Optically Clear Biomicroviscometer with Modular Geometry Using Disposable PDMS Chips

by

Niko Lee-Yow

Thesis submitted in partial fulfillment of the requirements for the

Master of Applied Science Degree in

Biomedical Engineering

Ottawa-Carleton Institute of Biomedical Engineering

Faculty of Engineering

University of Ottawa

Abstract

We have designed and fabricated a biomicroviscometer platform for measurement of microflows of biological fluids. The biomicroviscometer combines an optically clear biocompatible polydimethylsiloxane (PDMS) channel with on-chip integrated microfluidic differential pressure sensors and capabilities of modular channel geometries. This setup allows for a direct measurement of the change in pressure and flow rate, increasing the overall accuracy of the measurement of viscosity and optical observation. We present an introduction of this combined method of measurement with different channel dimensions, using Newtonian and non-Newtonian fluids, and the corresponding calculations. This measurement technique has potential applications in measuring rheological properties at the micro level to further blood disease analysis, and lab-on-a-chip fabrication and analysis.

Acknowledgements

I would like to first acknowledge my supervisor, Dr. Marianne Fenech. Without her support, expertise of research, and excellent advice throughout my thesis, the experimental research would not have been possible. Her guidance and encouragement have helped greatly with the many versions of the thesis. I would not have become the researcher I am without the inspiration you have given me to complete my research.

I would also like to thank Dr. Laura Haya for assisting me by providing excellent suggestions while reading through my thesis which have improved the thesis.

I am thankful to both Dr. Patrick Dumond and Dr. Metin Yaras for agreeing to examine my thesis and taking their time for its evaluation.

The whole microfluidic laboratory group has been a pleasure to work with and has brought together great minds and great memories. I am very grateful to all of the people that I have met in the lab, who have all aided with experiments and conversed about various research. In particular, I would like to thank Katie Pitts, Rym Mehri, Erfan Niazi, Mark Stroobach, Eleanor Gerson, and Omemah Gliah for their assistance in various stages of my project.

I would especially like to thank my friends and family, of whom I do not consider mutually exclusive to those previously mentioned. My parents, sister and grandparents have provided constant support and wisdom throughout this endeavour. I would also like to thank my girlfriend, Caitlin, for her support and patience while listening to all the times I have practised presentations and helping to edit my thesis. Additionally, I would like to thank my mother for reading through my thesis and providing grammatical support. Friends and family have made the journey throughout my research outstanding, both while working on my thesis and during everyday tasks.

I would also like to acknowledge funding from Canada Foundation for Innovation (CFI) and Natural Sciences and Engineering Research Council of Canada (NSERC); funding and design assistance from CMC Canada, especially Zhao Lu; as well as design and chip fabrication assistance from Robert Stevenson of A.L.Bi.C.S. Ltd, Canada.

List of Symbols

2Δ	Distance between two adjacent microposts
a	Radius of a cylindrical channel or longest side length of a rectangular channel
a^*	Non-Newtonian wall shear rate formula constant
A	Cross-sectional area of the flow region
A'	Portion of the selected area which the fluid can flow through
$A"$	The total area comprising both the flow area and the fibre area
b_1	Longer parallel side of the trapezoid
b_2	Shorter parallel side of the trapezoid
b^*	Non-Newtonian wall shear rate formula constant
cP	Centipoise
C_{hyd}	Compliance in the system
d_1	Depth/height of the channel
d_2	Width of the channel
D	Diameter of the input channel
D_h	Hydraulic diameter
E	Young’s modulus
f^*	Wall shear rate formula constant
g	Acceleration due to gravity
h	Height of the chamber
K	Permeability constant
$K_\perp$	Permeability constant for a quadratic arrangement of pillars
$K_{power\ law}$	Non-Newtonian power law constant relating to viscosity
K_{POLYOX}	Power law constant K for polyox
K_{settle}	Settling constant
l_i	Length of the input channel
l_o	Length of the output channel
L	Length of the channel
L_c	Characteristic length

L_e	Entrance length
$n_{power\ law}$	Non-Newtonian power law constant
n_{POLYOX}	Power law constant n for polyox
P	Pressure
Q	Flow rate
r^2	Coefficient of determination
r_{RBC}	Radius of a red blood cell
R	Radius of a micropost
R_{hyd}	Hydraulic resistance
R_i	Inner radius of the channel
R_o	Outer radius of the channel
Re	Reynolds number
t_{settle}	Settling time
$V_{f\ max}$	Maximum possible fibre volume fraction
v	Voltage
V	Average flow velocity
V_f	Volume fraction between the pillars
V_{int}	Internal volume
w	Channel width
x	Aspect ratio constant
X_{hyd}	Hydraulic resistance constant
$\dot{\gamma}$	Shear rate
$\dot{\gamma}_a$	Wall shear rates for Newtonian fluids
$\dot{\gamma}_w$	Wall shear rate for the non-Newtonian fluids
ΔP	Pressure drop
μ	Dynamic viscosity
μ_{app}	Apparent dynamic viscosity
ρ	Density
σ	Poisson’s coefficient
τ	Shear stress

$\tau_{air\ compliance}$	Relaxation time due to compliance of air pockets
$\tau_{deformable\ walls}$	Relaxation time considering deformable walls
$\tau_{rigid\ walls}$	Relaxation time considering rigid walls
τ_{slit}	Shear stress for slit geometry
$°C$	Degree Celsius

List of Acronyms

3D	Three-Dimensional
$CaCO_3$	Calcium Carbonate
CFD	Computational Fluid Dynamics
CFI	Canada Foundation for Innovation
CFL	Cell-Free Layer
DI	De-ionized
FFHMT	Fluid Flow, Heat and Mass Transfer
FRP	Flow-Rate Platform
GUI	Graphical User Interface
HF	Hydrofluoric Acid
I.D.	Inner Diameter
IRPHE	Institut de Recherche sur les Phénomènes Hors Equilibre
MATLAB	Matrix Laboratory
MeMeA	Medical Methods and Applications
MEMS	MicroElectroMechanical Systems
MFCS-EZ	MultiFluidic Control System-EZ
MSDS	Material Safety Data Sheets
m-VROC	m-Viscometer/Rheometer-On-a-Chip
NOA60	Norland Optical Adhesive 60
NSERC	Natural Sciences and Engineering Research Council of Canada
O.D.	Outer Diameter
PBS	Phosphate Buffered Saline
PDMS	Polydimethylsiloxane
PEEK	Polyetheretherketone
PMMA	Polymethylmethacrylate
POLYOX	Polyethylene Oxide
PPE	Personal Protective Equipment
RBC	Red Blood Cell

SOP	Standard Operating Procedure
USB	Universal Serial Bus
UV	Ultraviolet
w/v	Weight/Volume
w/w	Weight/Weight
μPIV	Microparticle Image Velocimetry
μVISC	Micro-Viscometer

Table of Contents

List of Figures

List of Tables

1. Introduction

The introduction section first presents the rheology of blood and its importance in diagnoses of many diseases. The influence of blood flow phenomena, such as aggregation and the cell-free layer (CFL), at the microscale is then explained. Current methods for measuring the viscosity of blood are described and their limitations investigated. This chapter concludes by outlining the research objectives, encompassing the research aims and project objectives.

1.1.Blood Rheology

Blood is an essential type of connective tissue that provides the body with transportation of dissolved gases, regulation of body functions, and protection from exterior organisms and matter [1]. Because of its importance, various blood flow phenomena are of great scientific interest, and because of its complex behaviour, there is much left unknown about blood flow phenomena, making it a topic for current research and experimental modelling, particularly at the microcirculation scale [2]. Blood is comprised of solid particles (mainly flexible cells and platelets) suspended in liquid plasma (a Newtonian matrix). The majority of the solid particles in blood consist of red blood cells (RBCs), white blood cells and platelets. RBCs are the most abundant of the blood cells and are shaped as biconcave discs. These deformable cells normally have a diameter of approximately 7 to 8 µm, and a thickness (at the thickest part of the disk) of about 2 – 2.5 µm [1], [3], [4]. The plasma matrix is composed of about 91.5% water, 7% being proteins and 1.5% of other dissolved solutes [1]. These are average values, since they vary from one person to the next, depending on factors such as genetics (uncontrollable) and diet (controllable).

Healthy human beings have approximately 40% to 45% of the blood volume comprising of RBCs [1]. This volume fraction of RBCs within the blood is called the hematocrit, and is often given as a percentage. Healthy RBCs have an average density of approximately 1.10 kg/m^3 [4], [5]. The plasma portion by itself has an average density of approximately 1.03 kg/m^3, while whole blood with all its constituents at 45% hematocrit has an average density of approximately 1.06 kg/m^3 [4], [6].

The rheological behaviour of blood is complex at the micro level, where the rheological characteristics of a blood sample depend on factors such as hematocrit, temperature, vessel dimension scale, flow rate, and shear rate. Scale-dependent behaviour is especially important in the microcirculation, which is the main site for nutrient and gas exchange within the vascular network. As the size of the channel and/or the shear rate decreases, the non-homogeneity of the blood becomes increasingly important [2–4]. The size of the cellular components in comparison to the size of the channels, and the interactions between the cells with each other and with the vessel walls become significant. Other factors that influence the blood behaviour include diseases or pathologies affecting the shape and material properties of the RBCs.

Blood flow at this micro level is particularly challenging to model, due to the local non-homogeneity, since the continuum assumption of the Navier-Stokes equations (the governing equations describing fluid motion) breaks down. Newtonian fluids, such as water and blood plasma, exhibit a constant viscosity with applied shear rate. Blood, as a whole, however, is a non-Newtonian fluid and its viscosity is dependent on both the level of applied shear rate and the direction of the rate of shear loading (whether it is increased or decreased) due to cellular aggregation and deformation [7]. Unlike for a Newtonian fluid such as de-ionized water (DI water) (shown in Figure 1 as blue), the viscosity of non-Newtonian blood is dependent on the level of shear rate applied to it. Blood flows easier (i.e. has a lower viscosity) when increased levels of shear are applied to it – a behaviour called shear thinning (illustrated as the red curve in Figure 1).

Blood also exhibits a fluidic hysteresis, in which the viscosity depends on whether the shear rate on the fluid is being increased or decreased (shown as a non-linear path in Figure 2). This dependency on the shear rate is linked to the ability of the suspended fluid particles (RBCs in the case of blood) to move around in the three-dimensional (3D) fluid structure of blood over time [9]. At body temperature, the viscosity of normal human blood roughly ranges between 4 and 5 mPa·s at shear rates above 100 s^{-1} [10]. The viscosity increases with further decreases in the shear rate, with example values listed in Table 1. The hysteresis behaviour of blood is characterised as "thixotropic", meaning that the viscosities are higher when the shear rates are increasing (loading) than when they are decreasing (deloading); this is depicted in Figure 2, in which the loading and deloading paths are labelled 1 and 2, respectively.

Table 1. Average viscosity of human blood at various shear rates at human body temperature (37 °C) [10].

Shear Rate (s^{-1})	**Viscosity of Human Blood (mPa·s)**
> 100	4 to 5
10	10
1	20
0.1	100

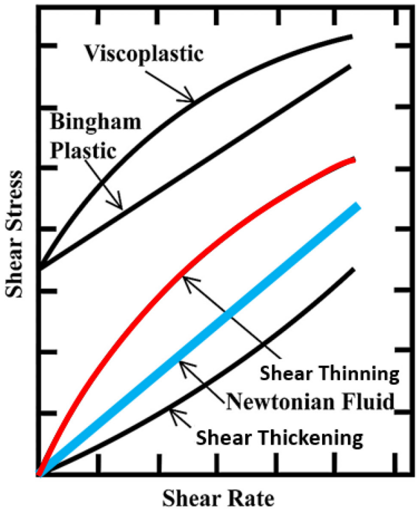


Figure 1. Plot of different types of fluids, where the slope of the line indicates the viscosity. Two curves in colour compare shear thinning (red) to a Newtonian fluid (blue). (Adapted from [11]*)*

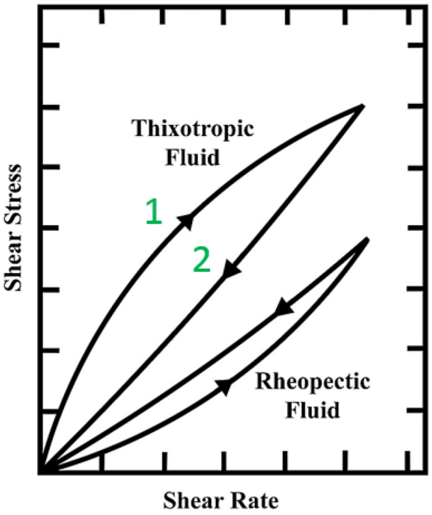


Figure 2. Plot of fluid indicating a rheological hysteresis. Numbers indicate the path that the viscosity would take for a thixotropic fluid when increasing (1) and then decreasing (2) the shear rate. (Adapted from [11]*)*

The thixotropic nature of blood is due to the time dependence of the RBCs' abilities to aggregate, deform, and orient themselves in a 3D orientation [3], [9]. Since these effects are time dependent, these phenomena might not fully occur while the shear rate increases or decreases, and cause the apparent viscosity to depend on the direction of shear rate loading [9]. Bureau *et al.* [7] studied rheological hysteresis of blood using Couette flow viscometers for healthy human blood, to compare with samples having pathological conditions. They linearly varied the shear rate from 0 to 0.12 s^{-1} and back down to 0 s^{-1}, in triangular steps as shown in Figure 3, in addition to linearly increasing from 0 to 1.03 s^{-1} and back down to 0 s^{-1} [7]. The model of Bureau *et al.* [7] showed that certain model constants relating to the thixotropic or viscoelastic properties varied between the healthy and diseased blood flows, depending on how blood components were affected by the pathological conditions.

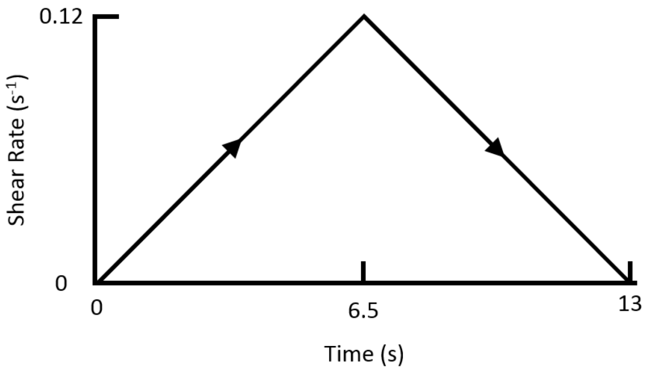


Figure 3. Variation of shear rate in triangular steps as used by Bureau et al. [7].

Constitutive models have since been proposed to model the aggregation and disaggregation of blood [3], [8], which agree with the experimental data from Bureau *et al.* [7]. Owens [3] derived a microstructure based constitutive equation considering the elastic stress and average aggregate size, which closely agreed with the experimental data from Bureau *et al.* [7], and indicated stress hysteresis with viscoelastic and shear thinning properties. Fenech *et al.* [8] derived a particle-based equation considering the elastic and aggregation forces, which also closely agreed with the experimental data from Bureau *et al.* [7] and model from Owens [3], when comparing the mean average aggregate size of RBCs to the shear rate.

These models, however, only consider the flow of blood in simple geometries and do not consider the effect of the network geometry itself. In this thesis work, we aim to investigate the rheology of blood in the microcirculation. More specifically, we aim to investigate the viscous behaviour of blood, which has often been reduced to an equivalent viscosity in a single microchannel [3], [8], and we will consider the network as a whole.

1.1.1. Blood Viscosity as a Diagnostic Measurement

Abnormal levels of blood viscosity, as a diagnostic measurement, have been related to acute myocardial infarction, acute stroke and chronic cerebrovascular disease [12], [13]. For acute myocardial infarction, the blood viscosity was seen to have increased after admission to hospital (within a 24-hour time span from symptoms of chest pain), and then reduce as a function of time [12]. An increase in blood viscosity after an event would cause higher fluidic resistance for the heart to overcome. A treatment to reduce the viscosity and improve the microcirculation could be of benefit for such patients [12].

For acute stroke or chronic cerebrovascular disease, echocardiogram measurements have been used to identify abnormalities, with there being a significant difference between healthy and marked patients, with the latter exhibiting a “smoke-like” appearance of blood [13]. These “smoke-like” images had higher contrast than those of the healthy patients. This was found to be linked to the blood viscosity, which was altered by abnormal levels of aggregation caused by abnormal concentrations of proteins in the blood of the patients suffering from acute stroke or chronic cerebrovascular disease [13]. Blood viscosity has also been investigated in its relation to individuals with coronary heart disease [14]. Regions inside the vessels having lower shear stress tend to correspond with regions where plaque formations occur [14]. Measuring the viscosity and the flow within vessels could give an estimation of where plaque formation would occur before an event occurs. Since blood is comprised of RBCs, plasma, dissolved ions, fibrinogen and other proteins, it is important to investigate the effects of all these constituents on the blood viscosity at the microchannel size.

1.2.Blood Flow in Microcirculation

It is hypothesized that the tissue perfusion (the passage of blood through the tissue) will be influenced by the network geometry, channel size, and bifurcation angles of the vessels within it [2], [15]. Changes to these factors affect the RBC interactions with both each other and with the vessel walls. Under low shear rates, these interactions cause the natural clustering of RBCs over time, called aggregation, and the formation of an RBC CFL. Aggregation is a reversible process, where the RBC clusters will separate at high shear rates. These phenomena cause physiological effects such as changes in the shear forces acting on the vessel walls, oxygen exchange rate, and nutrient transport [2]. For example, in the microcirculatory system, in addition to observing RBC aggregation and CFL, it has been shown that partitioning of RBCs at bifurcations is often unbalanced [15]. The repartition of the RBCs in the mother or daughter branches could be affected, among other factors, by the RBC deformability, the system geometry or the local hematocrit. In addition, RBC aggregation and CFL thickness are directly influenced by the local hydrodynamics.

1.2.1. RBC Deformation

RBCs are specialized flexible cells that contain no nucleus and maximize oxygen transport without consuming any of the oxygen that they are transporting [1]. Their biconcave shape, as shown in Figure 4 (a), allows for the ratio of surface area to volume to be high and also for the cell to be flexible [16], [17]. These cells deform in the capillaries to increase the surface contact area with the vessel walls, facilitating gas exchange. Since some of the smaller capillaries are 3 to 4 μm in diameter, the cells must be able to deform in order to pass without breaking their outer membrane [18]. The deformation properties of RBCs also contribute to the variation of blood viscosity as it flows in vessels or channels. In the microcirculation, the deformation of RBCs depends on the vessel conditions, such as the dimensions and shape. In cylindrical vessels ranging from 3 to 13 μm, RBCs can take the shape of a parachute [19]. The parachute shape is depicted in Figure 4 (c), and other examples of RBC shapes include "slippers", as shown in Figure 4 (b), with different slipper shapes depending on the fluid interactions [20]. For these "slipper" shapes, the cell membrane is in flexible rotation relative to the cell's contents, similar to how a tractor's or tank's treads would rotate around its wheels while moving across land.

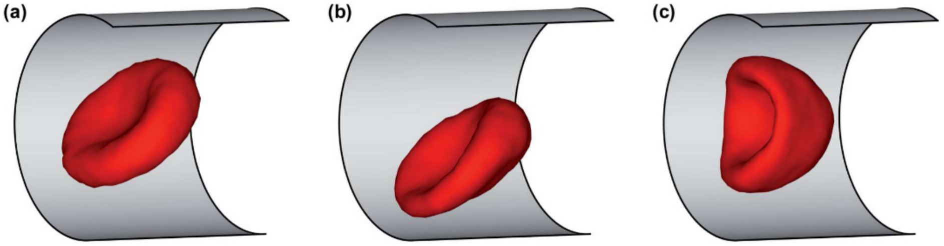


Figure 4. Simulation of RBCs in flow with (a) biconcave, (b) slipper cell, and (c) parachute shape [20].

The ability of RBCs to deform can thus affect a person's health, if they are not deformable enough. This can result, for example, from advanced cell age or from some diseases. The effect of RBC age was investigated using a cone-plate rheoscope, where older cells were observed to tumble compared to the younger ones which deformed more easily [18]. This result suggests that, while doing experiments *in vitro*, it is important to keep in mind the age of RBCs relative to the time of collection. A number of diseases can also affect the deformation of RBCs, with the membrane displaying abnormal shapes or having abnormal stiffnesses. The deformability of RBCs can be affected by hereditary diseases, such as type II diabetes mellitus, sickle cells, or hypercholesterolemia. In diabetes, the recovery time for a RBC to return to its original shape after being deformed is slowed, and cell deformability is additionally impaired [17]. Sickle cell disease causes the cells to have a reduced deformability due to abnormal sickle haemoglobin, which causes a cell to form with a sickle shape [21]. The RBCs are also known to deform due to being enriched in cholesterol, which causes the cells to become pancake-shaped with a wavy edge [17]. Since the viscosity depends on how the RBCs flow, the differentiation between normal and diseased RBCs is important.

1.2.2. Aggregation

Aggregation, illustrated in Figure 5, is the natural clustering of the blood cells over time. Aggregation differs from clotting, which is the trapping of RBCs in a fibrin net following platelet activation and fibrin synthesis. The RBCs naturally tend to cluster together and form structures that resemble coin stacks, called rouleaux [22], [23]. This process is one of the mechanisms that cause the viscosity of blood to increase as the shear rate decreases, because aggregation typically occurs under low shear rate. The effect of aggregation starts being apparent at shear rates lower than 100 s^{-1}, with an increase of viscosity at shear rates lower than 10 s^{-1} [24]. It is a reversible process, since at higher levels of shear the RBC aggregates are broken up into individual RBCs in a process called "disaggregation". The aggregation phenomenon and the mechanisms that cause it to occur are still not fully understood. There are a few competing hypotheses explaining the cause of aggregation, and several parameters are known to affect the aggregation rate. Diseases such as sickle cell disease, for example, can interfere with the aggregation process by affecting the plasma proteins or cellular shape[21].

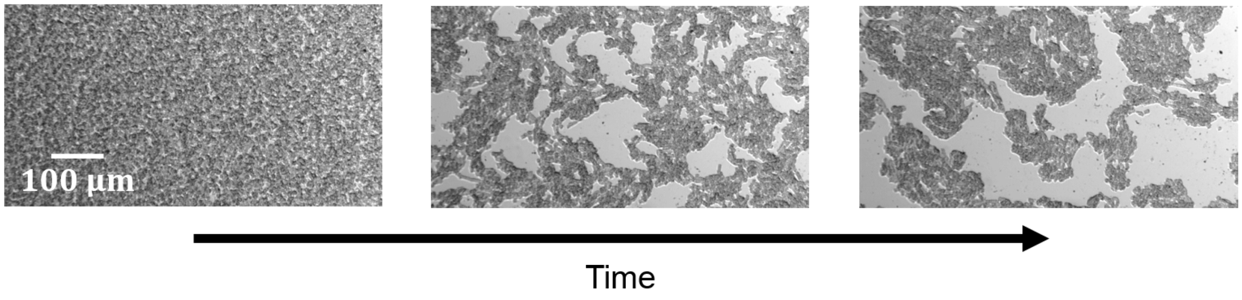


Figure 5. Human blood's fluidic complexity due to cells and aggregation, where the blood cells aggregate together over time. (Images courtesy of Rym Mehri, University of Ottawa)

The mechanisms which are believed to cause aggregation are based either on (1) the bridging of adjacent RBCs, or (2) the depletion of molecules between cells [23]. The "bridging" theories purport that macromolecules such as fibrinogen, that are attached to the surface of the RBCs, are able to contact molecules of their type on adjacent cells, and bridge under favourable conditions [22], [25]. If the forces acting to keep the cells apart are sufficiently high, e.g. due to fluid flow or flow interaction, the macromolecules are unable to connect to cause aggregation. In comparison, the depletion theories purport that when the surfaces of adjacent cells are sufficiently close, the concentration of macromolecules between the cells becomes relatively low (compared to the bulk fluid), and a resulting osmotic pressure is generated that causes the cells to be attracted towards each other [23]. The aggregation effect could be the result of a combination of these explanations, but a complete understanding of aggregation would warrant investigating the molecular interactions involved.

Several measurable parameters are known to affect aggregation, such as the fluid shear rate, blood hematocrit, and temperature. The effect of shear rate on aggregation of RBCs has been investigated extensively [3], [7], [8], [26]. At high shear rates, the RBCs do not have sufficient time to interact with each other, and the flow forces are too great for them to aggregate. Lower shear rates allow the RBCs to aggregate together and this increased aggregation, in turn, increases the viscosity of blood. Since hematocrit is the relative percentage of RBCs in blood, a greater hematocrit increases the interactions or collisions between the cells. A greater number of interactions then causes an increase in aggregation. The temperature of the blood affects the rate of aggregation, with the speed depending on the temperature range [22], [27]. Maeda *et al.* [22] found that, at a shear rate of 7.5 s^{-1}, the aggregation speed decreases when increasing the blood temperature from 5 °C to 20 °C, whereas the aggregation speed would increase as the temperature increased further, from about 20 °C to 43 °C; in other words, the rate of aggregation was minimum at approximately 20 °C. Singh *et al.* [27] performed a study of the aggregation indices measured in blood and found similar results using a fully automatic aggregometer, which could apply shear rates up to 600 s^{-1}. This trend of temperature-dependent aggregation has been explained to be the result of the combined effects of various phenomena that change with temperature: the frequency of cell collisions (which increases with temperature and promotes aggregation), fluid shear forces (which increase with temperature and counteracts aggregation), and the properties of macromolecules and RBCs, themselves [22]. The effect of temperature on the storage of the blood cells should also be considered when planning experiments. The storage conditions need to be considered, since RBC deformability changes significantly after six hours at room temperature (of 25 °C) and the RBC aggregation changes after only four hours at room temperature [28].

1.2.3. Cell-Free Layer (CFL)

As blood travels through a channel, the RBCs tend to concentrate towards the centre of the channel, described by a phenomenon called the Fåhræus–Lindqvist effect [29]. Since RBCs are suspended in plasma, a core of RBCs is formed with a cell-free plasma layer existing between the outer diameter of the RBC core and the channel wall, as shown in Figure 6.

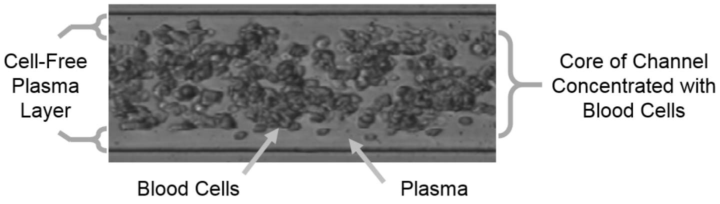


Figure 6. Flow of blood in a microchannel which is 100 μm in width, depicting the cell-free layer phenomena of blood flow in microchannels. The cells are concentrated in the core of the channel and leave a cell-free plasma layer. (Adapted from image provided by Omemah Gliah, University of Ottawa)

The formation of the CFL is not yet well understood, but potential causes include collisions, lift force and shear diffusion [30]–[32]. The formation of the CFL is complicated by the fact that these potential causes are in opposition to the interactions between the cells within the flow and the tendency for the cells to move toward the channel walls [30]. The effect of collisions depends on factors such as the hematocrit of the blood flowing through the channel, and aggregation forces. An increase in the hematocrit causes there to be a greater RBC-to-plasma volumetric ratio in the channel and decrease the CFL thickness, whereas the increased frequency of collisions promotes aggregation and increases the CFL thickness [30]. RBCs experience a lift force away from the wall due to the ability for the RBC membrane to deform and create asymmetric shapes which causes changes in the flow [31], [33]. Shear diffusion phenomena is caused by the random collisions of particles within a flow causing large particles to migrate to regions of low shear rate and cause a diffusive effect [34].

Fåhræus and Lindqvist [29] found from experiments using a glass capillary viscometer that the Poiseuille law could not be used to accurately determine the viscosity of blood in cylindrical channels having diameters smaller than 300 μm, owing to the significant effect of the CFL. Since the viscosity of the RBC core is greater than that of the plasma, the apparent viscosity of the fluid will be affected if the CFL is considerably large, relative to the vessel size. It has been mathematically estimated that if the CFL is greater than about 20% of the tube radius, the apparent viscosity can be approximated as being independent of the flow rate (i.e. shear rate) [35]. Differences between measurement methods of the CFL and viscosity require consideration due to the effect of the biological environment on the CFL. *In vivo*, the glycocalyx proteins on the endothelial cells of the vessel walls as well as the microvessel geometry can reduce the formation of the CFL, compared to in glass capillary tubes [32].

The CFL thickness is affected by factors such as the channel dimensions, shear rate, and aggregation. As the ratio of channel diameter to RBC diameter decreases, the RBCs have a more significant effect on the flow. Several studies have shown that vessels with radii between 5 μm and 8 μm (on the order of RBC diameters) show a slight increase of CFL thickness, relative to the vessel radius, but for vessels larger than this, the relative thickness ratio decreases [30]. The thickness of the CFL has also been shown to decrease as the shear rate of the flow increases [36]. This effect is linked to the aggregation, as studies show that the layer thickness was greater when the aggregation rates were increased (which occurs under low shear/flow conditions) [37]. The increased presence of larger aggregates in the core of the flow was due to (1) larger aggregates moving more towards the vessel centre than smaller aggregates and individual cells, and (2) stronger aggregation occurring at lower shear rates (as the shear rates near the vessel centre are lower than near the walls), causing the core RBCs to aggregate more [37]. Conversely, an increase in the shear rate causes disaggregation of the RBCs, which causes the CFL thickness to thin [37].

An interesting effect that occurs within the CFL is that particles smaller than RBCs within the blood tend to migrate to the layer, an effect called margination. This effect has been investigated both in experiments and computer simulations framed toward the application of drug delivery. It has been found that using certain sizes and shapes of small drug particles, in this margination range, resulted in a more uniform or targeted dose uptake [38], [39].

1.3.Existing Rheometric Measurement Technologies

There are currently many commercial ways to measure rheological properties in Newtonian and non-Newtonian fluids. These methods include capillary viscometry, cone and plate and truncated cone and plate rheometry, spinning or oscillating parallel disk or plate rheometry, Couette viscometry and axial annular flow measurements. There are entire texts devoted to standardizing rheometry methods [40], [41].The cone-plate, capillary and rectangular-slit viscometry methods are further investigated in the following subsections.

These methods, however, all work at the macroscopic level and have significant limitations when applied to micro scale samples. Each of the standard methods results in errors, such as those due to zeroing the gap between cones or plates. In these instances, as the gap narrows, the error increases. Additionally, significant time and effort must be put into calibration using standardized fluids. Test samples are also difficult to contain in these calibrated devices. When applied to microlitre samples sizes, these methods can result in unacceptably high levels of error and the loss of sample is too significant. These drawbacks are where the drive to use microrheometric methods arises.

Microrheometric methods have taken many forms in recent research. There are optical methods such as trapped air compression in microfluidic channels filled with water [42] and fluid-fluid interface measurements [43]–[45]. The trapped air compression method uses a dead-end channel which is connected to the channel where the pressure is to be measured. When pressure is applied to the main channel, a fluid plug within the measurement channel moves into the measurement channel and compresses the air in the dead-end channel. The amount that the air is compressed correlates to the pressure within the main channel, but also depends on the geometry of the measurement channel and compressibility of the air [42]. In fluid-fluid interface measurements, two immiscible fluids (a sample fluid and a reference fluid) are flowed in parallel, into a single channel. The location of the interface between the liquids, the fluids' respective flow rates, and the viscosity of the reference liquid can be used to calculate the pressure drop and viscosity of the sample fluid [43]–[45]. The equations relating the pressure drop to the viscosity, for various channel shapes, are located in the Working Equations Section 2.2.1. Other methods that are used to measure the viscosity of biological fluids, such as blood, include measuring the impedance of microlitre sized samples [46] or using continuous flows of the sample and reference fluid in an array of parallel channels [47]. Finally, there are micro pressure sensor measurements [48] which have been applied to water and non-Newtonian fluids, but they have not yet been applied to blood.

1.3.1. Cone-Plate Viscometer

In a cone-plate viscometer, a cone rotates on a flat surface called the plate, as shown in Figure 7. The analysis fluid is located between the surfaces of the cone and the plate and the sheared flow is generated as the cone rotates. Measurement of viscosity is completed by using the dimensions of the setup and measuring the torque resistance created by the viscosity of the fluid as the cone rotates. Variations of the geometry of this viscometer exist such as plate-plate or truncated cone-plate configurations. The rotation of this viscometer can be modulated to test various patterns of shear rates, which is important for fluids exhibiting hysteresis and it is more difficult to do using flow systems with rigid/fixed geometries [49]. The setup can be paired with a transparent plate and an inverted microscope so that the fluid flow may be viewed.

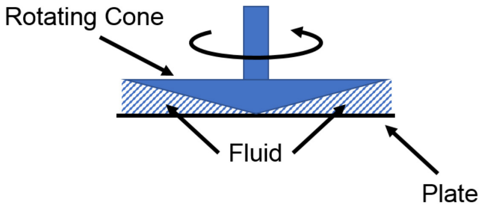


Figure 7. Diagram of cone-plate viscometer measurement method indicating where the fluid is located and the rotating mechanism. The viscosity can be measured using the resistance torque experienced by the rotating cone, due to the viscosity of the fluid.

In this viscometer, the velocity profile of the fluid is linear between the surfaces, instead of being parabolic like in cylindrical channels. Since blood flows through near-cylindrical channels *in vivo*, this means that the velocity profiles are not indicative of true blood flow. The ability to correlate the viscosity measurement to the velocity profile, in three dimensions, of the RBCs in a cylindrical channel is missing and would not account for important information about the RBC interactions. Furthermore, due to the experimental setup, the fluid is exposed to the air. Depending on the temperature the measurement is performed at, this could cause significant evaporation over the duration of the experiment, which could greatly affect the measurement. Proteins in blood tend to concentrate at the surface of the blood-air and plasma-air interfaces which also affects the shear force measured [50]. Proteins at these surfaces can add significant resistance torque at low shear rates, due to the non-Newtonian time-dependent effects of these proteins during an experiment [50].

1.3.2. Capillary Viscometer

Capillary viscometers work by having a sample fluid flow through a capillary tube of known diameter, which is usually made of glass, as shown in Figure 8. A pressure differential between the two ends of the tube is generated using methods such as a syringe pump or a height difference between the inlet and outlet using raised volumes [29], [51]. The pressure difference, which causes the fluid to flow through the tube, is measured using manometers or pressure sensors along the flow [51]–[53]. The viscosity is computed using the flow rate of the fluid, the pressure measurement, and the dimensions of the capillary tube. The use of transparent material for the tubes allow the fluid flow to be imaged as it moves through the tube.

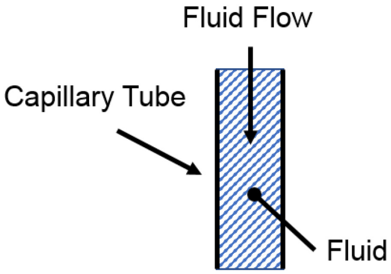


Figure 8. Diagram of capillary viscometer measurement method indicating the fluid flow through a capillary tube. The viscosity can be determined using the fluid flow rate and the differential pressure between the tube ends.

The cylindrical cross-section of the capillary tube more closely mimics a single blood vessel and allows a parabolic velocity profile to develop. This type of viscometer has been used to measure the aggregation at low shear rates in both vertical and horizontal tubes [29], [35], [54]. Horizontal arrangements of the capillary viscometer, however, can result in sedimentation of suspended particles, like blood cells, to occur during an experiment and affect the viscosity measurement. Additionally, the range of pressures and flow rates that are measurable by capillary viscometers are limited by the length of the capillary tubes. Shorter capillary tubes would be preferred for lower shear rates since they do not require a high pressure drop, but the flow would be more difficult to image.

1.3.3. Rectangular-Slit Viscometer

Viscosity can be measured using a rectangular-slit viscometer by flowing the sample fluid at a known flow rate through a channel with a cross-section of a rectangular-slit, where the channel's width is much larger than the height, and is driven by a pressure difference between the two sides, as is shown in Figure 9. The pressure difference can be generated using a syringe, compressed gas or using volumes of liquid attached to the inlet and outlet set at different heights. The pressure is measured using several pressure sensors integrated within the rectangular-slit. By using a sufficiently high slit aspect ratio, whereby one width of the rectangle is much greater than the height, the calculation of the viscosity based on the flow is simplified compared to a cross-section with a lower aspect ratio. Using the measured pressure and the flow rate, the apparent viscosity can be measured.

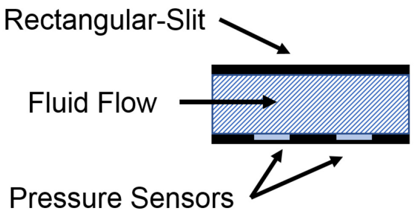


Figure 9. Diagram of rectangular-slit viscometer measurement method indicating the flow direction and placement of pressure sensors. The viscosity can be determined using the flow rate and pressure difference between the pressure sensors.

Examples of commercial instruments that use a rectangular-slit viscometer (and are used for this project to obtain reference viscosity measurements) are the m-Viscometer/Rheometer-on-a-Chip (m-VROC) and Micro-Viscometer μVISC viscometers (RheoSense, San Ramon, United States). Both devices use a syringe pump to flow the sample fluid through selectable sensors. Due to the placement of the components and having the pressure sensors located within the channel, it is difficult to visually inspect the chips. The visual inspection would allow for the measuring of the aggregation and CFL, which are both important phenomena that affect the viscosity. Having the pressure sensors located within the channel itself, however, means that the pressure along the channel can be measured directly, instead of it being inferred. Since the devices are made with high precision and sensitive components, each selectable sensor has a working range of flow rate or shear rate. Measurements at low shear rates give a low linear correlation coefficient, which means a higher uncertainty in the measurement; this is a problem, in particular, for measurements made at the shear rates found in the microcirculation. It is important, however, to measure at this range for non-Newtonian fluids such as blood, in order to investigate the rheology under physiological conditions.

1.4.Research Objectives

The objective of this thesis is to design a viscometer to measure blood viscosity under microcirculation conditions with the ability to correlate the viscosity with the physiological phenomena of aggregation and the CFL. The context of the research is to enable the investigation of more complicated microchannel networks than previously analyzed straight tubes, closer to the 3D vessel structures found in the human body. Future developments resulting from this research may support diagnostic comparisons between healthy and diseased blood flows.

1.4.1. Research Question

What aspects of current viscometers could be used or improved upon to help model the flow of blood in the microcirculation?

1.4.2. Gaps in Research and Existing Technologies

To our knowledge, no previous study has investigated fluid viscosity in a network while simultaneously imaging the flow. The ability to do this using Newtonian, Non-Newtonian as well as biological fluids would help contribute to the understanding of blood rheology in the microcirculation. The existing technology is not able to combine low shear capabilities along with the visualization of the flow and abilities to interchange complex microchannel networks.

1.4.3. Research Aims

The aim of this study is to investigate the blood viscosity at low shear rates in rectangular channels with dimensions comparable to smaller blood vessels (hydraulic diameter of 100 micrometres and below). The study will start with rectangular channels for better visualization of the aggregation and CFL effects. These channels are to be arranged in a network of microchannels, using fluidic features to allow the simultaneous measurement of viscosity and imaging the flow in the channels. Investigating the effects of microchannel dimensions and geometrical network arrangement in optically clear channels will allow the simultaneous investigation of human blood cell behaviour and viscosity.

1.4.4. Hypothesis

The hypothesis is that microchannel dimensions and arrangement influence the apparent viscosity of blood at low shear/flow rates.

1.4.5. Project Objectives

The specific scope of this project is to design an optically clear device that measures the viscosity of fluids while having fast pressure response and, at the same time, can be programmed to vary the applied shear rate. This device should be able to measure the viscosity of the blood as the cell deformation, aggregation, and CFL effects occur, and correlate the measured values with the aggregation size, CFL thickness and (eventually) the velocity profile determined from images of the blood flow that are made simultaneously.

A viscosity measurement platform will be designed to estimate the viscosity of blood at low shear rates, ideally below 100 s^{-1}, and record the blood flow. The viscosity at this scale is investigated by measuring pressure drop and blood flow rate using cutting edge microfluidics technology. The optimal chip material will facilitate investigations of the RBC phenomena and the measurement flow parameters. A material which is optically clear, enables fast fabrication and capabilities with gas exchange, would be optimal. The chip will be created with capabilities of being able to change the network of channels being analyzed.

Flow variables such as pressure drop or flow rate can be measured using sensors, whereas more complicated optical analysis is required to measure aggregation level, CFL and hematocrit partition. These requirements lead to the need for:

1) A customizable measurement platform

   Pressure and velocity measurement can be combined with optical observation, using a customizable viscosity measurement platform in which the range of pressures and flows is representative of physiological values.

2) Disposability of the microfluidic chip

   Since biological samples are used, it is desired that the chip be disposable for health and safety concerns, while also having the flexibility to introduce any configuration into the channel design.

3) Fast pressure response

   The ability to reach steady state flow quickly will allow short measurement times and prevent the RBCs from settling due to gravity.

4) Programmable flow/shear rate and ability to measure low flow rates

   Maintaining the flow/shear rate at set time intervals will allow the results to be correlated with the known effects of flow/shear rate on the RBC aggregation and CFL.

To these ends, we present the design of a biomicroviscometer, where the ability for optical access has been combined with high accuracy micro pressure and flow sensors in optically clear microfluidic channels. Due to the biological nature of blood, these devices are aimed to be for one-time use. The goal is to obtain accurate rheological measurements of complex fluids such as blood. Verification of the system's ability to estimate the apparent viscosity will be completed with DI water first. Validation of the system's ability to estimate blood viscosity will be tested by first using reference fluids with known viscosity behaviour that progressively reach blood's viscosity, non-Newtonian nature and RBC contents, before actually using blood itself.

2. Experimental Design

The experimental method is to measure the pressure drop, flow rate and temperature as the blood flows through microchannel features in an optically clear fluidic chip in order to estimate the blood viscosity at low shear rates. The blood is pressure driven, via compressed air, through a microchannel network, which is fabricated in a polydimethylsiloxane (PDMS) chip. The PDMS chip allows optical access to simultaneously observe and record the visual blood flow properties.

2.1.Rheology Measurement Method

2.1.1. Design Principle

The design of the viscosity measurement platform began with the placement of the channels and sensors. The flow sensor was placed before the chip, and after the sample vial. As shown in Figure 10, a general equivalent circuit was created as a basis for where the pressure sensors would be placed and the layout of the channels. The letters on the equivalent circuit represent the ports in the design, which can be used for microfluidic connections. Thermocouples were placed near the chip to measure the fluid temperature, with the potential for them to be placed in the fluid. The fluid then flows through the outlet of the chip, into a tube and out to a waste container.

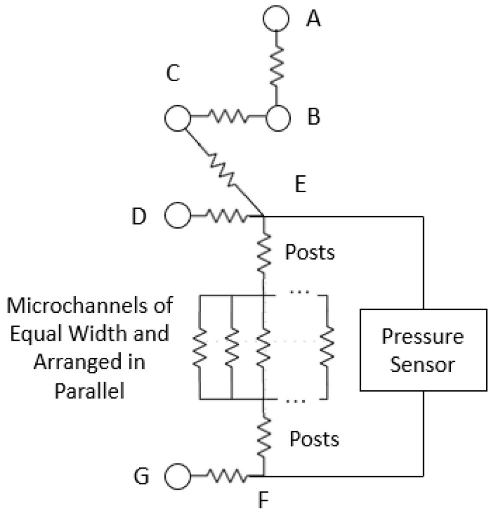


Figure 10. Equivalent hydraulic resistance circuit for the core of the microfluidic network design. The core of the design is located between ports E and F, which are used to measure the differential pressure.

The chip design consists of two tapered chambers connected together with parallel channels having dimensions comparable to those found in the microcirculation. Rectangular channels were chosen for the initial network instead of round ones to facilitate imaging and fabrication. When using imaging techniques with round channels, extra care is necessary to consider the shadowing effect of cells on top of each other [55], [56]. Techniques are available to create round channels, such as using metal wires or pressurized air, however these are more complicated to execute for the complex geometries that may be found in a modular network [57].

Due to the low flow rates required to investigate the blood properties, several channels of equal widths were chosen to be connected in parallel. The inclusion of several channels divides the total flow rate between them, which lowers the flow rate in the individual channels and causes the same pressure difference across each of the channels. These channels would be where the main visual inspection of the flow would occur. As shown in Figure 11, large tapering chambers (with a trapezoidal shape when viewed from the top) connect these channels together at their ends. In an effort to disaggregate the larger blood aggregates before entering the parallel channels, a grid of microposts were created in the large tapering channels. While being known for their sorting or capturing capabilities [58], [59], micropillars have also been used to investigate and improve mixing capabilities [60]. The rigidity of unbonded microposts has also previously been computed, which can be used to predict the post tips' ability to bend if not bonded to another surface [61]. The microposts would also provide stability to the channel so that the ceiling of the channel would not collapse under the large aspect ratio (being much wider than the depth).

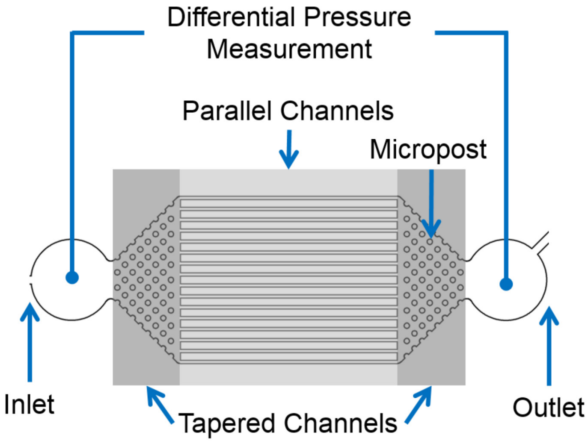


Figure 11. Depiction of main analysis sections, where the differential pressure is measured across, with large tapering channels and parallel channels.

2.2.Design of Experimental Setup

A network of channels was created based on the design specifications. The path of fluid would follow first through an inlet port, then to the section where the pressure would be measured, out of an outlet port to a flow sensor, and then to the waste. A high-speed camera could image the channels, and linking a microparticle image velocimetry (µPIV) setup would allow the tracking of the velocity profile.

A pressure driven system, instead of using a flow rate driven system, was chosen to minimize the pressure delay in the experiment. A small pressure delay was desired for our application since blood cells settle quite quickly over time and can affect the concentration of cells flowing through the system. If the arrangement of channels causes a different concentration of cells to flow through the system, the results will be for the incorrect concentration. The settling of RBCs is important to consider when planning and conducting experiments, since the experimental setup can cause the RBCs to start settling within that time frame. Settling can be most apparent as the fluid flows to a chip, both in the sample holding chamber and in the fluidic connections. It was found when experimenting with blood samples that the lowest volumetric flow rates used cause settling of RBCs that can be visually seen. This settling can be reduced by having short connections and a mechanism that continuously mixes the sample.

Our viscosity measurement platform design combines optical analysis with the known accuracy of micro scale pressure sensors. A pressure and temperature sensor module was previously made (by Dr. Katie Pitts) using D2 BASIC differential pressure transducers (5 INCH-S2-BASIC by All Sensors) and J2 thermocouples. Using this module, the differential pressure would be measured between ports E and F, as shown in Figure 10, while the temperature could be measured next to the channels. These sensors have low errors, which will allow for an overall reduction in error for measurements of biological samples in pressure driven flows. In addition to having a small error with sensitive sensors, our setup will use small and contained sample sizes, which is advantageous for biological samples such as blood.

The equivalent hydraulic resistance circuit is shown below in Figure 12 alongside the final chip design with the ports A through G labelled in each diagram. The overall dimensions of the PDMS portion of the chip is approximately 35 mm × 15 mm. The core of the design is located between ports E and F, comprising the area where the microposts and parallel channels reside. The low flow rates desired between ports E and F required that there be many parallel channels located in-between. Rigid Teflon tubing with inner diameter (I.D.) × outer diameter (O.D.) of 0.79375 mm × 1.5875 mm, and polyetheretherketone (PEEK) tubing (0.254 mm I.D. × 0.79375 mm O.D.) were used for all connections, except for pressure sensor connections, to increase the fast pressure response. Tygon tubing (2.38 mm O.D.) was used to connect the pressure sensor to the rigid tubing.

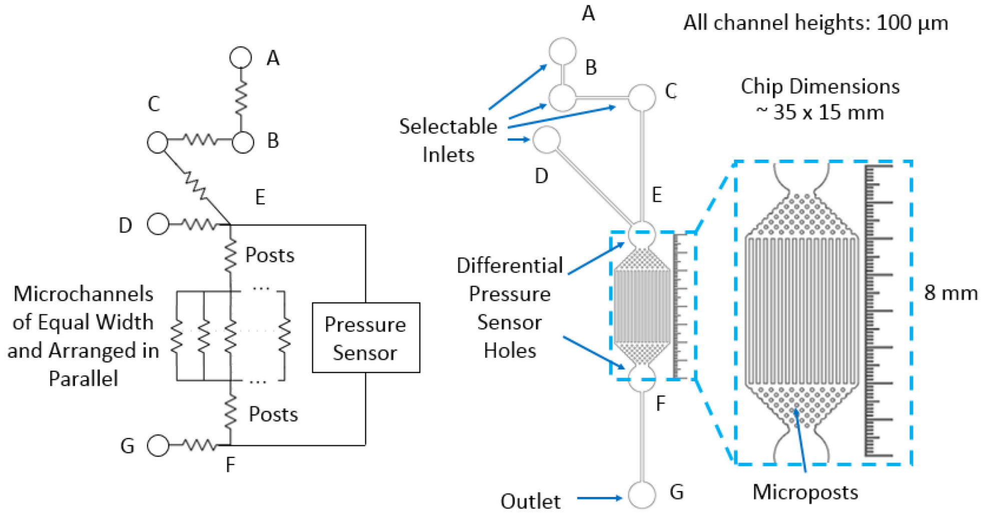

A
B
C
D
E
Posts
Microchannels of Equal Width and Arranged in Parallel
Pressure Sensor
Posts
G
F
All channel heights: 100 µm
Selectable Inlets
Chip Dimensions ~ 35 x 15 mm
Differential Pressure Sensor Holes
8 mm
Outlet
Microposts

Figure 12. Labelled ports of the equivalent microfluidic circuit of hydraulic resistances (R_{hyd}) (left) and top view of the final channel design (right).

Sufficient distances between the inlets, outlet and pressure sensor ports were needed to enable them to be placed in the chip. To ensure that the tubing connections had sufficient contact with the PDMS at the ports, and that the walls would not deform considerably (which could possibly cause leaks), a minimum of 4 mm was provided between key connections, with one exception. A shorter distance of 1 mm was allowed between ports A and B, since both would never be used as a fluidic connection at the same time.

