2-1-2013

Feasibility of capturing circulating tumor cells with a magnetized device

Jay Wheaton

Follow this and additional works at: <http://scholarworks.rit.edu/theses>

Recommended Citation

Wheaton, Jay, "Feasibility of capturing circulating tumor cells with a magnetized device" (2013). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Feasibility of Capturing Circulating Tumor Cells With a Magnetized Device

by

Jay Daniel Wheaton

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

Supervised by

Associate Professor Dr. Steven Day
Department of Mechanical Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
February 2013

Approved by:

Dr. Steven Day, Associate Professor
Thesis Advisor, Department of Mechanical Engineering

Dr. Risa Robinson, Associate Professor
Committee Member, Department of Mechanical Engineering

Dr. Linda Barton, Associate Professor
Committee Member, Department of Physics

Dr. Tony Harkin, Associate Professor
Committee Member, Center for Applied and Computational Mathematics

Dr. Alan Nye, Associate Department Head
Department Representative, Mechanical Engineering

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

Feasibility of Capturing Circulating Tumor Cells With a Magnetized Device

I, Jay Daniel Wheaton, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Jay Daniel Wheaton

Date

© Copyright 2013 by Jay Daniel Wheaton
All Rights Reserved

Dedication

This work is dedicated to my late grandfather, Gerald Wheaton, who raised me and inspired my passion for education. I would not be able to push through the demanding dual degree workload without his many life lessons guiding me. Reminiscing on his difficult battle with cancer has provided much needed motivation during long hours of research. I am forever grateful for his contribution to my success.

Acknowledgments

I appreciate all of the assistance I have received from friends, family, faculty, and staff. I especially would like to thank my girlfriend, Tracy Burlingham, for her patience and support throughout my college career. She took care of my responsibilities when research kept me on campus late into the night. I could not have made it this far without her help. My grandmother has also been important in my success and I can not thank her enough. Her kind nature and pride in my accomplishments has motivated me along the way. I would also like to thank my advisor, Dr. Steven Day, for providing me with the opportunity to partake in important research. He has provided the means to do experiments and provided insight when I reached major hurdles. Bishoy Faltas, M.D. from Rochester General Hospital has provided medical expertise and assisted with experiments. His contributions to this research has been extremely helpful and important. In addition, I thank my committee for the much needed assistance that made this work possible. Dr. Tony Harkin specifically helped build the Matlab groundwork that I evolved into an elaborate particle tracking algorithm. I truly appreciate the efforts of everyone that has helped me complete this thesis.

This work was funded by the RIT/RGH Seed Grant #63268

Abstract

Feasibility of Capturing Circulating Tumor Cells With a Magnetized Device

Jay Daniel Wheaton

Supervising Professor: Dr. Steven Day

A standard inferior vena cava (IVC) filter is plated with Nickel in order to be rendered magnetizable. In the presence of an applied magnetic field, the IVC filter has a strong magnetic gradient near the surface that captures circulating tumor cells (CTCs) tagged with paramagnetic nanoparticles. Particle motion and capture includes a balance of the magnetization force and hydrodynamic drag force. ANSYS Fluent CFD software and ANSYS Workbench Magnetostatics were used independently to obtain fluid velocity and magnetic field intensity solutions. The separate solutions were combined in Matlab for numerical particle tracking. Empirical capture efficiency was determined using in-flow spectrometer absorbance measurements as well as measuring image intensity from time-lapse fluorescent images. Three control experiments were used to ensure particles caught in the continuous flow loop are not falsely attributed to the magnetizable IVC filter. Fluorescent magnetite coated spheres, 8.34 micron in diameter, were used for developing methods while magnetically tagged cancer cells were used to determine practical feasibility. Results from the numerical model predict eight percent capture efficiency for a single pass through the IVC filter. Experimental results showed capture efficiency of paramagnetic particles in a continuous loop at 0.3 L/min was ninety-five percent after six hours. For the same experiment, tagged cancer cells resulted in capture efficiency up to ten percent after two hours. A second trial was unable to confirm the magnetized IVC filter had a distinguishable effect on the tagged cancer cells. This work has provided reasonable empirical evidence and numerical results to confirm the feasibility of using a nickel-plated IVC filter for capturing tagged cancer cells.

Contents

Dedication	iv
Acknowledgments	v
Abstract	vi
Acronyms	xvi
List of Symbols	xvii
1 Introduction	1
1.1 Background	1
1.2 Specific Aims	4
1.3 Societal Impact	4
1.4 Funding	5
1.5 Overview of Thesis	5
2 Literature Review	7
3 Numerical Model	9
3.1 Overview	9
3.2 Mathematical Model	10
3.2.1 Magnetization Force $\mathbf{F}_m$	11
3.2.2 Hydrodynamic Drag Force $\mathbf{F}_D$	11
3.2.3 Particle Tracking, $\mathbf{v}_{mp}$	12
3.3 ANSYS Fluent CFD	13
3.3.1 Overview	13
3.3.2 Model	13
3.3.3 Grid Independence	14
3.3.4 Results	19
3.4 ANSYS Workbench Magnetostatics	21
3.4.1 Overview	21
3.4.2 Model	21

3.4.3	Grid Independence	24
3.4.4	Results	30
3.5	Matlab particle tracking algorithm	34
3.5.1	Overview	34
3.5.2	Setup and Parameters	34
3.5.3	Algorithm	36
3.5.4	Wall Contact	39
3.5.5	Results	45
3.6	Discussion	50
4	Experimental Model with Magnetic Particles	52
4.1	Magnetic particle capture efficiency based on spectrometer	52
4.1.1	Overview	52
4.1.2	Setup and Procedure	53
4.1.3	Equations and Assumptions	55
4.1.4	Results	57
4.2	Magnetic particle capture efficiency based on fluorescent images	61
4.2.1	Overview	61
4.2.2	Equations and Assumptions	61
4.2.3	Setup and Procedure	62
4.2.4	Results	63
4.3	Discussion	70
5	Cancer Cell Experiments	72
5.1	Cancer cell capture efficiency based on spectrometer	72
5.1.1	Overview	72
5.1.2	Equations and Assumptions	73
5.1.3	Results	73
5.2	Cancer cell capture efficiency based on fluorescent images	76
5.2.1	Overview	76
5.2.2	Equations and Assumptions	76
5.2.3	Setup and Procedure	76
5.2.4	Results	79
5.3	Discussion	87
6	Conclusion and Discussion	89
6.1	Overview	89

6.2 Future Work	90
Bibliography	91
Appendices	92
A Supplemental Spectrometer Plots	94
B Matlab Particle Tracking Code	98
B.1 Particle Tracking Main GUI	98
B.2 ODE solver	116
B.3 Post-processing Function	127
B.4 Fluid import and interpolation functions	134
B.5 Magnetostatic import and interpolation functions	137
C Matlab Spectrometer Code	140
C.1 Spectrometer calibration processing	140
C.2 Spectrometer η_{capt} processing	148
D Matlab Image Processing Code	168
D.1 Image intensity main GUI	168
D.2 Image integration	175
D.3 Post-processing plots	179
E Spectrometer Tutorial	186
Glossary	194

List of Tables

3.1	Fluid properties for glycerin and water at 20°C [1].	14
3.2	Fluid grid independence mesh comparison.	14
3.3	Pure annealed nickel demagnetization values based on ANSYS Workbench material properties.	23
3.4	Magnetostatic grid independence mesh comparison.	25
3.5	Parameters used for Matlab particle tracking	34
4.1	Contributions to particle capture for four scenarios	53

List of Figures

1.1	Picture of actual Nickel plated IVC filter used in experiments.	2
1.2	Simplified force balance on magnetized particle	3
1.3	Magnetically Tagged Carcinoma Cell	4
3.1	Flowchart for computer simulation utilizing multiple programs.	9
3.2	Dimensions for analytical model.	10
3.3	Fluent model with inlet in blue, outlet in red.	13
3.4	Mesh FA containing 203,268 elements.	15
3.5	Closeup view of mesh FA containing 203,268 elements.	15
3.6	Mesh FB containing 268,287 elements.	15
3.7	Closeup view of mesh FB containing 268,287 elements.	15
3.8	Mesh FC containing 382,409 elements.	16
3.9	Closeup view of mesh FC containing 382,409 elements.	16
3.10	Mesh FD containing 759,908 elements.	16
3.11	Closeup view of mesh FD containing 759,908 elements.	16
3.12	Comparison of fluid velocity solution and convergence for four different meshes. The black, red, green, and blue lines for scaled residual plots represent continuity, x-velocity, y-velocity, and z-velocity respectively. . . .	18
3.13	Fluid velocity magnitude along $X = 2\text{cm}$ line (left of IVC filter tip) for all four meshes.	19
3.14	Velocity pathlines and vectors in m/s for mesh FD in ANSYS Fluent 13.0. .	20
3.15	Velocity pathlines, vectors, and cross-section contour in m/s for mesh FD. The contour plane is located at $x = 2\text{cm}$	20
3.16	Pressure contours in Pascals along the vertical and horizontal center planes for mesh FD.	21
3.17	Magnetic flux parallel boundary conditions, applied to the +X (red arrow) polarized magnets for the local coordinate system shown.	22
3.18	Pure annealed nickel demagnetization curve from values in Table 3.3. . . .	24
3.19	Mesh MA cross-section containing 113,982 elements.	26
3.20	Mesh MA filter	26
3.21	Mesh MB containing 141,532 elements.	27
3.22	Mesh MB filter	27

3.23	Mesh MC containing 289,475 elements.	28
3.24	Mesh MC filter	28
3.25	Mesh MD containing 317,226 elements.	29
3.26	Mesh MD filter	29
3.27	Screen-shot of settings and properties used for Matlab simulation.	35
3.28	Screen-shot of solver parameters used for Matlab simulation.	36
3.29	Screen-shot of initial conditions used for Matlab simulation. See Algorithm section for initial position.	36
3.30	Particle injection location for all twenty-six particles.	39
3.31	Geometry and points used to prevent particle from passing through the boundary (not to scale).	40
3.32	Example of a particle colliding with the cylinder wall. The initial point is cyan colored, projected point based on simulation is green, projected wall contacts are purple and orange, final bounce location is in red, black lines are trajectory paths.	41
3.33	Side view of an example particle bouncing off of the wall.	42
3.34	Three-dimensional plot of captured particles based on simulation. Three particles are captured, two of which are located on the filter.	45
3.35	Particle injection location for captured particles.	46
3.36	Captured particles from simulation with magnetic field intensity vectors on the xy-plane.	46
3.37	Side view of captured particles based on simulation.	47
3.38	Top view of captured particles based on simulation.	47
3.39	Front view of captured particles based on simulation.	48
3.40	Three-dimensional view of particles that escaped through the outlet.	48
3.41	Side view of particles that escaped through the outlet.	49
3.42	Top view of particles that escaped through the outlet.	49
3.43	Front view of particles that escaped through the outlet.	50
4.1	Experimental setup side view.	52
4.2	Experimental flow loop with magnets.	54
4.3	Absorbance is computed from intensity of light through the fluid particle mixture.	54
4.4	Absorbance for calibration of flow loop with $8.34\mu\text{m}$ diameter magnetic particles.	58
4.5	Integrated spectrometer absorbance for 21 different known concentrations.	58
4.6	Capture efficiency with magnetic particles for all four scenarios at 0.3 L/min.	59

4.7	Capture efficiency with magnetic particles for YFYM scenario at 0.3 L/min and 2 L/min.	60
4.8	Original RGB image (left) and masked image showing only processed area (right).	62
4.9	Capture efficiency over 6 hours using fluorescent images.	64
4.10	Time lapse images for YFYM scenario with magnetic particles over six hours.	65
4.11	Time lapse images for YFNM scenario with magnetic particles over six hours.	66
4.12	Time lapse images for NFYM scenario with magnetic particles over six hours.	67
4.13	Time lapse images for NFNM scenario with magnetic particles over six hours.	68
4.14	Time lapse images for the two liters per minute YFYM scenario with magnetic particles over six hours.	69
4.15	Preliminary concentration calibration experiment with stagnant fluid in four identical cuvettes.	70
5.1	Spectrometer capture efficiency for first set of cancer cell data over two hours. The four scenarios are tested in the sequence NFNM, NFYM, YFNM, YFYM.	74
5.2	Spectrometer capture efficiency for second set of cancer cell data over four hours. The four scenarios are tested in the sequence YFYM, YFNM, NFNM, NFYM.	75
5.3	Original YFYM RGB image (left) and masked image with processed area (right) for the first set of data.	77
5.4	Original YFNM RGB image (left) and masked image with processed area (right) for the first set of data.	77
5.5	Original NFNM RGB image (left) and masked image with processed area (right) for the first set of data.	77
5.6	Original NFYM RGB image (left) and masked image with processed area (right) for the first set of data.	77
5.7	Original YFYM RGB image (left) and masked image with processed area (right) for the second set of data.	78
5.8	Original YFNM RGB image (left) and masked image with processed area (right) for the second set of data.	78

5.9	Original NFNM RGB image (left) and masked image with processed area (right) for the second set of data.	78
5.10	Original NFYM RGB image (left) and masked image with processed area (right) for the second set of data.	78
5.11	Time lapse images for YFYM scenario with first set of cancer cells over two hours.	79
5.12	Time lapse images for YFNM scenario with first set of cancer cells over two hours.	80
5.13	Time lapse images for NFYM scenario with first set of cancer cells over two hours.	81
5.14	Time lapse images for NFNM scenario with first set of cancer cells over two hours.	82
5.15	Time lapse images for YFYM scenario with second set of cancer cells over four hours.	83
5.16	Time lapse images for YFNM scenario with second set of cancer cells over four hours.	84
5.17	Time lapse images for NFYM scenario with second set of cancer cells over four hours.	85
5.18	Time lapse images for NFNM scenario with second set of cancer cells over four hours.	86
5.19	Capture efficiency based on normalized camera images taken every thirty seconds converted to grayscale. Both experiments use the same setup except the second set is fluoresced with a continuous laser.	87
A.1	Spectrometer intensity with magnetic particles for all four scenarios at 0.3 L/min.	94
A.2	Spectrometer absorbance with magnetic particles for all four scenarios at 0.3 L/min.	94
A.3	Spectrometer intensity with magnetic particles for YFYM scenario at 0.3 L/min and 2 L/min.	95
A.4	Spectrometer absorbance with magnetic particles for YFYM scenario at 0.3 L/min and 2 L/min.	95
A.5	Spectrometer intensity for first set of cancer cell data over two hours. The four scenarios are tested in the sequence NFNM, NFYM, YFNM, YFYM. .	96
A.6	Spectrometer absorbance for first set of cancer cell data over two hours. The four scenarios are tested in the sequence NFNM, NFYM, YFNM, YFYM.	96

A.7	Spectrometer intensity for second set of cancer cell data over four hours. The four scenarios are tested in the sequence YFYM, YFNM, NFNM, NFYM.	97
A.8	Spectrometer absorbance for second set of cancer cell data over four hours. The four scenarios are tested in the sequence YFYM, YFNM, NFNM, NFYM.	97
B.1	Matlab particle tracking algorithm file structure. Green items are GUI's containing an m-file and fig file and blue items are function files.	98
C.1	Matlab spectrometer calibration processing file structure.	140
C.2	Matlab spectrometer capture efficiency processing file structure.	148
D.1	Matlab image processing file structure.	168
E.1	Spectrometer components	186
E.2	Begin new absorbance measurement	187
E.3	Select spectrometer	187
E.4	Set spectrometer absorbance acquisition parameters	188
E.5	Capture reference spectrum	188
E.6	Capture dark spectrum	189
E.7	Sample spectrum with no light	190
E.8	Sample spectrum with maximum light applied	190
E.9	Setup automatic saving	191
E.10	Automatic save settings	192
E.11	Begin automatic capture	192
E.12	Capture dark spectrum	193
E.13	Sample overlay spectrum	193

Acronyms

AC alternating current. 5

ANSYS ANalysis SYStems by ANSYS, Inc.. vi, x, 6, 8, 9, 14, 21, 23, 24

CFD computational fluid dynamics. vi, 6

CTCs circulating tumor cells. vi, 1, 4, 5

EpCAM Epithelial Cellular Adhesion Molecule. 2, 3, 72

IVC inferior vena cava. vi, xi, 1, 2, 4–8, 13, 17, 19–21, 23, 24, 45, 50, 52, 60, 62, 71, 72, 89

MRI magnetic resonance imaging. 2

NFNM No Filter No Magnet. 53

NFYM No Filter Yes Magnet. 53

ODE ordinary differential equation. 39

PBS phosphate buffered saline. 72, 73

RF radio frequency. 5

VTE venous thromboembolism. 1

YFNM Yes Filter No Magnet. 53

YFYM Yes Filter Yes Magnet. 53

List of Symbols

A Cross-sectional area. 13

η_{capt} Capture efficiency. ix, 4–6, 52, 55, 72, 75, 76, 87–90, 148

χ_{mp} Magnetic susceptibility of magnetic particle. 11

C_{mix} Concentration of mixture. 73

C_{susp} Concentration of cell suspension. 73

ρ Density. 12

D Diameter. 12

F_D Drag force. 10

H_a Applied magnetic field intensity. 11

H Magnetic field intensity. 11, 22

B Magnetic flux density. 11, 22

F_m Magnetic force on a particle. 10

M_s Magnetic saturation. 11

M Magnetization. 11

m_{mp} Mass of magnetic particle. 11

μ Magnetic permeability. 11

μ_0 Magnetic permeability of free space. 11, 34

μ_f Dynamic fluid viscosity. 12

∇ Del operator. 11

N_{mp} Number of magnetic particles. 11

Q Volumetric flow rate. 13

Re Reynolds number. 12

v Average velocity. 12

$\mathbf{v}_{mp}$ Velocity of magnetic particle. 10

V_{mp} Volume of magnetic particle. 11

V_{susp} Volume of cell suspension. 73

Chapter 1

Introduction

1.1 Background

Circulating Tumor Cell

The spread of cancerous tumors from a primary site to a secondary location, or systemic metastasis, are difficult to cure [2]. If circulating tumor cells (CTCs) spread to vital organs, it can be fatal. As a matter of fact, people who die of cancer most often have metastatic disease [2]. Fortunately, there are several methods quickly advancing that may interrupt the metastatic process or remove the CTCs. Advancements in technology have led to the ability for tagging virtually any cell with magnetic particles as long as the surface protein is known. Once tagged and magnetically susceptible, the particles can be attracted to a filter or stent, such as the magnetizable IVC filter proposed herein.

Inferior Vena Cava Filter

The inferior vena cava (IVC) filter has been used in the United States since the 1920s [3]. As stated by Weichman and Ansell, original success rates were low but the IVC filter has been improved and is common today with over 49,000 insertions in 1999 [3]. Traditionally the IVC filter is used to prevent pulmonary embolism, or blockage of the main artery of the lung [4]. This is typically caused by a blood clot from a vein in the leg, termed venous thromboembolism (VTE) [4]. The IVC is located in the abdomen and returns blood to the heart from the lower body [4]. Placement of an IVC filter in a patient is done with a catheter inserted “into a large vein in the neck or groin and advanced to the inferior vena cava in the abdomen” [4]. Due to the ease of insertion with a catheter, the IVC is used for patients with high risk of VTE such as pregnant, burn, cancer, trauma, and surgical patients [3]. Given

the commonality of an IVC filter, an application such as the one proposed herein is realistic and doesn't require invasive surgery. Unlike other stents considered in research, the IVC filter can collect particles at several locations, including the center of the vein due to the geometry of the filter. In addition, a major vein such as the IVC provides the opportunity to process a significant amount of blood over a relatively short period of time, increasing the chances of removing circulating tumor cells. Unfortunately this also means that a higher magnetization force is needed to overcome the large hydrodynamic force. The actual IVC filter used in experiments is shown in Figure 1.1.

Figure 1.1: Picture of actual Nickel plated IVC filter used in experiments.

Epithelial Cellular Adhesion Molecule (EpCAM)

The most common cancer in humans, carcinoma, strongly expresses Epithelial Cellular Adhesion Molecule (EpCAM) which is a surface protein contained in the GA733-2 gene [5]. Each tissue and tumor has a different level of EpCAM expressed. A comprehensive study comparing the level of expression for many different samples has been conducted by Went [5]. According to Went, prostate carcinoma has one of the highest expressions of EpCAM therefore it will be used for experiments. Knowing that the cells express EpCAM, dextran-coated magnetic iron particles can be attached to the cell as shown in Figure 1.3. This tagging of the cancer cells will allow them to be physically maneuvered in the presence of a magnetic field. The dextran coated iron oxides are known to work for magnetic

resonance imaging (MRI) contrast enhancement [6]. The particles “are biocompatible and are excreted via the liver after the treatment” [6]. The reticuloendothelial system removes particles over 10nm while the smaller particles are removed elsewhere in the body such as lymph nodes and bone marrow [6]. This is promising for the future feasibility of in-vivo implementation.

Fundamental Equations

The magnetization force, F_m , and drag force, F_D , are considered the major forces acting on each particle. Equations for the force balance are developed in Section 3.2. The free body diagram with major forces is shown in Figure 1.2.

Figure 1.2: Simplified force balance on magnetized particle

Magnetization force is provided by several magnetic nano-particles attached to the surface of a cell. Attachment is implemented knowing that the surface of epithelial cells express EpCAM. One end of a linking element, labeled the Bi-specific Tetrameric Antibody Complex attaches to the EpCAM surface of the carcinoma cell [7]. The other end links to the dextran-coated iron nano-particle. Once coupled, the magnetic particles on the surface render the carrier cell capable of being attracted to the filter. Figure 1.3 represents the magnetic particle tagging of a cell, as explained by the Stemcell Technologies product information sheet [7].

Figure 1.3: Magnetically Tagged Carcinoma Cell

1.2 Specific Aims

Aim 1: Develop a method for accurately quantifying microparticle capture efficiency, η_{capt} , in a continuous flow loop.

Aim 2: Perform particle tracking η_{capt} for magnetizable IVC filter with spherical particles.

Aim 3: Evaluate η_{capt} of magnetizable IVC filter with spherical particles and tagged cancer cells. Compare analytical and experimental results; report on theoretical validity.

1.3 Societal Impact

Many years of oncology research has allowed for significant advancements in the treatment of cancer. However, many treatments target the malignant tumor, not metastases. Removing the tumor can result in a cancer free patient as long as circulating tumor cells CTCs do not result in more tumors. The proposed method herein aims to inhibit the spread of CTCs. This is not a cure for cancer rather a means to prevent the spread of it. If this method is effective, it could reduce the amount of radiation or chemotherapy needed to eliminate cancer in a patient. Furthermore, the proposed method can be used on patients with weakened immune systems whereas chemotherapy can not. The design can be used for any type of cancer or biomedical application as long as magnetic particles can be attached to the cells.

This flexibility allows targeting of specific cells, which is not the case for all treatment methods. A serious implication of a successful magnetizable IVC filter is prolonged life and less fatalities due to cancer. It will provide doctors with another tool to make treatment possible for people battling cancer.

1.4 Funding

This thesis is in parallel to a project funded by the RIT/RGH Grant #63268. The aims of the grant include characterizing the magnetic properties of the magnetizable IVC filter, finding efficacy in capturing circulating magnetic nanospheres adherent to CTCs, and efficiency of radio frequency (RF) signal to destroy cancer cells. The work herein excludes the RF signal portion due to time constraints but the method has proved to be feasible by other researchers. The heating of “malignant cells while sparing surrounding healthy tissue” is referred as hyperthermia treatment [6]. An alternating current (AC) magnetic field is applied at a specific frequency that causes particles to heat up and kill surrounding diseased tissue. The convection cooling of blood flow is a major limitation of this method [6].

1.5 Overview of Thesis

It is important to capture circulating tumor cells (CTCs) so they do not form malignant tumors elsewhere in the body. Attaching magnetic particles to the CTCs allows an applied local magnet field to manipulate the cells. The results provided herein are targeted at demonstrating the possible effectiveness, or feasibility of capturing the CTCs. Cell capture occurs directly on the magnetized IVC filter or on the walls closest to the permanent magnets. Feasibility is determined by the percent of CTCs captured over a set period of time. This is referred to as η_{capt} .

Although blood flow in the inferior vena cava (IVC) is about 2-5 L/min, a slow flow rate of 0.3 L/min is initially used for methods development. This provides a larger data set because more particles are captured, reducing the percent of error. The more realistic IVC flow rate of 2 L/min is also considered in order to determine a baseline for future

improvement. The η_{capt} is experimentally determined using two quantification techniques due to the small size of the particles. This includes fluorescent image processing as well as measuring the energy absorbed by particles with a spectrometer. A simulation of the experimental setup is modeled in order to compare analytical solutions to actual results. ANSYS Fluent computational fluid dynamics (CFD) software is used to analytically model the experimental flow field. The model also incorporates magnetism from ANSYS Workbench Magnetostatics. Both the fluid and magnetic solution are combined in Matlab and used to determine η_{capt} numerically.

The purpose of this work is to accurately develop methods for determining time dependent η_{capt} of tagged cancer cells on an magnetized IVC filter, not optimization of multiple parameters. Actual tagged cancer cells are used experimentally, after method development with more consistent 8.34 micron fluorescent magnetite coated spheres. All aspects of the experiment are held constant except for flow rate and the type of particles used. The methods developed herein can be improved and used as a viable procedure for treating cancer patients.

Chapter 2

Literature Review

Biocompatible Magnetic Particle Manipulation

The technology needed for the magnetizable IVC filter has been already been partially developed in magnetic drug targeting. Tagged therapeutic drugs are injected in the bloodstream and manipulated to the target site with magnetic fields [6]. Magnetic drug targeting is useful because it can be applied to a tumor without adversely affecting healthy tissue. Another benefit is the required dosage of a drug may be reduced. Due to the hydrodynamic forces, magnetic drug targeting is limited to slow blood flow regions.

The magnetic drug targeting method has been improved using ferromagnetic intravascular stents [8, 9, 10, 11]. It has been shown that the magnetic field gradient is increased locally due to a paramagnetic wire, significantly increasing capture efficiency [8]. In addition, the optimal magnetic field is less than one Tesla because the materials become magnetically saturated [8]. This makes the practical implementation of a magnetizable IVC filter possible.

Similar to magnetic drug targeting, it is possible to perform magnetic particle separation of tagged cancer cells. Instead of directing tagged drugs to a specific location, tagged cells are removed from the bloodstream. This has been successfully conducted in female mice with ovarian cancer without a stent [12]. Malignant cells were magnetically tagged and fluoresced by illumination through the skin. However, a magnet with field strength near 2600 Gauss was placed on the skin of the abdomen which is not practical in a human because the magnetic gradient will not penetrate deep enough. A high-gradient filter such as a plated IVC filter or stent could increase the local magnetic field gradient, making this possible for human use. In addition to experimental validation, the theoretical understanding of a simple wire and magnetic micro particles has been developed [9, 10]. However,

to my knowledge, a magnetic IVC filter has not been considered. The complex geometry of an IVC filter should improve capture efficiency over the simple coil stents considered in literature [8, 9, 10]. Considering an IVC with a 1/2" diameter and volumetric flow rate of 2 liters/min., the linear flow speed in the IVC is average about 26 cm/s. This results in a significantly higher Reynolds number of 33 for the cross section of the tube compared to published results of a coil stent at a Reynolds number of 4.4 [8].

In addition, the flow path considered herein differs from methods considered in literature. Continuous flow over a long period of time is possible by using a pump and reservoir contrary to experiments that use a syringe or pump at the inlet and collect the fluid on the outlet [8, 11]. Multiple passes provides a more realistic and beneficial experiment because people will be wearing this device constantly and it is important to know when the system becomes saturated and can no longer attract particles.

