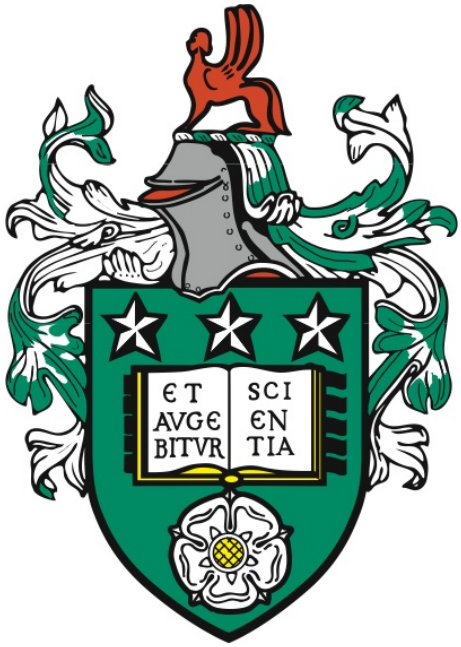


Fluid Mechanics of the Inspection of Pre-filled Medical Syringes

Hamza Liaquet

A thesis submitted in accordance with the requirements for the degree of

Doctor of Philosophy

The University of Leeds

School of Mechanical Engineering
School of Computing
October 2025

Intellectual Property and Publication Statements

I confirm that the work submitted is my own and that appropriate credit has been given where reference has been made to the work of others.

Acknowledgements

I would like to express my deepest gratitude to all supervisors, Professor Nikil Kapur, Dr Greg de Boer, and Dr Cédric Beaume for their continuous guidance, encouragement, and expertise throughout this research. Their insight and support have been instrumental in shaping both the direction and the quality of this work. I am especially grateful for their patience, constructive feedback and the balance of academic freedom and mentorship that they provided, which allowed me to grow as an independent researcher. Their combined expertise in computational modelling, fluid mechanics, and applied mathematics has deeply influenced my understanding and approach to this study.

I would also like to extend my sincere thanks to Christine Gardner and Paul Kinsey for their valuable discussions, insight, and support, which contributed meaningfully to the development of this work.

Finally, I would like to thank the University of Leeds for providing the facilities and academic environment that made this research possible.

I acknowledge the use of ChatGPT-5.0 (Open AI, https://chat.openai.com/) to correct grammar and proof read as well as develop codes used within the thesis.

Abstract

In this thesis, the process of the inspection of pre-filled syringes (PFS) is explored in the context of fluid flow and particle motion. The inspection of PFS is a a key stage in ensuring drug safety and quality. This work combines computational modelling with experimental testing to understand the fluid dynamics of the inspection process and links this to particle flow within the PFS. The process of inspection involves the fluidisation of particles by spinning a PFS (spin-up) and suddenly stopping its rotation (spin-down), leading to flow features enabling particle lift. A series of computational fluid dynamics (CFD) simulations are developed in COMSOL Multiphysics to replicate the spin-up and spin-down phases used in industry. A custom-built experimental rig is also designed to reproduce real inspection conditions. Research focusses on how parameters such as Reynolds number, aspect ratio, and viscosity influence flow stability, jet formation, and particle movement.

The results show that during the spin-down phase, strong axial jets form within the syringe, creating a short but vital window when the particles become suspended. This is essential for visions systems to detect particles during inspection and it is found that higher Reynolds numbers and larger aspect ratios promote fluidisation, whereas, higher viscosity dampens motion. By using simulations and experiments, operational maps are created to help manufacturers optimise inspection settings for different syringe types based on their aspect ratio. The findings have direct industrial relevance discussed in the thesis. The study also highlights directions for future research, including multi-particle simulations, 3D modelling, and enhanced imaging methods. Ultimately, this work bridges the gap between theoretical fluid mechanics and practical pharmaceutical inspection. Offering tools and insights that can improve safety, efficiency, and consistency in the production of pre-filled syringes.

Contents

1 Introduction **1**
1.1 Brief Overview of The Pre-filled Medical Syringe . . . 1
1.2 Context: The Industrial Significance . . . 2
1.3 Research Aim and Objectives . . . 3
1.4 Methodology . . . 4
1.5 Thesis Layout . . . 5

2 Literature Review **6**
2.1 Challenges In Production of pre-filled syringes . . . 6
2.2 Visible and Sub-visible Particles . . . 8
2.3 The Inspection Process . . . 10
2.3.1 Human Inspection . . . 10
2.3.2 Robotic Inspection . . . 11
2.3.3 Importance of Headspace . . . 14
2.4 Rotational Flows . . . 14
2.4.1 Non-Dimensional Parameters . . . 14
2.4.2 Fluid Instabilities . . . 17
2.4.3 Spin-up . . . 19
2.4.4 Spin-down . . . 22
2.5 CFD and Particle Tracking . . . 25
2.6 Visual Inspection Methods . . . 26
2.6.1 Syringe Damage . . . 26
2.6.2 Particle Inspection . . . 26
2.6.3 Vial Inspection . . . 29
2.7 Summary of Literature Review . . . 29

3 Materials and Methods: Computational Methods **30**
3.1 Introduction . . . 30
3.2 Fluid Flow Setup . . . 31
3.2.1 Problem Setup . . . 31
3.2.2 Definitions & Non-dimensionalisation . . . 32
3.2.3 The Computational Domain . . . 32
3.2.4 Boundary & Initial Conditions . . . 33
3.2.5 Meshing Strategy . . . 35
3.2.6 Governing Equations and The Weak Formulation . . . 36
3.2.7 Spacial Discretisation and the Finite Element Method . . . 37
3.2.8 Temporal Discretisation (Time-Stepping Method) . . . 38
3.2.9 Solving the Discretised Equations . . . 38
3.3 Lagrangian Particle Tracking Model . . . 39
3.3.1 Particle Motion Formulation . . . 40
3.3.2 Forces Acting on Particles . . . 40

3.3.3 2D to 3D Mapping (COMSOL Specific) . . . 42
3.3.4 Boundary and Initial Conditions . . . 43
3.3.5 Solver settings . . . 44
3.4 Hardware Specifications . . . 45
3.5 Mesh Independence Study . . . 45
3.5.1 Grid Convergence Index (GCI) . . . 46

4 Fluid Flow: Data Collection and Simulation 54
4.1 Spin-up . . . 54
4.1.1 Primary Flow . . . 54
4.1.2 Secondary Flow . . . 62
4.2 Spin-Down . . . 67
4.2.1 Primary Flow . . . 67
4.2.2 Secondary Flow . . . 74
4.3 Effect of the Reynolds number on spin-down . . . 77
4.3.1 Below the critical Reynolds number . . . 77
4.3.2 Jet interactions at the critical Reynolds Number, Re_{crit}. . . . 79
4.3.3 The development and progression of the vortices beyond the critical Reynolds Number, Re_{crit}. . . . 82
4.4 Aspect Ratio Study . . . 84
4.5 Conclusion . . . 86

5 Particle Motion Studies: Computational Data Collection and Analysis 87
5.1 Massless Particles . . . 88
5.1.1 Spin-up: General Particle Motion . . . 88
5.1.2 Spin-up: Impact of Initial Radial Seeding . . . 90
5.1.3 Spin-up: Massless Particle Quarter Grid Seeding . . . 93
5.1.4 Spin-down: Impact of Initial Radial Seeding (1000 RPM or Re $\approx$ 364) 95
5.1.5 Spin-down: Effect of increased RPM (3000 RPM) . . . 98
5.1.6 Spin-down: Impact of Initial Height Seeding (1000 RPM) . . . 100
5.1.7 Spin-down: Geometry Study (Conical Plunger) . . . 101
5.2 Full Cycle Heatmaps (massless particles) . . . 104
5.2.1 Aspect Ratio Study . . . 105
5.2.2 Reynolds Number Study . . . 119
5.3 Particles With Mass . . . 123
5.3.1 Single Mass Particle . . . 123
5.3.2 Particle Diameter Study with Grid Seeding . . . 126

6 Materials and Methods: Experimental Setup 135
6.1 Overview of the Experimental Objectives . . . 135
6.2 Apparatus . . . 135
6.3 Syringe and Particle Preparation . . . 140
6.3.1 Fluid Station . . . 140

6.3.2 Particle Station . . . 141
6.3.3 Cutting Station . . . 142
6.3.4 Syringe Assembly . . . 143
6.4 Experimental Procedure . . . 144
6.5 Post Processing . . . 145
6.5.1 Conversion to Tiff . . . 146
6.5.2 Sorting Relevant Test Image Frames . . . 146
6.5.3 Background Subtraction Algorithm . . . 146
6.5.4 Data clean-up . . . 147
6.5.5 Analysis_tool.m . . . 149
6.6 Experimental Plan . . . 152

7 Experimental Results 154
7.1 RPM Study . . . 154
7.2 Particle Diameter Study . . . 157
7.3 Particle Density Study . . . 159
7.4 Viscosity Study . . . 162
7.5 Aspect Ratio Study . . . 169
7.6 Headspace Study . . . 171
7.7 Experimental and Computational Validation . . . 173
7.7.1 Consistent Qualitative Trends . . . 173
7.8 Experimental Summary . . . 174

8 Conclusion & Industrial Focus 175
8.1 What has been learned from the fluid dynamics of inspection? . . . 175
8.2 What has been learned about the motion of particles during inspection? . . . 175
8.3 Major outputs from the PhD project . . . 176
8.4 Practical implications in syringe inspection for industry . . . 177
8.5 Ideal inspection of particles in Pre-filled Syringes. . . . 177
8.6 Future Work . . . 178

9 Appendix 181
9.1 Conversion Code: Conversion_code.m . . . 181
9.2 Run Sort Code: Run_Sort.m . . . 183
9.3 Background Subtraction Algorithm: main_.m . . . 189
9.3.1 File 1: main_.m . . . 189
9.3.2 File 2: sec2hms_.m . . . 191
9.3.3 File 3: track_.m . . . 191
9.3.4 File 4: tracker_.m . . . 213
9.4 XY Plot Editor Tool: "XYPlotEditorGUI_.m . . . 232

10 References 248

1 Introduction

1.1 Brief Overview of The Pre-filled Medical Syringe

A parenteral drug delivery system administers drugs by injection, infusion, or implantation rather than through the alimentary canal. A pre-filled syringe (PFS) is one of these parenteral drug delivery systems, where other types of drug delivery systems include oral (most common), pulmonary (using the respiratory system for transport), and transdermal (absorbed through skin) (HealthManagement, 2012). PFSs have gained widespread adoption because of their safety, efficiency, and ease of use. Figure 1 shows the typical components of a PFS.

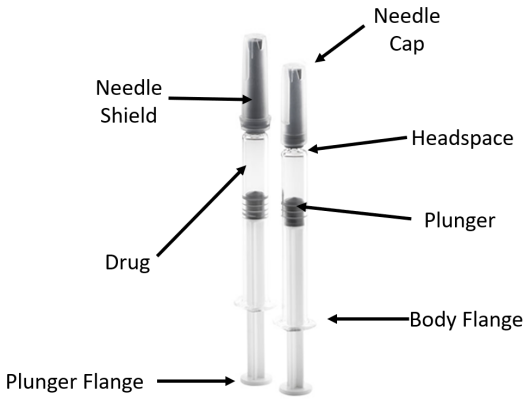


Figure 1: Components within a PFS, modified from West Pharmaceutical Services (2021).

The primary components of a PFS include the drug solution, needle (with cap and shield), plunger, and flanges. Depending on the application, there may be slight variations in the components shown, such as the inclusion of flange extenders and syringe stoppers.

The PFS drug delivery method continues to grow in popularity and has become the primary container of choice for parenteral drug delivery systems. PFS devices are the fastest growing segment of the injectable delivery device market (Kale et al., 2015). By having the primary advantages of greater medication safety and increased convenience due to a prefillable device, PFS devices have become the container of choice since their inception (Romacker et al., 2008). An alternative method of drug delivery via injection is the traditional vial and syringe method, but this requires various steps to be followed by the healthcare practitioner for the preparation of the drug before it can be administered. Initially, the PFS delivery system was an insignificant niche product; the initial success of PFS began with the heparin syringe by Sanofi and Rhone Poulenc-Rorer around the 1980s in Europe (Romacker et al., 2008). The initial success helped the PFS market grow from its origin to ever-rising popularity to become the most popular method by the late 1990s and early 2000s. As of 2008, the global market had a PFS presence of almost 2.2 billion syringes. With its increased popularity and higher demand for production, new challenges arise within various aspects of the production of PFS. Some of these challenges and the current market trends are discussed in the latter part of this section.

PFSs are chosen due to many advantages over traditional vial and syringe methods. These can apply to both the patients as well as the company manufacturing them. For the patient, there is an ease of self-administration of the drug alongside the decreased risk of contamination. This is due to fewer steps required for preparation of the dose in comparison to vial and syringe methods. From the manufacturer's perspective, there is a conservation of drug solution as no overfill volume and less preparation is needed for manufacturing (Sacha et al., 2015). Some other benefits of PFS devices compared to the traditional vial and syringe method are shown below (Pironti et al., 2020):

1. There is a lower risk of microbial and chemical cross-contamination due to a single unit of syringe already prepared for administration. The procedure is as simple as removing the syringe from its packaging and administering the drug. There are no other procedures that are needed.

2. Drugs can be administered quickly. This is considered a key benefit in critical patient situations.

3. Patients are able to self administer treatments at home without the need to go to specialists or hospitals.

4. There is a low injury risk from physicians due to minor handling.

5. Improved dosage accuracy and reduced waste.

1.2 Context: The Industrial Significance

The pre-filled medical syringe device market falls under the global pharmaceutical industry. A report by Statista (2025) sets the value of the global pharmaceutical industry's worth at approximately $1.21 trillion in 2025. The global pre-filled syringes market was valued at $8.01 billion USD in 2024. This is a significant rise from $4.9 billion in 2018. This is projected to expand at a compound annual growth rate (CAGR) of 13.5% by 2030 to $16.73 billion predicted by ResearchAndMarkets.com (2025). Due to the COVID-19 pandemic, traditional syringe and vials have contributed to massive glass shortages that have led the pharma industry to adopt pre-filled syringes (Curia, n.d.). With large investments at stake across the world, the production of PFS and its sales are essential for both the producers as well as the users. Usable PFS devices are of critical importance, and these devices must be produced effectively. The initial wave of COVID vaccines does not include PFS delivery systems, primarily due to the emergency nature of the pandemic and the cheaper production costs associated with the syringe and vial method. Feinmann (2021) stated that the National Health Service (NHS) UK is in a position to look at the use of PFS devices as vaccination becomes routine (like the flu vaccine). Although not used initially, MHRA approved the use of Pfizer/BioNTech's Comirnaty KP.2 and Moderna's Spikevax JN.1 for the covid booster programs by late 2023 into 2024 in the UK. The inspection for these PFS devices needs to follow good manufacturing practices and regulatory standards must be met through quality assurance. This drives the motivation for this project which focuses on the inspection process of PFS devices.

During its production, unwanted foreign particles (visible and sub-visible) from within the manufacturing process have the potential to make their way into the PFS. Contamination of these particles within PFS devices can occur in various stages of production and these particles increase the risk of adverse side effects (Bouillet et al., 2021). Sub-visible particles are defined to be particles with a diameter of 0.1 µm to 100 µm (Carpenter et al., 2015). These fall outside the detection range of the naked eye. Robotic inspection methods have been introduced in the production of such PFS devices to limit the presence of visible and sub-visible particles in the final product released to the market. During inspection, particles can be anywhere within the syringe. These can be in the bulk fluid, resting on the plunger or in the walls. Automated Inspection is carried out by means of fluidising particles. It can be challenging to spot particles when located on the walls of the container, so the particles are forced into the bulk fluid, away from the walls, by a process of fluidisation. This involves rapid deceleration of a rotating syringe. Initially, the syringe is rotated at a fixed rpm (also called spin-up), followed by an abrupt deceleration (spin-down) to a complete stop. This method is widely adopted in industry as it provides a rapid and effective means of fluidising particles within the syringe. The sudden deceleration generates flow structures such as axial jets, which lift particles away from the walls and into the bulk fluid. This significantly improves the likelihood of particle detection using imaging systems activated once spin-down begins, making it a reliable and efficient approach for automated inspection. Particle tracking algorithms are also used alongside cameras to obtain particle tracks within the PFS device during an inspection. These tracking algorithms are relatively simple and computationally inexpensive. After detection of external particles takes place, the PFS device is rejected and cannot be sold by the manufacturer. These can often be re-inspected by the manufacturer to ensure false rejects. The presence of particles, even the smallest in their size, poses health risks and is extremely undesirably a hazard, as they indicate a failure in the manufacturing, sterilisation, or storage processes.

1.3 Research Aim and Objectives

The project is titled "Fluid Mechanics Of The Inspection Of Pre-filled Medical Syringes". The fluid behaviour during the spin-up and spin-down phases varies and can impact the detection of visible and sub-visible particles. The project aims to understand the fluid dynamics of the inspection process and relate this to particle motion in order to support the improvement of current inspection techniques applied within the pharmaceutical industry. In this context, improvement refers to providing a clearer understanding of how fluid flow influences particle behaviour, which can be used to inform the optimisation and customisation of inspection conditions in industrial systems.
The aim of the project can be achieved through a set of objectives outlined below:

1. Gain an understanding of the PFS inspection process by studying the fluid flow during spin-up and spin-down. Understand how fluid flow can help with particle detection and imaging. This includes studying how the particle moves within the given flow field and can be achieved by studying literature around rotational flows as well as their application in the inspection of PFS.

2. Develop a set of computational fluid dynamics (CFD) models with varying complexity,

that are capable of capturing the flow behaviour during the spin-up and spin-down phase. Understand both phases and their resulting effect on the fluid dynamics. Study the effects of various fluid properties (e.g., density and viscosity, etc.) and parameters (PFS container dimensions, rotational speed, etc.) on the fluid behaviour through the spin-up and spin-down phase. These can be achieved through the following models;

Part 1- **Flow model**: A flow model able to capture the fluid behaviour on a simple cylinder-type container with no free-surface.

Part 2- **Particle tracking model**: A model with particle tracking implemented within the flow model. Using a grid-like method for particle seeding, study the motion of particles during the spin-up and the spin-down phase.

3. Build an experimental rig that replicates key physical aspects of the real inspection process. In particular,the particle motion during spin-up and spin-down. The goal is to experimentally observe how parameters (e.g., rotational speeds, fluid properties) influence particle movement. These findings will help to provide insights that may guide future optimisation of the inspection processes or parameter selection in industrial contexts.

4. Map the parameter space in terms of the most dominant parameters so that the industry could use these results for customised testing based on different PFS devices. Use this to create a limited set of profiles based on the most dominant parameters that can later be used for inspection, providing ease of operation in an industrial setting.

1.4 Methodology

To develop a CFD model that can replicate fluid behaviour, COMSOL Multiphysics 5.6 (COMSOL AB, 2021) has been chosen. Its ability to couple a 2D axisymmetric fluid study with 3D particle tracking provides a computationally efficient way to analyse particle motion. This flexibility and ease of use make it well suited for the particle-tracking focus of the project. For the CFD component of the project, the procedure of geometry, meshing, solution, results, and post-processing will be followed. To complete the various objectives outlined above, a simple model replicating a rotating cylinder filled with a fluid of known properties will be modelled.

The validation of results obtained through the CFD analysis will be done by comparison to a carefully designed experimental set-up. Real PFS devices obtained from an industrial partner will be used in the experiments where the inspection process is replicated. By using the advantages of CFD and experimentation, a detailed analysis of the flow behaviour within the PFS device as it goes through inspection can be obtained. Using tools such as particle tracking on COMSOL can allow for a more detailed analysis of each particle injected into the domain and thus provides insight into particle motion overall. This may not be easy if only experiments are used to study particle tracking.

1.5 Thesis Layout

This thesis is structured to guide the reader logically from the context and motivation of PFS inspection research to the results and industrial implications. Here is an outline of each chapter.

Chapter 1 Introduction

In this chapter, background and industrial motivation for studying the inspection of PFS is presented. The research aim, objectives, and the methodology adopted to achieve them are outlined.

Chapter 2 Literature Review

This chapter critically reviews the prior research related to PFS, particle contamination, inspection technologies, and the underlying fluid mechanics of rotational flows. The gaps identified in this chapter form the basis for the present study.

Chapter 3 Materials and Methods: Computational Methods

This section covers the computational methods used to simulate the spin-up and spin-down of a PFS during inspection. It explains the governing equations, non-dimensionalisation, boundary conditions, and particle-tracking approach used in COMSOL Multiphysics.

Chapter 4 Fluid Flow: Data Collection and Simulation

This chapter presents the results of fluid flow during spin-up and spin-down, highlighting the influence of Reynolds number and aspect ratio on flow stability and vortex development.

Chapter 5 Particle Motion Studies: Computational Data Collection and Analysis

Extends the analysis conducted in Chapter 4 to particles. It explores how particle trajectories evolve under various flow regimes and geometrical conditions.

Chapter 6 Materials and Methods: Experimental Setup

In this chapter, the design of the experimental apparatus replicating the inspection process is presented. This includes the preparation of test samples, and the data-processing techniques to produce particle tracking results.

Chapter 7 Experimental Results

This chapter presents the experimental results on particle motion under varying conditions (e.g., RPM, viscosity, particle density, and aspect ratio).

Chapter 8 Conclusions and Industrial Focus

The final chapter summarises the key findings and discusses their implications for syringe-inspection practices. The chapter concludes with industrial recommendations and directions for future work.

2 Literature Review

This chapter outlines and reviews previous research related to the inspection of PFS. Namely, the manufacturing challenge, particle contamination and the inspection techniques. It summarises key findings on rotational fluid flows and the use of CFD and particle tracking to analyse particle motion during the inspection of a PFS. The review concludes by identifying current gaps linking transient flow behaviour to particle detection in syringe inspection systems.

2.1 Challenges In Production of pre-filled syringes

The process of manufacturing a PFS is by no means simple. Pironti et al. (2020) describes this as a complex process that requires substantial investment of resources. Getting PFS devices on the market in a timely, compliant, and sustainable manner is not easy. One of the reasons for this is the variety of components present in a PFS (Kale et al., 2015). These are usually variations of different metals, glass, plastic, or rubber. An example of where these types of materials can be used is shown in Figure 2.

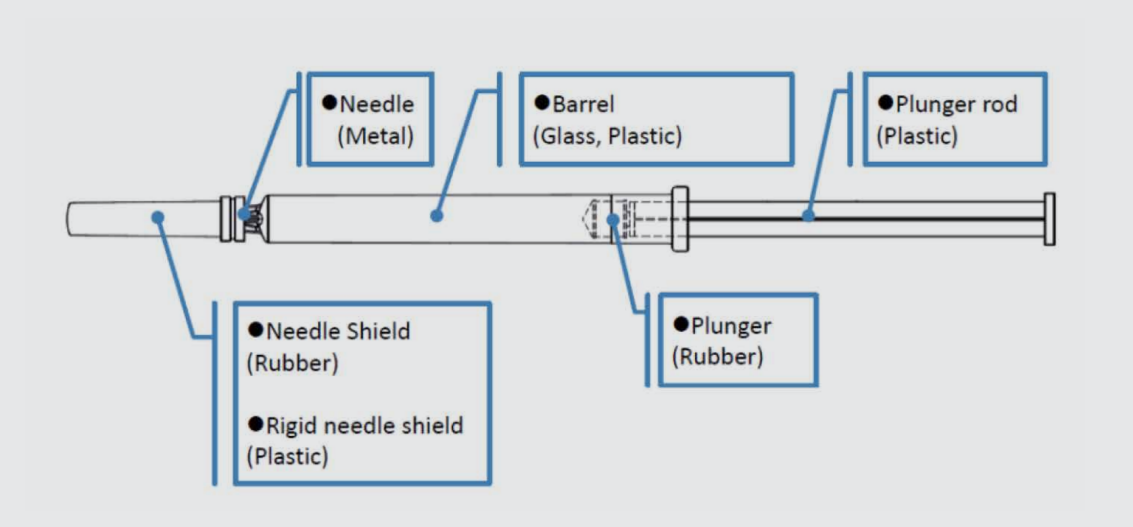


Figure 2: Possible materials used within a syringe from Pironti et al. (2020).

During its production, a PFS goes through various manufacturing processes for all its components, and some of these processes, if not performed correctly, can cause the PFS components to be damaged or render the device unsafe for use. Good manufacturing practices and standard regulations require these PFS devices to be detected and discarded. To ensure that defective PFS are not sent to the market, manufacturers are required to perform thorough inspections. To name a few; visible inspection is conducted on; metals, o-ring seals, filters, pumps, pipes, cracks in containers, plunger management, missing needle caps, etc. For the final PFS produced, these defects can be summarised as plunger, fluid (drug), or container defects. Figure 3 shows a list of some of the inspections performed for certain types of defects during the production of a PFS. A detailed discussion of the types of inspections conducted

on pre-filled syringes alongside the standards for each inspection is provided by Challener (2019). The various inspections conducted are typically static inspection processes. However, inspections within the dashed box require a dynamic process. These are challenging research problems for which CFD and experimentation can provide insight into the complex dynamic behaviour observed. For this project, these are the main areas of interest.

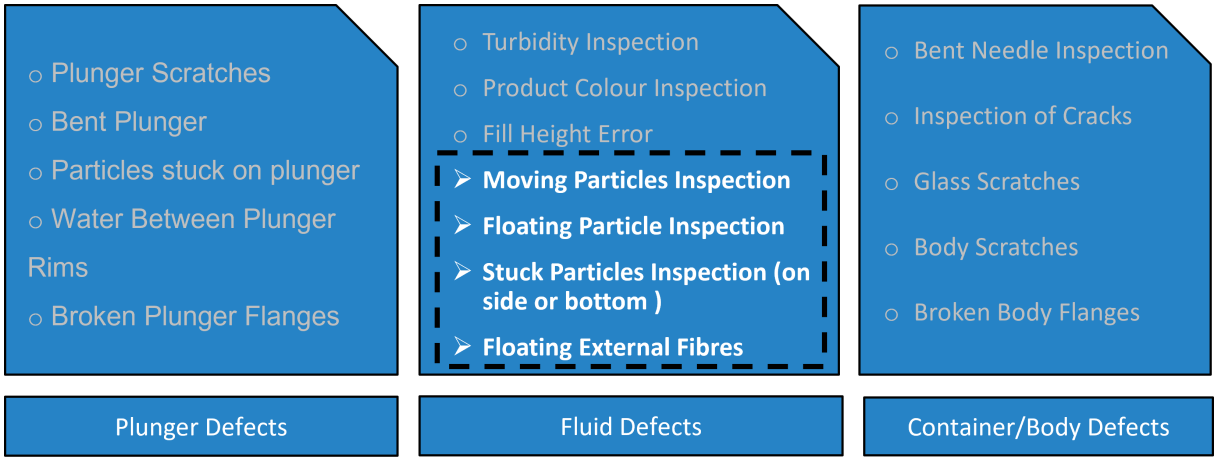


Figure 3: Potential defects within the production of PFS, where the dashed box represents the inspections of interest to this project.

An important process within the production of a PFS is surface treatment or lubrication, commonly achieved by siliconization. This is usually applied to glass syringes, plungers, needles, and some polymeric syringes. Colas et al. (2006) defines the process as the addition of silicone fluid by applying a low concentration of either a solvent or an emulsion-based solution. Then, this is baked or air-dried at elevated temperatures to improve film adhesion. Silicone fluid is also referred to as polydimethylsiloxane (PDMS), and there are three types of silicone products in use; non-reactive silicone fluids, non-reactive silicone emulsion, and reactive silicone dispersion. Siliconization of parental packaging components provides clear benefits of its use, especially for drainage and lubrication of glass cartridges. However, if not performed correctly, this can be a potential source of contamination. The U.S. Food and Drug Administration (FDA) guides the industry on the criteria needed for the safety, quality, and purity of a drug product. It is stated in FDA (2004), that the appropriate quality control criteria should be met and there should not be any adverse effects. An example of an adverse effect is the aggregation of particles through the degradation of protein-based drugs or tungsten residue used within a PFS. This is a well-known problem within the production of PFS and examples of solutions to silicone-based lubrication are the use of plastic syringes or silicone oil-free syringe systems, such as the Daikyo Crystal Zenith by Daikyo Seiko and West (Hlobik, 2019). The manufacturing and challenges of PFS are discussed in detail in Sacha et al. (2015). Other sources of contamination mentioned are alkali metals that can leach out of the glass container and result in pH shifts and decreased potency of the drug. Some of the potential interactions between drug products and PFS materials are shown in Table 1. This highlight the complexity of figuring out where potential particles may come from.

Table 1: Potential interactions of drug solutions with PFS materials from Pironti et al. (2020).

Phenomenon	**Causing factor**	**Related material**
Physical	Aggregation by silicon oil	Independent of material
	Aggregation by tungsten	Glass
	Interaction with glue	Dependent on manufacture
	Excessive shaking	Independent of material
Chemical	Alkali elution	Glass
	Gas permeability	Polymer
	Residual radicals	Dependent on sterilisation

2.2 Visible and Sub-visible Particles

One of the leading authorities in the world in the regulation of pharmaceuticals is the U.S. Food and Drug Administration (FDA). United States Pharmacopeia (USP) are a set of standards used by the FDA. USP chapter < 788 > explains how to test for external particles that are usually mobile and undissolved. These refer to particles other than the drug itself (including gas bubbles), and there are many ways in which visible and subvisible particles have the potential to make their way into the final PMS product. Subvisible particles are defined as particles within the size range of 0.1 µm to 100 µm and these particles are not large enough to be seen by the naked eye (Carpenter et al., 2015). Visible particles are those greater than 100 µm. Subvisible particles can make their way through the shedding of process equipment during manufacture. If not detected, these particles may remain in the syringe following quality control after production. As mentioned above, the presence of tungsten or silicon oil within the container can also introduce particles into a PFS (Sacha et al., 2015). This is achieved through the aggregation or degradation of protein-based solutions. Moreover, glass-based containers can shed small particles, which can then enter the drug solution. Another source of particles within the PFS during manufacturing can occur if plungers are inserted into a syringe by an insertion rod. Sacha et al. (2015) suggests that black marks can occur during this process, and these can cause damage. Moreover, if the insertion rod is not aligned properly, sliding against the insertion tube could allow stainless steel particles to enter the solution. These are just some of the many ways in which unwanted particles can pass through to the final product. Some of the various regulations that apply to the testing of particulate matter in parental solutions are provided in the following USP standards;

- **Injections:** USP < 1 >
- **Visible Particulates:** USP < 790 >
- **Sub-Visible Particulates:** USP < 787 >, USP < 788 >, USP < 789 >

In general, all articles for the administration of parental drugs should be designed to exclude particulate matter. Foreign particles within the human body can cause serious health problems. An example of this outside of PFS are the leakage of metal particles within the body

from metal-on-metal wear. These lead to local tissue irritation and systemic metal toxicity. For PFS, there are no visible particles allowed with a limit on subvisible particles that can pass through inspection. These regulations also provide and approve methods of detection such as "Light Obscuration Particle Count" and "Microscopic Particle Count". Bouillet et al. (2021) states that a PFS device can lose functionality or performance due to the presence of these visible and subvisible particles within the drug solution. In addition, this can also cause the development of immunological responses and adverse reactions in patients. For the case of immunological responses, Carpenter et al. (2015) refers to it as a particular problem that can cause some patients to become secondary nonresponders due to the formation of neutralising antibodies (i.e., non-responsive to maintenance treatment). In turn, this can affect the success of miracle drugs (Carpenter et al., 2015). Typical particles detected using inspection are fibres, glass, rubber, and various metals. Some examples of such particles can be seen in Figure 4 (taken from a major UK pharmaceutical manufacturer during robotic inspection testing in 2019). US pharmacopoeia has established numerical limits on particle sizes at less than 10 µm. This is mainly due to the fact that particles at this size block human capillaries. However, the average diameter of a capillary is 7 µm, which suggests that there is still a possibility of particles passing through inspection. If this is the case, it may cause capillary blockage that would lead to more side effects for the patients. It is therefore essential that these particles are detected within the manufacturing process, and the PFS devices containing such particles are rejected.

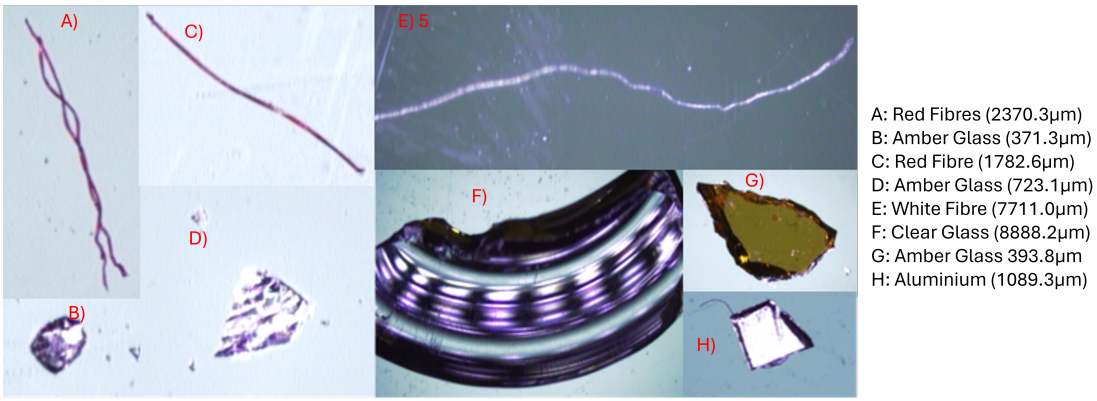


Figure 4: Examples of particles and their size used by a major UK pharmaceutical manufacturer during robotic inspection testing

2.3 The Inspection Process

Inspection of visible and subvisible particles is essential during the production of PFS. As this project focuses on the inspection process for particle detection, other types of inspection will not be discussed (i.e inspections mentioned on Figure 3, shown outside the dashed box). In this section, the inspection process needed for the detection of particles is explained as well as different methods of inspection (i.e., human and robotic inspection).

2.3.1 Human Inspection

Human inspection, sometimes called manual inspection, serves as a secondary check on defects already discussed in the previous section. After filling and sealing, the integrity of the container closure must be free of defects and it should be ensured that there is no breach in sterility and that there are no free visible particles in the final product. Manual inspection can be performed on machines such as the Syntegon MIH-1 (shown in Figure 5).

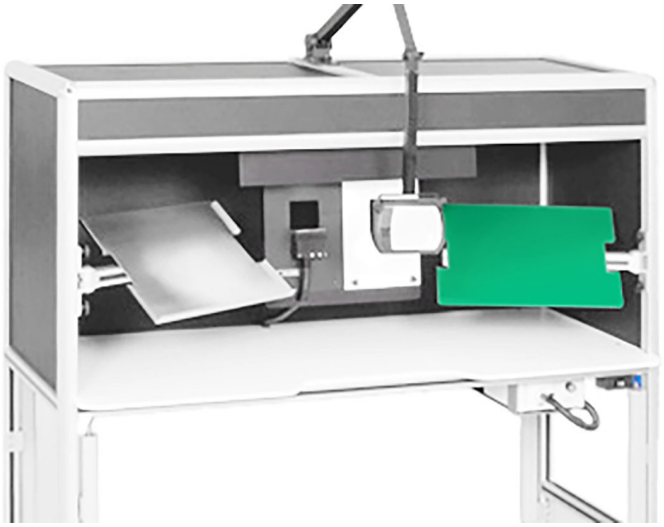


Figure 5: MIH-1 workstation from Syntegon (2021).

The MIH-1 allows humans (trained and certified) to test the product in a controlled environment. The machine can be adjusted to various height requirements and consists of two non-flickering fluorescent lamps. To help with inspection, machines such as the MIH-1 contain a magnifying glass to visualise the syringe. This is a static process in which a human can inspect against various backgrounds (white or black, etc.) in constant light. This is viewed from multiple angles to detect any foreign objects within the PFS. It can sometimes be used for re-inspection of both previous rejects from robotic inspection or manual inspection. It should also be noted that manual inspectors require special training that is needed for successful detection.

The advantages of manual inspection include reduced setup costs for manufacturers since expensive equipment is not needed as opposed to robotic inspection. Moreover, it may not always be suitable to use expensive automated machines depending on the size of the company manufacturing the product. However, the reliability of manual inspection is not consistent. After collecting data from various sources, Shabushnig et al. (1995) produced Figure 6. Note that the black trend line on the figure extends beyond 100% but is not physically meaningful. In practice, detection probability is bounded between 0% and 100%, and the linear trend should be interpreted only as an indication of increasing detectability with particle size.

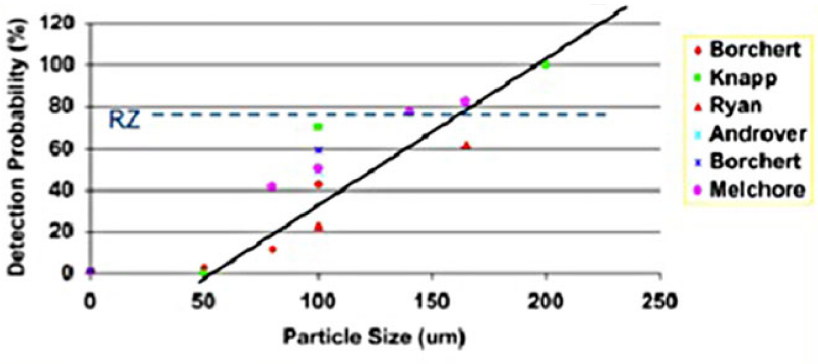


Figure 6: Probability of detection using manual inspection from Shabushnig et al. (1995).

It can be seen that the probability of detection was 100% for particles greater than 200 µm. If detected, these particle sizes are too large to pass through regulations. It is clear that, for the range of subvisible particles, a manual inspection may not be reliable. This is because the chance of detection varies from 20% to 40%, even at the end of the sub-visible particles range of 100 µm. From USP regulations, depending on the volume of solution, the size of the detected particles cannot exceed 10 µm or 25 µm. Based on the results presented in Figure 6, it is clear that these will not be detected. Therefore, better methods of detection are required and this is where automatic (robotic) inspection are more reliable.

In 2008, 33% of the companies that inspect particulate matter used manual (human) inspection methods (Leversee and Shabushnig, 2008). Semi-automated or automated inspections contributed to 24% and 43%, respectively. Within these inspections, the most common source of defects identified is particulate matter, followed by scratches. Within the identified particulate matter, the order of the most common particulates is lint/fibre, glass, product-related particulates, rubber and metals.

2.3.2 Robotic Inspection

Fully automatic inspection machines are an alternative to manual inspection. Using both static and dynamic processes, these machines are capable of performing most of the inspections in Figure 3. Like manual inspection machines, robotic inspection machines create a controlled environment for inspection. This is achieved through the use of high-speed cameras and lighting. Particulate inspection is a small part of what these machines are capable

of, and since that is the area of interest for this project, it will be the main focus. An example of such a machine is an AIM 8000 series from Syntegon, and it is shown in Figure 7.

Figure 7: AIM 8000 Series by Syntegon (2021 a).

For the inspection of particulate matter, robotic methods use a dynamic process in which the syringe is rotated about its long axis (typically 1000 to 8000 RPM), and this is known as the spin-up phase. The syringe is then abruptly decelerated to rest in the spin-down phase. A vortex forms during the spin-up phase and collapses during the spin-down phase. The fluidisation of particles observed in the inspection stage is anecdotally associated with the collapse of the vortex when a free surface is present in the syringe. However, this has yet to be tested.

As the syringe rotates, the fluid inside also starts to rotate with it. When the syringe is abruptly decelerated, the fluid still has momentum and continues to rotate until it is once again stationary. Between this, particles and bubbles (if any) are fludised within the syringe and can then be detected using cameras and various particle tracking algorithms. In this case, the rotation of the fluid has been used as a tool for inspection by fluidisation of particles. Figure 8 shows a schematic of the particulate matter inspection process. It should be noted that this specific inspection only lasts a couple of seconds and understanding the fluid behaviour during this time is essential for inspection.

In general, the USP regulations require that 100% of the liquid products and containers be inspected. One of the main concerns for manufacturers is the time taken for inspection, as well as the reliability of the inspection process. Ullherr (2008) explains that using robotic inspection machines (like the AIM 8000 series), up to 36,000 syringe devices can be inspected per hour in 10 filling positions. Humans are no match for robotic inspection as it is beyond the capability of what humans can do reliably. Since the value to the manufacturer increases as the time taken for a reliable inspection is shortened, manufacturers prefer the robotic inspection method. Moreover, the use of high-speed cameras with the capability of obtaining 100s or 1000s of images on each particle inspection unit allows for a low risk of false rejects. Although there are many advantages to this method of inspection, the initial cost of a robotic inspection machine is extremely high compared to manual testing. It is therefore dependent on the manufacturer and their PFS production rates to decide which form of inspection to conduct.

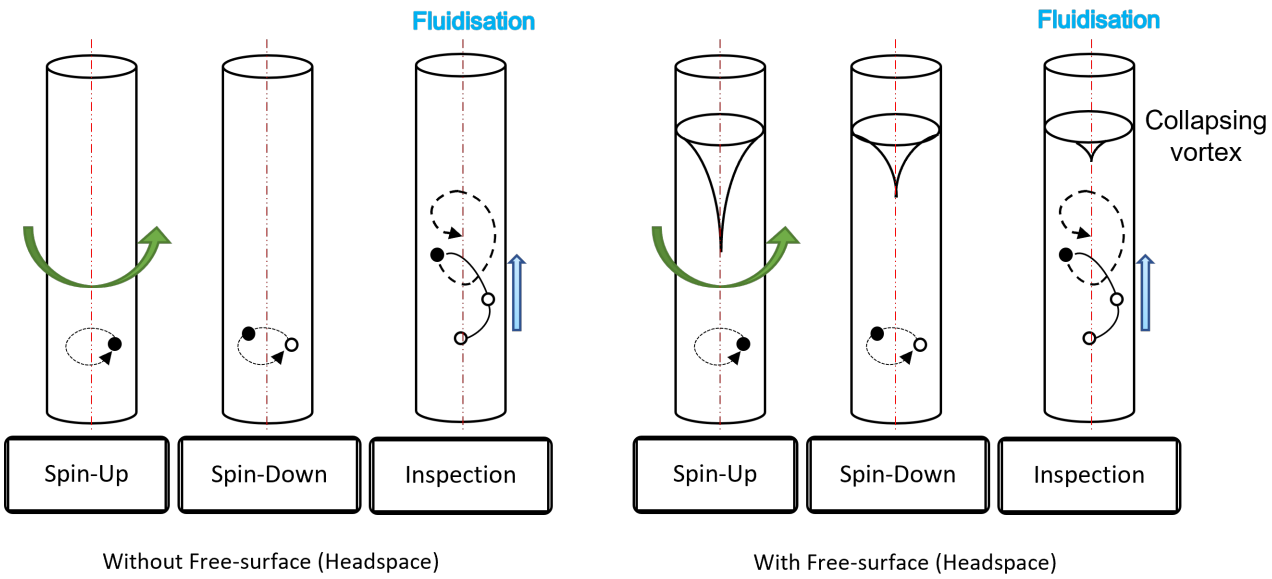


Figure 8: Schematic showing the dynamic process for fluidization of particles within a PFS device (without and with a free-surface).

The efficiency of an inspection method is typically established through a Knapp test. It is done by comparing an existing inspection method with one that needs to be tested. Originally created by Knapp and Kushner (1980), it provides the industry with a method of using statistical analysis for measuring the efficiency of the inspection process (Melchore, 2011). Consider the example for a batch of known contaminated vials where a manual inspection finds 53 total defects (ΣM) and an automatic inspection finds 57 (ΣA) after multiple tests. One can calculate the efficiency of automatic inspection using the following equation;

$$\text{Efficiency of Automatic Inspection} = \frac{\Sigma A}{\Sigma M} \times 100 = \frac{57}{53} \times 100 = 107.5\% \tag{1}$$

For the comparison of robotic inspection with manual inspection, Kardel et al. (2005) performed the Knapp test using known contaminated vials. The results show that the robotic inspection obtained a rating of 103.8 %. A rating equivalent to or greater than 100% is considered a better inspection method. Kardel et al. (2005) concludes that robotic inspection is not always successful, but performs better compared to manual inspection.

2.3.3 Importance of Headspace

PFS devices can be produced with a headspace, also known as free surface (as shown in Figure 8). The air injected into the muscle forms an airlock that prevents the medication from seeping out (Public Health England, 2014). In addition, patients are advised not to expel the air bubble, as it can waste some of the vaccine, leading to a smaller dose for the patient. The needle is flicked in an attempt to avoid injecting air into patients, as trapped bubbles can lead to health concerns. During inspection, when there is headspace, the rotation forms a vortex at the top of the syringe (Figure 8). During the deceleration process, the collapse of this vortex induces lift and rotation of the particle (Rathore et al., 2009).

However, there is no literature that specifically studies the effect of the presence of a headspace. Hence, it is not completely clear whether the presence of the headspace leads to the fluidisation of particles within the PFS inspection process. This is an important factor to consider, as the presence of a free surface can produce potential contamination hazards, as suggested by Turner (2021). During air transport (the most common type of transport for PFS devices), the free surface expands and contracts, leading to changes in dosage, evaporation, or residue depositing in the needle aperture. This issue is also highlighted by Waxman et al. (2012) where the plunger within a glass syringe moves almost by 9 mm to 9.5 mm when 18,000 feet of altitude is reached. At this height, plastic syringes do not move at all. However, at a height of 21,500 feet, both glass and plastic syringe materials are exposed to contaminated space.

2.4 Rotational Flows

This section covers the literature for the theoretical background of fluid flows observed within the inspection of a PFS.

2.4.1 Non-Dimensional Parameters

There are a few non-dimensional parameters to consider for the fluid flows exhibited within a PFS. In this section, these non-dimensional parameters are defined.

Reynold's Number

To determine whether the flow is laminar or turbulent, the Reynolds number (non-dimensional parameter) can be calculated using the equation below;

$$Re = \frac{\text{Inertial forces}}{\text{Viscous forces}} = \frac{\Omega R^2}{\nu} \tag{2}$$

where Ω , R and ν represent angular velocity ($\mathrm{rad\,s^{-1}}$), radius (m) and kinematic viscosity ($\mathrm{m^2\,s^{-1}}$) respectively. The laminar-to-turbulent transition is associated with the critical Reynolds number, Re_c. If $Re \geq Re_c$, the flow is turbulent and an appropriate turbulence model is needed when performing a CFD analysis. Using $\Omega = \frac{\alpha\pi}{30}$, where α is the RPM, $C = \frac{\pi R^2}{30\nu}$, the Reynolds number can be written as;

$$Re = \frac{\alpha\pi R^2}{30\nu} = C\alpha \tag{3}$$

Rossby and Ekman numbers

The Rossby and Ekman numbers are also two important non-dimensional parameters needed for the study of rotational flows. Rossby number is the ratio of the inertial forces to Coriolis forces (forces that act on a mass where its motion is described using a non-inertial frame of reference), whereas the Ekman number is the ratio of the viscous forces to Coriolis forces. These are defined as;

$$\text{Rossby Number: } \Rightarrow Ro = \frac{\text{Inertial force}}{\text{Coriolis force}} = \frac{U}{\Omega L} \tag{4}$$

$$\text{Ekman Number: } \Rightarrow Ek = \frac{\text{Viscous force}}{\text{Coriolis force}} = \frac{\nu}{\Omega L^2} \tag{5}$$

where U and L are the characteristic velocity and length scales (Boubnov and Golitsyn, 2012). A high Rossby number refers to inertial forces dominating over the Coriolis forces, whereas a high Ekman number suggests viscous forces dominating over the Coriolis forces.

Taylor number

The Taylor number is the ratio between the inertial forces and the viscous forces; it can be calculated using:

$$\text{Taylors Number, } Ta = \frac{\Omega^2 R_1 (R_2 - R_1)^3}{\nu^2} \tag{6}$$

where R_1 and R_2 are the diameters of the internal and external cylinders, respectively.

Dean number

The Dean number is the ratio between viscous forces and centrifugal forces. It is defined as;

$$\text{Dean Number, } D = Re\sqrt{\frac{D}{2R_c}} \tag{7}$$

where Re is the Reynolds number, D is the diameter of the pipe (m), R_c is the radius of curvature of the channel path (m).

Görtler number

The Görtler number is ratio of centrifugal effects to viscous effects within a boundary layer developing over a curved surface. The Görtler number is used to assess the stability of the flow where higher values indicate a greater tendency to centrifugal instabilities (i.e. streamwise vortices) that can have significant influence over the particle motion. The Görtler number can be calculated using (Drazin and Reid, 2004);

$$\text{Görtler Number, } G = \frac{U_\infty \theta}{\nu}\left(\frac{\theta}{R}\right)^{\frac{1}{2}} \tag{8}$$

where U_∞ is the free stream velocity of the boundary layer and θ is the momentum thickness.

The momentum thickness therefore provides a measure of the reduction in momentum within the boundary layer due to viscous effects. It is an important parameter for characterising the development and strength of the boundary layer, and is commonly used in stability analyses. In this context, it is used as a key length scale in determining the Görtler number and assessing the onset of flow instabilities. The total loss of momentum flux in a boundary layer is equivalent to the removal of momentum through a distance θ (Fagbenle et al., 2020). The momentum thickness can be calculated using;

$$\theta = \int_0^\infty \frac{u}{U_\infty}\left(1 - \frac{u}{U_\infty}\right) dy \tag{9}$$

It should also be noted that the critical Görtler number for neutral stability is $(G_\theta)_c \approx 0.3$ (Drazin and Reid, 2004).

2.4.2 Fluid Instabilities

In this section, three centrifugal fluid instabilities are highlighted. Figure 9 shows a schematic of the type of flows that can lead to centrifugal instabilities subject to various conditions. Figure 9a, 9b and 9c (from Drazin and Reid (2004)) show Taylor-Couette flow (fluid is contained within two coaxial cylinders), flow in a cured channel and flow in a boundary layer on a concave wall respectively.

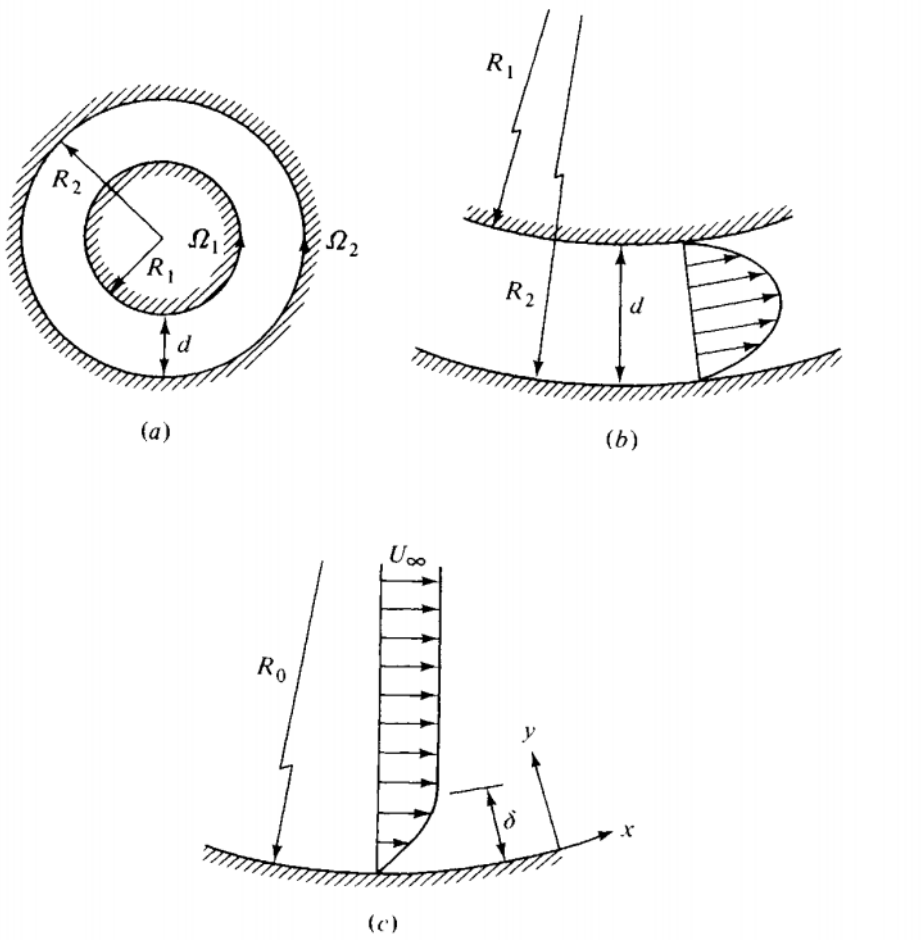


Figure 9: (a) Taylor-Couette flow. (b) Flow in a curved channel. (c) Flow in a boundary layer on a concave wall. Adapted from Drazin and Reid (2004) where the variables shown in Figure 9 are defined as follows: R_1 and R_2 represent the inner and outer cylinder radii in the Couette flow configuration, and Ω_1 and Ω_2 are the corresponding angular velocities of the cylinders. The parameter d denotes the gap between the cylinders. In the curved channel and boundary layer configurations, R_0 represents the radius of curvature of the wall. The free-stream velocity is denoted by U_∞, and δ represents the boundary layer thickness, defined as the distance from the wall to the point where the flow velocity reaches approximately 99% of the free-stream velocity.

Taylor-Couette flow instability

Taylor-Couette flows look at two coaxial concentric cylinders containing a fluid between them while moving at some speed differential. These may consist of one spinning and one stationary cylinder or two moving at different speeds. Depending on the spin conditions, the resulting flow can lead to various types of Taylor-Couette instabilities.These are classically studied analytically where infinite-cylinder geometries are used. There are also examples of experimental studies that study finite cylinders such as Andereck et al. (1986).

Figure 10 is taken from Andereck et al. (1986) and shows a flow-regime diagram for circular Couette systems with independently rotating cylinders. Re_i refers to the Reynolds number of the inner cylinder and Re_o refers to the Reynolds number of the outer cylinder. The plot highlights the transition boundaries (dashed lines) between various types of flow regime such as Taylor vortex flow, spiral turbulence, wavy vortex flow, etc. The line $Re_o = 0$ resembles the condition of a single cylinder during spin-down where the outside walls are no longer moving and therefore is extremely relevant during the inspection of prefilled syringes.

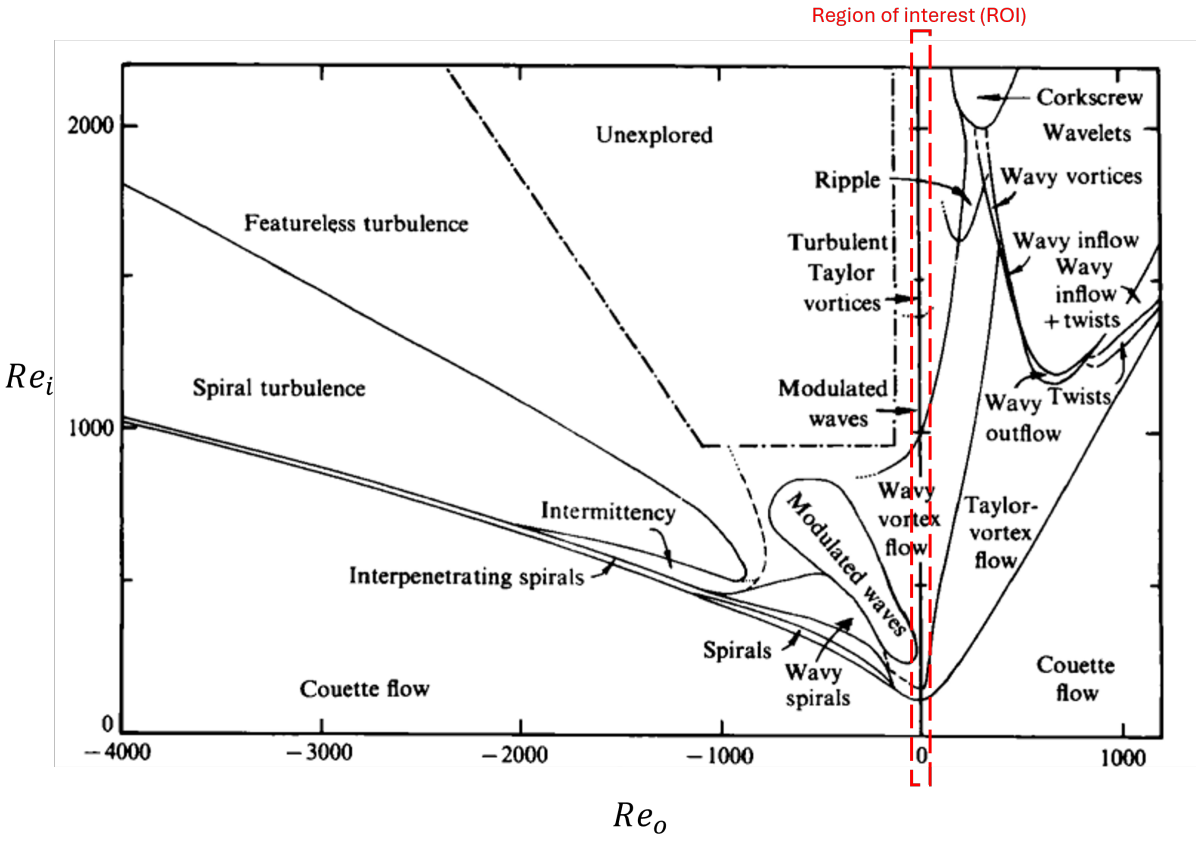


Figure 10: Flow-regime diagram for the circular Couette systems with independently rotating cylinders where Re_o is the Reynolds number of the outer cylinder and Re_i is the Reynolds number of the inner cylinder. Adapted from Andereck et al. (1986) where region of interest relevant to a PFS is highlighted by a red dashed box.

Dean vortices

The flow instability that occurs in a curved channel (Figure 9b) results in Dean vortices. This can be applied to a pipe with a 90° bend, and these are cross-sectional counter-rotating vortices that appear immediately after the bend. The presence of Dean vortices are seen beyond a critical Dean number. These vortices can cause redistribution of particles within the flow. However, since a PFS does not have any 90° bends, this is not likely applicable for a PFS.

Görtler vortices

Figure 9c shows the case in which Görtler vortices can form within a boundary layer on a concave wall. It occurs because of the centrifugal forces acting in the boundary layers over the curved surface, and this results in streamwise vortices. A schematic of an example of Görtler vortices is shown in Figure 11. The result is a superposition of the boundary layer and the counter-rotating vortex pairs (Floryan, 1992). The onset of Görtler vortices is associated with the critical Görtler stability parameter, also called the Görtler number.

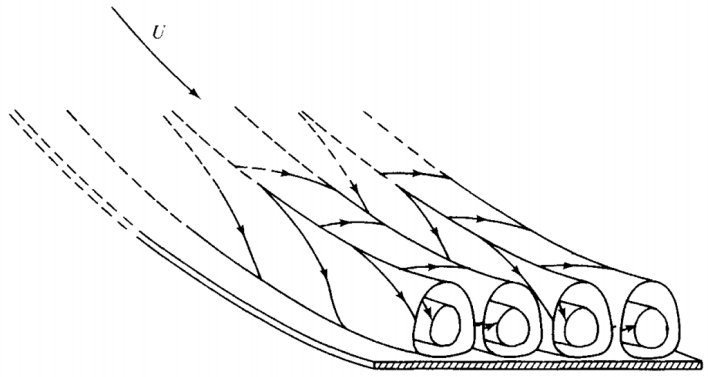


Figure 11: Görtler vortices seen as secondary flow within a boundary layer along a concave wall. Adapted from Drazin and Reid (2004)

2.4.3 Spin-up

There is a vast amount of literature covering the spin-up of a cylinder filled with fluid. The analytical theory of unsteady flow within a cylinder when it is spun about its axis is provided by Wedemeyer (1964). The work shows that the flow can be categorised into two regions that are divided by a moving front. Namely, these regions are;

- Boundary layer flow (near the end-walls)
- Core flow (rest of the flow)

As the spin-up begins, this moving front propagates from the outer walls of the cylinder, radially towards the central axis. Wedemeyer (1964) provides a method to couple the core

and secondary flow behaviour. The general behaviour of the flow is relatively well understood, and it is shown that secondary flows (boundary layer flows) occur near the end-walls (top and bottom walls). The fluid ahead of the front remains non-rotating whereas the fluid behind the front rotates with the moving boundary. A visual schematic of this flow is given in Figure 12.

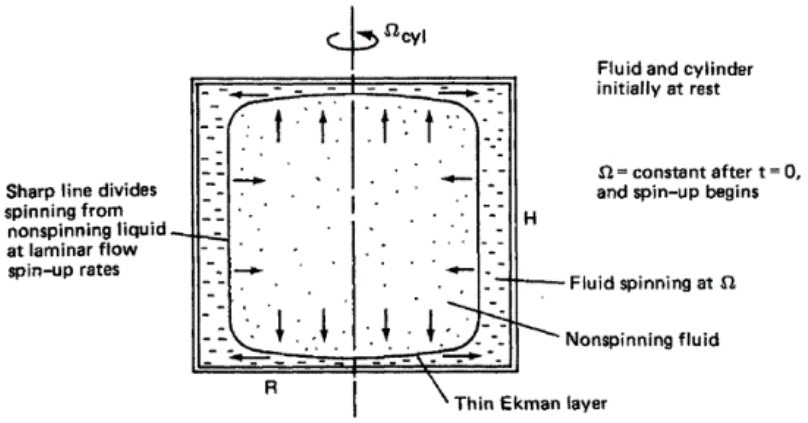


Figure 12: Schematic of the fluid behaviour as explained by Vanyo (2015)

Vanyo (2015) explains the formation of the secondary flow during spin-up as:

- Boundary layers formed at the top and bottom surfaces produce a radial pressure differential that is unbalanced by the central non-spinning fluid, and a secondary flow (radially) outward occurs in the top and bottom layers.
- The fluid is pumped axially from the centre into the boundary layers at the top and bottom of the cylinder. This flow is then pushed out radially into the periphery. A sharp line forms dividing the spinning and non-spinning fluid (i.e. the front).
- Due to continuity, the flow at the end wall near the outside of the cylinder moves axially towards the centre of the cylinder, opposite to the non-spinning flow.
- As more flow is pumped into the end walls, the thickness of the cylindrical layer increases.

The original Wedemeyer model (Wedemeyer, 1964) was a theoretical attempt focused on the spin-up from rest with some experiential support. The work used analytical modelling using the boundary layer theory and momentum integral methods. This was the first model showing the role of the end-wall convection in spin-up. This work was extended by Weidman (1976*a*) for small Ekman numbers. The work also focused on transient spin-up and spin-down velocities at constant acceleration using theoretical methods and detailed experimental study. Weidman (1976*b*) compares this extended model to experimental data collected using a Laser-Doppler-Velocimeter. It is found that the shear layer propagates away from the cylinder walls during non-linear spin-up, but it remains attached as a boundary layer during

spin-down. This provides the setting for centrifugal instability, namely, Görtler vortices (Weidman, 1976*b*).

When a single cylinder is spun, the maximum fluid velocity occurs near the outside walls. Near the centreline, the fluid is stationary. This virtually results in the same condition as two-concentric cylinders where the outer cylinder is moving and the inner cylinder is stationary. Figure 9 (a) shows an example of geometry (with two concentric cylinders) in which a Taylor-Couette flow instability can arise. These instabilities start to form once a critical Taylor number (≈ 1700) has been reached (White, 2008).

Upon reviewing previous work in the literature, Benton (1974) looks at spin-up for homogeneous and stratified fluids. It is found that various other attempts at extending the Wedemeyer models have been made, but the underlying assumptions of the model (i.e. laminar flow) and trying to parametrize the entire domain using a simple Ekman suction formula are inadequate. In these models, the transport of angular momentum is primarily attributed to secondary flows induced by boundary layers, often represented through approximate Ekman-type suction relations. Benton (1974) highlights that these assumptions are insufficient to fully capture the transient and spatially varying nature of the flow, particularly when non-linear effects and flow instabilities are present. Hence, a better model is needed, as various forms of instability can occur that may disturb the interior flow as well as the outer walls. As a result, more advanced modelling approaches are required, incorporating non-linear Ekman compatibility conditions that more accurately describe the coupling between the boundary layers and the interior flow, allowing for the influence of disturbances and complex flow development within the domain.

A detailed history of work on spin-up is also presented by Hyun et al. (1983). A numerical investigation is conducted into the spin-up from rest of a homogeneous fluid in a cylinder at small Ekman numbers. Data is collected using a finite-difference method and for an axisymmetric analysis. This is then compared to data collected experimentally using a rotating laser-Doppler velocimeter. Excellent agreement is found between the experimental data and the numerical solution. This is then compared to Wedemeyer's inviscid solution, and these results are shown in Figure 13 where azimuthal velocities, $\frac{\nu}{r\Omega}$, are plotted for non-dimensional time, T.

The numerical solution also fills in the details not predicted by the Wedemeyer model or its extensions. Comparison of the inviscid and viscous terms of the Navier-Stokes equations is made using functions of time and radius. As the front moves in, no spin-up occurs until the front arrives inward.

Hyun et al. (1983) describes the early stages of impulsive spin-up as being initially dominated by viscous diffusion at the sidewalls before the Ekman layers can fully develop. As time progresses, the influence of non-linearity and the advancing front weakens where the flow gradually becomes governed by linear coriolis acceleration effects.

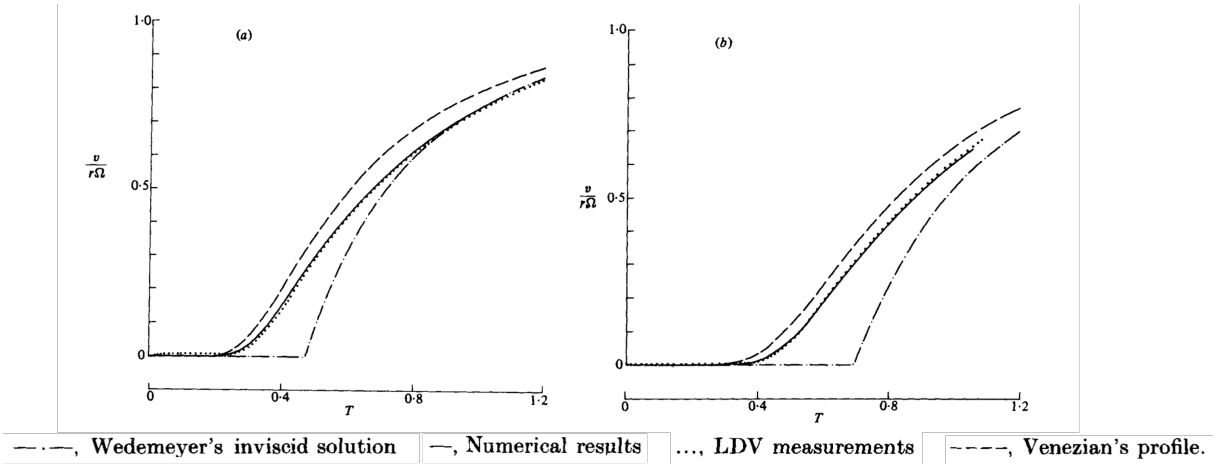


Figure 13: Figure 13: Comparison of scaled azimuthal velocity, $v/r\Omega$, as a function of non-dimensional time, T, at two radial locations: (a) $r/a = 0.75$ and (b) $r/a = 0.5$. The figure compares Wedemeyer's inviscid solution, numerical simulations, LDV experimental measurements, and Venezian's analytical profile. Adapted from Hyun et al. (1983).

2.4.4 Spin-down

As for the spin-down, Vanyo (2015) describes and summarises the process as follows:

- Once the container is impulsively stopped, a radial pressure now exists in the spinning fluid but not in the boundary layers of the (stationary) fluid.
- The secondary flow at the end boundary layers now pumps fluid radially inwards and towards the centre of the cylinder in the axial direction.
- Usually, before spin-down is complete, the secondary flow is distorted and instabilities occur.
- A central column of non-rotating fluid forms and grows in diameter.

Briley and Walls (1971) uses a numerical solution to the Navier-Stokes equations for the spin up and spin down. The time-dependent motion of a viscous incompressible fluid is modelled for a cylinder filled with fluid. The procedure involves solving the vorticity and tangential velocity equations over time using a modification of an alternating-direction-implicit (ADI) method. This method improves computational time when solving the equations of the stream function. For a low and moderate Reynolds number (405 and 1167 respectively), Figure 14 shows the stream functions after spin-down produced by Briley and Walls (1971).

It is observed that at a low Reynolds number (405), the streamlines do not show any instability. For the moderate Reynolds number of 1167, the onset of instability can be seen along the outer walls of the cylinder. Although no data are provided for the Görtler number in these cases, the observed wall-adjacent waviness at Re=1167 suggests that centrifugal effects may have exceeded critical thresholds for instability, consistent with Görtler- or Taylor-Couette-type behaviour.

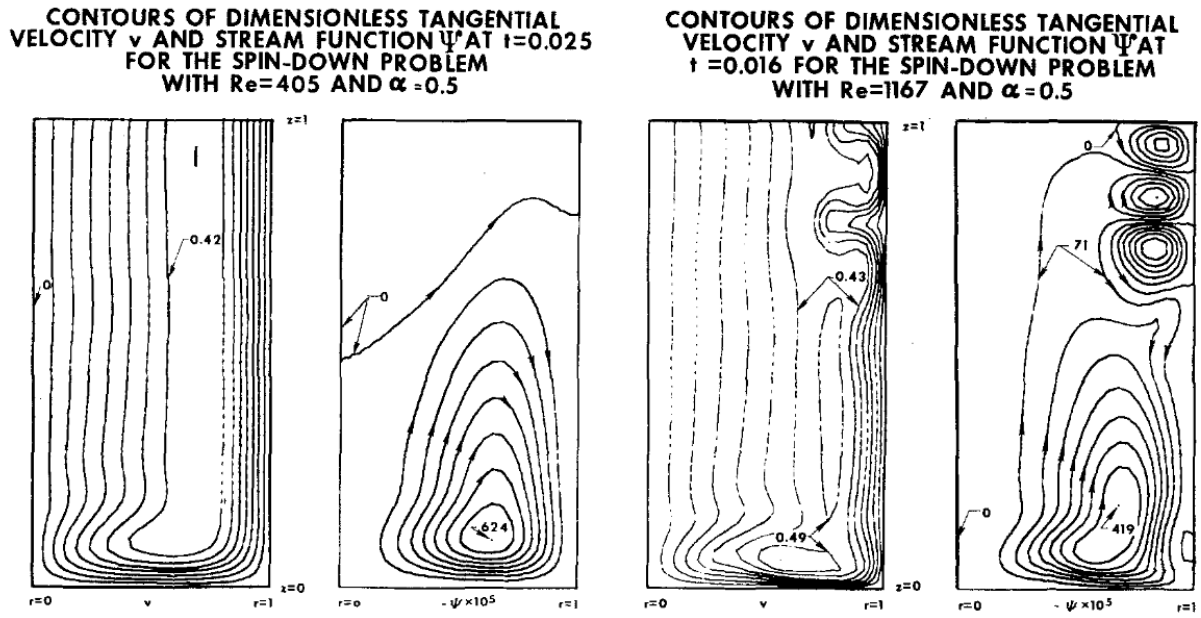


Figure 14: Stream functions for a low and moderate Reynolds number after spin-down at two different times. Adapted from Briley and Walls (1971)

More recently, Kaiser et al. (2019) tries to describe the spin-down process through all its phases. These include the onset of centrifugal instabilities, as well as the decay of turbulence. The stages of spin-down described by the author are presented in Figure 15.

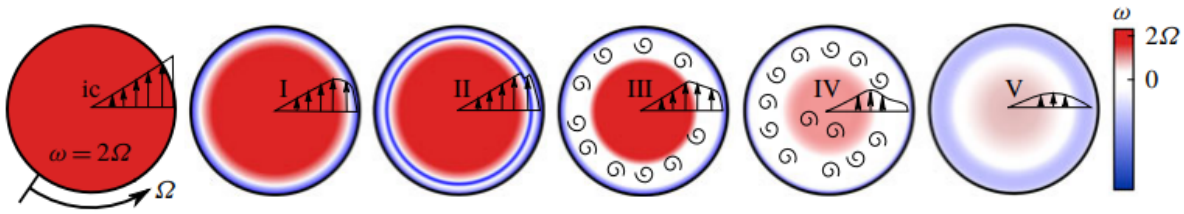


Figure 15: Schematic representation of the evolution of vorticity (colour contours) and azimuthal velocity (vector arrows) during the spin-down process in a rotating cylinder. The flow progresses through distinct stages: (I) initial laminar state, (II) onset of instabilities and transition, (III) development of turbulent structures with a coherent vortex core, (IV) decay of turbulence and breakdown of the vortex core, and (V) relaminarisation. Spiral patterns indicate the presence of turbulent fluctuations during the transitional stages. Adapted from Kaiser et al. (2019)

Kaiser et al. (2019) states that a complete breakdown of all stages of the spin-down process is yet to be studied as a whole. In the past, work by other authors has covered both numerical and experimental methods. Up to stage II (instabilities and transition to turbulence), some work tried to use both analytical solutions and experimental methods. For Reynolds numbers up to 28,000, Kaiser et al. (2019) uses a Direct Numerical Simulation (DNS) for an infinitely

long cylinder. Figure 16 shows a finite-time Lyapunov exponent (FTLE) visualisation for Re=12000 at various times. The Lyapunov exponents are a measure of the instability of chaotic objects. One can visualise the flow by identifying which modes grow along the trajectory of the time-dependent system.

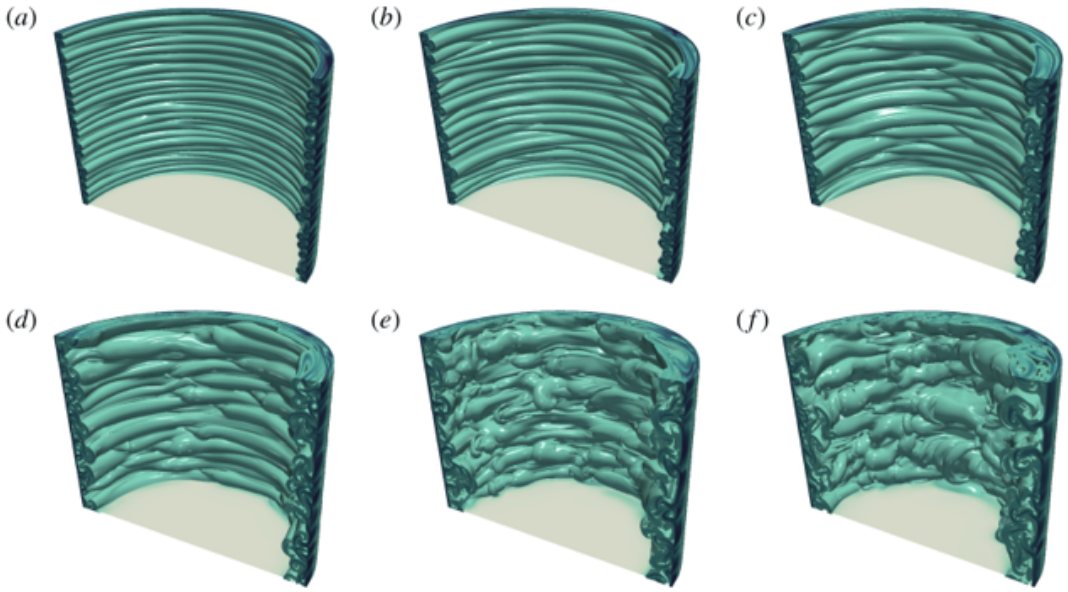


Figure 16: Finite-time Lyapunov exponent (FTLE) visualisation of flow structures during spin-down at $Re = 12000$, highlighting regions of particle separation. Adapted from Kaiser et al. (2019).

In Figure 16, the stages of temporal development are as follows:

(a) $\Omega t = 4.3$, primary instability;
(b) $\Omega t = 4.8$, asymmetric growth of primary instability;
(c) $\Omega t = 5.3$, observable streamwise ends of streamwise vortices;
(d) $\Omega t = 5.8$, onset hairpin-like vortices;
(e) $\Omega t = 6.8$, corruption of streamwise vortices;
(f) $\Omega t = 7.8$, late transition.

The extent of turbulence at various stages is dependent on the Reynolds number of the flow. For PFS devices, the maximum Reynolds number is limited to the robotic inspection machines, and therefore the stability of spin-down is yet to be determined. Despite this work focused on the flow field during the spin-up and spin-down of a cylinder, there is no application to particle motion within these flow fields. Moreover, PFS are finite cylinders and the boundary layer effects from the top and bottom walls on the fluid flow during spin-up or spin-down are not studied. This gap in the literature is significant for understanding the fluid flow for the inspection of PFS. This project aims to study these effects and link the fluid flow to particle tracking during the spin-up and spin-down phases.

2.5 CFD and Particle Tracking

CFD is a fundamental tool for investigating flow behaviour in geometries where analytical or experimental approaches become limited. In essence, CFD works by solving the Navier-Stokes equations using numerical solutions. This is achieved by using discretisation methods such as finite difference, finite volume, and finite element techniques to satisfy the conservation of mass and momentum across a defined computational domain. Each method works by transforming the governing equations into algebraic form over discrete spatial and temporal domains. The most commonly used method is the finite-volume method because of its inherently conservative properties. The Finite-Element Method provides greater geometric flexibility at higher costs, while the finite difference schemes are mainly suited for structured grids and simpler domains (Ferziger et al., 2020). The finite volume method is used in commercial and research CFD codes such as FLUENT and OpenFOAM and are particularly robust for solving compressible and incompressible flow problems Versteeg and Malalasekera (2007). For this project, the domain is the PFS represented as a cylinder (more on this in Chapter 3). Modern solvers enable detailed exploration of unsteady, three-dimensional, rotational flows that are otherwise difficult to visualise experimentally.

Analytical and experimental studies such as those by Wedemeyer (1964), Weidman (1976*a*) and Weidman (1976*b*), provided valuable insight into the spin-up and spin-down behaviour of rotating cylinders. However, these approaches are limited in resolving full, transient three-dimensional structure of the flow. The emergence of CFD offers a means of solving the Navier-Stokes equations and enabling complete visualisation of the velocity and pressure fields during the rotational transients.

Early numerical investigations by Briley and Walls (1971) and Hyun et al. (1983) demonstrated that CFD is more than capable of reproducing the transient evolution of the flow fields in such rotational flows including the formation of Ekman and secondary vortices. These studies presented the first computational attempts for Wedemeyer's theoretical model. More recently, high-fidelity Direct Numerical Simulation (DNS) such as the work by Kaiser et al. (2019) has resolved the onset of centrifugal instability and turbulence decay with three-dimensional resolution.

CFD is an essential tool for better understanding the fluid flow with a PFS. So far, none of the existing literature has coupled CFD with particle motion analysis within vertically rotating cylinders, such as those representing a PFS. In addition to resolving the fluid flow, CFD can be extended to model trajectories of particles through a Lagrangian particle tracking approach. Most commonly, the discrete particle method (DPM) or discrete element method (DEM) is used, Sommerfeld et al. (2019). In this framework, individual particles are treated as point masses and their motion is obtained through the use of Newton's second law under hydrodynamic forces such as drag, lift, gravity or buoyancy. DEM is used for particle-particle interactions whereas DPM is particularly used for dilute, non-contact-dominated flows. This technique enables the investigation of how transient structures influence the movement of particles and can be extended to a rotational system such as a PFS inspection study.

2.6 Visual Inspection Methods

Part of this work will be the use of experimental methods in conjunction with CFD to understand fluid behaviour during the inspection process of a PFS device. Experimental rigs with visual inspection can be built in both an academic and an industrial environment. In this section, we aim to cover various inspection methods.

2.6.1 Syringe Damage

Visual inspection has long been applied across a wide range of industrial processes. For example, to detect corn-plants diseases, assess seed quality, and sort fruits using rule-based vision systems (Jia, 2009). In the context of PFS, Jia (2009) developed a laboratory visual inspection rig that focuses on geometric parameters such as barrel fixture locations, flange angle, and thumb-tab to shroud distance. The inspection machine uses a ten-camera assembly line setup that can inspect 300 to 600 parts per minute. Due to their inspection rate, automatic inspection machines are highly desirable as manual inspection has clear limits. Another example of successful replication of an inspection process for particle tracking as well as other defects is Zhou et al. (2008). However, in this case, instead of PFS devices, injection vials are inspected. The process for the inspection techniques overlaps with the inspection of PFS devices, and the inspection rig can inspect more than 8000 vials per hour.

Within the industrial setting, many different automatic inspection machines are used. Automated particle inspection systems within these machines originated in the 1970s with Static Division systems developed by Eisai Machinery (Baczewski, 2015). This specific system worked by differentiating static images from moving objects. The process involves the transmission of lights through the solution with the container onto an optical sensor. Any particles inside the PFS devices can block a portion of the light and cast a shadow. This shadow is then detected by the sensors. More advanced methods now use low and high-speed cameras to take a sequence of images. These images are then analysed by algorithms specifically designed to filter through to the particle and plot their trajectories (Baczewski, 2015).

2.6.2 Particle Inspection

An example of an academic system is one of a master group at the University of Leeds by Hannam et al. (2016). The apparatus shown in Figure 17 has been used by Hannam et al. (2016) to perform an automated inspection of a PFS device. Using the apparatus, one can potentially recreate the inspection method in a laboratory setting. In the experimental setup, a motor rotates the syringe body at the prescribed RPM. The camera mounted alongside the LED panel is used to track any external particles within the syringe. Particle tracking techniques, such as the background subtraction algorithm, are also applied to the images recorded to calculate a particle path during the inspection process. Various types of image processing techniques are used that are proprietary to a particular manufacturer. A visual analysis of the background subtraction algorithm used by Hannam et al. (2016) is shown in Figure 18.

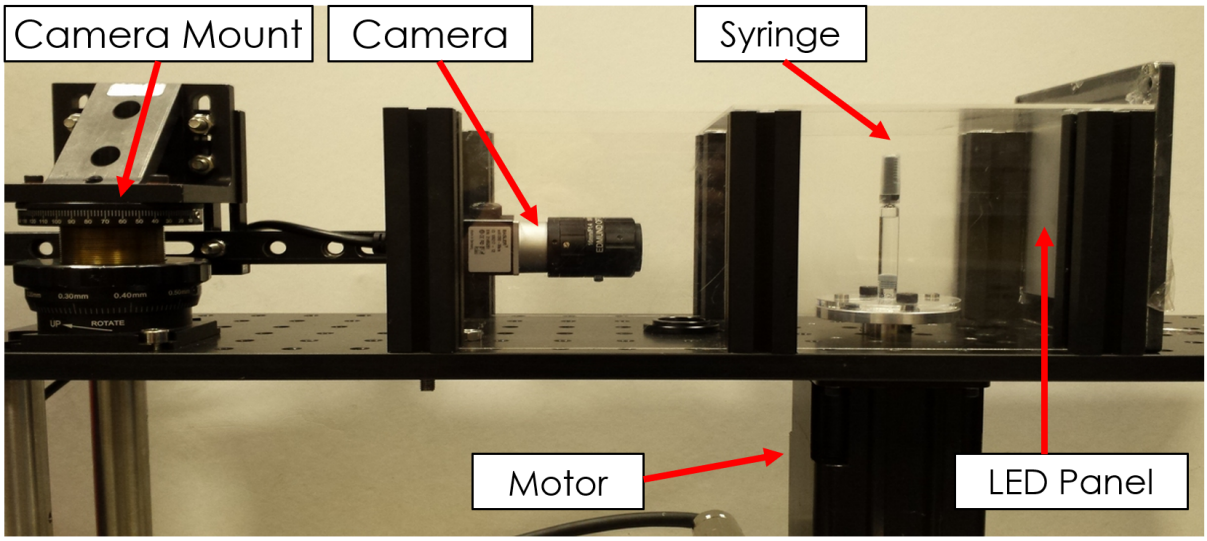


Figure 17: Schematic of the the inspection apparatus by Hannam et al. (2016)

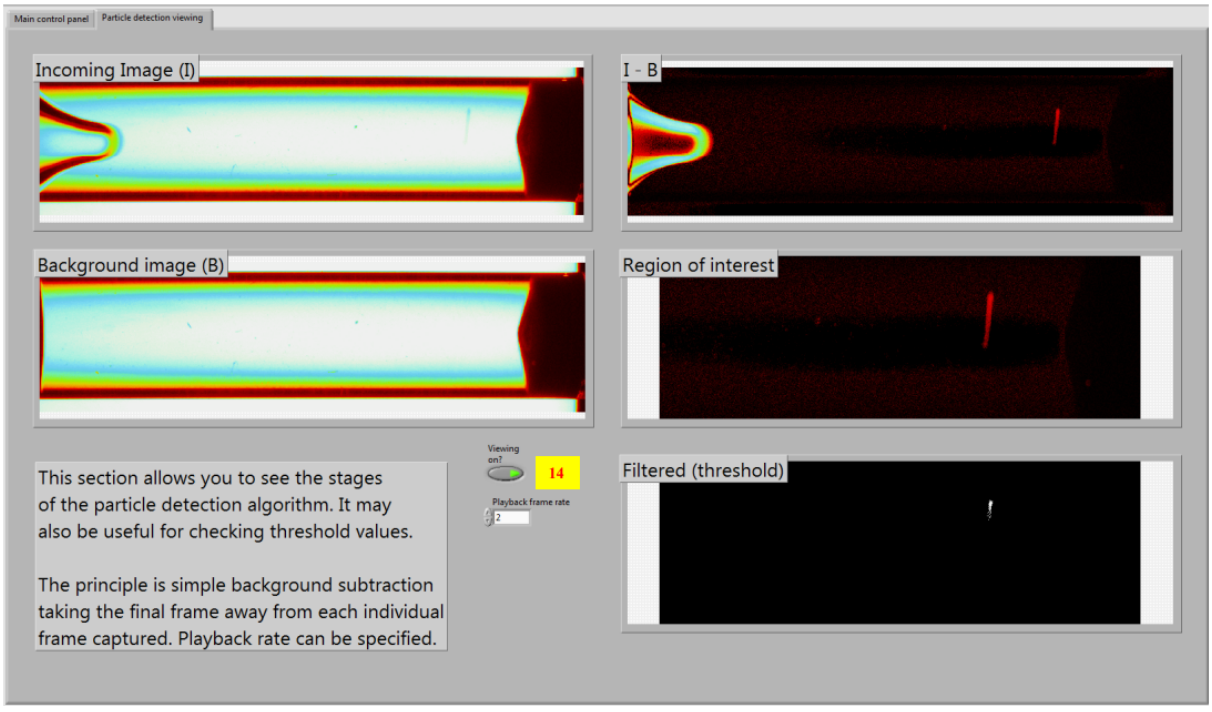


Figure 18: Screenshot of Background subtraction algorithms viewing panel from Hannam et al. (2016)

The background subtraction algorithm works by subtracting an incoming image from a background image stored at an earlier time. Once the subtraction is performed, the region of interest is selected. In Figure 18, this is the red region where the particle is present (as opposed to the vortex formation). The algorithm can be used for more than just particle tracking. Figure 19 shows the background subtraction result as well as the use of the algorithm to detect various features within the syringe. These include plunger, vortex, and particle detection. The general idea within particle tracking is to fluidise particles and monitor them as they move slowly through the frame. By taking various images, one can subtract one frame from another. If there is a moving particle, it will appear as a white dot and will be detected through background subtraction algorithms. This white dot can then be tracked across different frames to produce a trajectory plot (as shown in Figure 19).