Since the other inlets, that are not in use, are dead-end channels, the hydraulic resistance of the channels between the inlet and the first pressure sensor port can be calculated. The resulting difference in resistance between the different designs were calculated using hydraulic resistance formulas located in Design of Experimental Setup Section 2.2. Entrance ports were created so that there are different selectable inlets, with details of the calculation and values in Hydraulic

Resistances of Features Common between Designs and Addition of Inlet Channels for Parallelization Section 2.4.2. A series of inlets are located at the beginning of the chip and have different lengths of channels between them, as shown in Figure 13. Chips with the smaller width for the parallel channels have the higher resistance. The design of the inlet port started with attempting to match the two designs with the smallest resistance between E and F ($R_{hyd_{EF}}$), for the designs with the largest parallel channel widths of 100 and 60 µm. With each inlet port option, another section of channel is added which corresponds to the difference in resistance between it and the previous chip. Selecting an inlet location to punch along the channels allows for tuning of the inlet resistance. For example, the difference in $R_{hyd_{EF}}$ between the 100 µm and 60 µm is added as resistance between ports A and B. The resistance between ports A and G ($R_{hyd_{AG}}$) for the 100 µm design would then equal the resistance between ports B and G ($R_{hyd_{BG}}$) for the 60 µm design. The intention was to create chips that could be run in parallel with the same overall resistance. Running experiments with the same pressure drop across chips with the same resistance will result in similar flow rates through the analysed channels. A panel of tests can then be run on a sample, subjecting it to the same flow conditions, just in different width channels.

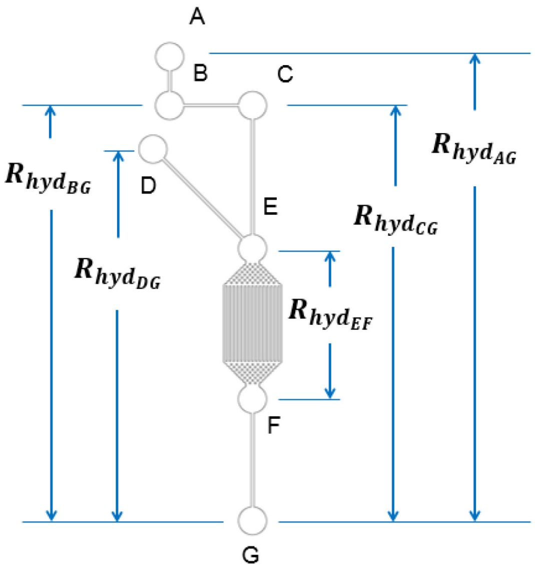


Figure 13. Extra ports and sections of channels were added to obtain the same resistance between designs with different channel widths. Additional sections of channels near the inlet would add hydraulic resistance for designs with lower hydraulic resistance between ports E and F ($R_{hyd_{EF}}$).

Compliance is the measure of a volume change of a system, compared to the initial volume, as a ratio to the pressure change induced. Compliance in the system was investigated since this fluidic system elasticity would cause the fluidic circuit to take time before it reached steady state. Rigid tubing was chosen for the main experimental connections over flexible tubing in order to reduce any compliance in the system. Minimal flexible tubing was used to connect the rigid tubing together. Flexible tubing was also located within the pressure sensor to route from the differential pressure sensors to outside of the pressure/temperature sensor casing. The air pocket that formed inside the flexible tubing was reduced by having several concentric tubing pieces, which were tight fitting and had a rigid tubing in the centre. These layers of tubing effectively reduced the volume of air created between the pressure sensors and the fluid being measured, which will be further explained in Determining Equilibrium and Reduction of Compliance Section 2.5.3. The channels were analysed initially assuming that the channels are rigid. The lengths of tubing between (1) the outlet hole and flow sensor; and (2) the flow sensor outlet and waste were also considered. These segments of tubing added to the overall chip resistance as the flow sensor and the tubing used to connect to it have very small I.D.'s of 125 µm and 397 µm, respectively.

In the initial design, the flow sensor was positioned downstream of the chip, but this was found to cause high pressures in the PDMS channels and tubing, owing to the high resistance (and therefore pressure drop) across the flow sensor. For the final design, the flow sensor was relocated upstream of the chip. This would allow any variances in the input pressure to be dampened before reaching the channels being measured at a lower pressure.

2.2.1. Working Equations

To derive the equation, relating the apparent fluid viscosity to the measured parameters (flow rate, pressure gradient and temperature), some important assumptions about the fluid flow must be made:

1) Fully developed flow
2) Steady flow
3) Laminar flow
4) The fluid is incompressible
5) The viscosity is independent of the pressure

Considering that the geometry of the proposed network is complex (i.e. not unidirectional, axisymmetric, etc.) and the blood properties will not be known, the relationship between the flow and pressure drop cannot be solved either mathematically or numerically. Therefore, only a global apparent viscosity (μ_{app}), instead of the local viscosity, can be determined based on these assumptions.

The flow and pressure within a fluidic network can be analysed similarly to an electric circuit with the resistance and voltage. In the overall network, the pressure drop *(*ΔP) across a fluidic feature is equal to the flow rate (Q) through it, multiplied by its hydraulic resistance (R_{hyd}); i.e. $\Delta P = Q \cdot R_{hyd}$.

The equivalent hydraulic resistance of the complex flow network depends on the viscosity of the fluid flowing through the channels and the geometry of the channels. The hydraulic resistance is a function of the fluid pressure drop and flow rate of the fluid, similar to how the resistance of electrical circuits is related to the voltage potential and current through them. The equivalent hydraulic resistance is computed by summing up the individual resistances in series or parallel in the same way as with electrical circuits, using Equations (1) and (2).

$$R_{eq_{hyd_{series}}} = R_{hyd_1} + R_{hyd_2} + \cdots + R_{hyd_n} \tag{1}$$

$$R_{eq_{hyd_{parallel}}}{}^{-1} = R_{hyd_1}{}^{-1} + R_{hyd_2}{}^{-1} + \cdots + R_{hyd_n}{}^{-1} \tag{2}$$

The shear rate of the fluid, $\dot{\gamma}$, within the channels can be calculated by using the equilibrium pressure gradient (ΔP) and the shear stress (τ) acting over a fluid element (e.g. a channel segment). For Newtonian fluids, the shear rate is related to the shear stress and fluid viscosity as $\dot{\gamma} = \tau/\mu$. In the following subsections, the flow-pressure relationships and the shear rate equations (given the corresponding hydraulic resistances) will be presented for the channel geometries that may be used in the design.

2.2.1.1. Round Channels

Tubing used for fluidic connections within the design will be considered as round channels with a linear pressure drop. The flow rate and shear rate of cylindrical tubing were calculated using Equations (3) and (4), respectively [62], [63].

$$Q = \Delta P \frac{\pi a^4}{8\mu L} \tag{3}$$

$$\dot{\gamma} = \frac{4Q}{\pi a^3} = \frac{a\Delta P}{2\mu L}, \tag{4}$$

where Q is the flow rate, $\dot{\gamma}$ is the shear rate, ΔP is the pressure drop across the channel, a is the radius of the channel, μ is the viscosity, and L is the length of the channel.

2.2.1.2. Rectangular Slit

The rectangular slit is not found within the design geometry, but it could be implemented in future channel designs. The flow rate and shear rate can be calculated for a rectangular slit using Equations (5) and (6), where the width of the channel (d_2) is much larger than the depth/height of the channel (d_1) [62], [63].

$$Q = \Delta P \frac{d_2 d_1^3}{12\mu L} \tag{5}$$

$$\dot{\gamma} = \frac{6Q}{d_2 d_1^2} = \frac{d_1 \Delta P}{2\mu L} \tag{6}$$

where Q is the flow rate, $\dot{\gamma}$ is the shear rate, ΔP is the pressure drop across the channel, d_1 is the length of the rectangular cross-section's shorter edge, d_2 is the length of the rectangular cross-section's longer edge, μ is the viscosity, and L is the length of the channel.

2.2.1.3. Rectangular Channels

The flow rate of a rectangular channel was calculated using the following Equations (7) and (8) [64], [65].

$$Q = \Delta P \frac{8d_1}{\mu d_2 L} \sum_{n=1}^{\infty} \frac{1}{\beta_n^4} \left(1 - \frac{2}{\beta_n d_1} \tanh\left(\beta_n \frac{d_1}{2}\right)\right) \tag{7}$$

where d_1 is the length of the rectangular cross-section's shorter edge, d_2 is the length of the rectangular cross-section's longer edge, μ is the viscosity, Q is the flow rate, and:

$$\beta_n = (2n-1)\left(\frac{\pi}{d_2}\right). \tag{8}$$

The wall shear rates for the parallel rectangular channels with Newtonian fluids ($\dot{\gamma_a}$) were estimated using Equations (9), (10) and (11), and were verified to be within the desired range in the analysed channels [66].

$$\dot{\gamma_a} = \left(\frac{6Q}{{d_1}^2 d_2}\right)\left(1 + \frac{d_1}{d_2}\right) f^*\left(\frac{d_1}{d_2}\right) \tag{9}$$

$$f^*(x) = \left[\left(1 + \frac{1}{x}\right)^2 \left(1 - \frac{192}{\pi^5 x}\sum_{i=1,3,5,\ldots}^{\infty} \tanh\frac{\left(\frac{\pi}{2} i x\right)}{i^5}\right)\right]^{-1}. \tag{10}$$

$$x = d_1/d_2 \tag{11}$$

In these equations, Q is the flow rate, d_1 is the length of the rectangular cross-section's shorter edge, d_2 is the length of the rectangular longer edge, and x is the aspect ratio constant [66].

The wall shear rate for a non-Newtonian fluid ($\dot{\gamma_w}$) with a viscosity that follows a power law was calculated using geometric constants for a rectangular duct cross-section and Equation (12) [67], [68]:

$$\dot{\gamma_w} = K_{power\ law}\left[\dot{\gamma_a}\left(\frac{2}{3}\right)\left(\frac{b^*}{f^*} + \frac{a^*}{f^*}\frac{1}{n_{power\ law}}\right)\right]^{n_{power\ law}} \tag{12}$$

where $K_{power\ law}$ is a non-Newtonian power law constant relating to viscosity, $n_{power\ law}$ is a non-Newtonian power law constant, and for this equation, a^* and b^* are equation constants that can be calculated using the following Equations (13) and (14) [64], [65]:

$$a^* = \frac{1}{2(1+x)^2\left[1 + 4\sum_0^{\infty}\left(\frac{(-1)^{n+1}}{\left(\frac{2n+1}{2}\pi\right)^3}\frac{1}{\cosh\left(\frac{(2n+1)\pi}{2x}\right)}\right)\right]} \tag{13}$$

where x is calculated using Equation (11).

$$a^* + b^* = \frac{3}{2(1+x)^2\left[1 - \frac{192}{\pi^5} x \sum_0^{\infty}\left(\frac{1}{(2n+1)^5}\tanh\left(\frac{(2n+1)\pi}{2x}\right)\right)\right]} \tag{14}$$

The apparent viscosity was estimated using the shear stress relations shown in Equation (15) and the wall shear rate with corrections calculated for a power law fluid.

$$\tau = \mu_{app}\dot{\gamma} = K_{power\ law}\dot{\gamma}^{n_{power\ law}} \tag{15}$$

Rearranging Equation (15) and isolating for the viscosity term gives Equation (16), which was used to estimate the viscosity of the fluid based on the wall shear rate:

$$\mu_{app} = K_{power\ law}\dot{\gamma}_w{}^{n_{power\ law}-1} \tag{16}$$

2.2.1.4. Flow Around Microposts

Darcy's law for flow through regularly ordered parallel fibres was used to calculate the resistance through the micropillars [64]:

$$Q = \Delta P \frac{KA}{\mu L} \tag{17}$$

where K is a permeability constant, A is the cross-sectional area of the flow region, μ is the viscosity, Q is the flow rate and L is the streamwise length [64]. The pillars were created with a quadratic arrangement and perpendicular fluid flow, as shown in Figure 14, which are modelled by Equations (18) to (24) [64]. Equations involve analyzing the singular case and are used with the overall dimensions of the actual section to calculate the resistance. The top-down area was considered for the volume fraction, as shown in Figure 14, for the calculation of the permeability constant for a quadratic arrangement and fluid flow which is perpendicular to the axes of the microposts $K_{\perp}$:

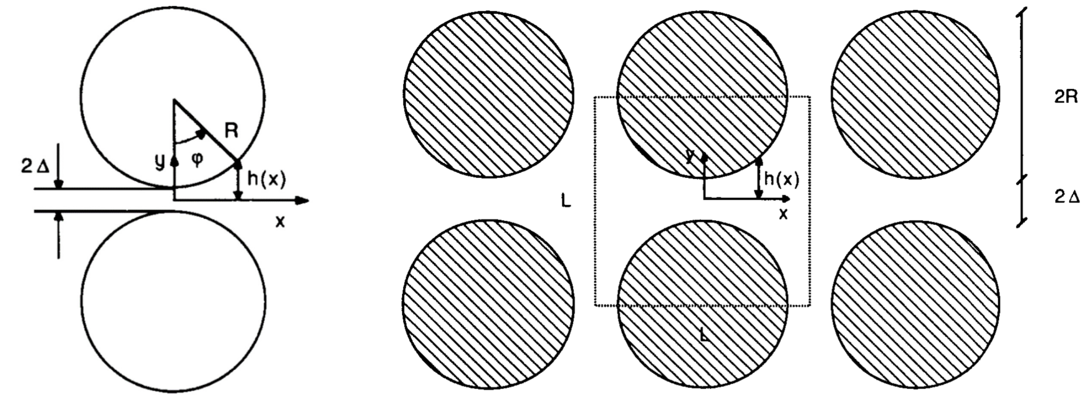


Figure 14. Diagram of dimensions for the fibre analysis (left), and area being considered for the volume fractions of a singular case (right), which are used to calculate the resistance for the whole tapered channel [64].

$$K_{\perp} = C_1 \left(\sqrt{\frac{V_{f\,max}}{V_f}} - 1 \right)^{\frac{5}{2}} R^2 \tag{18}$$

where C_1 for a quadratic arrangement is $\frac{16}{9\pi\sqrt{2}}$, V_f is the volume fraction between the pillars, R is the radius of a micropost, $2\varDelta$ is the distance between two adjacent microposts, and $V_{f\,max}$ is the maximum possible fibre volume fraction (i.e. if the cylinders were touching, $V_{f\,max} = \pi/4$) [64]. Combining Equations (17) and (18) together, the flow rate is given as:

$$Q = \Delta P \frac{\frac{16}{9\pi\sqrt{2}} \left(\sqrt{\frac{V_{f\,max}}{V_f}} - 1 \right)^{\frac{5}{2}} R^2 A}{\mu L} \tag{19}$$

To obtain the volume fractions and the cross-sectional area, the geometry of our chip design was considered. Rearranging Equation (19) to solve for the resistance from the form $Q = \Delta P / R_{hyd}$ gives the following relationship:

$$R_{hyd} = \frac{\mu L}{\frac{16}{9\pi\sqrt{2}} \left(\sqrt{\frac{V_{f\,max}}{V_f}} - 1 \right)^{\frac{5}{2}} R^2 A} \tag{20}$$

The following equation was used to calculate the fibre volume fraction:

$$V_f = 1 - \frac{A'}{A''} \tag{21}$$

where A' is the portion of the selected area which the fluid can flow through, and A'' is the total area comprising both the flow area and the fibre area [64]. For the selected region,
$A' = (2R + 2\varDelta)^2 - \pi R^2$ and $A'' = (2R + 2\varDelta)^2$. Using these areas and Equation (21),

$$V_f = 1 - \frac{A'}{A''} = 1 - \frac{(2R + 2\varDelta)^2 - \pi R^2}{(2R + 2\varDelta)^2} = \frac{(2R + 2\varDelta)^2 - (2R + 2\varDelta)^2 + \pi R^2}{(2R + 2\varDelta)^2}$$

$$V_f = \frac{\pi R^2}{(2R + 2\varDelta)^2} \tag{22}$$

For the tapered channel shown in Figure 15 (right), the cross-sectional area through which the fluid would flow (including the pillars) is

$$A = \frac{b_1 + b_2}{2}\mathrm{h} \tag{23}$$

where b_1 is the longer parallel side of the trapezoid, b_2 is the shorter parallel side, and h is the height of the chamber.

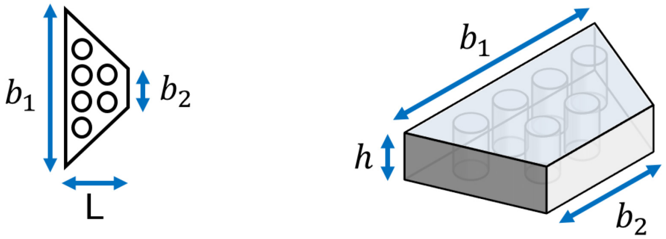


Figure 15. Top-down tapered channel shape, which contains the microposts, and dimensions (left), and the flow path considering the cross-sectional area to flow through (right).

Substituting Equations (22) and (23) into Equation (20), and simplifying:

$$R_{hyd} = \frac{\mu L}{\frac{16}{9\pi\sqrt{2}}\left(\sqrt{\frac{\pi/4}{\pi R^2/(2R + 2\Delta)^2}} - 1\right)^{\frac{5}{2}} R^2 \left(\frac{b_1 + b_2}{2}\right)\mathrm{h}}$$

$$= \frac{\mu L}{\frac{16}{9\pi\sqrt{2}}\left(\sqrt{\frac{1}{R^2/(R + \Delta)^2}} - 1\right)^{\frac{5}{2}} R^2 \left(\frac{b_1 + b_2}{2}\right)\mathrm{h}}$$

$$= \frac{\mu L 9\pi\sqrt{2}}{16\left(\sqrt{\frac{(R + \Delta)^2}{R^2}} - 1\right)^{\frac{5}{2}} R^2 \left(\frac{b_1 + b_2}{2}\right)\mathrm{h}} +$$

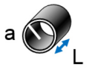
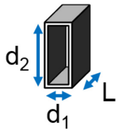
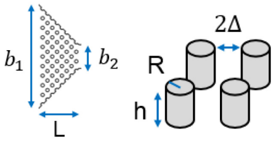
$$= \frac{9\mu L\pi\sqrt{2}}{8\left(\frac{(R + \Delta)}{R} - 1\right)^{\frac{5}{2}} R^2 (b_1 + b_2)\mathrm{h}} = \frac{9\mu L\pi\sqrt{2}}{8\left(\frac{\Delta}{R}\right)^{\frac{5}{2}} R^2 (b_1 + b_2)\mathrm{h}} = \frac{9\mu L\pi\sqrt{2}}{8R^2\mathrm{h}\left(\frac{\Delta}{R}\right)^{\frac{5}{2}} (b_1 + b_2)}$$

$$R_{hyd} = \frac{9\mu L\pi\sqrt{2}}{8R^2\mathrm{h}\left(\frac{\Delta}{R}\right)^{\frac{5}{2}} (b_1 + b_2)} \tag{24}$$

2.2.2. Viscometer Principle and Operating Range

In each section of the microfluidic circuit, the pressure drop (ΔP) across a fluidic feature is equal to the flow rate (Q) through it multiplied by its hydraulic resistance (R_{hyd}). Since R_{hyd} is proportional to the viscosity of the fluid, R_{hyd} can be represented by the viscosity multiplied by a hydraulic resistance constant (X_{hyd}) as $\mu_{app} \cdot X_{hyd}$. MATLAB was used to calculate these hydraulic resistant constants, based on Equations (3), (7) and (24). For the X_{hyd} calculations of Newtonian flows in the channel features, Table 2 was used as an accurate relation between viscosity and the experimentally measured quantities (Q, ΔP). This constant X_{hyd} was not used to calculate the apparent viscosity of non-Newtonian fluids; we applied the correction for the wall shear rate as presented in Equations (12) to (16).

Table 2. Channel features, which are found in the design and their respective hydraulic resistant constants.

Shape		**Hydraulic Resistance Constant (X_{hyd})** [62], [64], [65]
Cylindrical Channel		$\frac{8}{\pi}L\frac{1}{a^4}$
Rectangular Channel		$\frac{Ld_2/8d_1}{\sum_{n=1}^{\infty}\frac{1}{\beta_n^4}\left(1-\frac{2}{\beta_n d_1}\tanh\left(\beta_n\frac{d_1}{2}\right)\right)}$ $\beta_n = (2n-1)\left(\frac{\pi}{d_2}\right)$
Micropposts in Quadratic Arrangement		$\frac{9L\pi\sqrt{2}}{8R^2\mathrm{h}\left(\frac{\Delta}{R}\right)^{\frac{5}{2}}(b_1+b_2)}$

Since $\Delta P = Q \cdot R_{hyd}$, where $R_{hyd} = \mu_{app} \cdot X_{hyd}$, the equivalent resistance of the network of channels could be calculated from measurements of Q and ΔP.

$$\Delta P = Q \cdot R_{hyd} = Q \cdot \mu_{app} \cdot X_{hyd} \tag{25}$$

Rearranging the equation to solve for μ_{app} gives the following equation:

$$\mu_{app} = \frac{\Delta P}{Q \cdot X_{hyd}} \tag{26}$$

Once the apparent viscosity is determined for the fluid, it will be associated with the shear rate calculated in the network's channels by using the flow rate and channel dimensions. The apparent viscosity and shear rate can then be plotted to investigate how the apparent viscosity is affected when changing the shear rate, which is of particular importance for non-Newtonian fluids, such as blood.

The operating range of the designed channel configuration to measure apparent viscosity will be estimated based on the dimensions and geometry of the chip (i.e. the hydraulic resistance coefficient), and the minimum and maximum pressure drops that the pressure sensor can measure, at various shear rates. This range gives an indication towards the capabilities of the viscometer to measure certain viscosities at various shear rates. If the intended viscosity or shear rate lies within the range, it is possible for the device to measure the viscosity. As stated previously, the shear stress (τ) of a Newtonian fluid is equal to the shear rate ($\dot{\gamma}$) multiplied by the viscosity (μ):

$$\tau = \dot{\gamma}\mu. \tag{27}$$

Considering a fluid in equilibrium, the force due to the driving pressure must be equal to the opposing force of the shear stress in a channel [69]. For a rectangular channel, the shear stress and pressure are related to the geometry as in the following equation, where ΔP is the pressure drop, *w* is the channel width, *h* is the channel height, and *L* is the channel length.

$$\Delta P(wh) = \tau(2wL + 2hL)$$

$$\tau_{rec} = \frac{\Delta P}{2L}\frac{(wh)}{(w+h)} \tag{28}$$

For a slit geometry, $w \;>>\; h$ and Equation (28) becomes:

$$\tau_{slit} = \frac{\Delta P}{2L}h \tag{29}$$

Substituting Equation (28) or (29) into Equation (27) and isolating for the viscosity gives the following equations, which can be used to estimate the operating range of a chip with a rectangular or rectangular slit channel. The operating range of the proposed design was chosen so that the viscosity of blood could be measured at low shear rates. This was completed by using the pressure drop range of the pressure sensor and changing the geometry so that a desired shear rate was obtained within them.

Rectangular Slit:
$$\mu = \frac{1}{\dot{\gamma}}\frac{\Delta P}{2L}h \tag{30}$$

Rectangular Channel:
$$\mu = \frac{1}{\dot{\gamma}}\frac{\Delta P}{2L}\frac{(wh)}{(w+h)} \tag{31}$$

For example, using the device's specifications, as tabulated in Table 3, the operating range for the m-VROC viscometer (RheoSense, San Ramon, United States) with an A05 chip can be computed using Equation (30) for the rectangular slit.

Table 3. Specifications used to estimate the operating range for m-VROC viscometer chips (RheoSense, San Ramon, United States) [70], [71].

Specification	Value for A05	Value for B05
Minimum Pressure Drop (1% full scale) (Pa)	100	400
Maximum Pressure Drop (Pa)	10000	40000
Length (mm)	6.2	6.2
Depth (mm)	0.05	0.05
Width (mm)	3.02	3.02

Plotting the operating range for these chips along with the predicted viscous behaviour of the intended testing fluid, as shown in Figure 16 and Figure 17, depicts the range of shear rates at which the device is able to measure desired viscosity. As can be seen when comparing Figure 16 (Chip A05) to Figure 17 (Chip B05), because the sensor Chip B05 has a higher pressure limit, it can measure higher viscosities. The viscosity of blood with 45% hematocrit, according to Chien [24], was also plotted on the same graph to compare to the operating range of these devices. It can be seen that the Chip A05 has the capability to measure this viscosity of blood above 100 s^{-1}, whereas Chip B05 has the capability to measure this viscosity of blood above 500 s^{-1}.

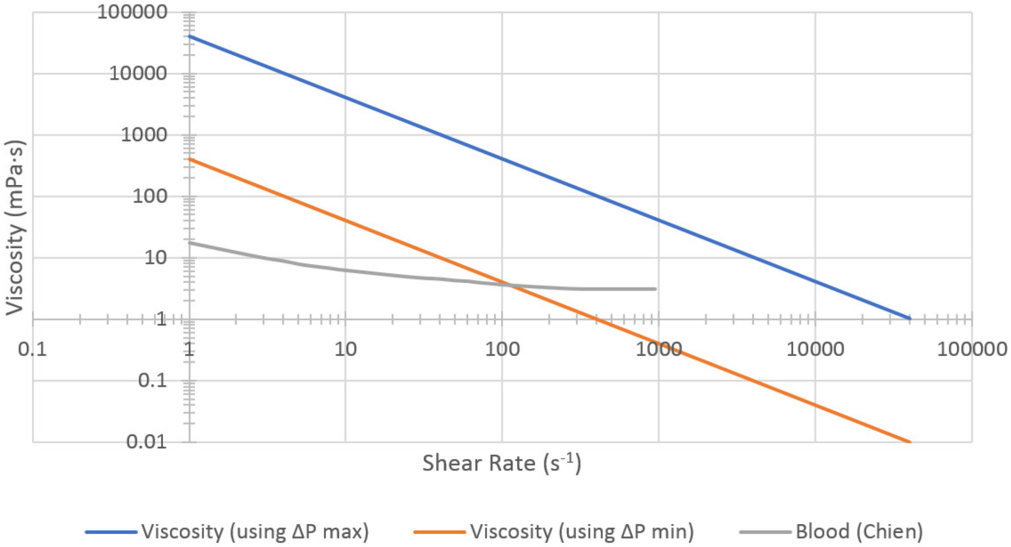


Figure 16. Operating range of m-VROC viscometer using A05 chip (RheoSense, San Ramon, United States) using the max differential pressure (blue), min differential pressure (orange) in comparison to blood viscosity at 45% hematocrit (grey) as by Chien [24].

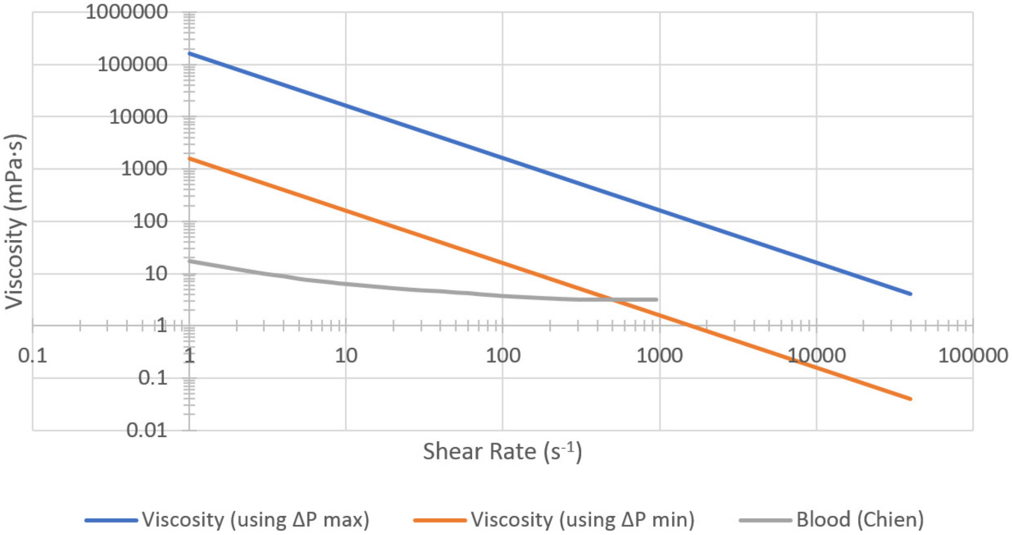


Figure 17. Operating range of m-VROC viscometer using B05 chip (RheoSense, San Ramon, United States) using the max differential pressure (blue), min differential pressure (orange) in comparison to blood viscosity at 45% hematocrit (grey) as by Chien [24].

2.3.Design Specifications

Both physiological and instrumental specifications were taken into consideration during the design of the device. Vessels in the microcirculation range in diameter from 300 μm (microscopic arterioles) down to 5 μm (capillaries) [1]. A width range of 20 to 100 μm was chosen for the channel dimensions, in order to model the smaller vessels in the microcirculation range. In vessels of this scale, the physiological flow rate ranges from 37.7 nL/min to 5.89 μL/min and the flow rate in the device channel was aimed to be within this range [72]. As mentioned previously, blood viscosity depends on a number of factors, such as the hematocrit and shear rate, which vary between people and are affected by pathologies [73]. Normal adult human blood viscosity at body temperature ranges between 3-5 mPa·s for shear rates greater than 100 s^{-1}; the indication of hyperviscosity in neonates is a measurement of >14.6 mPa·s at a shear rate of 11.5 s^{-1} [74]. To enable measurements using water (viscosity of 1 mPa·s at 20 °C), and also accommodate potential differences in blood viscosity between patients, the design range of viscosity was chosen to be 1 mPa·s to 20 mPa·s. Hematocrit in the microcirculation vessels smaller than 100 μm in diameter decreases from the normal range of 40-50% of larger vessels, down to 10% found in the capillaries [63]. Thus, a target microcirculation range for the hematocrit was chosen to be between 10% and 35% for the platform.

Specifications for equipment include the Flow-Rate Platform (FRP) Unit S flow sensor (Fluigent, Villejuif, France), with a range of 17.5 nL/min to 7 μL/min [75]. The microelectromechanical systems (MEMS) pressure sensors were chosen based on the design pressure range, sensitivity, and physical footprint. 5 INCH-D2-BASIC pressure transducers (AllSensors, Morgan Hill, United States) were chosen for low pressure measurements, with a differential range of 1.25 to 1245 Pa [76]. The MultiFluidic Control System-EZ (MFCS-EZ) pressure controller (Fluigent, Villejuif, France), which drives the fluid, uses filtered compressed air and has an output range from 210 Pa to 700 kPa [77]. Table 4 summarizes these specifications.

Table 4. Design specifications for biomicroviscometer.

Specification Type	Specification	Range		
Physiological	Hydraulic Diameter	20 μm	-	100 μm
	Blood Flow Rate	37.7 nL/min	-	5.89 μL/min
	Viscosity	1 mPa·s	-	20 mPa·s
	Hematocrit	10%	-	35%
Equipment	Flow Sensor	17.5 nL/min	-	7 μL/min
	Pressure Sensor	1.25 Pa	-	1245 Pa
	Pressure Controller	210 Pa	-	700 kPa

In addition to these fluidic design specifications, the design of the chip was to be optically clear and biocompatible. Polymers that are commonly used in microfluidics, such as PDMS or polymethylmethacrylate (PMMA), were considered for the creation of the microfluidic chip. PDMS was chosen as the substrate to bond to a glass slide for its optical clarity, non-toxic and fast fabrication properties [78]. PDMS is flexible and can be treated to be made hydrophilic (whereas it is naturally hydrophobic [79]). PDMS is a cost-effective material, which is important when using biological samples, since the chips can be disposed of after use of these samples. Soft lithography with PDMS was chosen as the fabrication technique with SU-8 as the photopolymer to create microfeatures on a silicon wafer.

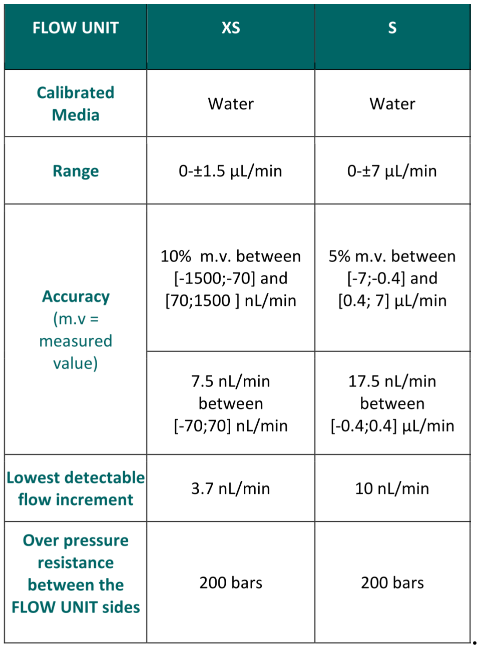
Using chips with these materials allows for visual inspection of RBC aggregation or deformation and CFL effects. A range of channel widths can be examined through image processing techniques to help investigate their influence on the CFL.The apparent viscosity will be evaluated by measuring pressure drop and flow rate through a series of microfluidic channels, using the previously mentioned instrumentation. The effect of temperature, pressure, shear/flow rate, hematocrit, and channel dimensions will be investigated. The ability to create different networks of channel geometries allows for a modular nature for these chips. Computer controlled timing capabilities of the system will allow precise control of the pressure and flow rate enabling the study of viscosity hysteresis in the channels and the behaviour in microcirculation networks.

2.4.Design Proposal

2.4.1. Core of the Design

An Excel spreadsheet was first created to estimate the various lengths required for each segment of the design so as to result in certain overall resistances to ensure the range is within the capabilities of the pressure sensor, flow sensor and input pressure. Values in the spreadsheet were varied to ensure that the viscosity range and dimensions of the channels would yield measurable pressures with the necessary input pressures. A compromise between the specifications was to create multiple channels, to allow the pressure drop and flow rate in the individual channels to be relatively low, while operating in the relatively higher range of the pressure controller (which drove the higher magnitude total flow).

Two different flow sensors, with specifications shown in Figure 18, were initially considered in the design of the experiment: the XS Flow Unit and S Flow Unit (Fluigent, Villejuif, France) models of the FRP flow sensor were used in the calculations to estimate the flow rates, pressures necessary, and the feasibility of experiments. The flow rates measurable by the XS sensor are lower than the range of the S sensor, which would allow finer measurements. It was found during early testing, though, that the XS sensor was more susceptible to clogs due to the smaller I.D. of its sensor tube, and thus sensor S was chosen for the design. To compensate for the larger flow rate range, multiple parallel channels were chosen, with the intent to run analysis on one of these channels (or the differences between several of them).



FLOW UNIT	XS	S
Calibrated Media	Water	Water
Range	0-±1.5 µL/min	0-±7 µL/min
Accuracy (m.v = measured value)	10% m.v. between [-1500;-70] and [70;1500] nL/min	5% m.v. between [-7;-0.4] and [0.4; 7] µL/min
	7.5 nL/min between [-70;70] nL/min	17.5 nL/min between [-0.4;0.4] µL/min
Lowest detectable flow increment	3.7 nL/min	10 nL/min
Over pressure resistance between the FLOW UNIT sides	200 bars	200 bars

Figure 18. Fluigent flow sensor specifications for flow unit sensors, S and XS, which were tested for use in our design [75].

The final design, as previously shown in Figure 12, features parallel channels 100 µm in height, having widths of either 20, 40, 60, or 100 µm. Multiple inlet ports A to D, as indicated in Figure 12, were included for the option of having the same overall hydraulic resistance with two designs of different parallel channel widths. A chip design with a wider parallel channel width would require an additional resistance source to match the overall resistance of another chip (with a narrower set of parallel channels). Extra lengths of channels were added before the core of the design to provide these additional resistance sources. The aim was to allow for simultaneous experiments between two or more chips with each of them having the same overall resistance, as shown highlighted in Table 11.

The hydraulic resistances across the different sections of the chip geometries were calculated using Table 2 for all the designs, with a set of example values shown in Table 5 to Table 14. The internal volume of each component was also calculated to estimate the fluid sample priming volumes. The lengths of the external tubing were overestimated to ensure that the model's measurement was still possible, without exceeding the pressure drop or flow rate range. These example values were calculated for a setup using the S Flow Unit with DI water, channel heights of 100 µm and all four design widths, for the associated flow parameters shown in Table 5.

Table 5. Variables of viscosity, flow rate, channel height and density, which were modified to ensure the design met the design specifications, with the example of DI water.

Variables	Estimated Value
Viscosity (mPa·s)	1
Flow Rate (μL/min)	5
Flow Rate (m^3/s)	8.33×10^{-11}
Channel Height (μm)	100
Density (kg/m^3)	1

2.4.1.1. Hydraulic Resistance of the Parallel Channels and Tapered Chambers

The analysis was not made in the order of the flow sequence, but addressed first the parallel channel section to ensure they would produce the appropriate pressure drop and the desired flow rates in the ranges that the pressure and flow sensors could measure. Example dimensions and variables used for the calculation are shown in Table 6, indicating the total hydraulic resistance constant, dimensions and internal volume of all the channels. There are 16 parallel channels for the designs with widths of 100 μm, 60 μm, and 40 μm, whereas there are 32 channels for the 20 μm width design. This was to ensure that the flow rate within each parallel channel was below 0.4375 μL/min for the 100 μm, 60 μm and 40 μm wide channels, and below 0.21875 μL/min for the 20 μm wide channels. The shear rate was estimated to be less than 43.75 s^{-1}, 72.92 s^{-1}, 109.375 s^{-1}, and 109.375 s^{-1} for the channel widths 100 μm, 60 μm, 40 μm, and 20 μm, respectively. The lengths of the parallel channels for the designs with 100 μm, 60 μm, and 40 μm, was 4000 μm. A different length was assigned to the 20 μm design, of 1200 μm, so that the pressure drop across the channels would be within the pressure sensor's measurement range. The shear rate and Reynolds number were also estimated for a single parallel channel for each of the channel designs, as shown in Table 6. The shear rate was calculated using Equations (9) and (10) and checked to be within the lower operating range of the chip and the Reynolds number was verified to indicate laminar fluid flow.

Table 6. Hydraulic resistance constants and total internal volume for each parallel channel section, which were calculated using equations from Rheology Measurement Method Section 2.1 and the volume of standard objects. Highlighted values have been calculated for the case of DI water and input variables as shown in Table 5.

Parameter	**Channel Design**			
Ideal Width (µm)	100	60	40	20
Ideal Height (µm)	100	100	100	100
Length of Each Parallel Channel (µm)	4000	4000	4000	1200
Individual Channel Hydraulic Resistance Constant (1/m^3)	1.14×10^{15}	3.57×10^{15}	1.00×10^{16}	2.06×10^{16}
Number of Parallel Channels	16	16	16	32
Equivalent Resistance Constant of All Parallel Channels (1/m^3)	7.10×10^{13}	2.23×10^{14}	6.27×10^{14}	6.44×10^{14}
Total Internal Priming Volume of Parallel Channels (µL)	0.64	0.384	0.256	0.0768
Individual Hydraulic Resistance (Pa·s/m^3)	1.14×10^{12}	3.57×10^{12}	1.00×10^{13}	2.06×10^{13}
Individual Channel Flow Rate (µL/min)	0.3125	0.3125	0.3125	0.15625
Total Pressure Drop of Parallel Channels (Pa)	5.92×10^{0}	1.86×10^{1}	5.22×10^{1}	5.36×10^{1}
Shear Rate (s^{-1})	31.25	52.08	78.125	78.125
Reynolds Number	5.21×10^{-8}	6.51×10^{-8}	7.44×10^{-8}	4.34×10^{-8}

The resistances of the tapered chambers with pillars were then calculated since, with the exception of the 20 µm design, the tapered channels were of the same dimensions. The distance between the pillars, the pillar diameter, and the pillar height are all 100 µm. The microposts were chosen to have an aspect ratio of 1:1 (diameter:height) with dimensions of 100 µm :100 µm. The aspect ratio of 1 meets our design target of being between 0.2 and 2 in order to prevent bending of the pillars or a collapse of the channel when being fabricated [61], [80]. In addition to the manufacturing requirements, the hydraulic resistance caused by the microposts should be small enough so that measurements in the core of the design can be made within the equipment specifications. The hydraulic resistance caused by the microposts should also be relatively small compared to the resistance in the parallel channels, to allow the calculation of the apparent viscosity without being on the scale of the equipment error. The hydraulic resistance and internal volume of these sections were calculated using the values shown in Table 7 and Equation (24).

Table 7. Dimensions of tapered channels used in the calculations for micropillar analyses for channel widths 20, 40, 60, and 100 µm. Variable symbols are described in Flow Around Microposts Section 2.2.1.4, with the dimension symbols shown in Figure 14 and Figure 15. Highlighted values have been calculated for the case of DI water and input variables as shown in Table 5.

Parameter	Symbol	20 µm Parallel Channel Width	40, 60, or 100 µm Parallel Channel Widths
Width of Entrance (μm)	b_1	500	
Width of Exit (μm)	b_2	3740	3100
Height of Channel (μm)	h	100	
Radius of Micropillar (μm)	R	50	
Half Distance between Fibres (μm)	Δ	50	
Length of Tapered Channel (μm)	L	1591	1270
Maximum Volume Fraction of Flow Area to Total Area	$V_{f_{max}}$	π/4	
Volume Fraction of Flow Area to Total Area	V_f	0.0625π	
Hydraulic Resistance Constant ($1/m^3$)	X_{hyd}	7.50×10^{12}	7.06×10^{12}
Internal Volume (μL)	V_{int}	2.71×10^{-1}	1.84×10^{-1}
Hydraulic Resistance ($Pa \cdot s/m^3$)	R_{hyd}	7.50×10^{9}	7.06×10^{9}

2.4.2. Hydraulic Resistances of Features Common between Designs and Addition of Inlet Channels for Parallelization

The hydraulic resistance constants and internal volumes were then calculated for the rest of the fluidic connections going to, and coming out of, the microfluidic chip. Since these components were the same for all the channel designs, these values were the same for all designs. The first component of the chip to have a significant pressure drop would be in the tubing located between the sample vial and the flow sensor. To go from the tubing in the sample vial to the flow sensor, different diameter tubing was required to be connected, of estimated lengths being 10 and 25 mm, for the larger and smaller diameter tubing, respectively. The length of the tubing within the flow sensor was estimated to be 30 mm, using the dimensions of the flow sensor. More tubing was then required to connect the flow sensor to the selected inlet for the microfluidic chip, also with estimated lengths being 10 mm and 25 mm. These first connections were all using cylindrical channels, with the values used and estimates of the hydraulic resistance and internal volume estimated in Table 8.

Table 8. Example estimates of hydraulic resistance and internal volume of tubing to the flow sensor and from the flow sensor to the microfluidic chip. Highlighted values have been calculated for the case of DI water and input variables as shown in Table 5.

Fluidic Component		Radius (μm)	Length (μm)	Hydraulic Resistance Constant (1/m^3)	Internal Volume (μL)	Hydraulic Resistance (Pa·s/m^3)
Tubing from Outlet to Flow Sensor	Tubing 1	793.75	10000	1.03×10^{12}	4.95	1.03×10^{9}
	Tubing 2	396.875	25000	4.11×10^{13}	3.09	4.11×10^{10}
S Flow Sensor		150	30000	2.41×10^{15}	1.5	2.41×10^{12}
Tubing from Flow Sensor to Selected Inlet Port	Tubing 1	396.875	25000	4.11×10^{13}	3.09	4.11×10^{10}
	Tubing 2	793.75	10000	1.03×10^{12}	4.95	1.03×10^{9}

The hydraulic resistance between the inlet port and the first pressure sensor port (E) depends on which port was chosen, and the hydraulic resistance depended on the difference in resistance between designs of different parallel channel widths, as previously shown in Table 6. The widths of these channels were chosen to be the same, since the resulting lengths did not space ports too far or too close together. The chips with smaller widths of parallel channels have a higher resistance than those with wider widths.

To compensate for the difference in the resistances, additional lengths of channel were chosen depending on the differences between the hydraulic resistances of different designs. The differences in hydraulic resistance between ports E and F are shown in Table 9. As an example, to account for the difference in hydraulic resistance between a 40 and 60 μm width design, the channel between ports C and B required a hydraulic resistance of $4.03 \times 10^{11}\ Pa \cdot s/m^3$. This value is used to calculate the length of channel required between the ports as 2935 μm.

Table 9. Example values of the difference in hydraulic resistance between the channel designs. Highlighted values have been calculated for the case of DI water and input variables as shown in Table 5.

Designs Which are Compared	Difference in Hydraulic Resistance Constant (1/m^3)	Difference in Hydraulic Resistance (Pa·s/m^3)	Width (μm)	Height (μm)	Length Required (μm)	Length Selected (μm)
40 μm and 20 μm	1.78×10^{13}	1.78×10^{10}	150	100	129.11	129
60 μm and 40 μm	4.03×10^{14}	4.03×10^{11}			2924.48	2925
100 μm and 60 μm	1.52×10^{14}	1.52×10^{11}			1103.83	1104

From the hydraulic resistance confirmation analysis, summarized in Table 10, it was found that the length of channel between ports C and D would need to be 129 μm. The ports would be too close together and it would be difficult to manipulate fluidic connections with this small amount of space. It was chosen that port D should have its own connection to port E, with port C connecting to E with the length of D plus the necessary difference. The other two lengths were long enough to feasibly work with during experiments.

Table 10. Example of resistance differences between different selectable inlet ports.

Inlet to First Pressure Sensor Hole	1st Resistance Path (D)	2nd Resistance Path (C)	3rd Resistance Path (B)	4th Resistance Path (A)
Width (µm)	150			
Depth (µm)	100			
Length (µm)	6000	129	2925	1104
Hydraulic Resistance Constant (1/m^3)	8.28×10^{14}	1.78×10^{13}	4.03×10^{14}	1.52×10^{14}
Internal Volume (µL)	9.00×10^{-2}	1.94×10^{-3}	4.39×10^{-2}	1.65×10^{-2}

The effect of having these resistances for these ports can be further seen in Table 11, which shows the equivalent total hydraulic resistance constant when selecting certain ports (e.g. the resistance constant between ports B and G represented as X_{BG}), as shown in Figure 19. The highlighted values show that for each design, a different section can be chosen to obtain the same hydraulic resistance. This similarity would allow chips having different channel widths to be run in parallel; i.e. if there is a single pressure/flow rate source, then multiple chip designs can be used simultaneously and have the same pressure/flow rate going through the whole chip.

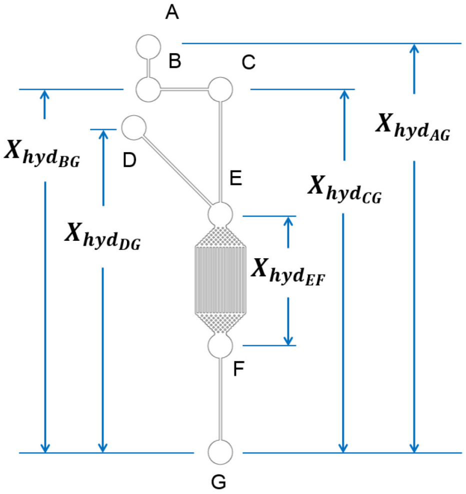
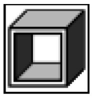
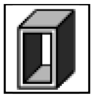
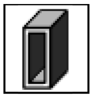


Figure 19. Design's modularity in choosing ports with the aim to enable chips to have the same fluidic resistance. Segments were labelled for the hydraulic resistance constants to determine required resistances in each segment for parallelization of chip experiments.

Table 11. Calculated values of equivalent hydraulic resistance constants between certain ports for chips with different parallel channel widths.

Shape				
Length (µm)	4000	4000	4000	1200
Height (µm)	100	100	100	100
Width (µm)	100	60	40	20
$\boldsymbol{X_{hyd_{AG}}\left(\frac{1}{m^3}\times 10^{15}\right)}$	43.5	46.6	54.6	55
$\boldsymbol{X_{hyd_{BG}}\left(\frac{1}{m^3}\times 10^{15}\right)}$	40.5	43.5	51.6	51.9
$\boldsymbol{X_{hyd_{CG}}\left(\frac{1}{m^3}\times 10^{15}\right)}$	32.4	35.5	43.5	43.9
$\boldsymbol{X_{hyd_{DG}}\left(\frac{1}{m^3}\times 10^{15}\right)}$	32.1	35.1	43.2	43.5
$\boldsymbol{X_{hyd_{EF}}\left(\frac{1}{m^3}\times 10^{15}\right)}$	1.7	4.75	12.8	13.2

The second tapered channel in the flow is a mirror image of the first one and has the same hydraulic resistance. The last microfluidic channel, located between ports F and G as shown in Figure 19, has a rectangular cross-section with the hydraulic resistance and internal volume shown in Table 12.

Table 12. Example values of hydraulic resistance and internal volume for tubing from the second pressure sensor port, measured at F, to the outlet at G. Highlighted values have been calculated for the case of DI water and input variables as shown in Table 5.

	Width (µm)	**Depth (µm)**	**Length (µm)**	**Hydraulic Resistance Constant (1/m³)**	**Internal Volume (µm)**	**Hydraulic Resistance (Pa·s/m³)**
Second Pressure Sensor Port to Outlet	150	100	5000	6.90×10^{14}	0.075	6.90×10^{11}

The last fluidic connection was tubing from the outlet to the waste vial, with the tubing submerged under the surface of the waste fluid. A waste vial collected the sample after it had flowed through the whole chip and its contents were disposed of after the experiment. This waste vial, which was different from the sample vial, was angled lower than 45 degrees from the vertical position to ensure that the fluid surface area was maximized to minimize the effects of the waste volume increasing over the duration of an experiment. An increase of the volume of the waste vial, and thus the height of the fluid within the vial, could cause there to be an increase in a counter pressure against the compressed air, causing the flow rate to decrease from the desired value. Table 13 outlines the values used to calculate the resistance and internal volume, assuming that there would be 15 cm of tubing connection between the outlet port G and the waste vial.

Table 13. Example values of hydraulic resistance and internal volume for tubing from the sample to the selected inlet. Highlighted values have been calculated for the case of DI water and input variables as shown in Table 5.

	Diameter (μm)	**Length (μm)**	**Hydraulic Resistance Constant (1/m^3)**	**Internal Volume (μL)**	**Hydraulic Resistance (Pa·s/m^3)**
Tubing from Outlet Port to Waste Container	793.75	150000	9.62×10^{11}	7.42×10^{1}	9.62×10^{8}

For this example of microfluidic chip design, Table 14 summarizes the hydraulic resistance, internal volume and estimated pressure drop for each component or fluidic feature.

Table 14. Example summary of hydraulic resistances, pressure drops, and internal volumes for the fluidic circuit from Table 6, Table 7, Table 8, Table 12, and Table 13. Highlighted values have been calculated for the case of DI water and input variables as shown in Table 5.

