Numerical Simulation and Performance Analysis of Magnetic Bioseparation in Microfluidic Devices

by Matin Golozar, Bachelor of Science

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Degree of
Master of Science
in the field of Mechanical Engineering

Advisory Committee:

Majid Molki, Co-chair

Jeff Darabi, Co-chair

Terry Yan

Graduate School
Southern Illinois University Edwardsville
July, 2016

ABSTRACT

NUMERICAL SIMULATION AND PERFORMANCE ANALYSIS OF MAGNETIC BIOSEPARATION IN MICROFLUIDIC DEVICES

by

MATIN GOLOZAR

Chairpersons: Professors Majid Molki and Jeff Darabi

This research presents the modeling and optimization of a magnetophoretic bioseparation chip for isolating cells, such as circulating tumor cells (CTCs), red, and white blood cells from the peripheral blood. The chip consists of a continuous flow through a microfluidic channel that contains locally engineered magnetic field gradients. The high gradient magnetic field produced by the magnets is spatially non-uniform and gives rise to an attractive force on magnetic particles moving through the channel. Simulations of the particle-fluid transport and the magnetic force were performed using the open-source software OpenFOAM to predict the trajectories and capture lengths of the particles within a fluidic channel. The computational model takes into account key forces, such as the magnetic and fluidic forces and their effect on design parameters for an effective separation. The results show that the microfluidic device has the capability of separating various cells with different sizes from their native environment. Additionally, to improve the performance of the separation device, a parametric study was performed to investigate the effects of the magnetic bead size, cell size, number of beads per cell, and flow rate on the cell separation performance. Computational results indicate that the trapping length decreases with increasing the number of beads per cell and the bead size. Also, the trapping length increased as the cell size was increased. Finally, an experimental study was performed to verify and validate the simulation results.

ACKNOWLEDGEMENTS

Firstly, I would like to express my special appreciation and thanks to my thesis advisors Prof. Majid Molki and Prof. Jeff Darabi for the continuous support of my Master study and related research, for their patience, motivation, and immense knowledge. Your guidance and encouragement helped me in all the time of research and writing of this thesis. I would also like to acknowledge Prof. Terry Yan for serving as my committee member even at hardship, and thank you for letting my defense be an enjoyable moment, and I am gratefully indebted to your insightful and brilliant comments on this thesis, thanks to you.

Above all, I would like to express my very profound gratitude to my family. Thanks to my parents Mohammad Ali Golozar and Roghayeh Fazel Anvar Yazdi who raised me with love of science and supported me in all my pursuits. My brother Mehdi Golozar for providing me with unfailing support and continuous encouragement throughout two of my years here. My sister Maryam Golozar who encouraged me and expressed confidence in my abilities. Thank you for your love, support, and unwavering belief in me. You all have been supportive and caring and thanks for all the sacrifices that you've made on my behalf. This accomplishment would not have been possible without you.

Author

Matin Golozar

TABLE OF CONTENTS

Chapter

LIST OF FIGURES

LIST OF TABLES

Table | Page

NOMENCLATURE

Latin Characters

A	face area vector
a_p	central coefficient
B	magnetic field
C	closed loop circulating around area A, particle volume concentration
d_p	diameter of the particle
dL	arc length
E	electric flux density
E	east
$\lvert F \rvert$	the magnitude of the total force acting on the particle
$\vec{F}_m$	magnetic force
$\vec{F}_d$	drag force
$\vec{F}_g$	gravity force
$\vec{F}_L$	lift force
$\vec{F}_B$	Brownian force
$\vec{F}_c$	particle collision force
f	flux vector through the face
f_{se}	flux vector of southeast face
g	body force
H	magnetic field intensity, matrix coefficients

i,j	unit vectors
J	current density
K_B	Boltzmann constant
M	remanence field
$\vec{M}_0$	initial magnetization of the bead
m_b	magnetic dipole moment, mass of the magnetic bead
M_b	magnetization of the bead
m_c	mass of the cell
m_p	mass of the particle, mass of the cell-bead complex
$\vec{M}_{sat}$	saturation magnetization
N	the number of magnetic particles
N	north
N_x	number of mesh in the x direction
N_y	number of mesh in the y direction
NE	northeast
NW	northwest
n	normal vector
P	pressure, point in the center of the control volume
q_ϕ	source or sink of ϕ
Re	Reynolds number
R_p	radius of the particle, radius of the cell-bead complex
S_ϕ	source term

S	south
S	surface
S_e	east surface
SE	southeast
SW	southwest
t	time
T	absolute temperature
$\vec{U}_p$	velocity of the particle
u	velocity
u_i	velocity components in x,y,z directions
V	volume
$\vec{V}_f$	velocity of the fluid
$\vec{V}_p$	velocity of the particle
V_b	volume of the bead
V_c	volume of the cell
V_p	volume of the particle, volume of the cell-bead complex
W	west
$\vec{X}_p$	position vector
x,y,z	components of the Cartesian coordinates

Greek Characters

ρ_v	electric charge density
ρ_p	cell-bead complex density, particle density
ρ_c	cell density
ρ_v	bead density
ρ	fluid density
ϵ_0	permittivity of free space
μ_0	permeability of free space
ν	fluid kinematic viscosity
χ_b	the magnetic susceptibility of the bead
η	fluid viscosity
Γ	diffusivity for the quantity ϕ
Ω	volume
τ_{ij}	stress tensor
ψ	magnetic scalar potential

Abbreviations

BCC	Breast Cancer Cell
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
CTC	Circulating Tumor Cell
CV	Control Volume
DPS	Discrete Particle Modeling
FV	Finite Volume
GAMG	Generalized Geometric-Algebraic Multi-Grid
HGMC	High-Gradient Magnetic Field Concentrator
IMP	Immune-Magnetophoresis
LOC	Lab-on-a-Chip
LPT	Lagrangian Particle Tracking
MFS	Micromachined Ferromagnetic Stripes
MT-MACS	Multitarget Magnetic Activated Cell Sorter
PCG	Preconditioned Conjugate Gradient Solver
PCM	Paramagnetic Capture Mode
PMMS	Paramagnetic Mode Magnetophoretic Separation
RBC	Red Blood Cells
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
WBC	White Blood Cells

CHAPTER 1

INTRODUCTION

1.1 Magnetic Separation

Developing integrated microfluidic devices with magnetism is relatively new and in the past few years, much research has been done to design and fabricate microfluidic-based magnetic cell separation devices to capture and analyze biological entities. Such microsystems typically range in size from millimeters to centimeters and are usually fabricated using polymer-based or planar glass-silicon substrates [1]. Implementing magnetic functionality into the microfluidic devices can be done by placing magnetic elements in the vicinity of the microchannel. Such elements provide magnetic field inside the channel which exert an attraction force on the labeled biomaterials as they flow through the channel. Magnetic elements could be passive or active. Passive magnetic elements have soft- or hard- magnetic microstructures. Soft magnetic elements, such as nickel-based microbars, are those which become magnetized in the presence of an external field and return to their non-magnetic state as the external field is removed. In contrast to soft magnetic element, a hard magnetic element retains its permanent magnetization [2]. Therefore, using the former elements we can turn on and off the magnetic force, while the latter elements do not have such flexibility and in order to remove the magnetic force, we need to move the permanent magnets away from the substrate. Another group of magnetic elements are active ones, such as voltage-driven conductors, where an electrically switchable magnetic force is created inside the channel. Comparing active and passive elements, the passive magnets produce substantially higher magnetic field inside the channel, resulting in a higher force on the magnetic particles, thereby enabling easier magnetophoretic control to sort or immobilize the particles [3–5].

1.2 Magnetic Beads

Magnetic particles play an important role in biomedical and clinical diagnostic applications and are widely used in microfluidic systems to separate and sort biological entities. Micrometer-sized magnetic beads are usually fabricated by encapsulating magnetite (Fe_3O_4) nanoparticles in an organic (polymeric) or inorganic matrix [6]. There are two common methods to bind the magnetic beads to biomaterials, direct and indirect binding. In direct binding the surface of the magnetic beads are coated with biocompatible molecules and are directly attached to the specific target biomaterial. Indirect binding requires at least one antigen to make it possible for the beads to attach to the target biomaterial [7].

1.3 Superiority of the Magnetophoretic Technique

When it comes to separating specific biological entities from their native environment, magnetic cell-separation devices appear to show some advantages over comparable electrostatic-based micro devices. Microfluidic devices that are engineered with passive magnets enable on-chip manipulation of magnetically labeled biomaterials using magnetic field that has been created by the permanent magnets, which requires no power consumption. Consequently, there is no need for a power source, drive circuitry and integrated electrodes that are necessary for electrophoretic manipulation [1]. In addition, since the sample fluids usually exhibit non-magnetic property, they are not interfered by the applied magnetic field and the magnetically labeled materials. While, in electrophoresis the carrier fluids, due to their intrinsic electrical properties, are affected by the external electric field. Also, magnetic cell-separation devices are less sensitive to factors such as iconic concentration, pH, and surface charge [1]. Another benefit of using magnets, is the capability of this method to prevent sample contamination by using external magnets, where the sample fluids are no longer in contact with the magnets. The nature of continuous flow in microfluidic devices significantly reduces the amount of non-target materials that are trapped beneath target materials and it allows for real-time monitoring of separation efficiency [1,8]. Microfluidic-based continuous-flow magnetophoresis is significantly faster than competing techniques and can isolate the target biomaterials directly from the crude samples such as blood, bone marrow, and tissue homogenates [9]. It is worth mentioning that, while the current work is based on magnetophoretic method, there are numerous other techniques to separate biomaterials from their native environment including, biochemical interactions, sheath flow and streamline sorting, fluorescence activated cell sorting, and deterministic lateral displacement [10,11].

1.4 The Proposed Magnetic Cell Separation Device

The microfluidic-based magnetophoretic bioseparation chip that has been modeled and investigated in this thesis has localized engineered permanent magnets at the bottom of the chip and the throughput of the device is significantly higher comparing to the recently developed microfluidic devices where the flow rates were limited to less than 1 ml/h [12,13]. The chip consists of a continuous flow through a microfluidic channel that contains high gradient magnetic fields along the channel. The high gradient magnetic field produced by the magnets is spatially non-uniform and provides large attractive forces on magnetic particles that travel through the channel. The overall dimensions of the device are 20 mm by 60 mm by 1 mm. Figure 1.1 shows a schematic representation of the magnetic separation chip in different steps. A mixture of target and non-target cells, is introduced into the separation channel. Among the particles moving along the channel, those with beads attached to them experience a magnetic force, which is governed by the gradient of the magnetic flux, and are pulled downward and immobilized on the magnetic surfaces. As the particles move forward, all the target cells are trapped by the magnetic field and the non-target cells that are not affected by the magnetic field will flow out through the separation channel.

1.5 Scope of Research

In this study, a numerical simulation model has been developed to predict the transport and separation of bioparticles by using an external magnetic field. The model is based on an Eulerian-Lagrangian CFD approach that takes into account the dominant forces on the particles. Simulations of the particle-fluid motion and magnetic field were performed using the open-source software OpenFOAM. It was shown that the proposed microfluidic magnetic-based device is capable of isolating different cells, such as circulating tumor cells (CTCs), red, and white blood cells from the peripheral blood. The numerical simulation has also been verified through experimental observations, where the microfluidic chip was made in the lab and was tested to observe the transport and capture of the magnetic beads inside the microchannel.

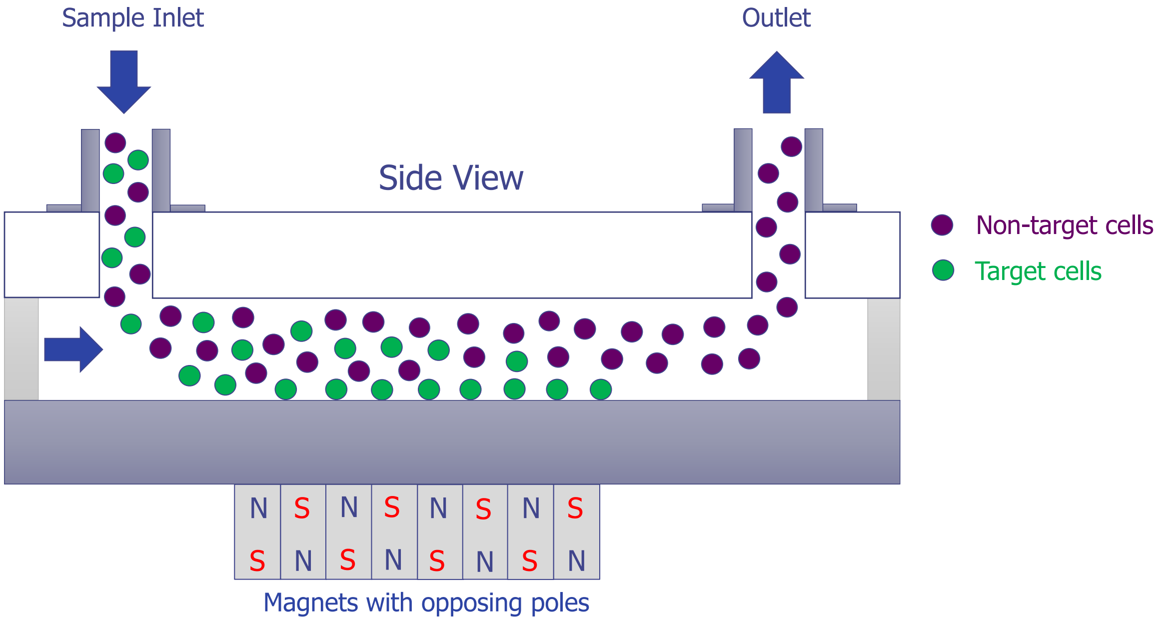


Figure 1.1: Schematic of a magnetic cell separation device.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Introduction

Microfluidic devices, due to a number of capabilities such as small size, low cost, low sample and reagent consumption, portability, and fast analysis time, are becoming a common fixture in laboratories and have shown to be a promising platform for separation and concentration of rare cells [14]. Over the past decade, the use of lab-on-a-chip (LOC) devices for preparing samples and separating cells has tremendously increased and a wide range of research has been done to develop new, more compact, and more versatile separation devices. Among the various methods, the continuous-flow magnetic separation technique offers the merits of simplicity, efficiency, and biocompatibility. This literature review provides some of the recently published works in this area, which illustrate the state of the art fabrication and detection techniques.

2.2 Methods to Enhance the Magnetic Field Gradient

In continuous-flow magnetic separation technique, the magnetic field gradient is a key parameter to create a high magnetic force inside the channel. Different methods have been studied and reported to increase the magnetic field gradient. Xia et al. [15] developed a high-gradient magnetic field concentrator (HGMC) for isolating contaminant bacteria and particulates from biological fluids. The HGMC consists of a micro-comb and micro-needle structure to generate a non-uniform magnetic field. Results indicate that this device is capable of separating living E. coli bacteria tagged with magnetic nanoparticles from red blood cells. Liu et al. [16] designed a microfluidic system in which a traveling magnetic field can be created inside the channel by using four current-carrying conductors to separate particles with two different sizes. Their simulation and experimental results show that the combination of current-carrying conductors and a uniform external field could increase the magnetic field gradient inside the channel up to 3 mT, which led to higher particle deflection trajectories. Jung and Han [17] introduced a lateral-driven technique to separate red blood cells (RBC) and white blood cells (WBC) from peripheral blood. A ferromagnetic wire array has been placed on a glass substrate to achieve a high-gradient magnetic field, resulting in an even lateral magnetic force inside the microchannel. Results show that, for a given flow rate of 20 l h^{-1} and the magnetic field of 0.3 T, the device is capable to continuously separate out 93.9% of RBCs and 89.2% of WBCs from the whole blood. Adams et al. [18] presented a multitarget magnetic activated cell sorter (MT-MACS) to continuously and simultaneously sort two types of target cells into two different outlets. Micromachined ferromagnetic strips (MFS) were used to create short-range and strong magnetic gradient within the microchannel. To get the desired combination effect of the magnetophoretic force and the hydrodynamic force, one group of cells are labeled with large radius and large magnetization tags, while others are labeled with small radius and small magnetization tags. Results indicate that the MT-MACS is capable of sorting the target cells with purity higher than 90% for multiple bacterial cell types at a throughput of 10^9 cells per hour. Derec et al. [19] suggested a magnetophoresis cell separation device, where he used a line conductor parallel to the microchannel to create a changeable magnetic field inside the channel. The only disadvantage of the device was the heat generation, due to an electrical current passing through the copper conductor. Numerical and experimental results indicate that the chip was capable of isolating tumor cells labelled with magnetic nano-particles. Lee et al. [20] invented a high-speed RNA microextractor for the direct separation of RNA from blood lysate using magnetic oligo-dT. The working principle is based on a ferromagnetic wire array which creates lateral magnetic force inside the channel. Within 1 minute, the device was capable of extracting 80% magnetic beads with a flow rate as high as 20 ml/h. The produced magnetic field by the ferromagnetic wire array was strong enough to direct the labeled particles into the collecting tube. Shen et al. [21] devised a label-free cell separation device where the magnetic susceptibility of the medium was improved to increase the magnetophoresis force acting on the cells. Micro-fabricated ferromagnetic structures are engineered beneath the microchannel, which allows the concentration of magnetic field being generated from external permanent magnets, resulting in a very high magnetic field gradient. This high-gradient magnetic field was later used to successfully separate white and red blood cells from whole blood based on their intrinsic magnetic properties.

One of the main characteristics of microseparator devices, is their ability for sample processing which is still performed off chip. Verbarg et al. [22] developed a MagTrap device, which is a sample manipulation unit that can be used to capture, mix, and release target particles in a microfluidic channel. The device consists of 6 pairs of permanent magnets in a rotating wheel, mounted beneath the microchannel. The time-dependent magnetic field produced by the rotating magnets, were efficient enough to separate, mix, and release the target particles.

Different configurations of the magnets around a microchannel, provide different magnetic field gradients inside the channel. Wilbanks et al. [23] adjusted the magnets around a microchannel asymmetrically to concentrate diamagnetic particles in a ferrofluid flow through a straight rectangular microchannel. Experimental and theoretical studies show that the magnetic asymmetry configuration increases the rate for particle trapping. Jung et al. [24] presented a six-stage cascade paramagnetic mode magnetophoretic separation (PMMS) device to separate white and red blood cells based on their native magnetic properties. Experimental results indicate that the PMMS system is able to continuously separate RBCs at a volume flow rate of 28.8 L /hr with a separation efficiency of 89.5%. Khashan et al. [25] proposed a magnetic bioseparation device with flow-invasive magnetic elements to separate magnetically-labeled bioparticles. The magnetic elements are magnetized using a bias field and are placed beneath the chip with their length transverse to the flow. The proposed scheme overcomes the limitation of the channel size and the disadvantage of short-range magnetic force. Also this configuration improved the capture efficiency compared to devices with magnetic structures engineered in the channel wall.

2.3 Methods to Enhance the Susceptibility Mismatch

In order to be able to magnetically separate the particles, it is required to modify their magnetic properties. Labeling the target cells with high magnetic susceptibility beads and changing the susceptibility of the medium are two common methods to increase the susceptibility mismatch [14]. Kim et al. [26] introduced an immune-magnetophoresis (IMP) cell sorting chip to separate T-cell from biological suspensions, like whole blood. There are two different inlets for blood stream, containing cells, and antibody coated magnetic particles. Using the magnetic force, the binding objects are pulled towards the blood stream, where the labeling of the target cells is performed by relative motion of the cells and the particles. Finally, at the outlet of the channel a permanent magnet separates the label target cells from the blood sample. Pamme and Wilhelm [27] created a microfluidic platform for continuous sorting of cells loaded with magnetic nanoparticles. Simulation and experimental results matched with each other and mouse macrophages and human ovarian cancer cells were successfully separated by the use of a magnetic field gradient inside the channel. Forbes and Forry [28] developed and characterized a microfluidic system with angled permanent magnet beneath the microchannel. The device can be used for the lateral magnetophoresis of labeled cell-bead complexes and superparamagnetic beads. A dimensionless parameter has been introduced to efficiently investigate the design parameters, such as, number of beads, magnet orientation, magnet type (permanent, ferromagnetic, electromagnet), flow rate, channel size, and buffer. Using the defined dimensionless parameter, the transition between regimes from the magnetically dominated to hydrodynamically dominated is experimentally verified. Zeng et al. [29] studied the continuous separation of diamagnetic particles and cells in ferrofluid flow using a pair of permanent magnets at the inlet and outlet of the microchannel. 3 m- and 10 m-diameter diamagnetic polystyrene particles were separated successfully and the effect of flow speed and magnet distance examined and verified using both experimental and numerical approaches. Liang et al. [30] experimentally and numerically demonstrated that replacing a water-based diamagnetic solution with a dilute ferrofluid can significantly improve the separation of magnetic and diamagnetic particles. By using a ferrofluid medium, magnetic particles experience positive magnetic force and diamagnetic particles experience negative magnetophoresis when applied to an external magnetic field. A T-microchannel ferrofluid magnetic based separation device was developed, which was able to successfully separate magnetic and diamagnetic particles from each other. Darabi et al. [31] developed a magnetophoretic bioseparation device for continuous separation of CD4+T cells from whole blood. An array of thin nickel stripes is deposited on a glass substrate, which produces high magnetic field gradient inside the channel when exposed to an external magnetic field. Experimental results indicate a cell purity of greater than 95% at a sample flow-rate of 50 ml/h.

2.4 Magnetic Separation of Circulating Tumor Cells

Plouffe et al. [34] designed a microfluidic magnetophoretic cell separation platform for isolating cancer cells from suspension. The device consists of an electromagnet wire array which creates magnetic field along the microchannel. When magnetic field is applied inside the channel, the magnetic particles are deflected towards the center collection stream. The device is capable of separating hematopoietic stem cells and endothelial progenitor cells from whole blood. Hoshino et al. [35] developed a microchip-based immunomagnetic circulating tumor cell (CTC) detection device for separating cancer cells labeled with magnetic nanoparticles from whole blood. An array of magnets with opposing poles were used to create high magnetic field gradient inside the channel. From experimental results capture rate 90% and 86% for COLO205 and SKBR3, respectively, cells were reported. The results are also predicted by the numerical calculation. Han et al. [36] devised a microsystem for isolating suspended non-labeled breast cancer cells (BCC) in whole blood. Initially red blood cells are captured by means of a continuous paramagnetic capture mode (PCM) and are drawn closer to the ferromagnetic wire and the BCCs behave as diamagnetic particles and are forced away from the wire. Therefore, RBCs and BCCs are forced into different outlets and get separated. About 94.8% of breast cancer cells were successfully isolated and detected.

2.5 Analytical Modeling of Microfluidic Devices

Furlani et al. [32] proposed a microfluidic device based on a mathematical modeling for continuous separation of white and red blood cells from plasma. An external magnetic field is used to magnetize an array of integrated soft-magnetic elements that are placed transverse to the microchannel. Diamagnetic and paramagnetic behavior of WBCs and RBCs is a key factor for the cell-separation process in this study. In another study, Furlani [1] presented a numerical approach to investigate effects of particle-fluid coupling on particle transport and capture in a magnetic-based microfluidic device. Analysis show that for high particle concentration, one-way coupling, where the particles do not affect the fluid flow, overpredicts the magnetic force required to capture the particles, while for more accurate predictions, two-way particle-fluid coupling is required. Cheng et al. [33] proposed an analytical three-dimensional modeling that can predict non-magnetic particles trajectory in a magnetic fluid being exposed to permanent magnets. The effects of magnetic susceptibility of particles and the carrier fluid, flow-rate, as well as different geometrical parameters of the system on the particle deflection has been studied.

CHAPTER 3

THEORETICAL BACKGROUND

In general, three distinct physical phenomena are involved in the process of particle separation in a microfluidic channel. The magnetic field which is governed by Gauss's magnetism law, fluid flow which is determined by Navier-Stokes and continuity equations, and particle dynamics which is governed by Newton's second law of motion. This chapter introduces the fundamental theory of each phenomenon and reveals its significance on particle separation.

3.1 Magnetic Field

A magnetic field is a region around a moving electric charge or a magnetic material within which the force of magnetism acts. The spinning and orbiting of the nucleus of an atom produces a magnetic field as does electrical current flowing through a wire. The direction of the spin and orbit determine the direction of the magnetic field [37]. In 1861, James Clerk Maxwell developed the widely known Maxwell's equations which are a set of four partial differential equations that describe how electric and magnetic fields propagate, interact, and how they are influenced by objects. The four equations are Gauss' Law, Gauss' Magnetism Law, Faraday's Law, and Ampere's Law, which are discussed in the following sections.

3.1.1 Gauss' Law

Gauss' Law is the first of the Maxwell's equations which indicates how the electric field behaves around electric charges. Gauss' Law can be written in terms of the electric flux density and the electric charge density as

$$\nabla \cdot E = \rho_v \tag{3.1}$$

where E is the electric flux density and ρ_v is the electric charge density. By assuming some arbitrary volume V, Gauss' Law can be written in an integral form as

$$\iint_A E \cdot dA = \frac{1}{\varepsilon_0} \iiint_V \rho_v \, dV \tag{3.2}$$

where A is the boundary of the volume and ε_0 is the permittivity of free space [38].

3.1.2 Gauss' Magnetism Law

Gauss' Magnetism Law states that the net magnetic flux out of any closed surface must be zero, and it simply means that all magnetic field lines form complete loops. In other words, Gauss' Law for Magnetism indicates that magnetic monopoles do not exist or at least they haven't been discovered yet. The differential and integral forms of this law are as follows

$$\nabla \cdot B = 0 \tag{3.3}$$

$$\iint_A B \cdot dA = 0 \tag{3.4}$$

where B is the magnetic field and A is the area of any closed surface [38].

3.1.3 Faraday's Law

Faraday's Law is the third equation in Maxwell's Equations, which states that a changing magnetic flux within a circuit (or closed loop of wire) produces an induced electromotive force, which is the opposite of the time-rate-of-change of the magnetic flux through the circuit. The differential and integral equations are

$$\nabla \times E = -\frac{\partial B}{\partial t} \tag{3.5}$$

$$\oint_C E \cdot dl = -\frac{d}{dt} \iint_A B \cdot dA \tag{3.6}$$

where E is the electric field, B is the magnetic field, dl is an infinitesimal arc length along the wire, and C is the closed loop circulating around the area A [38].

3.1.4 Ampere's Law

Ampere's Law is the last equation in Maxwell's equations, which relates the electric current passing through a closed loop to the integrated magnetic field around the loop. In other words, as a result of Ampere's Law one can determine a flowing electric current which gives rise to a magnetic field that circles the current and also a time-changing electric flux density which gives rise to a magnetic field that circles the E field. The differential and integral forms of this law are

$$\nabla \times B = \mu_0 (J + \varepsilon_0 \frac{\partial E}{\partial t}) \tag{3.7}$$

$$\oint_C B \cdot dl = \mu_0 \iint_A J \cdot dA + \mu_0 \varepsilon_0 \frac{d}{dt} \iint_A E \cdot dA \tag{3.8}$$

where μ_0 is the permeability of free space and J is the total current density [38].

3.2 Fluid Flow

Governing equations that predict the motion of fluids and can be used to estimate particular parameters such as velocity, pressure, and temperature are conservation of mass (continuity), Navier-Stokes, and energy equations. In this study, the fluid is isothermal and, therefore, energy equation is not used.

Conservation of mass states that the mass of a particular system is conserved, which means that there is no accumulation or depletion of mass, so mass is conserved within the domain. Continuity equation is stated as

$$\frac{d\rho}{dt} + \nabla \cdot (\rho u) = 0 \tag{3.9}$$

where ρ is the fluid density and u is the fluid velocity. For an incompressible flow, Eq. (3.9) reduces to:

$$\nabla \cdot u = 0 \tag{3.10}$$

The Navier-Stokes equations represent the conservation of momentum and can be obtained by applying Newton's second law of motion to fluids. In the case of an incompressible Newtonian fluid, the Navier-Stokes equation can be expressed as:

$$(\frac{\partial u}{\partial t} + u \cdot \nabla u) = -\frac{\nabla P}{\rho} + g + \nu \nabla^2 u \tag{3.11}$$

where u is the fluid velocity, P is the fluid pressure, ν is the fluid kinematic viscosity, and g is the acceleration of gravity.

3.3 Force Analysis

The motion of the colloidal particles in the presence of the magnetic field is governed by several forces including the magnetic force, the hydrodynamic drag force, the gravitational force, lift force, particle-fluid coupling, Brownian motion, and particle effects, e.g. magnetic dipole-dipole and particle-particle interactions. A comprehensive study including all the forces is very complex and beyond the scope of this study. In this section, the focus is on determining dominant forces that might affect the particle motion in a microfluidic channel. By comparing the order of magnitude of each force, we are able to identify the key forces and their effect on design parameters for effective separation. According to Newton's second law, the trajectory of a labeled particle in a laminar flow through a microfluidic channel is determined by the combined influence of colloidal and hydrodynamic interactions, as described by the general force balance equation [39]

$$m_p \frac{d\vec{u}_p}{dt} = \vec{F}_m + \vec{F}_d + \vec{F}_g + \vec{F}_L + \vec{F}_B + \vec{F}_c \tag{3.12}$$

where m_p is the mass of the particle, $\vec{u}_p$ is the particle velocity, and $\vec{F}_m$, $\vec{F}_d$, $\vec{F}_g$, $\vec{F}_L$, $\vec{F}_B$, and $\vec{F}_c$ are the magnetic force, drag force, gravity force, lift force, Brownian force, and particle collision force, respectively.