Finite Element Methods

ANSYS Fluent was used by Haverkort to model magnetic particle capture in arterial flows for magnetic drug targeting [13]. Particle tracking was implemented to quantify capture efficiency based on the number of particles that contacted a wall. A magnetic field was modeled as a circular current loop in ANSYS Fluent with a user defined function. The geometry was based on a left coronary artery with an inlet diameter of 4.5 mm. A total of 260 particles were tracked with an inlet velocity of 0.1 m/s. About half of the four micron particles were captured in his model. The main difference between Haverkort's simulation and the magnetic IVC filter simulation provided herein is that his applied magnetic field does not include high-gradient wires. Inclusion of the wires significantly complicates the multi-physical model requiring complicated user defined functions in ANSYS Fluent, large models in Comsol, or a combination of multiple solvers. Solutions have been obtained in FEMLAB (now Comsol) for magnetic particle capture using a magnetizable wire stent which may be more accurate than the methods used herein [8, 9, 14]. However, the simplicity of the Matlab model provided herein makes it easy to change several parameters without re-running the large fluid or magnetic models.

Chapter 3

Numerical Model

3.1 Overview

Particle tracking computer simulation allows for direct comparison to experimental results. The magnetic particles are exposed to hydrodynamic forces and magnetic forces, as explained in Section 3.2. Although it may be possible to run the entire simulation using ANSYS Fluent with the Magneto-Hydrodynamic Module and user defined functions, there are several limitations to this approach. Consequently, multiple commercially available programs are used from start to finish in order to have unhindered control of all parameters. The fluid solution and magnetic solution are solved separately in ANSYS Fluent and ANSYS Magnetostatics, respectively. The geometry is shared between simulations however each requires a different mesh type due to the nature of each solver. Results from both simulations are imported into Matlab and combined in order to track particles.

Figure 3.1: Flowchart for computer simulation utilizing multiple programs.

Figure 3.2: Dimensions for analytical model.

3.2 Mathematical Model

The governing magnetic equation is based on a magnetic-dipole method. Many forces contribute to the capture of particles but the most significant forces are the magnetization force, $\mathbf{F}_m$, and drag force, $\mathbf{F}_D$ [15]. Newton's force balance is shown in Figure 1.2 where $\mathbf{v}_{mp}$ is the velocity vector of the magnetic particle.

$$\mathbf{F}_m + \mathbf{F}_D = m_{mp} \frac{d\mathbf{v}_{mp}}{dt} \quad (3.1)$$

The mass, m_{mp} , and acceleration are assumed to be small and are neglected. As a result,

$$\mathbf{F}_m = -\mathbf{F}_D \quad (3.2)$$

3.2.1 Magnetization Force $\mathbf{F}_m$

The total magnetic force due to N_{mp} particles below saturation attached to a carrier cell exposed to an external field, $\mathbf{H}_a$, is developed by Furlani [15].

$$\mathbf{F}_m = \begin{cases} \mu_0 N_{mp} V_{mp} \frac{3\chi_{mp}}{(\chi_{mp}+3)} (\mathbf{H}_a \cdot \nabla) \mathbf{H}_a & \mathbf{H}_a < \frac{\chi_{mp}+3}{3\chi_{mp}} M_s \\ \mu_0 N_{mp} V_{mp} \frac{M_s}{\mathbf{H}_a} (\mathbf{H}_a \cdot \nabla) \mathbf{H}_a & \mathbf{H}_a > \frac{\chi_{mp}+3}{3\chi_{mp}} M_s \end{cases} \quad (3.3)$$

Where ∇ is the del operator, μ_0 is the permeability of air, V_{mp} is the volume of the particle, χ_{mp} is the magnetic particle susceptibility, and M_s is the magnetic saturation. This equation can be redefined in terms of permeability, μ , using fundamental equations [16]. These equations include magnetic flux density, $\mathbf{B}$, magnetic field intensity, $\mathbf{H}$, and magnetization, $\mathbf{M}$.

$$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}) \quad (3.4)$$

Plugging in $\mathbf{M} = \chi_{mp}\mathbf{H}$ and $\mathbf{B} = \mu\mathbf{H}$

$$\mu = \mu_0(\chi_{mp} + 1) \quad (3.5)$$

This can be rearranged as

$$\chi_{mp} = \frac{\mu_{mp}}{\mu_0} - 1 \quad (3.6)$$

Plugging back into equation 3.3 gives magnetic force in terms of permeability

$$\mathbf{F}_m = \begin{cases} \mu_0 N_{mp} V_{mp} \frac{3(\mu_{mp}-\mu_0)}{(\mu_{mp}+2\mu_0)} (\mathbf{H}_a \cdot \nabla) \mathbf{H}_a & \mathbf{H}_a < \frac{(\mu_{mp}+2\mu_0)}{3(\mu_{mp}-\mu_0)} M_s \\ \mu_0 N_{mp} V_{mp} M_s (\mathbf{H}_a \cdot \nabla) & \mathbf{H}_a > \frac{(\mu_{mp}+2\mu_0)}{3(\mu_{mp}-\mu_0)} M_s \end{cases} \quad (3.7)$$

3.2.2 Hydrodynamic Drag Force $\mathbf{F}_D$

Drag force occurs due to the particle moving in the viscous fluid and always opposes motion. Stokes' drag force equation is used assuming the Reynolds number based on particle

diameter is ≤ 1 . Stokes' drag force equation is shown below [17].

$$\mathbf{F}_D = -6\pi\mu_f r(\mathbf{v}_{mp} - \mathbf{v}_f) \quad (3.8)$$

where μ_f is the dynamic fluid viscosity, r is the radius of the particle, $\mathbf{v}_{mp}$ and $\mathbf{v}_f$ are the respective magnetic particle and fluid velocity. The Reynolds number, Re , is calculated after running simulation in order to verify it is indeed less than one [17].

$$Re = \frac{\rho v D}{\mu_f} \quad (3.9)$$

Equation 3.9 is used to calculate Reynolds number with density, ρ , at 1092.4 kg/m^3 , velocity, v , of 0.082 m/s , sphere diameter, D , of $8.34 \text{ }\mu\text{m}$ and dynamic viscosity, μ_f , of 0.47 Ns/m^2 . This results in a Reynolds number of about $1.6\text{e-}3$ which is much less than one, confirming that the assumption is valid. It should be noted that the velocity used to calculate Re is the maximum velocity of the fluid which assumes that the particle is stopped. There may have been times when the particle moved opposite of the fluid due to the magnetic force, increasing relative velocity to the fluid, but it was never substantially higher.

3.2.3 Particle Tracking, $\mathbf{v}_{mp}$

Equations 3.7 and 3.8 are plugged into 3.2 and solved for $\mathbf{v}_{mp}$.

$$\mathbf{v}_{mp} = \mathbf{v}_f + \frac{\mu_0 N_{mp} V_{mp} (\mu_{mp} - \mu_0)}{2\pi\mu_f r (\mu_{mp} + 2\mu_0)} (\mathbf{H}_a \cdot \nabla) \mathbf{H}_a \quad (3.10)$$

This velocity is used to track the position of the particle using a variable time step ODE solver. For example, the distance moved in the x , y or z coordinate direction, denoted d , is represented by:

$$\Delta d = \mathbf{v}_{mp} \Delta t \quad (3.11)$$

where Δt is a small time step.

3.3 ANSYS Fluent CFD

3.3.1 Overview

The L-shaped section of the experiment containing the IVC filter is considered for analysis. Dimensions are equivalent to the experimental model and can be seen in Figure 3.2. The volumetric flowrate, Q , is related to the cross-sectional area, A , and linear velocity $\mathbf{v}$ as follows:

$$Q = \mathbf{v} \cdot A \quad (3.12)$$

The flowrate is assumed to be constant at 0.3 L/min through the 1.3cm diameter chamber, which correlates to a linear velocity of 3.77 cm/s using Equation 3.12. The fluid is a glycerin/water mixture which correlates to the actual fluid used in experiments.

3.3.2 Model

The Fluent model is shown in Figure 3.3 with the inlet highlighted in blue and the outlet highlighted in red. A laminar solver is used because the Reynolds number is small and the model is relatively simple. Solution methods settings include a SIMPLE scheme, Least Squares Cell Based Gradient, Second Order Pressure and Second Order Upwind Momentum. Default Under-Relaxation Factors are used including 0.3 for pressure, 1 for density and body forces, and 0.7 for momentum. Continuity and the absolute convergence criteria for x, y, and z velocity are set to 1e-06.

Figure 3.3: Fluent model with inlet in blue, outlet in red.

Fluid properties are set to match the experimental fluid. In the experiments, glycerin

and water are mixed to a viscosity of about 4 Centipoise, close to that of blood [18]. The required amount of glycerin is 41.5 wt% or about 36% by volume based on interpolation from an aqueous glycerol solution table at 20°C [1]. Fluid properties for glycerin and water are shown in Table 3.1 based on published values [1].

Fluid	Glycerin	Water	Mixture
Volume %	36%	64%	100%
wt%	41.5%	58.5%	100%
Dynamic Viscosity μ (cP)	1410	1.002	4.062
Density ρ (g/cm³)	1.2613	0.9982	1.103

Table 3.1: Fluid properties for glycerin and water at 20°C [1].

3.3.3 Grid Independence

Four different meshes are developed in order to check grid independence. The first mesh, labeled Mesh FA, is the coarsest at 203,268 elements. Sizing options are applied in the ANSYS DesignModeler environment to get more refined grids up to 759,908 elements. The mesh properties are shown in Table 3.2.

Mesh	FA	FB	FC	FD
# Elements	203,268	268,287	382,409	759,908
# Nodes	44,794	69,361	133,273	360,778
Min Convergence	$5e^{-7}$	$1e^{-7}$	$5e^{-7}$	$5e^{-7}$
Iterations	375	475	800	1100
Max Velocity (m/s)	0.078	0.078	0.081	0.082

Table 3.2: Fluid grid independence mesh comparison.

The meshes from Table 3.2 are shown below.

Figure 3.4: Mesh FA containing 203,268 elements.

Figure 3.5: Closeup view of mesh FA containing 203,268 elements.

Figure 3.6: Mesh FB containing 268,287 elements.

Figure 3.7: Closeup view of mesh FB containing 268,287 elements.

Figure 3.8: Mesh FC containing 382,409 elements.

Figure 3.9: Closeup view of mesh FC containing 382,409 elements.

Figure 3.10: Mesh FD containing 759,908 elements.

Figure 3.11: Closeup view of mesh FD containing 759,908 elements.

A velocity magnitude contour plot is created for each mesh at the center plane ($Z=0$). This is used to visually compare mesh solutions. Scaled residual plots are shown next to each solution in Figure 3.12. An analytical grid independence validation is provided in Figure 3.13 based on velocity values across a line positioned at $X = 2$ cm. This position is just to the left of the IVC filter tip.

(3.12a) Velocity contour (m/s) at the center cross section for mesh FA.

(3.12b) Scaled residuals solution convergence for mesh FA.

(3.12c) Velocity contour (m/s) at the center cross section for mesh FB.

(3.12d) Scaled residuals solution convergence for mesh FB.

(3.12e) Velocity contour (m/s) at the center cross section for mesh FC.

(3.12f) Scaled residuals solution convergence for mesh FC.

(3.12g) Velocity contour (m/s) at the center cross section for mesh FD.

(3.12h) Scaled residuals solution convergence for mesh FD.

Figure 3.12: Comparison of fluid velocity solution and convergence for four different meshes. The black, red, green, and blue lines for scaled residual plots represent continuity, x-velocity, y-velocity, and z-velocity respectively.

Figure 3.13: Fluid velocity magnitude along $X = 2\text{cm}$ line (left of IVC filter tip) for all four meshes.

The wide ranging mesh solutions are very similar, which signifies a grid independent solution. Mesh FD was chosen for particle tracking because it provides closely spaced points which help track the particle. This does introduce more time consuming computations but helps convergence for particle tracking.

3.3.4 Results

The fluid simulation solution for mesh FD is shown below. The velocity pathlines and vectors shown in Figure 3.14 contain vortices that would not exist if the L-shape geometry was reduced to a straight cylinder.

Figure 3.14: Velocity pathlines and vectors in m/s for mesh FD in ANSYS Fluent 13.0.

Figure 3.15: Velocity pathlines, vectors, and cross-section contour in m/s for mesh FD. The contour plane is located at $x = 2\text{cm}$.

Based on the fluid streamlines, more particles will flow towards the bottom. The particles must overcome large hydrodynamic drag force on the bottom due to high velocities but they are also closest to the lower magnet which increases the magnetic force. Fortunately, the streamlines point to the center of the IVC filter which helps attract particles towards the filter where the magnetic gradient is the highest. The force balance is discussed further in Section 3.5.

Figure 3.16: Pressure contours in Pascals along the vertical and horizontal center planes for mesh FD.

3.4 ANSYS Workbench Magnetostatics

3.4.1 Overview

The geometry is not magnetically symmetrical which requires a full three-dimensional model in order to be accurate. Some of the asymmetrical geometry includes the NdFeB magnets offset from the center of the flow loop and the complex shape of the IVC filter. Another reason for doing a full model is to share the same geometry that is used in fluid simulation. This makes Matlab superposition of the magnetostatic and fluid solutions more straightforward.

3.4.2 Model

A box enclosure modeled as air is placed around all objects in ANSYS Workbench. A Magnetic Flux Parallel condition is applied to all six faces of this box, shown in Figure 3.17. The local coordinate system shown in the figure is used to define the polarization of the

magnets, both in the +X direction. All simulation controls are set to be program controlled. The temperature is 22°C and convergence parameters include target change set to 30, allowable change set to max, and max refinement loops set to four. The properties of the magnets are approximated by a linear relationship between the coercive force and residual induction. The coercive force for a grade N42 permanent neodymium magnet is 11,000 Oe with a residual induction at 13,200 gauss at 20°C [19]. A non-linear model requires several data points to make a **BH** curve, or flux density vs. field intensity curve. A linear magnetization solver is chosen over a non-linear model for several reasons. These include poor non-linear convergence due to irregular mesh, faster computation time, lack of accurate curve data, and most importantly the number of elements is limited by the available license. A non-linear magnetic material model was tested using a **BH** curve but it did not solve without significantly reducing convergence criteria.

Figure 3.17: Magnetic flux parallel boundary conditions, applied to the +X (red arrow) polarized magnets for the local coordinate system shown.

Default ANSYS Workbench magnetic material properties for pure annealed nickel are used for the IVC filter. The list of values is shown in Table 3.3. In the physical experiments, the IVC filter is only coated with nickel. As a result, the analytical model is slightly overestimating the actual magnetic field strength.

Magnetic Field Intensity (Oe)	Magnetic Flux Density (Gauss)
0	0
0.59816	800
0.99777	1280
1.9943	2200
3.9886	3500
5.9828	3900
7.9874	4000
9.9739	4100
19.943	4400
39.898	4750
59.841	5050
99.739	5500
199.47	5800
398.93	6100
797.86	6700
1595.7	7800
3191.5	9800
6382.9	13600
8796.5	16300

Table 3.3: Pure annealed nickel demagnetization values based on ANSYS Workbench material properties.

Figure 3.18: Pure annealed nickel demagnetization curve from values in Table 3.3.

3.4.3 Grid Independence

A different mesh for the magnetostatic solution is required because it needs electromagnetic physics preference. This includes specific elements and connections that are designed for magnetostatic equations. The magnetic model also must include the volume of the IVC filter whereas the fluid model only models the exterior surface. Four different meshes are developed in order to check the sensitivity of the magnetic solution. These are referred to as MA, MB, MC, MD ranging from coarse to medium-fine. Various body sizing options were implemented in ANSYS DesignModeler in order to obtain each mesh. Due to minimal mesh control in ANSYS Workbench and complicated IVC filter geometry, a well defined grid with smooth transitions was not achieved. Refining the mesh is beyond the scope of this work and is left for future improvement. With that said, the difference between the four meshes is minimal. The magnetic field intensity vectors were similar across all four meshes. The mesh properties are shown in Table 3.4 followed by figures of each mesh.

Mesh	MA	MB	MC	MD
# Elements	113,982	141,532	289,475	317,226
# Nodes	172,629	217,928	392,811	472,849
CSG Convergence	71.1	227	354	87.6
Iterations	9	8	8	9
Max $\mathbf{H}$ (A/m)	$8.15e^5$	$8.01e^6$	$8.07e^5$	$8.02e^5$

Table 3.4: Magnetostatic grid independence mesh comparison.

Figure 3.19: Mesh MA cross-section containing 113,982 elements.

Figure 3.20: Mesh MA filter

Figure 3.21: Mesh MB containing 141,532 elements.

Figure 3.22: Mesh MB filter

Figure 3.23: Mesh MC containing 289,475 elements.

Figure 3.24: Mesh MC filter

Figure 3.25: Mesh MD containing 317,226 elements.

Figure 3.26: Mesh MD filter

3.4.4 Results

The solution for the magnetic field intensity is shown for all four meshes. The vector direction and magnitude are very similar for each model. A closer look at the filter shows different maximum magnetic field intensity values. This is caused by a sharp edge at the tip of the filter. The highest gradients are clearly on the filter and near the corners of the magnets.

(3.27a) H (A/m) for mesh MA.

(3.27b) H (A/m) on the filter for mesh MA.

(3.27c) H (A/m) for mesh MB.

(3.27d) H (A/m) on the filter for mesh MB.

(3.27e) H (A/m) for mesh MC.

(3.27f) H (A/m) on the filter for mesh MC.

(3.27g) H (A/m) for mesh MD.

(3.27h) H (A/m) on the filter for mesh MD.

3.5 Matlab particle tracking algorithm

3.5.1 Overview

Particle tracking is implemented in Matlab R2011a using the independent fluid and magnetic solutions shown in Sections 3.3 and 3.4. Specifically, fluid velocity $\mathbf{v}_f$ and applied magnetic field intensity $\mathbf{H}_a$ are imported into Matlab with respective coordinates. Matlab's TriScatteredInterp function arranges solution values using nearest-neighbor interpolation. The hydrodynamic and magnetic force are interpolated at each time step. Combining these values using Equation 3.2 yields the distance traveled in three dimensional space for a small time step. This is repeated until the particle escapes through the outlet or reaches a threshold particle capture velocity, set at 0.5 mm/sec. A graphical user interface (GUI) is used to streamline the import, processing, and post-processing steps. This helps avoid user error and can improve particle tracking setup time.

3.5.2 Setup and Parameters

A summary of the parameters used for simulation are shown in Table 3.5.

Variable	Value	Unit
Paramagnetic Particle		
μ_0	1.25663706e-06	H/m
μ_{mp}	8.75e-04	H/m
Dia_{mp}	8.34	μm
ρ_{mp}	1580	kg/m^3
$H_{sat_{mp}}$	500	A/m
Fluid Properties		
ρ_{fl}	1092.4	kg/m^3
Dia_{tube}	1.3	cm
μ_{fl}	0.47	$N \cdot s/m^2$

Table 3.5: Parameters used for Matlab particle tracking

In order to run the program, several GUI's interact with sub-functions that are subsequently used to solve particle motion with forward dynamics. The driving forces are

obtained from pre-solved fluid and magnetic data. The time-marching algorithm is further explained in Section 3.5.3. The main Matlab GUI is shown in Figure 3.27 followed by the Solver Parameter GUI and Initial Condition GUI. The main GUI will set default values and call other GUI's depending on user input. The Verify All Values Set button will ensure Fluid and Magnetic solution data has been imported and all initial conditions and parameters are set. Afterwards, a Solve button becomes visible which calls the main particle tracking function upon selection.

Figure 3.27: Screen-shot of settings and properties used for Matlab simulation.

Figure 3.28: Screen-shot of solver parameters used for Matlab simulation.

Figure 3.29: Screen-shot of initial conditions used for Matlab simulation. See Algorithm section for initial position.

3.5.3 Algorithm

Several steps are required to track the magnetic particles. The first pre-processing step is to import the fluid and magnetic solutions which were created by ANSYS Fluent and Workbench respectively. The fluid and magnetic properties must also be set because they are used for calculations. After the fluid and magnetic solutions are imported, the initial conditions should be determined. All initial velocities are assumed to be zero because the

particle will reach the fluid velocity after a few iterations. The solver parameters include relative and absolute tolerance set to $1e-05$, interpolation distance set to 0.1mm, capture velocity of 0.5mm/s or less, and maximum step size of 0.1 second. The time and iteration limits are set very high at 42 minutes and 15,000 iterations per particle in order to allow the program to reach a solution while still being reasonable. Once all parameters are set, the number of particles to track is set to twenty-five plus one particle at the center for a total of twenty-six particles. Solving the algorithm requires several steps. The overview of the code is as follows.

1. Create interpolation function based on imported fluid solution data.
2. Create interpolation function based on imported magnetic solution data.
3. Set ODE45 initial conditions based on the pre-processed inputs.
4. Determine fluid velocity at the current point in the x, y, and z direction using the fluid interpolation function.
5. Determine magnetic $\mathbf{H}_a$ field at the current point in the x, y, and z direction using the magnetic interpolation function.
6. Determine magnetic $\mathbf{H}_a$ field offset from the current point by x - 0.1mm, y - 0.1mm, and z - 0.1mm.
7. Determine divergence of $\mathbf{H}_a$ by summing the gradient in the x, y, and z direction using the previous two steps.
8. Calculate the magnetic force on the particle after checking if the particle is magnetically saturated (Equation 3.7).
9. Calculate the velocity of the magnetic particle by combining the magnetic and fluid forces using Equation 3.10.
10. Send the current velocity information back to ODE45 in order to determine the next time step and check events.

11. The ODE45 solver checks if iteration or time limit has been reached, if particle exited through the outlet, if the particle is captured by comparing velocity to the minimum threshold parameter (0.5mm/s), or if the particle passed through the wall boundary. If the particle is none of the above, a time step is chosen based on convergence criteria and the next iteration begins. If one of the above is met, excluding passing the boundary, the ODE45 solver performs a cleanup routine and ends the simulation.
12. If the particle passes through the boundary, a function is called to interpolate the location of the particle. The location is intended to represent a particle bouncing off the wall. The equations behind this function are explained in detail below. After the coordinates are determine, the ODE45 solver is essentially reset. The interpolated location becomes the new initial condition along with the current time.
13. After each particle is tracked, the data is sent back to the workspace and saved. A new particle location is chosen and the ODE45 solver is called again. This process is repeated until all particles are tracked.
14. Once all particles are tracked and saved, several post-processing options are available. See Figure 3.27 for these options.

Matlab code is provided in the appendix section for further details. Particle injection location is determined based on the number of particles. Twenty-six particles are tracked in this work, all starting near the inlet. There are three circles concentric with the inlet at a radius of $\frac{1}{8}dia$, $\frac{1}{6}dia$, and $\frac{1}{4}dia$. Each circle contains eight evenly spaced particles. There is also one particle directly in the center. A visual representation of this is shown in Figure 3.30.

Figure 3.30: Particle injection location for all twenty-six particles.

3.5.4 Wall Contact

Tracking the particles only using the fluid solution is straight forward. A simple iterative ordinary differential equation (ODE) solver can interpolate the fluid solution data and project the particles trajectory. Combined fluid and magnet particle tracking is complicated by multiple interpolations as well as particles escaping through the walls. The fluid velocity at the wall is zero due to a no slip boundary condition but the magnetostatic solution attracts particles through the fluid wall boundary. An algorithm is included in the particle tracking code that prevents a particle from escaping the fluid/wall boundary by treating it as an inelastic collision. Figure 3.31 represents the 3D geometry (not to scale) and shows the generic points used to develop boundary contact equations.

Figure 3.31: Geometry and points used to prevent particle from passing through the boundary (not to scale).

To determine contact with the wall, two different equations are required depending on the location of the particle. The slope of a particle with respect to the lower left corner is computed with equation 3.14. This value is used to determine which boundary contact equation to use.

$$m_{ab} = \frac{Y_a - Y_b}{X_a - X_b} \quad (3.13)$$

$$m_{mp} = \frac{Y_{mp} - Y_b}{X_{mp} - X_b} \quad (3.14)$$

The L-shaped geometry is split into two cylinders. The algorithm contains if statements and equations that automatically take care of the transition point from the upper cylinder to the lower cylinder, reducing computationally intensive loops that check for wall boundaries. For the geometry used in simulation and experiments, slope m_{ab} equals one, which is used to check which cylinder's equations should be implemented. Consequently if the slope to the magnetic particle m_{mp} is less than or equal to one, it is contained in the lower, horizontal cylinder. If m_{mp} is greater than one, it is contained in the upper, vertical cylinder. This if statement is implemented in Matlab using an Event structure in order to check if the particle has reached the wall boundary. When the particle is in the upper cylinder, the boundary circle lies somewhere on the X-Z plane and must satisfy Equation 3.15 if it is inside the

fluid boundary.

$$(X - X_1)^2 + (Z - Z_1)^2 \leq (dia/2)^2 \quad (3.15)$$

When the particle is in the lower cylinder, the boundary circle lies somewhere on the Y-Z plane and must satisfy Equation 3.16 if it is inside the fluid boundary.

$$(Y - Y_2)^2 + (Z - Z_2)^2 \leq (dia/2)^2 \quad (3.16)$$

If the particle trajectory lies outside of the boundary, an interpolation function places the particle in the appropriate location inside of the cylinder as if it bounced against the wall. The governing equations for the interpolation are explained in depth below. Figure 3.32 shows an example of the particle bounce interpolation.

Figure 3.32: Example of a particle colliding with the cylinder wall. The initial point is cyan colored, projected point based on simulation is green, projected wall contacts are purple and orange, final bounce location is in red, black lines are trajectory paths.

Figure 3.33: Side view of an example particle bouncing off of the wall.

Interpolation equations are based on three points, shown in Figure 3.33. The first point, P_0 , represents the the location of the particle immediately before crossing the boundary. The second point, P_1 , is the projected particle location based on the time step and velocity. The slope of the line between the current point and previous point, projected onto the x-y plane is represented by Equation 3.17.

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \quad (3.17)$$

Rearrange to obtain an equation for the line.

$$y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0 \quad (3.18)$$

Similarly, the slope of the line between the current point and previous point, projected onto the x-z plane is represented by Equation 3.19.

$$\frac{z - z_0}{x - x_0} = \frac{z_1 - z_0}{x_1 - x_0} \quad (3.19)$$

Rearrange to obtain an equation for the line.

$$z = \frac{z_1 - z_0}{x_1 - x_0}(x - x_0) + z_0 \quad (3.20)$$

Equation 3.21 represents the circle along the x-axis:

$$y^2 + z^2 = R^2 \quad (3.21)$$

Plug Equations 3.18 and 3.20 in to Equation 3.21.

$$\left[\frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0 \right]^2 + \left[\frac{z_1 - z_0}{x_1 - x_0}(x - x_0) + z_0 \right]^2 = R^2 \quad (3.22)$$

At this point, it makes sense to introduce new variables. Let $a = x_1 - x_0$, $b = y_1 - y_0$, $c = z_1 - z_0$, $d = x - x_0$. These variables are plugged in to Equation 3.22 and rearranged to form a quadratic equation in terms of d .

$$d^2 \left(\frac{b^2 + c^2}{a^2} \right) + d \left(\frac{2by_0 + 2cz_0}{a} \right) + y_0^2 + z_0^2 - R^2 = 0 \quad (3.23)$$

This quadratic equation can now be solved in Matlab given two known particle locations. Since there are two solutions, the root that is closest to the original point is chosen. The solution to Equation 3.23 only represents the x-coordinate at which the particle crossed the boundary. The solution to Equation 3.23 is plugged in to Equations 3.20 and 3.18 in order to get all three coordinates. The total length from P_0 to P_1 is referred to as L_1 .

$$L_1 = \sqrt{a^2 + b^2 + c^2} \quad (3.24)$$

Similarly, the total length from P_0 to P_n is referred to as L_2 .

$$L_2 = \sqrt{(x_n - x_0)^2 + (y_n - y_0)^2 + (z_n - z_0)^2} \quad (3.25)$$

Subtracting L_2 from L_1 yields the length from P_n to P_1 and is referred to as L_3 .

$$L_3 = L_1 - L_2 \quad (3.26)$$

The kinematics of the particle-wall collision are not considered in detail. Rather, the bounce distance is arbitrarily assumed to be half of the total distance traveled outside of the boundary, equal to L_3 . In order to determine the coordinates of the particle after bouncing off

the wall, a line perpendicular to the P_0 - P_1 line is determined. The slopes for the y and z equations are simply the negative inverse of Equations 3.18 and 3.20.

$$m_y = -a/b \quad (3.27)$$

$$m_z = -a/c \quad (3.28)$$

The final point after bouncing off the wall is referred to as P_f . The equations for the y and z coordinate in terms of x are as follows.

$$y_f = m_y(x_f - x_n) + y_n \quad (3.29)$$

$$z_f = m_z(z_f - z_n) + z_n \quad (3.30)$$

As previously stated, the distance of P_f away from the intersection point will be ζL_3 where ζ is assumed to be 1/2.

$$\sqrt{(x_f - x_n)^2 + (y_f - y_n)^2 + (z_f - z_n)^2} = \zeta L_3 \quad (3.31)$$

For further simplification, variables α , β , and γ are introduced. Let $\alpha = x_f - x_n$, $\beta = y_f - y_n$, $\gamma = z_f - z_n$. These are plugged into Equation 3.31 to obtain the following equation.

$$\alpha^2 + \beta^2 + \gamma^2 = \zeta^2 L_3^2 \quad (3.32)$$

Combining and simplifying Equations 3.29, 3.30, and 3.32 results in the following quadratic equation.

$$\alpha^2 (1 + m_y^2 + m_z^2) - \zeta^2 L_3^2 = 0 \quad (3.33)$$

The solution for α can be used to obtain P_f .

$$x_f = \alpha + x_n \quad (3.34)$$

$$y_f = m_y \alpha + y_n \quad (3.35)$$

$$z_f = m_z \alpha + z_n \quad (3.36)$$

3.5.5 Results

The simulations represent a single pass of twenty-six particles. There are three particles that slowed down below the capture threshold of 0.5mm/sec. All particles converged, therefore the other twenty-three particles escaped through the outlet. A three-dimensional plot of the captured particles is shown in Figure 3.34. It is apparent that one of the three captured particles is due to a vortex in the fluid, not the magnetized IVC filter. Consequently, about eight percent of particles are captured on a single pass.