Resistance Pathway	**Hydraulic Resistance Constant with S Sensor ($1/m^3$)**	**Hydraulic Resistance with S Sensor ($Pa \cdot s/m^3$)**	**Internal Volume with S Flow Sensor (μL)**	**Pressure Drop with S Flow Sensor (Pa)**
Tubing from Sample to Flow Sensor	4.21×10^{13}	4.21×10^{10}	8.04×10^{0}	3.51×10^{0}
S Flow Sensor	2.41×10^{15}	2.41×10^{12}	1.50×10^{0}	2.01×10^{2}
Tubing from Flow Sensor to Selected Inlet	4.21×10^{13}	4.21×10^{10}	8.04×10^{0}	3.51×10^{0}
Inlet to First Pressure Sensor Hole	8.28×10^{14}	8.28×10^{11}	1.52×10^{-1}	6.90×10^{1}
First Pressure Sensor Hole to Parallel Channels (Tapered Channel)	7.06×10^{12}	7.06×10^{9}	1.84×10^{-1}	5.88×10^{-1}
All the Parallel Channels	6.27×10^{14}	6.27×10^{11}	3.84×10^{-1}	5.22×10^{1}
Parallel Channels to Second Pressure Sensor Hole (Tapered Channel)	7.06×10^{12}	7.06×10^{9}	1.84×10^{-1}	5.88×10^{-1}
Second Pressure Sensor Hole to Outlet	6.90×10^{14}	6.90×10^{11}	7.50×10^{-2}	5.75×10^{1}
Tubing from Outlet to Waste Container	9.62×10^{12}	9.62×10^{8}	7.42×10^{1}	8.02×10^{-2}
Total Pathway	4.66×10^{15}	4.66×10^{12}	9.28×10^{1}	3.88×10^{2}

The measured pressures and flow rates were tested to ensure they fall within the equipment specification ranges for the different design configurations of channel width and inlet ports, and the different sample fluids. The low end of the viscosity range was tested using DI water (viscosity of 1 mPa·s), with 100 μm wide parallel channels at a flow rate of 1 μL/min. It was shown that using a channel with a depth of 100 μm and using the shortest inlet path, that the channels could measure flow rates as low as 1 μL/min for a measured pressure drop of 1.42 Pa. It would have a total hydraulic resistance of 4.10×10^{12} Pa•s/m^3, pressure input of 68.37 Pa and an internal priming volume of 92.98 μL. These conditions would give the lowest shear rates, whereas the highest shear rates would be created with the following conditions.

The maximum capabilities were calculated for a fluid viscosity of 20 mPa·s (the highest viscosity expected for blood at the lowest design shear rate), and 20 μm wide parallel channels at a flow rate of 7 μL/min. It was found that with channels 100 μm in height, and using all the added resistances, that these channels could measure at flow rates as high as 5.75 μL/min for a measured pressure drop of 1261 Pa. Under these conditions, the chip would have a total hydraulic resistance of 1.05×10^{14} Pa•s/m^3, pressure input of 1.01×10^4 Pa and an internal priming volume of 92.48 μL. This analysis shows that, except at these conditions which would provide the lowest/highest shear rates, the chip and sensor designs can adequately measure the desired ranges of viscosity and shear rate for the desired fluids.

2.5.Design Analysis

2.5.1. Operating Range of Device Design

The operating range for each design was estimated using Equation (31) and the values in Table 15.

Table 15. Specifications used to estimate the operating range for the proposed designs.

Specification	100 µm Design	60 µm Design	40 µm Design	20 µm Design
Minimum Pressure Drop (Pa)	1.25	1.25	1.25	1.25
Maximum Pressure Drop (Pa)	1245	1245	1245	1245
Channel Length (mm)	4	4	4	1.2
Channel Depth (μm)	100	100	100	100
Width of Parallel Channels (μm)	100	60	40	20

In Figure 20 to Figure 23, the operating range for each of the chip designs is plotted with the viscosity of blood vs. shear rate, as reported by Chien [24]. At each shear rate, the minimum viscosity that can be measured is indicated by the orange line and the maximum viscosity that can be measured is indicated by the blue line. These plots show that the viscosity of blood falls within the proposed design's operating range for shear rates between approximately 1 s^{-1} to 1000 s^{-1}. This lower shear rate range is of interest because it enables the testing of the aggregation effects of blood, as was required by the design specifications (aggregation normally occurs at shear rates less than 100 s^{-1}.) [24]. Viscosity levels near that of blood can be measured for shear rates as high as almost 1000 s^{-1}, using certain chip designs (notably, the 20 and 100 μm width designs). The estimated operating range of our device can be compared to the example commercial operating ranges shown in Figure 16 for Chip A05 and Figure 17 for Chip B05, which show that the lowest shear rates that they can measure the viscosity of blood at are 50 s^{-1} and 300 s^{-1}, respectively. These shear rates are near or above the value which blood viscosity can be measured at, whereas the proposed design is more suitable for testing at the lower shear rates in the range of aggregation and the range typical of flow rates in the microcirculation.

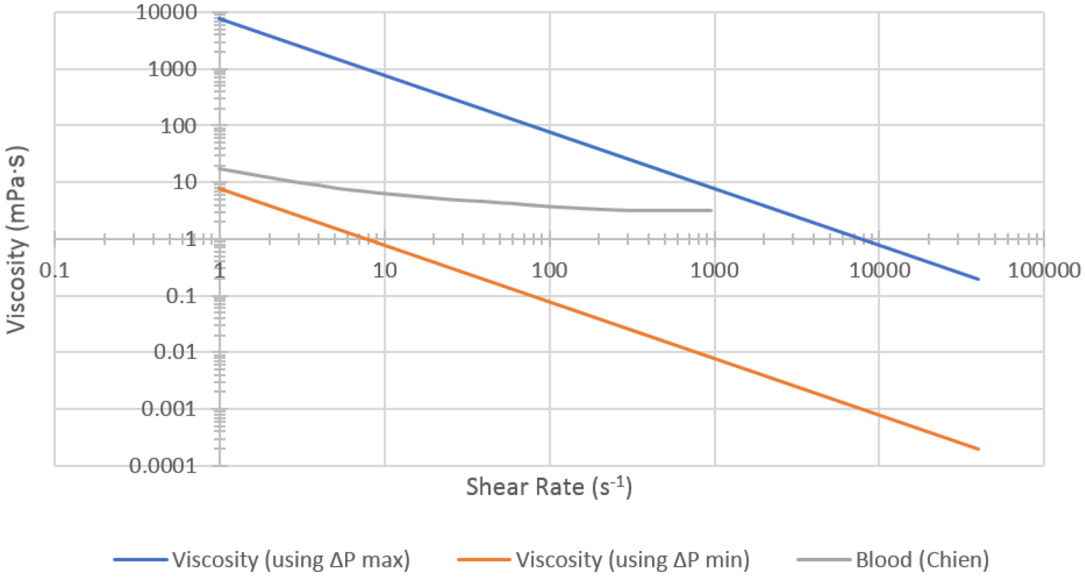


Figure 20. Operating range for the design using 100 μm wide parallel channels and using the max differential pressure (blue), min differential pressure (orange) in comparison to blood viscosity at 45% hematocrit (grey) as by Chien [24].

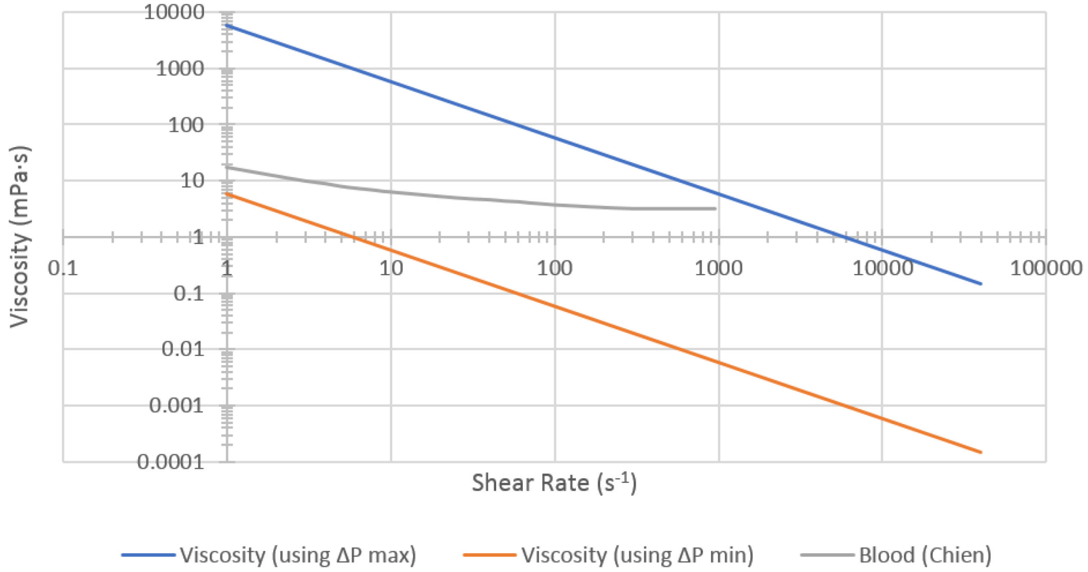


Figure 21. Operating range for the design using 60 µm wide parallel channels and using the max differential pressure (blue), min differential pressure (orange) in comparison to blood viscosity at 45% hematocrit (grey) as by Chien [24].

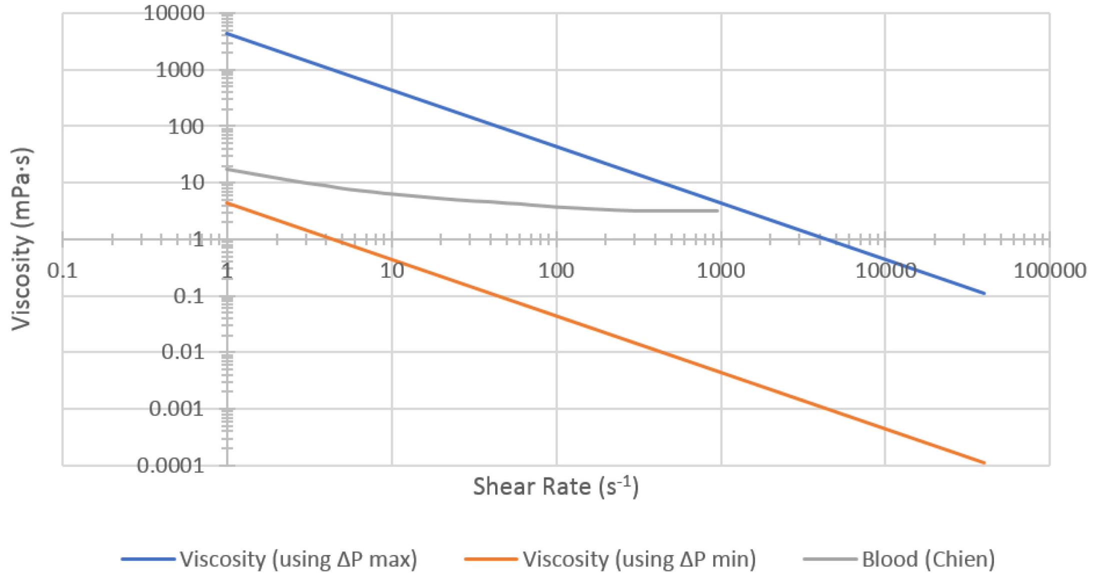


Figure 22. Operating range for the design using 40 µm wide parallel channels and using the max differential pressure (blue), min differential pressure (orange) in comparison to blood viscosity at 45% hematocrit (grey) as by Chien [24].

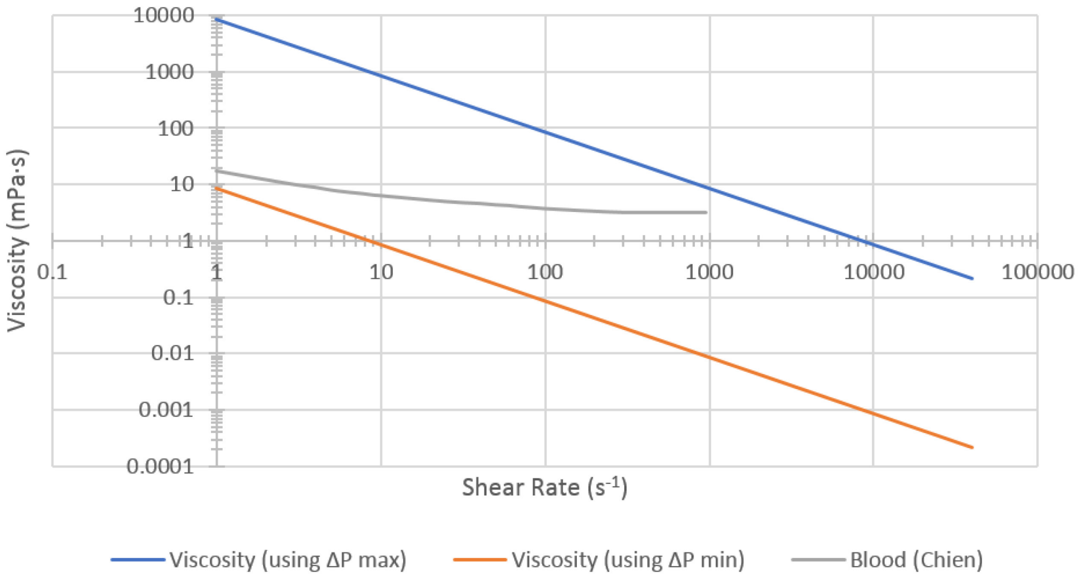


Figure 23. Operating range for the design using 20 μm wide parallel channels and using the max differential pressure (blue), min differential pressure (orange) in comparison to blood viscosity at 45% hematocrit (grey) as by Chien [24].

The operating range of the device is limited by the capabilities of the instrumentation, namely the pressure sensor, pressure controller, and the flow rate sensor, which are further examined in the discussion located in Limitations and Errors Section 5.1. It is not recommended to measure the viscosity of water at the pressure sensor's lowest limit, since it falls outside of the device operating range. A limitation also exists with the shear rates capabilities within the chip since the flow rate sensor itself is only able to measure within a certain range.

2.5.2. Channel Deformability

The deformability of PDMS was investigated since it is a soft polymer that can even be easily deformed by using human physical force. It is important to note that at maximum experimental conditions, the pressure at the entrance of the chip was estimated to be 4.99×10^3 Pa, using the same conditions for the maximum capabilities in the previous section. This value was calculated as the sum of the pressure drops from the tubing in the waste container to the inlet port of the microfluidic chip. This maximum pressure is much less than PDMS's breaking limit, when plasma bonded to a piece of glass, which is approximately 700 kPa [81].

Studies have been conducted for deformation of different microchannel structures in PDMS bonded to glass, such as deformation of micropillars [81], [82], rectangular channels due to solvents [83], [84], and rectangular channels due to DI water solutions [83], [85]. The specification range and microfluidic chip configuration of the present design are different from these studies in terms of aspect ratio, applied pressure and dimensions. Micropillars with diameter × height × spacing of 25 × 6 × 30 µm in a microfluidic chip 1 mm thick experienced about 1% change in deformation due to a pressure of approximately 4.99×10^3 Pa [82]. Meanwhile, micropillars with diameter × height × spacing of 22 × 40 × 44 µm in a microfluidic chip 1.5 mm thick deformed less than 1% due to a pressure of the same magnitude [81]. The micropillars in the present design have diameter × height × spacing of 100 × 100 × 100 µm, and the chip is approximately 8 mm thick, so the micropillars would be expected to deform negligible amounts due to the microposts' dimensions and the thickness of the chip.

Studies investigating the deformation of rectangular channels have shown negligible changes in the channel widths due to deformation, and have therefore instead investigated the channel height increase due to flow pressure [83], [85]. Estimates of channel height deformation as by Hardy *et al.* [85] with width × height of 250 µm × 37 µm with 1.5 mm and 3 mm thick PDMS layer, indicated less than 1% difference in height is to be expected with a pressure of approximately 4.99×10^3 Pa. Similarly, estimates of channel height deformation as by Kim *et al.* [83] with width × height as 400 µm × 40 µm with unknown thickness of PDMS layer, also indicated less than 1% difference in height is to be expected with a pressure of approximately 4.99×10^3 Pa. Since the largest channels of the present chip designs are width × height of 150 µm × 100 µm with approximately 8 mm thick PDMS layer, the microchannel height would be expected to deform negligible amounts.

2.5.3. Determining Equilibrium and Reduction of Compliance

The entrance lengths required to establish fully developed flow in the channels were estimated using conditions that would require the longest distance in the microfluidic chip's parallel channels, which would give the worst-case scenario. The entrance length for a given channel depends on the Reynolds number and the channel dimensions, whereby an increase in either of these would cause an increase of the entrance length. For the present designs, the conditions resulting in the longest entrance length occur in the parallel channels of the chip, even though fluidic flows with higher Reynolds numbers are usually laminar due to the inertial effects being much smaller than the viscous effects.

Combining the Reynolds number equation and the flow velocity relation gives the following Equations (32) to (34):

$$Re = \frac{\rho V L_c}{\mu} \tag{32}$$

$$V = \frac{Q}{A} = \frac{Q}{aL_c} \tag{33}$$

$$Re = \frac{\rho L_c}{\mu}\frac{Q}{aL_c} = \frac{4Q\rho}{a\mu} \tag{34}$$

where V is the average flow velocity, ρ is the fluid density, μ is the fluid dynamic viscosity, and L_c is the characteristic length. The average flow velocity (V) depends on the flow rate (Q) and cross-sectional area of the channel (A), which can be measured for the experiments. For a rectangular channel, the shortest side length is L_c and longest side length is a.

The largest Reynolds number expected in any of the chip designs, across the full operating range, was estimated to be 7.29×10^{-8} using Equation (34) and the values in Table 16. As anticipated, this value confirms that the flow in this channel will be laminar.

Table 16. Variables used to calculate the largest expected Reynolds Number in the parallel channels of the microfluidic design.

Parameter	Variable	Value
Viscosity Estimate (mPa·s)	μ	1
Flow Rate (μL/min)	Q	7
Channel Height (μm)	a	100
Channel Width (μm)	L_c	100
Density (kg/m^3)	ρ	1
Reynolds Number	Re	7.29×10^{-8}

The entrance length (L_e) for this low Reynolds number is estimated by using the following equation, which was presented by Ahmad and Hassan [86] for a square cross-section, with a hydraulic diameter of D_h. The range of Reynolds number for the parallel channel being investigated is much lower than the range tested by Ahmad and Hassan [86], but using their equation should give an estimate of the entrance length for our design.

$$L_e = \left(\frac{0.6}{0.14Re + 1} + 0.0752Re\right) D_h \tag{35}$$

Using Equation (35), an entrance length of approximately 6 μm was found, as presented in Table 17. Since this is shorter than any of the designs' parallel channel lengths, the flows are assumed to be fully developed well within the channels upstream of the central regions, and that imaging is possible between the margins.

Table 17. The parameters used in the calculation of the maximum entrance length.

Parameter	Variable	Value
Reynolds Number	Re	7.29×10^{-8}
Hydraulic Diameter (μm)	D_h	100
Entrance Length (μm)	L_e	6

The time to reach steady state flow within the microfluidic chips was also estimated. The relaxation time was calculated based on the compliance in the system. The largest source of compliance in the system came from air pockets within the tubing connecting the pressure sensor to our chip and any flexible tubing being used. These air pockets compress under pressure and cause the experiment to take longer to get to steady state to make measurements.

An estimation of the compliance in the system was completed to determine the time required to reach steady state, since the RBCs suspended in a fluid would settle in the channels over time. If the measurement time is exceptionally long, and significant settling occurs, the viscosity measurement may be affected by the resulting non-homogeneity of the sample.

Since the flow has a very low Reynolds number, the settling time was estimated using the Stokes' settling time. Approximating the RBC as a sphere of radius 3.5 μm in a 100 μm tall channel gives a settling time of approximately 54 seconds using values from Table 18 and Equation (36) shown below. This would allow approximately 54 seconds while doing an experiment with no external influences until the RBCs are expected to fully settle. One aspect to note is that this is a rough approximation of the settling time of blood, and it would vary in samples from different donors. The viscosity and density of the suspending fluid would also depend on whether human plasma or synthetic plasma were used.

Table 18. Fluidic parameters used for the estimation of the settling time of a RBC in a channel filled with plasma with a channel height of 100 μm.

Fluidic Parameter	Variable	Value	Units
Viscosity of Plasma	μ	1.00×10^{-3}	Pa•s
Density of RBC	ρ_{RBC}	1100	kg/m^3
Density of Plasma	ρ_{plasma}	1030	kg/m^3
Radius of RBC	r_{RBC}	3.50×10^{-6}	m
Acceleration due to Gravity	g	9.81	m/s^2
Velocity	V	1.60×10^{-6}	m/s
Channel Height	h	1.00×10^{-4}	m
Settling Time	t_{settle}	53.5	s

$$V = \frac{2}{9}\frac{(\rho_{RBC} - \rho_{plasma})}{\mu} g r_{RBC}^2$$

$$t_{settle} = \frac{h}{V} = \frac{9}{2}\frac{h\mu}{(\rho_{RBC} - \rho_{plasma}) g r_{RBC}^2} \tag{36}$$

The fluidic relaxation times being considered were for the tubing to expand due to the fluid flowing through it, for the fluid itself to change volume, and for any possible air pockets within the system.

Relaxation times were calculated using equations presented by Tabeling [65] for compressibility due to the fluid flowing from one channel to another. The relaxation time for a system with rigid walls ($\tau_{rigid\ walls}$) is given by Equation (37) and the relaxation time for a system with deformable walls ($\tau_{deformable\ walls}$) is given by Equation (38) [65].

$$\tau_{rigid\ walls} = \frac{3\pi l_o \mu D^2 l_i}{E w_o b^3} \tag{37}$$

$$\tau_{deformable\ walls} = \frac{6\pi l_o \mu D^2 l_i}{K w_o b^3} \tag{38}$$

$$K_{settle} = E\frac{R_0^2 - R_i^2}{2R_0^2(1 - \sigma^2)} \tag{39}$$

where l_o is the length of the output channel, μ is the dynamic viscosity, D is the diameter of the input channel, l_i is the length of the input channel, E is the Young's modulus, K_{settle} is a constant that depends on the Young's modulus of the tube material, w_o is the width of the output channel and b is the height of the output channel [65]. The K_{settle} constant is calculated using Equation (39), where R_0 is the outer radius of a channel, R_i is the inner radius of a channel and σ is the Poisson's coefficient of the tubing material [87].

The relaxation time of the system was initially measured using DI water as a testing fluid. To calculate the largest relaxation time for both of these equations, the connection between the tubing from the flow rate sensor to the first inlet was investigated. Relaxation times and the values used to calculate them are listed in Table 19. It shows that the relaxation times are much shorter in comparison to a potential experiment time scale, since these times are much less than 1 second.

Table 19. Values used to calculate the relaxation times for the connection between the tubing from the flow rate sensor to the first inlet, assuming there are rigid walls or deformable walls and using DI water.

Channel Parameters	Variable	Value	Units
Length of Microchannel	l_o	1.02×10^{-2}	m
Dynamic Viscosity	μ	1.00×10^{-3}	Pa•s
Hydraulic Diameter of Input	D	7.94×10^{-4}	m
Length	l_i	1.50×10^{-1}	m
Width of Channel	w_o	1.50×10^{-4}	m
Depth of Channel	d	1.00×10^{-4}	m
Young's Modulus	E	12.1×10^{6}	Pa
Outer Radius	R_0	1.19×10^{-3}	m
Inner Radius	R_i	3.97×10^{-4}	m
Poisson's Coefficient	σ	0.5	
Tube Dilatability	K_{settle}	7.17×10^{6}	Pa
Relaxation time Considering Rigid Walls	$\tau_{rigid\ walls}$	4.98×10^{-3}	s
Relaxation time Considering Deformable Walls	$\tau_{deformable\ walls}$	1.68×10^{-2}	s

The first design had sources of compliance in the form of tiny air bubbles, extra air in the interface between the pressure sensor and fluid, and in the flexible tubing. The tiny air bubbles, as shown in Figure 24 (left), were removed by re-treating the PDMS microfluidic chip to oxygen plasma, and by priming the chip, as shown in Figure 24 (right). PDMS chips which have been freshly bonded to glass have a hydrophilic surface, with a contact angle of approximately 111 degrees [88], [89]. This hydrophilicity can be lost, however, if the chip is exposed to atmospheric air for long periods of time, but the microfluidic chip can be re-treated to cause the surface to become hydrophilic again. For priming the chip, a high viscosity fluid was flowed through first, followed by the sample fluid. Compliance in the system (C_{hyd}) was reduced by using mainly rigid tubing and reducing the amount of air pockets in the system. The relaxation time before and after reducing these air pocket volumes was calculated to measure the change in relaxation time.

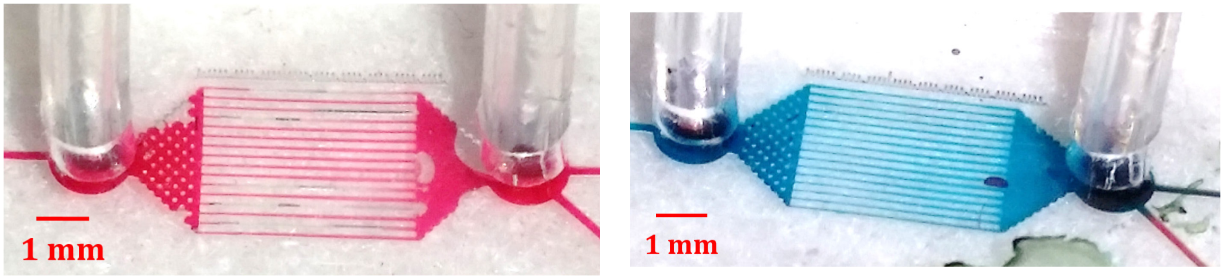


Figure 24. Reduction of bubbles in microchannels, using food colouring dyed water, between flowing through a non-plasma-treated chip (left) and flowing through a plasma-treated chip (right).

Modelling the fluidic network using resistance and compliance components, the equivalent circuit was reduced to having the large resistance and a compliance from each of the pressure sensor tubing filled with air. The initial tubing had an internal diameter of approximately 2.38 mm, and a length of 40 cm. This initial air volume was successfully reduced by replacing the tubing to one having an internal diameter of 0.397 mm, and a length of 15 cm. By reducing this volume, the relaxation time due to compliance of air pockets ($\tau_{air\ compliance}$) was calculated using Equation (40), and was reduced from 197.17 s down to 18.4 s.

$$\tau_{air\ compliance} = 2R_{hyd}C_{hyd} \tag{40}$$

Pressures due to capillary force were not considered, since they only occur due to a fluid/fluid/solid interface. Once a material has been wetted and filled with a fluid, there is no longer a capillary force created from a surface tension source.

3. Materials and Methods

3.1. Experimental Methods

3.1.1. Soft Lithography and Fabrication of PDMS Chips

Classical soft lithography techniques were used to create the chips using a silicon wafer (University Wafer Inc., South Boston, United States), SU-8 photopolymer (Dow Corning, Midland, United States) and PDMS. The PDMS chips were made within a clean room environment to maintain clean working surfaces, and a fume hood was used to extract toxic fumes.

3.1.1.1. Mask Creation

A digital 3D model of the final design was created using SolidWorks 2012 and exported to AUTOCAD in the DWG file format. AUTOCAD was used to group design lines together and fill in portions of the design which were to be transparent to light. Feature lines, and the distances between them, were kept above a 10 µm size for mask resolution. Once the file was processed with AUTOCAD, the design was exported into a DXF file format, as shown in Figure 25 (left). The DXF file was sent to CAD/Art Services Inc. (Bandon, United States), a company specializing in creating photomasks, to create a mask. The mask was printed on a transparency with the channel mould sections being transparent to light.

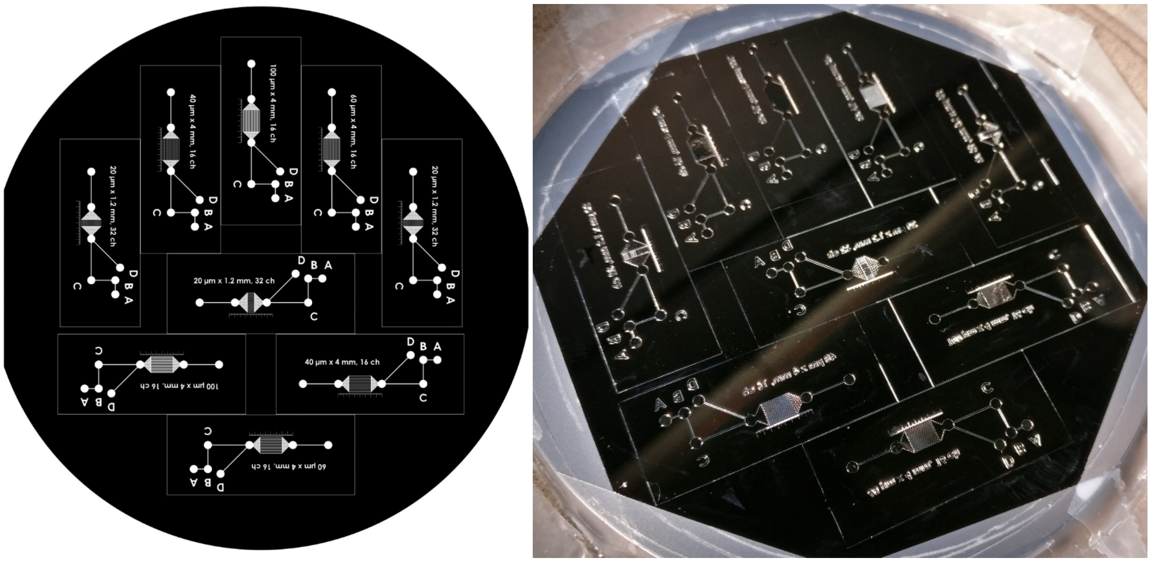


Figure 25. Mask of wafer design (10 cm diameter) indicating the arrangement of chip designs, with white sections being transparent to light (left). Wafer mould (10 cm diameter) with the final designs with parallel channel widths of 20 µm, 40 µm, 60 µm and 100 µm (right).

Additional design considerations were made while making the wafer design, as shown in Figure 25. A ruled line was added below the parallel channels, on each chip, for positioning purposes of cutting the chip for measurements or for visual alignment during tests. Each design was labelled with the width, the length, and number of the parallel channels; and the selectable ports on each design were labelled A to D. A border was created around the designs as a guide for where to cut each of the designs and leave sufficient material around each of the ports. This extra material is important to ensure there is not a leak path for the liquid to come out of when the fluidic connections were made.

A duplicate of each design was included in the mask in case one of the two identical designs were damaged. An additional copy of the design with the smallest parallel channel width was also added, due to the increased risk that the delicate features would become broken or clogged (after repeated PDMS pours). The design features were placed with an extra 5 mm margin from the wafer edge. This margin was used to avoid the mask being aligned too close to the edge of the wafer and to avoid any effects near the edge of the wafer, such as photopolymer lift-off or distortion of wafer features.

3.1.1.2. Wafer Mould Fabrication

To create the wafer mould, soft lithography was used as shown in Figure 26. A new 10 cm (4 inch) silicon wafer, as seen in Figure 27, was first cleaned in a clean room using hydrofluoric acid (1% w/v) to remove silicon dioxide from the surface. This surface preparation was used to maximize the adhesion of the photopolymer to the surface. A summary of the cleaning procedure follows, with the full SOP located in Appendix A. The wafer was first placed in a bath of HF for 2 – 4 minutes, before washing it in a DI water bath for a minute, and then rinsed with DI water. The hydrophobicity of the surface was checked to ensure it was hydrophobic, by examining how the water beaded up on the surface. Since pure silicon is hydrophobic, a high droplet contact angle and beading would indicate a successful hydrofluoric acid cleaning. The cleaned wafer was kept in the clean room environment inside a chamber with a desiccant until it was processed.

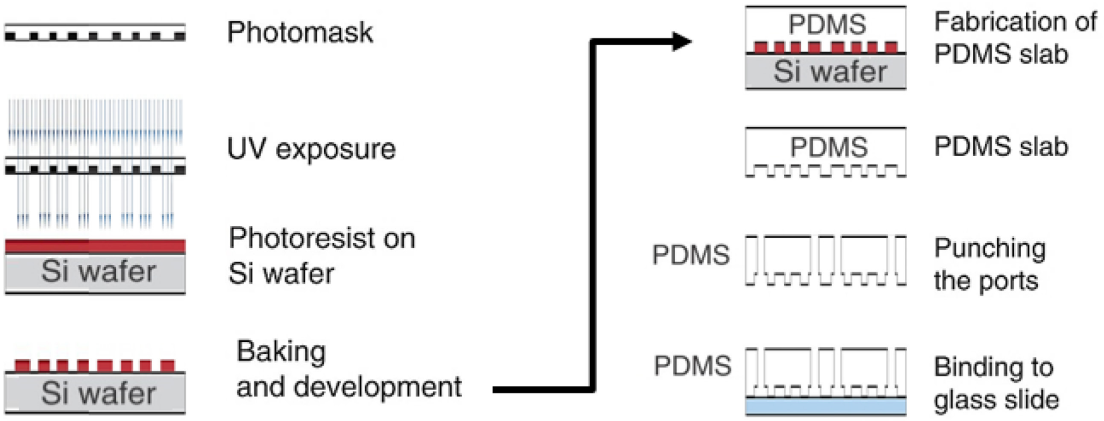


Figure 26. Soft lithography was used to create our microfluidic chips. The column on the left depicts the process to create the wafer mould and the right depicts the process to use the wafer mould to make a PDMS chip and create the microfluidic chip. (Adapted from Mazutis et al. [90]*)*

Figure 27. Testing the hydrophobicity of the 10 cm silicon wafer using DI water to see if water beaded on the surface.

The manufacturer recommended procedure for using SU-8 50 (MicroChem, Westborough, United States) to create a 100 µm thick layer was used to create the wafer mould [91]. The cleaned wafer was placed in a spin coater (WS-650MZ-23NPPB; Laurell Technologies Corporation, North Wales, United States) using a provided alignment tool to centre the wafer to the axis of rotation. The wafer was held in place using vacuum suction during every spin sequence. Test spins were done to ensure that the wafer was centred properly. Approximately 4 mL of the SU-8 50 polymer was then deposited in the middle of the wafer using a disposable plastic syringe. The creation of bubbles was carefully avoided both during the drawing and depositing of the polymer. The wafer was then spin-coated using a two-step process to first spread the polymer, and secondly to create a uniform thickness, as shown in Figure 28. The spread step consisted of a ramp up to a constant acceleration of the wafer of 500 rev/min at an acceleration of 100 rev/min/s for 10 seconds [91].

Immediately after the spread step, a final speed of 1000 rev/min was achieved at an acceleration of 300 rev/min/s and held for a total of 30 seconds [91].

Figure 28. The wafer was placed in a spin coater to obtain a level surface of SU-8 polymer. The spin coater spins the wafer at a certain angular velocity to obtain a desired polymer thickness.

The polymer layer is solidified through a two-step soft bake to evaporate the solvent using an Isotemp hot plate (ThermoFisher, Waltham, United States) at 65 ºC and 95 ºC, for 10 minutes and 30 minutes, respectively [91]. A mask containing the negative of the design was then aligned over the spin-coated silicon wafer. To improve the contact between the wafer and the mask, a thin layer of glycerol was used between the mask and wafer due to it having a similar refractive index to SU-8 50 [92]. Choosing a fluid with a similar refractive index would ensure that the light would not refract in the gap (or material) between the mask and the wafer, to result in straight-walled channels. A couple millilitres of glycerol was placed at the centre of the wafer, on top of the solidified SU-8 50 layer, with the mask placed on top of the glycerol. Pressure was applied upon the mask to gradually spread out the glycerol and avoid air bubbles. Once the glycerol was evenly spread over the whole SU-8 50 surface, pressure was applied to the wafer by a flat solid surface to press out the extra glycerol. Excess glycerol was cleaned away from the edges of the mask, while ensuring the mask designs were aligned over each other.

The mask/glycerol/SU-8 50/wafer stack was then exposed to ultraviolet (UV) light using an Intelli-Ray 600 (Uvitron International, Inc., West Springfield, United States) UV lamp with a UV1080 Rayven curing oven box (Uvitron International, Inc., West Springfield, United States) to cross-link (i.e. cure) the exposed features. A glass UV polarizing filter was placed on top of the mask/glycerol/SU-8 50/wafer stack to further align the UV rays. The UV box outputs the majority of its light in the 350 nm – 400 nm wavelength range, as shown in Figure 29, which is the optimal range for SU-8 50 exposure [91]. For the 100 μm thickness of the applied SU-8 layer, an exposure dose of 500 $\mathrm{mJ/cm^2}$- 650 $\mathrm{mJ/cm^2}$ was required to fully cure the exposed SU-8. The UV box outputs an irradiance of 175 $\mathrm{mW/cm^2}$ at 3 inches from the bottom of the lamp surface. Using 100% intensity for the box, an unfiltered exposure time of 3 to 4 seconds was necessary to fit these parameters. Due to the use of a polarizing filter, which would reduce the intensity, four seconds was selected to ensure a full exposure dose.

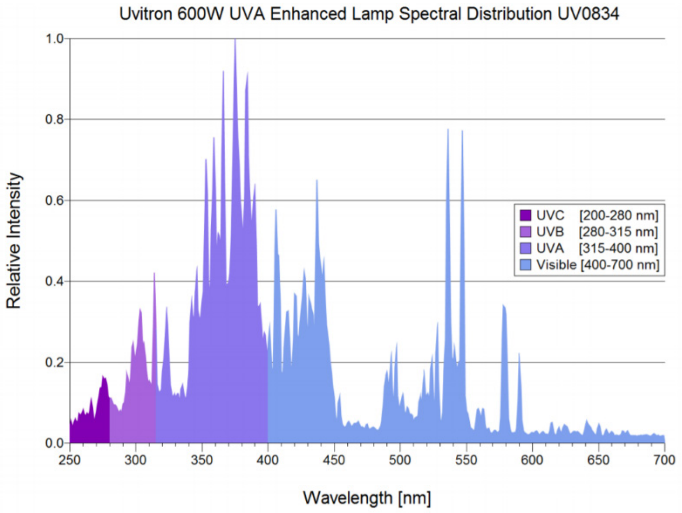


Figure 29. Intelli-Ray 600 spectral distribution, which indicates that the majority of the light output is located in the 350 nm – 400 nm wavelength range.

A post exposure bake was then completed for the SU-8 50 at 100 μm thickness, using an Isotemp hot plate (ThermoFisher, Waltham, United States), first at 65 °C for 1 minute and then at 95 °C for 10 minutes [91]. After letting it cool to room temperature, the wafer was then immersed into a glass dish containing SU-8 Developer (MicroChem, Westborough, United States) for approximately 10 minutes. This process removes the non-cross-linked SU-8, leaving raised portions of the cross-linked SU-8, which will make the microchannels of the final PDMS chip. Once the developing time was over, the wafer was lifted out of the solution using tweezers, and held over a glass waste container. Isopropyl alcohol (99%) was used to wash away any developing solution left on the SU-8 50 features and silicon wafer. Underdevelopment of the wafer was indicated if any cloudy opaque fluid was created when applying the isopropyl alcohol. If the wafer was underdeveloped, the wafer was re-immersed into the developing solution again for another 30 seconds and rinsed again with isopropyl alcohol to test for underdevelopment. The re-immersion and rinsing would be repeated until the wafer was fully developed. After the rinse indicates a fully developed wafer, it is then dried off with nitrogen gas. To prevent particles from settling on the surface before being used, the SU-8 features are kept in a clean room environment in a closed container.

3.1.1.3. PDMS Chip Fabrication

PDMS was chosen as the polymer to be used to create the microfluidic channels. This soft and optically clear polymer was poured on top of the wafer mould and took the shape of the features on the wafer. PDMS (Sylgard 184, Ellsworth Adhesives, Germantown, United States) in a 10:1 (base:curing agent) w/w ratio was then mixed and degassed before pouring over the wafer mould. The PDMS was further degassed after being poured on the channels to remove air bubbles created near the wafer features. A PDMS thickness of approximately 8 mm was used on the chip designs. The liquid PDMS was then baked for 10 minutes on an Isotemp hot plate (ThermoFisher, Waltham, United States) at 150 ºC, as shown in Figure 30.

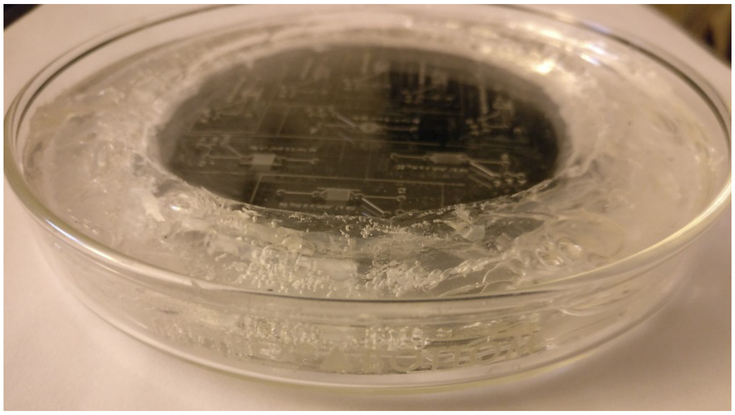


Figure 30. Wafer mould inside of a glass petri dish with PDMS cured on top. PDMS over the wafer is cut out to be used for microfluidic chips. The ring of PDMS that is higher than the central portion is from previous uses of the wafer mould.

After cooling to room temperature, the solidified PDMS over the wafer was cut (using the chip borders as a guide) using a scalpel and then peeled off the wafer. Pieces of clear adhesive tape were placed on the channel side of the PDMS and wafer to prevent dust particles from entering the channels. The individual chip designs were then cut out of the PDMS and labelled as to which part of the wafer the design came from, as designated in Figure 31 (top right). Inlet and outlet ports were then punched out of the PDMS chips, before the chips were bonded to a glass microscope slide using oxygen plasma in a PE-50 Compact Benchtop Plasma Cleaning System (Plasma Etch, Inc.). Both clean surfaces of the glass microscope slide and PDMS were plasma-treated for

2 minutes before bonding. The bond was further solidified by placing the bonded chip on a hot plate for 5 minutes at 95 ºC.

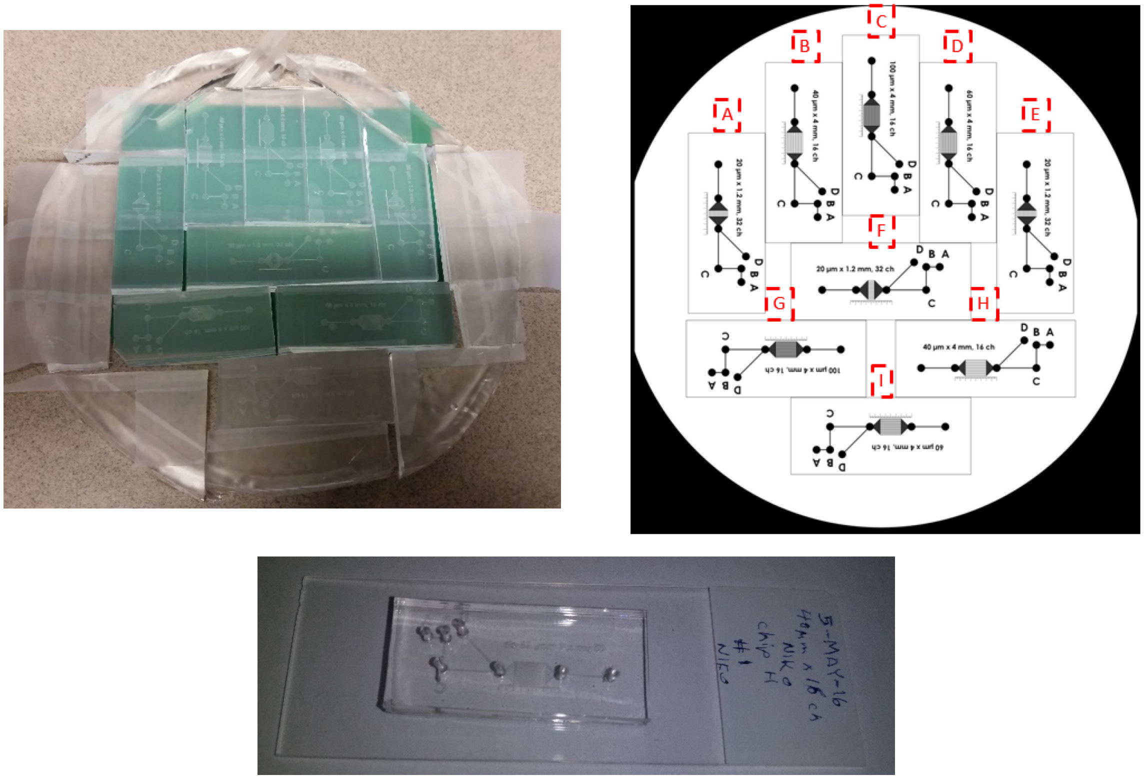


Figure 31. The PDMS was cut into the individual chips along the design edges (top left). Chip designs on the wafer (10 cm diameter) were labelled for identification due to the wafer design containing duplicates of the same design (top right). After punching holes for the ports, chip designs were plasma bonded to a glass microscope slide (bottom).

3.1.2. Experimental Setup

The PDMS chip and experimental setup are shown in the flow diagram in Figure 32, with a flow diagram shown in Figure 33, which indicates how the fluid sample moves to the microfluidic chip. To perform the experiments, software programs by Fluigent were used to control input pressures and collect data from the flow sensor and pressure controller. A separate program, custom-made in MATLAB, collected data from differential pressure (placed across the resistance portion of the chip) and temperature sensors (that currently measure the temperature of the room).

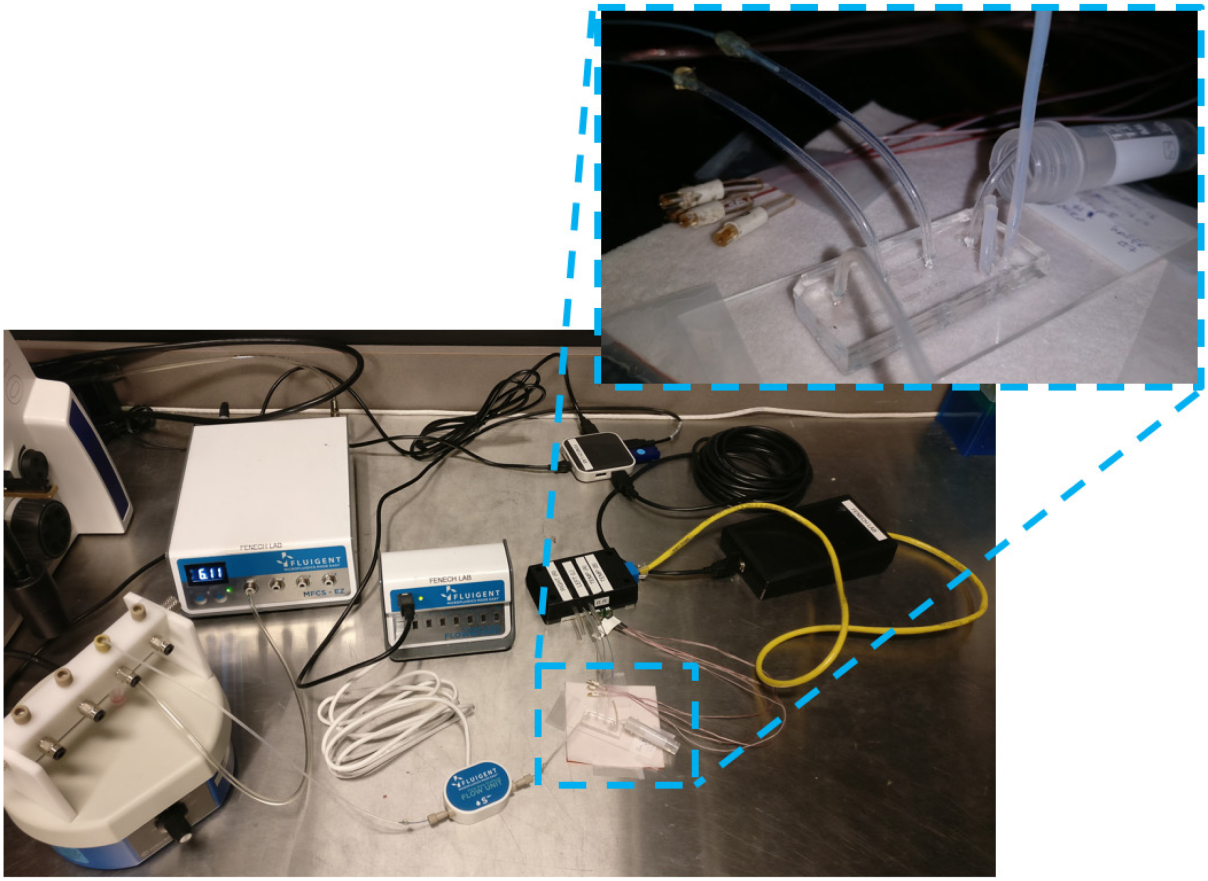


Figure 32. Experimental setup of pressure-driven flow (bottom left) with the fluidic connections to the microfluidic chip highlighted (top right).

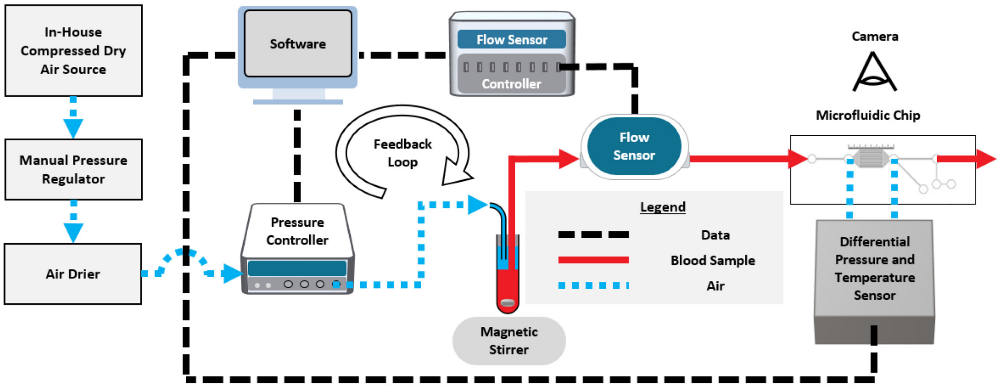


Figure 33. Total flow diagram of experimental setup with the pressure-driven flow, flow of sample through the microfluidic chip and feedback loop pathways (not drawn to scale).

Key elements of the experimental setup will be described here in detail, beginning with the compressed air system which drove the blood flow, highlighted in Figure 34.

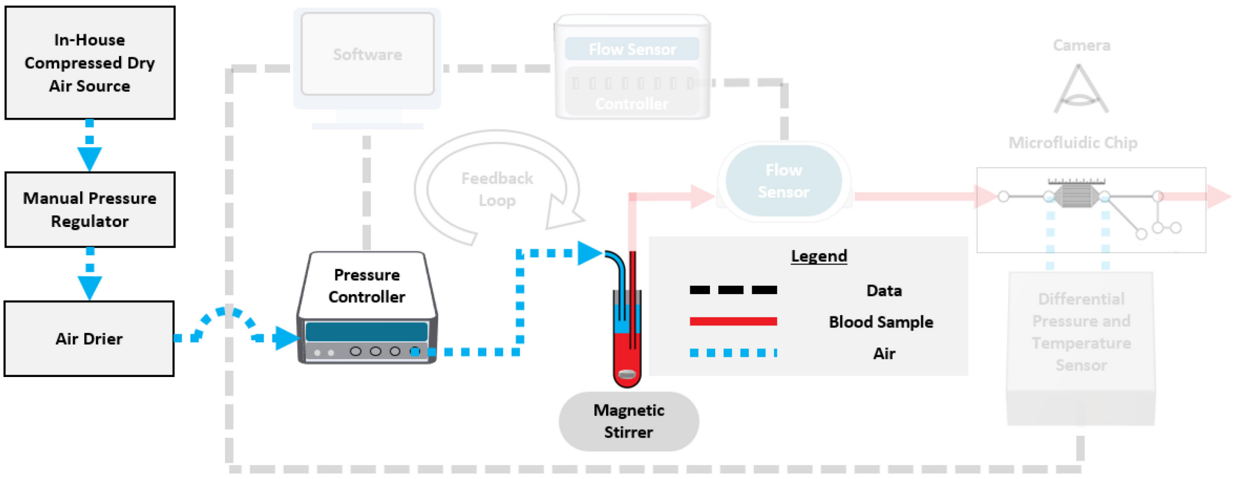


Figure 34. Introduction of the compressed air to the sample is controlled by a pressure controller. The input air pressure from an in-house compressed dry air source is regulated using a manual pressure regulator and dried (not drawn to scale).