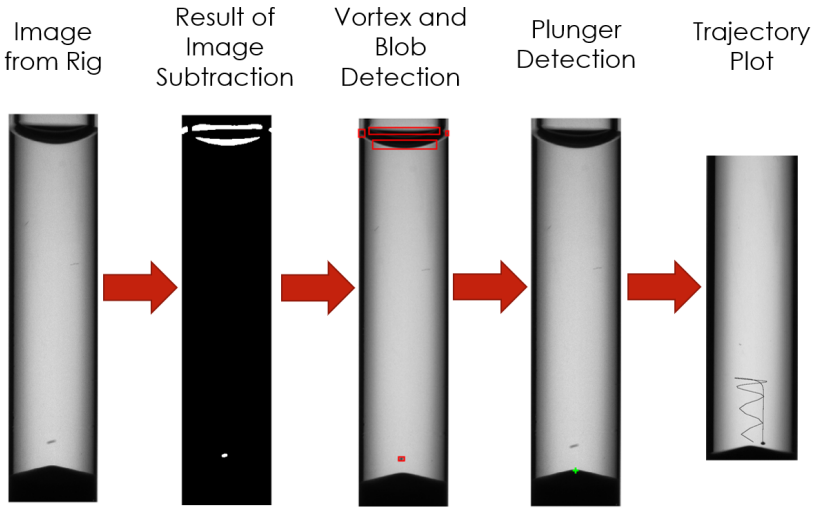


Figure 19: Example of how background subtraction algorithms have been used by Hannam et al. (2016)

Background subtraction algorithms used in the industry follow the same fundamental principles, but are a lot more complex in their functionality. For example, detecting changes between sequential images to identify moving particles is difficult, but commercial machines incorporate high-speed multi-camera assembly lines integrated with advanced illumination control and real-time classification algorithms to achieve rapid and automated detections. Therefore, the method used in this lab-controlled environment represents a simplified, research-orientated implementation of the same core ideas of image detection.

This project aims to build on the model by Hannam et al. (2016) and use it to complement the CFD model. In doing so, the advantages of both experimental and computational methods can be employed to understand the inspection process of particles within a PFS device.

2.6.3 Vial Inspection

The literature for the use of CFD in syringe inspection is quite scarce. One particular work (computational and experimental) by Asadi et al. (2025) has been found that looks at spin-stop and how it helps with the detection of particles. Specifically, the study models the wall shear stress (WSS) distribution for a standard vial to determine how agitation can optimise particle mobilisation. This is different in a way because it is a vial and not a syringe, but is still applicable because mathematically, both can be modelled using a cylinder with various heights. It is found that a higher angular acceleration during spin-down maximises the WSS distribution across the interior surface. This results in maximum mobilisation of the particles contained within the vial. This work focusses on agitating particles rather than trying to understand a typical track a particle might take. In our study, we will try to determine that using various spin parameters.

2.7 Summary of Literature Review

During the initial literature search, a wide range of research has been covered. For example, the existence of sub-visible particles due to silicon oil or tungsten within a PFS device. The process of aggregation has also been covered in depth for protein-based solutions. It has also been found that there seems to be a gap in the literature regarding the testing of these particles. This includes both the inspection process of fluidising particles and the use of this for detection. To be more specific, there has been limited literature on the use of CFD and particle tracking for this type of inspections. The focus during the literature review was to gain an understanding of the PFS market before any work is done within a specific area, such as the role of fluid dynamics in inspection. As for the fluid dynamics, the basic model of Wedemeyer for the spin-up of a cylinder has been studied alongside its various extensions. The numerical method approach has also been studied in the literature. A gap found in the literature within rotational flows has to do with the motion of particles within these flow fields. Experimental and computational methods are yet to be seen for particle motion within the spin-up and spin-down of a cylinder.

3 Materials and Methods: Computational Methods

3.1 Introduction

In this chapter, the numerical approach used to simulate the fluid flow and motion of the particles within a prefilled syringe during its inspection is described. The study investigates the transient flow dynamics of a rotating cylindrical syringe, subject to spin-up and then followed by an abrupt stop. This will cover the various methods selected to be able to conduct a numerical study that captures the flow during both spin-up and spin-down. Due to the complexity of the flow observed, an analytical solution is impractical, as extending the solution to model particle flow is extremely challenging. CFD provides an efficient means to analyse the behaviour of rotational flows and its effects on fluidised particles. For this project, COMSOL Multiphysics has been selected due to its use of the finite element method (FEM) and its ability to efficiently couple fluid flow with Lagrangian particle tracking. The software allows the problem to be formulated in a 2D axisymmetric domain, while still capturing three-dimensional particle trajectories. This significantly reduces computational cost compared to fully three-dimensional simulations, while retaining the key flow physics relevant to particle transport during spin-up and spin-down.

COMSOL Multiphysics has been selected due to its ability to handle coupled CFD as well as particle tracking simulations effectively. The software enables high-fidelity modelling of transient flow dynamics with flexible meshing and solver options. COMSOL also has a powerful toolkit for post-processing both 2D and 3D plots in a compact manner. It also has the capability of running 2D simulations and mapping the results to produce particle tracks in 3D space.This Allows for a less computationally expensive study while producing reliable 3D particle flow results. This is a feature that will be used to determine particle-flow behaviour. To obtain a solution, a structured workflow tailored to this study is followed:

1. Definition of the computational domain using a 2D axisymmetric representation of the syringe geometry
2. Specification of fluid properties and rotational conditions, including spin-up and abrupt spin-down profiles
3. Mesh generation with refinement near the walls to resolve boundary layer effects
4. Discretisation of the governing Navier–Stokes equations using COMSOL's finite element formulation
5. Application of boundary and initial conditions, including no-slip conditions at the walls and prescribed rotational motion
6. Solution of the transient flow field during spin-up and spin-down
7. Coupling of the flow solution with Lagrangian particle tracking to simulate particle trajectories
8. Post-processing and analysis of velocity fields and particle motion

3.2 Fluid Flow Setup

3.2.1 Problem Setup

To perform the numerical study, a simplified representation of the syringe geometry is defined as the domain in which the fluid flow is solved. A cylinder with height h and radius a, spinning about its axis with angular velocity ω, is considered, as shown in Figure 20. During inspection a PFS device can be modelled using a polar coordinate system where (r, θ, z) are the radial, azimuthal, and axial coordinates, respectively. Similarly, the radial, azimuthal, and axial velocity components are u, v, and w, respectively.

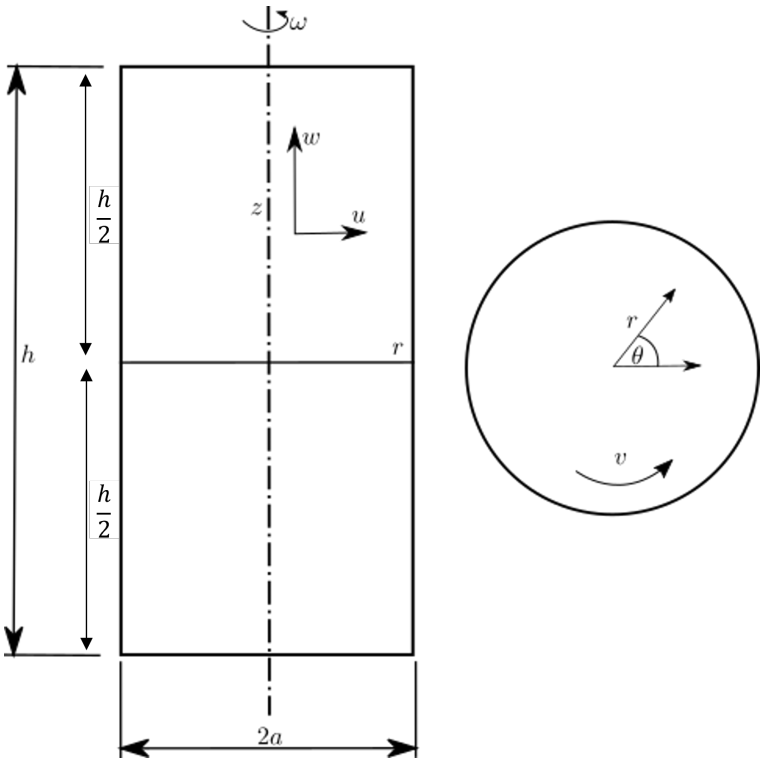


Figure 20: Schematic of the problem using a polar coordinate system adapted from Wedemeyer (1964)

In this model, several geometric simplifications to the PFS are applied. The plunger and needle are omitted, and it is assumed that the domain is completely filled with fluid bounded by flat top and bottom surfaces. These assumptions remove the influence of the free surface and plunger deformations, allowing the flow field to be analysed in a controlled manner. In practice, small plunger motion and free surface effects may occur. However, these are expected to have a limited influence on the dominant rotational flow and bulk fluid behaviour during spin-up and spin-down. The flat top and bottom boundaries therefore provide a reasonable approximation for capturing the primary flow dynamics relevant to the inspection process. Moreover, the plunger is manufactured with tight tolerances, which significantly restrict its motion and minimise its influence on the flow.

3.2.2 Definitions & Non-dimensionalisation

This subsection highlights the definitions of the terminology used in the upcoming chapters. The most important non-dimensional parameter to consider is the Reynolds number. This is a measure of the inertial forces relative to the viscous forces. A large Reynolds number implies that inertia dominates over viscous effects. The Reynolds number for a cylindrical body can be calculated using Equation 3.

Another essential non-dimensional parameter is the aspect ratio, AR. It is given by

$$AR = H/D \tag{10}$$

where H is the height (mm) and D is the diameter (mm) of the syringe. The aspect ratio physically represents the geometric slenderness of the syringe, indicating relationship between its height and diameter. This parameter is an important geometric factor in several of the studies presented in this thesis.

It is also necessary to consider non-dimensional velocity and radial positions obtained in the results. This allows for a much more effective discussion for the results obtained. The velocity magnitude and the radial position are non-dimensionalised using the spin speed and radius of the cylinder. This can be achieved using:

$$\bar{u} = \frac{u}{\omega a} \qquad , \qquad \bar{r} = \frac{r}{a} \tag{11}$$

where u is the velocity magnitude ($\mathrm{m\,s^{-1}}$), r is the radial position (m), ω is the angular velocity ($\mathrm{rad\,s^{-1}}$), and a is the radius of the cylinder (m).

3.2.3 The Computational Domain

The computational domain is created in COMSOL Multiphysics using a 2D axisymmetric representation of the syringe. Using a 2D axisymmetric study reduces the computational cost but retains the essential flow physics as the problem is rotationally symmetric. The geometry dimensions were selected to resemble a typical syringe used in the industry. These dimensions are provided with permission of a major UK pharmaceutical manufacturer for a particular syringe. These parameters represent a 1 mL standard pre-filled syringe configuration but do not encompass all possible industrial geometries. For other typical syringe dimensions, refer to International Organization for Standardization (2015). While there is variation in syringe geometries across manufacturers, this is largely driven by the required fill volume (e.g., 0.5 mL, 1 mL), which primarily affects the barrel dimensions while maintaining a similar cylindrical configuration. The dimensions of the syringe used in the computational setup are given in Table 2. The kinematic viscosity, ν, is related to the dynamic viscosity, μ, and density, ρ, and is given by $\nu = \mu/\rho$. This relationship is used within COMSOL when defining the fluid properties.

Table 2: Some of the values for the input parameters used in the setup (data obtained from a major UK pharmaceutical manufacturer)

Parameter	Symbol	Value
Cylinder height (mm)	h	33.02
Cylinder radius (mm)	a	3.175
Acceleration due to gravity ($\mathrm{m\,s^{-2}}$)	g	9.81
Angular velocity ($\mathrm{rad\,s^{-1}}$)	ω	104.72
Fluid density ($\mathrm{kg\,m^{-3}}$)	ρ	1035.4
Fluid dynamic viscosity (Pa s)	μ	0.003

3.2.4 Boundary & Initial Conditions

Fluid properties such as dynamic viscosity and density are selected on the basis of typical products within a PFS device. The fluid being modelled is Newtonian and incompressible. It is assumed that the flow is purely driven by wall motion (rotational velocity). The model selected is also isothermal, so buoyancy effects are not considered. Moreover, the fluid is also considered to be laminar, and no turbulence effects are studied. This is a modelling assumption used to isolate the primary flow behaviour, rather than a strict representation of all possible flow regimes. There are no inlet/outlet boundary conditions as the flow is driven solely by rotation and the fluid is enclosed within the cylindrical syringe. Figure 21 shows the geometry where the axis of rotation is shown in red. The walls highlighted in blue have the wall motion set to $v = \omega r$ where ω is the angular velocity. For spin-up, the wall rotation is set to a fixed rotational speed (i.e. $\approx 100\,\mathrm{rad\,s^{-1}}$) for 1 second. The cylinder is very rapidly decelerated to zero velocity. The function $\omega(t)$ is shown in Figure 22.

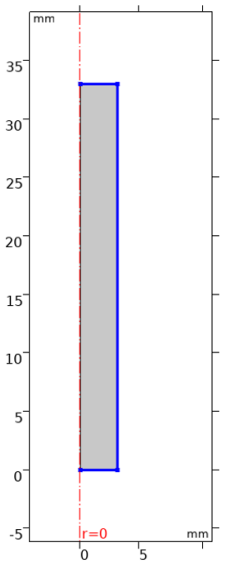


Figure 21: Axisymmetric geometry where the walls are indicated by the blue lines and the red line represents the central axis

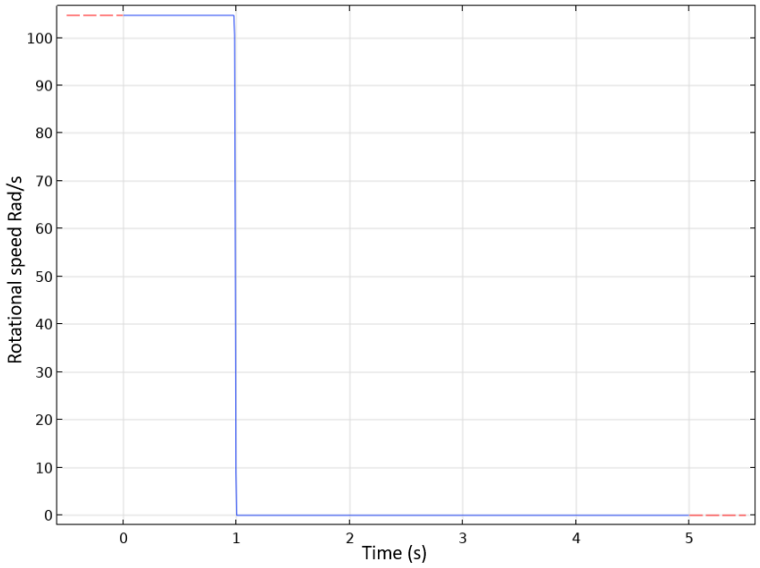


Figure 22: Function $\omega(t)$ for angular velocity at given times during spin up and spin down

In Figure 22, the sharp drop at $t = 1\,\mathrm{s}$ is modelled as an instantaneous reduction to zero angular velocity, representing an idealised abrupt stop. This simplification enables a clear distinction between the spin-up and spin-down phases. In reality, a physical motor would decelerate over a finite time due to inertia and control system limitations, typically within a fraction of a second depending on load and operating speed. The effect of varying deceleration rates is not considered in this study, as the deceleration profile is kept constant for both the CFD and experimental investigations to isolate the primary flow behaviour.

Summary of boundary and initial conditions;

1. Centreline: At $r = 0$, the symmetry condition is set to ensure that there is no flow across the axis $u_r = 0$ (i.e., the velocity components remain symmetric about the axis).

2. Outer wall of the cylinder:

   - at $r = a$, no-slip condition with wall motion is set as the wall is rotating with the angular velocity function $\omega(t)$. $u = 0, w = 0, v = \omega a$. The function $\omega(t)$ represents the spin-up and spin-down process.
   - at $r = a, z = -h$, pressure constraint at a single point on the outer wall the pressure is set to $p = 0$ Pa (Gauge Pressure). COMSOL requires this to define a unique solution (since incompressible flow has no absolute pressure reference). This pressure constraint is applied at a single reference point, preventing numerical singularities.

3. Top and Bottom Walls: at $z = 0, z = h$, the no-slip condition with wall motion is applied. $u = 0, w = 0, v = \omega r$.

3.2.5 Meshing Strategy

A good meshing strategy is essential for an accurate numerical study. The shape elements in a mesh can be a line (i.e. a line between nodes in 1D), a surface (i.e. 2D shapes such as triangles or quadrilaterals) or solid elements (3D tetrahedral or hexahedral). Therefore, the generation of the mesh can be structured or unstructured depending on the types of elements used. Unstructured meshes can be used for complex shapes, whilst structured meshes tend to solve faster and provide greater control of the grid quality (Ferziger et al., 2019).

Depending on the geometry, the type of mesh needs to be determined. In this study, since a 2D axis-symmetric geometry is used, a structured approach using quadrilateral elements is selected. This allows the mesh to be generated more easily and focuses refinement in areas of high gradients. In doing so, more control is obtained within the meshing process, and the quality of the mesh is further improved. Moreover, a structured mesh reduces computational costs and memory usage due to its regular connectivity and more efficient matrix assembly, while providing a uniform distribution of nodes (Ferziger et al., 2019). The refinement process on structured meshes is also easier compared to that on unstructured meshes, as the mesh distribution can remain constant even with increasing mesh elements. The focus during the meshing procedure is to capture the flow near the centre of the cylinder where the particles are expected to lift. This region corresponds to the expected jet of fluid that forms during spin-down. In addition, boundary layers are expected to form around all walls. Therefore, a higher density of mesh elements is required to accurately resolve the shear layer flows near these walls (Ferziger et al., 2019). The regions in between do not require a higher mesh density, and to reduce computational time, a bias has been set to distribute the mesh along the required regions. This mesh distribution along the cylindrical geometry is presented in Figure 23, where the axial and radial distribution of the increasing mesh elements is shown. A mesh independence study for the PFS problem is outlined in Section 3.5.

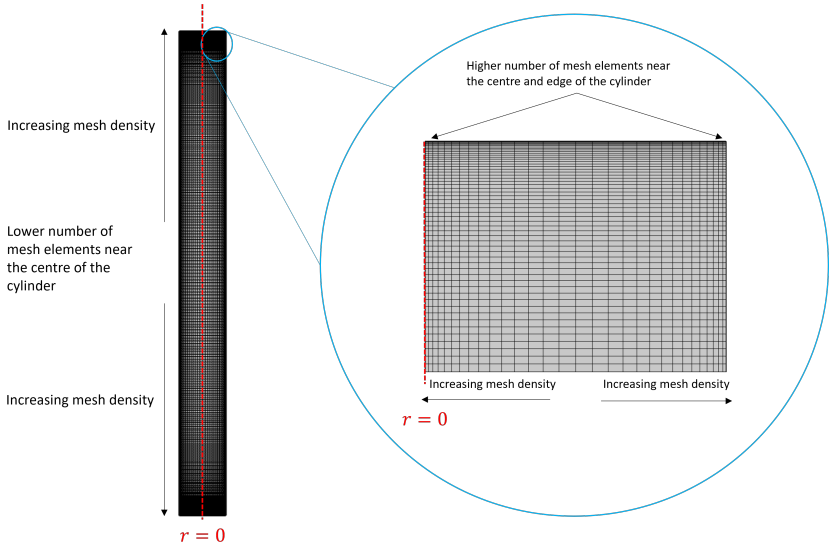


Figure 23: Mesh distribution along the cylinder

3.2.6 Governing Equations and The Weak Formulation

For a solution to the Navier-Stokes equations using the finite element method, the weak form is derived by multiplying the governing equations by test functions and integrating over the computational domain. This approach follows the standard finite element formulation of fluid mechanics and is consistent with the implementation used in COMSOL Multiphysics (COMSOL AB, 2021). Consider the incompressible Navier-Stokes equations in their strong form:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \quad \text{in } \Omega \tag{12}$$

$$\nabla \cdot \mathbf{u} = 0, \quad \text{in } \Omega \tag{13}$$

where $\mathbf{u}$ is the velocity field, p is the pressure, ρ is the density, ν is the kinematic viscosity and $\mathbf{f}$ represents body forces.

Consider the test functions $\mathbf{v}$ (for momentum conservation) and q (for mass conservation) and integrate it over the domain Ω:

$$\int_\Omega \mathbf{v} \cdot \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) d\Omega - \int_\Omega \mathbf{v} \cdot \left(-\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \right) d\Omega = 0 \tag{14}$$

$$\int_\Omega q \nabla \cdot \mathbf{u} \, d\Omega = 0 \tag{15}$$

For the pressure term, using the divergence theorem and applying integration by parts results in

$$\int_\Omega \mathbf{v} \cdot (-\nabla p) \, d\Omega = \int_{\partial\Omega} (-p\mathbf{v} \cdot \mathbf{n}) \, d\Gamma + \int_\Omega p(\nabla \cdot \mathbf{v}) \, d\Omega \tag{16}$$

where $\mathbf{n}$ is the outward normal in $\partial\Omega$.

Similarly, applying integration by parts to the viscous term results in

$$\int_\Omega \mathbf{v} \cdot \nu \nabla^2 \mathbf{u} \, d\Omega = -\int_\Omega \nu \nabla \mathbf{v} \cdot \nabla \mathbf{u} \, d\Omega + \int_{\partial\Omega} \nu (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \mathbf{n} \cdot \mathbf{v} \, d\Gamma \tag{17}$$

Substituting the above results, the final weak form is:

$$\int_\Omega \mathbf{v} \cdot \frac{\partial \mathbf{u}}{\partial t} \, d\Omega + \int_\Omega \mathbf{v} \cdot (\mathbf{u} \cdot \nabla \mathbf{u}) \, d\Omega - \int_\Omega p \frac{1}{\rho} \nabla \cdot \mathbf{v} \, d\Omega + \int_\Omega \nu \nabla \mathbf{v} \cdot \nabla \mathbf{u} \, d\Omega = \int_\Omega \mathbf{v} \cdot \mathbf{f} \, d\Omega \tag{18}$$

$$\int_\Omega q \nabla \cdot \mathbf{u} \, d\Omega = 0 \tag{19}$$

This weak form has reduced from a continuous partial differential equation (PDE) to a system of semi-discrete ordinary differential equations. COMSOL uses the weak form to discretise in both space and time.

3.2.7 Spacial Discretisation and the Finite Element Method

Initially, COMSOL discretises the weak form in space over a number of smaller finite elements that are connected at nodes using the Finite Element Method (FEM). FEM is widely used in the engineering industry to solve numerical problems that usually consist of irregular geometries, multiphysics interactions, and high-fidelity models. The collection of nodes and elements is what is defined as the mesh. For FEM, the rewritten weak (integral) form is applied to each element. Between the nodes, it is assumed that the function to be solved is expressed by a series of predefined functions, these are called shape functions. Shape functions can be linear or higher-order polynomials. Since the shape functions are known, one can use the values of the shape functions at the nodes to determine the values of the shape function throughout the domain. The entire information about the function is represented by a finite number of elements. Hence, this reduces the infinite number of unknowns in a continuous function to a finite number of unknowns. This procedure is called discretisation. Since this is an approximation, an inherent error known as the discretisation error is introduced. This error is known to decrease with increasing elements, and a mesh independence study is essential to ensure minimisation of this error.

The elements can be of first, or higher order. For the study, a P2 + P2 element option is used. This is the second order elements for both velocity and pressure. These offer a higher accuracy for flow problems compared to P1 + P1 elements (first order for velocity and pressure), which may require fewer computational resources but with reduced accuracy. With second-order elements, numerical diffusion is reduced resulting in a more stable solution. Figure 24 shows various 2D and 3D elements with their nodes. Higher-order elements consist of the solution at a higher number of nodes and improve the accuracy of the solution.

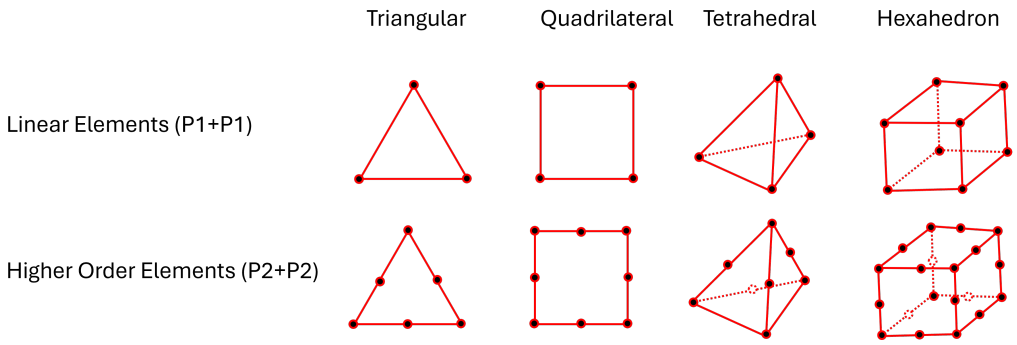


Figure 24: Examples of 2D and 3D elements with their nodes (shown as black circles) for linear and higher order elements

3.2.8 Temporal Discretisation (Time-Stepping Method)

As the study needs to consider the temporal evolution of the flow field, the weak form needs to be discretised with time as well. This is done using the time-stepping method. There are two types of solver that determine which method will be used to discretise the weak form. These are either implicit or explicit solvers (Ferziger et al., 2020). Explicit solvers compute the next time step directly from the current state using known values from the previous time step. The implicit time-stepping method computes the next time step using both the current value and the future value. This requires solving a system of equations for each time step. Explicit time-stepping schemes are conditionally stable, with the allowable time step restricted by the Courant–Friedrichs–Lewy (CFL) condition, whereas implicit schemes are generally more stable and permit larger time steps (Ferziger et al., 2020; Versteeg and Malalasekera, 2007). This distinction is particularly important for transient flow problems involving strong gradients and rapid temporal changes, where stability constraints can significantly limit explicit methods (Versteeg and Malalasekera, 2007). A stiff problem is one in which the governing equations contain processes that evolve on significantly different time scales, requiring very small time steps for stable integration when using explicit methods, even if the overall solution varies slowly. If the problem is not stiff, small time steps are acceptable, and the computational cost per step is critical, explicit solvers are a better option. However, the flow exhibits multiple temporal scales during spin-up and spin-down, making stability an important consideration. Therefore, an implicit time-stepping method is chosen.

There are two methods available within COMSOL for fluid flow when the implicit solver type is selected. These are Generalised-α and Backward Differentiation Formula (BDF) (COMSOL AB, 2021). Generalised-α introduces artificial damping, but is less stable for the Navier-Stokes equations. BDF for time stepping is stable and provides adaptive time stepping. This is when the solver automatically adjusts the time step size, i.e. uses smaller time steps when flow changes rapidly. However, it is computationally expensive. Backward Differentiation Formula (BDF) methods are widely used for transient simulations due to their stability and suitability for stiff systems, particularly in fluid flow applications (Ferziger et al., 2020).Therefore, For PFS inspection studies, BDF has been selected.

3.2.9 Solving the Discretised Equations

The system of equations that need to be solved can be done through two types of methods in COMSOL (COMSOL AB, 2021). One method uses a segregated step approach, which generates a set of algebraic equations for all the physics models. The segregated approach solves each variable, that is, the velocity first and then pressure. However, this is done separately at each time step. This method can struggle with convergence for complex flows but usually requires less memory, but is much slower in contrast with the second type of method, a fully coupled approach.

The fully coupled method generates a single set of algebraic equations that are solved for all physics models simultaneously. This is then implemented in a single iteration scheme, which

is repeated until convergence has been achieved. This method converges very quickly, but requires a lot of memory to store the entire system of equations for each time step. It is also better for strongly coupled physics (like incompressible Navier-Stokes). It is computationally expensive, but increases stability. Since the PFS inspection involves transient rotational flows, shear layer evolution & potential instabilities as well as an axisymmetric formulation, the flow requires strong coupling of velocity and pressure. As suggested by COMSOL AB (2021), a fully coupled method is selected for the PFS inspection problem.

Finally, the fully coupled system needs to be solved at each time step. This is done through the use of linear solvers. There are three direct solvers to be selected in COMSOL from the previous setup (COMSOL AB, 2021). These are:

1. Multifrontal Massively Parallel Solver (MUMPS)
2. Sparse Object-Oriented Linear Equations Solver (SPOOLES)
3. Parallel Direct Sparse Solver (PARDISO) (Common for CFD and Finite Element Methods)

MUMPS works well for parallel computing, as it can handle distributed computing and is suitable for large problems. It is slower than PARDISO and can have memory scaling issues in extreme cases. SPOOLES is an alternative to PARDISO but is not as widely used and is not optimised for modern CPUs. PARDISO is efficient for sparse matrices and is best for small to medium size problems in which memory is not an issue. Although it has a high memory consumption and scales poorly for large problems, it is stable and handles incompressible flows efficiently. Therefore, it has been selected for the PFS inspection study.

3.3 Lagrangian Particle Tracking Model

So far, a setup has been modelled that solves for fluid flow within the inspection of pre-filled syringes. The results obtained from this study are used to determine the flow of particles within these fluid flow fields. The motion of particles in the context of inspection, is as important as the fluid flow itself. The main purpose of inspection is to determine the existence of unwanted particles. The fluid is subjected to a particular velocity field so that particles can be fluidised and detected. A second study is therefore conducted to solve for the particle motion resulting from the previously obtained fluid flow fields. These results are obtained by mapping the results of the velocity and pressure fields from the fluid flow study and solving for the particle trajectories. Keeping in mind that the fluid study is 2D axisymmetric, the 2D data from the previous study needs to be mapped to 3D particle tracks. This section provides the setup used for a Lagrangian Particle Tracking model (LTP). An LPT approach resolves individual particle trajectories instead of a Eulerian approach, which looks at track concentrations. Since individual particles are relevant for this PFS studies, an LPT approach is ideal. Moreover, the system is setup using one-way coupling. This means that the fluid flow affects the motion of the particles, but not vice versa. This assumption is justified because the particles are small, and have densities close to that of the surrounding fluid and exist at very low concentrations. Under such conditions, the momentum exchange

from the particles to the surrounding fluid is negligible (Balachandar and Eaton, 2010). The one way coupling assumption provides a valid and computationally efficient approximation for this study. Consequently, the motion of the particles does not influence the flow of the dominant fluid within the small window of particle inspection.

3.3.1 Particle Motion Formulation

The general governing equations for the particle motion are Newton's Second Law for particle motion and these can be represented by (Maxey and Riley, 1983):

$$m_p \frac{d\mathrm{v}}{dt} = F_t \tag{20}$$

where m_p is the mass of the particle (kg), v_p is the velocity of the particle ($\mathrm{m\,s^{-1}}$), and F_t is the total force acting on the particle (N). The total forces acting on the particle consist of the drag force, the virtual mass force, gravity, the buoyancy force, and lift.

3.3.2 Forces Acting on Particles

There are various forces that can potentially act on the particle if selected in the COMSOL model. One of such forces is the drag force. This is the force that acts in the direction of the flow and opposes motion. For the drag force, the Schiller-Naumann drag law is selected as it is valid for a wide range of particle sizes and Reynolds numbers. Although the Schiller-Naumann correlation was derived for an isolated sphere, it is widely used in dilute liquid–particle flows at moderate Re_p (defined in Equation 23). For small particles, the standard single particle drag law is adequate (Balachandar and Eaton, 2010). COMSOL also recommends it for liquid flows laden with particles and works well for both laminar and moderate turbulent flows. Other options include the Stokes drag law, but the particle sizes for the PFS studies are much larger than 1 µm and the Reynolds number is expected to be moderate to large. The equation of the drag force is given by the following:

$$F_{Drag} = \frac{1}{2} C_D \rho_f A_p (\mathbf{u} - \mathbf{v}_p) \mid \mathbf{u} - \mathbf{v}_p \mid \tag{21}$$

and

$$C_D = \frac{24}{Re_p}(1 + 0.15 Re_p^{0.687}) \tag{22}$$

where C_D is the drag coefficient, ρ_f is the fluid density ($\mathrm{kg\,m^{-3}}$), A_p is the projected cross-sectional area of the particle ($\mathrm{m^2}$), $\mathbf{u}$ is the fluid velocity ($\mathrm{m\,s^{-1}}$), $\mathbf{v}_p$ is the particle velocity ($\mathrm{m\,s^{-1}}$), and Re_p is the Reynolds number of the particle, defined as:

$$Re_p = \frac{\rho_f d_p \mid \mathbf{u} - \mathbf{v}_p \mid}{\mu_f} \tag{23}$$

where d_p is the particle diameter and μ_f is the dynamic viscosity of the fluid.

When a fluid accelerates a particle adding to its inertia, an added mass force is needed. This accounts for the additional resistance a particle experiences when it accelerates relative to the surrounding fluid. This is called the virtual mass force, and it is important when the particle density resembles the fluid density or when the relative acceleration between the particle and the fluid is significant. The virtual mass force F_{vm} is given by:

$$F_{vm} = C_{vm}\rho_f V_p \frac{Du}{Dt} \tag{24}$$

where C_{um} is the virtual mass coefficient (this is 0.5 for spherical particles), ρ_f is the density of the fluid, V_P is the volume of the particle and $\mathbf{u}$ is the fluid velocity, $\frac{Du}{Dt}$ is the rate of change of the fluid velocity following the particle)

Another force that contributes to the total forces experienced by a particle is the gravity force given by:

$$F_g = m_p g \tag{25}$$

where g is the acceleration due to gravity and m_p is the mass of the particle.

A particle suspended in a fluid also consists of a buoyancy force given by:

$$F_b = -\rho_f V_p g \tag{26}$$

where $-\rho_f$ is the density of the fluid, V_p is the volume of the particle and g is the acceleration due to gravity.

The final force needed is the lift force. This is the force experienced by a particle as a result of asymmetric flow around a particle causing the particle to move perpendicular to the flow direction. Lift forces can be induced by shear flows or rotating flows. There are two types of lift forces that can be selected in the COMSOL model. These are Saffman Lift force or wall-induced lift force. Saffman lift force is associated with shear-induced lift when the particle moves in a velocity gradient (i.e. shear flow) and is caused by the difference in fluid velocity that generates a net force perpendicular to the flow direction (Saffman, 1965), given by the following:

$$F_L = 6.46\rho_f \nu^{1/2} \mid \mathbf{u} - \mathbf{v}_p \mid (\nabla \times \mathbf{u}) \tag{27}$$

where ν is the fluid kinematic viscosity, $\mid \mathbf{u} - \mathbf{v}_p \mid$ is the relative velocity between the particle and the fluid and $\nabla \times \mathbf{u}$ is the fluid vorticity.

The wall-induced lift force is the best model for particles moving near solid boundaries that are lifted away from the walls. To study particles stuck to walls, this may be the best model. If the intention is to study particles in the flow that are generally away from the walls, then the Saffman lift force is better, as the flow consists of shear effects, especially near the centreline. In this study, the Saffman lift model is selected, as it is appropriate for shear-dominated flow conditions and small spherical particles at low Reynolds numbers. This model captures the lift induced by velocity gradients in the bulk flow, which is relevant for particles suspended away from solid boundaries.

3.3.3 2D to 3D Mapping (COMSOL Specific)

To use particle tracking, the 2D flow solution obtained from the fluid flow study needs to be mapped onto a 3D domain where particle tracks can be solved in three dimensions. In this section, the outline of the setup to achieve this mapping is provided.

1. Geometry Component 1: Fluid Flow Solution
   A rectangle geometry representing the cross section of the syringe is used with cylindrical coordinates (r, z) due to the symmetric nature of the axis. The 2D axis-symmetric rectangle with a line of symmetry obtains the fluid flow results. The velocity of the fluid is solved only in the (r,z) plane, and the assumption is that the solution is identical at every azimuthal angle. The model solves for (u, w) and pressure (p) where u and w are the components of radial and axial velocity, respectively.

2. Geometry Component 3: 2D to 3D Conversion
   The axis-symmetric results are mapped onto a 3D domain (x,y,z). This is done by creating a separate geometry component (component 3). COMSOL uses general extrusion operators to achieve this. The 2D velocity field is transferred to a 3D system through the following mappings:

   Mapping 1: General Extrusion 1 (genext1)

   - r- expression: $r \Rightarrow$ keeps the radial coordinate the same
   - z- expression: $-z + b \Rightarrow$ Adjusts the origin for the height component, b.

   Mapping 2: General Extrusion 3 (genext3)
   A second general extrusion operator is used to ensure that the velocity field remains aligned. It also allows for the next geometry component (component 2) to reference consistently.

   - r-expression: r
   - z-expression: z

   Mapping 3: General Extrusion 2 (genext2)
   Particle tracking operates in Cartesian coordinates. The 2D results (in cylindrical coordinates) need to be mapped to the Cartesian coordinate system. This is a critical step to ensure that the mapping is done correctly. So, another general extrusion is needed for this mapping:

   - r- expression: $\sqrt{(x^2 + y^2)} \Rightarrow$ converts r into Cartesian (x,y)
   - z- expression: $z \Rightarrow$ Keeps the axial component the same.

3. Geometry Component 2: Defining the Velocity Field in 3D
From genext2, one can extract

   - Radial velocity, $ur = genext2(u)$
   - Azimuthal velocity, $u_\phi = genext2(v)$
   - Axial velocity, $w = genext2(w)$

   To ensure each azimuthal slice has the correct velocity vector, this needs to be converted into Cartesian coordinates (x, y, z):

$$u = ur \cdot cos(\phi) - u_\phi \cdot sin(\phi)$$

$$v = ur \cdot sin(\phi) + u_\phi \cdot cos(\phi)$$

$$w = genext2(w)$$

With this arrangement, the 2D axisymmetric fluid flow study can be extended to obtain 3D particle results. By doing so, the required flow field is obtained with minimal computational resources. However, this mapping assumes that the velocity field is axisymmetric and identical at all azimuthal angles. As a result, any 3D or chaotic flow features are not captured directly. This mapping, therefore, provides a reliable approximation for studying mean particle motion and overall transport behaviour, but not the exact particle trajectories in highly unstable or non-axisymmetric flow regimes.

3.3.4 Boundary and Initial Conditions

Depending on the study, the particles can be released based on various initial and boundary conditions. The release times are set to release at t=0 s if the study focuses on the trajectory of the particle during spin-up and spin-down, or the particle can be released at t=1 s where spin-down has just begun. The particles can be released as

1. Singular; released at a selected height and radius. (i.e., 1 particle at $r = 0.5\,\text{mm}$, $z = 0.5\,\text{mm}$)
2. Distributed along a line; at a fixed height and a selected radial gap (i.e., 5 particles at 0.4 mm apart at a height of 1 cm)
3. Grid; Rectangular grid release (i.e. 5x4 particles with radial and height gaps at predetermined values)

The particles have zero initial velocity and only motion influenced by the local flow field is tracked. That is, they do not have movements outside those induced by fluid motion. At the walls, the particles have a bounce condition. This means that the particles reflect off walls using an elastic collision model, where the kinetic energy of the particle is conserved. This idealisation maintains energy conservation and avoids artificial damping of motion near the boundaries. The particles are defined as spherical in which their size and density can be prescribed (i.e., density: 1200 kg/m^3, diameter: 50 µm). They are also solid particles (no

deformation effects are considered) and no temperature effects are included. Since the fluid domain is assumed to be Newtonian and incompressible, the particle motion is computed according to Newton's second law of motion. The "Newtonian" assumption here applies to the fluid, not the particles, which are modelled as rigid solids responding to hydrodynamic forces.

3.3.5 Solver settings

As before, the solvers options available can use implicit or explicit methods. Implicit methods solve the velocity and position of the particles simultaneously by solving a system of nonlinear equations at every time step. Explicit methods advance these quantities sequentially. They are also conditionally stable and limited by the Courant–Friedrichs–Lewy (CFL) condition, whereas implicit methods are unconditionally stable for linear problems and allow larger time steps (Ferziger et al., 2020). This makes them suitable for stiff systems such as coupled particle-fluid problems such as a PFS inspection model.

For initialisation (calculation of the first step), a first-order implicit method called backward Euler is used. It is very stable, but introduces significant numerical dissipation, meaning that it artificially damps solution variations over time and can smooth sharp gradients more than higher-order methods such as the generalised-α model. Therefore, it is only used to initialise and obtain a solution from unknown initial conditions without losing stability. After this, the solver is switched to generalised-alpha, which is an implicit second-order accurate time integration scheme. It is used for transient studies where high-frequency oscillations, i.e. rapid fluctuations in the numerical solution arising from discretisation errors or sharp temporal changes, need to be damped to ensure a stable and physically meaningful solution. It reduces numerical oscillations in the particle trajectories and is a good balance between stability and accuracy (Chung and Hulbert, 1996). This adjustable numerical dissipation makes it ideal for fluid-particle interactions.

For the solution of non-linear systems (the particle motion equations), a Newton iteration method is selected. This method is an iterative method that refines the solution until the convergence criteria have been achieved. with this method, equations are linearised and solved repeatedly until convergence is achieved. Convergence is determined based on a relative tolerance, where the solution is considered converged when the change between successive iterations falls below a specified threshold in COMSOL Multiphysics. COMSOL AB (2021) recommends it for particle-fluid interaction models. As this method leads to large linear systems at each iteration, a linear solver is needed. COMSOL automatically assigns an appropriate sparse linear solver based on the problem size and physics coupling, in this case the default Generalised Minimal Residual (GMRES) solver is used. At each Newton iteration, the resulting linear system is solved using the GMRES method. GMRES is an iterative solver designed for large, sparse systems, and works by minimising the residual of the solution.

3.4 Hardware Specifications

All studies are conducted on COMSOL Multiphysics 6.1 without parallel processing. The studies are conducted on a custom built PC with a 16-core AMD Ryzen 9 5950X processor. For the studies, parallel processing was not needed as the simulations did not take a lot of time. Although, for large grid studies, it could have been used to save some time, but this feature was not used. The Random Access Memory (RAM) for the PC is 32GB, and the graphics card is a RTX 3080Ti.

For post-processing, Spyder (Python 3.11), MatLAB R2023b, and Microsoft Excel are used alongside the built-in post processing tools in COMSOL Multiphysics.

3.5 Mesh Independence Study

To conduct a formal study of mesh independence, three different mesh sizes have been created. The number of mesh elements for the coarse mesh has been doubled in the axial and radial directions to produce the medium mesh. The same is then replicated for the medium mesh to create a fine mesh. The fine mesh has four times the number of mesh elements along each direction in comparison to the coarse mesh. By increasing the number of mesh elements in this way, the uneven mesh distribution as previously discussed is kept constant for each mesh while refinement is made. A quadrant of the top end of the cylinder is shown in Figure 25 for the coarse, medium, and fine mesh, respectively, for a visual representation.

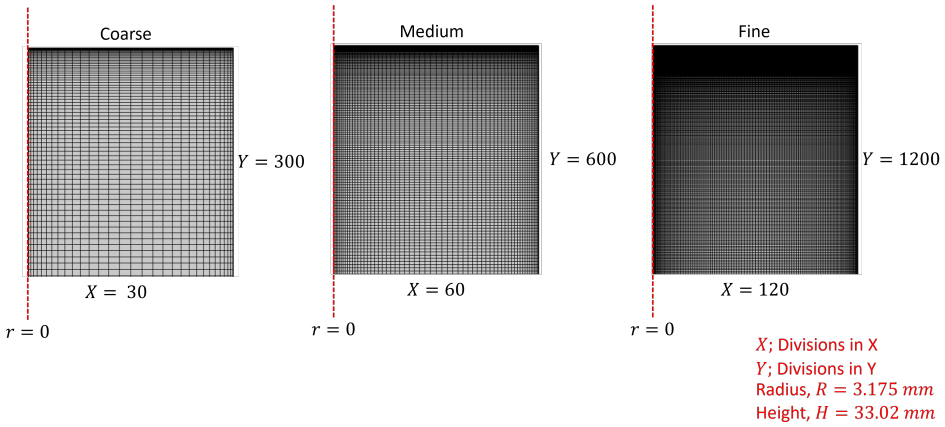


Figure 25: Coarse, medium, and fine mesh distributions used for the mesh independence study

Of the three types of mesh above, a mesh needs to be selected that produces results with relatively good accuracy while reducing computational costs (memory and time). The setup shown in the previous section is used to obtain solutions using both first-order and second-order discretisation schemes. These correspond to different levels of numerical accuracy, with second-order schemes providing improved spatial resolution. Comparing both allows assessment of numerical convergence and ensures that the results are not sensitive to discretisation order. To get a general (quantitative) idea of the performance of each mesh, the z-component

of the velocity can be plotted across the axial centreline of the cylinder. The first-order results for the velocity field using the coarse, medium, and fine mesh are shown in Figure 26 at a time of 0.1 s (spin-up). Overall, the velocity profile across the centreline for the three mesh sizes shows small differences. Figure 26b shows a zoomed plot of the highest velocity region for each mesh. One can determine that the three meshes produce the highest velocity in a small range ($-0.0158 < z < -0.0156$). The same qualitative analysis can be performed on the second-order results. Figures 27 a and 27 b show the same results using second-order discretisation. The main difference seen using the second order is that the three mesh sizes produce a velocity profile much closer to each other and the highest velocity recorded has a much smaller range than the first order (approximately $-0.0157 < z < -0.0156$). However, a better method than this qualitative method to determine the performance of each mesh is to use a quantitative approach such as the Grid Convergence Index (GCI) at the most extremely sensitive areas of the domain in terms of fluid flow.

3.5.1 Grid Convergence Index (GCI)

The GCI is a uniform method proposed by Roache (1994) to determine the discretisation error between various grid refinements. A percentage error is calculated between successive grid refinements to provide confidence within the solution obtained. This method is used to determine whether the mesh grids selected for this model are reliable. To calculate the GCI, the solution error (e) must be calculated. This is given by;

$$e = \frac{(f_2 - f_1)}{f_1} \tag{28}$$

where f_1 and f_2 are the solutions obtained from finer and coarser grids, respectively. The location of the data points is selected on the basis of the regions where the z-component of the velocity is maximum. These are regions close to the bottom (or top) of the cylinder near the centreline (r=0). The precise location of the data points where f_1 and f_2 are recorded is shown in Figure 28.

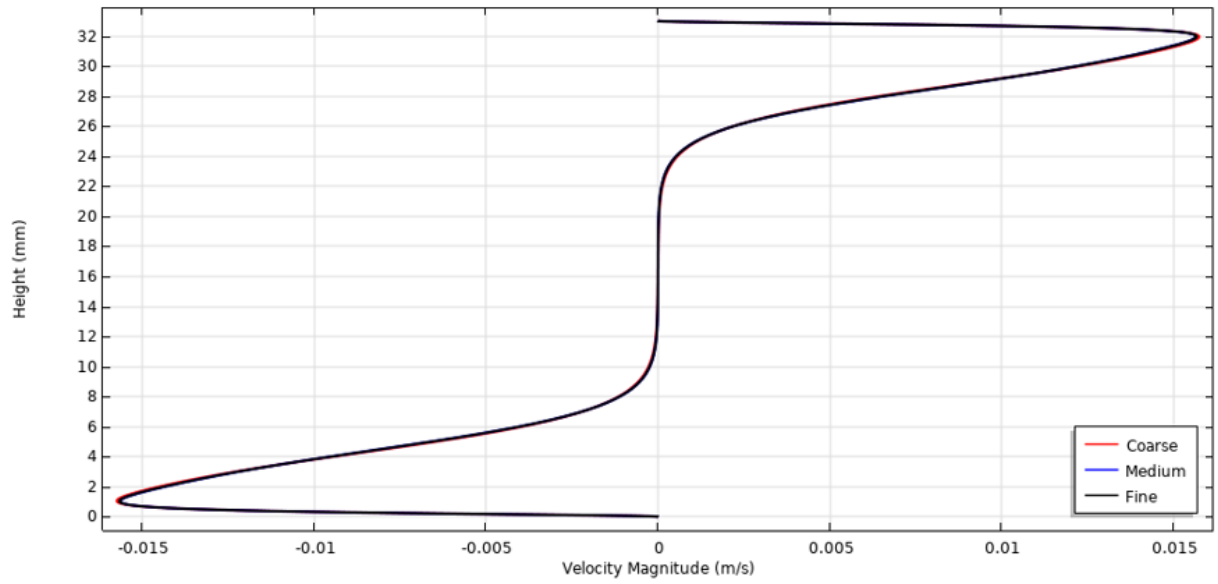
The grid refinement ratio, r, is given by

$$r = \frac{h_2}{h_1} \tag{29}$$

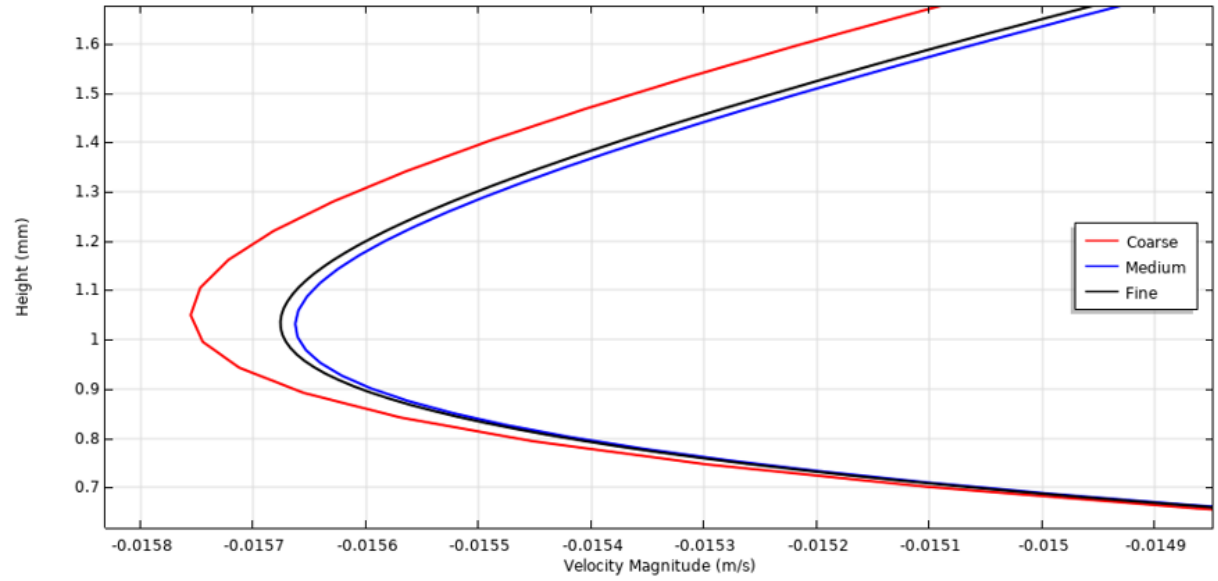
where h_1 and h_2 are the edge lengths for the fine and coarse grid, respectively. For the grids used for this model, the grid refinement ratio between the coarse and medium meshes as well as the medium and fine mesh is set to 2. The discretisation error can now be calculated using the Richardson Extrapolation;

$$E = \frac{e}{(r^p - 1)} \tag{30}$$

where p is the formal order of accuracy. For the analysis, first-order and second-order accuracy is tested. Roache (1994) extended this error calculation to include a safety factor (F_s) that produces the GCI equation as shown;

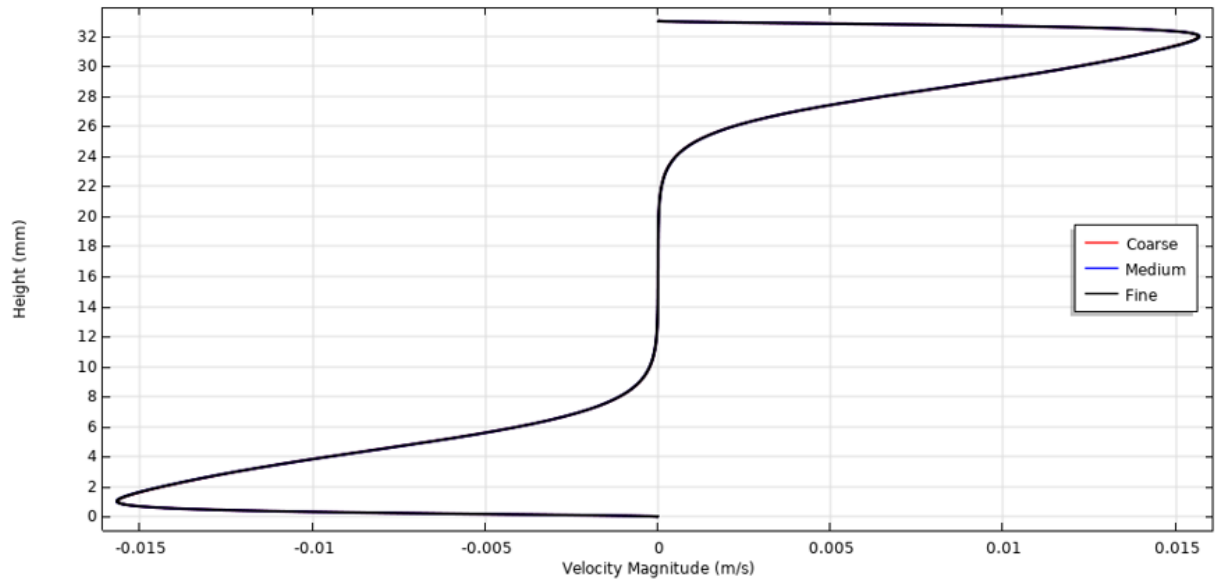


(a)

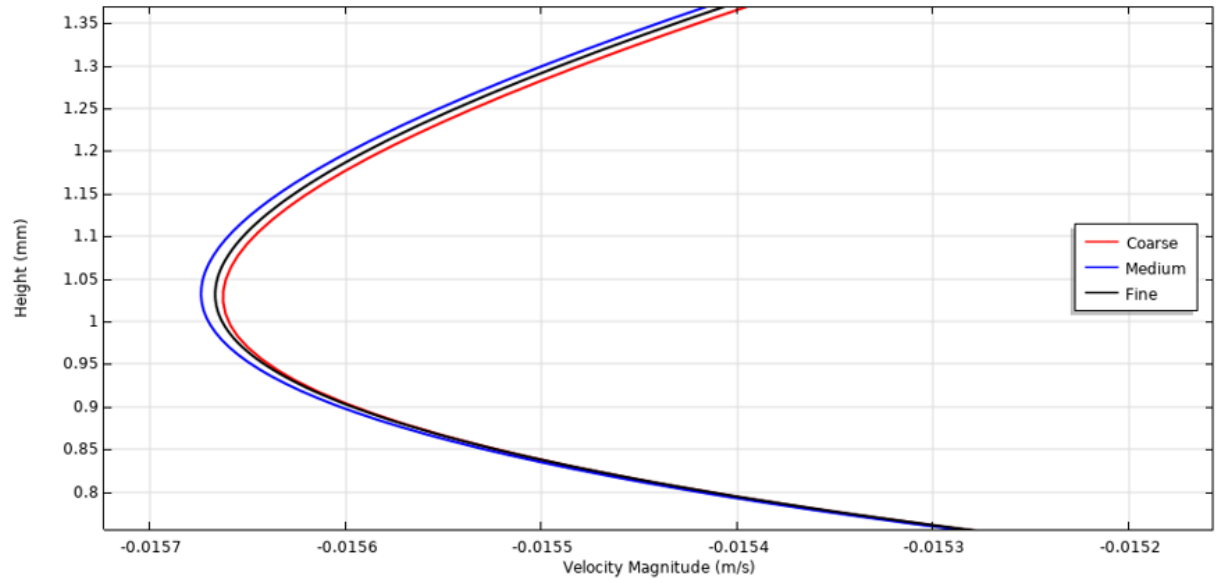


(b)

Figure 26: (a) Full domain comparison, (b) zoomed view of the high-velocity region near the cylinder head, highlighting mesh-induced variations for the first order, z-component of velocity across the centreline at time t=0.1s for coarse, medium, and fine meshes.



(a)



(b)

Figure 27: (a) Full domain comparison, (b) zoomed view of the high-velocity region near the cylinder head, highlighting mesh-induced variations for the second order, z-component of velocity across the centreline at time t=0.1s for coarse, medium, and fine meshes.

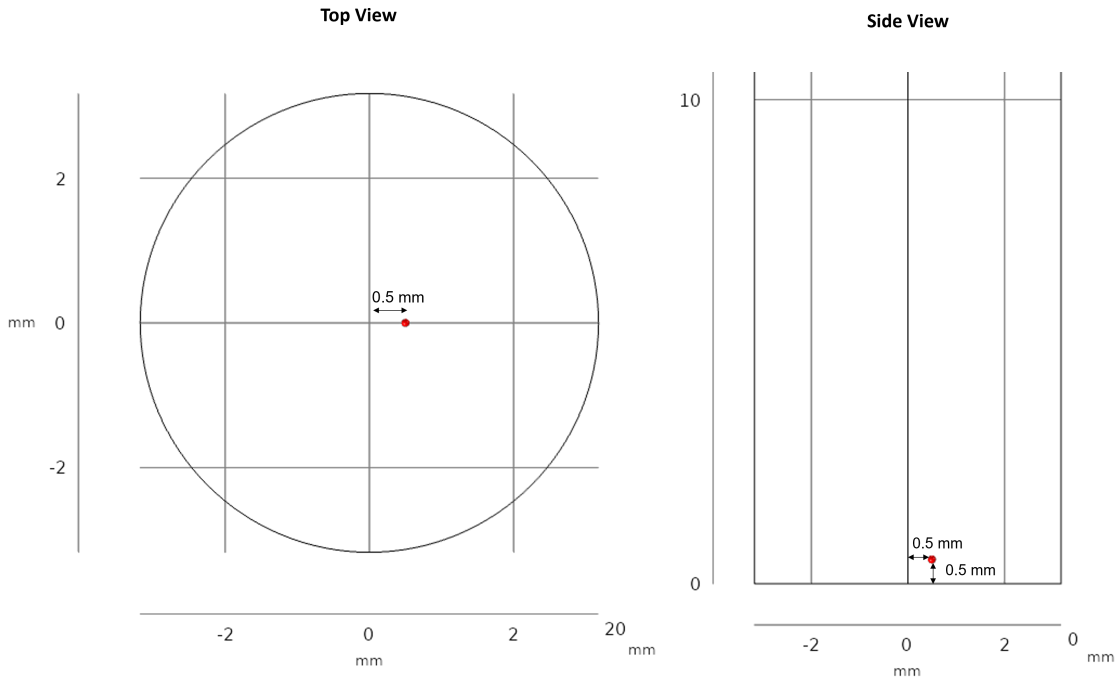


Figure 28: Location of the selected data points for GCI calculation

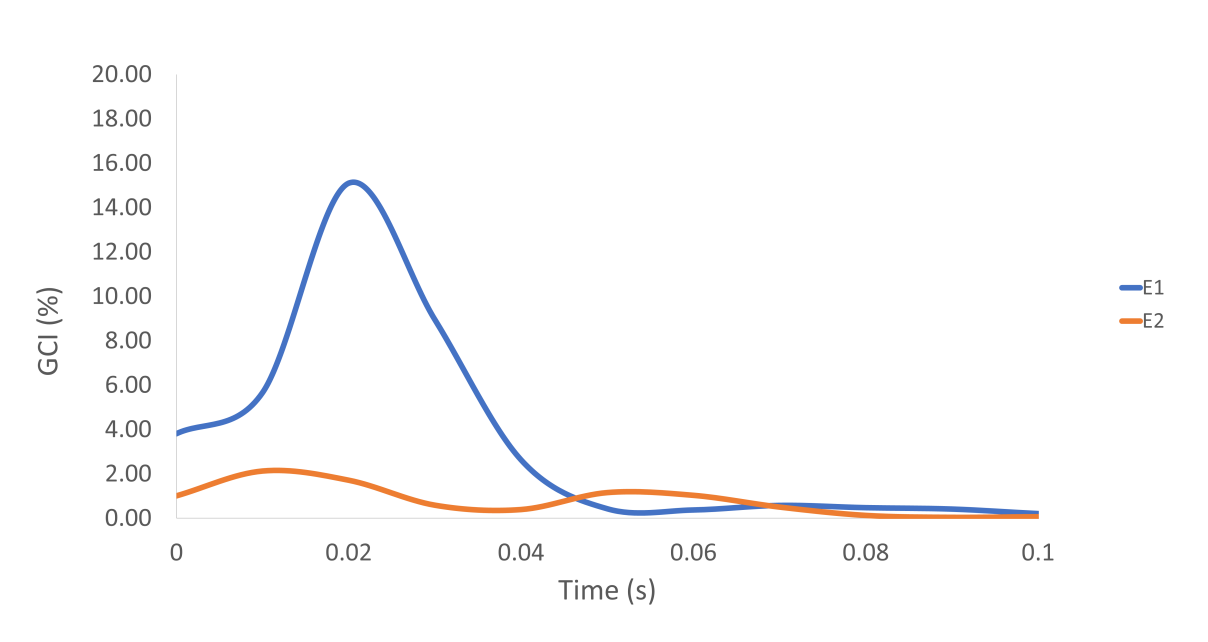
$$GCI = \frac{F_s|e|}{(r^p - 1)} \tag{31}$$

Common values of the safety factor are within the range of $1 < F_s < 3$, where 3 is considered conservative and 1 is equivalent to the Richardson extrapolation.

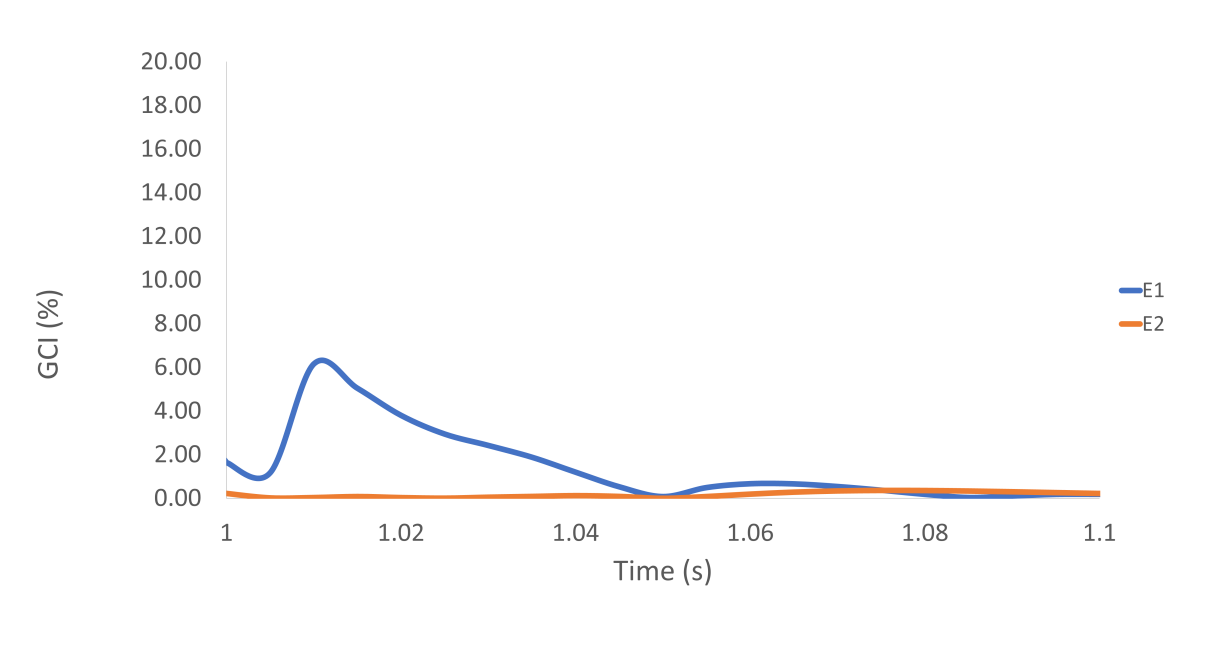
GCI can be calculated for both spin-up and spin-down. At the end of spin-up, the cylinder stops rotating, initiating the spin-down process. This sudden transition causes strong transient flow adjustments near t=1, which can introduce numerical noise in the GCI calculations. To minimise this, the error is evaluated in regions of peak velocity, where discretisation errors are most representative for each mesh. Therefore, the error is calculated for the time frame within 5% of the maximum velocity. For spin-up, these velocities occur between $(0 < t < 0.2)$ seconds, where the same time frame for spin-down is $(1 < t < 1.3)$. The GCI (using a conservative value of $F_S = 3$) for the first-order solution is shown in Figure 29, where E1 is the GCI between coarse-medium meshes and E2 is the GCI between medium-fine meshed. The results show a maximum error of approximately 15 % for E1 (between the coarse and medium meshes) during spin-up. This occurs as soon as the cylinder walls begin to move. For E2 (between medium and fine mesh), this error is significantly smaller than 2 %. For E1 after the initial spike, the average error in the time frame is below 3 %. E2 remains below 2 % throughout this period of time. During spin-down, the highest recorded error occurs at approximately 6 % for E1 where E2 produces errors of less than 1%. The average error for both E1 and E2 is under 2 % during spin-down. This shows that even with a conservative safety factor of 3, the results produced show that the medium mesh is appropriate with regard

to the error produced. Doubling the mesh elements from medium to fine does not result in a significant error reduction and therefore the medium mesh can be used for further analysis.

Conducting a similar second-order analysis produces Figure 30. The results show all errors below 2% for spin-up and spin-down with first-order accuracy. The second-order results show below 0.5 % error during spin-up and a maximum error of 3 % during spin-down.

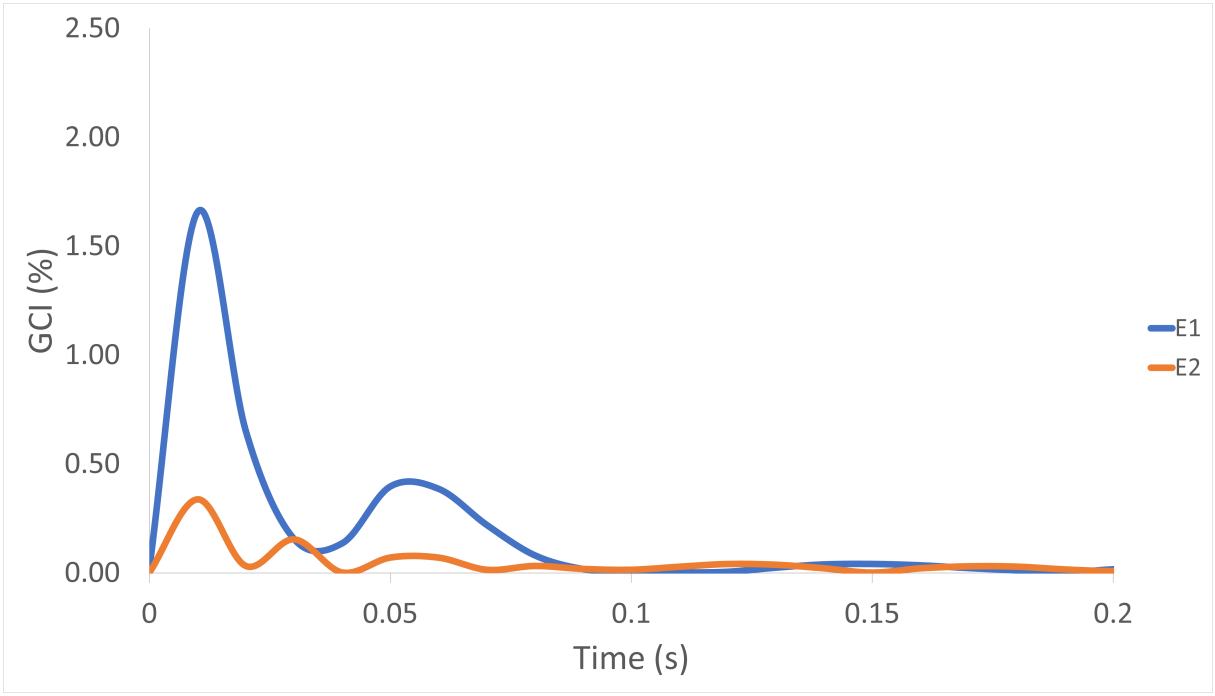


(a)

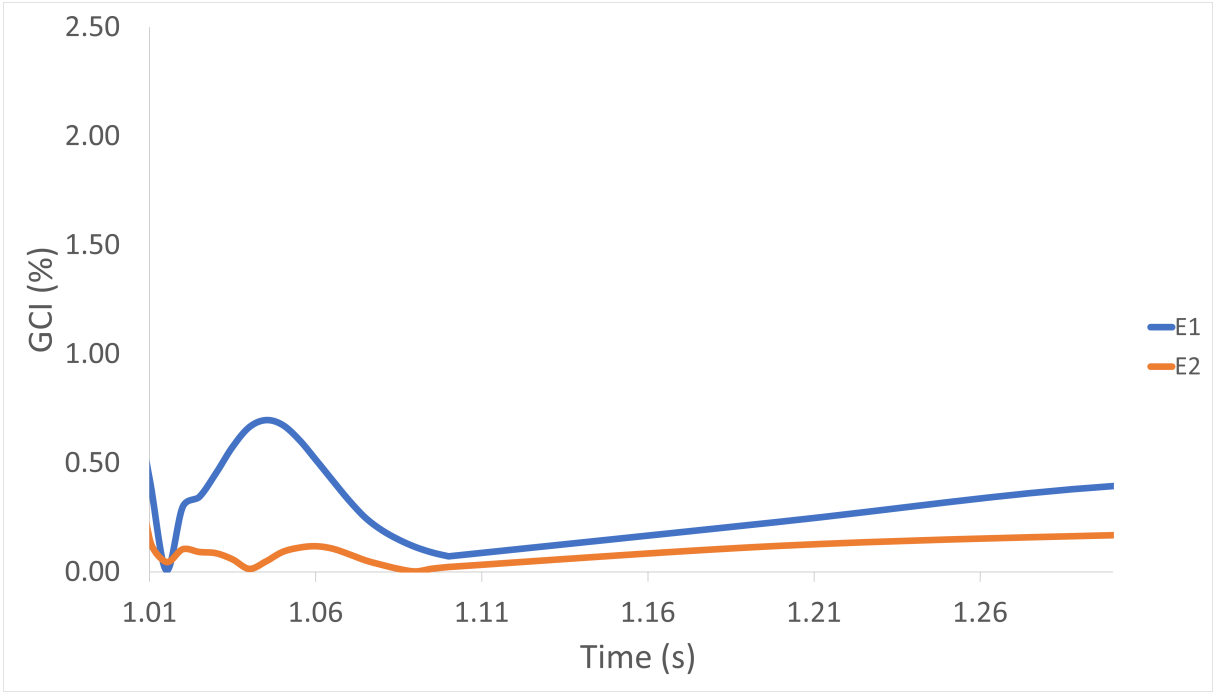


(b)

Figure 29: (a) spin-up phase, (b) spin-down phase, showing the GCI variation with time for the first-order solution. E1 represents the GCI between coarse–medium meshes, and E2 represents the GCI between medium–fine meshes



(a)



(b)

Figure 30: (a) spin-up phase, (b) spin-down phase, showing the GCI variation with time for the second-order solution. E1 represents the GCI between coarse–medium meshes, and E2 represents the GCI between medium–fine meshes

Consider the runtime for the coarse, medium, and fine mesh for both first and second order discretisation, as shown in Table 3.

Table 3: Computational Time Analysis for Different Mesh Resolutions

Discretisation	**Coarse Mesh**	**Medium Mesh**	**Fine Mesh**
First-Order	2 min 7 sec	7 min 13 sec	35 min 17 sec
Second-Order	7 min 58 sec	38 min 5 sec	119 min 23 sec

The time increase from first-order to second-order was between 3x and 6x. A similar increase is seen between the jump from coarse to medium and medium to fine mesh. The medium mesh takes $\approx$ 7 minutes and for second-order, it takes $\approx$ 38 min. Since no multi-threading was enabled, the solver ran on a single CPU core. This is why the fine mesh significantly increased the computation time. The increase in time from first to second order is also due to higher degrees of freedom on second-order elements. For the fine mesh, the second-order result took $\approx$ 119 minutes compared to 35 minutes for the first-order elements. This is a significant increase in time and needs to be considered for large simulations. These results indicate that while finer meshes and higher-order schemes provide marginal accuracy, they lead to disproportionately higher computational costs. The medium mesh with second-order discretisation therefore provides an optimal balance between accuracy and efficiency for the present model. With this configuration, the discretisation error remains below 5% even with a conservative safety factor, while maintaining a reasonable runtime for subsequent parametric studies.

4 Fluid Flow: Data Collection and Simulation

The aim of this chapter is to analyse the fluid flow patterns that evolve during automated robotic inspection. To do this, the inspection cycle of a prefilled syringe will be split into various phases to determine the behaviour of fluid in these phases. The results presented in this chapter are obtained using the CFD model and boundary conditions defined in Chapter 3 (Materials and Methods: Computational Methods). This is so that the flow field can be explained in general.

For clarity, these specific definitions are used in this chapter:

- Primary flow: The dominant flow behaviour of the bulk fluid during a specific inspection phase.
- Secondary flow: The less dominant flow structures that occur within the same phase.
- Spin-up: The phase in which the syringe is spun at a fixed rate of rotation, allowing the fluid to start its initial rotation. Before spin-up, the fluid was assumed to be stationary for this study. This is represented on the angular velocity against time plot in Figure 22 for the time between 0 and 1 seconds.
- Spin-down: The phase in which the rotation of the syringe has stopped and the fluid reacts to this halt in rotation. This is also the most important phase of inspection as it leads to the fluidisation of particles needed to be detected. In Figure 22, this is between 1 and 5 seconds, where the wall rotational speed is zero.

During spin-up or spin-down, two types of flow are observed. These types of flow are defined as primary or secondary flows. The primary flows are the most dominant features observed within the flow, where as secondary flow features are those that are smaller (less dominant) such as recirculation, etc. In this chapter, the primary and secondary flows for both spin-up and spin-down are discussed.

4.1 Spin-up

4.1.1 Primary Flow

The primary motion of fluid in a spinning cylinder during the spin-up phase can be understood from the magnitude of the fluid velocity within the domain. To get an idea of how the fluid behaves, consider the velocity magnitude, u, across the domain, as well as the normalised velocity magnitude, $\bar{u}$ (defined in the previous chapter), at the midpoint of the cylinder along the radial direction (shown by the red line in Figure 31).

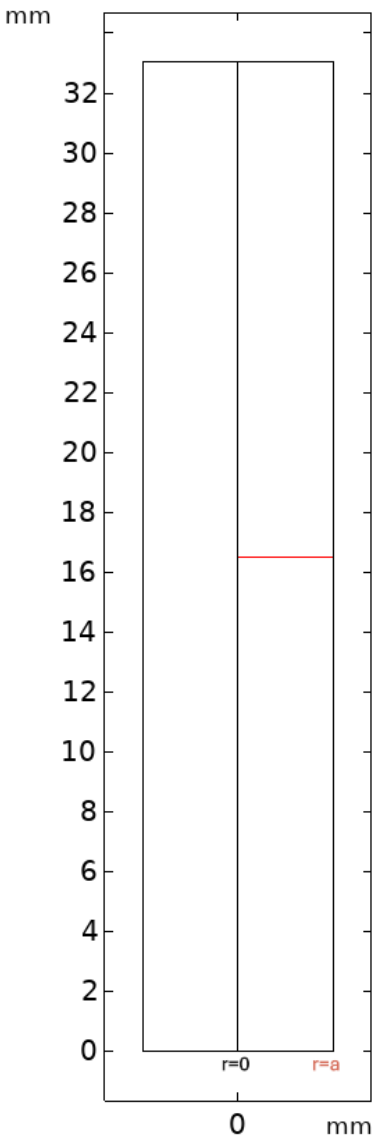


Figure 31: Red line used to plot radial data at the centre of the cylinder

For this analysis, consider the Reynolds number (Re) of 500 ($\approx$ 1372 RPM). Figure 32 shows the velocity magnitude contours throughout the domain and Figure 33 plots the non-dimensionalised velocity magnitude, $\bar{v}$, along the red line in Figure 31. Each curve in Figure 33 corresponds to $\bar{v}$ at a fixed time during the spin-up phase. The times (with respect to ω) of 0.01 s, 0.05 s, 0.1 s, 0.5 s, 0.90 s, and 0.98 s (intermediate times) are shown. These capture the evolution of the velocity profile at the centre of a cylinder undergoing spin-up.

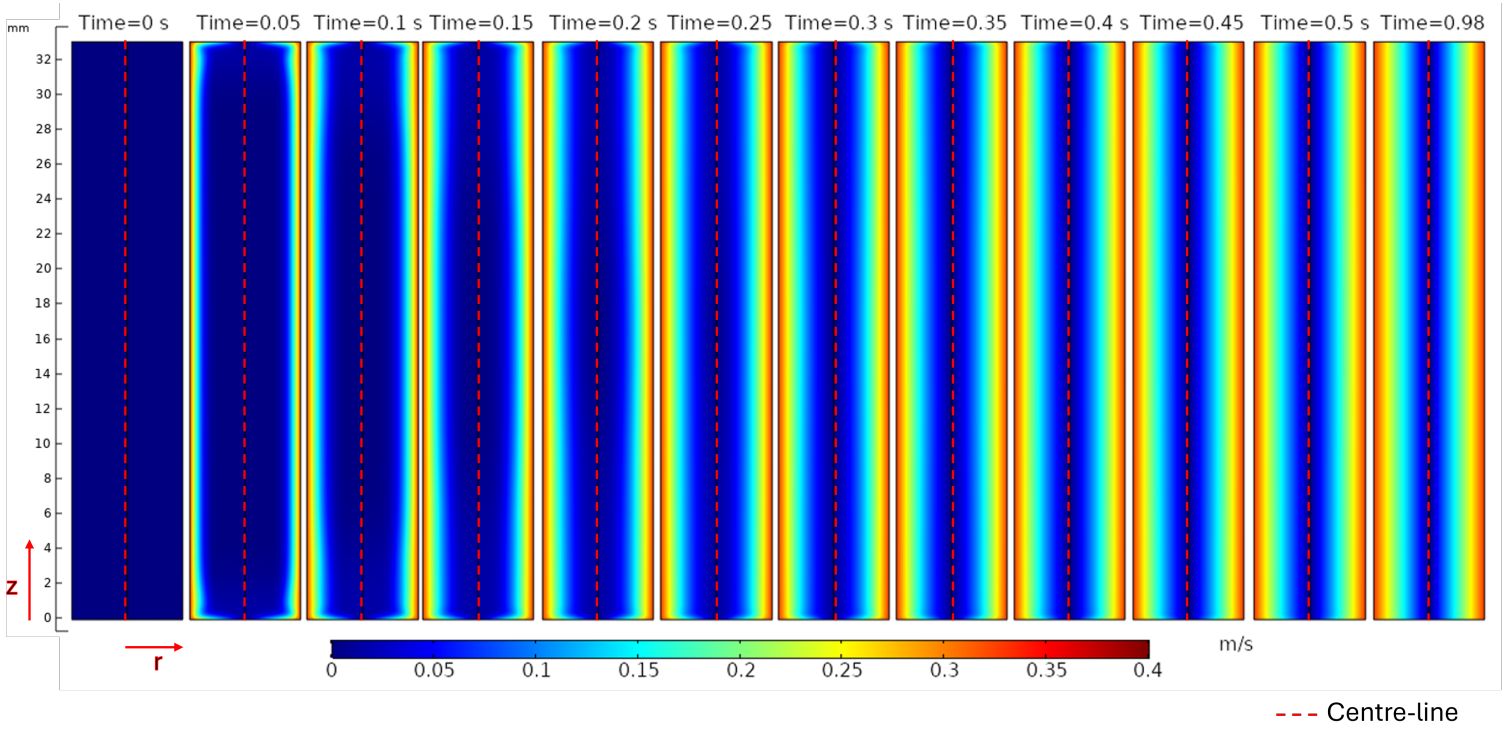


Figure 32: Contours of velocity magnitude during spin-up represented on a 2D cut of the cylinder where the axial direction is represented vertically and the radial direction is represented horizontally at different times from t=0s (start) to t=0.98s (end).

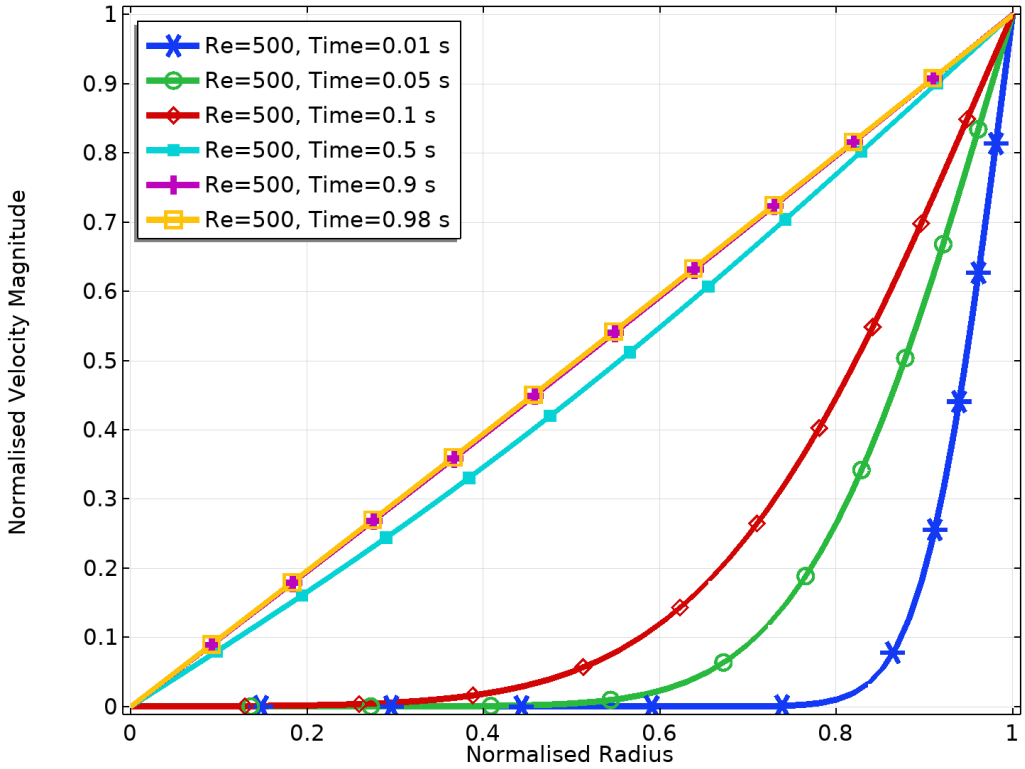


Figure 33: Normalised Velocity Magnitude against Normalised Radius calculated using Equation 11 for Re = 500 at times of 0.01s, 0.05s, 0.1s, 0.1s, 0.5s and 0.98s

On Figure 32 at t=0 s, the cylinder is stationary, and the dark blue velocity contour represents zero velocity (non-spinning fluid) across the domain at this time. On Figure 33, the $\bar{v}$ at t=0.01 s is shown by the blue curve. At this point, the cylinder sidewalls have started to move. The fluid in the entire domain is essentially zero, except for the cylinder sidewalls, where the highest velocity is observed. The fluid in immediate contact with the sidewalls of the cylinder starts to move with the moving sidewall. This is a consequence of the no-slip condition. Applied to the boundary, the no-slip condition is an experimentally observed boundary condition that states that the fluid velocity at a moving wall must be equal to the velocity of the wall itself. In this case, this is set to $v = \omega r$. Initially, most of the fluid away from the sidewalls remains stationary as the viscous effects have not yet influenced the fluid near the central axis of the cylinder.