Figure 3.34: Three-dimensional plot of captured particles based on simulation. Three particles are captured, two of which are located on the filter.

The corresponding inlet location viewed from above is shown in Figure 3.35.

Figure 3.35: Particle injection location for captured particles.

A closeup of the captured particles and $\mathbf{H}_a$ in the x-y plane is shown in Figure 3.36. The spacing of the vectors is dependent on the mesh.

Figure 3.36: Captured particles from simulation with magnetic field intensity vectors on the xy-plane.

Figure 3.37: Side view of captured particles based on simulation.

Figure 3.38: Top view of captured particles based on simulation.

Figure 3.39: Front view of captured particles based on simulation.

The rest of the particles were not slowed down enough to be captured. Particle tracking solutions for these are shown below.

Figure 3.40: Three-dimensional view of particles that escaped through the outlet.

Figure 3.41: Side view of particles that escaped through the outlet.

Figure 3.42: Top view of particles that escaped through the outlet.

Figure 3.43: Front view of particles that escaped through the outlet.

3.6 Discussion

The three dimensional multi-physics particle tracking model developed herein has provided promising results. Two out of twenty-six particles are captured by the filter from a single pass. Given the empirical setup with a total fluid volume of 750mL and an average flowrate of 0.3L/min, a single flow loop pass takes about 2.5 minutes. Simply put, the numerical model predicts eight percent of particles captured every 2.5 minutes based on the experimental setup. Several parameters can be modified in Matlab to obtain better results such as particle diameter, particle material, fluid viscosity, magnetic saturation, and possibly others. Although not considered in this work, the magnetostatic model and IVC filter can be re-oriented and modified to obtain optimal results without changing the Matlab code. There are some limitations to the final algorithm. These include the requirement of an L-shaped flow chamber (unless the wall bounce function is ignored), no inertial term, and a

simplified one-dimensional magnetic field intensity gradient calculation. With that considered, the groundwork provided herein is easily adapted and can be used to predict realistic single-pass capture efficiency.

Chapter 4

Experimental Model with Magnetic Particles

4.1 Magnetic particle capture efficiency based on spectrometer

4.1.1 Overview

The experimental setup is comprised of a centrifugal pump, two Neodymium magnets, and a flow loop. A side view is shown in Figure 4.1.

Figure 4.1: Experimental setup side view.

() for the magnetized IVC filter is continuously monitored through an acrylic window for six hours. An OceanOptics USB2000+ spectrometer with an LS-1 tungsten halogen lamp captures the light intensity across the fluid. This is used to calculate absorbance which correlates to the amount of particles in the fluid. If the absorbance is zero, no particles are in the fluid. Consequently, higher absorbance correlates to more particles in the

fluid. Particle concentration is linearly proportional to absorbance according to the Beer-Lambert law shown in Equation 4.2 [20]. The proportionality constant relating absorbance to concentration is determined by evaluating the slope of absorbance vs. known concentration through calibration. The benefit of knowing this relationship is that an unknown concentration of particles can be quantitatively determined from an absorbance measurement during the experiments. For experimental results, the amount of particles captured for four different scenarios is plotted as a percentage. The test scenario is known as Yes Filter Yes Magnet (YFYM). There are three control scenarios that determine particle accumulation due to the flow loop, magnet, and filter itself. These are referred to as No Filter No Magnet (NFSM), No Filter Yes Magnet (NFYM), and Yes Filter No Magnet (YFSM). Table 4.1 below shows what factors contribute to particles being captured for each scenario.

	Particles Captured By:		
Scenario	Pump/Tubing	Filter	Magnet
NFSM	x		
YFSM	x	x	
NFYM	x		x
YFYM	x	x	x

Table 4.1: Contributions to particle capture for four scenarios

4.1.2 Setup and Procedure

In order to maintain a similar viscosity to blood, a mixture of 36% glycerin to water by volume is used for each scenario. A single blank solution is mixed in a large container to ensure consistency across each scenario prior to separating into smaller containers at 750mL each. The four scenarios used in experiments are identical in fluid and setup except for the magnet or filter being added or removed in the flow loop section shown in Figure 4.2.

Figure 4.2: Experimental flow loop with magnets.

Spectrometer data acquisition is automated using the Ocean Optics Spectrasuite software. The spectrometer setup consists of 4 major components, shown in Figure 4.3. A tungsten halogen light source provides bright visible light through a small fiber optic cable. This cable is positioned perpendicular to flow with a polished acrylic fixture. Another fiber optic cable receives the light on the other side of the fixture. The light is sent into the spectrometer which separates the light into intensity at specific wavelengths ranging from 200-1100 nm. Finally, intensity data is converted to absorbance using computer software.

Figure 4.3: Absorbance is computed from intensity of light through the fluid particle mixture.

For calibration, the NFNM scenario is implemented which is the experimental flow loop minus the filter and magnets. This provides an accurate baseline but introduces error in measurement due to the dynamic, non-homogenous fluid. Calibration is implemented by first measuring absorbance of a 750 ml blank solution of 36 % glycerin to water by volume. Afterwards, 35 μ L of particles is added, stirred with a glass stir rod, then run at 0.3 L/min for five minutes in order to mix particles. Immediately following, the spectrometer is turned on and data is collected every 1.8 seconds for five minutes. This is repeated at increments of 35 μ L until the total volume of particles reaches 0.7 ml.

As for the η_{capt} experiments, only one concentration is injected. All subsequent data is collected automatically using the *Ocean Optics Spectrasuite* software. The flow rate is checked periodically with a Transonic T206 ultrasonic flow meter in order to obtain consistent data. Prior to each η_{capt} data set, the entire flow loop is rinsed and flushed with at least 1 gallon of distilled water in order to remove any particles from previous runs. Once rinsed, 750mL of the blank solution is poured into the reservoir and pumped through several times. Afterward, the spectrometer is calibrated using this blank solution at the specified flow rate. This is known as a reference spectrum and it is used in the absorbance calculation. A dark spectrum is also captured by blocking the light source. It is important to make sure the light source is warm before capturing any spectrum data. With the spectrometer calibrated, particles are carefully pipetted into the reservoir and stirred with a glass stir rod. This is not a perfect mixture of particles but it mixes quickly once the pump is turned on. The spectrometer begins collecting data slightly before the pump is started. The spectrometer acquisition time is 1.8 seconds and averages 10 samples in order to reduce noise in the data. This results in a new spectrum file every 18 seconds. Specific settings and a brief tutorial of the software can be found in Appendix E.

4.1.3 Equations and Assumptions

A reference intensity, I_R , and dark intensity, I_D , are calibrated at each wavelength for a blank solution. The sample intensity, I_S , is measured for each concentration and used to

calculate the absorbance, A , using Equation 4.1.

$$A = -\log_{10} \left(\frac{I_S - I_D}{I_R - I_D} \right) \quad (4.1)$$

The Beer-Lambert law is used to correlate absorbance of light to the properties of the fluid/particle mixture. This results in a direct relationship between absorbance and concentration [20].

$$A = \epsilon c l \quad (4.2)$$

Where ϵ is the extinction coefficient of particles, c is the concentration, and l is the optical path length.

The particle concentration is determined based off of known values from SpherotechTM. The particles are supplied in a suspension of 1% w/v which means 10 mg of particles per 1 mL of solution. Mean particle diameter, d_{mp} and particle density, ρ_{mp} are 8.34 μm and 1.58 g/cm^3 respectively [Spherotech data sheet]. Equation 4.4 can be used to determine the number of particles per ml of suspension fluid.

$$\%w/v = \frac{\rho_{mp} V_{mp} N_{mp}}{V_{sf}} \quad (4.3)$$

Where V_{mp} is the volume of each magnetic particle, N_{mp} is the number of particles, and V_{sf} is the volume of the suspension fluid. This equation can be rearranged to solve directly for the number of particles for a given amount of suspension of fluid.

$$N_{mp} = \frac{6 * (\%w/v) * V_{sf}}{\rho_{mp} \pi d_{mp}^3} \quad (4.4)$$

For the calibration experiments, 35 μL of suspension fluid is added, equating to approximately 730,000 particles for each injection as shown in Equation 4.5.

$$N_{mp} = \frac{6 * (10\text{g}/\text{L}) * (35 * 10^{-6}\text{L})}{(1.58\text{g}/\text{cm}^3) * \pi * (8.34 * 10^{-4}\text{cm})^3} = 730,000 \quad (4.5)$$

The actual concentration of particles in the entire flow system is simply the total number

of particles divided by the total amount of fluid in the system. The volume of the particles is neglected when calculating concentration. Given the initial glycerin-water mixture of 750mL, each addition of 730,000 particles yields a concentration increase on the order of $1e^6$ particles per liter of fluid. Several assumptions are made for the spectrometer experiments. The primary assumptions are listed below.

1. The particles are perfectly distributed throughout the fluid.
2. The reference and dark spectrum do not change once calibrated.
3. All particles contribute equally to absorbance.
4. No particles are captured in the flow loop.

4.1.4 Results

Calibration

The spectrometer absorbance measurement for concentrations between 0 and about 19 million particles per liter is shown in Figure 4.4. These absorbance measurements are integrated from 450 to 700nm in order to collect a sufficiently large set of data. The integrated absorbance is plotted against known concentrations in order to obtain an equation relating absorbance and concentration. This is shown in Figure 4.5 and color coding is used to show the different concentrations, not the wavelengths.

Figure 4.4: Absorbance for calibration of flow loop with $8.34\mu\text{m}$ diameter magnetic particles.

Figure 4.5: Integrated spectrometer absorbance for 21 different known concentrations.

Capture Efficiency

All four scenarios are analyzed as well as the two liter per minute YFYM scenario. The intensity plot is based off of the raw spectrometer data. This is converted to absorbance through Equation 4.1 which is finally converted to capture efficiency based on the light and dark spectrometer calibration. Thousands of data points are plotted for both the intensity and absorbance measurements which are included in Appendix A, Figures A.1 and A.2. The resulting capture efficiency plots for all four 0.3L/min scenarios are shown in Figure 4.6.

Figure 4.6: Capture efficiency with magnetic particles for all four scenarios at 0.3 L/min.

Similarly, the YFYM scenario for both 0.3 L/min and 2 L/min flowrates is shown in Figure 4.7 below. The supporting absorbance and intensity plots are located in Appendix A,

Figures A.3 and A.4.

Figure 4.7: Capture efficiency with magnetic particles for YFYM scenario at 0.3 L/min and 2 L/min.

Capture efficiency at the idealistic 0.3 L/min is around ninety-five percent after subtracting the control experiments. When flowrate is increased to 2.0 L/min the efficiency drops to about ten percent. It is important to note the magnets alone contribute to about half of the capture efficiency while the IVC filter alone has minimal effect when magnets are not present.

4.2 Magnetic particle capture efficiency based on fluorescent images

4.2.1 Overview

High resolution images are captured every minute for 6 hours simultaneously with the spectrometer experiment in Section ???. A 532nm continuous 300mW laser is used to fluoresce the magnetic particles. A wavelength specific optical filter is placed in front of the camera lens in order to filter out the laser light at 532nm. The image intensity at each pixel is added and plotted versus time. Capture efficiency trends are compared to results obtained using the spectrometer.

4.2.2 Equations and Assumptions

The image processing equations are kept very simple. Three-dimensional matrices obtained from RGB images are converted to two-dimensional matrices in the form of grayscale images using Matlab's *rgb2gray* function. There are 3072 by 4608 pixels per grayscale image. Summation of the pixel intensity of each image, I_n , is calculated with masked regions set to zero.

$$I_n = \sum_{i=1}^{3072} \sum_{j=1}^{4608} p_n(i, j) \quad (4.6)$$

Where n represents the current image, p stands for pixel, i and j are the horizontal and vertical pixel locations, and I is the image intensity.

The capture efficiency is a normalized value. It is determined by assuming the highest intensity image is one-hundred percent capture efficiency and all other images are a relative percentage of that value. This is clearly not to be interpreted as a quantitative measurement of how many particles are captured. It is only intended to show the relative percent of particles captured over time.

$$\eta_{capt_n} = \frac{I_n}{\max(I_n) \big|_{n=1}^{n=end}} \quad (4.7)$$

Several assumptions are required in order to use this simplified image processing technique.

1. The mask does not cover any of the particles.

2. Image location does not shift; the mask covers the same region for all images.
3. The only source of intensity is from particles.
4. Max image intensity occurs when the most particles have collected on the IVC filter.
5. Illumination source and surrounding light is constant.
6. All particles fluoresce the same amount.

4.2.3 Setup and Procedure

The RGB images are captured with the high resolution DSLR camera and converted to grayscale images in order to reduce computation time. A mask is applied during post-processing in order to minimize intensity due to reflection and out-of-focus particles. The mask traces the shape of the IVC filter.

Figure 4.8: Original RGB image (left) and masked image showing only processed area (right).

Excitation of the fluorescent magnetic particles is provided by a 300mW continuous 532nm laser. The laser beam is pointed at a twenty to forty degree angle to the axis of the flow loop in order to illuminate the entire filter. The laser beam is expanded to the height of the flow chamber using cylindrical lens, which means it is no longer collimated light. Timing for the laser and camera is not required because of the continuous laser. In order to have consistent image spacing, the camera was automatically triggered using an Agilent 33220A function generator and a relay. Matlab is used to set the function generator parameters. The camera timing procedure is outlined below.

1. Download driver and interface program from Agilent.

Driver :agilent_33220A.mdd

Program: IOLibSuite_16_2_15823.exe

2. Install and Run Lib_Suite file
3. Connect the Agilent device to the computer, open Matlab, type *tmttool* in the command window, connect to the Agilent device.
4. Write an m-file to control the function generator. An example is shown below that turns the function generator on (signal = 1), pauses sixty seconds, and turns it off.


```
getsig = AgilentSig(frequency,Vpp,Voffset,1)
pause(60)
getsig = AgilentSig(frequency,Vpp,Voffset,0)
```
5. Strip the wires on a remote camera trigger for the Nikon D3100. There will be a white, red, and yellow wire.
6. Connect the function generator to the supply voltage side (+ and - terminals). Connect the white remote wire to one end of the relay switch (terminal 1). Connect the red and yellow remote wires to the other side of the relay switch (terminal 2). When the relay activates, the wires will be connected which emulates the push button switch on the remote trigger.
7. Set voltages, frequency, offset, etc. in Matlab and run the m-file.

4.2.4 Results

The normalized capture efficiency for each of the four scenarios is plotted in Figure 4.9. A series of sample images used to obtain this figure are included below, starting with Figure 4.10.

Figure 4.9: Capture efficiency over 6 hours using fluorescent images.

(4.10a) Image taken after 0 minutes.

(4.10b) Image taken after 15 minutes.

(4.10c) Image taken after 60 minutes.

(4.10d) Image taken after 120 minutes.

(4.10e) Image taken after 240 minutes.

(4.10f) Image taken after 360 minutes.

Figure 4.10: Time lapse images for YFYM scenario with magnetic particles over six hours.

(4.11a) Image taken after 0 minutes.

(4.11b) Image taken after 15 minutes.

(4.11c) Image taken after 60 minutes.

(4.11d) Image taken after 120 minutes.

(4.11e) Image taken after 240 minutes.

(4.11f) Image taken after 360 minutes.

Figure 4.11: Time lapse images for YFNM scenario with magnetic particles over six hours.

(4.12a) Image taken after 0 minutes.

(4.12b) Image taken after 15 minutes.

(4.12c) Image taken after 60 minutes.

(4.12d) Image taken after 120 minutes.

(4.12e) Image taken after 240 minutes.

(4.12f) Image taken after 360 minutes.

Figure 4.12: Time lapse images for NFYM scenario with magnetic particles over six hours.

(4.13a) Image taken after 0 minutes.

(4.13b) Image taken after 15 minutes.

(4.13c) Image taken after 60 minutes.

(4.13d) Image taken after 120 minutes.

(4.13e) Image taken after 240 minutes.

(4.13f) Image taken after 360 minutes.

Figure 4.13: Time lapse images for NFNM scenario with magnetic particles over six hours.

(4.14a) Image taken after 0 minutes.

(4.14b) Image taken after 15 minutes.

(4.14c) Image taken after 60 minutes.

(4.14d) Image taken after 120 minutes.

(4.14e) Image taken after 240 minutes.

(4.14f) Image taken after 360 minutes.

Figure 4.14: Time lapse images for the two liters per minute YFYM scenario with magnetic particles over six hours.

4.3 Discussion

Two methods for determining capture efficiency of 8.34 micron magnetite coated fluorescent spheres have provided reasonable results. The digital camera provided less promising results than the spectrometer, however the algorithm is simply the summation of pixel intensity. An enhancement of this method should provide useful results since the camera was capable of yielding a well defined curve in the YFYM scenario.

Calibration

The experiments tested for calibration resulted in absorbance below the recommended operating range. Unfortunately, it appears that errors are compounded for the calibration, resulting in sub-par correlation to expected results. It is expected that the absorbance and concentration are linearly related according to the Beer-Lambert law, Equation 4.2. Calibration with a stagnant fluid would likely improve correlation. Preliminary data was collected with a stagnant fluid in a cuvette and matched expected values with an R^2 correlation value near 0.99. This is shown in Figure 4.15.

Figure 4.15: Preliminary concentration calibration experiment with stagnant fluid in four identical cuvettes.

However, stagnant absorbance measurements may not relate directly to measurements in a moving fluid. This can be a consequence of using a cuvette versus flow loop which means different lengths of fluid processed, different wall materials, position of fiber optic cables, and possibly other variables.

Spectrometer

Although the spectrometer is considered to be more accurate, caution should be used when analyzing the data. As shown in the spectrometer calibration, there is a large potential for inconsistencies in the data. Several variables for inflow spectrometer absorbance measurements can lead to noticeable inaccuracies in the data. Firstly, the light passes through a large cross section of fluid in which particles are assumed to be perfectly dispersed. It is possible for a majority of particles to go above or below the light beam depending on density, leading to a lower absorbance than expected. Secondly, particles can get stuck somewhere in the flow loop from a previous experiment and released later, resulting in an increased absorbance. Thirdly, the pipetter may contain particles from a previous experiment when injecting into a new experiment. Lastly, there is inherent error in the spectrometer at low absorbance measurements due to sensitivity of the ccd array. Assuming the calibration was correct, the actual spectrometer capture efficiency data appears extremely promising. At a slow flow rate, the magnetized IVC filter is capable of capturing most of the particles. For the 2 L/min case, it was visually observed that the particles attach to the filter momentarily and slide along the filter before detaching. This is an important observation which may be useful for clinical treatment. Since the particles can be slowed down momentarily, it may be possible to target the cancer cells while they are attached to the filter.

Fluorescent Imaging

From a qualitative standpoint, fluorescent imaging provides decent visual results. It is apparent that particles begin to attract to the IVC filter almost immediately in the YFYM scenario. This is not as apparent in any of the other three scenarios. The other scenarios did not have a well defined trend using the camera. This is expected because there shouldn't be any particles if they are truly not being captured. However, the shape of the curve for the YFYM scenario does not match the one obtained with the spectrometer. A higher resolution image with a camera focused on a planar laser sheet will likely improve this method. As for now, this method can only reliably provide qualitative results.

Chapter 5

Cancer Cell Experiments

5.1 Cancer cell capture efficiency based on spectrometer

5.1.1 Overview

Following method developments with spherical particles, tagged cancer cells are run through the same flow loop that was previously discussed in Chapter 4. A visual representation of a tagged cell is shown in Figure 1.3. One important difference for testing particles versus cancer cells was the cells required strict protocols due to the health dangers such as bloodborne pathogens including Hepatitis B. Fortunately, Karen Rosell at the University of Rochester Medical Center provided cultured cells for testing. This allowed focus to be placed on experiments rather than training on cell culturing. Bishoy Faltas, M.D. from Rochester General Hospital assisted in the cell tagging portion of experiments. The experiments were conducted twice but with the scenario sequence re-arranged the second time in order to determine dependency on the scenario order. () for the magnetized IVC filter was continuously monitored over four hours for the first data set and two hours for the second data set. The cells had a twenty-four hour life expectancy which limits the duration of experimental testing. The fluid media was phosphate buffered saline (PBS) and the cells were cultured from an epithelial PS3 cell line. EasySep magnetic nano-particles were attached to the cells using EpCAM positive cell selection cocktail based on the protocol defined by Stemcell Inc. [7].

5.1.2 Equations and Assumptions

The same spectrometer absorbance equations used for paramagnetic spheres in Section 4.1.3 were used for the cancer cells. The only additional equations are related to mixture concentrations. The cell suspension was comprised of cancer cells, positive selection cocktail, and magnetic nano-particles. Adding the PBS media to the cell suspension yields what is referred to as the cell mixture. Through a preliminary dilution experiment, the experimental setup was capable of detecting mixture concentrations (C_{mix}) as low as $1.5e4 \text{ cells/ml}_{mix}$. Experimental mixture concentration was initially set at a factor of about seven above this, at $1e5 \text{ cells/ml}_{mix}$. The Stemcell protocol recommends a cell suspension concentration (C_{susp}) of $1e8 \text{ cells/ml}_{susp}$ in order to achieve properly tagged cells. The final C_{susp} is comprised of 100 μL cocktail, 50 μL magnetic nano-particles, and 1 mL of cells at a concentration of $1e8 \text{ cells/mL}$. Therefore, the suspension volume (V_{susp}) that must be added to the 500 mL of PBS is determined to be about 0.5 ml_{susp} from Equation 5.1.

$$V_{susp} = \frac{C_{mix}}{C_{susp}} V_{mix} \quad (5.1)$$

Several assumptions are made in order to process the data.

1. The particles have consistent properties and are perfectly distributed throughout the fluid.
2. The reference and dark spectrum do not change once calibrated.
3. The same spectrometer absorbance correlates to the same concentration of mixture
4. Magnetic nano-particles are attached to the cells
5. The concentration of mixture remains above the spectrometer's minimum limit of $1.5e4 \text{ cells/mL}$

5.1.3 Results

Nearly 1200 intensity data sets were collected on the spectrometer's CCD sensor for each scenario. This first set of experimental data was collected in the order NFNM, NFYM,

YFNM, YFYM. There were thousands of data points contained in each set of data because it captures intensity at many points across the visible spectrum, resulting in millions of data points for each scenario. This provides very precise results due to the large data set however the accuracy is at the mercy of experimental errors. The raw intensity values for each scenario are plotted in Figure A.5 along with the corresponding absorbance values in Figure A.6, located in Appendix A. Absorbance data is used to calculate the capture efficiency. By assuming full concentration, or no particles captured at time zero, the absorbance is converted to a relative percentage of cells captured over time. The data is reduced by integrating all absorbance values from 450 to 700nm, reducing hundreds of data points into a single data point.

Figure 5.1: Spectrometer capture efficiency for first set of cancer cell data over two hours. The four scenarios are tested in the sequence NFNM, NFYM, YFNM, YFYM.

The same experiment is repeated to determine if the order of scenarios is significant.

This second set of data was conducted over four hours for each scenario instead of two hours as shown above. Results for the YFYM, YFNM, NFNM, NFYM sequence are shown below.

Figure 5.2: Spectrometer capture efficiency for second set of cancer cell data over four hours. The four scenarios are tested in the sequence YFYM, YFNM, NFNM, NFYM.

It is apparent that the results are not consistent. The YFYM scenario is very similar for both sets of data. However, the control experiments do not align well. In fact, the highest capture efficiency for the second set of data is attributed to the NFNM scenario, which is hypothesized to be the smallest contributor to η_{capt} .

5.2 Cancer cell capture efficiency based on fluorescent images

5.2.1 Overview

The same camera setup used for the magnetic particle experiments in Section 4.2 was used again. However, for the first data set, a laser is not used and the second time a continuous laser is introduced. Expectations of this as a viable method for determining are very low since the consistent magnetic spheres had less than desirable results in Section 4.2.

5.2.2 Equations and Assumptions

The image intensity is simply a sum of the pixels after setting the masked areas to a value of zero. Equation 4.7 was developed for processing the magnetic particle images and it is also used for the cancer cell images.

5.2.3 Setup and Procedure

Setup for the cancer cells is very similar to the magnetic particle methods used in Section 4.2. The main difference is that the cells are not fluorescent. Consequently, the laser was not used for the first set of data. After looking at the images, it is apparent that the image resolution is not high enough to distinguish the cells. The laser was re-implemented for the second set of data. The optical filter is not used to filter out the 532nm wavelength light because the cells do not emit a noticeably different wavelength once excited. However, the laser light does scatter off the cells which is detected by the camera.

Figure 5.3: Original YFYM RGB image (left) and masked image with processed area (right) for the first set of data.

Figure 5.4: Original YFNM RGB image (left) and masked image with processed area (right) for the first set of data.

Figure 5.5: Original NFNm RGB image (left) and masked image with processed area (right) for the first set of data.

Figure 5.6: Original NFYM RGB image (left) and masked image with processed area (right) for the first set of data.

Figure 5.7: Original YFYM RGB image (left) and masked image with processed area (right) for the second set of data.

Figure 5.8: Original YFNM RGB image (left) and masked image with processed area (right) for the second set of data.

Figure 5.9: Original NFNM RGB image (left) and masked image with processed area (right) for the second set of data.

Figure 5.10: Original NFYM RGB image (left) and masked image with processed area (right) for the second set of data.

5.2.4 Results

Time lapse images for the first cancer cell data set are shown below. It does not appear that particles are stuck to the filter or cylinder wall for any of the four scenarios.

(5.11a) Image taken after 3 minutes.

(5.11b) Image taken after 15 minutes.

(5.11c) Image taken after 30 minutes.

(5.11d) Image taken after 60 minutes.

(5.11e) Image taken after 90 minutes.

(5.11f) Image taken after 120 minutes.

Figure 5.11: Time lapse images for YFYM scenario with first set of cancer cells over two hours.

(5.12a) Image taken after 0 minutes.

(5.12b) Image taken after 15 minutes.

(5.12c) Image taken after 30 minutes.

(5.12d) Image taken after 60 minutes.

(5.12e) Image taken after 90 minutes.

(5.12f) Image taken after 120 minutes.