A MFCS-EZ pressure controller (Fluigent, Villejuif, France) was connected to the in-house compressed dry air line through a manual pressure regulator (Watson Smith 5310R0400R, IMI Precision Engineering, Fradley Park, United Kingdom) and air drier (Norgren F07-200-A3TG, Birmingham, United Kingdom). MAESFLO 3.2.1 software (Fluigent, Villejuif, France) was used to control the pressure controller to introduce compressed air at specific pressures to the plastic vial containing the fluid sample. The sample vial is connected to a Fluiwell vial holder (Fluigent, Villejuif, France) with a rigid Teflon tubing (1.59 mm O.D.) with the end located close to the bottom of the vial. A magnetic stir bar can be put in the sample vial and the Fluiwell can be placed above a magnetic stirrer to keep the sample mixed. A uniformly mixed sample is important for fluids which separate or settle, such as blood. These effects would cause the fluid flowing to the chip to be of different properties compared to the sample in the vial. For a liquid like blood, in which the RBCs settle, this would mean that a higher hematocrit would initially flow to the chip than intended, as the fluid sample is drawn from the bottom of the vial.

The sample fluid is then flowed through a flow sensor (Fluigent, Villejuif, France), as shown in Figure 35, using a combination of Tygon tubing (2.38 mm O.D.), rigid Teflon tubing (1.59 mm) and PEEK tubing (0.794 mm) into the chip's inlet. Accuracy of tubing placement could be verified visually with a microscope. The differential pressure sensor box measures the pressure at the previously mentioned ports E and F, using air as a medium. The same flow section between ports E and F can be viewed using a high-speed camera for visual processing of the parallel channels or tapered channels. Tubing identical to that used for inlet connections was used to connect the channel to a 2.5 mL plastic vial waste reservoir.

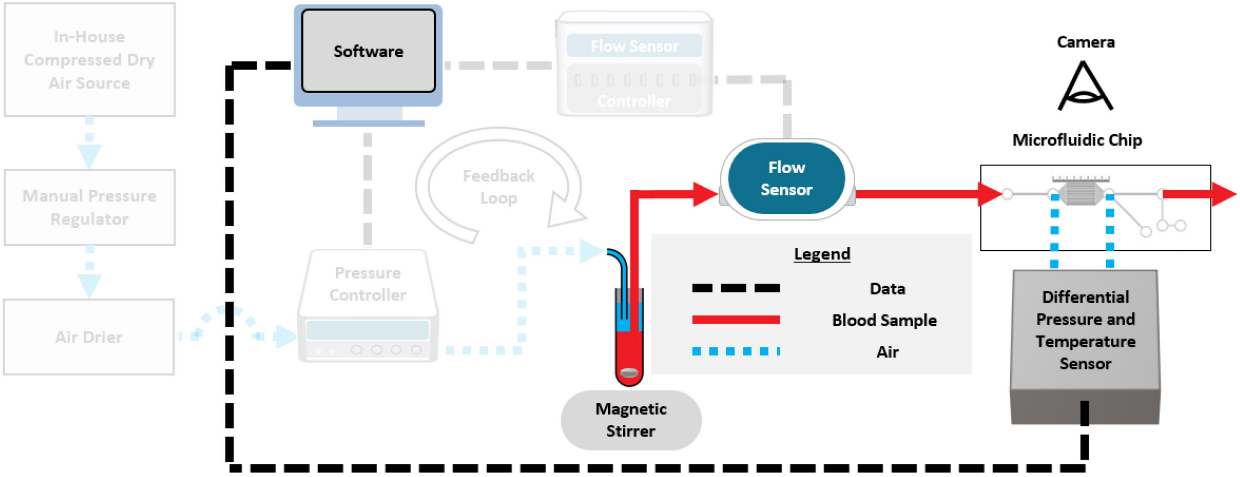


Figure 35. The sample fluid travels through the flow sensor and onto the chip. A differential pressure and temperature are measured on the chip, with capabilities for imaging using a high-speed camera. The sample exits the chip through the outlet port and flows into a waste container (not drawn to scale).

The software by Fluigent can communicate with the pressure controller and flow sensor controller in a feedback loop to obtain desired input pressures or flow rates at certain time intervals during experiments, with this communication path shown in Figure 36.

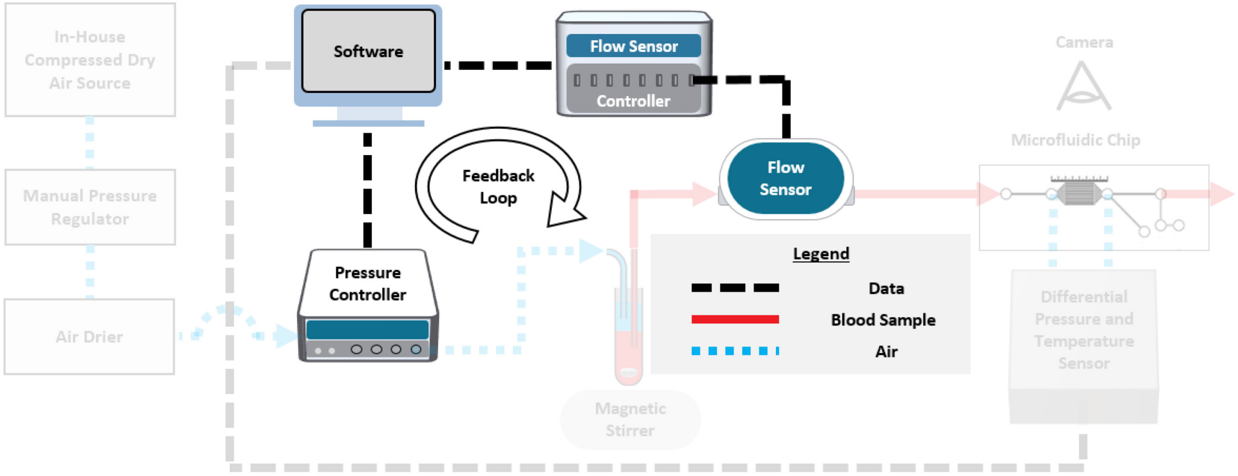


Figure 36. A software feedback loop controls and records the flow rate and pressure. Our scripts are able to control the pressure and flow rate to reach certain values at desired time intervals (not drawn to scale).

The differential pressure sensor and thermocouple module was custom designed and fabricated for the low-pressure range and sensitivity by A.L.Bi.C.S. Ltd. in conjunction with Dr. Katie Pitts (a former Ph.D. student in Dr. Marianne Fenech's research group). The measurement ports are connected to the pressure sensors within the module via tubing, using a friction fit and lubricating grease. Temperature sensors located in the module are J type thermocouple sensors with an amplifier having built-in cold junction compensation and a transfer function of 5 mV/°C signal. Each of the pressure sensors and thermocouples within this module have built-in low pass filters with cut off frequencies of 15.9 Hz and 34.3 Hz, respectively. The temperature sensors are in the form of deformable wire leads that are capable of being repositioned and placed within the microfluidic chip. Ends of these leads have a tubing sheath to seal pressure when inserted into the fluidic chip. The thermocouple measuring junctions protrude slightly to facilitate measurements when inserted into a fluid.

This setup enables the measurement of pressure drop between points E and F on the microfluidic chip, of the tapered chambers and parallel channels as shown in Chapter 4 (Experimental Methods), with a maximum linear hysteresis of 0.5% over the full pressure range of 5 inches of water (~1245 Pa). It also allows the measurement of the flow rate, within an accuracy of 5% of the measured value between the range of -7 to -0.4 μL/min or 0.4 to 7 μL/min, and 17.5 nL/min in the range of -0.4 to 0.4 μL/min [75]. These measurements of pressure and flow rate can be used to estimate the apparent viscosities of sample fluids, using equations as shown in Section 2.2.1.

3.1.2.1. Compliance Reduction in the System

During testing of the chip, it was found that any additional source of compliance could negatively affect the response time of the chip. A few measures were taken to reduce the compliance during the first iterations of the experimental setup.

Initially, the flow passed from the inlet ports A, B, C and D, to G as shown in Design Principle Section 2.1.1. The unpunched A - D ports, which were not used for an experiment, would have air trapped in them once the chip was primed. This air would increase the compliance within the system and cause the time taken to reach steady state to increase. This increased compliance was resolved by flowing the fluid from the port F (downstream of the tapered chambers and parallel channels) to the selected inlet port A - D, but also punch out the other unused ports (A to D). Once the sample was primed in the channels, the selected outlet would be plugged with a solid plastic plug (same diameter as tubing). Extra fluid would then be flowed out of the other outlet ports and solidly plugged using the same cylinders, creating dead end chambers. The plugs were also applied by contacting the end to the fluid once the sample rose out of the outlet port.

It was also found that bubbles could form in the channels if the sample was introduced into a dry chip. The creation of these bubbles was prevented by first priming a plasma-treated chip with a dyed fluid having a higher viscosity than the sample fluid. PDMS that has just been plasma-treated is hydrophilic, allowing fluids like water to wet the surface and avoid bubble creation. Bonded PDMS channels can be re-treated with plasma to induce hydrophilic surfaces, with the effect duration depending on the plasma treatment time [93]. The reduction of the contact angle was minimal for exposure times greater than 300 seconds [93]. Before running the experiment, the already bonded chip was plasma-treated for 300 seconds. The fluidic tubing, the sample vials and the tubing were also treated at the same time for cleaning purposes.

The priming fluid was flowed slowly through the channels, allowing the pressures to equilibrate. It was important to ensure the parallel channels were all at the same pressure, or it was observed that the fluid would burst through one of the channels first (one with slightly less resistance and usually one of the edge channels). If there were any bubbles or blocked channels, they could be detected in the dyed fluid. Higher fluid pressure as well as light force was applied using finger pressure to the PDMS chip to remove the bubbles or channel clogs. Once the channels were completely filled with the priming liquid and were without bubbles, the fluid could be flowed through at a higher pressure to fill the rest of the chip.

A denser and higher viscosity priming fluid was also used to minimize the amount of residual priming fluid from staying in the pressure/temperature sensor tubing. After flowing the priming fluid in, the fluid sample would be introduced to the chip. Once the lighter and less viscous test sample fluid was present, it pushed out the more viscous and denser dyed fluid. If a lighter and less viscous priming fluid were used, the lighter fluid would rise to the top in the pressure sensor air interface and get trapped. Since the priming fluid is in contact with the sample during priming, it is necessary that the priming fluid be compatible with the sample. For the case of blood, a denser and more viscous isotonic fluid would be used.

The outlet tubing was connected to the end of the chip, with the end submerged in a sample vial containing some fluid. A pre-filled vial was used to avoid fluctuations of hydraulic resistance occurring from the creation of drops at the end of the tube. Earlier experiments, in which the outlet tube was not submerged, found there to be an almost cyclic pressure pattern when a drop would form and then fall due to the fluid's weight. The hydraulic resistance increased due to the increase of pressure required to overcome the increasing surface tension of the water droplet.

3.1.3. Fluid Preparation

A sequence of fluids was chosen to test the experimental setup, as shown in Table 20, starting with fluids of well-known properties and progressing to ones that more closely match the characteristics of whole blood. The initial fluid tested was DI water, chosen due to its availability and well-known viscosity at different temperatures [94]. To test the system with a fluid with a viscosity close to that of average human blood, a solution of DI water and glycerol was chosen as the second test fluid. Formulas for the viscosity of glycerol solutions have also been established [95]. A solution of polyethylene oxide (polyox) WSR-301 (Dow Corning, Midland, United States) in DI water was chosen as the non-Newtonian fluid for the next step. Solutions having different concentrations of polyox yield different shear thinning behaviours, which can be customized to be within a range of viscosities at certain shear rates [96].

Next, fluid suspensions containing RBCs would be tested; RBCs would then be isolated and removed from whole blood samples and mixed with synthetic plasma. PBS would be used as a synthetic plasma for this step for its isotonic fluid properties to not damage the RBCs. The composition of the synthetic plasma would be adjusted by adding compounds, such as Dextran, in order to induce various amounts of aggregation in the RBCs depending on the compound concentration and molecular size [97], [98]. The final fluid to be tested would be reconstituted blood using the autologous (i.e. native) plasma and RBCs, where the RBCs would be cleaned with the white blood cells and platelets removed. These components should be removed since they minimally affect the blood flow and the platelets can promote clotting during experiments [49].

Table 20. Sequence of experimental fluids to test fluids of known properties and progressing towards blood, with each step introducing a different rheological parameter.

Step	Sample Fluid	Known Viscosity	Similar Viscosity to Blood	Non-Newtonian	Contains Blood Cells	Whole Blood
1	DI Water	✓	✗	✗	✗	✗
2	DI Water and Glycerol	✓	✓	✗	✗	✗
3	Polyox	✓	✓	✓	✗	✗
4	Synthetic Plasma and RBCs	✗	✓	✓	✓	✗
5	Plasma and RBCs	✗	✓	✓	✓	✓

The DI water used in the experiments, for fluid preparation, and for cleaning, was collected from a Prima Filter (ELGA PURElab, Paris, France).

Reference Newtonian fluid and non-Newtonian viscosities were measured using a μVISC and m-VROC device (RheoSense, San Ramon, United States). Newtonian fluid viscosities were averaged for several shear rates (e.g. 300, 500 and 1000 s^{-1}). The non-Newtonian power law fluid, polyox (Dow Corning, Midland, United States) solution, was measured in our experiments over the whole range of the commercial viscometer to determine the power law constants.

Additional steps are required when testing with the fluids which contain RBCs (sample fluid steps 4 and 5). New tubing, chips and plastic vials would be used for each blood sample, to avoid the possibility of clogging or improperly cleaned chips. For components that are not disposable, such as the flow sensor, a sequence of fluids would be passed through them following measurements of the RBC-containing fluids, to ensure that they are cleaned properly. A proposal of such a procedure is explained in the Next Prototype to Implement Section 5.3.

3.1.3.1. De-ionized Water and Glycerol

Glycerol solutions with viscosities of 3.65, 8.4 and 17.5 mPa·s were chosen to be analyzed based on normal human blood viscosity and the full specification range. To obtain these viscosities, an initial volume/volume ratio was determined using the methods of Cheng [95]. Based on these ratios, glycerol and DI water was measured into a 15 mL plastic vial and mixed.

3.1.3.2. De-ionized Water and Polyox

A 0.1% polyox non-Newtonian solution was created for the experiments, since its viscosity range is within the design specifications. This solution was created following the manufacturer's recommended procedure [99], summarized here. The polyox powder was first pre-suspended in anhydrous 2-propanol and constantly stirred using a magnetic stirring bar. The stir bar was set to spin at 600 rpm while adding in the DI water at a constant rate to avoid clumping. The rate was decreased gradually until 60 rpm and then set to stir (covered) for 30 min to 60 min until the solution was homogenous. The fluid was then transferred to a 15 mL plastic tube for storage.

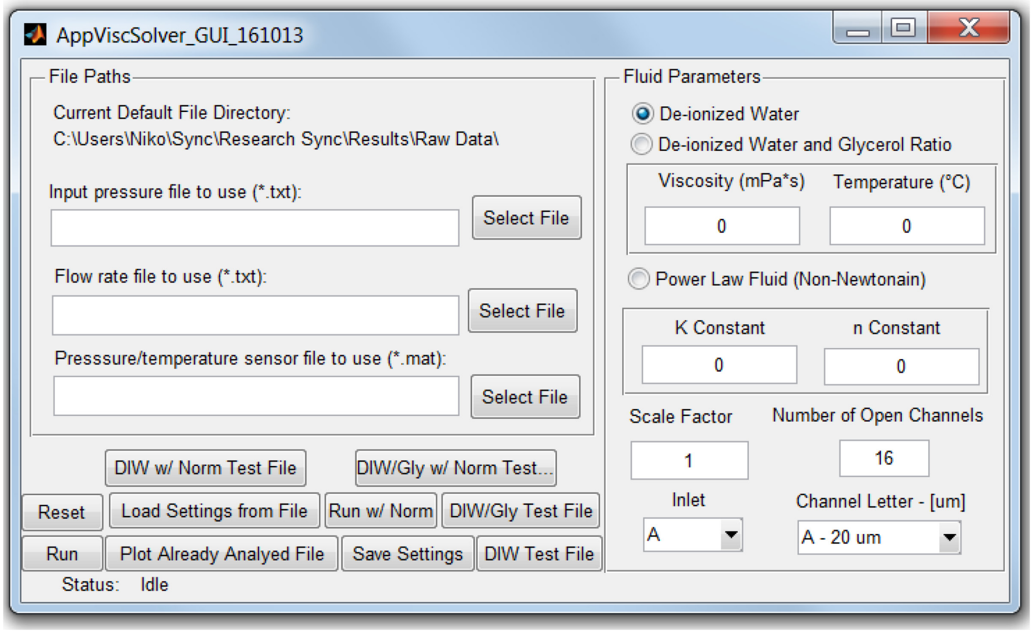
3.2.Signal Processing and Data Collection

3.2.1. Data Collection

MATLAB's signal processing toolbox was used to collect and pre-process the data signals from the pressure and temperature sensors, within a transducer/signal conditioning module, through a USB-6008 (National Instruments Corporation, Austin, United States) data acquisition device. The signals were pre-processed to convert from the raw voltage signal to the reading in Pascals for pressure and Celsius for temperature. The flow rate sensors and input pressure sensors collect data at 0.1 s intervals, whereas the pressure sensors and temperature sensors measure at 0.02 s intervals. The software used to measure the flow rate and input pressures, called FRP and MAESFLO, respectively, wrote the output data in a tab-delimited type of file. The flow rate or applied fluid pressure could be automatically changed at specific time intervals using Fluigent's SciTE Script Editor. Each fluid was tested for a sequence of flow rates which spanned the design range, each held constant over equal time intervals. By measuring over the design range, an indication could be made about the platform's ability to test at the higher and lower shear rates. The order of the flow rates in the sequence was randomized for each test.

3.2.2. Data Analysis

A graphical user interface (GUI) was created to facilitate the analysis of different fluids, with options to save and load settings for different tests and fluids, as shown in Figure 37. Data files from the experiment and known fluid parameters can be entered into the GUI to be analyzed and plotted. The analyzed data can then be saved for future analysis.



AppViscSolver_GUI_161013

File Paths

Current Default File Directory:
C:\Users\Niko\Sync\Research Sync\Results\Raw Data\

Input pressure file to use (*.txt):

Flow rate file to use (*.txt):

Presssure/temperature sensor file to use (*.mat):

Fluid Parameters

● De-ionized Water
○ De-ionized Water and Glycerol Ratio

Viscosity (mPa*s): 0
Temperature (°C): 0

○ Power Law Fluid (Non-Newtonain)

K Constant: 0
n Constant: 0

Scale Factor: 1
Number of Open Channels: 16

Inlet: A
Channel Letter - [um]: A - 20 um

Status: Idle

Figure 37. The MATLAB GUI allows users to process data files from experiments using our system. Current functionality includes analyzing experiments with DI water; DI water and glycerol; or a power law fluid. The GUI includes capabilities of saving the settings of an experiment, running analyses on data files, and plotting previously analyzed data.

The file data were first parsed, and the measurements were aligned based on time, to the closest 0.1 s interval. The measured pressure was then zeroed based on the first 10 seconds of the experiment, in which no fluid was flowing. Flow rate was scaled based on the fluid selection and scale factor. The channel dimensions were determined from the drop-down menu selection of channel design and inlet used. The program completes its calculations using the pre-measured dimensions of the channels for the selected configuration, with values found in Calibration of the Channel Dimensions Section 4.1.1. The hydraulic resistance constants were calculated by the MATLAB code using equations, as previously presented in Table 2, found in the Working Equations Section 2.2.1 (with $n = 50$ for the summation). The apparent viscosity was then calculated using the flow rate, measured pressure and the hydraulic resistance constant using Equations (25) and (26). Theoretical viscosities for the fluid were calculated using either the fluid temperature (for DI water), a viscosity measured from µVISC (RheoSense, San Ramon, United States) (for glycerol solution), or power law constants (for the non-Newtonian polyox). An estimation of the wall shear rate was calculated using Equations (9) and (10) for the Newtonian fluids (with $n = 16$ for the summation).

The input pressure from the MAESFLO software, the measured pressure from the pressure sensor, flow rate from the Flow Rate Platform software, the measured experimental viscosity, and the theoretical viscosity were then plotted, with the results presented in the next section. The data contained noise and were filtered before plotting on a graph. For initial analysis, the signal was smoothed using a moving average filter with a 1-second time window and time corrected for the filter delay.

4. Results

4.1.Calibration

Calibration of the experimental setup was completed before verifying and validating with testing sample fluids. These include calibration of the channel dimensions, sensors, and fluid properties.

Calibrations, which involved fluid flow, were completed using the channel design H (width 40 µm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.

4.1.1. Calibration of the Channel Dimensions

Dimensions of the channels were verified by cutting cross-sections of each of the designs (labelled from A to I, as shown in Figure 31) and measuring using a Lab A1 microscope (Carl Zeiss, Oberkochen, Germany). A micro ruler was used to calibrate the microscope image and lengths were calculated using the images of the cross-sections. Multiple measurements were taken for each design along their lengths. The height, average minimum width and average maximum width were measured multiple times along the length of a channel to be averaged. Since the channels did not have perpendicular sides, the average minimum and maximum width was measured at the narrowest and widest portion of the channels, respectively. Experimental calculations of the apparent viscosity used the averaged value for the respective designs when the channel dimensions were required. Examples of measurements for wafer N1 are shown in Table 21.

Table 21. Dimensions of the cross-section of the parallel channels for wafer N1, as shown in Figure 31.

Chip	**Desired Width [µm]**	**Average Narrowest Width [µm]**	**Average Widest Width [µm]**	**Average Width [µm]**	**Desired Height [µm]**	**Average Height [µm]**
A	Parallel channels did not work for this wafer					
B	40	33.89	72.25	53.07	100	108.69
C	100	98.58	134.14	116.60	100	116.60
D	60	61.31	86.20	73.75	100	113.56
E	20	18.17	43.40	30.79	100	108.21
F	20	23.42	27.71	25.57	100	100.86
G	100	98.47	115.80	107.13	100	109.05
H	40	43.92	66.71	55.31	100	101.11
I	60	59.95	87.24	73.59	100	106.91

Comparing the desired and average values, in Table 22, shows that the chips' average width and heights are all higher than the desired dimension. The width difference does not seem to be proportional as a percentage of the width, but has an average width distance difference of approximately 11.917 µm. The desired height was the same for all the channels, and there was a similar height difference with all designs. Chip F, which is the chip closest to the centre of the wafer with the average height value emboldened in Table 22, shows this dimension is the closest to the desired height. This closeness suggests that the final design could be improved by using a smaller wafer containing a unique chip at its centre. While these values differ from the desired values, the measured average width and height were used for the channel dimensions in the MATLAB program to determine the apparent viscosity.

Table 22. Difference between desired width and height with measurements for the wafer N1 of chip designs B to I, sorted by the desired channel width

Chip	**Desired Width [µm]**	**Average Width [µm]**	**Width Difference [µm]**	**Width Difference [%]**	**Desired Height [µm]**	**Average Height [µm]**	**Height Difference [µm]**	**Height Difference [%]**
E	20	30.79	10.79	53.93	100	108.21	8.21	8.21
F	20	25.57	5.57	27.83	100	100.86	0.86	**0.86**
B	40	53.07	13.07	32.68	100	108.69	8.69	8.69
H	40	55.31	15.31	38.29	100	101.11	1.11	1.11
D	60	73.75	13.75	22.92	100	113.56	13.56	13.56
I	60	73.59	13.59	22.66	100	106.91	6.91	6.91
C	100	116.60	16.60	16.60	100	116.60	16.60	16.60
G	100	107.13	7.13	7.13	100	109.05	9.05	9.05

4.1.2. Calibration of the Pressure Sensor with De-Ionized Water and Syringe Pump

The pressure sensor required calibration in order to convert from the voltage measurement signal to the pressure in Pascals. The specifications of the transducer module indicated a linear relationship with the signal voltage from -2.5 V to 2.5 V corresponding with -1245.41 Pa to 1245.41 Pa, respectively, for a linear relationship of 498.16 Pa/V. For a more precise relationship, considering environmental factors such as room temperature, the calibration was repeated within our setup.

The calibration of the sensors was completed by flowing DI water through a straight microfluidic channel defined in Table 23. Chip dimensions, as shown in Table 23, were measured using the same method as in the calibration of the chip dimensions.

Table 23. Channel dimensions of the calibration chip used to calibrate the pressure sensor using DI water and a syringe pump.

Channel Parameters	**Variable**	**Value**	**Units**
Length	L	5000	μm
Height	h	39.16	μm
Width	w	111.68	μm

Precise flows for calibration were obtained using a Nexus 3000 precision syringe pump (Chemyx, Stafford, United States) connected to the straight channel via a flexible 3/32” × 1/32” O.D. × I.D. (0.79 mm × 2.38 mm O.D. × I.D.) tubing. The theoretical pressure drops for each flow rate were calculated using Equation (7) presented in the Rectangular Channels Section 2.2.1.3. In this equation the summation was carried out to n = 10 giving a value within 0.02% of the value obtained for n = 100. The viscosity of DI water was approximated to be 1 mPa·s.

To prevent any damage of the sensor, it was found that the flow rate which generates the maximum pressure was 1.084×10^{-10} m^3/s (i.e. 6.5 μL/min). The voltages of the differential pressure sensor, which is connected between two ports of the chip, were recorded at flow rate intervals within the pressure sensor’s range. Due to the system’s compliance, the voltage measurement was taken after equilibrium.

The steady state voltage was identified by visual inspection, with the average value being recorded once the required flow rate was attained for 10 seconds. Figure 38 (left) shows an example fluid calibration with a randomized flow rate pattern. A flow rate was flowed until it reached steady state before switching to another flow rate, as shown in Figure 38 (right).

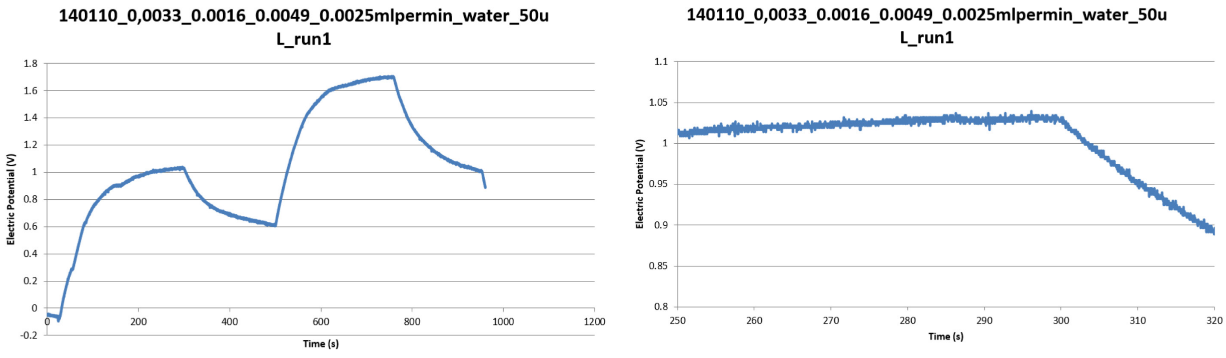


Figure 38. Example of calibration plot of pressure sensor voltage using a syringe pump and DI water. Example of a full calibration random sequence (left), and a close up of the steady state region of the plot (right).

The linear relationship between theoretical pressure - associated with the input flow rates - and the voltage recorded was determined. Figure 39 shows the electric potential measurements plotted against the theoretical pressure (Pa).

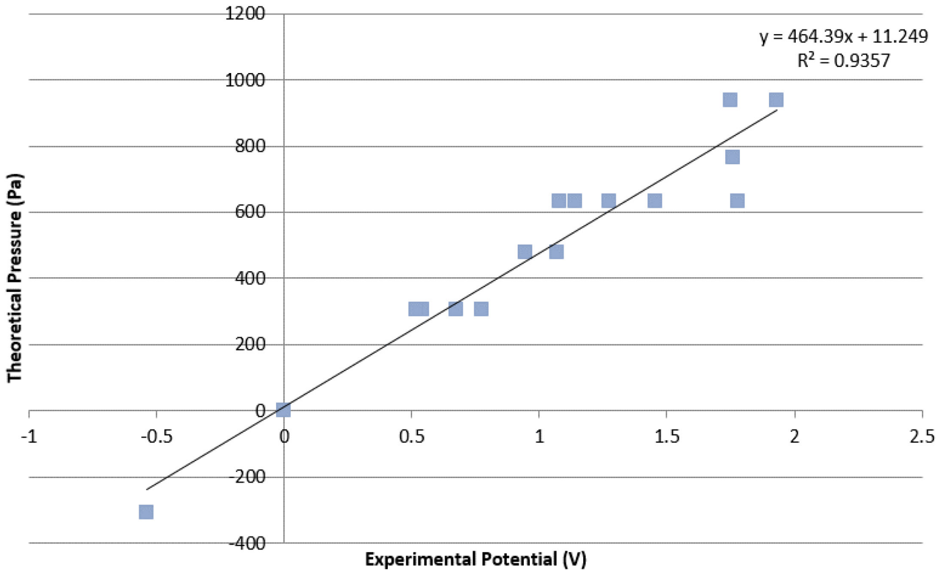


Figure 39. Determination of pressure sensor calibration constants by plotting the theoretical pressure generated in the channel against the voltage measured by the pressure sensor, where $\Delta P = 464.39v + 11.49$*,* $r^2 = 0.9357$*.*

The slope of the line, as well as the offset were used in the MATLAB Simulink program to convert from the electric potential to the pressure.

4.1.3. Calibration of the Temperature Sensors Using a Thermally Heated Block

The J type thermocouples in the pressure/temperature sensor module were calibrated using a viscometer thermal block in the m-VROC (RheoSense, San Ramon, United States) connected to a thermoelectric chiller/heater ThermoCube 300 (Solid State Cooling Systems), which was able to maintain temperatures within ±0.05 °C [100]. This device uses a coolant to heat or cool down a component by running the fluid through a thermal block designed to keep syringes at a certain temperature during an experiment. Temperature sensors were labelled based on the communication port and which branch of the lead that the thermocouple belonged to (i.e. Labelled {port}_{branch}). The temperature sensors were all placed in the thermal block at the same time, touching the inside of the block. The ThermoCube was then set at different temperatures using the manufacturer software at intervals between 20 °C to 43 °C, to record the voltage signal from the sensors. Measurements were taken, once the software indicated that it had reached a steady state, over a 5-minute period, with results shown in Figure 40.

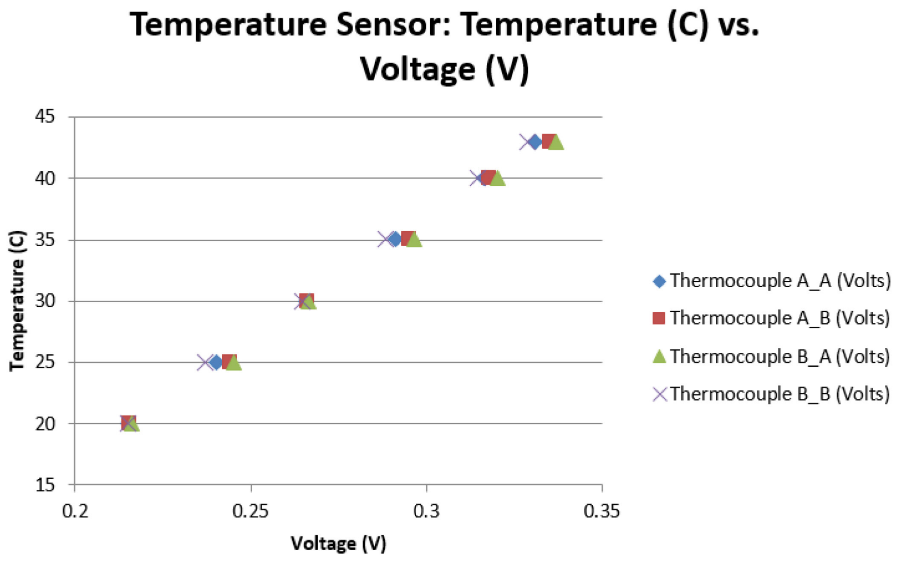


Figure 40. Calibration curve of thermocouples in the pressure/temperature module. The slope and y-axis intercept were computed and used as constants in a MATLAB Simulink program to convert the voltage to temperature in Celsius.

A linear equation was computed for each thermocouple using the measurements, which are shown in Table 24, and used in a MATLAB Simulink program, as shown in Figure 41. These values were tested by setting the thermal block at various temperatures and verifying calibrated sensors' readings. Using these calibrated probes within the platform's channels, along with heating elements, will allow for the analysis of temperature effects.

Table 24. Calibration constants, calculated from Figure 40, and used in a MATLAB Simulink program to correct the temperature.

	Temperature A_A (Volts)	**Temperature A_B (Volts)**	**Temperature B_A (Volts)**	**Temperature B_B (Volts)**
Slope	198.16	194.22	191.79	200.11
Intercept	-22.64	-22.00	-21.57	-22.82
r^2	1.000	0.999	0.998	0.999

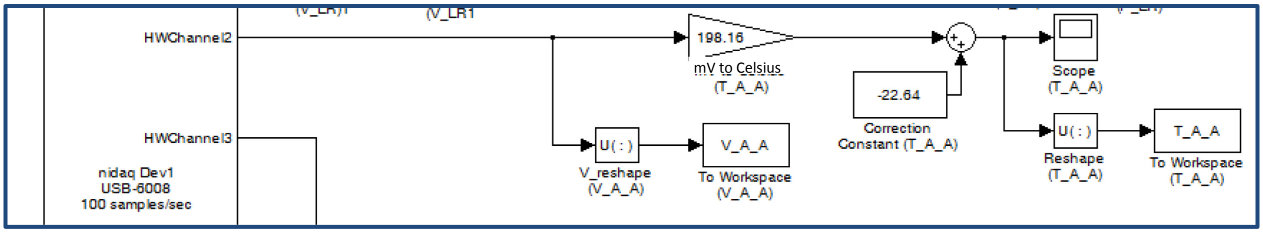


Figure 41. Example section of MATLAB Simulink diagram for converting the thermocouple A_A signal to Celsius. The voltage signal travels from HWChannel2, is multiplied by a constant and adjusted by a correction constant before being recorded by the MATLAB Simulink program.

4.1.4. Calibration of the Flow Rate Platform with Different Fluid Samples

The S Flow Unit flow sensor (Fluigent, Villejuif, France) measures the flow rate by adding a small amount of heat between two temperature sensors and measuring the temperature spread between the sensors [75]. The sensor requires calibration for fluids other than the factory calibration of water due to this heat method of measuring the flow rate. Following the calibration protocol from Fluigent [75], a scale factor for each fluid was created on the day of testing.

The flow sensor was connected to a Nexus 3000 syringe pump (Chemyx, Stafford, United States) and infused with a selected fluid at several flow rates within the platform's range. To obtain the scaling factor over the range of the sensor, 1, 3, 5, and 7 µL/min (which correspond to 14, 43, 71, and 100% of the positive range) were chosen as the intervals. A glass syringe (50 µL Hamilton Gastight) was connected to the sensor using rigid Teflon tubing (1.59 mm) and PEEK tubing (0.794 mm). Once the flow rate stabilized, the flow rate was measured and averaged over 1 to 2 minutes. Scale factors for each fluid were determined by plotting the syringe pump's flow rate, taken as the actual flow rate, against the sensor's measurement and calculating the slope, as shown in Figure 42. The scale factors were used to correct the measured flow rate for the different fluids to the correct rates.

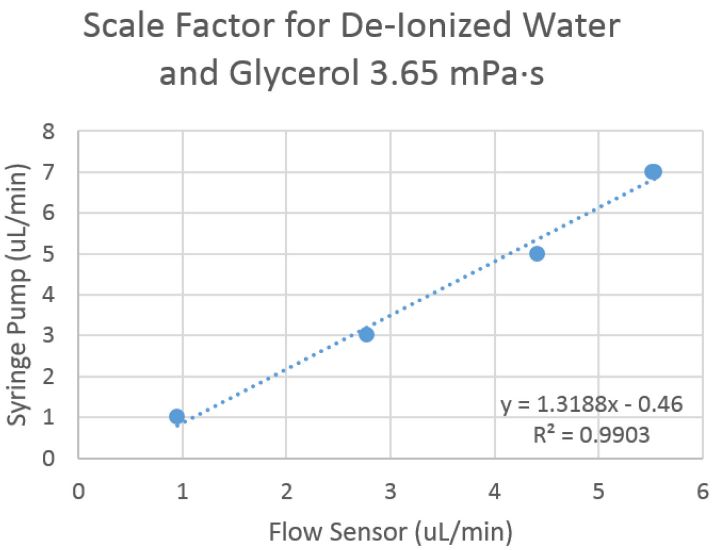


Figure 42. Example of the determination of fluid scale factor for de-ionized water and glycerol solution. The scale factor of 1.3188 was used to correct the reading from the flow sensor to the actual flow rate.

4.1.5. Calibration of the Reference Fluid Viscosities

At the current stage of the project, DI water, glycerol solutions, and polyox are the fluids which have been tested. Once the DI water, glycerol solutions, and polyox were prepared, their theoretical viscosities were determined with different methods, using a combination of theoretical and equipment measurements.

Calibrations were completed using the channel design H (width 40 µm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.

4.1.5.1. DI Water

For DI water, temperature and reference instrument, m-VROC (RheoSense, San Ramon, United States) were used. The system was first calibrated and tested with readily available Newtonian DI water, as its viscosity properties are well known within a range of temperatures [94]. The temperature reading from the thermocouples were used to estimate the viscosity of DI water. The thermocouples were placed next to the chip, instead of in the flow, so as to not disturb the flow and because the fluid was being tested at room temperature. M-VROC with sensor 14RA05100259 was used to compare the biomicroviscometer's capabilities at relatively low shear rates. These values were measured using mVROC_Control_v3.1.1 (RheoSense, San Ramon, United States) software to program the software in order to measure viscosities at different shear rates.

4.1.5.2. DI Water and Glycerol

The determination of glycerol solution viscosities was performed using a µVISC viscometer (RheoSense, San Ramon, United States).

4.1.5.3. Polyox

For the non-Newtonian fluid polyox, power law constants were calculated numerically. Viscosity measurements were performed on a µVISC viscometer (RheoSense, San Ramon, United States). The effect of shear rate on the viscosity was measured between 1700 s^{-1} and 110 s^{-1}, with smaller intervals at the lower shear rates. The power law constants were determined by plotting the natural logarithm of shear rate to the natural logarithm of the viscosity, as shown in Figure 43 and calculating the slope (n_{POLYOX} constant) and e raised to the y-intercept of the graph (K_{POLYOX}, the viscosity related constant). For this example, n_{POLYOX} = -0.1380 and K_{POLYOX} = 17.58.

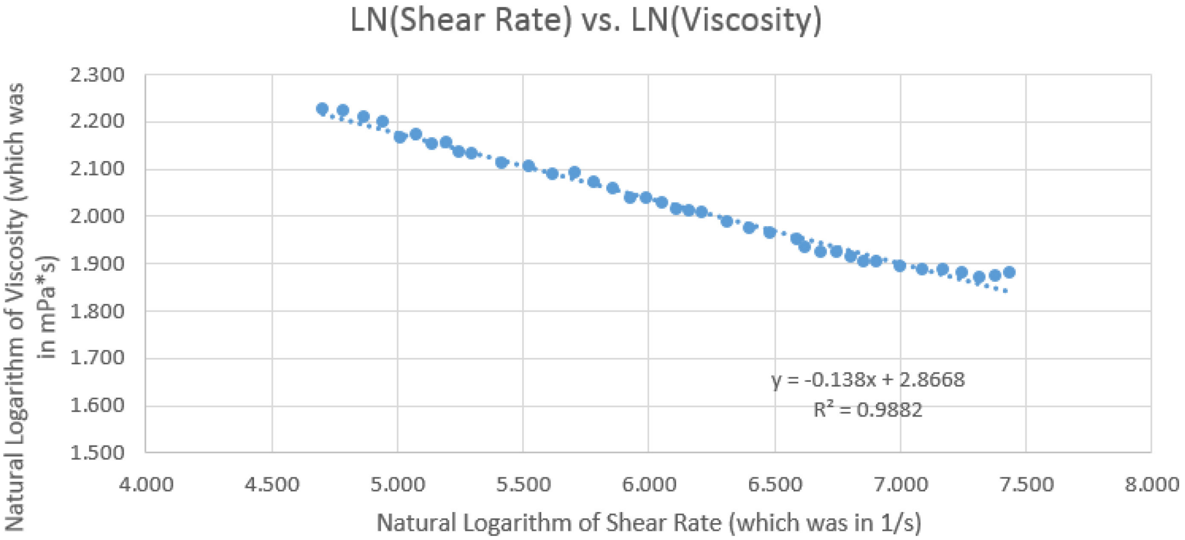


Figure 43. Log-log plot of viscosity vs shear rate to determine power law constants n_{POLYOX} and K_{POLYOX} using the slope and e^($y - intercept$), respectively.

4.2.Verification with De-Ionized Water

An initial verification of the experimental setup and measurement system, the viscosity of DI water was measured using the channel design H (width 40 μm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.

Plots, such as the one shown in Figure 44, were generated using the MATLAB GUI code. These plots indicate:

1. The input pressure (top left), in Pascals, as measured by the internal sensor of the MAESFLO pressure controller
2. The measured pressure (top right), in Pascals, as measured by the differential AllSensors pressure sensors located in the pressure and temperature sensor module
3. The flow rate (bottom left), in mPa·s, as measured by the Fluigent Flow Rate Platform flow sensor
4. The experimental apparent viscosity (bottom right), in mPa·s, as calculated using the measured pressure, flow rate and channel geometry as a blue line; and the theoretical viscosity, in mPa·s, as calculated using the temperature from the J2 temperature sensors as a red line

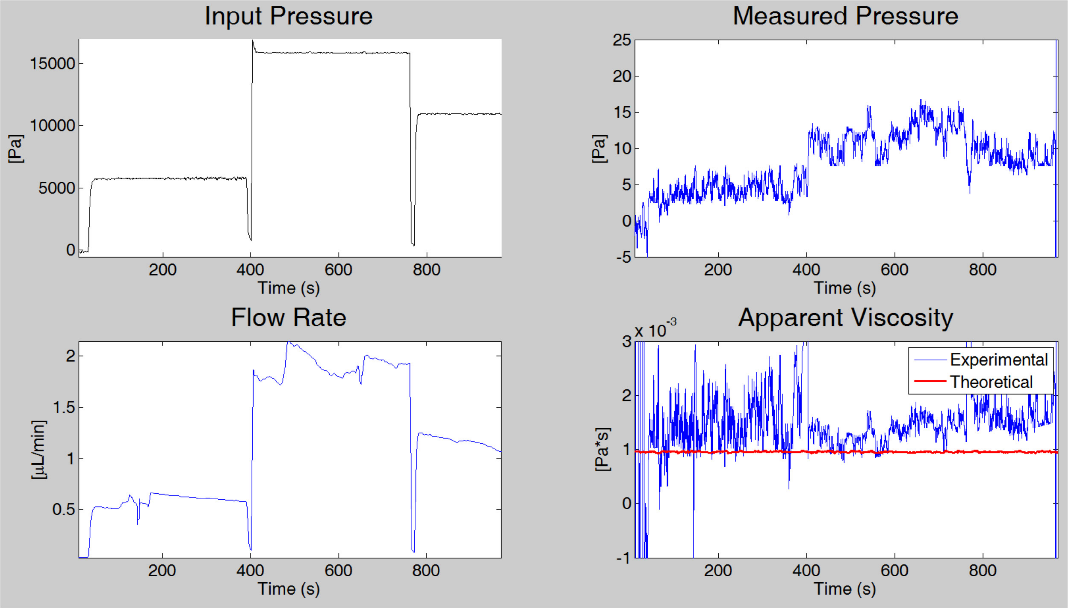


Figure 44. Output plots of measured values for a de-ionized water experiment (input pressure, measured pressure and flow rate) are used to calculate the apparent viscosity, which is compared to the theoretical viscosity. Measurements were performed using channel design H (width 40 μm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.

Input pressures were for the whole system, including the pressure difference required to flow the sample through the flow rate sensor. After each experiment was set up, input pressures were tested to see what range would produce equal intervals within the flow sensor range (e.g. which pressures would obtain 25%, 50%, 75% and 100% of the flow rate sensor range). This check was performed before each experiment, since differences in the component connections would change the resistance within the system. For example, the overall resistance was sometimes increased by the overtightening of certain threaded connections within the system, which would deform the internal diameter of rigid tubing slightly. These slight changes in experimental setup caused the input pressure to vary between experiments. The measured pressure and flow rate, however, were always within their sensor's ranges.

The experiments had a randomly assigned flow rate, at equal intervals within the sensor range, for specified time segments. MATLAB's "randperm(N,K)" function was used to randomize the range, which returned a random list of K numbers between 1 and N. This function was carried out a random number of times, where N = K and was the number of flow rate intervals being tested. Within each experiment, the time interval being used was the same for each different flow rate.

Early tests were carried out with an additional step in-between two input pressure intervals, where the input pressure was set to 0 Pa for 5 seconds, as can be seen in Figure 44. This step was used to gauge whether there was enough time for the fluid to get to a measured pressure of 0 Pa, and from the desired conditions from 0 Pa. A reset between the different pressures would mean that the system was at 0 Pa before there was a change in pressure. Time intervals of 5 or 6 minutes were chosen for the flow rates, to give the system time to get to steady state (due to any residual pressure effects) or check if there were any inherent response times related to the equipment. It was found that the 5-second segment at 0 Pa did not greatly affect the response compared to omitting it.

Analysis of the plots indicated that our system can measure relatively low flow rates of 7 µL/min and below, and pressures of approximately 130 Pa and below, as shown in Figure 44. Error ranges were also calculated for the case of measurement error in the channel dimensions of ±3 µm (for both the width and the height), as shown in Figure 45. This range was the measurement error in identifying the dimension of the channel due the inability to precisely know where the channel walls were in the images. The extreme cases occurred when the height and width had 3 µm added to each value, or when each of the height and width had 3 µm subtracted from the values. In addition to calculating the apparent viscosity of each sample with the average dimensions, the larger and smaller dimension values were also used. These apparent viscosities were then plotted as an example range of error for the apparent viscosity, if there was a ±3 µm error in the dimension measurements. Comparing both the apparent viscosity and estimated channel dimension error range to the theoretical line shows that there appeared to be agreement with the theoretical plotted line. Since the device was designed with water viscosity being at the lower level of our device's operating range, the observed noise was to be expected.

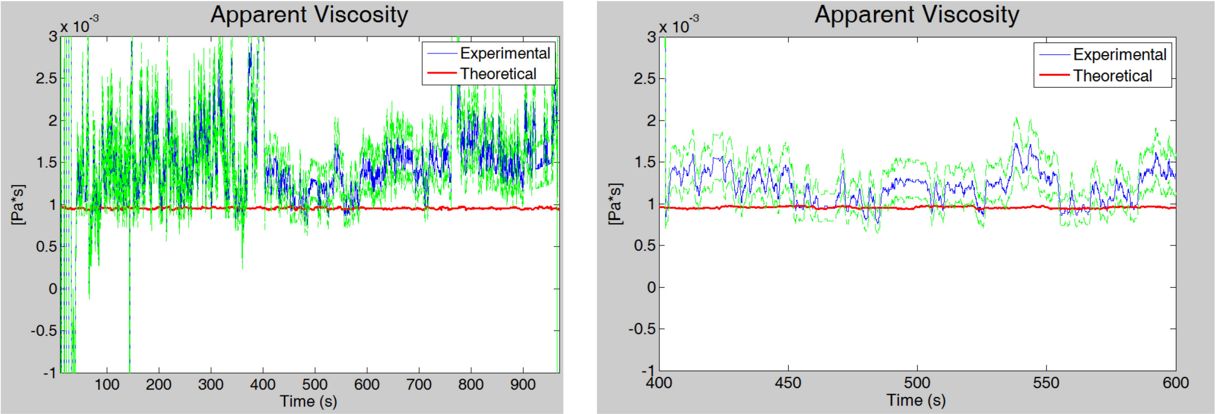


Figure 45. Apparent viscosity of DI water with the maximum error estimate for a ±3 µm error, indicated by the green dotted lines, in the height or width dimension for a whole experiment (left) and the segment from 400 seconds to 600 seconds only (right). Measurements were performed using channel design H (width 40 µm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.

4.3.Validation of Newtonian Fluid with Glycerol - DI Water Solutions

Glycerol solutions with viscosities of 3.65, 8.4 and 17.5 mPa·s were analyzed similarly to how DI water was, with different degrees of agreeability to the theoretical values, as shown in Figure 46. For comparison between the glycerol solutions, the percent error between the experimental and theoretical viscosities were calculated for each fluid and flow rate. Experimental data were removed from the plot, if it was clear that there was a bubble or blockage, though the data were kept for methodological analysis. These viscosity data were plotted against the shear rate, as shown in Figure 47, with the percent error based on the reference viscosity represented by the colour. The percent error ranges from 0% (dark green) to greater than 50% (red). Different fluid viscosities are represented by different marker shapes. Reference viscosities for each fluid are indicated on both plots as thin dotted lines.

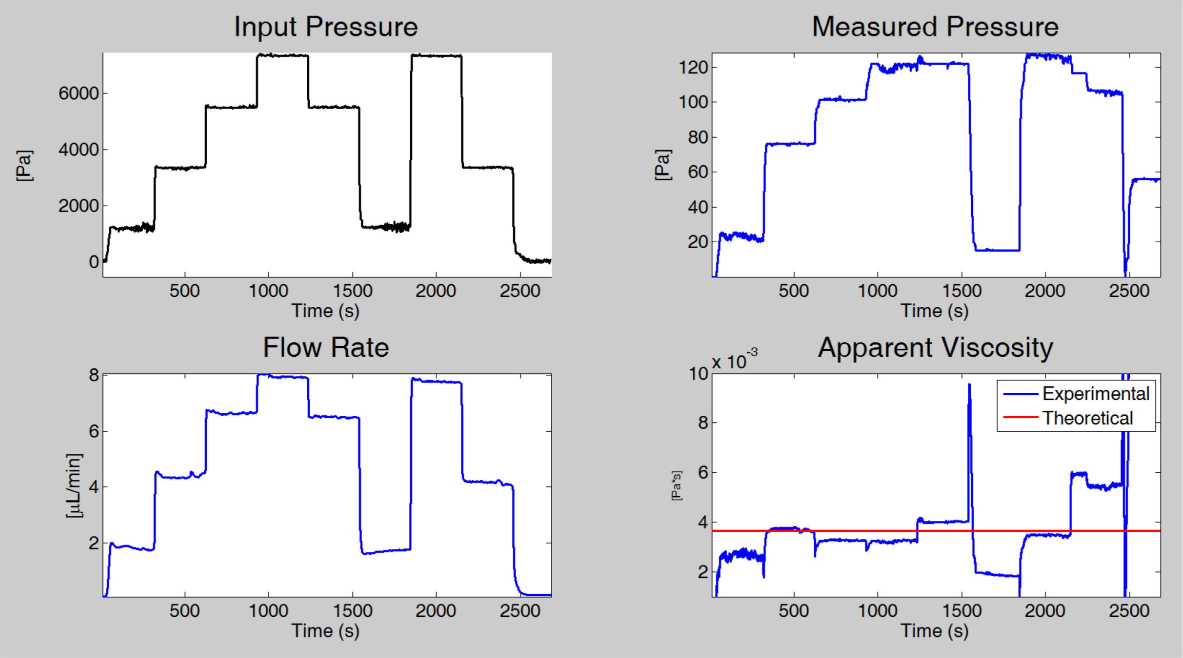


Figure 46. Output plots of measured values for a 3.65 mPa·s glycerol solution experiment (input pressure, measured pressure and flow rate) are used to calculate the apparent viscosity, which is compared to the equipment reference viscosity. Measurements were performed using channel design H (width 40 μm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.