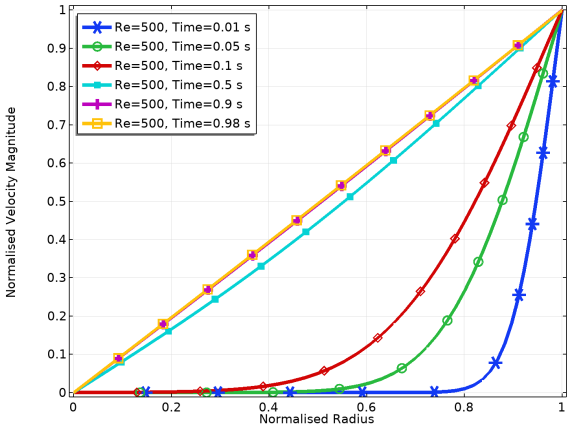
As the cylinder sidewalls continue to rotate after spin-up begins, a boundary layer (B.L) develops. Classically in fluid dynamics, for a stationary wall (or object) with fluid moving over the wall (i.e. a flow over a flat plate or an aerofoil), the B.L is a thin layer of fluid where the velocity of the flow ranges from zero (at the walls) to the free-stream velocity (velocity of the fluid far away from the walls) over a short distance. The thickness of the boundary layer is defined as the distance from the wall and the point at which the velocity of the fluid is 99% that of the velocity of the free stream (Chanson, 2004). A B.L can form when there is a relative velocity between a wall and a fluid. In this study, it is the sidewalls of the cylinder that move. Therefore, the velocity of the fluid at the cylinder sidewalls will match the sidewall velocity, and the fluid near the centreline of the cylinder will be stationary. The boundary layer is a region in which viscous forces dominate the movement of the fluid.

With the progression of time, the adjacent fluid layers also gradually accelerate, due to momentum diffusion, radially inward towards the central axis of the cylinder. Hence, the non-spinning region starts to shrink, and the boundary layer region moving with the sidewalls grows as time increases. This confirms the observation made by Wedemeyer (1964) on how momentum propagates from the boundary towards the inner core. It is also confirmed by the curves $\bar{v}$ showing $\bar{v} = 1$ on the side walls of the cylinder. At t=0.5 s, the green curve shows that nearly 30% of the outer radial positions (0.7 to 1) have a $\bar{v}$ of at least 0.1 (10% the velocity of the outer walls) for Re = 500 ($\approx$ 1372 RPM). More of the fluid near the cylinder sidewall is now moving at increased velocity, and this region is starting to expand closer towards the central axis of the cylinder. However, the velocity of the fluid near the central axis at this time is still near zero, indicating that the momentum transfer from the sidewall of the cylinder has not yet reached the inner cylinder.

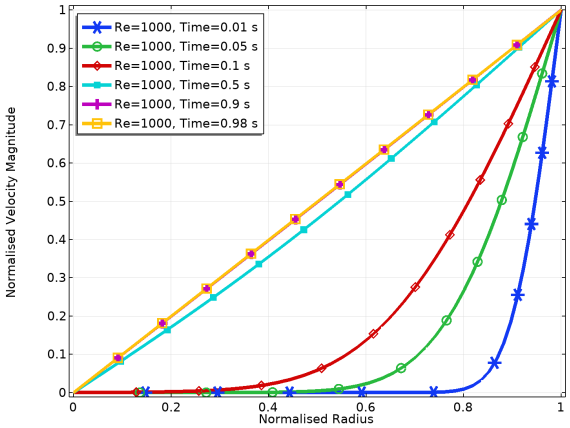
The gradient of the velocity profile is very steep in the earlier stages of spin-up. This steepness is reduced as more momentum is transferred from the cylinder sidewalls to the inner central axis. The diffusion of momentum is driven by the viscosity of the fluid. The fluid viscosity is a measure of the internal resistance to flow which acts as the mechanism that enables the transfer of momentum from the cylinder sidewalls to the stationary fluid near the central axis of the cylinder, resulting in an increase in rotational velocity of the entire fluid within the domain. The velocity profiles observed are in line with Hyun et al. (1983) which has also been validated experimentally.

Some other factors that influence momentum transfer are the speed, radius, and density of the fluid (also encompassed by the Reynolds number). At lower Reynolds numbers, the viscous forces dominate the inertial forces, causing a more gradual and diffusive momentum transfer. The boundary between the spinning and non-spinning fluid can also be referred to as the "moving front". At t=0.1 s, almost 60% of the outer radial position has at least 10% of the cylinder sidewall velocity showing the progression of the momentum transfer. Note that the contour of velocity magnitude at a time of 0.5 s resembles the time of 0.98 s (just before the end of the spin-up phase). However, for $\bar{v}$ at t=0.5 s and 0.98 s, there is still some momentum transfer that occurs due to the difference in the cyan and magenta curves. The percentage difference of the normalised velocity magnitude between the two curves is 4.7%. This is much smaller between 0.9 and 0.98 seconds at 0.3%. This suggests that the fluid exhibits near-steady state behaviour. For these two times, only the inner 10% of the radius of the cylinder has speeds less than 10% of the velocity of the cylinder sidewall.

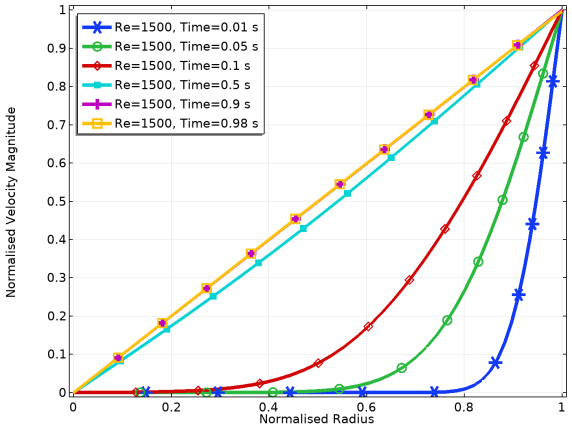
The general trend for the velocity magnitude across different Reynolds numbers is consistent and this can be seen on Figure 34. The sub-figures plot various Reynolds numbers ranging from 500 to 3000 in increments of 500. Each curve within the subplot corresponds to the non-dimensionalised velocity magnitude, $\bar{v}$, at a various times during the spin-up phase. In all cases of the Reynolds numbers studied, the magenta and yellow curves (corresponding to t=0.9 and 0.98s) show very little change, suggesting that solid body rotation has been reached by 0.98 seconds for all these Reynolds numbers. Moreover, one can also look at $\bar{v}$ for the same time with different Reynolds numbers, as shown in Figure 35. As expected, the subplots show that the faster Reynolds numbers reach the steady-state solution earlier than the lower Reynolds numbers. This is shown by the higher Reynolds number curves being above the lower Reynolds number curves. The plots for t=0.9 s and t=0.98 s for all Reynolds numbers line up on top of each other, confirming a steady state solution. Moreover, this fits with the work of Weidman (1976*a*) at low Ekman numbers, where diffusion depends on the viscosity and the rotational speed (essentially the Reynolds number). At t=0.98 s, all the Reynolds number cases collapse onto a single linear profile. This shows that the flow has reached near solid-body rotation. This linear relationship between normalised velocity and radius shows that the entire fluid column is now rotating uniformly. As the velocity profile has been normalised, the collapse also demonstrates that the velocity distribution becomes independent of Reynolds number once the spin-up is completed. This confirms a quasi-steady state behaviour at the end of this phase.



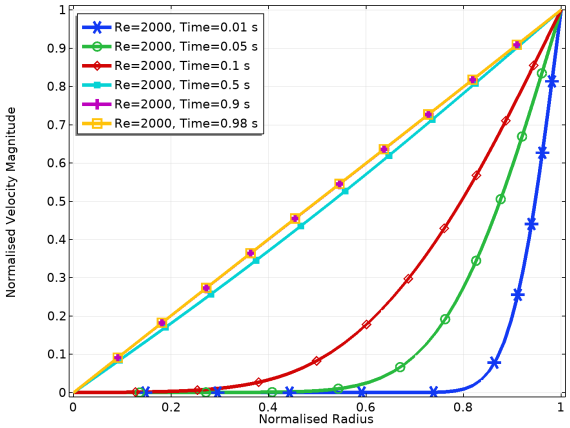
(a) Re 500



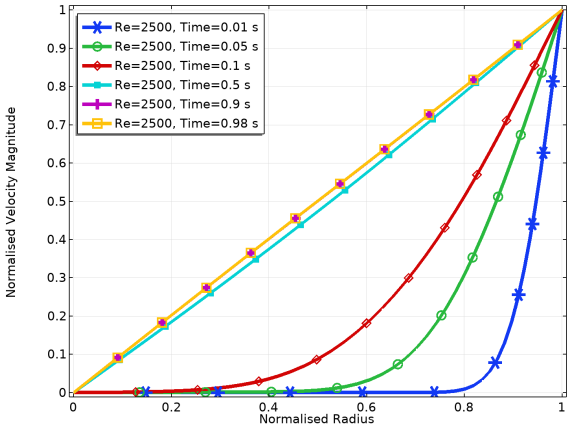
(b) Re 1000



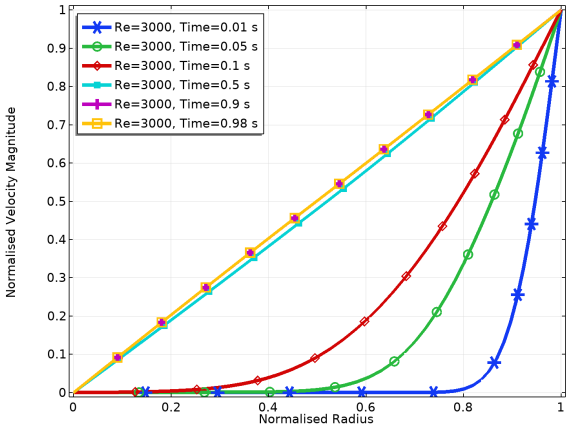
(c) Re 1500



(d) Re 2000

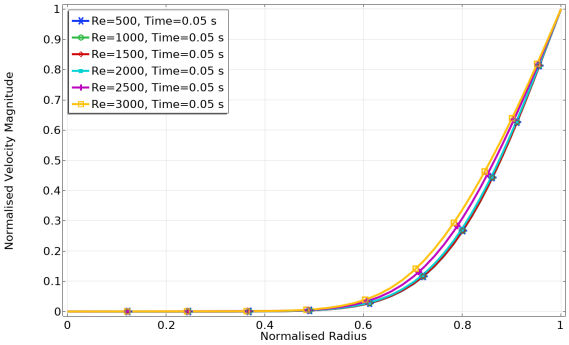


(e) Re 2500

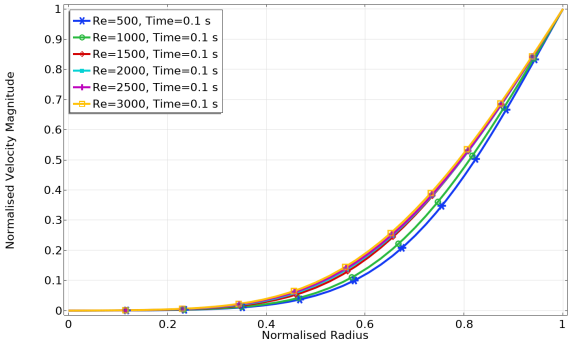


(f) Re 3000

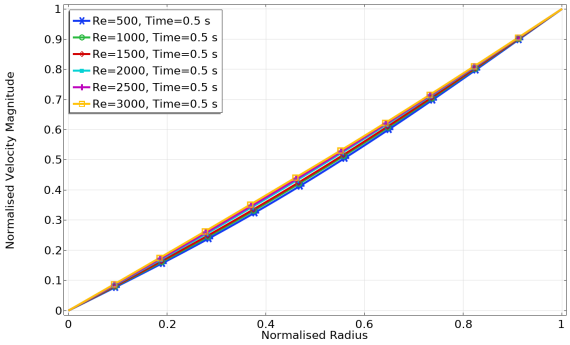
Figure 34: Spin Up: Normalised Velocity Magnitude against Normalised Radius calculated using Equation 11 for various Reynolds Numbers of: (a) Re 500, (b) Re 1000, (c) Re 1500, (d) Re 2000, (e) Re 2500, (f) Re 3000



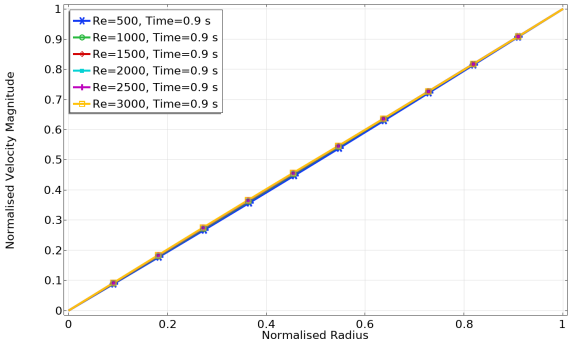
(a) t=0.05s



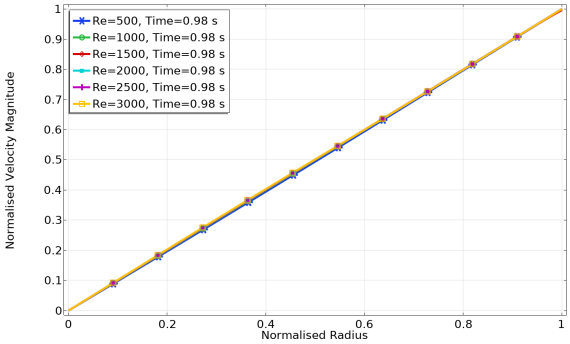
(b) t=0.1s



(c) t=0.5s



(d) t=0.9s



(e) t=0.98s

Figure 35: Spin Up: Normalised Velocity Magnitude against Normalised Radius calculated using Equation 11 for Various Reynolds Numbers at set times of: (a) t=0.05s, (b) t=0.1s, (c) t=0.5s, (d) t=0.9s, (e) t=0.98s

4.1.2 Secondary Flow

In addition to the primary flow, a secondary flow is present, located at the top and bottom walls of the cylinder. The secondary flow exhibits fluid velocities that are an order of magnitude smaller than those of the primary flow. Figure 37 shows the axial velocity for the duration of the spin-up, where red represents the positive velocity (upwards) and blue represents the negative velocity (downwards). As the cylinder spins about its central axis, the flow near the axis of symmetry($\bar{r} = 0$), moves axially towards the top and bottom walls of the cylinder. At the same time, the flow near the sidewall($\bar{r} = 1$) moves axially in the opposite direction toward the centre of the cylinder. As time increases, the area of fluid affected also increases in these respective directions to 0.25 s. To understand the secondary flows, one needs to consider both the axial and radial components of velocity simultaneously. Figure 38 shows the radial component of the velocity during the first instance of spin-up. As t=0.05 s, a differential radial velocity is observed on the top and bottom walls of the cylinder. This causes the fluid at the top and bottom of the cylinder to move from the central axis to the sidewalls of the cylinder. This observation is consistent with the work of Vanyo (2015) highlighting the presence of a secondary flow where the fluid moves from the centre radially towards the outside walls. Further increases in time reduce the radial velocity exhibited near the top and bottom of the cylinder. The axial and radial velocity across the domain is reduced by t = 0.6 s. Compared with the velocity magnitude after t = 0.5 s in Figure 32, the flow again suggests that the steady-state solution has been reached. This can also be confirmed by Figure 36 showing the axial velocity across the centreline is zero at t= 0.9 s. The velocities of the secondary flow are much weaker compared to the dominating primary flow, and this has been highlighted by Wedemeyer (1964). However, the secondary flows are essential for effective dissipation of momentum, specifically in the radial and axial directions. As the secondary flow acts over regions of high velocity gradients where viscous stresses are dominant, these weak circulations govern the overall approach to equilibrium, facilitating the radial diffusion of angular momentum.

A summary of the secondary flow exhibited during the early stages of spin-up near the top wall of the cylinder is shown in Figure 39a where the velocity magnitude of spin-up at 0.05 s is presented. The black arrows represent the axial and radial velocity vectors, and the dashed black line represents the dividing line between the upward and downward motion of the fluid. The numbered white arrows indicate the overall motion of the fluid during these early instances of spin-up. Arrow 1 shows the fluid moving axially to the top of the cylinder, where it is radially pushed outward towards the outside walls (represented by arrow 2). This is where the fluid is pushed down axially toward the middle of the cylinder (arrow 3). Figure 39b shows the direction of the same arrows for secondary flow observed near the bottom wall ($z = -\frac{1}{2}h$) of the cylinder. The direction of the arrows between the top and bottom of the wall is essentially a rotation of 180°.

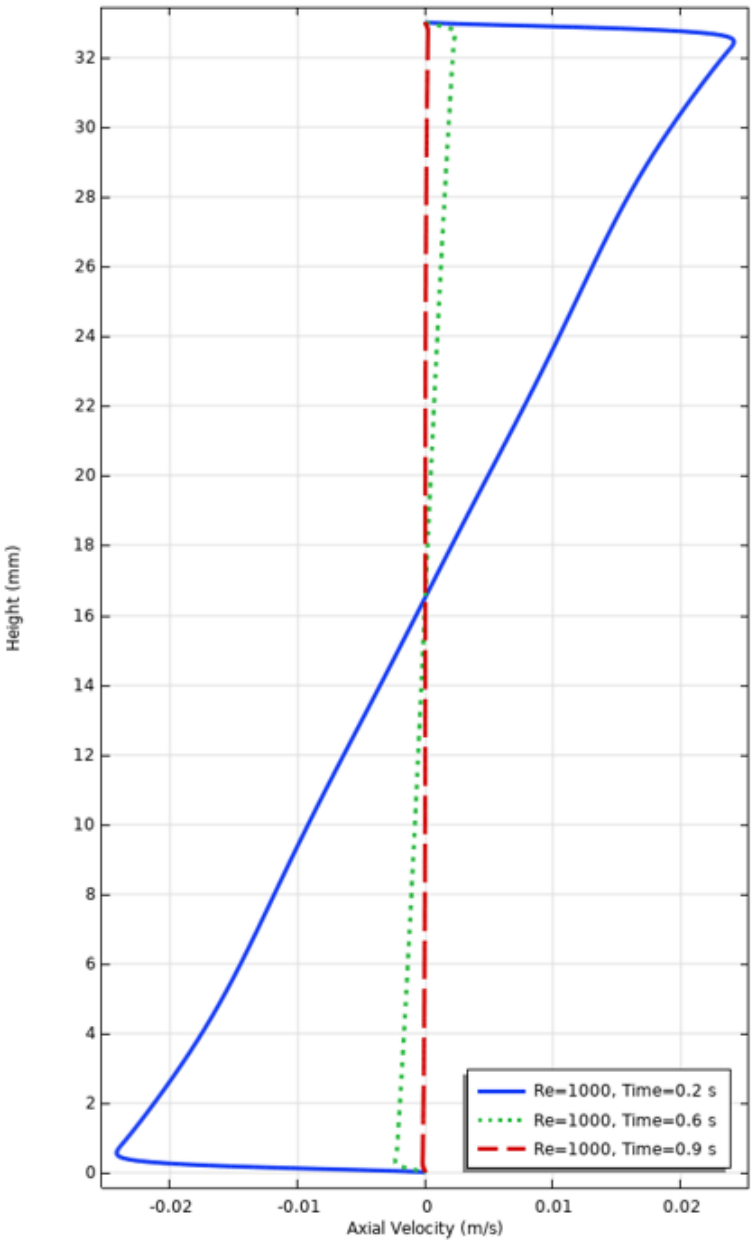


Figure 36: Axial velocity across the centreline for Re=1000 showing steady state solution has been reached

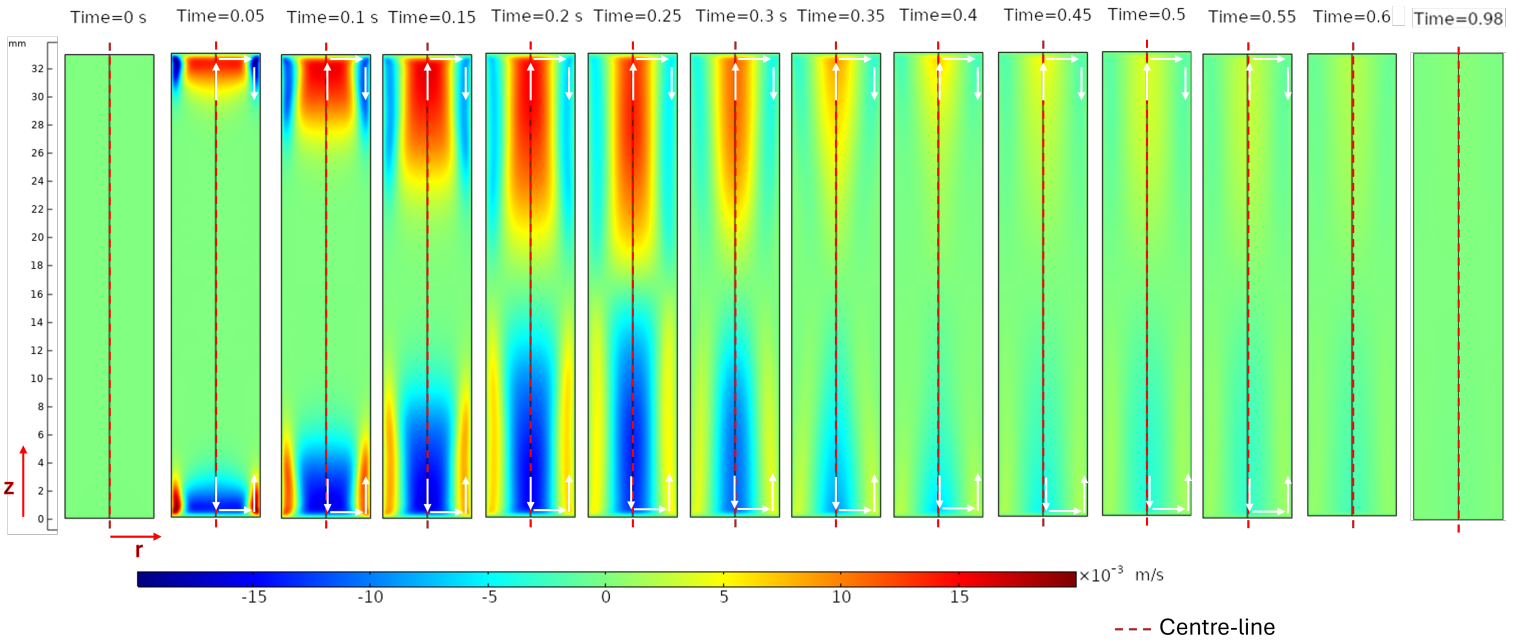


Figure 37: Contours of axial velocity during spin-up represented on a 2D cut of the cylinder where the axial direction is represented vertically and the radial direction is represented horizontally at different times from t=0 s (start) to t=0.98 s (end). The white arrows represent the direction of secondary flows.

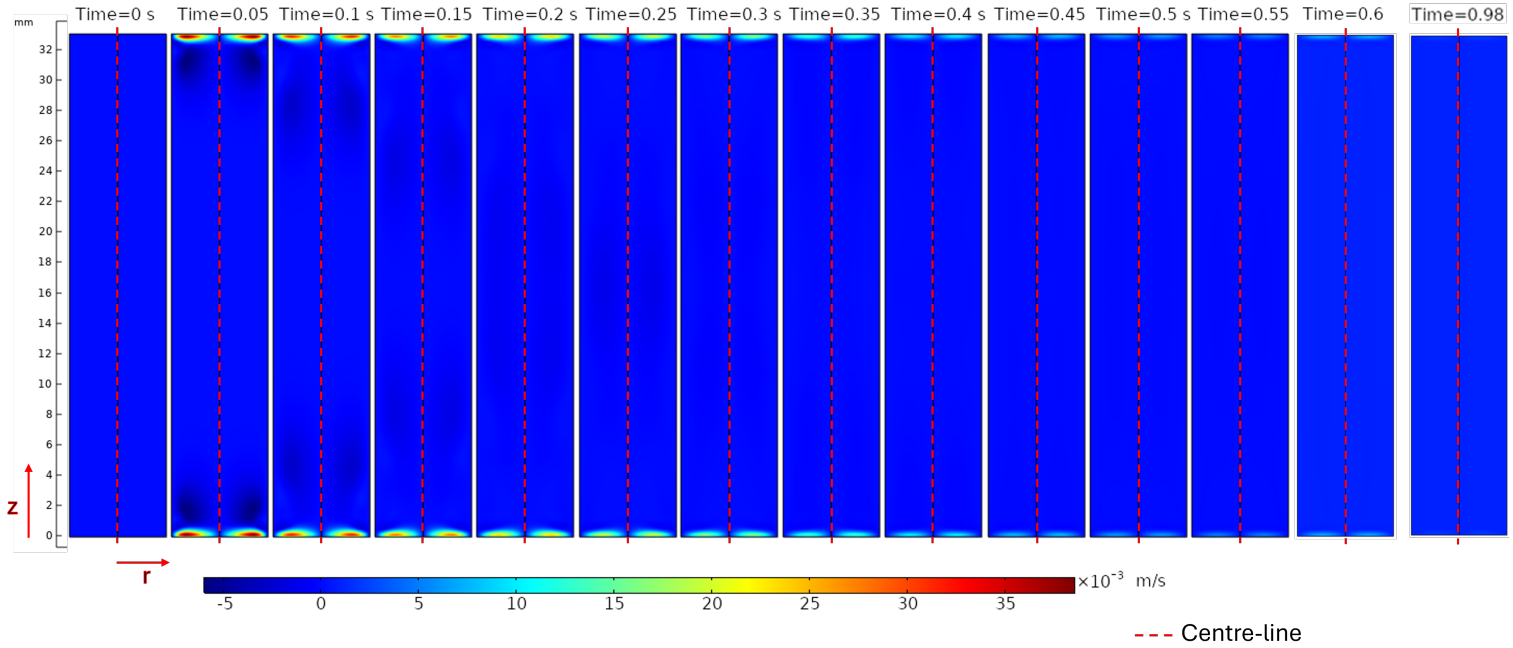


Figure 38: Contours of radial velocity during spin-up represented on a 2D cut of the cylinder where the axial direction is represented vertically and the radial direction is represented horizontally at different times from t=0 s (start) to t=0.98 s (end).

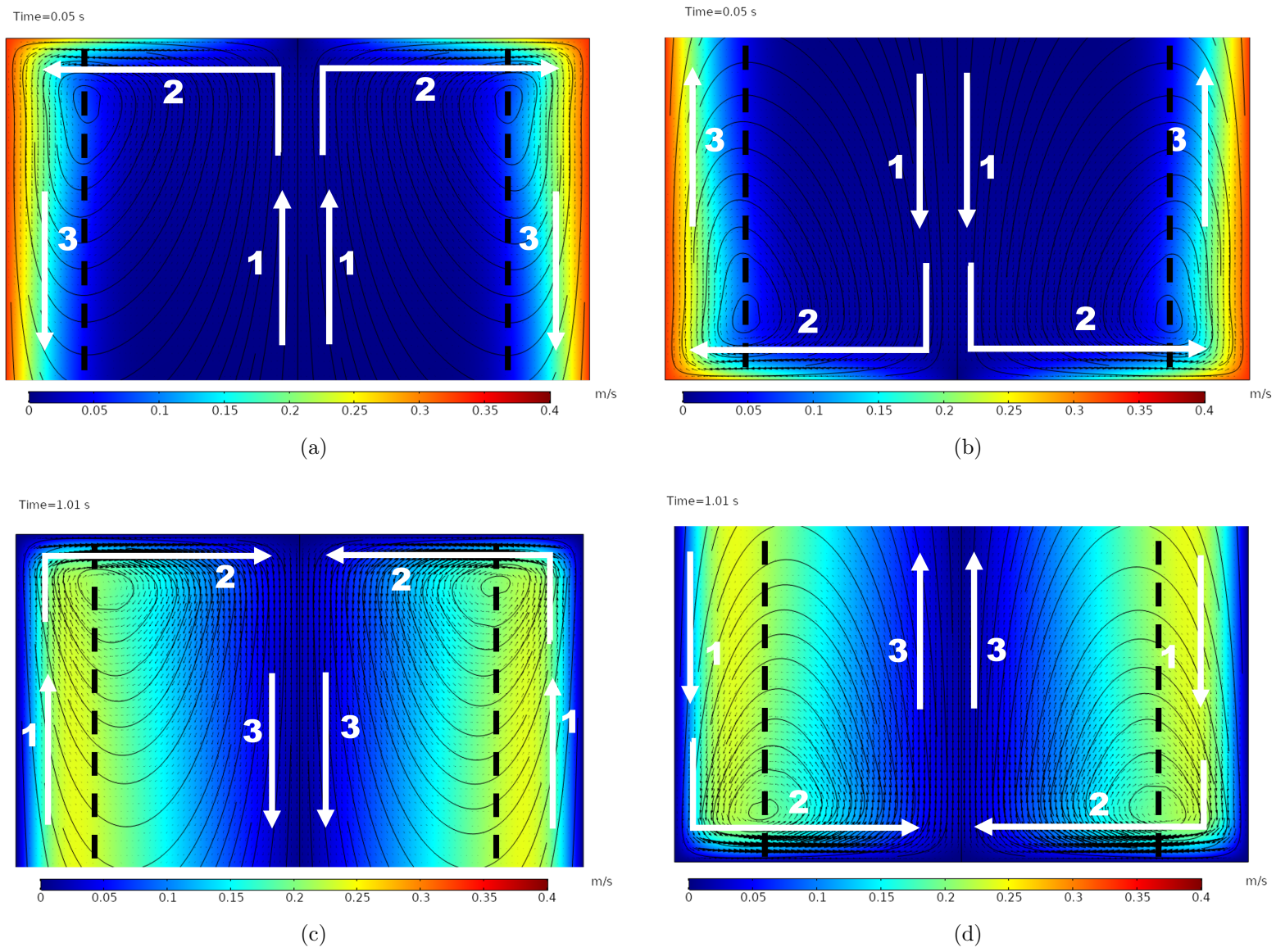


Figure 39: Comparison of secondary flow directions at top and bottom of the cylinder during spin up (t=0.05 s) and spin-down(t=1.01 s). Each plot shows white arrows indicating general fluid movement with black dashed lines dividing the upward and downward motion. The streamlines are also plotted. (a) Top of the syringe during spin-up, (b) bottom of the syringe during spin-up, (c) Top of the syringe during spin-down, (b) bottom of the syringe during spin-down

4.2 Spin-Down

4.2.1 Primary Flow

From the spin-up analysis, the results show that the fluid is exhibiting solid body rotation at a time of 0.98 s for the range of parameters considered here. In this section, the focus is on what happens as a fluid in a solid body rotation is instantaneously stopped. The start of spin-down is defined as the point at which the rotation of the cylinder about its central axis is stopped instantaneously. As before, the primary motion during spin-down can best be understood by examining the velocity magnitude contours, as shown in Figure 40, where spin-down starts at 1 s. For the fluid along the central axis (red dashed line), there is a line of $\omega = 0$ where the fluid is stationary. Near the cylinder sidewalls, the blue region shows that the velocity of the cylinder sidewall is zero and the fluid has stopped instantly with the sidewalls because of the no-slip condition. Between 1 s and 1.09 s, the shear stress between the inner core and the outer walls causes the fluid to slow down resulting in a rapid decrease in velocity within the boundary layer. Moreover, as time increases, the velocity of the fluid decreases as a result of diffusion of momentum. This continues until all the fluid comes to rest. This matches stage 1 (laminar boundary layer stage) of the five stages described by Kaiser et al. (2019) for a cylinder coming to rest. For a more intricate understanding, one can apply the same analysis conducted for spin-up to spin-down by looking at the normalised velocity magnitude, $\bar{v}$, across the same red line used for spin-up in Figure 31. The resulting plots for various Reynolds numbers at different times through spin-down are shown in Figure 41.

Figure 41a shows the lowest Re of 500, and the diffusion of momentum of the fluid to rest, over a period of 0.8 s. A short time after spin-down begins, the highest velocity recorded relative to the cylinder sidewalls velocity during spin-up is approximately 80% of the maximum velocity. This is consistent across the range of Reynolds numbers. Moreover, the maximum $\bar{v}$ in the earlier time frames is recorded near the outer cylinder, and as time increases, this peak is reduced and moves towards the centre of the radius. This can be compared to the velocity contours showing dark green areas (higher velocities) between r=0 and r=a, with blue contours (i.e. slower velocities) both at the central axis and at the cylinder sidewalls. The general behaviour observed in Figure 41a shows that the peak reduces and moves towards a normalised radius of 0.5. This diffusion of momentum in the velocity profile is relatively consistent compared to higher Reynolds numbers. For example, Figures 41b to Figure 41f do not show a similar consistency in the reduction of the peak as time passes. Instead, more of the peak is flattened and reduced near the cylinder sidewalls than closer to the central axis of the cylinder. All the curves exhibit a similar behaviour. Early time points see the velocity near the walls drop rapidly, and a boundary layer grows from the outside walls towards the core. There are some differences in the dynamics of the dissipation of energy as the fluid comes to rest. For Low Re cases (Re=500), the peak in velocity, decrease in size and moves towards the core in a monotonic manner, whilst the higher Reynolds number case show a quicker movement of this peak towards the core. By scaling the velocity, it appears that the lower Re case takes longer to slow than the higher Re case. Comparison can be made with the maximum normalised velocity peak at t=1.2 s for Re=500 (0.25), with the maximum

normalised velocity peak of Re=3000 (peak of black curves for both Reynolds numbers).

By scaling the velocity, as follows:

$$\frac{Peak_{Re=3000}}{Peak_{Re=250}} = \frac{V_{norm@Re=3000} \times 3000}{V_{norm@Re=500} \times 500} = \frac{0.1 \times 3000}{0.25 \times 500} = 2.4$$

the peak of Re=500 is moving 2.4 times slower than the peak at Re=3000.

For most Reynolds numbers, the inner 30% of the radius maintains more or less its $\bar{v}$ with a slight reduction due to momentum diffusion. One can also compare the same velocity profiles at a fixed time for various Reynolds numbers. This is clearly shown in Figure 42 and Figure 43. These show the velocity profiles across the centre of the cylinder of the various Reynolds numbers at fixed times, starting at t = 1.01 s (slightly after the spin-down begins) to t= 2.0 s. The general trend shows the diffusion of the velocity profile through the reduction of the peak and its highest point moving towards the centre of the radius. However, at t=1.04 s, between Re = 1000 (green) and Re = 1500 (red), the flow behaviour changes from smooth velocity profiles to more flat curves. This change indicates that the flow may have moved from stage 1 to stage 2 described by Kaiser et al. (2019) and chaotic flows may be present within the cylinder, and this will be discussed later in the chaotic flow section of this chapter.

Finally, Figure 43e shows that for all Reynolds numbers, at 2 seconds, the maximum velocity of the fluid relative to the initial spin condition of the cylinder is less than 0.75%. The fluid in the centre of the cylinder is essentially stationary at this point. Another result to observe is that the higher Reynolds number diffuses momentum much faster than the lower Reynolds number. For example, Re 3000 shows all $\bar{v}$ values below 20% of the sidewall spin velocities at t = 1.1 s, while the same occurs for Re = 500 between 1.2 and 1.4 s. To make a direct comparison, one can compare the normalised velocity magnitude against the normalised radius for two instances of Reynolds number with comparable time scales. For example, Reynolds number 500 at t=1.08 s (0.08 s after spin-down) can be compared to Reynolds number 1000 at t=1.04 s. This is shown in Figure 44a. The normalised velocity magnitude is relatively higher towards the outer edge of the cylinder (higher radius) for a larger Reynolds number. This is expected as the fluid at high Reynolds number is expected to move faster. Figure 44 shows the same plot for Reynolds number 1500 at t=1.08 and Reynolds number 3000 at t=1.04 s. This goes well beyond the critical Reynolds number of 675, which is associated with the onset of centrifugal instabilities in rotating flows for this case, beyond which the flow transitions from laminar to unsteady and potentially chaotic behaviour. The non-uniformity in the reduction of the peak at higher Reynolds numbers could be due to the detection of chaotic flows, which are explored in a later section of this chapter.

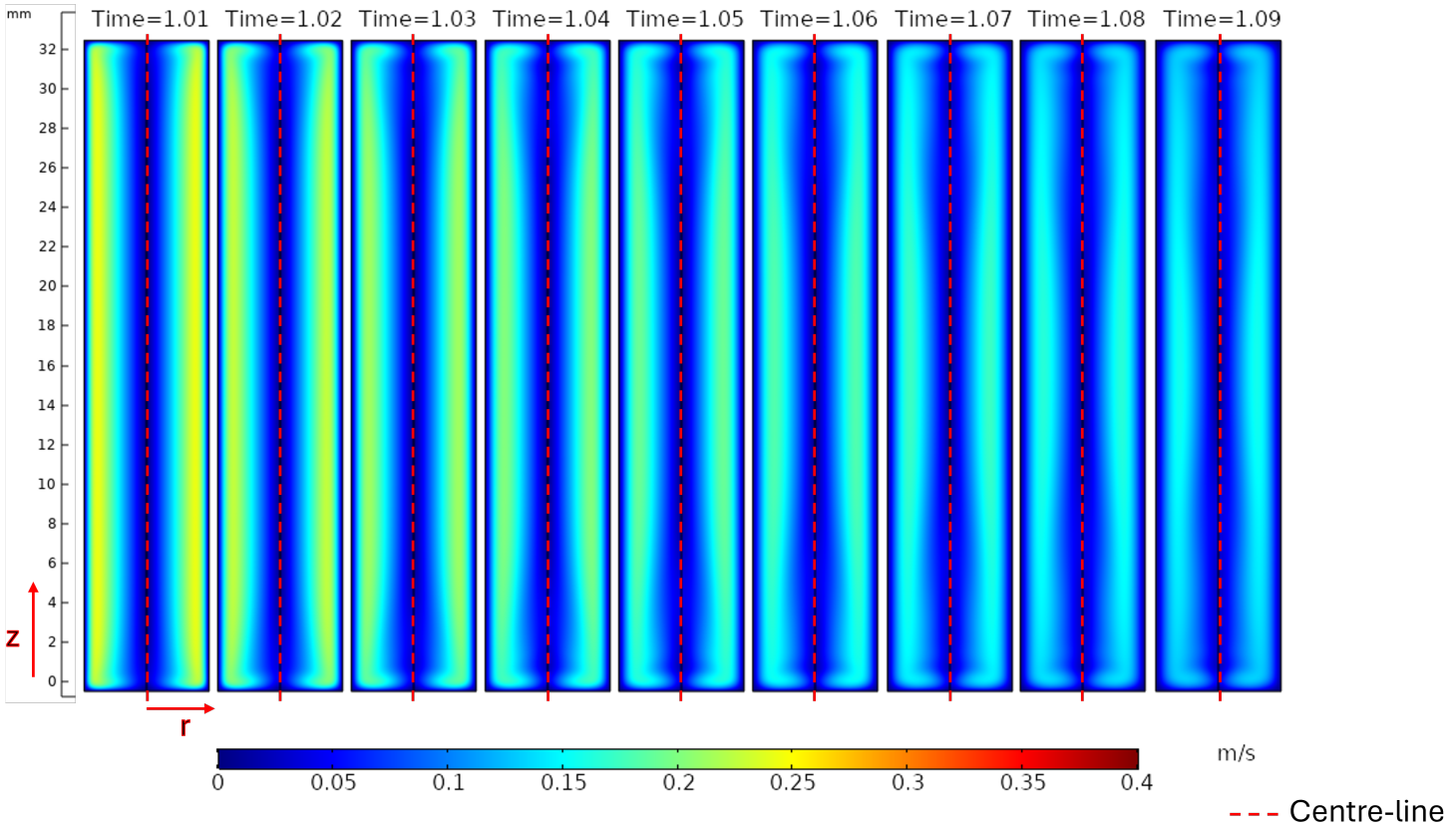
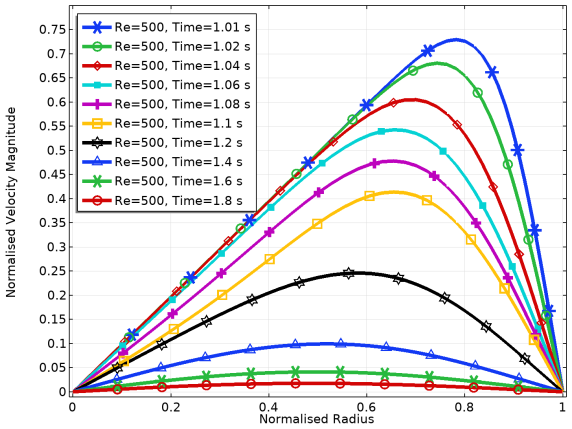
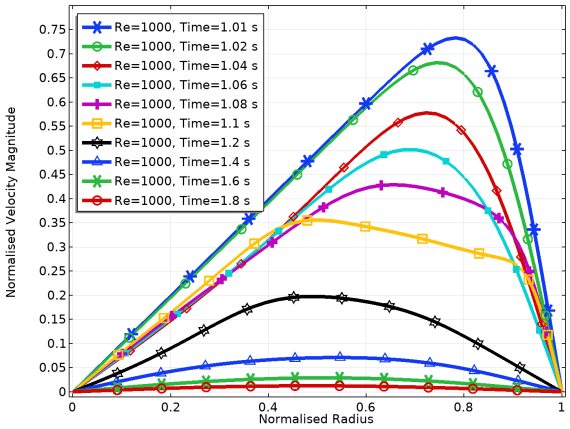


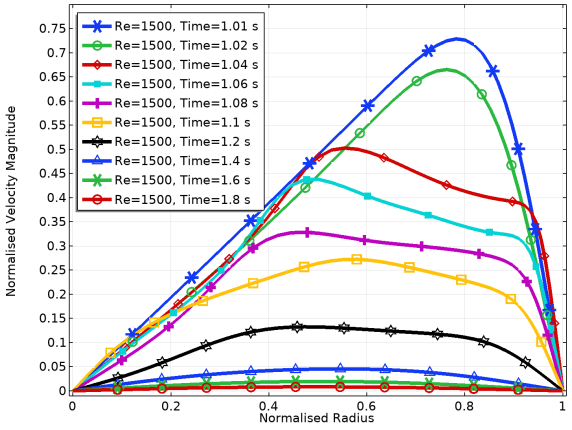
Figure 40: Contours of velocity magnitude during spin-down represented on a 2D cut of the cylinder where the axial direction is represented vertically and the radial direction is represented horizontally at different times from t=1.01 s (start) to t=1.09 s (end).



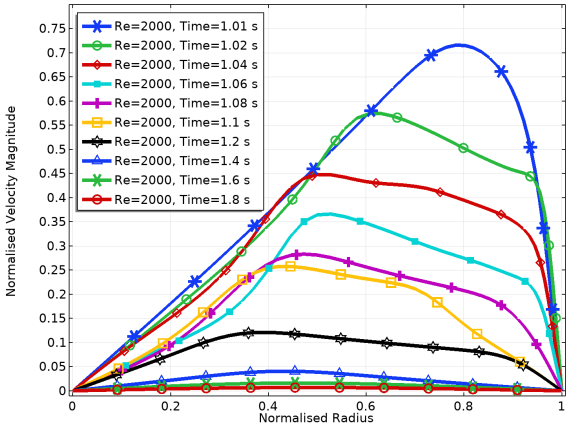
(a) Re 500



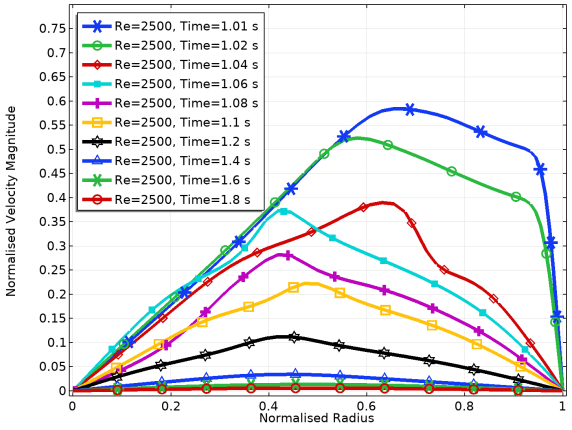
(b) Re 1000



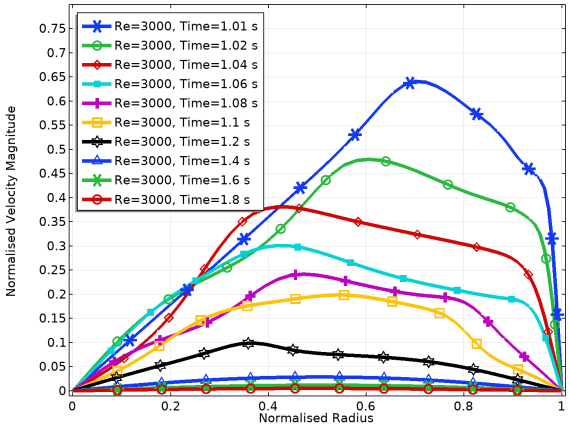
(c) Re 1500



(d) Re 2000

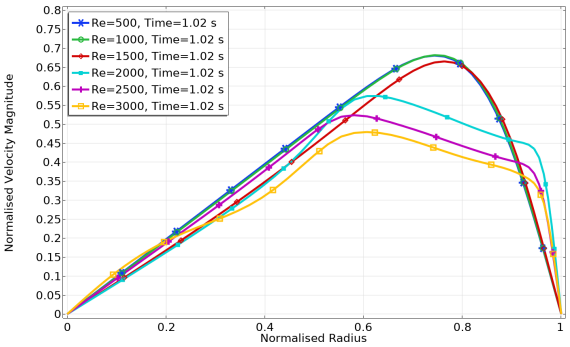


(e) Re 2500

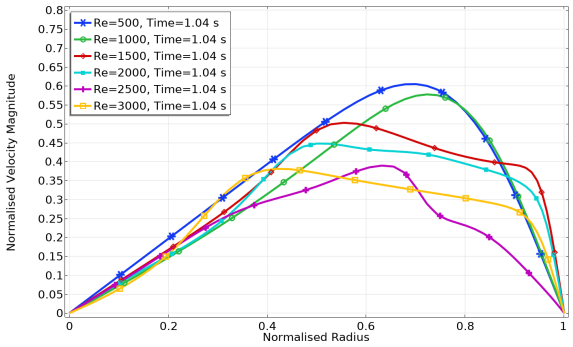


(f) Re 3000

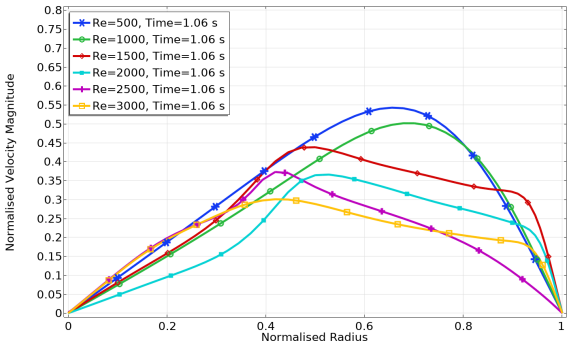
Figure 41: Spin Down: Normalised Velocity Magnitude against Normalised Radius for Reynolds Numbers of (a) Re = 500, (b) Re = 1000, (c) Re = 1500, (d) Re = 2000, (e) Re = 2500, (f) Re = 3000 at various times



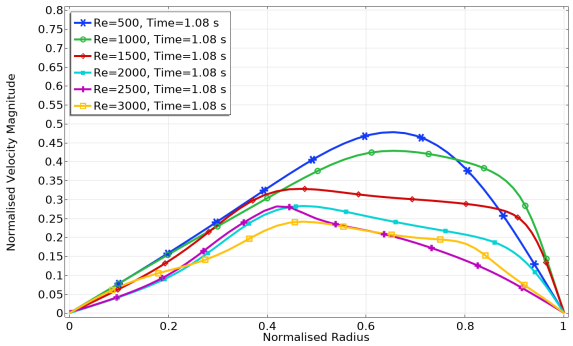
(a) t=1.02 s



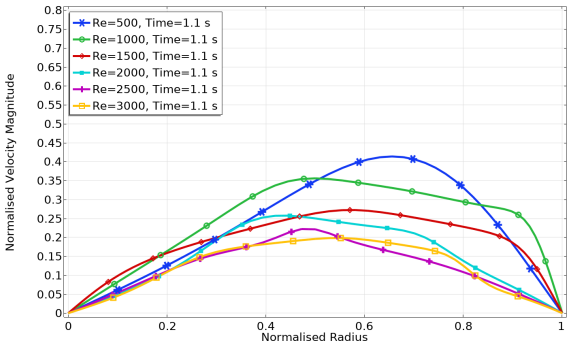
(b) t=1.04 s



(c) t=1.06 s

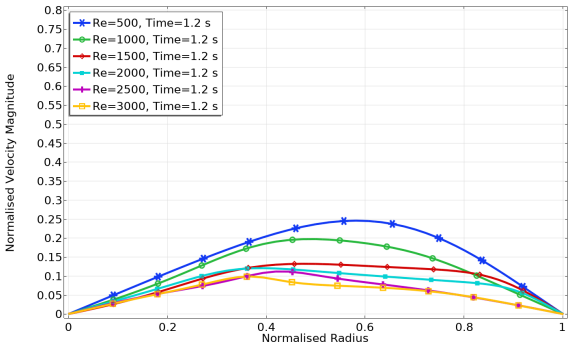


(d) t=1.08 s

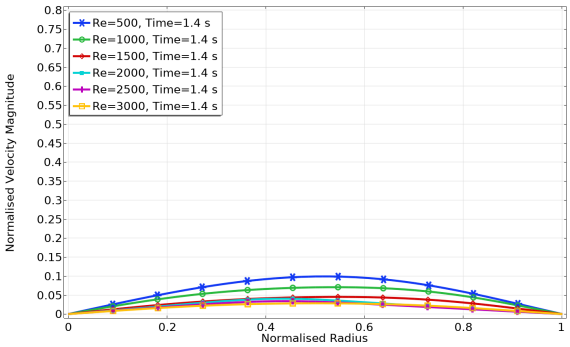


(e) t=1.10 s

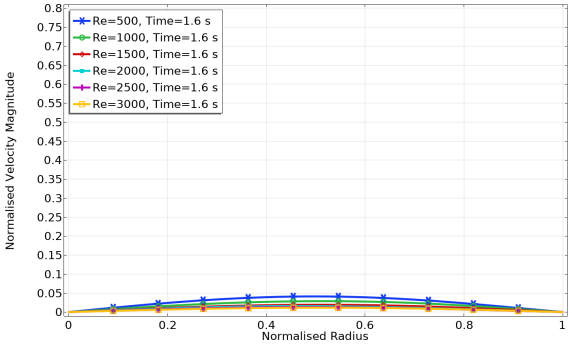
Figure 42: Spin Down: Normalised Velocity Magnitude against Normalised Radius for Various Reynolds Numbers at (a) t=1.02 s, (b) t=1.04 s, (c) t=1.06 s, (d) t=1.08 s and (e) t=1.10 s



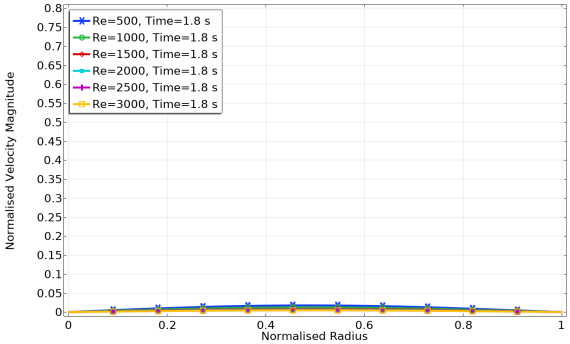
(a) t=1.2 s



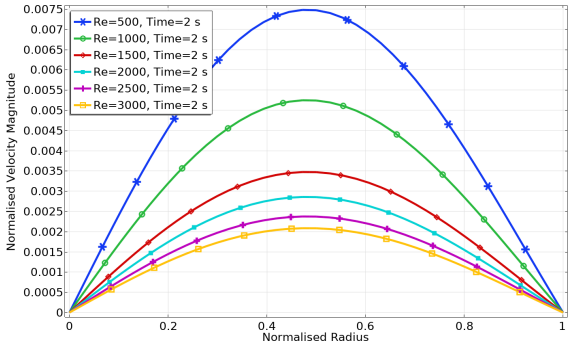
(b) t=1.4 s



(c) t=1.6 s

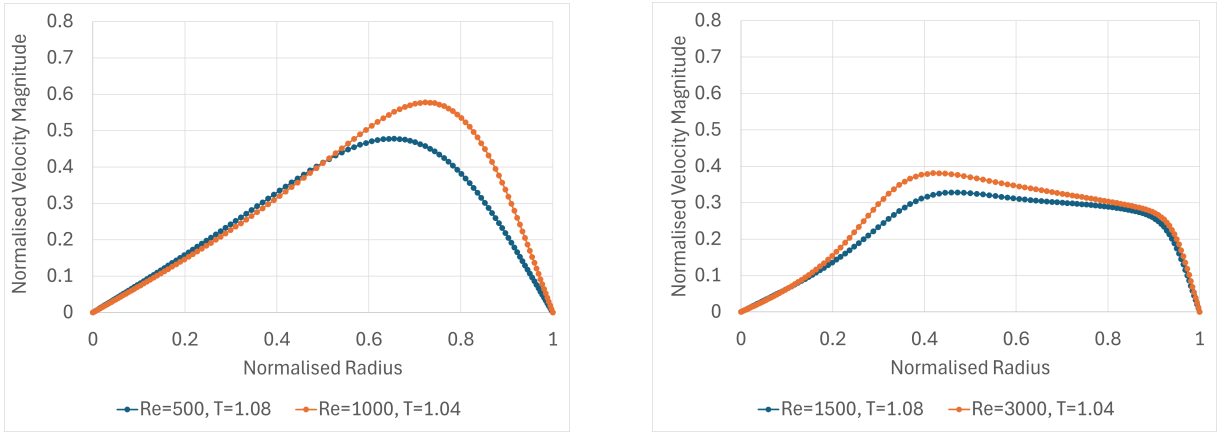


(d) t=1.8 s



(e) t=2.0 s

Figure 43: Spin Down: Normalised Velocity Magnitude against Normalised Radius for Various Reynolds Numbers at (a) t=1.2 s, (b) t=1.4 s, (c) t=1.6 s, (d) t=1.8 s, (e) t=2.0 s (Unscaled)



(a) (b)

Figure 44: (a): Normalised velocity magnitude against normalised radius before chaotic flows at Reynolds number 500 at t=1.08 s vs Reynolds number 1000 at t=1.04 s, (b): normalised velocity magnitude against normalised radius after chaotic flows for Reynolds number 1500 at t=1.08 s vs Reynolds number 3000 at t=1.04 s

4.2.2 Secondary Flow

Similar to the spin-up, the secondary flow velocities observed during the spin-down are at most an order of magnitude smaller than the velocity magnitude discussed in the primary flow. The contour of axial velocity during spin-down is shown in Figure 45, which shows the emergence of two axial jets from the top and bottom walls of the cylinder. As time increases to 1.01 seconds, the length of the jets increases as they permeate towards and reach the centre of the cylinder around t = 1.1 s. Near the sidewalls, the flow moves in the opposite direction to the axial jet towards the top and bottom walls. Within a few milliseconds of the onset of spin-down, the jets produced are sustained, and eventually the axial velocity is reduced as a result of diffusion. This can be observed in Figure 45, where the height of the jet is reduced. Eventually, all the fluid comes to rest (approximately t=3 s from the onset of spin-down). The radial velocity after the spin-down is shown on Figure 46. Similarly to the spin-up, the radial velocity is only non-zero at the top and bottom walls. Unlike spin-up, the direction of flow has been reversed, as represented by the change in colour of the contour plots. This also fits the description of the spin-down by Vanyo (2015) where the fluid near the sidewalls now moves towards the central axis. A summary of the resulting secondary flows for spin-down at the top of the cylinder is shown in Figure 39c, where the direction of the flow is reversed in comparison to the spin-up case. As before, the contour represents the velocity magnitude, and the black vector plots represent the axial and radial components of the velocity. The black dashed line represents the dividing lines between opposing fluid motions. Arrow 1 (white) shows the fluid near the side walls moving towards the top wall of the cylinder. This flow is then pushed radially from the sidewalls to the central axis of the cylinder (arrow 2). Finally, the flow at the central axis of the cylinder is pushed down towards the centre of the cylinder. For the secondary flow observed on the bottom wall ($z = -\frac{1}{2}h$) of the cylinder during the spin-down, Figure 39d can be rotated 180°.

One of the aims of the project is to understand the fluid flow so that the movement and inspection of the particles can be optimised. To do this, understanding particle fluidisation is essential. From the results shown in this chapter, spin-up is not ideal for a few reasons. Consider the Reynolds number of 1000 shown in Figure 36, the axial velocity across the centreline is near zero at the end of spin-up. This is will not aid with the fluidisation of particles needed for inspection purposes. Moreover, due to the specific directions of the flow associated with secondary flows during spin-up, any particles away from the top or bottom walls may move axially towards the top and bottom walls, then radially outwards and near the walls of the cylinder (see the arrows in Figure 39b). The strength of these jets do not last very long. That is, for Re = 500, the strength of the jet shows almost complete diffusion by 0.4 s as shown in Figure 37. For industrial applications, cameras focused near the centreline are easier to setup than those looking at sidewalls. The platform on which the PFS are mounted is also rotating and moving along a conveyor belt system. In comparison, the jets formed during spin-down last longer (see Figure 45), giving a larger inspection window. The position of the jet during spin-down allows for better camera placement to aid with the inspection process. For all the reasons mentioned above, the use of spin-down for the purpose of fluidising and inspecting particles is ideal.

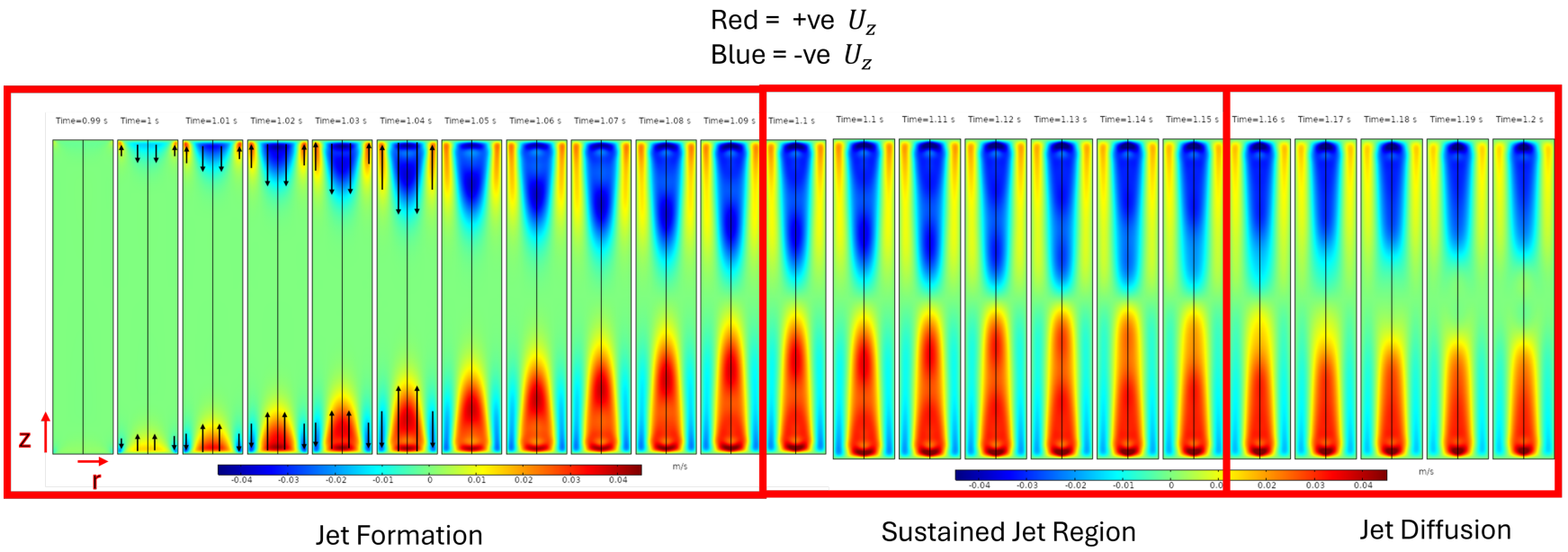


Figure 45: Contours of axial velocity during spin-down showing jet formation and breakup during spin-down

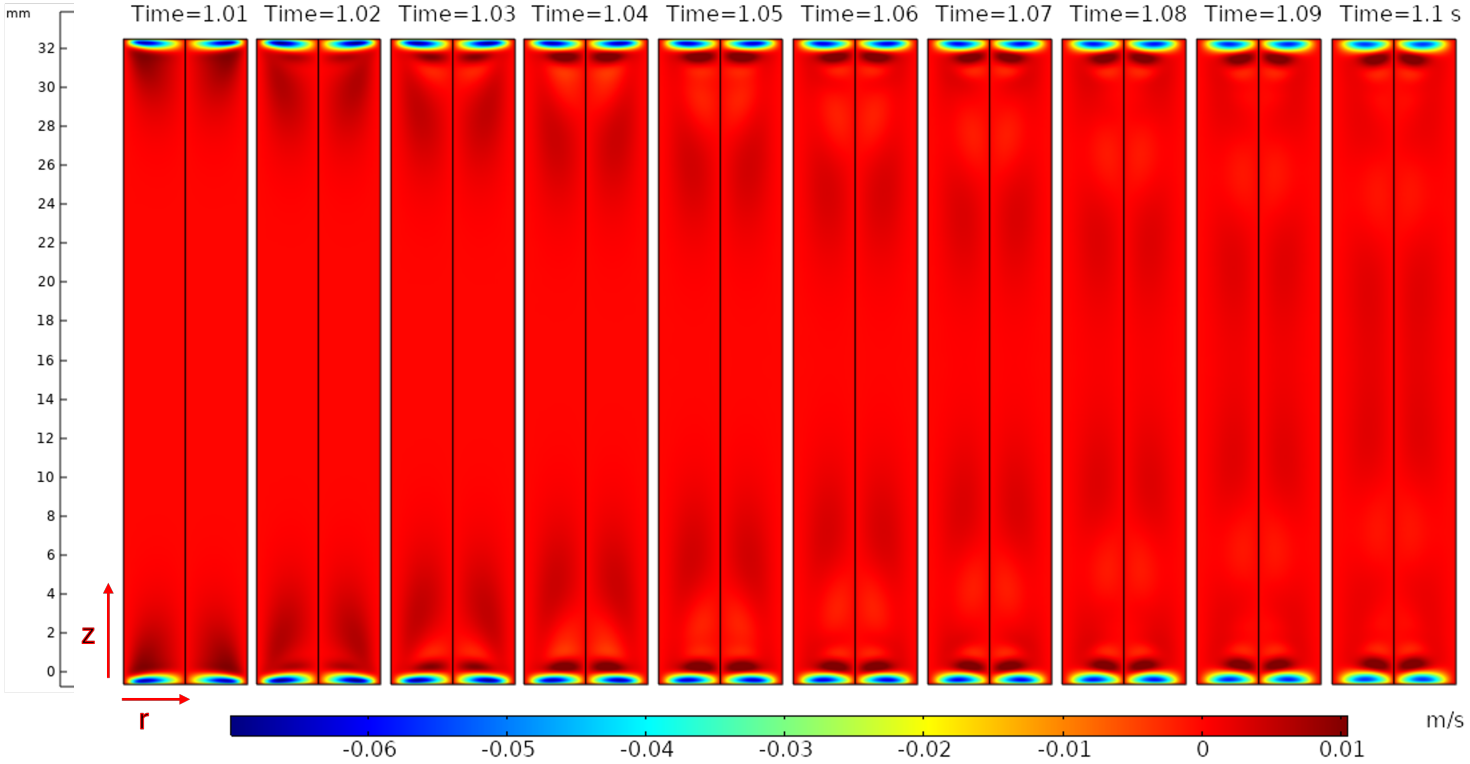


Figure 46: Contours of radial velocity during spin-down represented on a 2D cut of the cylinder where the axial direction is represented vertically and the radial direction is represented horizontally at different times from t=1.01s (start) to t=1.1s (end).

4.3 Effect of the Reynolds number on spin-down

Spin-down plays a critical role in the detection of particles during an inspection. During spin-down, the abrupt stopping of the cylinder walls leads to a transient fluid state where competing viscous and inertial forces dictate the evolution of the flow patterns. These flow patterns result in the fluidisation of the particles that are to be detected. Although earlier sections have detailed the primary and secondary flow dynamics during both spin-up and spin-down, this section focusses on the behaviour of these flows at higher Reynolds numbers. For Reynolds numbers exceeding the critical threshold ($Re_{crit} \approx 675$), the fluid exhibits complex structures marked by jet collisions, the formation of counter-rotating vortices, and the transition to chaos (as shown by Kaiser et al. (2019) on Figure 16). These mark a distinct difference between the low Reynolds number flows described in the previous section, where the flow is laminar.

By identifying key flow features such as jet interaction and vortex formation, one can begin to understand the chaotic flow patterns that are observed at these higher Reynolds numbers. This will help with the understanding of both the theoretical fluid flow and the practical implications of this kind of flow for the application to prefilled syringes and, in particular, the movement of particles during inspection.

4.3.1 Below the critical Reynolds number

Lets start by considering a cylinder where the aspect ratio, AR, is calculated using Equation 10. Now lets consider the axial velocity contours for a cylinder with AR=1 and Re = 500 ($\approx 1372\,\mathrm{RPM}$), which is below the Re_{crit} of 1120 (for AR=1). Figure 47 shows the square domain where the green box is the area of focus for which the axial velocity is shown. These are five different times (t = 1.05 s, 1.10 s, 1.15 s, 1.20 s, and 1.25 s) after the spin-down begins. As expected, the flow moves down near the outside cylinder walls, as represented by the blue region. The red region represents the upwards motion. Looking at the bottom left quadrant of the cylinder, i.e. bottom left quadrant of the domain, the flow moves in an anticlockwise direction around the stagnation point labelled with the number 1 in a yellow circle. The dissipation of momentum as time passes is observed as the fluid comes to rest. This is evident from the intensity of the red and blue regions at different times.

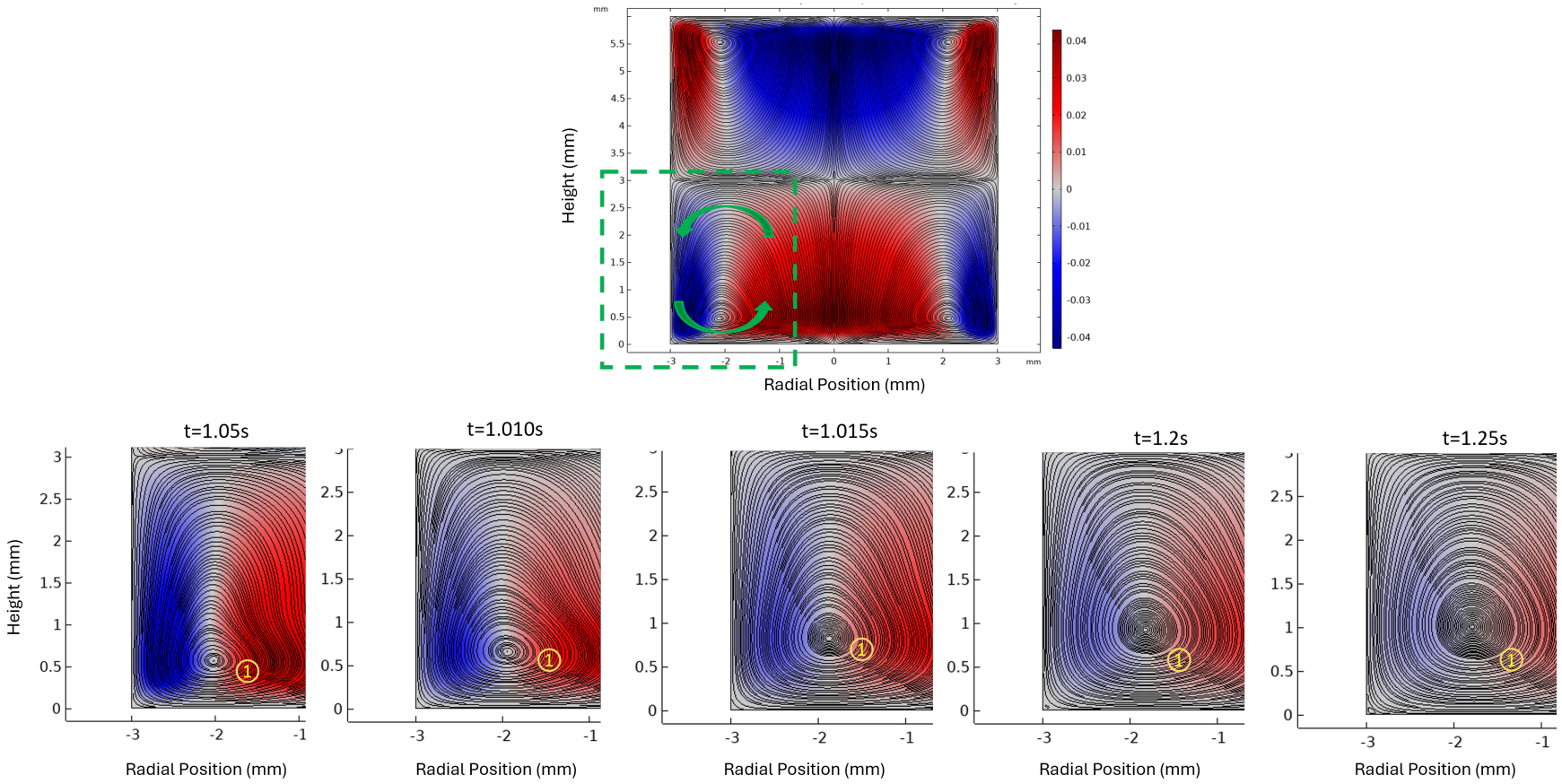


Figure 47: Re_{crit} = 1120 ≈ 3074 RPM , Re= 500 (≈ 1372 RPM), AR=1, bottom left quadrant showing streamlines and axial velocity components for a time of t= 1.05 s , 1.010 s , 1.015 s ,1.20 s and 1.25 s. The yellow circles labelled "1" represents stagnation point

4.3.2 Jet interactions at the critical Reynolds Number, Re_{crit}.

Re_{crit} is the point at which the two vertical jets collide, leading to the first formation of three initial vortices, where their interactions govern the fluid behaviour. Continuing on from the last section, lets consider the same AR=1 domain and keep the focus on the bottom left side of the quadrant. Recognising this is an axisymmetric problem, so the other quadrants reflect similar fluid behaviour. Figure 48 shows images captured at various times after the spin-down begins, highlighting the change in the red and blue flow regions due to the interaction of the jets. A schematic is also drawn for the blue region, which specifically highlights the fluid changes between each image. Initially, the region associated with downward motion (blue region) is symmetric. This is just before the interaction of the jets. Once the vertical jets collide, the blue region experiences necking caused by the radial flow from the centreline towards the outside cylinder walls. This radial flow is highlighted by the red arrow in the second schematic, causing the necking.

The next observation is the formation of additional recirculation points, labelled "2" and "3". These are marked with green and black circles to distinguish them from each other. Although "3" is a recirculation point, it is very weak and does not stay active very long. However, it does move down along with the core flow as it diffuses. The recirculation point "2" is much more interesting and appears to form due to the necking as the blue region moves to follow the shape of "S". It should be noted that points "1" and "3" are technically the same vortex as the fluid moves around in the anti-clockwise direction. The appearance of these recirculation regions is associated with the interaction of the opposing jet structures and the resulting shear layers formed during the necking process. As time progresses, the recirculation point "2" is convected downward as it reaches the bottom third of the quadrant and weakens as viscous diffusion reduces the strength of the velocity gradients. Figure 49 can be used as a reference as it shows the same plots but at various times. This shows the formation and eventual dissipation of the new recirculation points after spin-down. Note that the final stage is very similar to the initial stage with only one strong recirculation point (point "1").

From the different stages described by Kaiser et al. (2019), as the Reynolds number is increased further, the behaviour of these (and more) recirculation points will be observed. The interaction between the different boundary layers and the core jet flows will be analysed.

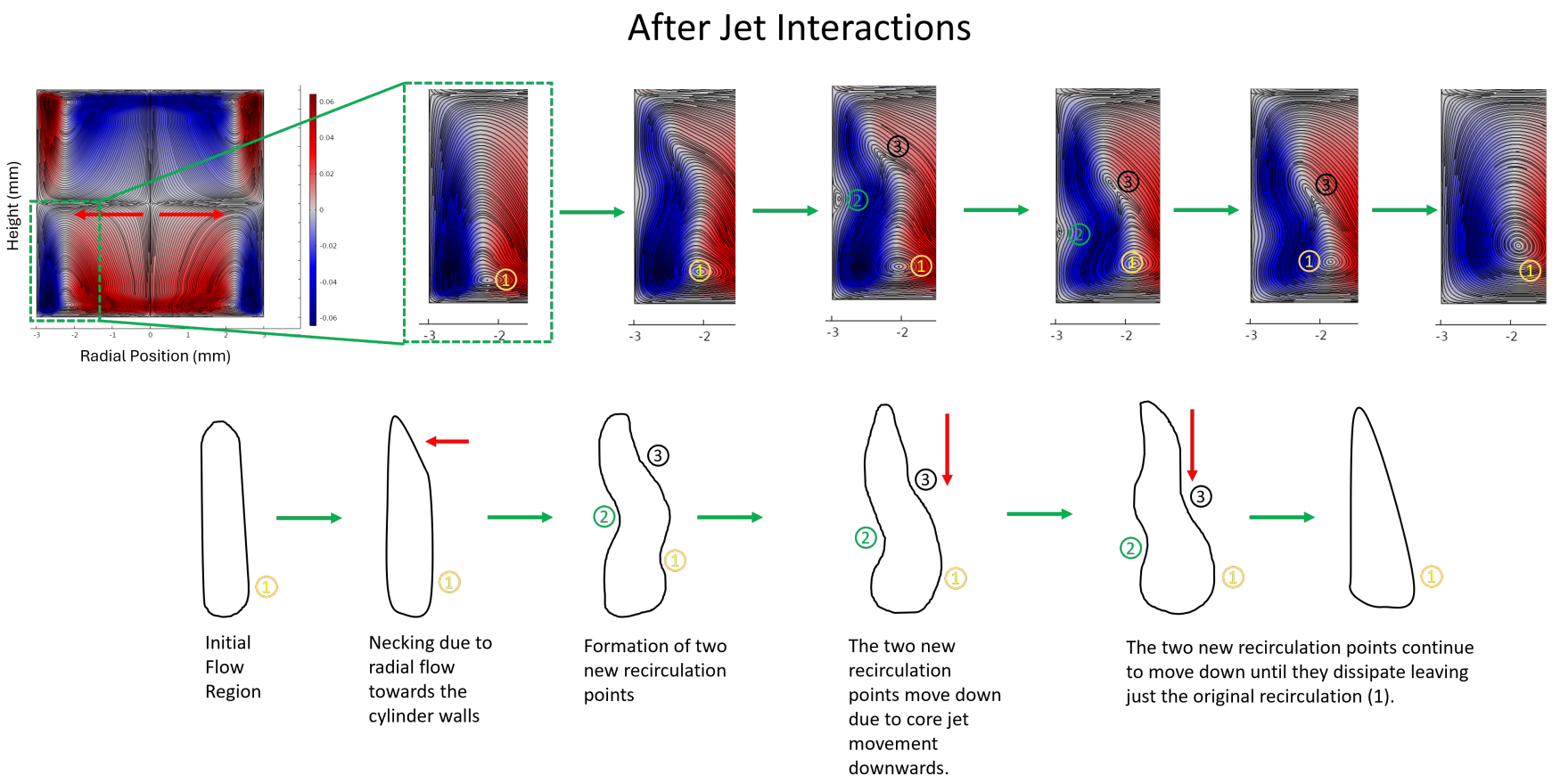


Figure 48: $Re = Re_{crit} = 1120 \approx 3074\,\text{RPM}$, AR=1, Top figures show the bottom left quadrant with streamlines and axial velocity component, coloured circles representing stagnation point. The bottom figures show a schematic of the main flow features observed.

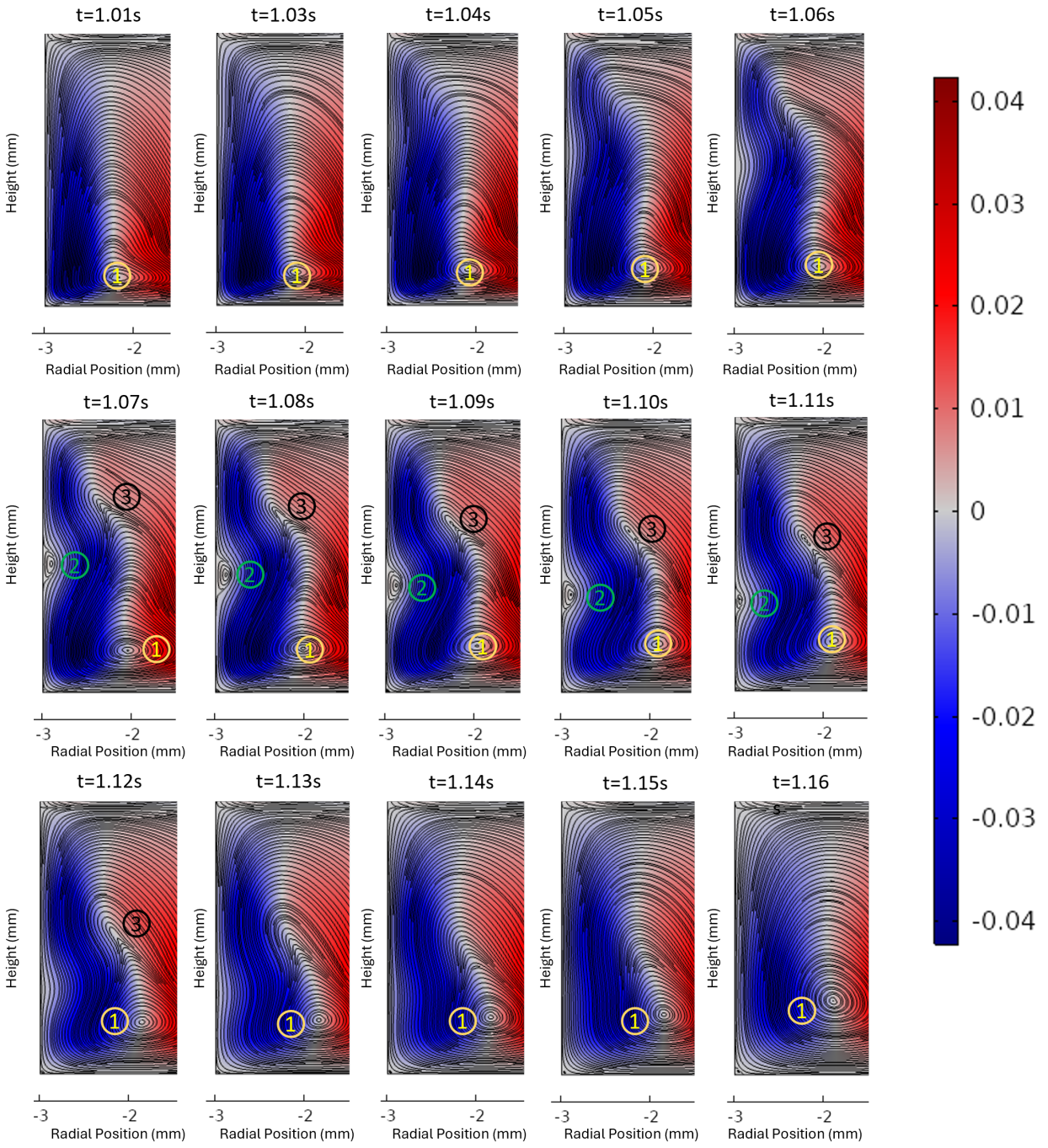


Figure 49: $Re = Re_{crit} = 1120 \approx 3074\,\text{RPM}$, AR=1, bottom left quadrant showing streamlines and axial velocity (represented by the blue/red colours) component, coloured circles representing stagnation point. A schematic summarises the observed flow features.

4.3.3 The development and progression of the vortices beyond the critical Reynolds Number, Re_{crit}.

As the Reynolds number exceeds well beyond the critical value ($Re_{crit} = 1120 \approx 3074\,\text{RPM}$), the fluid behaviour drastically changes from what has been discussed in the previous section of jet interactions.

Consider the bottom right side of the domain as shown in Figure 50, where two different types of fluid behaviours being observed; the formation of the boundary layers and Taylor-Couette vortices forming on the top right (shown by the rolls at t=1.006s). The figure shows the early development of the flow through some of its stages. There is also a sharp boundary layer at the bottom because of the bottom wall of the cylinder. Due to the axis-symmetric nature of the setup, there is also an imposed symmetry along the top wall, and this is reflected onto the other quadrants.

At t=1.000 s, the spin-down has started. At that point, near the cylinder walls, the no-slip condition causes the fluid near the walls to come to rest. The inner core is still spinning due to inertia; hence, there is also a boundary layer that forms between the cylinder walls and the core of the flow (i.e. the jet). The red region shows the dominant core flow with +ve u_z and the blue region with -ve u_z. At t= 1.003 s, the recirculation points 2 and 3 in Figure 49 form and are the result of the interaction of the jets explained in the previous section.

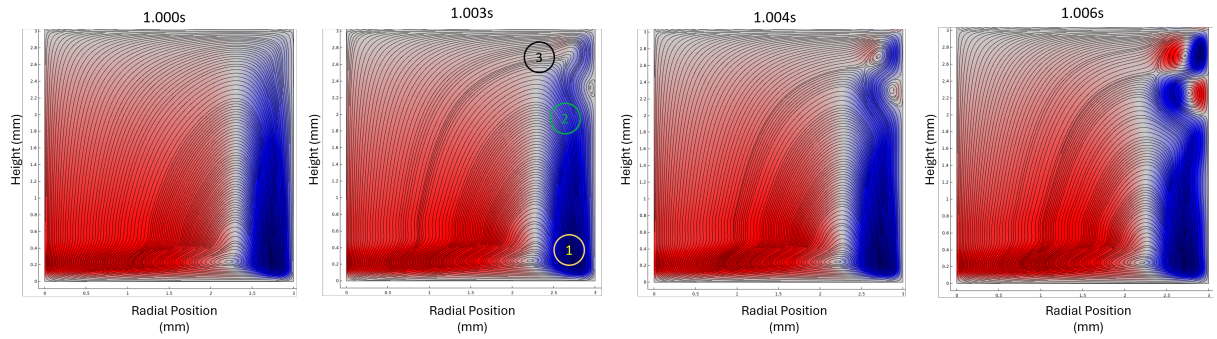


Figure 50: Formation of Taylor-Couette vortices near the top right section of the domain. (AR = 1, Re = 2000 for T=1.000 s, 1.003 s, 1.004 s and 1.006 s)

The two vortices forming at the top right are similar to a classic example of Taylor-Couette flow discussed in the literature. Due to the axis-symmetric nature of the model setup, one can observe only Taylor-Couette flows. This diagram is used to understand how different flow behaviours change under varying rotation rates of the cylinders. In this case, there are not two concentric cylinders. However, for spin-down, the outer radius, i.e., the outside walls are stationary. The core of the flow near the centreline is still moving, acting as a virtual inner cylinder with some velocity. The ratios between the inner and outer cylinders result in the behaviours seen with Taylor-Couette.

After 1.006s, the fluid interactions become chaotic. The interaction between the Taylor vortices with the side boundary layer as well as the core deviates from the traditional monotonic decay of the Taylor-Couette instability. In a infinite cylinder, this would normally contaminate the entire vertical domain, but here, this is not the case. Because of restrictions close to the bottom and top walls and no-slip boundary conditions, which do not allow the rolls to grow as easily, the growth occurs towards the centre of the domain first. Figure 51 shows the progression with time of some of these interactions.

From 1.010 s to 1.015 s, the Taylor-Couette instability on the top right grows vertically downward. At 1.015 s, there is another small blue region (representing negative velocity or downward motion) near the bottom right showing some pinching of the vortices. This shows a clear interaction between the Taylor vortex boundary layer and the bottom boundary layer. After 1.019 s, a vortex forms (highlighted by the blue region) at the top left of the domain which interacts with the Taylor-Couette vortices and the other boundary layers in the domain. At 1.025 s, there is another vortex that does the same but has its direction reversed. The evolving flow field shows a high degree of complexity as multiple vortices interact. This happens until the fluid comes to rest and is fully dissipated. This is probably due to Taylor-Couette taking place in a contained vessel. A lot more work needs to be done in this area to understand any further progress of the fluid behaviours, and this is well outside the scope of this project.