3.3.1 Magnetic Force

Magnetic force is the dominant force in a magnetophoresis process, which pulls the magnetically-labeled particles towards the permanent magnets. The magnetic force is applied to the magnetic beads which show superparamagnetic behavior in the presence of the applied magnetic field. Superparamagnetism is a type of magnetism that occurs in small ferromagnetic or ferrimagnetic nanoparticles [9]. This includes sizes around a few nanometers to a couple of tenth of nanometers, depending on the material. As the magnetic field is removed, the superparamagnetic particles will no longer experience magnetic force and will behave like a neutral particle.

In general, the magnetic force acting on a magnetic bead in the presence of an applied magnetic field can be written as [40]

$$\vec{F}_m = (\vec{m}_b \cdot \nabla)\vec{B} \tag{3.13}$$

where m_b is the magnetic dipole moment of the bead and B is the applied magnetic field. If the bead has a volumetric magnetization M_b, then the dipole moment on the bead is given by [40]

$$\vec{m}_b = V_b\vec{M}_b \tag{3.14}$$

The Magnetization of the bead is a function of the applied magnetic field. As can be seen in Figure 3.1, the magnetization is a linear function of the field intensity up to saturation (0.2 T) after which it remains a constant with a value M_{sat} that is independent of the applied magnetic field. The magnetization data in Figure 3.1 correspond to Dynabeads (Invitrogen Corporation) that have been used in the experiment. When the magnetic particle is saturated the magnetization is nearly constant, $M_b = M_s$, and the magnetic force can be approximated as

$$\vec{F}_m = V_b(\vec{M}_{sat} \cdot \nabla)\vec{B} \tag{3.15}$$

Since the magnetic force is proportional to the number of magnetic particles attached to each cell, Eq. (3.15) must be multiplied by the number of beads

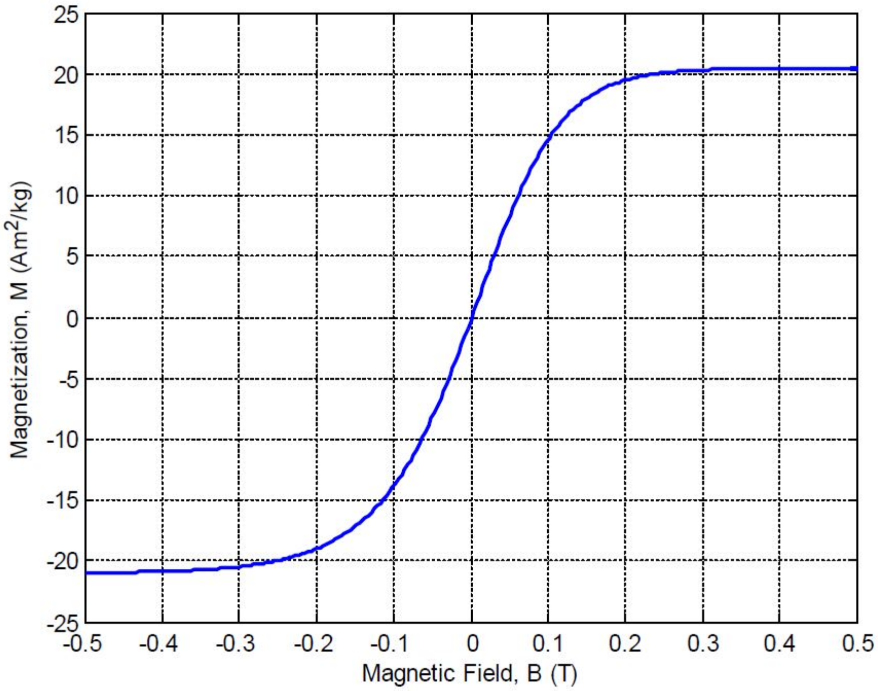


Figure 3.1: The schematic behavior of superparamagnetic Dynabeads in an external magnetic field. (Adapted from Guo C. [9])

$$\vec{F}_m = N\, V_b(\vec{M}_{sat} \cdot \nabla)\vec{B} \tag{3.16}$$

Below saturation, where the magnetic particle is unsaturated, the magnetization can be approximated as

$$\vec{M}_b = \vec{M}_0 + \frac{\chi_b}{\mu_0}\vec{B} \tag{3.17}$$

where M_0 is the initial magnetization of the bead, χ_b is the magnetic susceptibility of the bead, and μ_0 is the permeability of free space. With this definition, the magnetic force acting on N unsaturated magnetic beads can be expressed as

$$\vec{F}_m = N\,\frac{V_b \chi_b}{2\mu_0} \nabla \vec{B}^{\,2} \tag{3.18}$$

Also, when the magnetic beads are magnetized, they exert force on each other. However, since the magnetic beads are much smaller than the surrounding medium and the particle they are attached to, this force is considerably small and can be neglected.

3.3.2 Drag Force

Drag force is a mechanical force which is generated by the interaction and contact of the particles with the fluid within the microchannel. In the study of magnetophoretic bioseparation process, depending on the cell size, different number of magnetic beads are attached to them in order to implement an effective separation. Thus, the drag force is applied to the cell-bead complex. In general, if there are N magnetic beads attached to a cell, the effective mass, volume, density, and radius of the cell-bead structure can be estimated by using the approach presented by Safaryk et al. [41]

$$m_p = m_c + N m_b = \rho_c V_c + N \rho_b V_b \tag{3.19}$$

$$V_p = V_c + N\,V_b \tag{3.20}$$

$$\rho_p = \frac{\rho_c V_c + N\,\rho_b V_b}{V_c + N\,V_b} \tag{3.21}$$

$$R_p = (\frac{3}{4\pi} V_p)^{1/3} \tag{3.22}$$

where N is the number of magnetic beads and subscripts p, c, and b represent cell-bead complex, cell, and bead, respectively. Since the magnetic beads are much smaller comparing to the cell they are attached to, and the distance between the beads is much smaller than their distance from the permanent magnets, the cell-bead complex can be assumed as a spherical structure. Considering that the magnetically labeled cells are spherical, the hydrodynamic drag force is given by Stokes' law

$$\vec{F}_d = 6\pi\eta R_p(\vec{V}_p - \vec{V}_f) \tag{3.23}$$

where R_p is the radius of the cell-bead structure, η is the fluid viscosity, and $\vec{V}_p$ and $\vec{V}_f$ are the velocities of the particle and fluid, respectively.

3.3.3 The Gravitational Force

Considering Buoyancy, the gravitational force can be defined as an upward force exerted by fluid that opposes the weight of an immersed particle. The net gravitational force can be written as

$$\vec{F}_g = -V_p\left(\rho_p - \rho_f\right)\vec{g} \tag{3.24}$$

where V_p, ρ_p, ρ_f, and $\vec{g}$ are volume of the particle, the density of the particle and fluid, and the acceleration due to gravity, respectively.

3.3.4 Lift Force

Spherical particles in a shear field experience a lift force perpendicular to primary streamlines. Depending on the position of the particle in the flow, there are two different types of lift forces on particles, shear gradient-induced lift force and boundary layer lift force. The inertial force depends on the fluid velocity and the particle size relative to the length of the channel [14]. At high Reynolds numbers $Re_p \gg 1$, the inertial force becomes significant among the other forces and it can be used for lateral particle separation. On the other hand, for low Reynolds numbers $Re_p \ll 1$, the hydrodynamic drag force is dominant and the magnetophoresis technique can be used to separate particles. Thus, due to the fact that magnetic separation and inertial separation are suitable for different ranges of velocities and particle diameters, a separation process cannot gain from both techniques at the same time [42]. In this study due to small particle size and low flow velocity, approximately $0.01\, m/s$, the Reynolds number is much less than unity and, consequently, the magnetic force becomes dominant. Hence, the effect of lift force has been ignored.

3.3.5 Brownian Force

Brownian motion is the random movement of particles suspended in a fluid resulting from random collisions of molecules of the fluid with the particles. If the diameter of particles is less than threshold diameter approximated by Eq. (3.25), the Brownian motion can affect the motion of suspended particles [43].

$$\mid F \mid d_p \ll K_B T \tag{3.25}$$

In Eq. (3.25), $\mid F \mid$ is the magnitude of the total force acting on the particle ,K_B is the Boltzmann constant, T is the absolute temperature, and d_p is the diameter of the particle. It has been shown that for particles' diameter smaller than d_p, trajectories cannot be obtained by Newton's Eq. (3.12) [44].

3.4 Significant Forces for the Magnetic Cell Separation Application

Based on the particle size and the magnitude of the magnetic field, some of the mentioned forces could be neglected for magnetic cell separation application. Particle-particle interaction and particle-fluid interaction can be ignored for particle suspensions with small particle volume concentration ($c \ll 1$) [44]. Figure 3.2 has been produced by Hejazian et al. [14] to schematically illustrate the order of magnitude of different forces as a function of characteristic particle diameter (at an average velocity of $1\,mm\,s^{-1}$ and channel height and depth of $100\,\mu m$). As can be seen in Figure 3.2, in a relatively high magnetic field ($0.5\,T$) and large particle diameter of $20\,\mu m$, drag force, magnetic force, and gravitational force are the most significant forces, and other forces could be ignored.

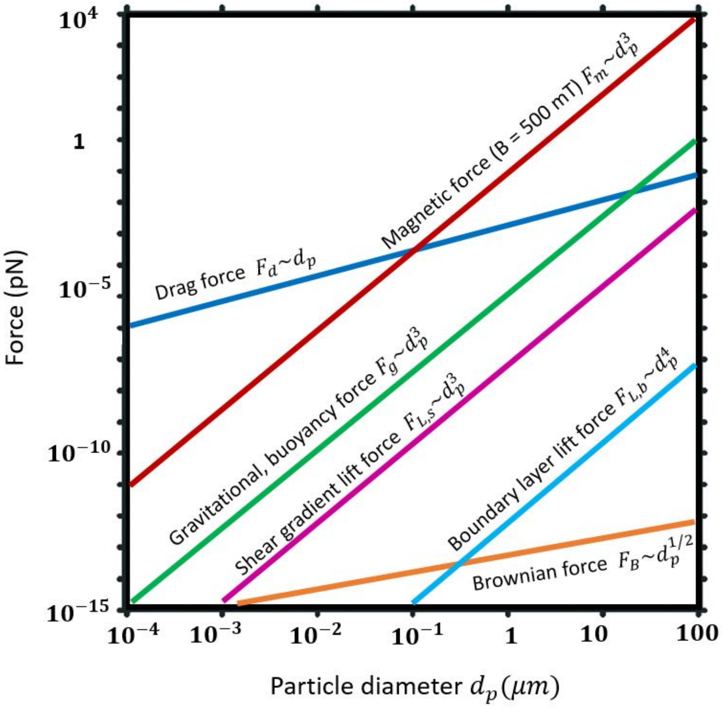


Figure 3.2: Orders of magnitude of forces acting on a magnetic particle as a function of the characteristic particle diameter. (Adapted from Hejazian et al. [14])

As a result of the above discussion, Figure 3.3 illustrates the schematic of the dominant forces acting on a magnetically labeled particle flowing through a microchannel in the presence of an applied magnetic field.

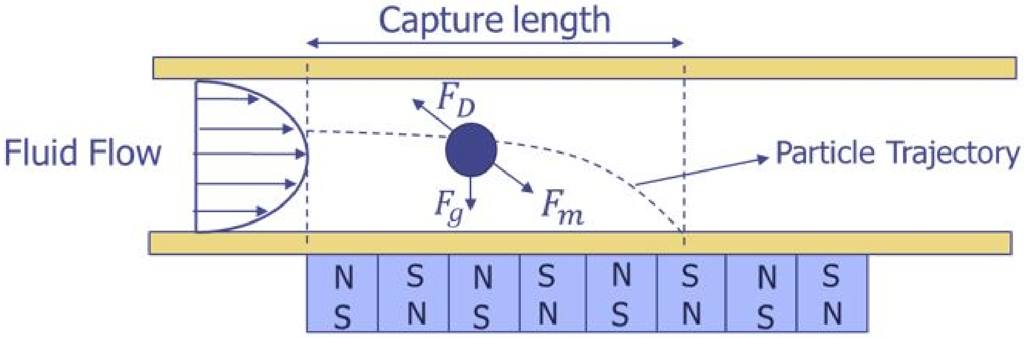


Figure 3.3: Schematic view of the dominant forces acting on a magnetically labeled particle flowing through a channel in the presence of an applied magnetic field.

3.5 Geometry and Boundary Conditions

Boundary conditions for the fluid velocity are the no-slip boundary condition. Fluid velocity at the inlet of the channel changes from 0.0139 m/s to 0.0416 m/s, depending on flow rates. For the walls, zero pressure gradient is applied. The pressure at the outlet of the channel is uniform and equal to zero. An array of Neodymium magnets with opposing poles were placed underneath the channel to produce high magnetic field gradient inside the channel. Boundary conditions for the permanent magnets are uniform magnetic flux density of 0.7 T at the surface of each magnet. A mixture of target and non-target cells, is introduced into the separation channel. The initial velocity of each particle is equal to the corresponding fluid velocity at that point at the inlet of the channel. In microfluidic channels the aspect ratio of the height of the channel to its width is very small, consequently the symmetry boundary condition is applied to the symmetry planes located at z = 0 and z = 0.02 m.

CHAPTER 4

COMPUTATIONAL METHODOLOGY

In this research project, the geometry of the microfluidic channel is a simple rectangular enclosure, but modeling the separation process requires solving three different cases. First, modeling the flow field and magnetic field in two separate computational domains, and then modeling a third case to find the particle trajectories by using the values obtained from the flow and magnetic fields.

The computational domain in this study, is a rectangular microchannel with the overall dimensions of $20 \times 60 \times 0.2\,(mm^3)$. Because the inlet velocity and the hydraulic radius of the channel are considerably small, the fluid flow can be treated as a fully developed laminar flow inside the channel. Hence, the flow field can be modeled within a separate computational domain, with a similar geometric dimensions, and the obtained velocity field can be further used in the main case, where the particles are injected into the channel. On the other hand, the magnetic field needs to be treated differently. As can be seen in Figure 4.1, the magnetic field produced by the permanent magnets, forms magnetic field lines, which are distributed all over the space around the magnets. Depending on the location of the channel in the magnetic field, one gets different magnetic field gradients within the channel. As a result, the magnetic field needs to be modeled within a very large computational domain around the magnets, and based on the location of the channel, the corresponding magnetic field values are used to find the particle trajectories.

For all three cases, the mesh is clustered near the walls and the interfaces of the magnets, but uniform elsewhere. Table 4.1 presents the total number of cells, the mesh type, dimensions, end time, and data-write interval for each of the three cases. Details of the codes, setups, and implementation of the parameters for all three cases are provided in the Appendices section.

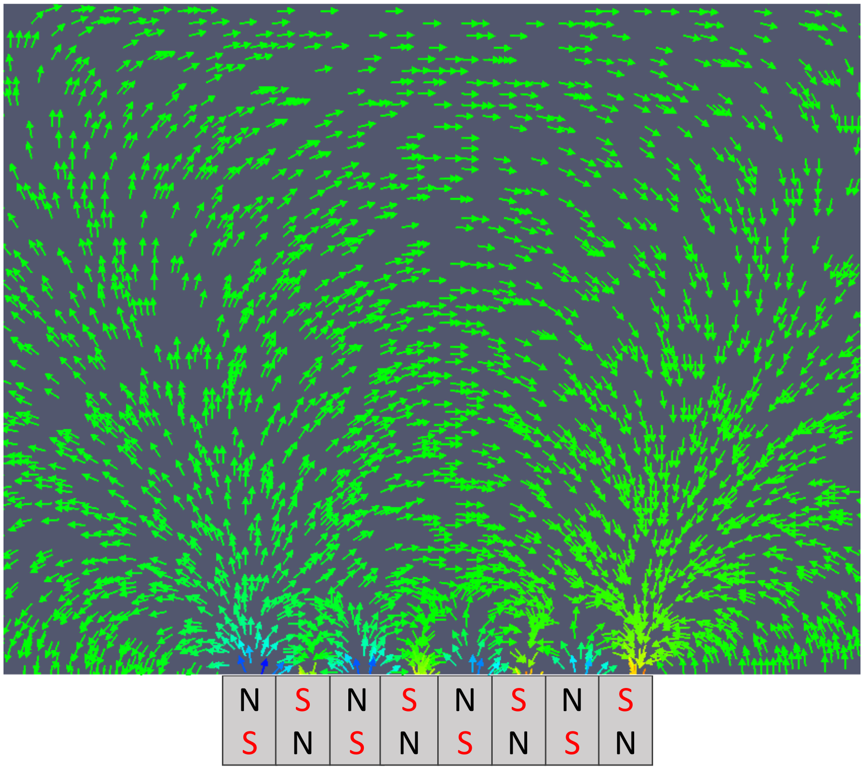


Figure 4.1: The magnetic field lines produced by the permanent magnets.

For the fully developed laminar flow in the channel, the computation needs to be run for a long time to reach steady state. It is also necessary to run the other two cases up to 1 and 3 seconds to observe the essential features of the phenomena.

So far, non-discretized governing equations, geometry, and the boundary conditions have been presented. In this chapter, the computational methodology used in this research is described. In the rest of this chapter, the computer software (OpenFOAM), the method used, Finite Volume Method, the mesh generation and refinement, and the discretization of the governing equations will be discussed.

Table 4.1: Information about the case setups

Case	Mesh Type	Number of Cells	Dimensions	End Time	Write-Interval
Flow Field	Clustered near the wall and the interfaces of the magnets, Uniform elsewhere	1,164,500	$20 \times 60 \times 0.2$ mm^3	3000 s	1 s
Magnetic Field	Clustered near the wall and the interfaces of the magnets, Uniform elsewhere	3,960,000	$20 \times 60 \times 10$ mm^3	1 s	1 s
Particle Trajectory	Clustered near the wall and the interfaces of the magnets, Uniform elsewhere	1,164,500	$20 \times 60 \times 0.2$ mm^3	3 s	1e-6 s

4.1 OpenFOAM Software

OpenFOAM (Open Field Operation and Manipulation) [45] is a free, open source CFD software package developed by OpenCFD Ltd. OpenFOAM has a well-developed user base across different areas of engineering and science. OpenFOAM includes a wide range of features which makes this software a powerful tool to solve anything from complex fluid flows involving turbulence, heat transfer, and chemical reactions to electromagnetics and solid dynamics.

One of the main advantages of using OpenFOAM, is the complete freedom of accessibility to source codes to customize and extend existing functionalities within the software. However, trusting an open source software may cause some problems in terms of accuracy. For OpenFOAM, the provided features are regularly checked by developers and maintainers at OpenFOAM Ltd, where the developments are permanently updated through the continual release-cycle of OpenFOAM. Also, many major universities around the world are using the same software every day to solve a variety of problems and the results are published in well-known journals. Thus all the provided features are being tested permanently and efficiency and accuracy of the solvers are reliable.

OpenFoam includes different tools, such as meshing in and around complex geometries, pre- and post-data processing and visualization, and parallel computing. As mentioned above, one of the most important features of OpenFOAM, is that it allows users to freely contribute, develop and modify the solvers and codes. This is probably the main reason why OpenFOAM is the preferred software package for this thesis work. Magnetophoresis cell separation process is a multiphysics phenomenon consisting of fluid flow, magnetic field, and particles motion. By using an appropriate solver for each physic, not only we were able to model each case separately and finally couple them together to solve the main case, but also we were able to modify the code and change the forces applying to each particle. Also, parallel computing is another significant feature of OpenFOAM, which considerably helped to speed up computations for cases with a very fine mesh. OpenFOAM has no limitation on number of processors, and based on the level at which one's hardware supports parallelism, up to 1000 processors can be used to solve computationally and data-intensive problems [45]. This is true while other commercial software packages have certain limitations based on the license agreement.

While OpenFOAM is famous among current existing software packages as being free and open source, there are some major challenges with this software. First and foremost, is the familiarity with object-oriented programming in C++. Object orientation attempts to resolve the complexity by building individual software components (classes) which classify data and functions together and maintain data from accidental corruption [46]. OpenFOAM contains several tutorials to solve particular engineering and science problems and there are few documents for beginners on how to pre-process, run and post-process different cases. Using OpenFOAM to solve multiphysics problems requires careful examination of relevant tutorials for each phenomenon and applying proper modifications to solve the main case. At this point, OpenFOAM has no specific flow chart to follow, however people around the world share their ideas and experiences about OpenFOAM in CFD online discussion forums and useful information can be found on how to further develop proper solvers for a variety of problems. Another challenge in using OpenFOAM is that the errors, which address the problem, do not provide the users with solutions and sometimes the user needs to spend a considerable amount of time to fix them. All in all, every software package has its own advantages and disadvantages and this should be kept in mind that, every problem is doable, but it needs patience and continuous effort to find the solution.

4.2 Finite Volume Method

OpenFOAM uses Finite Volume (FV) Method to discretize differential equations in order to solve them in a discretized solution domain. To discretize the solution domain, the space is divided into a finite number of discrete regions, called control volumes. Discretization of the solution domain creates a computational mesh where the discretized equations are subsequently solved. As discussed earlier in this chapter, Table (4.1) provides a summary of the mesh type and the total number of cells being used in this study.

Furthermore, FV method uses a volume integral form of the problem with a finite partitioning set of volumes, to discretize the equations by using an appropriate approximation. In FV method, by using the divergence theorem, volume integrals in a partial differential equation, which contain a divergence term, are converted to surface integrals.

Surface integrals in control volume faces are evaluated as fluxes at the surfaces of each finite volume, and since the flux entering a given volume is equal to that leaving the neighboring volume, the FV method is conservative. Conservative schemes are popular to discretize partial differential equations, because if they converge, they can be proven [47] to converge to a weak solution of the conservation law. By writing integral form of governing equations for each control volume and canceling out the surface integrals, the global governing equations are obtained.

4.2.1 Discretization Procedure for the Navier-Stokes Equation

The standard form of Navier-Stokes and continuity equations for an incompressible flow is:

$$\nabla \cdot u = 0 \tag{4.1}$$

$$\frac{\partial u}{\partial t} + \nabla \cdot (uu) - \nabla \cdot (\nu \nabla u) = -\nabla P \tag{4.2}$$

The Navier-Stokes equation is a special case of a convection-diffusion equation with some source term, which is a negative of the pressure gradient. In general, the convection-diffusion equation, which is also called generic transport equation, is a combination of the diffusion and convection terms, and describes physical phenomena where physical quantities such as mass, energy, and momentum are transferred inside a physical domain. In this section, the general differential equation of generic transport equation will be used to demonstrate discretization methods for convection, diffusion, and source terms that are common in all conservation equations (continuity, energy, and momentum). The contribution from the pressure in the Navier-Stokes equation, has no analog in the transport equation and it requires special attention which is also included in this chapter. Finally, since this is a time-independent problem, the unsteady term will be dropped in the calculations.

The coordinate-free vector form of the transport equation, describing conservation of a quantity, ϕ, is [48]

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho v \phi) - \nabla \cdot (\Gamma_\phi \nabla \phi) = S_\phi(\phi) \tag{4.3}$$

The four terms are unsteady term, convection term, diffusion term, and source term, respectively, and Γ is the diffusivity for the quantity ϕ. In finite volume method, the integral form of the transport equation must be used to discretize the equations [48]

$$\int_S \rho\phi v \cdot n dS = \int_S \Gamma \nabla \phi \cdot n dS + \int_\Omega q_\phi d\Omega \tag{4.4}$$

To discretize Eq. (4.4), the solution domain is subdivided into a finite number of control volumes (CV) by choosing a suitable grid and allocating the computational node to each CV center. Then the integral Eq. (4.4) applies to each CV, as well as to the entire solution domain. Finally, the global conservation equation is obtained by adding up all the equations for all CVs. To obtain algebraic equations, the surface and volume integrals are approximated by using quadrature formulas.

4.2.2 Approximation of Surface Integrals

Figure 4.2 and 4.3 show 2D and 3D Cartesian control volumes. Each control volume is assigned to a node (P) and the CV surface, which consists of four (in 2D) and six (in

3D) plane faces, is denoted by lower-case letters corresponding to their direction (e, w, n, s, t, and b) [48].

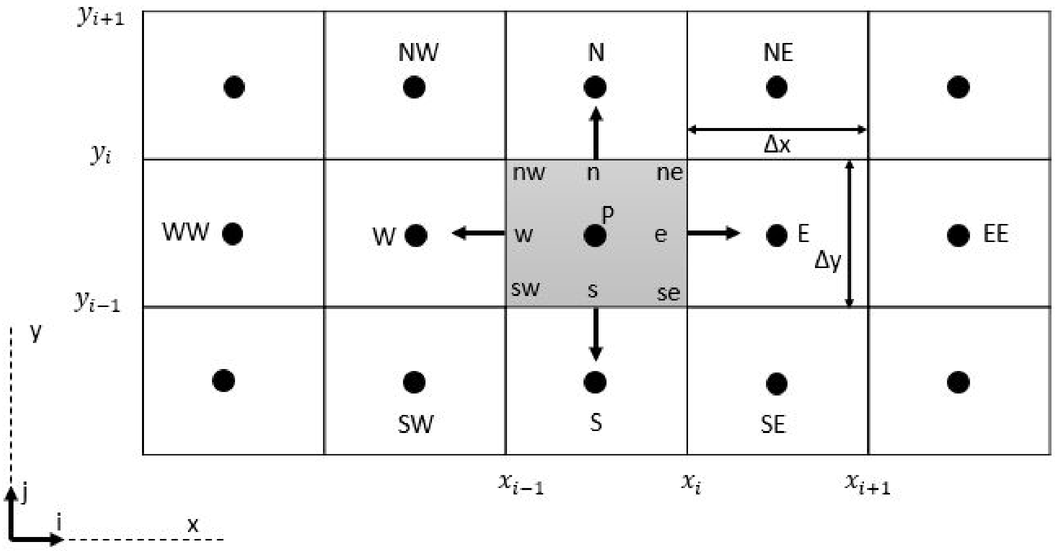


Figure 4.2: Control volume for a Cartesian 2D grid.

As discussed earlier, the surface and volume integrals are important for FV method to obtain algebraic equations. The net flux through the CV boundary is the sum of integrals over the four (in 2D) or six (in 3D) CV faces [48]

$$\int_S f dS = \sum_n \int_{S_e} f dS \tag{4.5}$$

where f is either the convective ($\rho\phi v \cdot n$) or diffusive ($\Gamma\nabla\phi \cdot n$) component in the flux vector. Since the value of the integrand f is not available everywhere on the surface S_e, an approximation must be introduced to calculate the surface integral in Eq. (4.5). A second-order approximation for the surface integral is the trapezoid rule, which is [48]

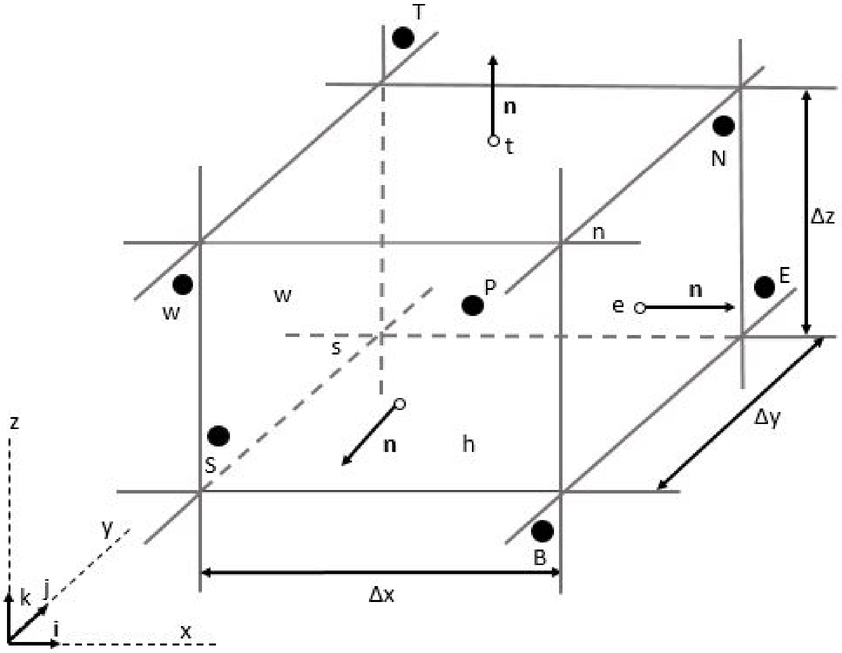


Figure 4.3: Control volume for a Cartesian 3D grid.

$$F_e = \int_{S_e} f dS \approx \frac{S_e}{2}(f_{ne} + f_{se}) \tag{4.6}$$

where f_{ne} and f_{se} are fluxes at the CV corners. Using higher order approximations such as Simpson's rule, which is a fourth-order approximation, requires evaluating the flux at more than two points.

4.2.3 Approximation of Volume Integrals

Some terms in the generic transport equation need integration over the volume of a CV. The simplest approximation is a second-order, which approximates the volume integral by the product of the mean value of the integrand and CV volume [48]

$$Q_p = \int_{\Omega} q d\Omega = \bar{q}\Delta\Omega \approx q_p \Delta\Omega \tag{4.7}$$

where q_p is the value of q at the CV center.

The discretization technique explained in this section applies to the momentum and continuity equations, however there are terms in the momentum equation which differ from those in the transport equation and require special attention.