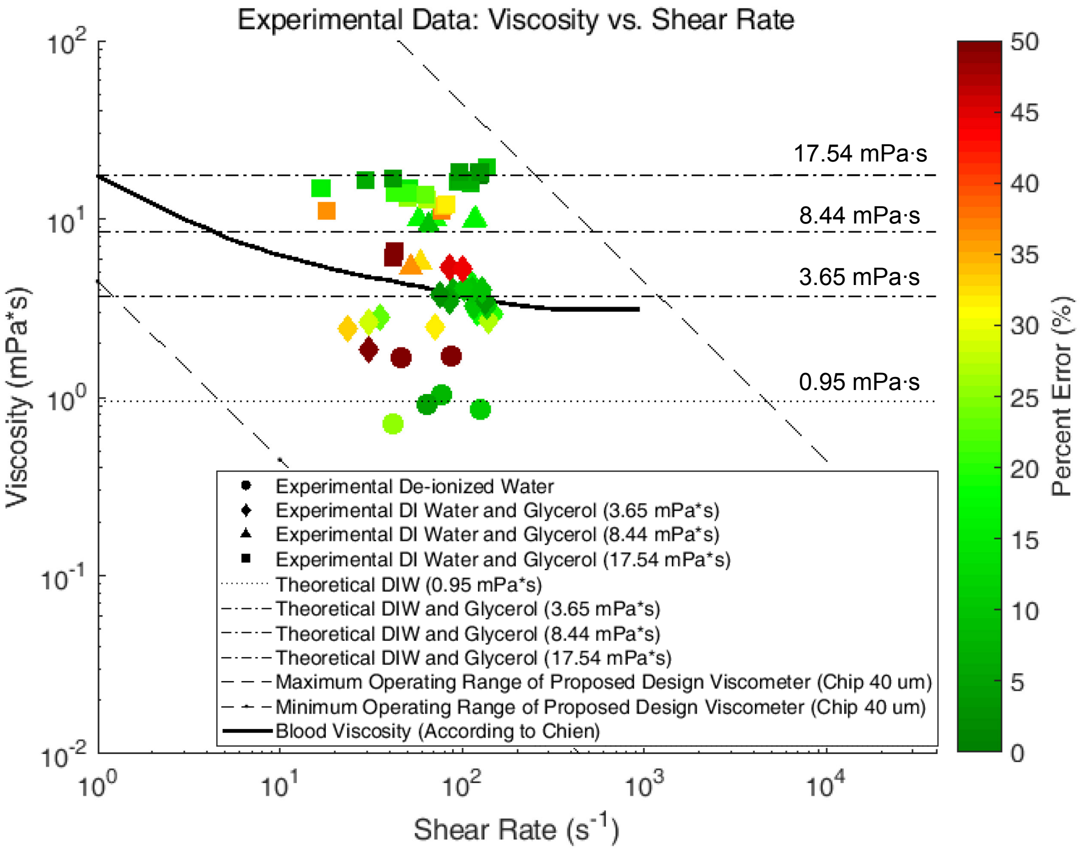


Figure 47. Plot of our optically clear biomicroviscometer measurements versus the shear rates, with colour representing the error. Horizontal dashed lines represent reference viscosities, with the marker shapes corresponding to a certain fluid viscosity. The operating range of our experimental design is shown as diagonal dashed lines. Blood viscosity at 45% hematocrit as reported by Chien [24] *is also included as a solid black line for comparison to the operating range of our device. Measurements were performed using channel design H (width 40 µm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.*

As an overview comparison between the commercial devices and the reference viscosities, Figure 47 and Figure 48 show the relationship between each measurement compared to its respective reference against the shear rate. To give a perspective on the influence in regard to the magnitude of the shear rate, the apparent viscosities and their percent error in comparison to the reference instruments is shown in Figure 48.

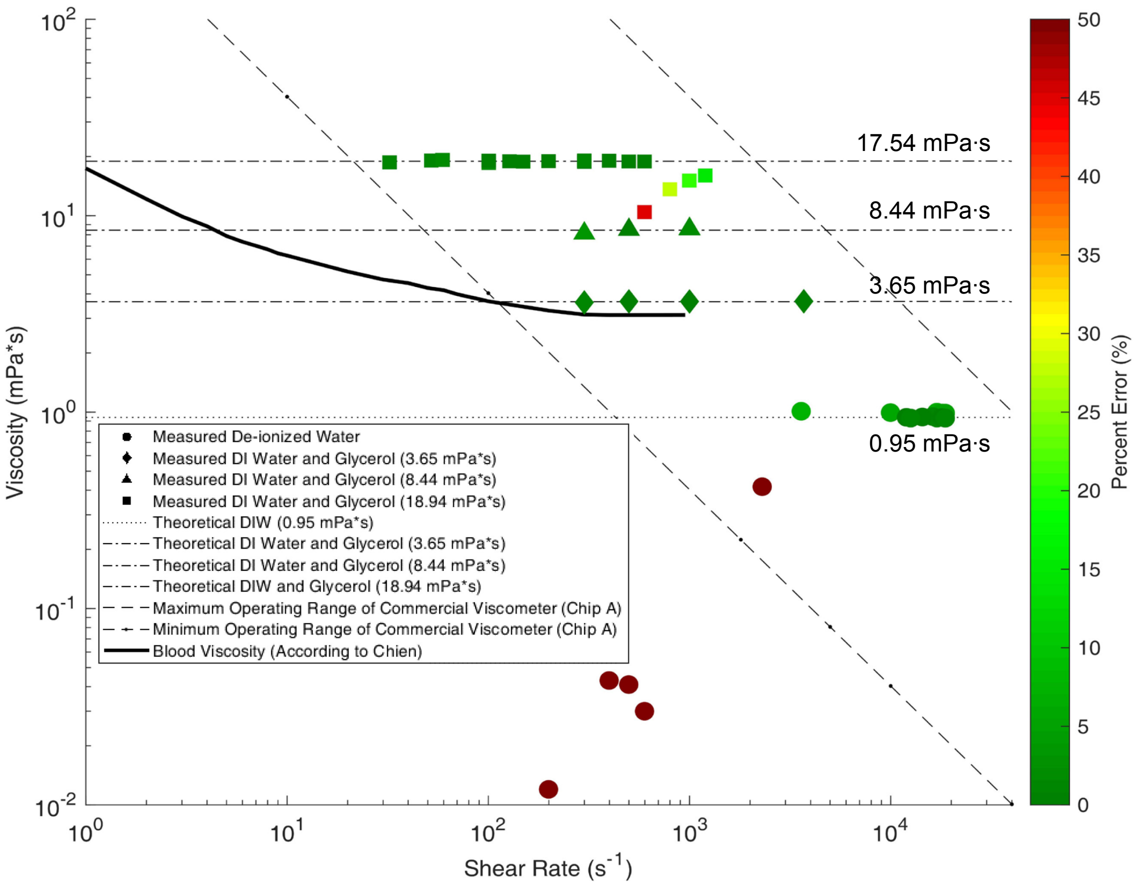


Figure 48. Log-log plot of the commercial viscometer's measurements versus the shear rates, with colour representing the error. Horizontal dashed lines represent reference viscosities, with the marker shapes corresponding to a certain fluid viscosity. The operating range of the commercial viscometer is shown as diagonal dashed lines. Blood viscosity at 45% hematocrit as reported by Chien [24] *is also included as a solid black line for comparison to the operating range of the commercial device.*

The commercial instrument was tested at higher shear rates, within the operating range of the device, which were higher than the experimental values. At shear rates lower than the operating range, the m-VROC software indicates that the shear rate value would not yield good results and these are demonstrated in Figure 48 with DI water under 500 s^{-1}. For the glycerol solution of 17.5 mPa·s, the commercial devices had less than 10% error approximately above 30 s^{-1}, whereas our device has less than 10% error above 100 s^{-1}. At 8.4 mPa·s, the commercial device has less than 10% error approximately above 30 s^{-1} compared to our 50 s^{-1}. At 3.65 mPa·s, the commercial devices have less than 10% error approximately above 300 s^{-1} compared to our 70 s^{-1}. For DI water at 1 mPa·s, the commercial devices have 10% error approximately above 3000 s^{-1} compared to our 60 s^{-1}. Initial results appear to show that our device can operate at the comparable shear rate range as commercial devices.

4.4.Validation with Non-Newtonian Fluid (Polyox)

The platform's capabilities were tested with a non-Newtonian fluid by using polyox with a 0.1% concentration. The theoretical curve was calculated differently than the Newtonian fluids, with further details in the Polyox Section 4.1.5.3. The power law constants were calculated from multiple measurements of apparent viscosity at different shear rates. The shear rates were estimated by first using the flow rate and the dimensions of the channel for a Newtonian fluid case ($\dot{\gamma_a}$) and then calculating the wall shear rate for the non-Newtonian power law fluid ($\dot{\gamma_w}$) as described by Equations (9) to (14). The theoretical apparent viscosity was then calculated using $\dot{\gamma_w}$, along with the power law constants $K_{power\ law} = 17.58$ and $n_{power\ law} = -0.1380$ as found in Polyox Section 4.1.5.3, using Equation (16).

The theoretical values were then calculated using these constants with the shear rate and plotted, as shown in Figure 49. Error limits, similar to the Newtonian fluid plots, using a ±3 µm error in the height or width dimension for a whole experiment were completed, as shown in Figure 50. Plots of the flow rate and measured pressure have quite agreeable plots, however there is a relatively constant difference between the experimental and theoretical plots. These initial tests indicate feasibility with non-Newtonian fluids with tuning of the theoretical estimates and estimates with the hydraulic resistances. However, further tests would be required to investigate these phenomena.

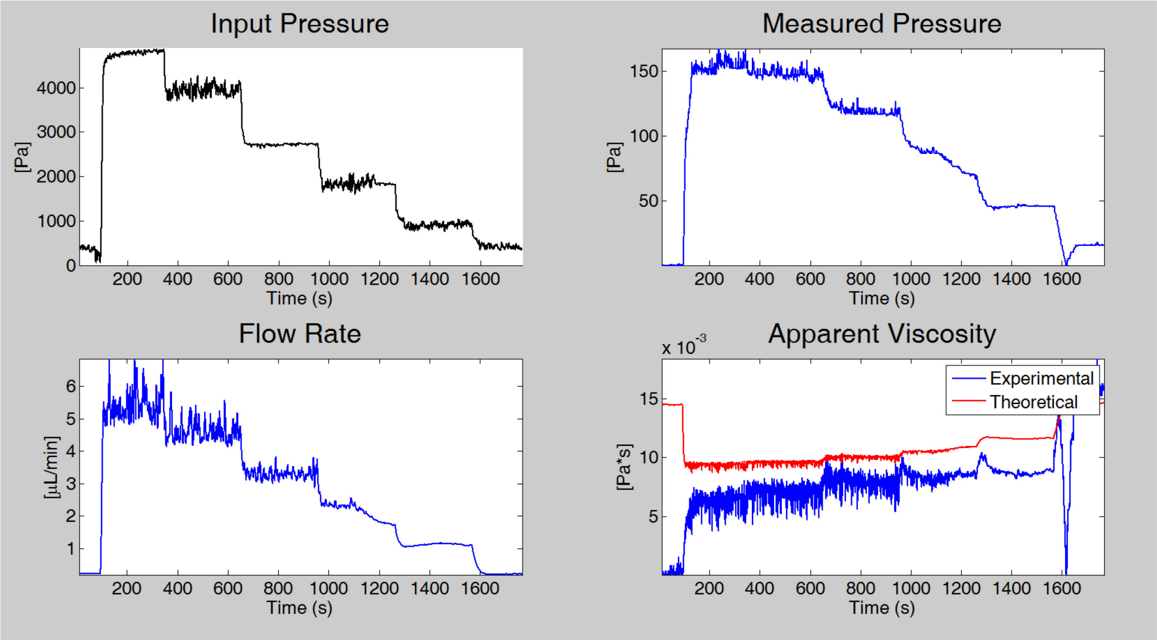


Figure 49. Output plots of measured values for a polyox experiment (input pressure, measured pressure and flow rate) are used to calculate the apparent viscosity, which is compared to the theoretical viscosity. Measurements were performed using channel design H (width 40 µm and 16 parallel channels*), see Figure 31 for location of chip on the wafer mould*.

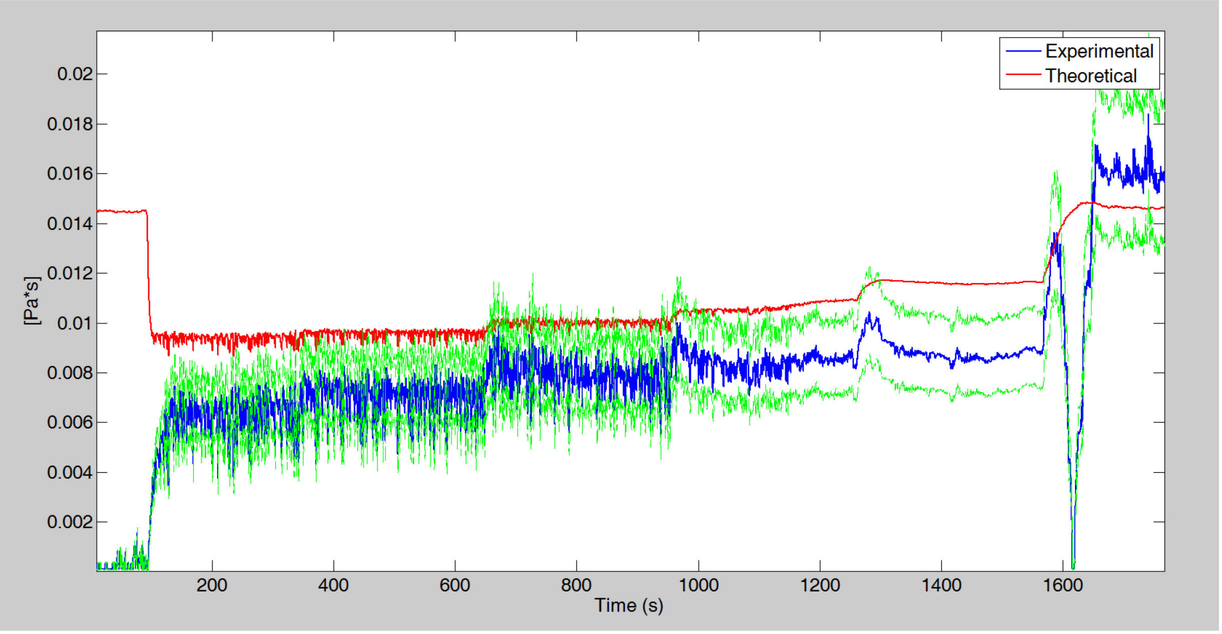


Figure 50. The experimentally determined apparent viscosity of polyox 0.1% (blue line), with the maximum error estimate for a ±3 µm error in the height or width dimension (green line) is compared to the theoretically calculated apparent viscosity. Measurements were performed using channel design H (width 40 µm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.

5. Discussion

The scope of the project was to design an optically clear viscometer, with a fast pressure response and the ability to be programmed to vary the applied shear rate. The aim was to measure the viscosity of blood, ideally at shear rates below 100 s^{-1}, to investigate the deformation, aggregation, and CFL effects. The use of PDMS as the microfluidic chip material allows for visual inspection of the flows, as indicated by visualizing different dyes in the system and by its common use in microfluidics for this purpose. A fast pressure response in the system was attained by using a pressure-driven flow, as demonstrated in the measurement plots of the sensors and the analysis plot of the apparent viscosity, as previously shown in Figure 44, Figure 46 and Figure 49. The shear rates can be controlled indirectly to occur at certain time intervals through programming scripts to vary the flow rate. Disposability of the microfluidic chip was also desired due to the biological nature of blood, which is realized in choosing PDMS due to its relatively low cost. At the current stage of the project, blood cells and blood plasma have not been tested. However, the microfluidic chips' operating ranges were designed to include the viscosity of blood at its various shear rates, with the ability to test fluids at low shear rates. The design of the experimental setup and methodology also allows for different geometries to be tested. Although it requires the resistance analysis of the new intended geometry, the system can test the network pressure and flow rate to investigate the rheology of fluids in various geometries. MATLAB programs to collect the data and analyze the data in a GUI interface were also successfully created for the project. The MATLAB Simulink program collected the signals from the pressure and temperature sensors, converted them to their respective measurement units using calibrated values and saved them in a readable format. The GUI collected the data files and analyzes them according to the fluid chosen and given geometry of the chip (which are programmed for each chip). The MATLAB GUI code can be modified to work with new geometries or fluids.

Improvements towards reducing the compliance and improving the channels' dimensions were also completed during the design of the experimental setup. The reduction of the system's relaxation time, from 197.17 seconds down to 18.4 seconds, gives the system the capability to investigate rheological properties before the RBCs settle since this settling occurs after 54 seconds. Modifications to the wafer processing technique for the clean room setup for our lab resulted in channels with walls which are more perpendicular. The perpendicular aspect of the channels can still be improved upon, since the channels with the smaller widths experienced larger percent width differences compared to channels with larger widths.

Calibration was performed for the pressure sensors, temperature sensors, fluid scale factors, and fluid viscosities. The calibration of the pressure sensors, temperature sensors, and fluid scale factors yielded values which were used to convert signals to calibrated measurements. The reference viscosities were important values with which to compare with experimental results. Apparent viscosities of Newtonian fluids (DI water and glycerol solutions), with viscosities of approximately 0.95, 3.65, 8.44, and 17.54 mPa·s, were measured using our device at shear rates between 20 to 200 s^{-1}. These same fluids were tested with the commercial device (m-VROC) at higher shear rates, due to the lower shear rates falling outside of the operating range of the commercial device. Although the measurements from our device are not precisely the reference viscosity, values were generally agreeable to the reference viscosities. Our device also has an operating range lower than that of the commercial one, allowing for testing of fluids with a lower viscosity at lower shear rates in addition to the ability to simultaneously investigate the rheological phenomena of blood. Testing with polyox indicated feasibility for the system to measure the apparent viscosity of non-Newtonian fluids. Further testing is required to understand if the difference between measurements and the theoretical value is due to fluid preparation or measurement technique. It is possible that the solution was not homogenous in viscosity due to the complexity of the fluid preparation steps and the tendency for polyox to have wetted particles that may never dissolve [96]. Another possibility to test is the possibility for these wetted particles to settle over time and create a vertical density gradient within the solution and cause the top or bottom of the sample vial to have a different viscosity.

Limitations exist with the experimental setup and measurements, such as the measurement of the channel cross-section’s dimensions and the occurrence of blockages in the microchannels. Errors affecting the measurement of the apparent viscosity within the system, their estimates, and the methods of mitigating the errors, are further discussed with respect to the microchannel dimensions and pressure/flow rate measurements. Several inaccuracies inherent in the system due to equipment precisions were also present and estimated. The error in the calculations of viscosity due to microchannel dimensions was investigated. Error was also investigated with respect to the pressure and flow as a percent change in calculated value due to these errors and the attempt to mitigate them. The change in calculated value due to just measurement errors was calculated using the results from the glycerol solution with a viscosity of approximately 3.6 mPa·s.

Demonstrating verification and validation, the next stage of the design is to improve upon the design before using fluids containing blood cells. The use of μPIV will then be discussed with regard to understanding the three-dimensional velocity profile with the intent to increase the ability to calculate the flow and shear rate within the microchannels.

5.1.Limitations and Errors

5.1.1. Cross-Sectional Shape of Channels

The microfluidic channels in the chip are tapered in shape, which can affect the flow and velocity profile. The tapered cross-section is due to the fabrication process. The flow path is especially important when using fluids containing RBCs, since the interaction between particles and channels are more complex. The fabrication techniques can be improved in order to ensure that the walls are perpendicular. Techniques could be applied for future creations of wafers to ensure better quality of channels, such as the modification of the wafer mould or the creation of a more precise mould.

Modification of the wafer mould surface to release the PDMS better could be done using a silicon wafer passivated with a silanizing agent, or by using a mould made out of PDMS [60], [80]. Having a silanizing agent (which contains a silane) as a monolayer on a wafer facilitates the removal of PDMS from the SU-8 on the silicon wafer mould. These silanes require the use of a fume hood and are toxic, being corrosive and extremely reactive with water [101]. Creation of a more precise mould could be completed using further improvements on the wafer mould fabrication. Another analysis of changes to the fabrication process could be completed with small variations to the process. Since the amount of tapering in the channel was not severe, as is shown in Figure 51, the flow path or velocity profile is expected to not be greatly affected, but should be experimentally confirmed with rectangular channels.

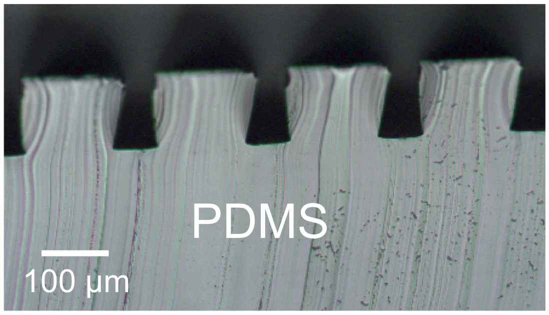


Figure 51. Cross-section of the parallel channels for Chip H (40 µm with 16 channels) at 10× magnification with PDMS material as the light grey material, see Figure 31 for location of chip on the wafer mould.

The range of confidence in the measurements of the viscosity and error limits is based on a dimension measurement error of ±3 µm. These errors and their influence on the apparent viscosity depend on the relative dimensions of the channels and were, therefore, calculated using the MATLAB program. Estimates of the microchannel dimension effects on the apparent viscosity for the glycerol solution were calculated as a percentage of the measured value to be 18.2% at the maximum value (between 0.33 mPa·s and 0.735 mPa·s difference) and 16.3% at the minimum value (between 0.3 mPa·s and 0.65 mPa·s difference), as shown in Figure 52. The experiments were carried out with one width (nominally 40 µm for the parallel channels). It would be expected that this error would have a greater effect on the 20 µm width channels due to the relative error and the high dependence on the smallest dimension. To reduce the amount of measurement error, the measurements were completed by one researcher and were taken in the middle of the visible edge and the blurred edge. Digital images of all the measured channels were also kept for improving upon the technique or refining the dimensions. Further estimate of the possible error can be approximated using multiple measurements from different researchers to obtain an average variance.

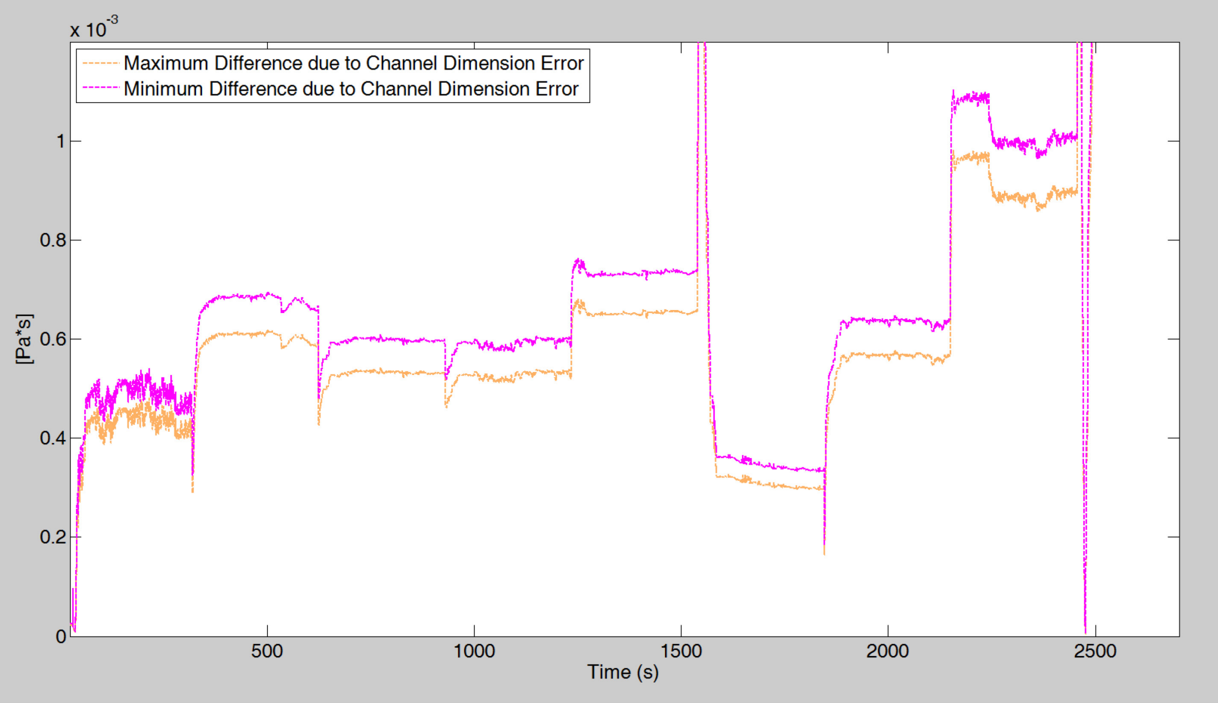


Figure 52. Minimum and maximum difference in the apparent viscosity estimate with a dimension measurement error of ±3 µm for the DI water and glycerol solution with a theoretical viscosity of 3.65 mPa·s example in Section 4.3. Measurements were performed using channel design H (width 40 µm and 16 parallel channels), see Figure 31 for location of chip on the wafer mould.

5.1.2. Blockages of Microchannels

The flow sensor and microchannels are susceptible to blockages throughout an experiment. These blockages, whether they are partial or complete, can affect the flow rate or path within a microchip. Blockages from dust or fibres can come from the surrounding connecting components or from the preparation of the fluid. The possibility of these particulates entering the channels was decreased by cleaning the connecting components before their use (both inside and outer surfaces) and storing them in closed containers between experiments. Fluid preparation is completed within fume hoods where possible and the goal is to ensure that the measurement tools are clean and that containers are closed as soon as possible.

Blockages within the flow channels can be checked using a microscope and high-speed camera while an experiment is running. Since the pressure drop matters only for the channels under investigation, it matters less for the connective tubing and connections – as long as they do not dislodge and enter the channels which are under investigation. For components which cannot be easily checked, such as the flow sensor, it is critical for these components to be cleaned between experiments and to check whether the flow seems to be blocked by testing a reference fluid. The cleaning process for blood, as an extreme case for blockages, ensures that no blood components are left within the flow sensor.

5.1.3. Different Temperature Scale Factors for Flow Rate Sensor

Each different fluid requires calibration with the flow sensor to accurately measure the flow rate, since the sensor operates by introducing a small quantity of heat into the fluid. Even the same formulation of a liquid requires a new calibrated scale factor if it may introduce differences in the thermal conductivity. For complex fluids such as blood, the scale factor might be different within the fluid at different times of the day as more time passes since the blood sample was collected. As such, an experiment to test the scale factor of different hematocrit over several intervals of time would be necessary. These tests would give insight towards whether the hematocrit, plasma and/or sample age influences the flow rate measurement. Since the fluids which have already been tested do not have a complex thermal conductivity, this limitation is not yet apparent.

Even though there might be subtle variations between the sides of each chip, the chips were carried out at the temperature of the ambient room conditions. The sample fluids were allowed to equilibrate to room temperature before using them in the experiments. The temperature was measured throughout each experiment with the thermocouples placed next to the chip. During the experiments, the temperature sensors were not placed within the fluid itself, since they have the possibility to disturb the fluidic flow. In future versions of the experiments, the temperature of the chip should be measured on the chip using holes placed next to the channels, but not within them. They would be sufficiently close to the flow to measure the temperature, but not cause deformation of the channel walls by being placed down. Once experiments are to be carried out using a heated (or cooled) sample fluid and/or chip, a distribution of temperature across the chip should be completed. A correlation formula would allow the temperature in certain channels to be calculated from the measurement(s) of the temperature provided by sensors in specified locations.

5.1.4. Settling Time for RBCs

Blood cells start settling as soon as there is insufficient force or motion to keep them suspended. The amount of settling and the speed of settling depends on the dimensions of the channel in comparison to the cells as well as the fluid properties around the cells. As in the Determining

Equilibrium and Reduction of Compliance Section 2.5.3, the settling time was estimated to be approximately 54 seconds.

This effect is difficult to mitigate since gravity is the force causing the cells to settle. Ideally, the cells can be resuspended passively within channels instead of an external active mixing. These changes need to be gentle enough for the cells not to be damaged, but also not greatly affect the flow of the fluid through the microchannels. Even though the RBCs would be resuspended (for fluid samples containing these cells) by the magnetic stir bar within the sample container, there would not be a resuspension method within the channels themselves. If the time taken for the sample to travel through channels exceeds 54 seconds, with the fluid flow being perpendicular to the force of gravity, features would be necessary to ensure proper resuspension of the RBCs. The settling within channels could be mitigated by placing the microfluidic chip in a vertical orientation, with fluidic connections also being vertical. This orientation would cause the force due to gravity and the pressure gradient to act in the same direction. This arrangement is not optimal for our experimental setup due to the limitation of our microscope, which can currently only be used for horizontal samples.

5.1.5. Pressure Controller Delay and Viscosity Measurement Error

During the starting steps of the pressure controller introducing pressure to the sample, there is often a delay if the pressure is set to a low value. For these low values of the pressure input, it can take a few seconds for the fluid to start moving due to the pressure. This extra time could allow a fluid, like blood, to have its cells settle within the channels or tubing, since the fluid is not moving. It was seen that priming a different fluid ahead of time reduced this effect and the system would allow the pressure controller software enough time to change the pressure to the system with uniform change throughout the experiment.

Both the pressure and flow rate sensors were used in their calibrated states, or recalibrated as needed. The differential pressure sensors have an estimate error of approximately 0.5% of the full scale, which equates to approximately 6.225 Pa [76]. MATLAB was used to estimate the percentage difference in apparent viscosity of the calculated value compared to the theoretical value, which gave an error between approximately 8.15% (0.308 mPa·s) for the higher pressure, and 5.11% (0.168 mPa·s) for the lower pressure, as both the minimum and maximum for the DI water and glycerol solution with a theoretical viscosity of 3.65 mPa·s example in Section 4.3. For tests with lower shear rates and pressures, this percentage is expected to increase if the measured pressure value is relatively close to the accuracy.

The effect of the flow sensor's accuracy on the apparent viscosity was estimated using an error of ±5% of the measured value [75]. Using MATLAB to calculate the error and difference in apparent viscosity measurement gave approximately 4.76% minimum error (between 0.191 mPa·s and 0.088 mPa·s difference) and a 5.26% maximum error (between 0.212 mPa·s and 0.101 mPa·s difference), as shown in Figure 53 for the DI water and glycerol solution with a theoretical viscosity of 3.65 mPa·s example in Section 4.3.

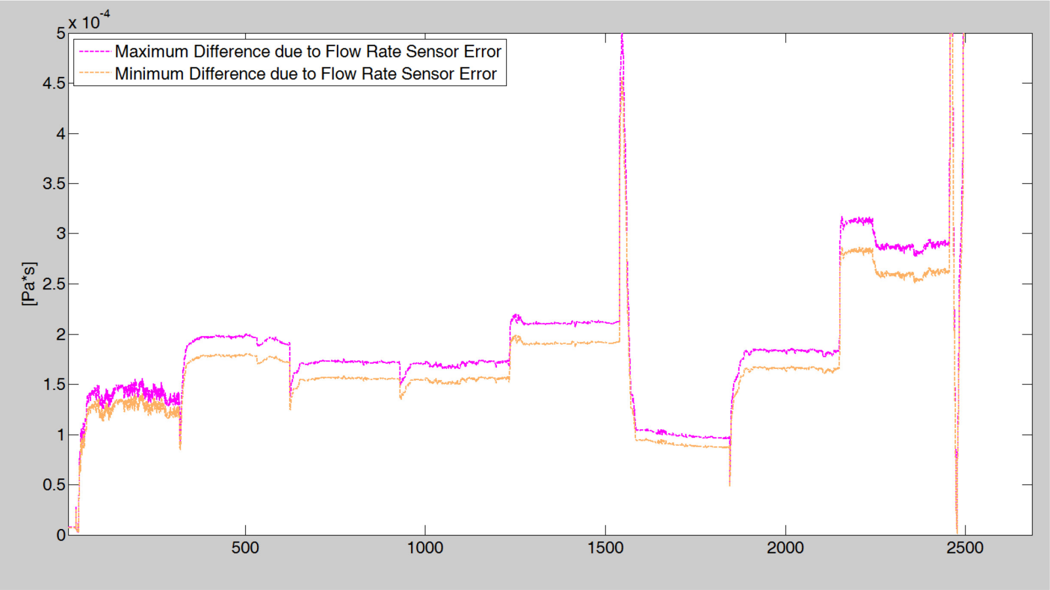


Figure 53. Minimum and maximum difference in the apparent viscosity estimate with a flow rate sensor error of ±5% for the DI water and glycerol solution with a theoretical viscosity of 3.65 mPa·s example in Section 4.3.

To mitigate these errors, the measurements were carried out at higher pressures and flow rates as possible. Initial tests used fluids with a viscosity greater than water, which helped in mitigating the errors since these require higher pressure to flow through the channels. The use of parallel channels to obtain a fraction of the flow rate within the channels also helped mitigate the need to use very low flow rates.

5.2. Discussion of Limitations and Errors

It is of interest to measure at the lower shear rates, since the aggregation effects within blood start to show below about 150 s^{-1} [102]. Aggregation of the RBCs in the blood causes the blood viscosity to increase. While improvements need to be realized regarding the accuracy of the platform at lower shear rates, initial results show that the device is capable of measuring at these lower shear rates to examine blood viscosity.

Response time between the fluid flow and the pressure change could be reduced by decreasing the compliance in the platform. Improvements have been made by reducing the amount of flexible tubing and trapped air located between the pressure sensors and the chip. A reduction of trapped air bubbles in the channels was completed by exposing the fabricated chip to oxygen plasma. A temporary hydrophilic surface was created in the channels by completing a 5-minute plasma treatment [93].

All fluids at the micro scale have many inherent issues and behaviours such as bottlenecking and residence time [65] as well as entrance effects [86]. The placement of the sensors in the system took consideration of these effects, but each new geometry introduced into the setup will require verification. A frequently applied method for measuring fluid flow properties in microfluidic channels is µPIV. Previous research using PDMS channels and µPIV with blood dilutions, without pressure sensors, found that a wider, rectangular cross-section was easier to visualize blood flows in the µPIV system. Literature revealed that as long as the width to height ratio was kept under 10, a rectangular solution could be achieved and slit-flow could be avoided [62], [65]. Our current designs are under this width to height ratio, ranging from a ratio of 1 to 5, and would avoid slit-flow which could give structural weakness and give flows that are unrepresentative of that which is found in the microcirculation. In addition, to apply standard rheological calculations to micro scale measurements, the 2D Weissenberg-Rabinowitsch correction can be applied to find the wall shear rate for a fluid with shear dependent viscosity (i.e. non-Newtonian flows) [103].

The microfluidic chip has the capability of being used with these technologies as it currently stands. The current experimental setup would require rearrangement to fit within the same box as the high-speed microscope or µPIV setup, to reduce the amount of tubing required to and from the chip. For applications with µPIV, it is important to keep the device contained within a box for safety reasons and to avoid ambient light from affecting measurements. Especially for this box, automation of the fluid changes and proper sample mixing are favourable for the setup. A platform to hold the chip and any required tubing or equipment would be ideal and should be implemented in the next prototype to ensure stability of the microfluidic chip, measurement devices and any fluidic connections.

Programs have already been created to detect the CFL thickness, using image processing software by our research group, and also count the size and number of aggregates [44]. A time synchronization program would need to be created to correlate the CFL thickness and RBC aggregation values to the apparent viscosity. Having a real-time program to see the flow and these values would prove helpful in correlating the values to physiological or fluidic phenomena. By incorporating MEMS into such a transparent channel, full rheological understanding of microcirculation could finally be achieved.

The biomicroviscometer was designed to eventually be coupled with a µPIV system. Using µPIV, a flow profile can be estimated by taking velocity profiles at the centre plane within the flow geometry [104]. Comparing the experimental flow profile to a theoretical 3D solution, a 3D experimental profile can be computationally extrapolated [105]. By directly measuring the pressure drop with the platform, the viscosity can be determined experimentally and the accuracy of the flow rate measurement can be significantly increased.

To our knowledge, fabrication and validation of a transparent biomicroviscometer for eventual use with a µPIV system for blood research has not been done before. Such a design is a necessary ‘missing link’ in current microcirculation literature and study. By incorporating MEMS into a transparent channel, full rheological understanding of microcirculation will finally be achieved.

5.3.Next Prototype to Implement

5.3.1. Modular Geometry and Mimicking Physiological Channels

The design of chips with different width parallel channels was not tested extensively. Testing the other chips should be completed after testing the experimental setup with the fluids containing RBCs. As is intended in the design of the chip, analysing with different widths of the parallel channels would allow the investigation of the CFL. Checking the precision and accuracy of the wafer moulds could be completed by measuring the features using a confocal displacement sensor such as the Acuity CCS Prima MG210 (Schmitt Industries, Inc., Portland, United States). These feature height measurements could ensure that the topographical geometry of the wafer is understood and correlated to the PDMS channels. One drawback to such displacement sensors is that they use light to measure distance to a surface, but they cannot measure features hidden under an overhanging edge.

The modular nature of the experimental setup means that PDMS chips with other geometries can be readily obtained. In the current design, one change could be the pillar arrangement dimensions and shapes. The pillar diameters could be changed for easier removal from the wafer. Centre-to-centre distances could be reduced to increase the shear rate of the fluid flowing in-between. Pillar arrangement could be changed from a square pattern to a hexagonal pattern for uniform distances between adjacent pillars with an expected fluid flow [64]. In addition to the microchannels, a branching network of straight channels could be used to ensure the same velocity in all parallel channels with verification using Computational Fluid Dynamics (CFD) software [58].

For simple geometries, an equivalent theoretical hydraulic resistance can be calculated for comparison. A next step for a change in the geometry could be the use of round channels instead of rectangular ones. Techniques to create round channels in PDMS, such as reheating the wafer mould to reflow the raised polymer features [57], could be used. For more complex 3D geometries, such as those found in the microcirculation, a dissolvable 3D printed mould could be used.

Other techniques to attain better cross-section shapes are possible using more rigid materials for the microfluidic channels, such as polymethylmethacrylate (PMMA) or Norland Optical Adhesive 60 (NOA60, Norland Products, Cranbury, United States) [106], [107]. In the case of NOA60, it has been demonstrated to be transparent, UV light curable and micro-machinable to create microfluidic features [106], [107]. Capabilities to machine micro-features in the more rigid material and stack layers would allow for more complex geometries on different levels. Using NOA60 would ensure rigid channel walls for more precise cross-sectional shapes and reduce the compliance, in comparison to PDMS channels.

5.3.2. Automatic Change of Fluid Samples

It would be of interest to automate the experimental process and test sequential fluid samples with the same chip using a script. The use of a M-Switch fluidic sample switcher (Fluigent, Villejuif, France) and ESS Control software (Fluigent, Villejuif, France) have both been successfully tested with the SciTE scripting software (Fluigent, Villejuif, France). A potential series of fluids would include the priming fluid first and then have a series of experiments with blood, each with a fluid flush in-between different experimental tests. Having the fluids in a sequence allows the fluid flows to be analysed in the same geometry without having to disconnect and reconnect tubing. The disconnection and reconnection require the flushing out of any air bubbles that are created, which is the step that takes the longest to complete during experiments. In addition to reducing the preparation time, flushing out air bubbles would reduce the pressure controller delay and compliance of the system.

An important aspect is that the M-Switch requires extra tubing from the sample to the microfluidic chip. Samples which are of limited volumes, like human blood samples, may see some of the fluid relegated to being part of a priming volume. The effect of extra tubing can be reduced by using the minimum length of tubing possible for the setup and tubing with a small diameter (e.g. PEEK tubing with 0.794 mm O.D.). The extra tubing also means that particles (such as blood cells) in the samples may settle during testing, requiring a different mixing technique in the experimental setup. Using minimum tubing between the M-switch and the chip will reduce the settling time for RBCs.

5.3.3. Wet-Wet Pressure Measurement Interface

A main source of compliance in the system, which increases the time for the system to reach steady state, was in the connection between the pressure sensors and the chip's ports labelled E and F. Since air was used as a medium to measure the fluid pressure, the volume of air between the sensor and the fluid introduced compliance within the system. The sensors used in the design were used for their precision, but could be improved upon by using a fluid interface instead of an air interface.

Sensors are available which allow for fluids, in addition to gases, to come into contact with the measuring diaphragm. The use of such a 'wet-wet' measurement interface would reduce compliance, which would reduce the time to steady state, in addition to reducing any possible settling that the fluid might exhibit. These sensors are capable of measuring at a low-pressure range just like the ones used in our design. We identify that the DP15 Variable Reluctance Pressure Sensor (Validyne Engineering, Northridge, United States) is capable of measuring in the ± 552 Pa (2.22 inH2O) range.

5.3.4. Heated Chip with Feedback Control Loop

The incorporation of heating components to the chip, with a computer controlled feedback loop, would allow analysis of the effect of temperature on fluids. Temperature sensors would be designed to be located as close to the flow as possible. The sensors would be sufficiently close to measure the same temperature as the fluid but not disturb the flow. Infrared cameras or additional temperature sensors could be used to calculate the difference in temperature between the fluid and measurement location for correction.

This would be especially important for the analysis of blood at body temperature, which is approximately 37 °C. The speed of aggregation of RBCs has been shown to be reduced with a reduction in temperature from 30 °C to 16 °C [22], which is where room temperature typically falls within. Analysis of measurements as a function of temperature and shear rate could be done to test their effects on the apparent viscosity.

For fluids which have known viscosities at certain temperatures, the temperature could be used to directly compute the theoretical viscosity of the sample fluid. Calibration of sample fluids with reference instruments would need to be done first to ensure they follow the theoretical formulae. For instance, the viscosity of a glycerol solution follows a theoretical formula, as described by Cheng [95], which can be used to determine the ratio of glycerol and DI water required to mimic the viscosity of blood at body temperature.

5.3.5. Fluidic Testing with Blood Analogue and Blood at Specific Hematocrits

In addition to doing the preliminary tests with porcine blood, human blood would also be tested. Aspects of testing with RBC samples are the differences between donor samples on the experimental tests and the hematocrit of the sample. The blood samples would be cleaned, as described below, and reconstituted at specific hematocrits for the samples. Using samples with a specific hematocrit ensures that the ratio of the RBCs to the plasma would be similar between experiments. The similarity of scale factors between samples of the same hematocrit would be investigated. This investigation would check if the scale factor for a specific hematocrit is the same between different donors.

To obtain different hematocrits, the blood would be separated into its RBC and plasma constituents with the following procedure. The blood would be collected into a container with anticoagulants. 2 g of EDTA is measured out into the collection container and suspended in 2 g of water for the case of porcine blood. For human blood samples, the blood collection vacuum tubes would already be coated. For both cases, the blood would be inverted several times to mix in the anticoagulant.

The blood would then be centrifuged at 3000 rpm for 10 minutes before the plasma is collected into a separate container. The buffy coat layer is then disposed of, before resuspending the RBCs in PBS. PBS is used to clean the RBCs to remove any leftover buffy coat. The RBCs and PBS are centrifuged again at 3000 rpm for 10 minutes, before disposing of the PBS and buffy coat. The RBCs are cleaned once more using PBS, with any remaining PBS and buffy coat being discarded. The RBCs and plasma can then be reconstituted into different hematocrit samples. For the case of the sample fluid containing Dextran and RBCs, the aggregation rate would be investigated by varying the amount of Dextran.

An important change for the experimental test procedure is the cleaning of the flow sensor. Since the flow sensor is sensitive and difficult to access the internal components, care is necessary to ensure the components are not damaged. RBC components, mineral deposits, or thin films can be left inside the chip if it is not properly cleaned. For future tests with a smaller flow sensor (XS Flow Sensor by Fluigent), it is crucial that the 25 μm diameter capillary tube in the sensor is not blocked.

A series of fluids would be used to flush out and clean the flow sensor after an experiment. First a few mL of PBS solution would be flowed through the flow sensor to remove as much blood components as possible. Since PBS is an isotonic solution, the RBCs would not hemolyze and would not leave cell fragments. This is followed by using a few mL of a 10% bleach solution to decontaminate the flow sensor from biological material. A few mL of DI water is then used to remove the bleach solution. Urine blood test strips (URS-1B, LW Scientific, Inc., Lawrenceville, United States) are used to test drops of liquid for haemoglobin traces coming out of the outlet tubing (not in the waste container). DI water is continuously used to flush the sensor until there is no trace of haemoglobin in order to ensure no blood components are left in the channel. Isopropyl alcohol is lastly used since it can clean out any biological films and it evaporates without leaving any mineral deposits.

6. Conclusion

Experimental results, thus far, indicate feasibility to control and measure the pressure and flow rate using the software. An optically clear rheometer was successfully designed with capabilities to measure the effect of hematocrit and channel width on the viscosity at low shear rates using the microfluidic designs. PDMS was chosen as the fabrication material for the microfluidic chip due to the ability to visually inspect the flow, the ease of fabrication and disposability. At the current stage of the project, blood cells and blood plasma have not been tested. However, the microfluidic chips' operating ranges includes the range of blood viscosity at its various shear rates, with the ability to test fluids at low shear rates.

Our viscosity measurement platform design combines optical analysis with the known accuracy of micro scale pressure sensors. In addition, the use of small, contained sample sizes is advantageous for biological samples such as blood. The design of the experimental setup and methodology enables other geometries to be analyzed with different fluid viscosities and properties through modification of a MATLAB GUI. Improvements in the design were completed, such as the reduction of compliance in the system and creation of more perpendicular walls. Calibration was performed for the pressure sensors, temperature sensors, fluid scale factors, and fluid viscosities. Apparent viscosities of DI water and glycerol solutions were measured using our device at shear rates between 20 to 200 s^{-1} with agreeable, but not exact, results compared to that of the reference. It was found that our device, compared to a commercial device, has a lower operating range, a better capability to analyze the apparent blood viscosity at lower shear rates, and the ability to visually analyze blood flow. Preliminary results of testing with a non-Newtonian fluid (polyox) indicated feasibility of measuring the apparent viscosity of the power law fluid, with further testing required to investigate the effect of fluid preparation on the viscosity.

This research contributes to the knowledge of the fluid flow in channels comparable to human microcirculation and the characterization of the rheology of blood at low shear rates. This research can lead to the investigation of more complicated microchannel networks, similar to the 3D structures found in the human body. The parameters and model can also be used for the validation of computational models of microcirculation, and to help biomedical engineers designing lab-on-a-chip devices. Future developments resulting from this research might lead to comparisons between healthy and diseased blood flows. One application is to characterize the rheology of blood in microcirculation with hyperaggregation or high hematocrit conditions. Another application is the investigation of how a drug delivery capsule would travel within blood vessels, for the vectorization and delivery of localized drug substances. They will additionally be able to characterize the rheology of blood in microcirculation with hyperaggregation (inflammation) or high hematocrit conditions.

The next design prototype being created for this biomicroviscometer will use a more rigid channel material, such as NOA60, to obtain precise channel cross-sections and reduce compliance. Blood samples of specified hematocrit will be used to calculate scale factors for the flow sensor and variances between samples. An automatic fluid sample changer will allow for priming and experimental tests to run with reduced settling time. The time to reach steady state, and thus compliance, will be reduced through the use of a pressure sensor with a wet-wet interface instead of using air as a medium. To mimic the blood phenomena in the human body, the temperature of the blood in the chip will be maintained using a heat lamp controlled using a feedback control loop. These changes will improve upon the current design, which is currently an optically clear device that measures the viscosity of fluids, while having fast pressure response and, at the same time, can be programmed to vary the applied shear rate.

This work has been presented as a poster at the University of Ottawa's Engineering and Computer Science Graduate Poster Competition in 2015. It was accepted by, and presented as, an extended abstract at the 2nd International Conference on Fluid Flow, Heat and Mass Transfer (FFHMT) in 2015. The design was accepted by, and presented at, the Medical Methods and Applications (MeMeA) conference in 2017. A Student Travel Grant Award was awarded for the paper submission and presentation at MeMeA 2017.

Research Ethics

Research ethics were required for the use of human blood with the system. The research ethics board at University of Ottawa reviewed the ethics under file #H11-13-06.

References

[1] G. J. Tortora and B. Derrickson, *Principles of anatomy and physiology*, 12th ed. Hoboken, N.J.: John Wiley & Sons, 2010.

[2] K. H. K. Wong, J. M. Chan, R. D. Kamm, and J. Tien, “Microfluidic Models of Vascular Functions,” *Annu. Rev. Biomed. Eng*, vol. 14, pp. 205–30, 2012.

[3] R. G. Owens, “A new microstructure-based constitutive model for human blood,” *J. Nonnewton. Fluid Mech.*, vol. 140, no. 1–3, pp. 57–70, Dec. 2006.

[4] P. C. Sousa, F. T. Pinho, M. A. Alves, and M. S. N. Oliveira, “A review of hemorheology: Measuring techniques and recent advances,” *Korea Aust. Rheol. J.*, vol. 28, no. 1, pp. 1–22, 2016.

[5] E. Ponder, “The Relation between Red Blood Cell Density and Corpuscular Hemoglobin Concentration,” *J. Biol. Chem.*, no. 144, pp. 333–339, 1942.

[6] R. J. Trudnowski and R. C. Rico, “Specific gravity of blood and plasma at 4 and 37 °C,” *Clin. Chem.*, vol. 20, no. 5, pp. 615–616, 1974.

[7] M. Bureau, J. C. Healy, D. Bourgoin, and M. Joly, “Rheological hysteresis of blood at low shear rate.,” *Biorheology*, vol. 17, no. 1–2. pp. 191–203, Jan-1980.

[8] M. Fenech, D. Garcia, H. J. Meiselman, and G. Cloutier, “A particle dynamic model of red blood cell aggregation kinetics.,” *Ann. Biomed. Eng.*, vol. 37, no. 11, pp. 2299–309, Nov. 2009.

[9] A. M. Robertson and R. G. Owens, *Cardiovascular Mathematics*, 1st ed. Milan: Springer Milan, 2009.

[10] O. K. Baskurt and H. J. Meiselman, “Blood Rheology and Hemodynamics,” *Semin. Thromb. Hemost.*, vol. 29, no. 5, pp. 435–450, 2003.

[11] R. P. Chhabra, “Non-Newtonian fluids: An introduction,” *Rheol. Complex Fluids*, pp. 3–34, 2010.