From the point of view of diffusing energy as time progresses, there is a pulsing of the core jet flow near the centreline, which is alternating between upward (red) and downward (blue) velocities. These oscillations act as a driving force that can feed energy into the smaller vortices that have already formed because of the collision of opposing jets. The central oscillations can transfer energy to the smaller vortices. This can either strengthen or weaken the vertical vortices depending on timing. The pulsing motion helps to initiate the formation of new vortices. As the central flow changes direction, it allows the fluid regions to pinch off new vortices. This explains why a regular pattern of alternating vortices is seen along the interface. There is also a synchronisation that takes place because of the central jet pulsing. This is seen in the somewhat regular spacing between the vertical vortices. These oscillations are likely the key driving force for the transition to chaos from the regular momentum diffusion seen in the pre-critical Reynolds numbers. With each pulse, the disturbances in the flow seem to be amplified as well as nullified in the alternating vertical vortex. Hence, this leads to complex flow patterns as the instability grows. One should note that this takes place during spin-down, where the flow is eventually coming to rest in its end state.

A more detailed analysis of the evolving flow fields in such geometries would serve as a useful adjunct to the more conventional Taylor-Couette literature. However, this is beyond the scope of the present study. This work is more interested in the overall flow and its interactions with particle motion.

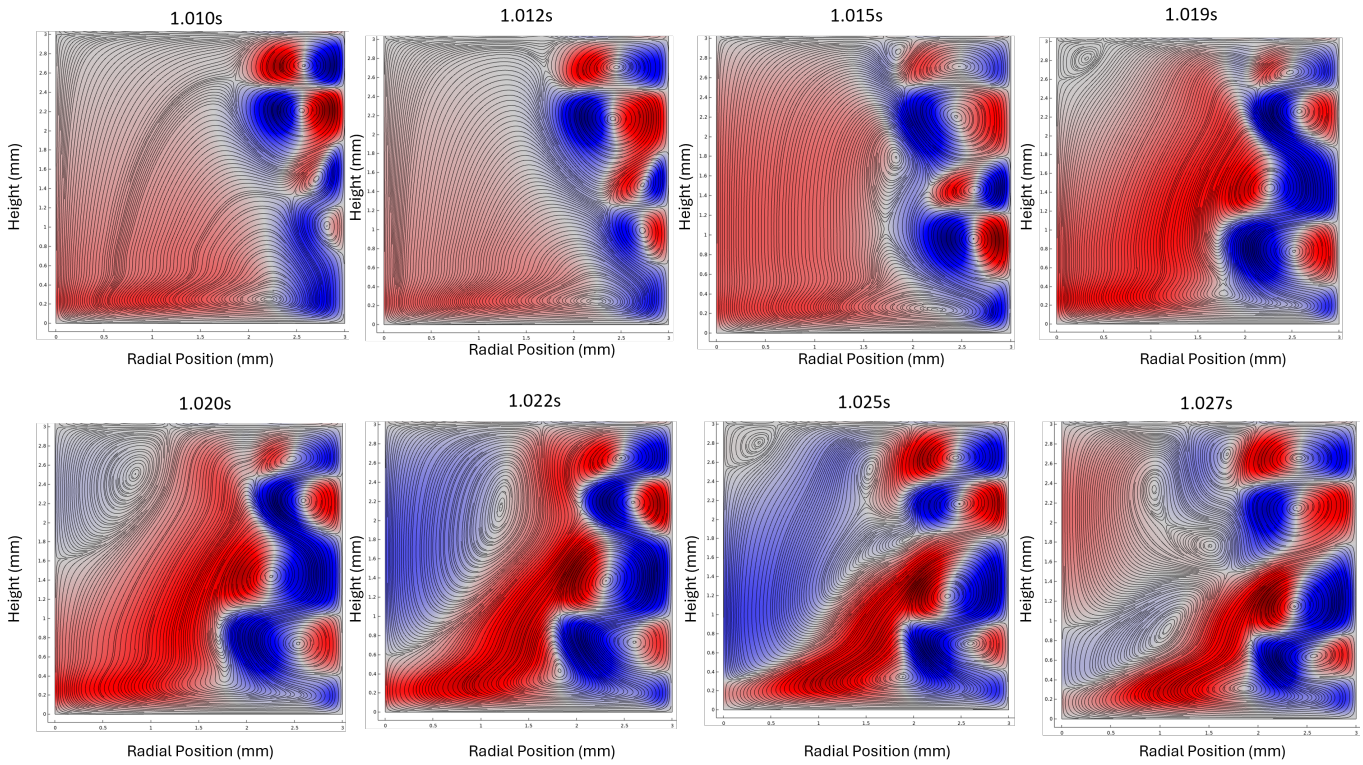


Figure 51: Interaction of Taylor-Couette vortices near the top right section of the domain with various boundary layers within the domain. The red and blue represent the axial velocity of the flow (AR = 1, Re = 2000 for various times after 1.010 s)

4.4 Aspect Ratio Study

As the industry uses various aspect ratios for their syringes or vials, the instance at which the flow becomes chaotic can be determined with respect to the geometry. By doing so, one can produce an operability map that can be used during inspection. To produce such an operability map, one can conduct the analysis like the Reynolds number study in the previous section but repeat this for various aspect ratios. This study is conducted for aspect ratios ranging from 0.5 to 6 as these cover the extent of medical devices used in the pharmaceutical industry. A typical syringe has about an aspect ratio of 5, but the fluid within can be much smaller depending on the dosage.

The critical Reynolds number has been approximated by the presence of the vortices detected in the previous section for each aspect ratio. As stated above, the critical Reynolds number, Re_{crit}, is identified as the Reynolds number at which two vertical jets interact, leading to the first formation of the 3 recirculation points discussed in Figure 48. This is approximated using velocity vectors, streamlines, as well as velocity and vorticity contour plots overlaid to determine the presence of these vortices for the first time. The initial instance in which all three vorticities are present before they diffuse is determined as the critical Reynolds number. This method is somewhat subjective and provides an approximation of the critical Reynolds number.

One method would be to look at the vorticity around the recirculation points and set a minimum threshold for the vorticity. However, due to time constraints, this method has not been adopted and is suggested as future work. As such, there is an error bar associated with the detection of these vortices with the current method. However, it does give an approximation of the critical Reynolds number, which is a good starting point for operators in the industry. The time at which each aspect ratio shows the 3 vortices the Reynolds number tested is shown in Figure 52. It is clear that the detection of the initial vortices occurs between 1 and 1.4 seconds for the given Re. This fits well with the inspection window used by the industry, and one can assume that the effects of the vortices will definitely play a role in the motion of particles within the inspection process. There is a sawtooth effect that is visible between AR = 2 and AR = 2.5. This is also observed between AR = 4.5 and AR = 5.0. A possible explanation for this could be due to bifurcations of the Taylor vortices that become prevalent and that change the time at which the vortex is detected. This has parallels with the complexity of Taylor-Couette bifurcation explored by Andereck et al. (1986) (see Figure 10).

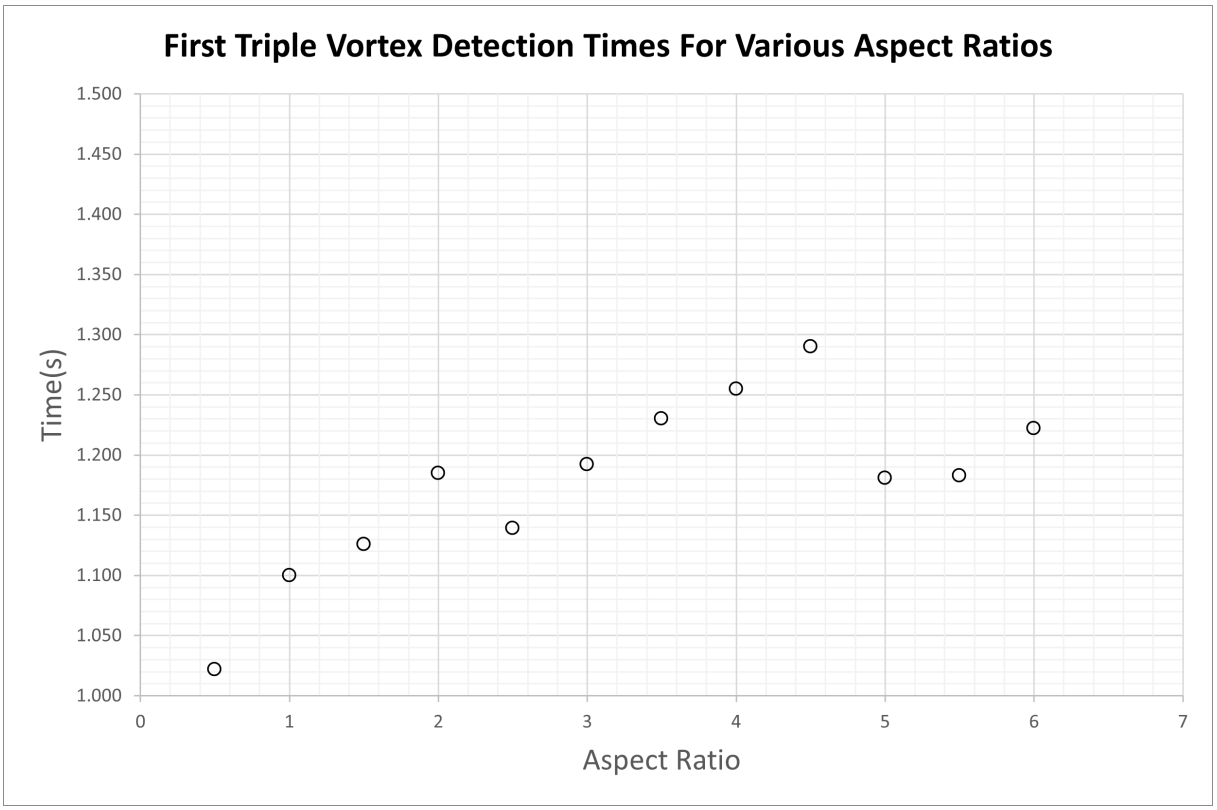


Figure 52: The time after spin-down until the detection of chaotic vortices for various aspect ratios for Reynolds numbers in Figure 53.

Figure 53 shows the Reynolds number at which the flow exhibits vortices against aspect ratio. This is defined as the critical Reynolds number for each aspect ratio. Above the red crosses highlighted in the figure, the flow exhibits vortices within the domain. As the aspect ratio decreases, the critical Reynolds number increases. From the aspect ratio of 3 onwards, the critical Reynolds number does not change significantly. This figure can be used as an

operability plot within the industry for syringes of known aspect ratios. Operators, knowing the aspect ratio of their PFS, can use this plot to determine the critical Reynolds numbers. for their PFS. The impact of the vortices on particles is explored in the later chapters.

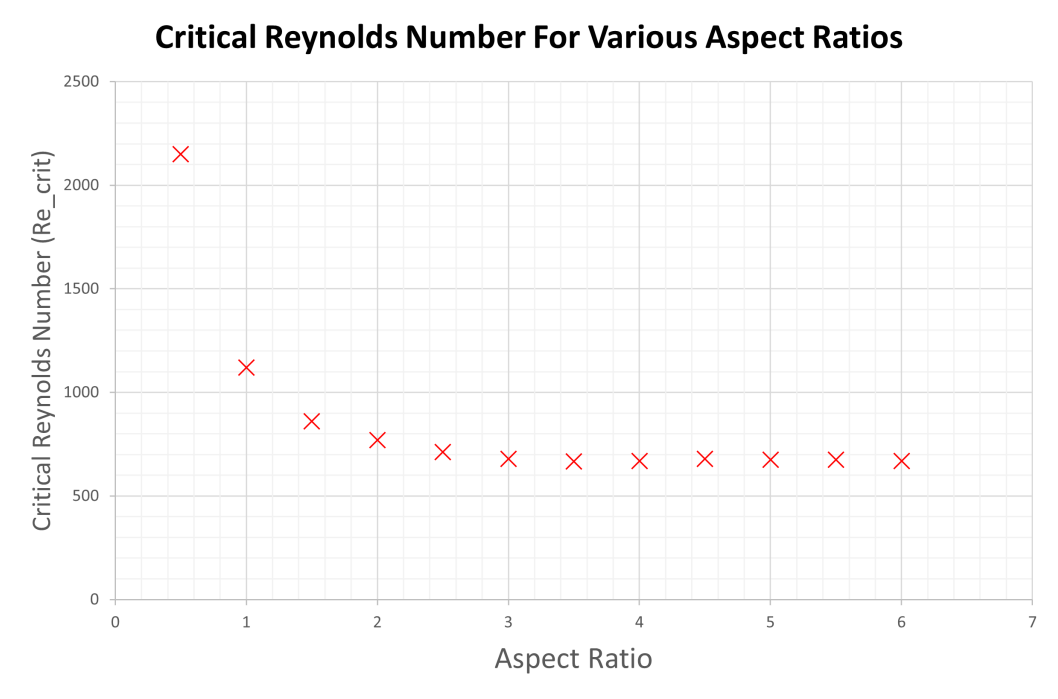


Figure 53: Operability map showing critical Reynolds number for various aspect ratios.

4.5 Conclusion

In this chapter, a deep analysis has been conducted for the typical fluid flow behaviour observed during a robotic inspection for a PFS. A cylinder has been used to model the flow of the syringe at various Reynolds numbers and aspect ratios. Initially, the spin-up phase is explored and primary and secondary flow features are explored. This is then expanded to spin-down where the reasons to the fluidisation of particles are investigated. In this study, one can now understand why the particles are fluidised and lifted during inspection. This is done by means of the axial jets that form during spin-down. Finally, observing the extreme cases at which the Reynolds number is at or beyond the critical value and its consequences on the fluid flow. The velocity profiles, boundary layer development, and momentum transfer mechanisms have been analysed across various Reynolds numbers, highlighting key differences in flow behaviour at subcritical, critical, and post-critical Reynolds numbers. These will also have an impact on the movement of particles within these flows. This analysis will allow the definition of the conditions that inspection should take place in to either avoid post-critical flows or use them to our advantage such as dislodging stuck particles on the surfaces of various parts of the syringe (i.e. plunger or wall). This study has also been expanded to create an operability map for various aspect ratios that covers different types of syringes and vials.

5 Particle Motion Studies: Computational Data Collection and Analysis

It is an extremely difficult challenge to model particles within the inspection of a PFS. This is due to the initial position of the particle and its properties, such as shape and material type. There are particles that may be stuck, that is, on the walls or plunger of the syringe, and may need to be dislodged. These particles may also be floating within the fluid or sink to the bottom immediately above the plunger, depending on the particle density. The behaviour of particles is also very different during spin-up and spin-down. The particle shape is also unpredictable and therefore makes modelling particles challenging.

In a typical real-world inspection, a syringe undergoes many processes before entering the inspection window that observes particles if they are present. These syringes may have been rotated, shaken, or even inverted prior to particle inspection. All these features of the inspection process have influence over the initial position of the particle and make this initial position of the particle highly random within the domain. This can also influence the initial position of the particle before the particle inspection can begin. This initial position of the particle matters because it will determine which features of the local flow influence the particle position over time. The motion of the particles is also affected by the shape and size of the particles. With that in mind, this chapter will study the motion of the particles under various initial releases based on different locations within the domain. These releases may be individual particles, a line of particles, or a grid of particles, each exploring the resulting particle tracks at those locations.

In this chapter, various types of analysis on the placement of the particles with mass and particles without mass are conducted. In COMSOL, massless particles are purely tracer particles that do not experience any inertia, momentum, or drag forces because they have no mass. Therefore, Newton's second law also does not apply, and acceleration or deceleration of the particle is not considered. By studying massless particles, the particle trajectories provide a Lagrangian perspective of the flow, allowing the paths taken by individual fluid elements to be tracked over a period of time (potentially, inspection window). While streamline and velocity field plots describe the instantaneous structure of the flow, particle tracking reveals how particles are transported within the domain. This is particularly important for understanding particle behaviour during the inspection window, where the likelihood of detection depends on the trajectories and residence time of particles within specific flow regions. These studies show the impact of the fluid motion on the particle and are a one-way coupled system. Therefore, in these studies, the effect that the motion of particles on the fluid is also not considered. Massless particle studies are useful for visualising streamlines, recirculation patterns, and flow structures using particles. With these studies, some limitations include that there is no complex interaction with the walls other than stopping (freezing) or bouncing elastically (without losing any energy). Although some studies use massless particles, others include the mass, density, and diameter of particles as well. The aim of this chapter is to determine the effect on the motion of particles resulting from the initial radial position, the initial vertical position, the change in the RPM or a change in the aspect ratio of the syringe. These are conducted at spin-up and spin-down to get an idea of the particle motion

during these phases. By doing so, one can get a general idea of what to expect for particle behaviours depending on these initial conditions.

5.1 Massless Particles

5.1.1 Spin-up: General Particle Motion

In this study, the effects of spin-up on particle motion is analysed. This will provide a clearer understanding of the location of particles at the end of the spin-up phase before spin-down can begin. Massless particles are injected at known radial distances from the centreline at a height of 0.5 mm. This height has been selected to represent particles that are likely to settle near the bottom region of the syringe prior to inspection. In practice, particles typically found (such as those in Figure 4) are denser than the fluid tend to migrate towards the lower regions of the domain under gravity, particularly during periods of low fluid motion. As such, this location provides a representative initial condition for particles that are most relevant to the inspection process. Figure 54 shows the location of particles injected at the start of spin-up (t = 0 s) for radial distances of 0.5, 1.0, 1.5 and 2 mm.

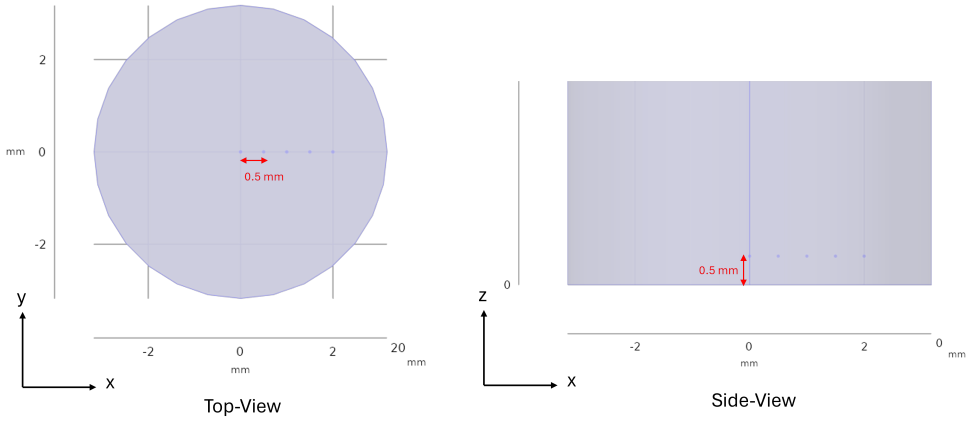


Figure 54: location of injected massless particles at t=0 to study the spin-up phase

To obtain a visual representation of the location of the particles during the spin-up phase, one can plot the particle tracks at a time of t= 1 s (end of the spin-up) interpolated over the spin-up time frame. This is shown in Figure 55 where the colour map represents the velocity magnitude. From the side-view and the 3D view, it can be seen that the particle tracks reach various heights based on their initial radial position. As mentioned earlier, these particle tracks are essentially streamlines or tracers suggesting a path the particles may take depending on their properties and the boundary conditions. The particle tracks at the centreline falls vertically downward and stops as it hits the wall. This is due to the downward jet produced by the secondary flow in these time intervals (see white arrows showing the secondary flow in Figure 39b). Particles further away from the centreline also reduce in height initially for the same reason before rising to reach their final state at the end of spin-up. The top view shows that particles are pushed from the centreline towards the cylinder walls, and this is due to the centrifugal forces experienced by the fluid, resulting

in these particles tracers. To get a better idea of the changes in height these particle tracers experience, one can look at the particle position plots against time. Figure 56 shows the vertical and radial positions of the particles with time. From the radial position plot, it is clear that all the particles move towards the outside walls immediately as spin-up begins with the exception of the particle at the centreline. This is important to keep in mind because it explains why there is a change in particle height. It should also be noted that particles have a higher probability of being located near the outside walls of the cylinder during spin-up, based on this outward movement experienced by all particles. The particles gain height near the outer walls because of the upward region of the flow explained in the secondary flow explanation of spin-up (see white arrows showing secondary flow in Figure 39b).

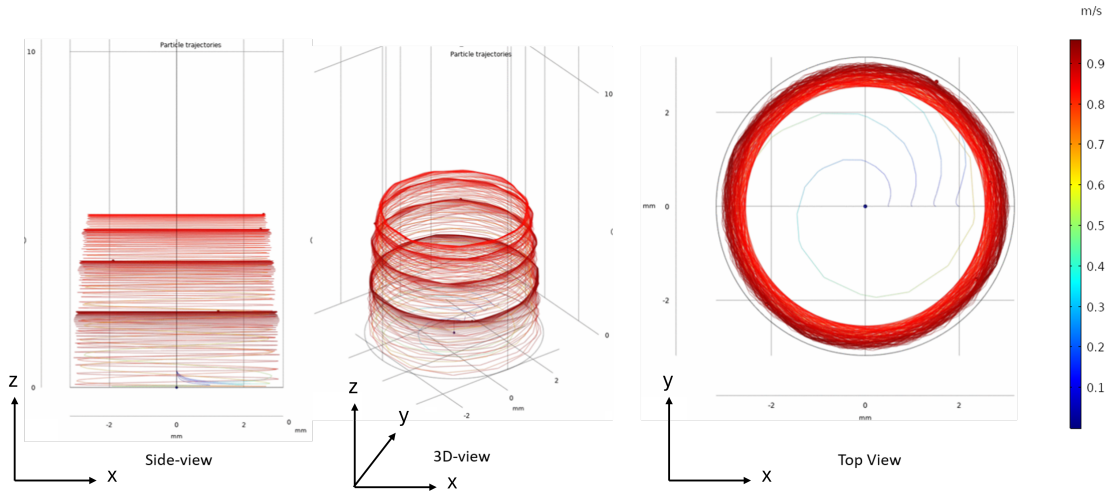


Figure 55: Various viewpoints and of particle tracks at t=1 s (end of spin-up: 3000 RPM $Re \approx 1090$)

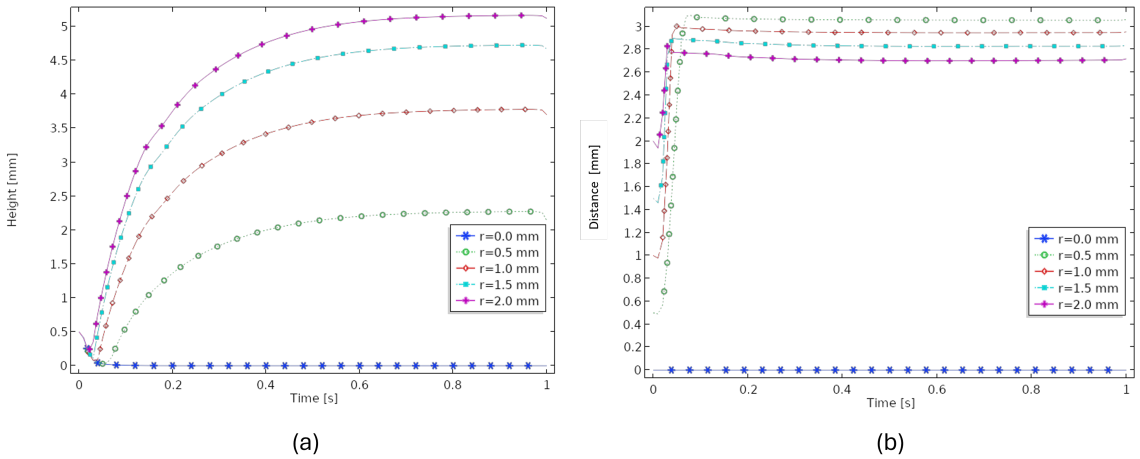


Figure 56: (a) Vertical position, (b) radial position, of particles against time during spin-up at 3000 RPM ($Re \approx 1090$)

5.1.2 Spin-up: Impact of Initial Radial Seeding

In this study, the impact of the initial radial seeding on the final position of a particle (radial and height) during spin-up for a range of Reynold numbers is examined. By doing so, one can get a better idea of the general trend radial seeding has for the motion of particles. The particles are released at a particular height of 1.5 mm and range from $R = 0$ mm to $R = 3.1$ mm in increments of 0.02 mm. In doing so, 150 particles are seeded. Figure 57 shows the top and side view of the 150 seeded particles. The aim of this study is to understand how the initial radial particle position changes the final height and radial position. This study is very similar to the previous study with five particles as particles are released at varying radial distances, but it analyses a large dataset to get a better look at the general trend within the domain.

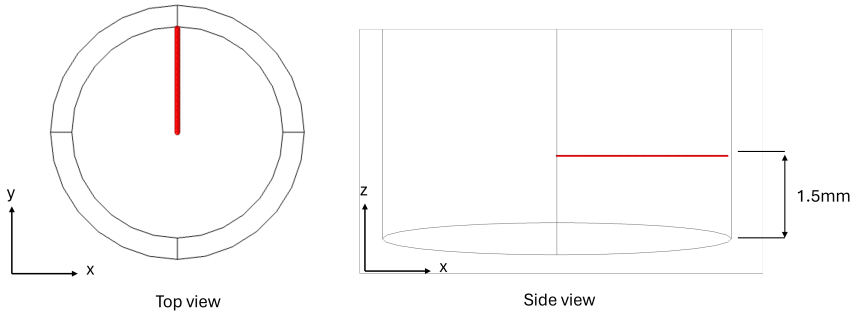
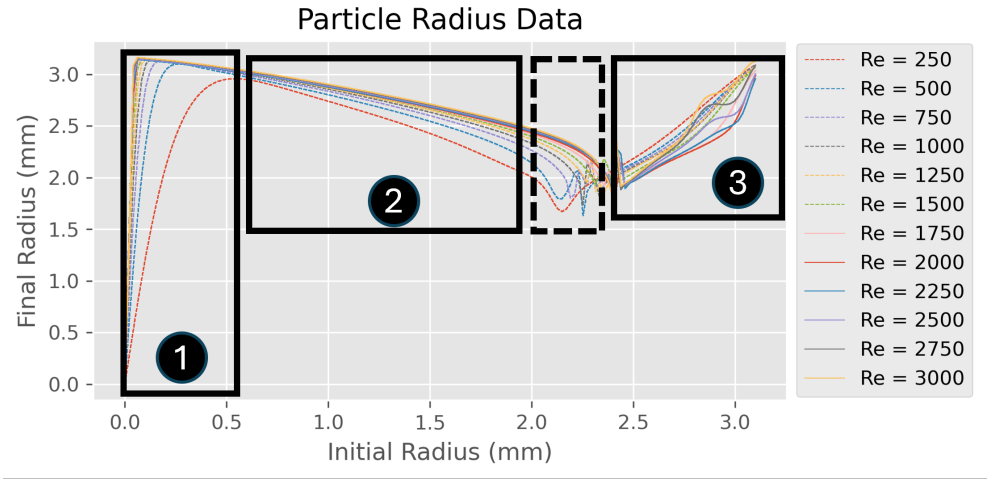


Figure 57: Top and side view of the location of 150 massless particles released along a radial line

First, consider the impact that the radial position has on the radial position of the particle at the end of the spin-up cycle $R_{t=0.95s}$.For this study, the time at the end of the spin-up is taken as 0.95 s to avoid the effects of the spin-down. Figure 58 shows the initial particle radial position $R_{t=0s}$ and against the final particle radial position $R_{t=0.95s}$ for a range of Reynold's numbers for all particles. Similarly for the same data, Figure 59 shows the change in the radial position at t=0.95 s from the initial radial position at $t = 0s$.

In Figure 58, the "black boxed region 1" shows that the particles shoot drastically outward for the initial $R < 0.5$ mm. This is due to centrifugal flow as the cylinder rotates. This is exaggerated for higher Reynold numbers because of stronger centrifugal flows. As this is identical to the previous study (Section 5.1.1), it is known that this radial movement occurs very early during spin-up. For $0.5\,\text{mm} < R < 2\,\text{mm}$, the "black boxed region 2" shows there is a slight increase in radial position. This is less drastic than for the $R < 0.5$ mm particles. There is a transitional region between $2\,\text{mm} < R < 2.5\,\text{mm}$ followed by a reversal of the direction from outward to inward at $2.5\,\text{mm} < R$ highlighted by the "black boxed region 3". This is likely due to the secondary flows observed near the outer wall of the cylinder (see white arrows showing the secondary flow in Figure 39b). Overall, the trend is the same across all Reynold's numbers, where the momentum of the fluid varies. Figure 59 shows change in radial position from the initial seeding radial position. Region 1 shows an increase in the radial position, while region 2 shows a decrease in the radial position from 2.5 mm to 3.175 mm.

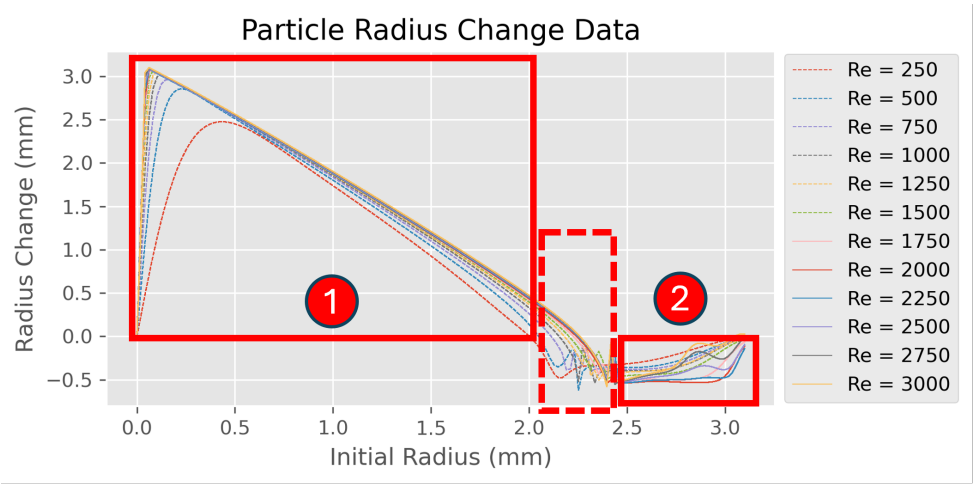


1 Greatest increase in radial position: $0 < R_0\ (mm) < 0.5$

2 Significant increase in radial position: $0.5 < R_0\ (mm) < 2$

3 Minor change in radial position: $2.5 < R_0\ (mm) < 3.175$

Figure 58: Initial particle radial position $R_{t=0s}$ against the particle radial position $R_{t=0.95s}$ for a range of Reynold's numbers for all 150 particles at a height of 1.5 mm during spin-up



1 Increase in radial position: $0 < R_0\ (mm) < 2$

2 Decrease in radial position: $2.5 < R_0\ (mm) < 3.175$

Figure 59: Change in the particle radial position at $t = 3s$ from the initial particle radial position at $t = 1s$ for 150 particles at a height of 1.5 mm for various Reynold's numbers during spin-up

Now lets consider the final height change a particle achieves based on its initial radial position. This is shown in Figure 60 for a range of relevant Reynolds numbers. The figure is divided into 3 different regions. The "Red Boxed Region 1" shows the particles near that are radially near the centreline, 0 mm is 0.5 mm. These particles show a reduction in height. This is expected for particles near the centreline as a result of the downward jet during the early stages of spin-up. "Red Boxed Region 2" shows an increase in height from the initial height of 1.5 mm to a maximum peak of 6 mm between $0.5\,\text{mm} < R < 2.0\,\text{mm}$. The increase in final height is greater for large Reynolds numbers. This region is followed by a transitional region between $2\,\text{mm} < R < 2.5\,\text{mm}$, shown by the dashed red box. Finally, particles seeded near the outside walls at $2.5\,\text{mm} < R < 3.1\,\text{mm}$ highlighted by the "Red boxed region 3", show the largest increase in height of around 10 mm. This largest change is expected as a result of the secondary flow that exhibits vertical motion near the cylinder sidewalls during spin-up (as shown by white arrows in Figure 39b). This shows that particles at the end of the spin-up phase are likely to move towards the outside walls and generally are lifted by the secondary flows with the exception of the region near the centreline and particles in the transition region.

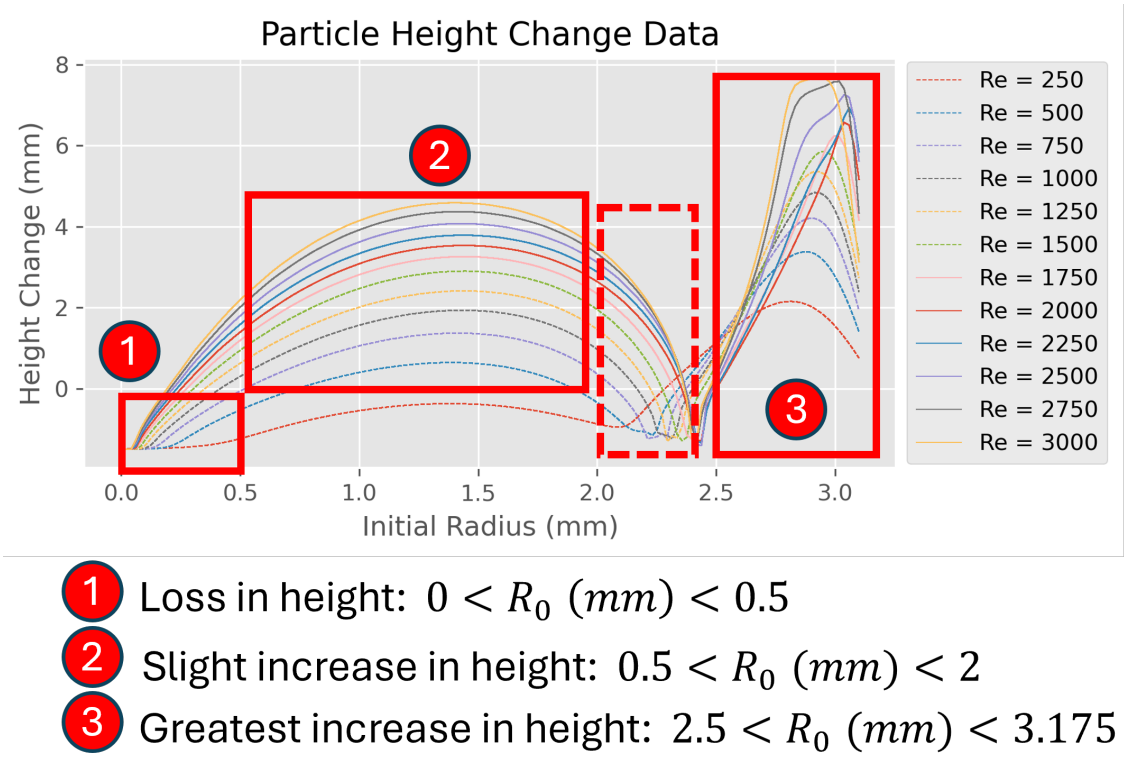


Figure 60: Change in the final height position at $t = 3s$ from an initial radial seed at a height of 1.5 mm at $t = 1s$ for 150 particles at a height of 1.5 mm for various Reynold's numbers after spin-down begins

5.1.3 Spin-up: Massless Particle Quarter Grid Seeding

In this study, a grid of 64 x 64 massless particles released at the bottom right quadrant of the cylinder with the aim of analysing the particle redistribution during spin-up. Figure 61 shows this initial release position for the Reynolds numbers of 250 and 3000. The aim is to analyse how a concentrated grid of particles redistributes itself spatially during spin-up. This will enable the identification of regions exhibiting high particle transport. The results are taken up to a time of t=0.95 s. This allow the areas of greater interest to be determined within the domain to look for particle motion during the spin-up phase of a real-life inspection.

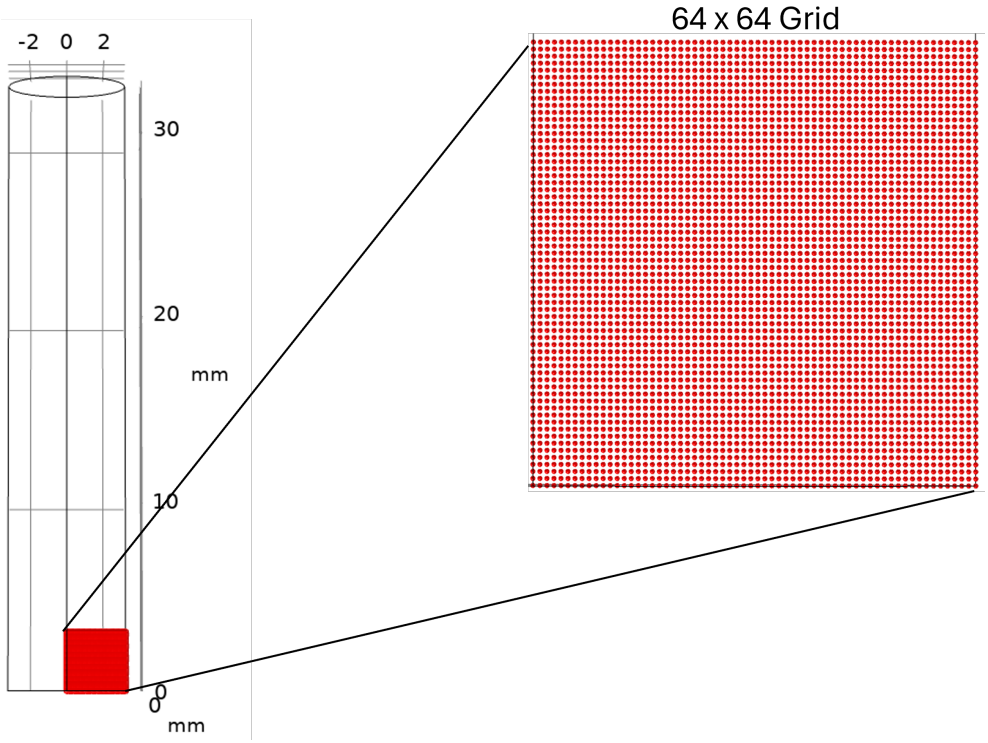


Figure 61: Initial release location of a 64 x 64 grid of massless particles seeded at the bottom right quadrant of the cylinder for Reynolds number of 250 and 3000.

Figure 62 shows the final result for the Reynolds number of 250 where the green dots are the starting position of the particles grid and the red dots highlighted by the black border area show the movement of the initial grid at $t = 0.95\,\mathrm{s}$. Overall, it is observed that during spin-up, the particles migrate towards the outside walls and gain height forming a hill-shaped region. This reflects the influence of the secondary flow structures in which motion is vertically upwards at the bottom right side of the domain due to recirculating vorticities (see Figure 39b for clarification). For comparison, the same analysis is conducted for the Reynolds number of 3000 shown in Figure 63. The particle dispersal is much more pronounced and tightly packed. Interestingly, an area where no particles (a void) is present within the bounds of the black boarder that is not seen with the lower Reynolds number. This suggests particles are pushed out of this region at higher Reynolds numbers during the spin-up duration. A sharper particle distribution region is due to the stronger centrifugal forces associated with higher Reynolds numbers.

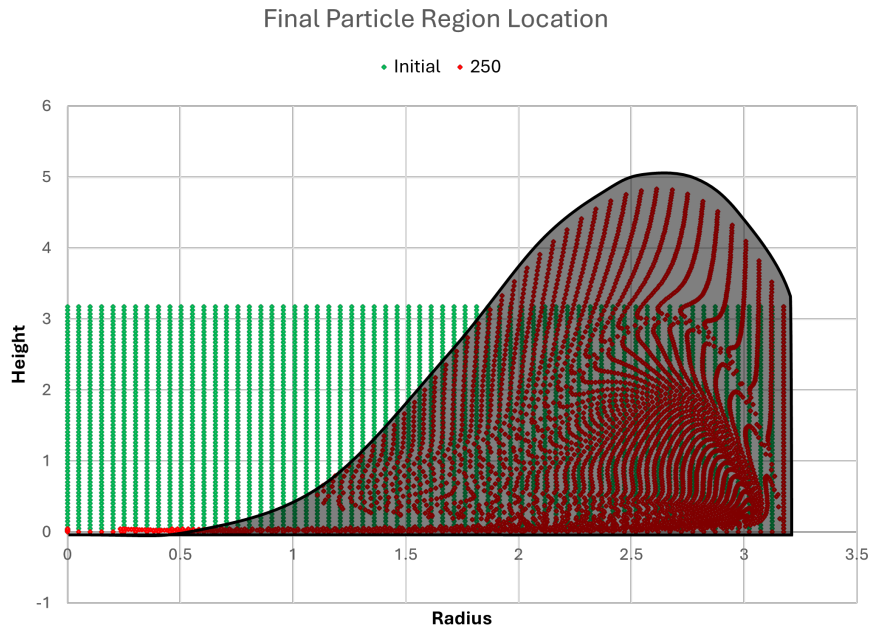


Figure 62: Final position of particles (red) at t=0.95 s for Re=250. Green points indicate the initial location and the grey region highlighting the new displacement area.

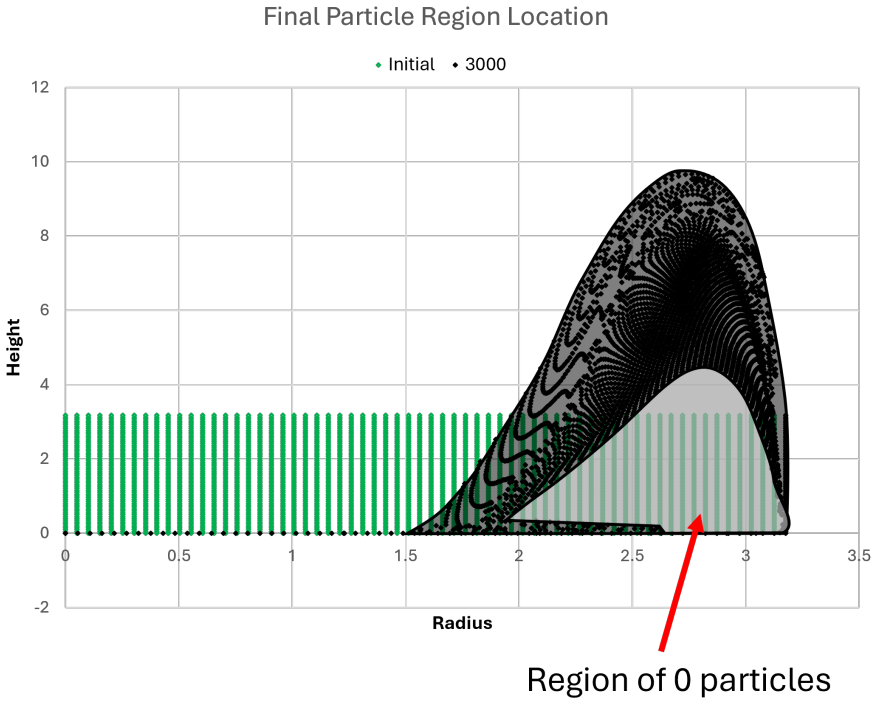


Figure 63: Final position of particles (red) at t=0.95 s for Re=3000. Green points indicate the initial location and the grey region highlighting the new displacement area.

By colour tracking and adding other grids up to the centre of the cylinder for the 250 Reynolds number case, the initial position of the particle grid highlighted in Figure 64 is obtained. The resulting motion of particles is also shown in the figure for the final position of the total grid. It is evident that particles in different vertical layers experience different intensities of the fluid effects that cause them to displace. The highest affected regions are closer to the bottom of the cylinder with grid 1 & 2 and less pronounced motion in grid 5. For inspection, this means that particles near the bottom of the syringe during spin-up are most likely to be significantly transported in comparison to particles floating near the centre of the syringe at these instances of spin-up.

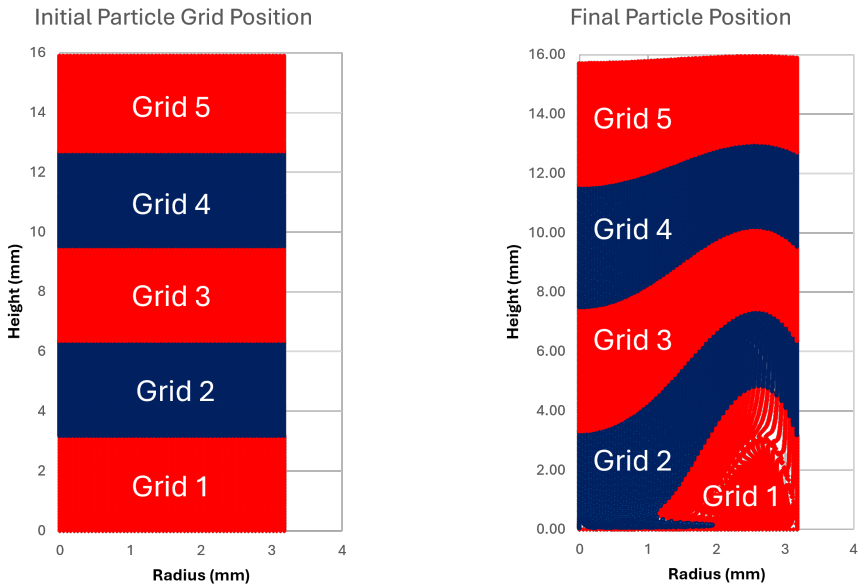


Figure 64: Initial seeding grid (left) and final particle positions at t=0.95 s (right) of a 64 x 64 grid of particles released for Reynolds number 250 (≈ 686 RPM where red and blue colours are used to represent alternating grids.

5.1.4 Spin-down: Impact of Initial Radial Seeding (1000 RPM or Re ≈ 364)

To analyse the effect of radial distance, massless particles are placed at varying distances from the centreline at heights of 1 mm and 4 mm respectively (shown in Figure 65). In this study, typical particle tracks are determined for particles released at equal arbitrary radial distances from each other. For comparison, this is done at heights of 1 mm and 4 mm near the bottom of the syringe to see the effect the lower jet has on particle movement. In a real inspection, cameras have a particular inspection window (for time) when they are activated and monitor movement. This is after spin-down begins. Therefore, in this study, only spin-down is considered. The initial seeding of the particles shown in Figure 65 is released at a time of t=1 s where the spin speed is 1000 RPM reflecting particles at these particular locations at this time. The initial seeding for spin-down are not the same particles examined at the end of spin up.

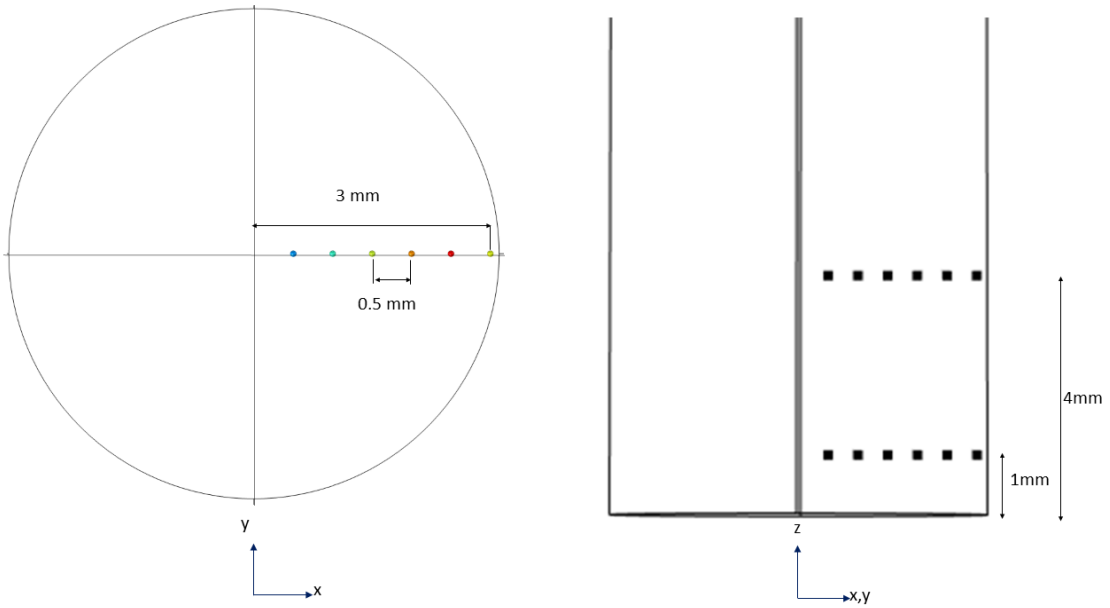


Figure 65: Injection of particles at spin-down (t=1 s) placed radially from the centreline at two given heights.

The resulting particle tracks are seen in Figure 66 represent the streamlines of massless particles up to a time of t=3 s (2 s after spin-down begins). The figure shows a 3D view and a 2D side view which would typically be seen by a camera in a real inspection. The scale represents the velocity magnitude of the particles and is shown by the colour along the trajectory lines. In the case of 1000 RPM, it is found that all particles are completely stationary by t = 3 s, as shown by the dark blue dot at the end of the trajectory. None of the particles has completed a full revolution. More details on the position of the particle against time can be found in Figure 67 for particles released at heights 1 mm and 4 mm. The final height observed for each released particle, $Z_{t=3s}$, is shown in Table 4. Particles closer to the centreline gain more height than those closer to the outside walls of the cylinder. This is because particles closer to the centreline are within the range where the axial jet is propelled from the bottom of the syringe. In contrast, particles placed further than 1.5 mm from the centreline are found to lose height instead of gaining it. This is due to the secondary flow discussed for the axial flow during spin down, where the flow near the outside walls is reversed from the jet direction. This flow feature causes the particles to loose height as the particles get closer to the outside walls (see white arrows in Figure 39d). The dividing region between the upward and downward motion for the 1000 RPM case lies between 2.0 mm and 2.5 mm. An interesting result is shown by particles 5 and 11 in Table 4 (particles in this region) show that the height is lower than the particles placed after them.

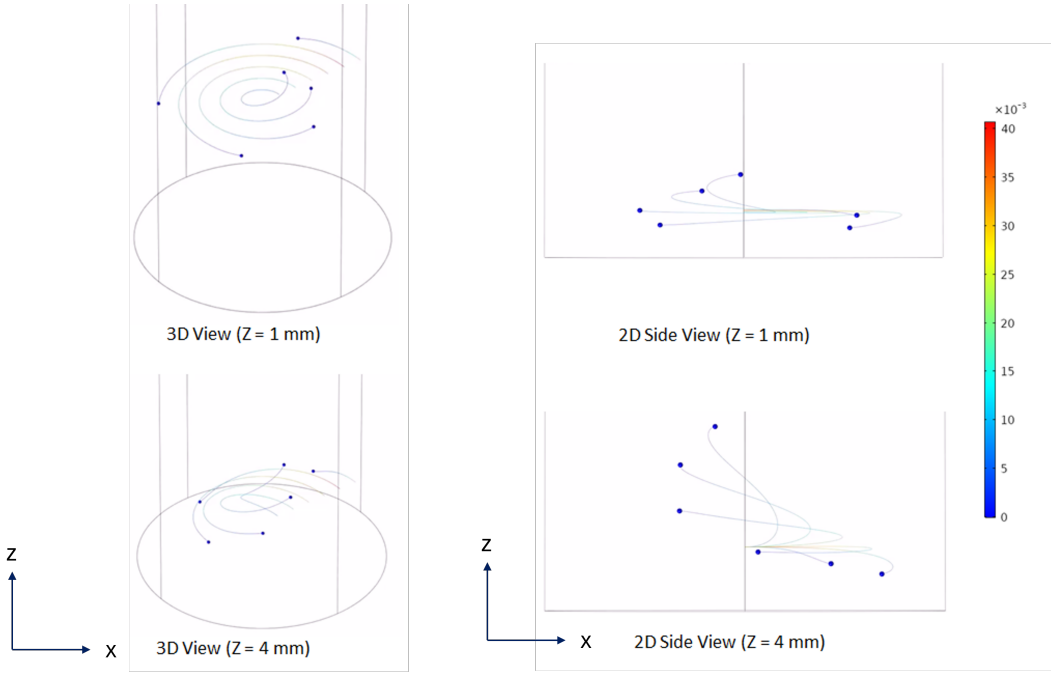


Figure 66: 2D (side-view) and 3D view of particle tracks of two different heights at 1000 RPM, where colour represents velocity magnitude in $\mathrm{m\,s^{-1}}$.

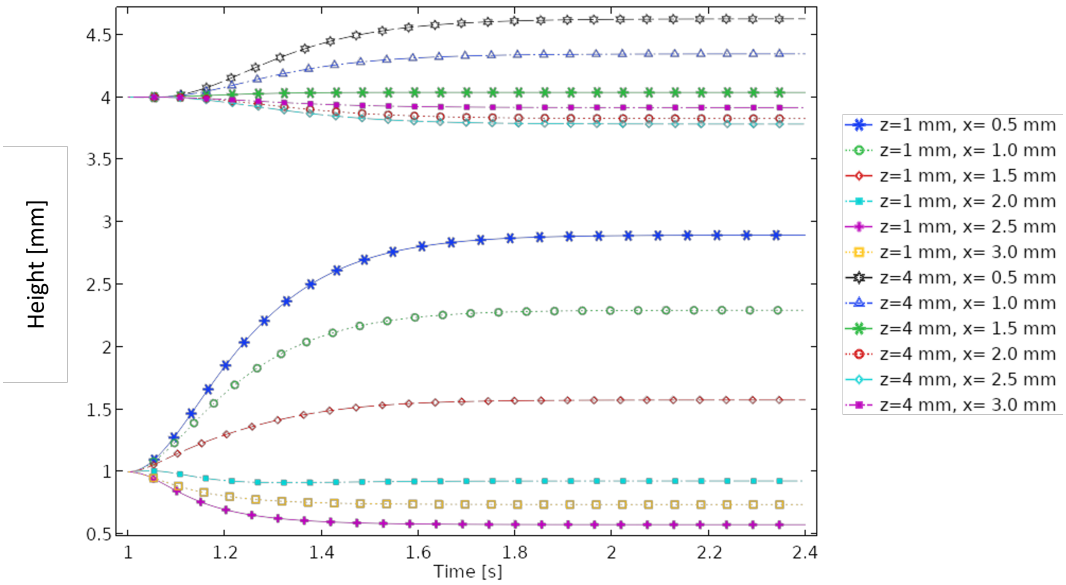


Figure 67: Particle height against time for two given initial heights, $Z_{t=1s}$, at 1000 RPM

Table 4: Particle position and maximum height increase, $Z_{t=3s}$ for two given initial heights, $Z_{t=1s}$ at 1000 RPM.

Particle	$Z_{t=1s}$ [mm]	X_0 [mm]	$Z_{t=3s}$ [mm]
1	1	0.5	2.895
2	1	1.0	2.295
3	1	1.5	1.575
4	1	2.0	0.927
5	1	2.5	0.578
6	1	3.0	0.739
7	4	0.5	4.626
8	4	1.0	4.346
9	4	1.5	4.037
10	4	2.0	3.828
11	4	2.5	3.784
12	4	3.0	3.916

5.1.5 Spin-down: Effect of increased RPM (3000 RPM)

The effect of rotation speed on particle movement is studied by increasing the RPM from 1000 to 3000. The position of the particles is the same as the setup in Figure 65 at a height of 1 mm. This height is selected to model particles that are settled near the bottom of the syringe just above the plunger. The results in Figure 68 show the particle tracks up to 5 seconds after spin-down begins. The colours represent the velocity magnitude of the particles and show that all particles are stationary at 5 seconds. Compared to Figure 66, the main effect of increased RPM is the number of revolutions on the particle track. At 1000 RPM, half of a revolution is observed, where the 3000 RPM track shows approximately 2.5 revolutions. Moreover, Figure 69 shows the height of particles with time. It can be seen that the height of the particle increased from its initial height compared to Figure 67. This is because the fluid has a much higher momentum at the start of spin-down.

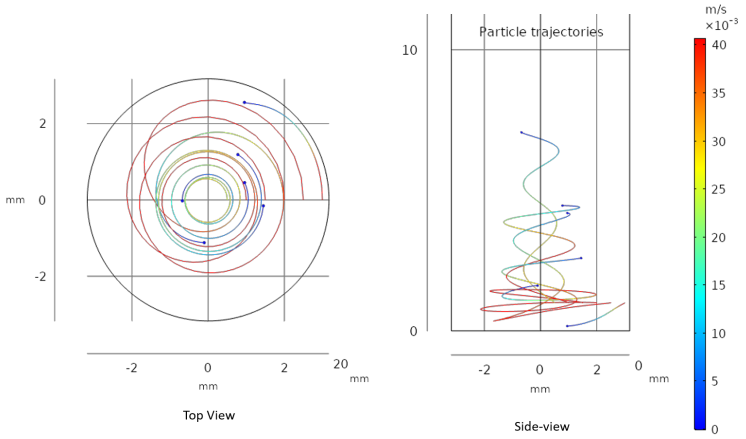


Figure 68: Particle tracks showing top-view and side-view for 6 particles released at a height of 1 mm at 3000 RPM at the end of spin-down

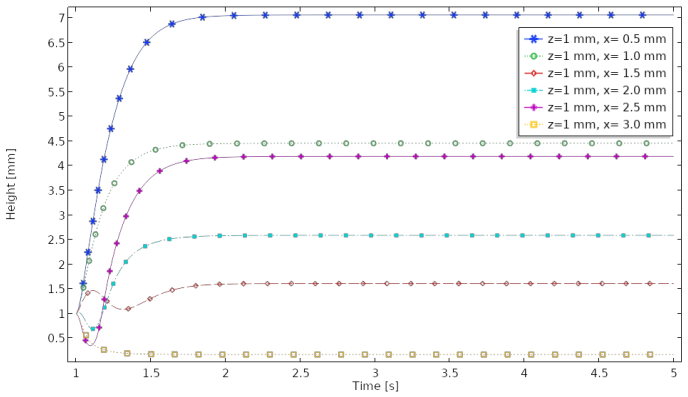


Figure 69: Particle Height against time for 6 particles released at a height of 1 mm at 3000 RPM at the end of spin-down

5.1.6 Spin-down: Impact of Initial Height Seeding (1000 RPM)

In this next study, consider the effect of the vertical particle position during spin-down and how this changes as the line of initial vertical seeding is moved radially through the domain. To achieve this, a line of massless tracer particles is placed starting at a height of $Z = 0.1\,\text{mm}$ along the centreline. The next particles are then placed in increments of $0.01\,\text{mm}$ (ΔZ_0) up to a height of $16.5\,\text{mm}$ (half the height of the cylinder). All particles are released at $t = 1s$, when spin-down begins after a spin of 1000 RPM. The purpose of the study is to determine how the maximum height to which the particles reach varies as the initial height and the radial position change. Figure 70 shows the placement of the particles.

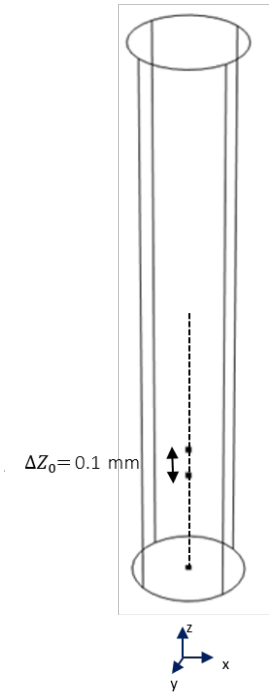


Figure 70: Injection of particles at spin-down (t=1 s) placed vertically along the centreline at a height of 0.1 mm and incremental increase of also 0.1 mm.

Along the centreline, Figure 71 shows the maximum height change achieved by each particle based on its initial height at the start of the spin down. The results show a noticeable increase in the change in particle height between the initial heights of 0.1 mm and 2 mm. The maximum height gained is also found to be 2.18 mm and this occurs at an initial height of 0.75 mm. After this point, as the initial height increases, the height gained by particles gradually decreases. This is to be expected because particles seeded too high may miss the most intense region of the upward axial jet which is formed during early spin-down (See the formation of the jet in Figure 45). Particles caught by the jet during this early time frame are in the optimal position to be displaced by the jet for the longest duration, resulting in the largest vertical displacement.

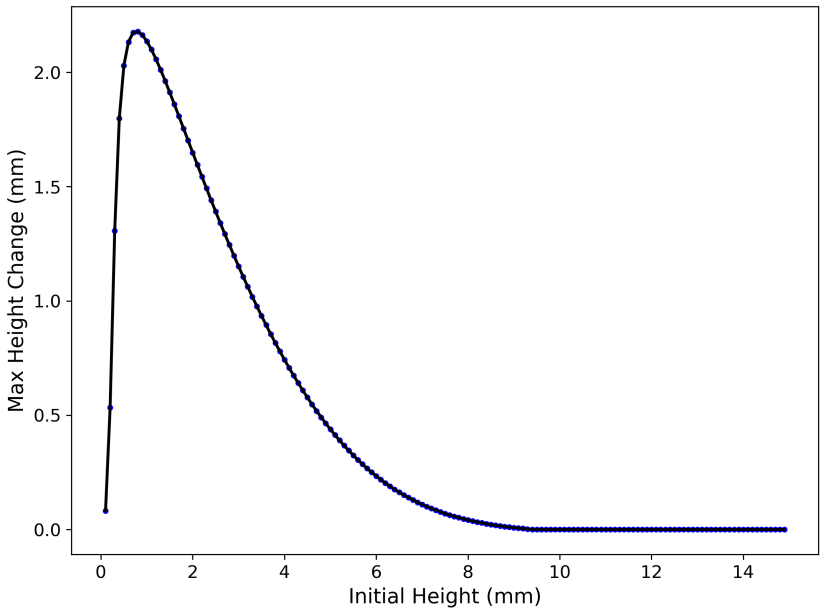


Figure 71: Maximum height change as a function of initial vertical position at the centreline (r=0 mm) showing a clear peak at approximately $z = 0.75$ mm for a speed of 1000 RPM

To assess how the radial location influences vertical transport, the study is extended to particles released at various radial locations. Figure 72 shows the trend of the change in maximum height with the initial height as one moves away from the centreline. The maximum height change at the peak is seen at the centreline and occurs below 2 mm for this geometry at 1000 RPM. The reduction in maximum height is associated with the position of the axial jet that acts closer towards the centreline than areas near the outside walls (as shown in Figure 45. The general trend observed across each vertical placement of particles is the same, where a peak is reached closer the lower initial heights. This is expected as the jet develops at the lower heights immediately after spin-down begins. The particles placed at higher initial heights also gain height, but are less effected by these jets. For all particles injected, no height change is observed after 9 mm. This is because the fluid near the centre heights of the cylinder is relatively stationary.

5.1.7 Spin-down: Geometry Study (Conical Plunger)

In this study, the impact of the syringe geometry of a particular particle track during spin-down is analysed. The aim is to observe how the presence of a plunger can alter the fluid behaviour, leading to any changes a massless tracer particle may have on its trajectory and height displacement. Two cases are analysed under the same initial conditions, where one has a conical plunger and the other does not. The plunger is a conical installation at the bottom of the cylinder with a cone height of 1.75 mm (typical plunger dimensions). Figure 73 shows a schematic of the conical geometry. The spin speed is set to 1000 RPM and the particle is released with a height of 3 mm and a radial position of 1 mm. The 1000 RPM is selected as it ensures that no chaotic flow is present within the flow.

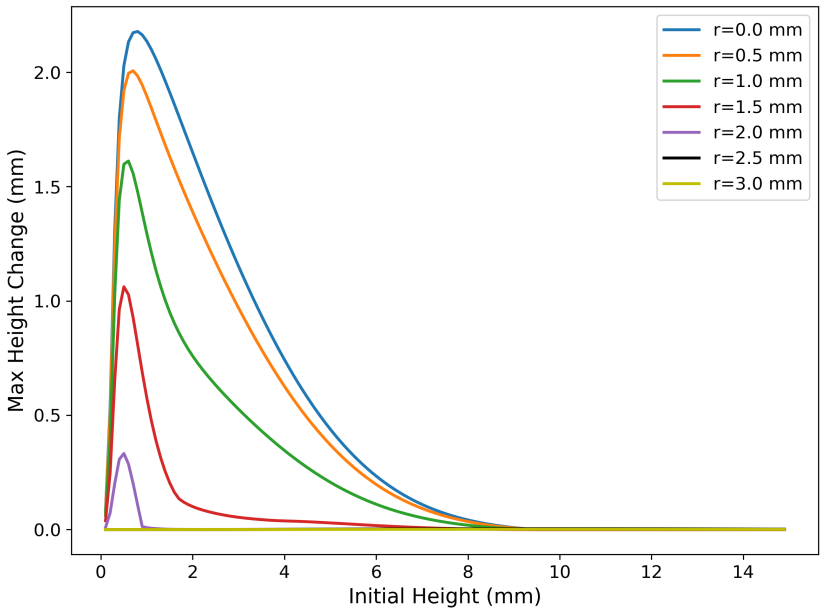


Figure 72: Maximum height change against initial height at various radial distances within the cylinder

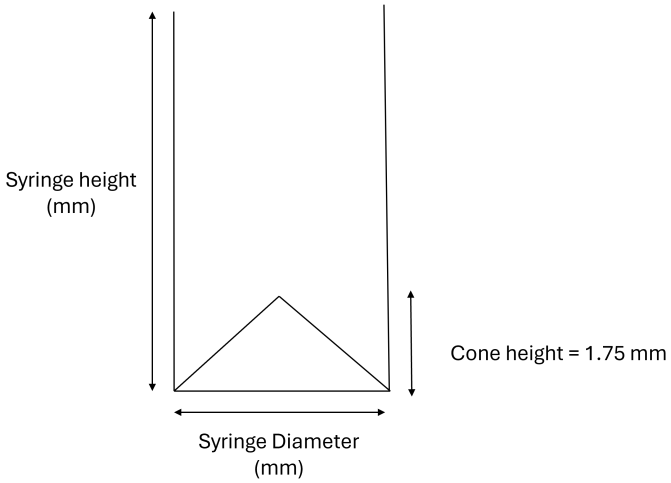


Figure 73: Schematic of the gemetry used for the plunger model with a conical base of height 1.75 mm

Figure 74 shows the 3D track(left) and the height position against time (right) from the start of spin-down to 5 seconds. Figure 75 shows the same for the plunger case. In the absence of the plunger, the resulting track is tight and helical with the particle lifting up to a height of 12 mm. Adding a plunger shows that initially the tight helical track is present. However, after a height peak at 8 mm, the particle starts to lose height and settles at 6 mm. With the same initial conditions, no gravitational or inertial effects, just the presence of the plunger reduces the maximum height. This suggests that the formation of the jet during spin-down is altered, reducing its strength in the presence of a plunger.

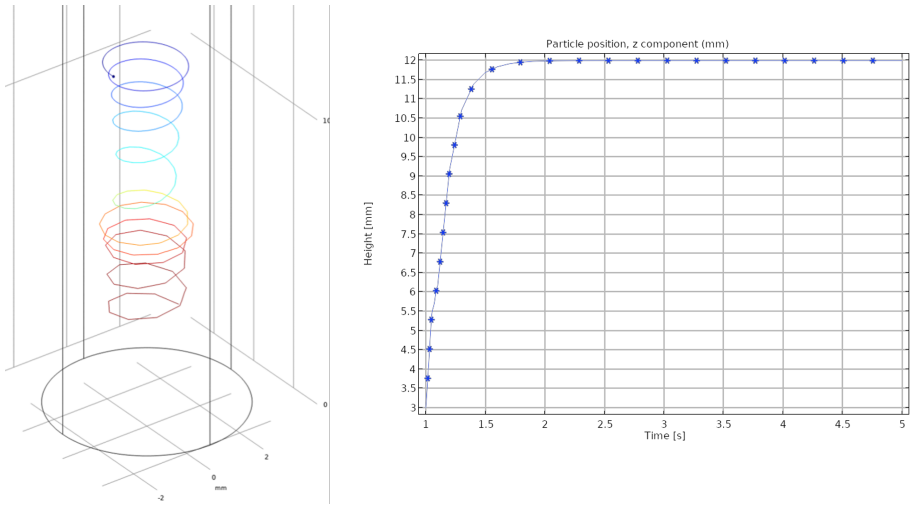


Figure 74: 3D track(left) and the height position against time (right) from the start of spin-down to 5 seconds for the no-plunger geometry at 1000 RPM for a particle released with a height of 3 mm and a radial position of 1 mm

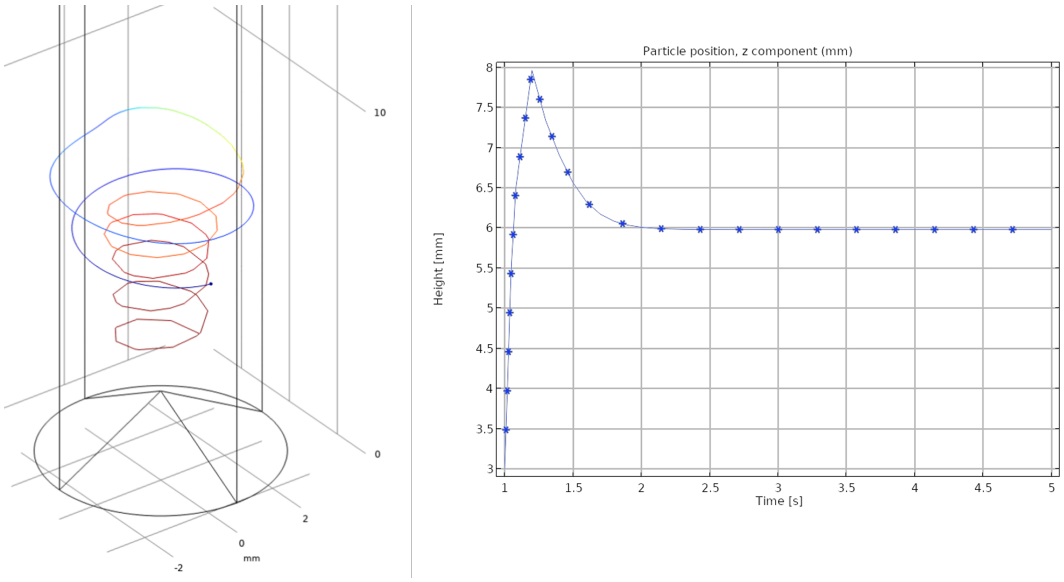


Figure 75: 3D track(left) and the height position against time (right) from the start of spin-down to 5 seconds for the plunger geometry at 1000 RPM for a particle released with a height of 3 mm and a radial position of 1 mm

5.2 Full Cycle Heatmaps (massless particles)

This section aims is to understand the behaviour of particles under two critical parameters. These are Reynolds number (changed by spin speed) and aspect ratio (changed by varying height of domain). For each study, two different grids are used to seed the particles. For the aspect ratio study (Section 5.2.1, 100 particles are seeded through a grid release of 5 x 20 massless particles. For the Reynolds number study (Section 5.2.2), 500 massless particles are seeded using a 10 x 50 particle grid. Both grids cover the lower right quadrant of the cylinder. This initial seeding is shown in Figure 76 for the aspect ratio study seeding and Figure 77 for the Reynolds number study seeding. The particles are seeded at $t = 0s$ and are influenced by both spin-up and spin-down. The behaviour of the particles from $t = 1.1s$ to $t = 1.5s$ is analysed. This resembles a typical inspection window for industrial practice. The massless analysis is used initially, and this is likely representative of particles with mass, as these can be used as tracer particles to see the impact of the fluid on a particle at a given location. Mass studies are conducted in Section 5.3. The main executed change with the inclusion of particles with mass is the effect of gravity at the end of the particle's trajectory causing the particle to sink.

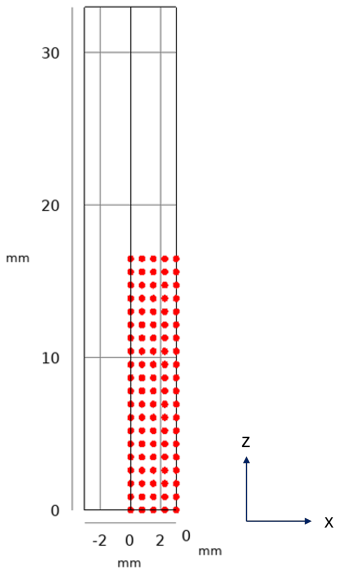


Figure 76: Initial seeding location of a grid of 100 particles (5 x 20) covering the bottom right quadrant of the cylinder for the aspect ratio study

The radial position and height of the particles are recorded every 0.01 s from t = 0 to t = 1.5. Data for the time between $t = 1.1s$ and $t = 1.5s$ are processed at equal time intervals for the two studies. Initially, an aspect ratio of 5.5 and Reynolds number of 500 is discussed for both studies as this is the case that resembles a typical prefilled syringe. Other aspect ratios starting at 0.5 increasing in increments of 0.5 up to 5.5 are later compared.

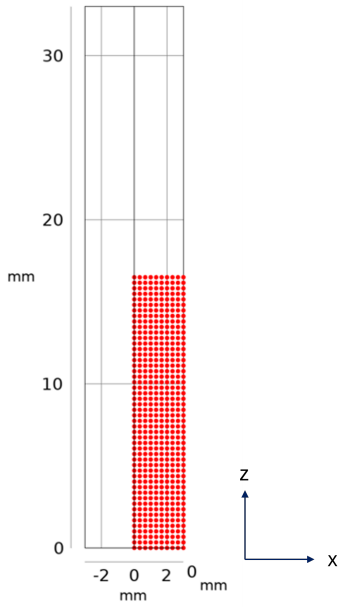


Figure 77: Initial seeding location of a grid of 500 particles (10 x 50) covering the bottom right quadrant of the cylinder for the Reynolds number study

5.2.1 Aspect Ratio Study

The aspect ratio study is conducted with a fixed Reynolds number of 500 ($\approx$ 1372 RPM). Figure 78 shows a scatter plot in the radial-axial plane for all particles during the inspection window for an aspect ratio of 5.5 during spin-down. Note that this plot shows a 2D projection of the position and ignores its angular component. Hence, providing a flattened 2D view. Using such a plot is still very useful, as it simplifies the interpretation of motion but removes the rotational complexity. From the plot, a comprehensive visualisation of particle motion patterns overtime within the cylinder is captured during spin-down. Each dot represents the radial and vertical location of the particle at a certain time within the inspection window. By producing this plot, a general idea of the global trajectory of particles through curved arcs and various densities showing cluster zones can be obtained.

The length of the cluster zones shows where particles might be slowed, trapped, or redirected by the fluid motion. The visible arches and radial spread reveal how vortex structures affect the motion of particles. The particle instances near the outer walls of the cylinder show smaller clusters through particles that overlap each other within the inspection window. This suggests that the particles experience very little motion during the inspection window at this location. The particles released at $z = 0$ mm and in the centre of the cylinder ($z = 16.5$ mm) only show radial motion. No significant change is seen in the height at these locations. For the particular case of AR= 5.5, symmetric arches near $r = 1.5$ mm show that the particles experience repeated upward and downward jet structures. At the centreline, larger heights show more clustering and lower heights show streaks of recorded data points suggesting more movement within the same time frame. This shows that the red jet as shown in Figure 45 influences the lower regions and does not reach the higher. Finally, the wide vertical extent

for $r < 1.5\,\text{mm}$ and $z < 7.5\,\text{mm}$ suggests that the particles move the fastest in these initial seeds, suggesting a strong axial movement.

Figure 79 shows the same data in various aspect ratios. Starting with the lower aspect ratio of 0.5, the general trend of the clockwise motion of the particle instances is pretty clear. There are clear arches formed by particles detected in given time frames. As the aspect ratio increases, the general clockwise motion is stretched in the vertical direction.

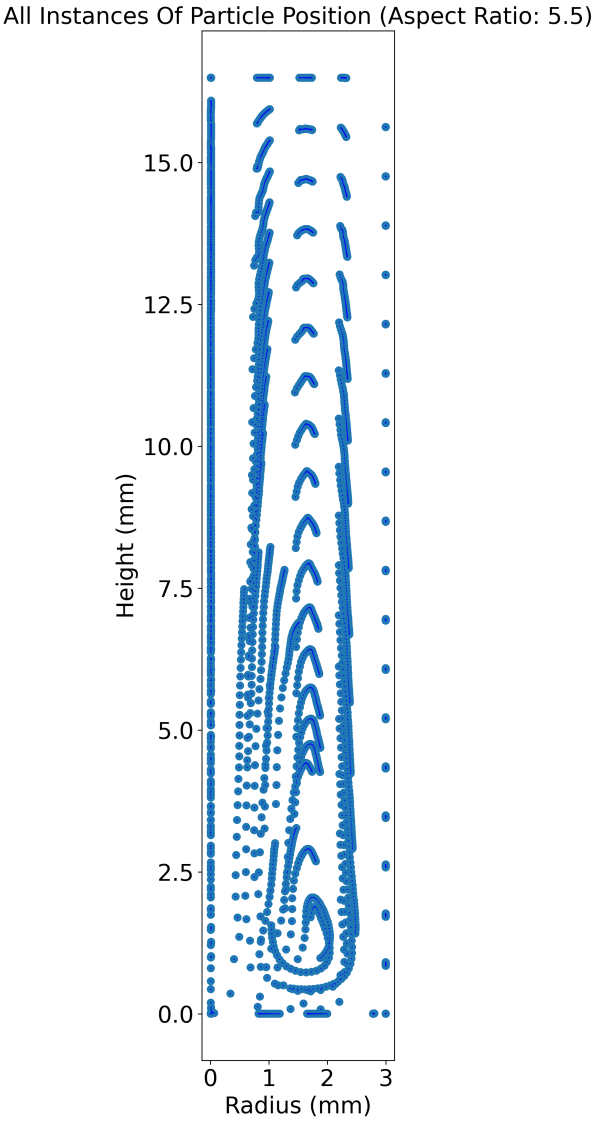


Figure 78: Scatter plot showing all recorded radial particle positions and heights between t = 1.1 s and t = 1.5 s for Aspect ratio = 5.5 at Re=500 ($\approx 1372\,\text{RPM}$)

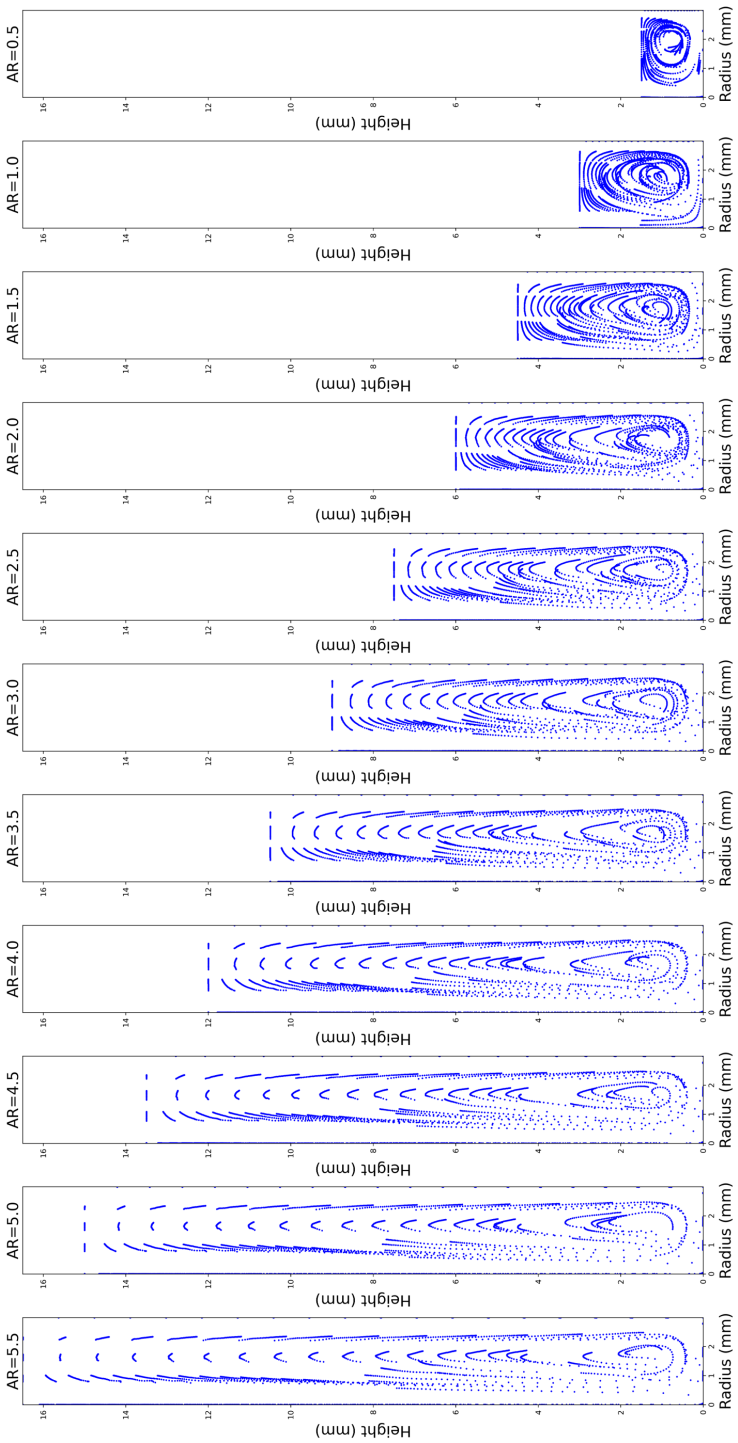


Figure 79: Scatter plot showing all recorded radial particle positions and heights between t = 1.1 s and t = 1.5 s for various aspect ratio at Re=500 ($\approx$ 1372 RPM)

Now, lets consider the movement of the particles from the time the inspection window begins to its end. This is essentially the individual displacement vector of each particle during the inspection window. Figure 80 shows this through blue and orange dots representing the initial and final positions of the particles respectively. The green arrows represent the displacement vector for the inspection window. The numbers (blue and orange) on the dots represent the particle number of 100. For an aspect ratio of 5.5 and Reynolds number of 500, particles near the centreline show small movement at high initial heights. However, for particles released closer to the bottom, large vertical motion is observed which is consistent with particles getting lifted by the jet formation. Near the outer radial positions, very little motion is detected. Most of the displacement occurs near the mid-radial positions. The particles show both an increase and a decrease in both radial positions and height. Some particles at lower heights (less than 5 mm) and mid to high initial radii (0.5 mm to 0.5 mm) show a reduction in radial position overtime. This is expected from the radial motion of the fluid and its secondary flows during spin-down. Near the mid-radial initial position, the majority of the particles show an increase in final radial position. Particles at lower heights are more displaced in the vertical direction compared to larger initial seeds. The largest displacement vectors are seen for $1.5\,\text{mm} < r < 2.5\,\text{mm}$ and $z < 5\,\text{mm}$. In particular, these are particles 43 and 23. These show a reduction in the radial position but an increase in height. Some particles show only a reduction in radial position and virtually no change in height (particles 22, 42, 45, etc.). Overall, the results can be summarised as a general clockwise motion observed for most of the particles, and this is expected from the secondary flow discussion in Chapter 4 and the literature such as Vanyo (2015).

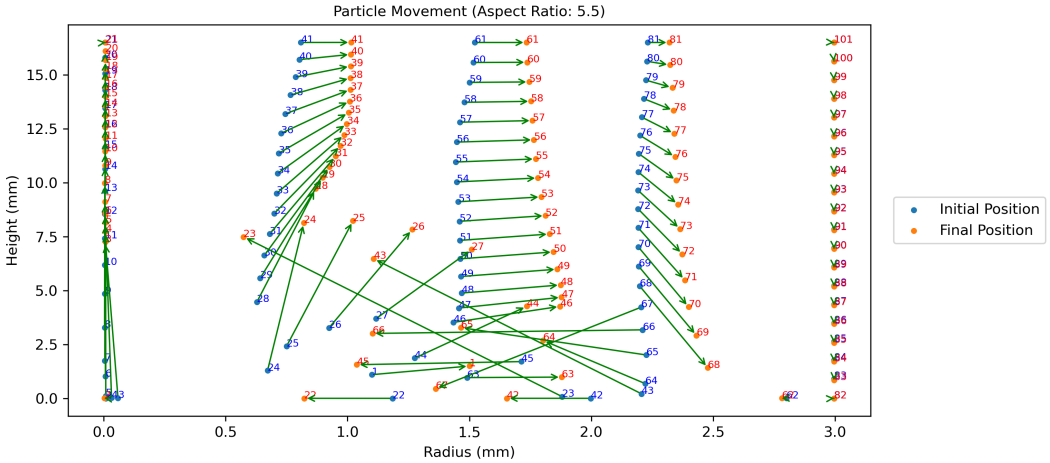


Figure 80: Displacement vector plot showing the initial and final radial position and height of particles between t = 1.1 s and t = 1.5 s for aspect ratio = 5.5 at Re=500 ($\approx$ 1372 RPM).

Figures 81 and 82 show displacement vectors for various aspect ratios (0.5 to 5.5 in increments of 0.5). It is found that as the aspect ratio increases, there is a larger vector displacement in and a highly pronounced clockwise motion. It should also be noted that the largest displacement in all aspect ratios occurs for lower initial heights. This is because the jets form from this region (as shown in Figure 45) and the particles are caught in the vertical jet the longest at these initial heights.