4.2.4 Discretization of Convective and Viscous Terms

The convective term in the momentum equation introduces non-linearity and its differential and integral forms are as follows [48]

$$\frac{\partial(\rho u_i u_j)}{\partial x_j} \quad \textit{and} \quad \int_S \rho u_i v \cdot n dS \tag{4.8}$$

Same method as described for the convective term in the transport equation can be used to discretize the convective term in the momentum equation. However, since the viscous term in the momentum equation is a vector equation, the discretization technique is more complicated than the one in the transport equation and it requires further simplification to discretize the viscous term.

The viscous term in the momentum equation is [48]

$$\frac{\partial(\tau_{ij})}{\partial x_j} \quad \textit{and} \quad \int_S (\tau_{ij} i_j) \cdot n dS \tag{4.9}$$

where τ_{ij} is the stress tensor and for an incompressible and a Newtonian fluid can be expressed as

$$\tau_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) \tag{4.10}$$

As mentioned earlier, because the momentum equation is a vector equation, the viscous term is not identical to the diffusive term in the transport equation and a different treatment must be considered to discretize this term. Eq. (4.11) is a part of the viscous term that is similar to the diffusive term in the generic transport equation and can be discretized by using the same methods for the corresponding terms in the transport equation.

$$\frac{\partial}{\partial x_j}(\mu\frac{\partial u_i}{\partial x_j}) \quad and \quad \int_S \mu \nabla u_i \cdot ndS \tag{4.11}$$

The Eq. (4.12) is the additional term which is non-zero when the flow is incompressible and the viscosity is variable. Since this term goes to zero when the viscosity ia constant, it is treated separately.

$$\frac{\partial}{\partial x_j}(\mu\frac{\partial u_j}{\partial x_i}) \quad and \quad \int_S (\mu\frac{\partial u_j}{\partial x_i} i_j) \cdot ndS \tag{4.12}$$

4.2.5 Discretization of Pressure Term

To obtain the pressure equation, the following semi-discretized form of the momentum equation will be used [49]

$$a_p U_p = H - \nabla P \tag{4.13}$$

where H is the matrix coefficients for all neighbors multiplied by their velocities and corresponding time values. The continuity equation can be discretized in the following form

$$\nabla \cdot U = \sum_f A \cdot U_f = 0 \tag{4.14}$$

where U can be found from:

$$U_p = \frac{H}{a_p} - \frac{1}{a_p}\nabla P \tag{4.15}$$

Applying the face interpolation into Eq. (4.15) gives:

$$U_f = (\frac{H}{a_p})_f - (\frac{\nabla P}{a_p})_f \tag{4.16}$$

Substituting Eq. (4.16) into Eq. (4.14) yields:

$$\nabla \cdot (\frac{\nabla P}{a_p}) = \nabla \cdot (\frac{H}{a_p}) = \sum_f A \cdot (\frac{H}{a_p})_f \tag{4.17}$$

Both sides of Eq. (4.17) can be discretized by using similar methods which were used to discretize the diffusion term. The final form of discretized incompressible Navier-Stokes equations are:

$$a_p U_p = H - \sum_f A \cdot P_f \tag{4.18}$$

$$\sum_f A \cdot [(\frac{1}{a_p})_f \nabla P_f] = \sum_f A \cdot (\frac{H}{a_p})_f \tag{4.19}$$

At this point, the SIMPLE (Semi-implicit Method for Pressure-Linked Equations) algorithm can be used to solve the Navier-Stokes equations with an iterative procedure. The iterative procedure can be summarized as follows:

1. Providing boundary conditions for unknown parameters such as velocity and pressure fields.
2. Solving the discretized momentum equation for the mentioned unknown parameters in step 1.
3. Using step 2 values for the velocity and pressure fields to solve the momentum equation for the new approximated velocity field.
4. Using the velocity obtained in step 3 to generate implementation of H to the pressure equation.

5. Calculating the new pressure field by using the updated values obtained in steps 3 and 4.

6. Correcting the velocity field by using an iterative approach on the basis of the new pressure field, which eventually leads to the corrected pressure field.

7. Repeating steps 4-6 until the unknown variables stop varying.

8. By using the obtained values as initial guesses, the next time step is generated for the next loop which eventually gives a new solution.

4.3 Magnetic Field Solution

The appropriate equation for solving the magnetic field generated by the permanent magnets is:

$$B = \mu_0(H + M) \tag{4.20}$$

where B is the magnetic flux density, H is the magnetic field intensity, M is the remanence field, and μ_0 is the permeability constant. In a current free region, where

$$\nabla \times H = 0 \tag{4.21}$$

it is possible to define the magnetic scalar potential, ψ, from the relation

$$H = -\nabla\psi \tag{4.22}$$

By using the Eq. (4.20) together with the Eq. (3.3), the Poisson's equation can be obtained as:

$$-\nabla \cdot (\mu_0 \nabla \psi - \mu_0 M_0) = 0 \tag{4.23}$$

The OpenFOAM uses the Poisson's equation to first solve the magnetic scalar potential, ψ, from which the magnetic flux density, B ,and the magnetic field intensity, H ,are obtained.

4.4 Lagrangian Particle Tracking

Lagrangian particle tracking (LPT) is a numerical method for tracking individual particles within an Eulerian phase. This method is also known as Discrete Particle Simulation (DPS) and is valid for particles smaller than the grid size. In a Lagrangian frame, the position vector $\vec{X}_p$ of each particle is obtained from the equation:

$$\frac{d\vec{X}_p}{dt} = \vec{U}_p \tag{4.24}$$

where the $\vec{U}_p$ is the velocity of the particle P and the position X_p is located at the center of the particle. Particle P is defined with its diameter D_p and its density ρ_p.

The motion of each particle with a mass of m_p is governed by Newton's second law:

$$m_p \frac{d\vec{U}_p}{dt} = \sum \vec{F} \tag{4.25}$$

where $\sum \vec{F}$ as discussed in chapter 3 is the sum of the magnetic, drag, and gravitational forces. Hence Eq. (4.25) can be rewritten as:

$$m_p \frac{d\vec{U}_p}{dt} = \vec{F}_m + \vec{F}_d + \vec{F}_g \tag{4.26}$$

By substituting the corresponding expressions for $\vec{F}_m$, $\vec{F}_d$, and $\vec{F}_g$ into Eq. (4.26) and rearranging the equation with respect to $\vec{U}_p$, an appropriate differential equation for the velocity of a particle can be obtained as:

$$\frac{d\vec{U}_p}{dt} + a\vec{U}_p = b \tag{4.27}$$

where a and b are constants. Finally, the position of a particle at each time step can be obtained by solving the differential Eq. (4.27) and substituting the $\vec{U}_p$ values into Eq. (4.24).

4.5 OpenFOAM Standard Solvers

In order to solve the equations effectively, it is required to use appropriate solvers. OpenFOAM includes different linear solvers, i.e. iterative solvers for linear sets of equations $Ax = b$. For this research, the Generalized Geometric-Algebraic Multi-Grid Solver (GAMG) and the Preconditioned Conjugate Gradient Solver (PCG) are the most efficient solvers that are discussed in the following.

Multi-grid (MG) methods are algorithms used in numerical analysis for solving differential equations using coarse grid with fast solution time intervals to smoothen out high frequency errors and to provide a starting solution for the finer grid. The Generalized Geometric-Algebraic MG solver follows the same logic, where an initial solution is first made by solving a coarse problem compared to the grid first made by the user, and then the solution is mapped onto a finer mesh. This method is suitable for solving the pressure equation which requires the same approach. In other words, the solution starts by an initial guess and the solution generated by the coarse mesh is used as a boundary condition for the finer mesh. In this way, initial guess is improved by solving the coarse problem first.

In addition, the Preconditioned Conjugate Gradient Solver is another solver that is very effective in solving a large sparse symmetric positive-definite system. The PCG is an iterative algorithm for the numerical solution of systems of linear equations, where the matrix is symmetric and positive-definite. This algorithm is applicable to sparse systems, arise when numerically solving partial differential equations, which are too large to be handled by a direct implementation.

4.6 Mesh Generation

In computational solutions of partial differential equations, discretizing the entire computational domain into subdomains is of a great importance. Meshing is a discrete representation of the geometry that is involved in the problem. A finer mesh commonly gives better calculation results, but the main question is, what is the best resolution of the mesh that gives reasonably accurate results?

The mesh quality depends on different factors such as, rate of convergence, solution accuracy, and CPU time required to solve the problem. The better the mesh quality, the better the convergence rate, which means the solution has been achieved in a shorter time. Also more accurate results can be obtained by improving the mesh quality. For example, at specific spots of the geometry where the gradients are high, increasing the resolution in that area provides a more accurate solution at that point. CPU time is another factor which is considered as an undesirable factor. As the mesh gets finer, the larger CPU time is required to solve the equations.

Whether the problem is 2-D or 3-D, different cell shapes are used to discretize the geometry into subdomains. In 2-D problems, triangle and quadrilateral and in 3-D problems, tetrahedron, pyramid, triangular prism, and hexahedron are the most common cell shapes. (Figure 4.4)

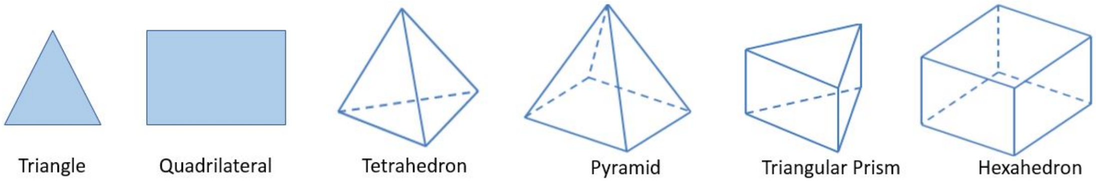


Figure 4.4: Basic two-dimensional and three-dimensional cell shapes.

Grids are classified into three main categories. Structure grids, unstructured grids, and hybrid grids. Structure grids follow a uniform pattern and are comprised of quadrilateral in 2-D and hexahedra in 3-D elements. Unstructured grids do not follow a uniform pattern and any of the mentioned elements can be used to create the mesh. Finally, the hybrid grid contains a mixture of both structured and unstructured grids [50]. Among mentioned grids, the structured grid is well suited for many applications and if the accuracy of solution is of highest concern, this type of grid is the most preferable one. It must be noted that, in order to be able to capture all the details, the density of the mesh is required to be high at the areas with highest gradients.

In this study, a structured grid was used to discretize the computational domain. As discussed earlier in this chapter, separation process requires solving appropriate equations for three different cases. In simulating the magnetic field, the surface and the interfaces of the magnets are the areas with highest gradients, thus the mesh at these regions needs refinement to capture every detail. In modeling the fluid flow, the gradients are near walls and the mesh at these regions needs refinement. Also, the mesh needs to be fine enough to ensure the particle motion is accurate through the modeling domain. The mesh refinement can be done by either increasing the total number of cells per unit area or by clustering the mesh at mentioned areas. In general, the total number of cells in mesh clustering is lower than the mesh refinement technique, which leads to a faster solution. Also, clusterization provides more accurate results at the areas with higher gradients and more details can be captured if this method is used to discretize the domain.

Studying different grids for solving the magnetic field, flow field, and particle motion equations, revealed the fact that the magnetic field requires a finer mesh to capture all the details, comparing to the other two physics. Hence, a grid study has been done to find the best grid for solving the magnetic field, and the obtained grid was further used to ensure that it gives reasonable results for the flow field.

As discussed above, the surface and the interfaces of the magnets are of great importance and as can be seen in Figure 4.5 and Figure 4.6, the mesh is clustered at these regions and is uniform elsewhere.

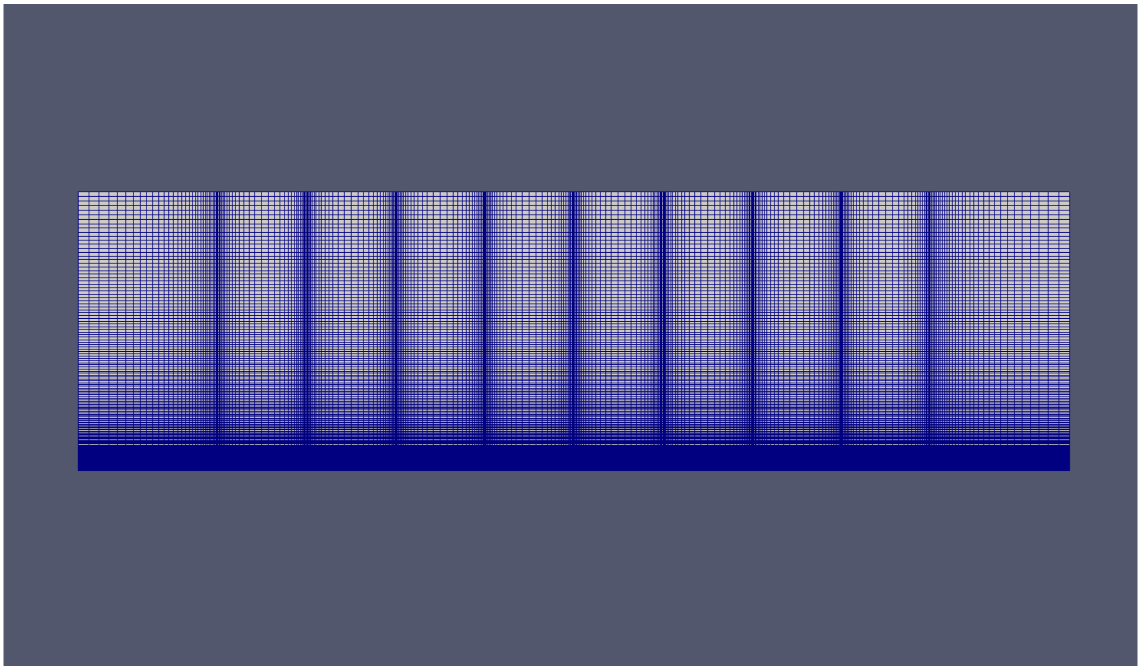


Figure 4.5: Clustered mesh near the surface and the interfaces of the magnets, front-view.

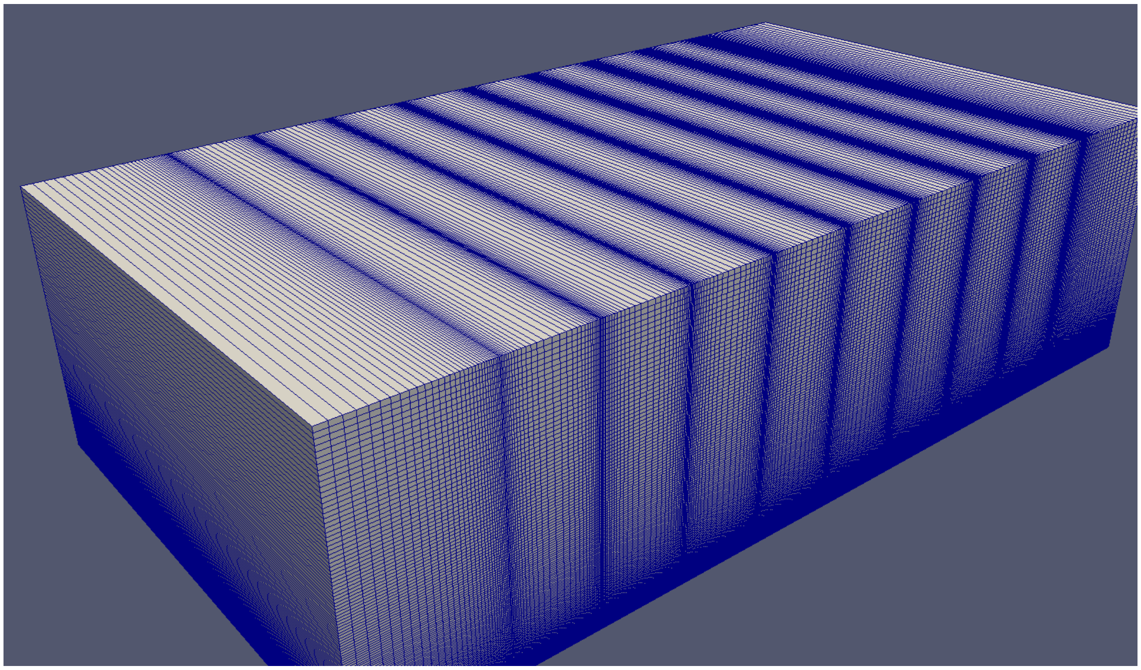


Figure 4.6: Clustered mesh near the surface and the interfaces of the magnets, edge-view.

4.7 Grid Study

To model the magnetic field, it is required to first solve the equations within a very large computational domain around the magnets, and then based on the location of the channel, the corresponding magnetic field values are mapped onto the channel. As a result, it is very important to choose an appropriate grid for the smaller domain, which is the channel, to ensure it represents the correct values when the magnetic field values are mapped onto it. Hence, a super fine mesh (660 × 6000) was used to discretize the larger domain and a grid study was performed on the channel domain to find the best mesh that represents the most reasonable results. It is worth mentioning that, due to memory limits, the mesh in the larger domain could not be any finer and the grid size 660 × 6000 was the ultimate choice. In other words, the result obtained from the super fine mesh in the larger domain was considered as the most accurate result that could be obtained by using the computers in the lab. Because the mesh inside the channel is coarser than the mesh inside the larger domain, the values obtained from mapping the data onto the channel would necessarily have some error, which the goal of grid study is to minimize this error.

Determining the correct parameter and location that grid study will be performed are crucial. Therefore, since the magnetic force is the most important parameter that will be used in finding the particle trajectory, it is required to perform the grid study based on this parameter. Also, the channel is located 1 mm away from the surface of the magnets and the channel height is 200 μm. Consequently, the distance of 1.2 mm from the surface of the magnets was chosen for the purpose of grid study. It is worth mentioning that, the mesh is clustered at the surface of the magnets and as the distance increases from the surface, the mesh becomes coarser. Thus, if the chosen grid in the channel gives reasonable results for the magnetic force at the top surface of the channel, we can ensure the magnetic force inside the channel is also acceptable.

An appropriate mesh size can be obtained by performing grid studies in x and y directions. In the following, N_x and N_y represent the number of mesh in x and y-directions, respectively. First, a proper N_y was obtained by investigating the effect of mesh size in y-direction on the discretization error, while N_x was kept constant and equal to $N_x = 660$. Then, a similar approach was used to find a proper value for N_x. This time N_x was the parameter to study and N_y was kept constant at the value obtained from the previous step, $N_y = 50$.

Figure 4.7 indicates the change of magnetic force error as a result of grid refinement in y-direction. To obtain the magnetic force discretization error, the magnetic force values along x-axis at y = 200 (μm) were obtained for different mesh sizes and were compared with the values obtained from the case with the finest mesh. As can be seen in Figure 4.7, the discretization error decreases with an increase in mesh refinement in y-direction and for $N_y = 40$ and $N_y = 50$, the error is below 1% and either of these two mesh sizes can be used to obtain the magnetic force values within 1% error. To be more accurate, $N_y = 50$

was chosen as the number of mesh in y-direction. The next step is to find a proper N_x value by performing grid study in x-direction, while N_y is constant and equal to 50.

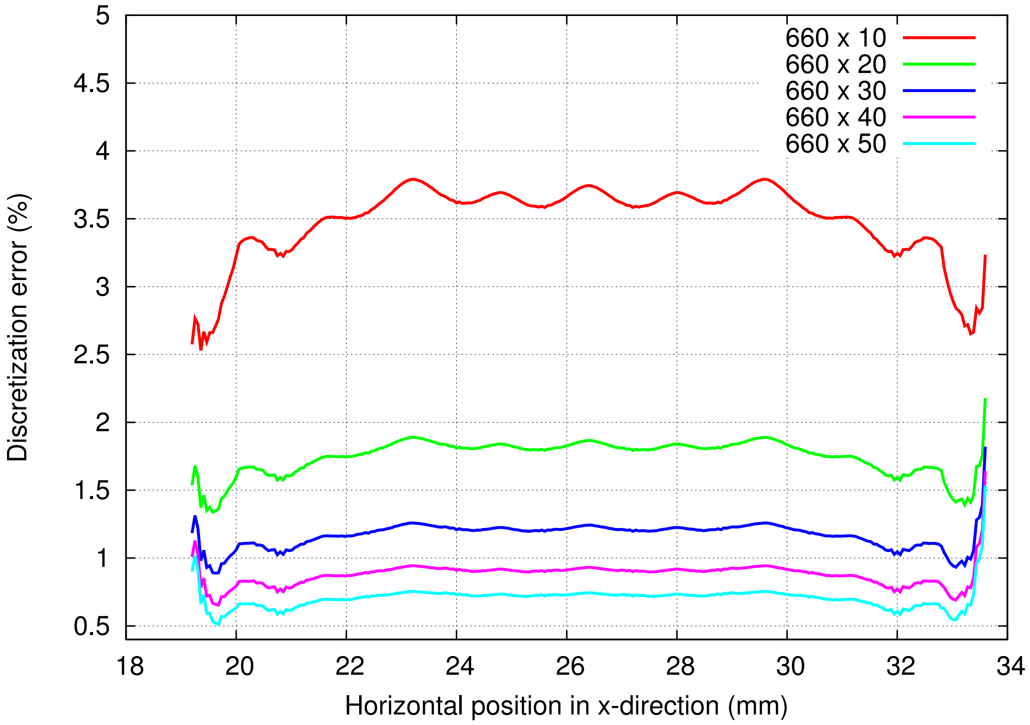


Figure 4.7: Magnetic force discretization error along x-axis at y $= 200\,(\mu m)$ for five different mesh sizes. Grid refinement is in y-direction.

Figure 4.8 illustrates the change of magnetic force error as a result of grid refinement in x-direction. As can be seen, the discretization error due to inadequate grid resolution is high and as the mesh gets finer, $N_x = 275$ and $N_x = 365$, the error curve becomes smoother and it reduces to below 0.8%. To maintain a good level of accuracy, the grid resolution of 365×50 was chosen for our study to solve the appropriate partial differential equations within 1% error.

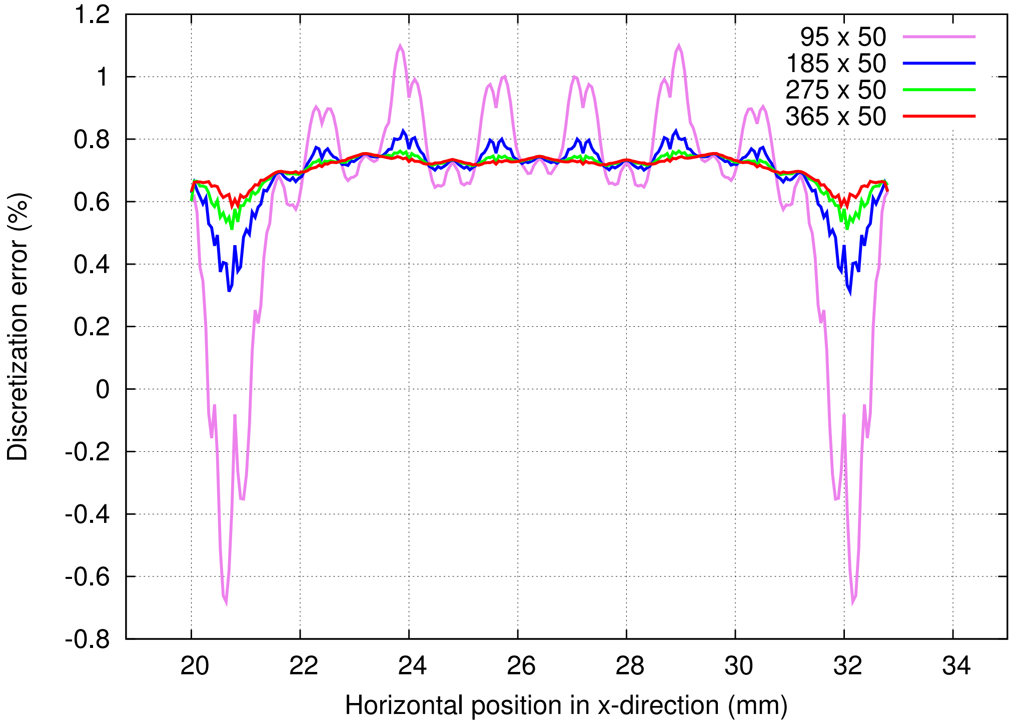


Figure 4.8: Magnetic force discretization error along x-axis at y $= 200\,(\mu m)$ for four different mesh sizes. Grid refinement is in x-direction.

To ensure the chosen grid size was suitable for solving the flow field equations, the simulation result for the velocity profile inside the channel was compared with the analytical solution of poiseuille flow inside a rectangular micro-channel. As shown in Figure 4.9, the simulation result agrees very well with the analytical solution and verifies the validity of the chosen mesh size for solving Navier-Stokes equations.

Since the flow inside the channel is assumed to be steady-state and fully-developed, it is required to monitor the residuals of all equations to make sure the approximate solution satisfies them. As a result, Figure 4.10 has been provided to illustrate the convergence of the pressure and velocity equations that are steadily progressing towards the final solution.

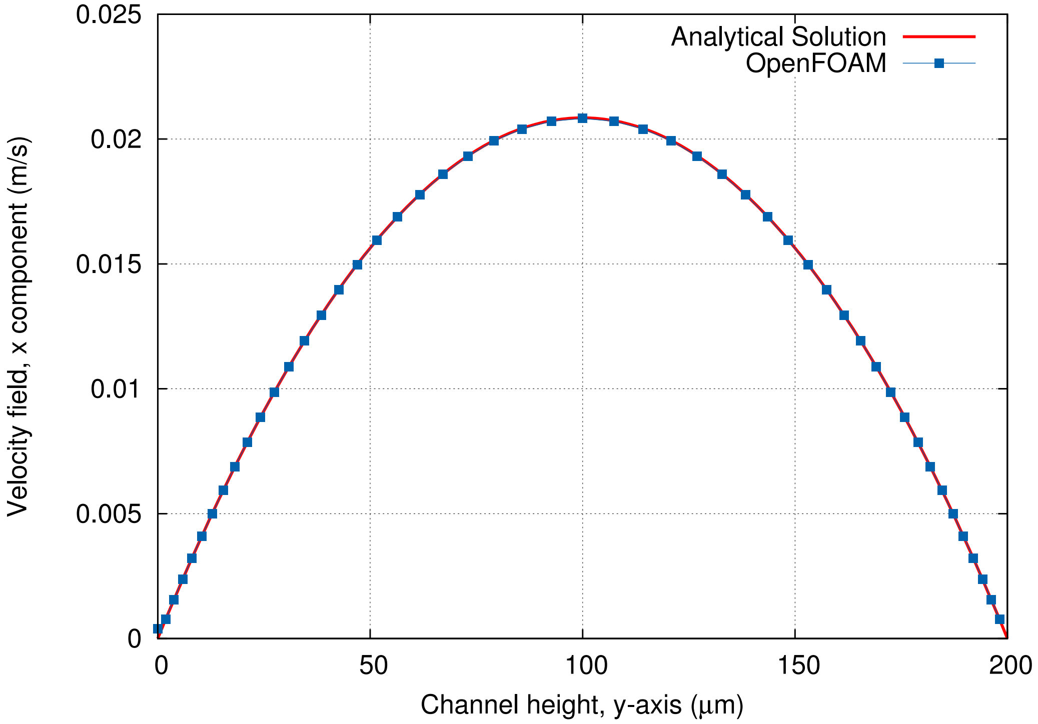


Figure 4.9: The simulation result and analytical solution for the velocity profile inside the channel in a fully-developed laminar flow case.

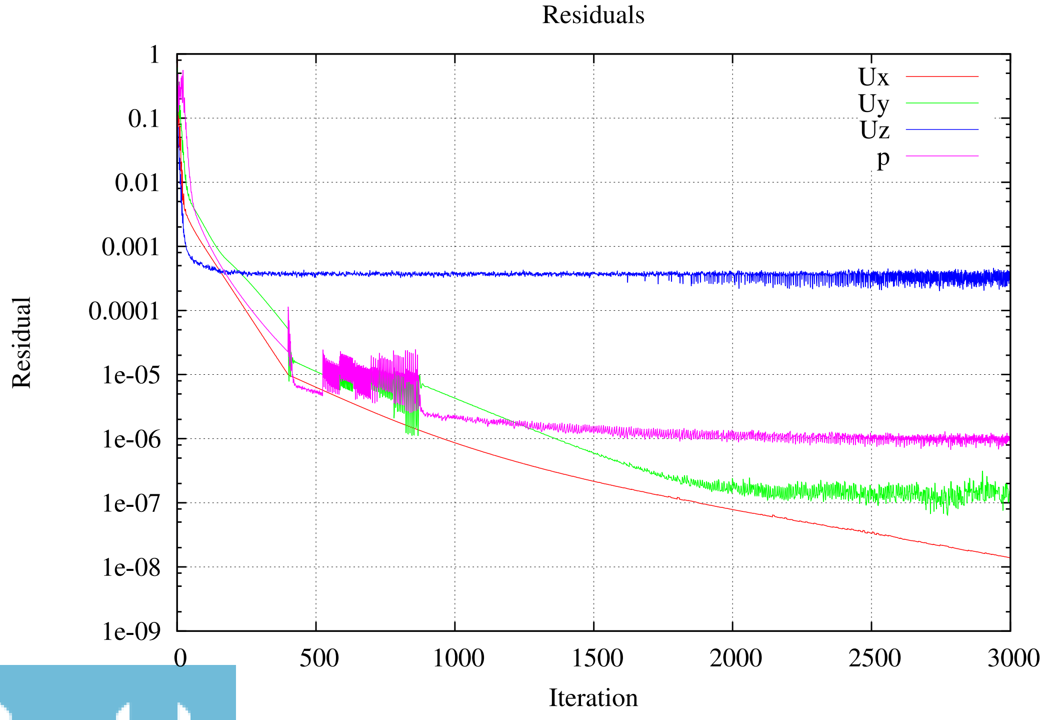


Figure 4.10: Residual monitors for the fluid flow simulation inside the channel.