Figure 5.12: Time lapse images for YFNM scenario with first set of cancer cells over two hours.

(5.13a) Image taken after 0 minutes.

(5.13b) Image taken after 15 minutes.

(5.13c) Image taken after 30 minutes.

(5.13d) Image taken after 60 minutes.

(5.13e) Image taken after 90 minutes.

(5.13f) Image taken after 120 minutes.

Figure 5.13: Time lapse images for NFYM scenario with first set of cancer cells over two hours.

(5.14a) Image taken after 0 minutes.

(5.14b) Image taken after 15 minutes.

(5.14c) Image taken after 30 minutes.

(5.14d) Image taken after 60 minutes.

(5.14e) Image taken after 90 minutes.

(5.14f) Image taken after 120 minutes.

Figure 5.14: Time lapse images for NFNM scenario with first set of cancer cells over two hours.

The time lapse images for the second cancer cell experiment are shown below. The continuous laser is green in color and helps illuminate the small cancer cells.

(5.15a) Image taken after 0 minutes.

(5.15b) Image taken after 15 minutes.

(5.15c) Image taken after 30 minutes.

(5.15d) Image taken after 60 minutes.

(5.15e) Image taken after 120 minutes.

(5.15f) Image taken after 240 minutes.

Figure 5.15: Time lapse images for YFYM scenario with second set of cancer cells over four hours.

(5.16a) Image taken after 0 minutes.

(5.16b) Image taken after 15 minutes.

(5.16c) Image taken after 30 minutes.

(5.16d) Image taken after 60 minutes.

(5.16e) Image taken after 120 minutes.

(5.16f) Image taken after 240 minutes.

Figure 5.16: Time lapse images for YFNM scenario with second set of cancer cells over four hours.

(5.17a) Image taken after 0 minutes.

(5.17b) Image taken after 15 minutes.

(5.17c) Image taken after 30 minutes.

(5.17d) Image taken after 60 minutes.

(5.17e) Image taken after 120 minutes.

(5.17f) Image taken after 240 minutes.

Figure 5.17: Time lapse images for NFYM scenario with second set of cancer cells over four hours.

(5.18a) Image taken after 0 minutes.

(5.18b) Image taken after 15 minutes.

(5.18c) Image taken after 30 minutes.

(5.18d) Image taken after 60 minutes.

(5.18e) Image taken after 120 minutes.

(5.18f) Image taken after 240 minutes.

Figure 5.18: Time lapse images for NFNM scenario with second set of cancer cells over four hours.

Normalized capture efficiency for each of the four scenarios is plotted in Figure 5.19.

(5.19a) Normalized image intensity for the first set of cancer cell data.

(5.19b) Normalized fluorescent image intensity for the second set of cancer cell data.

Figure 5.19: Capture efficiency based on normalized camera images taken every thirty seconds converted to grayscale. Both experiments use the same setup except the second set is fluoresced with a continuous laser.

5.3 Discussion

The first set of cancer cell data shows a noticeable increase in η_{capt} when the filter and magnet are present. A very similar η_{capt} is demonstrated in the second set of data but the control experiments were much different. There are several variables that can not be controlled which could cause this. The cells were carefully tagged with magnetic particles using the manufacturers protocol. It is possible that the first set of cancer cells were more receptive to the magnetic nano-particles. Other possible causes include the order of experiments, not flushing the flow chamber effectively between scenarios, spectrometer light and dark calibration errors, different concentration of cells between both data sets, and pipette errors. These may have contributed to the unexpected fluorescent image η_{capt} results in Figure 5.19b. The likely reason for the decrease rather than increase in η_{capt} is attributed to cells sticking to the walls and obstructing the laser. Consequently, the overall light intensity will decrease which is detected by the camera. Although the results are not consistent,

two methods have been developed for determining η_{capt} of cancer cells in real-time. It is clear that the spectrometer is capable of distinguishing small changes in cell concentration. Fluorescent imaging is also feasible if a more complicated image processing algorithm is used on images that have well defined particles. The main lesson learned from the cancer cell experiment is the flow loop needs to be more consistent and repeatable including the spectrometer. It is hard enough to control the organic variables without having errors introduced due to the experimental setup and procedure. The filter should be implantable without disconnecting the flow loop in order to prevent contaminating or disturbing the fluid. In addition, the spectrometer section should have a much smaller volume, possibly on the micro scale. It is advisable to use an automated particle injection mechanism in order to prevent pipetting errors. A larger camera lens or microscope should also be implemented with a laser sheet on the cells.

Chapter 6

Conclusion and Discussion

6.1 Overview

All objectives set in Section 1.2 have been considered in this work. The η_{capt} of a magnetizable IVC filter is determined theoretically using commercially available software as well as experimentally using a spectrometer and digital camera. Both 8.34 micron spheres and magnetically tagged cancer cells are analyzed. As shown previously, the spectrometer appears to be the more precise method for experimentally determining η_{capt} . The theoretical model predicts spherical particle η_{capt} to be about eight percent for a single pass. This correlates to about 2.5 minutes in the flow loop. Experimental results show only three percent capture efficiency over a 2.5 minute period for the particles. This discrepancy is partly due to the simplified magnetostatic model used for simulation and partly due to the inaccuracies of the spectrometer. The model assumes the sphere is completely made out of magnetite while the actual particles were only surface coated with magnetite. In addition, only twenty-six particles were tracked numerically whereas thousands of particles were used in physical experiments.

The capture efficiency for the cancer cells yielded mixed results. On the first run it appeared that the cells were sticking to the magnetized filter in the YFYM scenario much more than any other scenario. However, the same experiment was tested again in a different order. This time the capture efficiency could not be attributed to the magnetized filter any more than the control scenarios. Considering the promising results using magnetite spheres, it is likely that the cancer cell tagging method was not well executed.

Proper cell tagging and slightly improved methods should provide reasonable results.

The analytical model that has been developed herein can be used to optimize the filter geometry and magnet orientation. Combining the optimized model with better cancer cell tagging should prove to be a viable option of treating cancer patients.

6.2 Future Work

1. Experimental improvements

- The camera should be perpendicular to a laser sheet obtained from a spherical and cylindrical lens. The laser sheet should be traversed on a motorized linear mount while keeping the camera in focus. This will accent the particles without losing the ability to assess the entire filter.
- The spectrometer could use fluorescent detection rather than white light. This may provide more accurate results.
- Shorten the flow loop and possibly consider a single pass setup rather than continuous flow loop for analytical and experimental comparisons.
- Improve the camera η_{capt} algorithm to include a search for particles rather than simply measuring intensity.

2. Analytical improvements

- Add acceleration term to the ODE45 solver.
- Use a non-linear magnetostatic model.
- Optimize filter geometry and magnet orientation.
- Implement a more robust method for determining the gradient of the magnetic field intensity.

3. Design improvements

- Optimize filter geometry and location.
- Optimize magnet orientation and strength.
- Consider clinical use such as a vest with magnets.

Bibliography

- [1] *Physical Properties of Glycerine and Its Solutions*. New York: Glycerine Producer's Association, 1963. http://www.aciscience.org/docs/Physical_properties_of_glycerine_and_its_solutions.pdf.
- [2] J. E. Talmadge and I. J. Fidler, "The biology of cancer metastasis: Historical perspective," *American Association for Cancer Research*, vol. Cancer Res 2010, pp. 5649–5670, July 2010.
- [3] K. Weichman and J. E. Ansell, "Inferior vena cava filters in venous thromboembolism," *Progress in cardiovascular diseases*, vol. 49, no. 2, pp. 98–105, 2006.
- [4] RadiologyInfo.org, "Inferior Vena Cava Filter Placement and Removal," 2011. <http://www.radiologyinfo.org/>.
- [5] P. T. H. Went, A. Lugli, S. Meier, M. Bundi, M. Mirlacher, G. Sauter, and S. Dirnhofer, "Frequent EpCam Protein Expression in Human Carcinomas," *Human Pathology*, pp. 122–128.
- [6] Pankhurst, Q A and Connolly, J and Jones, S K and Dobson, J, "Applications of magnetic nanoparticles in biomedicine," vol. 167, 2003.
- [7] Stemcell, "Human EpCAM Positive Selection Kit," vol. 3.1.2, pp. 1–2, Mar. 2009. http://www.stemcell.com/~media/TechnicalResources/2/28860PIS_3_1_2.ashx.
- [8] M. Aviles, H. Chen, A. Ebner, A. Rosengart, M. Kaminski, and J. Ritter, "In vitro study of ferromagnetic stents for implant assisted-magnetic drug targeting," *Journal of Magnetism and Magnetic Materials*, vol. 311, pp. 306–311, Apr. 2007.
- [9] H. Chen, A. Ebner, M. Kaminski, A. Rosengart, and J. Ritter, "Analysis of magnetic drug carrier particle capture by a magnetizable intravascular stent: 2 Parametric study with multi-wire two-dimensional model," *Journal of Magnetism and Magnetic Materials*, vol. 293, pp. 616–632, May 2005.
- [10] H. Chen, A. Ebner, A. Rosengart, M. Kaminski, and J. Ritter, "Analysis of magnetic drug carrier particle capture by a magnetizable intravascular stent: 1 Parametric study

- with single wire correlation,” *Journal of Magnetism and Magnetic Materials*, vol. 284, pp. 181–194, Dec. 2004.
- [11] Z. G. Forbes, B. B. Yellen, D. S. Halverson, G. Fridman, K. a. Barbee, and G. Friedman, “Validation of high gradient magnetic field based drug delivery to magnetizable implants under flow,” *IEEE transactions on bio-medical engineering*, vol. 55, pp. 643–9, Feb. 2008.
 - [12] K. E. Scarberry, E. B. Dickerson, J. F. McDonald, and Z. J. Zhang, “Magnetic nanoparticle-peptide conjugates for in vitro and in vivo targeting and extraction of cancer cells,” *Journal of the American Chemical Society*, vol. 130, pp. 10258–62, Aug. 2008.
 - [13] J. W. Haverkort, S. Kenjeres, and C. R. Kleijn, “Computational simulations of magnetic particle capture in arterial flows,” *Annals of biomedical engineering*, vol. 37, pp. 2436–48, Dec. 2009.
 - [14] J. Ritter, “Application of high gradient magnetic separation principles to magnetic drug targeting,” *Journal of Magnetism and Magnetic Materials*, vol. 280, pp. 184–201, Sept. 2004.
 - [15] E. Furlani, “A model for predicting magnetic targeting of multifunctional particles in the microvasculature,” *Journal of Magnetism and Magnetic Materials*, vol. 312, pp. 187–193, May 2007.
 - [16] D. K. Cheng, *Field and Wave Electromagnetics*. Addison-Wesley, second ed., 1990.
 - [17] R. W. Fox, P. J. Pritchard, and A. T. McDonald, *Introduction to Fluid Mechanics*. Wiley, seventh ed., 2009.
 - [18] I. M. Kundu, Pijush K.; Cohen, *Fluid Mechanics*. Elsevier, fourth ed., 2008.
 - [19] K&J Magnetics, Inc., “Neodymium magnet physical properties,” Nov 2012. <http://www.kjmagnetics.com/specs.asp>.
 - [20] Ocean Optics, Inc., “Spectrasuite® spectrometer operating software.” Document Number 000-20000-300-02-201110. <http://www.oceanoptics.com/technical/SpectraSuite.pdf>.

Appendices

Appendix A

Supplemental Spectrometer Plots

Figure A.1: Spectrometer intensity with magnetic particles for all four scenarios at 0.3 L/min.

Figure A.2: Spectrometer absorbance with magnetic particles for all four scenarios at 0.3 L/min.

Figure A.3: Spectrometer intensity with magnetic particles for YFYM scenario at 0.3 L/min and 2 L/min.

Figure A.4: Spectrometer absorbance with magnetic particles for YFYM scenario at 0.3 L/min and 2 L/min.

Figure A.5: Spectrometer intensity for first set of cancer cell data over two hours. The four scenarios are tested in the sequence NFNM, NFYM, YFNM, YFYM.

Intensity data is converted to absorbance using the calibrated light and dark spectrum and Equation 4.1. The absorbance plot is shown in Figure A.6 below.

Figure A.6: Spectrometer absorbance for first set of cancer cell data over two hours. The four scenarios are tested in the sequence NFNM, NFYM, YFNM, YFYM.

Figure A.7: Spectrometer intensity for second set of cancer cell data over four hours. The four scenarios are tested in the sequence YFYM, YFNM, NFNM, NFYM.

Figure A.8: Spectrometer absorbance for second set of cancer cell data over four hours. The four scenarios are tested in the sequence YFYM, YFNM, NFNM, NFYM.

Appendix B

Matlab Particle Tracking Code

Figure B.1: Matlab particle tracking algorithm file structure. Green items are GUI's containing an m-file and fig file and blue items are function files.

B.1 Particle Tracking Main GUI

This code corresponds to a .fig file that is created using Matlab's GUIDE tool. It is included because there are several calculations as well as calls to other functions and GUI's.

```

function varargout = Particle_Tracking_Main(varargin)
% PARTICLE_TRACKING_MAIN M-file for Particle_Tracking_Main.fig
%     PARTICLE_TRACKING_MAIN, by itself, creates a new
%     PARTICLE_TRACKING_MAIN or raises the existing singleton*.
%
%     H = PARTICLE_TRACKING_MAIN returns the handle to a new
%     PARTICLE_TRACKING_MAIN or the handle to the existing singleton*.
%
%     PARTICLE_TRACKING_MAIN('CALLBACK',hObject,eventData,handles,...)
%     calls the local function named CALLBACK in PARTICLE_TRACKING_MAIN.M
%     with the given input arguments.
%
%     PARTICLE_TRACKING_MAIN('Property','Value',...) creates a new
%     PARTICLE_TRACKING_MAIN or raises the existing singleton*. Starting
%     from the left, property value pairs are applied to the GUI before

```

```

% Particle_Tracking_Main_OpeningFcn gets called. An unrecognized
% property name or invalid value makes property application stop.
% All inputs are passed to Particle_Tracking_Main_OpeningFcn via
% varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Particle_Tracking_Main

% Last Modified by GUIDE v2.5 12-Nov-2012 16:37:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',   @Particle_Tracking_Main_OpeningFcn, ...
                  'gui_OutputFcn',    @Particle_Tracking_Main_OutputFcn, ...
                  'gui_LayoutFcn',    [] , ...
                  'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Particle_Tracking_Main is made visible.
function Particle_Tracking_Main_OpeningFcn(hObject, eventdata, handles,...
    varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Particle_Tracking_Main (see VARARGIN)

```

[illegible]

```

% --- Executes when user attempts to close figure.
function figure_CloseRequestFcn(hObject,eventdata,handles)
% Hint: delete(hObject) closes the figure
if isequal(get(hObject,'waitstatus'),'waiting')
    % The GUI is still in UIWAIT, use UIRESUME and return
    uiresume(hObject);
else
    % The GUI is no longer waiting, so destroy it now.
    delete(hObject);
end

% --- Executes on button press clc.
function clcbtn_Callback(hObject, eventdata, handles)
evalin('base','clc')

% --- Executes on button press close.
function closebtn_Callback(hObject, eventdata, handles)
evalin('base','close')

% --- Executes on button press in clear.
function clearbtn_Callback(hObject, eventdata, handles)
evalin('base','clearvars -except fl mg fm postp')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Fluid_Callback(hObject, eventdata, handles)
flstr = 'fl';
assignin('base','flstr',flstr)
exst = evalin('base','exist(flstr)');
if exst ==1
    fl = evalin('base','fl');
end

mainnm = 'Particle_Tracking_Main'; % Name of main GUI
Particle_Tracking_Fluid(mainnm, handles.figure1);

```



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function IC_Callback(hObject, eventdata, handles)
```

```
% Set the initial position and velocities
```

```
mainnm = 'Particle_Tracking_Main'; % Name of main GUI
```

```
Particle_Tracking_ICs(mainnm,handles.figure1)
```

```
function solverparam_Callback(hObject, eventdata, handles)
```

```
% Set the solver parameters such as relative tolerance, max step size, etc.
```

```
mainnm = 'Particle_Tracking_Main'; % Name of main GUI
```

```
Particle_Tracking_SP(mainnm,handles.figure1)
```

```
function trackmult_Callback(hObject, eventdata, handles)
```

```
% Track multiple particle settings
```

```
set(handles.tracksing, 'Value', 0);
```

```
set(handles.trackmult, 'Value', 1);
```

```
set(handles.uipanel4, 'Visible', 'on');
```

```
function tracksing_Callback(hObject, eventdata, handles)
```

```
% Track single particle settings
```

```
set(handles.trackmult, 'Value', 0);
```

```
set(handles.tracksing, 'Value', 1);
```

```
set(handles.uipanel4, 'Visible', 'off');
```

```
% --- Executes on button press in verify.
```

```
function verify_Callback(hObject, eventdata, handles)
```

```
fm = evalin('base', 'fm');
```

```
mg = evalin('base', 'mg');
```

```
fl = evalin('base', 'fl');
```

```
chk{1,1} = 'maxiter'; % choose any variable set in SP
```

```
    chk{1,2} = 'Solver Parameters';
```

```
    chk{1,3} = 'fm';
```

```
chk{2,1} = 'x0'; % choose any variable set IC
```

```
    chk{2,2} = 'Initial Conditions';
```

```
    chk{1,3} = 'fm';
```

```
chk{3,1} = 'rho'; % choose any variable set in main
```

```
    chk{3,2} = 'Fluid Properties (main)';
```

```
    chk{1,3} = 'fl';
```

```
chk{4,1} = 'Fvx'; % choose any variable set in Import FL
```

```
    chk{4,2} = 'Import Fluid Data (button)';
```


[illegible]

```

function solveptrack_Callback(hObject, eventdata, handles)
% Make sure variables are stored
fl = evalin('base','fl');
mg = evalin('base','mg');
fm = evalin('base','fm');

mg.sf = str2num(get(handles.sf_m,'string'));
mg.u0 = str2num(get(handles.mu0,'string'));
mg.Nmp = str2num(get(handles.Nmp,'string'));
mg.Dmp = 10^(-6)*str2num(get(handles.Dmp,'string'));
mg.ump = str2num(get(handles.mump,'string'));
mg.rho = str2num(get(handles.rhomp,'string'));
mg.Hsat = str2num(get(handles.hsat,'string'));
mg.Msat = mg.Hsat*(mg.ump/mg.u0-1);
mg.Vmp = 4/3*pi*(mg.Dmp/2)^3;
mg.mmp = mg.rho*mg.Vmp;

fl.rho = str2num(get(handles.rho,'string'));
fl.dia = str2num(get(handles.tdia,'string'));
fl.mu = str2num(get(handles.mu,'string'));
fl.A = pi*(fl.dia/2)^2;

ptyp = get(handles.trackmult,'Value');
varargin{1} = fm.solvertyp;% ode solver type
varargin{2} = fm.outtype; % ode plot type

%%%%%%%%%%%%%%
% Track single particle
%%%%%%%%%%%%%%
figure(1)
pause(0.0001)% let program think...
if ptyp == 0
[mg.Fhx mg.Fhy mg.Fhz fm.xtrack fm.ytrack fm.ztrack tm fm.soln pcapt...
 pout plim] = fl_mg_soln_new(fl,mg,fm,[fm.x0,fm.y0,fm.z0],varargin);

fm.particles_in = 1;
fm.particles_capt = pcapt;
fm.particles_out=pout;
fm.particles_lim = plim;
fm.particles_nan = 1-pout-pcapt-plim;

fprintf('\n\n -----Particle Tracking Results ----- \n')

```

```

fprintf('%3.0f particles in\n',fm.particles_in)
fprintf('%3.0f particles captured\n',sum(fm.particles_capt))
fprintf('%3.0f particles escaped through the outlet\n',...
        sum(fm.particles_out))
fprintf('%3.0f particles resulted in NaN\n',sum(fm.particles_nan))
fprintf('%3.0f particles reached time/iteration limit\n',...
        sum(fm.particles_lim))
ptracknum = str2num(get(handles.ptrack,'string'));

fm.ptrack{1,ptracknum} = [fm.xtrack, fm.ytrack, fm.ztrack];
fm.T{1,ptracknum} = tm;
fm.MagForce{1,ptracknum} = fm.soln.MagForce;
fm.DragForce{1,ptracknum} = fm.soln.DragForce;
InertFx = mg.mmp*diff(fm.soln.vel(:,1))/diff(tm);
InertFy = mg.mmp*diff(fm.soln.vel(:,2))/diff(tm);
InertFz = mg.mmp*diff(fm.soln.vel(:,3))/diff(tm);
fm.InertForce{1,ptracknum} = [InertFx, InertFy, InertFz];
set(handles.ptrack,'string',num2str(ptracknum+1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Track multiple particles
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

elseif ptyp ==1
%Initial conditions
atx = max(fl.xbnd)-fl.dia/2;% Auto Track - X start position
% aty = max(fl.ybnd)-fl.dia;% Auto Track - Y start position
atz = 0;% Auto Track - Z start position
numpart = str2num(get(handles.numpart,'string'));
ptracknum = str2num(get(handles.ptrack,'string'));
fm.particles_in = numpart;
aty = str2num(get(handles.mult_init_y,'string'));
for kk = ptracknum:numpart+ptracknum-1
    figure(1)
    if kk==1
        fm.x0 = atx;
        fm.y0 = aty;
        fm.z0 = atz;
    elseif kk>1 && kk<=5
        theta = 90*(6-kk)-45;
        fm.x0 = atx+(fl.dia/4)*cosd(theta);
        fm.y0 = aty;
        fm.z0 = (fl.dia/4)*sind(theta);
    end
end

```

```

elseif kk>5 && kk<=9
    theta = 90*(10-kk);
    fm.x0 = atx+(fl.dia/8)*cosd(theta);
    fm.y0 = aty;
    fm.z0 = (fl.dia/8)*sind(theta);
elseif kk>9 && kk<=13
    theta = 90*(14-kk);
    fm.x0 = atx+(fl.dia/4)*cosd(theta);
    fm.y0 = aty;
    fm.z0 = (fl.dia/4)*sind(theta);
elseif kk>13 && kk<=17
    theta = 90*(18-kk)-45;
    fm.x0 = atx+(fl.dia/8)*cosd(theta);
    fm.y0 = aty;
    fm.z0 = (fl.dia/8)*sind(theta);
elseif kk>17 && kk<=21
    theta = 90*(22-kk)-45;
    fm.x0 = atx+(fl.dia/16)*cosd(theta);
    fm.y0 = aty;
    fm.z0 = (fl.dia/16)*sind(theta);
elseif kk>21 && kk<=25
    theta = 90*(26-kk);
    fm.x0 = atx+(fl.dia/16)*cosd(theta);
    fm.y0 = aty;
    fm.z0 = (fl.dia/16)*sind(theta);
else
    theta = 90*(30-kk)-45;
    fm.x0 = atx+(fl.dia/3)*cosd(theta);
    fm.y0 = aty;
    fm.z0 = (fl.dia/3)*sind(theta);
end

% Run simulation
[mg.Fhx mg.Fhy mg.Fhz fm.xtrack fm.ytrack fm.ztrack tm fm.soln{1, kk} ...
    pcapt pout plim] = fl_mg_soln_new(fl, mg, fm, [fm.x0, fm.y0, fm.z0], varargin);
fprintf('\n Particle %2.0f checked\n\n', kk)
fm.ptrack{1, kk} = [fm.xtrack, fm.ytrack, fm.ztrack];
fm.T{1, kk} = tm;
fm.MagForce{1, kk} = fm.soln.MagForce;
fm.DragForce{1, kk} = fm.soln.DragForce;
InertFx = mg.mmp*diff(fm.soln.vel(:, 1))/diff(tm);
InertFy = mg.mmp*diff(fm.soln.vel(:, 2))/diff(tm);
InertFz = mg.mmp*diff(fm.soln.vel(:, 3))/diff(tm);
fm.InertForce{1, kk} = [InertFx, InertFy, InertFz];

```

```

set(handles.ptrack,'string',num2str(kk+1));

fm.particles_capt(1,kk) = pcapt;
fm.particles_out(1,kk)=pout;
fm.particles_lim(1,kk) = plim;
fm.particles_nan(1,kk) = 1-pout-pcapt-plim;

fprintf('\n\n -----Particle Tracking Results ----- \n')
fprintf('%3.0f particles in\n',fm.particles_in)
fprintf('%3.0f particles captured\n',sum(fm.particles_capt))
fprintf('%3.0f particles escaped through the outlet\n',...
        sum(fm.particles_out))
fprintf('%3.0f particles resulted in NaN\n',sum(fm.particles_nan))
fprintf('%3.0f particles reached time/iteration limit\n',...
        sum(fm.particles_lim))
close figure 1
% Save results
saveas1 = get(handles.svm,'string');
fm.saveas = [saveas1,'_',num2str(kk)];
assignin('base','fm',fm)
evalin('base','save(fm.saveas)');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Didn't select particle tracking method %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else
    msgbox('Select single or multiple particle tracking.')
end
assignin('base','mg',mg)
assignin('base','fl',fl)
assignin('base','fm',fm)
evalin('base','save(fm.saveas)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Post Processing %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in SavePDF.

```

```

function SavePDF_Callback(hObject, eventdata, handles)
qual = num2str(get(handles.DPI, 'String'));
qualstr = ['-r',qual];
svnm = get(handles.SaveImAs, 'String');
savenm = svnm;
n = get(0,'children');
nprev = n(2);
set (figure(nprev), 'PaperOrientation','landscape','PaperUnits',...
    'normalized','PaperPosition',[0 0 1 1])
print (figure(nprev),'-dpdf',qualstr,savenm)

```

```

function plot_sel_Callback(hObject, eventdata, handles)
postp = evalin('base','postp');
% Set particle plot type if fm plot is selected
if get(handles.fmsol, 'Value')==1
pplot = get(handles.particles, 'Value');
    if pplot ==2
        postp.particles = 'all';
    elseif pplot ==3
        postp.particles = 'pcapt';
    elseif pplot ==4
        postp.particles = 'pout';
    elseif pplot ==5
        postp.particles = 'plim';
    elseif pplot ==6
        postp.particles = 'pnan';
    end
end
% Set plot view type
pv = get(handles.plotview, 'Value');
ps = get(handles.plotview, 'String');
postp.plotview = cell2mat(ps(pv));
assignin('base','postp',postp)
postproc(postp)% run custom plot function

```

```

% --- Executes on button press in cax.
function cax_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base','ppstr',ppstr)
exst = evalin('base','exist(ppstr)');
if exst ==1
    postp = evalin('base','postp');
end

```

```

val = get(handles.cax, 'Value');
postp.coord =val;% Zero coordinates 'cax'
assignin('base', 'postp', postp)

% --- Executes on button press in pin.
function pin_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base', 'ppstr', ppstr)
exst = evalin('base', 'exist(ppstr)');
if exst ==1
    postp = evalin('base', 'postp');
end
val = get(handles.pin, 'Value');
postp.inlet =val;% Start point 'pin'
assignin('base', 'postp', postp)

% --- Executes on button press in mvect.
function mvect_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base', 'ppstr', ppstr)
exst = evalin('base', 'exist(ppstr)');
if exst ==1
    postp = evalin('base', 'postp');
end
val = get(handles.mvect, 'Value');
postp.mgvect =val;% 'mvect'

if val ==1
    set(handles.zpln, 'Visible', 'on')
    set(handles.ypln, 'Visible', 'on')
    set(handles.mvectall, 'Visible', 'on')
    postp.vectstyle{1} = 'zpln';
else
    set(handles.zpln, 'Visible', 'off')
    set(handles.ypln, 'Visible', 'off')
    set(handles.mvectall, 'Visible', 'off')
end
assignin('base', 'postp', postp)

% --- Executes on button press in zpln.
function zpln_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base', 'ppstr', ppstr)

```

```

exst = evalin('base','exist(ppstr)');
if exst ==1
    postp = evalin('base','postp');
end
val = get(handles.zpln,'Value');
if val ==1
    postp.vectstyle{1} = 'zpln';
elseif val ==0
    postp.vectstyle{1} = 'none';
end
assignin('base','postp',postp)