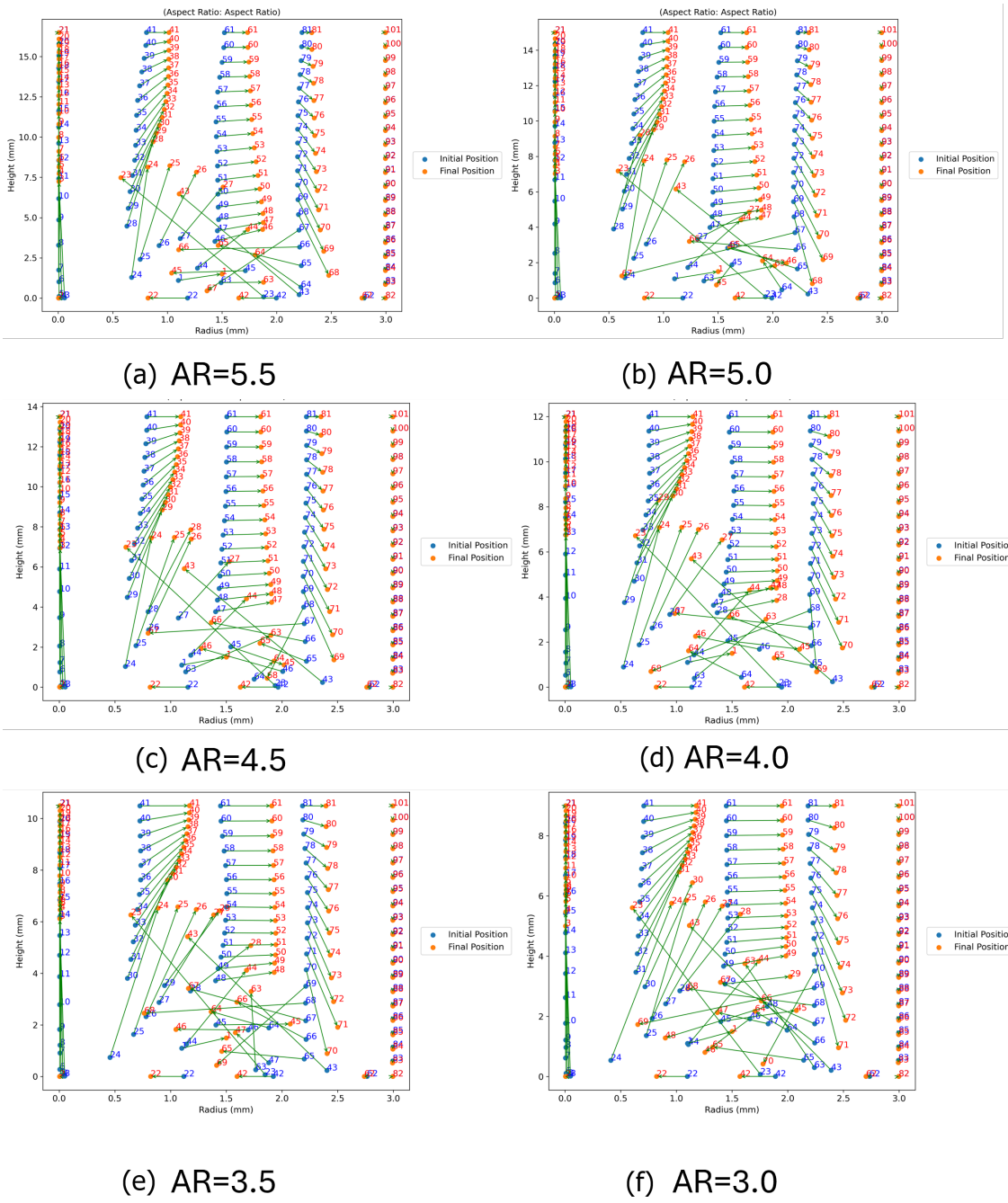


Figure 81: Displacement vector plot showing the initial and final radial position and height of particles between t = 1.1 s and t = 1.5 s for aspect ratio of 5.5, 5.0, 4,5, 4,0, 3.5, and 3.0 shown in (a), (b), (c), (d), (e) and (f) respectively at Re=500 ($\approx$ 1372 RPM).

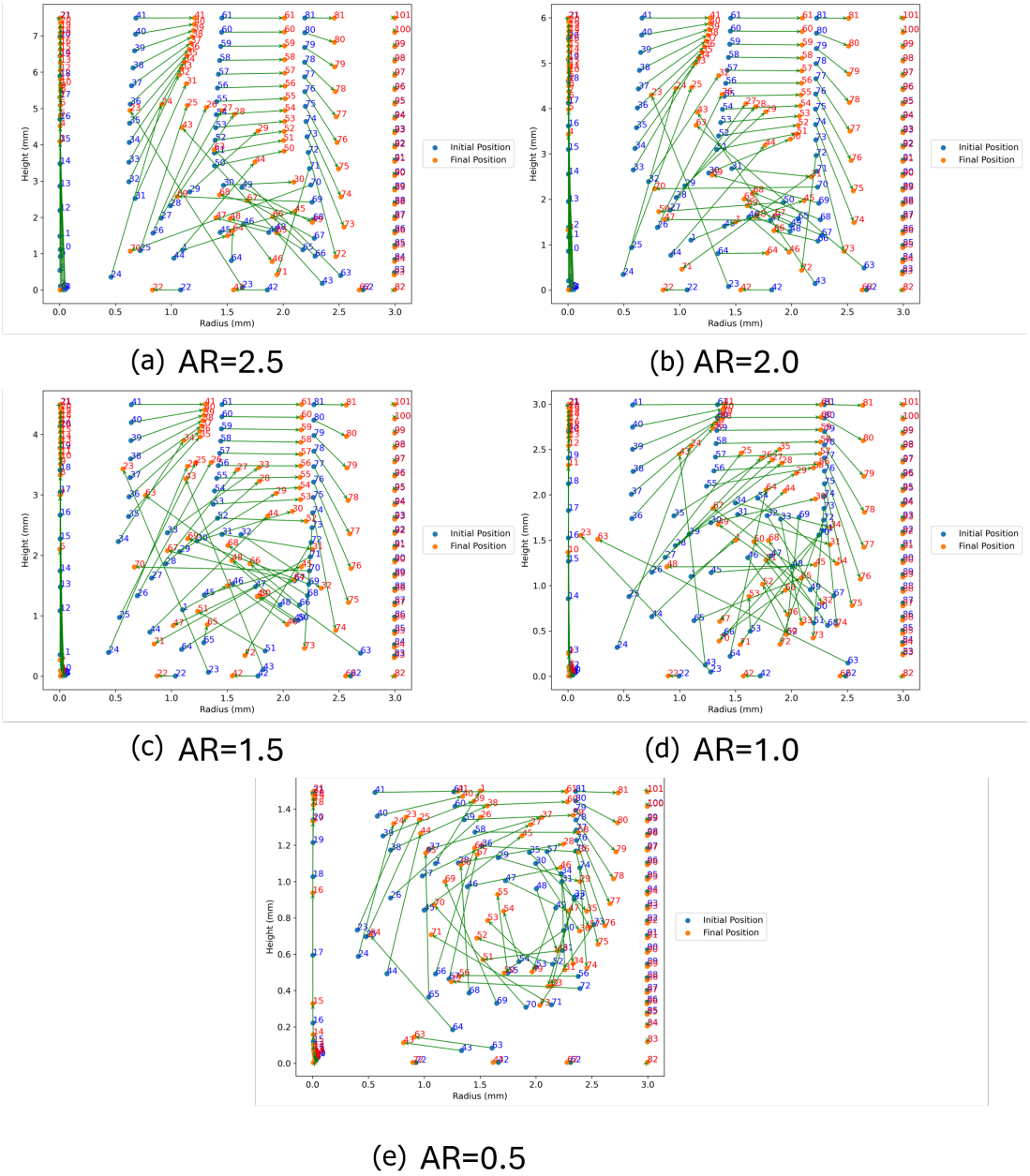


Figure 82: Displacement vector plot showing the initial and final radial position and height of particles between t = 1.1 s and t = 1.5 s for aspect ratio of 2.5, 2.0, 1,5, 1,0 and 0.5 shown in (a), (b), (c), (d) and (e) respectively at Re=500 ($\approx$ 1372 RPM).

Using the initial and final position, Figure 83 shows the calculated change in radial position and height using a box-plot (left) and a strip-plot (right). The box plot shows the statistical distribution of the net change in radial position and height between the initial and final positions of the particles during the inspection window. The median is represented by the horizontal line inside the blue and orange plots, the interquartile range is the height of the box-plots and the black diamonds represent the outliers. The outliers are defined as values satisfying

$$O > Q_3 + 1.5\,\mathrm{IQR} \quad \text{or} \quad O < Q_1 - 1.5\,\mathrm{IQR},$$

where O denotes an individual data point, Q_1 and Q_3 are the first and third quartiles, respectively, and $\mathrm{IQR} = Q_3 - Q_1$. The strip-plot shows every individual particle's displacement in radial position and height. The distribution helps to visualise the spread and clustering of the radial position and heights by the particles. Both plots show that the axial transport (height displacement) is more evenly distributed and has a greater variance. The radial displacement is much more clustered near 0 mm to 0.5 mm. This suggests that during the inspection window, strong axial effects dominate the radial ones.

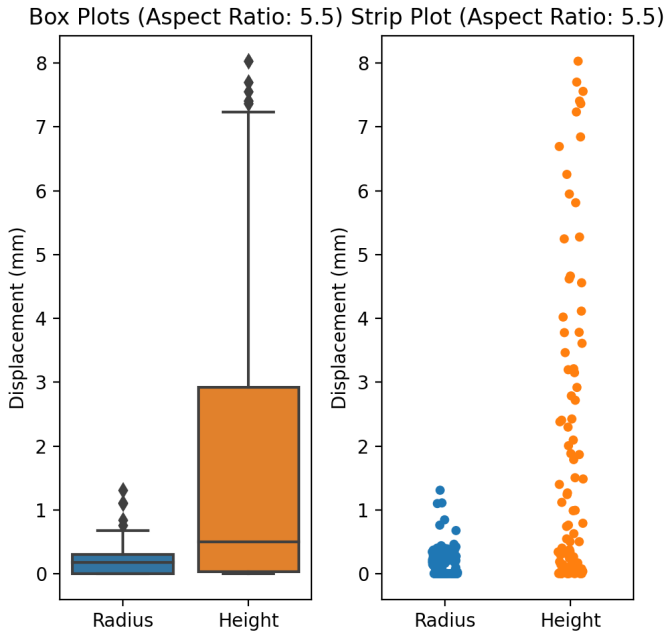


Figure 83: Displacement Magnitude represented by a box-plot (left) and a strip-plots (right) for the initial and final radial position and height of particles between t = 1.1 s and t = 1.5 s with Aspect ratio = 5.5 at Re=500 ($\approx$ 1372 RPM).

Figure 84 shows the magnitude of the displacement with a box and strip plot for all the aspect ratios studied. As the aspect ratio is reduced, the maximum height recorded is also reduced. For higher aspect ratios, the height displacement is the most dominant change compared to the radial changes. Overall, the radial effects increase slightly with a reducing aspect ratio. For example, at AR = 0.5 (Figure 84a), the interquartile range is approximately 0.55 mm compared to approximately 0.4 mm at AR = 5.5 (Figure 84k). However, this is not a significant change, and the radial effects are minimal.

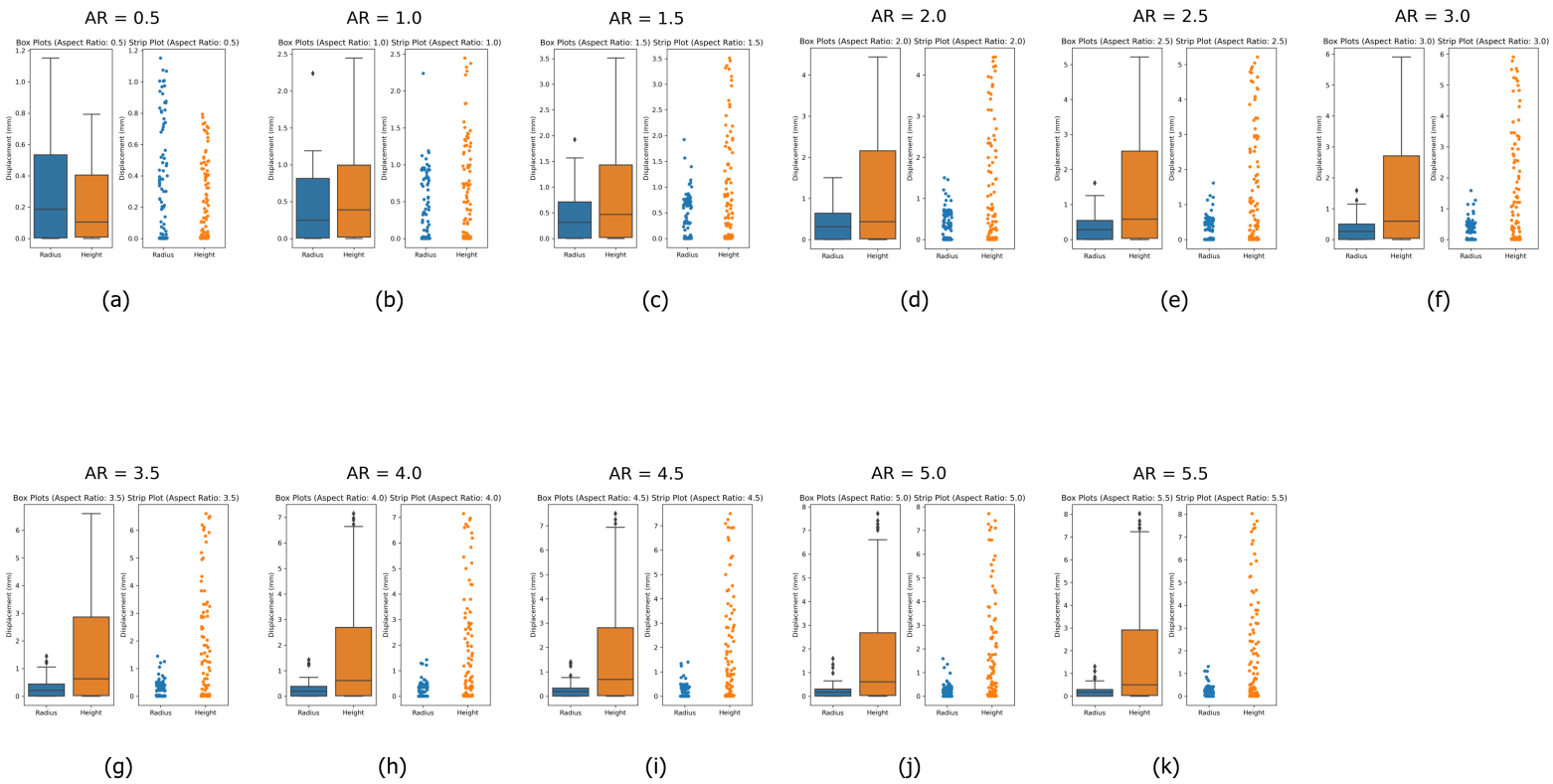


Figure 84: Displacement magnitude represented by a box-plot (left) and a strip-plots (right) for the initial and final radial position and height of particles between t = 1.1 s and t = 1.5 s for aspect ratios of 0.5, 1,0, 1,5, 2.0, 2.5, 3.0, 3.5, 4,0, 4,5, 5.0 and 5.5 in (a), (b), (c), (d), (e), (f), (g), (h), (i), (j) and (k) respectively at Re=500 ($\approx$ 1372 RPM).

To quantitively assess the spatial distribution of the particle position during spin-down, one can further process the results through data binning. This method is used to partition a continuous space, such as the cylindrical domain, into discrete sections or bins, both in the radial and axial directions. By doing so, a rectangular subgrid is generated within the original domain. For this study, the grid consists of 5 radial divisions and 20 height divisions. Each bin represents a small and fixed region of physical space within the domain. By counting the number of instances a particle is found within each bin during the inspection window, one can find the localised variation in particle concentration. The count per bin for the number of instances recorded is normalised by the total data points recorded and plotted on a colour-coded heat map so that different aspect ratios can be compared, where areas of higher concentration are indicated by darker shades. For an aspect ratio of 5.5, Figure 85 shows this result. The darkest regions (blue) are near the bottom right, and the centre of the cylinder near 16 mm height. There is a region between the height of 1 mm and 4 mm where there is no significant movement or clustering. Note that this is specifically true for massless particles released anywhere within the grid. Realistically, most particles will have sunk to the bottom as they will have mass, and this region should have more clustering.

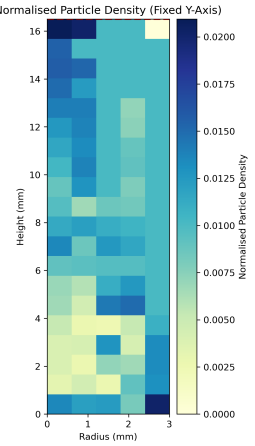


Figure 85: 2D heatmap showing normalised relative density of particle in a binned domain for Aspect Ratio 5.5 at a Reynolds number of 500 ($\approx$ 1372 RPM) from t = 1.1 s to t = 1.5 s

Figures 86 shows the heatmap plot discussed above for all aspect ratios in the study. The figure has been scaled to 16.5 mm for a direct comparison between the aspect ratios. For a realistic representation, the same data is plotted in Figure 87 where the red dashed line represents the maximum height of the cylinder within that particular aspect ratio without any scaling. The results for the massless particles show that the absence of particles is detected in the region between 1 mm and 4 mm. This is present across the aspect ratios (when looking at the scaled figures). This shows the same general trend across the aspect ratios in terms of particle clustering for massless grids. Outside of this region, most of the particles are present within the inspection window. The darker regions grow as the aspect ratio increases, where more blue squares are present for an aspect ratio of 5.5 when compared to the 0.5 case.

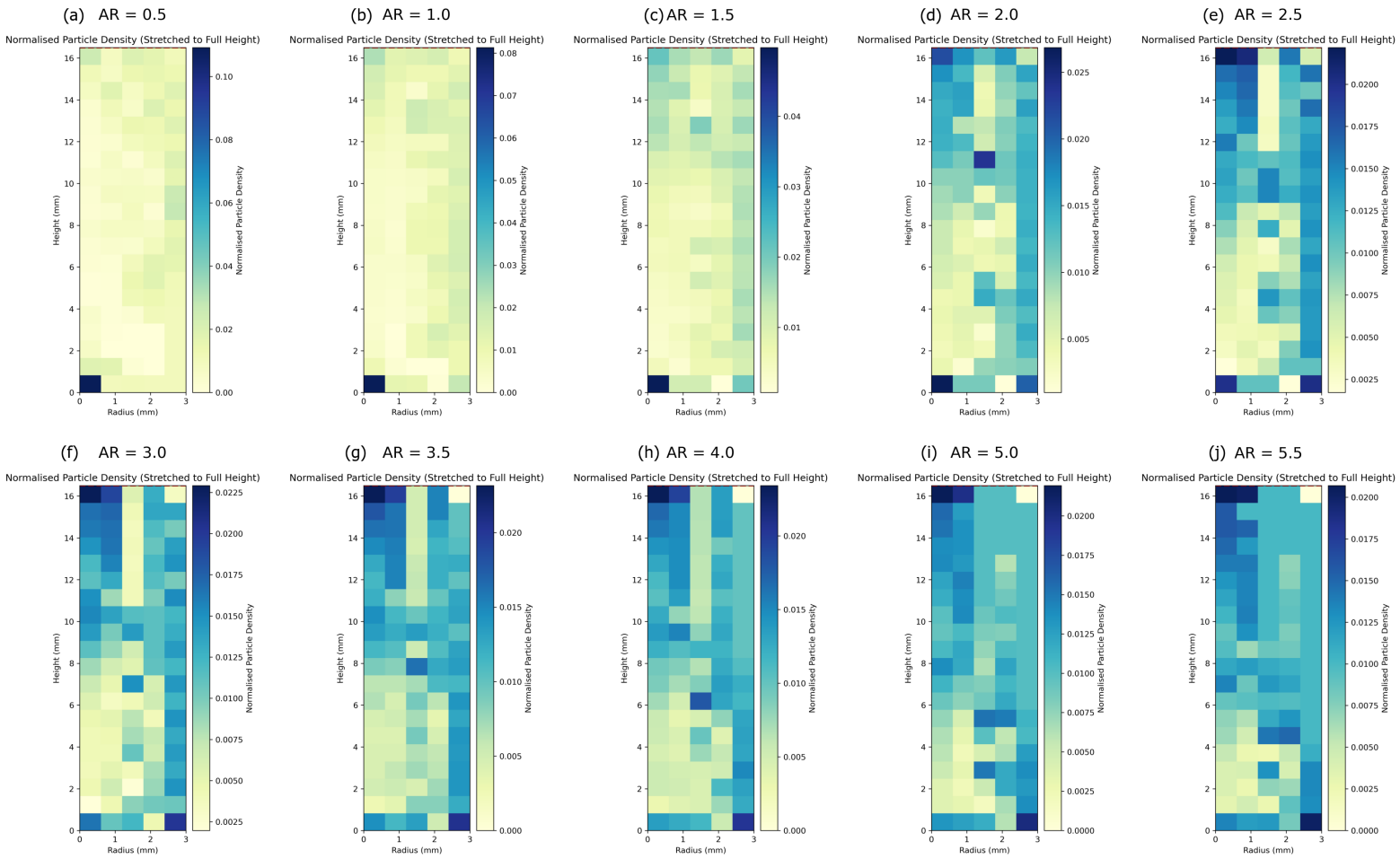


Figure 86: Scaled 2D heatmap showing normalised relative density of particle in a binned domain for aspect ratios of 0.5, 1,0, 1,5, 2.0, 2.5, 3.0, 3.5, 4,0, 5.0 and 5.5 in (a), (b), (c), (d), (e), (f), (g), (h), (i) and (j) respectively at a Reynolds number of 500 ($\approx 1372\,\mathrm{RPM}$) from t = 1.1 s to t = 1.5 s.

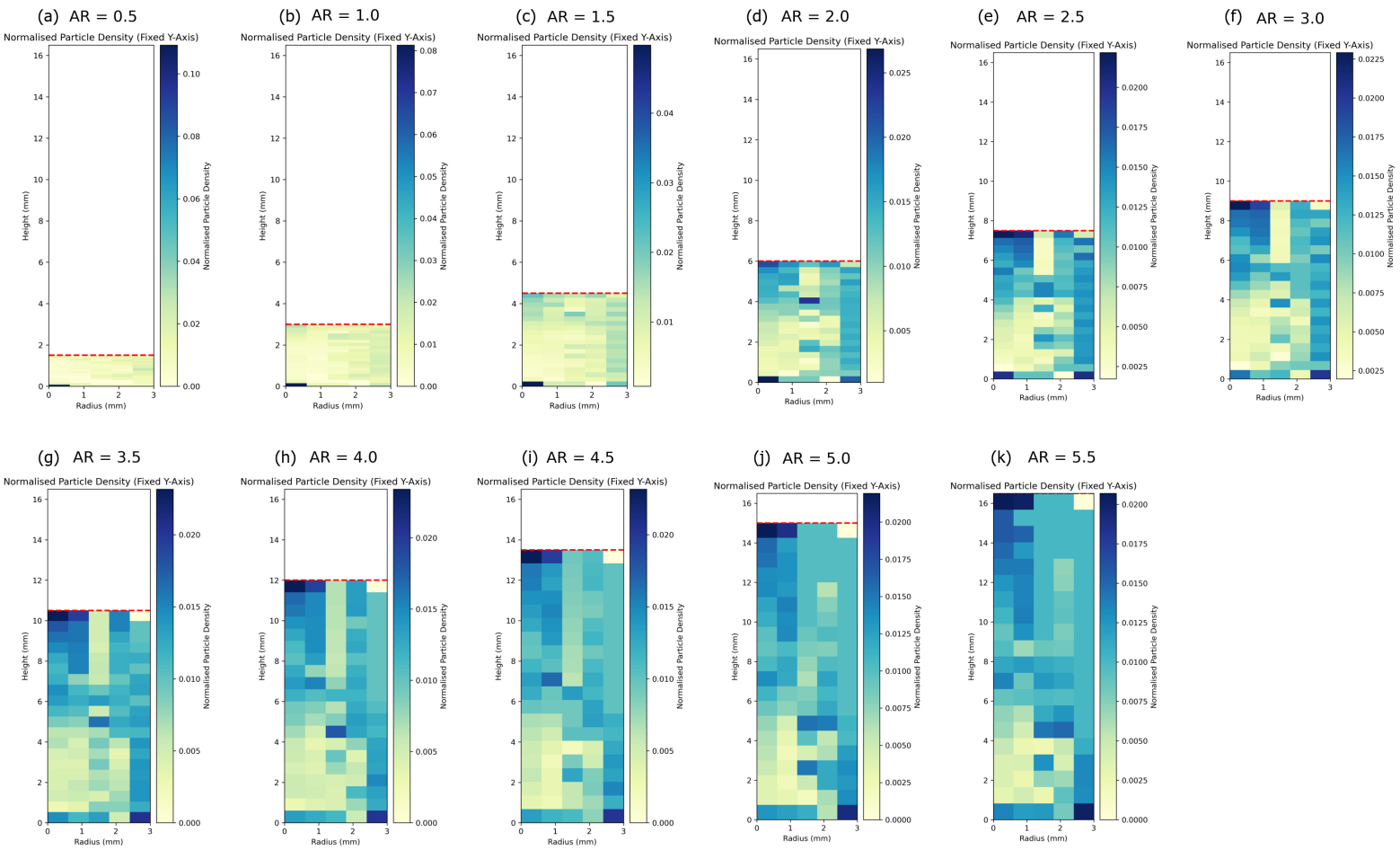


Figure 87: Unscaled 2D heatmap showing normalised relative density of particle in a binned domain for aspect ratios of 0.5, 1,0, 1,5, 2.0, 2.5, 3.0, 3.5, 4,0, 4,5, 5.0 and 5.5 in (a), (b), (c), (d), (e), (f), (g), (h), (i), (j) and (k) respectively at a Reynolds number of 500 ($\approx$ 1372 RPM) from t = 1.1 s to t = 1.5 s.

The same data are then displayed using kernel density estimate (KDE) plots showing smoothing over the data to help to see the regions of activity clearly. A KDE plot is used to represent the distribution of particle positions in a continuous manner. Unlike a histogram, which groups data into discrete bins, the KDE produces a smooth approximation of the underlying probability density function by summing kernel functions centred at each data point. These show the likelihood of finding a particle in each part of the domain. As before, Figure 88 shows the plot in which the y-axis is scaled to match the highest aspect ratio for direct comparison. Figure 88 shows the unscaled version for a true representation. For these plots, the data are normalised by 0 for the minimum number of detections and 1 for the maximum number of particle activity.

In these plots, the lighter region represents the absence of particle detection while the dark blue region represents the region in which most of the particles are tracked within the inspection window. For an aspect ratio of 0.5, two significant regions are seen. One is the 0 to 0.2 mm height near the centreline and the second is the top right of the domain. The region near the centreline dissipates as the aspect ratio increases. The region on the top right becomes more prominent as the aspect ratio increases with regional features. For higher aspect ratios, vertical streaks start to form, splitting the top right region into smaller hotspots. The lack of particles near the centreline highlighted by the lighter colours shows that the particles have been lifted very early and that they are tracked at greater heights within the inspection window.

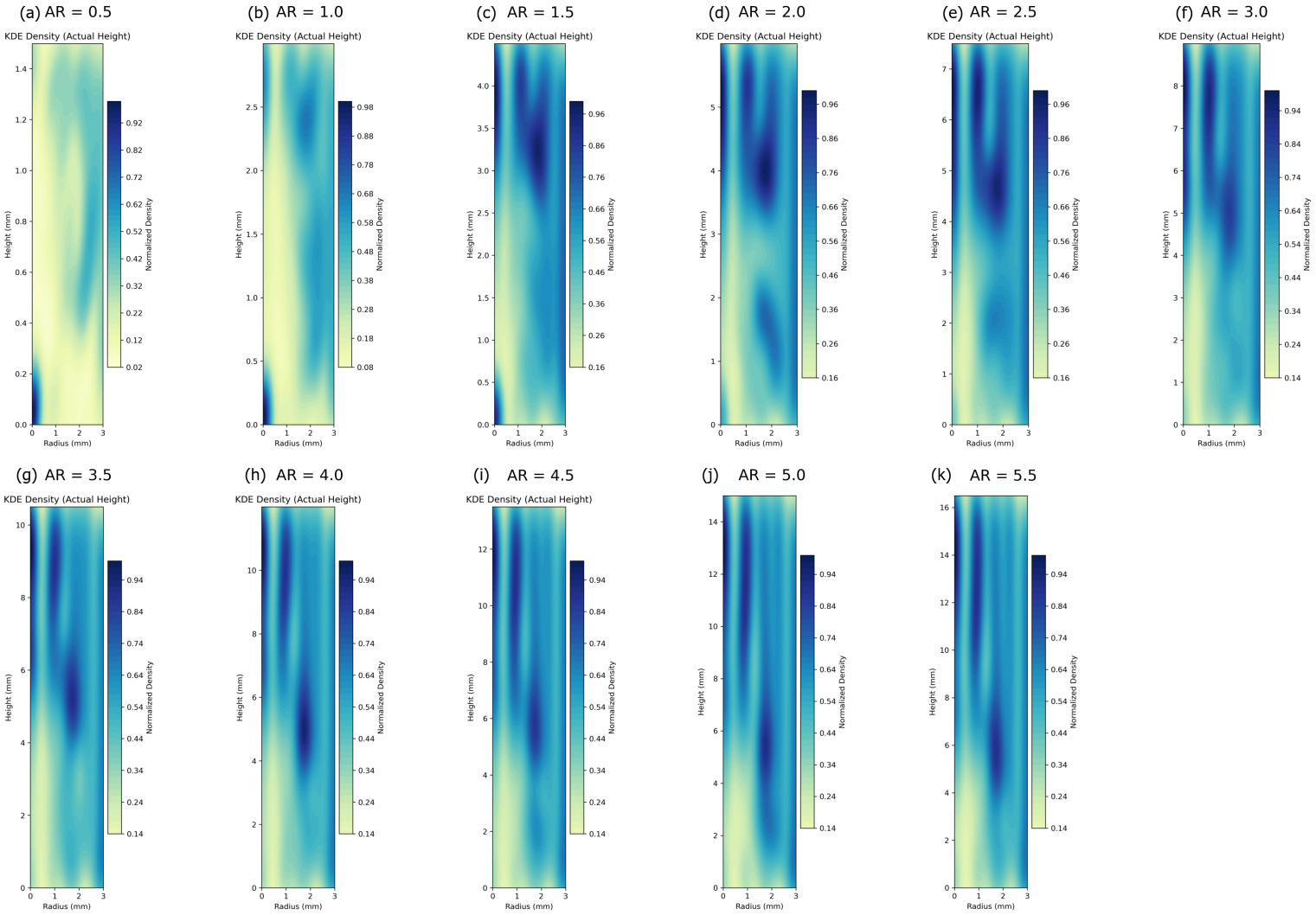


Figure 88: Scaled 2D KDE plot showing normalised density smoothing of particle in a binned domain for aspect ratios of 0.5, 1,0, 1,5, 2.0, 2.5, 3.0, 3.5, 4,0, 4,5, 5.0 and 5.5 in (a), (b), (c), (d), (e), (f), (g), (h), (i), (j) and (k) respectively at a Reynolds number of 500 ($\approx$ 1372 RPM) from t = 1.1 s to t = 1.5 s.

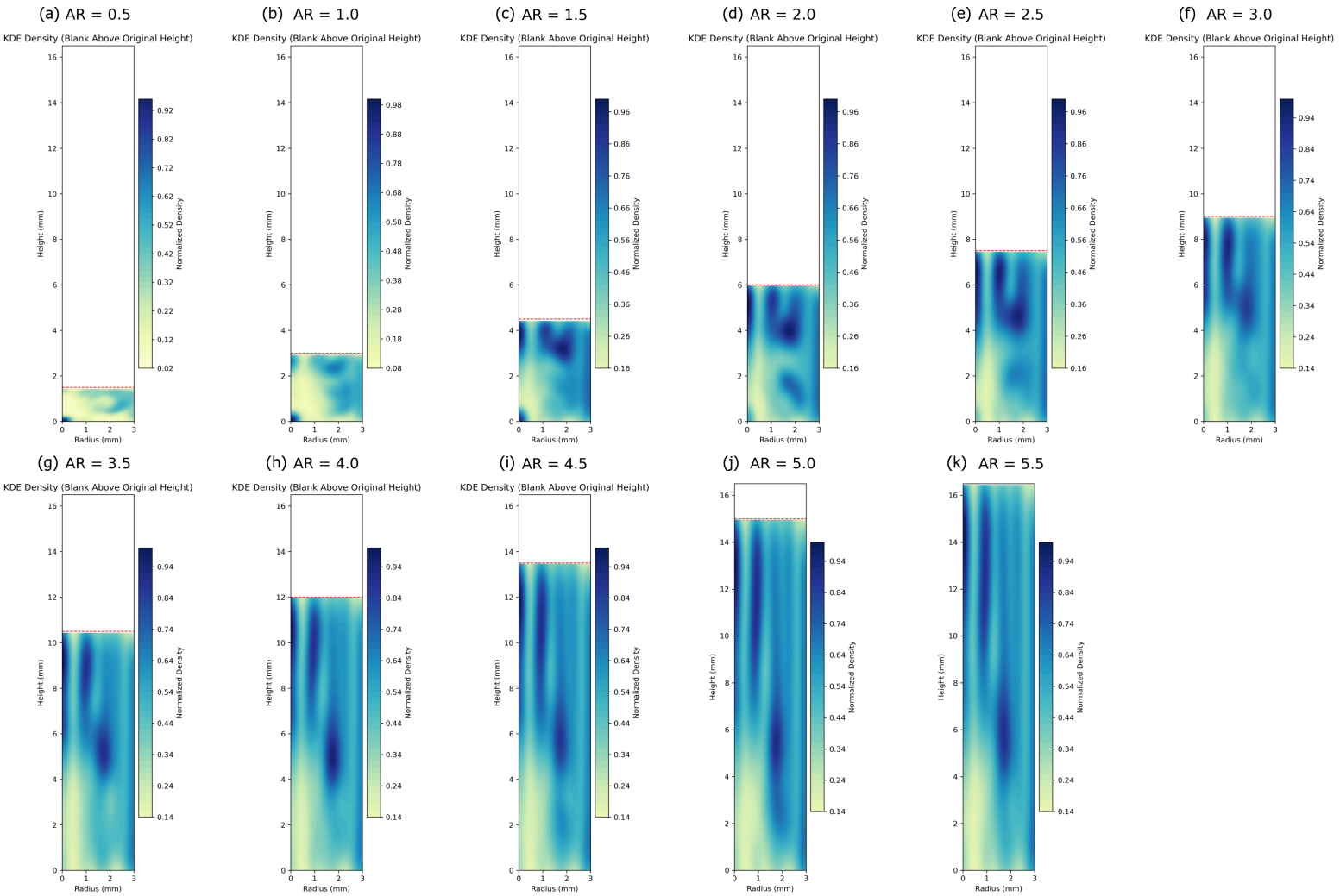


Figure 89: Unscaled 2D KDE plot showing normalised density smoothing of particle in a binned domain for aspect ratios of 0.5, 1,0, 1,5, 2.0, 2.5, 3.0, 3.5, 4,0, 4,5, 5.0 and 5.5 in (a), (b), (c), (d), (e), (f), (g), (h), (i), (j) and (k) respectively at a Reynolds number of 500 ($\approx$ 1372 RPM) from t = 1.1 s to t = 1.5 s.

5.2.2 Reynolds Number Study

A similar analysis to the aspect ratio study in Section 5.2.1 is conducted for the Reynolds number study. However, the number of particles is 500 instead of 100 released as a 10 x 50 grid (shown in Figure 77). This is to cover more of the domain compared to the aspect ratio study. First, consider the radial and height distributions of particles as the Reynolds number changes. Figure 90 shows the box plot and the strip plot showing the distribution of the 500 massless particles. As before, there is a larger change in height when compared to radial position, across all Reynolds numbers. At higher Reynolds numbers (Re=3000), the maximum height change observed is approximately 12 mm. Whereas for Re = 500, this is approximately 10 mm. Across the various Reynolds numbers, there is not much change in the radial data ($< 1\,\mathrm{m\,m}$).

Figure 91 shows the normalised particle density obtained for various Reynolds numbers. The KDE for the same data showing the smoothing is shown in Figure 92. For Reynold's number of 500, The histogram and the KDE plot shows particles are detected in most of the domain except the 0 to 4 mm ranges near the centreline. As the Reynolds number increases, the blue regions start to dissipate into smaller areas until the majority of particle density is focused either near the top of the centreline or the outside walls. The regions are also unevenly distributed, and this is intrinsically linked to the chaotic flow at higher Re, causing more disturbance to individual particles.

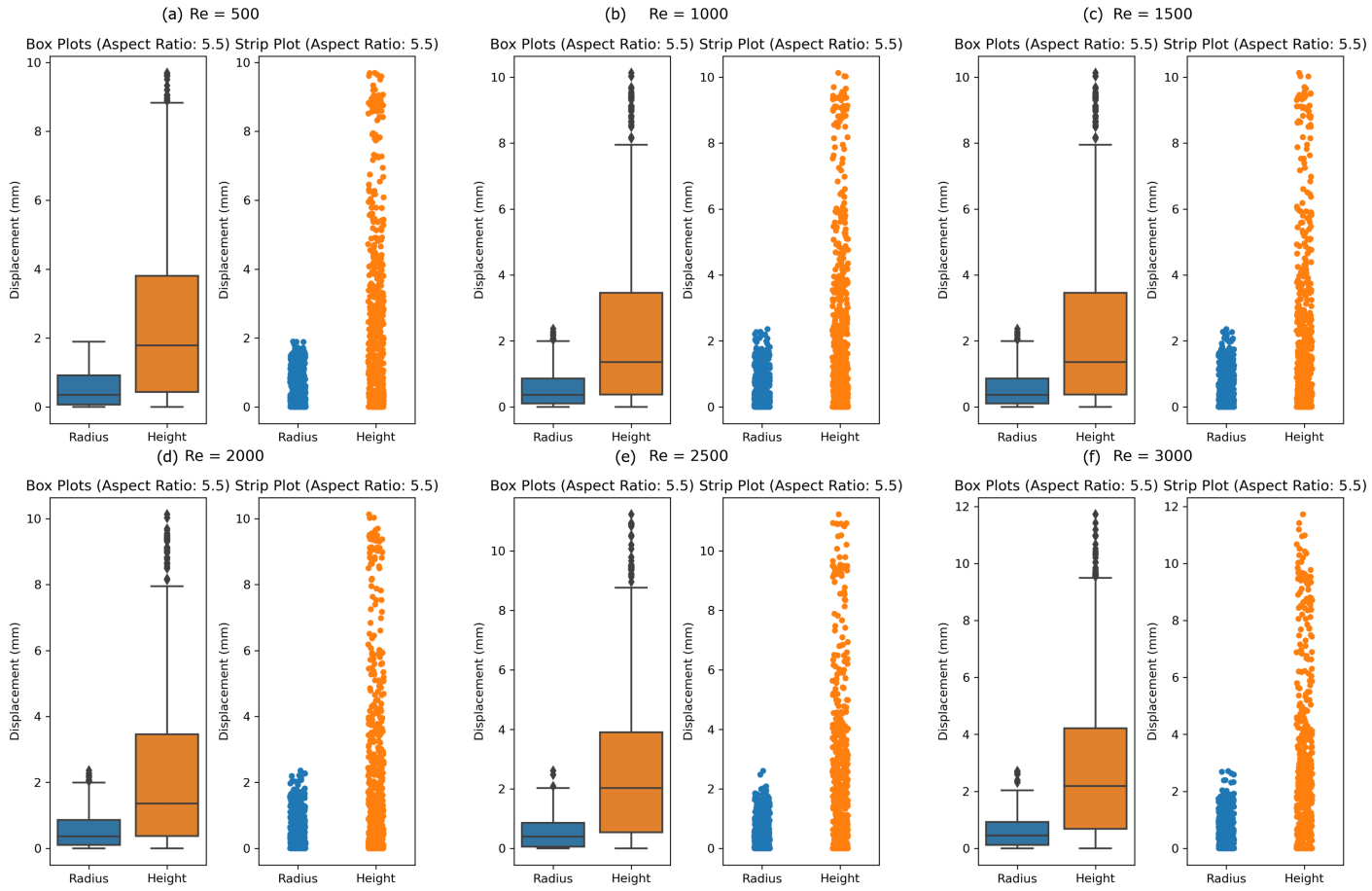


Figure 90: Displacement Magnitude represented by a box-plot (left) and a strip-plots (right) for the initial and final radial position and height of particles between t = 1.1 s and t = 1.5 s for Reynolds numbers of 500, 1000, 1500, 2000, 2500 and 3000 shown by (a), (b), (c), (d), (e) and (f) respectively

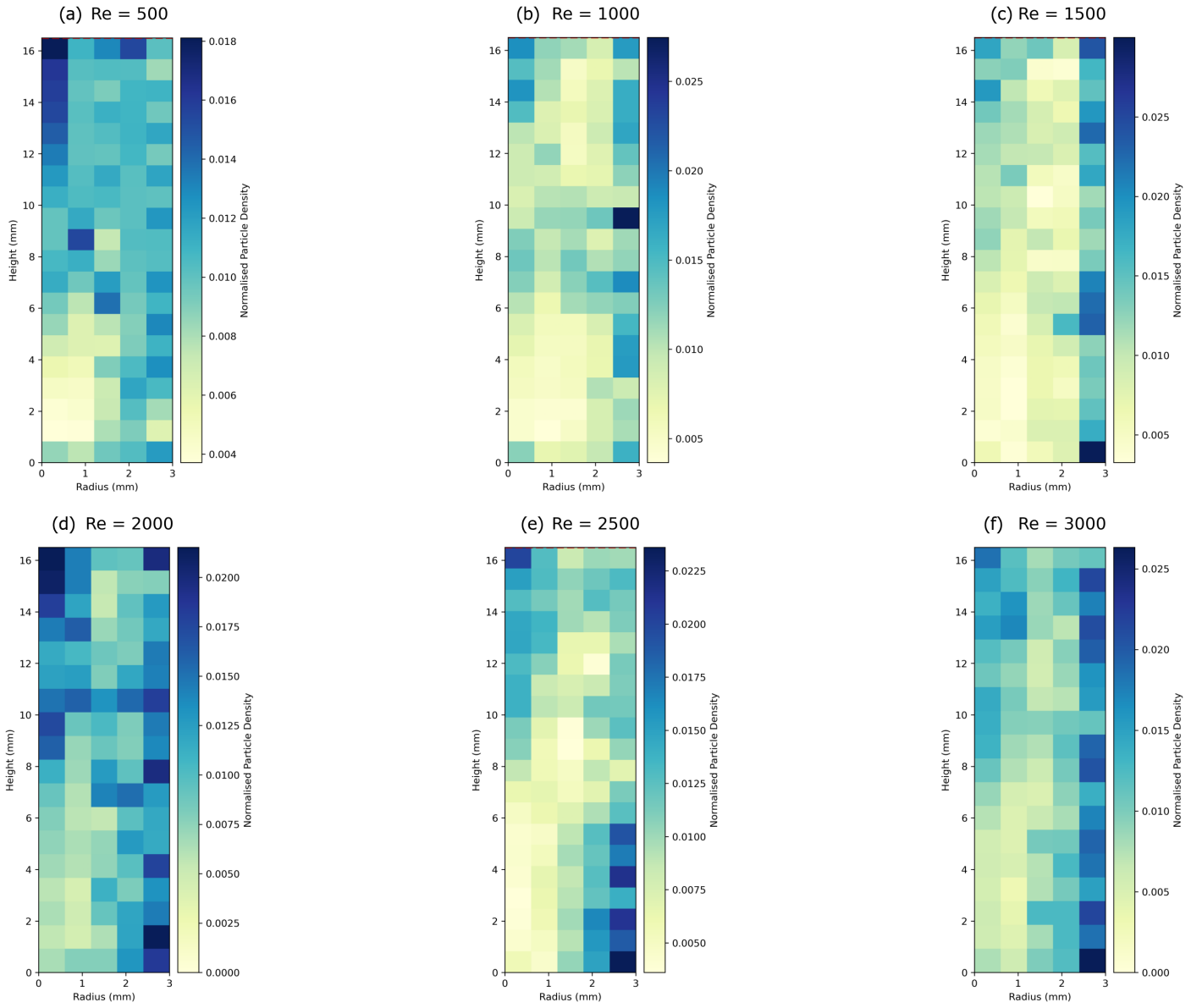


Figure 91: 2D heatmaps showing normalised relative density of particle in a binned domain for Reynolds numbers of 500, 1000, 1500, 2000, 2500 and 3000 shown by (a), (b), (c), (d), (e) and (f) respectively from t = 1.1 s to t = 1.5 s

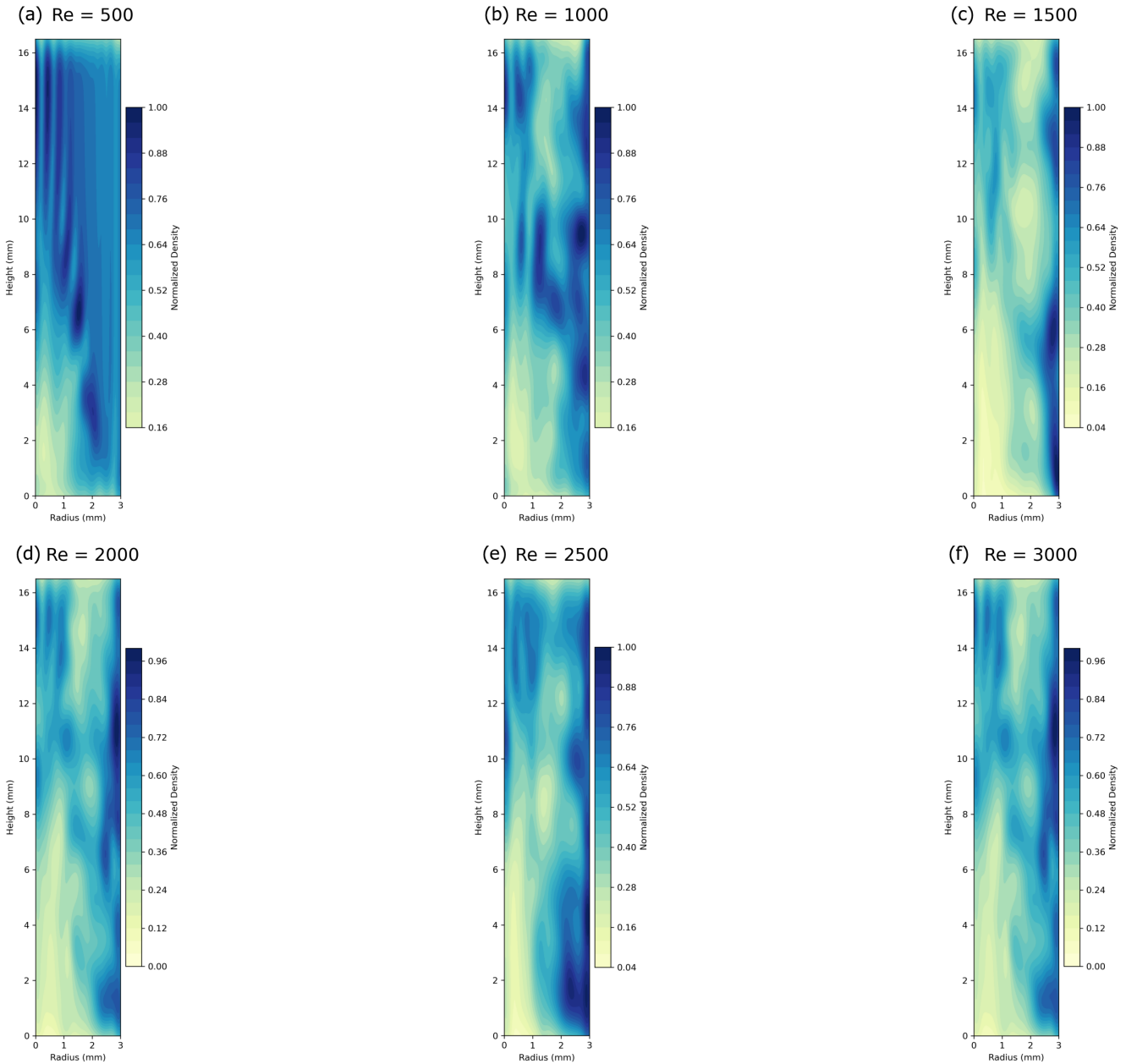


Figure 92: 2D KDE plot showing normalised density smoothing of particle in a binned domain for Reynolds numbers of 500, 1000, 1500, 2000, 2500 and 3000 shown by (a), (b), (c), (d), (e) and (f) respectively from t = 1.1 s to t = 1.5 s

5.3 Particles With Mass

In this section, the motion of particles with mass is examined and the effect of the force of gravity is compared to the massless case. Unlike massless tracers, particles with a mass experience inertia and respond differently to massless particles. From the standards, PFS cannot contain particles of both visible and sub-visible ranges. These can be small or large and some examples are shown on Figure 4. These particles are a much better approximation to particles in real-life inspection processes and their mass can be modified by varying the diameter, D (in mm), of the particle and keeping the density the same.

5.3.1 Single Mass Particle

Four particles are released at a radial position of 1 mm and a height of 3 mm, and the spin speed is set to 1000 ($Re \approx 364$). This initial seeding is selected to model particles close to the plunger and offset from the central axis so that the particle does not just move vertically upward along the centreline. These seeding conditions are kept the same for all particles released in this study. These properties resemble the typical values of properties associated with particles during an inspection. A massless particle (Figure 93a) is compared to, D=25 µm ((Figure 93b), 50 µm (Figure 93c) and 75 µm (Figure 93d). All particles excluding the 0 mm particles have a density of 1200 kg m^{-3} as they are slightly heavier than the surrounding fluid, making the particles sink outside the influence of the axial jets. This particle density has been selected as it was also used in the experimental studies conducted. In this study, the full cycle of spin-up and spin-down is considered to get an idea of a typical particle track during inspection. This allows one to observe how the diameter and inertia of the particles influence the trajectory, the radial displacement, and height under the influence of fluid motion.

Figures 93 show the side view of the corresponding particle tracks observed for all the particles. During spin-up, the massless particle has a lower maximum radial position and height when compared to particles with larger diameter. As the diameter (mass) increases, the radial spread increases, and the final height also increases. In a rotating fluid, heavier particles are expected to follow a larger radial position as a result of stronger inertial and centrifugal forces.

Figure 94, Figure 95 and Figure 96 show the particle height, radial position and velocity magnitude against time for up to 5 seconds. Generally, all particle tracks exhibit a similar trend. The particles have a helical trajectory with initial downward movement in the axial direction because of the secondary effects of the downward axial jet during spin-up. The 0 µm (massless), 25 µm and 50 µm show a maximum drop down to 0.16 mm until the end of spin-up and 1 second. At the same time, the particles are transported radially outward near the outer walls. The radial position increasing with increasing diameter with 0 µm (massless), 25 µm, 50 µm and 70 µm reaching 2.0 mm, 2.1 mm, 2.3 mm and 2.8 mm respectively. This behaviour is consistent with the behaviour of the fluid during spin-up, where centrifugal effects are dominant. This previously explained through Figure 39b.

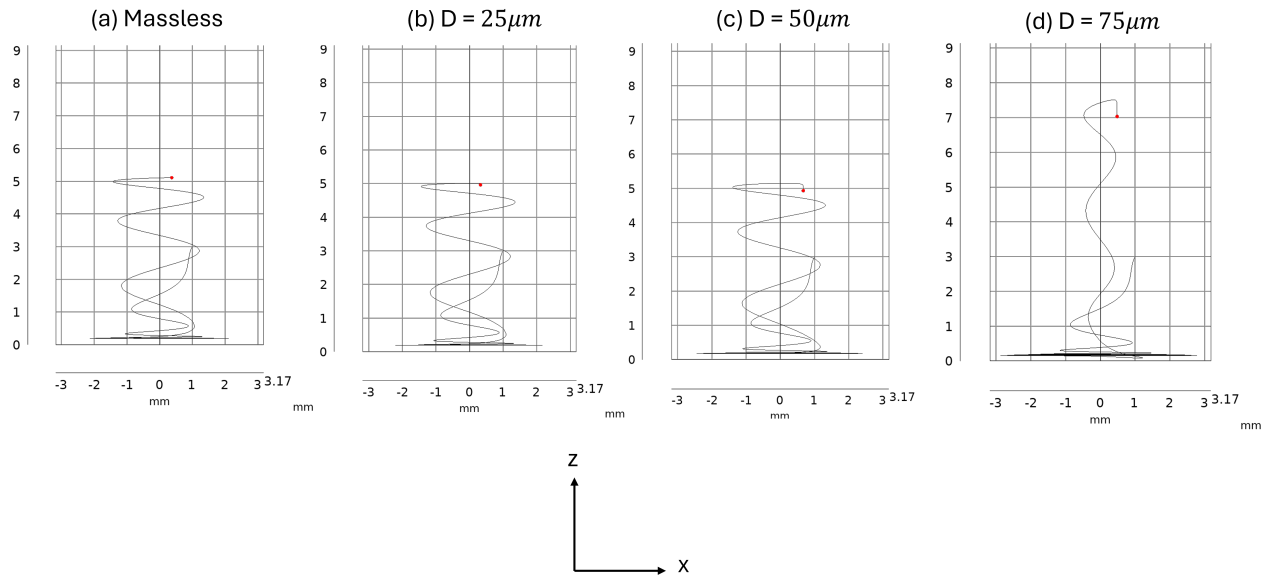


Figure 93: Side-views showing particle track up to 5 seconds for a particular mass and a density of 1200 kg m^{-3} released at $z = 3mm$, $r = 1mm$ for Re = 1000 (where 0 µm = massless).

As spin-down begins, all cases tested show a reduction in particle radial position. For the massless particle and particles with diameters of 25 µm and 50 µm, the reduction is 1.5 mm when the 75 µm particle is reduced to a lower 0.5 mm. The particles move radially inward towards the centre of the cylinder and are then caught by the axial jet lifting the particles to their maximum height. For 0 µm, 25 µm and 50 µm, the maximum height is recorded at about 5 mm and the 75 µm particle is lifted to a much higher 7.5 mm. Once the fluid is stationary, there is a drop in height in the mass of the particles as a result of gravitational effects taking over. This is further highlighted in Figure 94, which shows the final position as a red dot. The 75 µm particle shows a greater radial and vertical displacement due to increased inertia during both spin-up and spin-down. This is shown by a radial position of approximately 2.8 mm compared to 2.3 mm for the 50 µm particle during spin up and 0.5 compared to 1.5 for the 50 µm particle. Both particles are affected by the jets, but settle with a variation. This shows that particles with mass are affected not only by centrifugal forces but also by local flow fields. In Figure 96, particles with a larger diameter have much larger acceleration (steeper rate of change on Figure 95) compared to smaller diameters. They also have a larger velocity at the end of spin up. During spin-down, these particles also slow down much faster but keep moving vertically downwards due to gravitational effects until the end of the cycle.

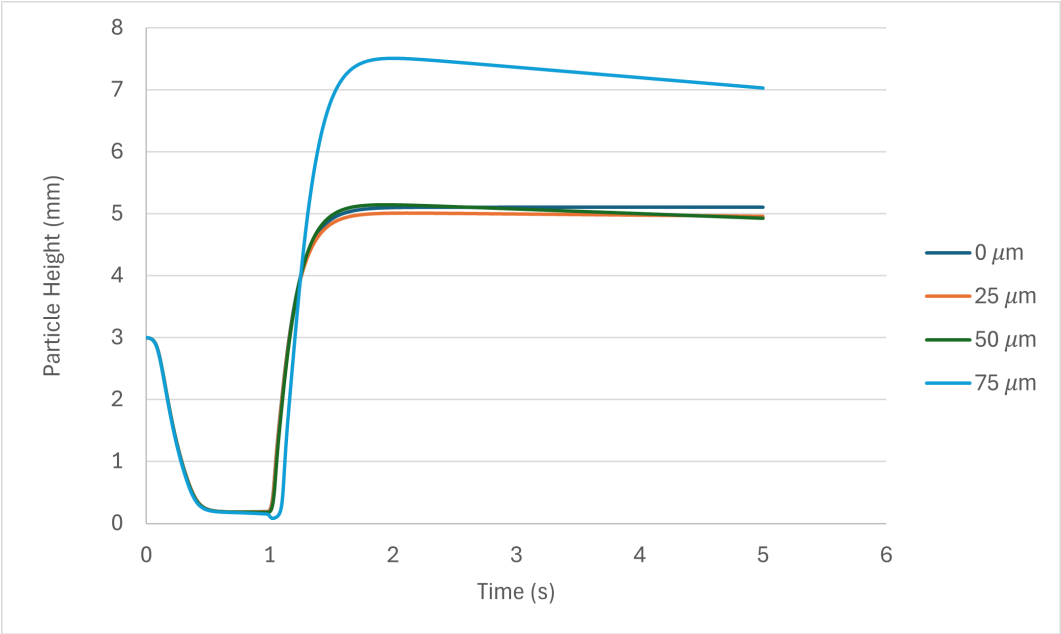


Figure 94: Height of the particle over time for various particles diameters with density $1200\,\mathrm{kg\,m^{-3}}$ released at $z = 3\,\mathrm{mm}$, $r = 1\,\mathrm{mm}$ during the full spin-up and spin-down cycle (where $0\,\mu\mathrm{m}$ = massless).

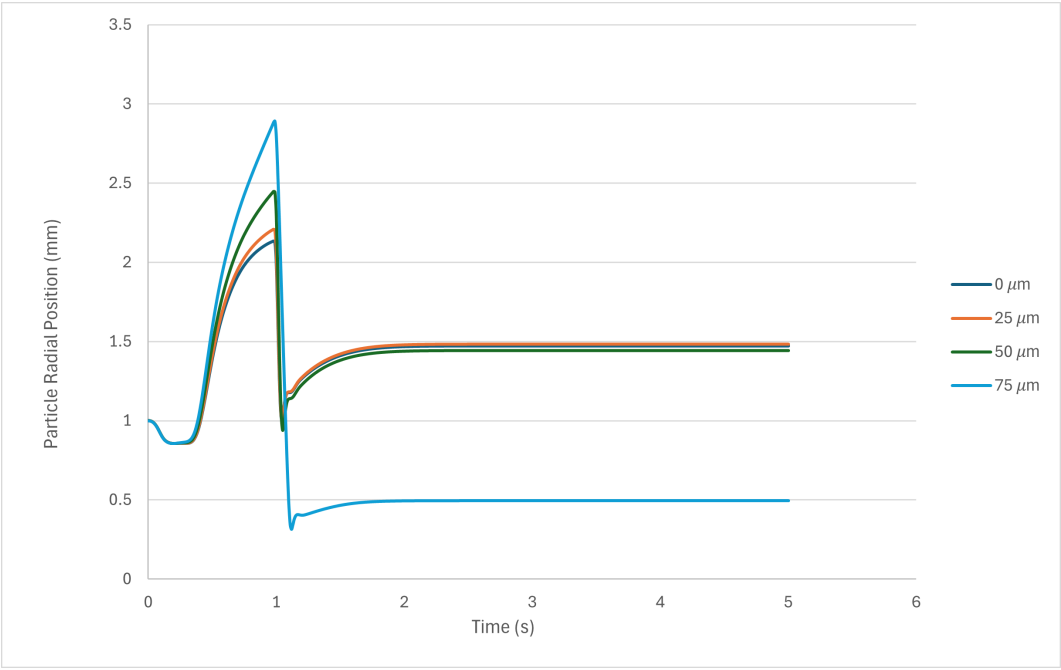


Figure 95: Radial position of over time for various particles with density $1200\,\mathrm{kg\,m^{-3}}$ released at $z = 3\,\mathrm{mm}$, $r = 1\,\mathrm{mm}$ during the full spin-up and spin-down cycle (where $0\,\mu\mathrm{m}$ = massless).

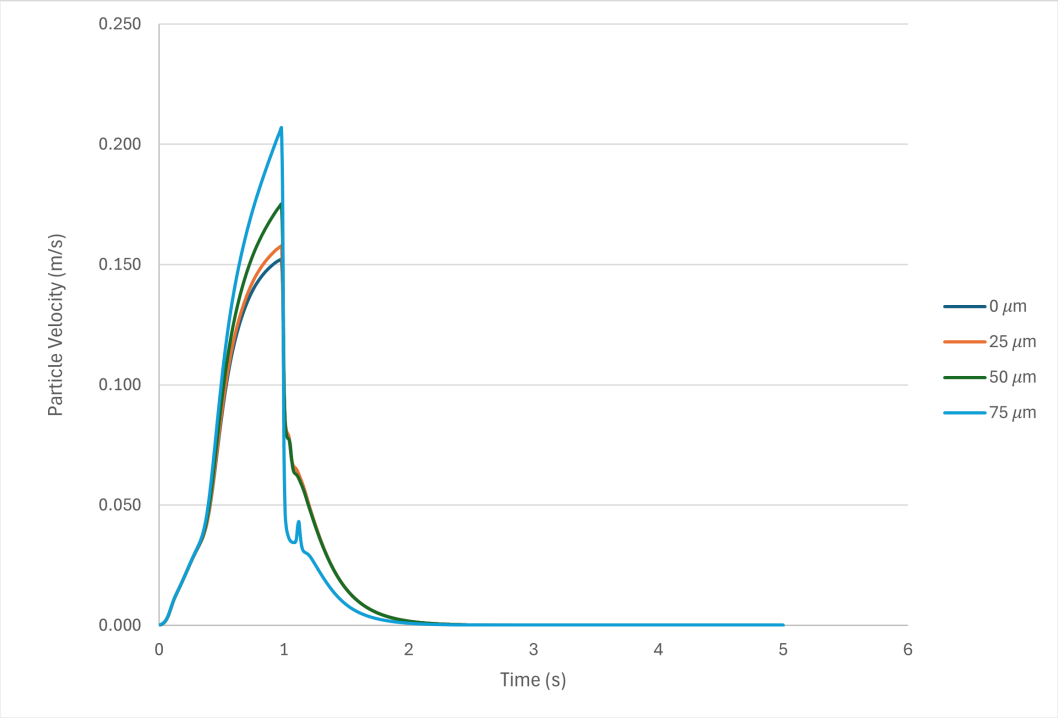


Figure 96: Velocity of various particle diameters over time with a density 1200 kg m^{-3} released at $z = 3$ mm, $r = 1$ mm during the full spin-up and spin-down cycle (where 0 µm = massless).

5.3.2 Particle Diameter Study with Grid Seeding

In this study the influences of particle diameter on the motion during the spin-up phase for a Reynolds number of 250 is observed. To do so, a 10 x 10 grid of particles with mass is seeded in the bottom right quadrant of the cylinder. The particle diameters range from 1 µm to 50 µm. All particles undergo the same fluid field over time, with the only variation being the diameter of the particles. For the study, the final position of the particles in the grid at t=0.95 s (end of the spin-up phase) is analysed. Figure 97 shows the initial location of the grid at t=0.

The final position of the grid for all diameters of size at t=0.95 s is shown in Figure 98 and the same effect is also observed on Figure 95. It is clear that the smaller the particle diameter, the closer it is to the centre of the cylinder. This is because larger diameter particles have more inertia and travel farther out during spin-up compared to smaller diameter particles. The smaller particles collected near the outside walls also have higher heights than the larger diameters collected in the same area. This is reversed between 2 mm and 2.5 mm

Analysing the status of the motion of the particles, Figure 99 shows whether the particles are moving at t=0.95 s, stopped moving before the end of spin-up or froze at $t = 0$. The results show specifically for particles of 50 µm diameter. This is another factor to consider as even the same particle diameters placed at different locations around the grid will exhibit different behaviour. From the data recorded, 10 particles immediately froze at $t = 0s$. These

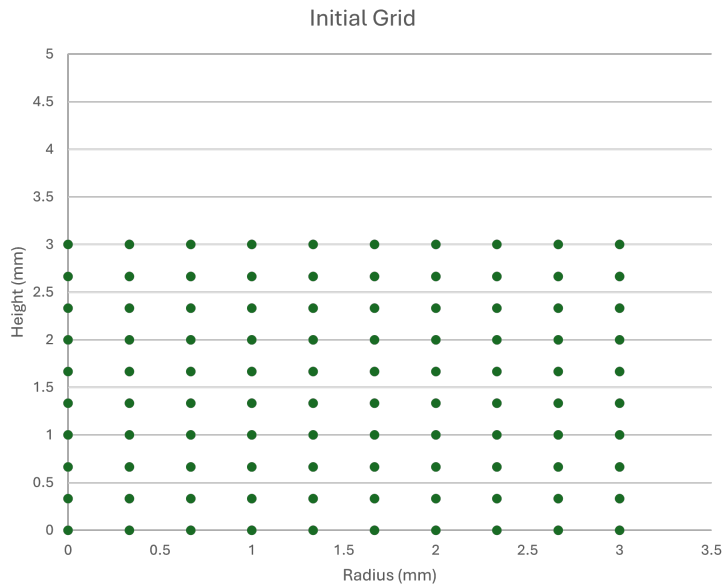


Figure 97: Initial release of the 10 x 10 grid for particles with mass at Reynolds number 250.

are the particles at the centreline and are expected to fall immediately after spin-up begins, and they freeze at the bottom at $r = 0$, $h = 0$. This is expected due to the axial symmetry condition in the problem setup. 21 particles stopped moving before the end of the spin-up and 69 particles were still moving at t=0.95 s. Extending this analysis to all diameters tested, Figure 100 shows the percentage of particles still moving for various diameters. The results show that particles with a larger diameter are more likely to stop moving before the end of the spin up because the percentage of moving particles is close to 50% for the 50 µm diameter compared to nearly 90% for the range of 1 µm diameter to 25 µm diameters.

Figure 101 shows the maximum average velocity of the entire grid for the various diameters. The general trend shows that particles with larger diameters will have a higher peak velocity across the grid. This is a trivial and expected result as larger particles will have higher inertia. In general, the maximum average velocity range for a Reynolds number of 250 is between 0.143 m/s and 0.021 m/s. The average grid position compared to the average initial grid position is highlighted in Figure 102. The results show that the smaller particles between 5 µm and 25 µm (shown by the green ellipse) show an increase in lift, while particles from 30 µm to 50 µm show the opposite through a reduction in lift with increasing diameter. Radial positions are fairly consistent, which implies that axial effects are more dominant than centrifugal effects. In general, the average radial range is between 2.2 mm and 2.43 mm, resulting in a range of 0.23 mm. For the average height, it is between 1.28 mm and 1.43 mm and a range of 0.15 mm. When thinking about inspection, these changes are not really significant. However, the behaviour of particles clearly varies beyond a particle diameter of 30 µm.

The last result of this study is the average distance travelled by the particles within each grid for various diameters. The result is shown in Figure 103 and highlights an increase in the

average distance travelled with increasing particle diameter from 5 µm to 25 µm. This trend is then reversed after 30 µm where the following increases in diameter deduce the average distance travelled by the particles in the grid.

In conclusion, particle diameter significantly affects motion. Smaller particles remain suspended longer and can move higher, whereas larger particles reach a larger radial position. The large particles also dominate early and reach higher peaks, but settle quicker and travel shorter distances than the smaller particles, which are fluidised more during spin-up. When looking at the actual changes between the different diameters, the changes do not seem significant or show a clear trend. Overall, all of this highlights the need to study particle motion to better develop inspection methods for prefilled syringes.

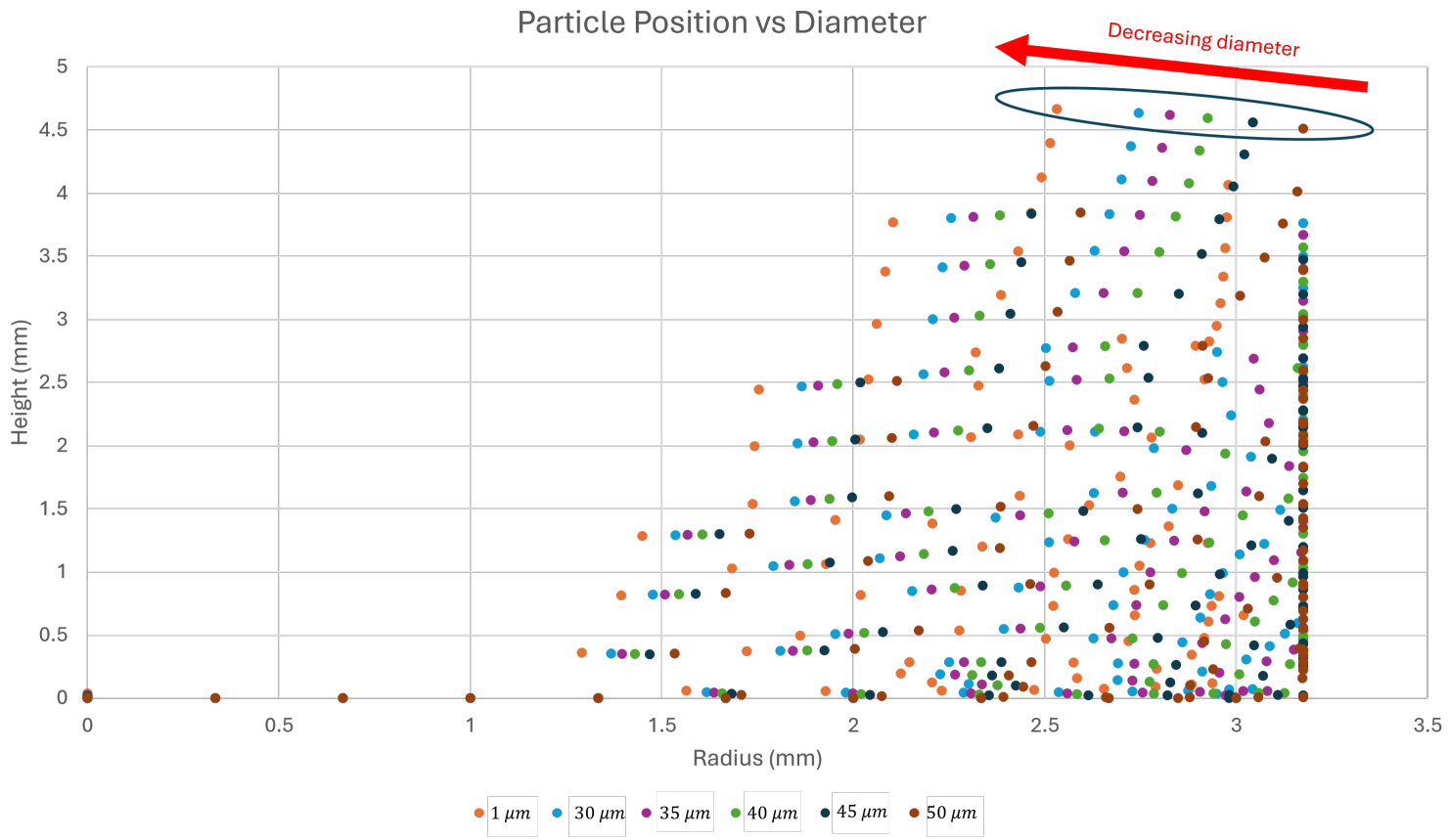


Figure 98: Final particle location (height and radial position) of the 10 x 10 grid of particles with varying diameters at Reynolds number of 250 at $t = 0.95s$

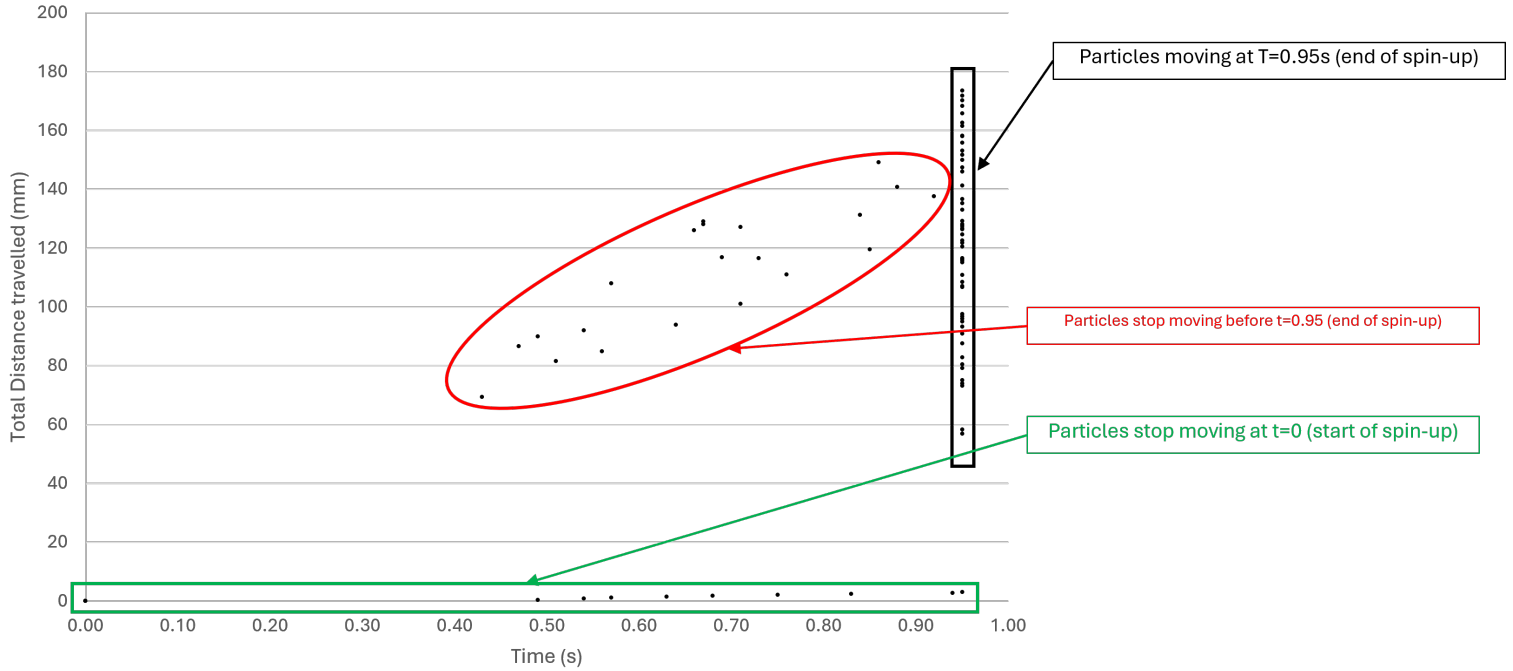


Figure 99: The status of particles using total distance travelled against time for the 10 x 10 grid of particles with varying diameters at Reynolds number of 250 at t=0.95 s

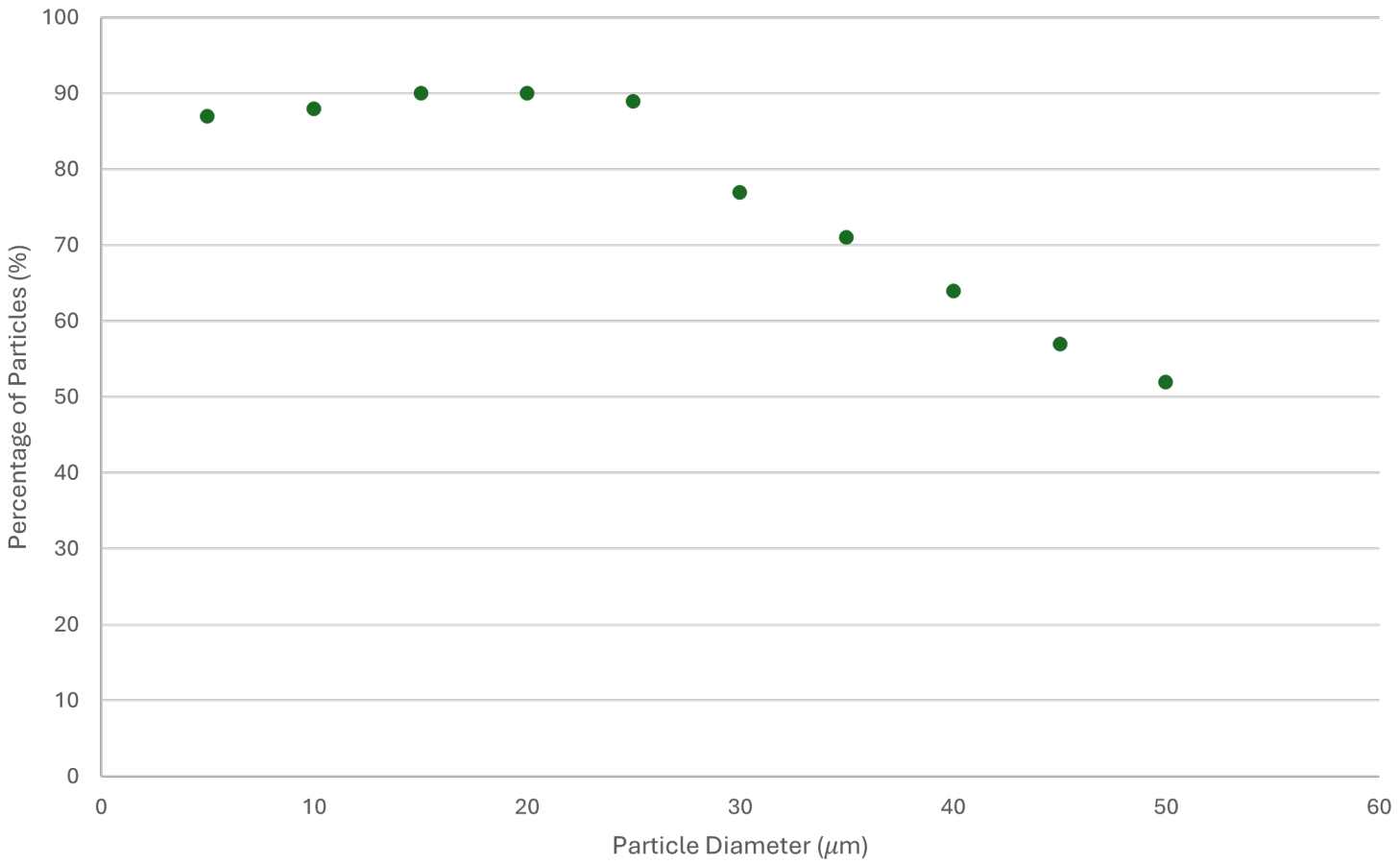


Figure 100: Percentage of particles moving for the 10 x 10 grid of particles with varying diameters at Reynolds number of 250 at $t = 0.95s$

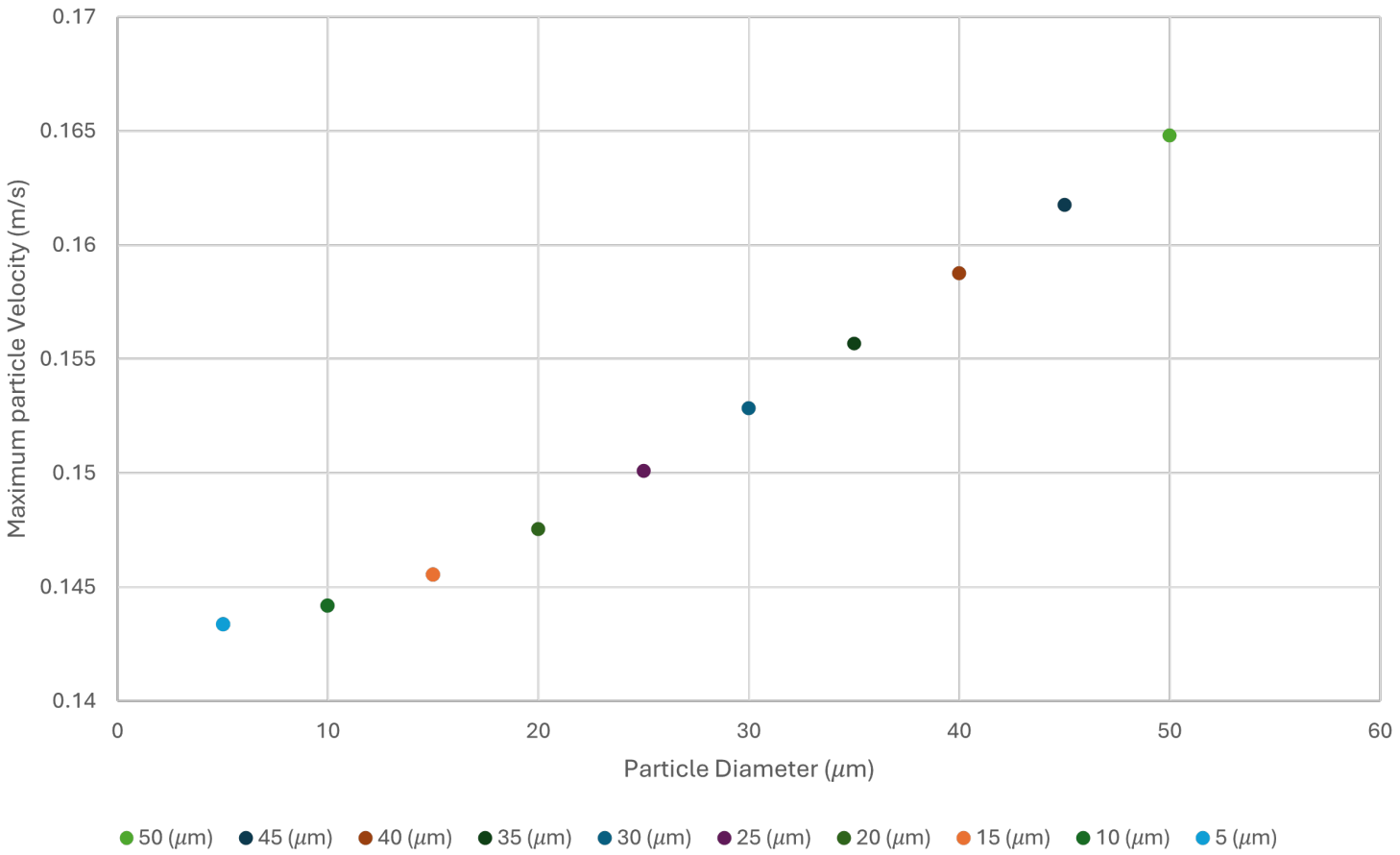
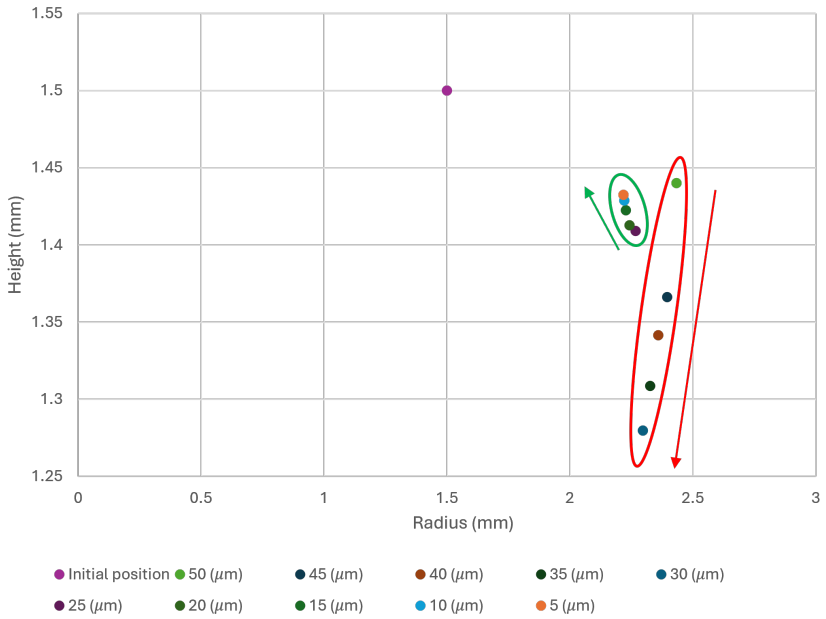


Figure 101: Maximum particle velocity for the 10 x 10 grid of particles with varying diameters at Reynolds number of 250 at $t = 0.95s$

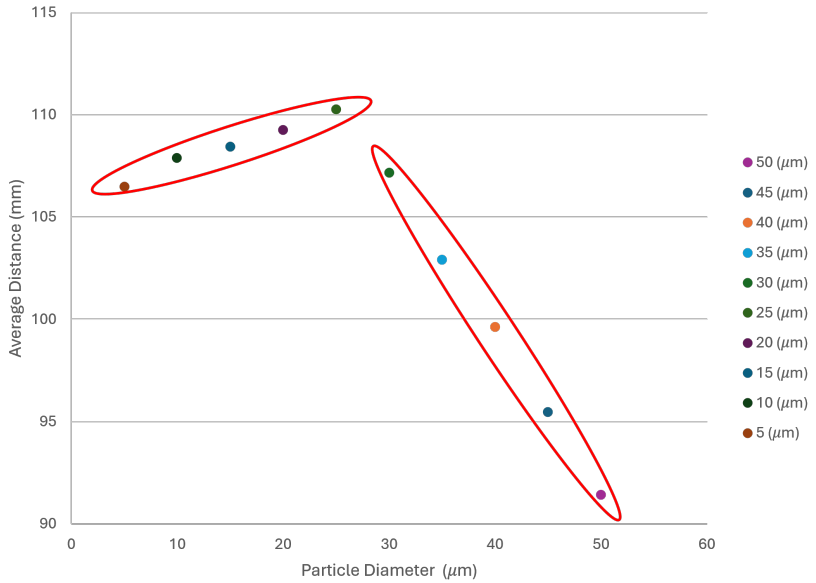


- Increasing height, decreasing radius for : 5 μm - 25μm
- Decreasing height, decreasing radius for : 30 μm - 50μm

Overall:

- Radial range ($2.2 \leq r \leq 2.43$) [change $\sim 0.23mm$]
- Height range ($1.28 \leq h \leq 1.43$) [change $\sim 0.15mm$]

Figure 102: Average grid position for the 10 x 10 grid of particles with varying diameters at Reynolds number of 250 at $t = 0.95s$



- Increasing average distance for : 5 μm - 25μm
- Decreasing average distance for: 30 μm - 50μm

Overall:

- Average distance range ($91.4 \leq D \leq 110.3$) [change ~ 20%]

Figure 103: Average distance travelled for the 10 x 10 grid of particles with varying diameters at Reynolds number of 250 at t=0.95 s

6 Materials and Methods: Experimental Setup

6.1 Overview of the Experimental Objectives

In this section, experiments are conducted in a controlled laboratory setting to replicate the conditions of the robotic inspection process for pre-filled syringes. The primary objective of the experiments is to investigate how a single particle flows under various conditions that mimic the spin cycle of prefilled syringes. Key conditions are altered by varying parameters including the spin speed, fluid aspect ratio, fluid viscosity, particle diameter, and particle density. The resulting particle trajectories are analysed. Understanding how the trajectory of a single particle changes with different parameters, especially those parameters that cause maximum effects on this trajectory, can help to select better spin conditions, enhancing the detection of particles, as well as supporting a general understanding of flows.