CHAPTER 5

THE EXPERIMENTAL SETUP

This chapter describes the experimental setup with a focus on the development of a magnetophoretic bioseparation device. The chip was designed and fabricated to further verify the simulation results obtained in this study.

5.1 Design and Development

The chip consists of a bottom glass substrate, inlet outlet ports, a thin double-sided polyimide tape in which the separation channel is made, and a top coverslip. The bottom glass substrate is a microscope coverslip with dimensions of 25 mm × 75 mm × 1 mm. The top substrate is another coverslip with the same dimensions, but a thinner thickness of 160 μm. This thin coverslip made it possible to view the 1 μm magnetic particles at high magnifications. Two 1 mm holes were drilled in the top substrate for the inlet and outlet ports. The flow channel was cut out of a 100 μm thin double-sided polyimide tape and sandwiched between the bottom and top substrates. A length of two plastic tubes were carefully attached to the inlet and outlet ports, located on the top substrate. An array of 16 NdFeB magnets were implemented beside the chip to produce magnetic field within the channel. The block NdFeB magnets were grade 52 and were purchased from the K&J Magnetic company [51]. The overall dimensions of the magnets were 19 mm × 1.6 mm × 6.35 mm.

The pattern of the microfluidic channel was drawn using AutoCAD software. Since modeling observations were based on the side-view of the channel, the attempt was to come up with a design to give us the capability to monitor the trapping of the magnetic beads from the side-view. After investigating different aspects and limitations of the experiment, such as the location and position of the microscope lens and magnets with respect to the chip, the following design was suggested in order to be able to see the particles from the side-view. (Figure 5.1).

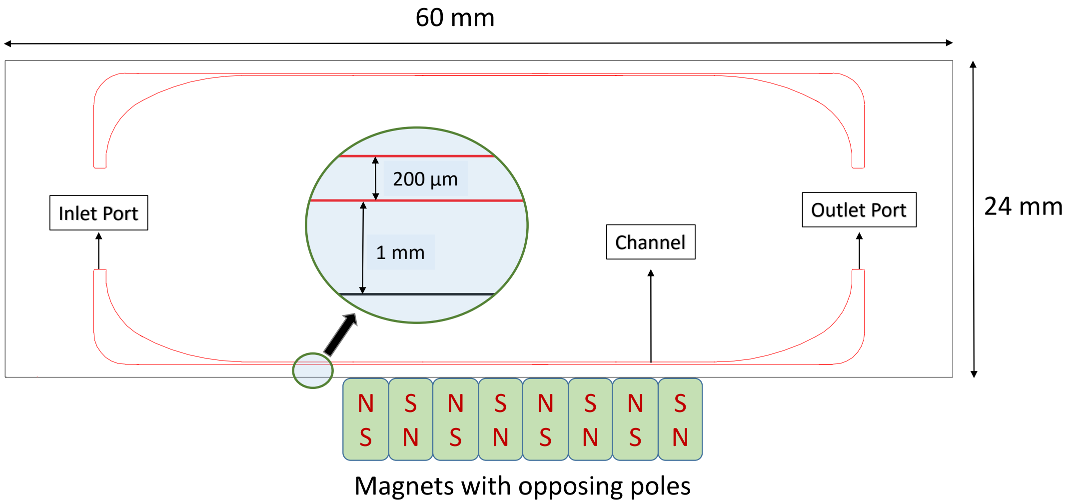
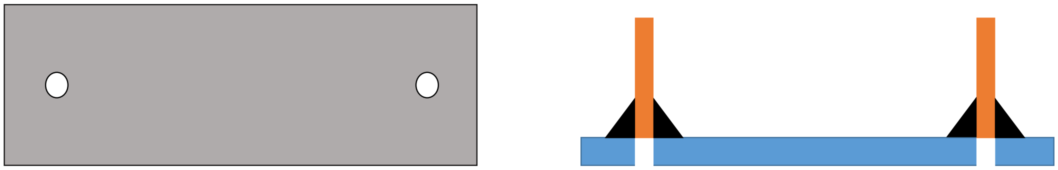


Figure 5.1: The AutoCAD drawing of the microfluidic channel.

As can be seen in Figure 5.1, the design of the channel is such that the distance between the bottom surface of the channel and the magnets is 1 mm and the channel height is 200 μm. The suggested design provides this possibility to monitor the magnetic particles trajectories along the channel. A sample containing magnetic beads can be introduced into the chip from the inlet port and leave the channel from the outlet port to the waste collection tube. The depth of the channel is equal to the thickness of the double-sided polyimide, which is 100 μm. It is worth mentioning that, due to common sources of error in physics lab experiments, it is required to fabricate chips until the desirable device is obtained. As a result, to save time and material, two identical channels were considered for each device. However, the smoothest and the most desirable one was chosen to run the sample.

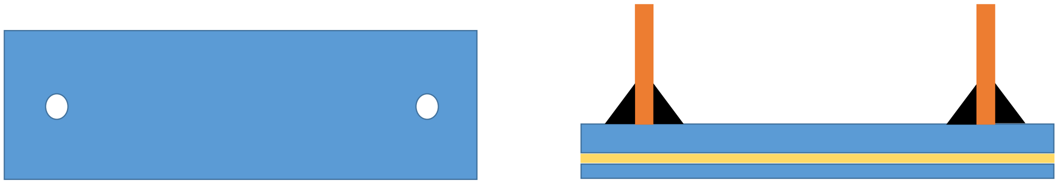
5.2 Fabrication of the Chip

Figure 5.2 shows the fabrication process of a magnetic bioseparation chip by using glass substrates. First, a mask was designed by using the AutoCAD software to drill two holes for the inlet and outlet ports in a glass substrate. As mentioned in the previous section, the microfluidic channel and the mask were drawn on a 100 μm double-sided polyimide tape and a masking tape, respectively. A Graphtec CraftRobo Pro Plotter (Graphtec USA, Santa Ana, CA) was used for the drawings. Next step was to remove the excess material by a tweezer to make the microfluidic channel out of the polyimide tape. By using the masking tape for the inlet and outlet ports, the corresponding ports were drilled into a microscope slide. The substrate was then cleaned and two pieces of Tygon tubing were attached to the substrate by using epoxy adhesive and cured for 30 minutes in the oven. While waiting for the tubes to bond to the substrate, the other substrate was cleaned with alcohol and the fluidic channels that were made in the polyimide tape were carefully aligned and attached to it. The cured substrate was also cleaned with alcohol and attached to the other side of the polyimide tape, such that the tape was sandwiched between the bottom and top substrates. Finally, the chip was clamped by clips and placed inside the oven and cured at $70°C$ for 3 hours. Figure 5.3 shows the final image of the fabricated chip. The next step is to place the channel and magnets under the microscope such that the magnetic field can be produced within the channel for the purpose of capturing the magnetic beads.



a. Mask for drilling inlet and outlet holes on the top substrate.

c. Bonding inlet and outlet tubes by epoxy adhesive.



b. Drilling holes on the substrate.

d. Bonding the substrate, double-sided tape and the coverslip together.

Figure 5.2: Fabrication process of a magnetic bioseparation chip.

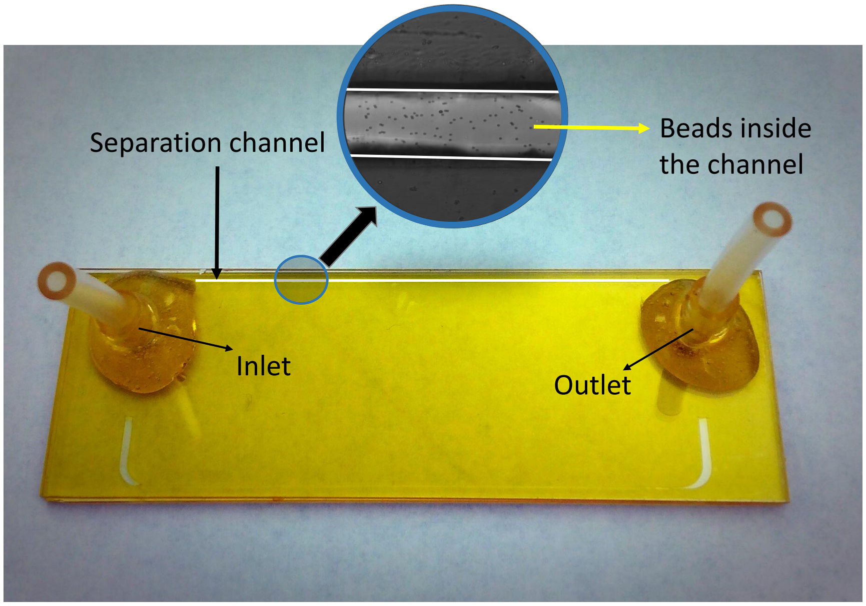


Figure 5.3: The fabricated glass based microfluidic device.

5.3 The Experimental Procedure

As shown in Figure 5.4, the experimental setup consists of a bidirectional MilliGat pump with a MicroLynx controller, a syringe pump for the buffer, light microscope to monitor the particle motion within the channel, and plastic tubing for the connections between the pumps and the chip.

To perform the experiment, a 1:100 dilution sample was prepared by mixing 10 μL magnetic beads with 990 μL deionized water mixing it using a microcentrifuge under gentle shaking. The diameter of the magnetic beads was chosen as 1 μm. Before injecting the solution inside the channel, the chip and tubing were washed with deionized water and soaked with 20 % bovine serum albumin (BSA) for 30 minutes and rinsed with an isolation buffer. Next, by using the bidirectional milliGAT pump, 1 mL of the sample was withdrawn into the tubing which was then connected to the inlet port of the chip. To obtain an average velocity of 13.9 mm/s within the channel, the flow rate was set to 0.27 μL/s using the milliGAT pump. While the sample is flowing through the channel, it is possible to observe particles motion and trajectories within the channel, however, capturing good quality images requires high speed imaging systems. In the absence of a high speed imaging technology, which is the case here, the sample pump must be stopped to be able to take images using the integrated camera of the microscope. The camera was then moved along the channel to monitor particle trapping at various distances from the inlet port. The results have been provided in the Results and Discussion part. Once the experiment was done, the magnets were removed from the chip and the captured beads were eluted to a collection tube by flowing buffer through the channel.

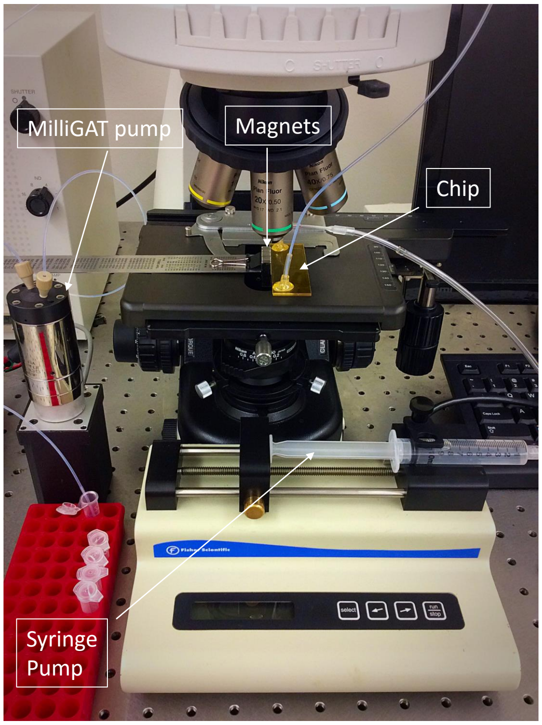


Figure 5.4: An image of the experimental setup and an assembled chip.

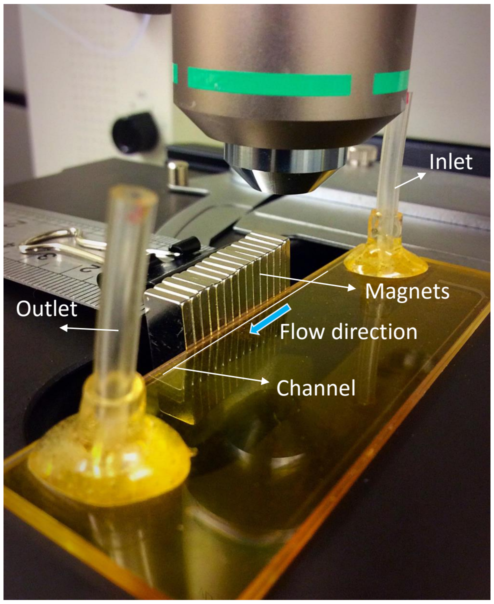


Figure 5.5: An assembled chip with external magnets placed beside the chip.

CHAPTER 6

RESULTS AND DISCUSSION

In this section, results and discussion for the magnetic field and particles motion will be presented. The magnetic field was simulated for an array of magnets with opposing poles and the results were then used to study the behavior of particles motion in the presence of the magnetic field. A parametric study was also performed to study the effect of flow rate, particle size, bead size, and number of beads per cell on the particle trajectory. Finally, the numerical simulation results were verified through experimental observations, where the fabricated microfluidic chip was tested to observe the transport and capture of the magnetic beads inside the microchannel.

6.1 Magnetic Field Simulation

The permanent magnets used in this study were Neodymium (NdFeB) block magnets grade N52 with overall dimensions of $19\,mm \times 1.6\,mm \times 6.35\,mm$ (3/4" × 1/16" × 1/4"). These NdFeB magnets were purchased from the K&J Magnetics company and are the strongest magnets available [51]. The magnets are magnetized through the 6.35 mm dimension. In the following, the computational modeling results of a single magnet and an array of magnets are discussed.

6.1.1 Modeling of a Single Magnet

A 3-D model of a single block magnet was created in OpenFOAM software. Eq. (4.20) is solved in OpenFOAM to obtain the magnetic flux density. For NdFeB N52 grade magnets, a relative permeability of 1.05 and a normal value of M = 1083769 A/m for the remanence field were used [52].

Figure 6.1 shows the variation of the magnetic flux density norm along vertical center line of the magnet. As can be seen, on the surface of the magnet the magnetic flux density is 0.7 T and it decreases as the distance from the surface of the magnet increases.

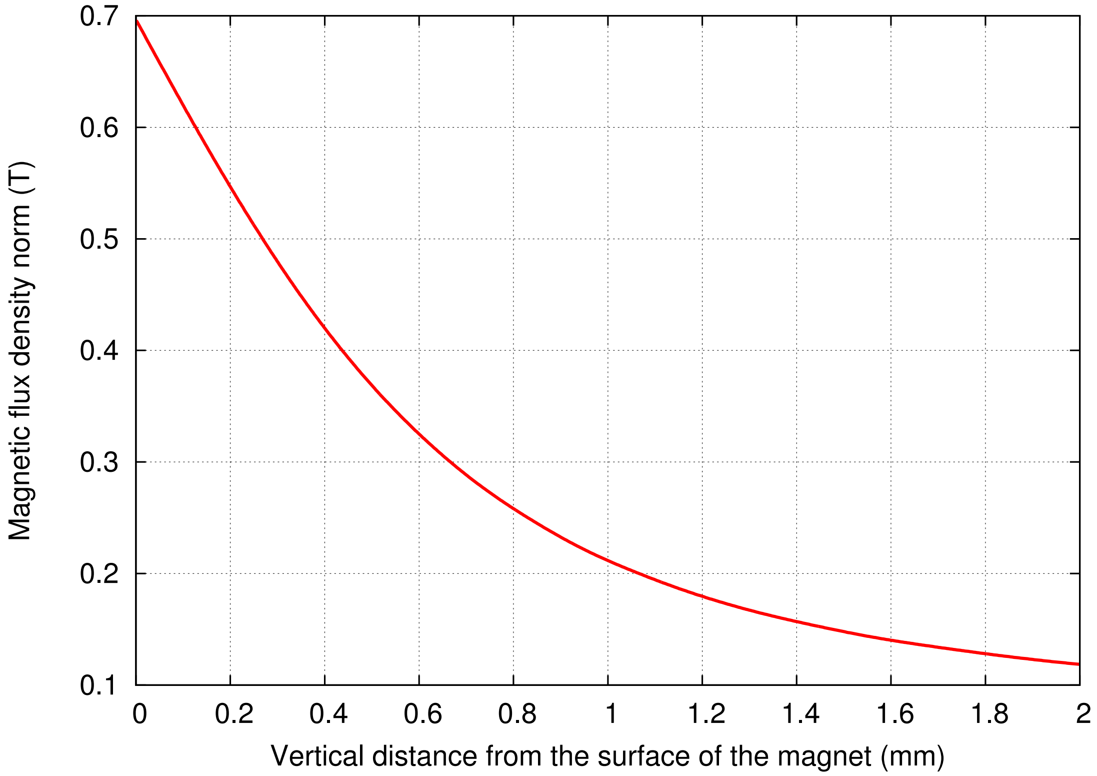


Figure 6.1: Magnetic flux density norm along vertical center line of a magnet.

6.1.2 Modeling an Array of Magnets

To investigate the effect of an array of magnets on the magnetic field distribution, the number of magnets was varied from 2 to 16. A repeatable magnetic field pattern at the interface of the magnets was observed when more than eight magnets was used. Figure 6.2 shows the distribution of the magnetic field for an array of 8 magnets. As can be seen, the magnetic flux density is nearly the same at the surface of each magnet and reaches its maximum value at the interface of the magnets, which is due to the alternating polarity arrangement (first magnet has north facing up, second magnet has south facing up, third magnet has north facing up, etc.). Therefore, the magnetic field along the channel has a repeating pattern between two peaks. Also, as the distance from the surface of the magnets increases, the peak values of the magnetic field decreases.

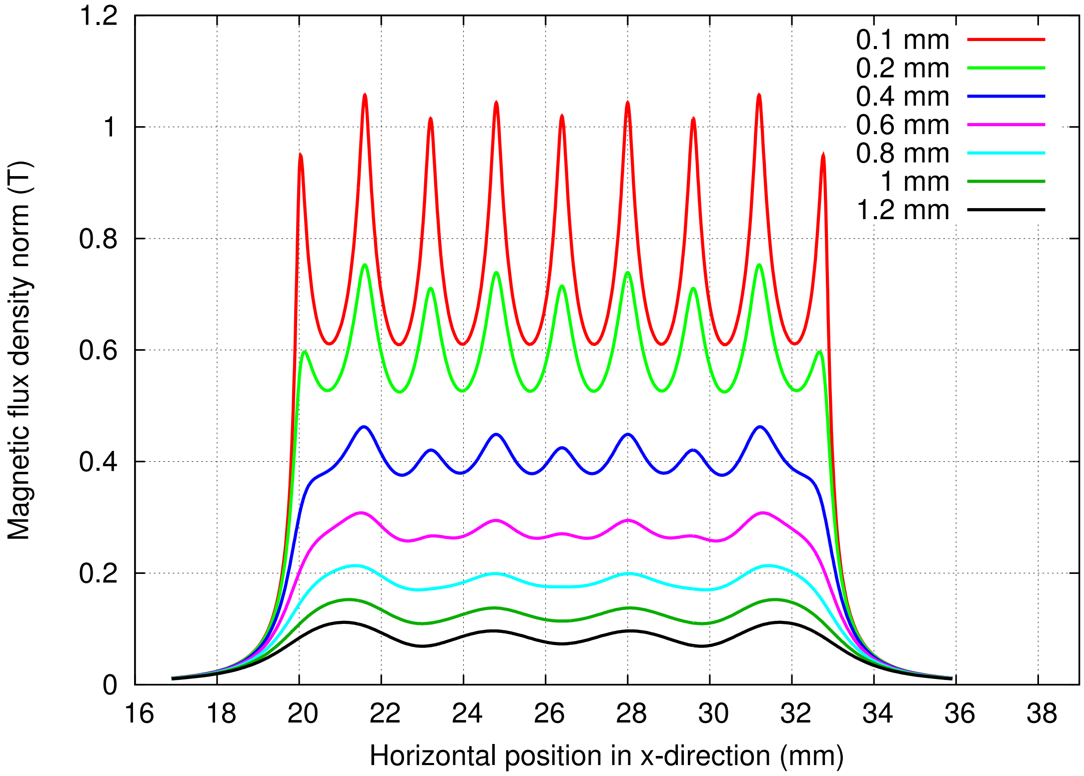


Figure 6.2: Normal magnetic flux density distribution for an array of 8 magnets at various distances from the surface of the magnets.

Figure 6.2 can also be used to determine whether the channel is in the saturated or unsaturated region. As shown in Figure 3.1, superparamagnetic particles such as magnetic beads are saturated at magnetic fields greater than 0.2 T, and therefore the corresponding magnetic force is given by Eq. (3.16). Below saturation, where the magnetic particles are unsaturated, the magnetic force corresponds to Eq. (3.18). The location of the bottom of the channel is at the distance of 1 mm away from the surface of the magnets and since the channel height is 200 μm, the top surface of the channel is at the distance of 1.2 mm from the magnets. According to Figure 6.1, the magnetic flux density within the channel varies from 0.1 T to 0.2 T. Thus, it can be concluded that the region within the channel is in the transitional zone from unsaturated to saturated regimes (Figure 3.1). Unfortunately, there is no analytical solution for the magnetic force in this region, however, reasonable assumptions can be made to use the saturated force as an approximation for the magnetic force within the channel.

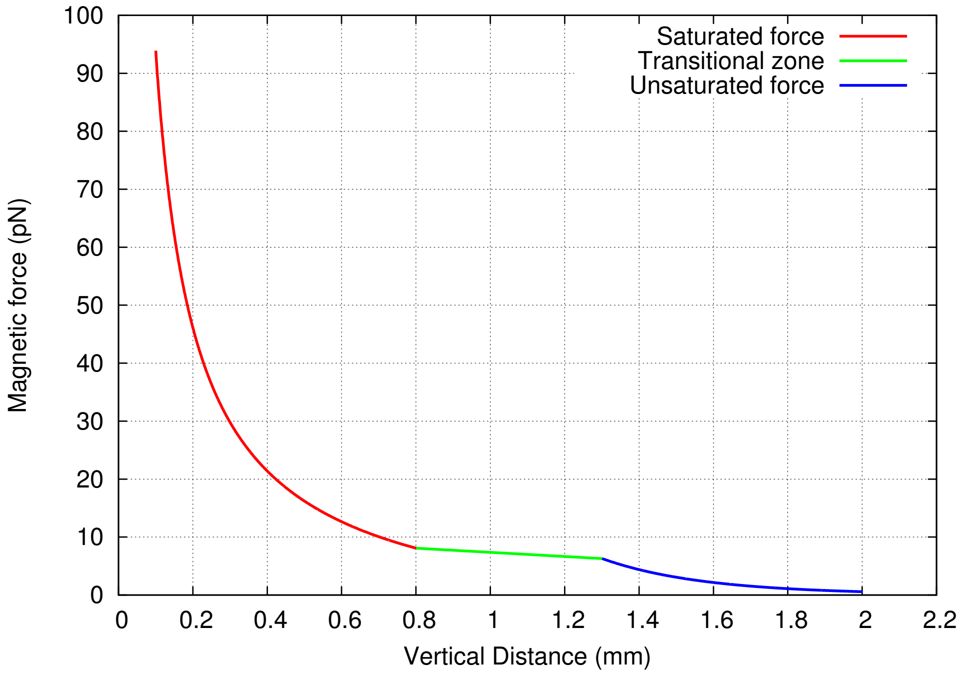


Figure 6.3: Magnetic force norm along vertical center line of a magnet.

Figure 6.3 shows the variation of the magnetic force norm along vertical center line. As can be seen, superparamagnetic beads are saturated up to y = 0.8 mm from the surface of the magnets and the saturated force is applied to them. From y = 0.8 mm to y = 1.3 mm, which involves the channel domain, the transitional zone from saturated to unsaturated regime exists, and above y = 1.3 mm, the particles are in the unsaturated region and the unsaturated force is applied to them. The slope of the green line, which was drawn to show the transitional region, is closer to the slope of the saturated force curve, thus it's more suitable to use the saturated force as the dominant force within the channel. Also, it is worth mentioning that, the saturated magnetic force gives a smaller magnetic force within the channel compared to the unsaturated force, and the particle separation results would be more conservative when the saturated magnetic force is used.

Figure 6.4 shows the variation of the x-component of the magnetic force, F_x, along the channel at various distances from the surface of the magnets. Due to the alternating polarity arrangement of the magnets, F_x changes direction from one magnet to another. This oscillatory feature of the magnetic force in x-direction is more visible at distances near the surface of the magnets and as the distance from the magnets increases to 1 mm away from the surface, the location of the channel, the magnetic force in x-direction largely reduces.

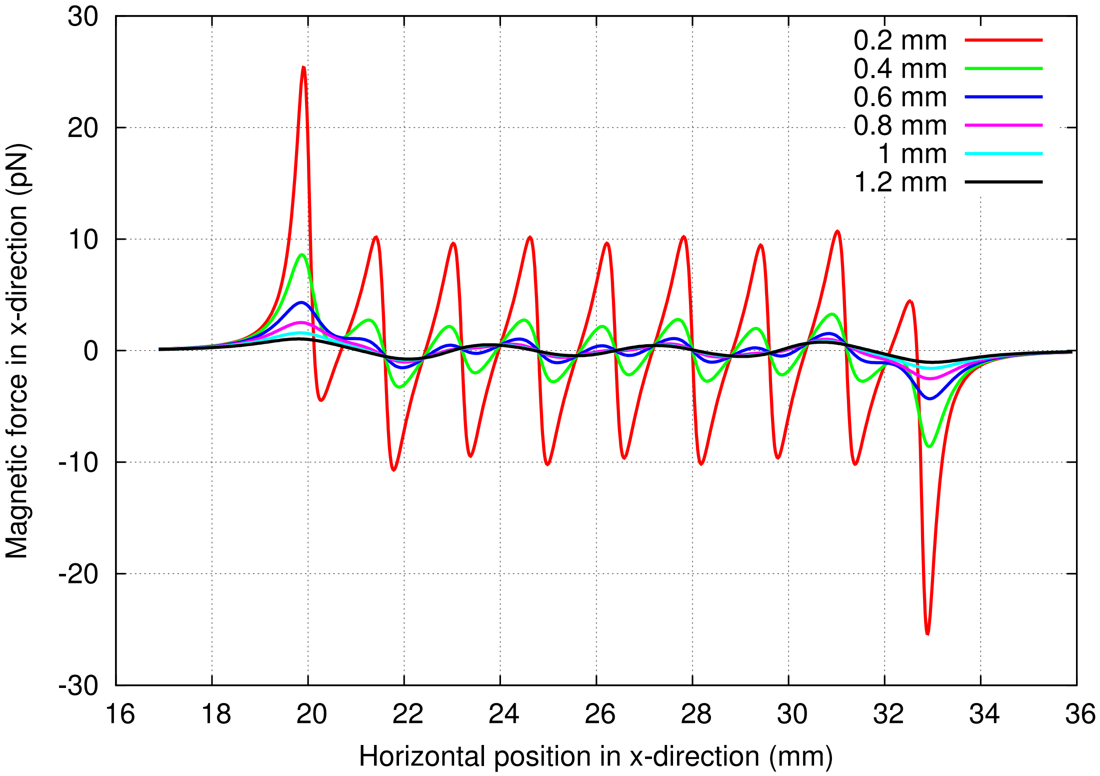


Figure 6.4: Variation of the x-component of the magnetic force along the channel at various distances from the surface of the magnets.

Figure 6.5 depicts the variation of the y-component of the magnetic force, F_y, along the channel at various distances from the surface of the magnets. The negative sign indicates the direction of F_y which is towards the surface of the magnets. This downward force is one of the advantages of using an array of magnets to capture the particles. As discussed in the literature review section, when soft magnetic elements are used to isolate particles, the y component of the magnetic force oscillates along the channel and at some regions the magnets repel the magnetic particles, resulting in lower capture efficiencies. Furthermore, by comparing F_x and F_y values, it can be concluded that the dominant component of the magnetic force inside the channel is the y component of the force, which is approximately 5 times larger than F_x.