% --- Executes on button press in ypln.
function ypln_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base','ppstr',ppstr)
exst = evalin('base','exist(ppstr)');
if exst ==1
    postp = evalin('base','postp');
end
val = get(handles.ypln,'Value');
if val ==1
    postp.vectstyle{2} = 'ypln';
elseif val ==0
    postp.vectstyle{2} = 'none';
end
assignin('base','postp',postp)

% --- Executes on button press in mvectall.
function mvectall_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base','ppstr',ppstr)
exst = evalin('base','exist(ppstr)');
if exst ==1
    postp = evalin('base','postp');
end
val = get(handles.mvectall,'Value');
if val ==1
    postp.vectstyle{3} = 'all';
elseif val ==0
    postp.vectstyle{3} = 'none';
end
assignin('base','postp',postp)

```



```

% --- Executes on button press in fmsol.
function fmsol_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base','ppstr',ppstr)
exst = evalin('base','exist(ppstr)');
if exst ==1
    postp = evalin('base','postp');
end
val = get(handles.fmsol,'Value');
postp.fm =val;% Fluid-Mag solution 'fmsol'
if val ==1
    set(handles.fm_an,'Visible','on')
    set(handles.fm_noan,'Visible','on')
    set(handles.particles,'Visible','on')
    postp.fm_noan =1;
else
    set(handles.fm_an,'Visible','off')
    set(handles.fm_noan,'Visible','off')
    set(handles.particles,'Visible','off')
end
assignin('base','postp',postp)

% --- Executes on button press in fm_an.
function fm_an_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base','ppstr',ppstr)
exst = evalin('base','exist(ppstr)');
if exst ==1
    postp = evalin('base','postp');
end
val = get(handles.fm_an,'Value');
if val ==1
    set(handles.fm_noan,'Value',0)
    postp.fm_noan =0;
end
postp.fm_an =val;% Fluid streamlines 'fstrm'
assignin('base','postp',postp)

% --- Executes on button press in fm_noan.
function fm_noan_Callback(hObject, eventdata, handles)
ppstr = 'postp';
assignin('base','ppstr',ppstr)

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% function plot_fm_an_Callback(hObject, eventdata, handles)
%
function ptrack_Callback(hObject, eventdata, handles)
function ptrack_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function load_fm_Callback(hObject, eventdata, handles)
function rho_Callback(hObject, eventdata, handles)
function rho_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function tdia_Callback(hObject, eventdata, handles)
function tdia_CreateFcn(hObject, eventdata, handles)

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function mu_Callback(hObject, eventdata, handles)
function mu_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Nmp_Callback(hObject, eventdata, handles)
function Nmp_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function mu0_Callback(hObject, eventdata, handles)
function mu0_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

end
function Dmp_Callback(hObject, eventdata, handles)
function Dmp_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function mump_Callback(hObject, eventdata, handles)
function mump_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function sf_m_Callback(hObject, eventdata, handles)
function sf_m_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function mult_init_y_Callback(hObject, eventdata, handles)
function mult_init_y_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function numpart_Callback(hObject, eventdata, handles)
function numpart_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function svm_Callback(hObject, eventdata, handles)
function svm_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function particles_Callback(hObject, eventdata, handles)
function particles_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

```

```
function plotview_Callback(hObject, eventdata, handles)
function plotview_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function rhomp_Callback(hObject, eventdata, handles)
function rhomp_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function hsat_Callback(hObject, eventdata, handles)
function hsat_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function SaveImAs_Callback(hObject, eventdata, handles)
function SaveImAs_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function DPI_Callback(hObject, eventdata, handles)
function DPI_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
    'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

B.2 ODE solver

This m-file is called to solve the particle tracking algorithm for a single particle. There are several nested functions that are used to solve the ODE.

```
function [Fhx Fhy Fhz xtrack ytrack ztrack T soln pcapl pout plim] =...  
    fl_mg_soln_new(fl,mg,fm,xyz0,varargin)  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
% Set constant values from inputs
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u0 = mg.u0; % Permeability of free space
ump = mg.ump; % Permeability of free space
Nmp = mg.Nmp; % Number of magnetic particles per cell
Vmp = mg.Vmp; % Volume of magnetic particles
Tstop = fm.tstop;
reltol = fm.reltol;
abstol = fm.abstol;
init_t= cputime;
simtime = fm.simtime;
stopdlg = fm.stopdlg;
fcnt =fm.iterskip;
pcapt = 0;
pout = 0;
plim = 0;
limhit =0;
simokchk = 0;% check if the simulation is okay. 1=no
iplot = 1; % initialize 3D plot
xchk = max(fl.xbnd);% xlocation for exit
vmag = 100; % initial magnitude of velocity
styp = cell2mat(varargin{1}(1));
outtyp = cell2mat(varargin{1}(2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 1: Set start point and find smallest dx,dy,dz
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dxyz = fm.dinterp; % meters

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 2: Interpolate to get Fluid velocity, Vf(x,y,z)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Fvx = fl.Fvx;% TriScatteredInterp for velocity x
Fvy = fl.Fvy;% TriScatteredInterp for velocity y
Fvz = fl.Fvz;% TriScatteredInterp for velocity z

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 3: Interpolate to get Magnetic Field Intensity, H(x,y,z)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Fhx = mg.Fhx;% TriScatteredInterp for Hx
Fhy = mg.Fhy;% TriScatteredInterp for Hy
Fhz = mg.Fhz;% TriScatteredInterp for Hz

```

```

mag_const = mg.sf*(u0*Nmp*Vmp*3*(ump-u0)/(ump+2*u0));
fprintf('\nThe magnetic constant is %12.4e (H*m^3/kg)\n\n',mag_const)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 6: Find distance moved
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

qq = 1;
Tstart =0; % initial start time
tin = 0.0005; % initial step size
%Initial condition for x,y,z, coordinates at t = 0 is set in xyz0
%Calculate particle velocity using Matlab ode solver
tout = Tstart;
x0 = xyz0(1);
y0 = xyz0(2);
z0 = xyz0(3);
yout = [xyz0(1),xyz0(2),xyz0(3)];
% Initial velocities are zero
vx_i1 =0;
vy_i1 = 0;
vz_i1 = 0;

tic
for zz = 1:fm.loopstp
    options = odeset('RelTol',reltol,'AbsTol',abstol,'OutputFcn',@fcn1,...
        'MaxStep',fm.maxstep,'InitialStep',tin,'Events',(@event1));
    % Set ODE solver
    switch styp
        case {'ODE45','ode45'}
            [T,Y] = ode45(@ode_dist,[Tstart Tstop],xyz0,options);
        case {'ODE23','ode23'}
            [T,Y] = ode23(@ode_dist,[Tstart Tstop],xyz0,options);
        otherwise
            [T,Y] = ode45(@ode_dist,[Tstart Tstop],xyz0,options);
    end
    xtrack = Y(:,1); ytrack = Y(:,2); ztrack = Y(:,3);
    to{zz} = T;
    yo{zz} = Y;
    assignin('base','to',to)
    assignin('base','yo',yo)
    disp('ode refreshed')

```

```

% Accumulate output. This could be passed out as output arguments.
nt = length(T);
% Don't save output if time limit or iteration limit is hit
pause(0.001)
if limhit ==1;
    break;
else
    tout = [tout; T(2:nt)];
    yout = [yout; Y(2:nt,:)];

    % Set the new initial conditions
    R = fl.dia/2; % Radius of circle
    p0 = [Y(end-1,1),Y(end-1,2),Y(end-1,3)];
    p1 = [Y(end,1),Y(end,2),Y(end,3)];

    [xyz0,simokchk] = bounce_phys(R,p0,p1)
end

assignin('base','xtrack',xtrack)
assignin('base','ytrack',ytrack)
assignin('base','ztrack',ztrack)

end

toc

function dy = ode_dist(t,xyz)
xi = xyz(1); yi = xyz(2); zi = xyz(3);
xchk = xi;
vxi = Fvx(xi,yi,zi); % Fluid Velocity x
vyi = Fvy(xi,yi,zi); % Fluid Velocity y
vzi = Fvz(xi,yi,zi); % Fluid Velocity z
hxi = Fhx(xi,yi,zi); % Magnetic Field Intensity x
hyi = Fhy(xi,yi,zi); % Magnetic Field Intensity y
hzi = Fhz(xi,yi,zi); % Magnetic Field Intensity z

xi1 = xi-dxyz;%xi-dx because it should move in the -x direction
yi1 = yi-dxyz;%yi-dy because it should move in the -y direction
zi1 = zi+dxyz;%zi+dz

```



```

hxi1 = Fhx(xi1,yi,zi);% hx(x+dx)
hyi1 = Fhy(xi,yi1,zi);% hy(y+dy)
hzi1 = Fhz(xi,yi,zi1);% hz(z+dz)

gradHx = (hxi1-hxi)/(xi1-xi);% Magnetic Field Intensity x+dx
gradHy = (hyi1-hyi)/(yi1-yi);% Magnetic Field Intensity y+dy
gradHz = (hzi1-hzi)/(zi1-zi);% Magnetic Field Intensity z+dz
divH =abs(gradHx)+abs(gradHy)+abs(gradHz);

% Check for magnetic saturation
Ms = mg.Msat;
if abs(hxi)<(ump+2*u0)/(3*ump-3*u0)*Ms % Not Saturated
    % Force on particle due to magnet in x-dir
    f_mx = (mg.sf*u0*Nmp*mg.Vmp*divH*hxi)*3*(ump-u0)/(ump+2*u0);
else % Saturated
    f_mx = mg.sf*u0*Nmp*mg.Vmp*Ms*sign(hxi)*divH;
end

if abs(hyi)<(ump+2*u0)/(3*ump-3*u0)*Ms % Not Saturated
    % Force on particle due to magnet in y-dir
    f_my = (mg.sf*u0*Nmp*mg.Vmp*divH*hyi)*3*(ump-u0)/(ump+2*u0);
else % Saturated
    f_my = mg.sf*u0*Nmp*mg.Vmp*Ms*sign(hyi)*divH;
end

if abs(hzi)<(ump+2*u0)/(3*ump-3*u0)*Ms % Not Saturated
    % Force on particle due to magnet in z-dir
    f_mz = (mg.sf*u0*Nmp*mg.Vmp*divH*hzi)*3*(ump-u0)/(ump+2*u0);
else % Saturated
    f_mz = mg.sf*u0*Nmp*mg.Vmp*Ms*sign(hzi)*divH;
end

% Set current velocity of particle
vx_cur = vxi+f_mx/(3*pi*fl.mu*mg.Dmp);
vy_cur = vyi+f_my/(3*pi*fl.mu*mg.Dmp);
vz_cur = vzi+f_mz/(3*pi*fl.mu*mg.Dmp);

vmag = sqrt(vx_cur^2+vy_cur^2+vz_cur^2);

% Save current iteration results
soln.gradH(qq,1:3) = [gradHx,gradHy,gradHz];

```

```

soln.H(qq,1:3) = [hxi,hyi,hzi];
soln.H1(qq,1:3) = [hxi1,hyi1,hzi1];
soln.coord(qq,1:3) = [xi,yi,zi];
soln.coord1(qq,1:3) = [xi1,yi1,zi1];
soln.vx(qq,1:3) = [vxi,vyi,vzi];
soln.v_m(qq,1:3) = [vx_cur-vxi,vy_cur-vyi,vz_cur-vzi];
soln.vel(qq,1:3) = [vx_cur,vy_cur,vz_cur];
soln.MagForce(qq,1:3) = [f_mx,f_my,f_mz];
soln.DragForce(qq,1:3) = 3*pi*f1.mu*mg.Dmp*[vxi,vyi,vzi];
soln.t(qq) = t;

% Display results each specified iteration; i.e. every 10 iterations
if fcnt==qq || qq==1
    fprintf('grad(Hx) = %12.4e (A/m^2)\n',gradHx)
    fprintf('grad(Hy) = %12.4e (A/m^2)\n',gradHy)
    fprintf('grad(Hz) = %12.4e (A/m^2)\n',gradHx)
    fprintf('div(H) = %12.4e (A/m)\n',divH)

    fprintf('x(i) = %12.4e and x(i+1) = %12.4e (m)\n',xi,xi1)
    fprintf('y(i) = %12.4e and y(i+1) = %12.4e (m)\n',yi,yi1)
    fprintf('z(i) = %12.4e and z(i+1) = %12.4e (m)\n',zi,zi1)

    fprintf('Hx(i) = %12.4e and Hx(i+1) = %12.4e (A/m)\n',hxi,hxi1)
    fprintf('Hy(i) = %12.4e and Hy(i+1) = %12.4e (A/m)\n',hyi,hyi1)
    fprintf('Hz(i) = %12.4e and Hz(i+1) = %12.4e (A/m)\n',hzi,hzi1)

    fprintf('Fluid Velocity (x) = %12.4e (m/s)\n',vxi)
    fprintf('Fluid Velocity (y) = %12.4e (m/s)\n',vyi)
    fprintf('Fluid Velocity (z) = %12.4e (m/s)\n',vzi)

    fprintf('Velocity due to Magnet (x) = %12.4e (m/s)\n',vx_cur-vxi)
    fprintf('Velocity due to Magnet (y) = %12.4e (m/s)\n',vy_cur-vyi)
    fprintf('Velocity due to Magnet (z) = %12.4e (m/s)\n',vz_cur-vzi)

    fprintf('Particle Velocity (x) = %12.4e (m/s)\n',vx_cur)
    fprintf('Particle Velocity (y) = %12.4e (m/s)\n',vy_cur)
    fprintf('Particle Velocity (z) = %12.4e (m/s)\n',vz_cur)

    fprintf('\n Iteration %g \n\n',qq)

    fcnt = fcnt+fm.iterskip;
end

dy = zeros(3,1); % Column Vector

```

```

dy(1) = vx_cur; % Vmpx = dx/dt
dy(2) = vy_cur; % Vmpy = dy/dt
dy(3) = vz_cur; % Vmpz = dz/dt
if isnan(dy)
    disp('Your new velocity is NaN!!!')
    pause(15)
    simokchk = 1;
    t = Tstop
end
qq=qq+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 8: Determine if time/iteration limit has been reached and plot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function status = fcn1(T,Y,flag)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot current step based on plot type
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch outtyp
case 'odeplot'
    % ODE PLOT
    odeplot(T,Y,flag);
    legend('x_p_o_s','y_p_o_s','z_p_o_s')
    title('Position of Particle')
    xlabel('Time (s)')
    ylabel('Position (m)')
case '3D'
    % 3D PLOT
    if iplot ==1
        figure(1)
        plot3(fl.xbnd,fl.ybnd,fl.zbnd,'k.','MarkerSize',0.01)
        axis([-0.1 0.1 -0.1 0.1 -0.1 0.1])
        xlabel('x');ylabel('y');zlabel('z')
        hold on; axis square; zoom(2.5)
        iplot = 2;
    elseif iplot ==2 && isempty(Y)==0
        % Plot solution
        figure(1)
        plot3(Y(1),Y(2),Y(3),'bo','MarkerSize',4,'MarkerFaceColor','b')
    end
end

```

```

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine if limits have been reached
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Check to see if time limit has been reached
if cputime - init_t>=simtime
    status = 1; %
    disp('Time limit has been reached')
    limhit =1;
    plim = 1;
% Check to see if max iteration has been reached
elseif qq >=fm.maxiter
    status = 1;
    disp('Iteration limit has been reached')
    limhit =1;
    plim = 1;
% Check to see if user stopped simulation
elseif qq>=stopdlg
    ststr = 'Enter 1 if you would like to stop or 0 to continue'
    status = str2num(cell2mat(inputdlg(ststr)));
    if status ==1
        disp('Simulation terminated by user!')
        limhit =1;
    end
    stopdlg = stopdlg+fm.stopdlg;
    plim = 1;
% Check to see if particle exited through outlet
elseif xchk<=0.9*min(fl.xbnd)
    status =1;
    disp('Particle exited through the outlet!')
    limhit =1;
    pout = 1;
% Check to see if particle escaped boundaries
elseif simokchk ==1
    status =1;
    disp('Particle tracking failure, NaN result.')
    limhit =1;
% Check to see if particle velocity is less than threshold (captured)
elseif vmag<=fm.vthresh
    status =1;
    disp('Particle Captured!')

```

```

        limhit =1;
        pcapt = 1;
% Check to see if NaN solution occurred
        elseif isnan(Vmp(end,1))==1
            cntr =length(t);
            disp('Particle tracking failure, NaN result.')
            limhit =1;
% Otherwise continue
        else
            status=0;
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 9: Determine if particle has crossed boundary
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [stop,isterminal,direction] = event1(T,Y)
% Locate when the particle contacts the wall.
% End the integration for that step and move on.

% Check to see if the particle hits the wall
x = Y(1);
y = Y(2);
z = Y(3);
    cbmax = [max(fl.xbnd),max(fl.ybnd),max(fl.zbnd)];
    cbmin = [min(fl.xbnd),min(fl.ybnd),min(fl.zbnd)];
    dia = fl.dia;
x1 = cbmax(1)-dia/2;
x3 = cbmax(1)-dia;
y3 = cbmin(2)+dia;
x4 = cbmax(1);
y4 = cbmin(2);
m_cross = (y3-y4)/(x3-x4);%Slope where L shape meets w.r.t. btm rt corner
m_particle = (y-y4)/(x-x4);%Slope of current particle
if isnan(x) ==1
    stop =0;% Stops ODE when value crosses zero
    isterminal =1;% End current ode iteration
elseif m_particle<m_cross; % Less than because slope is negative
    if (x-x1)^2+z^2>(dia/2)^2 % Check to see if it is outside of the
        % xz-plane circle (upper cylinder)
        stop = 0;
        isterminal =1;
%
    Tstart = fm.tstop;

```

```

    else
        stop = 1;
        isterminal = 0;
    end
elseif m_particle >= m_cross;
    if (y)^2 + (z)^2 > (dia/2)^2 % Check to see if it is outside of the
        % yz-plane circle (lower cylinder)
        stop = 0;
        isterminal = 1;
        % Tstart = fm.tstop;
    else
        stop = 1;
        isterminal = 0;
    end
end
direction = 0;
fprintf('Boundary Checked, value = %2.0g \n', stop)
end

function [xyz0, simokchk] = bounce_phys(R, p0, p1)
    % Should pause here and plot the p0 and p1 points to make sure the
    % interpolation function is working properly...

    % Interpolate intersection

    a = p1(1) - p0(1);
    b = p1(2) - p0(2);
    c = p1(3) - p0(3);

    A = (b^2 + c^2) / a^2;
    B = 2 * p0(2) * b / a + 2 * p0(3) * c / a;
    C = p0(2)^2 + p0(3)^2 - R^2;
    if isfinite(A + B + C)
        simokchk = 0;

        r = roots([A B C]);

        % Solution (roots of quadratic eqn)
        valu1 = min(r) + p0(1); valu2 = max(r) + p0(1); % Parametric solution
        if abs(valu1) < abs(valu2)
            rt = valu1
        else
            rt = valu2
        end
    end
end

```

```

end

pn(1) = rt;
pn(2) = b/a*(pn(1)-p0(1))+p0(2);
pn(3) = c/a*(pn(1)-p0(1))+p0(3);
% set point to wall
xn = pn(1); yn = pn(2); zn = pn(3);

% Largest root
%   rtmx = max(r)+p0(1);
%   pnmx(1) = rtmx;
%   pnmx(2) = b/a*(pnmx(1)-p0(1))+p0(2);
%   pnmx(3) = c/a*(pnmx(1)-p0(1))+p0(3);

% plot3(xyz0(1),xyz0(2),xyz0(3),'mo')
% move point based on velocity
L2 = sqrt(a^2+b^2+c^2);
L3 = sqrt((pn(1)-p0(1))^2+(pn(2)-p0(2))^2+(pn(3)-p0(3))^2);
L4 = abs(L2-L3);
% Final bounce point
my = -a/b;% x-y slope for bounce line (perp. to p1 p0 line)
mz = -a/c;% x-z slope for bounce line

zeta = .5; % Reduction of distance due to collision w/ wall
Af = 1+my^2+mz^2;% Second order term
Bf = 0;% First order term
Cf = -zeta^2*L4^2;% Constant
alpha = roots([Af Bf Cf]);
xrt = real(min(alpha));

xyz0(1) = xrt+pn(1);% final x point
xyz0(2) = my*xrt+pn(2);% final y point
xyz0(3) = mz*xrt+pn(3);% final z point
% Plot point where particle jumped
plot3(xyz0(1),xyz0(2),xyz0(3),'ro','MarkerSize',8,'MarkerFaceColor','r')
if T(nt) ~= Tstop
    Tstart = T(nt);
    tin = T(nt)-T(nt-4)
end
else
    simokchk = 1;
end
end
end

```

```
end
```

B.3 Post-processing Function

This post-processing function uses inputs from the main GUI to plot a figure. There are several options available such as side view solution for captured particles.

```
function postproc(postp,varargin)
% This function will plot the solutions based on what is choosen in postp.
fl = evalin('base','fl;');
mg = evalin('base','mg;');
fm = evalin('base','fm;');
hsv_scl =1.0; % Scale Color HUE (1 = full range, 2 = 1/2 range, etc.)
hsv3 = 1;% HSV value (0 to 1)
hsvshift = 2.5;% Shift HSV by a value
hsvs= 1.1; %hsvs - HSV
ms = 10; % Particle size
% Plot boundary points
figure
skipnm = 3;
plot3(fl.xbnd(1:skipnm:end),fl.ybnd(1:skipnm:end),fl.zbnd(1:skipnm:end),...
      'k.','MarkerSize',0.01);
axis([-0.1 0.1 -0.1 0.1 -0.1 0.1])
xlabel('X (m)','fontsize',12,'fontweight','bold')
ylabel('Y (m)','fontsize',12,'fontweight','bold')
zlabel('Z (m)','fontsize',12,'fontweight','bold')
title('3D View','fontsize',12,'fontweight','bold')
set(gca,'fontsize',12)
hold on
axis square
zoom(2.5)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plot Interpolation Function   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if postp.interp == 1;
% Pick two arbitrary points and plot them
p0 = [0.065,-0.005,0];% Interior point
p1 = [0.06,-0.009,0.005];% Exterior point
interp_example_figures(fl, postp,p0,p1)
```



```

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Plot Coordinates %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if postp.coord ==1
    plot3(0,0,0,'ko','MarkerSize',10,'MarkerFaceColor','k'), hold on
    qsz = 0.015;
    quiver3(0,0,0,qsz,0,0,'LineWidth',2)% x,y,z,u,v,w
    quiver3(0,0,0,0,qsz,0,'LineWidth',2)% x,y,z,u,v,w
    quiver3(0,0,0,0,0,qsz,'LineWidth',2)% x,y,z,u,v,w
    text(qsz,0,0,'X','FontSize',12,'FontWeight','bold')
    text(0,qsz,0,'Y','FontSize',12,'FontWeight','bold')
    text(0,0,qsz,'Z','FontSize',12,'FontWeight','bold')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Plot fl streamlines %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if postp.flstrm ==1
    nend =length(fm.ptrack(1,:));
    % Check if fl.stream exists
    exstchk = 'stream';
    assignin('base','exstchk',exstchk)
    exst = evalin('base','isfield(fl,exstchk)');
    if exst ==1
str1 = ['Streamlines found. Calculate new streamlines based',...
'on particle tracking start position?'];
        strmcalt = questdlg(str1);
        switch strmcalt
            case 'Yes'
                exstcont =1;
            case {'No','Cancel'}
                exstcont =0;
        end
    else
str1 = ['Streamlines not found. Are you sure you want to calculate'...
'them? This will take a while...'];
        strmcalt = questdlg(str1);
        switch strmcalt
            case 'Yes'
                exstcont =1;
            case {'No','Cancel'}

```

```

        exstcont =2;
    end
end

if exstcont <2
    for nn =1:nend
        % Calculate streamlines if they don't exist
        if exstcont ==1
            curstr = ['Calculating Streamline number ',num2str(nn),'...'];
            disp(curstr)
        % Initial coordinates
        xyz0cell = cell2mat(fm.ptrack(nn));
        x0 = xyz0cell(1,1);
        y0 = xyz0cell(1,2);
        z0 = xyz0cell(1,3);
        xyz0 = [x0, y0, z0];
        % Calculate streamline using Matlab ode solver
        [xstream ystream zstream T] = SL_plot(fl.Fvx,fl.Fvy,fl.Fvz,xyz0);
        fl.stream{1,nn} = [xstream ystream zstream T];
        curstr = ['Finished calculating streamline'];
        disp(curstr)
    end

    % Plot results
    fsol = [];
    yo = fl.stream;
    fsol = cell2mat(yo(nn));
    fsol_x = fsol(:,1);
    fsol_y = fsol(:,2);
    fsol_z = fsol(:,3);
    t = fsol(:,4);
    HSVc(1) = hsvs -(nn+hsvshift)/(hsv_scl*nend); % Color
    HSVc(2) = 1;% Saturation
    HSVc(3) = hsv3;
    clr_rgb1 = hsv2rgb(HSVc);
    plot3(fsol_x,fsol_y,fsol_z,'Marker','.', 'MarkerEdgeColor',clr_rgb1(1:3),...
        'MarkerFaceColor',clr_rgb1(1:3), 'LineWidth',1.5, 'MarkerSize',ms)
    end
end
end
assignin('base','fl',fl)

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plot mg vectors %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if postp.mgvect ==1
    for kk =1:length(postp.vectstyle)
        vsty = cell2mat(postp.vectstyle(kk));
        switch vsty
            case 'zpln'
                mg.vect_min = -0.0005;
                mg.vect_max = 0.0005;
                [ir,ic] = find(mg.z<=mg.vect_max & mg.z>= mg.vect_min);
                x = mg.x(ir,1);
                y = mg.y(ir,1);
                z = mg.z(ir,1);
                Hx = mg.hx(ir,1);
                Hy = mg.hy(ir,1);
                Hz = mg.hz(ir,1);
                quiver3(x,y,z,Hx,Hy,Hz,2,'filled','Color',[255/255 140/255 0]);
                hold on
            case 'ypln'
                mg.vect_min = -0.0005;
                mg.vect_max = 0.0005;
                [ir,ic] = find(mg.y<=mg.vect_max & mg.y>= mg.vect_min);
                x = mg.x(ir,1);
                y = mg.y(ir,1);
                z = mg.z(ir,1);
                Hx = mg.hx(ir,1);
                Hy = mg.hy(ir,1);
                Hz = mg.hz(ir,1);
                quiver3(x,y,z,Hx,Hy,Hz,1,'filled','b')
                hold on
            case 'all'
                x = mg.x;
                y = mg.y;
                z = mg.z;
                Hx = mg.hx;
                Hy = mg.hy;
                Hz = mg.hz;
                quiver3(x,y,z,Hx,Hy,Hz,1,'filled','g')
                hold on
        end
    end
end