6.2 Apparatus

Experimental Chamber Overview

Considering safety first, experiments are conducted in an enclosed environment. Figure 104 shows a schematic of the enclosure in which the experiments are performed. Using a custom-designed mount, a syringe is mounted to a servomotor in the centre of the enclosure with a camera on one side and a backlight on the opposite side. The door of the enclosure is transparent, creating an inspection window. The brightness of the backlight is adjusted by controlling the current and the voltage it receives through a power supply. The table on which the enclosure is fixed is made of steel with wheels allowing the apparatus to be mobile. The wheels can also be locked in place to ensure that the test bench does not move during inspection testing. The arrangement ensures a mobile test bench with a clear view of the particles as the syringe is spun, while ensuring operator safety behind a window.

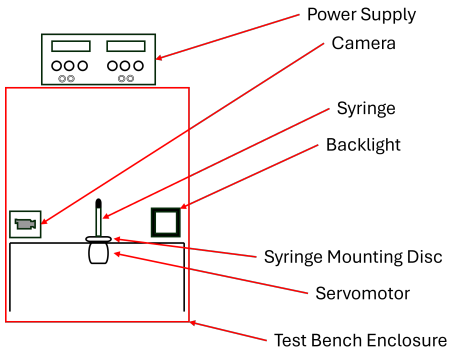


Figure 104: A schematic of the experimental test enclosure containing the essential equipment needed to run a typical pre-filled syringe inspection.

Servomotor

The servomotor used is the Allen-Bradley MPL-A210V-VJ72AA motor with the specifications shown on Table 5. This motor is selected because it allows for a speed range of up to 8000 RPM (Re $\approx$ 2914) which is ideal for inspection.

Table 5: Technical specifications for Servomotor MPL-A210V

Specification	**Value**
Catalog Number	MPL-A210V
Rated Speed (rpm)	8000
Rated Output (kW)	0.37
Rotor Inertia (kg-m^2)	0.000015
Continuous Stall Torque (Nm)	0.55
Peak Stall Torque (Nm)	1.52
Continuous Stall Current (A)	3.1
Peak Stall Current (A)	10.2

Syringe Dimensions

The syringes prepared and used in the experiments are made of glass and provided by a major UK pharmaceutical manufacturer. They have an inner diameter of 6.3 mm and a maximum height of 33 mm when enclosed by a plunger. This is very typical of a syringe used for the inspection of a PFS and only this particular diameter model will be tested.

Lighting

A variable diffused white LED backlight panel with a maximum brightness of 2000 lux is used. The brightness can be adjusted by altering the current and voltage set by the power supply. The size of the LED light is larger than the region of interest captured by the camera to ensure that the entire syringe is lit adequately. The voltage is set to 18 V (giving a current draw of 0.32 A), to ensure uniformity in all recorded data. The LED panel is directly behind the syringe in line with the camera position. The power supply is not connected to the rest of the apparatus as the power to the LED is delivered separately.

Imaging and Data Capture

The camera is mounted on a steel base and does not move. The focal length is determined by placing a thin wire behind the syringe and then in front until both show optimal sharpness. For the calibration of pixels to millimetres, the real diameter of the syringe (6.3 mm) is used. Virtual syringe diameter is measured in pixels and an equivalent ratio is determined from pixels to mm. The camera, lens and settings used are shown in Table 6.

Table 6: Camera, Lens Specifications and settings

Specification	**Details**
Camera Model	Basler acA 1300–30 µm, 1.3 MP, 1/3", Mono, CCD
Frame Rate	75 fps
Exposure Time	9000 µs
Resolution Settings	1296 × 350 pixels
File Format	MP4 Video Recording
Software	Pylon Viewer (to record footage)
Lens Model	Edmund Optics 16 mm fixed focal length (f/1.4–f/16)
Magnification Levels	Constant at focused syringe length

Control Systems

To perform each test safely and accurately, various buttons are used. In total, six different functions, each attributed to a particular action, have been mapped to a controller action. Table 7 shows the different buttons used and their relative functions. Emergency stop buttons have been placed to shut down the apparatus in case of an emergency. The blue test control button is used as a reset button as well as for other mappings if needed. One such example of this mapping is a custom spin profile that alternates between ±8000 rpm to fluidise the particles stuck on the walls or the plunger. This button is also used to create spin profiles that can remove bubbles from a syringe after its preparation.

Table 7: Color-Coded Button Functions for Power and Test Controls

Category	**Color**	**Functions**
Power Control Panel	Red	Power off (to the entire system)
	Green	Power on (to the entire system)
	Blue	Emergency Stop/Reset
Test Control Panel	Red	Emergency Stop
	Green	Begin Test
	Blue	Reset / Other mappings

All wiring from the motor, power controls and test controls leads to the controller powered by a 240 V mains line. The controller is placed inside the power box. The mapping of all buttons is done through a Kinetix 5100C controller by Rockwell Automations. The software used is Rockwell Automation's KNX5100C. The "Test" or "Reset" buttons (green and blue, respectively) for the test controls are set to create the spin profiles used for the experiments. Parameters such as rotational speed, rotation time, acceleration, and deceleration rates are set during this stage. Figure 105 shows the parameters that can be set together with the step function affected when creating a spin profile. Figure 106 shows a summary through various images of the entire experimental rig needed to perform a typical syringe inspection.

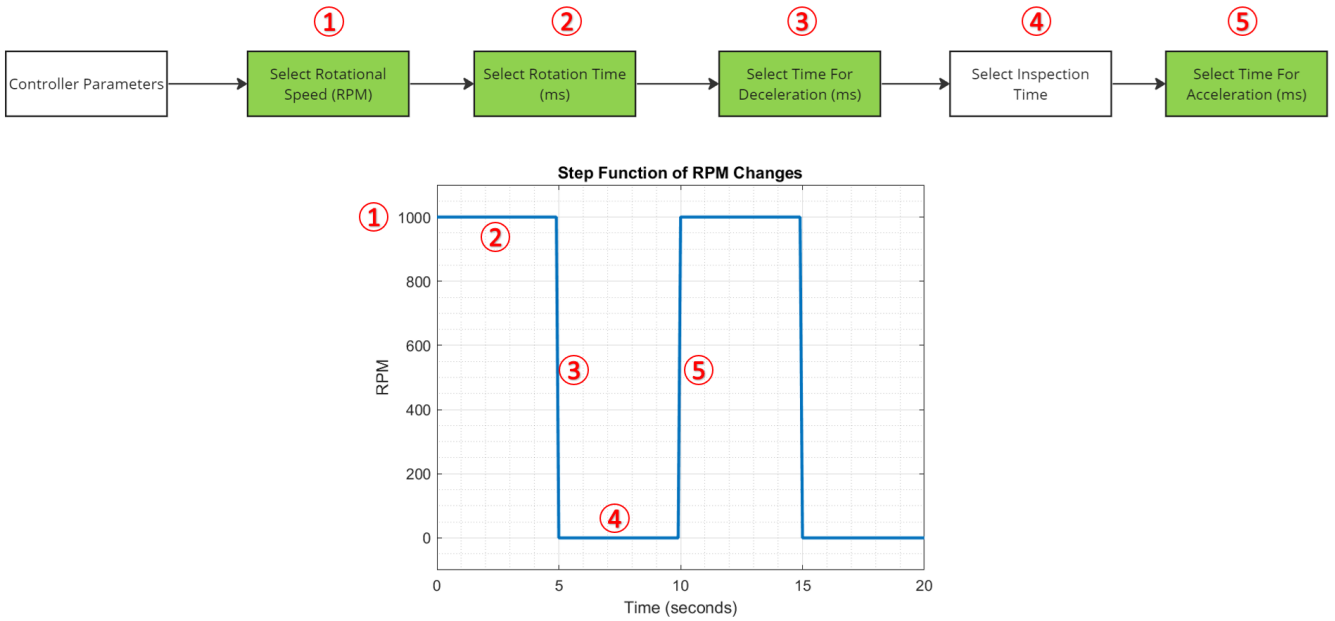


Figure 105: Controller parameters and the step function needed to create a spin-profile for testing.

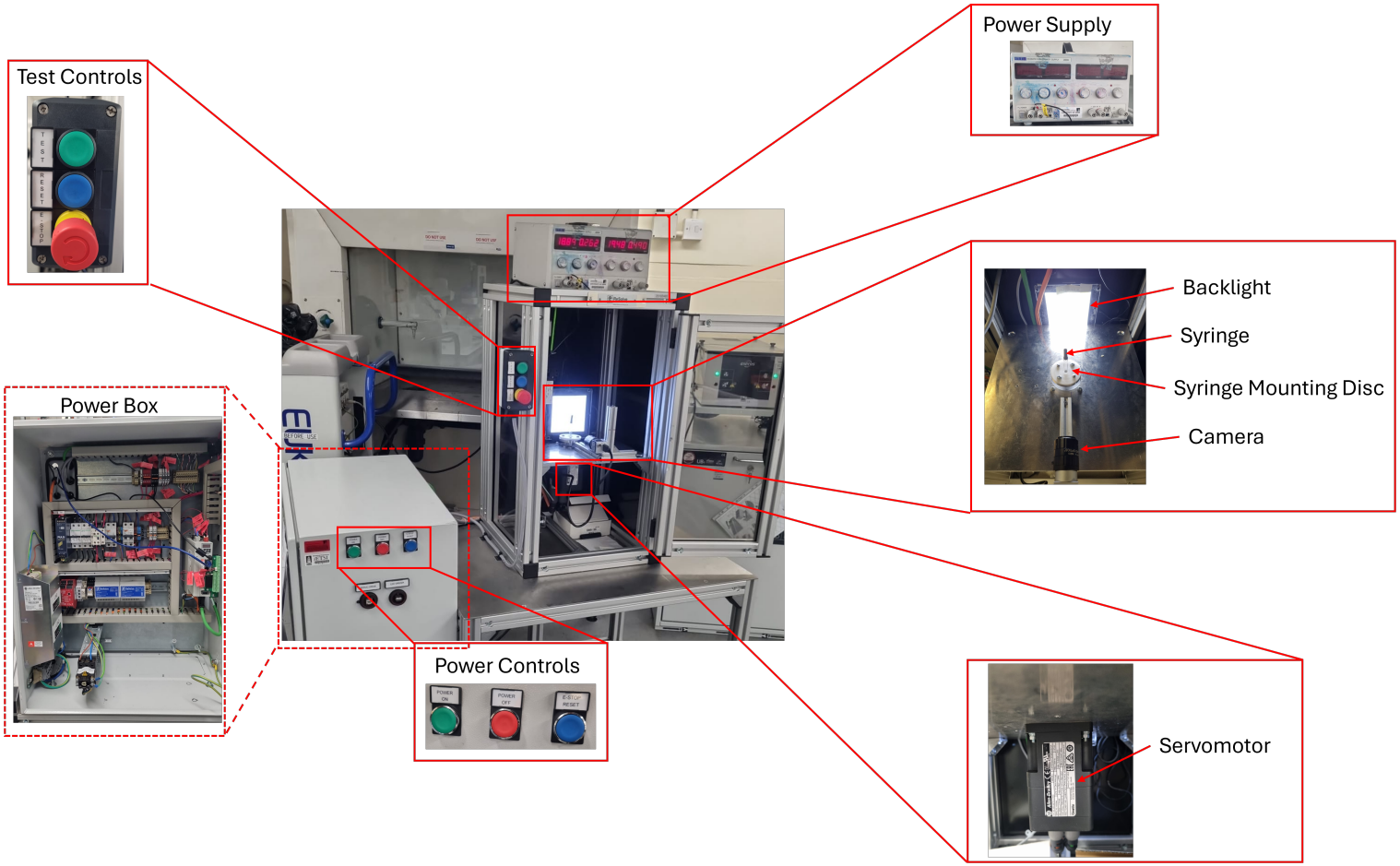


Figure 106: Experimental test rig setup containing the essential equipment needed to run a typical pre-filled syringe inspection.

6.3 Syringe and Particle Preparation

To prepare a syringe for testing, various standard processes are developed and need to be completed for each syringe. For the experiment, a syringe needs to pass through four different stations, each station being set to ensure a particular objective. These are "Syringe Preparation Station", "Fluid Preparation Station", "Particle Station", and "Cutting Station". A summary of the different stages at each station is provided in Figure 107 and detailed processes are discussed later.

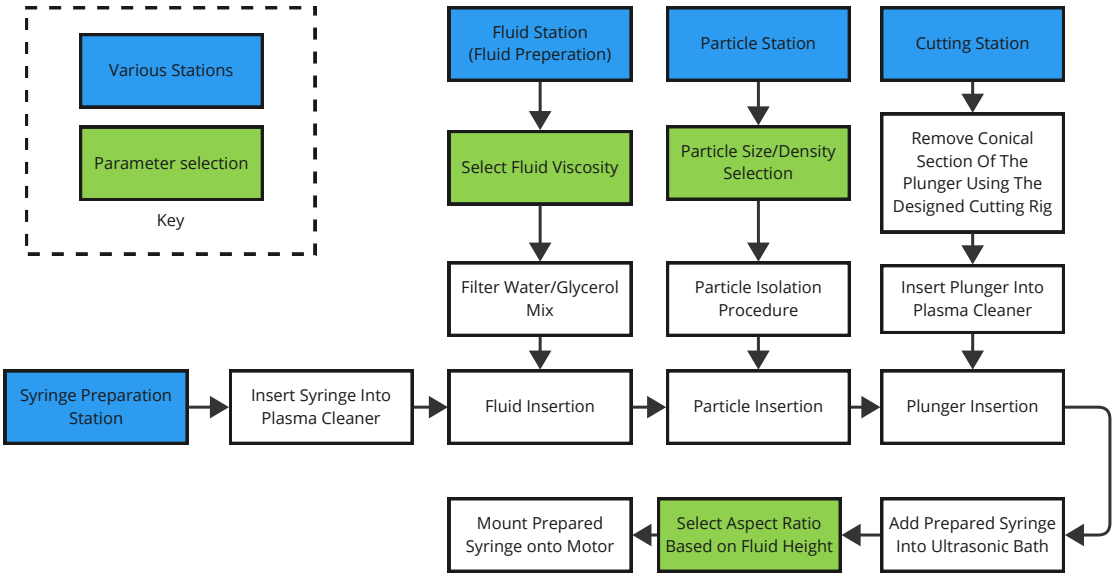


Figure 107: A summary of the different stages a syringe goes through at each station where blue boxes refer to stations and green boxes refer to parameter selection.

6.3.1 Fluid Station

At the fluid station, water and glycerol mixtures are created to produce fluids of varying viscosity that resemble typical PFS fluids. For the experiments, fluids of viscosity 1 cP, 9cP, 18 cP, 23 cP are produced as these cover the range of viscosities used in pre-filled syringes. The fluid density range from $1000\,\mathrm{kg\,m^{-3}}$ to $1150\,\mathrm{kg\,m^{-3}}$ and industry grade de-ionised water is used and further filtered to ensure no contaminants are present in the water. The purpose of the filling station is to ensure a clean mixture at a desired viscosity is produced so that it can be used for testing.

The working fluid is prepared by selecting a target viscosity and determining the corresponding water–glycerol ratio at a controlled laboratory temperature of 25 °C. To minimise contamination, all equipment is carefully cleaned prior to use. A beaker is rinsed with isopropyl alcohol and then with deionised water. The required amount of glycerol is first transferred into the beaker using a syringe fitted with a 0.2 µm filter. The corresponding volume of deionised water is then added, and the mixture is gently stirred until a uniform solution is obtained. Throughout the preparation, care is taken to avoid unnecessary contact with external surfaces and to minimise agitation, reducing the risk of introducing particles or air bubbles.

Once prepared, the mixture is transferred into a sealed container and stored in a temperature-controlled environment until required. The process is repeated for different water–glycerol ratios to obtain fluids across the desired range of viscosities.

6.3.2 Particle Station

The purpose of the particle station is to ensure that a single particle is isolated and stored away until it is needed for final assembly into a syringe that is needed for inspection. The particles used for the experiments are supplied by Cospheric, and are polyethylene microspheres. There are three range of spherical particle sizes purchased for testing, these are sized as (106 µm to 125 µm), (180 µm to 212 µm) or (250 µm) respectively. These particles have a density of either 1.0 g cm^{-3} or 1.2 g cm^{-3}. A sample image of an orange microsphere particle is shown on Figure 108 as a small orange dot.

Figure 108: Sample image of a syringe with a orange spherical microparticles

Microparticles of sizes 106 µm to 125 µm are first selected. To minimise contamination, a container is cleaned using isopropyl alcohol before a small quantity of particles is placed into it along with a controlled volume of deionised water.

Using a micropipette in conjunction with a microscope, a single particle is carefully isolated within a small volume of deionised water and transferred into a Falcon tube for storage. Care is taken to ensure that the particle is positioned close to the centreline of the tube, reducing the likelihood of it adhering to the walls.

The isolated particle is then stored in the Falcon tube with the small volume of deionised water until final assembly into the syringe. This procedure is repeated for particles of sizes 180 µm to 212 µm and 250 µm.

6.3.3 Cutting Station

The cutting station is used to remove the conical section of the plunger to form flat-top rubber plungers. This step is essential, as all the previous work focusses on flat plunger geometries. It has also been established that a conical plunger changes the particle track in comparison to a flat top (from Chapter 5, Conical plunger Study). Figure 109 shows a schematic of the mould used to cut the plungers consistently.

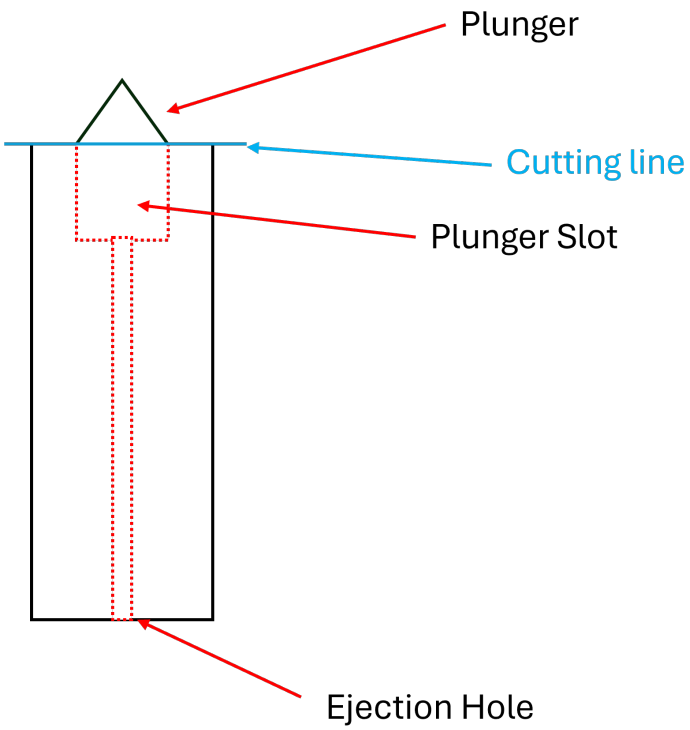


Figure 109: A schematic showing the mould used to cut plungers with consistency.

The plunger is placed into a cutting mould, with the cylindrical section aligned within a precision-fit slot to ensure stability during the cutting process. The plunger is then checked to confirm that it is securely positioned, preventing any movement. It is arranged such that the conical section protrudes from the mould.

Using disposable gloves, a scalpel is dipped in a surfactant and used to carefully remove the protruding conical rubber section. The cut is performed along a predefined cutting line using a steady and controlled motion, ensuring that the conical section is removed while preserving a flat cylindrical surface at the top. Appropriate care is taken when handling the blade to ensure safe operation.

Following the cut, the plunger is removed from the mould using a small rod inserted through a hole located behind the plunger slot, allowing for controlled and safe ejection.

6.3.4 Syringe Assembly

Once the fluid is prepared, the particle is isolated and the plunger is cut flat, the final syringe can be prepared and mounted. The syringe is first cleaned by rinsing with isopropyl alcohol using disposable gloves, followed by a further rinse with filtered deionised water. Compressed air is then used to remove any residual liquid and provide an initial clean. The syringe is then placed in a plasma cleaner to remove any remaining organic and inorganic contaminants from the surface.

Once thoroughly cleaned, the syringe is filled with the prepared test fluid. The filling process is carried out slowly from the bottom up to minimise the formation of air bubbles. The test particle, previously stored in a Falcon tube, is then drawn into a micropipette and carefully introduced into the syringe. Care is taken to ensure that the particle is positioned along the centreline, reducing the likelihood of it sticking to the syringe walls.

The syringe is subsequently sealed using the prepared plunger, and the fluid height is adjusted to achieve the desired aspect ratio. The assembled syringe is then placed in an ultrasonic bath to degas the fluid, reducing the presence of any microbubbles that may be present.

Finally, the syringe is mounted onto the motor using a custom-designed holder with two reference pins and securing screws, as shown in Figure 110. The mounting system ensures alignment of the syringe along its central axis. The syringe is positioned carefully within the mount to avoid contamination, and the screws are tightened using Allen keys to provide a stable and uniformly rotating setup.

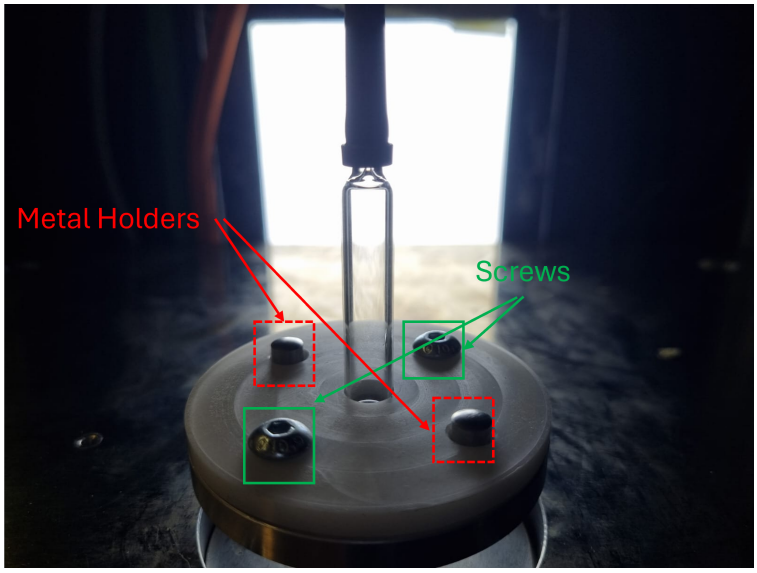


Figure 110: A close-up image showing the screws and metal holders used to mount the syringe onto the motor.

Note
Throughout the process, ensure the use of gloves to maintain a clean environment. After the insertion of the particle and plunger, tap the syringe repeatedly. The tapping action aims to dislodge and remove any air bubbles present in the syringe. This step is a preliminary measure, and the complete removal of micro-bubbles will be ensured using the ultrasonic bath.

6.4 Experimental Procedure

Once a syringe is prepared and mounted,

1. Turn on the power to the power box using the green “Power On” button and the power supply for the backlight using the power supply switch.
2. Open the KNX5100C Software to set spin-profiles to the test conditions.
3. Open the Pylon viewer software and ensure that the camera is ready to record and the camera settings are as described in the previous section.
4. Start the camera recording in Pylon Viewer and verify the ability to record continuously for at least 10 minutes.
5. Sync the experiment time with a stopwatch to keep track of the time by pressing the stopwatch and the green “Test” button using the test controls at the same time.
6. Once the experiment time has reached a pre-defined time, stop the recording on Pylon viewer and press the blue “Reset” button using the test controls at the same time.

7. Once the data video is recorded, save it to a collection folder renaming the video to reflect the experiment conducted.

8. Conduct all the tests needed for a particular syringe and remove the syringe using the Allen key.

9. Remount the next test syringe and repeat Steps 1 to 8 until all syringes have been tested.

6.5 Post Processing

The post-processing section explains how a video recorded during the experiment is processed to obtain particle data. To give a better context to the reader and help in reading this section, Figure 111 shows two typical images with the particle manually highlighted. The image on the left is a mounted syringe showing the camera-view where as the image on the right is an unmounted syringe showing similar features. The plunger, test particle, and air bubbles are also highlighted in the image. They are not the same syringe, but are provided to aid the reader to understand how the prepared syringes look before and after being mounted. The aim of this section is to covert the movement of these particles through various techniques to analyse particle tracks.

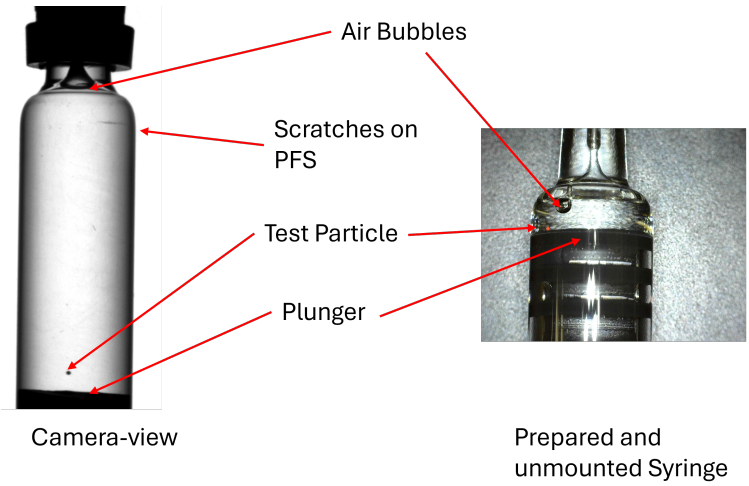


Figure 111: Two typical images of examples of prepared syringes where one (on the left) shows the camera-view for a mounted syringe and the other (on the right) shows an unmounted syringe

To obtain particle track data, various steps are as follows. The data of interest are five parameters, namely;

1. Settling time (s) - Chosen to understand how long a particle takes to settle (i.e. stop moving)

2. Max height change (mm) - To understand how much vertical travel a particle may have during inspection

3. Maximum projected radius (mm) - To calculating how wide a particle is detected.
4. Projected path length (mm) - To find the total projected travel distance (larger distance has more movement within inspection window hence more likely to be detected).
5. The number of revolutions - To understand how many times a particle may rotate about the central axis.

In this section, each post processing step is explained.

6.5.1 Conversion to Tiff

The initial step is to extract the images from the raw inspection videos. Using video FPS, the MATLAB script (Conversion_ code.m) allows the conversion of multiple video files within a folder to a sequence of TIFF frames. This can be done in bulk if the folder contains multiple video files. The result consists of a folder per video, where all files are converted into TIFF frames, each with the naming convention FileName_Frame0001.tiff. (See the full code provided in the Appendix 9.1 for more details)

6.5.2 Sorting Relevant Test Image Frames

A second MATLAB script is used to sort the large number of TIFF frames into their respective test runs. This is done by calculating the TIFF frame numbers at which

1. Spin-up starts,
2. Spin-up ends and spin-down begins (this is also when inspection begins)
3. Next spin-up (for Test run 2) starts,
4. And so on

Once the frame numbers have been calculated, only the frames relating to the spin-down/inspection of each test run are kept. The spin-up frames are deleted to save computational resources. With the camera used in this particular experiment, the particles move too fast to capture any relevant data outside of the inspection time frame. The frames relevant to the inspection window are then sorted into folders with their test run number. Once these test run folders have been sorted, the next script can be used to learn more about each run. The code used for this is the (Run_ Sort.m) presented in the Appendix 9.2.

6.5.3 Background Subtraction Algorithm

The folders that have already been sorted into their run folders can further be processed. This is achieved through a background subtraction method. The goal of using the background subtraction method is to extract two data sets. These are the x-position (projected radius) vs time and the y-position (height) vs time. This data can be used to perform a statistical analysis of particle motion under various conditions. The background subtraction method

can detect moving particles in each frame of the video and form links for the moving particles between each frame to show the movement of the particles.

A background subtraction method works by isolating moving particles from a static image (i.e. the background image). For this particular work, the final image is used (once the particle settles) as it is an ideal background image. Each frame is then compared with the background image of each test run through pixel-wise subtraction. Pixels that are sufficiently darker than the background are marked as foreground; all others are marked as background. If the image is darker than the background (subject to a selected threshold), it is a particle; otherwise, there is no change. A one-sided threshold is used where 0 refers to black and 255 refers to white. This is then used to produce a clean greyscale image through conversion to a logical mask, where the particle is white against a black stationary background. The resulting mask is cleaned through morphological opening and closing to remove specks, smooth the image, and fill small gaps, yielding a single blob for the particle (Russ and Neal, 2018). To help determine the location of various objects within the domain, a datum is set to the plunger. Using this datum, one can determine the relative height above the plunger or the projected radial position from the centre of the plunger. This is then used to express the centre of the particle in the frame. This process is repeated for each frame until the particle settles, and the Crocker–Grier algorithm by Crocker and Grier (1996) is used to form a single particle trajectory. The Crocker–Grier algorithm makes a list of possible trajectories between all the particle detections through and all the frames and picks the path by minimising the total distance travelled. It also works when particles temporarily disappear. The final output of this method is a video file showing a video of the track in real-time and a text file containing the data for the projected radius, height, and time. The set of codes used for this are provided in Appendix 9.3.

6.5.4 Data clean-up

Particle tracks under the same conditions generally have similar behaviour. To get a good idea of the statistical behaviour of particles, at least 20 particle tracks with the same conditions are exported. The data exported from the background subtraction method does not always produce a clean particle track. Sometimes, if the background image contains a noisy 'spec' of data (false particle) that is misidentified as a particle, each frame interpolates between its particle position and the false particle on the background image. A false particle can also be detected if the threshold is not optimal, and between each frame, the trajectory linking between each frame can fail. In these cases, the particle track can still be extracted and needs to be cleaned up first. A tool that can delete invalid data points while maintaining the time record of each track has been created precisely to fine-tune any particle tracks and ensure that the background subtraction method has not failed. Figure 112 shows a screenshot of the graphical user interface for the tool and here is how the tool works;

1. "Select Main Folder" is initially selected, containing subfolders relating to the particle data in text files exported from the background subtraction method. A drop-down on the right side of the button can pick the individual subfolders to process multiple studies simultaneously.

2. "Available file" section shows the detected text files within a particular selected sub-folder.

3. The figures on the right are interactive and can be used to select any points that need to be deleted. The same can be done by selecting or using the row index. The final tracks are saved for final analysis. The false particle points are identified using the videos files generated during the background subtraction method.

4. Once the desired points are deleted, the "Save Files & Plots" button exports the final text files that have been cleaned up to represent the real particle track.

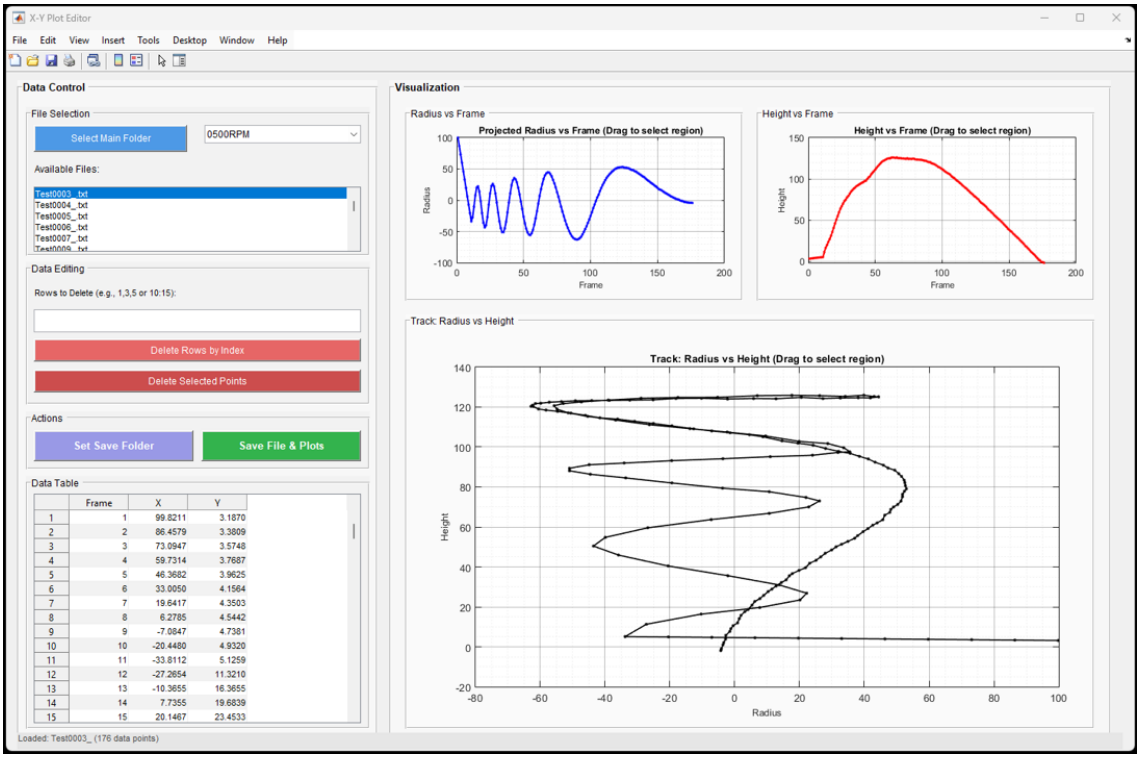


Figure 112: Screenshot of the graphical user interface for the data clean-up tool showing the various features discussed.

This tool can be used to select a false particle location and delete any location (projected radius and height) positions at this point. The frame number is kept to ensure that the time increments are not missed. The files exported using this tool are ready for track analysis. From the collected data, at least 20 tracks are exported to ensure that the statistical analysis can encapsulate the particle behaviour. In some cases, 20 particle tracks could not be exported because there are not enough reliable particle tracks processed by the background subtraction methods. These could be due to a number of reasons. For example, if the lighting is not suitable, the imaging threshold may produce false tracks. If a second particle or artifact is found in a frame, the threshold may link the track of one particle to the other. These would lead to false particle detection that does not follow the particle track. The MATLAB code for the particle track clean-up is provided in Appendix 9.4 filename "XYPlotEditorGUI.m"

6.5.5 Analysis_tool.m

A multi-tab MATLAB tool has been developed to analyse the data obtained during the experiments. It works by loading the folders (study) containing subfolders (parameters tested) with final text files (position data). Tab 1 within the tool calculates and generates tables for variables such as settling time (s), max height change (mm), max projected radius (mm), projected path length (mm) and number of revolutions. These averages are calculated across all files within the subfolder. Figure 113 shows an example data set for 500 RPM calculated for 20 different test runs. Other useful information within the selected dataset is also provided in the Data summary section on the bottom left. The calculations depend on the calibrations performed on the basis of the frames per second (fps) of the recorded data and the millimetre to pixel conversion. For this experiments, 1 mm equals 43 pixels.

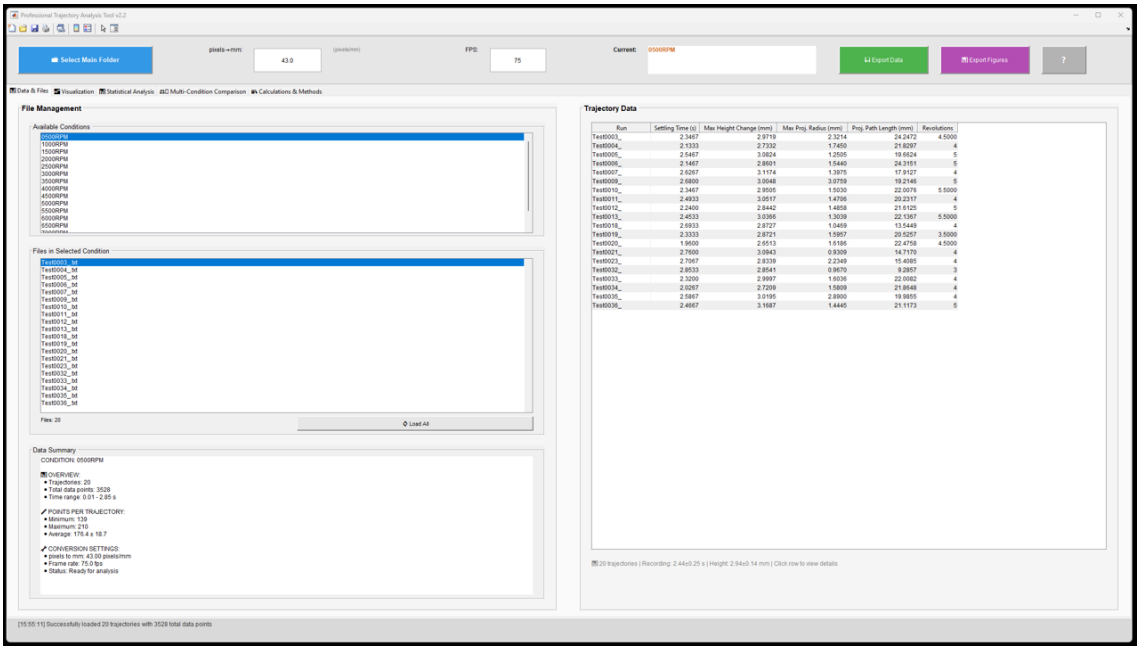


Figure 113: Screenshot of the graphical user interface for Tab 1 showing an example trajectory data calculated across various test runs.

Tab 2 of the tool is used as a visual aid, showing various interactive figures related to the loaded data. One of these figures is the all-trajectory overview (projected radius vs. height), which shows all the particle tracks loaded within a subfolder. It also highlights the current selection for a particular test run, highlighted in yellow against a backdrop of all other particle tracks. This is a useful figure for comparing runs within the same data set.

For a selected test run, three other figures are also produced. These are the project radius vs time, height vs time, and the projected radius vs projected height. These are useful for visualising one particular track and its temporal evolution. A screenshot of Tab 2 for the same dataset as above is presented in Figure 114.

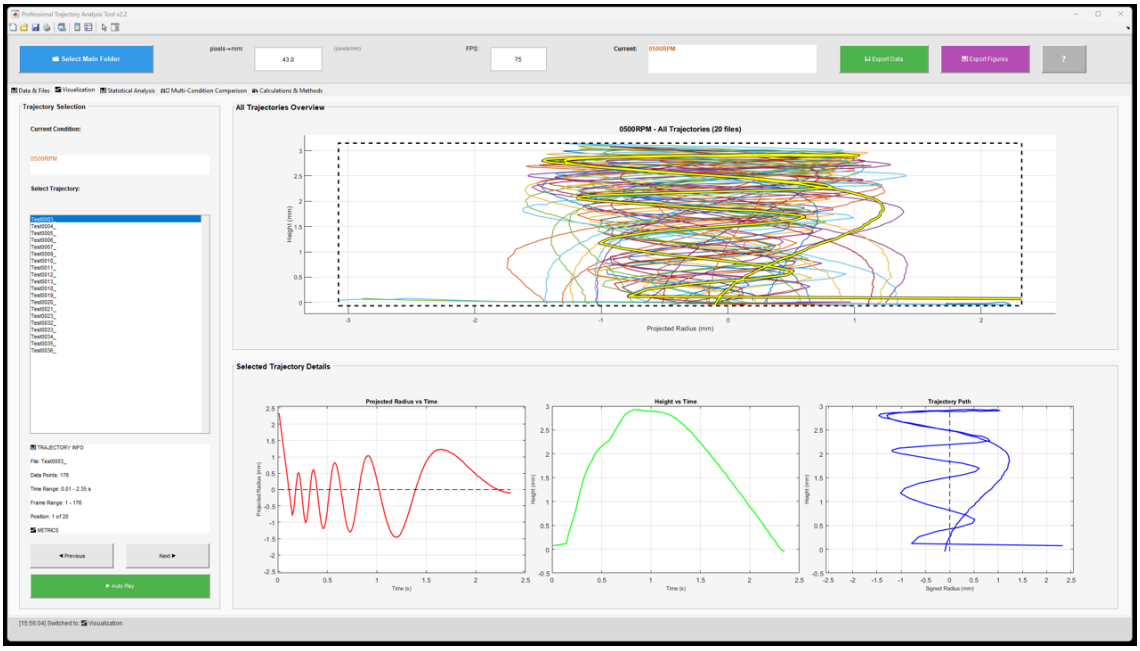


Figure 114: Screenshot of the graphical user interface for Tab 2 showing the trajectory data for the 500 RPM test runs.

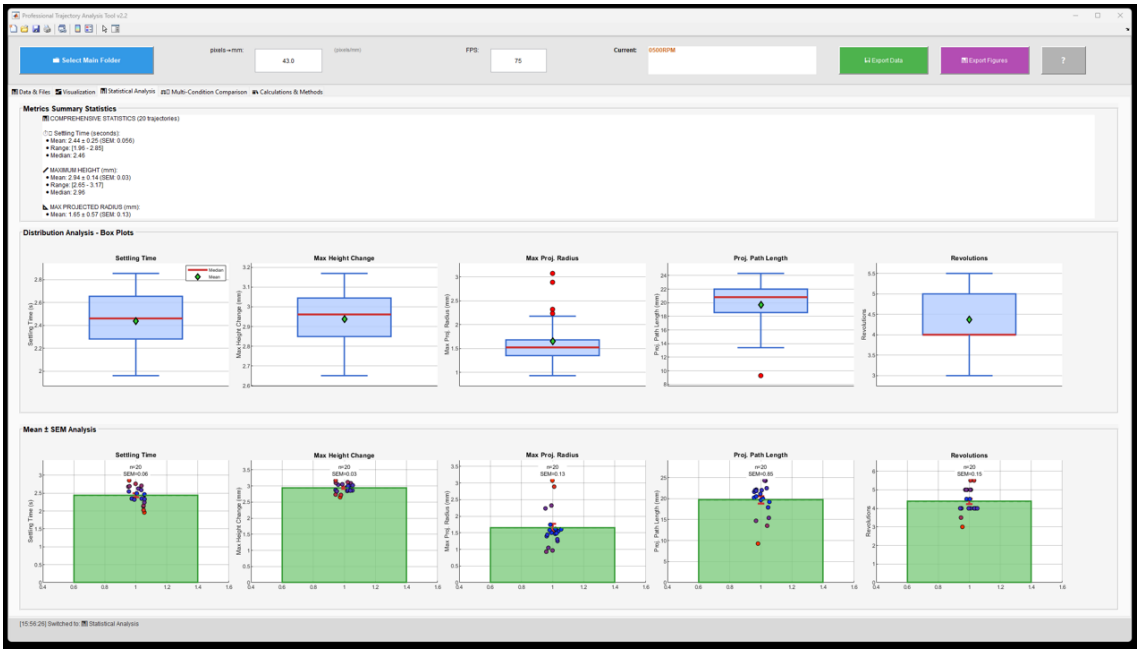


Figure 115: Screenshot of the graphical user interface for Tab 3 showing the statistical data for the 500 RPM test runs.

Tab 3 is used to calculate and display statistical data within a selected subfolder. The variables calculated shown in the table on Tab 1 are presented here as box plots and mean ± SEM plots (standard error of mean). For comparison of this data to other parameters within the same study (i.e. 500 RPM vs 1000 RPM), Tab 4 (Figure 116) presents the multi-condition comparison plots. These are the same five parameters presented for each parameter in the study. All this data can be exported and saved using the tool.

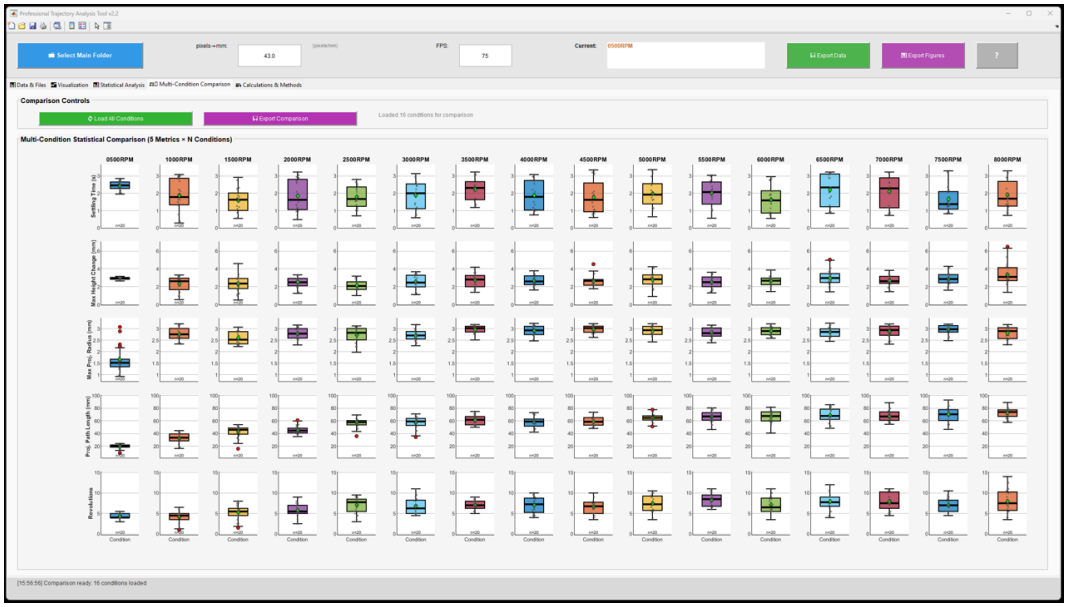


Figure 116: Screenshot of the graphical user interface for Tab 3 showing the multi-condition comparison tab with the five parameters from the trajectory data.

The five parameters exported are settling time (s), max height change (mm), max projected radius (mm), projected Path length (mm), and number of revolutions. In reality, the settling time (s) is the time taken for the particle to stop moving. In this particular case, it is the time recorded in the final frame for each test run within the video. For the majority of the studies, the particles settle well before the next run and the data is cut when the particle stops, hence the final frame time is the settling time. The max height change (mm) is calculated as $H_{max} - H_i$ where H_{max} is the highest vertical position of the particle during its trajectory and H_i is the initial height recorded within a test run. The max projected radius (mm) is the maximum radius observed by the camera. All these measurements are from an origin point fixed at the top centre of the plunger, as seen from the point of view of the camera.

The projected path length, L, is the total distance the particle travels across the trajectory. It is not the true distance as there is no data in the third dimension (z). This is calculated using:

$$L = \sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

where x_i and y_i are the initial coordinate of the particle from the centre top of the plunger and x_{i+1} and y_{i+1} are the next position of the particle.

The number of revolutions are estimated using three different methods. A zero-crossing method where the number of times zero (origin) is crossed in the radial direction. The number of times zero is crossed is halved to estimate the number of revolutions. This method underestimates when trajectory data flatlines (damped motion). A second method to calculate the number of revolutions is by counting the number of times the direction changes from positive to negative to positive to negative. This count is then divided by 4. This method overestimates the number of revolutions especially if there is a lot of noise in the data. Finally, the last method of estimating the number of revolutions is the peak-trough method. One complete revolution is one peak and one trough. Therefore, the number of revolutions can be estimated using number of $(peak + through)/2$. This method can both overestimate and underestimate the number of revolutions of the threshold is either too low or high or if the data has micro peaks. Therefore, all the methods above are calculated and the median across the three methods is used to estimate the number of revolutions.

6.6 Experimental Plan

Table 8 shows the list of syringes needed to carry out the experiments planned in this section.

Table 8: Particle and Fluid Properties Used in Syringe Experiments

Syringe	**Average Particle Diameter (µm)**	**Particle Density (g/cm³)**	**Fluid Viscosity**	**Aspect Ratio**
1	180	1.2	1 cP	4.1
2	106	1.2	1 cP	3.9
3	250	1.2	1 cP	3.96
4	106	1.0	1 cP	4.01
5	180	1.2	9 cP	3.80
6	180	1.2	18 cP	3.96
7	180	1.2	23 cP	4.0

The independent variables in the studies are the geometry of the fluid environment (specified by aspect ratios), the fluid viscosity, the RPM, and the particle sizes (106 µm, 180 µm, 250 µm). The dependent variables are particle motion (trajectory, settling locations), max height reached by particles, the settling/transit time of particles, and the number of revolutions completed by particles. The planned experiments for various studies are presented in Table 9. The results obtained using this entire apparatus are given and discussed in Chapter 7.

Table 9: Variable Conditions and Constants for each planned study

Variable Conditions	**Constants**
RPM Study • RPM: 500–8000 (step 500)	• Density: 1.2 g/cm^3 • Viscosity: 1 cP • Aspect Ratio: 4 • Particle Diameter: 180 µm • Head Space: 0
Particle Diameter Study • Particle Diameter: 106, 180, 250 µm	• RPM: 1000 • Density: 1.2 g/cm^3 • Viscosity: 1 cP • Aspect Ratio: 4 • Head Space: 0
Particle Density Study • Particle Density: 1.0 and 1.2 g/cm^3	• RPM: 1000 • Particle Diameter: 106 µm • Viscosity: 1 cP • Aspect Ratio: 4 • Head Space: 0
Fluid Viscosity Study • RPM: 3000, 5000, 7000 • Viscosity: 1, 9, 18, 23 cP	• Particle Diameter: 180 µm • Density: 1.2 g/cm^3 • Aspect Ratio: 4 • Head Space: 0
Aspect Ratio Study • Aspect Ratio varied	• RPM: 1000 • Particle Diameter: 180 µm • Density: 1.2 g/cm^3 • No Headspace • Viscosity: 1 cP
Head Space Study • Aspect Ratio and Head Space varied	• RPM:3000 • Particle Diameter: 180 µm • Density: 1.2 g/cm^3 • Viscosity: 1 cP

7 Experimental Results

In this chapter, the final results exported from the analysis tool for the data obtained from the experiments are presented and discussed. There are six different studies testing different parameters and their impact on the typical track of a particle. These are the studies with the variable conditions presented in Table 9 in Chapter 6. The results obtained in this chapter are processed from particle tracks under various conditions. An example of a typical track is shown on Figure 117.

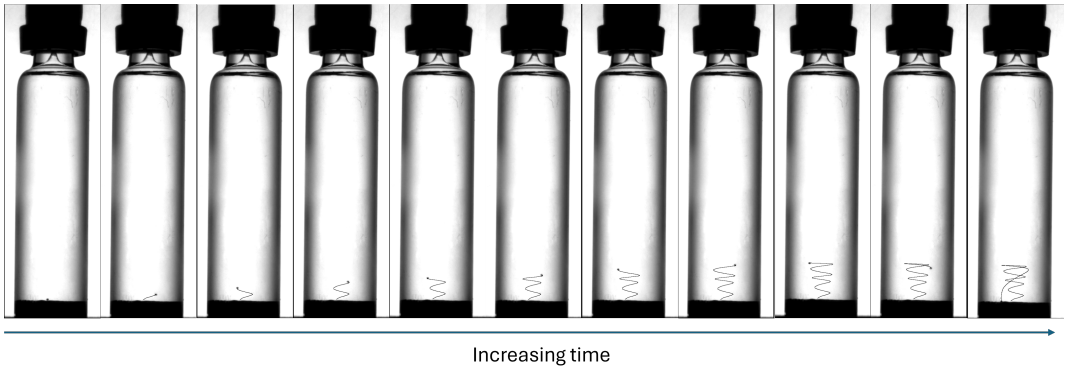


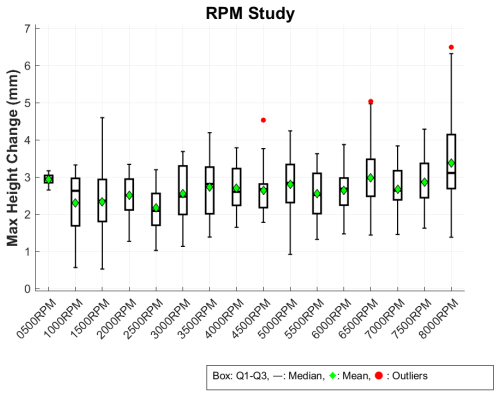
Figure 117: A typical particle track observed after spin-down. The screenshots from left to right show the progression of time within the inspection window.

7.1 RPM Study

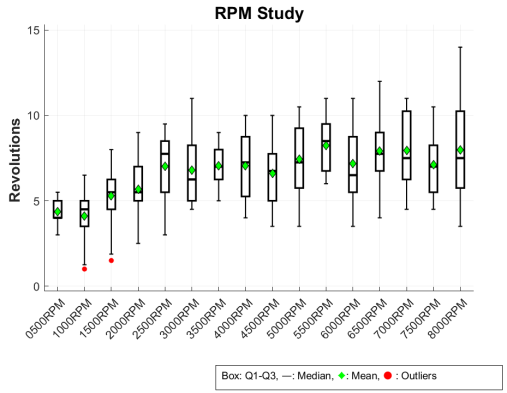
In this study, the maximum rotational speed prior to stop (RPM) is varied starting at 500 RPM ($Re \approx 182$) with increasing increments of 500 RPM ($Re \approx 182$) up to 8000 RPM ($Re \approx 2914$). Each test is repeated 20 times to obtain an average particle track for each condition. The purpose of the study is to analyse the impact the spin-speed has on the particle track. Figures 118a, 118b, 118c, 118d and 118e show the maximum height change, the number of revolutions the particle makes, the settling time, the maximum projected radius, and the projected path length, respectively. The maximum height change across the various RPMs shows a mean between 2 to 4 millimetres. As the RPM increases, the mean maximum height does not show a particular trend, but the range of the collected data (shown by the minimum and maximum values) increases. For the 500 RPM case ($Re \approx 182$), the error bars are very small and this applies to the other parameters too. However, there is a large increase in the data range from 1000 RPM onwards. This is due to larger momentum of the fluid during the inspection window. Moreover, some particles are observed to remain on the plunger and not lift as the momentum is not large enough to cause fluidisation at 500 RPM.

As the max rotational speed increases, the number of revolutions the particle makes as well as the projected path length, also gradually increase. This is expected as the fluid has larger momentum with the higher RPM speed, the particle have higher inertia. However, the mean settling time remains fairly consistent between 1.5 and 2.5 s. The range of settling times is also fairly consistent. This suggests that although there is higher inertia, the fluid and the particles come to rest around the same time, irrespective of the rpm. This is in line with the particles reaching largely the same height, and gravity dominating the settling as the vertical fluid motion dissipates quickly. The maximum projected radius across most studies is expected to remain the same as all particles are pushed radially towards the outer walls. All RPMs except 500 RPM ($Re \approx 182$) remain fairly consistent with respect to the maximum projected radius seen near the outer walls. The 500 RPM ($Re \approx 182$) case does not have enough acceleration to reach the outer walls near 3 mm.

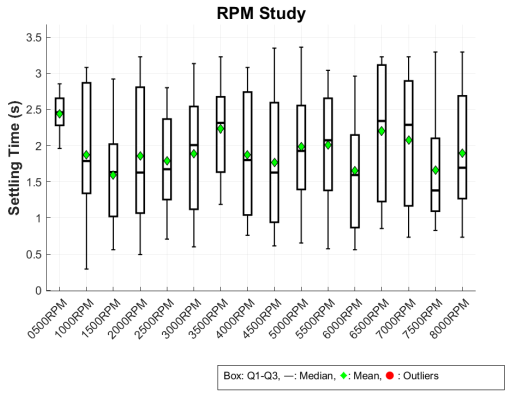
In summary, the change in RPM affects the maximum height recorded, the number of revolutions, and the projected path length, but does not impact the settling time, or the maximum projected radius (with the exception of the 500 RPM case). Note that flow becomes unstable at high RPMs, reaching transitional or turbulent regimes. For this aspect ratio, turbulence is expected around 2000 RPM (Re $\approx$ 728) from Figure 53 from Chapter 4. This may be a possible explanation for the increased observed error bars.



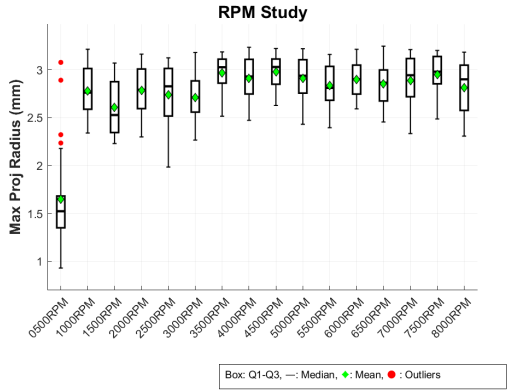
(a) Max Height Change(mm)



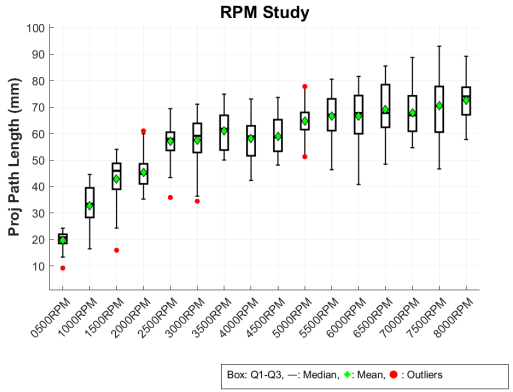
(b) Number Of Revolutions



(c) Settling time (s)



(d) Maximum Projected Radius (mm)



(e) Projected Path Length (mm)

Figure 118: RPM study results for various parameters for aspect ratio of 4.0 and 1cP fluid viscosity.

7.2 Particle Diameter Study

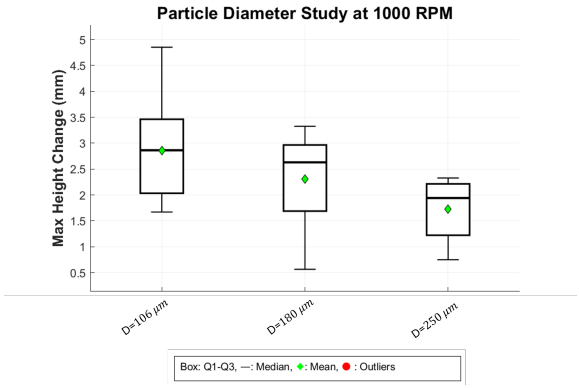
In this study, three particle diameters (106 µm, 180 µm and 250 µm) are varied and their resulting particle trajectories are analysed. Figures 119a, 119b, 119c, 119d and 119e show the change in the maximum height, the number of revolutions, the settling time, the maximum projected radius, and the projected path length, respectively.

As the particle diameter increases, there is a decrease in the change in height that a particle observes (in mm). The smallest particle diameter (106 µm) has a mean maximum height change of almost double that of the largest particle (250 µm). The smaller particles have lower inertia and match the fluid velocity more quickly, whilst the larger particles require more inertia. Since the velocity field is dissipating, the height reached by the larger particles is reduced. This is also expected as the smaller particles are lighter and can be lifted much more easily than larger diameter particles (with a constant particle density). The range of the change in maximum height recorded decreases with increasing particle diameter. For example, the range of the change in height of 106 µm is around 4 mm, where as the largest particles diameter of 250 µm has a range of around 2 mm.

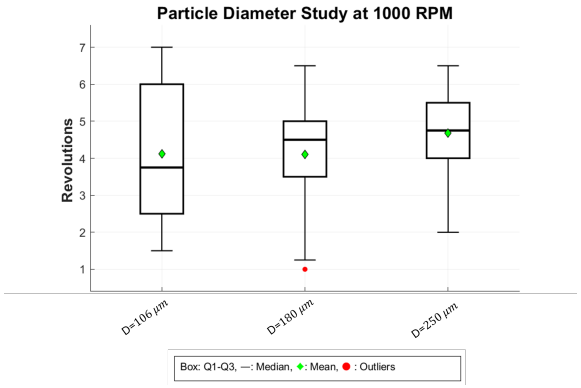
The number of revolutions for the three particles show a mean between 4 and 5 revolutions. Although the mean is roughly the same, the range of the data recorded is significantly tighter for the larger diameter particle, i.e. the interquartile range (IQR) of the 250 µm particle is around 1.5 where as the 106 µm particle has an IQR of 3.5. This means that from the 20 particle tracks observed for the larger diameters, there is less variation in the number of revolutions.

The mean settling time recorded for the 106 µm particle is around 3s. For the 250 µm this is almost half at 1.5s. The range of the data is also much smaller for the 250 µm particle just like the maximum height change data. The smaller particles also show a smaller maximum radius, suggesting that they remain close to the centre of the syringe, whereas larger particles are accelerated much closer to the outer walls (at maximum radius). The projected path length for all particles is around 35 mm at 1000RPM and no significant trend is observed.

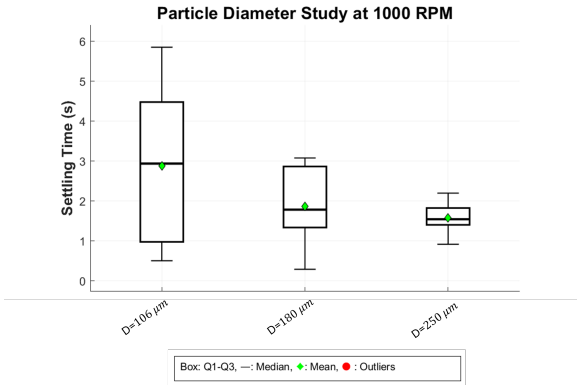
This suggests that smaller particles are in motion for a lot longer on average and also show larger height changes. They also have larger ranges for the same parameters in comparison with larger diameter particles.



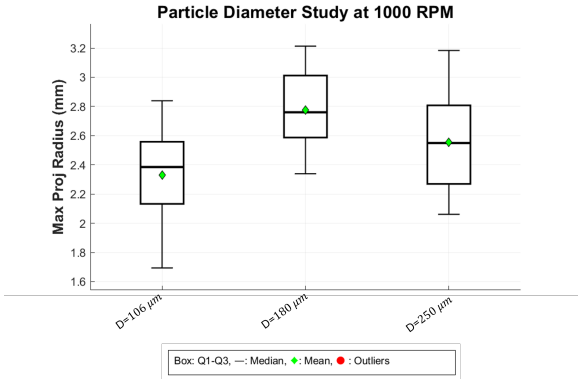
(a) Max Height Change(mm)



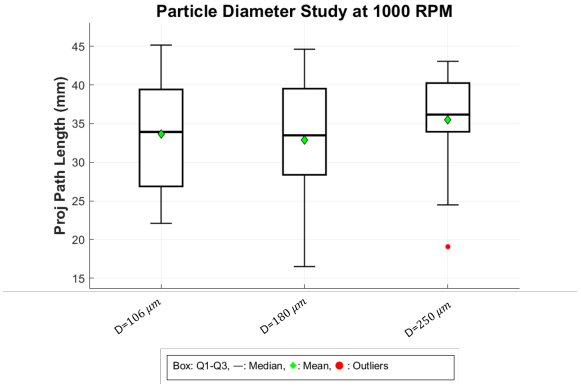
(b) Number Of Revolutions



(c) Settling time (s)



(d) Maximum Projected Radius (mm)



(e) Projected Path Length (mm)

Figure 119: Particle diameter study for particle sizes of 106 µm, 180 µm and 250 µm at aspect ratio of 4.0 and 1cP fluid viscosity.

7.3 Particle Density Study

The particle density study is slightly different from the other studies as it compares two different particle densities. One of these is $1.0\,\mathrm{g\,cm^{-3}}$, where the particle density is comparable to the fluid density (0.998 kg/m^3 at 25 °C) , and $1.2\,\mathrm{g\,cm^{-3}}$, where the particle is denser than the fluid. The behaviour of particles with the two densities varies even before any spin profile is applied. The $1.0\,\mathrm{g\,cm^{-3}}$ particles do not settle at the bottom; they are observed across the domain before any spin has occurred. On the other hand, the $1.2\,\mathrm{g\,cm^{-3}}$ particles almost always settle at the bottom of the plunger at the end of the test run. Therefore, all results shown in this study should be strictly taken into account for these two densities. Due to a lack of time, only particles with $1.0\,\mathrm{g\,cm^{-3}}$ and $1.2\,\mathrm{g\,cm^{-3}}$ were studied. It may be relevant to study the increase in particle densities that are larger than $1.0\,\mathrm{g\,cm^{-3}}$ in one instance and particles less than $1.0\,\mathrm{g\,cm^{-3}}$ in a separate study where the critical case is the $1.0\,\mathrm{g\,cm^{-3}}$. It is not known how the data changes if a density of less than $1.0\,\mathrm{g\,cm^{-3}}$ is tested. Figures 120a, 120b, 120c, 120d and 120e show the change in the maximum height, the number of revolutions, the settling time, the maximum projected radius, and the projected path length, respectively.

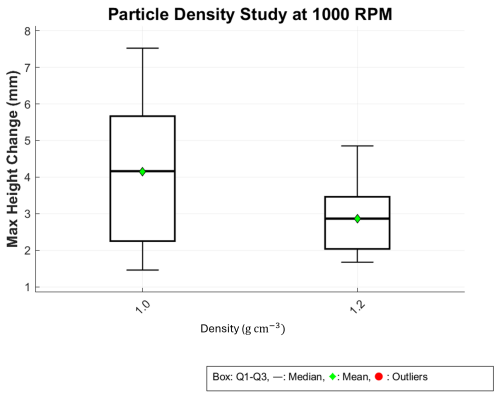
The maximum height decreases with increasing density of particle. The range of maximum height changes detected for particles reduces from 6 mm (at a density of $1.0\,\mathrm{g\,cm^{-3}}$) to 3.5 mm. The mean maximum height change also reduces from 4 mm to approximately 3 mm. The data also shows a large spread for the $1.0\,\mathrm{g\,cm^{-3}}$ shows inconsistent change in height compared to the $1.2\,\mathrm{g\,cm^{-3}}$. This is expected as the particles with density $1.0\,\mathrm{g\,cm^{-3}}$ is similar the fluid density and these particles are less affected by external forces such as weight. Particles with a density of $1.2\,\mathrm{g\,cm^{-3}}$ also start at the bottom of the domain where the plunger is placed in comparison to $1.0\,\mathrm{g\,cm^{-3}}$ which can be anywhere within the fluid volume.

The number of revolutions shows roughly the same mean value of 4.0 revolutions but with a much larger spread for the case of $1.2\,\mathrm{g\,cm^{-3}}$. The $1.0\,\mathrm{g\,cm^{-3}}$ shows a range of 3.5 revolutions compared to the particle $1.2\,\mathrm{g\,cm^{-3}}$ showing a range of 5.5 revolutions. A similar trend is seen with the settling time. Particles with a density larger than the fluid shows a much larger spread of results for the settling time, although both densities have a mean settling time of around 3 seconds. The large spread in settling time could also be a result of the larger spread of data seen for the number of revolutions. As the number of revolutions increases, particles are suspended longer.

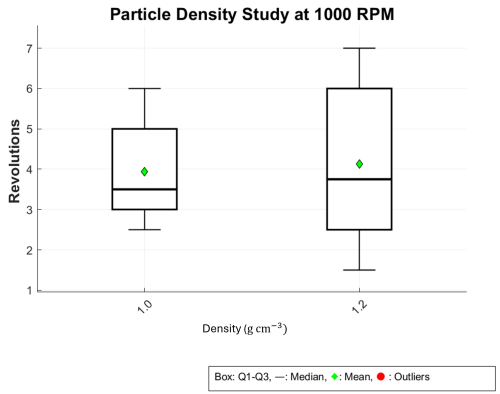
The maximum projected radius does not seem to be affected by a change in density, and both results show similar mean values and spread. The spin speed is the same but both particles do not have the same acceleration. For the $1.0\,\mathrm{g\,cm^{-3}}$ particle, lower mass means higher acceleration for the same force (a = F/m). This means that the $1.2\,\mathrm{g\,cm^{-3}}$ particle will have a lower acceleration. However, the results show either a geometric constraint, i.e. where the walls limit the projected radial position regardless of the acceleration. This suggests that there is an equilibrium position when it comes to the projected radius. The $1.0\,\mathrm{g\,cm^{-3}}$ particles have a notably longer total projected path lengths i.e. $\approx$ 35-40 mm vs $\approx\ 25 - 35mm$. This may be due to the higher maximum height change adding to the total distance travelled by the particles.

This study demonstrates that particle density significantly influences the behaviour of transient motion after spin-down begins. Particle density affects the mean of all observed measurements with the exception of maximum projected radius.

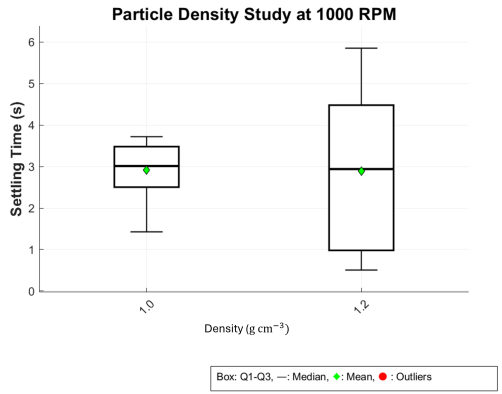
One might expect a for particles lighter than the fluid to show a similar result to the particles observed at the bottom near the plunger, just reversed. However, the exact impact of the presence of a headspace or vortex near the top of the syringe is not known once spin-down begins. This is beyond the scope this work and is suggested as future work.



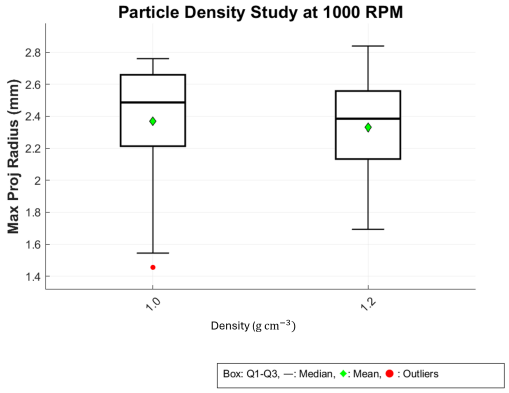
(a) Max Height Change(mm)



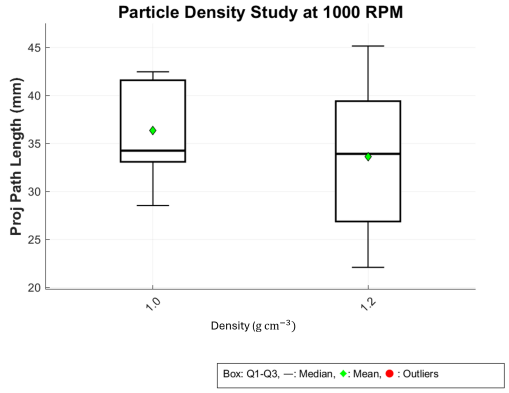
(b) Number Of Revolutions



(c) Settling time (s)



(d) Maximum Projected Radius (mm)



(e) Projected Path Length (mm)

Figure 120: Particle density study for particles of $1.0\,\mathrm{g\,cm^{-3}}$ and $1.2\,\mathrm{g\,cm^{-3}}$ for aspect ratio of 4 and RPM of 1000 with 1 cP fluid viscosity.

7.4 Viscosity Study

The viscosity study was tested to compare water-based drugs with those with higher viscosity. In this case, RPM of 3000 (Re $\approx$ 1094), 5000 (Re $\approx$ 1821) and 7000 (Re $\approx$ 2550) are tested for fluid viscosities of 1 cP, 9 cP, 18 cP and 23 cP. These values are selected on the basis of the range of viscosities used within the industry (Wu et al., 2024). During this study, the particles for the 1 cP case settle down to the plunger position. However, particles in the higher cP fluid do not start from the plunger position between each run. This is because the inspection of a syringe has many processes that may cause particles to be anywhere within the domain. The particles with the 1 cP fluid that are not neutrally buoyant, settle very quickly to the plunger, but this may not necessarily be the case for the particles in higher viscosity fluids. So in these studies, the initial location of the particle can be anywhere within the domain, as the particle does not settle in time. The 1 cP test is added for basic comparison.

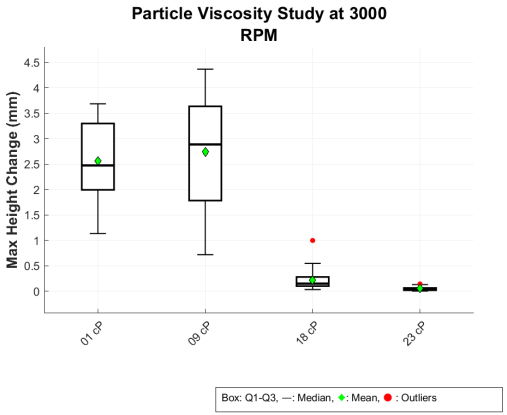
The maximum height change in Figure 121 shows a clear viscosity dependence across all RPMs. The 3000 RPM result shows that the particles are only fluidised at 1 and 9 cP. The mean height change recorded at 1 cP is around 2.5 mm. Unexpectedly, this increases to around 2.75 mm with a larger spread for 9 cP. The large spread of data may contribute to the slightly higher increase in the recorded data. As the RPM is increased to 5000, the particle in 18cP fluidise with a mean height change of 0.5 mm. The 23 cP particle is still not fluidised. As the RPM is further increased to 7000, all particles are fluidised, and the general trend shows a decreasing maximum height change from $\approx$ 3 mm for 1 cP dropping to $\approx$ 0.5 mm for 23 cP. This shows that viscous damping severely restricts vertical motion and that higher viscosity fluids create greater resistance to particle displacement from the initial position.

For the number of revolutions data recorded in Figure 122, all different RPMs show the same general trend of the reduction in the number of revolutions. This effect is more pronounced at higher RPMs. For example, at 7000 RPM, there is a large reduction from $\approx$ 8 revolutions at 1 cP to less than 1 revolution at 23 cP. In general, low viscosity and low RPM fluidises the particles and allows sustained rotational motion, while high viscosity quickly dampens it.

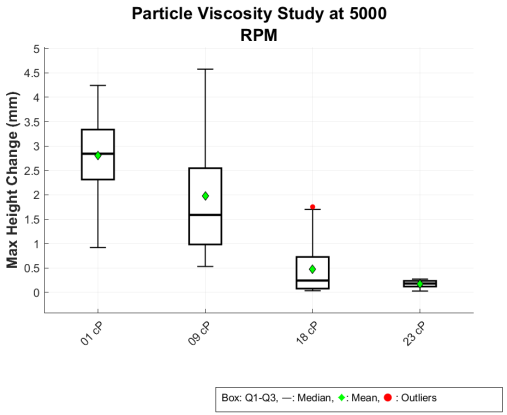
The results for the settling time show some complexity and inconsistency in Figure 123. As mentioned above, the particles for 1 cP settle at the bottom of the plunger. However, this is not the case for the particles in higher-viscosity fluids. The 1 cP test is conducted because it is widely used in the industry as water-based drugs. However, if the viscosity is increased, the particles could be anywhere within the domain as a syringe moves through the inspection line. Variable starting positions for particles that are not in the 1 cP fluid have a wider range of trajectories and settle behaviours. Generally, from 9 cP to 23 cP, the 3000 and 7000 RPMs show a decreasing settling time regardless of the initial position of particles in the domain. For many of the higher RPM results, the particles were visually observed near the centre of the syringe. These areas are less affected by the jet and more affected by the rotational flow in which the particle is. Once the fluid stops moving, the particles also stopped quickly because they do not have a lot of inertia within these flows. To better capture the results

for settling time in this study, better cameras are needed as the 75 FPS camera used, is too small to capture slow motion behaviour exhibited for these cases.

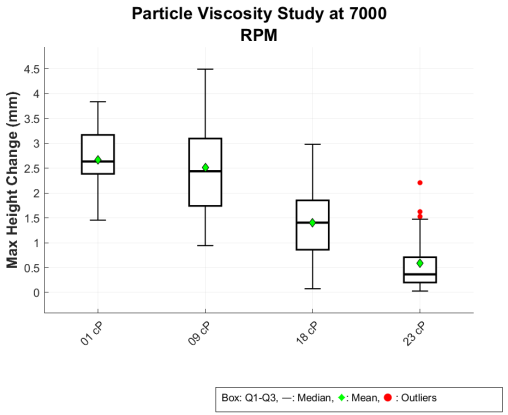
When looking at the maximum projected radius in Figure 124, the higher-viscosity particles are not observed near the outer radius. The particles show a maximum radius of around 1.5 mm for the maximum projected radius of both 18 cP and 23 cP across all RPMs. Note that this result may be due to very limited motion and the projected nature of the measurement discussed earlier. The projected path length in Figure 125 also shows a decreasing measurement as the viscosity is increased. This is expected behaviour has the particles do not have a lot of momentum at higher viscosities and this can be seen with near 0 projected length for the higher viscosities across all spin speeds.



(a) 3000 RPM

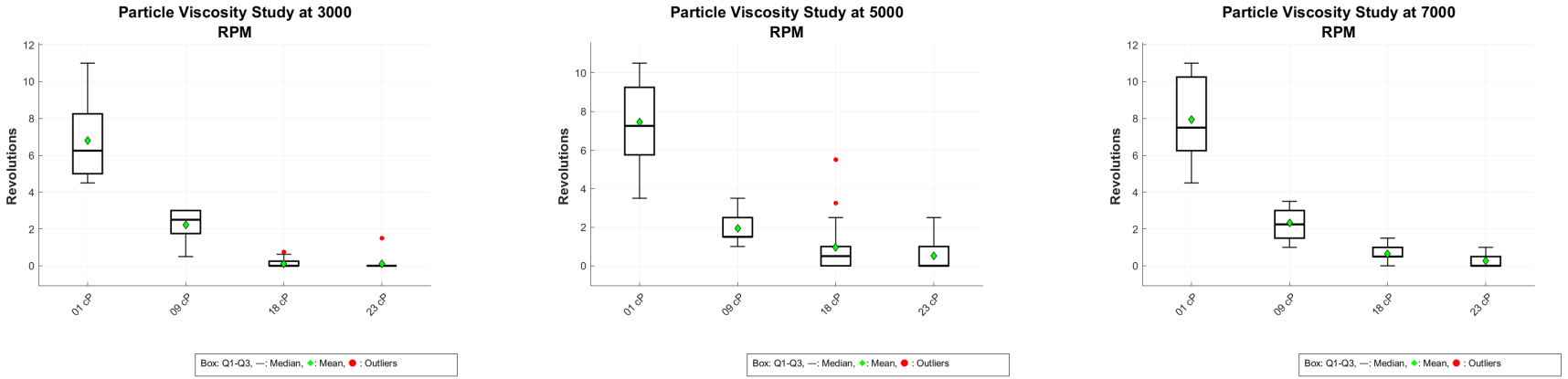


(b)5000 RPM



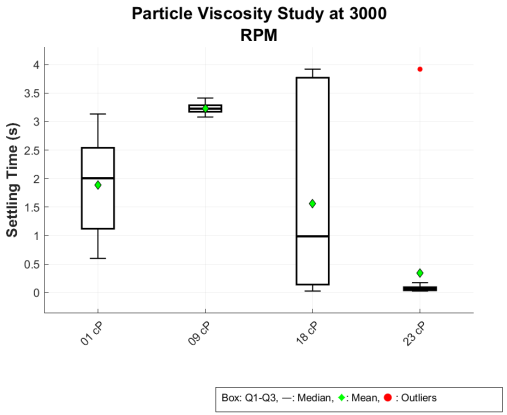
(c) 7000 RPM

Figure 121: Maximum height change for 1 cP, 9 cP, 18 cP and 23 cP at 3000, 5000 and 8000 RPM respectively.

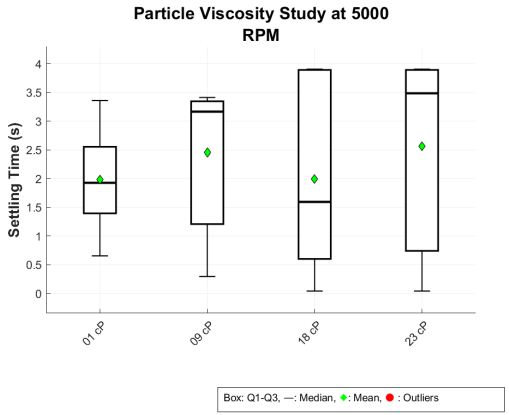


(a) 3000 RPM (b)5000 RPM (c) 7000 RPM

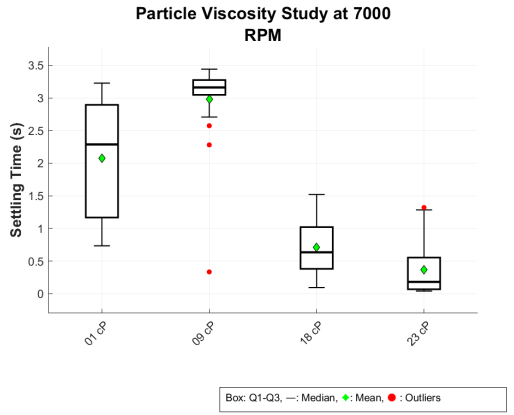
Figure 122: Number of revolutions for 1 cP, 9 cP, 18 cP and 23 cP at 3000, 5000 and 8000 RPM respectively.



(a) 3000 RPM

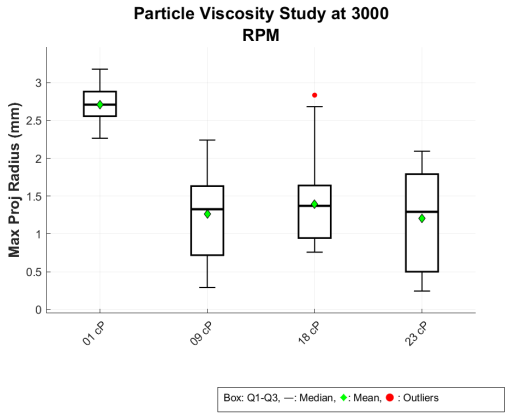


(b)5000 RPM

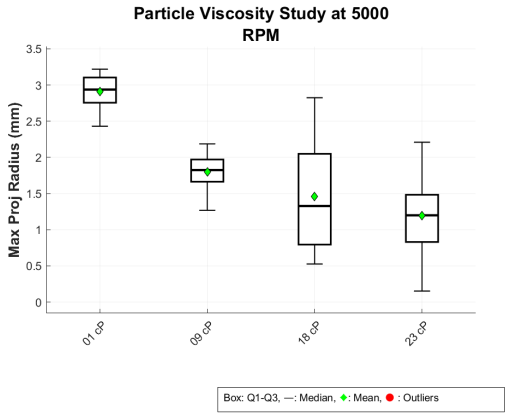


(c) 7000 RPM

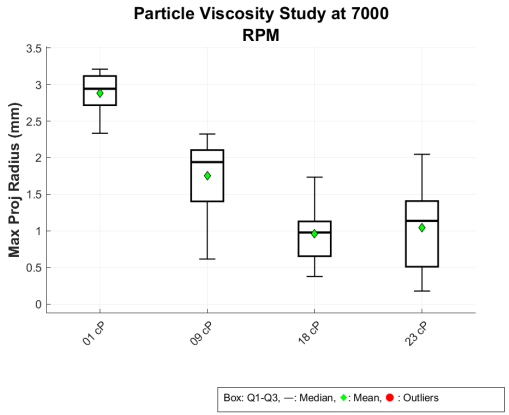
Figure 123: Settling time (s) for 1 cP, 9 cP, 18 cP and 23 cP at 3000, 5000 and 8000 RPM respectively.