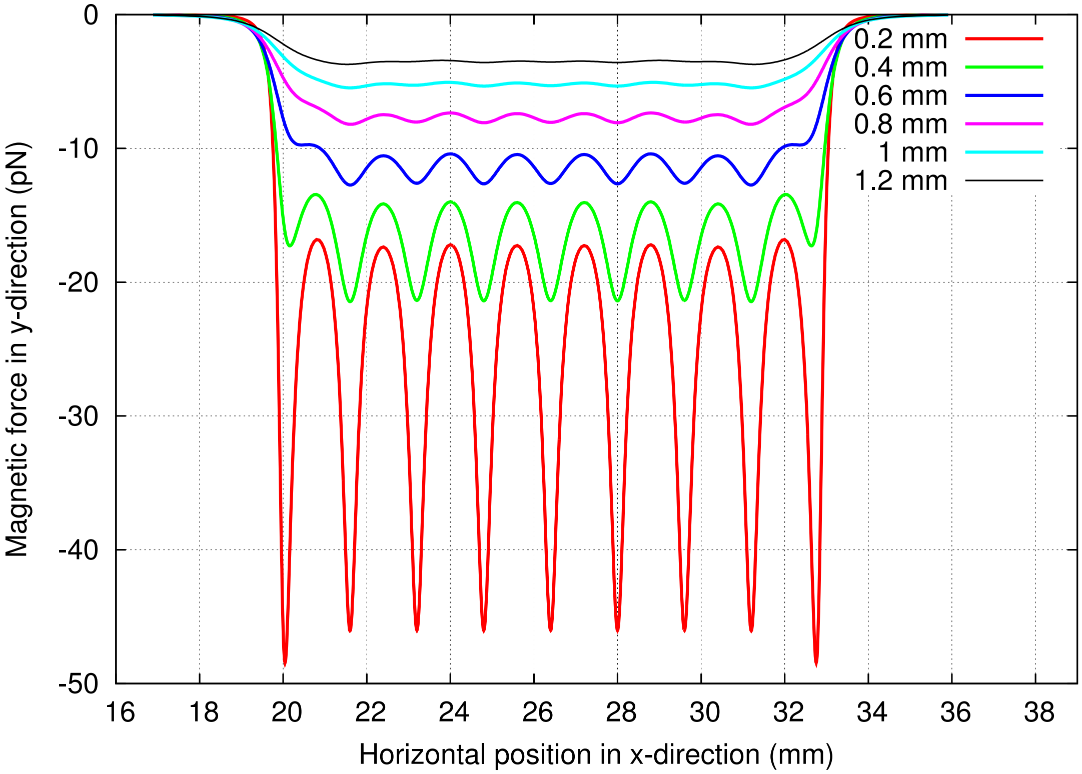


Figure 6.5: Variation of the y-component of the magnetic force along the channel at various distances from the surface of the magnets.

The magnetic force norm along the channel at different distances from the surface of the magnets is shown in Figure 6.6. Here, the magnetic force has its highest values at the interface of the magnets. Due to polarity arrangement of the magnets, the force produced inside the channel is larger than a single magnet and the net magnetic force on each particle is as high as 5 pN.

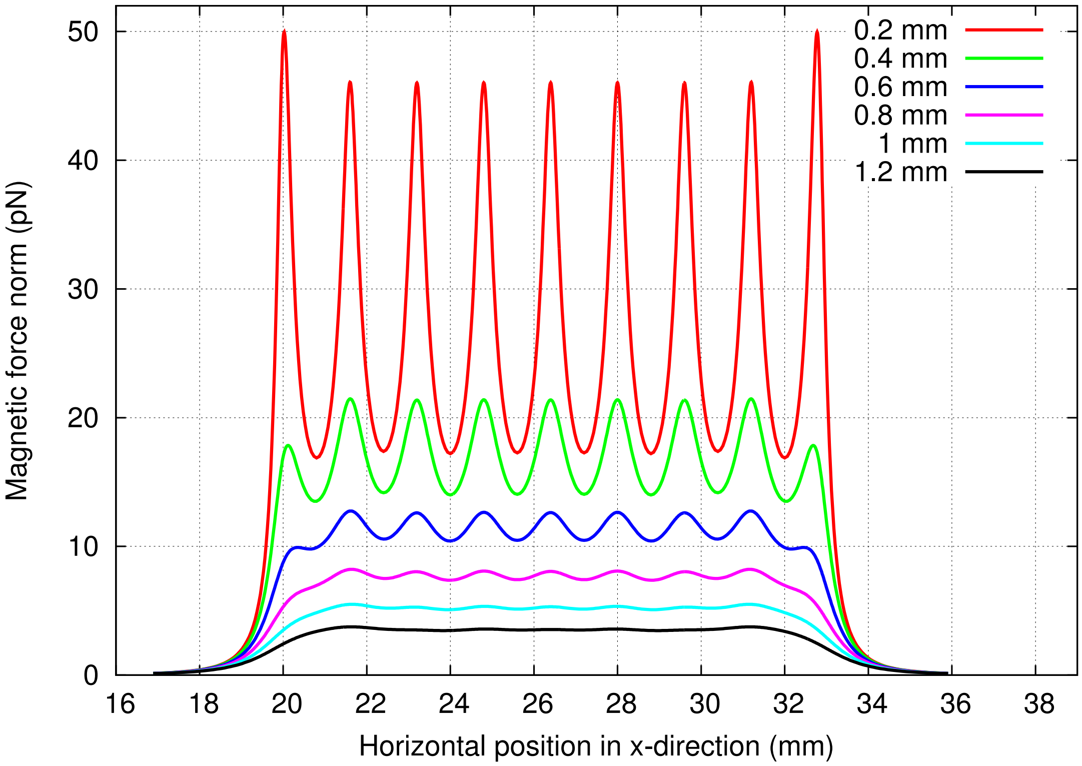


Figure 6.6: Variation of the magnetic force norm along the channel at various distances from the surface of the magnets.

6.2 Particle Transport Analysis

This section provides the results and analysis of the motion of particles under the influence of the magnetic field. First, a parametric study was performed to obtain the minimum channel length, also called the trapping length, to capture particles. The trapping length is a horizontal distance, measured from the first magnet, that a particle travels inside the channel before it gets trapped at the bottom surface of the channel.

To ensure all the particles can be trapped within the trapping length, the calculations were done for an extreme case where the particles travel the longest distance within the channel. Since the particles at the top surface of the channel travel the longest distance, this point was chosen to inject the particles into the channel. Also, an entrance region with the length of 7 mm was considered to ensure the fringing effect of the first magnet is not neglected.

In previous studies the particles were released from the top channel wall above the leading edge of the first magnet. As can be seen in Figure 6.7, the assumption of releasing particles from the first magnet is not accurate and because the magnetic field around the first magnet extends to a portion of the entrance region, the capture length is shorter. As shown in Figure 6.7, the first magnet is located at x = 7 mm, as indicated by the dashed line, and 20 μm particles are released with their center located at 10 μm away from the top channel wall.

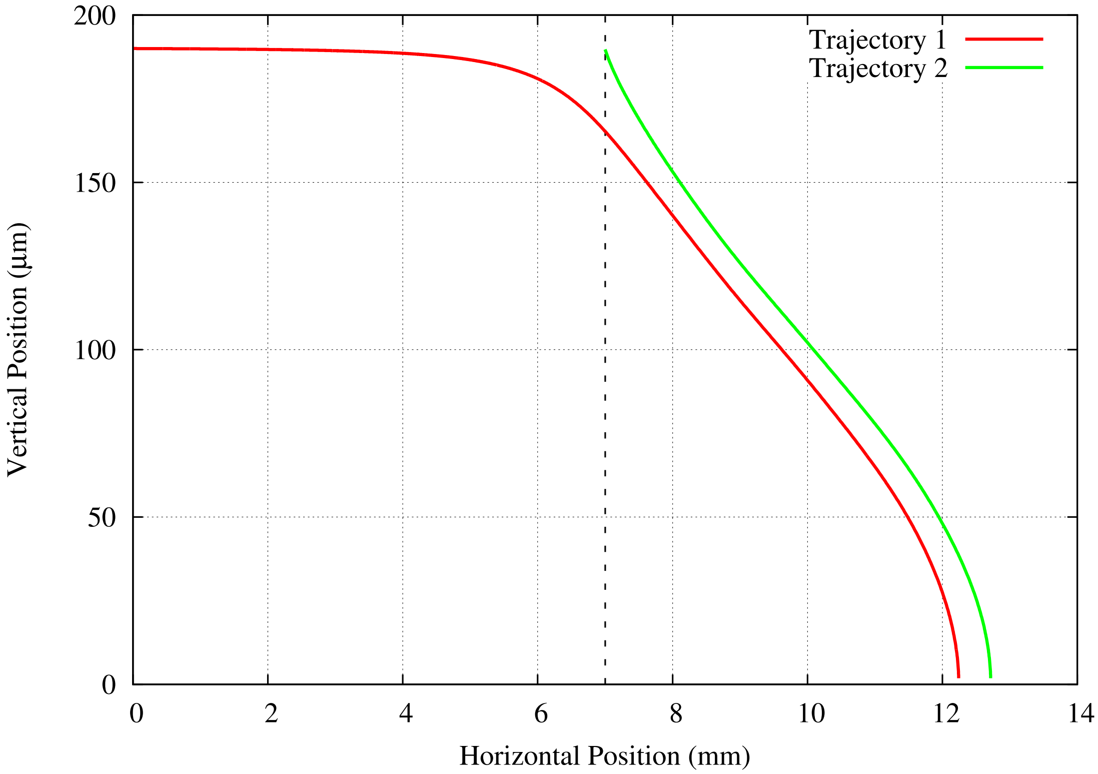


Figure 6.7: Particle trajectory for different releasing points inside the channel. The dashed line indicates the location of the first magnet.

6.3 Parametric Study

Four different parameters that have the most influence on the particle trajectory, are:

1. Number of magnetic beads per cell
2. Cell size
3. Flow rate
4. Bead size

In the following, the results of the parametric study are discussed. For all cases, the height of the channel and the thickness of the substrate were considered constant and equal to 200 μm and 1 mm, respectively. Other parameters such as number of beads, cell size, flow rate, and bead size were variable, unless otherwise stated. Also, the location of the first magnet was represented by a dashed line at x = 7 mm.

6.3.1 Number of Magnetic Beads Per Cell

Figure 6.8 shows the particle trajectory for different number of beads attached to each cell. The cell size and bead size were 20 μm and 1 μm, respectively. The flow rate was 100 ml/h which corresponds to the average velocity of 13.9 mm/s within the channel.

The results indicate that the trapping length decreases as the number of beads increases. This is because the magnetic force is proportional to the number of beads, and as the number of beads increases, the magnetic force increases as well. The results also show that for the cell size of 20 μm and the flow rate of 100 ml/h, at least 5 magnetic beads are required to capture the cells.

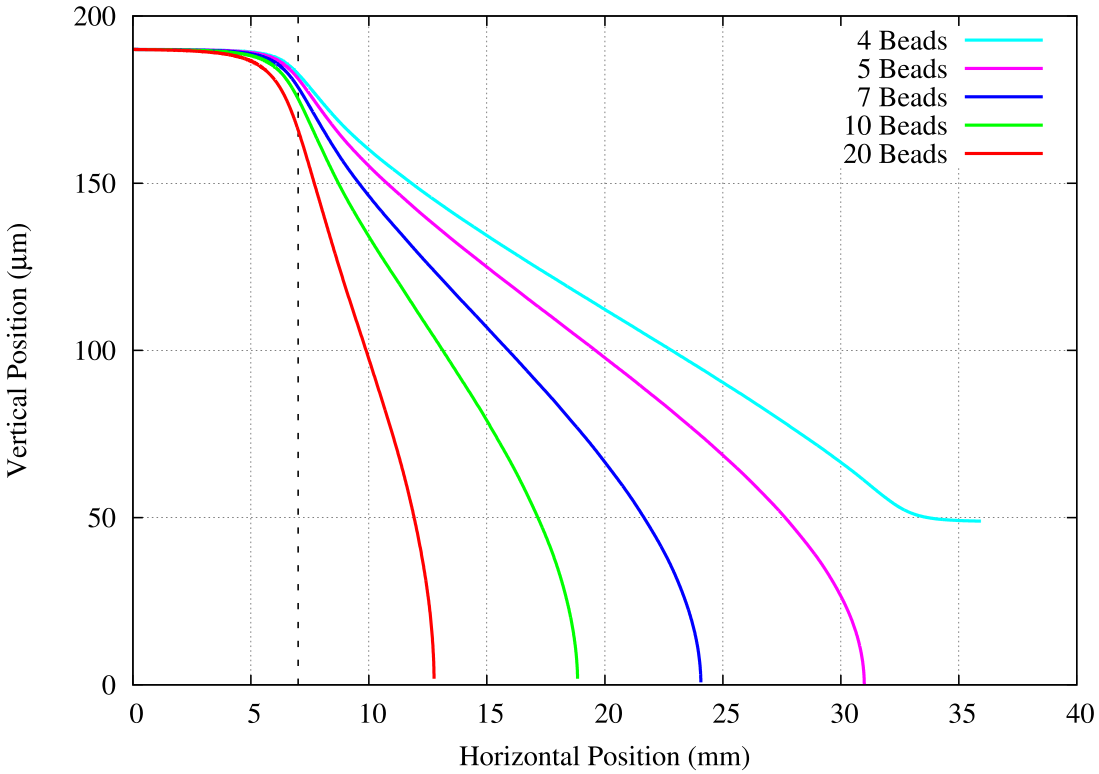


Figure 6.8: Particle trapping for different number of beads per cell. The cell size, bead size, and flow rate were 20 μm, 1 μm, and 100 ml/h.

6.3.2 Cell Size

The correct number of beads that are required to separate cells depends on the average cell size. Red blood cells can vary in size due to pathologies but they are mostly 7.5-8 μm in diameter. The average size of a white blood cell is around 10 μm in diameter and circulating tumor cells appear to have a diameter between 12 μm and 25 μm.

Figure 6.9 shows the particle trajectory for cell sizes of 10 μm, 15 μm, 20 μm, 25 μm, and 30 μm. Depending on the cell size, the average number of beads attached to each cell can vary, however, here the number of beads was assumed constant and equal to 10. The results show that for the flow rate of 100 ml/h, the capture length increases as the size of the cell increases. This is due to the fact that, the drag force is proportional to

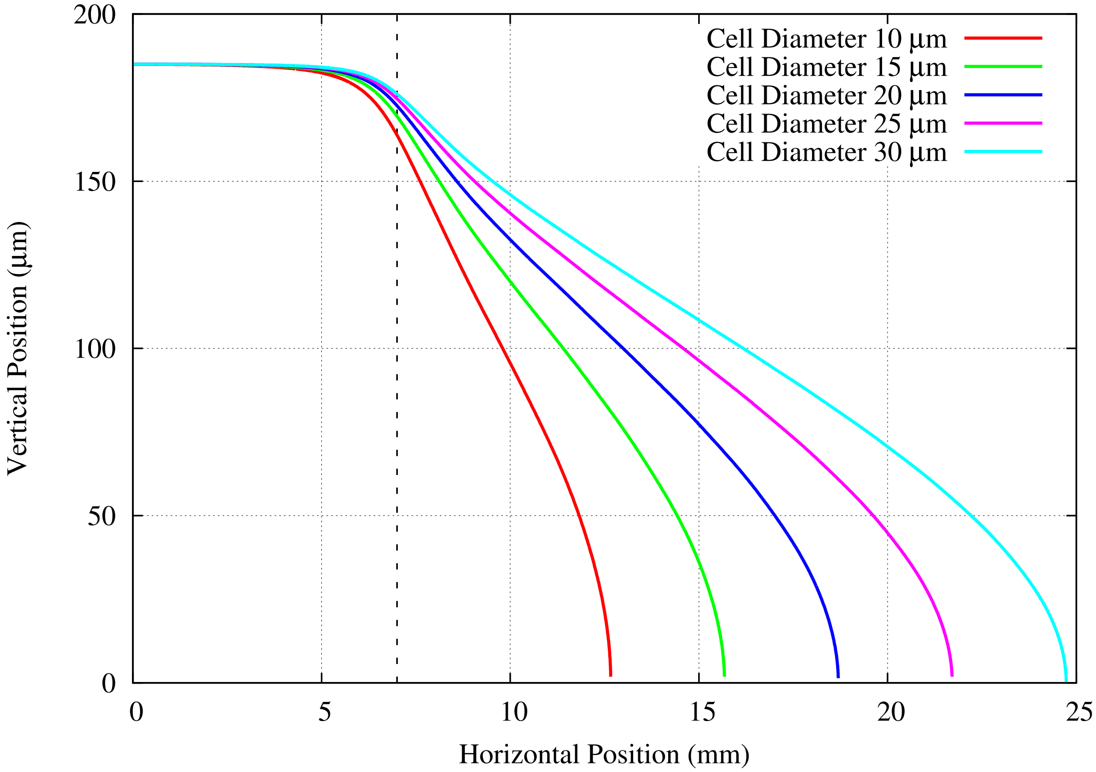


Figure 6.9: Particle trapping for cells with different sizes. The number of beads, bead size, and flow rate were 10 , 1 μm, and 100 ml/h.

the cell size, and as the cell size increases, the drag force increases. Thus, for a constant magnetic force, the capture length increases with increasing the cell size.

6.3.3 Flow Rate

Separation of small micro particles at different flow rates highly depends on the magnitude of the magnetic force inside the channel. For larger magnetic forces, higher input flow rates can be used to inject the particle-fluid solution inside the channel. As shown in Figure 6.10, the capture length increases as the flow rate increases. This is because the particle velocity along the channel increases with increasing the flow rate, resulting in a larger capture length. The results also show that as the flow rate exceeds 250 ml/h, depending on the position of the particle inside the channel, some of the particles

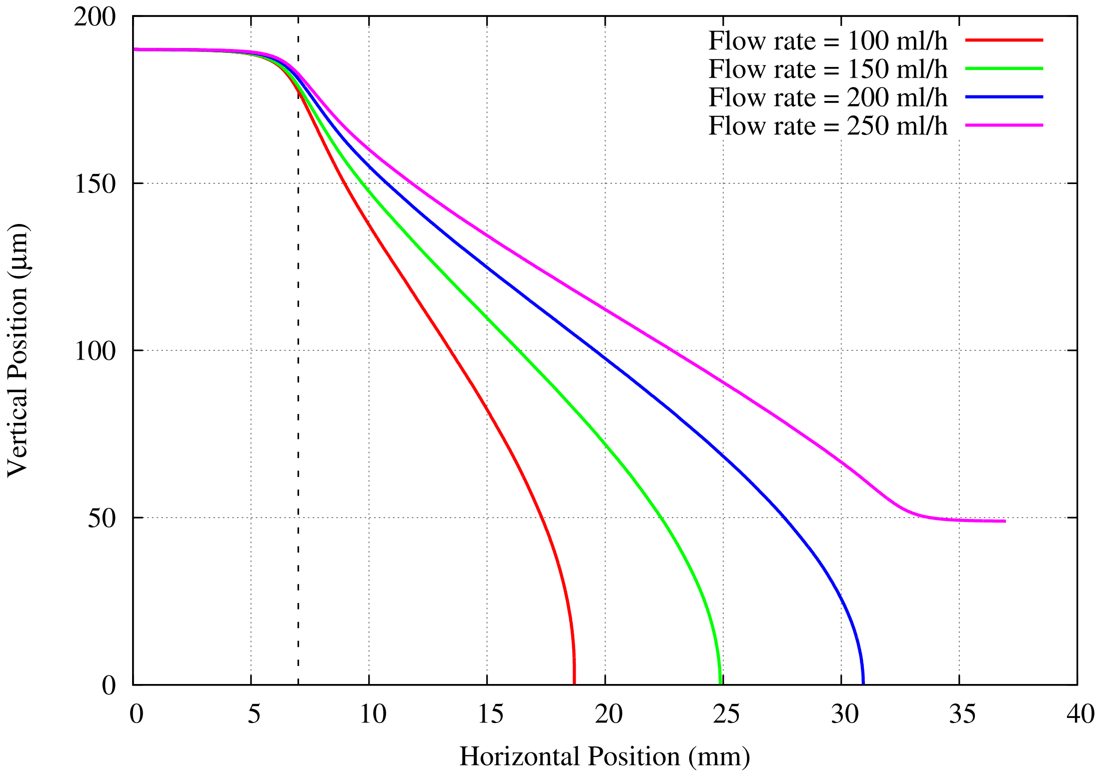


Figure 6.10: Particle trapping for different flow rates. The number of beads, cell size, and bead size were 10, 20 μm, 1 μm.

do not get captured and would leave the channel.

6.3.4 Bead Size

Magnetic beads come in several sizes for use in a wide variety of applications. Dynabeads are available in three different diameter sizes - 1 μm, 2.8 μm, and 4.5 μm [53]. Figure 6.11 shows the effect of different bead size on the particle trajectory. The number of beads, cell size, and flow rate were 1, 20 μm, and 100 ml/h, respectively. As can be seen, the capture length decreases as the bead size increases. This is due to the fact that the magnetic force is proportional to the cube of the diameter of the bead, Eq. (3.16), and as the bead size increases from 1 μm to 2.8 and 4.5 μm, the magnetic force increases 20 and 90 times, respectively, resulting in a shorter capture length. The results show that

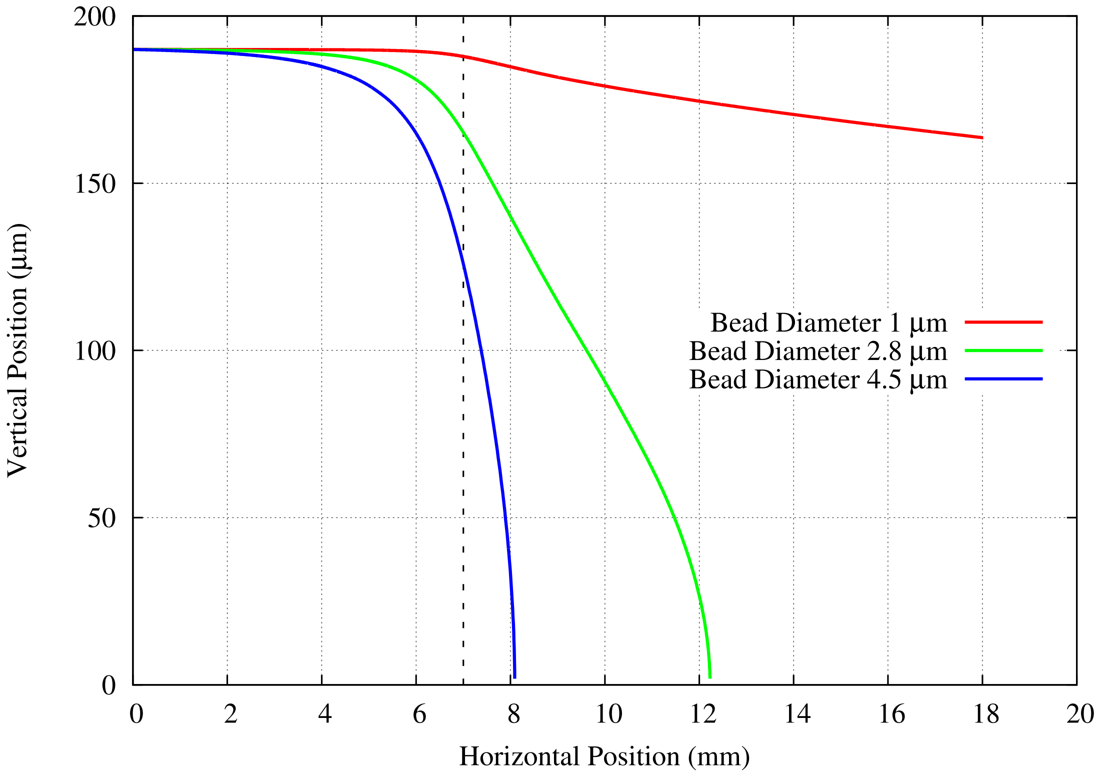


Figure 6.11: Particle trapping for cells with different bead size attached to them. The number of beads, cell size, and flow rate were 1, 20 μm, and 100 ml/h.

the particles tagged with 2.8 and 4.5 μm beads are captured within 5 mm and 1 mm from the first magnet, respectively, while a 1 μm bead size is not sufficient to trap the particles.

Figure 6.12 illustrates the effect of flow rate on particle trajectory for a 2.8 μm bead size. The number of beads and cell size were 1 and 20 μm, respectively. As seen there, a higher flow rate up to 300 ml/h can be achieved when 2.8 μm bead size is used.

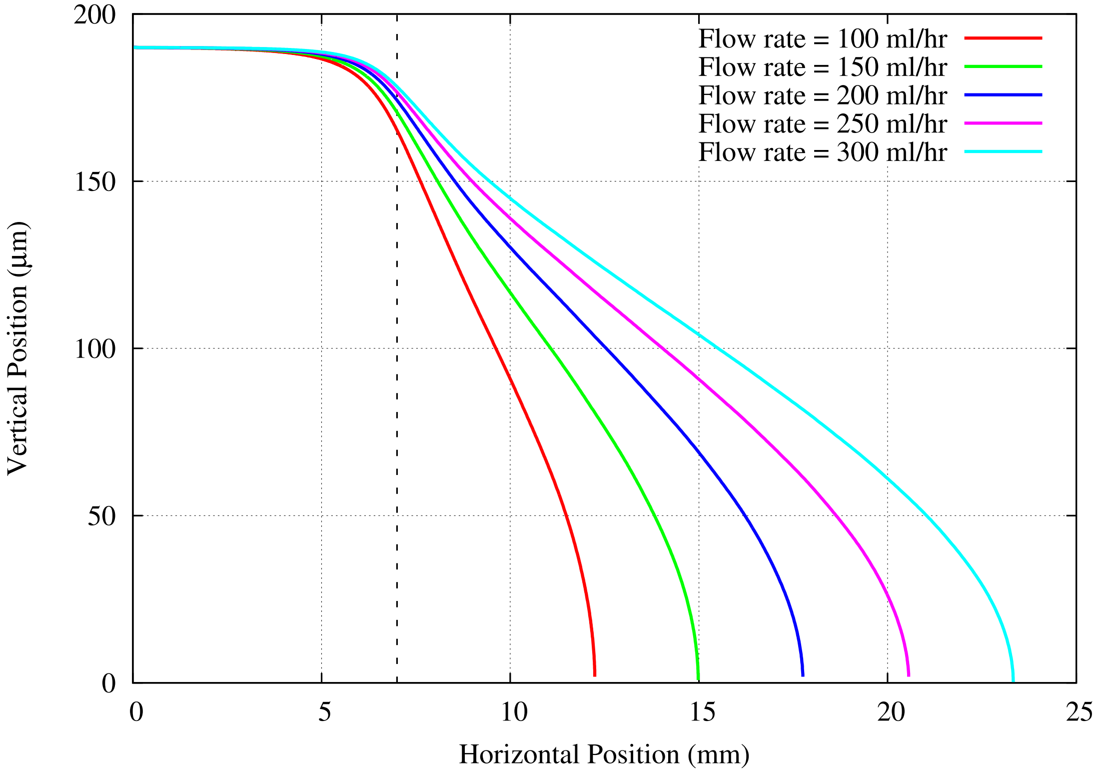


Figure 6.12: Particle trapping for different flow rates. The number of beads, cell size, and bead size were 1, 20 μm, 2.8 μm.

6.4 Separation Process Modeling

Figure 6.13 is a demonstration of a two-dimensional cell separation process for a case where 4000 particles with different number of beads attached to them were injected into the separation channel. Red, yellow, light blue, and dark blue cells are tagged with 20, 10, 5, and 2 beads, respectively. The bead size and flow rate were 1 μm and 100 ml/h. The top, middle, and bottom windows demonstrate the snap shot images of different regions along the channel. The first window shows the entrance region which is free of magnets and the particles flow through this region without any deflection. Separation of the particles starts as soon as the particles experience the magnetic force. As the particles flow through the separation channel, they are gradually pulled down towards the magnets and are immobilized on the magnetic surfaces. The second window represents the channel above the 3rd and 4th magnets (11-14 mm from the inlet port). Red particles with 20 beads attached to them are separated within this region. As the number of beads decreases, the particles travel a longer distance before being captured. Yellow particles with 10 beads attached to them, are isolated within the 7th and 8th magnets (17-20 mm from the inlet port). For this particular case, 15 magnets are required to capture light blue cells with five beads attached to them (third window, 29-32 mm from the inlet port). Finally, the dark blue particles with 2 beads attached to them do not get captured and flow out of the separation channel.

Figure 6.14 represents a two-dimensional cell separation process for a case where 3000 particles with different sizes were injected into the separation channel. The diameter of red, green, and blue cells are 15 μm, 20 μm, and 25 μm, respectively. Similar to the previous case, different windows demonstrate different regions within the channel. Initially, the particles travel through the entrance region without any deflection in their trajectory. Smaller particles experience smaller drag force compared to larger particles, as a result, smaller particles are expected to get captured faster. Second window shows the separation of 15 μm diameter red particles inside the channel above the 5th and 6th magnets (14-17 mm from the inlet port). 8 magnets are required to fully separate the green particles with particle size of 20 μm (17-20 mm from the inlet port). Finally, the largest particles with the diameter of 25 μm are separated within 10 magnets (20-23 mm from the inlet port).