```

```

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Plot inlet particles %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if postp.inlet ==1
    % Choose which particles to plot
    particles = postp.particles;
    yo = fm.ptrack;
    nend = length(yo);
    switch particles
    % Particles_captured
    case 'pcapt'
        cntrchk = fm.particles_capt;
    case 'pnan'
        cntrchk = fm.particles_nan;
    case 'pout'
        cntrchk = fm.particles_out;
    case 'plim'
        cntrchk = fm.particles_lim;
    case 'all'
        cntrchk = ones(1,nend);
    end
    [I,cntr] = find(cntrchk ==1); % cntr corresponds to particles to capture

    for nn = cntr
        % Initial coordinates
        x0 = fm.ptrack{1,nn}(1,1);
        y0 = fm.ptrack{1,nn}(1,2);
        z0 = fm.ptrack{1,nn}(1,3);
        xyz0 = [x0, y0, z0];
        % Plot each particle
        HSV(nn,1) = hsvs -(nn+hsvshift)/(hsv_scl*nend); % Color
        HSV(nn,2) = 1;% Saturation
        HSV(nn,3) = hsv3;
        clr_rgb1 = hsv2rgb(HSV);
        plot3(x0,y0,z0,'Marker','o','MarkerSize',ms,'MarkerEdgeColor',...
            clr_rgb1(nn,1:3),'MarkerFaceColor',clr_rgb1(nn,1:3),'LineWidth',1.5)
        hold on
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Plot fm solution %%%%%%%%%

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if postp.fm ==1
    % Plot solution
    fsol = [];
    yo = fm.ptrack;
    to = fm.T;
    nend = length(yo);
    skp =100;% animation values to skip
    cnum = 1;
    % Choose which particles to plot
    particles = postp.particles;
    switch particles
    % Particles_captured
        case 'pcapt'
            cntrchk = fm.particles_capt;
        case 'pnan'
            cntrchk = fm.particles_nan;
        case 'pout'
            cntrchk = fm.particles_out;
        case 'plim'
            cntrchk = fm.particles_lim;
        case 'all'
            cntrchk = ones(1,nend);
    end
    [I,cntr] = find(cntrchk ==1);% cntr corresponds to particles to capture

    for nn = cntr
        if isempty(cell2mat(yo(nn)))~=1
            fsol = cell2mat(yo(nn));
            tsol = cell2mat(to(nn));
            fsol_x = fsol(:,1);
            fsol_y = fsol(:,2);
            fsol_z = fsol(:,3);
            t = tsol(:,1);
            HSVc(1) = hsvs -(nn+hsvshift)/(hsv_scl*nend); % Color
            HSVc(2) = 1;% Saturation
            HSVc(3) = hsv3;
            clr_rgb1 = hsv2rgb(HSVc);
            if postp.fm_noan ==1
plot3(fsol_x,fsol_y,fsol_z,'Marker','.', 'MarkerEdgeColor',clr_rgb1(1:3),...
      'MarkerFaceColor',clr_rgb1(1:3), 'MarkerSize',ms)
            elseif postp.fm_an ==1
                for kk =1:skp:length(t)

```

```

plot3(fsol_x(kk),fsol_y(kk),fsol_z(kk),'Marker','.', 'MarkerEdgeColor',...
      clr_rgb1(1:3), 'MarkerFaceColor',clr_rgb1(1:3), 'MarkerSize',ms)
      if kk>1
          pause(t(kk)-t(kk-skp))
      end
    end
  end
end
end
end
end
end
% Set current axes properties
viewopt = postp.plotview;
switch viewopt
    case 'side'
        dar = [1 1 1];
        axview = [-180 -90];
        axxlim = [-0.075 0.1];
        axylim = [-0.02 0.06];
        set(gca, 'View', axview, 'XLim', axxlim, 'YLim', axylim, 'DataAspectRatio', dar)
        title('Side View', 'fontsize', 12, 'fontweight', 'bold')
        grid on
        zoom(0.8)
    case 'top'
        dar = [1 1 1];
        axview = [-180 0];
        axxlim = [-0.075 0.1];
        axzlim = [-0.02 0.02];
        set(gca, 'View', axview, 'XLim', axxlim, 'ZLim', axzlim, 'DataAspectRatio', dar)
        title('Top View', 'fontsize', 12, 'fontweight', 'bold')
        zoom(0.8)
        grid on
    case 'inlet'
        dar = [1 1 1];
        axview = [-180 0];
        axxlim = [max(fl.xbnd)-fl.dia max(fl.xbnd)];
        axzlim = [-fl.dia/2 fl.dia/2];
        set(gca, 'View', axview, 'XLim', axxlim, 'ZLim', axzlim, 'DataAspectRatio', dar)
        title('Particle Inlet Top View', 'fontsize', 12, 'fontweight', 'bold')
        zoom(5)
        grid on
    case 'front'
        dar = [1 1 1];
        axview = [-90 0];

```

```

axyylim = [-0.01 0.055];
axzylim = [-0.02 0.02];
cuv = [0 1 0];
set(gca,'View',axview,'ZLim',axzylim,'YLim',axyylim,'DataAspectRatio',...
    dar,'CameraUpVector',cuv)
title('Front View','fontsize',12,'fontweight','bold')
zoom(1)
grid on
end
end

```

B.4 Fluid import and interpolation functions

These functions import the fluid data and determine boundary points and streamlines.

```

function [xstream ystream zstream T] = SL_plot(Fvx,Fvy,Fvz,xyz0)

% Use same ODE options as fl_mg_soln if fm exists
fmstr = 'fm';
assignin('base','fmstr',fmstr)
exst = evalin('base','exist(fmstr)');
if exst ==1 % use fm parameters
    fm = evalin('base','fm');
    options = odeset('RelTol',fm.reltol,'AbsTol',fm.abstol);
    Tstop = fm.tstop;
% Set ODE solver
styp = fm.solvertyp;
switch styp
    case {'ODE45','ode45'}
        [T,Y] = ode45(@oderhs,[0 Tstop],xyz0,options);
    case {'ODE23','ode23'}
        [T,Y] = ode23(@oderhs,[0 Tstop],xyz0,options);
    otherwise
        [T,Y] = ode45(@oderhs,[0 Tstop],xyz0,options);
end

else % use standard parameters
    Tstop = 16;
%calculate streamline using Matlab ode solver
options = odeset('RelTol',1e-5);

```

```

    [T,Y] = ode23(@oderhs,[0 Tstop],xyz0,options);
end
tic
xstream = Y(:,1); ystream = Y(:,2); zstream = Y(:,3);
toc

function dy = oderhs(t,xyz)
xi = xyz(1); yi = xyz(2); zi = xyz(3);
vxi = Fvx(xi,yi,zi);
vyi = Fvy(xi,yi,zi);
vzi = Fvz(xi,yi,zi);
dy = zeros(3,1);    % a column vector
dy(1) = vxi;
dy(2) = vyi;
dy(3) = vzi;
end
end

function [xbnd ybnd zbnd]=boundary_points(x,y,z,vx,vy,vz,n)
m=1;
for i = 1:n
    speed = sqrt(vx(i)^2+vy(i)^2+vz(i)^2);
    if speed == 0
        xbnd(m,1)=x(i);
        ybnd(m,1)=y(i);
        zbnd(m,1)=z(i);
        m = m+1;
    else
        xbnd(1,1)=0;
        ybnd(1,1)=0;
        zbnd(1,1)=0;
    end
end
end
end

function FLstrm(ptype)
% Plots streamline and boundary points.
% Input options are "an" for animated or "no_an" for no animation, just
% plotting

```



```

fl = evalin('base','fl');
figure
plot3(fl.xbnd,fl.ybnd,fl.zbnd,'b.','MarkerSize',0.5);% plot boundary points
axis([-0.1 0.1 -0.1 0.1 -0.1 0.1])
xlabel('x')
ylabel('y')
zlabel('z')
hold on

switch ptype
case 'no_an'
    strm_end = 2;
    for kk = 1:fl.nend*2
        fl.xstrmkk = cell2mat(fl.strm(kk,1));
        fl.ystrmkk = cell2mat(fl.strm(kk,2));
        fl.zstrmkk = cell2mat(fl.strm(kk,3));
        strm_end = length(fl.xstrmkk);
        HSV(kk,1) = kk/(2*fl.nend); % Color
        HSV(kk,2) = 1;% Saturation
        HSV(kk,3) = 1;%jj/strm_end; % Brightness
        clr_rgb1 = hsv2rgb(HSV);
    plot3(fl.xstrmkk,fl.ystrmkk,fl.zstrmkk,'Marker','.', 'MarkerEdgeColor',...
        clr_rgb1(kk,1:3), 'MarkerFaceColor',clr_rgb1(kk,1:3), 'LineWidth',1.5)
    end

case 'an'
    h = msgbox('Click OK to begin animation');
    uiwait(h);
    for kk = 1:fl.nend
        fl.xstrmkk = cell2mat(fl.strm(kk,1));
        fl.ystrmkk = cell2mat(fl.strm(kk,2));
        fl.zstrmkk = cell2mat(fl.strm(kk,3));
        strm_end = length(fl.xstrmkk);
        for jj = 1:strm_end
            HSV(jj,1) = kk/(fl.nend); % Color
            HSV(jj,2) = 1;% Saturation
            HSV(jj,3) = 1;%jj/strm_end; % Brightness
            clr_rgb1 = hsv2rgb(HSV);
        plot3(fl.xstrmkk(jj),fl.ystrmkk(jj),fl.zstrmkk(jj), 'Marker','.',...
            'MarkerEdgeColor',clr_rgb1(jj,1:3), 'MarkerFaceColor',clr_rgb1(jj,1:3),...
            'LineWidth',1.5)
        pause(0.0001)
    end
end

```

```

        end
    end
end

clear fl.xstrmkk fl.ystrmkk fl.zstrmkk
disp('Finished plotting streamlines!')
assignin('base','fl',fl)

end

function [x y z vx vy vz n D] = Import_fl_data(filename,delim,headers,clm)
%Import_fl_data Summary of this function goes here
% Detailed explanation goes here
D=importdata(filename,delim,headers);
fluid=D.data;
x = fluid(:,clm(1));
y = fluid(:,clm(2));
z = fluid(:,clm(3));
vx = fluid(:,clm(4));
vy = fluid(:,clm(5));
vz = fluid(:,clm(6));
n = length(x);
end

```

B.5 Magnetostatic import and interpolation functions

These functions import the magnetostatic data and determine boundary points and streamlines.

```

function MGstrm(ptype)
% Plots streamline and boundary points.
% Input options are "an" for animated or "no_an" for no animation, just
% plotting
fl = evalin('base','fl');
mg = evalin('base','mg');
figure
plot3(fl.xbnd,fl.ybnd,fl.zbnd,'b.','MarkerSize',0.5);% plot boundary points
axis([-0.1 0.1 -0.1 0.1 -0.1 0.1])

```

```

xlabel('x')
ylabel('y')
zlabel('z')
hold on

switch ptype
case 'no_an'
    strm_end = 2;
    for kk = 1:mg.nend*2
        mg.xstrmkk = cell2mat(mg.strm(kk,1));
        mg.ystrmkk = cell2mat(mg.strm(kk,2));
        mg.zstrmkk = cell2mat(mg.strm(kk,3));
        strm_end = length(mg.xstrmkk);
        HSV(kk,1) = kk/(2*mg.nend); % Color
        HSV(kk,2) = 1;% Saturation
        HSV(kk,3) = 1;%jj/strm_end; % Brightness
        clr_rgb1 = hsv2rgb(HSV);
        plot3(mg.xstrmkk,mg.ystrmkk,mg.zstrmkk,'Marker','.', 'MarkerEdgeColor',...
            clr_rgb1(kk,1:3), 'MarkerFaceColor',clr_rgb1(kk,1:3), 'LineWidth',1.5)
    end

case 'an'
    h = msgbox('Click OK to begin animation');
    uiwait(h);
    for kk = 1:mg.nend
        mg.xstrmkk = cell2mat(mg.strm(kk,1));
        mg.ystrmkk = cell2mat(mg.strm(kk,2));
        mg.zstrmkk = cell2mat(mg.strm(kk,3));
        strm_end = length(mg.xstrmkk);
        for jj = 1:strm_end
            HSV(jj,1) = kk/(mg.nend); % Color
            HSV(jj,2) = 1;% Saturation
            HSV(jj,3) = 1;%jj/strm_end; % Brightness
            clr_rgb1 = hsv2rgb(HSV);
            plot3(mg.xstrmkk(jj),mg.ystrmkk(jj),mg.zstrmkk(jj), 'Marker','.',...
                'MarkerEdgeColor',clr_rgb1(jj,1:3), 'MarkerFaceColor',clr_rgb1(jj,1:3),...
                'LineWidth',1.5)
            pause(0.0001)
        end
    end
end
end

```

```

clear mg.xstrmkk mg.ystrmkk mg.zstrmkk
disp('Finished plotting streamlines!')
assignin('base','mg',mg)

```

```

end

```

```

function [x y z hx hy hz n D] = Import_mg_data(filenm,delim,headers,clm)
%Import_fl_data Summary of this function goes here
% Detailed explanation goes here
D=importdata(filenm,delim,headers);
mag=D.data;
x = mag(:,clm(1));
y = mag(:,clm(2));
z = mag(:,clm(3));
hx = mag(:,clm(4));
hy = mag(:,clm(5));
hz = mag(:,clm(6));
n = length(x);
end

```

Appendix C

Matlab Spectrometer Code

C.1 Spectrometer calibration processing

Figure C.1: Matlab spectrometer calibration processing file structure.

This function sets all parameters for the spect_cal.m file.

```

% Call function to process spectrometer for Exp #2
% Written by Jay Wheaton, 2-29-2012
clear all, clc, close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Input parameters %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
datapts = 2048;% Total # of wavelength points
num_samp = 17;% # of samples for each concentrations
num_conc = 21;% # of concentrations collected
num_tot = num_conc*num_samp;% Total # of data points
vol_tot = 0.750;% Total volume in liters
vol_par = 35*10^-6;% Particle volume added each time in liters
specnum=800;%Corresponds to row of input data i.e. 560nm @ data point 595?
int_rg = [450, 700]; % Range of integration for absorbance plots
% Plots include:
%   'intensity'      all intensity data
%   'absorbance'     all absorbance data
%   'av_abs'         averaged absorbance data
%   'ab_int'         integrated absorbance
%   'av_int'         integrated av_abs
plots = {'intensity','absorbance','av_abs','ab_int','av_int'};
  
```

```

plots = {'intensity','absorbance','av_abs','ab_int','av_int','abs_color'};
% Concentration styles (conc_style) for x-axis include:
%   'nums'          1,2,3,...,n
%   'particles'     # of particles
%   'volume'        # of particles / Liter of fluid
conc_style = {'volume'};
% Include error bars:
%   'ebar'          include error bars
%   'noebar'        do not include error bars
plot_style = {'noebar'};

fdir = ['E:\Jay_Wheaton\Wheaton_thesis\mivcf\web\public\',...
       'Final_thesis_experiments\Experiment2\Spectrometer'];
%fdir = uigetdir

fs = 12;% Axis font size
fst = 14;% Title font size

% Call function to read data and make plots
[wa,prospec,absorbance,ab_avg,int_ab,int_ab_avg,particle_conc] = ...
    spect_cal(fdir,datapts,num_samp,num_conc,vol_tot,...
    vol_par,int_rg,plots, conc_style, plot_style,fs,fst);

```

This m-file uses the raw spectrometer intensity data to determine absorbance and concentration.

```

function [wa,prospec,absorbance,ab_avg,int_ab,int_ab_avg,particle_conc]...
    = spect_cal(fdir,datapts,num_samp,num_conc,vol_tot,vol_par,int_rg,...
    plots,conc_style,plot_style,fs,fst)

%spect_cal is used to input, analyze, and plot spectrometer data.
% This function is used to process spectrometer calibration
% concentrations data. Files should be saved as ".jdx".
% the plots can be 'intensity', 'absorbance', 'av_abs', or 'av_int'
% corresponding to all intensity data, all absorbance data,
% averaged absorbance data, or integrated average absorbance
num_tot = num_conc*num_samp;% Total # of data points
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Import Data %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for nn = 1:num_tot
    % Correct the counter for file names

```

```

if nn <=10
    nzero = '0000';
elseif nn <=100
    nzero = '000';
elseif nn<=1000
    nzero = '00';
elseif nn <= 10000
    nzero = '0';
end
% Read in calibration file named calfile
calfile = [fdir, '\cal', nzero, num2str(nn-1), '.jdx'];
fid = fopen(calfile, 'r');
specdata = textread(calfile, '', 'delimiter', ',', 'commentstyle', ...
    'shell', 'headerlines', 28);
fclose(fid);
% Store wavelength and intensity data
procspec(:, nn) = specdata(1:datapts, 2); % Processed spectrum (column 2)
if nn==1
    % Wavelength (column 1), same for all data
    wa(:, 1) = specdata(1:datapts, 1);
    % Dark spectrum (column 2), same for all
    darkspec(:, 1) = specdata(datapts+1:datapts*2, 2);
    % Reference spectrum (column 2). same for all
    refspect(:, 1) = specdata(datapts*2+1:datapts*3, 2);
else
end
% calculate absorbance from intensity
diff(:, nn) = (procspec(:, nn) - darkspec) ./ (refspect - darkspec);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Process Data %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute Absorbance from intensity
absorbance = real(-10*log10(diff));
% Compute concentration of particles
wv = 10; % w/v (g/L)
rho_p = 1.58; % Particle density (g/cc)
diap = 8.34*10^-4; % Particle diameter (cm)
NpV = 6*wv/(rho_p*pi*diap^3); % Number of particles per Liter of fluid
er_spect = 0.005; % spectrometer error (percent/100)
er_pip = 0.008*vol_par; % pipette error (volume)
% Split data into cells
for jj = 1:num_conc

```



```

% Plot absorbance over entire spectrum
    case 'absorbance'
figure
plot(wa,absorbance)
    axis([360,1000,-0.05,0.3])
    ti2 = ['Spectrometer Absorbance for ',num2str(num_conc),...
        ' Known Concentrations'];
    title(ti2)
    xlabel('Wavelength (nm)')
    ylabel('Absorbance (OD)')
    grid minor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot average absorbance over entire spectrum
    case 'av_abs'
figure
    val3 = cell2mat(ab_avg);
plot(wa,val3)
    axis([360,1000,-0.05,0.2])
    ti3 = ['Average Spectrometer Absorbance for ',num2str(num_conc),...
        ' Known Concentrations'];
    title(ti3)
    xlabel('Wavelength (nm)')
    ylabel('Absorbance (OD)')
    grid minor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot absorbance integration over specified range
    case 'ab_int'
        qq = char(conc_style);% choose x-axis type
        switch qq
            case 'nums'
                particle_conc = 1:1:num_conc;
                xlab = 'Concentration (concentration #)';
            case 'particles'
                particle_conc = num_part*10^-6; %#particles to # million particles
                xlab = 'Concentration (million particles)';
            case 'volume'
                particle_conc = percent_vol*10^-6;
                xlab = 'Concentration (million particles per liter of fluid)';
        end
    end
% Calculate error

for rr = 1:num_conc
    erry(rr) = int_ab(rr*10)*0.2;

```

```

        errx(rr) = er_part(rr)*10^-6+particle_conc(rr)*0.2;
end
assignin('base','errx',errx)
assignin('base','erry',erry)
assignin('base','er_part',er_part)

% Make nice colormap for plotting
eb = char(plot_style);
figure
tt = 1;

    for ss = 1:num_conc
        for uu =1:num_samp
            HSV(tt,1) = ss/num_conc; % Color
            HSV(tt,2) = 1;% Saturation
            HSV(tt,3) = uu/(num_samp); % Brightness
            clr_rgb1 = hsv2rgb(HSV);
            % plot data
            plot(particle_conc(ss),int_ab(tt),'o','MarkerFaceColor',...
                clr_rgb1(tt,1:3),'MarkerSize',10,'MarkerEdgeColor',...
                'k'), hold on
            % save concentration for all data points
            particle_conc2(tt)=particle_conc(ss);
            tt=tt+1;
        end
        % plot error
        switch eb
        case 'ebar'
            errorbar(particle_conc(ss),int_ab(tt-1),erry(ss))
            herrorbar(particle_conc(ss),int_ab(tt-1),errx(ss));
        case 'noebar'
            eb2 = 0;
        end
    end

ax1 = 1.1*max(particle_conc);
ax2 = 1.1*max(int_ab_avg);
axis([-0.001,ax1,-0.001,ax2])
ti4 = ['Integrated Spectrometer Absorbance from ',...
        num2str(int_rg(1)),'nm to ',num2str(int_rg(2)),'nm'];
title(ti4,'FontSize',fst)
xlabel(xlab,'FontSize',fs)
ylabel('Integrated Absorbance (OD*nm)','FontSize',fs)

```

```

grid minor

colormap(clr_rgb1)
%h=colorbar;
%caxis([particle_conc(1) particle_conc(num_conc)])
% Fit a line to the data
ft4 = fittype('A*x','coeff',{'A'});
[f4,g4] = fit(particle_conc2',int_ab',ft4,'start',1);
y4 = @(x) f4.A *x;
% Round values for text output
Aco = num2str(floor(f4.A*100000)/100000);
Rco = num2str(floor(g4.rsquare*100000)/100000);

txt4 = ['R^2 = ', Rco(1:5)];
txt5 = ['y = ',Aco(1:6),'*x'];
% plot best fit line with text (R^2 and equation)
plot(particle_conc2',y4(particle_conc2'),'k--','LineWidth',2)
text(12,0.55*max(int_ab),txt4,'HorizontalAlignment','center',...
'BackgroundColor',[1 1 1],'EdgeColor',[0 0 0],'FontSize',fs);
text(12,0.5*max(int_ab),txt5,'HorizontalAlignment','center',...
'BackgroundColor',[1 1 1],'EdgeColor',[0 0 0],'FontSize',fs);

% Save figure
fignm = ['Spect_Cal_AbInt_',date];
h = gcf; set(h,'PaperOrientation','landscape','PaperUnits','normalized',...
'PaperPosition',[0 0 1 1])
print(gcf,'-dpdf','-r300',fignm)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot average absorbance integration over specified range
case 'av_int'
    qq = char(conc_style);% choose x-axis type
    switch qq
        case 'nums'
            particle_conc = 1:1:num_conc;
            xlab = 'Concentration (concentration #)';
        case 'particles'
            particle_conc = num_part*10^-6; %%particles to # million particles
            xlab = 'Concentration (million particles)';
        case 'volume'
            particle_conc = percent_vol*10^-6;
            xlab = 'Concentration (million particles per liter of fluid)';
    end
% Make nice colormap for plotting

```

```

color_conc = 150;
figure
for rr = 1:num_conc
    H(rr,1) = rr/num_conc; % Color
    H(rr,2) = 1;% Saturation
    H(rr,3) = 1;%rr/(num_conc); % Brightness
    clr_rgb = hsv2rgb(H);
    plot(particle_conc(rr),int_ab_avg(rr),'o','MarkerFaceColor',...
        clr_rgb(rr,1:3),'MarkerSize',10,'MarkerEdgeColor','k'), hold on
end
ax1 = 1.1*max(particle_conc);
ax2 = 1.1*max(int_ab_avg);
axis([0,ax1,0,ax2])
ti4 = ['Integrated Average Spectrometer Absorbance from ',...
    num2str(int_rg(1)),'nm to ',num2str(int_rg(2)),'nm'];
title(ti4,'FontSize',fst)
xlabel(xlab,'FontSize',fs)
ylabel('Integrated Absorbance (OD*nm)','FontSize',fs)
grid minor
set(gca,'FontSize',fs)
colormap(clr_rgb)
h=colorbar;
caxis([particle_conc(1) particle_conc(num_conc)])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot absorbance with colorbar and save
case 'abs_color'
    clear HSV clr_rgb1 cmap h
    particle_conc = percent_vol*10^-6;
ylab = 'Concentration (million particles per liter of fluid)';

% Make nice colormap for plotting
figure
    for tt = 1:num_conc
        % for uu =1:num_samp
        HSV(tt,1) = tt/num_conc; % Color
        HSV(tt,2) = 1;% Saturation
        HSV(tt,3) = 1;%uu/(num_samp); % Brightness
        clr_rgb1 = hsv2rgb(HSV);
        % plot data
        plot(wa,ab_avg{tt},'.','MarkerFaceColor',...
            clr_rgb1(tt,1:3),'MarkerSize',6,'MarkerEdgeColor',...
            clr_rgb1(tt,1:3)), hold on
        tt=tt+1;
    end

```

```

end

axis([360,1000,-0.05,0.3])

ti2 = ['Average Spectrometer Absorbance for ',num2str(num_conc),...
      ' Known Concentrations'];
title(ti2,'FontSize',fst)
xlabel('Wavelength (nm)','FontSize',fs)
ylabel('Absorbance (OD)','FontSize',fs)
grid minor
set(gca,'FontSize',fs)

cmap = colormap(clr_rgb1);
h=colorbar('Location','EastOutside');
caxis([particle_conc(1) particle_conc(num_conc)])

set(get(h,'Ylabel'),'String',ylab,'VerticalAlignment','cap',...
     'Rotation',270,'FontSize',fs)
hloc = get(h,'Ylabel');
set(hloc,'Position',get(hloc,'Position').*[1.5 1 1])

set(gca,'FontSize',fs)
% Save figure
fignm = ['Spect_Cal_Abs_',date];
h = gcf; set(h,'PaperOrientation','landscape','PaperUnits',...
            'normalized','PaperPosition',[0 0 1 1])
print(gcf,'-dpdf','-r300',fignm)
otherwise
    disp('No plots choosen for spect_cal function')
end
end
end

```

C.2 Spectrometer η_{capt} processing

Figure C.2: Matlab spectrometer capture efficiency processing file structure.

This m-file is used to set parameters for image processing of cancer cells and setting a dual y-axis plot.

```
% This code is used to process the spectrometer data for the cancer cells
datapts = 2048;% Total # of wavelength points
vol_par = 750; % Particle volume added each time in micro-liters
int_rg = [450, 700]; % Range of integration for absorbance plots
cancer_exp = 2; % 1 = First experiment, 2 = second experiment
if cancer_exp == 1
    % Cancer Cells experiment 1
    %c = {'YFNM','YFYM','YFNM2','NFMN','NFYM'};
    c = {'YFYM','YFNM2','NFMN','NFYM'};
    skpnm =1; % Step size for collecting data
    skipdata = 240; % Points to skip due to initial mixing (1 = no skip)
    vol_tot = 0.5; % Total volume in liters
    num_tot =10220;% Total # of samples+1
else
    % Cancer Cells 2. This one corrects the error in YFYM w/ 100ms
    % integration instead of 750ms
    c = {'YFYM','YFNM','NFMN','NFYM'};
    skpnm =8; % Step size for collecting data
    skipdata = 240; % Points to skip due to initial mixing (1 = no skip)
    vol_tot = 0.5; % Total volume in liters
    num_tot =6300;% Total # of samples+1
end

% Process data
for kk = 1:length(c)
    if cancer_exp == 1
        pathnm = ['E:\Jay_Wheaton\Wheaton_thesis\mivcf\web\public\',...
            'Final_thesis_experiments\Cancer_Cells\Spectrometer\'],...
            cell2mat(c(kk)), '\']; % Cancer Cells
    else
        pathnm = ['E:\Jay_Wheaton\Wheaton_thesis\mivcf\web\public\',...
            'Final_thesis_experiments\Cancer_Cells2\']...
            ,cell2mat(c(kk)), '\Spectrometer\']; % Cancer Cells 2
    end
    cyfnm2 = cell2mat(c(kk));
    switch cyfnm2;
        case 'YFNM2'
            basenam = 'YFNM'; % File basename, i.e. 'cal' for cal00123
```

```

        ctyp = 1;
    case 'YFYM'
        if cancer_exp == 1
            ctyp = 1; % Cancer Cells
        else
            ctyp = 2; % Cancer Cells 2. This is where correction occurs
        end
        basenam = c(kk); % File basename, i.e. 'cal' for cal00123
    otherwise
        ctyp = 1;
        basenam = c(kk); % File basename, i.e. 'cal' for cal00123
    end

if ctyp == 2
    procdata{kk} = spect_tl_c3(datapts,num_tot,vol_tot,vol_par,int_rg,...
        basenam,skipdata,pathnm,skpnm);
else
    procdata{kk} = spect_tl_c2(datapts,num_tot,vol_tot,vol_par,int_rg,...
        basenam,skipdata,pathnm,skpnm);
end
procdata{kk}.cancer_exp = cancer_exp;
procdata{kk}.c = c{kk};
procdata{kk}.int_rg = int_rg;
end

%% Plot properties
%   Plots include:
%   'intensity'      all intensity data
%   'absorbance'     all absorbance data
%   'ab_int'         integrated absorbance
%   'capteff'        percent captured based on integrated abs.
%   'capteffpar'     #particles captured based on integrated abs.
ceffit(1) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
ceffit(2) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
ceffit(3) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
ceffit(4) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
ceffit(5) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
    %plots = {'intensity','absorbance','ab_int','capteff','capteffpar'};
plots = {'capteff'};
suball = 0; % Combine each scenario with subplots
color(1,:) = [1 0 0];
color(2,:) = [0 0.7 0];
color(3,:) = [0 0 1];
color(4,:) = [255/255 165/255 0/255];

```

```

color(5,:) = [0 1 1];

for kk = 1:length(procddata)
    procddata{kk}.color = color(kk,:);
    procddata{kk}.cefffit = cefffit(kk);
% Shorten the strings to be 4 dimensions. This is needed for the YFNM2
% case. I want that to show up as YFNM on the legend.
    if procddata{kk}.cancer_exp == 1
        tempstr = char(procddata{kk}.c);
        procddata{kk}.c = tempstr(1:4);
    end
end

%% Plot data
plots = {'capteff'};
suball = 0; % Combine each scenario with subplots
for kk = 1:length(procddata)
d = procddata{kk};% Compilation of data
    [p,f,cnst,r2_temp,cfiteq] = abs_plots2(d,procddata{kk}.c,...
        procddata{kk}.cefffit,procddata{kk}.int_rg,plots,suball,...
        procddata{kk}.color,kk);
    leg1 = [char(procddata{kk}.c),' at 0.3 L/min'];
    procddata{kk}.p = p;
    procddata{kk}.f = f;
    procddata{kk}.cnst = cnst;
    procddata{kk}.r2 = r2_temp;
    procddata{kk}.cfiteq = cfiteq;
end

%% Plot all data w/ curve fits
clearvars -except procddata
%c = zeros(1,length(procddata))
figure
ms = 3;
for jj = 1:length(procddata)
H =plot(procddata{jj}.timeab,procddata{jj}.ceff,'o','MarkerFaceColor',...
    procddata{jj}.color,'MarkerEdgeColor',procddata{jj}.color,...
    'MarkerSize',ms);hold on
c{jj} = procddata{jj}.c;
end
fs = 14;
fst = 16;

a1 = legend(char(c{1}),char(c{2}),char(c{3}),char(c{4}),...