[12] K. M. Jan, S. Chien, and J. T. Bigger, “Observations on blood viscosity changes after acute myocardial infarction.,” *Circulation*, vol. 51, no. 6, pp. 1079–1084, 1975.

[13] D. P. Briley, G. D. Giraud, N. B. Beamer, E. M. Spear, S. E. Grauer, J. M. Edwards, W. M. Clark, G. J. Sexton, and B. M. Coull, “Spontaneous echo contrast and hemorheologic abnormalities in cerebrovascular disease.,” *Stroke.*, vol. 25, no. 8, pp. 1564–9, 1994.

[14] B. Liu and D. Tang, “Computer simulations of atherosclerotic plaque growth in coronary arteries.,” *Mol. Cell. Biomech.*, vol. 7, no. 4, pp. 193–202, 2010.

[15] Z. Shen, G. Coupier, B. Kaoui, B. Polack, J. Harting, C. Misbah, and T. Podgorski, “Inversion of hematocrit partition at microfluidic bifurcations,” *Microvasc. Res.*, vol. 105, pp. 40–46, 2016.

[16] F. C. Mokken, M. Kedaria, C. P. Henny, M. R. Hardeman, and A. W. Gelb, “The clinical importance of erythrocyte deformability, a hemorrheological parameter,” *Ann. Hematol.*, vol. 64, no. 3, pp. 113–122, 1992.

[17] G. Tomaiuolo, “Biomechanical properties of red blood cells in health and disease towards microfluidics,” *Biomicrofluidics*, vol. 8, no. 5, pp. 1–19, 2014.

[18] C. Pfafferott, G. B. Nash, and H. J. Meiselman, “Red blood cell deformation in shear flow. Effects of internal and external phase viscosity and of in vivo aging.,” *Biophys. J.*, vol. 47, no. 5, pp. 695–704, 1985.

[19] H. Noguchi and G. Gompper, “Shape transitions of fluid vesicles and red blood cells in capillary flows,” *Proc. Natl. Acad. Sci.*, vol. 102, no. 40, pp. 14159–14164, 2005.

[20] D. A. Fedosov, M. Peltomäki, and G. Gompper, “Deformation and dynamics of red blood cells in flow through cylindrical microchannels.,” *Soft Matter*, vol. 10, no. 24, pp. 4258–67, 2014.

[21] A. M. Robertson and M. V Kameneva, “Hemorheology,” vol. 37, pp. 63–120, 2008.

[22] N. Maeda, M. Seike, and T. Shiga, “Effect of temperature on the velocity of erythrocyte aggregation,” *BBA - Biomembr.*, vol. 904, no. 2, pp. 319–329, 1987.

[23] C. Wagner, P. Steffen, and S. Svetina, “Aggregation of red blood cells: From rouleaux to clot formation,” *Comptes Rendus Phys.*, vol. 14, no. 6, pp. 459–469, 2013.

[24] S. Chien, “Shear Dependence of Effective Cell Volume as a Determinant of Blood Viscosity,” *Adv. Sci.*, vol. 168, no. 3934, pp. 977–979, 1970.

[25] M. Yamamoto, “Effects of fibrinogen, globulin, albumin and hematocrit on the kinetics of erythrocyte aggregation in man,” *Angiology*, p. 663, 1986.

[26] M. Bureau, J. C. Healy, D. Bourgoin, and M. Joly, “Etude rhéologique en régime transitoire de quelques échantillons de sangs humains artificiellement modifiés,” *Rheol. Acta*, vol. 18, no. 6, pp. 756–768, Nov. 1979.

[27] M. Singh and J. F. Stoltz, “Influence of temperature variation from 5 degrees C to 37 degrees C on aggregation and deformability of erythrocytes.,” *Clin. Hemorheol. Microcirc.*, vol. 26, no. 1, pp. 1–7, 2002.

[28] M. Uyuklu, M. Cengiz, P. Ulker, T. Hever, J. Tripette, P. Connes, N. Nemeth, H. J. Meiselman, and O. K. Baskurt, “Effects of storage duration and temperature of human blood on red cell deformability and aggregation.,” *Clin. Hemorheol. Microcirc.*, vol. 41, no. 4, pp. 269–78, Jan. 2009.

[29] R. Fåhræus and T. Lindqvist, “The Viscosity of the Blood in Narrow Capillary Tubes,” *Am. J. Physiol.*, vol. 96, no. 3, pp. 562–568, 1931.

[30] S. Kim, P. K. Ong, O. Yalcin, M. Intaglietta, and P. C. Johnson, “The cell-free layer in microvascular blood flow,” *Biorheology*, vol. 46, no. 3, pp. 181–189, 2009.

[31] Q. M. Qi and E. S. G. Shaqfeh, “Theory to predict particle migration and margination in the pressure-driven channel flow of blood,” *Phys. Rev. Fluids*, vol. 2, no. 9, p. 93102, 2017.

[32] D. A. Fedosov, B. Caswell, A. S. Popel, and G. E. Karniadakis, “Blood Flow and Cell-Free Layer in Microvessels,” *Microcirculation*, vol. 17, no. 8, pp. 615–628, 2012.

[33] M. Abkarian and A. Viallat, “Vesicles and red blood cells in shear flow,” *Soft Matter*, vol. 4, no. 4, p. 653, 2008.

[34] R. G. Henríquez Rivera, X. Zhang, and M. D. Graham, “Mechanistic theory of margination and flow-induced segregation in confined multicomponent suspensions: Simple shear and Poiseuille flows,” *Phys. Rev. Fluids*, vol. 1, no. 6, p. 60501, 2016.

[35] J. J. Bishop, A. S. Popel, M. Intaglietta, and P. C. Johnson, “Rheological effects of red blood cell aggregation in the venous network: a review of recent studies.,” *Biorheology*, vol. 38, no. 2–3, pp. 263–74, Jan. 2001.

[36] W. Reinke, P. Gaehtgens, and P. C. Johnson, “Blood viscosity in small tubes: effect of shear rate, aggregation, and sedimentation.,” *Am. J. Physiol.*, vol. 253, no. 3 Pt 2, pp. H540–H547, 1987.

[37] M. a Moyers-Gonzalez and R. G. Owens, “Mathematical modelling of the cell-depleted peripheral layer in the steady flow of blood in a tube.,” *Biorheology*, vol. 47, no. 1, pp. 39–71, Jan. 2010.

[38] J. A. Champion and S. Mitragotri, “Role of target geometry in phagocytosis,” *Proc. Natl. Acad. Sci.*, vol. 103, no. 13, pp. 4930–4934, 2006.

[39] K. Müller, D. a Fedosov, and G. Gompper, “Margination of micro- and nano-particles in blood flow and its effect on drug delivery.,” *Sci. Rep.*, vol. 4, p. 4871, 2014.

[40] K. Walters, *Rheometry*, 1st ed. London: Chapman and Hall, 1980.

[41] G. Schramm, “A Practical Approach to Rheology and Rheometry,” *Rheology*, 1994.

[42] N. Srivastava and M. A. Burns, “Microfluidic pressure sensing using trapped air compression.,” *Lab Chip*, vol. 7, no. 5, pp. 633–7, 2007.

[43] P. Guillot, P. Panizza, J. Salmon, A. Colin, C. Bruneau, T. Colin, and M. Joanicot, “Viscosimeter on a Microfluidic Chip,” *Society*, no. 6, pp. 6438–6445, 2006.

[44] R. Mehri, “Red Blood Cell Aggregation Characterization: Quantification and Modeling Implications of Red Blood Cell Aggregation at Low Shear Rates,” 2016.

[45] R. Mehri, “Micro PIV and Numerical Investigation of a Micro-Couette Blood Flow,” 2012.

[46] H. Zeng and Y. Zhao, “Study of Whole Blood Viscosity Using A Microfluidic Device,” *2008 ASME Int. Mech. Eng. Congr. Expo. IMECE2008*, vol. 2, pp. 223–229, 2008.

[47] Y. J. Kang, S. Y. Yoon, K. H. Lee, and S. Yang, “A highly accurate and consistent microfluidic viscometer for continuous blood viscosity measurement,” *Artif. Organs*, vol.

34, no. 11, pp. 944–949, 2010.

[48] C. J. Pipe, T. S. Majmudar, and G. H. McKinley, “High shear rate viscometry,” *Rheol. Acta*, vol. 47, no. 5–6, pp. 621–642, 2008.

[49] D. Liepsch, *Biofluid mechanics.*, vol. 43, no. 4. 1998.

[50] A. L. Copley, *Clinical Hemorheology: Applications in Cardiovascular and Hematological Disease, Diabetes, Surgery and Gynecology*, 1st ed., vol. 74. Dordrecht: MARTINUS NIJHOFF PUBLISHERS, 1987.

[51] A. R. Pries, D. Neuhaus, and P. Gaehtgens, “Blood viscosity in tube flow: dependence on diameter and hematocrit.,” *Am. J. Physiol.*, vol. 263, no. 6 Pt 2, pp. H1770-8, Dec. 1992.

[52] S. Kim, B. Namgung, P. K. Ong, Y. I. Cho, K. J. Chun, and D. Lim, “Determination of rheological properties of whole blood with a scanning capillary-tube rheometer using constitutive models,” *J. Mech. Sci. Technol.*, vol. 23, no. 6, pp. 1718–1726, Apr. 2010.

[53] S. Shin, D. Keum, and Y. H. Ku, “Blood Viscosity Measurements Using a Pressure-Scanning Capillary Viscometer,” *KSME Int. J.*, vol. 16, no. 12, pp. 1719–1724, 2002.

[54] C. Alonso, A. R. Pries, and P. Gaehtgens, “Time-dependent rheological behavior of blood at low shear in narrow vertical tubes.,” *Am. J. Physiol.*, vol. 265, no. 2 Pt 2, pp. H553–H561, 1993.

[55] B. Chayer, K. L Pitts, G. Cloutier, and M. Fenech, “Velocity measurement accuracy in optical microhemodynamics: experiment and simulation,” *Physiol. Meas.*, vol. 33, no. 10, pp. 1585–1602, 2012.

[56] K. L. Pitts and M. Fenech, “Micro-particle image velocimetry for velocity profile measurements of micro blood flows.,” *J. Vis. Exp.*, no. 74, p. e50314, Jan. 2013.

[57] Z. Huang, X. Li, M. Martins-Green, and Y. Liu, “Microfabrication of cylindrical microfluidic channel networks for microvascular research.,” *Biomed. Microdevices*, vol. 14, no. 5, pp. 873–83, Oct. 2012.

[58] L. Saias, J. Autebert, L. Malaquin, and J.-L. Viovy, “Design, modeling and characterization of microfluidic architectures for high flow rate, small footprint microfluidic systems.,” *Lab Chip*, vol. 11, no. 5, pp. 822–832, 2011.

[59] D. R. Gossett, W. M. Weaver, A. J. Mach, S. C. Hur, H. T. K. Tse, W. Lee, H. Amini, and D. Di Carlo, “Label-free cell separation and sorting in microfluidic systems.,” *Anal. Bioanal. Chem.*, vol. 397, no. 8, pp. 3249–67, Aug. 2010.

[60] N. Tran-Minh, T. Dong, and F. Karlsen, “An efficient passive planar micromixer with ellipse-like micropillars for continuous mixing of human blood,” *Comput. Methods Programs Biomed.*, vol. 117, no. 1, pp. 20–29, 2014.

[61] M. T. Yang, J. Fu, Y. Wang, R. A. Desai, and C. S. Chen, “Assaying stem cell mechanobiology on microfabricated elastomeric substrates with geometrically modulated

rigidity.," *Nat. Protoc.*, vol. 6, no. 2, pp. 187–213, 2011.

[62] H. Bruus, "Theoretical Microfluidics." MIC – Department of Micro and Nanotechnology at Technical University of Denmark, p. 247, 2006.

[63] D. Rubenstein, W. Yin, and M. D. Frame, *Biofluid Mechanics: An Introduction to Fluid Mechanics, Macrocirculation, and Microcirculation*, 1st ed. Academic Press, 2011.

[64] B. R. Gebart, "Permeability of Unidirectional Reinforcements for RTM," *J. Compos. Mater.*, vol. 26, no. 8, pp. 1100–1133, 1992.

[65] P. Tabeling, *Introduction to microfluidics*. Oxford; New York: Oxford University Press, 2010.

[66] Y. Son, "Determination of shear viscosity and shear rate from pressure drop and flow rate relationship in a rectangular channel," *Polymer (Guildf).*, vol. 48, pp. 632–637, 2007.

[67] J. P. Hartnett and M. Kostic, "Heat Transfer to Newtonian and Non-Newtonian Fluids in Rectangular Ducts," pp. 247–356, 1989.

[68] W. Kozicki, C. H. Chou, and C. Tiu, "Non-Newtonian flow in ducts of arbitrary cross-sectional shape," *Chem. Eng. Sci.*, vol. 21, no. 8, pp. 665–679, 1966.

[69] Y. Son, "Determination of shear viscosity and shear rate from pressure drop and flow rate relationship in a rectangular channel," *Polymer (Guildf).*, vol. 48, no. 2, pp. 632–637, 2007.

[70] J. A. Pathak, R. R. Sologuren, and R. Narwal, "Do clustering monoclonal antibody solutions really have a concentration dependence of viscosity?," *Biophys. J.*, vol. 104, no. 4, pp. 913–923, 2013.

[71] RheoSense Inc., *m-VROC (Viscometer/Rheometer-On-a-Chip) User's Manual*, 900003.5-M. 2012.

[72] R. K. Hobbie and B. J. Roth, *Intermediate physics for medicine and biology*, 4th ed. Berlin: Springer Science & Business Media, 2007.

[73] O. K. Baskurt and H. J. Meiselman, "Blood Rheology and Hemodynamics," *Semin. Thromb. Hemost.*, vol. 29, no. 5, pp. 435–450, 2003.

[74] J. P. Cloherty, *Manual of Neonatal Care*, 6th ed. Wolters Kluwer, 2007.

[75] FLUIGENT, *Flow-Rate Platform User Manual*, 15/03/14. FLUIGENT, 2014.

[76] All Sensors, *Miniature Basic Pressure Sensors*, DS-0105 Re. 2007.

[77] Fluigent, *MFCS-EZ Microfluidic Flow Control System - User Manual*, 24/07/13. 2013.

[78] S. Mitra and S. Chakraborty, *Microfluidics and Nanofluidics Handbook: fabrication, implementation, and applications*. 2011.

[79] A. del Campo and E. Arzt, Eds., *Generating Micro- and Nanopatterns on Polymeric*

Materials, 1st ed. Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA, 2011.

[80] Y. Xia and G. M. Whitesides, "Soft lithography," *Annu. Rev. Mater. Sci.*, vol. 28, no. 12, pp. 153–184, 1998.

[81] D. W. Inglis, "A method for reducing pressure-induced deformation in silicone microfluidics," *Biomicrofluidics*, vol. 4, no. 2, pp. 1–8, 2010.

[82] H. Madadi, M. Mohammadi, J. Casals-Terré, and R. C. López, "A novel fabrication technique to minimize poly(dimethylsiloxane)-microchannels deformation under high-pressure operation.," *Electrophoresis*, vol. 34, no. 22–23, pp. 3126–32, Dec. 2013.

[83] M. "Mike" Kim, Y. Huang, K. Choi, and C. H. Hidrovo, "The improved resistance of PDMS to pressure-induced deformation and chemical solvent swelling for microfluidic devices," *Microelectron. Eng.*, vol. 124, pp. 66–75, Jul. 2014.

[84] R. Dangla, F. Gallaire, and C. N. Baroud, "Microchannel deformations due to solvent-induced PDMS swelling," *Lab Chip*, vol. 10, no. 21, p. 2972, 2010.

[85] B. S. Hardy, K. Uechi, J. Zhen, and H. Pirouz Kavehpour, "The deformation of flexible PDMS microchannels under a pressure driven flow," *Lab Chip*, vol. 9, no. 7, pp. 935–938, 2009.

[86] T. Ahmad and I. Hassan, "Experimental Analysis of Microchannel Entrance Length Characteristics Using Microparticle Image Velocimetry," *J. Fluids Eng.*, vol. 132, no. 4, p. 41102, 2010.

[87] B. Algret, "Design of microfluidic chips for blood," 2010.

[88] K. L. Pitts, S. Abu-Mallouh, and M. Fenech, "Contact angle study of blood dilutions on common microchip materials," *J. Mech. Behav. Biomed. Mater.*, vol. 17, pp. 333–336, 2012.

[89] Diversified Enterprises, "Surface Energy Data for PDMS: Polydimethylsiloxane, CAS #9016-00-6 Source," vol. 1947, no. 44, pp. 1–2, 2009.

[90] L. Mazutis, J. Gilbert, W. L. Ung, D. a Weitz, A. D. Griffiths, and J. a Heyman, "Single-cell analysis and sorting using droplet-based microfluidics.," *Nat. Protoc.*, vol. 8, no. 5, pp. 870–91, 2013.

[91] MicroChem, "NANO SU-8: Negative Tone Photoresist Formulations 50-100." p. 4, 2011.

[92] Y. J. Chuang, F. G. Tseng, and W. K. Lin, "Reduction of diffraction effect of UV exposure on SU-8 negative thick photoresist by air gap elimination," *Microsyst. Technol.*, vol. 8, no. 4–5, pp. 308–313, 2002.

[93] S. H. Tan, N. T. Nguyen, Y. C. Chua, and T. G. Kang, "Oxygen plasma treatment for reducing hydrophobicity of a sealed polydimethylsiloxane microchannel," *Biomicrofluidics*, vol. 4, no. 3, pp. 1–8, 2010.

[94] J. Kestin, M. Sokolov, and W. A. Wakeham, "Viscosity of Liquid Water in the Range -8 to

150C.pdf," *J. Phys. Chem. Ref. Data*, vol. 7, no. 3, pp. 941–948, 1978.

[95] N.-S. Cheng, "Formula for the Viscosity of a Glycerol−Water Mixture," *Ind. Eng. Chem. Res.*, vol. 47, no. 9, pp. 3285–3288, May 2008.

[96] The Dow Chemical Company, *Polyox Water-Soluble Resins*. 2002.

[97] A. Pribush, D. Zilberman-Kravits, and N. Meyerstein, "The mechanism of the dextran-induced red blood cell aggregation," *Eur. Biophys. J.*, vol. 36, no. 2, pp. 85–94, 2007.

[98] J. K. Armstrong, R. B. Wenby, H. J. Meiselman, and T. C. Fisher, "The hydrodynamic radii of macromolecules and their effect on red blood cell aggregation.," *Biophys. J.*, vol. 87, no. 6, pp. 4259–70, 2004.

[99] Dow, "POLYOX Water-Soluble Resins Dissolving Techniques." Dow, p. 14, 2003.

[100] Solid State Cooling Systems, *ThermoCube 200/265/300/400 Thermoelectric Chiller Manual*, M14 ed. 2002.

[101] Dow Corning, "A Guide to Silane Solutions: The Basics of Silane Chemistry," pp. 1–6, 2009.

[102] H. Schmid-Schönbein, P. Gaehtgens, and H. Hirsch, "On the shear rate dependence of red cell aggregation in vitro.," *J. Clin. Invest.*, vol. 47, no. 6, pp. 1447–1454, 1968.

[103] C. W. Macosko, *Rheology : principles, measurements, and applications*, 1st ed. New York: Wiley-VCH, 1994.

[104] K. L. Pitts, R. Mehri, C. Mavriplis, and M. Fenech, "Micro-particle image velocimetry measurement of blood flow: validation and analysis of data pre-processing and processing methods," *Meas. Sci. Technol.*, vol. 23, pp. 1–9, 2012.

[105] K. L. Pitts and M. Fenech, "An analytic study on the effect of alginate on the velocity profiles of blood in rectangular microchannels using microparticle image velocimetry.," *PLoS One*, vol. 8, no. 8, p. e72909, Jan. 2013.

[106] G. Pitingolo, R. Vecchione, and P. A. Netti, "A simple and low cost method to fabricate NOA microfluidic chips," *Chip and Tips*, 2015. [Online]. Available: http://blogs.rsc.org/chipsandtips/2015/01/09/a-simple-and-low-cost-method-to-fabricate-noa-microfluidic-chips/. [Accessed: 04-Oct-2017].

[107] E. P. Dupont, R. Luisier, and M. A. M. Gijs., "NOA 63 as a UV-curable material for fabrication of microfluidic channels with native hydrophilicity," *Microelectron. Eng.*, vol. 87, no. 5–8, pp. 1253–1255, 2010.

[108] Sporian Microsystems Inc., Ed., "Su-8 adhesion to Si wafers," 2007. [Online]. Available: http://www.sporian.com/cgi-bin/HyperNews/get/SU-8/54.html?inline=1. [Accessed: 01-Jan-2014].

[109] S. Basrour, J. C. Jeannot, and V. Soumann, "SU-8: Thick Photo-Resist for MEMS," 2013. [Online]. Available: http://memscyclopedia.org/su8.html. [Accessed: 01-Jan-2014].

[110] S. Heidemann, “SU-8: Thick Photo-Resist for MEMS,” 2013. [Online]. Available: http://memscyclopedia.org/su8.html. [Accessed: 01-Jan-2014].

[111] A. Del Campo and C. Greiner, “SU-8: a photoresist for high-aspect-ratio and 3D submicron lithography,” *J. Micromech. Microeng. J. Micromech. Microeng*, vol. 17, no. 17, pp. 81–95, 2007.

Appendix

Appendix A Standard Operating Procedure (SOP) for Working with Hydrofluoric Acid (HF) 1% w/v in Dr. Fenech's Lab

A.1. Disclaimer

This document, and those to which it refers, have been created in reference to Dr. Fenech's laboratory and experimental procedures in mind. Its use for other laboratories, procedures or researchers should be modified or remade accordingly. Researchers who are tailoring their procedure should check with appropriate safety coordinators (e.g. Pierre Laflamme at uOttawa), supervisors/PI's, and/or lab mates to verify safe handling.

A.2. Purpose

This standard operating protocol outlines the procedure to safely clean silicon wafers with hydrofluoric acid 1% w/v.

A.3. References

Consulted with Health, Safety and Risk Manager, and Office of Risk Management at the University of Ottawa.

UCIrvine's "Cleaning Procedures for Silicon Wafers":

https://www.inrf.uci.edu/wordpress/wp-content/uploads/sop-wet-silicon-solvent-clean.pdf

A.4. Definitions

Clean room – A room that is kept clean from contaminants in the form of particulate or bacteria. They are used in laboratories for procedures which require a low number of particulates in the air.

A.5. Materials Required

A.5.1. Recommended Gowning Apparel/Personal Protective Equipment (PPE)

The table below lists the personal protective equipment (PPE) necessary for use in the procedure, and their locations (if a set place is designated). The first items in the list are those required for clean room gowning, which has its own standard operating procedure. An empty row separates the clean room gowning equipment from the extra equipment needed for this procedure.

Equipment	Location
Regular lab coat	Determined by each person
Shoe covers	Cabinet containing disposable clean room equipment
Clean room frock lab coat	Inside clean room, on hooks
Disposable clean room lab coat	Inside clean room, on hooks
Disposable face masks	Cabinet containing disposable clean room equipment
Hair net	Cabinet containing disposable clean room equipment
Standard nitrile gloves	Inside clean room on top of flammables cabinet
Chemical splash goggles or safety glasses: for eye protection	Inside clean room, on hooks; or in cabinet containing disposable clean room equipment
Blue disposable personal protection gown	Inside clean room; in drawers
Long, disposable clean room nitrile gloves	Inside clean room; in drawers
Disposable face shield	Inside clean room; in drawers

Equipment	Location
Secondary plastic containers	Inside and outside of clean room

Thick butyl rubber, neoprene, nitrile, or polyethylene gloves may be appropriate. Check glove manufacturer for recommendations on a suitable glove for the specific chemical. Ensure that personal protective equipment is worn at all times while in the clean room.

A.5.2. Waste Containers

The table below lists the hydrofluoric acid contaminated waste containers/equipment necessary for use in the procedure, and their locations (if a set place is designated). Care should be taken when disposing of hydrofluoric contaminated waste so as to not contaminate the outside of any waste container or other surfaces.

Equipment	Location
Plastic container for liquid hydrofluoric (HF) waste - designated for use with HF	Kept inside of fume hood.
Plastic container for solid waste contaminated with HF - designated for use with HF	Kept outside of fume hood, underneath fume hood.
Plastic funnel - designated for use with HF	Inside clean room, with the plastic container for fluid hydrofluoric acid.

A.5.3. Equipment

The table below lists the equipment necessary for use in the procedure, and their locations (if a set place is designated).

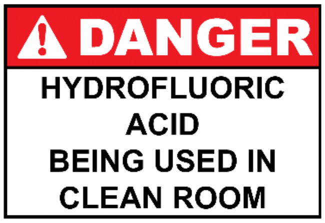
Equipment	Location
Fume hood	Inside clean room in D511.
De-ionized water (DI water)	From the Prima Filter machine above the sink in D511.
Plastic beakers	On drying racks in either D509 or D511.
HF exposure/spill kit	Kept outside of fume hood, underneath fume hood.
Three small plastic tubs/containers - designated for used with HF	Inside clean room acid cabinet. Stored in a plastic secondary container.
Squirt bottle (for DI water)	Inside clean room acid cabinet. Stored in a plastic secondary container.
Plastic scissor tweezer clamps - designated for use with HF	Inside clean room acid cabinet. Stored in a plastic secondary container.
Nitrogen gas	Inside clean room, from gas cylinder with gun nozzle attachment.
Silicon wafer(s)	In specified drawer in clean room.
HF pre-diluted at 1%	Inside clean room acid cabinet. Contained in a plastic bottle and stored in a plastic secondary container.
Timer (with magnetic back)	Attached to the side of the fume hood.
Container for processed wafers	Inside clean room.

Equipment	Location
Calcium carbonate powder (CaCO3)	Inside clean room top drawer to the right near the fume hood.
Scoopula or equivalent	Inside clean room drawer. Top right near the fume hood.
Plastic weighing boats	Inside clean room drawer. Top right near the fume hood.
pH test strips (in a box)	Inside clean room drawer. Top right near the fume hood.

A.6. Procedure

A.6.1. Preparation

1) Ensure you have allotted enough time to do the whole procedure (e.g. going to washroom beforehand).
2) Clean the clean room surfaces and items to be used (either day before or right before using the clean room) using clean room cleaning cloths and clean room floor sweeper. It is advised to do this the day before to allow particles, which are generated by cleaning, to settle.
3) Turn on the clean room fan and leave it running for at least 5 minutes before entering the clean room. This allows the clean room to evacuate particles.
4) Prepare cautionary signage, to caution that HF is being used, at the entrance of the clean room and outside the door to enter the laboratory area. An example is shown in Figure 54.



⚠ DANGER

HYDROFLUORIC
ACID
BEING USED IN
CLEAN ROOM

Figure 54. Example signage to use for procedure.

5) Before handling the hydrofluoric acid (HF), go to the Fenech Lab's Network Accessible Storage (NAS) and:
   a) Carefully read the corresponding Material Safety Data Sheets (MSDS) sheet and sign it.
   b) Learn the HF Presentation "Working with Hydrofluoric Acid".
   c) Read the HF hazard assessment sheets to know the potential hazards.
   d) Read the sheet assessment "Hydrofluoric Acid (HF) Considerations for Dr. Fenech's Lab"
   e) Understand the spill/exposure kit contents, procedures and situations to use them.

   Note: Undergraduate students should never be given the task of mixing Hydrofluoric Acid solutions. Only experienced persons familiar with its properties should handle the concentrated acid.
6) Arrange to have someone else, who has also followed step 5, working in the laboratory at the same time for safety.
7) Check that the sink is clean, as shown in Figure 55 (left), and ready for items to be cleaned, and that there is enough space inside for the white secondary container, as shown in Figure 55 (right).

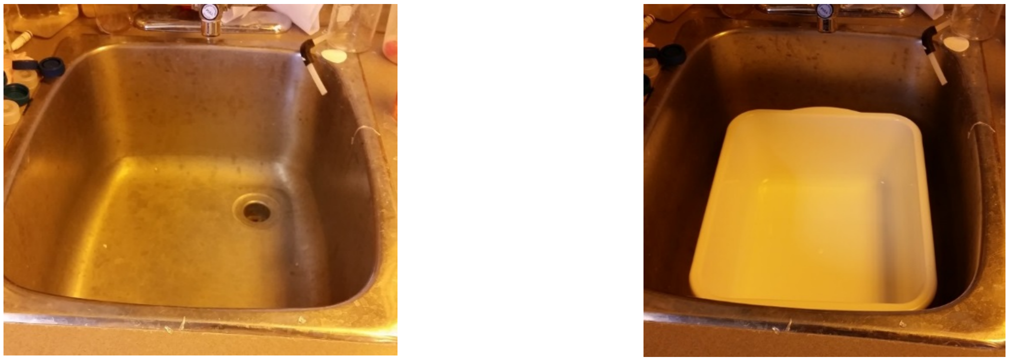


Figure 55. Clean sink and surrounding surfaces (left), and check that the secondary white container fits within the sink (right).

8) Ensure all unnecessary items are removed from the fume hood, as shown in Figure 56.

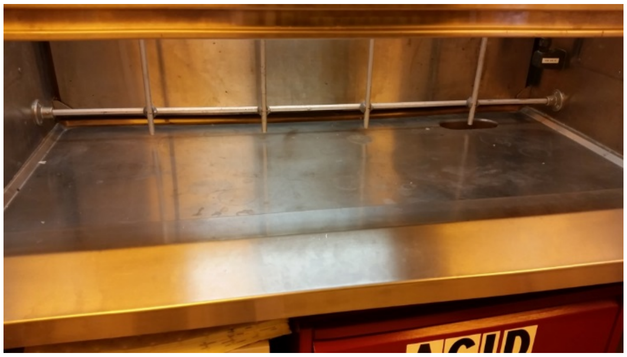


Figure 56. Clean fume hood, ready for use during this procedure.

9) Verify that there are enough silicon wafers necessary for the intended fabrication amount.

10) Verify visually in the acid cabinet, as shown in Figure 57, that there are 3 small plastic containers in the secondary container that holds the HF labelled as follows. Do not remove the containers yet.

    a) "#1: Hydrofluoric Acid (1%)".

    b) "#2: First De-Ionized Water Rinse Bath for Hydrofluoric Acid (1%) – CONTAINS HYDROFLUORIC ACID".

    c) "#3: Rinse Container for Wafers – CONTAINS HYDROFLUORIC ACID".

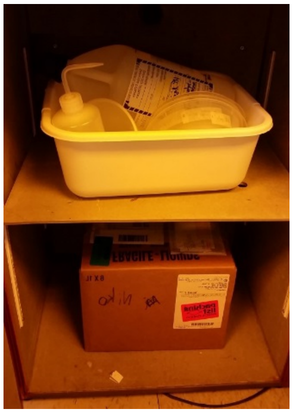


Figure 57. Materials inside the ACID cabinet, with the secondary container containing the three plastic containers (on the top) and the box containing the plastic bottles of Hydrofluoric Acid (1%) (on the bottom).

11) Place the larger and shallower secondary container inside the fume hood, to contain any spills.

12) Place a laboratory absorbent sheet inside the fume hood, inside the container from the previous step.

13) Prepare a 1L plastic beaker with DI water and place in the fume hood, inside the container in step 11.

14) Prepare calcium carbonate ($CaCO_3$ powder) container and plastic weighing boats, which will be used to neutralize the hydrogen fluoride solutions.

    Note: The final molarity of calcium carbonate needs to be about 0.025 M in the neutralized solution. Measuring out more than needed will help if the pH of the neutralized solution is not neutral. The final pH needs to be measured after the neutralization process, so more calcium carbonate is better than less.

15) Check that there is enough nitrogen gas in the cylinder for the procedure, indicated in Figure 58 (left). Open the valves for the nitrogen gas tank and move the nitrogen gun to hook onto part of the fume hood for ease of access, as shown in Figure 58 (right).

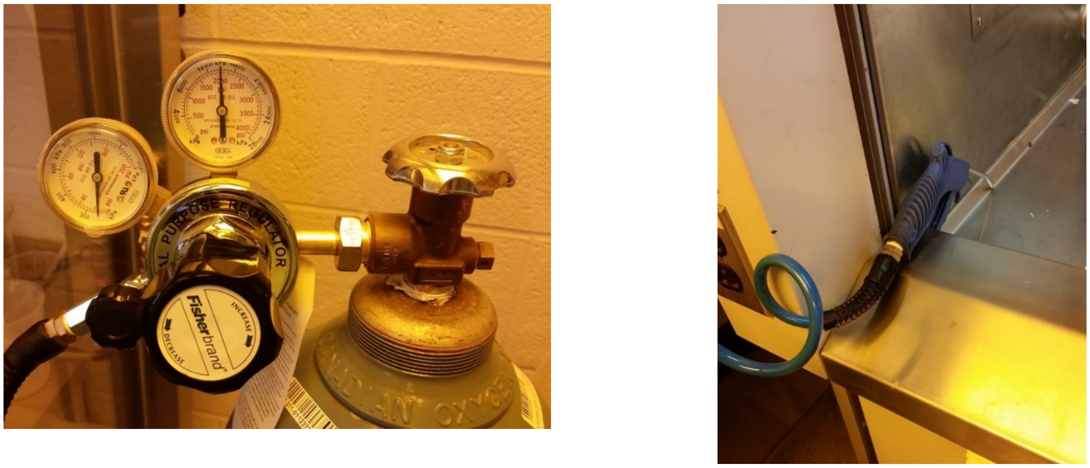


Figure 58. Checking that there is enough nitrogen in the cylinder for the procedure (left) and (right) Placement of nitrogen gun in fume hood for easy access during the procedure (right).

16) Ensure again that the fume hood is free of extraneous items.

17) Place the solid waste container in the work station (fume hood).

18) Turn on the tap to a reasonable flow rate and leave it running for use once the procedure is completed.

A.6.2. Using Hydrofluoric Acid

19) Wear the Personal Protective Equipment (PPE) required when handling HF, located in the clean room PPE drawers as shown in Figure 59.

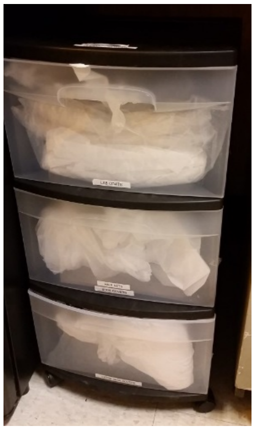


Figure 59. Clean room drawers containing Personal Protective Equipment (PPE), which are necessary to enter the clean room.

a) Put on safety glasses or safety goggles (if not already wearing them).

b) Over your regular lab coat sleeves, put on the disposable long nitrile clean room gloves, as shown in Figure 60.

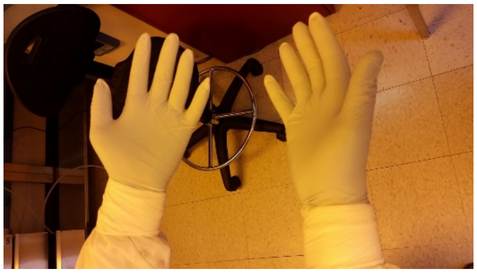


Figure 60. A set of disposable long nitrile clean room gloves goes over the clean room frock and disposable clean room lab coat sleeve cuffs.

c) Put on a disposable personal protection gown (blue plastic). These are unfolded from a small plastic pouch.

   i) The hole where your head goes through is created by removing a pre-perforated piece of plastic, as shown in Figure 61. Place your head through the hole so that the plastic protection gown covers the front of your torso and the open side is to your back.

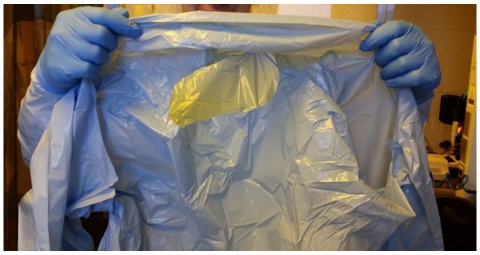


Figure 61. Unfolded personal protection gown with head hole opened.

ii) Ensure your thumbs go through the appropriate thumb-holes at the end of each sleeve, indicated in Figure 62.

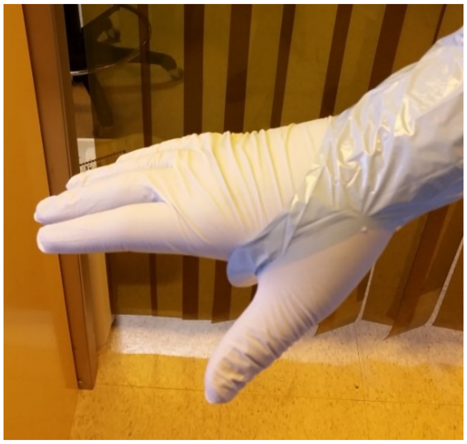


Figure 62. Researcher's thumb goes through the personal protection gown sleeve loop.

iii) Tie the personal protection gown's ribbons together at the back, as shown in Figure 63. It should be tight enough not to come undone easily, but still allow the user to move their arms around freely while completing the procedure.

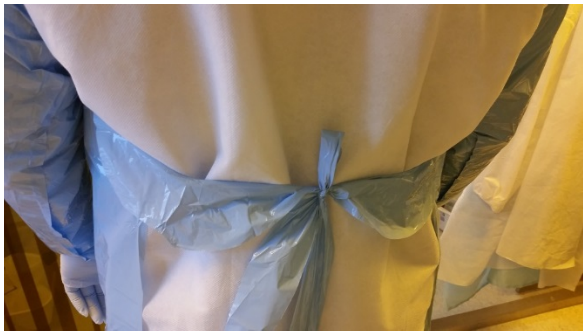


Figure 63. Tied personal protection gown ribbons to keep the apron secure, but not restricting movements.

d) Over the disposable personal protection gown, put on a second layer of the disposable long nitrile clean room plastic gloves.

e) Over the gloves, put on blue disposable nitrile gloves.

f) Over the safety glasses or safety goggles, put on the face shield. A fully gowned researcher is shown in Figure 64.

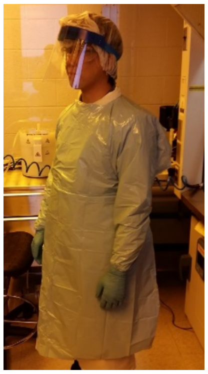


Figure 64. Fully gowned researcher with all personal protective equipment, who is ready to use HF acid.

20) While handling items that are labelled as “For HF processing” (nitrogen gun, DI water bottle and solid/liquid HF chemical waste), change blue outermost gloves before touching them.
21) Remove the other secondary plastic bin (containing HF, 3 plastic containers, funnel, DI water bottle) from the acid cabinet in the clean room and place it in the fume hood.
22) Remove the 3 plastic containers from the bin and place them in order inside the fume hood on the absorbent sheet very close to each other.
23) Fill the second container (#2), up to the indicated fill line (about 500 mL) with de-ionized water from the beaker.
24) If necessary, take out the plastic DI water bottle from the bin and fill it with DI water (also from the beaker).
25) Arrange the work flow in the fume hood to go right-to-left (in their appropriate safety containers), for example:
    a) New silicon wafers to be processed
    b) Container #1
    c) Container #2
    d) Container #3
    e) De-ionized water squirt bottle
    f) Nitrogen gas gun
    g) Container for processed wafers
26) Lower the sash as low as practically possible to work safely inside.

    Note: Work in the fume hood with the sash as low as possible at all times, moving it up or down as necessary with uncontaminated gloves.

27) Take out the HF (pre-diluted at 1%) container from the secondary container bin (located inside the fume hood) and place it on the absorbent sheet, as shown in Figure 65.

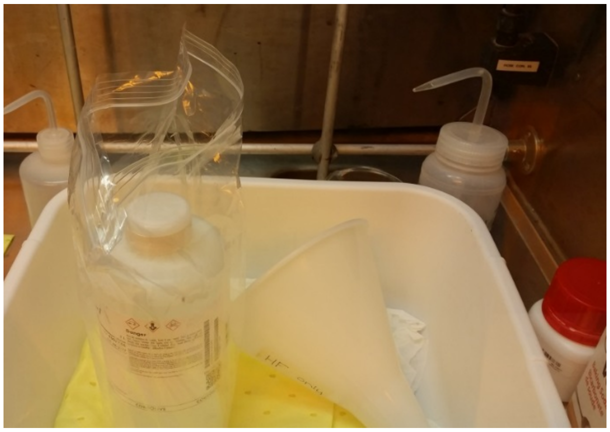


Figure 65. Hydrofluoric Acid (1%) container inside plastic bag, sitting within the secondary plastic container.

a) Uncap the HF container.
b) Safely fill the first plastic container (#1) with about 100 mL of HF up to the indicated fill line, avoiding any splashes.
c) Close the HF container.
d) Put the closed HF container onto absorbent pad, inside the secondary container, in the fume hood.

28) Using plastic scissor tweezer clamps, pick up a new silicon wafer and put it into the first container (#1) filled with HF diluted at 1%.

Note: Do not hold the wafer so that it is tilting sideways as it could then be prone to slipping out of the plastic scissor tweezer clamps. Avoid also twisting the clamps as the silicon wafer might break.

29) Let the silicon wafer soak for 2-4 minutes in the HF diluted at 1%.

30) Using the same plastic scissor tweezer clamps, transfer the silicon wafer from the first container (#1) to the second container (#2).

31) Let the silicon wafer rinse in the liquid for another minute in the DI water.

32) Using the plastic scissor tweezer clamps, transfer the silicon wafer from the second container by moving it over the third container.

a) Holding the DI water bottle in a clean gloved hand, use the bottle to thoroughly rinse the silicon wafer.

b) Rinse the silicon wafer with DI water for 1 minute, holding the wafer over the third container.

c) Use a clean plastic scissor tweezer clamp (uncontaminated with HF) with the clean gloved hand to take the wafer out of the contaminated clamp.

d) Transfer the clean clamp to the contaminated gloved hand safely.

e) Rinse the area where the previous clamp was with more DI water.

f) Test the hydrophobicity of the surface of the silicon wafer by checking the contact angle that the water makes with the wafer. A successful wafer etch will be evident when the wafer surface becomes hydrophobic, as shown in Figure 66.

Note: The oxide layer (which we are trying to remove in this SOP) is hydrophilic, whereas pure silicon is hydrophobic.

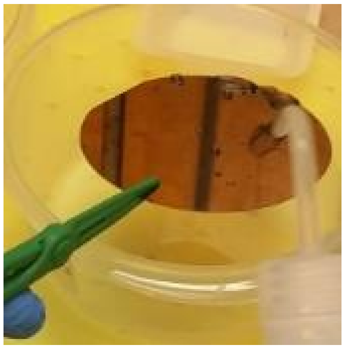


Figure 66. Beading up of DI water on the surface of the processed silicon wafer.

33) Dry the silicon wafer inside the fume hood with nitrogen gas, using the nitrogen gun nozzle.

a) Hold the silicon wafer using the plastic scissor tweezer clamps and slope the wafer down so that any liquid will be blown into the container.

b) Using a clean gloved hand, face the nozzle of the nitrogen gun at the silicon wafer to dry it.

c) Blow the gas over the wafer slowly, so that it does not spray the liquid, until there is no more liquid on it.

d) Place the nitrogen gun back in its holder near the fume hood.

34) Place the cleaned wafer in a wafer holder.

35) Inspect gloves for any possible tears or holes.

a) If there are any holes or cleaning wafers for more than 1 hour, safely replace the outermost blue gloves.

36) Repeat steps 18-25 for each new wafer to be cleaned.

A.6.3. Clean-Up Procedure

37) Neutralize the solutions in each container.

    a) Measure out calcium carbonate ($CaCO_3$) powder into a weighing boat to pour into the plastic containers.

    b) Lower the sash as low as reasonably and safely as possible.

    c) Slowly pour $CaCO_3$ powder into each container (#1, #2, and #3).

    Note: Slowly pouring the solution will avoid an exothermic reaction, since heat will vaporize HF and increase the risk of exposure to HF.

    d) Gently agitate the largest secondary container to move the neutralizing solutions around.

    e) Wait at least 2 minutes for the solution to neutralize. The end of bubble formation indicates that the neutralization is finished.

    f) Test the pH of the solution, using pH papers, of each container. The target pH is a value of at least 5.5 (ensuring the solution is not acidic).

    g) Repeat steps c) to f) as needed, to reach the target pH.

38) Using the clean gloved hand, open the HF liquid waste container.

39) Use the plastic funnel to carefully transfer the liquid from each plastic container (#1, #2 and #3) into the plastic HF liquid waste container, as shown in Figure 67.

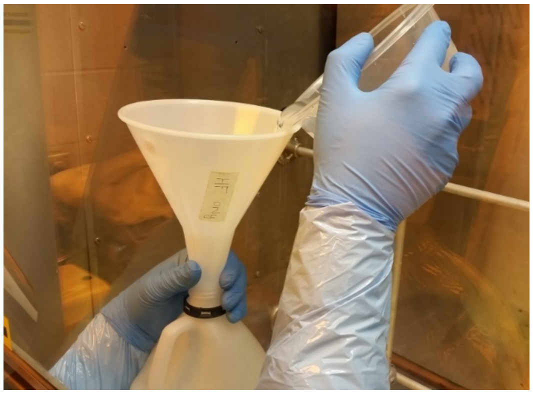


Figure 67. Pouring neutralized liquid HF waste into the HF liquid waste container.

    a) Pour most of the liquid into the liquid waste container, leaving a little bit to gather the sediment powder.

  b) Swish around the liquid to suspend the powder into the liquid and pour into the liquid waste container.
  c) Use DI water to rinse out the powder sediment and repeat the previous sub-step until there is no more powder left.

40) Take out the plastic funnel and place it in the secondary container in the fume hood, next to the 3 plastic containers and inside the large and shallow secondary container.

41) Close the waste container with the designated clean gloved hand.

42) Place the hydrofluoric acid (1%) bottles back into the other container in the acid cabinet.

43) Take the secondary container (containing the 3 plastic containers, plastic beakers and the funnel) and slowly exit the clean room by going backwards. Move slowly and watch that there are any tripping hazards. This process will avoid hydrofluoric acid from contaminating the curtains from the secondary container or personal protective equipment.

44) Carefully place the secondary container (containing the 3 plastic containers and the funnel) in the sink.

45) Arrange the plastic bin so that the faucet will not splash water when the bin is placed in the sink.

46) Using the clean gloved hand, slowly open the tap or increase the flow of water to pour water over the equipment to thoroughly clean the containers, beaker, funnel, and equipment (let water run in each container for at least 2 minute).
  a) Avoid the water splashing out of the sink as it hits any surfaces in the sink.
  b) Thoroughly wash each surface for at least 1 minute, as shown in Figure 68, for each piece of equipment, being sure not to get water into the gloves while washing.

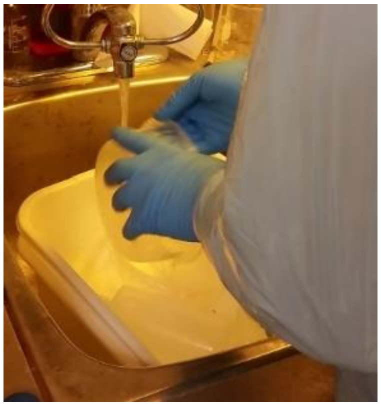


Figure 68. Thoroughly washing each surface of the materials used in the procedure.

c) Keep the water running while doing the proceeding steps.

d) Check that all surfaces do not have any residual powder sediment created from $CaCO_3$ as shown in Figure 69.

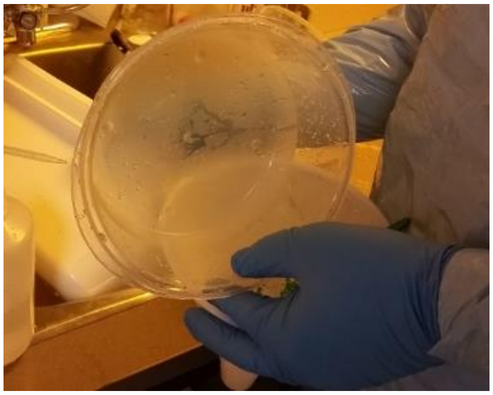


Figure 69.Checking that there is no residual $CaCO_3$ left in the plastic containers.

47) Open the chemical solid waste container with the clean gloved hand.

48) Safely fold the absorbent sheets and used gloves, and throw them away into the HF chemical solid waste container, as shown in Figure 70.

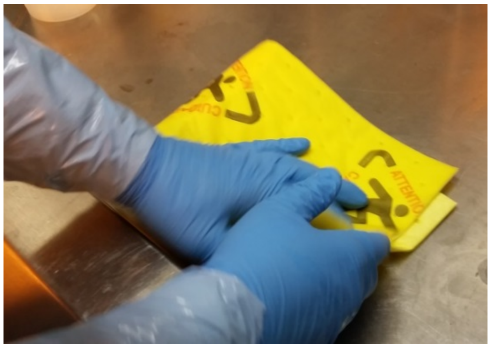


Figure 70. Folding the absorbent sheets to place in the HF chemical solid waste container.

49) Remove the PPE:

a) Safely remove the outermost, blue gloves and put them into the HF chemical solid waste container.

b) Safely remove the disposable face shield and put it into the HF chemical solid waste container.

c) Safely remove the outermost, long gloves and put them into the HF chemical solid waste container.

d) Safely remove the personal protection gowns, as shown Figure 71, and put it into the HF chemical solid waste container.

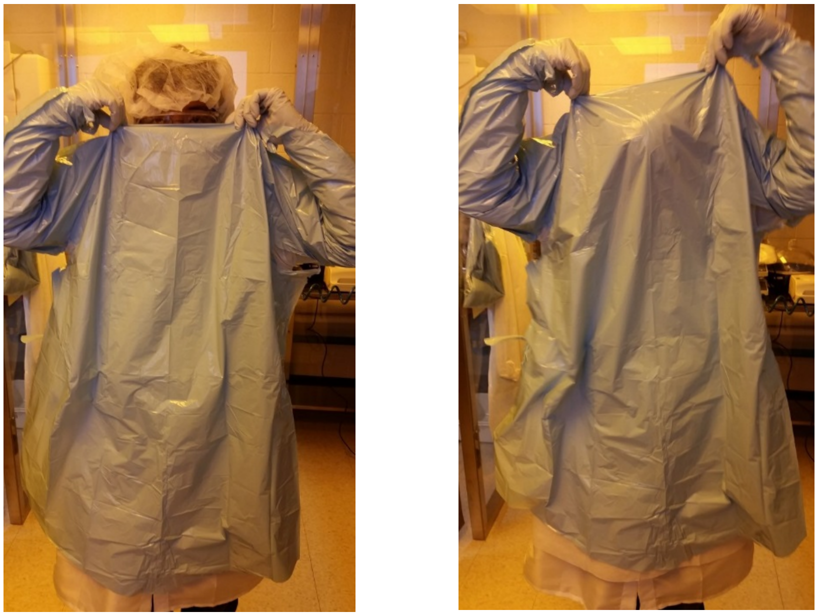


Figure 71. Removal of personal protective gown, starting by pulling the gown over the researcher's head.

e) Safely remove the plastic gloves, as shown in Figure 72, and put them into the HF chemical solid waste container.