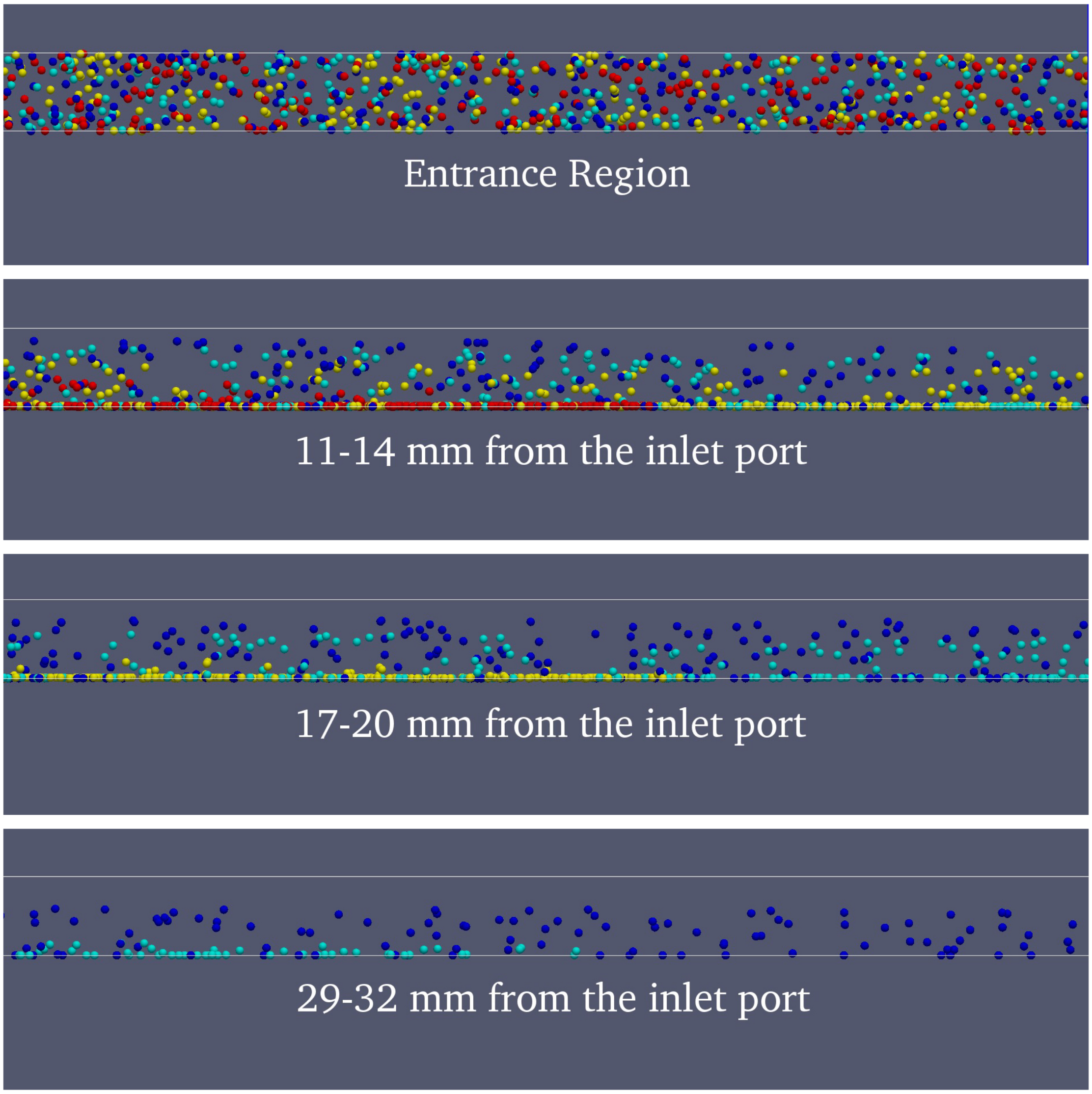


Figure 6.13: Particle trajectories for 20 μm cells with different number of beads attached to them. Red, yellow, light blue, and dark blue cells are tagged with 20, 10, 5, and 2 beads, respectively. The bead size and flow rate were 1 μm and 100 ml/h.

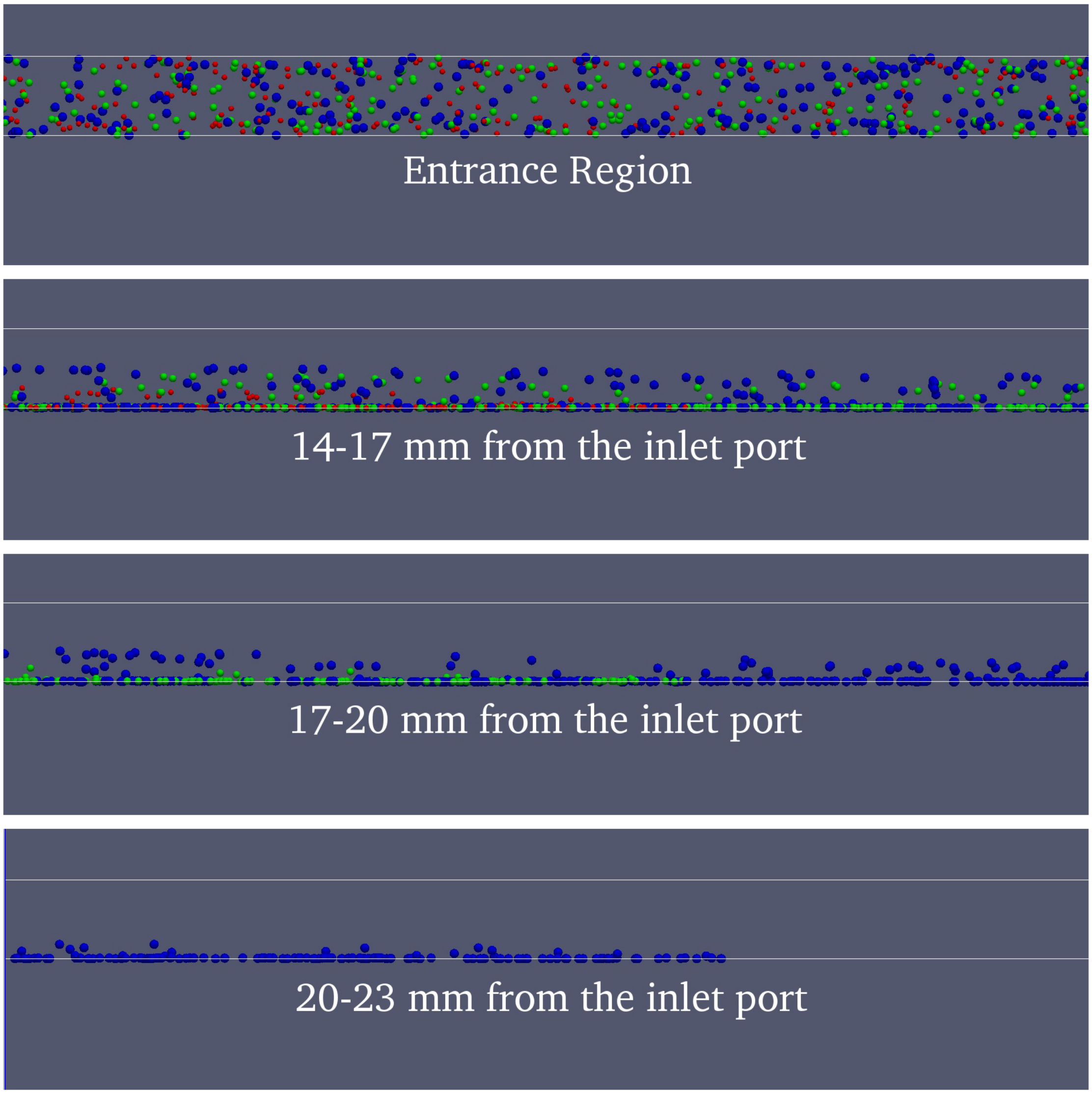


Figure 6.14: Particle trajectories for cells with different diameters. The diameter of red, green, and blue cells are 15 μm, 20 μm, and 25 μm, respectively. The number of beads, bead size, and flow rate were 10, 1 μm, and 100 ml/h.

6.5 Experimental Results

Figure 6.15 displays a comparison between the experimental and simulation results for 1 μm magnetic beads at an average fluid velocity of 13.9 mm/s. The channel height is 200 μm and the length of each frame is 400 μm which is approximately equal to 1/4 of a magnet. Efforts were made to ensure a close approximation between the model and the actual bead separation experiments. 10,000 magnetic beads were injected into the channel and the results were compared in Figure 6.15. In the absence of magnets, particles uniformly spread across the entrance region of the channel and move with fluid in the x-direction. This is evidenced by the experimental image in window (a), and the simulated particle trajectories in window a$'$. When the magnetic particles approach the magnets, they are deflected towards the bottom of the channel due to the downward magnetic force, resulting in a particle-free region near the upper wall of the channel. Window b depicts a region above the first magnet which indicates that the magnetic beads are confined within approximately the bottom 2/3 of the channel. This observation is consistent with the predicted particle trajectories as shown in frame b$'$. Frames c and d demonstrate the particle trajectories above the 2nd the 3rd magnets, respectively. The results indicate that the magnetic beads are further pulled down and confined within the bottom 1/2 and 1/3 of the channel. Also, as can be seen from these images the experimental and simulation results are in good agreement.

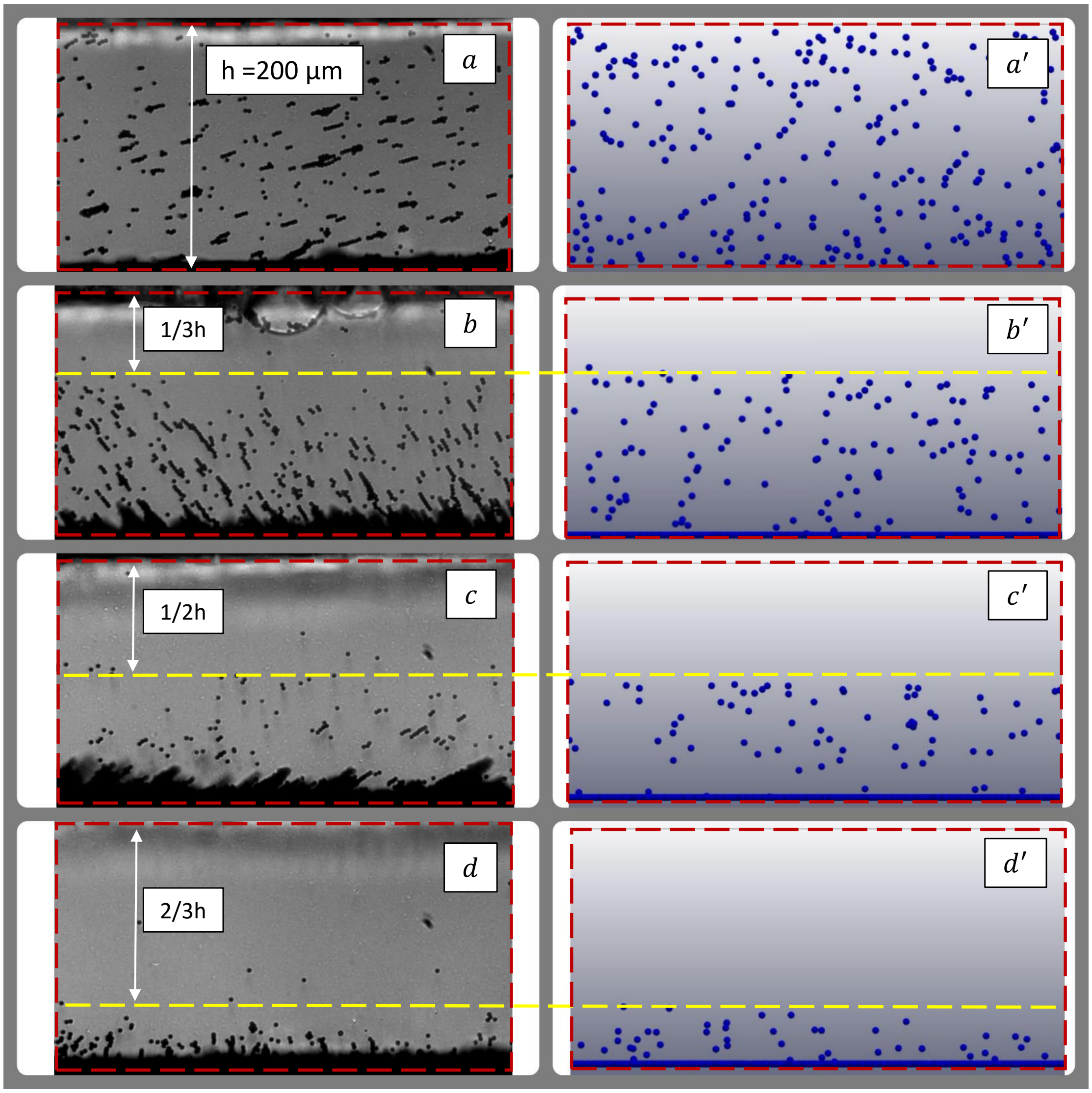


Figure 6.15: A comparison between the experimental and simulation results for the deflection of 1 μm magnetic beads at an average flow speed of 13.9 mm/s. Left images show micrographs of experimental observations while the right figures show the simulated trajectories. Panels a and a′: Entrance Region. Panels b and b′: First Magnet. Panels c and c′: Second Magnet. Panels d and d′: Third Magnet.

CHAPTER 7

CONCLUSION

Trajectories and capture of bioparticles were successfully simulated in a magnetophoretic bio-separation chip. The device consists of a continuous flow microfluidic channel and employs an array of magnets on the bottom side of the chip to produce high magnetic fields. It was shown that, due to the alternating polarity arrangement of the magnets, magnetic fields as large as 0.2 T were produced inside the channel. Simulations were performed using the OpenFOAM software to predict the trajectories and capture length of the particles within the separation channel. A comprehensive parametric study indicated that the trapping length decreases with increasing the number of beads per cell and the bead size. However, the trapping length increases as the size of the cell increases. The size of the beads greatly influenced the capture efficiency. As the bead size increases from 1 μm to 2.8 and 4.5 μm, the magnetic force increases by a factor of 20 and 90 times, respectively, allowing the chip to operate at higher sample flow rates, such as 300 ml/h, which is suitable for many real world applications. Finally, the numerical simulation results were verified by conducting an experiment, where the deflection of 1 μm magnetic beads at an average flow speed of 13.9 mm/s inside the separation channel with 200 μm in height was studied. A good qualitative agreement between the experimental and numerical results was obtained.

REFERENCES

[1] Khashan SA and Furlani EP., *Effects of particle–fluid coupling on particle transport and capture in a magnetophoretic microsystem*, Microfluidics and nanofluidics **12** (2012), no. 1-4, 565–580.

[2] Khashan SA and Furlani EP, *Coupled particle–fluid transport and magnetic separation in microfluidic systems with passive magnetic functionality*, Journal of Physics D: Applied Physics **46** (2013), no. 12, 125002.

[3] Choi JW, Ahn CH, Bhansali S, and Henderson HT, *A new magnetic bead-based, filterless bio-separator with planar electromagnet surfaces for integrated bio-detection systems*, Sensors and Actuators B: Chemical **68** (2000), no. 1, 34–39.

[4] Choi JW, Liakopoulos TM, and Ahn CH, *An on-chip magnetic bead separator using spiral electromagnets with semi-encapsulated permalloy*, Biosensors and Bioelectronics **16** (2001), no. 6, 409–416.

[5] Kong TF, Sugiarto HS, Liew HF, Wang X, Lew WS, Nguyen NT, and Chen Y, *An efficient microfluidic sorter: implementation of double meandering micro striplines for magnetic particles switching*, Microfluidics and nanofluidics **10** (2011), no. 5, 1069–1078.

[6] Arruebo M, Fernndez-Pacheco R, Ibarra MR, and Santamara J, *Magnetic nanoparticles for drug delivery*, Nano today **2** (2007), no. 3, 22–32.

[7] Hale C and Darabi J, *Magnetophoretic-based microfluidic device for DNA isolation*, Biomicrofluidics **8** (2014), no. 4, 044118.

[8] Pamme N, *On-chip bioanalysis with magnetic particles*, Current opinion in chemical biology **16** (2012), no. 3, 436–443.

[9] Guo C, *A Magnetophoretic Bioseparation Chip*, SOUTHERN ILLINOIS UNIVERSITY AT EDWARDSVILLE, 2014.

[10] Pratt ED, Huang C, Hawkins BG, Gleghorn JP, and Kirby BJ, *Rare cell capture in microfluidic devices*, Chemical engineering science **66** (2011), no. 7, 1508–1522.

[11] Autebert J, Coudert B, Bidard FC, Pierga JY, Descroix S, Malaquin L, and Viovy JL, *Microfluidic: an innovative tool for efficient cell sorting*, Methods **57** (2012), no. 3, 297–307.

[12] Furdui VI and Harrison DJ, *Immunomagnetic T cell capture from blood for PCR analysis using microfluidic systems*, Lab on a Chip **4** (2004), no. 6, 614–618.

[13] Inglis DW, Riehn R, Austin RH, and Sturm JC, *Continuous microfluidic immunomagnetic cell separation*, Applied Physics Letters **85** (2004), no. 21, 5093–5095.

[14] Hejazian M, Li W, and Nguyen NT, *Lab on a chip for continuous-flow magnetic cell separation*, Lab on a Chip **15** (2015), no. 4, 959–970.

[15] Xia N, Hunt TP, Mayers BT, Alsberg E, Whitesides GM, Westervelt RM, and Ingber DE, *Combined microfluidic-micromagnetic separation of living cells in continuous flow*, Biomedical Microdevices **8** (2006), no. 4, 299–308.

[16] Liu C, Lagae L, Wirix-Speetjens R, and Borghs G, *On-chip separation of magnetic particles with different magnetophoretic mobilities*, Journal of applied physics **101** (2007), no. 2, 024913.

[17] Jung J and Han KH, *Lateral-driven continuous magnetophoretic separation of blood cells*, Applied Physics Letters **93** (2008), no. 22, 223902.

[18] Adams JD, Kim U, and Soh HT, *Multitarget magnetic activated cell sorter*, Proceedings of the National Academy of Sciences **105** (2008), no. 47, 18165–18170.

[19] Derec C, Wilhelm C, Servais J, and Bacri JC, *Local control of magnetic objects in microfluidic channels*, Microfluidics and Nanofluidics **8** (2010), no. 1, 123–130.

[20] Lee H, Jung J, Han SI, and Han KH, *High-speed RNA microextraction technology using magnetic oligo-dT beads and lateral magnetophoresis*, Lab on a Chip **10** (2010), no. 20, 2764–2770.

[21] Shen F, Hwang H, Hahn YK, and Park JK, *Label-free cell separation using a tunable magnetophoretic repulsion force*, Analytical chemistry **84** (2012), no. 7, 3075–3081.

[22] Verbarg J, Kamgar-Parsi K, Shields AR, Howell PB, and Ligler FS, *Spinning magnetic trap for automated microfluidic assay systems*, Lab on a chip **12** (2012), no. 10, 1793–1799.

[23] Wilbanks JJ, Kiessling G, Zeng J, Zhang C, Tzeng TR, and Xuan X, *Exploiting magnetic asymmetry to concentrate diamagnetic particles in ferrofluid microflows*, Journal of Applied Physics **115** (2014), no. 4, 044907.

[24] Jung Y, Choi Y, Han KH, and Frazier AB, *Six-stage cascade paramagnetic mode magnetophoretic separation system for human blood samples*, Biomedical microdevices **12** (2010), no. 4, 637–645.

[25] Khashan SA, Alazzam A, and Furlani EP, *Computational analysis of enhanced magnetic bioseparation in microfluidic systems with flow-invasive magnetic elements*, Scientific reports **4** (2014).

[26] Kim J, Steinfeld U, Lee HH, and Seidel H, *Development of a novel micro immune-magnetophoresis cell sorter*, Sensors, 2007 IEEE, 2007, pp. 1081–1084.

[27] Pamme N and Wilhelm C, *Continuous sorting of magnetic cells via on-chip free-flow magnetophoresis*, Lab on a Chip **6** (2006), no. 8, 974–980.

[28] Forbes TP and Forry SP, *Microfluidic magnetophoretic separations of immunomagnetically labeled rare mammalian cells*, Lab on a chip **12** (2012), no. 8, 1471–1479.

[29] Zeng J, Deng Y, Vedantam P, Tzeng TR, and Xuan X, *Magnetic separation of particles and cells in ferrofluid flow through a straight microchannel using two offset magnets*, Journal of Magnetism and Magnetic Materials **346** (2013), 118–123.

[30] Liang L, Zhang C, and Xuan X, *Enhanced separation of magnetic and diamagnetic particles in a dilute ferrofluid*, Applied Physics Letters **102** (2013), no. 23, 234101.

[31] Darabi J and Guo C, *On-chip magnetophoretic isolation of CD4+ T cells from blood*, Biomicrofluidics **7** (2013), no. 5, 054106.

[32] Furlani EP, *Magnetophoretic separation of blood cells at the microscale*, Journal of Physics D: Applied Physics **40** (2007), no. 5, 1313.

[33] Cheng R, Zhu T, and Mao L, *Three-dimensional and analytical modeling of microfluidic particle transport in magnetic fluids*, Microfluidics and nanofluidics **16** (2014), no. 6, 1143–1154.

[34] Plouffe BD, Mahalanabis M, Lewis LH, Klapperich CM, and Murthy SK, *Clinically relevant microfluidic magnetophoretic isolation of rare-cell populations for diagnostic and therapeutic monitoring applications*, Analytical chemistry **84** (2012), no. 3, 1336–1344.

[35] Hoshino K, Huang YY, Lane N, Huebschman M, Uhr JW, Frenkel EP, and Zhang X, *Microchip-based immunomagnetic detection of circulating tumor cells*, Lab on a Chip **11** (2011), no. 20, 3449–3457.

[36] Han KH, Han A, and Frazier AB, *Microsystems for isolation and electrophysiological analysis of breast cancer cells from blood*, Biosensors and bioelectronics **21** (2006), no. 10, 1907–1914.

[37] Jiles D, *Introduction to magnetism and magnetic materials*, CRC press, 2015.

[38] Chow TL, *Introduction to electromagnetic theory: a modern perspective*, Jones & Bartlett Learning, 2006.

[39] Ma H, Pedel J, Fife P, and Johnson WP, *Hemispheres-in-cell geometry to predict colloid deposition in porous media*, Environmental Science & Technology **43** (2009), no. 22, 8573–8579.

[40] Purcell EM and Morin DJ, *Electricity and magnetism*, Cambridge University Press, 2013.

[41] Hfeli U, Schtt W, Teller J, Zborowski M, and editors, *Scientific and Clinical Applications of Magnetic Carriers*, Hafeli, U., Schutt, W., Teller, J., Zborowski, M., Eds (1997).

[42] Park JS, Song SH, and Jung HI, *Continuous focusing of microparticles using inertial lift force and vorticity via multi-orifice microfluidic channels*, Lab on a Chip **9** (2009), no. 7, 939–948.

[43] Gerber R, Takayasu M, and Friedlaender F, *Generalization of HGMS theory: the capture of ultra-fine particles*, IEEE Transactions on Magnetics **19** (1983), no. 5, 2115–2117.

[44] Furlani EP, *Analysis of particle transport in a magnetophoretic microsystem*, Journal of Applied Physics **99** (2006), no. 2, 024912.

[45] OpenFOAM, *The Open Source Computational Fluid Dynamics (CFD) Toolbox*, `http://www.openfoam.com/`. Accessed Jul 2016, 01.

[46] Jasak H, *OpenFOAM: open source CFD in research and industry*, International Journal of Naval Architecture and Ocean Engineering **1** (2009), no. 2, 89–94.

[47] Santos J and de Oliveira P, *A converging finite volume scheme for hyperbolic conservation laws with source terms*, Journal of computational and applied mathematics **111** (1999), no. 1, 239–251.

[48] Ferziger JH and Peric M, *Computational methods for fluid dynamics*, Springer Science & Business Media, 2012.

[49] Jasak H, *Error analysis and estimation for finite volume method with applications to fluid flow* (1996).

[50] Mavriplis DJ, *Mesh generation and adaptivity for complex geometries and flows*, Handbook of Computational Fluid Mechanics (1996), 417–459.

[51] K&J, *K&J Magnetics Company*, `https://www.kjmagnetics.com/`. Accessed Jul 2016, 01.

[52] Campbell P, *Permanent magnet materials and their application*, Cambridge University Press, 1996.

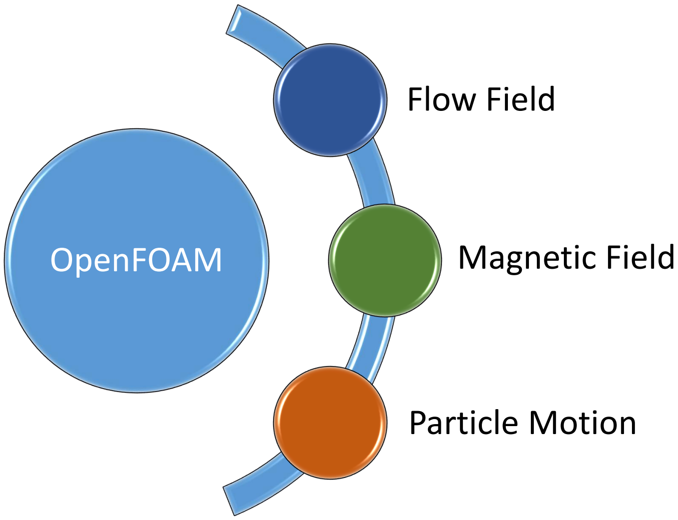
[53] ThermoFisher, *ThermoFisher Scientific*, `https://www.thermofisher.com/`. Accessed Jul 2016, 01.

APPENDIX A

Solver Modification and Compilation in OpenFOAM

Appendix A provides an instruction on how to edit and compile the OpenFOAM solvers being used in this study. Three different solvers, magneticFoam, simpleFoam, and icoUncoupledKinematicParcelFoam were used to simulate the separation process. There have been some changes that needed to be made in order to use the solvers properly. To change any solver in OpenFOAM, the necessary files must be copied into the user source directory before making any change in the codes, and the next step would be to apply the changes and recompile them.

Also, at the end of this chapter, the step by step procedure on running different cases is provided to help the reader run his/her own separation cases.



OpenFOAM

Simulating the separation process with the help of OpenFOAM requires using the following solvers:

- magneticFoam
- simpleFoam
- icoUncoupledKinematicParcelFoam

Lagrangian Directory

In order to be able to change the paramagnetic particle force, we first need to recompile the **lagrangian/intermediate** library.

In this study we are only interested in changing the paramagnetic particle force. Before making any change in the code, we recompile the files needed in the **$WM_PROJECT_USER_DIR/src/lagrangian** directory.

To do so, enter the following commands in the terminal window: (Next Slide)

- cd $WM_PROJECT_USER_DIR
- mkdir -p src/lagrangian
- cp -r $FOAM_SRC/lagrangian/intermediate $WM_PROJECT_USER_DIR/src/lagrangian

Once this is done, the necessary files will be copied into the user source directory and the next step would be to recompile them.

Now go to the following directory **lagrangian/intermediate/Make/files**

Replace the last line in the **files** directory as follows:

Replace this:

Code:

LIB = $(FOAM_LIBBIN)/liblagrangianIntermediate

with this:

Code:

```
LIB = $(FOAM_USER_LIBBIN)/libmylagrangianIntermediate
```

Now, recompile the lagrangian/intermediate library:

- ➢ wmake

New Paramagnetic Force

In our study the paramagnetic particle force has been calculated as:

$$\rho V \frac{|M|}{|B|}(\vec{B}.\nabla\vec{B})$$

By substituting the values of ρ, V, and M we end up getting the following equation for the paramagnetic particle force applied to one bead:

$$2.13\text{e-}14\,\frac{(\vec{B}.\nabla\vec{B})}{|B|}$$

Later on we will see that by making necessary changes, we would be able to calculate $\frac{(\vec{B}.\nabla\vec{B})}{|B|}$ using the following command:

```
mymagneticFoam -HdotGradH
```

Paramagnetic Particle Force

To change the paramagnetic particle force, go to the following directory:

```
cd
$WM_PROJECT_USER_DIR/src/lagrangian/intermediate/submodels/Kinematic/ParticleForces/Paramagnetic
```

In ParamagneticForce.C at line 110, the force has been defined as:

Code:

```
forceSuSp value(vector::zero, 0.0);

const interpolation<vector>& HdotGradHInterp = *HdotGradHInterpPtr_;
```

```
value.Su()=
mass*3.0*constant::electromagnetic::mu0.value()/p.rho()
*magneticSusceptibility_/(magneticSusceptibility_ + 3)
*HdotGradHInterp.interpolate(p.position(), p.currentTetIndices());
return value;
```

Which needs to be replaced by the following lines:

Code:

```
   if (mass > 5e-13)          /* Diameter = 10 micrometer */
      {
      forceSuSp value(vector::zero, 0.0);
      const interpolation<vector>& HdotGradHInterp = *HdotGradHInterpPtr_;
       value.Su()=
       2.13e-14*HdotGradHInterp.interpolate(p.position(), p.currentTetIndices());
      return value;
      }
   else                  /* Zero force to other particles */
      {
     forceSuSp value(vector::zero, 0.0);
  const interpolation<vector>& HdotGradHInterp = *HdotGradHInterpPtr_;
      value.Su() =
    0*HdotGradHInterp.interpolate(p.position(), p.currentTetIndices());
    value.Sp() = 0.0;
        return value;
    }
```

Next step is to copy the solvers we look forward to work with.

The solvers we are interested in working with, are:

- magneticFoam
- icoUncoupledKinematicParcelFoam

MagneticFoam Solver

This section shows how to recompile the magneticFoam solver before making any change in the codes.

```
cd $WM_PROJECT_USER_DIR
mkdir -p applications/solvers
cd $FOAM_SOLVERS
cd electromagnetics
cp -r magneticFoam $WM_PROJECT_USER_DIR/applications/solvers/mymagneticFoam
cd $WM_PROJECT_USER_DIR/applications/solvers/mymagneticFoam
```

Now a couple of alternations need to be made to *make* files in order for everything to compile and not overwrite the original solver. First, rename the primary file to your new solver name and delete the old dependency file:

```
mv magneticFoam.C mymagneticFoam.C
rm  magneticFoam.dep
```

Now go to the **Make** subdirectory and open the **'files'** file with your favorite editor.