```



```

        'Location','NorthWest');
a2 = get(a1,'children');
set(a2(1:3:end),'MarkerSize',ms)
grid minor
ti5 = ['Capture Efficiency Based on Integrated Absorbance from ',...
       num2str(procdata{1}.int_rg(1)),'nm to ',...
       num2str(procdata{1}.int_rg(2)),'nm'];
title(ti5,'FontSize',fst)
xlabel('Time (min)','FontSize',fs)
ylabel('Percent of Cells Captured (%)','FontSize',fs)
xmx = max(procdata{1}.timeab)+5;
axis([0 120 -2 50])

set(gca,'FontSize',fs,'XTick',[0:30:240])

% Plot best fit lines

for jj = 1:length(procdata)
f4 = procdata{jj}.cnst;
y4 = procdata{jj}.f;
plot(procdata{jj}.timeab,y4(procdata{jj}.timeab),'--','LineWidth',2.5,...
     'Color','k')
eqtot = [' ' procdata{jj}.cfiteq, ' ',procdata{jj}.r2, ' '];
    if procdata{jj}.cancer_exp ==1
        text(50,39-3*jj,eqtot,'HorizontalAlignment','center',...
             'BackgroundColor',[1 1 1],'EdgeColor',procdata{jj}.color,...
             'FontSize',fs,'LineWidth',1.75);
    %text(52,35-2*jj,procdata{jj}.r2,'HorizontalAlignment','center',...
    % 'BackgroundColor',[1 1 1],'EdgeColor',procdata{jj}.color,'FontSize',fs);
    else
        text(50,48-3*jj,eqtot,'HorizontalAlignment','center',...
             'BackgroundColor',[1 1 1],'EdgeColor',procdata{jj}.color,...
             'FontSize',fs,'LineWidth',1.75);
        %text(120,57-3*jj,procdata{jj}.r2,'HorizontalAlignment','center',...
        % 'BackgroundColor',[1 1 1],'EdgeColor',procdata{jj}.color,'FontSize',fs);
    end
end
fignm = ['Capt_Eff_120_',num2str(procdata{1}.cancer_exp)];
h = gcf; set(h,'PaperOrientation','landscape','PaperUnits','normalized',...
             'PaperPosition',[0 0 1 1])
print(gcf,'-dpdf',fignm)
%%
% Plot raw absorbance

```

```

figure
fs = 11;
fst = 12;
for zz = 1:4
    subplot(2,2,zz)
    plot(procdata{zz}.wa,procdata{zz}.absorbance)
    xlabel('Wavelength (nm)','FontSize',fs)
    ylabel('Absorbance (OD)','FontSize',fs)
    axis([375,1000,0,0.34])
%    axis([375,1000,0,0.2])
    title(char(c(zz)),'FontSize',fst)
    grid on
    set(gca,'FontSize',fs-2)
end
fignm = ['Abs',num2str(procdata{1}.cancer_exp)];
h = gcf; set(h,'PaperOrientation','landscape','PaperUnits','normalized',...
    'PaperPosition',[0 0 1 1])
print(gcf,'-dtiffn',fignm)

%%
% Plot processed spectrum
figure
fs = 11;
fst = 12;
for zz = 1:4
    subplot(2,2,zz)
    plot(procdata{zz}.wa,procdata{zz}.procspec)
    xlabel('Wavelength (nm)','FontSize',fs)
    ylabel('Intensity','FontSize',fs)
    axis([300,1100,0,60000])
    title(char(c(zz)),'FontSize',fst)
    grid on
    set(gca,'FontSize',fs-2)
end
fignm = ['Int',num2str(procdata{1}.cancer_exp)];
h = gcf; set(h,'PaperOrientation','landscape','PaperUnits','normalized',...
    'PaperPosition',[0 0 1 1])
print(gcf,'-dtiffn',fignm)

```

This m-file is slightly different than the previous file. Parameters are set for processing images with the magnetic particles, which have a different file structure.

```

clear all, clc, close all
%global txt4 txt5
datapts = 2048;% Total # of wavelength points
vol_par = 750; % Particle volume added each time in micro-liters
int_rg = [450, 700]; % Range of integration for absorbance plots
% Magnetic Particle Spectrometer Ceff
    c = {'YFYM_spect', 'YFNM_spect', 'NFMNM_spect', 'NFYM_spect'};
    basenam = {'YFYM', 'YFNM', 'NFMNM', 'NFYM'};
    skpnm = 1; % Step size for collecting data
    skipdata = 15; % Points to skip due to initial mixing (1 = no skip)
    vol_tot = 0.5; % Total volume in liters
    num_tot = 1197; % Total # of samples+1
% Process data
for kk = 1:length(c)
    pathnm = ['E:\Jay_Wheaton\Wheaton_thesis\mivcf\web\public\',...
        'Final_thesis_experiments\Experiment4\Spectrometer\',...
        cell2mat(c(kk)), '\']; % Cancer Cells
    basenm = basenam(kk); % File basename, i.e. 'cal' for cal00123
    procdata{kk} = spect_tl_c2(datapts, num_tot, vol_tot, vol_par, int_rg, ...
        basenm, skipdata, pathnm, skpnm);
procdata{kk}.c = c{kk};
procdata{kk}.int_rg = int_rg;
end
%% Plot properties
%   Plots include:
%       'intensity'           all intensity data
%       'absorbance'         all absorbance data
%       'ab_int'             integrated absorbance
%       'capteff'            percent captured based on integrated abs.
%       'capteffpar'         #particles captured based on integrated abs.
cefffit(1) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
cefffit(2) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
cefffit(3) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
cefffit(4) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
cefffit(5) = 1; % 0 = no curve fit, 1 = exponential, 2 = linear
    %plots = {'intensity', 'absorbance', 'ab_int', 'capteff', 'capteffpar'};
plots = {'capteff'};
suball = 0; % Combine each scenario with subplots
color(1,:) = [1 0 0];
color(2,:) = [0 0.7 0];
color(3,:) = [0 0 1];
color(4,:) = [255/255 165/255 0/255];
color(5,:) = [0 1 1];

```

```

for kk = 1:length(procddata)
    procddata{kk}.color = color(kk,:);
    procddata{kk}.cefffit = cefffit(kk);
% Shorten the strings to be 4 dimensions. This is needed for the YFNM2
% case. I want that to show up as YFNM on the legend.
    tempstr = char(procddata{kk}.c);
    procddata{kk}.c = tempstr(1:4);
end
%% Plot data
plots = {'capteff'};
suball = 0; % Combine each scenario with subplots
for kk = 1:length(procddata)
d = procddata{kk}; % Compilation of data
    [p,f,cnst,r2_temp,cfiteq] = abs_plots2(d,procddata{kk}.c,...
        procddata{kk}.cefffit,procddata{kk}.int_rg,plots,suball,...
        procddata{kk}.color,kk);
leg1 = [char(procddata{kk}.c), ' at 0.3 L/min'];
procddata{kk}.p = p;
procddata{kk}.f = f;
procddata{kk}.cnst = cnst;
procddata{kk}.r2 = r2_temp;
procddata{kk}.cfiteq = cfiteq;
procddata{kk}.basenam = basenam(kk);
end
%% Plot all data w/ curve fits
%c = zeros(1,length(procddata))
figure
ms = 3;
for jj = 1:length(procddata)
H =plot(procddata{jj}.timeab,procddata{jj}.ceff,'o','MarkerFaceColor',...
    procddata{jj}.color,'MarkerEdgeColor',procddata{jj}.color,'MarkerSize',ms);
hold on
c{jj} = procddata{jj}.c;
end
fs = 14;
fst = 16;

a1 = legend(char(procddata{1}.basenam),char(procddata{2}.basenam),...
    char(procddata{3}.basenam),char(procddata{4}.basenam),...
    'Location','NorthWest');
a2 = get(a1,'children');
set(a2(1:3:end),'MarkerSize',ms)
grid minor

```

```

ti5 = ['Capture Efficiency Based on Integrated Absorbance from ',...
      num2str(procdata{1}.int_rg(1)), 'nm to ',...
      num2str(procdata{1}.int_rg(2)), 'nm'];
title(ti5,'FontSize',fst)
xlabel('Time (min)','FontSize',fs)
ylabel('Percent of Cells Captured (%)','FontSize',fs)
xmx = floor(max(procdata{1}.timeab))+5;
axis([0 xmx -5 100])

set(gca,'FontSize',fs,'XTick',[0:30:360])
% Plot best fit lines
for jj = 1:3:length(procdata)
f4 = procdata{jj}.cnst;
y4 = procdata{jj}.f;
plot(procdata{jj}.timeab,y4(procdata{jj}.timeab),'--','LineWidth',2.5,...
      'Color','k')
eqtot = [' ' procdata{jj}.cfiteq, ' ',procdata{jj}.r2, ' '];
text(200,85-15*jj,eqtot,'HorizontalAlignment','center',...
      'BackgroundColor',[1 1 1],'EdgeColor',procdata{jj}.color,...
      'FontSize',fs,'LineWidth',1.75);
end
fignm = ['Capt_Eff_particles_yy'];
h = gcf; set(h,'PaperOrientation','landscape','PaperUnits','normalized',...
      'PaperPosition',[0 0 1 1])
print(gcf,'-dpdf',fignm)
%%
% This is a dual y-axis plot that was commented out because the axes were
% the same
particles_in = 0.3*2.08*10^7/(10^6);
% 0.3ml * 2.08e7 particles/ml --> millions (divide by 1e6)
fs = 12;
fst = 14;
figure
[AX,H1,H2]=plotyy(procdata{1}.timeab,procdata{1}.ceff,...
      procdata{1}.timeab,procdata{1}.ceff/100*particles_in,'plot'); hold on
for jj = 2:length(procdata)
plot(procdata{jj}.timeab,procdata{jj}.ceff, '.', 'MarkerFaceColor',...
      color(jj,:), 'MarkerEdgeColor', color(jj,:), 'MarkerSize',8),hold on
end
set(AX,{'ycolor'},{'k';'k'})
set(H1,'LineStyle','.', 'MarkerFaceColor',color(1,:), 'MarkerEdgeColor',...
      color(1,:), 'MarkerSize',8)
set(H2, 'LineStyle','.', 'MarkerFaceColor',color(1,:), 'MarkerEdgeColor',...

```

```

        color(1,:), 'MarkerSize', 1)
a1 = legend(char(procdata{1}.basenam), char(procdata{2}.basenam), ...
            char(procdata{3}.basenam), char(procdata{4}.basenam), ...
            'Location', 'NorthWest');
a2 = get(a1, 'children');
set(a2(1:3:end), 'MarkerSize', 16)
grid minor
ti5 = ['Capture Efficiency at 0.3 L/min Based on',
      'Integrated Absorbance from ', num2str(int_rg(1)), 'nm to ', ...
      num2str(int_rg(2)), 'nm'];
title(ti5, 'FontSize', fst)
xlabel('Time (min)', 'FontSize', fs)
set(get(AX(1), 'Ylabel'), 'String', 'Percent of Particles Captured (%)', ...
    'FontSize', fs)
set(get(AX(2), 'Ylabel'), 'String', ...
    'Number of Particles Captured (Millions)', ...
    'VerticalAlignment', 'cap', 'Rotation', 270, 'FontSize', fs)
axis(AX(1), [0, 365, -5, 100])
axis(AX(2), [0, 365, -.05*particles_in, particles_in])
set(AX(2), 'XTickLabel', '')
set(gca, 'FontSize', fs, 'XTick', [0:30:360])
set(AX(1), 'YTick', [0:10:100], 'FontSize', fs)
set(AX(2), 'YTick', [0:0.5:10], 'FontSize', fs)
% Shift yy label
yylabel = get(AX(2), 'Ylabel');
set(yylabel, 'Position', get(yylabel, 'Position').*[1.01 1 1])
% Plot best fit lines
for jj = 1:3:length(procdata)
    f4 = procdata{jj}.cnst;
    y4 = procdata{jj}.f;
    plot(procdata{jj}.timeab, y4(procdata{jj}.timeab), '--', 'LineWidth', 2.5, ...
        'Color', 'k')
    eqtot = [' ' procdata{jj}.cfiteq, ' ', procdata{jj}.r2, ' '];
    text(220, 85-15*jj, eqtot, 'HorizontalAlignment', 'center', ...
        'BackgroundColor', [1 1 1], 'EdgeColor', procdata{jj}.color, ...
        'FontSize', fs, 'LineWidth', 1.75);
end
fignm = ['Capt_Eff_particles_yy'];
h = gcf; set(h, 'PaperOrientation', 'landscape', 'PaperUnits', 'normalized', ...
    'PaperPosition', [0 0 1 1])
print(gcf, '-dpdf', '-r300', fignm)
%%
% Plot raw absorbance

```

```

figure
fs = 11;
fst = 12;
for zz = 1:2
    subplot(1,2,zz)
    plot(procdata{zz}.wa,procdata{zz}.absorbance)
    xlabel('Wavelength (nm)','FontSize',fs)
    ylabel('Absorbance (OD)','FontSize',fs)
    axis([375,1000,-.01,0.025])
%     axis([375,1000,0,0.2])
nm{1} = ' 0.3LPM';
nm{2} = ' 2LPM';
    til = char(c(zz));
    ti = [til(1:4), nm{zz}];
    title(ti,'FontSize',fst)
    grid on
    set(gca,'FontSize',fs-2)
    axis square
end
fignm = ['Abs_mag_', '2LPM']; %num2str(procdata{1}.cancer_exp)];
h = gcf; set(h,'PaperOrientation','landscape','PaperUnits','normalized',...
    'PaperPosition',[0 0 1 1])
print(gcf,'-dtiffn',fignm)
%%
% Plot processed spectrum
figure
fs = 11;
fst = 12;
for zz = 1:2
    subplot(1,2,zz)
    plot(procdata{zz}.wa,procdata{zz}.procspec)
    xlabel('Wavelength (nm)','FontSize',fs)
    ylabel('Intensity','FontSize',fs)
    axis([300,1100,0,60000])
%     ti = char(c(zz));
%     title(ti(1:4),'FontSize',fst)
nm{1} = ' 0.3LPM';
nm{2} = ' 2LPM';
    til = char(c(zz));
    ti = [til(1:4), nm{zz}];
    title(ti,'FontSize',fst)
    grid on
    set(gca,'FontSize',fs-2)

```

```

        axis square
    end
    fignm = ['Int_mag_', '2LPM']; %num2str(procdata{1}.cancer_exp)];
    h = gcf; set(h, 'PaperOrientation', 'landscape', 'PaperUnits', 'normalized', ...
        'PaperPosition', [0 0 1 1])
    print(gcf, '-dtiffn', fignm)

```

The following function is called by both main m-files to process the images.

```

function [procdata] = spect_tl_c2(datapts,num_tot,vol_tot,vol_par,...
    int_rg,basenam,skipdata,pathnm,skpnm)
%spect_cal is used to input, analyze, and plot spectrometer data.
% This function is used to process spectrometer absorbance data.
% Files should be saved as ".jdx".
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Import Data %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    basenm = char(basenam); % basename for files
    cntrl = 1;
    for nn = skipdata:skpnm:num_tot
        % Correct the counter for file names
        if nn <=10
            nzero = '0000';
        elseif nn <=100
            nzero = '000';
        elseif nn<=1000
            nzero = '00';
        elseif nn <= 10000
            nzero = '0';
        else
            nzero = '';
        end
        % Read in calibration file named calfile
        calfile = [pathnm,basenm,nzero,num2str(nn-1),'.jdx'];
        %fid = fopen(calfile,'r');
    specdata = textread(calfile, '', 'delimiter', ',', 'commentstyle', 'shell');
        %fclose(fid);
        % Store wavelength and intensity data
        if nn ==skipdata
            reffile = [pathnm,'refspec.jdx'];
            fid = fopen(reffile,'r');
            reffdata = textread(reffile, '', 'delimiter', ',', 'commentstyle', 'shell');

```



```

        fclose(fid);
        darkfile = [pathnm,'darkspec.jdx'];
darkdata = textread(darkfile, '', 'delimiter', ',', 'commentstyle', 'shell');
wa(:,1) = specdata(1:datapts,1); % Wavelength (column 1), same for all data
darkspec(:,1) = darkdata(1:datapts,2); % Dark spectrum (col 2), same for all
refspec(:,1) = reldata(1:datapts,2); % Reference spect (col 2). same for all
    % Initial time
    FileInfo = dir(calfile);
    [Y1, M1, D1, H1, MN1, S1] = datevec(FileInfo.datenum);
    S(cntrl) = S1;
    t = 0;
end
    % Set time between each capture
    FileInfo = dir(calfile);
    [Y, M, D, H, MN, S(cntrl+1)] = datevec(FileInfo.datenum);
    if S(cntrl+1)-S(cntrl)>=0
        t = t+S(cntrl+1)-S(cntrl); % Seconds
    else
t = t+60+S(cntrl+1)-S(cntrl); % Correct for switch from i.e. 58 to 2 seconds
    end
        timeab(cntrl) = t/60; % Time in minutes
        procspec(:,cntrl) = specdata(1:datapts,2); % Processed spectrum (column 2)
        cntrl = cntrl+1;
end
msg1 = ['All files for ',basenm,' have been read successfully'];
disp(msg1)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Process Data %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for cntrl2 =1:length(procspec(1,:))
    diff(:,cntrl2) = (procspec(:,cntrl2)-darkspec)./(refspec-darkspec);
    absorbance(:,cntrl2) = real(-10*log10(diff(:,cntrl2)));
end
    % Compute Absorbance from intensity
    absorbance=real(-log10(diff));
    % Set time between captures
    %timeab_sec = 0:0.75:(num_tot-skipdata)*0.75-0.75; % Time in seconds
    %timeab = timeab_sec/60; % Time in minutes
percent_vol = vol_par/vol_tot; % percent particles : total volume
num_part = 2.08*10^7*vol_par*10^-3/vol_tot; % #particles per liter of fluid
    % Integrate absorbance data to get a single data point for absorbance
    for ii = 1:cntrl-1

```

```

        val5 = absorbance(:,ii); % absorbance at concentration # ii
        % Integrate data over spectrum specified at beginning of code
int_ab(ii) = trapz(wa(int_rg(1):int_rg(2),1),val5(int_rg(1):int_rg(2),1));
        ceff(ii) = 100*(1-(int_ab(ii)/(int_ab(1))));
        ceffpart(ii) = num_part*(1-(int_ab(ii)/(int_ab(1))));
    end
    ab_avg = mean(absorbance(:,1:cntrl-1));
% Store data in a single variable
    procdata.wa=wa;
    procdata.procspec=procspec;
    procdata.absorbance=absorbance;
    procdata.int_ab=int_ab;
    procdata.timeab=timeab;
    procdata.darkspec=darkspec;
    procdata.refspec=refspec;
    procdata.percent_vol=percent_vol;
    procdata.num_part=num_part;
    procdata.ab_avg=ab_avg;
    procdata.ceff=ceff;
    procdata.ceffp=ceffpart;
end

```

This code is called by `spect_tl_cancer2.m` in order to average spectrometer data that was collected using a different integration time.

```

function [procdata] = spect_tl_c3(datapts,num_tot,vol_tot,vol_par,...
    int_rg,basenam,skipdata,pathnm,skpnm)
%spect_cal is used to input, analyze, and plot spectrometer data.
% This function is used to process spectrometer absorbance data.
% Files should be saved as ".jdx".
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Import Data %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    basenm = char(basenam); % basename for files
    cntrl = 1;
for kk = skipdata:skpnm:num_tot
    procspec_temp = zeros(2048,1);
    for nn = kk:kk+7
        % Correct the counter for file names
        if nn <=10
            nzero = '0000';

```

```

elseif nn <=100
    nzero = '000';
elseif nn<=1000
    nzero = '00';
elseif nn <= 10000
    nzero = '0';
else
    nzero = '';
end
% Read in calibration file named calfile
calfile = [pathnm,basnm,nzero,num2str(nn-1),'.jdx'];
%fid = fopen(calfile,'r');
specdata = textread(calfile,'','delimiter',' ','commentstyle','shell');
%fclose(fid);
assignin('base','specdata',specdata)
% Store wavelength and intensity data
if nn ==skipdata
reffile = [pathnm,'refspec.jdx'];
refdata = textread(reffile,'','delimiter',' ','commentstyle','shell');
darkfile = [pathnm,'darkspec.jdx'];
darkdata = textread(darkfile,'','delimiter',' ','commentstyle','shell');
wa(:,1) = specdata(1:datapts,1); % Wavelength (column 1), same for all data
darkspec(:,1) = darkdata(1:datapts,2);% Dark spectrum (col 2), same for all
refspec(:,1) = refdata(1:datapts,2);% Reference spect (col 2) same for all
% Initial time
FileInfo = dir(calfile);
[Y1, M1, D1, H1, MN1, S1] = datevec(FileInfo.datenum);
S(cntrl) = S1;
t = 0;
end
% Sum 7 spectrum data points @ 100ms + 1/2 of the 8th point.
% This is equivalent to 750ms integration.
if nn == kk+7
    % Half of the Processed spectrum
    procspec_temp = procspec_temp+0.5*specdata(1:datapts,2);
else
    % Processed spectrum (column 2)
    procspec_temp = procspec_temp+specdata(1:datapts,2);
end
end
procspec(:,cntrl) = procspec_temp;
assignin('base','procspec',procspec)
% Set time between each capture

```

```

FileInfo = dir(calfile);
[Y, M, D, H, MN, S(cntrl+1)] = datevec(FileInfo.datenum);
if S(cntrl+1)-S(cntrl)>=0
    t = t+S(cntrl+1)-S(cntrl);% Seconds
else
    % Correct for switch from i.e. 58 to 2 seconds
    t = t+60+S(cntrl+1)-S(cntrl);
end
timeab(cntrl) = t/60;% Time in minutes
% calculate absorbance from intensity
cntrl = cntrl+1;
end
msg1 = ['All files for ',basenm,' have been read successfully'];
disp(msg1)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Process Data %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for cntr2 =1:length(procspec(1,:))
diff(:,cntr2) = abs((procspec(:,cntr2)-darkspec))./abs((refspec-darkspec));
absorbance(:,cntr2) = real(-log10(diff(:,cntr2)));
end
    assignin('base','diff',diff)
    % Compute Absorbance from intensity
%     absorbance=real(-10*log10(diff));
    % Set time between captures
    %timeab_sec = 0:0.75:(num_tot-skipdata)*0.75-0.75; % Time in seconds
    %timeab = timeab_sec/60; % Time in minutes
percent_vol = vol_par/vol_tot; % percent particles : total volume
num_part = 2.08*10^7*vol_par*10^-3/vol_tot; % #particles per liter of fluid
    % Integrate absorbance data to get a single data point for absorbance
    for ii = 1:cntrl-1
        val5 = absorbance(:,ii); % absorbance at concentration # ii
        % Integrate data over spectrum specified at begining of code
int_ab(ii) = trapz(wa(int_rg(1):int_rg(2),1),val5(int_rg(1):int_rg(2),1));
        ceff(ii) = 100*(1-(int_ab(ii)/(int_ab(1))));
        ceffpart(ii) = num_part*(1-(int_ab(ii)/(int_ab(1))));
    end
    ab_avg = mean(absorbance(:,1:cntrl-1));
% Store data in a single variable
procdata.wa=wa;
procdata.procspec=procspec;
procdata.absorbance=absorbance;

```

```

    procdata.int_ab=int_ab;
    procdata.timeab=timeab;
    procdata.darkspec=darkspec;
    procdata.refspec=refspec;
    procdata.percent_vol=percent_vol;
    procdata.num_part=num_part;
    procdata.ab_avg=ab_avg;
    procdata.ceff=ceff;
    procdata.ceffp=ceffp;
end

```

The following function is used to plot processed images. There are several options available.

```

function [p2,y4,f4,r2_temp,cfiteq] = abs_plots2(d,basenam,cefffit,...
    int_rg,plots,suball,pcol,cntr)
%abs_plots Create Plots for particle capture with spectrometer
%global txt4 txt5

wa = d.wa;
procspec = d.procspec;
absorbance = d.absorbance;
int_ab = d.int_ab;
timeab = d.timeab;
ceff = d.ceff;
ceffp = d.ceffp;
refspec = d.refspec;
darkspec = d.darkspec;
p2=size(plots,2);
    basenm = char(basenam); % basename for files
for kk =1:p2
    mm = char(plots{kk}); % choose what plots to output
switch mm
% Plot intensity over entire spectrum
    case 'intensity'
figure
plot(wa,procspec,wa,refspec,wa,darkspec)
    axis([360,1000,0,60000])
    til = ['Spectrometer Intensity for ',basenm];
    title(til)
    xlabel('Wavelength (nm)')

```

```

ylabel('Intensity (counts)')
grid minor
if suball ==1
    figure(21)
    plot(wa,procspec,wa,refspec,wa,darkspec), hold on
    axis([360,1000,0,60000])
    title(ti1)
    xlabel('Wavelength (nm)')
    ylabel('Intensity (counts)')
    grid minor
end
% Plot absorbance over entire spectrum
case 'absorbance'
figure
plot(wa,absorbance)
axis([360,1000,-0.001,0.3])
ti2 = ['Spectrometer Absorbance for ',basenm];
title(ti2)
xlabel('Wavelength (nm)')
ylabel('Absorbance (OD)')
grid minor
if suball ==1
    figure(22)
    plot(wa,absorbance),hold on
    axis([360,1000,-0.001,0.3])
    title(ti2)
    xlabel('Wavelength (nm)')
    ylabel('Absorbance (OD)')
    grid minor
end
% Plot absorbance integration over specified range
case 'ab_int'
    xlab = 'Time (min)';
figure
plot(timeab,int_ab,'o','MarkerFaceColor',pcol,...
    'MarkerEdgeColor',pcol,'MarkerSize',3)
ax1 = 1.1*max(timeab);
ax2 = 1.1*max(int_ab);
axis([-0.001,ax1,-0.001,ax2])
ti4 = ['Integrated Spectrometer Absorbance from ',...
    num2str(int_rg(1)),'nm to ',num2str(int_rg(2)),'nm'];
title(ti4)
xlabel(xlab)

```

```

ylabel('Integrated Absorbance (OD*nm)')
grid minor
if suball ==1
    figure(23)
    plot(timeab,int_ab,'o','MarkerFaceColor',pcol,...
        'MarkerEdgeColor',pcol,'MarkerSize',3), hold on
    axis([-0.001,ax1,-0.001,ax2])
    title(ti4)
    xlabel(xlab)
    ylabel('Integrated Absorbance (OD*nm)')
    grid minor
end
% Plot percent capture efficiency over time
case 'capteff'
    xlab = 'Time (min)';
    txt4x = 0.7*max(timeab');
    txt4y = 0.6*max(ceff);
    txt5y = 0.5*max(ceff);

figure
plot(timeab,ceff,'o','MarkerFaceColor',pcol,...
    'MarkerEdgeColor',pcol,'MarkerSize',5), hold on
% Curve Fit
if cefffitt == 1
    % y-intercept set at zero
    ft4 = fitttype('A*(1-exp(B*x))','coeff',{'A','B'});
    [f4,g4] = fit(timeab',ceff',ft4,'start',[100,-0.05]);
    Aco = num2str(floor(f4.A*100000)/100000);
    Bco = num2str(floor(f4.B*100000)/100000);
    rsqd = num2str(floor(g4.rsquare*100000)/100000);
    y4 = @(x) f4.A *(1-exp(f4.B*x));
    plot(timeab,y4(timeab'),'r--','LineWidth',1.5)
    txt4 = ['R^2 = ', rsqd(1:5)];
    r2_temp = txt4;
    txt5 = ['y = ',Aco(1:6),'*(1-exp(',Bco(1:6),'*x))'];
    cfiteq=txt5;
    text(txt4x,txt4y,txt4,'HorizontalAlignment','center',...
        'BackgroundColor',[1 1 1],'EdgeColor',[1 0 0]);
    text(txt4x,txt5y,txt5,'HorizontalAlignment','center',...
        'BackgroundColor',[1 1 1],'EdgeColor',[1 0 0]);
else
    y4='No curve fit applied';
end
% Axis and labels

```

```

ax1 = 1.1*max(timeab);
axis([-0.001,ax1,-5,115])
ti5 = ['Capture Efficiency for ',basenm,...
      ' Based on Integrated Absorbance from ',num2str(int_rg(1)),...
      'nm to ',num2str(int_rg(2)),'nm'];
title(ti5)
xlabel(xlab)
ylabel('Capture Efficiency (Percent of Particles Captured)')
grid minor
if suball ==1
    figure(24)
    plot(timeab,ceff,'o','MarkerFaceColor',pcol,...
          'MarkerEdgeColor',pcol,'MarkerSize',3), hold on
    axis([-0.001,ax1,-5,115])
    title(ti5)
    xlabel(xlab)
    ylabel('Capture Efficiency (Percent of Particles Captured)')
    grid minor
end
% Plot percent capture efficiency over time
case 'capteffpar'
    xlab = 'Time (min)';
    txt4x = 0.7*max(timeab');
    txt4y = 0.6*max(ceffp);
    txt5y = 0.5*max(ceffp);
figure
    plot(timeab,ceffp,'o','MarkerFaceColor',pcol,...
          'MarkerEdgeColor',pcol,'MarkerSize',5), hold on
    % Axis and labels
    ax1 = 1.1*max(timeab);
    axis([-0.001,ax1,-1,10^7])
    ti5 = ['Capture Efficiency Based on Integrated Absorbance from ',...
          num2str(int_rg(1)),'nm to ',num2str(int_rg(2)),'nm'];
    title(ti5)
    xlabel(xlab)
    ylabel('Capture Efficiency (# of Particles)')
    grid minor
otherwise
    disp('No plots choosen for abs_plots function')
end
end
end

```


Appendix D

Matlab Image Processing Code

Figure D.1: Matlab image processing file structure.