(a) 3000 RPM

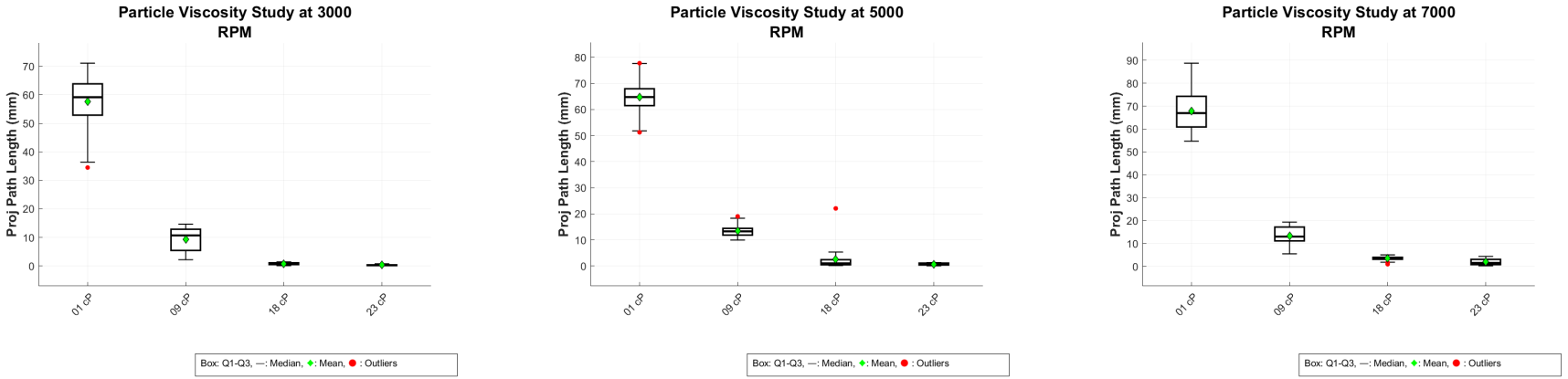


(b)5000 RPM



(c) 7000 RPM

Figure 124: Maximum projected radius (mm) for 1 cP, 9 cP, 18 cP and 23 cP at 3000, 5000 and 8000 RPM respectively.



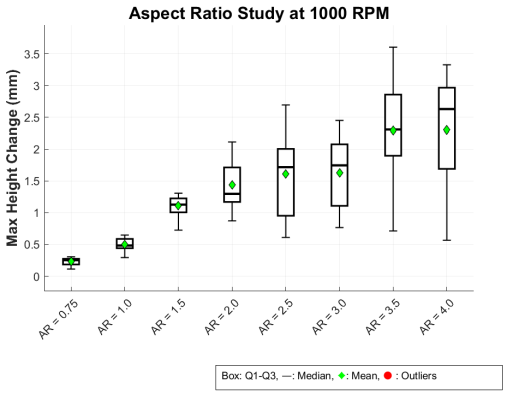
(a) 3000 RPM (b)5000 RPM (c) 7000 RPM

Figure 125: Projected path length (mm) for 1 cP, 9 cP, 18 cP and 23 cP at 3000, 5000 and 8000 RPM respectively.

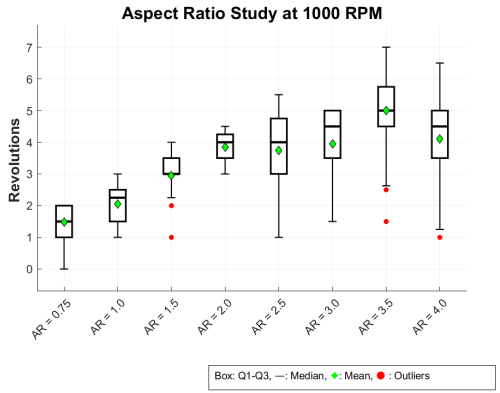
7.5 Aspect Ratio Study

For the aspect ratio study, the max rotational speed is set to 1000 RPM so that the chaotic flows discussed in the earlier chapters do not influence the results obtained. The purpose of the study is to determine the effects of aspect ratio of the fluid within the syringe and its impact on the measurement parameters that can be used to improve the inspection of the prefilled syringes. Figures 126a, 126b, 126c, 126d and 126e show the change in the maximum height, the number of revolutions, the settling time, the maximum projected radius, and the projected path length, respectively.

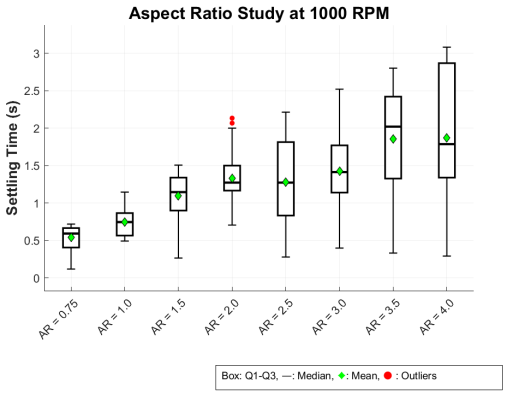
The results for the maximum height change shows a large increase with increasing aspect ratio. At an AR of 0.75, the mean maximum height change is ≈ 0.3 mm. It increases ≈ 3 mm at AR=4.0 and this makes physical sense because taller containers allow more vertical space for the jets to travel causing particles to lift more. The spread of data also increases with increasing aspect ratio. A similar trend is seen for all other measured parameters. Particles experience more oscillations in taller geometries. The number of revolutions increases from 1.5 at AR=0.75 to $\approx 4-5$ at the higher aspect ratios. This is very important for inspection as increasing motion increases the chance of the particle being detected. This also applies to all other parameters. For the settling time, AR of greater than 2.0 shows much wider spread where the mean rise gradually up to 2s. Taller syringes (higher AR) allow particles to take longer to settle, possibly due to greater fluid inertia or longer travel distance seen with the projected path length. Also, this suggests that the system is more sensitive to the initial aspect ratios. In general, a linear increase in both mean and spread is seen for all the measurements. The max projected radius is the least affected, but still shows a slight increase from AR 0.75 to 2.0 from 2.2 mm to around 2.8 mm.



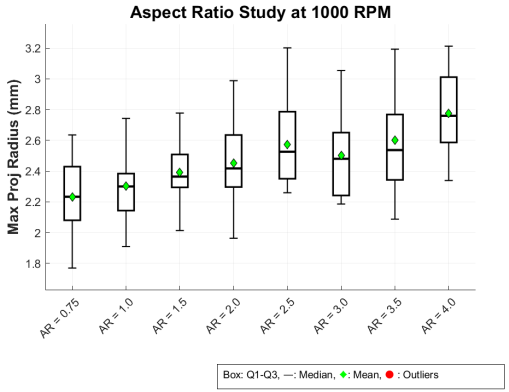
(a) Max Height Change(mm)



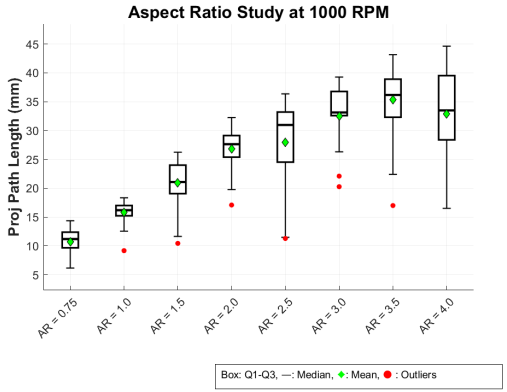
(b) Number Of Revolutions



(c) Settling time (s)



(d) Maximum Projected Radius (mm)



(e) Projected Path Length (mm)

Figure 126: Aspect ratio study for AR = 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, and 4.0.

7.6 Headspace Study

For the final study, the headspace (i.e. the air gap at the top of the syringe) is modified to assess the impact it has on the resulting measurements for the relevant parameters such as maximum height change (mm), number of revolutions, settling time (s), maximum projected radius (mm) and projected path length (mm). An example of a collapsing vortex as spin-down begins is shown in Figure 127 through various images as time passes after spin-down. The presence of the this headspace in a syringe and its impact on the particle track is another aspect of this work that is of interest. In this study, this impact is analysed.

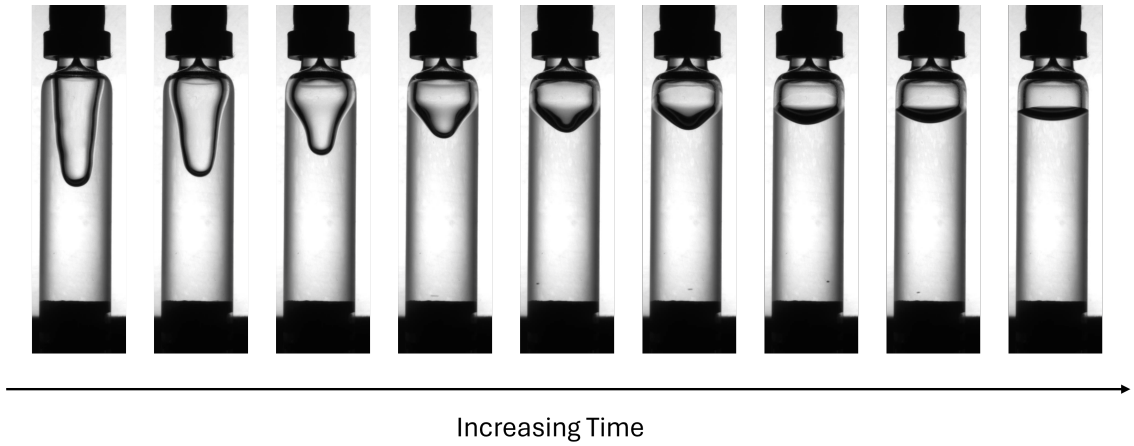
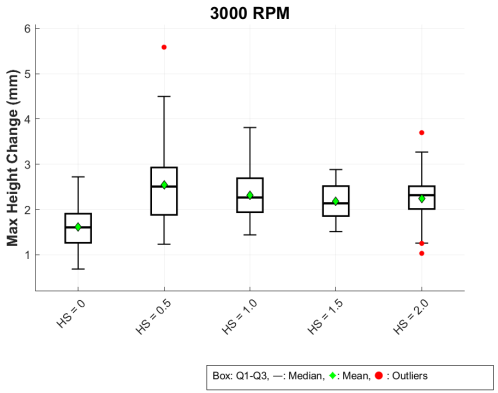


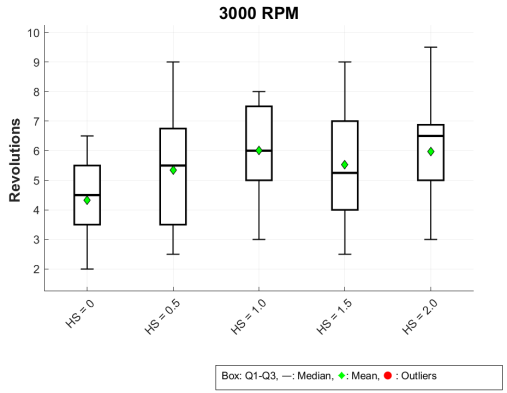
Figure 127: Collapse of a vortex present due to headspace through various images as time after spin-down increases at a maximum rational speed of 3000 RPM.

For this study, the headspace is increased in increments of 0.5 aspect ratio, up-to a maximum of 2.0 (i.e. half the fluid is air and the other half of the fluid column is a water-based fluid at 1 cP viscosity. Therefore the range of headspace (HS) are HS = 0, HS = 0.5, HS = 1.0, HS = 1.5 and HS = 2.0. Figures 128a, 128b, 128c, 128d and 128e show the change in the maximum height, the number of revolutions, the settling time, the maximum projected radius, and the projected path length, respectively.

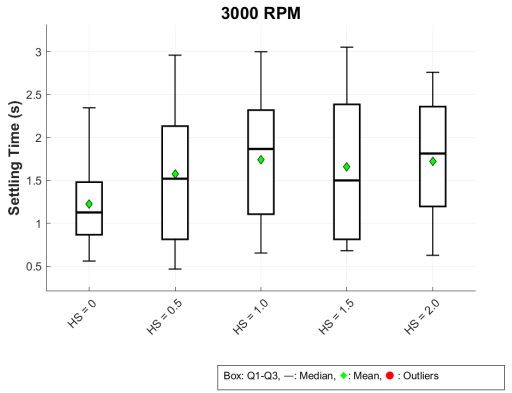
For maximum height change, there is an initial increase from no headspace but no significant change after that. This suggests that there is an optimal headspace effect that can be utilised to increase the maximum height change so that particle detection within syringe inspection can be improved. The number of revolutions show an increase from ≈ 4.5 to a mean of 6 revolutions. The presence of headspace seems to enable a more sustained rotation of the particle. A similar effect is seen for the settling time where no headspace resulted in a settling time of ≈ 1.25 s which increased to approximately 1.75 s for the higher headspace. The maximum projected radius does not change much between the different headspace tested, and behaviours very similar to the other studies where the particle reaches the outer walls and is limited by the radius. Finally, the projected path length has a very similar effect as the maximum height change where increasing headspace is allowing for more sustained motion of the particles. This can be very useful for inspection as a particle that moves more, is likely to get detected.



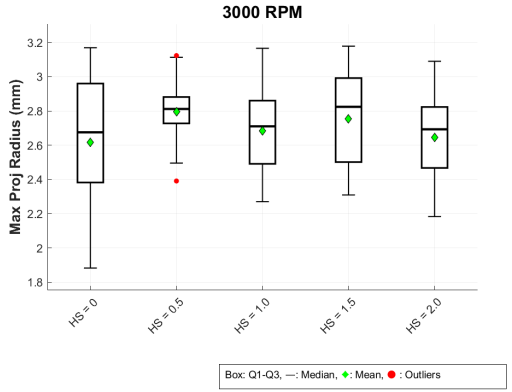
(a) Max Height Change(mm)



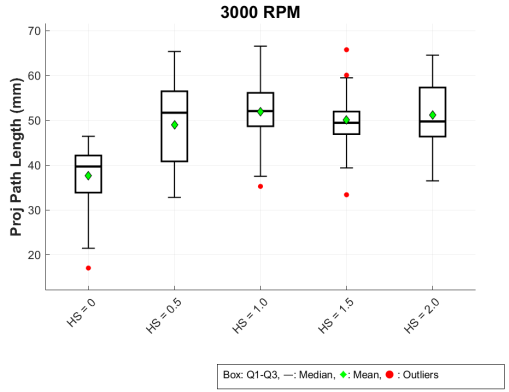
(b) Number Of Revolutions



(c) Settling time (s)



(d) Maximum Projected Radius (mm)



(e) Projected Path Length (mm)

Figure 128: Headspace study for HS= 0, 0.5, 1.0, 1.5 and 2.0 AR

7.7 Experimental and Computational Validation

In this section, a comparison between the computational model and the experimental results is made. Firstly, it must be acknowledged that there is no direct one-to-one comparison between the results obtained through the two methods. However, a justification can be made for the computational model when compared to the experimental results. This section will make attempt to make such a justification. Note that the COMSOL model is a 2D axisymmetric model that utilises massless and mass particles, where as the experimental rig results in 2D projected particle tracks. Keeping this in mind, lets consider some of the results previously discussed.

7.7.1 Consistent Qualitative Trends

Consider the results obtained for the RPM study in Sections 5.1.4 and 5.1.5, where increasing the spin speed from 1000 RPM (where less than 1 revolution occurred) to 3000 RPM, increased the number of revolutions to 2.5. All other variables are kept constant. Now comparing this to the results obtain in the experimental RPM study shown on Figure 126b, the trend in the number of revolutions agrees with the COMSOL model. Although the number of revolutions between the two studies are numerically far apart (mean range of 4 mm to 7 mm), there are variations that lead to this result. For example, the COMSOL studies are massless particles where as the experiments are conducted for particles with a diameter of 180 µm. Moreover, the initial location of the particle is not know during the experimental setup. However this is somewhat mitigated by repeating the experiment 20 times and looking at the ranges of the key parameters recorded.

Another comparison can be made between the radial position of the particle in relation to the position of the axial jet during spin-down. Consider the jet formation shown on Figure 45. When comparing the radial seed of a particle during spin down, its clear that the particles near the centreline line lift much higher than those near the outer walls. This is shown in the results from the COMSOL model shown on Table 4 (Section 5.1.4) where particles within 2 mm of the centreline gain the most height during spin-down. This is because this is where the axial jet is most dominant during spin-down. A typical particle track as shown on Figure 117 confirms this experimentally as the particle gains height near the centreline and drops as the radial position increases. This shows that when the particle is outside the axial jet region shown on Figure 45 and therefore falls under gravity.

Looking at the results obtained during the experimental aspect ratio study, it is found that the maximum height change observed increases with an increasing aspect ratio (as shown on Figure 126 a). At an aspect ratio of 4.0, the maximum height change is just over 3 mm. The COMSOL aspect ratio study shows similar results for an aspect ratio of 4.0 where the box plot on Figure 84h shows a similar range. Moreover, Figure 84 also shows the same trend with the maximum height change increasing with aspect ratio.

The experimental RPM study for the maximum projected radius (Figure 118d) shows that maximum particle radius for 1000 RPM is between 2.5 mm and 3 mm. This can be compared to the particle tracks obtained for particles of varying diameters using the COMSOL CFD model in Figure 95. These results show that maximum particle radius for all particles tested is between 2 and 3. This range of data is very similar for the experimental result showing maximum radial position near the outer wall. This also agrees with theory, as rotating fluids will cause particles to transport outwards during the spin-up phase, due to centrifugal centrifugal forces. Similarly, the CFD results on Figure 94 show that the maximum height change at 1000 RPM is 4.5 mm (7.5 mm - 3 mm). Experimentally, Figure 118a shows at the maximum height change is ($\approx$ 2 to 3 mm). Considering various differences discussed earlier with particle types, varying diameters, initial seed locations etc, similar trends between the experimental and CFD results are still observed.

The trends analysed in this section cover different aspects of fluid and particle motion and help to provide an overall picture of what happens during the inspection of PFS syringes. Even though a direct comparison has not been made, from all these experimental results, the CFD model can be qualitatively validated. The purpose of this project is to try and understand the inspection process and the CFD model helps to gain this understanding. There are limitations to the model as the accuracy of the data can further be perfected through direct comparison to experiments. However, that is not in the scope of the project as the aim is to understand the fluid dynamics and particle motion trends within inspection to aid with developing better inspection methods.

7.8 Experimental Summary

These experimental investigations demonstrate that particle motion during syringe inspection can be significantly enhanced or suppressed through careful control of operational and design parameters. RPM and aspect ratio emerge as the primary motion enhancers, with higher rotation speeds and taller geometries allowing more sustained particle trajectories, longer settling times, and increased detection opportunities. However, viscosity acts as the dominant motion suppressor, with higher viscosities severely restricting particle movement across all measured parameters. Particle density and diameter show intermediate effects, while headspace provides modest motion enhancement, especially with the change in height observed. Using these findings, one can better adapt the inspection of prefilled syringes using variations of different parameters. One such example is to combine higher viscosity flows with faster rotational speeds in an attempt to increase the Reynolds number so that the inefficiencies for inspection leads from high-viscosity fluids can be enhanced by using the advantages of increased RPMs. Using these results, better test conditions can be developed for new prefilled syringes as well as the current ones that are already in production to ensure that particles are thoroughly detected.

8 Conclusion & Industrial Focus

8.1 What has been learned from the fluid dynamics of inspection?

This research has offered, for the first time, a clear picture of the fluid mechanics that drive automated syringe inspection. By looking closely at how the flow develops and decays, one can now see how two distinct but connected flow regimes work together to determine whether particles inside the syringe can be detected. In the spin-up phase, the fluid begins to rotate as diffusion of momentum spreads inward from the syringe walls. Within the time frame of a second, the whole fluid body reaches solid body motion. During this process, centrifugal forces push particles outward toward the walls, where secondary flows lift them upward along the syringe boundaries. This is drastically different in the spin-down phase, which is the key to particle detection. When the syringe stops suddenly, two jets of fluid emanate inward from the top and bottom ends, meeting at the centre. These jets are strongest after spin-down begins and loose their intensity as the flow decays. The jets create the crucial inspection window when particles are caught within these jets becoming fluidised. This fluidisation of the particle allows for detection of movement through the inspection algorithms.Aspect ratio and Reynold's numbers have shown strong effects on the behaviour of these flows and in turn on these particles that have been tested both experimentally and computationally. As the aspect ratio increases from 0.5 to 6.0, the critical Reynolds number for the flow transition decreases from around 2100 to 650, levelling off beyond AR = 3.0. Above these thresholds, flow instabilities develop in the form of Taylor–Couette vortices. Whether the resulting flow mobilises the particles or hinders their detection by suppression motion depends on the initial conditions of the spin. Understanding the fluid dynamics of the inspection process has enabled us to understand why particles during inspection move the way they do.

8.2 What has been learned about the motion of particles during inspection?

The particle motion studies revealed a clear order of influence among the different parameters, showing which ones matter most for optimising syringe inspection. By analysing more than a thousand individual particle trajectories in six systematic studies, a quantitative link between system parameters and how particles actually move inside the syringe has been establish. Reynolds number and aspect ratio show a strong influence on the inspection in some way. Higher rotational speeds or relatively taller fluid columns (with fixed base size) allow for a sustained particle tracks, improving projected total distances particles travel as well as the number of revolutions or maximum height changes recorded. Within the inspection window, these measurements are crucial to detect any foreign particles. At the same time, a higher viscosity dampens the flow structures and so, particles are hardly fluidised. To enhance the fluidisation of these particles, a faster rotational speed or higher aspect ratio can be used to make inspection easier even for higher viscosity fluids. Larger particle sizes (diameter) show reduced maximum height change and settling times while having a similar number of revolutions. and projected path lengths. The particle density between a neutrally buoyant article and a denser particle are compared showing decreasing maximum heights and projected path lengths. The number of revolutions, settling time and maximum projected

radius relatively stay the same. However, a much wider spread of results is recorded, reflecting their sensitivity to small differences in starting position or flow structure. This variability makes detection more challenging, but also increases the chance of seeing subtle particle movements when the inspection is successful. Finally, headspace volume shows a saturation effect. Adding a small headspace significantly increased particle motion, but further increases beyond and air gap of AR = 1.0, gives only marginal benefits. The number of revolutions increased from 4.5 to 6.5, while the settling time increased from 1.25 to 1.75 seconds hence, extending the inspection window.

8.3 Major outputs from the PhD project

This research has delivered several key outcomes that move the field forward, both in terms of fundamental understanding and practical application to pharmaceutical inspection. A reliable CFD modelling approach has been built using COMSOL Multiphysics. By using a 2D axisymmetric setup with mesh-independence testing, the simulations cut the computational cost dramatically compared to full 3D runs. A 2D-to-3D mapping method makes it possible to track particles efficiently in three dimensions without losing this computational advantage. Alongside the modelling, a full inspection test rig has been designed and built. This system can reproduce industrial operating conditions and allows systematic parameter studies. It combines precise servo-motor control up to 8000 RPM with imaging at 75 fps but can be adapted by using high speed cameras if needed. To handle the large volume of video data, a dedicated MATLAB analysis suite has been developed. This includes background subtraction (adapted from Crocker and Grier (1996)) with automated trajectory linking, and statistical analysis routines. These tracking algorithms achieve sub-pixel accuracy in locating particles and can automatically link trajectories across video frames. Together, these tools allow efficient processing of thousands of particle tracks and extraction of meaningful motion statistics.

From the combined simulations and experiments, critical Reynolds number charts (Figure 53) have been created for aspect ratios ranging from 0.5 to 6.0. These maps show where the flow remains laminar and where it begins to transition to chaotic flows. They provide industry with a practical guide for better inspection regimes depending on syringe geometry. By systematically varying conditions, the work highlights the importance of different system parameters. Viscosity, aspect ratio, and RPM emerge as the main control variables. Finally, the work provides new insights into the physics of syringe inspection. It explains how primary and secondary flows interact, how jets form during spin-down, and how particles couple with these flows. These results provide the foundation needed for the development of more effective inspection solutions in the future.

8.4 Practical implications in syringe inspection for industry

The findings from this research can be applied to improve current inspection protocols for pre-existing prefilled syringes and to shape the design of future inspection methods. Several clear strategies emerge from the relationships identified between fluid and particle behaviour. One of these strategise is the viscosity–Re optimisation. To inspect a product with a known viscosity, it is now known that an increases in aspect ratio and RPM can fluidise particles that may not initially be fluidised. These tests can be further expanded across a range of viscosities, particle diameters, and fluid densities. Another strategy needed might be a geometry-based protocol selection. The aspect ratio studies provide practical guidance for container design and inspection parameters. Syringes that have aspect ratios below 2.0 need stronger spin speeds, as the geometry limits the development of the axial jets during spin-down. On the other hand, taller containers where the aspect ratio is larger than 2 allows for improved parameters that help with all parameters needed for inspection. Small increases in fluid height within existing containers can lead to significant improvements of the parameters needed for detection.

The challenge of neutral buoyancy means inspection must account for greater variability in particle motion. Such formulations may require longer inspection windows or multiple spin cycles to capture different particle behaviours. There needs to be a process to design protocols that account for random starting positions, rather than assuming consistent initial placement. This will improves the detection reliability for these cases. From the headspace studies, results show that maintaining a small airgap helps with all detection parameters. Particle motion is enhanced without adding unnecessary complexity. Larger headspace overall offers little additional benefit and can complicate other considerations. For example, dosing accuracy during transport.

Temporal analysis highlights that the best time to detect particles is in the earlier stages of spin-down as soon as it begins. During this window, axial jets are at their strongest but viscous dissipation of flow fields has not yet taken over. Aligning camera activation with this period can improve detection efficiency and reduce the burden of processing unnecessary data. Finally, the parameter sensitivity results provide a framework for tailoring inspection efforts. Challenging cases should receive more intensive inspection, while more favourable formulations can be inspected under standard protocols. This approach balances reliability with efficiency across product lines. An example of such a case is high-viscosity fluids with neutrally buoyant and small particles.

8.5 Ideal inspection of particles in Pre-filled Syringes.

An ideal inspection should take be adaptive to the product type. The inspection protocol should depend on various parameters with prefilled syringes and any parameters that suppress motion should be enhanced through optimisation of the inspection process as previously discussed. In practice, this would mean a system that automatically selects the right inspection settings using a decision matrix. By considering viscosity, particle density, container geometry, and particle size, the system could set the most appropriate RPM, spin duration,

and inspection timing for each product. Such a decision matrix does not yet exist and the studies included in this research only scratch the surface. For a particular product, initial testing can be done using the CFD providing crucial flow details with resect to the fluid and container parameters. This can then be enhanced using test kits to figure out the most optimal parameters for that particular product.

The imaging system used in an ideal inspection should be able to adjust frame rate and exposure based on how particles are expected to move. For fast-moving cases (low viscosity, tall containers), higher frame rates would capture the detail. For slower-moving cases (high viscosity, short containers), longer exposures and extended observation windows would be more effective. For these particular cases, there might need extensive parameter optimisation to ensure most parameters can be fluidised.

Although these studies focus on a single cycle inspection, an ideal inspection system is a multicycle one. For especially difficult products, the system could run multiple spin cycles with varied parameters. Since different starting positions produce different trajectories, repeating the cycle would improve coverage and increase detection confidence with the drawback of increased inspection time per syringe as well as increased hardware costs. Inspection settings would also be tailored to container geometry. For example, RPM profiles, spin-up durations, and inspection windows could be matched to aspect ratio, ensuring that axial jet formation and particle mobilisation are always as effective as possible. Beyond running inspections, the system could also monitor itself. By analysing distributions of particle motion parameters, it could detect early signs of wear or drift in the system, as well as formulation changes that might affect inspection reliability. These highlight what an ideal inspection should consist of and the aim should be to incorporate as much of this as possible.

8.6 Future Work

This work provides a strong foundation for fluid flow and the motion of particles within the inspection of prefilled syringes. However, several areas remain open for future exploration. By addressing these areas, the prefilled syringe inspection process can be truly optimised. Real pharmaceutical products rarely contain only one particle and that too of a similar type. In reality, metal, glass, or a polymer of different sort could both be present in the syringe. A method for detecting the type of particle would benefit the industry improving not just inspection, but it could also provide a solution to reducing particle in future inspections. This is because knowing what particle exists in the syringe provides essential diagnostic information to determine the location of particle leakage. Dealing with multiple particles will require capturing the complex movements of two or more particles simultaneously. This will most likely require advanced computational techniques and 3D modelling. However, regulations for passing or failing a product inspection do not require this.

The focus of this research was to mobilise free particles resting at the bottom of the plunger. Practically, particles can be stuck on any surface within the syringe, not just the walls. Better spin profiles are needed to mobilise these particles so that they can also be inspected. This is one of the biggest challenges within the industry. These are often the hardest to

dislodge but also the most concerning from a safety perspective. Future studies could explore higher Reynolds number flows or entirely new methods, such as ultrasonic agitation or electromagnetic fields, to specifically target and mobilise wall-adhered contaminants.

Another important aspect of particle inspection is to differentiate between air bubbles and particles. During experimentation, it was found that bubbles in a syringe can be aggregated using spin profiles. Spinning a syringe full of bubbles at extremely high RPMs (8000) and then reversing the direction but keeping the RPM the same causes bubbles to collect at the centreline and rise to the top leaving the syringe at the needle. This can be an important preparation step before inspection to remove these bubbles. However, the optimal spin conditions still need to be mapped out for syringes with different viscosities and aspect ratios, which extends Reynolds number validation.

Moreover, the chaotic flow regime for these industrial flows needs to be better understood. A lot more work is needed to understand this particular application of the Taylor-Couette vortices. The models used in this work are laminar and 2D. These are a good starting point but do not cover the extent of the real inspection process and these models are needed to be developed further. This research identified critical Reynolds numbers up to 3000. In practice inspection systems may operate at much higher speeds. Therefore, this can be extended to 10,000 to provide a clearer picture of chaotic flow regimes and their potential to mobilise particles under extreme conditions. Furthermore, the present COMSOL model assumes laminar flow throughout, which is a reasonable approximation below the critical Reynolds number. However, as discussed in Chapter 4, the flow transitions toward chaotic and potentially turbulent regimes at higher Reynolds numbers. In this transitional regime, a turbulence model is the next logical step that is able to resolve some of the vortices exhibited within the flow field it would provide more accurate results. Even though the current model has its limitations, it has been able to provide with a blueprint as to how particles move during a spin cycle with the inspection of PFS. Future work should assess whether implementing such a model yields meaningfully different particle trajectory predictions, particularly for the high-RPM cases where experimental results showed increased variability."

Moving from laboratory demonstrations to production environments will require further study of real-time control systems, sensor integration, and automated parameter adjustment. This step is crucial for translating scientific insight into robust industrial practice. While this work focused on syringes, the pharmaceutical industry also relies on vials, cartridges, and specialised delivery devices. Applying the same framework to these geometries could reveal new optimisation opportunities and extend the impact of this research. This also applies to the food processing industry, where an example is the inspection of sealed bottles that contain various fluids. The present study centred on conventional optical methods. Future research should investigate how improved understanding of particle motion can be used to push detection sensitivity further. Some examples include advanced technologies such as laser-based systems, acoustic detection, as well as machine learning to enhance image analysis.

Any technical improvements must also be balanced with cost. Any future studies should con-

sider economic modelling to compare detection improvements against operational expenses. This will enable practical, cost effective strategies across different product lines and markets. Finally, there is an great opportunity to link the quantitative understanding gained with this study to the regulatory frameworks. Doing so could provide science-based justification for inspection protocols and potentially streamline approval processes for new formulations. In summary, this research has laid the groundwork for both immediate improvements and long-term innovation. It represents the first comprehensive investigation of fluid mechanics in pharmaceutical inspection, bridging fundamental science with industrial application. By combining computational modelling, systematic experimentation, and quantitative analysis, it provides a solid platform for the next generation of particle detection technologies and for the advancement of pharmaceutical quality assurance.

9 Appendix

9.1 Conversion Code: Conversion_code.m

```
1
2       % Allow the user to select a specific test folder
3   selectedTestFolderPath = uigetdir('Select the test folder to process');
4
5   % Check if the user canceled the folder selection
6   if selectedTestFolderPath == 0
7       disp('Folder selection canceled. Exiting.');
8       return;
9   end
10
11  % Define the path for the converted folder within the selected test folder
12  convertedFolderPath = fullfile(selectedTestFolderPath, 'converted');
13
14  % Create the 'converted' folder if it doesn't exist
15  if ~exist(convertedFolderPath, 'dir')
16      mkdir(convertedFolderPath);
17  end
18
19  % Check if the 'converted' subfolder already contains TIFF files
20  if ~isempty(dir(fullfile(convertedFolderPath, '*.tiff')))
21      fprintf('Skipping %s as conversion has already been done.\n',
        selectedTestFolderPath);
22  else
23      % Process MP4 files in the selected test folder
24      mp4Files = dir(fullfile(selectedTestFolderPath, '*.mp4'));
25
26      for j = 1:length(mp4Files)
27          currentMP4File = fullfile(selectedTestFolderPath, mp4Files(j).name
        );
28
29          [~, baseFileName, ~] = fileparts(currentMP4File);
30          videoReader = VideoReader(currentMP4File);
31
32          % Display the frame rate
33          fprintf('Frame Rate for %s: %.2f frames per second\n',
        currentMP4File, videoReader.FrameRate);
34
35          fprintf('Processing %s...\n', currentMP4File);
36
37          frameCount = 0;
38          totalFrames = videoReader.NumFrames; % Get the total number of
        frames
39
40          % Create a folder with the same name as the .mp4 file within the '
        converted' folder
41          currentMP4Folder = fullfile(convertedFolderPath, baseFileName);
42          if ~exist(currentMP4Folder, 'dir')
43              mkdir(currentMP4Folder);
44          end
```

```
45
46          while hasFrame(videoReader)
47              frameCount = frameCount + 1;
48              try
49                  currentFrame = readFrame(videoReader);
50              catch ME
51                  warning('Error reading frame %d: %s', frameCount, ME.
    message);
52                  continue;
53              end
54
55              % Save each frame as a separate TIFF file in the folder with .
    mp4 name
56              tiffFileName = sprintf('%s_Frame%04d.tiff', baseFileName,
    frameCount);
57              tiffFilePath = fullfile(currentMP4Folder, tiffFileName);
58              imwrite(currentFrame, tiffFilePath, 'tiff', 'compression', '
    none');
59
60              % Retrieve the current time in seconds
61              currentTimeInSeconds = videoReader.CurrentTime;
62
63              % Calculate and display progress
64              progress = frameCount / totalFrames * 100;
65              fprintf('Processed frame %d of %d (%.2f%%) at time %.2f
    seconds\n', frameCount, totalFrames, progress, currentTimeInSeconds);
66          end
67      end
68
69      fprintf('Conversion for %s completed.\n', selectedTestFolderPath);
70  end
71
72  disp('Frame conversion and organization completed.');
```

Listing 1: Code to convert MP4 files to .Tiff frame images

9.2 Run Sort Code: Run_Sort.m

```
1
2  if isempty(gcp('nocreate'))
3      parpool;  % Automatically uses all available cores
4  end
5
6  % === TOTAL TIMER START ===
7  overallStart = tic;
8
9  % Define RPM values and corresponding delays
10 RPM_values = [500 1000 1500 2000 2500 3000 3500 4000 4500 5000 5500 6000
       6500 7000 7500 8000];
11 Spin_Up_Delay = [0.09, 0.09 + (RPM_values(2:end) - 500) / 500 * 0.01];
12 Spin_Down_Delay = Spin_Up_Delay / 2;
13
14 % Define headers for the table
15 headers = {'Spin-Up Start', 'Inspection Start', 'Inspection End'};
16
17 % Create the custom input dialog
18 fig = uifigure('Name', 'Input');
19 prompt1 = uilabel(fig, 'Position', [20, 270, 250, 22], 'Text', 'Enter the
       value for First Frame Selection:');
20 input1 = uieditfield(fig, 'numeric', 'Position', [270, 270, 100, 22]);
21
22 prompt2 = uilabel(fig, 'Position', [20, 240, 250, 22], 'Text', 'Select the
       RPM:');
23 input2 = uidropdown(fig, 'Position', [270, 240, 100, 22], 'Items', string(
       RPM_values), 'Value', '500');
24
25 prompt3 = uilabel(fig, 'Position', [20, 210, 250, 22], 'Text', 'Enter the
       Spin-Up Time (s):');
26 input3 = uieditfield(fig, 'numeric', 'Position', [270, 210, 100, 22]);
27
28 prompt4 = uilabel(fig, 'Position', [20, 180, 250, 22], 'Text', 'Enter the
       Spin-Down Time (s):');
29 input4 = uieditfield(fig, 'numeric', 'Position', [270, 180, 100, 22]);
30
31 btn = uibutton(fig, 'Position', [150, 140, 100, 22], 'Text', 'Submit', ...
32     'ButtonPushedFcn', @(btn,event) uiresume(fig));
33
34 uiwait(fig);
35
36 % Extract user inputs
37 firstFrameSelection = input1.Value;
38 rpmValue = str2double(input2.Value);
39 spinUpTime = input3.Value;
40 spinDownTime = input4.Value;
41
42 % Close the figure
43 close(fig);
44
45 % Prompt user to select folder containing frames
46 folderPath = uigetdir('Select Folder Containing Frames');
```

```
47
48 % List all files in the selected folder that match the pattern
49 frameFiles = dir(fullfile(folderPath, '*RPM_Frame*.tiff'));
50
51 % Extract the RPM prefix dynamically
52 frameNames = {frameFiles.name};
53 rpmPrefixPattern = '^\d+RPM_';
54 rpmPrefixes = regexp(frameNames, rpmPrefixPattern, 'match', 'once');
55 uniquePrefixes = unique(rpmPrefixes(~cellfun('isempty', rpmPrefixes)));
56
57 if length(uniquePrefixes) ~= 1
58     error('Multiple or no RPM prefixes found in the file names. Ensure all
       files have the same prefix.');
59 end
60
61 rpmPrefix = uniquePrefixes{1};
62
63 % Sort frame files numerically
64 frameNums = cellfun(@(x) sscanf(x, [rpmPrefix 'Frame%d.tiff']), frameNames
     );
65 [~, sortIdx] = sort(frameNums);
66 frameFiles = frameFiles(sortIdx);
67
68 % Count number of frames
69 numFrames = length(frameFiles);
70 disp(['Number of frames in the selected folder: ', num2str(numFrames)]);
71
72 % Find corresponding delays
73 rpmIndex = find(RPM_values == rpmValue, 1);
74 spinUpDelay = Spin_Up_Delay(rpmIndex);
75 spinDownDelay = Spin_Down_Delay(rpmIndex);
76
77 % FPS
78 FPS = 75;
79 framesPerRun = ceil((spinUpTime + spinUpDelay + spinDownTime -
     spinDownDelay) * FPS);
80 numRuns = ceil((numFrames - firstFrameSelection + 1) / framesPerRun);
81
82 % Table
83 secondTable = cell(numRuns + 1, length(headers));
84 for col = 1:length(headers)
85     secondTable{1, col} = headers{col};
86 end
87
88 spinUpStart = firstFrameSelection;
89 inspectionStart = (spinUpTime + spinUpDelay) * FPS + spinUpStart;
90 %inspectionStart = (spinUpTime + spinUpDelay) * FPS + spinUpStart - 2; %
     For 23cP (Need two frames earlier than normal for 3000RPM)
91 inspectionEnd = (spinDownTime - spinDownDelay) * FPS + inspectionStart;
92
93 secondTable{2, 1} = spinUpStart;
94 secondTable{2, 2} = inspectionStart;
95 secondTable{2, 3} = inspectionEnd;
96
```

```
for run = 3:numRuns + 1
    spinUpStart = secondTable{run-1, 3};
    inspectionStart = (spinUpTime + spinUpDelay) * FPS + spinUpStart;
    inspectionEnd = (spinDownTime - spinDownDelay) * FPS + inspectionStart
   ;

    secondTable{run, 1} = spinUpStart;
    secondTable{run, 2} = inspectionStart;
    secondTable{run, 3} = inspectionEnd;
end

disp('Predicted Frame Ranges:');
disp(secondTable);

% Move initial frames
initialFramesFolder = fullfile(folderPath, 'Initial Frames');
if ~exist(initialFramesFolder, 'dir')
    mkdir(initialFramesFolder);
end

movedFiles = struct('source', {}, 'destination', {});
createdFolders = {initialFramesFolder};

% === SORTING & MOVING TIMER START ===
sortingStart = tic;

hWaitbar = waitbar(0, 'Processing frames...', 'Name', 'Frame Sorting
   Progress');
numInitialFrames = ceil(firstFrameSelection)-1;

for frameNum = 1:numInitialFrames
    if frameNum <= numFrames
        frameFile = frameFiles(frameNum).name;
        destination = fullfile(initialFramesFolder, frameFile);
        movefile(fullfile(folderPath, frameFile), destination);
        movedFiles(end+1).source = fullfile(folderPath, frameFile);
        movedFiles(end).destination = destination;
        waitbar(frameNum / numInitialFrames, hWaitbar);
    end
end

% Move run frames (no spin-up/inspection folders)
totalFramesToProcess = numFrames - numInitialFrames;
processedFrames = 0;

for run = 2:size(secondTable, 1)
    if isempty(secondTable{run, 1})
        break;
    end
    runFolder = fullfile(folderPath, sprintf('Test%04d', run-1));
    if ~exist(runFolder, 'dir')
        mkdir(runFolder);
        createdFolders{end+1} = runFolder;
    end
```

```
149
150     inspectionStartFrame = ceil(secondTable{run, 2});
151     inspectionEndFrame   = min(ceil(secondTable{run, 3}) - 1, numFrames);
152
153     for frameNum = inspectionStartFrame:inspectionEndFrame
154         if frameNum <= numFrames
155             frameFile = frameFiles(frameNum).name;
156             destination = fullfile(runFolder, frameFile);
157             movefile(fullfile(folderPath, frameFile), destination);
158             movedFiles(end+1).source = fullfile(folderPath, frameFile);
159             movedFiles(end).destination = destination;
160             processedFrames = processedFrames + 1;
161             waitbar(processedFrames / totalFramesToProcess, hWaitbar, ...
162                 sprintf('Processing frames... %d%% complete', round((
    processedFrames / totalFramesToProcess) * 100)));
163         end
164     end
165 end
166
167 close(hWaitbar);
168 disp('Frames have been sorted into respective run folders.');
169
170 % === SORTING & MOVING TIMER END ===
171 sortingTime = toc(sortingStart);
172 fprintf('Frame sorting and moving time: %.2f seconds (%.2f minutes).\n',
    sortingTime, sortingTime/60);
173
174 % Ask user for confirmation
175 answer = questdlg('Are you satisfied with the frame moving?', ...
176     'Confirmation', 'Yes', 'No', 'Yes');
177
178 if strcmp(answer, 'Yes')
179
180     % === DELETE SPIN-UP FRAMES ===
181     disp('Deleting unneeded spin-up frames...');
182     for run = 2:size(secondTable, 1)
183         spinUpStart = ceil(secondTable{run, 1});
184         inspectionStart = ceil(secondTable{run, 2});
185
186         for frameNum = spinUpStart:inspectionStart-1
187             if frameNum <= numFrames
188                 frameFile = frameFiles(frameNum).name;
189                 fileToDelete = fullfile(folderPath, frameFile);
190                 if exist(fileToDelete, 'file')
191                     delete(fileToDelete);
192                 end
193             end
194         end
195     end
196     disp('Spin-up frames deleted.');
197
198     % === RENAMING & ROTATION TIMER START ===
199     renamingStart = tic;
200
```

```
    % Renaming
    hWaitbar = waitbar(0, 'Renaming files...', 'Name', 'File Renaming Progress');
    totalFilesToRename = 0;

    for run = 2:size(secondTable, 1)
        if isempty(secondTable{run, 1}), break; end
        runFolder = fullfile(folderPath, sprintf('Test%04d', run-1));
        totalFilesToRename = totalFilesToRename + length(dir(fullfile(runFolder, '*.tiff')));
    end

    totalFilesToRename = totalFilesToRename + length(dir(fullfile(initialFramesFolder, '*.tiff')));
    renamedFiles = 0;

    for run = 2:size(secondTable, 1)
        if isempty(secondTable{run, 1}), break; end
        runFolder = fullfile(folderPath, sprintf('Test%04d', run-1));
        runFiles = dir(fullfile(runFolder, '*.tiff'));

        parfor k = 1:length(runFiles)
            oldName = fullfile(runFolder, runFiles(k).name);
            newName = fullfile(runFolder, sprintf('frame%04d.tiff', k));
            img = imread(oldName);
            rotatedImg = rot90(img, 2);
            imwrite(rotatedImg, newName);
            if ~strcmp(oldName, newName)
                delete(oldName);
            end
        end
    end

    % Rename initial frames
    initialFramesFiles = dir(fullfile(initialFramesFolder, '*.tiff'));
    parfor k = 1:length(initialFramesFiles)
        oldName = fullfile(initialFramesFolder, initialFramesFiles(k).name);
        newName = fullfile(initialFramesFolder, sprintf('frame%04d.tiff', k));
        img = imread(oldName);
        rotatedImg = rot90(img, 2);
        imwrite(rotatedImg, newName);
        if ~strcmp(oldName, newName)
            delete(oldName);
        end
    end

    close(hWaitbar);
    disp('Files have been renamed successfully.');

    % === RENAMING & ROTATION TIMER END ===
    renamingTime = toc(renamingStart);
    fprintf('Renaming + rotating time: %.2f seconds (%.2f minutes).\n',
```

```
    renamingTime, renamingTime/60);

    % Delete Initial Frames folder after renaming
    if exist(initialFramesFolder, 'dir')
        disp('Deleting Initial Frames folder...');
        rmdir(initialFramesFolder, 's');
        disp('Initial Frames folder deleted.');
    end

else
    % Revert
    hWaitbar = waitbar(0, 'Reverting files...', 'Name', 'Revert Progress')
    ;

    for k = 1:length(movedFiles)
        movefile(movedFiles(k).destination, movedFiles(k).source);
        waitbar(k / length(movedFiles), hWaitbar, ...
            sprintf('Reverting files... %d%% complete', round((k / length(
    movedFiles)) * 100)));
    end

    close(hWaitbar);
    disp('Frames reverted.');

    for k = 1:length(createdFolders)
        if exist(createdFolders{k}, 'dir')
            rmdir(createdFolders{k}, 's');
        end
    end
    disp('Created folders deleted.');
end

disp('Frame moving operation completed successfully.');

% === TOTAL TIMER END ===
totalTime = toc(overallStart);
fprintf('\n=== Total script execution time: %.2f seconds (%.2f minutes)
    ===\n', totalTime, totalTime/60);
```

Listing 2: Code used to sort the large number of TIFF frames into their respective test runs.

9.3 Background Subtraction Algorithm: main_.m

9.3.1 File 1: main_.m

```
1
2  % The main program to launch the tracking algorithm for the tests of
3  % interest. This particular program is for when bulk testing is being done
       .
4
5  tic;
6
7  clc;
8  close all;
9  clear variables;
10
11 % Ask the user to select the directory where all of the seperate test
12 % folders are stored (ExampleImages)
13 path = uigetdir('*','Select the folder');
14
15 % List all of the directories inside the selected directory. First list
       all
16 % of the folders, and then remove non-folder items and items starting with
17 % . and .. as these are not valid folders.
18
19 % NOTE, this is for looking at folders, where each folder contains more
20 % folders. Structure:
21 % Overall folder ----> Different tests (viscosity, particle size) --->
22 % different repeats of that test.
23 % So this finds all of the different tests to do
24 folders = dir(path);
25
26 for k = length(folders):-1:1
27     % remove non-folders
28     if ~folders(k).isdir
29         folders(k) = [ ];
30     end
31
32     % remove folders starting with .
33     fname = folders(k).name;
34     if fname(1) == '.'
35         folders(k) = [ ];
36     end
37
38     % remove if the folder is Results
39     if strcmp(fname,'Results')
40         folders(k) = [ ];
41     end
42 end
43
44 for k=1:length(folders)
45
46     % Now find all of the repeats zs to analyse inside of each test. The
47     % results of these can have statistics found etc, and should be output
48     % to a summary file or 2 summary files, one for all results and
```

```
    another
     % for the summaries
     subfolders = dir([path filesep folders(k).name]);

     for j = length(subfolders):-1:1
         % remove non-folders
         if ~subfolders(j).isdir
             subfolders(j) = [ ];
         end

         % remove folders starting with .
         fname = subfolders(j).name;
         if fname(1) == '.'
             subfolders(j) = [ ];
         end
     end

     % Create a structure to store all of the results output by the tracker
     % algorithm. For now these include max and min height and width, total
     % distance and transit time, but this could be expanded to the time
    where
     % max height is reached, etc.
     results = struct('maxHeight',[],'minHeight',[],'maxWidth',[],'minWidth
    ',[],'totalDistance',[],'transitTime',[]);

     % Loop through each of these folders, calling the tracker algorithm to
     % generate the desired outputs.
     t = zeros(length(subfolders),1);
     avgtime = 0;
     for i=1:length(subfolders)
         t1=tic;
         fprintf('The current test number is %d, %d \n',k, i);
         results(i) = tracker([path filesep folders(k).name],subfolders(i).
    name);
         t(i)=toc(t1);
         if mod(i,5) == 0
             avgtime = mean(t(1:i));
             timeremain = avgtime * (length(subfolders)-i);
             [hours,mins,secs] = sec2hms(timeremain);
             fprintf('\nThe time remaining is %02d:%02d:%02d\n\n', hours,
    mins, round(secs));
         end
     end

     % Create a structure to store the processed results in.
     resultsProcessed = struct('parameters',[],'mean',[],'standardDeviation
    ',[]);

     % Populate the parameters field, i.e. max height, min height, etc.
     for i=1:length(fieldnames(results))
         fields = fieldnames(results);
         resultsProcessed(i).parameters = fields(i);
     end

```

```
97      % % Find the mean values for each of the parameters
98      % for i=1:length(fieldnames(results))
99      %     resultsProcessed(i).mean = nanmean([results.(fields{i})]);
100     % end
101     %
102     % % Find the standard deviation for each of the parameters
103     % for i=1:length(fieldnames(results))
104     %     resultsProcessed(i).standardDeviation = nanstd([results.(fields{
        i})]);
105     % end
106     %
107     % if ~exist([path filesep 'Results'], 'dir')
108     %     mkdir([path filesep 'Results']);
109     % end
110     %
111     % save([path filesep 'Results' filesep folders(k).name '_results.mat
        '], 'results');
112     % save([path filesep 'Results' filesep folders(k).name '
        _resultsSummary.mat'], 'resultsProcessed');
113 end
114
115
116 t0_end = toc;
117 [hours2,mins2,secs2] = sec2hms(t0_end);
118
119 fprintf('\nThe time taken was %02d:%02d:%02d\n\n', hours2, mins2, round(
        secs2));
```

Listing 3: Code used run the backgrond subtraction algorithm.

9.3.2 File 2: sec2hms_.m

```
1
2 function [hours, mins, secs] = sec2hms(t)
3     hours = floor(t / 3600);
4     t = t - hours * 3600;
5     mins = floor(t / 60);
6     secs = t - mins * 60;
7 end
```

Listing 4: Time conversion function

9.3.3 File 3: track_.m

```
1
2 function tracks = track(xyzs,maxdisp,param)
3
4 % ; Open source implementation of Crocker and Grieg's algorithm,
      programmed
5 % ; in MATLAB by Daniel Blair and Eric Dufresne, used in this project to
6 % ; link particle positions across frames
7
```

```
8
9  % ; see http://glinda.lrsm.upenn.edu/~weeks/idl
10 % ;   for more information
11 % ;
12 % ;+
13 % ; NAME:
14 % ; track
15 % ; PURPOSE:
16 % ; Constructs n-dimensional trajectories from a scrambled list of
17 % ; particle coordinates determined at discrete times (e.g. in
18 % ; consecutive video frames).
19 % ; CATEGORY:
20 % ; Image Processing
21 % ; CALLING SEQUENCE:
22 % ; result = track( positionlist, maxdisp, param )
23 % ;  set all keywords in the space below
24 % ; INPUTS:
25 % ; positionlist: an array listing the scrambled coordinates and data
26 % ;      of the different particles at different times, such that:
27 % ;  positionlist(0:d-1,*): contains the d coordinates and
28 % ;      data for all the particles, at the different times. must be
    positve
29 % ;  positionlist(d,*): contains the time t that the position
30 % ;      was determined, must be integers (e.g. frame number.  These values
     must
31 % ;               be monotonically increasing and uniformly gridded in
    time.
32 % ; maxdisp: an estimate of the maximum distance that a particle
33 % ;      would move in a single time interval.(see Restrictions)
34 %  OPTIONAL INPUT:
35 %   param:  a structure containing a few tracking parameters that are
36 %       needed for many applications.  If param is not included in the
37 %       function call, then default values are used.  If you set one value
38 %       make sure you set them all:
39 % ;         param.mem: this is the number of time steps that a particle
    can be
40 % ;             'lost' and then recovered again.  If the particle
    reappears
41 % ;             after this number of frames has elapsed, it will be
42 % ;             tracked as a new particle. The default setting is zero.
43 % ;             this is useful if particles occasionally 'drop out' of
44 % ;             the data.
45 % ;         param.dim: if the user would like to unscramble non-coordinate
     data
46 % ;             for the particles (e.g. apparent radius of gyration for
47 % ;             the particle images), then positionlist should
48 % ;             contain the position data in positionlist(0:param.dim-1,*)
49 % ;             and the extra data in positionlist(param.dim:d-1,*). It is
     then
50 % ;             necessary to set dim equal to the dimensionality of the
51 % ;             coordinate data to so that the track knows to ignore the
52 % ;             non-coordinate data in the construction of the
53 % ;             trajectories. The default value is two.
54 % ;         param.good: set this keyword to eliminate all trajectories
```

```
   with
% ;              fewer than param.good valid positions.  This is useful
% ;              for eliminating very short, mostly 'lost' trajectories
% ;              due to blinking 'noise' particles in the data stream.
%;           param.quiet: set this keyword to 1 if you don't want any text
% ; OUTPUTS:
% ; result:  a list containing the original data rows sorted
% ;     into a series of trajectories.  To the original input
% ;     data structure there is appended an additional column
% ;     containing a unique 'id number' for each identified
% ;     particle trajectory.  The result array is sorted so
% ;     rows with corresponding id numbers are in contiguous
% ;     blocks, with the time variable a monotonically
% ;     increasing function inside each block.  For example:
% ;
% ;     For the input data structure (positionlist):
% ;         (x)      (y)      (t)
% ;     pos = 3.60000      5.00000      0.00000
% ;           15.1000      22.6000      0.00000
% ;           4.10000      5.50000      1.00000
% ;           15.9000      20.7000      2.00000
% ;           6.20000      4.30000      2.00000
% ;
% ;     IDL> res = track(pos,5,mem=2)
% ;
% ;     track will return the result 'res'
% ;         (x)      (y)      (t)          (id)
% ;     res = 3.60000      5.00000      0.00000      0.00000
% ;           4.10000      5.50000      1.00000      0.00000
% ;           6.20000      4.30000      2.00000      0.00000
% ;           15.1000      22.6000      0.00000      1.00000
% ;           15.9000      20.7000      2.00000      1.00000
% ;
% ;     NB: for t=1 in the example above, one particle temporarily
% ;     vanished.  As a result, the trajectory id=1 has one time
% ;     missing, i.e. particle loss can cause time gaps to occur
% ;     in the corresponding trajectory list. In contrast:
% ;
% ;     IDL> res = track(pos,5)
% ;
% ;     track will return the result 'res'
% ;         (x)      (y)      (t)          (id)
% ;     res = 15.1000      22.6000      0.00000      0.00000
% ;                   3.60000      5.00000      0.00000      1.00000
% ;               4.10000      5.50000      1.00000      1.00000
% ;               6.20000      4.30000      2.00000      1.00000
% ;               15.9000      20.7000      2.00000      2.00000
% ;
% ;     where the reappeared 'particle' will be labelled as new
% ;     rather than as a continuation of an old particle since
% ;     mem=0.  It is up to the user to decide what setting of
% ;     'mem' will yeild the highest fidelity .
% ;
% ; SIDE EFFECTS:
```

```
% ; Produces informational messages.  Can be memory intensive for
% ; extremely large data sets.
% ; RESTRICTIONS:
% ; maxdisp should be set to a value somewhat less than the mean
% ; spacing between the particles. As maxdisp approaches the mean
% ; spacing the runtime will increase significantly. The function
% ; will produce an error message: "Excessive Combinatorics!" if
% ; the run time would be too long, and the user should respond
% ; by re-executing the function with a smaller value of maxdisp.
% ; Obviously, if the particles being tracked are frequently moving
% ; as much as their mean separation in a single time step, this
% ; function will not return acceptable trajectories.
% ; PROCEDURE:
% ; Given the positions for n particles at time t(i), and m possible
% ; new positions at time t(i+1), this function considers all possible
% ; identifications of the n old positions with the m new positions,
% ; and chooses that identification which results in the minimal total
% ; squared displacement. Those identifications which don't associate
% ; a new position within maxdisp of an old position ( particle loss )
% ; penalize the total squared displacement by maxdisp^2. For non-
% ; interacting Brownian particles with the same diffusivity, this
% ; algorithm will produce the most probable set of identifications
% ; ( provided maxdisp >> RMS displacement between frames ).
% ; In practice it works reasonably well for systems with oscillatory,
% ; ballistic, correlated and random hopping motion, so long as single
% ; time step displacements are reasonably small.  NB: multidimensional
% ; functionality is intended to facilitate tracking when additional
% ; information regarding target identity is available (e.g. size or
% ; color).  At present, this information should be rescaled by the
% ; user to have a comparable or smaller (measurement) variance than
% ; the spatial displacements.
% ;
% ; MODIFICATION HISTORY:
% ;  2/93 Written by John C. Crocker, University of Chicago (JFI).
% ;  7/93 JCC fixed bug causing particle loss and improved performance
% ;     for large numbers of (>100) particles.
% ; 11/93 JCC improved speed and memory performance for large
% ;     numbers of (>1000) particles (added subnetwork code).
% ;  3/94 JCC optimized run time for trivial bonds and d<7. (Added
% ;     d-dimensional raster metric code.)
% ;  8/94 JCC added functionality to unscramble non-position data
% ;     along with position data.
% ;  9/94 JCC rewrote subnetwork code and wrote new, more efficient
% ;     permutation code.
% ;  5/95 JCC debugged subnetwork and excessive combinatorics code.
% ; 12/95 JCC added memory keyword, and enabled the tracking of
% ;     newly appeared particles.
% ;  3/96 JCC made inipos a keyword, and disabled the adding of 'new'
% ;     particles when inipos was set.
% ;  3/97 JCC added 'add' keyword, since Chicago users didn't like
% ;     having particle addition be the default.
% ;  9/97 JCC added 'goodenough' keyword to improve memory efficiency
% ;     when using the 'add' keyword and to filter out bad tracks.
% ;       10/97 JCC streamlined data structure to speed runtime for >200
```

```
% ;                timesteps.  Changed 'quiet' keyword to 'verbose'. Made
% ;                time labelling more flexible (uniform and sorted is ok).
% ;  9/98 JCC switched trajectory data structure to a 'list' form,
% ;     resolving memory issue for large, noisy datasets.
% ;  2/99 JCC added Eric Weeks's 'uberize' code to post-facto
% ;     rationalize the particle id numbers, removed 'add' keyword.
% ;  1/05 Transmuted to MATLAB by D. Blair
% ;  5/05  ERD Added the param structure to simplify calling.
%    6/05  ERD Added quiet to param structure
%    7/05  DLB Fixed slight bug in trivial bond code
%    3/07  DLB Fixed bug with max disp pointed out by Helene Delanoe-Ayari
%
% ; This code 'track.pro' is copyright 1999, by John C. Crocker.
% ; It should be considered 'freeware'- and may be distributed freely
% ; (outside of the military-industrial complex) in its original form
% ; when properly attributed.
% ;
% ;-

dd = length(xyzs(1,:));

%use default parameters if none given
if nargin==2
    %default values
    memory_b=0; % if mem is not needed set to zero
    goodenough = 0;  % if goodenough is not wanted set to zero
    dim = dd - 1;
    quiet=0;
else
    memory_b     =   param.mem;
    goodenough   =   param.good;
    dim          =   param.dim;
    quiet        =   param.quiet;
end


% checking the input time vector
t = xyzs(:,dd);
st = circshift(t,1);
st = t(2:end) - st(2:end);
if  sum(st(find(st < 0))) ~= 0
    disp('The time vectors is not in order')
    return
end
info = 1;

w = find(st > 0);
z = length(w);
z = z +1;
if isempty(w)
    disp('All positions are at the same time... go back!')
    return
end

```

```
% partitioning the data with unique times

%res = unq(t);
% implanting unq directly
    indices = find(t ~= circshift(t,-1));
        count = length(indices);
        if count > 0
            res = indices;
        else
            res = length(t) -1;
        end
 %%%%%%%%%%%%%%%%%%%%%%%%

res = [1,res',length(t)];
ngood = res(2) - res(1) + 1;
eyes = 1:ngood;
pos = xyzs(eyes,1:dim);
istart = 2;
n = ngood;

zspan = 50;
if n > 200
    zspan = 20;
end
if n > 500
    zspan = 10;
end
resx = zeros(zspan,n) - 1;

bigresx = zeros(z,n) - 1;
mem = zeros(n,1);
%  whos resx
%  whos bigresx
uniqid = 1:n;
maxid = n;
olist = [0.,0.];

if goodenough > 0
    dumphash = zeros(n,1);
    nvalid = ones(n,1);
end

%  whos eyes;
resx(1,:) = eyes;
% setting up constants
maxdisq = maxdisp^2;

% John calls this the setup for "fancy code" ???
notnsqrd = (sqrt(n*ngood) > 200) & (dim < 7);
notnsqrd = notnsqrd(1);

if notnsqrd
    %;   construct the vertices of a 3x3x3... d-dimensional hypercube

```

```
270     cube = zeros(3^dim,dim);
271
272
273     for d=0:dim-1,
274         numb = 0;
275         for j=0:(3^d):(3^dim)-1,
276             cube(j+1:j+(3^(d)),d+1) = numb;
277             numb = mod(numb+1,3);
278         end
279     end
280
281     %   calculate a blocksize which may be greater than maxdisp, but which
282     %   keeps nblocks reasonably small.
283
284     volume = 1;
285     for d = 0:dim-1
286         minn = min(xyzs(w,d+1));
287         maxx = max(xyzs(w,d+1));
288         volume = volume * (maxx-minn);
289     end
290     volume;
291     blocksize = max( [maxdisp,((volume)/(20*ngood))^(1.0/dim)] );
292 end
293 %   Start the main loop over the frames.
294 for i=istart:z
295     ispan = mod(i-1,zspan)+1;
296     %disp(ispan)
297     % get new particle positions
298     m = res(i+1) - res(i);
299     res(i);
300     eyes = 1:m;
301     eyes = eyes + res(i);
302
303     if m > 0
304
305         xyi = xyzs(eyes,1:dim);
306         found = zeros(m,1);
307
308         % THE TRIVIAL BOND CODE BEGINS
309
310         if notnsqrd
311             %Use the raster metric code to do trivial bonds
312
313             % construct "s", a one dimensional parameterization of the
    space
314             % which consists of the d-dimensional raster scan of the
    volume.)
315
316             abi = fix(xyi./blocksize);
317             abpos = fix(pos./blocksize);
318             si = zeros(m,1);
319             spos = zeros(n,1);
320             dimm = zeros(dim,1);
321             coff = 1.;
```

```
322
323             for j=1:dim
324                 minn = min([abi(:,j);abpos(:,j)]);
325                 maxx = max([abi(:,j);abpos(:,j)]);
326                 abi(:,j) = abi(:,j) - minn;
327                 abpos(:,j) = abpos(:,j) - minn;
328                 dimm(j) = maxx-minn + 1;
329                 si = si + abi(:,j).*coff;
330                 spos = spos + abpos(:,j).*coff;
331                 coff = dimm(j).*coff;
332             end
333             nblocks = coff;
334             % trim down (intersect) the hypercube if its too big to fit in
     the
335             % particle volume. (i.e. if dimm(j) lt 3)
336
337             cub = cube;
338             deg = find( dimm < 3);
339             if ~isempty(deg)
340                 for j = 0:length(deg)-1
341                     cub = cub(find(cub(:,deg(j+1)) < dimm(deg(j+1))),:);
342                 end
343             end
344
345             % calculate the "s" coordinates of hypercube (with a corner @
    the origin)
346             scube = zeros(length(cub(:,1)),1);
347             coff = 1;
348             for j=1:dim
349                 scube = scube + cub(:,j).*coff;
350                 coff = coff*dimm(j);
351             end
352
353             % shift the hypercube "s" coordinates to be centered around
    the origin
354
355             coff = 1;
356             for j=1:dim
357                 if dimm(j) > 3
358                     scube = scube - coff;
359                 end
360                 coff = dimm(j).* coff;
361             end
362             scube = mod((scube + nblocks),nblocks);
363             % get the sorting for the particles by their "s" positions.
364             [ed,isort] = sort(si);
365
366             % make a hash table which will allow us to know which new
    particles
367             % are at a given si.
368             strt = zeros(nblocks,1) -1;
369             fnsh = zeros(nblocks,1);
370             h = find(si == 0);
371             lh = length(h);
```

```
372             if lh > 0
373
374             si(h) = 1;
375             end
376
377             for j=1:m
378                 if strt(si(isort(j))) == -1
379                     strt(si(isort(j))) = j;
380                     fnsh(si(isort(j))) = j;
381                 else
382                     fnsh(si(isort(j))) = j;
383                 end
384             end
385             if lh > 0
386             si(h) = 0;
387             end
388             coltot = zeros(m,1);
389             rowtot = zeros(n,1);
390             which1 = zeros(n,1);
391             for j=1:n
392
393
394                 map = fix(-1);
395
396                 scub_spos = scube + spos(j);
397                 s = mod(scub_spos,nblocks);
398                 whzero = find(s == 0 );
399                 if ~isempty(whzero)
400                     nfk = find(s ~=0);
401                     s = s(nfk);
402                 end
403
404                 w = find(strt(s) ~= -1);
405
406                 ngood = length(w);
407                 ltmax=0;
408                 if ngood ~= 0
409
410                     s = s(w);
411                     for k=1:ngood
412                         map = [map;isort( strt(s(k)):fnsh(s(k)))];
413                     end
414                     map = map(2:end);
415 %                     if length(map) == 2
416 %                         if (map(1) - map(2)) == 0
417 %                             map = unique(map);
418 %                          end
419 %                     end
420                     %   map = map(umap);
421                     %end
422                     % find those trival bonds
423                     distq = zeros(length(map),1);
424                     for d=1:dim
425                         distq = distq + (xyi(map,d) - pos(j,d)).^2;
```

```
426                     end
427                     ltmax = distq < maxdisq;
428
429                     rowtot(j) = sum(ltmax);
430
431                     if rowtot(j) >= 1
432                         w = find(ltmax == 1);
433                         coltot( map(w) ) = coltot( map(w)) +1;
434                         which1(j) = map( w(1) );
435                     end
436                 end
437
438             end
439
440
441             ntrk = fix(n - sum(rowtot == 0));
442
443             w = find( rowtot == 1);
444             ngood = length(w);
445
446
447             if ngood ~= 0
448                 ww = find(coltot( which1(w) ) == 1);
449                 ngood = length(ww);
450                 if ngood ~= 0
451                     %disp(size(w(ww)))
452                     resx(ispan,w(ww)) = eyes( which1(w(ww)));
453                     found( which1( w(ww))) = 1;
454                     rowtot( w(ww)) = 0;
455                     coltot( which1(w(ww))) = 0;
456                 end
457             end
458
459             labely = find( rowtot > 0);
460             ngood = length(labely);
461             if ngood ~= 0
462                 labelx = find( coltot > 0);
463
464                 nontrivial = 1;
465             else
466                 nontrivial = 0;
467             end
468
469         else
470
471             %   or: Use simple N^2 time routine to calculate trivial bonds
472
473             % let's try a nice, loopless way!
474             % don't bother tracking perm. lost guys.
475             wh = find( pos(:,1) >= 0);
476             ntrack = length(wh);
477             if ntrack == 0
478                 'There are no valid particles to track idiot!'
479                 break
```

```
480             end
481             xmat = zeros(ntrack,m);
482             count = 0;
483             for kk=1:ntrack
484                 for ll=1:m
485                     xmat(kk,ll) = count;
486                     count = count+1;
487                 end
488             end
489             count = 0;
490             for kk=1:m
491                 for ll=1:ntrack
492                     ymat(kk,ll) = count;
493                     count = count+1;
494                 end
495             end
496
497             xmat = (mod(xmat,m) + 1);
498             ymat = (mod(ymat,ntrack) +1)';
499             [lenxn,lenxm] = size(xmat);
500 %            whos ymat
501 %            whos xmat
502 %            disp(m)
503
504             for d=1:dim
505                 x = xyi(:,d);
506                 y = pos(wh,d);
507                 xm = x(xmat);
508                 ym = y(ymat(1:lenxn,1:lenxm));
509                 if size(xm) ~= size(ym)
510                     xm = xm';
511                 end
512
513                 if d == 1
514                     dq = (xm -ym).^2;
515                     %dq = (x(xmat)-y(ymat(1:lenxn,1:lenxm))).^2;
516                 else
517                     dq = dq + (xm-ym).^2;
518                     %dq = dq + (x(xmat)-y(ymat(1:lenxn,1:lenxm)) ).^2;
519                 end
520             end
521
522             ltmax = dq < maxdisq;
523
524             % figure out which trivial bonds go with which
525
526             rowtot = zeros(n,1);
527             rowtot(wh) = sum(ltmax,2);
528
529
530             if ntrack > 1
531                 coltot = sum(ltmax,1);
532             else
533                 coltot = ltmax;
```

```
534             end
535             which1 = zeros(n,1);
536             for j=1:ntrack
537                 [mx, w] = max(ltmax(j,:));
538                 which1(wh(j)) = w;
539             end
540 
541             ntrk = fix( n - sum(rowtot == 0));
542             w= find( rowtot == 1) ;
543             ngood = length(w);
544             if ngood ~= 0
545                 ww = find(coltot(which1(w)) == 1);
546                 ngood = length(ww);
547                 if ngood ~= 0
548                     resx( ispan, w(ww) ) = eyes( which1( w(ww)));
549                     found(which1( w(ww))) = 1;
550                     rowtot(w(ww)) = 0;
551                     coltot(which1(w(ww))) = 0;
552                 end
553             end
554 
555             labely = find( rowtot > 0);
556             ngood = length(labely);
557 
558             if ngood ~= 0
559                 labelx = find( coltot > 0);
560                 nontrivial = 1;
561             else
562                 nontrivial = 0;
563             end
564         end
565 
566         %THE TRIVIAL BOND CODE ENDS
567 
568         if nontrivial
569 
570             xdim = length(labelx);
571             ydim = length(labely);
572 
573             %  make a list of the non-trivial bonds
574 
575             bonds = zeros(1,2);
576             bondlen = 0;
577 
578             for j=1:ydim
579                 distq = zeros(xdim,1);
580 
581                 for d=1:dim
582                     %distq
583                     distq = distq + (xyi(labelx,d) - pos(labely(j),d)).^2;
584                     %distq
585                 end
586 
587                 w= find(distq <  maxdisq)' - 1;
```

```
588                 ngood = length(w);
589                 newb = [w;(zeros(1,ngood)+j)];
590 
591 
592                 bonds = [bonds;newb'];
593 
594                 bondlen = [ bondlen;distq( w + 1) ];
595 
596             end
597             bonds = bonds(2:end,:);
598 
599             bondlen = bondlen(2:end);
600             numbonds = length(bonds(:,1));
601             mbonds = bonds;
602             max([xdim,ydim]);
603 
604 
605             if max([xdim,ydim]) < 4
606                 nclust = 1;
607                 maxsz = 0;
608                 mxsz = xdim;
609                 mysz = ydim;
610                 bmap = zeros(length(bonds(:,1)+1),1) - 1;
611 
612             else
613 
614 
615                 %   THE SUBNETWORK CODE BEGINS
616                 lista = zeros(numbonds,1);
617                 listb = zeros(numbonds,1);
618                 nclust = 0;
619                 maxsz = 0;
620                 thru = xdim;
621 
622                 while thru ~= 0
623                     %  the following code extracts connected
624                     %   sub-networks of the non-trivial
625                     %   bonds.  NB: lista/b can have redundant entries due
    to
626                     %   multiple-connected subnetworks
627 
628 
629                     w = find(bonds(:,2) >= 0);
630 %                    size(w)
631 
632                     lista(1) = bonds(w(1),2);
633                     listb(1) = bonds(w(1),1);
634                     bonds(w(1),:) = -(nclust+1);
635                     bonds;
636                     adda = 1;
637                     addb = 1;
638                     donea = 0;
639                     doneb = 0;
640                     if (donea ~= adda) | (doneb ~= addb)
```

```
641                         true = 0;
642                     else
643                     true = 1;
644                     end
645
646                     while ~true
647
648                         if (donea ~= adda)
649                             w = find(bonds(:,2) == lista(donea+1));
650                             ngood = length(w);
651                             if ngood ~= 0
652                                 listb(addb+1:addb+ngood,1) = bonds(w,1);
653                                 bonds(w,:) = -(nclust+1);
654                                 addb = addb+ngood;
655                             end
656                             donea = donea+1;
657                         end
658                         if (doneb ~= addb)
659                             w = find(bonds(:,1) == listb(doneb+1));
660                             ngood = length(w);
661                             if ngood ~= 0
662                                 lista(adda+1:adda+ngood,1) = bonds(w,2);
663                                 bonds(w,:) = -(nclust+1);
664                                 adda = adda+ngood;
665                             end
666                             doneb = doneb+1;
667                         end
668                       if (donea ~= adda) | (doneb ~= addb)
669                           true = 0;
670                       else
671                           true = 1;
672                       end
673                     end
674
675                     [pp,pqx] = sort(listb(1:doneb));
676                     %unx =  unq(listb(1:doneb),pqx);
677                     %implanting unq directly
678                         arr = listb(1:doneb);
679                         q = arr(pqx);
680                         indices = find(q ~= circshift(q,-1));
681                         count = length(indices);
682                         if count > 0
683                             unx = pqx(indices);
684                         else
685                             unx = length(q) -1;
686                         end
687                     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
688
689                     xsz = length(unx);
690                     [pp,pqy] = sort(lista(1:donea));
691                     %uny =  unq(lista(1:donea),pqy);
692                     %implanting unq directly
693                         arr = lista(1:donea);
694                         q = arr(pqy);
```

```
695                         indices = find(q ~= circshift(q,-1));
696                         count = length(indices);
697                         if count > 0
698                             uny = pqy(indices);
699                         else
700                             uny = length(q) -1;
701                         end
702                     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
703
704
705
706
707                     ysz = length(uny);
708                     if xsz*ysz > maxsz
709                         maxsz = xsz*ysz;
710                         mxsz = xsz;
711                         mysz = ysz;
712                     end
713
714
715                     thru = thru -xsz;
716                     nclust = nclust + 1;
717                 end
718                 bmap = bonds(:,2);
719             end
720             % THE SUBNETWORK CODE ENDS
721             % put verbose in for Jaci
722
723             %   THE PERMUTATION CODE BEGINS
724
725             for nc =1:nclust
726                 w = find( bmap == -1*(nc));
727
728                 nbonds = length(w);
729                 bonds = mbonds(w,:);
730                 lensq = bondlen(w);
731                 [pq,st] = sort( bonds(:,1));
732                 %un = unq(bonds(:,1),st);
733                    %implanting unq directly
734                         arr = bonds(:,1);
735                         q = arr(st);
736                         indices = find(q ~= circshift(q,-1));
737                         count = length(indices);
738                         if count > 0
739                             un = st(indices);
740                         else
741                             un = length(q) -1;
742                         end
743                     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
744
745
746                 uold = bonds(un,1);
747
748                 nold = length(uold);
```

```
749
750                 %un = unq(bonds(:,2));
751
752                 %implanting unq directly
753                 indices = find(bonds(:,2) ~= circshift(bonds(:,2),-1));
754                 count = length(indices);
755                     if count > 0
756                         un = indices;
757                     else
758                         un = length(bonds(:,2)) -1;
759                     end
760                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
761
762                 unew = bonds(un,2);
763                 nnew = length(unew);
764
765                 if nnew > 5
766                     rnsteps = 1;
767                     for ii =1:nnew
768                         rnsteps = rnsteps * length( find(bonds(:,2) == ...
769                             unew(ii)));
770                         if rnsteps > 5.e+4
771                             disp('Warning: difficult combinatorics
    encountered.')
772                         end
773                         if rnsteps > 2.e+5
774                             disp(['Excessive Combinitorics you FOOL LOOK
    WHAT YOU HAVE' ...
775                                 ' DONE TO ME!!!'])
776                             tracks = 0;
777                             return
778                         end
779                     end
780                 end
781                 st = zeros(nnew,1);
782                 fi = zeros(nnew,1);
783                 h = zeros(nbonds,1);
784                 ok = ones(nold,1);
785                 nlost = (nnew - nold) > 0;
786
787
788                 for ii=1:nold
789                     h(find(bonds(:,1) == uold(ii))) = ii;
790                 end
791                 st(1) = 1 ;
792                 fi(nnew) = nbonds; % check this later
793                 if nnew > 1
794                     sb = bonds(:,2);
795                     sbr = circshift(sb,1);
796                     sbl = circshift(sb,-1);
797                     st(2:end) = find( sb(2:end) ~= sbr(2:end)) + 1;
798                     fi(1:nnew-1) = find( sb(1:nbonds-1) ~= sbl(1:nbonds-1)
    );
799                 end
```

```
%                   if i-1 == 13
%                       hi
%                   end
                   checkflag = 0;
                   while checkflag ~= 2

                       pt = st -1;
                       lost = zeros(nnew,1);
                       who = 0;
                       losttot = 0;
                       mndisq = nnew*maxdisq;


                       while who ~= -1

                           if pt(who+1) ~= fi(who+1)


                               w = find( ok( h( pt( who+1 )+1:fi( who+1 ) ) ) ); % check this -1
                               ngood = length(w);
                               if ngood > 0
                                   if pt(who+1) ~= st(who+1)-1
                                       ok(h(pt(who+1))) = 1;
                                   end
                                   pt(who+1) = pt(who+1) + w(1);
                                   ok(h(pt(who+1))) = 0;
                                   if who == nnew -1
                                       ww = find( lost == 0);
                                       dsq = sum(lensq(pt(ww))) + losttot*maxdisq;

                                       if dsq < mndisq
                                           minbonds = pt(ww);
                                           mndisq = dsq;
                                       end
                                   else
                                       who = who+1;
                                   end
                               else
                                   if ~lost(who+1) & (losttot ~= nlost)
                                       lost(who+1) = 1;
                                       losttot = losttot + 1;
                                       if pt(who+1) ~= st(who+1) -1;
                                           ok(h(pt(who+1))) = 1;
                                       end
                                       if who == nnew-1
                                           ww = find( lost == 0);
                                           dsq = sum(lensq(pt(ww))) + losttot*maxdisq;
                                           if dsq < mndisq
                                               minbonds = pt(ww);
                                               mndisq = dsq;
                                           end
```

```
851                                     else
852                                        who = who + 1;
853                                     end
854
855                                 else
856                                     if pt(who+1) ~= (st(who+1) -1)
857                                         ok(h(pt(who+1))) = 1;
858                                     end
859                                     pt(who+1) = st(who+1) -1;
860                                     if lost(who+1)
861                                         lost(who+1) = 0;
862                                         losttot = losttot -1;
863                                     end
864                                     who = who -1;
865                                 end
866                             end
867                         else
868                             if ~lost(who+1) & (losttot ~= nlost)
869                                 lost(who+1) = 1;
870                                 losttot = losttot + 1;
871                                 if pt(who+1) ~= st(who+1)-1
872                                     ok(h(pt(who+1))) = 1;
873                                 end
874                                 if who == nnew -1
875                                     ww = find( lost == 0);
876                                     dsq = sum(lensq(pt(ww))) + losttot*
    maxdisq;
877
878                                     if dsq < mndisq
879                                         minbonds = pt(ww);
880                                         mndisq = dsq;
881                                     end
882                                 else
883                                     who = who + 1;
884                                 end
885                             else
886                                 if pt(who+1) ~= st(who+1) -1
887                                     ok(h(pt(who+1))) = 1;
888                                 end
889                                 pt(who+1) = st(who+1) -1;
890                                 if lost(who+1)
891                                     lost(who+1) = 0;
892                                     losttot = losttot -1;
893                                 end
894                                 who = who -1;
895                             end
896                         end
897                     end
898
899                     checkflag = checkflag + 1;
900                     if checkflag == 1
901                         plost = min([fix(mndisq/maxdisq) , (nnew -1)]);
902                         if plost > nlost
903                             nlost = plost;
```

```
                        else
                            checkflag = 2;
                        end
                    end

                end
                %   update resx using the minimum bond configuration

                resx(ispan,labely(bonds(minbonds,2))) = eyes(labelx(bonds(
minbonds,1)+1));
                found(labelx(bonds(minbonds,1)+1)) = 1;

            end

            %   THE PERMUTATION CODE ENDS
        end

        w = find(resx(ispan,:) >= 0);
        nww = length(w);

        if nww > 0
            pos(w,:) = xyzs( resx(ispan,w) , 1:dim);
            if goodenough > 0
                nvalid(w) = nvalid(w) + 1;
            end
        end  %go back and add goodenough keyword thing
        newguys = find(found == 0);

       nnew = length(newguys);

        if (nnew > 0) % & another keyword to workout inipos
            newarr = zeros(zspan,nnew) -1;
            resx = [resx,newarr];

            resx(ispan,n+1:end) = eyes(newguys);
            pos = [[pos];[xyzs(eyes(newguys),1:dim)]];
            nmem = zeros(nnew,1);
            mem = [mem;nmem];
            nun = 1:nnew;
            uniqid = [uniqid,((nun) + maxid)];
            maxid = maxid + nnew;
            if goodenough > 0
                dumphash = [dumphash;zeros(1,nnew)'];
                nvalid = [nvalid;zeros(1,nnew)'+1];
            end
            % put in goodenough
            n = n + nnew;

        end

    else
        ' Warning- No positions found for t='
    end
    w = find( resx(ispan,:) ~= -1);
```

```
957     nok = length(w);
958     if nok ~= 0
959         mem(w) =0;
960     end
961 
962     mem = mem + (resx(ispan,:)' == -1);
963     wlost = find(mem == memory_b+1);
964     nlost =length(wlost);
965 
966     if nlost > 0
967         pos(wlost,:) = -maxdisp;
968         if goodenough > 0
969             wdump = find(nvalid(wlost) < goodenough);
970             ndump = length(wdump);
971             if ndump > 0
972                 dumphash(wlost(wdump)) = 1;
973             end
974         end
975         % put in goodenough keyword stuff if
976     end
977     if (ispan == zspan) | (i == z)
978         nold = length(bigresx(1,:));
979         nnew = n-nold;
980         if nnew > 0
981             newarr = zeros(z,nnew) -1;
982             bigresx = [bigresx,newarr];
983         end
984         if goodenough > 0
985             if (sum(dumphash)) > 0
986                 wkeep = find(dumphash == 0);
987                 nkeep = length(wkeep);
988                 resx = resx(:,wkeep);
989                 bigresx = bigresx(:,wkeep);
990                 pos = pos(wkeep,:);
991                 mem = mem(wkeep);
992                 uniqid = uniqid(wkeep);
993                 nvalid = nvalid(wkeep);
994                 n = nkeep;
995                 dumphash = zeros(nkeep,1);
996             end
997         end
998 
999         % again goodenough keyword
1000        if quiet~=1
1001            disp(strcat(num2str(i), ' of ' ,num2str(z), ' done.  Tracking
    ',num2str(ntrk),' particles  ', num2str(n),' tracks total'));
1002        end
1003        bigresx(i-(ispan)+1:i,:) = resx(1:ispan,:);
1004        resx = zeros(zspan,n) - 1;
1005 
1006 
1007        wpull = find(pos(:,1) == -maxdisp);
1008        npull = length(wpull);
1009 
```

```
1010        if npull > 0
1011            lillist = zeros(1,2);
1012            for ipull=1:npull
1013                wpull2 = find(bigresx(:,wpull(ipull)) ~= -1);
1014                npull2 = length(wpull2);
1015                thing = [bigresx(wpull2,wpull(ipull)),zeros(npull2,1)+
    uniqid(wpull(ipull))];
1016                lillist = [lillist;thing];
1017
1018            end
1019            olist = [[olist];[lillist(2:end,:)]];
1020
1021        end
1022
1023
1024
1025        wkeep = find(pos(:,1) >= 0);
1026        nkeep = length(wkeep);
1027        if nkeep == 0
1028                'Were going to crash now, no particles....'
1029        end
1030        resx = resx(:,wkeep);
1031        bigresx = bigresx(:,wkeep);
1032        pos = pos(wkeep,:);
1033        mem = mem(wkeep);
1034        uniqid = uniqid(wkeep);
1035        n = nkeep;
1036        dumphash = zeros(nkeep,1);
1037        if goodenough > 0
1038            nvalid = nvalid(wkeep);
1039        end
1040    end
1041
1042 end
1043
1044 if goodenough > 0
1045     nvalid = sum(bigresx >= 0 ,1);
1046     wkeep = find(nvalid >= goodenough);
1047     nkeep = length(wkeep);
1048     if nkeep == 0
1049         for i=1:10
1050         disp('You are not going any further, check your params and data')
1051         end
1052         disp('the code broke at line 1045')
1053         return
1054     end
1055     if nkeep < n
1056         bigresx = bigresx(:,wkeep);
1057         n = nkeep;
1058         uniqid = uniqid(wkeep);
1059         pos = pos(wkeep,:);
1060     end
1061 end
1062
```

```
1063
1064 wpull = find( pos(:,1) ~= -2*maxdisp);
1065 npull = length(wpull);
1066 if npull > 0
1067     lillist = zeros(1,2);
1068     for ipull=1:npull
1069         wpull2 = find(bigresx(:,wpull(ipull)) ~= -1);
1070         npull2 = length(wpull2);
1071         thing = [bigresx(wpull2,wpull(ipull)),zeros(npull2,1)+uniqid(wpull
     (ipull))];
1072         lillist = [lillist;thing];
1073     end
1074     olist = [olist;lillist(2:end,:)];
1075 end
1076
1077 olist = olist(2:end,:);
1078 %bigresx = 0;
1079 %resx = 0;
1080
1081 nolist = length(olist(:,1));
1082 res = zeros(nolist,dd+1);
1083 for j=1:dd
1084     res(:,j) = xyzs(olist(:,1),j);
1085 end
1086 res(:,dd+1) = olist(:,2);
1087
1088 % this is uberize included for simplicity of a single monolithic code
1089
1090 ndat=length(res(1,:));
1091 newtracks=res;
1092
1093
1094 %u=unq(newtracks(:,ndat));
1095
1096 % inserting unq
1097 indices = find(newtracks(:,ndat) ~= circshift(newtracks(:,ndat),-1));
1098         count = length(indices);
1099         if count > 0
1100             u = indices;
1101         else
1102             u = length(newtracks(:,ndat)) -1;
1103         end
1104
1105
1106 ntracks=length(u);
1107 u=[0;u];
1108 for i=2:ntracks+1
1109     newtracks(u(i-1)+1:u(i),ndat) = i-1;
1110 end
1111
1112 % end of uberize code
1113
```

```
1114 tracks = newtracks;
```