Change it to read:

mymagneticFoam.C

```
EXE = $(FOAM_USER_APPBIN)/mymagneticFoam
```

No changes are necessary for the **'options'** file. Delete the old binaries subdirectory:

- `rm -rf linuxGccDP0pt`
- `cd ..`

Now, test that the renamed solver works:

- `wmake`

If everything worked correctly, your new solver binary should appear in the **FOAM_USER_APPBIN** directory. Check this with:

```
ls $FOAM_USER_APPBIN
```

Defining HdotGradH ($\vec{H}.\nabla\vec{H}$)

- In the new paramagnetic particle force that we derived, $\vec{H}.\nabla\vec{H}$ term appears in the codes, which needs to be defined in order to give us $\frac{(\vec{B}.\nabla\vec{B})}{|B|}$ value, where B is the magnetic flux density.
- Since we get $\vec{H}.\nabla\vec{H}$ by using mymagneticFoam solver, we have to make necessary changes in this solver in order to get $\frac{(\vec{B}.\nabla\vec{B})}{|B|}$ value when we enter the command *mymagentictFoam -HdotGradH*

Changing $\vec{H}.\nabla\vec{H}$ to get $\frac{(\vec{B}.\nabla\vec{B})}{|B|}$

- `cd $WM_PROJECT_USER_DIR/applications/solvers/mymagneticFoam`

Open the mymagneticFoam.C with your favorite editor.

First edit the 'Application' to reflect the new name.

Change magneticFoam to *mymagneticFoam*

The 'if' loop where we have:

```
if (args.optionFound("HdotGradH"))
{
    Info<< nl
        << "Creating field HdotGradH for time "
        << runTime.timeName() << endl;
    volVectorField HdotGradH
    (
        IOobject
        (
            "HdotGradH",
            runTime.timeName(),
            mesh
        ),
        H & fvc::grad(H)
    );
    HdotGradH.write();
}
```

needs to be deactivated by using /* … */ sign.

The magnetic field loop where the magnetic field B is calculated, needs to be replaced by the following loop:

```
if (!args.optionFound("noB") || args.optionFound("HdotGradH"))
    {
        Info<< nl
            << "Creating field B for time "
            << runTime.timeName() << endl;
        volVectorField B
        (
```

```
      IOobject
      (
        "B",
        runTime.timeName(),
        mesh
      ),
      constant::electromagnetic::mu0
     *fvc::reconstruct(murf*fvc::snGrad(psi)*mesh.magSf() + murf*Mrf)
    );
if (!args.optionFound("BoH"))
    {
      Info<< nl
        << "Creating field B for time "
        << runTime.timeName() << endl;
      B.write();
    }
    if (args.optionFound("HdotGradH"))
    {
      Info<< nl
        << "Creating field HdotGradH for time "
        << runTime.timeName() << endl;
      volVectorField HdotGradH
(
        IOobject
        (
          "HdotGradH",
          runTime.timeName(),
          mesh
        ),
```

```
            (B & fvc::grad(B))*1/mag(B)
        );
        HdotGradH.write();
    }
}
```

In these lines, as you can see, the mymagneticFoam.C file has been changed in order to give us (B & fvc::grad(B))*1/mag(B), as the value of HdotGradH

Save your changes and run wmake:

- wmake

Recompiling icoUncoupledKinematicParcelFoam Solver

- cd $FOAM_SOLVERS
- cd lagrangian
- cp -r icoUncoupledKinematicParcelFoam $WM_PROJECT_USER_DIR/applications/solvers/myicoUncoupledKinematicParcelFoam
- cd $WM_PROJECT_USER_DIR/applications/solvers/myicoUncoupledKinematicParcelFoam

As before, a couple of alternations need to be made to make files in order for everything to compile and not overwrite the original solver. First, rename the primary file to your new solver name and delete the old dependency file:

- mv icoUncoupledKinematicParcelFoam.C myicoUncoupledKinematicParcelFoam.C
- rm icoUncoupledKinematicParcelFoam.dep

Now go into the **Make** subdirectory and open the **'files'** file with your favorite editor. Change it to read:

```
myicoUncoupledKinematicParcelFoam.C

EXE = $(FOAM_USER_APPBIN)/myicoUncoupledKinematicParcelFoam
```

Since we have a new library “lagrangianintermediate”, we need to replace in “Make/options” the following codes

Replace this:

Code:

```
    -I$(LIB_SRC)/lagrangian/intermediate/lnInclude \
```

with this:

```
    -I$(WM_PROJECT_USER_DIR)/src/lagrangian/intermediate/lnInclude \
```

And this:

Code:

```
    -llagrangianIntermediate \
```

with this:

```
    -L$(FOAM_USER_LIBBIN) -lmylagrangianIntermediate  \
```

Delete the old binaries subdirectory:

- `rm -rf linuxGccDP0pt`
- `cd ..`

Now, test that the renamed solver (and your installation of OpenFOAM) works:

- `wmake`

If everything worked correctly, your new solver binary should appear in the FOAM_USER_APPBIN directory. Check this with:

- `ls $FOAM_USER_APPBIN`

Running

Up to now we have recompiled the following directories:

- Lagrangian library
- mymagneticFoam
- myIcoUncoupledKinematicParcelFoam

Calculating the magnetic field

go to MagneticField case (Fine Mesh), enter the following commands in the terminal window:

- `blockMesh`
- `topoSet`
- `mymagneticFoam`
- `mymagneticFoam -HdotGradH`

go to MapField case, enter the following commands:

- `blockMesh`
- `topoSet`

Mapping the magnetic field to our case

go to the folder where we have both MapField and MagneticField cases, then enter the following commands:

- `cp -r MapField/0 MapField/1`
- `mapFields MagneticField -case MapField -sourceTime latestTime`

last step:

- go to the MapField case, and copy the HdotGradH file from folder 1, and paste it to the 0 folder in Separation case.

Calculating the velocity field

Go to FlowField case, and enter the following commands in the terminal window:

- blockMesh
- simpleFoam

Go to Separation case, then:

- blockMesh

Change the zero directory to 3000 and in the controlDict directory change the start time to 3000.

- Map the velocity field to our case:

Go to the directory where we have both FlowField and Separation cases, and enter the following commands:

- mapFields FlowField -case Separation -consistent

now you should be able to see the mapped velocity and pressure fields in the Separation case, 3000 directory.

Rename the file 3000 to 0 and in controlDict also change the start time to 0.

Separation Case

Now the main case is ready and you can run the Separation case by the following command:

- myicoUncoupledKinematicParcelFoam

APPENDIX B

OpenFOAM Dictionaries for the Simulation of the Flow Field

The flow field case includes files such as Pressure (P), Velocity (U), TransportProperties, RASProperties, blockMeshDict, ControlDict, fvSchemes, and fvSolution which will be presented in this section.

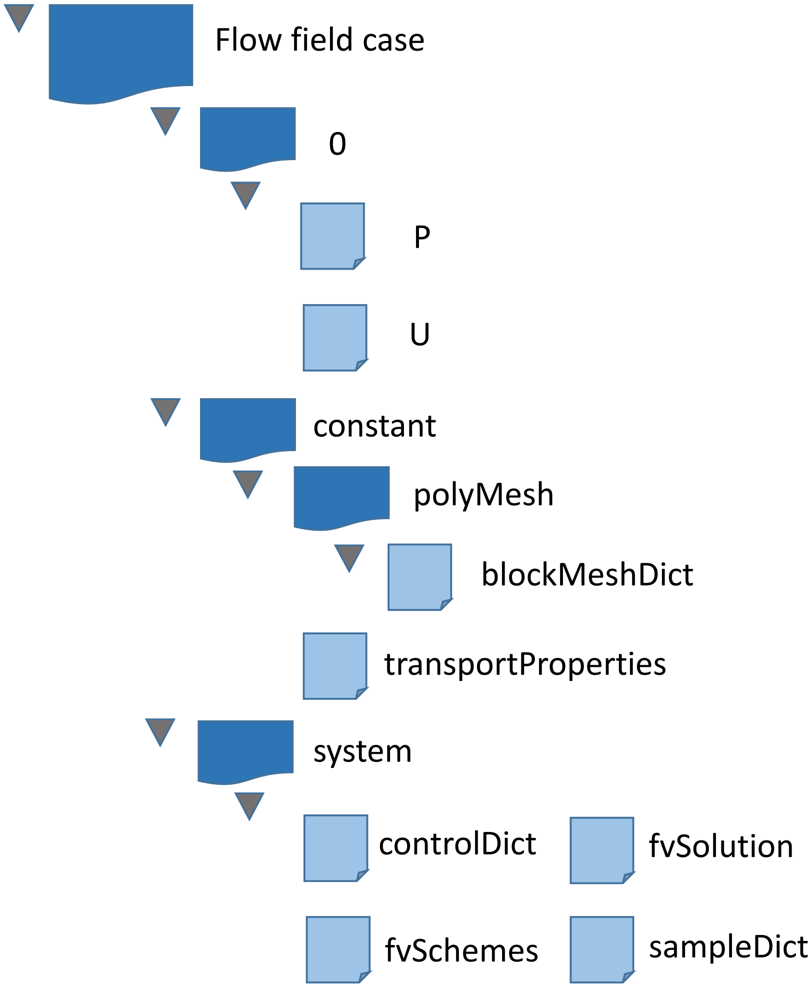


Figure B.1: Flow field case folder in OpenFOAM.

P (PRESSURE)

Description: The pressure boundary condition for water.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    northpatch
    {
        type            zeroGradient;
    }

    southpatch
    {
        type            zeroGradient;;
    }

    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        phi             phi;
        value           uniform 0;
    }

    frontAndBack
    {
        type            symmetry;
    }

    magnet1
    {
        type            zeroGradient;
    }
\\
.
.
Identical boundary condition for all 16 magnets
.
.
\\

    magnet32
    {
        type            zeroGradient;
    }
}
// ************************************************************************* //
```

U (Velocity)

Description: The velocity boundary condition for water.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      binary;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    northpatch
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    southpatch
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    inlet
    {
        type            fixedValue;
        value           uniform (0.0139 0 0); //Inlet Velocity
    }

    outlet
    {
        type            inletOutlet;
        phi             phi;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }

    frontAndBack
    {
        type            symmetry;
    }

    magnet1
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

\\
.
.
Identical boundary condition for all 16 magnets
.
.
\\

    magnet32
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}

// ************************************************************************* //
```

TransportProperties

Description: This file includes fuild (water) properties such as kinematic viscosity and density.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
transportModel  Newtonian;

// ro = 1 kg/m3
// mu = 2e-3 kg/m/s

nu              nu [ 0 2 -1 0 0 0 0 ] 1e-06;

CrossPowerLawCoeffs
{
    nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
    nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
    m               m [ 0 0 1 0 0 0 0 ] 1;
    n               n [ 0 0 0 0 0 0 0 ] 1;
}

BirdCarreauCoeffs
{
    nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
    nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
    k               k [ 0 0 1 0 0 0 0 ] 0;
    n               n [ 0 0 0 0 0 0 0 ] 1;
}
// ************************************************************************* //
```

RASProperties

Description: Flow regime (in this study the flow was considered laminar)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        laminar;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

blockMeshDict

Description: This file includes required information for making the geometry and the proper mesh. Geometry includes sixteen magnets and the mesh is clustered near the top and bottom walls, as well as at the interface of the magnets.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.001;

vertices
(
    (13 1 0)        // point 0
    (20 1 0)        // point 1
    (20.8 1 0)      // point 2
    (21.6 1 0)      // point 3
    (22.4 1 0)      // point 4
    (23.2 1 0)      // point 5
    (24 1 0)        // point 6
    (24.8 1 0)      // point 7
    (25.6 1 0)      // point 8
    (26.4 1 0)      // point 9
    (27.2 1 0)      // point 10
    (28 1 0)        // point 11
    (28.8 1 0)      // point 12
    (29.6 1 0)      // point 13
    (30.4 1 0)      // point 14
    (31.2 1 0)      // point 15
    (32 1 0)        // point 16
    (32.8 1 0)      // point 17
    (33.6 1 0)      // point 18
    (34.4 1 0)      // point 19
    (35.2 1 0)      // point 20
    (36 1 0)        // point 21
    (36.8 1 0)      // point 22
    (37.6 1 0)      // point 23
    (38.4 1 0)      // point 24
    (39.2 1 0)      // point 25
    (40 1 0)        // point 26
    (40.8 1 0)      // point 27
    (41.6 1 0)      // point 28
    (42.4 1 0)      // point 29
    (43.2 1 0)      // point 30
    (44 1 0)        // point 31
    (44.8 1 0)      // point 32
    (45.6 1 0)      // point 33
    (50 1 0)        // point 34
    (13 1.2 0)      // point 35
    (20 1.2 0)      // point 36
    (20.8 1.2 0)    // point 37
    (21.6 1.2 0)    // point 38
    (22.4 1.2 0)    // point 39
    (23.2 1.2 0)    // point 40
    (24 1.2 0)      // point 41
    (24.8 1.2 0)    // point 42
```

```
(25.6 1.2 0)    // point 43
(26.4 1.2 0)    // point 44
(27.2 1.2 0)    // point 45
(28 1.2 0)      // point 46
(28.8 1.2 0)    // point 47
(29.6 1.2 0)    // point 48
(30.4 1.2 0)    // point 49
(31.2 1.2 0)    // point 50
(32 1.2 0)      // point 51
(32.8 1.2 0)    // point 52
(33.6 1.2 0)    // point 53
(34.4 1.2 0)    // point 54
(35.2 1.2 0)    // point 55
(36 1.2 0)      // point 56
(36.8 1.2 0)    // point 57
(37.6 1.2 0)    // point 58
(38.4 1.2 0)    // point 59
(39.2 1.2 0)    // point 60
(40 1.2 0)      // point 61
(40.8 1.2 0)    // point 62
(41.6 1.2 0)    // point 63
(42.4 1.2 0)    // point 64
(43.2 1.2 0)    // point 65
(44 1.2 0)      // point 66
(44.8 1.2 0)    // point 67
(45.6 1.2 0)    // point 68
(50 1.2 0)      // point 69
(13 1 10)       // point 70
(20 1 10)       // point 71
(20.8 1 10)     // point 72
(21.6 1 10)     // point 73
(22.4 1 10)     // point 74
(23.2 1 10)     // point 75
(24 1 10)       // point 76
(24.8 1 10)     // point 77
(25.6 1 10)     // point 78
(26.4 1 10)     // point 79
(27.2 1 10)     // point 80
(28 1 10)       // point 81
(28.8 1 10)     // point 82
(29.6 1 10)     // point 83
(30.4 1 10)     // point 84
(31.2 1 10)     // point 85
(32 1 10)       // point 86
(32.8 1 10)     // point 87
(33.6 1 10)     // point 88
(34.4 1 10)     // point 89
(35.2 1 10)     // point 90
(36 1 10)       // point 91
(36.8 1 10)     // point 92
(37.6 1 10)     // point 93
(38.4 1 10)     // point 94
(39.2 1 10)     // point 95
(40 1 10)       // point 96
(40.8 1 10)     // point 97
(41.6 1 10)     // point 98
(42.4 1 10)     // point 99
(43.2 1 10)     // point 100
(44 1 10)       // point 101
(44.8 1 10)     // point 102
(45.6 1 10)     // point 103
(50 1 10)       // point 104
(13 1.2 10)     // point 105
(20 1.2 10)     // point 106
(20.8 1.2 10)   // point 107
(21.6 1.2 10)   // point 108
(22.4 1.2 10)   // point 109
(23.2 1.2 10)   // point 110
(24 1.2 10)     // point 111
(24.8 1.2 10)   // point 112
(25.6 1.2 10)   // point 113
```

```
    (26.4 1.2 10)    // point 114
    (27.2 1.2 10)    // point 115
    (28 1.2 10)      // point 116
    (28.8 1.2 10)    // point 117
    (29.6 1.2 10)    // point 118
    (30.4 1.2 10)    // point 119
    (31.2 1.2 10)    // point 120
    (32 1.2 10)      // point 121
    (32.8 1.2 10)    // point 122
    (33.6 1.2 10)    // point 123
    (34.4 1.2 10)    // point 124
    (35.2 1.2 10)    // point 125
    (36 1.2 10)      // point 126
    (36.8 1.2 10)    // point 127
    (37.6 1.2 10)    // point 128
    (38.4 1.2 10)    // point 129
    (39.2 1.2 10)    // point 130
    (40 1.2 10)      // point 131
    (40.8 1.2 10)    // point 132
    (41.6 1.2 10)    // point 133
    (42.4 1.2 10)    // point 134
    (43.2 1.2 10)    // point 135
    (44 1.2 10)      // point 136
    (44.8 1.2 10)    // point 137
    (45.6 1.2 10)    // point 138
    (50 1.2 10)      // point 139
);

blocks
(
    hex (0 1 36 35 70 71 106 105)    (25 50 1) simpleGrading (0.01 4 1)
    hex (1 2 37 36 71 72 107 106)    (20 50 1) simpleGrading (8 4 1)
    hex (2 3 38 37 72 73 108 107)    (20 50 1) simpleGrading (0.125 4 1)
    hex (3 4 39 38 73 74 109 108)    (20 50 1) simpleGrading (8 4 1)
    hex (4 5 40 39 74 75 110 109)    (20 50 1) simpleGrading (0.125 4 1)
    hex (5 6 41 40 75 76 111 110)    (20 50 1) simpleGrading (8 4 1)
    hex (6 7 42 41 76 77 112 111)    (20 50 1) simpleGrading (0.125 4 1)
    hex (7 8 43 42 77 78 113 112)    (20 50 1) simpleGrading (8 4 1)
    hex (8 9 44 43 78 79 114 113)    (20 50 1) simpleGrading (0.125 4 1)
    hex (9 10 45 44 79 80 115 114)   (20 50 1) simpleGrading (8 4 1)
    hex (10 11 46 45 80 81 116 115)  (20 50 1) simpleGrading (0.125 4 1)
    hex (11 12 47 46 81 82 117 116)  (20 50 1) simpleGrading (8 4 1)
    hex (12 13 48 47 82 83 118 117)  (20 50 1) simpleGrading (0.125 4 1)
    hex (13 14 49 48 83 84 119 118)  (20 50 1) simpleGrading (8 4 1)
    hex (14 15 50 49 84 85 120 119)  (20 50 1) simpleGrading (0.125 4 1)
    hex (15 16 51 50 85 86 121 120)  (20 50 1) simpleGrading (8 4 1)
    hex (16 17 52 51 86 87 122 121)  (20 50 1) simpleGrading (0.125 4 1)
    hex (17 18 53 52 87 88 123 122)  (20 50 1) simpleGrading (8 4 1)
    hex (18 19 54 53 88 89 124 123)  (20 50 1) simpleGrading (0.125 4 1)
    hex (19 20 55 54 89 90 125 124)  (20 50 1) simpleGrading (8 4 1)
    hex (20 21 56 55 90 91 126 125)  (20 50 1) simpleGrading (0.125 4 1)
    hex (21 22 57 56 91 92 127 126)  (20 50 1) simpleGrading (8 4 1)
    hex (22 23 58 57 92 93 128 127)  (20 50 1) simpleGrading (0.125 4 1)
    hex (23 24 59 58 93 94 129 128)  (20 50 1) simpleGrading (8 4 1)
    hex (24 25 60 59 94 95 130 129)  (20 50 1) simpleGrading (0.125 4 1)
    hex (25 26 61 60 95 96 131 130)  (20 50 1) simpleGrading (8 4 1)
    hex (26 27 62 61 96 97 132 131)  (20 50 1) simpleGrading (0.125 4 1)
    hex (27 28 63 62 97 98 133 132)  (20 50 1) simpleGrading (8 4 1)
    hex (28 29 64 63 98 99 134 133)  (20 50 1) simpleGrading (0.125 4 1)
    hex (29 30 65 64 99 100 135 134)  (20 50 1) simpleGrading (8 4 1)
    hex (30 31 66 65 100 101 136 135)  (20 50 1) simpleGrading (0.125 4 1)
    hex (31 32 67 66 101 102 137 136)  (20 50 1) simpleGrading (8 4 1)
    hex (32 33 68 67 102 103 138 137)  (20 50 1) simpleGrading (0.125 4 1)
    hex (33 34 69 68 103 104 139 138)  (20 50 1) simpleGrading (100 4 1)

)

edges
(
);
```

```
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 70 105 35)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (34 104 139 69)
        );
    }
    northpatch
    {
        type wall;
        faces
        (
            (35 105 106 36)
            (36 106 107 37)
            (37 107 108 38)
            (38 108 109 39)
            (39 109 110 40)
            (40 110 111 41)
            (41 111 112 42)
            (42 112 113 43)
            (43 113 114 44)
            (44 114 115 45)
            (45 115 116 46)
            (46 116 117 47)
            (47 117 118 48)
            (48 118 119 49)
            (49 119 120 50)
            (50 120 121 51)
            (51 121 122 52)
            (52 122 123 53)
            (53 123 124 54)
            (54 124 125 55)
            (55 125 126 56)
            (56 126 127 57)
            (57 127 128 58)
            (58 128 129 59)
            (59 129 130 60)
            (60 130 131 61)
            (61 131 132 62)
            (62 132 133 63)
            (63 133 134 64)
            (64 134 135 65)
            (65 135 136 66)
            (66 136 137 67)
            (67 137 138 68)
            (68 138 139 69)
        );
     }

     southpatch
     {
         type wall;
         faces
         (
            (1 71 70 0)
            (33 103 104 34)
         );
     }
```

```
magnet1
{
    type wall;
    faces
    (
        (2 72 71 1)
    );
}
magnet2
{
    type wall;
    faces
    (
        (2 72 73 3)
    );
}
magnet3
{
    type wall;
    faces
    (
        (3 73 74 4)
    );
}
magnet4
{
    type wall;
    faces
    (
        (4 74 75 5)
    );
}
magnet5
{
    type wall;
    faces
    (
        (5 75 76 6)
    );
}
magnet6
{
    type wall;
    faces
    (
        (6 76 77 7)
    );
}
magnet7
{
    type wall;
    faces
    (
        (7 77 78 8)
    );
}
magnet8
{
    type wall;
    faces
    (
        (8 78 79 9)
    );
}
magnet9
{
    type wall;
    faces
    (
        (9 79 80 10)
    );
```

```
}
magnet10
{
    type wall;
    faces
    (
        (10 80 81 11)
    );
}
magnet11
{
    type wall;
    faces
    (
        (11 81 82 12)
    );
}
magnet12
{
    type wall;
    faces
    (
        (12 82 83 13)
    );
}
magnet13
{
    type wall;
    faces
    (
        (13 83 84 14)
    );
}
magnet14
{
    type wall;
    faces
    (
        (14 84 85 15)
    );
}
magnet15
{
    type wall;
    faces
    (
        (15 85 86 16)
    );
}
magnet16
{
    type wall;
    faces
    (
        (16 86 87 17)
    );
}
magnet17
{
    type wall;
    faces
    (
        (17 87 88 18)
    );
}
magnet18
{
    type wall;
    faces
    (
        (18 88 89 19)
```

```
    );
}
magnet19
{
    type wall;
    faces
    (
        (19 89 90 20)
    );
}
magnet20
{
    type wall;
    faces
    (
        (20 90 91 21)
    );
}
magnet21
{
    type wall;
    faces
    (
        (21 91 92 22)
    );
}
magnet22
{
    type wall;
    faces
    (
        (22 92 93 23)
    );
}
magnet23
{
    type wall;
    faces
    (
        (23 93 94 24)
    );
}
magnet24
{
    type wall;
    faces
    (
        (24 94 95 25)
    );
}
magnet25
{
    type wall;
    faces
    (
        (25 95 96 26)
    );
}
magnet26
{
    type wall;
    faces
    (
        (26 96 97 27)
    );
}
magnet27
{
    type wall;
    faces
    (
```

```
        (27 97 98 28)
    );
}
magnet28
{
    type wall;
    faces
    (
        (28 98 99 29)
    );
}
magnet29
{
    type wall;
    faces
    (
        (29 99 100 30)
    );
}
magnet30
{
    type wall;
    faces
    (
        (30 100 101 31)
    );
}
magnet31
{
    type wall;
    faces
    (
        (31 101 102 32)
    );
}
magnet32
{
    type wall;
    faces
    (
        (32 102 103 33)
    );
}
frontAndBack
{
    type symmetry;
    faces
    (
        (0 35 36 1)
        (1 36 37 2)
        (2 37 38 3)
        (3 38 39 4)
        (4 39 40 5)
        (5 40 41 6)
        (6 41 42 7)
        (7 42 43 8)
        (8 43 44 9)
        (9 44 45 10)
        (10 45 46 11)
        (11 46 47 12)
        (12 47 48 13)
        (13 48 49 14)
        (14 49 50 15)
        (15 50 51 16)
        (16 51 52 17)
        (17 52 53 18)
        (18 53 54 19)
        (19 54 55 20)
        (20 55 56 21)
        (21 56 57 22)
        (22 57 58 23)
```

```
            (23 58 59 24)
            (24 59 60 25)
            (25 60 61 26)
            (26 61 62 27)
            (27 62 63 28)
            (28 63 64 29)
            (29 64 65 30)
            (30 65 66 31)
            (31 66 67 32)
            (32 67 68 33)
            (33 68 69 34)
            (70 71 106 105)
            (71 72 107 106)
            (72 73 108 107)
            (73 74 109 108)
            (74 75 110 109)
            (75 76 111 110)
            (76 77 112 111)
            (77 78 113 112)
            (78 79 114 113)
            (79 80 115 114)
            (80 81 116 115)
            (81 82 117 116)
            (82 83 118 117)
            (83 84 119 118)
            (84 85 120 119)
            (85 86 121 120)
            (86 87 122 121)
            (87 88 123 122)
            (88 89 124 123)
            (89 90 125 124)
            (90 91 126 125)
            (91 92 127 126)
            (92 93 128 127)
            (93 94 129 128)
            (94 95 130 129)
            (95 96 131 130)
            (96 97 132 131)
            (97 98 133 132)
            (98 99 134 133)
            (99 100 135 134)
            (100 101 136 135)
            (101 102 137 136)
            (102 103 138 137)
            (103 104 139 138)
        );
    }
);


mergePatchPairs
(
);

// ************************************************************************* //
```

ControlDict

Description: This file includes the case setup such as the end time, time step, precision of the time writing, write interval of the time, and other important parameters.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         3000;

deltaT          1;

writeControl    timeStep;

writeInterval   1000;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression compressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

// ************************************************************************* //
```

fvSchemes

Description: This file includes the used schemes, methods, desired residuals, tolerances, and iterations for the parameters that were defined in the zero folder, such as velocity and pressure.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss upwind;
    div(phi,k)      Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss upwind;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.01;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }

    k
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }

    epsilon
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }

    R
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }

    nuTilda
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    p               0.3;
    U               0.7;
    k               0.7;
    epsilon         0.7;
    R               0.7;
    nuTilda         0.7;
}


// ************************************************************************* //
```

APPENDIX C

OpenFOAM Dictionaries for the Simulation of the Magnetic Field

The magnetic field case includes files such as magnetic scalar potential (Psi), transportProperties, blockMeshDict, ControlDict, topoSetDict, fvSchemes, and fvSolution which will be presented in this section.

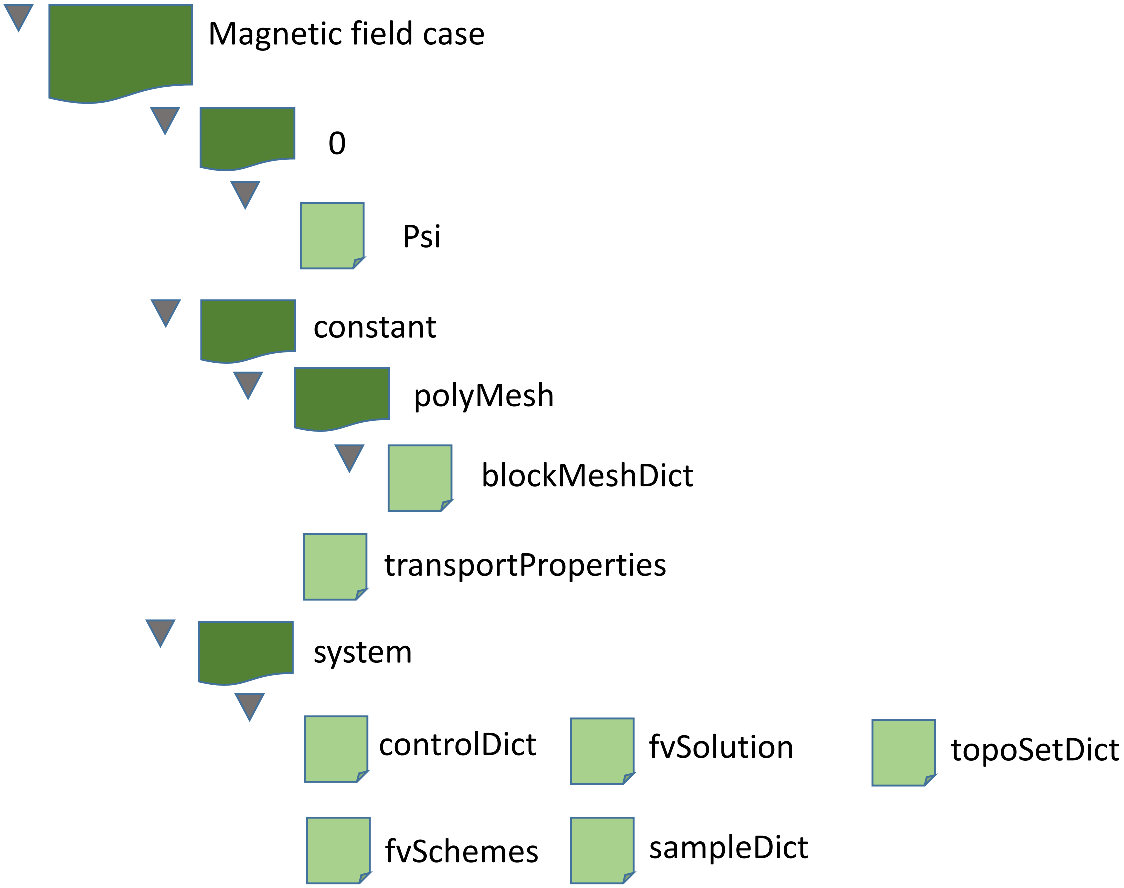


Figure C.1: Magnetic field case folder in OpenFOAM.