D.1 Image intensity main GUI

This code corresponds to a .fig file that is created using Matlab's GUIDE tool. It is used to call all relevant image processing sub-routines.

```

function varargout = intensity_calc2(varargin)
% INTENSITY_CALC2 MATLAB code for intensity_calc2.fig
%     INTENSITY_CALC2, by itself, creates a new INTENSITY_CALC2 or raises
%     the existing singleton*.
%
%     H = INTENSITY_CALC2 returns the handle to a new INTENSITY_CALC2 or
%     the handle to the existing singleton*.
%
%     INTENSITY_CALC2('CALLBACK',hObject,eventData,handles,...) calls the
%     local function named CALLBACK in INTENSITY_CALC2.M with the given
%     input arguments.
%
%     INTENSITY_CALC2('Property','Value',...) creates a new
%     INTENSITY_CALC2 or raises the existing singleton*. Starting from
%     the left, property value pairs are applied to the GUI before
%     intensity_calc2_OpeningFcn gets called. An unrecognized property

```

```

%      name or invalid value makes property application stop.
%      All inputs are passed to intensity_calc2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help intensity_calc2

% Last Modified by GUIDE v2.5 21-Oct-2012 17:30:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @intensity_calc2_OpeningFcn, ...
                  'gui_OutputFcn',  @intensity_calc2_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% --- Executes just before intensity_calc2 is made visible.
function intensity_calc2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to intensity_calc2 (see VARARGIN)

% Choose defaults command line output for intensity_calc2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

```

```

% UIWAIT makes intensity_calc2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = intensity_calc2_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)

% End initialization code - DO NOT EDIT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Load Content %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes on button press in defvals.
function defvals_Callback(hObject, eventdata, handles)
imlev = 0.1;
impath=(['E:\Jay_Wheaton\Wheaton_thesis\mivcf\web\public\'',...
        'Final_thesis_experiments\Cancer_Cells\Images\']);
imfolder = 'YFYM_images\'; % Image folder
imstart = 37; % First image #
imskip = 1; % Step size for image counter
imend = 284; % Last image #
    set(handles.edit2,'String',impath)
    set(handles.editYFYM,'String',imfolder)
    set(handles.edit4,'String',imstart)
    set(handles.edit5,'String',imskip)
    set(handles.edit6,'String',imend)
    set(handles.edit7,'String',imlev)
    set(handles.editYFYM,'String','YFYM\')
    set(handles.editYFNM,'String','YFNM\')
    set(handles.editNFYM,'String','NFYM\')
    set(handles.editNFMN,'String','NFMN\')
    set(handles.editYFYM2,'String','YFYM2LPM\')
    set(handles.initim,'String','DSC_0027.jpg')

% --- Executes on button press in defmac.
function defmac_Callback(hObject, eventdata, handles)
imlev = 0.1;

```

```

impath=('/Users/Jay/Desktop/backup/');
imfolder = 'YFYM_images/'; % Image folder
imstart = 3; % First image #
imskip = 1; % Step size for image counter
imend = 5; % Last image #
    set(handles.edit2,'String',impath)
    set(handles.editYFYM,'String',imfolder)
    set(handles.edit4,'String',imstart)
    set(handles.edit5,'String',imskip)
    set(handles.edit6,'String',imend)
    set(handles.edit7,'String',imlev)
    set(handles.editYFYM,'String','YFYM_images/')
    set(handles.editYFNM,'String','YFNM_images/')
    set(handles.editNFYM,'String','NFYM_images/')
    set(handles.editNFMN,'String','NFMN_images/')
    set(handles.editYFYM2,'String','YFYM2LPM_images/')

% --- Executes on button press in imloadadj.
function imloadadj_Callback(hObject, eventdata, handles)

    procopt(1,1) = {get(handles.yfym,'Value')};
    procopt(1,2) = {get(handles.editYFYM,'String')};
    procopt(1,3) = {'YFYM'};
    procopt(2,1) = {get(handles.yfnm,'Value')};
    procopt(2,2) = {get(handles.editYFNM,'String')};
    procopt(2,3) = {'YFNM'};
    procopt(3,1) = {get(handles.nfym,'Value')};
    procopt(3,2) = {get(handles.editNFYM,'String')};
    procopt(3,3) = {'NFYM'};
    procopt(4,1) = {get(handles.nfnm,'Value')};
    procopt(4,2) = {get(handles.editNFMN,'String')};
    procopt(4,3) = {'NFMN'};
    procopt(5,1) = {get(handles.yfym2,'Value')};
    procopt(5,2) = {get(handles.editYFYM2,'String')};
    procopt(5,3) = {'YFYM 2LPM'};
    assignin('base','procopt',procopt);
    impath = get(handles.edit2,'String');
    imstart = str2num(get(handles.edit4,'String'));
    imskip = str2num(get(handles.edit5,'String'));
    imend = str2num(get(handles.edit6,'String'));
    imlev = str2num(get(handles.edit7,'String'));
    maskim = get(handles.mim,'String');
    initim = get(handles.initim,'String'); % Initial image for timing

```

[illegible]

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes on button press in plotimint.
function plotimint_Callback(hObject, eventdata, handles)
int_calc = evalin('base','int_calc'); % intensity summation
procopt = evalin('base','procopt');
imparam = evalin('base','imparam');
plot_int(int_calc,procopt,imparam);
% --- Executes on button press in plot_mask.
function plot_mask_Callback(hObject, eventdata, handles)
imparam = evalin('base','imparam');
procopt = evalin('base','procopt');
immask = evalin('base','immask');
impath=imparam.impath;
maskim=imparam.maskim;
popt = 'both';
pltmask(immask,procopt,impath,maskim,popt)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Other Content %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function editYFYM_Callback(hObject, eventdata, handles)
function editYFYM_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit4_Callback(hObject, eventdata, handles)
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit5_Callback(hObject, eventdata, handles)
function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUiControlBackgroundColor'))

```

```

        set(hObject,'BackgroundColor','white');
    end
    function edit6_Callback(hObject, eventdata, handles)
    function edit6_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    function edit7_Callback(hObject, eventdata, handles)
    function edit7_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    function yfym_Callback(hObject, eventdata, handles)
    function nfnm_Callback(hObject, eventdata, handles)
    function editNFMN_Callback(hObject, eventdata, handles)
    function editNFMN_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    function nfym_Callback(hObject, eventdata, handles)
    function editNFYM_Callback(hObject, eventdata, handles).
    function editNFYM_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    function yfnm_Callback(hObject, eventdata, handles)
    function editYFNM_Callback(hObject, eventdata, handles)
    function editYFNM_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    function clcbutton_Callback(hObject, eventdata, handles)
    evalin('base','clc')
    function clearbutton_Callback(hObject, eventdata, handles)
    evalin('base','clear')
    function clbutton_Callback(hObject, eventdata, handles)
    evalin('base','close all')
    function mim_Callback(hObject, eventdata, handles)

```

```

function mim_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function pt_Callback(hObject, eventdata, handles)
function pt_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function maskopt_Callback(hObject, eventdata, handles)
function yfym2_Callback(hObject, eventdata, handles)
function editYFYM2_Callback(hObject, eventdata, handles)
function editYFYM2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function initim_Callback(hObject, eventdata, handles)
function initim_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

D.2 Image integration

The following m-file is used to integrate the images.

```

function [i_calc,immask] = imageint2(procopt,impath,imstart,imskip,...
    imend,imlev,maskim,pt,maskopt,imparam)
proctype = pt;
lp = length(procopt(:,1));
% Create a mask
for nn = 1:lp
if maskopt ==1;
    if cell2mat(procopt(nn,1)) ==1
        mimage = [impath,char(procopt(nn,2)),maskim];
        if nn==1

```



```

        mopt = '0';
    else
        pltmask(immask,procopt,impath,mimage,'masked')
mopt = inputdlg(['Do you want to create a new mask or use a ',...
    'previous one? (0 = new or enter the mask number you want to use)']);
        close figure 1
    end

    if str2num(char(mopt)) ==0
        immask(nn) = {jmask(mimage)};
    else
        immask(nn) = immask(str2num(char(mopt)));
    end

end

else
    immask(nn) = {ones(3072,4608)};
end
end
%% Process image
tic
for ii = 1:lp
    proc_op = cell2mat(procopt(ii,1));
    if proc_op ==1
        imfolder = char(procopt(ii,2));
        for kk = imstart:imskip:imend
            % Correct the counter for images
            if kk<=9
                nz = '000';
            elseif kk<=99
                nz = '00';
            elseif kk<=999
                nz = '0';
            else
                nz = '';
            end
            %Set name for current image
            imname=[impath,imfolder,'DSC_',nz,num2str(kk),'.JPG'];
            % Open current image
            imcurrent = imread(imname,'jpg');
            tm = int_time(imfolder,imparam,imname);
            itm(kk,ii) = tm; % minutes between images
            imnum = ['Image ',num2str(kk)];

```

```

switch proctype
    case 'gray'
        % Convert RGB to Grayscale intensity
        imp = im2double(rgb2gray(imcurrent));
        imp2 = cell2mat(immask(ii)).*imp;
        int_calc_gray(kk,ii) = sum(sum(imp2));
    case 'bw'
        % Convert RGB to BW to Double w/ specified level
        imp = im2double(im2bw(imcurrent,imlev));
        imp2 = cell2mat(immask(ii)).*imp;
        int_calc_bw(kk,ii) = sum(sum(imp2));
    case 'bwgray'
        % Convert RGB to Grayscale intensity
        imp1 = im2double(rgb2gray(imcurrent));
        imp2 = cell2mat(immask(ii)).*imp1;
        int_calc_gray(kk,ii) = sum(sum(imp2));
        % Convert RGB to BW to Double w/ specified level
        imp3 = im2double(im2bw(imcurrent,imlev));
        imp4 = cell2mat(immask(ii)).*imp3;
        int_calc_bw(kk,ii) = sum(sum(imp4));
    end
end
print1 = ['The ',char(procopt(ii,3)), ' images have been processed!'];
disp(print1);
end
end

switch proctype
    case 'gray'
        i_calc{1,1} = int_calc_gray;
        i_calc{1,3} = itm;
    case 'bw'
        i_calc{1,2} = int_calc_bw;
        i_calc{1,3} = itm;
    case 'bwgray'
        %Grayscale
        i_calc{1,1} = int_calc_gray;
        %BW
        i_calc{1,2} = int_calc_bw;
        % Serial Date of Image
        i_calc{1,3} = itm;
    end
end

toc
end

```

This code determines the exact time between each image based on the file information.

```
function [tm] = int_time(imfolder,imparam,imname)
%imfolder = char(procopt(qq,2));

% Find initial image info
    %Set name for initial image
    imname0=[imparam.impath,imfolder,imparam.initim];
    iminfo0 = iminfo(imname0,'jpg'); % Get info on image
    imd0 = iminfo0.FileModDate;
    imdt0 = datenum(imd0); % Initial image serial date
    imdate0 = datevec(imdt0,'mmm dd, yyyy HH:MM:SS');
% Find current image info
    %Set name for current image
    imname1=[imparam.impath,imfolder,imparam.initim];
    iminfo1 = iminfo(imname,'jpg'); % Get info on image
    imd1 = iminfo1.FileModDate;
    imdt1 = datenum(imd1); % Current image serial date
    imdate1 = datevec(imdt1);
% Compute time between images

tm = etime(imdate1,imdate0)/60; %Time between in minutes
end
```

This function applies a mask to an image based on user specified shapes.

```
function [immask] = jmask(imcurrent)
figure
imshow(imcurrent),hold on
imcurrent2 =imread(imcurrent);
immasktyp = inputdlg(['Mask with polygon lines or rectangles?','...
'(poly/rect) NOTE: Double Click to create mask!'],'Mask type',1,{'rect'});
switch char(immasktyp)
    case{'rect'}
        h = imrect;
        BW = createMask(h);
        pause_fig = wait(h);
    case{'poly'}
        nn=1;
        h=1;
        while h~=0
```

```

        BW1.mask{nn} = roipoly(imcurrent2);
h = str2double(inputdlg...
    ('Do you want to create another polygon? (1=yes,0=no)'));
    nn=nn+1;
end
BW = zeros(size(BW1.mask(1)));
for mm =1:length(BW1.mask)
    BW = BW+BW1.mask{mm};
end
BW = BW/mm;
otherwise
    disp('You did not pick the type of mask!')
end
immask = BW;
[m,n] = size(immask);
immaskrgb = zeros(m,n,3);
immaskrgb(:, :, 1) = uint8(immask);
immaskrgb(:, :, 2) = uint8(immask);
immaskrgb(:, :, 3) = uint8(immask);
immaskrgb = uint8(immaskrgb);
imshow(immaskrgb.*imcurrent2)
h2 = msgbox('Are you ready to continue?');
uiwait(h2)
close
pause(0.00001)
end

```

D.3 Post-processing plots

This function plots the results from image intensity integration.

```

function plot_int(int_calc,procopt,imparam)
% Determine which plots to use
proctype = imparam.pt;
% Plot colors
color1(5,:) = [0 1 1];
color1(4,:) = [0 0 1]; % NFNM
color1(3,:) = [0 0.7 0]; % NFYM
color1(2,:) = [255/255 165/255 0/255];% YFNM
color1(1,:) = [1 0 0];% YFYM

```

```

icgray = int_calc{1};
icbw = int_calc{2};
ictime = int_calc{3};

switch proctype
    case 'gray'
        nend = length(icgray(1,:));
    case 'bw'
        nend = length(icbw(1,:));
    case 'bwgray'
        nend = length(icgray(1,:));
end
nsum = sum(cell2mat(procopt(:,1)));
% Setup subplot dimensions
lp = length(procopt(:,1));
if nsum<4 && nend <=4
    p=1; a = 1; b = nsum; c = 0;
elseif nsum==4 && nend ==4
    p=1; a = 2; b = 2; c = 0;
elseif nend >5 && nend<8
    p=2; a = 1; b = nsum/2; c = 0;
else
    p=2; a = 2; b =2; c = 1;
end
% Plot results
pt = imparam.pt;
dblpltopt = 0;
nn=1;
figure(1), hold on
for qq = 1:4
    if cell2mat(procopt(qq,1))==1
        %subplot(a,b,nn)
        switch pt
            case 'bw'
                for vv = imparam.imstart:imparam.imskip:imparam.imend
                    normval = max(max(icbw(:,qq)));
                    plot(ictime(vv,qq),icbw(vv,qq)/normval,'Marker','.',...
                        'MarkerEdgeColor',color1(qq,:), 'MarkerFaceColor',...
                        color1(qq,:), 'MarkerSize',6), hold on
                end
            t = [procopt(qq,3),...
                'Normalized Capture Efficiency From Black and White Image Intensity'];

```

```

        title(t)
        grid on
        xlabel('Time (min)')
        ylabel('Normalized Image Intensity')
        axis([0 imparam.imend 0 1.1])
        case 'gray'
        for vv = imparam.imstart:imparam.imskip:imparam.imend
            normval = max(max(icgray(:,qq)));
            plot(ictime(vv,qq),icgray(vv,qq)/normval,'Marker','.',...
                'MarkerEdgeColor',color1(qq,:), 'MarkerFaceColor',...
                color1(qq,:), 'MarkerSize',6), hold on
        end
t = [procopt(qq,3),...
    'Normalized Capture Efficiency From Grayscale Image Intensity'];
        title(t)
        grid on
        xlabel('Time (min)')
        ylabel('Normalized Image Intensity')
        axis([0 imparam.imend 0 1.1])
        otherwise
        for vv = imparam.imstart:imparam.imskip:imparam.imend
            normval = max(max(icbw(:,qq)));
            plot(ictime(vv,qq),icbw(vv,qq)/normval,'Marker','.',...
                'MarkerEdgeColor',color1(qq,:), 'MarkerFaceColor',...
                color1(qq,:), 'MarkerSize',6), hold on
        end
t = [procopt(qq,3),...
    'Normalized Capture Efficiency From Black and White Image Intensity'];
        title(t)
        grid on
        xlabel('Time (min)')
        ylabel('Normalized Image Intensity')
        axis([0 imparam.imend 0 1.1])
        dblpltopt = 1;
    end
    nn=nn+1;
end
end

% Plot gray for all four cases when pt = bwgray
if dblpltopt ==1;
    figure
    fs = 12;

```

```

fst = 14;
for qq =1:4
    if cell2mat(procopt(qq,1))==1

        normval = max(icgray(:,qq));
        plot(ictime(:,qq),icgray(:,qq)./normval,'o',...
            'MarkerEdgeColor',color1(qq,:), 'MarkerFaceColor',...
            color1(qq,:), 'MarkerSize',5), hold on
t = ['Normalized Capture Efficiency From Grayscale Image Intensity'];
    title(t,'FontSize',fst)

    xlabel('Time (min)', 'FontSize',fs)
    ylabel('Normalized Image Intensity','FontSize',fs)
    axis([0 max(max(ictime))*1.1 0.7 1.1])
    end
end
legend(procopt{1,3},procopt{2,3},procopt{3,3},procopt{4,3},...
    'Location','NorthWest','FontSize',fs);
grid minor
set(gca,'FontSize',fs,'XTick',[0:30:240])
legend('Location','NorthWest')
usernum = cell2mat(inputdlg('What number do you want to save as?'));
fignm = ['Capt_Eff_Gray_CancerImage',usernum,'_',date];
h = gcf; set(h,'PaperOrientation','landscape','PaperUnits','normalized',...
    'PaperPosition',[0 0 1 1])
print(gcf,'-dpdf',fignm)
end

% Plot the YFYM 2LPM Case
qq=5;

if cell2mat(procopt(qq,1))==1
    figure
    switch pt
        case 'bw'
            for vv = imparam.imstart:imparam.imskip:imparam.imend
                normval = max(max(icbw(:,qq)));
                plot(ictime(vv,qq),icbw(vv,qq)/normval,'Marker','.',...
                    'MarkerEdgeColor',color1(qq,:), 'MarkerFaceColor',...
                    color1(qq,:), 'MarkerSize',6), hold on
            end
t = [procopt(qq,3),...
    'Normalized Capture Efficiency From Black and White Image Intensity'];

```

```

title(t)
grid on
xlabel('Time (min)')
ylabel('Normalized Image Intensity')
axis([0 imparam.imend 0 1.1])
case 'gray'
for vv = imparam.imstart:imparam.imskip:imparam.imend
normval = max(max(icgray(:,qq)));
plot(ictime(vv,qq),icgray(vv,qq)/normval,'Marker','.',...
      'MarkerEdgeColor',color1(qq,:), 'MarkerFaceColor',...
      color1(qq,:), 'MarkerSize',6), hold on
end
t = [procopt(qq,3),...
     'Normalized Capture Efficiency From Grayscale Image Intensity'];
title(t)
grid on
xlabel('Time (min)')
ylabel('Normalized Image Intensity')
axis([0 imparam.imend 0 1.1])
otherwise
for vv = imparam.imstart:imparam.imskip:imparam.imend
normval = max(max(icbw(:,qq)));
plot(ictime(vv,qq),icbw(vv,qq)/normval,'Marker','.',...
      'MarkerEdgeColor',color1(qq,:), 'MarkerFaceColor',...
      color1(qq,:), 'MarkerSize',6), hold on
end
t = [procopt(qq,3),...
     'Normalized Capture Efficiency From Black and White Image Intensity'];
title(t)
grid on
xlabel('Time (min)')
ylabel('Normalized Image Intensity')
axis([0 imparam.imend 0 1.1])
figure
for vv = imparam.imstart:imparam.imskip:imparam.imend
normval = max(max(icgray(:,qq)));
plot(ictime(vv,qq),icgray(vv,qq)/normval,'Marker','.',...
      'MarkerEdgeColor',color1(qq,:), 'MarkerFaceColor',...
      color1(qq,:), 'MarkerSize',6), hold on
end
t = [procopt(qq,3),...
     'Normalized Capture Efficiency From Grayscale Image Intensity'];
title(t)

```



```
        im4(:, :, nn) = 255*(1-m1)+im3(:, :, nn);
    end
    figure
    imshow(uint8(im4));
    t2 = [char(procopt(kk,3)), ' Masked Image'];
    title(t2)
    assignin('base', 'im2', im2);
    assignin('base', 'im3', im3);
    assignin('base', 'im4', im4);
end
end
end
```

Appendix E

Spectrometer Tutorial

The absorbance measurement software is relatively simple to setup. This tutorial gives a brief overview of the absorbance measurement but does not encompass some of the more advanced features in the software. It is helpful to calibrate using a reference medium before each capture with the Reference Monitor option but this is not always possible. The medium must be swappable, for instance using a cuvette. The in-line flow measurements in this thesis can only be calibrated at the beginning therefore the reference monitor option is not shown. See the actual software user manual for a more detailed description.

(E.1a) Light Source

(E.1b) Variable Attenuator

(E.1c) Spectrometer

Figure E.1: Spectrometer components

1. Open the Ocean Optics Spectrasuite Software and select **New** $\Rightarrow$ **Absorbance Measurement**

Figure E.2: Begin new absorbance measurement

2. Select the spectrometer and click **Next**

Figure E.3: Select spectrometer

3. Set the **Integration Time** so that the **Last peak value** is near the **Recommended peak value**. After the peak value is met, adjust the smoothing (if desired) using **Scans to Average**. Occasionally the capture time will be doubled if integration is used. Check the bottom right corner to see if the capture time is correct. If it is wrong, move on and adjust later. Select **Next** to continue. *Note: These settings can be adjusted later.*

Figure E.4: Set spectrometer absorbance acquisition parameters

4. Allow the spectrometer to capture at least once before obtaining the Reference Spectrum. Select the **light bulb** to store the Reference Spectrum. It is a good idea to visually compare a couple captures in the Reference Spectrum Preview window to make sure they are close. Select **Next** to continue. *Note: The Reference Spectrum is stored but not actually saved.*

Figure E.5: Capture reference spectrum

5. Block the light using the Variable Attenuator rather than turning off the light source. Allow the spectrometer to capture at least once before obtaining the Dark Spectrum. Select the **light bulb** to store the Dark Spectrum. It is a good idea to visually compare a couple captures in the Dark Spectrum Preview window to make sure they are close. Select **Finish** to continue. *Note: The Dark Spectrum is stored but not actually saved.*

Figure E.6: Capture dark spectrum

6. Absorbance with the Variable Attenuator set to block all light is shown below for reference.

Figure E.7: Sample spectrum with no light

7. Absorbance with the Variable Attenuator set to pass all light is shown below for reference. If the spectrum is not centered around zero then a new reference spectrum should be obtained. This can be done simply by clicking the yellow lightbulb, located near the middle of the icons. *Note: The Reference Spectrum is stored but not actually saved.*

Figure E.8: Sample spectrum with maximum light applied

8. A sample absorbance spectrum with fluid is shown below. Automatic saving can be setup by going to **File** ⇒ **Save** ⇒ **Save Spectrum**.

Figure E.9: Setup automatic saving

9. Select the spectrometer (make sure it is in Absorbance Mode). Set the Save Options and File Options. It is best to save as JCAMP-DX because this is a text file that contains the reference spectrum, dark spectrum, and processed spectrum with several decimal points. The values are saved as intensity rather than absorbance but a simple equation relates the two. Select **Accept** to continue.

Figure E.10: Automatic save settings

10. In order to begin capture, select **File** $\Rightarrow$ **Pause/Resume Export**. *Note: This step is only necessary if you selected **Pause until started by user** in the save settings.*

Figure E.11: Begin automatic capture

11. Choose the paused item and select **Resume** to begin automatic capture then **Close The Window**.

Figure E.12: Capture dark spectrum

12. The current spectrum can be stored for visual comparison at any time during data collection. Select the icon with a graph and pencil to freeze the spectrum on the screen (see figure below). *Note: The stored spectrum is displayed but not saved as a file.*

Figure E.13: Sample overlay spectrum

Glossary

Beer-Lambert law Correlation between light absorbance and solution mixture in terms of particle concentration. 56

Circulating Tumor Cell A rare cell derived from a primary tumor that can spread carcinoma and form secondary tumor(s). 1

Hydrodynamic Drag Force F_D The opposing force exerted on a spherical particle due to motion in a viscous fluid. vii, 11

Inferior Vena Cava Filter A filter inserted by catheter used to prevent pulmonary embolism. 1

Magnetization Force F_m The force exerted on a magnetic particle exposed to an external magnetic field based on the magnetic-dipole formulation. vii, 11

Tetrameric Antibody Complex A binding agent between carcinoma cells and dextran coated magnetic nanoparticles. 3