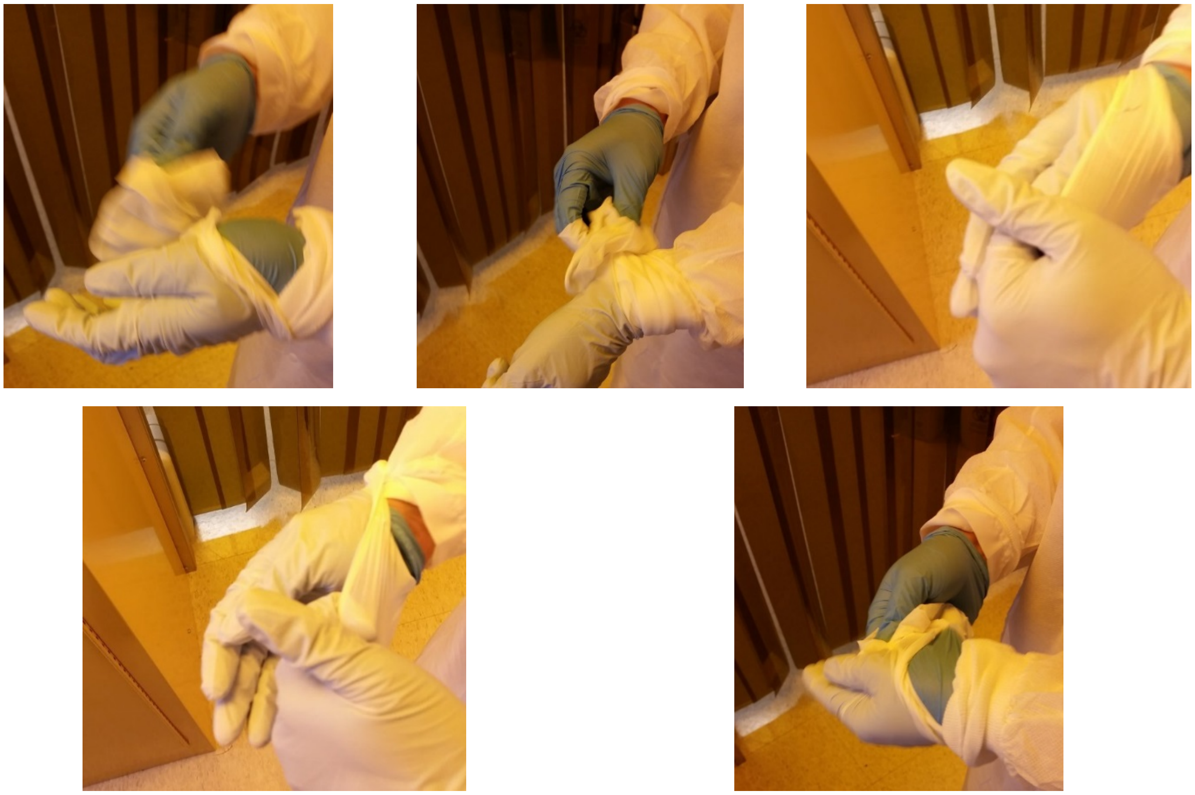


Figure 72. Process of removing gloves in a safe manner, without making contact with the outside of the gloves to the underlying layer.

f) Take off the safety glasses or safety goggles.

50) Close the HF chemical solid waste container.

51) Put on two pairs of blue gloves to turn off tap and put the funnel, plastic scissor tweezer clamps and 3 plastic containers in the secondary plastic bin.

52) Put the plastic beaker on the drying rack.

53) Close the valves on the nitrogen gas tank and put the nozzle back.

54) If any of the containers are damaged, replace with 3 <u>NEW</u> small plastic containers:

a) Label the first one with "#1: Hydrofluoric Acid (1%)".

b) Label the second one with "#2: First De-Ionized Water Rinse Bath for Hydrofluoric Acid (1%) – CONTAINS HYDROFLUORIC ACID".

c) Label last one with "#3: Rinse Container for Wafers – CONTAINS HYDROFLUORIC ACID".

55) Put the secondary container containing the 3 plastic containers, plastic beakers and the funnel back into the secondary container and back into the acid cabinet once they are dried.

56) Note whether the HF chemical solid waste container or the HF liquid waste containers are 75% full and arrange with chemical pickup for disposal, once the containers are over this limit.

Appendix B Wafer Processing Technique Tests

A very short study using wafers was completed since channel walls were initially not perpendicular to the ceiling and floor of the parallel channels of designs. Six changes to the protocol were tested and evaluated based on the PDMS channels that were created, such as adding relaxation time or increasing the exposure time. A control, using the original technique, was used and a combination of using all the techniques was also tested. It was found that using both (1) a thin layer of glycerol between the mask and SU-8 surface, and (2) using 4 seconds, with our setup, would give the most perpendicular channels.

B.1. Purpose

Determine if the wafer processing procedure would yield more perpendicular channels compared to the current technique, as following [91].

B.2. Hypothesis

Steps of the wafer processing technique can be improved upon, based on the how the wafer is treated with the SU-8 photopolymer. The processing technique could be tailored due to differences between laboratory setups.

B.3. Potential Solutions

Various research groups papers and their websites were investigated for possible “tips” that they use to modify the current procedure for their equipment to ascertain vertical/perpendicular channels. One of the first potential change was heating the silicon wafer at 120 °C to 140 °C for a few minutes to dry the surface, with the intent to cause higher hydrophilicity of the surface and better SU-8 adhesion [108]. The addition of relaxation time for the SU-8 to have time to relax from the structural changes both (1) between the spin coating and soft bake step, and (2) between the soft bake and exposure to UV light [109]. It has also been recommended that the heating up of the wafer should be completed at a higher temperature, ramped up from room temperature, for a longer period of time. Heidemann [110] suggests having the longer soft bake step to “reduce the trained solvent level” to help promote photopolymer adhesion.

Due to the non-uniform surface structure of the SU-8 layer, any air gaps can cause the UV rays to create features with "over-hangs", as shown in Figure 73 (left) or in Figure 74 (left to right) [92], [111].

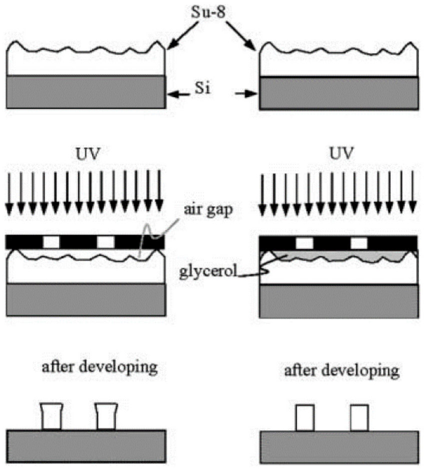


Fig. 8. Fabrication comparison of standard SU-8 process (left), and glycerol filled process (right)

Figure 73. Difference in fabrication procedure between (left) a standard SU-8 process (left) and (right) using glycerol to fill the gaps (right) [92].

Using a fluid with the same refractive index as the SU-8, such as glycerol, was shown to develop features with perpendicular walls (no overhangs) as in Figure 73 (right) [92].

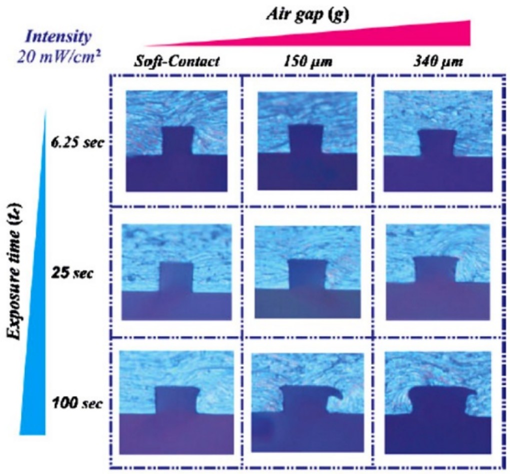


Figure 74. Cross-sections of channels showing the effect of increasing the air gap size between the mask and SU-8 layer (left to right), and the effect of increasing the exposure time (top to bottom) [111].

The effect of having increased UV light exposure time has been shown to increase perpendicular walls, but has the side effect of widening the channel slightly [111].

B.4. Methods

The changes to the wafer processing procedure were completed using masks with channels 100 µm × 100 µm. Wafer #1 was used as a control and underwent the standard procedure, whereas wafer #8 underwent all the changes in the procedure. Table 25 contains a summary of the wafers and any changes to the processing technique that the wafer experienced.

1) All wafers were processed using the hydrofluoric acid (1%) etching process as detailed in this report.
2) {Wafer #2 and #8 was separately allowed to dry at 140 °C for 3 minutes before going to the next steps.}
3) The SU-8 spin coated on the silicon wafers was completed to create a 100 µm thick layer.
4) {Wafer #3 and #8 was given a relaxation time: of 30 minutes before the soft bake procedure.}
5) The wafers, except for wafer #4 and #8, were soft-baked in a two-step process of 10 minutes at 65 °C and 30 minutes at 95 °C. {Wafer #4 and #8 experienced a soft bake of 100 °C for 100 minutes, starting from room temperature and ramping up.}
6) {Wafer #5 and #8 then had a relaxation time of 10 minutes between the soft bake step, before exposure to UV light.}
7) Masks of the designs were then placed over the wafers. {Wafer #6 and #8 had a layer of glycerol between the mask and the SU-8 layer, pressed to be as thin as possible, with the extra removed.}
8) The wafers were then placed in the UV oven, with a UV aligner on top, inside of a UV light for curing at 3 seconds, except for wafer #7 and #8. {Wafer #7 and #8 had an exposure time of 4 seconds to increase the curing.}
9) All wafers then had a hard bake step for 1 minute at 65 °C and 10 minutes at 95 °C.
10) All wafers were then developed in SU-8 developer and sufficiently rinsed with isopropyl alcohol.

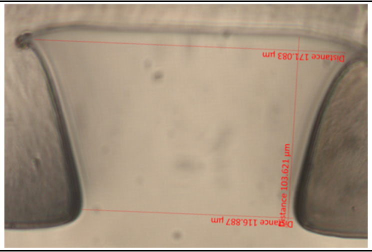
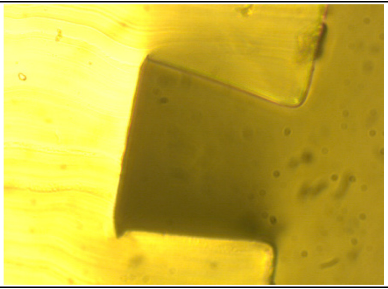
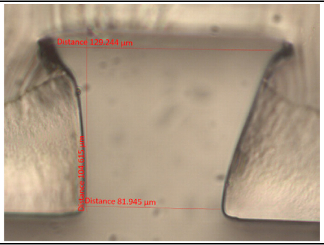
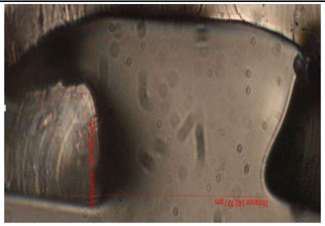
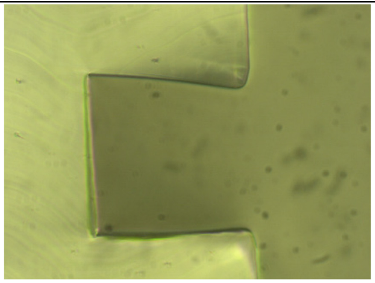
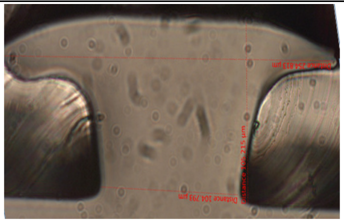
Table 25. Summary of changes to the wafer processing steps and which steps the changes would affect.

Wafer #	Wafer Processing Step Changed	Change in Wafer Processing Step
1	None (Control)	None
2	Wafer Preparation	Heating Wafer to Dry: at 140°C for 3 minutes
3	Between Spin Coating and Soft Bake	Relaxation Time: 30 minutes
4	Soft Bake	Higher Temperature for Longer Time: 100°C for 100 minutes
5	Between Soft Bake and Exposure to UV Light	Relaxation Time: 10 minutes
6	Between Soft Bake and Exposure to UV Light	Glycerol Layer: Added between Mask and SU-8 Layer on the Wafer
7	Exposure to UV Light	Extra Time (More Energy): 4 seconds at 100% Intensity
8	All Changes	All changes as wafer 2-7

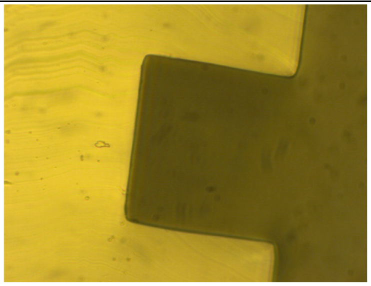
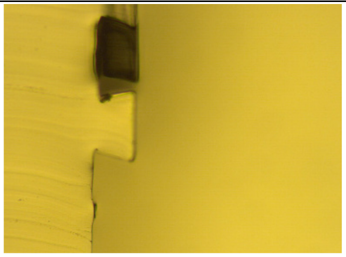
B.5. Results

Cross-sectional shape was tested by pouring polydimethylsiloxane (PDMS) on the wafer and imaging the cross-section of a cut channel. The resulting cross-sectional shapes of the 100 μm × 100 μm channels varied in the degree of how much the walls were perpendicular. Cross-sectional images are shown in Table 26, with the white/lighter portion being the cut PDMS channel from the respective wafer.

Table 26. Cross-sectional images for wafers corresponding to changes in the wafer processing techniques.[1]

Wafer #	**Wafer Processing Step Changed** ***And Changes Made***	**Cross-Section of 100 µm × 100 µm Channel at 40× Magnification**
1	None (Control)	 Distance 171.083 µm Distance 103.621 µm Distance 116.887 µm
2	Wafer Preparation *Heating Wafer to Dry: at 140 °C for 3 minutes*	
3	Between Spin Coating and Soft Bake *Relaxation Time: 30 minutes*	 Distance 129.244 µm Distance 104.615 µm Distance 81.945 µm
4	Soft Bake *Higher Temperature for Longer Time: 100 °C for 100 minutes*	
5	Between Soft Bake and Exposure to UV Light *Relaxation Time: 10 minutes*	
6	Between Soft Bake and Exposure to UV Light *Glycerol Layer: Added between Mask and SU-8 Layer on the Wafer*	 Distance 254.813 µm Distance 105.215 µm Distance 104.793 µm

[1] Images of wafer #1, #3, #4 and #6, courtesy of Mark Stroobach, University of Ottawa.

Wafer #	**Wafer Processing Step Changed** ***And Changes Made***	**Cross-Section of 100 µm × 100 µm Channel at 40× Magnification**
7	Exposure to UV Light *Extra Time (More Energy): 4 seconds at 100% Intensity*	
8	All Changes *All changes as wafer # 2-7*	 *Channels were broken/widened so much that the SU-8 features merged or broke. This image is from 10× zoom.

Images of wafers #2 to #8 were compared to the image of the cross-section. It can visually be seen that wafer #6 and wafer #7 contain channels that are relatively the most perpendicular. Since wafer #8 went through all of the procedure steps, it might have accumulated effects from all the changes and caused the SU-8 channel features to merge together. Using all the changes would not create desired perpendicular channels.

B.6. Conclusions

While the sample size and changes to the wafer processing procedure were not robustly investigated, the changes created channels more perpendicular in their cross-sectional shape. Further testing and a larger sample size would be necessary to quantify the degree to which these changes made to the processing technique. The addition of glycerol between the mask and SU-8 layer, along with the longer exposure time, were used for the creation of the wafer moulds. These changes seemed to create channels which were more perpendicular in cross-section.

Appendix C Standard Operating Procedure (SOP) for Pressure-Flow System

C.1. Disclaimer

This document, and those to which it refers, have been created in reference to Dr. Fenech's laboratory and experimental procedures in mind. Its use for other laboratories, procedures or researchers should be modified or remade accordingly. Researchers who are tailoring their procedure should check with appropriate safety coordinators (e.g. Pierre Laflamme at uOttawa), supervisors/PI's, and/or lab mates to verify safe handling.

C.2. Purpose

This standard operating protocol outlines the procedure to safely operate the pressure-flow control system.

C.3. References

Consulted with Fluigent with respect to cleaning methods for biological fluids like blood.

FLUIGENT, Flow-Rate Platform User Manual, 15/03/14. FLUIGENT, 2014.

C.4. Definitions

Pressure-flow control system – A system of flow sensors, pressure introduction system and accompanying software. Using a feedback loop with software allows for a certain pressure to be introduced to the sample, or for a specified flow rate, at selected time intervals.

MFCS – Microfluidic Flow Control System: takes an input pressure and outputs a desired pressure with stability to up to four outlet ports.

FRP – Flow Rate Platform: measures the flow rate from a selected sensor and converts it to a signal that the computer can process into a flow rate.

C.5. Materials Required

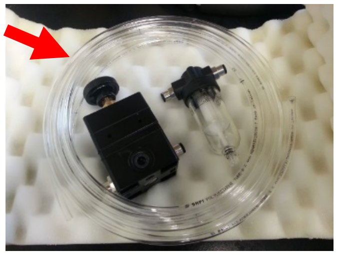
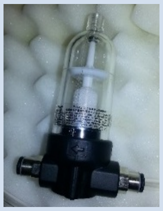
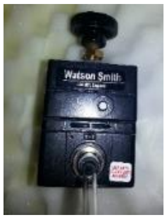
The required materials depend on which aspect of the microfluidic control system is needed. For any system to be used, a computer and the appropriate software needs to be installed. Refer to First-Time Use Section C.6 for details about the first-time use. The pressure control system is required to introduce air pressure to an experiment, such as moving a fluid or maintaining a certain pressure. The fluidic selection controller allows for either (1) the flow of a select a single fluid from multiple sources to flow to an output or (2) the flow of a sample out to a selected output. Flow rate sensor measurement allows the flow rate through a flow rate sensor to be measured of a calibrated fluid. The differential pressure and temperature can be measured using a module connected to pressure and temperature sensors.

The tables below list the equipment necessary for the respective functionality required, and their locations (if a set place is designated). The pressure controller component of the system is usually left connected, but could be kept safely in the designated drawer labelled “Pressure Control System” with other equipment used with it. Note that extra personal protective equipment should also be worn as required, and following the respective safety procedures, for any hazardous fluids (E.g. blood). The following subsections are ordered as they appear in the above paragraph’s list.

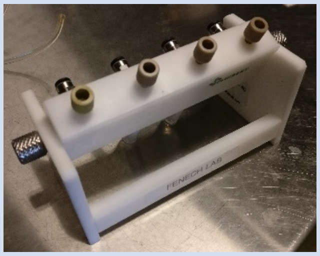
C.5.1. Computer

A desktop computer is located on the countertop in D509, but the software can be installed on a personal laptop as well.

C.5.2. Pressure Control System Equipment

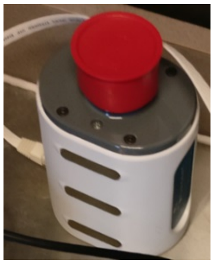
Equipment	Location
Compressed Dry Air Supply	Has the appearance of a tube with a valve at the end and usually has a quick connect adapter connected at the end, located in D509. Note: The MFCS-EZ Pressure Controller requires a minimum pressure of 3400 mbar (340 kPa) and nominal operating pressure of 7400 mbar (740 kPa).
 **6 mm × 4 mm (O.D. × I.D.) Flexible Tubing**	Usually left connected to quick connect adapter of the in-house compressed air line and air drier on the countertop in D509
 **Air Drier**	Usually left connected to 6 mm × 4 mm (O.D. × I.D.) flexible tubing on the countertop in D509
 **Manual Pressure Regulator**	Usually left connected to 6 mm × 4 mm (O.D. × I.D.) flexible tubing and flexible tubing with MFCS-EZ metal adapter on the countertop in D509

Equipment	Location
 6 mm × 4 mm (O.D. × I.D.) Flexible Tubing with MFCS-EZ Metal Adapter	Usually left connected to manual pressure regulator and MFCS-EZ on the countertop in D509
 MFCS-EZ Pressure Controller	Usually left connected to 6 mm × 4 mm (O.D. × I.D.) flexible tubing with MFCS-EZ metal adapter on the countertop in D509
MFCS-EZ Power Supply	Usually left connected to an AC outlet and the MFCS-EZ Pressure Controller on the countertop in D509
Universal Serial Bus (USB) Type "A to B" Cable	Usually left connected to the MFCS-EZ Pressure Controller on the countertop in D509
 USB Hub	Usually left connected to the USB "A to B" type cable on the countertop in D509
4 mm × 2.5 mm (O.D × I.D.) Flexible Tubing	On top of MFCS-EZ Pressure Controller in D509

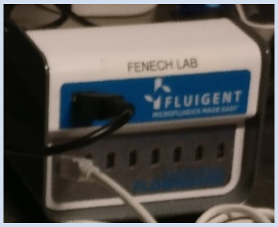
Equipment	Location
 **Fluiwell Sample Holder and Pressure Router**	On top of MFCS-EZ Pressure Controller in D509
Plastic Sample Vials with Threaded Necks	In a benchtop container in D511 (along with corresponding vial caps)

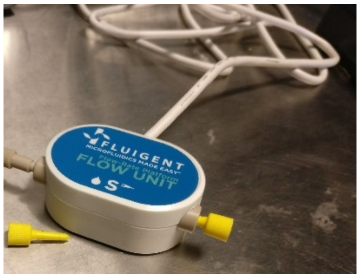
C.5.3. Fluidic Selection Controller Equipment

Equipment	Location
 Switchboard	In drawer labelled "Pressure Control System" in D509
Switchboard Power Supply	In drawer labelled "Pressure Control System" in D509
USB Type "A to B" Cable	In drawer labelled "Pressure Control System" in D509
 USB Hub	Usually left on the countertop in D509
Switchboard Communication Cable	In drawer labelled "Pressure Control System" in D509

Equipment	Location
 **M-Switch**	In drawer labelled "Pressure Control System" in D509
1/16" × 1/32" **(1.5875 mm × 0.79375 mm)** **O.D. × I.D. Rigid Tubing**	In drawer labelled "Pressure Control System" in D509
M-Switch Plastic Fluidic Plugs	In a plastic bag within the drawer labelled "Pressure Control System" in D509
 ESS Control Software License USB Dongle	Usually left connected to the USB hub on the countertop in D509

C.5.4. Flow Rate Sensor Measurement Equipment

Equipment	Location
 **FLOWBOARD – Flow Sensor Board**	On beside MFCS-EZ Pressure Controller on countertop in D509

Equipment	Location
 **"XS" and/or "S" Flow Sensor**	In drawer labelled "Pressure Control System" in D509
USB Type "A to B" Cable	In drawer labelled "Pressure Control System" in D509
 USB Hub	Usually left connected to the USB "A to B" type cable on the countertop in D509
Brown Ferrule Adapters	In drawer labelled "Pressure Control System" in D509
1/32" × 1/64" **(0.79375 mm ×0.396875 mm) Rigid Tubing**	In drawer labelled "Pressure Control System" in D509

C.5.5. Pressure and Temperature Sensor Equipment

Due to the pressure and temperature sensor requiring the use of custom software and GUI, the equipment to use this system is located in the “MATLAB GUI for Computing Apparent Viscosities” SOP.

C.6. First-Time Use

The first time that the system is used, with any combination of the subsystems, it is recommended to install and setup all the software required. The following describes the software, their installation files’ locations, and the procedure for correct setup. The procedure allows apps to run in Administrator mode which the Script Module needs.

Note: Refer to the “MATLAB GUI for Computing Apparent Viscosities” SOP if that subsystem is required.

C.6.1. Software Location

Access to the manual and installation files for the systems requires access to the laboratory network accessed storage (NAS), either by direct connection or through a laboratory computer with access rights. The installation files, and a brief summary of this first-time use procedure, are located in the following directory:

\\Volume_1\SHARE\Manuals and Installation Files\Fluigent USB

Note: The SDK folder is only for interfacing between MATLAB and/or LABVIEW

C.6.2. Software Setup

1. Install the following setup files by right-clicking the following setup files (in the corresponding folder name) and selecting "Run as Administrator":
    a. MAESFLO 3.2.1
        i. "setup.exe"
    b. ESS Control
        i. "setup.exe"

   c. Flow Rate Platform
      i. "setup.exe"
   d. Fluigent Script Module
      i. "Fluigent_ScriptModule_Installer.exe"
2. After installing the software, locate the following software shortcuts and put them in a single folder (to keep them all together). These shortcuts are usually on the Desktop, but could be found in the Start Menu.
   a. MAESFLO 3.2.1
   b. ESS Control 1.0.1
   c. Flow-Rate Platform
   d. SciTE.exe
3. For each of these shortcuts, right-click each shortcut and follow:
   a. "Properties"
   b. "Compatibility" tab
   c. Check "Run this program as an administrator"
   d. Click “Apply”
   e. Click “OK”

C.7. Preparation

1) Check that the components of the system to be used are available for use in the laboratory calendar and with other lab users.
2) Refer to First-Time Use Section C.6 for the first-time use for software setup of the computer.
3) Read and understand the necessary training or safety procedures required for the materials or environment that the system will be used with.
4) If the pressure and temperature sensor module is required to be used, refer also to “MATLAB GUI for Computing Apparent Viscosities” SOP.
5) Allocate necessary room required for the use of all the equipment components (including a laptop or desktop computer) for the experiment.
6) Ensure you have allotted enough time to do the whole procedure (e.g. going to washroom beforehand).

C.8. Subsystems

Equipment required for the procedure depend on which subsystem is being used. All subsystems are not required to be connected in order to run, i.e. the pressure controller system does not need the fluidic selection controller to function.

C.8.1. Pressure Control System (MFCS-EZ Controller)

1) Connect one end of the 6 mm × 4 mm (O.D. × I.D.) flexible tubing, shown in Figure 75, to the in-house compressed air outlet (using the quick connect adapter at the end). The in-house pressure is sufficient at 6 bar (600 kPa).

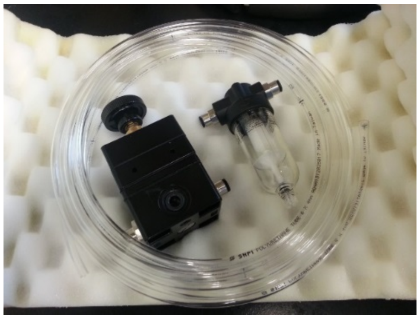


Figure 75. Initial package of (1) 6 mm × 4 mm (O.D. × I.D.) flexible tubing wrapped around (2) a manual pressure regulator to the left and (3) air drier to the right.

Note: It is recommended to run compressed air through this tubing, and other tubing used in this procedure, if they have not been left connected. This can be done by taking the tubing and connecting it temporarily to the compressed air quick connect, and slowly opening the valve to let some air evacuate the tubing of particles. Close the valve after clearing each tube.

2) Connect the 6 mm × 4 mm (O.D. × I.D.) flexible tubing to the air drier, as shown in Figure 76 (left). The flow direction of the gas should follow the arrow, as shown in Figure 76 (right).

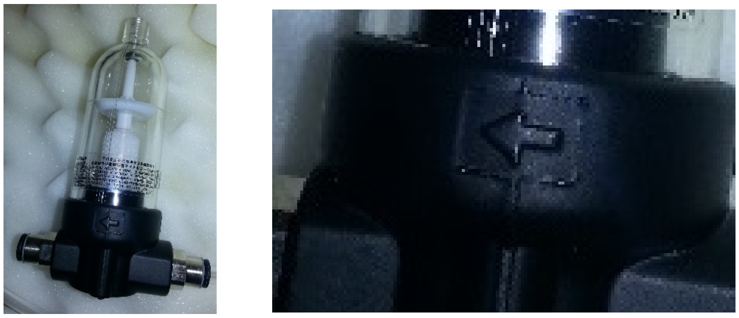


Figure 76. Air drier component (left) with the arrow indicating in which direction the air should flow (right).

3) Connect a small segment of the 6 mm × 4 mm (O.D. × I.D.) flexible tubing to the air drier and connect the other end to manual pressure regulator, as shown in Figure 77. The flow into the regulator should be at the side labelled 'IN', also indicated by a white circular dot, as shown in Figure 77 (right).

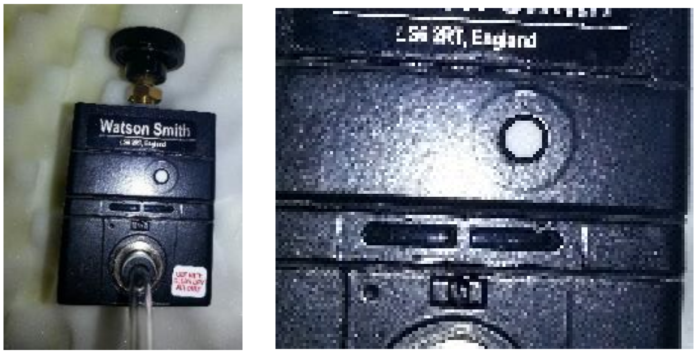


Figure 77. Manual pressure regulator (left) with the white circular dot indicating which side the air should go into (right).

4) Connect the pressure tube, as shown in Figure 78 (left), with a MFCS-EZ metal adapter, shown in Figure 78 (right) to the manual pressure regulator – using the end without the adapter.

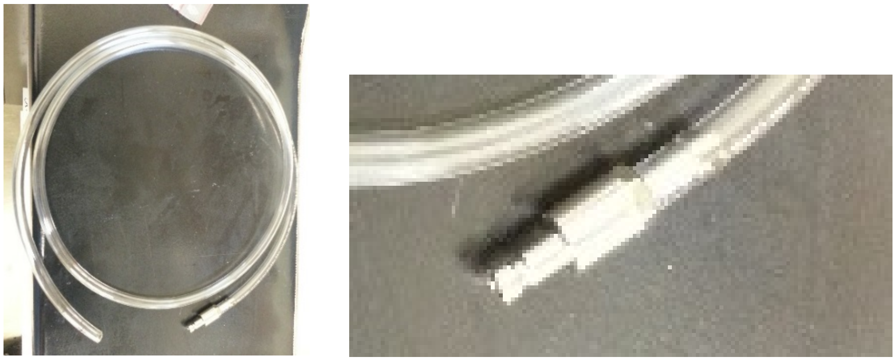


Figure 78. 6 mm × 4 mm (O.D. × I.D.) flexible tubing (left) with the MFCS-EZ metal adapter connected at one end (right).

5) Connect metal tubing adapter of the tubing into MFCS-EZ port, as shown in Figure 79 (right). Push the connector in until it clicks.

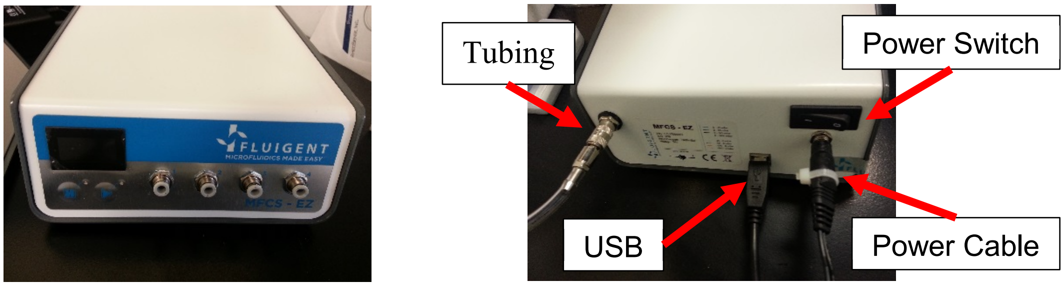


Figure 79. MFCS-EZ controller front (left) and back with connected flexible tubing (right), USB cable and power.

6) Plug in the power supply of the MFCS-EZ into an AC outlet.
7) Connect the cable end of the power supply and USB cable into MFCS-EZ, as shown in Figure 79 (right).
8) Connect the other end of the USB cable to computer. This connection can be done through the USB hub, as shown in Figure 80.

Figure 80. USB hub to connect multiple devices through the USB cable to a computer.

9) Turn on MFCS–EZ in the back, using the power switch, as depicted in Figure 79 (right). The pressure gauge should show values and a red LED above the pause button.
10) Press the Play button, indicated in Figure 81 (right), to let air into the system. The pressure (in bar) will now display the air pressure.

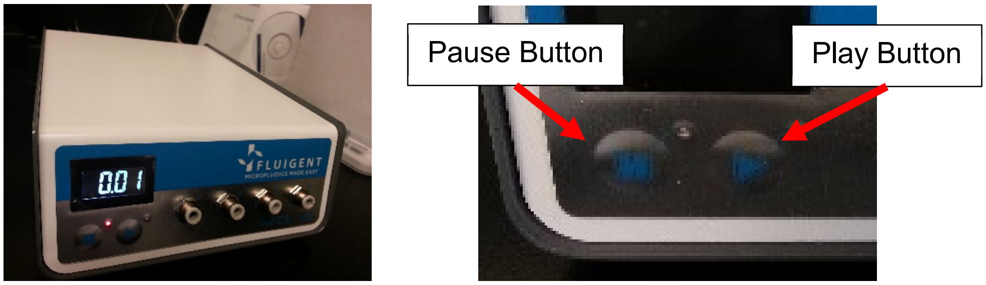


Figure 81. MFCS–EZ with the pressure display showing the input pressure (left) and the pause and play buttons on the front face of the MFCS–EZ (right).

11) Open air valve on the compressed air line to let air into system.

12) Turn manual pressure regulator, as shown by Figure 82, to allow pressure into the system.

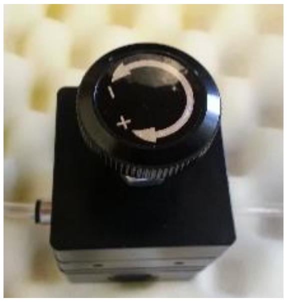


Figure 82. Manual pressure regulator knob with + indicating the direction in which to turn in order to increase the output pressure, and – to decrease the output pressure.

13) Run MAESFLO 3.2.1 software using the shortcut, as shown in Figure 83.

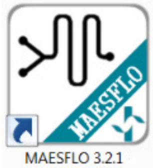
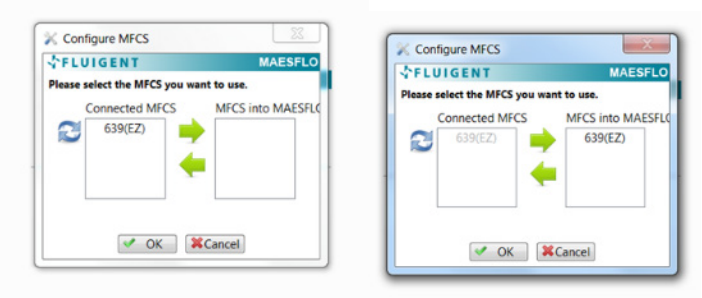


Figure 83. MAESFLO 3.2.1 shortcut icon to run the program.

Note: The program might pop-up in the background, minimize all other windows to see the software.

14) The first time MAESFLO 3.2.1 is run, a few prompts will pop-up asking to create new directories. Click "OK" to continue.

15) If a system is not already selected, as shown in Figure 84 (left), select the MFCS–EZ to use on the left and click the right arrow to move it to the left, as shown in Figure 84 (right), and click "OK".


Configure MFCS

Please select the MFCS you want to use.

Connected MFCS: 639(EZ)

MFCS into MAESFLO:

Configure MFCS

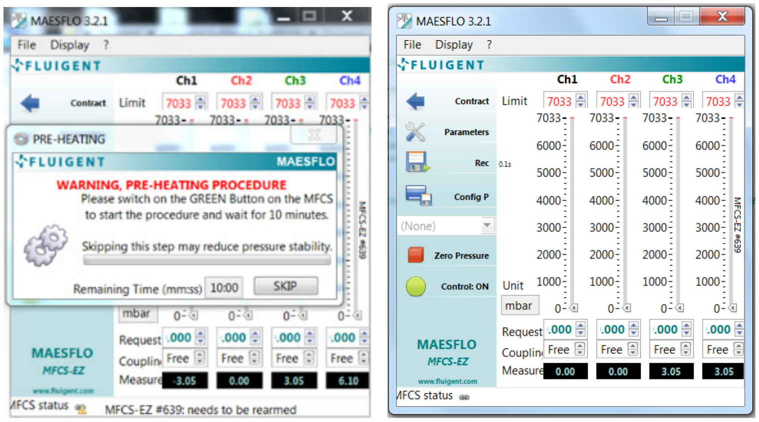
Please select the MFCS you want to use.

Connected MFCS: 639(EZ)

MFCS into MAESFLO: 639(EZ)

Figure 84. First-time use of MAESFLO 3.2.1 boxes with no pressure system selected (left) and after selecting a connected MFCS-EZ (right).

16) Every time the program is restarted, a “WARNING, PRE-HEATING PROCEDURE” will pop-up and automatically run for 10 minutes, as shown in Figure 85 (left). Once finished, the control screen will be presented to you for controlling the MFCS-EZ, as shown in Figure 85 (right). While waiting for this step to complete, the experiment can be set up.


PRE-HEATING

WARNING, PRE-HEATING PROCEDURE
Please switch on the GREEN Button on the MFCS
to start the procedure and wait for 10 minutes.

Skipping this step may reduce pressure stability.

Remaining Time (mm:ss) 10:00 SKIP

MFCS-EZ #639: needs to be rearmed

Figure 85. MAESFLO 3.2.1 pre-heating procedure pop-up upon initialization (left) and after pre-heating (right).

17) Complete the calibration of the MFCS-EZ controller, as shown in Figure 86, by selecting “File” → “Calibration” and following the prompts. Once finished (it should take ~10 seconds), the output pressures will be calibrated for 0 pressure (as there are no connected samples or pressures).

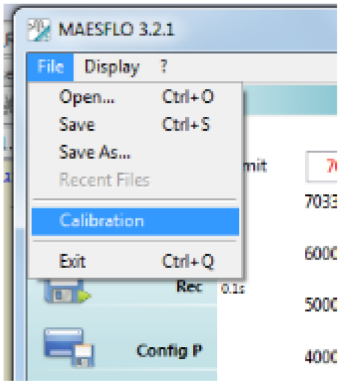


Figure 86. Calibration of the MFCS-EZ controller upon first use of the day.

18) Take the Fluiwell, as shown in Figure 87, and connect 4 mm × 2.5 mm (O.D × I.D.) flexible tubes to the quick connect ports in quantity to match the number of sample fluids.

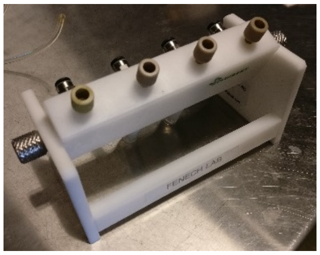


Figure 87. Fluiwell used to hold sample vials and route pressure to sample fluids.

19) Connect the other end of the tube and connect it to the MFCS-EZ controller and thread the plastic vials containing your sample fluids to the Fluiwell, as shown in Figure 88.

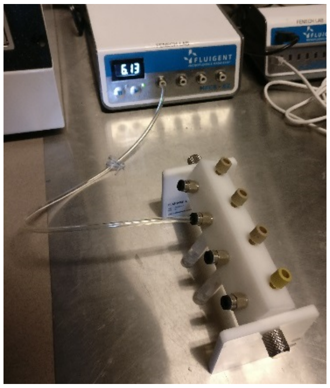


Figure 88. Fluiwell with plastic sample vials connected, with a 4 mm × 2.5 mm (O.D × I.D.) flexible tubing connecting it to the MFCS-EZ controller.

20) Loosen the tubing adapters at the top of the top of the Fluiwell to allow 1/16" (1.5875 mm) O.D. tubing to enter the sample vial.

    Note: Ensure that the end of the tube is sufficiently below the surface of the fluid, but not touching the bottom of the vial. Having the end at the bottom can cause the fluid path to be obstructed and make the fluid flow through a very tiny gap.

21) Tighten the tubing adapter enough so that the tube can no longer move up and down through the hole in the adapter.

    Note: Do not over-tighten these adapters, since it will permanently deform the plastic and prevent removal or insertion of tubing.

22) Connect the other end of the 1/16" (1.5875 mm) O.D. tubing to the experimental setup as appropriate.

23) The program can be controlled with the self-explanatory buttons, as shown in Figure 89, to make measurements and control the pressure of the fluid.

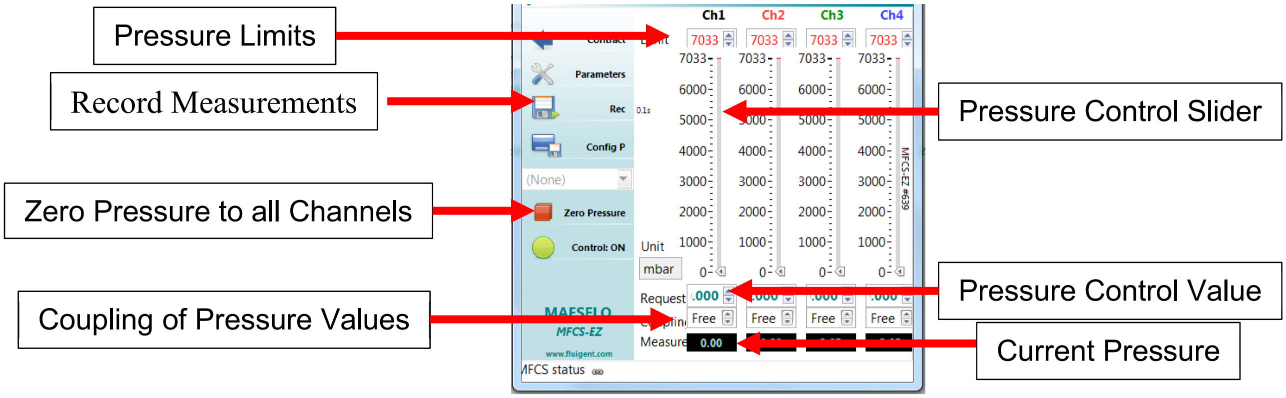


Figure 89. Main control setting for the MAESFLO 3.2.1 software.

C.8.2. Fluidic Selection Controller (Switchboard)

1) Plug in the Switchboard power adapter and USB type “A to B” cable to the Switchboard, as shown in Figure 90. Note: The power source connector can be difficult to plug fully, but ensure it is pushed in all the way.

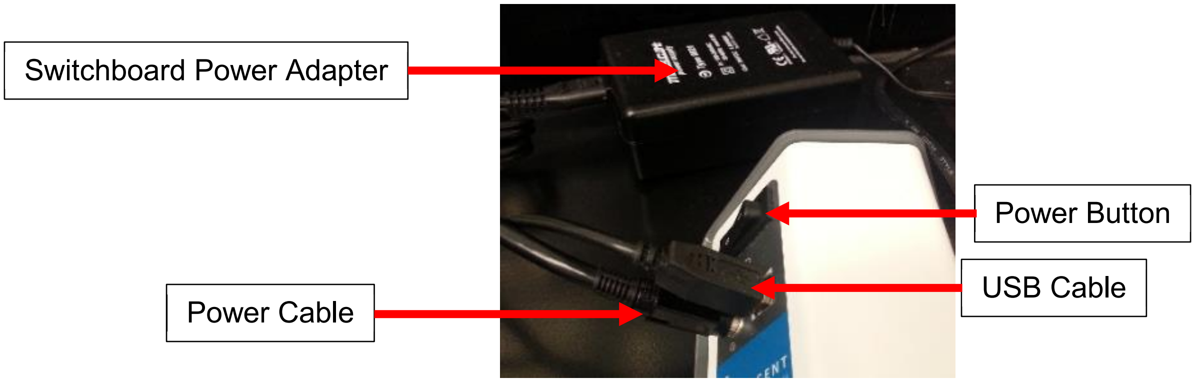


Figure 90. Switchboard’s location of connections, the power adapter and the power button.

2) Connect the communication cable a corresponding port in the M-switch area of the Switchboard, as indicated by Figure 91.

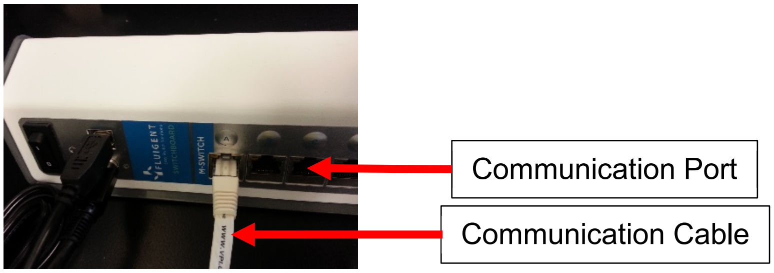


Figure 91. Connection of communication cable to the Switchboard.

3) Connect the other end of the communication cable to the M-switch, as shown in Figure 92.

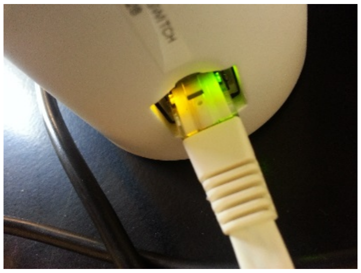


Figure 92. Connection of the communication cable to the M-Switch.

4) Determine how many inlet/outlet fluids are necessary for the experiment and obtain enough 1/16” × 1/32” (1.5875 mm × 0.79375 mm) O.D. × I.D. rigid tubing for connections.
5) Gather enough M-Switch plugs to fill the rest of the M-Switch top ports (there should be 11 fluidic connections, either tubing or plugs, in total).
6) Using a permanent marker, make a line near the end of the plugs and tubing, 2.4 cm from the end which will be inserted into the M-Switch. This indication line will ensure that the tubes are sufficiently connected to the M-Switch (which requires about 2.4 cm of tubing for a good connection.
7) Remove the red protective cap from the top of the M-Switch.
8) Loosen the fluidic connection ports of the M-Switch by twisting the outer grey ring of the device (counter-clockwise looking from the top).
9) Insert all rigid tubing which will be used to flow liquid, ensuring enough tubing is inserted. Note: It is recommended that the center tubing be inserted before the surrounding ports are filled. The tubing should be held close to the port as it is being pushed in to avoid the buckling of the tubing. The tubing will push past a locking feature in the device and can be felt during the insertion of the tubing.
10) Insert the plastic fluidic plugs (with the same technique as in step 9) in the rest of the ports.
11) Connect the ESS software license USB dongle to the computer, through the USB hub (if using it), as shown in Figure 93.

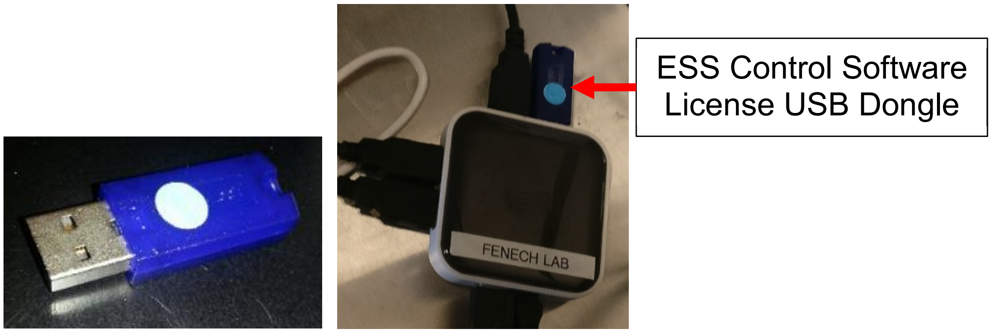


Figure 93. ESS Control Software License USB Dongle (left) and connected to the USB hub (right).

12) Connect the USB type "A to B" cable to the computer through the USB hub (if the computer is connected to the USB hub).

13) Turn on the power using the power switch, as shown in Figure 90.

    Note: It is normal for the M-Switch to make a short click when it is connected to the switchboard.

14) Check the connection by pressing on the button above the connected cable, as shown in Figure 94.

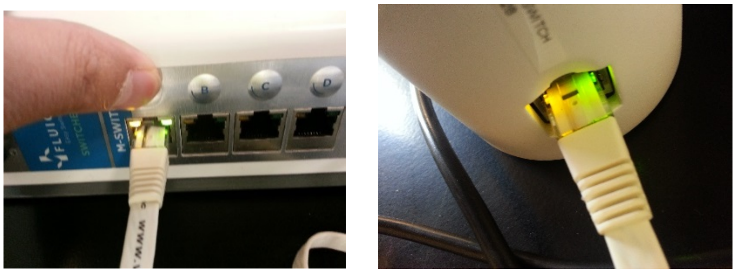


Figure 94. Checking the connection of the M-Switch by pressing the button above the connection cable (left) to check that the green light turns on for the Switchboard and the M-Switch (right).

15) Run ESS Control 1.0.1 software using the shortcut, as shown in Figure 95.

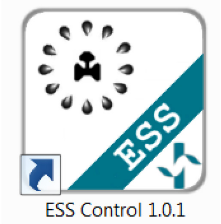


Figure 95. ESS Control 1.0.1 shortcut icon to run the program.

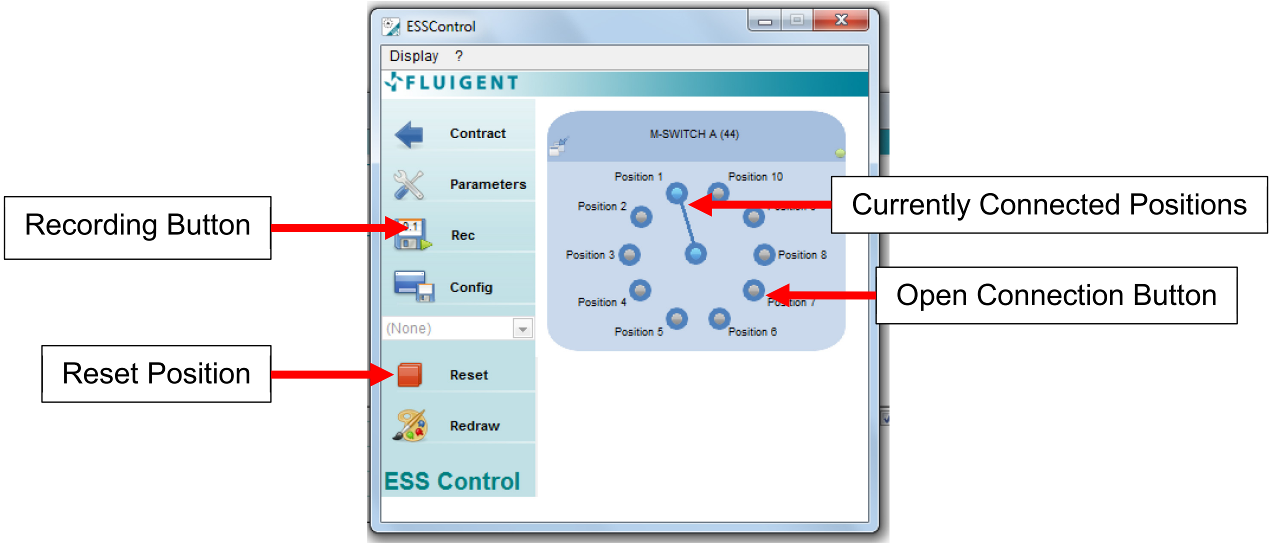


Figure 96. ESS Control software interface, illustrating possible control options for the M-Switch.

C.8.3. Flow Rate Sensor Measurement (FRP Controller)

1) Connect one end of the USB type “A to B” cable to the FLOWBOARD, as shown in Figure 97 and the other to the computer (through the USB hub if needed).