PSI (Magnetic scalar potential)

Description: This file provides the magnetic scalar potential boundary condition for solving the magnetic field around the permanent magnets. (The fixedGradient value for psi was calculated based on the surface magnetic flux density norm of 0.7 T, obtained from the K&J Magnetics company website.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      psi;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 1 0];

internalField   uniform 0;

boundaryField
{
    magnet1
    {
        type            fixedGradient;
        gradient        uniform -554252;  // -Magnetization
    }
    magnet2
    {
        type            fixedGradient;
        gradient        uniform -554252;  // -Magnetization
    }
    magnet3
    {
        type            fixedGradient;    // Magnetization
        gradient        uniform 554252;
    }
    magnet4
    {
        type            fixedGradient;    // Magnetization
        gradient        uniform 554252;
    }
\\
.
.
Identical boundary condition for all 32 magnets
.
.
\\
    magnet31
    {
        type            fixedGradient;    // Magnetization
        gradient        uniform 554252;
    }
    magnet32
    {
        type            fixedGradient;    // Magnetization
        gradient        uniform 554252;
    }
    inlet
    {
```

```
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    northpatch
    {
        type            zeroGradient;
    }
    southpatch
    {
        type            fixedValue;
        value           uniform 0;
    }
    frontAndBack
    {
        type            symmetry;
    }
}

// ************************************************************************* //
```

transportProperties

Description: This file includes permanent magnet properties such as relative permeability, remanence field, and the orientation of the magnets.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

magnets
(
  (magnet1 1.05 -1083769 (0 1 0))
  (magnet2 1.05 -1083769 (0 1 0))
  (magnet3 1.05 1083769 (0 1 0))
  (magnet4 1.05 1083769 (0 1 0))
\\
.
.
Identical lines for all 32 magnets
.
.
\\
  (magnet31 1.05 1083769 (0 1 0))
  (magnet32 1.05 1083769 (0 1 0))
);

// ************************************************************************* //
```

blockMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.001;

vertices
(
    (0 0 0)        // point 0
    (20 0 0)       // point 1
    (20.8 0 0)     // point 2
    (21.6 0 0)     // point 3
    (22.4 0 0)     // point 4
    (23.2 0 0)     // point 5
    (24 0 0)       // point 6
    (24.8 0 0)     // point 7
    (25.6 0 0)     // point 8
    (26.4 0 0)     // point 9
    (27.2 0 0)     // point 10
    (28 0 0)       // point 11
    (28.8 0 0)     // point 12
    (29.6 0 0)     // point 13
    (30.4 0 0)     // point 14
    (31.2 0 0)     // point 15
    (32 0 0)       // point 16
    (32.8 0 0)     // point 17
    (33.6 0 0)     // point 18
    (34.4 0 0)     // point 19
    (35.2 0 0)     // point 20
    (36 0 0)       // point 21
    (36.8 0 0)     // point 22
    (37.6 0 0)     // point 23
    (38.4 0 0)     // point 24
    (39.2 0 0)     // point 25
    (40 0 0)       // point 26
    (40.8 0 0)     // point 27
    (41.6 0 0)     // point 28
    (42.4 0 0)     // point 29
    (43.2 0 0)     // point 30
    (44 0 0)       // point 31
    (44.8 0 0)     // point 32
    (45.6 0 0)     // point 33
    (65.6 0 0)     // point 34
    (0 10 0)       // point 35
    (20 10 0)      // point 36
    (20.8 10 0)    // point 37
    (21.6 10 0)    // point 38
    (22.4 10 0)    // point 39
    (23.2 10 0)    // point 40
    (24 10 0)      // point 41
    (24.8 10 0)    // point 42
    (25.6 10 0)    // point 43
    (26.4 10 0)    // point 44
    (27.2 10 0)    // point 45
    (28 10 0)      // point 46
```

```
(28.8 10 0)    // point 47
(29.6 10 0)    // point 48
(30.4 10 0)    // point 49
(31.2 10 0)    // point 50
(32 10 0)      // point 51
(32.8 10 0)    // point 52
(33.6 10 0)    // point 53
(34.4 10 0)    // point 54
(35.2 10 0)    // point 55
(36 10 0)      // point 56
(36.8 10 0)    // point 57
(37.6 10 0)    // point 58
(38.4 10 0)    // point 59
(39.2 10 0)    // point 60
(40 10 0)      // point 61
(40.8 10 0)    // point 62
(41.6 10 0)    // point 63
(42.4 10 0)    // point 64
(43.2 10 0)    // point 65
(44 10 0)      // point 66
(44.8 10 0)    // point 67
(45.6 10 0)    // point 68
(65.6 10 0)    // point 69
(0 0 10)       // point 70
(20 0 10)      // point 71
(20.8 0 10)    // point 72
(21.6 0 10)    // point 73
(22.4 0 10)    // point 74
(23.2 0 10)    // point 75
(24 0 10)      // point 76
(24.8 0 10)    // point 77
(25.6 0 10)    // point 78
(26.4 0 10)    // point 79
(27.2 0 10)    // point 80
(28 0 10)      // point 81
(28.8 0 10)    // point 82
(29.6 0 10)    // point 83
(30.4 0 10)    // point 84
(31.2 0 10)    // point 85
(32 0 10)      // point 86
(32.8 0 10)    // point 87
(33.6 0 10)    // point 88
(34.4 0 10)    // point 89
(35.2 0 10)    // point 90
(36 0 10)      // point 91
(36.8 0 10)    // point 92
(37.6 0 10)    // point 93
(38.4 0 10)    // point 94
(39.2 0 10)    // point 95
(40 0 10)      // point 96
(40.8 0 10)    // point 97
(41.6 0 10)    // point 98
(42.4 0 10)    // point 99
(43.2 0 10)    // point 100
(44 0 10)      // point 101
(44.8 0 10)    // point 102
(45.6 0 10)    // point 103
(65.6 0 10)    // point 104
(0 10 10)      // point 105
(20 10 10)     // point 106
(20.8 10 10)   // point 107
(21.6 10 10)   // point 108
(22.4 10 10)   // point 109
(23.2 10 10)   // point 110
(24 10 10)     // point 111
(24.8 10 10)   // point 112
(25.6 10 10)   // point 113
(26.4 10 10)   // point 114
(27.2 10 10)   // point 115
(28 10 10)     // point 116
(28.8 10 10)   // point 117
```

```
    (29.6 10 10)   // point 118
    (30.4 10 10)   // point 119
    (31.2 10 10)   // point 120
    (32 10 10)     // point 121
    (32.8 10 10)   // point 122
    (33.6 10 10)   // point 123
    (34.4 10 10)   // point 124
    (35.2 10 10)   // point 125
    (36 10 10)     // point 126
    (36.8 10 10)   // point 127
    (37.6 10 10)   // point 128
    (38.4 10 10)   // point 129
    (39.2 10 10)   // point 130
    (40 10 10)     // point 131
    (40.8 10 10)   // point 132
    (41.6 10 10)   // point 133
    (42.4 10 10)   // point 134
    (43.2 10 10)   // point 135
    (44 10 10)     // point 136
    (44.8 10 10)   // point 137
    (45.6 10 10)   // point 138
    (65.6 10 10)   // point 139
);

blocks
(
    hex (0 1 36 35 70 71 106 105)    (90 5000 1) simpleGrading (0.01 4 1)
    hex (1 2 37 36 71 72 107 106)    (30 5000 1) simpleGrading (8 4 1)
    hex (2 3 38 37 72 73 108 107)    (30 5000 1) simpleGrading (0.125 4 1)
    hex (3 4 39 38 73 74 109 108)    (30 5000 1) simpleGrading (8 4 1)
    hex (4 5 40 39 74 75 110 109)    (30 5000 1) simpleGrading (0.125 4 1)
    hex (5 6 41 40 75 76 111 110)    (30 5000 1) simpleGrading (8 4 1)
    hex (6 7 42 41 76 77 112 111)    (30 5000 1) simpleGrading (0.125 4 1)
    hex (7 8 43 42 77 78 113 112)    (30 5000 1) simpleGrading (8 4 1)
    hex (8 9 44 43 78 79 114 113)    (30 5000 1) simpleGrading (0.125 4 1)
    hex (9 10 45 44 79 80 115 114)   (30 5000 1) simpleGrading (8 4 1)
    hex (10 11 46 45 80 81 116 115)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (11 12 47 46 81 82 117 116)  (30 5000 1) simpleGrading (8 4 1)
    hex (12 13 48 47 82 83 118 117)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (13 14 49 48 83 84 119 118)  (30 5000 1) simpleGrading (8 4 1)
    hex (14 15 50 49 84 85 120 119)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (15 16 51 50 85 86 121 120)  (30 5000 1) simpleGrading (8 4 1)
    hex (16 17 52 51 86 87 122 121)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (17 18 53 52 87 88 123 122)  (30 5000 1) simpleGrading (8 4 1)
    hex (18 19 54 53 88 89 124 123)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (19 20 55 54 89 90 125 124)  (30 5000 1) simpleGrading (8 4 1)
    hex (20 21 56 55 90 91 126 125)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (21 22 57 56 91 92 127 126)  (30 5000 1) simpleGrading (8 4 1)
    hex (22 23 58 57 92 93 128 127)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (23 24 59 58 93 94 129 128)  (30 5000 1) simpleGrading (8 4 1)
    hex (24 25 60 59 94 95 130 129)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (25 26 61 60 95 96 131 130)  (30 5000 1) simpleGrading (8 4 1)
    hex (26 27 62 61 96 97 132 131)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (27 28 63 62 97 98 133 132)  (30 5000 1) simpleGrading (8 4 1)
    hex (28 29 64 63 98 99 134 133)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (29 30 65 64 99 100 135 134)  (30 5000 1) simpleGrading (8 4 1)
    hex (30 31 66 65 100 101 136 135)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (31 32 67 66 101 102 137 136)  (30 5000 1) simpleGrading (8 4 1)
    hex (32 33 68 67 102 103 138 137)  (30 5000 1) simpleGrading (0.125 4 1)
    hex (33 34 69 68 103 104 139 138)  (90 5000 1) simpleGrading (100 4 1)

)

edges
(
);


boundary
(
    inlet
```

```
{
    type patch;
    faces
    (
        (0 70 105 35)
    );
}
outlet
{
    type patch;
    faces
    (
        (34 104 139 69)
    );
}
northpatch
{
    type wall;
    faces
    (
        (35 105 106 36)
        (36 106 107 37)
        (37 107 108 38)
        (38 108 109 39)
        (39 109 110 40)
        (40 110 111 41)
        (41 111 112 42)
        (42 112 113 43)
        (43 113 114 44)
        (44 114 115 45)
        (45 115 116 46)
        (46 116 117 47)
        (47 117 118 48)
        (48 118 119 49)
        (49 119 120 50)
        (50 120 121 51)
        (51 121 122 52)
        (52 122 123 53)
        (53 123 124 54)
        (54 124 125 55)
        (55 125 126 56)
        (56 126 127 57)
        (57 127 128 58)
        (58 128 129 59)
        (59 129 130 60)
        (60 130 131 61)
        (61 131 132 62)
        (62 132 133 63)
        (63 133 134 64)
        (64 134 135 65)
        (65 135 136 66)
        (66 136 137 67)
        (67 137 138 68)
        (68 138 139 69)
    );
 }

 southpatch
 {
     type wall;
     faces
     (
        (1 71 70 0)
        (33 103 104 34)
     );
 }

magnet1
{
    type wall;
    faces
```

```
    (
        (2 72 71 1)
    );
}
magnet2
{
    type wall;
    faces
    (
        (2 72 73 3)
    );
}
magnet3
{
    type wall;
    faces
    (
        (3 73 74 4)
    );
}
magnet4
{
    type wall;
    faces
    (
        (4 74 75 5)
    );
}
magnet5
{
    type wall;
    faces
    (
        (5 75 76 6)
    );
}
magnet6
{
    type wall;
    faces
    (
        (6 76 77 7)
    );
}
magnet7
{
    type wall;
    faces
    (
        (7 77 78 8)
    );
}
magnet8
{
    type wall;
    faces
    (
        (8 78 79 9)
    );
}
magnet9
{
    type wall;
    faces
    (
        (9 79 80 10)
    );
}
magnet10
{
    type wall;
```

```
    faces
    (
        (10 80 81 11)
    );
}
magnet11
{
    type wall;
    faces
    (
        (11 81 82 12)
    );
}
magnet12
{
    type wall;
    faces
    (
        (12 82 83 13)
    );
}
magnet13
{
    type wall;
    faces
    (
        (13 83 84 14)
    );
}
magnet14
{
    type wall;
    faces
    (
        (14 84 85 15)
    );
}
magnet15
{
    type wall;
    faces
    (
        (15 85 86 16)
    );
}
magnet16
{
    type wall;
    faces
    (
        (16 86 87 17)
    );
}
magnet17
{
    type wall;
    faces
    (
        (17 87 88 18)
    );
}
magnet18
{
    type wall;
    faces
    (
        (18 88 89 19)
    );
}
magnet19
{
```

```
    type wall;
    faces
    (
        (19 89 90 20)
    );
}
magnet20
{
    type wall;
    faces
    (
        (20 90 91 21)
    );
}
magnet21
{
    type wall;
    faces
    (
        (21 91 92 22)
    );
}
magnet22
{
    type wall;
    faces
    (
        (22 92 93 23)
    );
}
magnet23
{
    type wall;
    faces
    (
        (23 93 94 24)
    );
}
magnet24
{
    type wall;
    faces
    (
        (24 94 95 25)
    );
}
magnet25
{
    type wall;
    faces
    (
        (25 95 96 26)
    );
}
magnet26
{
    type wall;
    faces
    (
        (26 96 97 27)
    );
}
magnet27
{
    type wall;
    faces
    (
        (27 97 98 28)
    );
}
magnet28
```

```
{
    type wall;
    faces
    (
        (28 98 99 29)
    );
}
magnet29
{
    type wall;
    faces
    (
        (29 99 100 30)
    );
}
magnet30
{
    type wall;
    faces
    (
        (30 100 101 31)
    );
}
magnet31
{
    type wall;
    faces
    (
        (31 101 102 32)
    );
}
magnet32
{
    type wall;
    faces
    (
        (32 102 103 33)
    );
}
frontAndBack
{
    type symmetry;
    faces
    (
        (0 35 36 1)
        (1 36 37 2)
        (2 37 38 3)
        (3 38 39 4)
        (4 39 40 5)
        (5 40 41 6)
        (6 41 42 7)
        (7 42 43 8)
        (8 43 44 9)
        (9 44 45 10)
        (10 45 46 11)
        (11 46 47 12)
        (12 47 48 13)
        (13 48 49 14)
        (14 49 50 15)
        (15 50 51 16)
        (16 51 52 17)
        (17 52 53 18)
        (18 53 54 19)
        (19 54 55 20)
        (20 55 56 21)
        (21 56 57 22)
        (22 57 58 23)
        (23 58 59 24)
        (24 59 60 25)
        (25 60 61 26)
        (26 61 62 27)
```

```
            (27 62 63 28)
            (28 63 64 29)
            (29 64 65 30)
            (30 65 66 31)
            (31 66 67 32)
            (32 67 68 33)
            (33 68 69 34)
            (70 71 106 105)
            (71 72 107 106)
            (72 73 108 107)
            (73 74 109 108)
            (74 75 110 109)
            (75 76 111 110)
            (76 77 112 111)
            (77 78 113 112)
            (78 79 114 113)
            (79 80 115 114)
            (80 81 116 115)
            (81 82 117 116)
            (82 83 118 117)
            (83 84 119 118)
            (84 85 120 119)
            (85 86 121 120)
            (86 87 122 121)
            (87 88 123 122)
            (88 89 124 123)
            (89 90 125 124)
            (90 91 126 125)
            (91 92 127 126)
            (92 93 128 127)
            (93 94 129 128)
            (94 95 130 129)
            (95 96 131 130)
            (96 97 132 131)
            (97 98 133 132)
            (98 99 134 133)
            (99 100 135 134)
            (100 101 136 135)
            (101 102 137 136)
            (102 103 138 137)
            (103 104 139 138)
        );
    }
);


mergePatchPairs
(
);

// ************************************************************************* //
```

controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     magneticFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         100;

deltaT          1;

writeControl    timeStep;

writeInterval   20;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

graphFormat     raw;

runTimeModifiable true;


// ************************************************************************* //
```

topoSetDict

Description: In order to create magnets, the domain needs to be divided into zones. To do so, a subset of cells within the domain is selected to form a so called cellSet. In order to define cellSets and therefore the regions, the following topoSetDict directory was made.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      topoSetDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
     {
        name    magnet1;
        type    faceSet;
        action  new;
        source  patchToFace;
        sourceInfo
        {
            name magnet1;
        }
    }
    {
        name    magnet1;
        type    faceZoneSet;
        action  new;
        source  setToFaceZone;
        sourceInfo
        {
            faceSet magnet1;
        }
    }

\\
.
.
Identical lines for all 32 magnets
.
.
\\

     {
        name    magnet32;
        type    faceSet;
        action  new;
        source  patchToFace;
        sourceInfo
        {
            name magnet32;
        }
    }
    {
        name    magnet32;
        type    faceZoneSet;
        action  new;
        source  setToFaceZone;
        sourceInfo
        {
            faceSet magnet32;
        }
    }

);

// ************************************************************************* //
```

fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    //div(phi,T)      Gauss limitedLinear 1;
}

laplacianSchemes
{
    default         none;
    laplacian(murf,psi) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    T               ;
}


// ************************************************************************* //
```

fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    psi
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}


// ************************************************************************* //
```

APPENDIX D

OpenFOAM Dictionaries for the Simulation of Particle Trajectories

The separation case includes files such as kinematicCloudProperties, kinematicCloudPositions, ParticleTrackProperties, gravitational acceleration (g), mapFieldsDict, decomposeParDict, and ControlDict which will be presented in this section.

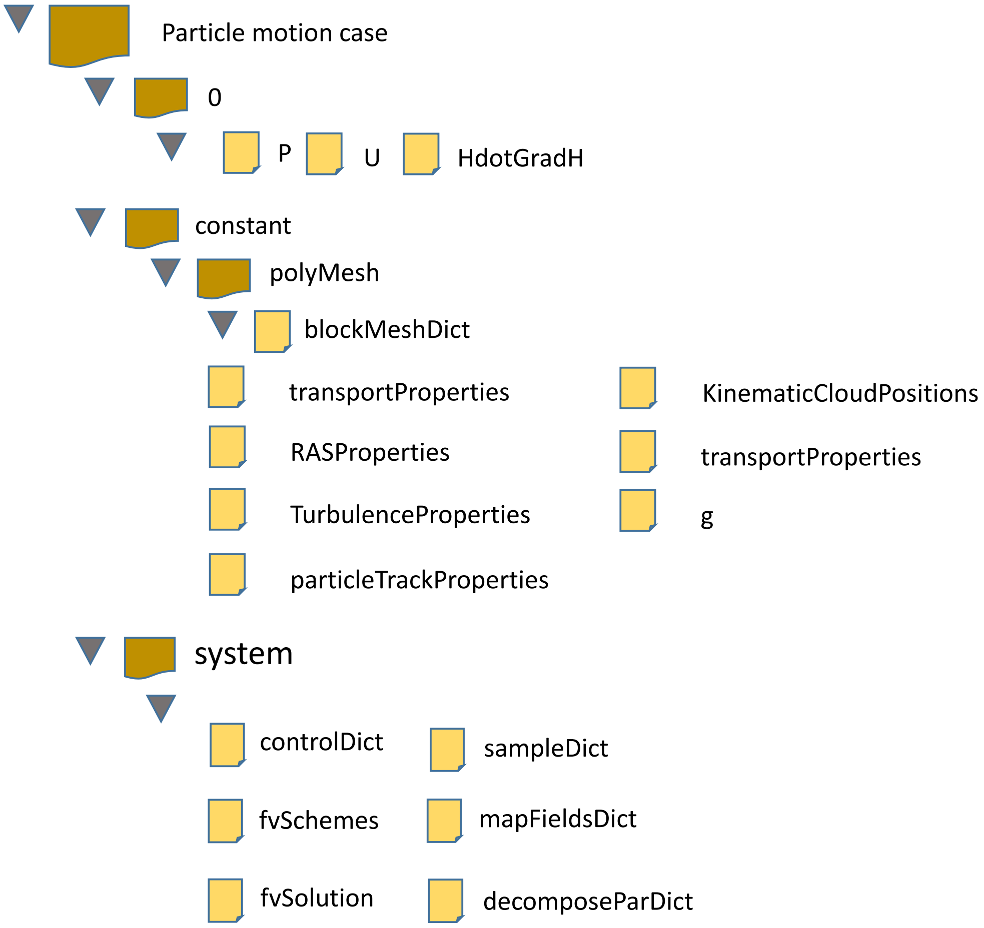


Figure D.1: Particle motion case folder in OpenFOAM.

kinematicCloudProperties

Description: kinematicCloudProperties dictionary is required to set up the properties of the particles. (Number of particles, diameter, initial velocity of each particle, duration of injection, injection model, collision model, wall interaction model, etc.)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      particleProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solution
{
    active          true;
    coupled         false; //true or false for coupled or uncoupled simulations
    transient       yes;   //no for steady-state calculations
    cellValueSourceCorrection off;
    maxCo           0.5;  //maximum Corrent number

    interpolationSchemes
    {
        rho         cell;
        U           cellPoint;
        mu          cell;
        HdotGradH         cellPoint;
    }

    averagingMethod basic;

    integrationSchemes
    {
        U               Euler;
    }

    sourceTerms
    {
        schemes
        {
            /*U semiImplicit 1;*/
        }
    }
}

constantProperties
{
    parcelTypeId 1;

    rho0            1000;   //particle density
    alphaMax        0.9;
    youngsModulus   1e8;
    poissonsRatio   0.35;
}

subModels
{
    particleForces
    {
        sphereDrag;     //drag force
```

```
        gravity;        //gravitational force

        paramagnetic    //magnetic force
        {
            magneticSusceptibility      8e-4; //m^3/kg for beads
            HdotGradH                   HdotGradH;
        }

    }


    injectionModels
    {

        model1
        {
            type            patchInjection;             //patchInjection or manualInjection
            patchName       inlet;                      //activated when we have patch injection
          //positionFile    "kinematicCloudPositions"   //activated when we have manual
injection
            parcelBasisType fixed;                      //three options: number,mass,fixed
            U0              (0.0139 0 0);               //particles' initial velocity
            nParticle       1;                          //number of particles
            parcelsPerSecond 1000;
            sizeDistribution
            {
                type        fixedValue;
                fixedValueDistribution
                {
                    value   15e-6;                      //size of the particles
                }
            }
            flowRateProfile constant 1;
            massTotal       0;                         //total mass to be injected
            SOI 0;                                     //start time of the injection
            duration   4;                              //duration of the injection
        }

        model2
        {
            type            patchInjection;
            patchName       inlet;
            parcelBasisType fixed;
            U0              (0.0139 0 0);
            nParticle       1;
            parcelsPerSecond 1000;
            sizeDistribution
            {
                type        fixedValue;
                fixedValueDistribution
                {
                    value   20e-6;
                }
            }
            flowRateProfile constant 1;
            massTotal       0;
            SOI 0;
            duration   4;
        }

        model3
        {
            type            patchInjection;
            patchName       inlet;
            parcelBasisType fixed;
            U0              (0.0139 0 0);
            nParticle       1;
            parcelsPerSecond 1000;
            sizeDistribution
            {
                type        fixedValue;
```

```
                fixedValueDistribution
                {
                    value   25e-6;
                }
            }
            flowRateProfile constant 1;
            massTotal       0;
            SOI 0;
            duration   4;
        }

    }

    dispersionModel none;

    patchInteractionModel localInteraction;     //interaction with patches

    localInteractionCoeffs
    {
        patches
        (
            frontAndBack
            {
                type stick;
            }
            northpatch
            {
                type stick;
            }
            southpatch
            {
                type stick;
            }
            magnet1
            {
                type stick;
            }
\\
.
.
Identical lines for all 32 magnets
.
.
\\
            magnet32
            {
                type stick;
            }
            inlet
            {
                type stick;
            }
            outlet
            {
                type escape;
            }
        );
    }

    heatTransferModel none;

    surfaceFilmModel none;

    stochasticCollisionModel none;

    collisionModel none; /*pairCollision;*/

    /*pairCollisionCoeffs
    {
        maxInteractionDistance  10e-6;// Maximum possible particle diameter expected at any time
```

```
        writeReferredParticleCloud no;

        pairModel pairSpringSliderDashpot;   //particle particle collision

        pairSpringSliderDashpotCoeffs        //spring slider damper model
        {
            useEquivalentSize   no;
            alpha               0.12;        //restitution coefficient
            b                   1.5;         //Hook's law exponent (b=1 linear)(b=1.5 Hertzian
Theory)
            mu                  0.52;        //friction coefficient
            cohesionEnergyDensity 0;         //binding energy
            collisionResolutionSteps 12;
        };

        wallModel    wallLocalSpringSliderDashpot;   //particle wall collision

        wallLocalSpringSliderDashpotCoeffs
        {
            useEquivalentSize no;
            collisionResolutionSteps 12;
            walls
            {
                type stick;
            }
            frontAndBack
            {
                type stick;
            }
            northpatch
            {
                type stick;
            }
            southpatch
            {
                type stick;
            }
            magnet1
            {
                type stick;
            }
\\
.
.
Identical lines for all 32 magnets
.
.
\\
            Magnet32
            {
                type stick;
            }
        };
    }*/

    packingModel none;

    /*explicitCoeffs
    {
        particleStressModel
        {
            type HarrisCrighton;
            alphaPacked 0.65;
            pSolid 10.0;
            beta 2.0;
            eps 1.0e-7;
        }
        correctionLimitingMethod
        {
            type absolute;
            e 0.9;
```

```
        }
    }

    implicitCoeffs
    {
        alphaMin 0.0001;
        rhoMin 1.0;
        applyGravity false;
        particleStressModel
        {
            type HarrisCrighton;
            alphaPacked 0.65;
            pSolid 5.0;
            beta 2.0;
            eps 1.0e-2;
        }
     }*/

    dampingModel none;

    /*relaxationCoeffs
    {
        timeScaleModel
        {
            type nonEquilibrium;
            alphaPacked 0.65;
            e 0.9;
        }
    }*/

    isotropyModel none;

    /*stochasticCoeffs
    {
        timeScaleModel
        {
            type isotropic;
            alphaPacked 0.58;
            e 0.9;
        }
    }*/

    radiation off;
}


cloudFunctions
{}


// ************************************************************************* //
```

KinematicCloudPositions

Description: This file is required for the injection of the particles manually inside the channel. For this purpose, the position of the particles must be defined in this dictionary.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       vectorField;
    object      kinematicCloudPositions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

(
(0.013  0.0012  0.005)
(x y x)
(x’ y’ z’)
(x” y” z”)
.
.
.
.
)
/ ************************************************************************* //
```

ParticleTrackProperties

Description: This dictionary is required for the post processing part in order to be able to extract the position of the particles at each time step. (The command for this purpose is: trackParticles)

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      particleTrackProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

cloudName       kinematicCloud;

sampleFrequency 1;

maxPositions    1000000;

setFormat       vtk;    // see sampleDict for set formats

// ************************************************************************* //
```

g (gravitational acceleration)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           ( 0 -9.81 0 );


// ************************************************************************* //
```

mapFieldsDict

Description: This dictionary is needed for mapping the flow and magnetic fields on the case where the motion of the particles are modeled. (The directory of this file is in the controlDict folder)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      mapFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

patchMap
(
);

cuttingPatches

(
inlet
outlet
northpatch
southpatch
magnet1
magnet2
magnet3
magnet4
magnet5
magnet6
magnet7
magnet8
magnet9
magnet10
magnet11
magnet12
magnet13
magnet14
magnet15
magnet16
magnet17
magnet18
magnet19
magnet20
magnet21
magnet22
magnet23
magnet24
magnet25
magnet26
magnet27
magnet28
magnet29
magnet30
magnet31
magnet32
frontAndBack
);

// ************************************************************************* //
```

decomposeParDict (Parallel Computing Setup File)

Description: This dictionary is required for parallel computations.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 4;

method          simple;

simpleCoeffs
{
    n               ( 4 1 1 );
    delta           0.001;
}

hierarchicalCoeffs
{
    n               ( 4 1 1 );
    delta           0.001;
    order           xyz;
}

manualCoeffs
{
    dataFile        "";
}

distributed     no;

roots           ( );


// ************************************************************************* //
```

controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     icoUncoupledKinematicParcelFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         3;

deltaT          1e-6;

writeControl    runTime;

writeInterval   0.001;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

// ************************************************************************* //
```