Listing 5: Open source implementation of Crocker and Grieg's algorithm used to creae particle tracks

9.3.4 File 4: tracker_.m

```
1
2  function [results] = tracker(folderPath,folderName)
3      close all;
4
5      % A constant to say whether to print messages to the command line or
6      % not
7      quiet = 0;
8
9      % Set the path for the folder being tested. Useful passing the test
10     % name separately as this is used to give the graphs and videos their
11     % names
12     path = [folderPath filesep folderName];
13
14     % Constants to determine whether to
15     % (a) write a single AVI of the trajectory
16     % (b) write an image of the x-y plot
17     % (c) write the X-Y Plot txt file
18     % (d) show a frame-by-frame video of the trajectory
19     % (e) Write the frames to a folder.
20     % If they are 0 it will not do it, if 1 it will
21     writeAVI = 1;
22     writeImage = 1;
23     writeTxt = 1;
24     showVid=0;
25     writeFrames=0;
26
27  %%
28
29     % ------------------------- Load VIDEO FILE -------------------------
       %
30
31     % Find the files inside the folder of interest
32     files = dir([path filesep '*.tiff']);
33
34     frame1 = imread([path filesep files(1).name]);
35
36     % Get number of pixels in height and width of the video
37     vidHeight = size(frame1,2);
38     vidWidth = size(frame1,1);
39
40     fprintf('Video Height in Pixels: %d\n', vidHeight);
41     fprintf('Video Width in Pixels: %d\n', vidWidth);
42
43     % Get the frame rate
44     frameRate = 75;
45     fprintf('Frame Rate: %d\n', frameRate);
```

```
46      % Get the number of frames in the video
47      numberOfFrames = length(files);
48      fprintf('Number of Frames: %d\n', numberOfFrames);
49
50  %%
51
52      % --------------------------- FIND ROI ------------------------------
        %
53
54      % Temporatily read in the middle frame of the video, rotate the image
55      % 90 degrees counterclockwise so it is position like in real life. The
56      % whole video will be rotated the same way later so it is between to
57      % get the max and min col and row positions in terms of this
58      % orientation. Half way through ensures that the syringe has stopped
59      % spinning, so the boundaries of the syringe should not change
60      % noticably between here and any other frame
61       temp = imrotate(frame1,-90);
62
63      % Find the rows and columns of any cells that contain a value less
        than
64      % or equal to 30 (close to complete black). The boundaries of the
65      % syringe (plunger, walls and cap) are all black, so finding the
        places
66      % where these are allow a smaller chunk of the video to be analysed,
67      % saving memory
68      [rows,cols] = find(temp<=30);
69
70      % From these vectors, extract the minimum and maximum values for both
71      % the columns and rows, giving the boundaries of the syringe and hence
72      % the ROI. While this will work with the clamp in view, it wont save
73      % any space as the whole clamp appears black in the videos.
74      mincol = min(cols);
75      maxcol = max(cols);
76      minrow = min(rows);
77      maxrow = max(rows);
78
79      % ROIHeight = 1087;
80      % ROIWidth = 349;
81      ROIHeight = maxrow-minrow+1;
82      ROIWidth = (maxcol-mincol+1)/3;
83
84      % Print the ROI boundaries and dimensions
85      fprintf('Min Column: %d\n', mincol);
86      fprintf('Max Column: %d\n', maxcol);
87      fprintf('Min Row: %d\n', minrow);
88      fprintf('Max Row: %d\n', maxrow);
89      fprintf('ROI Height: %d\n', ROIHeight);
90      fprintf('ROI Width: %d\n', ROIWidth);
91
92      % Plot ROI on original frame
93      % fig1=figure;
94      % imshow(imrotate(frame1,-90));
95      % hold on;
96      % rectangle('Position', [mincol, minrow, ROIWidth, ROIHeight], '
```

```
   EdgeColor', 'r', 'LineWidth', 2);
97    % title('Region of Interest (ROI)');
98    % hold off;
99 %
100
101
102    % ------ CONVERT VIDEO STRUCTURE TO 3D MATRIX WITH ROI DEFINED ------
      %
103
104    % Make an array to hold all of the frames. The number of frames is 1
105    % less than the reported number, as when videoReader is executed it
106    % seems to add an extra frame at the start that is the same as the 2nd
107    % frame but offset. Does not seem to be a problem with the actual
108    % video, so can only assume it is a quirk of the MATLAB function. The
109    % size of each frame is only to house the range minrow:maxrow and
110    % mincol:maxcol, so the difference plus 1 is needed for the size.
111    movie = zeros(vidHeight,vidWidth,numberOfFrames,'uint8');
112
113    % Read one frame at a time.
114    for k = 1 : numberOfFrames
115
116        % Read the current index frame into a temporary variable.
117        % Originally the whole video was read in to 1 file, but this held
118        % ~360MB in RAM, where using a temporary variable to store only 1
119        % frame stores only 1.22MB in memory.
120
121        temp = imread([path filesep files(k).name]);
122        temp_gray = rgb2gray(temp);  % Convert to grayscale
123        temp_gray = imrotate(temp_gray, -90);  % Rotate if necessary
124        movie(:, :, k) = temp_gray;  % Store entire frame
125    end
126
127    if quiet ~= 0
128        disp('Finished reading frames');
129    end
130
131 %%
132
133    % ------------------------ PLUNGER DETECTION ------------------------
      %
134
135    % Read in a .png file of the plunger isolated from everything else.
136    % Hopefully this will allow the centroid / max height of the plunger
      to
137    % be found in all frames so a relative distance a particle rises to
      can
138    % be found.
139    plunger = imread('plunger.png');
140    plunger = rgb2gray(plunger);
141
142    % Following code adapted from:
143    % http://uk.mathworks.com/help/images/ref/normxcorr2.html
144
145    % Find the normalised correlation between the image of the isolated
```

```
146     % plunger and the full image of the syringe. So far this has only been
147     % done for frame 50, either this can be used as the centre throughout
148     % or it could be calculated for each frame?
149 
150     middleFrame = round(numberOfFrames / 2);
151 
152     c = normxcorr2(plunger, movie(:,:,round(numberOfFrames / 2)));
153 
154     % Find the maximum normalised correlation and the location of it.
155     [ypeak,xpeak] = find(c==max(c(:)));
156 
157     fprintf('y-peak: %d\n', ypeak);
158     fprintf('x-peak: %d\n', xpeak);
159 
160     % The peak positions are offset by the size of the plunger image in
    the
161     % x and y directions. This gives the x and y position of the lower
    left
162     % corner (actually upper left when using imshow due to axes rotations)
    .
163     yoffset = ypeak-size(plunger,1);
164     xoffset = xpeak-size(plunger,2);
165 
166     fprintf('y-offset: %d\n', yoffset);
167     fprintf('x-offset: %d\n', xoffset);
168 
169     % To find the actual mid point of where the plunger is in the image,
170     % add half of each size to the offsets.
171     plunger_ycent = yoffset + size(plunger,1)/2;
172     plunger_xcent = xoffset + size(plunger,2)/2;
173 
174     % The max height of the plunger will actually be at height yoffset
175     % because of the axes notation, so this is the value all heights
    should
176     % be measured relative to.
177     plunger_yloc = plunger_ycent;
178     plunger_xloc = plunger_xcent;
179 
180     % Plot the plunger detection on the first frame
181     % fig_plunger = figure;
182     % imshow(movie(:,:,1));
183     % hold on;
184     % plot(plunger_xcent, plunger_ycent, 'g+', 'LineWidth', 2);
185     % hold off;
186 
187     % Save the figure with the green plus sign
188     if writeFrames ~= 0
189         imagePath = [folderPath, filesep, 'Plunger Detection'];  % Change
    folder name as needed
190         if ~exist(imagePath, 'dir')
191             mkdir(imagePath);
192         end
193         imageName = [folderName, '_Plunger Detection.png'];  % Adjust
    naming convention if needed
```

```
194         print(fig_plunger, fullfile(imagePath, imageName), '-dpng', '-r600
    ');
195     end
196 
197     if quiet ~= 0
198         disp('Finished plunger detection');
199     end
200 
201 %%
202 
203     % ---------------------- BACKGROUND SUBTRACTION ----------------------
    %
204 
205     % If the method of background subtraction changes, it should always
206     % work towards giving a single output, a heightxwidthxlength array of
207     % values assigned either 255 or 0, with 0 being the forground or the
208     % particle of interest, and 255 being anything that isn't the particle
    .
209     % This leaves it in the state needed for following morphological
210     % processing
211 
212     % Make a structure to store the subtracted image at each of the frames
    ,
213     % identical to the movie array.
214     movie_sub = zeros(size(movie,1),size(movie,2),size(movie,3),'double');
215 
216     % Perform the background subtraction. This simple method uses the last
217     % frame as the background, and takes this from the current frame. The
218     % numbers being compared have to be converted to doubles as uint8
219     % doesn't allow negative numbers. Only go up to numberOfFrames-1 so
    the
220     % last frame is black, i.e. not a particle
221     for k=1:numberOfFrames-1
222        movie_sub(:,:,k) = double(movie(:,:,k)) - double(movie(:,:,end));
223     end
224 
225     % Two logical operations are then done. Originally it had been done so
226     % only black particles on light background were counted, but to make
    it
227     % more general if the absolute difference between current and last
228     % frame is less than the threshold then it is not a particle, and if
229     % the difference is greater than the threshold (and less than 255 so
    it
230     % does not count those added in the previous operation) then it is a
231     % particle. Only error these can cause if there is a pure white
232     % particle on a pure white background it wouldn't be counted as a
233     % particle, but this should never occur.
234     threshold = 15;
235     % movie_sub(abs(movie_sub)<=threshold) = 255;
236     % movie_sub(abs(movie_sub)>=threshold & movie_sub<255) = 0;
237     movie_sub(movie_sub>=0 | (movie_sub<0 & movie_sub>=(-threshold))) =
    255;
238     movie_sub(movie_sub<0 & movie_sub<(-threshold)) = 0;
239 
```

```
240     if quiet~= 0
241         disp('Finished background subtraction');
242     end
243 
244 %%
245 
246     % -------------------- MORPHOLOGICAL PROCESSING -------------------
    %
247 
248     % Create the structuring element. In this case, use a 3x3 square, i.e.
249     % every pixel must have 8 neighbours.
250     SE = strel('square',4);
251 
252     % The processes use binary images, with 0 as the foreground and black
253     % as the background. This section converts to binary, with white being
254     % the particle and black being the background to match how imclose and
255     % imopen work.
256     movie_sub = logical(movie_sub~=255);
257 
258     % First perform an opening operation. This erodes the image and then
259     % dilates it. This is best to do first as the erosion removes any
    noise
260     % that is not of the structuring element size. The dilation then adds
261     % the pixels back that have been taken from the foreground objects of
262     % interest
263     for k=1:numberOfFrames-1
264         movie_sub(:,:,k) = imopen(movie_sub(:,:,k),SE);
265     end
266 
267     % A closing operation is then performed after the opening. This does a
268     % dilation, followed by an erosion. Performing the two together
269     % restores most of the features of the original particles while
270     % reducing the influence of all other objects.
271     for k=1:numberOfFrames-1
272         movie_sub(:,:,k) = imclose(movie_sub(:,:,k),SE);
273     end
274 
275     % Convert the matrix back to a 0-255 matrix with 0 = particles and 255
276     % = background. REGIONPROPS WORKS WITH WHITE ON BLACK SO MAY NOT BE
277     % NEEDED
278     %movie_sub_test = 255-255*uint8(movie_sub);
279 
280     if quiet~= 0
281         disp('Finished morphological processing');
282     end
283 
284  %%
285 
286 
287     % ------------------ PARTICLE LOCATIONS PER FRAME -------------------
    %
288 
289     % First thing to do is to detect all of the particles in each frame.
290     % Relevant particles uses the following assumptions: the particle will
```

```
291     % not rise oover vidheight/2, and they particle will not have an area
292     % of more than 700 pixels. The particles found are stored in an array
293     % with col 1 = x-direction, col 2 = y-direction, and col 3 = current
294     % frame. This is the form required by the IDL tracking algorithm. A
295     % preliminary loop is performed to find the number of particles so an
296     % appropriately sized array can be made, and a second loop performed
    to
297     % actually store the positions.
298
299     % Initialise an array to say how many relevant particles have been
300     % found so far
301     current = 0;
302
303     % Define the maximum area that will be passed
304     maxarea = 500;
305     minarea = 50;
306
307     for k=1:numberOfFrames-1
308         % Find the number of blobs in the image
309         s = regionprops(movie_sub(:,:,k));
310
311         % Collect all of the centroid positions into 1 array
312         c = cat(1,s.Centroid);
313         a = cat(1,s.Area);
314
315         % Compare each particle's position and area and see if it is a
316         % valid particle, and if so increment the current particles
    counter
317         % to find how many particles there are
318         for j=1:length(a)
319             %if (c(j,2) > vidHeight/2) && (a(j)<maxarea) && (a(j)>minarea)
320             if (c(j,2) <= plunger_yloc) && (a(j)<maxarea) && (a(j)>minarea
    ) && (c(j,2) >= 500)
321             %if (a(j) < maxarea) && (a(j) > minarea) && (c(j,2) >= 345) %
    Track particles in the entire domain
322                 current = current+1;
323             end
324         end
325     end
326
327     % Make an array with the correct size to store all of the particles
328     % identified in the previous loop
329     positions = zeros(current,3);
330     areas = zeros(current,1);
331
332     % Reset current
333     current = 0;
334
335     for k=1:numberOfFrames-1
336         s = regionprops(movie_sub(:,:,k));
337
338         c = cat(1,s.Centroid);
339         a = cat(1,s.Area);
340
```

```
341         % Perform the same comparisons as before but this time store the
342         % particles found in the positions array.
343         for j=1:length(a)
344             %if (c(j,2) > vidHeight/2) && (a(j)<maxarea) && (a(j)>minarea)
345             if (c(j,2) <= plunger_yloc) && (a(j)<maxarea) && (a(j)>minarea
    ) && (c(j,2) >= 500)
346             %if (a(j)<maxarea) && (a(j)>minarea) && (c(j,2) >= 345) %
    Track particles in the entire domain
347                 current = current+1;
348 
349                 positions(current,1) = c(j,1);
350                 positions(current,2) = c(j,2);
351                 positions(current,3) = k;
352 
353                 areas(current) = a(j);
354             end
355         end
356     end
357 
358     if quiet~= 0
359         disp('Finished detecting particles');
360     end
361 
362 %%
363 
364     % ------------------ PARTICLE LINKING / TRAJECTORY ------------------
    %
365 
366     % The particle linking algorithm used is called the IDL tracking
367     % algorithm, created by David Grier, John Crocker, and Eric Weeks,
    with
368     % the MATLAB implementation done by Daniel Blair and Eric Dufrense.
    The
369     % positions found in the previous loop in the form x,y,time are passed
370     % to the track algorithm, which returns a an array with trajectories
371     % sorted, in the form x, y, time, trajectory no. First a structure has
372     % to be made to store some variables.
373 
374     % Number of frames a particle can disappear for before reappearing.
375     % This is important and should be based on multiple observations of
    the
376     % largest observed disappearence.
377     mem_val = 3;
378 
379     % Number of dimensions being considered.
380     dim_val = 2;
381 
382     % Eliminate any trajectories with paths containing less than this
383     % number of points. Good for eliminating noise and portions where the
384     % particle splits due to high exposure time.
385     good_val = 0;
386 
387     % Set the quiet function to 1 if you don't want any verbose output
388     quiet_val = 1;
```

```
389
390     % Create a structure to store these options.
391     trackStruct = struct('mem', mem_val, 'dim', dim_val, 'good', good_val,
        'quiet', quiet_val);
392
393     % Set the maximum number of pixels a particle can move from
394     % frame-to-frame to be considered the same trajectory.
395     maxSeperation = 400;
396
397     if isempty(positions) || (size(positions,1)==1)
398         disp('No trajectories or only 1 point to track');
399         results = struct('maxHeight',NaN,'minHeight',NaN,'maxWidth',NaN,'
      minWidth',NaN,'totalDistance',NaN,'transitTime',NaN);
400         return;
401     end
402
403     % Send the positions and above values to the tracking algorithm, which
404     % returns the sorted trajectories.
405     trajectories = track(positions,maxSeperation,trackStruct);
406
407     if trajectories == 0
408         disp('Trajectory linking failed');
409         results = struct('maxHeight',NaN,'minHeight',NaN,'maxWidth',NaN,'
      minWidth',NaN,'totalDistance',NaN,'transitTime',NaN);
410         return;
411     end
412
413     if quiet ~= 0
414         disp('Finished linking particles');
415     end
416
417 %%
418
419     % ---------------------- TRAJECTORY SELECTION ----------------------
      %
420
421     % Find which trajectory is the one we want to plot and follow. This
422     % section needs a lot of work, how to determine which is the
      trajectory
423     % we want? Remove all that are less than 3 points long, see if it acts
424     % like a bubble (y values are always decreasing), any others?
425
426     % Find how many trajectories there are in total
427     numberOfTrajectories = max(trajectories(:,4));
428
429     % Make an array to store size of each trajectory
430     trajectorySizes = zeros(numberOfTrajectories,1);
431
432     % Store the number of points in each trajectory
433     for i=1:numberOfTrajectories
434        trajectorySizes(i) = find(trajectories(:,4)==i,1,'last') - find(
      trajectories(:,4)==i,1,'first') + 1;
435     end
436
```

```
437     % Remove any trajectory from the list of trajectories that has found
    to
438     % be one we are not interested in. To do this, go through all of the
439     % trajectory sizes, if one is NaN then find the first and last index
    to
440     % feature this trajectory and store trajectory as the same matrix
441     % without that specific trajectory
442     for i=1:length(trajectorySizes)
443         if isnan(trajectorySizes(i))
444             startPos = find(trajectories(:,4)==i,1,'first');
445             endPos = find(trajectories(:,4)==i,1,'last');
446             trajectories = trajectories([1:startPos-1,endPos+1:end],:);
447         end
448     end
449
450     % In this method then the remaining trajectories are used to form a
451     % single trajectory. To do this, first the size of the overall
452     % trajectory is needed.
453     firstTime = min(trajectories(:,3));
454     lastTime = max(trajectories(:,3));
455     trajectory = zeros(lastTime-firstTime+1,4);
456     times = firstTime:1:lastTime;
457
458     % Each time step is then looked at to match a row in 'trajectories' to
459     % give the best overall trajectory. To do this 3 passes are utilised.
460     % The first pass only adds rows where there is a unique incident of
461     % that time in 'trajectories', i.e. at time 2 if there is only 1
462     % trajectory with a t=2 then add it. However, if the particle splits
463     % due to exposure time etc, then there may be 3 trajectories all with
464     % different positions for t=2. This is pass 2, where it adds the
465     % positions that are non-unique. At this point there are different
466     % methods, the easiest being to take an average of them. In this
    method
467     % it looks at the indices around the current one to see whether there
468     % is a neighbour row that is populated, and if so compare all of the
469     % candidate positions to see which is closest to the neighbours, as
470     % this will give the most likely position. If there are no neighbours
471     % then a simple average can be taken. The third pass is to use linear
472     % interpolation to fill in any gaps that remain.
473
474     % PASS 1 - Find and add the unique positions. Also store a variable
475     % called second to find which index the second unique position is at.
476     second = 0;
477     count = 0;
478
479     for i=1:size(trajectory,1)
480         tempPos = find(trajectories(:,3)==times(i));
481         if length(tempPos)==1 % i.e. unique position values
482             count = count+1;
483             if count == 2
484                 second = i;
485             end
486             trajectory(i,:) = trajectories(tempPos,:);
487         end
```

```
488     end
489
490     % Check if first point is much greater than any other, removes
491     % possibility of slight motion of dust etc being counted as a particle
    .
492     % Checks if first one is higher 100 pixels above 2nd point.
493     if trajectory(1,2) < (trajectory(2,2) - 100)
494
495     end
496
497     % PASS 2 - Find and add the non-unique positions. This needs a lot of
498     % nested for and if loops because there are several combinations: if
    it
499     % is the first index (i==1) then it can only look forward for a
500     % neighbour, so there is either a neighbour or not, then if i>1 then
501     % there are 4 possibilities: 2 neighbours, 1 neighbour at i+1, 1
502     % neighbour at i-1 and no neighbours, each requiring a different
503     % solution
504     for i=1:size(trajectory,1)
505         tempPos = find(trajectories(:,3)==times(i));
506         if length(tempPos)>1 % i.e. more than 1 position at current time
507             if i==1 % i.e. cant compare 2 nearest neighbours
508                 if trajectory(i+1,3) ~= 0 % i.e. has a neighbour
509                     % Initialise a distance matrix
510                     dist=zeros(length(tempPos),1);
511
512                     % Find the distances between each of the candidates
    and
513                     % the neighbouring position
514                     for j=1:length(tempPos)
515                         dist(j) = sqrt((trajectory(i+1,1)-trajectories(
    tempPos(j),1))^2 + (trajectory(i+1,2)-trajectories(tempPos(j),2))^2);
516                     end
517
518                     % Find the index of the best candidate and add it to
519                     % trajectory
520                     [val,ind] = min(dist);
521                     trajectory(i,:) = trajectories(tempPos(ind),:);
522                 else % i.e. has no neighbour so an average must be used
523                     % Initialise x and y positions, and take an average of
524                     % the different candidates.
525                     x=0;
526                     y=0;
527                     for j=1:length(tempPos)
528                         x=x+trajectories(tempPos(j),1);
529                         y=y+trajectories(tempPos(j),2);
530                     end
531                     x=x/length(tempPos);
532                     y=y/length(tempPos);
533
534                     % Add these averaged x and y positions to the
535                     % trajectory
536                     trajectory(i,1) = x;
537                     trajectory(i,2) = y;
```

```
538                     trajectory(i,3) = trajectories(tempPos(1),3);
539                 end
540             else
541                 if i==size(trajectory,1) % i.e. cant have a neighbour at i
    +1
542                     if trajectory(i-1,3) ~= 0 % has 1 neighbour at i-1
543                         dist=zeros(length(tempPos),1);
544                         for j=1:length(tempPos)
545                             dist(j) = sqrt((trajectory(i-1,1)-trajectories
    (tempPos(j),1))^2 + (trajectory(i-1,2)-trajectories(tempPos(j),2))^2);
546                         end
547                         [val,ind] = min(dist);
548                         trajectory(i,:) = trajectories(tempPos(ind),:);
549                     else % has no neighbours so take an average
550                         x=0;
551                         y=0;
552                         for j=1:length(tempPos)
553                             x=x+trajectories(tempPos(j),1);
554                             y=y+trajectories(tempPos(j),2);
555                         end
556                         x=x/length(tempPos);
557                         y=y/length(tempPos);
558                         trajectory(i,1) = x;
559                         trajectory(i,2) = y;
560                         trajectory(i,3) = trajectories(tempPos(1),3);
561                     end
562                 else
563                     if trajectory(i+1,3) ~= 0 && trajectory(i-1,3) ~= 0 %
    has 2 neighbours so find average distance between
564                         dist=zeros(length(tempPos),1);
565 
566                         % In this case, find the distance between the
567                         % candidate and both neighbours and take an
568                         % average.
569                         for j=1:length(tempPos)
570                             dist(j) = sqrt((trajectory(i+1,1)-trajectories
    (tempPos(j),1))^2 + (trajectory(i+1,2)-trajectories(tempPos(j),2))^2);
571                             dist(j) = dist(j) + sqrt((trajectory(i-1,1)-
    trajectories(tempPos(j),1))^2 + (trajectory(i-1,2)-trajectories(tempPos
    (j),2))^2);
572                             dist(j) = dist(j)/2;
573                         end
574                         [val,ind] = min(dist);
575                         trajectory(i,:) = trajectories(tempPos(ind),:);
576                     elseif trajectory(i+1,3) ~= 0 % has 1 neighbour at i+1
577                         dist=zeros(length(tempPos),1);
578                         for j=1:length(tempPos)
579                             dist(j) = sqrt((trajectory(i+1,1)-trajectories
    (tempPos(j),1))^2 + (trajectory(i+1,2)-trajectories(tempPos(j),2))^2);
580                         end
581                         [val,ind] = min(dist);
582                         trajectory(i,:) = trajectories(tempPos(ind),:);
583                     elseif trajectory(i-1,3) ~= 0 % has 1 neighbour at i-1
584                         dist=zeros(length(tempPos),1);
```

```
585                         for j=1:length(tempPos)
586                             dist(j) = sqrt((trajectory(i-1,1)-trajectories
    (tempPos(j),1))^2 + (trajectory(i-1,2)-trajectories(tempPos(j),2))^2);
587                         end
588                         [val,ind] = min(dist);
589                         trajectory(i,:) = trajectories(tempPos(ind),:);
590                     else % has no neighbours so take an average
591                         x=0;
592                         y=0;
593                         for j=1:length(tempPos)
594                             x=x+trajectories(tempPos(j),1);
595                             y=y+trajectories(tempPos(j),2);
596                         end
597                         x=x/length(tempPos);
598                         y=y/length(tempPos);
599                         trajectory(i,1) = x;
600                         trajectory(i,2) = y;
601                         trajectory(i,3) = trajectories(tempPos(1),3);
602                     end
603                 end
604             end
605         end
606     end
607 
608     % Remove rows where all values are zero
609     trajectory(all(~trajectory,2),:) = [];
610 
611     emptyRow = [0 0 0 0];
612 
613     % Find the rows where the time skips, and also find the time skipped.
614     % This will make it easier later to allow multiple skipped frames to
    be
615     % extrapolated, where before it has only been possible to extrapolate
616     % across 1 frame skipped.
617     current = 1;
618     pos=0;
619     for i=1:size(trajectory,1)-1
620        if (trajectory(i+1,3) - trajectory(i,3)) ~= 1
621           pos(current) = i;
622           posNum(current) = trajectory(i+1,3) - trajectory(i,3) - 1;
623 
624           current = current+1;
625        end
626     end
627 
628     % Add empty rows in between the rows where there is a time gap. This
    is
629     % done through all of pos, and at each pos add the number of empty
    rows
630     % required, and then add the number of rows inserted to the next pos
631     % value (if there is one) so that the indexing is updates as the
    matrix
632     % grows. Only run these loops when there are empty rows that need to
    be
```

```
633     % added, if no rows need to be added pos = 0, so would cause error at
634     % length(pos).
635     if pos~=0
636         for i=1:length(pos)
637             for j=1:posNum(i)
638                trajectory = [trajectory(1:pos(i),:);emptyRow;trajectory(
    pos(i)+1:end,:)];
639             end
640 
641             if i<length(pos)
642                 for j=(i+1):length(pos)
643                     pos(j)=pos(j)+posNum(i);
644                 end
645             end
646         end
647 
648         % Use linear interpolation to fill in the missing gaps. This needs
649         % to be done so the number of empty rows to extrapolate across is
650         % used in whether the extrapolation needs to be 1/2 (for a 1 row
651         % gap, 1/2 and 2/3 (for a 2 row gap), etc. Tested with multiple
652         % gaps of different sizes and works fine.
653         for i=1:length(pos)
654            for j=1:posNum(i)
655                % Find the time that the current row takes place at, and
    the
656                % trajectory number if belongs to
657                trajectory(pos(i)+j,3) = trajectory(pos(i)+j-1,3)+1;
658                trajectory(pos(i)+j,4) = trajectory(pos(i)+j-1,4);
659 
660                % Use linear interpolation to find the x and y position of
661                % the particle at current time. Hopefully indexing works
662                % correctly, difficult to think about how to correctly
    refer
663                % to proper index when gap is say 2-3 rows large.
664                trajectory(pos(i)+j,1) = trajectory(pos(i),1) + (trajectory
    (pos(i)+posNum(i)+1,1)-trajectory(pos(i),1))*(trajectory(pos(i)+j,3)-
    trajectory(pos(i),3))/(trajectory(pos(i)+posNum(i)+1,3)-trajectory(pos(
    i),3));
665                trajectory(pos(i)+j,2) = trajectory(pos(i),2) + (trajectory
    (pos(i)+posNum(i)+1,2)-trajectory(pos(i),2))*(trajectory(pos(i)+j,3)-
    trajectory(pos(i),3))/(trajectory(pos(i)+posNum(i)+1,3)-trajectory(pos(
    i),3));
666            end
667         end
668     end
669 
670     % Check if first point is much greater than the second unique position
    ,
671     % removes possibility of slight motion of dust etc being counted as a
672     % particle. Checks if first one is higher 100 pixels above 2nd point.
673     % if trajectory(1,2) < (trajectory(second,2) - 100)
674     %     trajectory = trajectory(second:end,:);
675     % end
676 
```

```
677     if quiet ~= 0
678         disp('Finished trajectory selection and gap filling');
679     end
680 
681 %%
682 
683     % ---------------- FIND HEIGHTS RELATIVE TO PLUNGER -----------------
        %
684 
685     % In here all of the important information about the particle is
686     % extracted, e.g. height reached, time taken, total distance travelled
        ,
687     % etc. This will first be done in terms of pixels, and later we need
        to
688     % calibrate it to see whether each pixels size varies depending on
689     % whether it is in the centre, sides, top or bottom etc. Measure all y
690     % distances from plunger_yloc and all x distances from plunger_xloc.
691 
692     % Make a new array to store the trajectory with relative coordinates
693     trajectory_relative = zeros(size(trajectory,1),size(trajectory,2));
694     trajectory_relative(:,3) = trajectory(:,3);
695     trajectory_relative(:,4) = trajectory(:,4);
696 
697     % Make all of the trajectory points relative to the plunger
698     trajectory_relative(:,1) = trajectory(:,1) - plunger_xloc;
699     trajectory_relative(:,2) = plunger_yloc - trajectory(:,2);
700 
701 %%
702 
703     % --------------------- TRAJECTORY STATISTICS ----------------------
        %
704 
705     % EVENTUALLY CHANGE SO THE TIMES AND COORDINATES ARE IN SECONDS AND
706     % MICRONS, NOT PIXELS AND FRAMES
707 
708     % Find the maximum and minimum height and width reached in the
709     % trajectory. These points also form the bounding area of the
710     % trajectory, i.e. the area covered by
711     % (minWidthFrame:maxWidthFrame,minHeightFrame:maxHeightFrame)
712     [maxHeight,maxHeightIndex] = max(trajectory_relative(:,2));
713     maxHeightFrame = trajectory_relative(maxHeightIndex,3);
714     [minHeight,minHeightIndex] = min(trajectory_relative(:,2));
715     minHeightFrame = trajectory_relative(minHeightIndex,3);
716 
717     [maxWidth,maxWidthIndex] = max(trajectory_relative(:,1));
718     maxWidthFrame = trajectory_relative(maxWidthIndex,3);
719     [minWidth,minWidthIndex] = min(trajectory_relative(:,1));
720     minWidthFrame = trajectory_relative(minWidthIndex,3);
721 
722     % Find the range in height and width of the particle
723     rangeHeight = maxHeight-minHeight;
724     rangeWidth = maxWidth-minWidth;
725 
726     % Find the heights where the particle is in the middle 90% of the
```

```
727     % range, i.e. within 5% of either extremity. Useful to see whether it
728     % lingers at max and min heights for a while
729     rangeHeightTop = maxHeight-0.05*rangeHeight;
730     rangeHeightBottom = minHeight+0.05*rangeHeight;
731
732     % Find fraction of frames spent in middle 90%, top 5% and bottom 5%
733     timeMiddle = length(find(trajectory_relative(:,2) > rangeHeightBottom
    & trajectory_relative(:,2) < rangeHeightTop))/size(trajectory,1);
734     timeTop = length(find(trajectory_relative(:,2) > rangeHeightTop))/size
    (trajectory,1);
735     timeBottom = length(find(trajectory_relative(:,2) < rangeHeightBottom)
    )/size(trajectory,1);
736
737     % Find the total distance travelled in pixels using d = d +
738     % sqrt(x_diff^2 + y_diff^2)
739     distance = zeros(size(trajectory,1)-1,1);
740     for i=1:size(trajectory,1)-1
741         distance(i) = sqrt((trajectory(i,1)-trajectory(i+1,1))^2 + (
    trajectory(i,2)-trajectory(i+1,2))^2);
742     end
743     totalDistance = sum(distance);
744
745     % Find 'transit time'. We'll define this as the time from the start to
746     % the time the particle falls within 1% of the plunger over the y
    range
747     % (y_max - 0) for the last time (work backwards, as sometimes the
748     % particle swoops down below the plunger and comes back up. First
749     % define the 1% cut off.
750     y_limit = 20;
751
752     % Work backwards through the matrix until the centre of the particle
753     % passes the 1% cutoff and find the index this occurs at. This method
754     % shouldn't rely on the particle moving below the plunger as it will
755     % find the last position where it is above the plunger, which could be
756     % the last position if it settles on the top of the plunger. Only
    weird
757     % behaviour seen is when particle rests on plunger and falls, the
758     % particle tumbles slightly so moves above y_limit for 1 frame but
759     % previous 4 frames are all below it, fixable / any point fixing?
760     k=size(trajectory_relative,1);
761     if ~isempty(find(trajectory_relative(:,2)>y_limit,1))
762         while k>0 && trajectory_relative(k,2) < y_limit
763             k=k-1;
764         end
765
766         % From this define the transit time as the difference in frame
767         % where the particle rises 1% above the plunger and the start time
768         % of the trajectory.
769         transitTime = trajectory_relative(k,3);
770     else
771         transitTime = NaN;
772     end
773
774     % Other things that could be implemented: velocities throughout (would
```

```
775     % be crude due to crude distance approximations when particle is lost)
        ,
776     % free-fall velocity (speed it falls at right at the end where there
        is
777     % no swirling force, would be hard to find when this happens and may
778     % not happen for heavy particles and small chambers).
779
780 %%
781
782     % -------------------- PLOT IMPORTANT STATISTICS --------------------
        %
783
784     % Display the x and y positions throughout the video as 2D plots
785     fig3=figure('Visible','off');
786     set(gcf,'position',[711,434,530,550])
787     ax=plotyy(trajectory_relative(:,3), trajectory_relative(:,1),
        trajectory_relative(:,3),trajectory_relative(:,2));
788     xlabel('Frame','FontWeight','bold');
789     ylabel(ax(1),'x-position (pixels)','FontWeight','bold');
790     ylabel(ax(2),'y-position (pixels)','FontWeight','bold');
791     set(ax(2),'ylim',[0 300])
792     grid on;
793
794     if writeImage ~= 0
795         pause(1)
796         imagePath = [folderPath,filesep,'Graphs'];
797         if ~exist(imagePath,'dir')
798            mkdir(imagePath);
799         end
800         imageName = [folderName,'Graph.png'];
801         print(fig3,[imagePath,filesep,imageName],'-dpng','-r600');
802     end
803
804     % Save x-y positions as a text file
805     if writeTxt ~= 0
806         dataPath = [folderPath, filesep, 'Graphs'];
807         if ~exist(dataPath, 'dir')
808             mkdir(dataPath);
809         end
810         dataName = [folderName, '_.txt'];
811         dataFile = fopen(fullfile(dataPath, dataName), 'w');
812         fprintf(dataFile, 'Frame\tx-position (pixels)\ty-position (pixels)
        \n');
813         fprintf(dataFile, '%d\t%d\t%d\n', [trajectory_relative(:,3),
        trajectory_relative(:,1), trajectory_relative(:,2)]');
814         fclose(dataFile);
815     end
816 %%
817
818     % ---------------------- PLAY & SAVE MOVIE -------------------------
        %
819
820     % Bring up the image player interface
821     if showVid ~= 0
```

```
822         fig4 = figure; % Create a figure handle for the video display
823         current = 1;
824         for k = 1:trajectory(end,3)
825             % Show the syringe at frame k in the existing figure (fig4)
826             imshow(movie(:,:,k));
827             hold on;
828 
829             % If the current frame time corresponds to the frame time of
    the
830             % trajectory, plot all of the trajectory lines up to the
    current
831             % frame.
832             if k == trajectory(current,3)
833                 plot(trajectory(1:current,1),trajectory(1:current,2),'r-',
    'LineWidth',1);
834                 current = current + 1;
835             end
836 
837             % Pause for 1/frameRate to display the video in close to the
    intended time
838             pause(1/75);
839 
840             % Save the current frame as an image if writeFrames is enabled
841             if writeFrames ~= 0
842                 imagePath = [folderPath,filesep,'Trajectory'];
843                 if ~exist(imagePath,'dir')
844                     mkdir(imagePath);
845                 end
846                 imageName = [folderName,'Trajectory',num2str(k),'.png'];
847                 print(fig4,[imagePath,filesep,imageName],'-dpng','-r600');
848             end
849         end
850     end
851 
852 
853     % Make a matrix that stores the original video as well as the
854     % trajectory plotted on as a black line. A variable called current is
855     % used to determine whether the current index coincides with a
856     % trajectory index. A new matrix called movie_line is used to store
    the
857     % movie with the plotted line.
858 
859 
860     % Write the trajectory array to a video file
861     if writeAVI ~= 0
862 
863         current=2;
864         movie_line = movie(:,:,1:trajectory(end,3));
865         for k=1:trajectory(end,3)
866             % Make a temporaty matrix to store the frame until all of the
    lines
867             % have been plotted
868             temp_mat = movie_line(:,:,k);
869 
```

```
870             % Check if the current index is one in the trajectory
871             if k==trajectory(current,3)
872 
873                 % If it is, the loop through all of the trajectory points
    until
874                 % the current one and plot them on the matrix, the x and y
875                 % variables store the line coordinates, index rounds them
    to
876                 % the current matrix size and then logics are used to say
    at
877                 % all indices of the line change the pixel to a 0 to
    represent
878                 % the line.
879                 for i=2:current
880                     x=linspace(trajectory(i-1,1),trajectory(i,1),1000);
881                     y=linspace(trajectory(i-1,2),trajectory(i,2),1000);
882                     index = sub2ind(size(temp_mat),round(y),round(x));
883                     temp_mat(index) = 0;
884                 end
885 
886                 % Store this temporary frame as the finished frame
887                 movie_line(:,:,k) = temp_mat;
888 
889                 current = current+1;
890             end
891         end
892 
893         videoPath = [folderPath,filesep,'Videos'];
894         if ~exist(videoPath,'dir')
895            mkdir(videoPath);
896         end
897         v = VideoWriter([videoPath,filesep,folderName]);
898         v.FrameRate=30;
899         open(v)
900         for l=1:size(movie_line,3)
901             writeVideo(v,movie_line(:,:,l))
902         end
903         close(v)
904     end
905 
906     % Output the results to the testing program. The results are created
    as
907     % a structure with the variable name and it's value.
908     results = struct('maxHeight',maxHeight,'minHeight',minHeight,'maxWidth
    ',maxWidth,'minWidth',minWidth,'totalDistance',totalDistance,'
    transitTime',transitTime);
909 end
```

Listing 6: Code to control output track data

9.4 XY Plot Editor Tool: “XYPlotEditorGUI_.m

```
1
2  function XYPlotEditorGUI()
3      fig = figure('Name','X-Y Plot Editor','NumberTitle','off', ...
4          'Units','normalized','OuterPosition',[0.05 0.05 0.9 0.9], ...
5          'Color',[0.94 0.94 0.94],'Resize','on');
6
7      data = struct('allFiles',[],'currentFile',[],'editedData',[],'
       originalData',[],...
8                    'folder','', 'subfolders', [], 'currentFolder', '');
9      data.saveFolder = '';
10
11     % Create main panels for better organization
12     leftPanel = uipanel(fig,'Title','Data Control','FontSize',10,'
       FontWeight','bold',...
13         'Units','normalized','Position',[0.01 0.01 0.32 0.98],...
14         'BackgroundColor',[0.97 0.97 0.97]);
15
16     rightPanel = uipanel(fig,'Title','Visualization','FontSize',10,'
       FontWeight','bold',...
17         'Units','normalized','Position',[0.34 0.01 0.64 0.98],...
18         'BackgroundColor',[0.97 0.97 0.97]);
19
20     %% LEFT PANEL CONTROLS
21     % File Selection Section
22     filePanel = uipanel(leftPanel,'Title','File Selection','FontSize'
       ,9,...
23         'Units','normalized','Position',[0.02 0.75 0.96 0.23],...
24         'BackgroundColor',[0.95 0.95 0.95]);
25
26     uicontrol(filePanel,'Style','pushbutton','String','Select Main Folder'
       ,...
27         'Units','normalized','Position',[0.02 0.75 0.45 0.2],...
28         'FontSize',9,'Callback',@selectMainFolder,...
29         'BackgroundColor',[0.3 0.6 0.9],'ForegroundColor','white');
30
31     folderDropdown = uicontrol(filePanel,'Style','popupmenu',...
32         'String',{'<Select Subfolder>'},'Units','normalized',...
33         'Position',[0.52 0.75 0.46 0.2],'FontSize',9,...
34         'Callback',@subfolderChanged);
35
36     uicontrol(filePanel,'Style','text','String','Available Files:',...
37         'Units','normalized','Position',[0.02 0.52 0.3 0.15],...
38         'FontSize',9,'HorizontalAlignment','left','BackgroundColor',[0.95
       0.95 0.95]);
39
40     fileList = uicontrol(filePanel,'Style','listbox','String',{},...
41         'Units','normalized','Position',[0.02 0.02 0.96 0.48],...
42         'FontSize',8,'Callback',@fileSelected);
43
44     % Data Editing Section
45     editPanel = uipanel(leftPanel,'Title','Data Editing','FontSize',9,...
46         'Units','normalized','Position',[0.02 0.52 0.96 0.22],...
```

```
47          'BackgroundColor',[0.95 0.95 0.95]);
48
49      uicontrol(editPanel,'Style','text','String','Rows to Delete (e.g.,
    1,3,5 or 10:15):',...
50          'Units','normalized','Position',[0.02 0.75 0.96 0.15],...
51          'FontSize',8,'HorizontalAlignment','left','BackgroundColor',[0.95
    0.95 0.95]);
52
53      rowInput = uicontrol(editPanel,'Style','edit','Units','normalized',...
54          'Position',[0.02 0.55 0.96 0.18],'FontSize',9,...
55          'BackgroundColor','white');
56
57      uicontrol(editPanel,'Style','pushbutton','String','Delete Rows by
    Index',...
58          'Units','normalized','Position',[0.02 0.32 0.96 0.18],...
59          'FontSize',9,'Callback',@deleteRows,...
60          'BackgroundColor',[0.9 0.4 0.4],'ForegroundColor','white');
61
62      uicontrol(editPanel,'Style','pushbutton','String','Delete Selected
    Points',...
63          'Units','normalized','Position',[0.02 0.08 0.96 0.18],...
64          'FontSize',9,'Callback',@deleteSelectedPlotRows,...
65          'BackgroundColor',[0.8 0.3 0.3],'ForegroundColor','white');
66
67      % Action Buttons Section
68      actionPanel = uipanel(leftPanel,'Title','Actions','FontSize',9,...
69          'Units','normalized','Position',[0.02 0.42 0.96 0.09],...
70          'BackgroundColor',[0.95 0.95 0.95]);
71
72      % Set Save Folder button (left side)
73      uicontrol(actionPanel,'Style','pushbutton','String','Set Save Folder'
    ,...
74          'Units','normalized','Position',[0.02 0.15 0.47 0.7],...
75          'FontSize',10,'FontWeight','bold','Callback',@setSaveFolder,...
76          'BackgroundColor',[0.6 0.6 0.9],'ForegroundColor','white');
77
78      % Save File & Plots button (right side)
79      uicontrol(actionPanel,'Style','pushbutton','String','Save File & Plots
    ',...
80          'Units','normalized','Position',[0.51 0.15 0.47 0.7],...
81          'FontSize',10,'FontWeight','bold','Callback',@saveData,...
82          'BackgroundColor',[0.2 0.7 0.3],'ForegroundColor','white');
83
84      % Data Table Section
85      tablePanel = uipanel(leftPanel,'Title','Data Table','FontSize',9,...
86          'Units','normalized','Position',[0.02 0.02 0.96 0.39],...
87          'BackgroundColor',[0.95 0.95 0.95]);
88
89      dataTable = uitable(tablePanel,'Units','normalized',...
90          'Position',[0.02 0.02 0.96 0.96],'FontSize',8,...
91          'CellSelectionCallback', @rowSelected,...
92          'BackgroundColor',[1 1 1; 0.95 0.95 0.95]);
93
94      %% RIGHT PANEL PLOTS
```

```
    % Create subplot panels for better organization
    plot1Panel = uipanel(rightPanel,'Title','Radius vs Frame','FontSize'
    ,9,...
        'Units','normalized','Position',[0.02 0.68 0.47 0.3],...
        'BackgroundColor',[0.98 0.98 0.98]);

    plot2Panel = uipanel(rightPanel,'Title','Height vs Frame','FontSize'
    ,9,...
        'Units','normalized','Position',[0.51 0.68 0.47 0.3],...
        'BackgroundColor',[0.98 0.98 0.98]);

    plot3Panel = uipanel(rightPanel,'Title','Track: Radius vs Height','
    FontSize',9,...
        'Units','normalized','Position',[0.02 0.02 0.96 0.64],...
        'BackgroundColor',[0.98 0.98 0.98]);

    % Create axes within panels
    ax1 = axes('Parent',plot1Panel,'Units','normalized',...
        'Position',[0.15 0.2 0.8 0.7],'FontSize',8);
    ax2 = axes('Parent',plot2Panel,'Units','normalized',...
        'Position',[0.15 0.2 0.8 0.7],'FontSize',8);
    ax3 = axes('Parent',plot3Panel,'Units','normalized',...
        'Position',[0.1 0.1 0.85 0.8],'FontSize',8);

    % Set up figure-level mouse callbacks for drag selection (initially
    empty)
    set(fig, 'WindowButtonMotionFcn', []);
    set(fig, 'WindowButtonUpFcn', []);

    % Add status bar
    statusText = uicontrol(fig,'Style','text',...
        'String','Ready - Select a main folder to begin',...
        'Units','normalized','Position',[0.01 0.001 0.98 0.025],...
        'FontSize',8,'HorizontalAlignment','left',...
        'BackgroundColor',[0.9 0.9 0.9],'ForegroundColor',[0.3 0.3 0.3]);

    % Initialize selected points storage and drag state
    data.selectedPoints = [];
    data.dragState = struct('isDragging', false, 'startPoint', [], '
    currentAxes', [], 'rectHandle', []);

    %% CALLBACK FUNCTIONS
    function setSaveFolder(~, ~)
        folder = uigetdir(data.folder, 'Select Folder to Save Files');
        if folder == 0, return; end
        data.saveFolder = folder;
        statusText.String = ['Save folder set to: ' folder];
    end

    function selectMainFolder(~,~)
        folder = uigetdir;
        if folder == 0, return; end
        data.folder = folder;
        dirs = dir(folder);
```

```
144         subNames = {dirs([dirs.isdir]).name};
145         subNames = subNames(~ismember(subNames,{'.','..'}));
146         data.subfolders = subNames;
147 
148         if isempty(subNames)
149             folderDropdown.String = {'<No subfolders found>'};
150             statusText.String = 'No subfolders found in selected directory
    ';
151             return;
152         end
153 
154         folderDropdown.String = subNames;
155         folderDropdown.Value = 1;
156         statusText.String = sprintf('Found %d subfolders', length(subNames
    ));
157         subfolderChanged();
158     end
159 
160     function subfolderChanged(~,~)
161         sel = folderDropdown.Value;
162         if isempty(data.subfolders), return; end
163         selectedSubfolder = fullfile(data.folder, data.subfolders{sel});
164         files = dir(fullfile(selectedSubfolder,'*.txt'));
165         data.currentFolder = selectedSubfolder;
166         data.allFiles = fullfile({files.folder}, {files.name});
167 
168         if isempty(data.allFiles)
169             fileList.String = {'<No .txt files found>'};
170             statusText.String = 'No .txt files found in selected subfolder
    ';
171             return;
172         end
173 
174         relNames = cellfun(@(f) strrep(f, [data.currentFolder filesep], ''
    ), ...
175                            data.allFiles, 'UniformOutput', false);
176         fileList.String = relNames;
177         fileList.Value = 1;
178         statusText.String = sprintf('Found %d .txt files', length(relNames
    ));
179         fileSelected();
180     end
181 
182     function fileSelected(~,~)
183         idx = fileList.Value;
184         if isempty(data.allFiles), return; end
185         try
186             T = readtable(data.allFiles{idx},'FileType','text','Delimiter'
    ,'  ','HeaderLines',1);
187             T.Properties.VariableNames = {'Frame','X','Y'};
188             data.originalData = T;
189             data.editedData = T;
190             data.currentFile = data.allFiles{idx};
191             updateTableAndPlots();
```

```
192             highlightSuspiciousRows();
193             [~, filename] = fileparts(data.currentFile);
194             statusText.String = sprintf('Loaded: %s (%d data points)',
    filename, height(T));
195         catch ME
196             errordlg(['Could not read file: ' ME.message],'File Error');
197             statusText.String = 'Error loading file';
198         end
199     end
200 
201     function updateTableAndPlots()
202         T = data.editedData;
203         if isempty(T), return; end
204 
205         X = T.X;
206         Y = T.Y;
207         Frame = T.Frame;
208 
209         % Update table
210         set(dataTable,'Data',T{:,:},'ColumnName',T.Properties.
    VariableNames);
211 
212         % Plot 1: Radius vs Frame
213         axes(ax1); cla(ax1);
214         h1 = plot(ax1, Frame, X, 'b.-', 'LineWidth', 1.5, 'MarkerSize', 6)
    ;
215         title(ax1,'Projected Radius vs Frame (Drag to select region)','
    FontSize',9,'FontWeight','bold');
216         xlabel(ax1,'Frame','FontSize',8); ylabel(ax1,'Radius','FontSize'
    ,8);
217         grid(ax1, 'on'); grid(ax1, 'minor');
218         set(ax1,'GridAlpha',0.3,'MinorGridAlpha',0.1);
219         setupDragSelection(ax1, 1);
220 
221         % Plot 2: Height vs Frame
222         axes(ax2); cla(ax2);
223         h2 = plot(ax2, Frame, Y, 'r.-', 'LineWidth', 1.5, 'MarkerSize', 6)
    ;
224         title(ax2,'Height vs Frame (Drag to select region)','FontSize',9,'
    FontWeight','bold');
225         xlabel(ax2,'Frame','FontSize',8); ylabel(ax2,'Height','FontSize'
    ,8);
226         grid(ax2, 'on'); grid(ax2, 'minor');
227         set(ax2,'GridAlpha',0.3,'MinorGridAlpha',0.1);
228         setupDragSelection(ax2, 2);
229 
230         % Plot 3: Track
231         axes(ax3); cla(ax3);
232         h3 = plot(ax3, X, Y, 'k.-', 'LineWidth', 1.2, 'MarkerSize', 8);
233         title(ax3,'Track: Radius vs Height (Drag to select region)','
    FontSize',10,'FontWeight','bold');
234         xlabel(ax3,'Radius','FontSize',9); ylabel(ax3,'Height','FontSize'
    ,9);
235         grid(ax3, 'on'); grid(ax3, 'minor');
```

```
236         set(ax3,'GridAlpha',0.3,'MinorGridAlpha',0.1);
237         setupDragSelection(ax3, 3);
238     end
239 
240     function startDrag(src, ~, plotNum)
241         try
242             % Only start drag on left mouse button
243             if ~strcmp(get(fig, 'SelectionType'), 'normal')
244                 return;
245             end
246 
247             % Initialize drag state
248             data.dragState.isDragging = true;
249             data.dragState.currentAxes = src;
250             data.dragState.plotNum = plotNum;
251 
252             % Get starting point
253             cp = get(src, 'CurrentPoint');
254             data.dragState.startPoint = [cp(1,1), cp(1,2)];
255 
256             % Check for Ctrl key (for additive selection)
257             modifiers = get(fig, 'CurrentModifier');
258             data.dragState.ctrlPressed = any(strcmp(modifiers, 'control'))
    ;
259 
260             % Clear previous selection if not adding
261             if ~data.dragState.ctrlPressed
262                 data.selectedPoints = [];
263             end
264 
265             % Create selection rectangle
266             axes(src);
267             hold on;
268             data.dragState.rectHandle = rectangle('Position', [0 0 0 0],
    ...
269                 'EdgeColor', 'm', 'LineWidth', 2, 'LineStyle', '--', ...
270                 'FaceColor', [1 0 1 0.1], 'Tag', 'selection_rect');
271 
272             % Enable motion and button up callbacks only when dragging
    starts
273             set(fig, 'WindowButtonMotionFcn', @dragMotion);
274             set(fig, 'WindowButtonUpFcn', @endDrag);
275 
276         catch ME
277             fprintf('Error in startDrag: %s\n', ME.message);
278             data.dragState.isDragging = false;
279         end
280     end
281 
282     function dragMotion(~, ~)
283         try
284             % Only process if we're actually dragging
285             if ~data.dragState.isDragging || isempty(data.dragState.
    rectHandle) || ...
```

```
286                 ~isvalid(data.dragState.rectHandle)
287                     return;
288                 end
289 
290                 % Check if current point is still over the correct axes
291                 currentAxes = gca;
292                 if currentAxes ~= data.dragState.currentAxes
293                     return;
294                 end
295 
296                 % Get current point
297                 cp = get(data.dragState.currentAxes, 'CurrentPoint');
298                 currentPoint = [cp(1,1), cp(1,2)];
299 
300                 % Update rectangle
301                 x1 = min(data.dragState.startPoint(1), currentPoint(1));
302                 y1 = min(data.dragState.startPoint(2), currentPoint(2));
303                 width = abs(currentPoint(1) - data.dragState.startPoint(1));
304                 height = abs(currentPoint(2) - data.dragState.startPoint(2));
305 
306                 % Only update if rectangle handle is still valid
307                 if isvalid(data.dragState.rectHandle)
308                     set(data.dragState.rectHandle, 'Position', [x1, y1, width,
    height]);
309                 end
310 
311             catch ME
312                 fprintf('Error in dragMotion: %s\n', ME.message);
313                 % Clean up on error
314                 data.dragState.isDragging = false;
315                 set(fig, 'WindowButtonMotionFcn', []);
316             end
317         end
318 
319         function endDrag(~, ~)
320             try
321                 % Disable motion callback immediately
322                 set(fig, 'WindowButtonMotionFcn', []);
323                 set(fig, 'WindowButtonUpFcn', []);
324 
325                 if ~data.dragState.isDragging
326                     return;
327                 end
328 
329                 T = data.editedData;
330                 if isempty(T)
331                     data.dragState.isDragging = false;
332                     return;
333                 end
334 
335                 % Get final rectangle bounds
336                 cp = get(data.dragState.currentAxes, 'CurrentPoint');
337                 endPoint = [cp(1,1), cp(1,2)];
338 
```

```
339             x1 = min(data.dragState.startPoint(1), endPoint(1));
340             x2 = max(data.dragState.startPoint(1), endPoint(1));
341             y1 = min(data.dragState.startPoint(2), endPoint(2));
342             y2 = max(data.dragState.startPoint(2), endPoint(2));
343 
344             % Only process if we have a meaningful rectangle (not just a
    click)
345             if abs(x2-x1) < 0.01 && abs(y2-y1) < 0.01
346                 cleanupDrag();
347                 return;
348             end
349 
350             % Find points within rectangle
351             selectedIndices = [];
352             switch data.dragState.plotNum
353                 case 1 % Radius vs Frame
354                     withinX = T.Frame >= x1 & T.Frame <= x2;
355                     withinY = T.X >= y1 & T.X <= y2;
356                 case 2 % Height vs Frame
357                     withinX = T.Frame >= x1 & T.Frame <= x2;
358                     withinY = T.Y >= y1 & T.Y <= y2;
359                 case 3 % Track
360                     withinX = T.X >= x1 & T.X <= x2;
361                     withinY = T.Y >= y1 & T.Y <= y2;
362             end
363 
364             selectedIndices = find(withinX & withinY);
365 
366             % Add to selection (or replace if not Ctrl)
367             if data.dragState.ctrlPressed
368                 data.selectedPoints = unique([data.selectedPoints;
    selectedIndices]);
369             else
370                 data.selectedPoints = selectedIndices;
371             end
372 
373             % Update highlights and table
374             updatePlotHighlights();
375             if ~isempty(data.selectedPoints)
376                 dataTable.UserData = [data.selectedPoints, ones(length(
    data.selectedPoints), 1)];
377                 statusText.String = sprintf('%d points selected by drag (
    Ctrl+drag to add)', ...
378                     length(data.selectedPoints));
379             else
380                 statusText.String = 'No points in selection area';
381             end
382 
383         catch ME
384             fprintf('Error in endDrag: %s\n', ME.message);
385         end
386 
387         % Always clean up
388         cleanupDrag();
```

```
389     end
390
391     function cleanupDrag()
392         try
393             % Clean up drag state
394             if isfield(data.dragState, 'rectHandle') && ~isempty(data.
    dragState.rectHandle) && ...
395                isvalid(data.dragState.rectHandle)
396                 delete(data.dragState.rectHandle);
397             end
398
399             data.dragState.isDragging = false;
400             data.dragState.rectHandle = [];
401
402             % Make sure callbacks are cleared
403             set(fig, 'WindowButtonMotionFcn', []);
404             set(fig, 'WindowButtonUpFcn', []);
405
406         catch ME
407             fprintf('Error in cleanupDrag: %s\n', ME.message);
408         end
409     end
410
411     function setupDragSelection(axHandle, plotNum)
412         % Set up mouse callbacks for drag selection - only ButtonDownFcn
    per axis
413         set(axHandle, 'ButtonDownFcn', @(src,evt) startDrag(src, evt,
    plotNum));
414     end
415
416     function rowSelected(~, event)
417         dataTable.UserData = event.Indices;
418         if isempty(event.Indices)
419             data.selectedPoints = []; % Clear plot selection when table is
     cleared
420             updatePlotHighlights();
421             return;
422         end
423
424         rowIndices = unique(event.Indices(:,1));
425
426         % Sync plot selection with table selection
427         data.selectedPoints = rowIndices;
428         updatePlotHighlights();
429
430         statusText.String = sprintf('%d rows selected from table', length(
    rowIndices));
431     end
432
433     function deleteRows(~,~)
434         txt = rowInput.String;
435         if isempty(txt)
436             msgbox('Please enter row indices to delete.','No Input');
437             return;
```

```
438         end
439
440         try
441             indices = eval(['[', txt, ']']);
442             if any(indices < 1) || any(indices > height(data.editedData))
443                 errordlg('Invalid row indices. Check range.','Invalid
    Input');
444                 return;
445             end
446
447             numDeleted = length(indices);
448             data.editedData(indices,:) = [];
449             rowInput.String = ''; % Clear input
450             updateTableAndPlots();
451             highlightSuspiciousRows();
452             statusText.String = sprintf('Deleted %d rows. %d points
    remaining', ...
453                 numDeleted, height(data.editedData));
454         catch ME
455             errordlg(['Invalid input format: ' ME.message], 'Input Error')
    ;
456         end
457     end
458
459     function saveData(~,~)
460         if isempty(data.editedData)
461             msgbox('No data to save.','Warning');
462             return;
463         end
464
465         % Require a save folder first
466         if isempty(data.saveFolder)
467             msgbox('Please select a save folder first.','Save Folder
    Required');
468             return;
469         end
470
471         try
472             % Use current file name with "_edited" suffix
473             [~, baseName, ~] = fileparts(data.currentFile);
474             outFile = fullfile(data.saveFolder, [baseName '.txt']);
475
476             % Save table
477             writetable(data.editedData, outFile, 'Delimiter','  ');
478
479             % Save plots
480             %saveas(ax1, fullfile(data.saveFolder, [baseName 'Radius.png
    ']));
481             %saveas(ax2, fullfile(data.saveFolder, [baseName 'Height.png
    ']));
482             %saveas(ax3, fullfile(data.saveFolder, [baseName 'Track.png']
    );
483
484             statusText.String = sprintf('Saved: %s and plots to folder', [
```

```
    baseName '.txt']);
485             msgbox('Files saved successfully.','Success');
486         catch ME
487             errordlg(['Save failed: ' ME.message],'Save Error');
488         end
489     end
490 
491     function highlightSuspiciousRows()
492         T = data.editedData;
493         N = height(T);
494         if N < 5, return; end
495 
496         spikeRows = false(N, 1);
497 
498         % Method 1: Detect sharp deviations from local trend using moving
    window
499         windowSize = min(9, floor(N/10)); % Adaptive window size
500         if windowSize < 3, windowSize = 3; end
501 
502         for i = windowSize+1:N-windowSize
503             % Get local neighborhood
504             startIdx = max(1, i-windowSize);
505             endIdx = min(N, i+windowSize);
506 
507             % Calculate local statistics for X and Y separately
508             localX = T.X(startIdx:endIdx);
509             localY = T.Y(startIdx:endIdx);
510 
511             % Calculate deviation from local median (more robust than mean
    )
512             medianX = median(localX);
513             medianY = median(localY);
514             madX = median(abs(localX - medianX)); % Median Absolute
    Deviation
515             madY = median(abs(localY - medianY));
516 
517             % Avoid division by zero
518             if madX == 0, madX = std(localX); end
519             if madY == 0, madY = std(localY); end
520             if madX == 0, madX = 1; end
521             if madY == 0, madY = 1; end
522 
523             % Calculate z-scores using MAD (more robust than std)
524             zScoreX = abs(T.X(i) - medianX) / (1.4826 * madX); % 1.4826
    converts MAD to std equivalent
525             zScoreY = abs(T.Y(i) - medianY) / (1.4826 * madY);
526 
527             % Flag as spike if both X and Y deviate significantly
528             if zScoreX > 3.5 && zScoreY > 3.5
529                 spikeRows(i) = true;
530             end
531         end
532 
533         % Method 2: Detect points that create sharp angular changes in
```

```
    trajectory
534         % This catches interpolation artifacts that create unrealistic
    direction changes
535         for i = 3:N-2
536             % Get 5-point neighborhood for better stability
537             points = [T.X(i-2:i+2), T.Y(i-2:i+2)];
538 
539             % Calculate vectors
540             v1 = points(3,:) - points(1,:); % vector from i-2 to i
541             v2 = points(5,:) - points(3,:); % vector from i to i+2
542 
543             % Calculate angle between vectors
544             if norm(v1) > 0 && norm(v2) > 0
545                 cosTheta = dot(v1, v2) / (norm(v1) * norm(v2));
546                 cosTheta = max(-1, min(1, cosTheta)); % Clamp to valid
    range
547                 angle = acos(cosTheta) * 180 / pi;
548 
549                 % Calculate distances to assess if it's a sharp spike
550                 d1 = norm(points(3,:) - points(2,:)); % distance to
    previous point
551                 d2 = norm(points(4,:) - points(3,:)); % distance to next
    point
552                 avgDist = mean([norm(points(2,:) - points(1,:)), ...
553                                 norm(points(4,:) - points(3,:)), ...
554                                 norm(points(5,:) - points(4,:))]);
555 
556                 % Flag if sharp angle change AND distance is significantly
     larger than average
557                 if angle > 120 && (d1 > 2*avgDist || d2 > 2*avgDist)
558                     spikeRows(i) = true;
559                 end
560             end
561         end
562 
563         % Method 3: Detect isolated outliers using second derivatives
564         % This catches single-point spikes that deviate from smooth curves
565         if N > 6
566             % Calculate second derivatives for smoothness
567             for i = 4:N-3
568                 % Use 7-point stencil for robust second derivative
569                 x_vals = T.X(i-3:i+3);
570                 y_vals = T.Y(i-3:i+3);
571 
572                 % Calculate second derivatives using finite differences
573                 d2x = x_vals(1) - 2*x_vals(4) + x_vals(7); % i-3, i, i+3
574                 d2y = y_vals(1) - 2*y_vals(4) + y_vals(7);
575 
576                 % Calculate local curvature measure
577                 curvature = sqrt(d2x^2 + d2y^2);
578 
579                 % Get local curvature statistics (excluding current point)
580                 localIndices = max(1,i-10):min(N,i+10);
581                 localIndices(localIndices == i) = []; % Remove current
```

```
    point
582
583                 if length(localIndices) > 5
584                     localCurvatures = [];
585                     for j = localIndices
586                         if j > 3 && j <= N-3
587                             jx = T.X(max(1,j-3):min(N,j+3));
588                             jy = T.Y(max(1,j-3):min(N,j+3));
589                             if length(jx) >= 7 && length(jy) >= 7
590                                 jd2x = jx(1) - 2*jx(4) + jx(7);
591                                 jd2y = jy(1) - 2*jy(4) + jy(7);
592                                 localCurvatures(end+1) = sqrt(jd2x^2 +
    jd2y^2);
593                             end
594                         end
595                     end
596
597                     if ~isempty(localCurvatures)
598                         medianCurv = median(localCurvatures);
599                         madCurv = median(abs(localCurvatures - medianCurv)
    );
600                         if madCurv == 0, madCurv = std(localCurvatures);
    end
601                         if madCurv == 0, madCurv = 1; end
602
603                         % Flag if curvature is extremely high compared to
    local neighborhood
604                         if curvature > medianCurv + 4 * madCurv &&
    curvature > 5
605                             spikeRows(i) = true;
606                         end
607                     end
608                 end
609             end
610         end
611
612         % Clear previous spike markers
613         delete(findall(ax1, 'Tag', 'spike'));
614         delete(findall(ax2, 'Tag', 'spike'));
615         delete(findall(ax3, 'Tag', 'spike'));
616
617         % Mark suspicious points
618         numSpikes = sum(spikeRows);
619         for i = find(spikeRows)'
620             xf = T.Frame(i);
621             xv = T.X(i);
622             yv = T.Y(i);
623
624             axes(ax1); hold on;
625             plot(ax1, xf, xv, 'ro', 'MarkerSize', 10, 'LineWidth', 2, 'Tag
    ', 'spike');
626
627             axes(ax2); hold on;
628             plot(ax2, xf, yv, 'ro', 'MarkerSize', 10, 'LineWidth', 2, 'Tag
```

```
    ', 'spike');
629
630             axes(ax3); hold on;
631             plot(ax3, xv, yv, 'ro', 'MarkerSize', 10, 'LineWidth', 2, 'Tag
    ', 'spike');
632         end
633
634         if numSpikes > 0
635             statusText.String = sprintf('%d interpolation glitches
    detected and highlighted in red', numSpikes);
636         else
637             statusText.String = sprintf('No interpolation glitches
    detected in current data');
638         end
639     end
640
641     function deleteSelectedPlotRows(~,~)
642         selected = dataTable.UserData;
643         plotSelected = data.selectedPoints;
644
645         % Combine table and plot selections
646         allSelected = [];
647         if ~isempty(selected)
648             allSelected = [allSelected; unique(selected(:,1))];
649         end
650         if ~isempty(plotSelected)
651             allSelected = [allSelected; plotSelected(:)];
652         end
653         allSelected = unique(allSelected);
654
655         if isempty(allSelected)
656             msgbox('No rows selected. Click on table rows or plot points
    first.','No Selection');
657             return;
658         end
659
660         numDeleted = length(allSelected);
661         data.editedData(allSelected,:) = [];
662         dataTable.UserData = []; % clear table selection
663         data.selectedPoints = []; % clear plot selection
664         updateTableAndPlots();
665         highlightSuspiciousRows();
666         statusText.String = sprintf('Deleted %d selected rows. %d points
    remaining', ...
667             numDeleted, height(data.editedData));
668     end
669
670     function plotPointClicked(~, ~, plotNum)
671         T = data.editedData;
672         if isempty(T), return; end
673
674         % Get current point location
675         cp = get(gca, 'CurrentPoint');
676         x_click = cp(1,1);
```

```
677         y_click = cp(1,2);
678 
679         % Find closest data point based on which plot was clicked
680         switch plotNum
681             case 1 % Radius vs Frame plot
682                 distances = sqrt((T.Frame - x_click).^2 + (T.X - y_click)
    .^2);
683             case 2 % Height vs Frame plot
684                 distances = sqrt((T.Frame - x_click).^2 + (T.Y - y_click)
    .^2);
685             case 3 % Track plot
686                 distances = sqrt((T.X - x_click).^2 + (T.Y - y_click).^2);
687         end
688 
689         [~, closestIdx] = min(distances);
690 
691         % Handle multi-selection with Ctrl key
692         modifiers = get(gcf, 'CurrentModifier');
693         ctrlPressed = any(strcmp(modifiers, 'control'));
694 
695         if ctrlPressed
696             % Add to selection or remove if already selected
697             if any(data.selectedPoints == closestIdx)
698                 data.selectedPoints(data.selectedPoints == closestIdx) =
    [];
699             else
700                 data.selectedPoints = [data.selectedPoints; closestIdx];
701             end
702         else
703             % Single selection
704             data.selectedPoints = closestIdx;
705         end
706 
707         % Update highlights
708         updatePlotHighlights();
709 
710         % Update table selection to match
711         if ~isempty(data.selectedPoints)
712             dataTable.UserData = [data.selectedPoints, ones(length(data.
    selectedPoints), 1)];
713             statusText.String = sprintf('%d points selected from plots (
    Ctrl+click for multi-select)', ...
714                 length(data.selectedPoints));
715         else
716             dataTable.UserData = [];
717             statusText.String = 'No points selected';
718         end
719     end
720 
721     function updatePlotHighlights()
722         T = data.editedData;
723         if isempty(T), return; end
724 
725         % Clear all existing highlights
```

```
726         delete(findall(ax1, 'Tag', 'highlight'));
727         delete(findall(ax2, 'Tag', 'highlight'));
728         delete(findall(ax3, 'Tag', 'highlight'));
729 
730         % Highlight selected points from both table and plot selections
731         allSelected = [];
732         if ~isempty(dataTable.UserData)
733             allSelected = [allSelected; unique(dataTable.UserData(:,1))];
734         end
735         if ~isempty(data.selectedPoints)
736             allSelected = [allSelected; data.selectedPoints(:)];
737         end
738         allSelected = unique(allSelected);
739 
740         if isempty(allSelected), return; end
741 
742         for i = 1:length(allSelected)
743             idx = allSelected(i);
744             if idx > height(T), continue; end % Safety check
745 
746             X = T.X(idx);
747             Y = T.Y(idx);
748             Frame = T.Frame(idx);
749 
750             axes(ax1); hold on;
751             plot(ax1, Frame, X, 'mo', 'MarkerSize', 12, 'LineWidth', 3, '
    Tag', 'highlight');
752 
753             axes(ax2); hold on;
754             plot(ax2, Frame, Y, 'mo', 'MarkerSize', 12, 'LineWidth', 3, '
    Tag', 'highlight');
755 
756             axes(ax3); hold on;
757             plot(ax3, X, Y, 'mo', 'MarkerSize', 12, 'LineWidth', 3, 'Tag',
     'highlight');
758         end
759     end
760 end
```

Listing 7: Code to for the GUI tool developed to export XY plots

10 References

Andereck, C. D., Liu, S. S. and Swinney, H. L. (1986), 'Flow regimes in a circular couette system with independently rotating cylinders', *Journal of Fluid Mechanics* **164**, 155–183.

Asadi, H., Pearson, T. and Milne, G. (2025), 'Numerical modeling of conventional spin-stop agitation of liquid drug products for automated visual inspection', *International Journal of Pharmaceutics* **670**, 125148.
URL: *https://doi.org/10.1016/j.ijpharm.2024.125148*

Baczewski, J. (2015), 'Flexible inspection of prefilled syringes', *OnDrugDelivery* (61), 35 – 37.

Balachandar, S. and Eaton, J. K. (2010), 'Turbulent dispersed multiphase flow', *Annual Review of Fluid Mechanics* **42**, 111–133.

Benton, E. R. (1974), 'SPIN · UP', (1949), 257–280.

Boubnov, B. and Golitsyn, G. (2012), *Convection in Rotating Fluids*, Fluid Mechanics and Its Applications, Springer Netherlands.
URL: *https://books.google.co.uk/books?id=PaDnCAAAQBAJ*

Bouillet, A., Girois, L. and Weidenhaupt, M. (2021), 'Enhancing protein preservation during biological drug preparation and delivery', *Prefilled Syringes and Injection Devices* (113), 46–48.
URL: *https://www.ondrugdelivery.com/wp-content/uploads/2020/10/Prefilled-Syringes-Injection-Devices-ONdrugDelivery-No-113-Oct-2020-MR.pdf*

Briley, W. and Walls, H. A. (1971), *A numerical study of time-dependent rotating flow in a cylindrical container at low and moderate Reynolds numbers*, Springer, Berlin, Heidelberg, pp. 377–384.

Carpenter, J. F., Fradkin, A. H. and Vessely, C. (2015), 'Meeting biopharmaceutical analytical requirements for subvisible particle sizing and counting'.
URL: *https://www.europeanpharmaceuticalreview.com/article/35952/meeting-biopharmaceutical-analytical-requirements-for-subvisible-particle-sizing-and-counting/*

Challener, C. A. (2019), 'Test methods and quality control for prefilled syringes', *BioPharm International* **32**(3), 44–49. Accessed: 23 June 2025.
URL: *https://www.biopharminternational.com/view/test-methods-and-quality-control-prefilled-syringes*

Chanson, H. (2004), Environmental hydraulics for open channel flows, *in* 'Environmental Hydraulics for Open Channel Flows', Elsevier Butterworth-Heinemann, Oxford, UK, pp. 49–64.
URL: *https://www.sciencedirect.com/science/article/pii/B9780750661652500369*

Chung, J. and Hulbert, G. M. (1996), 'A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method', *Computer Methods in Applied Mechanics and Engineering* **137**(3-4), 175–188.

Colas, A., Siang, J. and Ulman, K. (2006), 'Silicones in Pharmaceutical Applications. Part 5: Siliconization of Parenteral Packaging Components', *Dow Corning Corporation* pp. 1–5.

COMSOL AB (2021), *COMSOL Multiphysics Reference Manual, Version 5.6*, Stockholm, Sweden. pp. 1004–1052.
URL: *https://www.comsol.com/documentation/COMSOL$_R$eferenceManual.pdf*

Crocker, J. C. and Grier, D. G. (1996), 'Methods of digital video microscopy for colloidal studies', *Journal of Colloid and Interface Science* **179**(1), 298–310.

Curia (n.d.), Flexibility and capability in pre-filled syringes: The future of pfs, White paper, Outsourced Pharma. Accessed via Outsourced Pharma; undated.

Drazin, P. G. and Reid, W. H. (2004), *Hydrodynamic Stability*, Cambridge Mathematical Library, 2 edn, Cambridge University Press.

Fagbenle, R. O., Amoo, O. M., Aliu, S. and Falana, A. (2020), *Applications of heat, mass and fluid boundary layers*, 1 edn, Woodhead Publishing.

FDA (2004), 'Sterile Drug Products Current Good Manufacturing Practice Guidance for Industry', *Filtration* (September), 17–18.
URL: *http://www.fda.gov/downloads/Drugs/GuidanceComplianceRegulatoryInformation/Guidances/ucm070342.pdf*

Feinmann, J. (2021), 'Why aren ' t covid-19 vaccines being manufactured in standard prefilled syringes ?', (August 2020), 2020–2021.

Ferziger, J. H., Perić, M. and Street, R. L. (2019), *Computational Methods for Fluid Dynamics*, 4th edn, Springer, Cham, Switzerland.

Ferziger, J. H., Perić, M. and Street, R. L. (2020), *Computational Methods for Fluid Dynamics*, 5th edn, Springer, Cham.

Floryan, J. M. (1992), 'On The Gortler Instability Of Boundary Layers', **28**, 235–271.

Hannam, G., Buck, J., Warburton, M. and Graham, S. (2016), *MECH5030M Team Project: Robotic Inspection Of Pre-Filled Medical Syringes.*

HealthManagement (2012), 'Drug Delivery Systems', **14**(1).

Hlobik, T. (2019), 'Simplifying the process of developing prefilled syringes', *ONdrugDelivery* (101), 76–79.

Hyun, J. M., Leslie, F., Fowlis, W. W. and Warn-Varnas, A. (1983), 'Numerical solutions for spin-up from rest in a cylinder', *Journal of Fluid Mechanics* **127**, 263–281.

International Organization for Standardization (2015), 'ISO 11040-4:2015, Prefilled syringes — Part 4: Glass barrels for injectables', `https://www.iso.org/standard/66047.html`.

Jia, J. (2009), 'A machine vision application for industrial assembly inspection', *2009 2nd International Conference on Machine Vision, ICMV 2009* pp. 172–176.

Kaiser, F., Frohnapfel, B. and Ostilla-mónico, R. (2019), 'On the stages of vortex decay in an impulsively stopped , rotating cylinder'.

Kale, N. S., Kazi, A. and Kale, S. S. (2015), 'Drug prefilled non-reusable syringes as drug device', *World Journal of Pharmaceutical Sciences* **3**(10), 2095–2110.

Kardel, M., Kutz, G., Rosito, R. and Von Zydowitz, H. (2005), 'Quality and economic efficiency of inspection- methods'.

Knapp, J. Z. and Kushner, H. K. (1980), 'Generalized methodology for evaluation of parenteral inspection procedures', *PDA Journal of Pharmaceutical Science and Technology* **34**(1), 14–61.
URL: *https://journal.pda.org/content/34/1/14*

Leversee, R. L. and Shabushnig, J. G. (2008), 'A survey of industry practice a survey of industry practice for the visual inspection of for the visual inspection of injectable products injectable products'.

Maxey, M. R. and Riley, J. J. (1983), 'Equation of motion for a small rigid sphere in a nonuniform flow', *Physics of Fluids* **26**(4), 883–889.

Melchore, J. A. (2011), 'Sound practices for consistent human visual inspection', *AAPS PharmSciTech* **12**(1), 215–221.

Pironti, V., Raghunathan, M. and Hlodan, R. (2020), Challenges And Practical Solutions For Switching To Prefilled Syringes For Injectables, Technical report.
URL: *https://patheon.com/wp-content/uploads/wp-challenges-and-practical-solutions-for-switching-to-prefilled-syringes.pdf*

Public Health England, . (2014), *Vaccine update (Issue 222)*.
URL: *https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/383514/VU_222_Nov_DEC_2014_09_Accessible.pdf*

Rathore, N., Chen, C. and Gonzalez, O. (2009), 'Challenges and strategies for implementing automated visual inspection for biopharmaceuticals'.
URL: *https://www.pharmtech.com/view/challenges-and-strategies-implementing-automated-visual-inspection-biopharmaceuticals*

ResearchAndMarkets.com (2025), 'Prefilled syringes market size, share and trends 2025: Steep rise in the development of biological therapies & high demand for improved safety in injectables – global forecasts 2024–2030'. Published January 2025, accessed 23 June 2025.
URL: *https://www.researchandmarkets.com/reports/4599575/prefilled-syringes-market-size-share-and-trends*

Roache, P. J. (1994), 'Perspective: A method for uniform reporting of grid refinement studies'.

Romacker, M., Schoenknecht, T. and Forster, R. (2008), 'The rise of prefilled syringes from niche product to primary container of choice: a short history', *Prefilled syringes: the container of choice for today's injectables* .
URL: *https://www.ondrugdelivery.com/wp-content/uploads/2018/11/Jun2008.pdf*

Russ, J. C. and Neal, F. B. (2018), *The Image Processing Handbook*, CRC Press.
URL: *https://www.perlego.com/book/2471968/the-image-processing-handbook-pdf*

Sacha, G., Rogers, J. A. and Miller, R. L. (2015), 'Pre-filled syringes: A review of the history, manufacturing and challenges', *Pharmaceutical Development and Technology* **20**(1), 1–11.

Saffman, P. G. (1965), 'The lift on a small sphere in a slow shear flow', *Journal of Fluid Mechanics* **22**(2), 385–400.

Shabushnig, J., Melchore, J. A., Geiger, M., Chrai, S. and Gerger, M. (1995), 'A proposed working standard for the validation of particulate inspection in sterile solutions', *PDA Annual Meeting* .

Sommerfeld, M., Schmidt, F., Kück, T., Möller, W. and Häussermann, S. (2019), 'Potential and constraints for the application of cfd combined with lagrangian particle tracking to dry powder inhalers', *European Journal of Pharmaceutical Sciences* **128**, 299–324.
URL: *https://doi.org/10.1016/j.ejps.2018.12.005*

Statista (2025), 'Pharmaceutical market worldwide'. Accessed: 23 June 2025.
URL: *https://www.statista.com/outlook/hmo/pharmaceuticals/worldwide*

Syntegon (2021), *MIH-1 Manual Workstation For Visual Inspection Of Pharmaceutical Products.*
URL: *https://www.syntegon.com/products/mih-1-manual-workstation-visual-inspection-pharmaceutical-products*

Syntegon (2021 a), *AIM 8000 Series.*
URL: *https://www.syntegon.com/products/aim-8000-series-automatic-inspection-machines*

Turner, M. (2021), 'Verification of injectables in transport and storage', *Prefilled Syringes And Injection Devices* (113), 56 – 58.
URL: *https://www.ondrugdelivery.com/wp-content/uploads/2020/10/Prefilled-Syringes-Injection-Devices-ONdrugDelivery-No-113-Oct-2020-MR.pdf*

Ullherr, K. (2008), 'New technologies for the processing of syringe nests', *Prefiller Syringes: The Container Of Choice For Today's Injectables* (1), 34 – 36.
URL: *https://www.ondrugdelivery.com/wp-content/uploads/2018/11/Jun2008.pdf*

Vanyo, J. P. (2015), *Rotating Fluids in Engineering and Science*, Dover Publications, Inc.

Versteeg, H. K. and Malalasekera, W. (2007), *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, 2nd edn, Pearson Education, Harlow, UK.

Waxman, L., Murray, H. and Vilivalam, V. (2012), 'Evaluation of piston movement and container integrity under severe storage conditions in plastic and glass prefilled syringes'. **URL:** *https://www.westpharma.com/en/blog/2012/May/Evaluation-of-Piston-Movement-under-Severe-Storage-Conditions-in-Plastic-Glass-Prefilled-Syringes*

Wedemeyer, E. H. (1964), 'The unsteady flow within a spinning cylinder', *Journal of Fluid Mechanics* **20**(3), 383–399.

Weidman, P. D. (1976*a*), 'On the spin-up and spin-down of a rotating fluid. Part 1. Extending the Wedemeyer model', *Journal of Fluid Mechanics* **77**(4), 685–708.

Weidman, P. D. (1976*b*), 'On the spin-up and spin-down of a rotating fluid. Part 2. Measurements and stability', *Journal of Fluid Mechanics* **77**(4), 709–735.

West Pharmaceutical Services, I. (2021), *Image of Syringe*. **URL:** *https://www.westpharma.com/en/blog/2017/October/west-rns-caps-now-available-in-two-designs*

White, F. M. (2008), *Fluid Mechanics*, 7 edn, The McGraw Hill.

Wu, L., Li, H., Wang, Y., Liu, C., Zhao, Z., Zhuang, G., Chen, Q., Zhou, W. and Guo, J. (2024), 'Advancing injection force modeling and viscosity-dependent injectability evaluation for prefilled syringes', *European Journal of Pharmaceutics and Biopharmaceutics* **197**, 114221. Available online 18 February 2024. **URL:** *https://doi.org/10.1016/j.ejpb.2024.114221*

Zhou, B., Wang, Y., Ge, J. and Zhang, H. (2008), 'A machine vision intelligent inspector for injection', *Proceedings - 2008 Pacific-Asia Workshop on Computational Intelligence and Industrial Application, PACIIA 2008* **2**, 511–516.