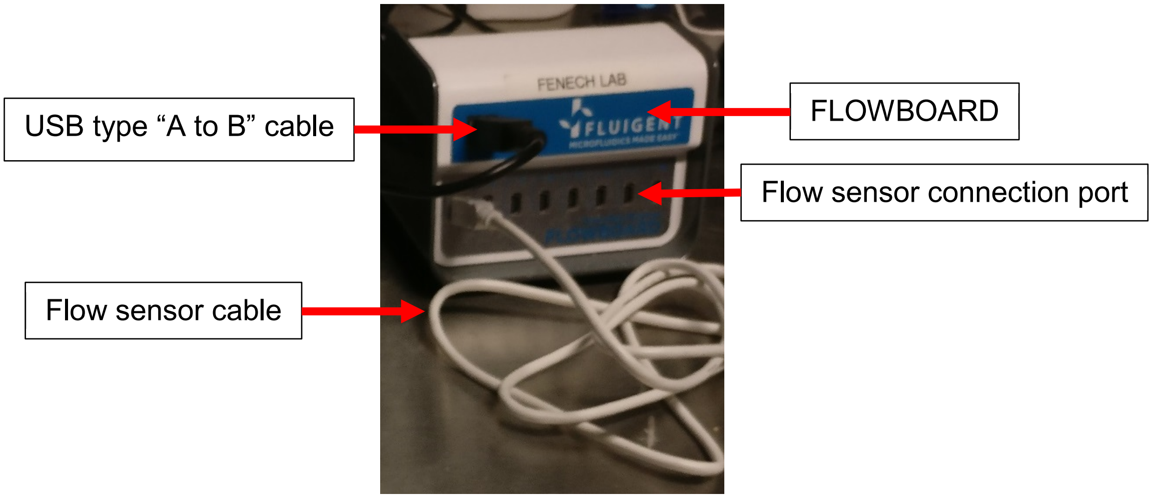


Figure 97. Flowboard with connections of USB type “A to B” and flow sensor.

2) Connect desired flow sensors, as shown in Figure 98, to the FLOWBOARD, as shown in Figure 97.

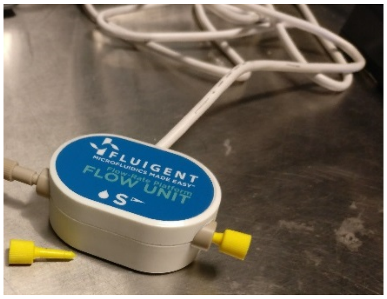


Figure 98. "S" flow unit model of the flow sensor.

3) Check the connection between the FLOWBOARD USB to computer (the green LED should be on).

4) Remove the yellow plugs from both ends of the flow sensor by pulling straight out. Keep these in a clean and safe place.

5) Insert the brown ferrule adapters over the 1/32" × 1/64" (0.79375 mm × 0.396875 mm) rigid tubing (one for each side of the sensor), as shown in Figure 99.

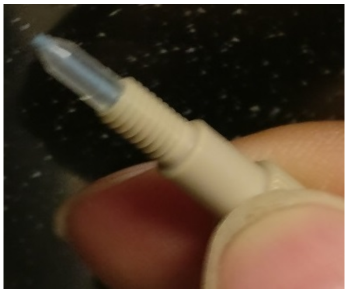


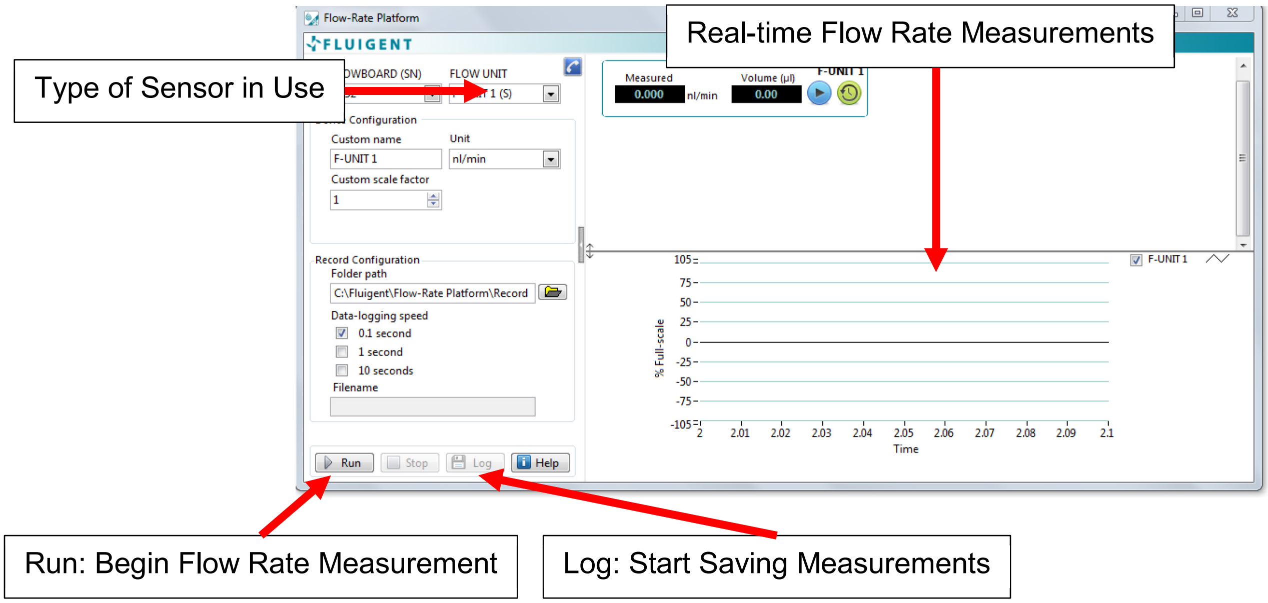
Figure 99. Brown ferrule adapter with 1/32" × 1/64" (0.79375 mm × 0.396875 mm) rigid tubing inserted

6) Connect the rigid tubing to the intended flow to be measured.

7) Run Flow Rate Platform software using the shortcut, as shown in Figure 100.

Figure 100. Flow Rate Platform shortcut icon to run the program.

8) The program can be controlled with the self-explanatory buttons, as shown in Figure 101, to control when to begin viewing and/or saving flow rate measurements.



Type of Sensor in Use

Real-time Flow Rate Measurements

Run: Begin Flow Rate Measurement

Log: Start Saving Measurements

Flow-Rate Platform

FLUIGENT

Measured 0.000 nl/min

Volume (µl) 0.00

Configuration

Custom name: F-UNIT 1

Unit: nl/min

Custom scale factor: 1

Record Configuration

Folder path: C:\Fluigent\Flow-Rate Platform\Record

Data-logging speed

☑ 0.1 second

□ 1 second

□ 10 seconds

Filename

Figure 101. Flow Rate Platform software interface, illustrating possible control options for the flow rate sensor.

C.8.3.1. Scale Factor Calibration

The flow sensor measures the flow rate by adding a small amount of heat between two temperature sensors and measuring the temperature spread between the sensors [75]. The sensor requires calibration for fluids other than the factory calibration of water due to this heat method of measuring the flow rate. Following the calibration protocol from Fluigent [75], a scale factor for each fluid should be created each day of testing.

1) Fill a glass syringe with the fluid intended to be used in experiments.
2) Load the syringe on a syringe pump.
3) Connect desired flow sensor to the syringe pump.
4) Infuse through the sensor using the fluid intended to be used in experiments. The infusion should be completed at several flow rates within the platform’s range. E.g. 25, 50, 75, and 100% of the sensor’s flow rate range.
5) Once the flow rate stabilizes, record the flow rate and average the viscosity values over a span of 1 to 2 minutes.

6) Repeat steps 4 and 5 until the whole range of flow rates has been tested in the intervals.

7) Plotting the syringe pump's flow rate, taken as the actual flow rate, against the sensor's measurement, as shown in Figure 102.

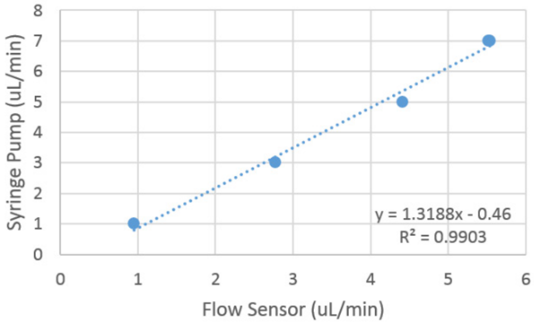


Figure 102. Determination of scale factor for a fluid by calculating the slope as the change in syringe pump value (actual) flow rate, with the flow rate sensor's flow rate measurement.

8) Calculating the slope, as also shown in Figure 102, of the actual flow rate with respect to the flow rate sensor measurement.

9) Use this scale factor to correct the measured flow rate, by the flow rate sensors, for the respective fluids to the correct rates.

C.8.4. Pressure and Temperature Sensor Module

Due to the pressure and temperature sensor requiring the use of custom software and GUI, the equipment to use this system is located in the "MATLAB GUI for Computing Apparent Viscosities" SOP.

C.8.5. Programmable Control Scripts

1) Run SciTE Script Editor software using the shortcut, as shown in Figure 103.

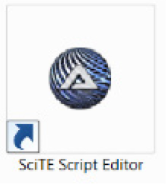


Figure 103. SciTE Script Editor shortcut icon to run the program.

Note: If the shortcut cannot be found, the program can be found:

Start Menu → AutoIT v3 → SciTE Script Editor

2) Open the programmed script using: File → Open, as shown in Figure 104, and navigate to the file path.

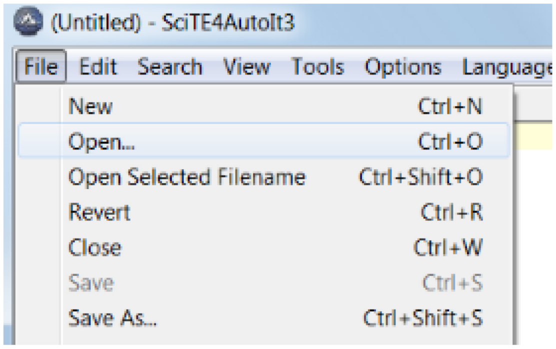
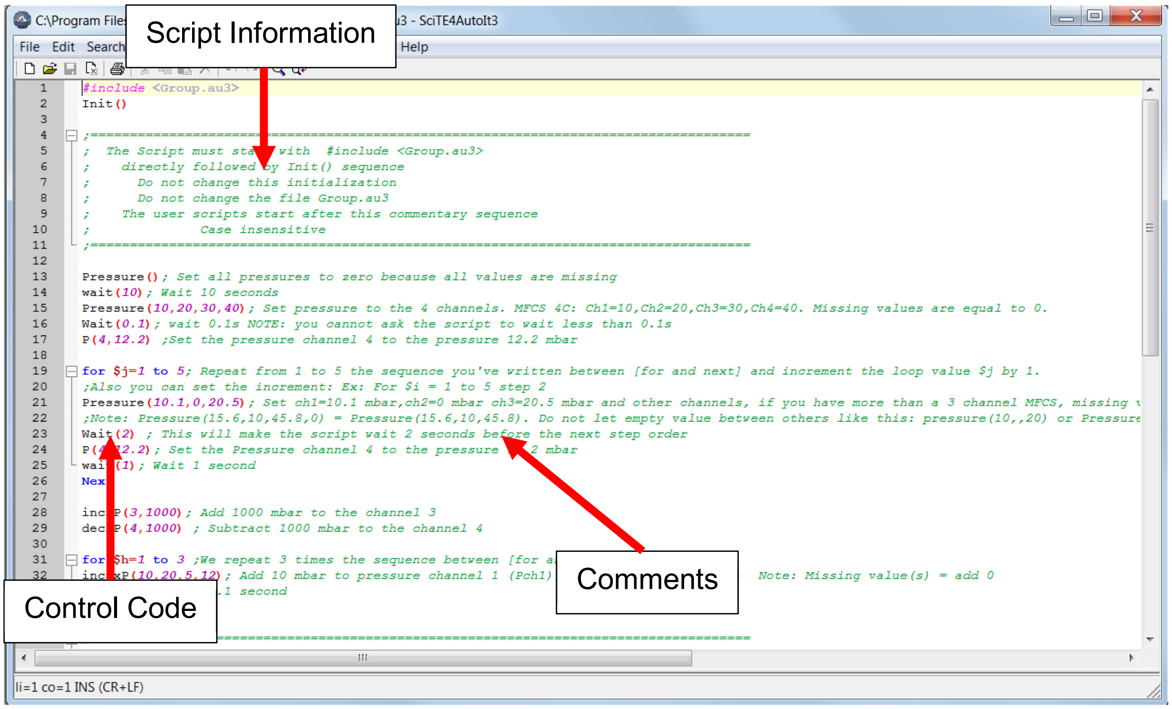


Figure 104. Navigation to the option to open a script file or example script file.

Note: There are example files to help understand the syntax and commands that can be used in the default directory of the "Open…" option.

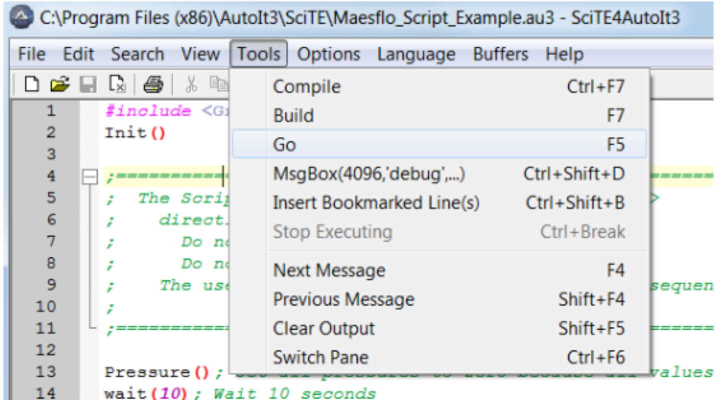
3) The script control code can be edited in the editing window, as shown in Figure 105, until the intended experimental procedure is created.



```
#include <Group.au3>
Init()

;=================================================================================
;  The Script must start with  #include <Group.au3>
;    directly followed by Init() sequence
;      Do not change this initialization
;      Do not change the file Group.au3
;    The user scripts start after this commentary sequence
;              Case insensitive
;=================================================================================

Pressure(); Set all pressures to zero because all values are missing
wait(10); Wait 10 seconds
Pressure(10,20,30,40); Set pressure to the 4 channels. MFCS 4C: Ch1=10,Ch2=20,Ch3=30,Ch4=40. Missing values are equal to 0.
Wait(0.1); wait 0.1s NOTE: you cannot ask the script to wait less than 0.1s
P(4,12.2) ;Set the pressure channel 4 to the pressure 12.2 mbar

for $j=1 to 5; Repeat from 1 to 5 the sequence you've written between [for and next] and increment the loop value $j by 1.
;Also you can set the increment: Ex: For $i = 1 to 5 step 2
Pressure(10.1,0,20.5); Set ch1=10.1 mbar,ch2=0 mbar ch3=20.5 mbar and other channels, if you have more than a 3 channel MFCS, missing v
;Note: Pressure(15.6,10,45.8,0) = Pressure(15.6,10,45.8). Do not let empty value between others like this: pressure(10,,20) or Pressure
Wait(2) ; This will make the script wait 2 seconds before the next step order
P(4,12.2); Set the Pressure channel 4 to the pressure 12.2 mbar
wait(1); Wait 1 second
Next

incP(3,1000); Add 1000 mbar to the channel 3
decP(4,1000) ; Subtract 1000 mbar to the channel 4

for $h=1 to 3 ;We repeat 3 times the sequence between [for a
incxP(10,20,5,12); Add 10 mbar to pressure channel 1 (Pch1)                Note: Missing value(s) = add 0
.1 second

=================================================================================
```

Figure 105. Flow Rate Platform software interface, illustrating possible control options for the flow rate sensor.

4) The program can be started using the "Go" command, as shown in Figure 106, or by pressing the "F5" button.


C:\Program Files (x86)\AutoIt3\SciTE\Maesflo_Script_Example.au3 - SciTE4AutoIt3
File
Edit
Search
View
Tools
Options
Language
Buffers
Help
Compile Ctrl+F7
Build F7
Go F5
MsgBox(4096,'debug',...) Ctrl+Shift+D
Insert Bookmarked Line(s) Ctrl+Shift+B
Stop Executing Ctrl+Break
Next Message F4
Previous Message Shift+F4
Clear Output Shift+F5
Switch Pane Ctrl+F6

```
#include <G
Init()

;==========                                        ======
;  The Scri                                        >
;     direct.
;        Do n
;        Do n
;     The us                                       sequen
;
;==========                                        ======

Pressure() ;                                       values
wait(10) ; Wait 10 seconds
```

Figure 106. The "Go" command, which is located in the Tools option, tests and compiles the script program before running the script.

C.9. Clean-Up

In general, the clean-up procedure is the reverse process of the procedure for use in an experiment. Care must be taken to ensure equipment that touches the sample fluid is properly cleaned. Steps to clean up the systems are in the following subsections and gives details for steps requiring more detail in the clean-up. The subsection for the pressure control system is located at the end since the clean-up of other subsystems is facilitated with the use of the pressure controller system.

C.9.1. Fluidic Selection Controller (Switchboard)

1) Use the appropriate fluid cleaning techniques and flow appropriate cleaning solutions through the M-Switch connections.

   E.g. For blood:

   a. Flow a few millilitres of Phosphate Buffered Saline (PBS) through the tubing and connections
   b. Flow a few millimetres of 10% bleach solution
   c. Flow de-ionized water through the tubing and connections
   d. Check that the outlet waste is void of blood, such as using blood test strips to test for haemoglobin

2) Once each fluidic connection is clean, it is recommended to flow isopropyl alcohol (to avoid mineral or biofilm build up) through each fluidic connection, followed by compressed air.

3) Reset the connection to that of position 1.

4) Close the software.
5) Turn off the Switchboard using the power button on its front.
6) Disconnect the USB type "A to B" cable from the computer and the Switchboard.
7) Loosen the fluidic connection ports of the M-Switch by twisting the outer grey ring of the device (counter-clockwise looking from the top) and remove all the inserted tubing or plastic fluidic plugs.
8) Place the red protective cap on the top of the M-Switch.
9) Clean the M-switch with the appropriate method according to the sample fluid.
    a. For blood, use a 10% bleach solution to wipe down all of the surfaces of the M-switch and the M-switch plastic fluidic plugs.

If this system is to be fully removed:

10) Disconnect the communication cable between the M-Switch and the Switchboard.
11) Disconnect the AC adapter from the AC outlet and remove the power cable from the Switchboard.
12) Store the M-Switch, plastic fluidic plugs, Switchboard, connection cable, USB type "A to B" cable, and AC adapter in the drawer labelled "Pressure Control System" in D509.

C.9.2. Flow Rate Sensor Measurement (FRP Controller)

1) Use the appropriate fluid cleaning techniques and flow appropriate cleaning solutions through the flow sensor.

    E.g. For blood:

    a. Flow a few millilitres of PBS through the flow sensor
    b. Flow a few millimetres of 10% bleach solution
    c. Flow de-ionized water through the flow sensor
    d. Check that the outlet waste is void of blood, such as using blood test strips to test for haemoglobin
2) Once the flow sensor is clean, it is recommended to flow isopropyl alcohol (to avoid mineral or biofilm buildup) through the sensor, followed by compressed air.
3) Stop logging the flow rate and close the Flow Rate Platform software.
4) Remove the brown ferrule adapters with the tubing.

5) Close the ends of the flow sensor with the yellow plastic plugs.
6) Remove the sensor(s) from the FLOWBOARD.
7) Remove the USB type "A to B" cable from both the computer and the FLOWBOARD.

If this system is to be fully removed:

8) Store the Flow Rate Sensor Flow Units, FLOWBOARD, USB type "A to B" cable in the drawer labelled "Pressure Control System" in D509.

C.9.3. Pressure and Temperature Sensor (USB-6001 Module)

Due to the pressure and temperature sensor requiring the use of custom software and GUI, the cleanup procedure for this system is located in the "MATLAB GUI for Computing Apparent Viscosities" SOP.

C.9.4. Pressure Control System (MAESFLO Controller)

1) Set all the pressures to zero.
2) Close the software.
3) Press on the "Pause" button on the front of the MFCS-EZ controller.
4) Close the valve of the in-house compressed air line.
5) Turn off the MFCS-EZ using the power button at its back.
6) Remove the 4 mm O.D. flexible tubing from the Fluiwell and MFCS-EZ controller.
7) Disconnect the sample vials and screw the caps on the vials (if being kept for more experiments)
8) Remove the 1/16" O.D. rigid tubing from the Fluiwell
    a. Loosen the brown removable tube ferrules at the top.
    b. Gently pull the rigid tubing through the adapter until it fully comes out.
9) Clean the Fluiwell with the appropriate method according to the sample fluid.
    a. For blood, use a 10% bleach solution to wipe down all of the surfaces of the Fluiwell and the brown removable tube ferrules.

If this system is to be fully removed:

10) Remove the flexible tubing from the back of the MFCS-EZ and the manual pressure regulator.
11) Detach the tubing between both:
    a. The manual pressure regulator and the air drier
    b. The air drier and the in-house compressed air line
12) Store components in the drawer labelled “Pressure Control System” in D509

Appendix D MATLAB GUI for Computing Apparent Viscosities

D.1. Disclaimer

This document, and those to which it refers, have been created in reference to Dr. Fenech's laboratory and experimental procedures in mind. Its use for other laboratories, procedures or researchers should be modified or remade accordingly.

The program in its current state is not in the final form and requires code optimization and GUI testing with many users. Default file paths and test files require change before using with new users.

D.2. Purpose

This standard operating protocol outlines the procedure to

1) Use the MATLAB Simulink file to measure pressure and temperature
2) Use the MATLAB graphical user interface (GUI) code to calculate the apparent viscosity from the data files

D.3. Definitions

MATLAB – Matrix Laboratory (MATLAB) is a programming software capable of completing matrix operations and has a suite of products, such as signal processing and data acquisition.

GUI – A graphical user interface (GUI) for ease of use of programming code.

D.4. Materials Required

D.4.1. Computer

A desktop computer is located on the countertop in D509, but the software can be installed on a personal laptop as well.

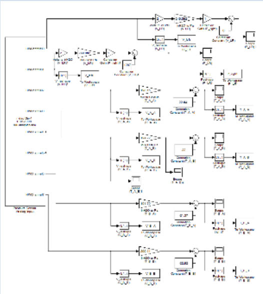
D.4.2. Required Software and Packages

The table below lists the data files necessary for the program to function. University of Ottawa has MATLAB available for use on select laboratory computers, which required network access. The university has a few licenses available of the following MATLAB packages.

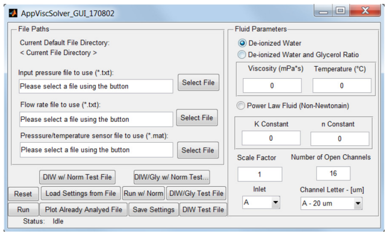
File	Purpose
MATLAB (32-BIT)	Main program used to record and analyze data files. 32-bit version is required to operate with the USB-6001 data acquisition module.
MATLAB Simulink	Interface used for collecting the pressure and temperature data. Appearance uses easy to use interface of flow blocks representing equipment and function transfers.
MATLAB Data Acquisition Pack	Allows MATLAB to communicate with the USB-6001 data acquisition module and handle the data signals.
MATLAB Signal Processing Pack	Allow MATLAB to process the data signals measured by the system.

D.4.3. Required Files for Pressure and Temperature Measurement

The following MATLAB files are required to use the pressure and temperature measurement Simulink program.

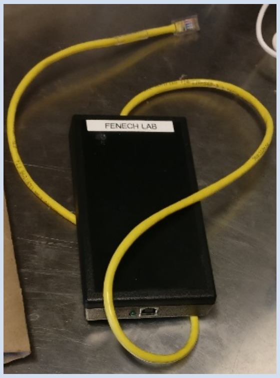
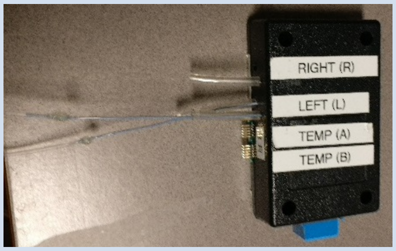
Required File	Location
 **Pressure_Temp_Measurement_160509.mdl**	Located on the network accessed storage (NAS) in \\NIKO\Matlab\Pressure_Temperature_System
NIDAQ1510f2.exe **NIDAQ Communication Software**	Located on the network accessed storage (NAS) in Z:\NIKO\OTHER, or located online as NIDAQ software for USB-6001

D.4.4. Required Files for Computing Apparent Viscosities

File	Purpose and Location
AppViscSolver_GUI_170802.m	Contains the main code for the GUI to calculate the apparent viscosity from the interface. Located on the network accessed storage (NAS) in \\NIKO\Matlab\Apparent_Viscosity_Solver
 AppViscSolver_GUI_170802 File Paths Current Default File Directory: < Current File Directory > Input pressure file to use (*.txt): Please select a file using the button Flow rate file to use (*.txt): Please select a file using the button Presssure/temperature sensor file to use (*.mat): Please select a file using the button Fluid Parameters ● De-ionized Water ○ De-ionized Water and Glycerol Ratio Viscosity (mPa*s): 0 Temperature (°C): 0 ○ Power Law Fluid (Non-Newtonain) K Constant: 0 n Constant: 0 Scale Factor: 1 Number of Open Channels: 16 Inlet: A Channel Letter - [um]: A - 20 um Status: Idle Select File Select File Select File DIW w/ Norm Test File DIW/Gly w/ Norm Test... Reset Load Settings from File Run w/ Norm DIW/Gly Test File Run Plot Already Analyed File Save Settings DIW Test File **AppViscSolver_GUI_170802.fig**	GUI interface for user to input the files and fluid information to analyze data. Located on the network accessed storage (NAS) in \\NIKO\Matlab\Apparent_Viscosity_Solver
Input Pressure Data File (Default_measurements_*.txt)**	Data file containing the input pressures from all the channels from the MFCS-EZ controller using MAEFLO software, the corresponding and the start time. User selects their own file.
Flow Rate Data File (FRPData_*.txt)**	Data file containing the flow rate from all the flow rate units attached to the Flowboard using Flow Rate Platform software, and the corresponding and the start time. User selects their own file.
MATLAB Simulink file (*.mat)	Contains the signal measurements from the differential pressure sensors, temperature sensors, the corresponding and the start time. User selects their own file.

D.4.5. Equipment

The table below lists the equipment necessary for use in the procedure, and their locations (if a set place is designated).

Equipment	Location
 **DAQ module with Attached Yellow Communication Cable**	In drawer labelled "Pressure Control System" in D509
 Universal Serial Bus (USB) Type "A to B' Cable	In drawer labelled "Pressure Control System" in D509
 **Pressure and Temperature Module**	In drawer labelled "Pressure Control System" in D509

Equipment	Location
 Thermocouple Pairs (Sheathed in Image)	In drawer labelled "Pressure Control System" in D509

D.5. First-Time Use

The first time that the system is used, it is recommended to install and setup all the software required as shown in Section D.4.2. Due to the unique build and sensitivity of the equipment, it is recommended to have a tutorial on the use of the equipment before initial use.

D.6. Measurement Procedure

D.6.1. Preparation

7) Check that the components of the system to be used are available for use in the laboratory calendar and with other lab users.
8) Refer to First-Time Use Section D.5 for the first-time use for software setup of the computer.
9) Read and understand the necessary training or safety procedures required for the materials or environment that the system will be used with.
10) Allocate necessary room required for the use of all the equipment components (including a laptop or desktop computer) for the experiment.
11) Ensure you have allotted enough time to do the whole procedure (e.g. going to washroom beforehand).

D.6.2. Data Acquisition Module

This module communicates with the pressure and temperature sensors to send to the computer.

1) Connect the yellow communication cable to the DAQ module (if not already connected), as shown in Figure 107.

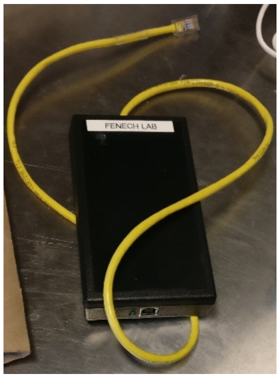


Figure 107. DAQ module with the communication cable already connected.

2) Connect one end of the USB type "A to B" cable, as shown in Figure 108, to the DAQ module and the other end to the computer.

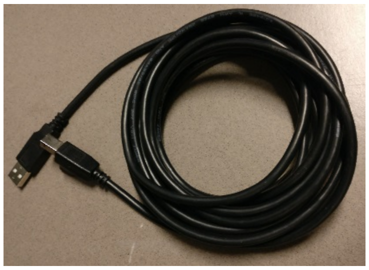


Figure 108. USB type "A to B" cable used to connect the computer to the DAQ module.

D.6.3. Pressure Sensors

The pressure sensors are of the differential pressure type, which allows the differential pressure to be measured, comparing between two ports. There are two sensors, each having a low and high-pressure tube, labelled as "LEFT (L)" and "RIGHT (R)". The low-pressure tube is indicated by a shorter length of tubing protruding from the side of the module.

Note: Excessive force should not be used with the pressure sensor tubes. These tubes can become disconnected from the pressure sensors if too much force is used.

1) Connect the other end of the yellow communication cable to the blue port of the pressure and temperature sensor module, which is depicted in Figure 109.

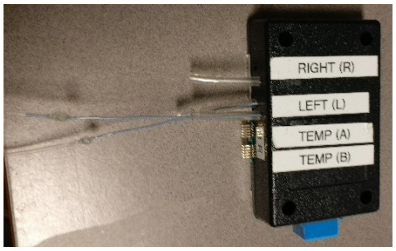


Figure 109. Pressure and temperature sensor module with pre-connected pressure measurement tubes.

2) Connect the low-pressure and high-pressure tubes appropriately to the fluidic ports to be measured.

D.6.4. Temperature Sensors

It is very important to connect the thermocouple pin connector to the DAW module correctly, since the program is calibrated in this fashion. The letter for a thermocouple pair requires to be matched up to the same letter on the DAQ. The "R" and "W" letters indicate the orientation that the red (R) and white (W) wires should be oriented, and should match the side that they are facing. I.e. "R" on the thermocouple pair should be closest to the "R" on the DAQ module.

The thermocouple pairs are named "A" and "B", as labelled and as shown in Figure 110. The pair of exposed thermocouple ends/leads are also individually labelled "A" and "B". When completing measurements with these pairs, the program labels are in the following format:

[Label of the thermocouple pin connector]_[Exposed thermocouple end]

Therefore, the thermocouple end "B" on thermocouple pin connector "A" would be called "A_B", moving outwards from the pressure and temperature sensor module.

Note: The thermocouples are very fragile, so be careful when handling the sensors. The wires should not be in tension, as this could cause the connections to break. Hold onto the black portion of the thermocouple pin connector when connecting or removing the thermocouples.

3) Connect each thermocouple to the matching letter port as on the DAQ module. The thermocouples, which are shown in Figure 110, do not need to be sheathed while being used.

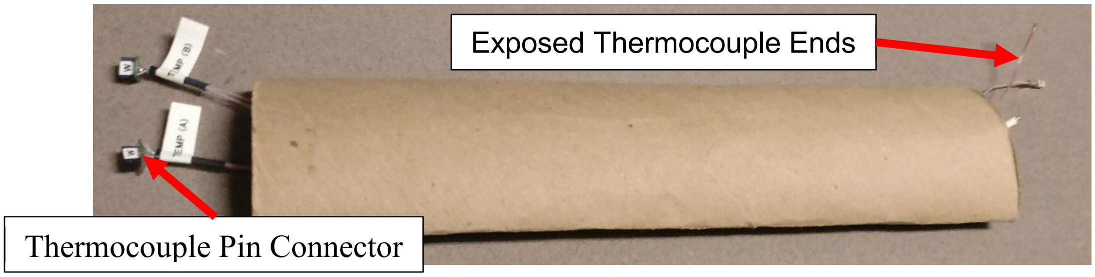


Figure 110. Two thermocouple pairs sheathed in a cardboard roll for protection.

Note: Ensure that all 10 pins are properly connected for each pair of thermocouples.

4) Connect the desired temperature sensor(s) to the desired ports or material.

D.6.5. Pressure and Temperature Measurement System

The pressure and temperature sensor data collection program, as shown in Figure 111, uses MATLAB's Simulink package to record the signal data.

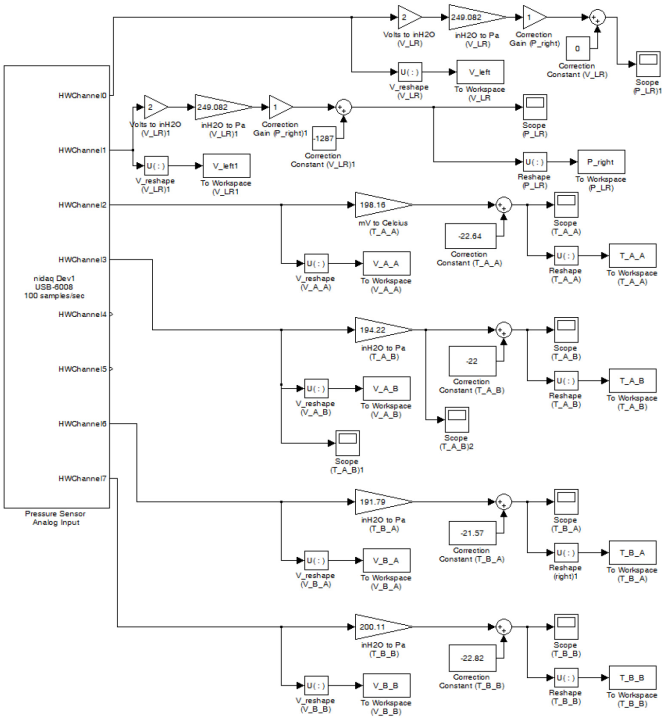
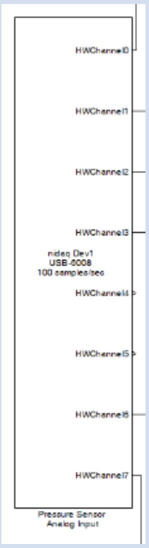
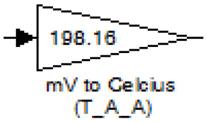
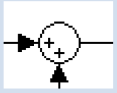
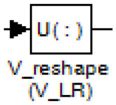
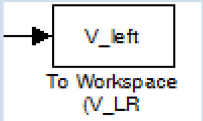
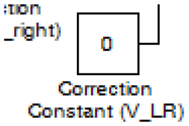
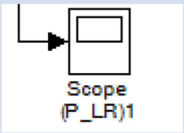


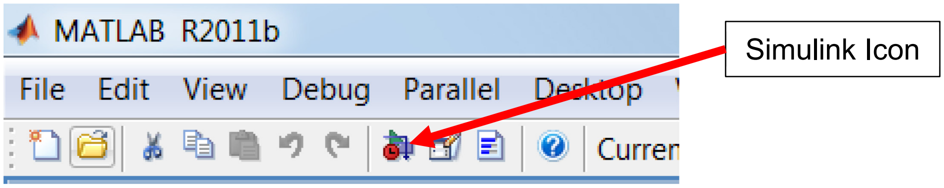
Figure 111. Pressure and temperature measurement Simulink program in flow diagram.

Refer to Table 27 for a description of each Simulink component functionality and use.

Table 27. List of the Simulink components and their descriptions, used in the pressure and temperature measurement Simulink flow program.

Simulink Component	Purpose and Information
 HWChannel0 HWChannel1 HWChannel2 HWChannel3 nidaq Dev1 USB-6008 100 samples/sec HWChannel4 HWChannel5 HWChannel6 HWChannel7 Pressure Sensor Analog Input **NIDAQ Block**	Represents the NIDAQ USB-6001 module with 8 possible output signals on its right side.
 **Multiplying Block**	Multiplies the input by the constant value within the triangle.
 **Summative Junction**	Sums the inputs (+/- as represented by the inputs) to an output.
 **Transfer Function Junction**	Reshapes the signal using the user-defined function.
 **Output to Workspace Block**	Outputs the input data to the MATLAB Workspace – note that this does not automatically save that data.

Simulink Component	Purpose and Information
 **Constant Block**	Outputs the constant value within the block.
 **Scope Block**	Gives the ability to monitor a signal in real-time.



1) Open Simulink within MATLAB.

   Note: It is normal for Simulink to take a few minutes to open.

2) Locate and open the Pressure_Temp_Measurement_160509.mdl file.
3) Double-click all scope blocks which are intended to be used to bring up a scope window, as shown in Figure 112.

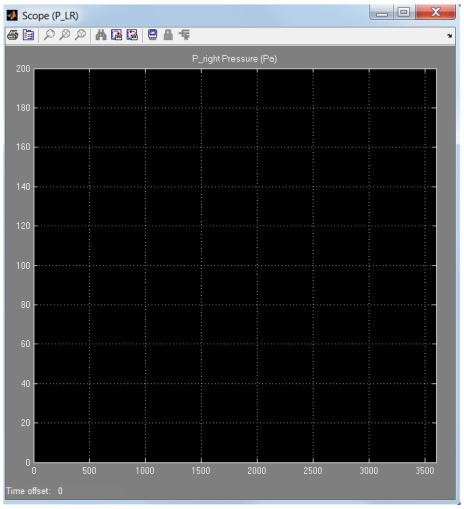
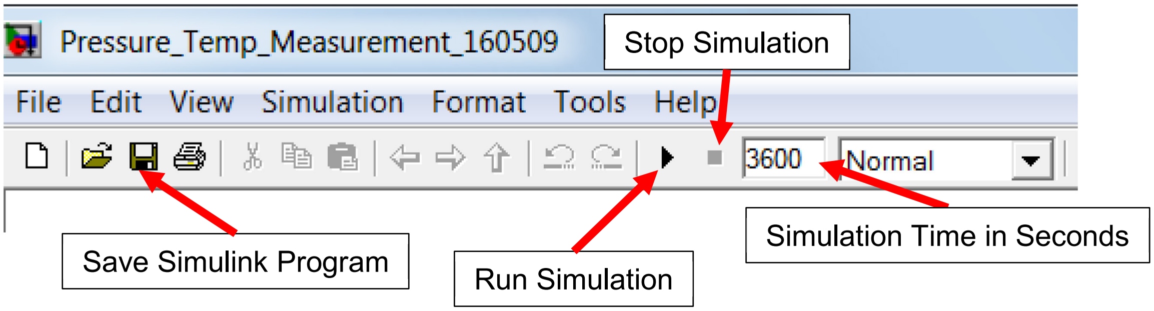


Figure 112. Opened scope window, ready to display real-time measurements.

4) Adjust the simulation time as needed for the required experiment, using the box as shown in Figure 113. The program can be stopped at any time during the simulation time (units are in seconds) without losing data from the Workspace.



Pressure_Temp_Measurement_160509
File Edit View Simulation Format Tools Help
3600
Normal
Stop Simulation
Save Simulink Program
Run Simulation
Simulation Time in Seconds

Figure 113. Key settings and buttons to operate and control the Simulink program.

5) Press the run button to start the Simulink simulation and start measuring data, as shown in Figure 113. The scope will start displaying data that the pressure and temperature sensor module is receiving.

6) If using the MATLAB GUI, as explained in GUI For Computing Apparent Viscosities Section D.8, for analyzing the data, leave a 10 second interval (no flow or input pressure). The GUI uses this time interval to determine the zero for the pressure.

7) Once the experiment is finished, the simulation can be stopped using the stop button, as shown in Figure 113.

   Note: The data has not been saved from the Workspace when pressing the stop button.

8) A prompt to save the Workspace data to a ".mat" file format will appear. Name and save the data file to a desired location. If it does not appear, manually save the data in the Workspace.

9) If a component of the Simulink program has been changed, use the save button (shown in Figure 113).

D.7. Clean-Up

In general, the clean-up procedure is the reverse process of the procedure for use in an experiment. Care must be taken to ensure equipment that touches the sample fluid is properly cleaned. Steps to clean up the systems are in the following subsections and gives details for steps requiring more detail in the clean-up.

1) Close the Simulink program and MATLAB.

D.7.1. Temperature Sensors

2) Remove the thermocouples by holding the thermocouple pin connector (not the wire) and pulling with gentle force away from the pressure and temperature sensor module.
3) Use the appropriate fluid cleaning techniques and use appropriate cleaning solutions for the end of the thermocouple sensors.

   E.g. For blood:

   a. Wipe the ends clean using PBS
   b. Wipe the ends with a minimal amount of 10% bleach solution
   c. Wipe the ends clean of the bleach using de-ionized water
   d. Check that the end is void of blood, such as using blood test strips to test for haemoglobin
4) Once each end is clean, it is recommended to use isopropyl alcohol (to avoid mineral or biofilm buildup) on just the end of the connections, followed by compressed air.
5) Place the thermocouples in the cardboard tube and back into the drawer labelled “Pressure Control System” in D509.

D.7.2. Pressure Sensors

6) Disconnect the pressure sensor tubes from the experiment’s fluidic ports. The pressure sensor tubes stay connected to the pressure sensor module.
7) Disconnect the communication cable from the pressure and temperature sensor module and the DAQ module.
8) Place the communication cable back into the drawer labelled “Pressure Control System” in D509.

9) Disconnect the USB type “A to B” cable from the pressure and temperature sensor module and the computer.
10) Place USB type “A to B” cable back into the drawer labelled “Pressure Control System” in D509.
11) Use the appropriate fluid cleaning techniques and use appropriate cleaning solutions for the end of the pressure sensor tubes.

    E.g. For blood:

    a. Wipe the ends clean using PBS
    b. Wipe the ends with a minimal amount of 10% bleach solution
    c. Wipe the ends clean of the bleach using de-ionized water
    d. Check that the end is void of blood, such as using blood test strips to test for haemoglobin
12) Once each end is clean, it is recommended to use isopropyl alcohol (to avoid mineral or biofilm buildup) on just the end of the connections.
13) Place the pressure and temperature sensor module back into the drawer labelled “Pressure Control System” in D509.

D.8. GUI For Computing Apparent Viscosities

The GUI uses the following data files to calculate the apparent viscosity using the programmed microfluidic design (*Refer to the “Pressure-Flow Control System SOP” for instructions on how to obtain the files in italics.)*:

a) Measured pressure and temperature data file, measured by the pressure and temperature sensor module (in .mat format)
b) Input pressure data file, measured by the MFCS-EZ (in .txt format)
c) Flow rate data file, measured by the MFCS-EZ (in .txt format)

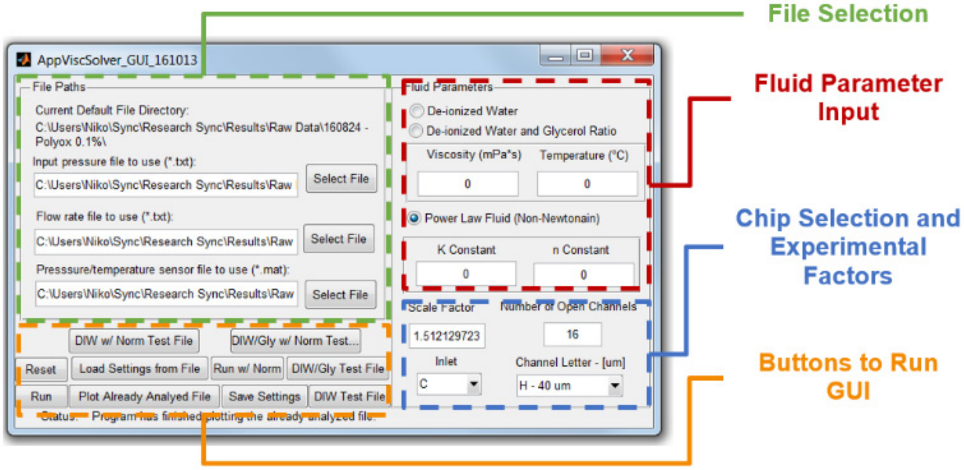
Currently, the options in the GUI are de-ionized water, a solution of de-ionized water and glycerol, and fluids which follow the power law. The procedure to use either of the Newtonian fluids or non-Newtonian power law fluids is similar, but has slightly different procedures. For ease of use, the full instructions for each of these conditions is located in the following subsections. Fluids which can be modelled using other laws are not implemented, but is possible by modifying the code.

D.8.1. Newtonian Fluids

1) Locate and open “AppViscSolver_GUI_170802.m” within MATLAB.

   Note: The file called “AppViscSolver_GUI_170802.fig” is also required to be in the same folder as “AppViscSolver_GUI_170802.m”.

2) Running the program will bring up an interface. The interface has been visually grouped in Figure 114 by the functionality of that section.


AppViscSolver_GUI_161013

File Paths

Current Default File Directory:
C:\Users\Niko\Sync\Research Sync\Results\Raw Data\160824 - Polyox 0.1%\

Input pressure file to use (*.txt): C:\Users\Niko\Sync\Research Sync\Results\Raw

Flow rate file to use (*.txt): C:\Users\Niko\Sync\Research Sync\Results\Raw

Presssure/temperature sensor file to use (*.mat): C:\Users\Niko\Sync\Research Sync\Results\Raw

Fluid Parameters

○ De-ionized Water

○ De-ionized Water and Glycerol Ratio

Viscosity (mPa*s): 0 Temperature (°C): 0

● Power Law Fluid (Non-Newtonain)

K Constant: 0 n Constant: 0

Scale Factor: 1.512129723 Number of Open Channels: 16

Inlet: C Channel Letter - [um]: H - 40 um

Status: Program has finished plotting the already analyzed file.

Figure 114. Graphical user interface (GUI) for computing the apparent viscosity and main components of the interface.

3) In “File Selection”, locate and select the input pressure, flow rate and pressure and temperature data files using the respective “Select File” buttons on the interface.

4) In “Fluid Parameter Input”, two options are available for the Newtonian fluids: (1) “De-ionized Water” and (2) a solution of “De-ionized Water and Glycerol Ratio”. Select the type of fluid using the radio button to the left of the name.

   a. For de-ionized water, the theoretical viscosity will be calculated using the temperature as measured by the temperature sensors.

   b. If using a solution of de-ionized water and glycerol, input the reference value of viscosity and the temperature it was measured at. These values will be used to display the theoretical apparent viscosity.

5) In “Chip Selection and Experimental Factors”, input the:
    a. Scale factor for the fluid, refer to “Pressure-Flow Control System SOP” for instructions on how to obtain the scale factor.
    b. Number of channels open when using the design, obtained by visually inspecting the channels during the experiment.
    c. The inlet used on the chip, which affects the length between the inlet port and first pressure tap on the design.
    d. The channel design designated letter, which affects which set of manually measured dimensions of the channels.
6) It is recommended to save the settings at this stage using the “Save Settings” button in “Buttons to Run GUI”. A prompt will appear with the option to select the name and location of the settings file.
7) Press the “Run” button in “Buttons to Run GUI”. A series of plots will appear on the screen, followed by a prompt to save the analyzed data. The file will have the default name of the pressure and temperature sensor file name, which can be changed. Select a name and location for the file before saving and viewing the plots.
8) Plots will appear that show the:
    a. Input pressure, measured pressure, flow rate, apparent viscosity and theoretical viscosity, as shown in Figure 115.

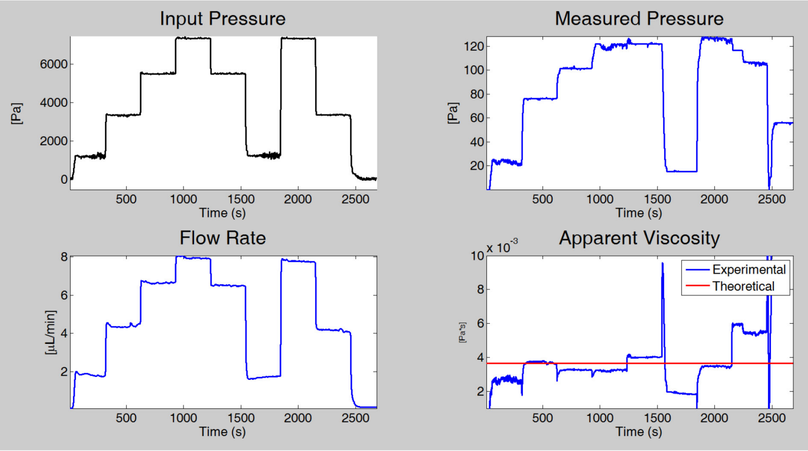


Figure 115. Example plot of signal processed and analyzed data with the input pressure (top left), measured pressure (top right), flow rate (bottom left), and experimental and theoretical apparent viscosities (bottom right).

b. Apparent viscosity with error curves indicating potential error with an error of ±3 μm in the measurement of the channel dimensions, in comparison to the theoretical apparent viscosity, as shown in Figure 116.

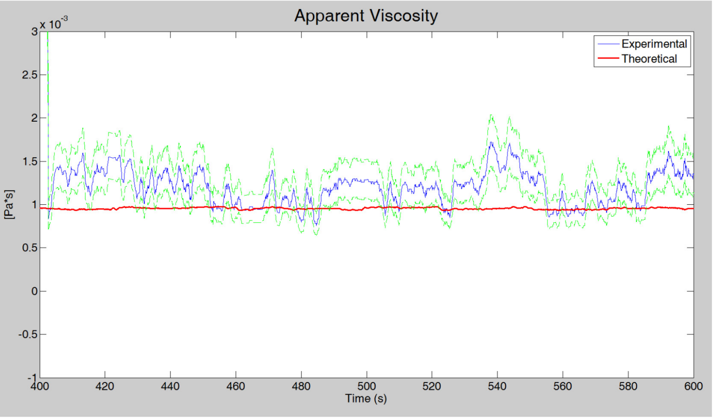


Figure 116. Example of a plot indicating the (blue line) experimental apparent viscosity, (red line) theoretical experimental viscosity, and (green lines) maximum and minimum error curves for an error of ±3 μm in the measurement of the channel dimensions.

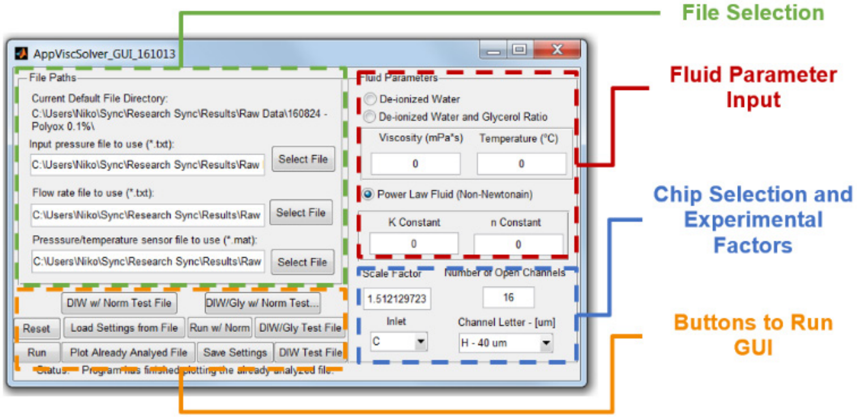
9) Once finished with the plots and GUI, close the plot windows and MATLAB.

D.8.2. Non-Newtonian Power Law Fluids

1) Locate and open "AppViscSolver_GUI_170802.m" within MATLAB.

   Note: The file called "AppViscSolver_GUI_170802.fig" is also required to be in the same folder as "AppViscSolver_GUI_170802.m".

2) Running the program will bring up an interface. The interface has been visually grouped in Figure 117 by the functionality of that section.



Current Default File Directory:
C:\Users\Niko\Sync\Research Sync\Results\Raw Data\160824 - Polyox 0.1%\

Input pressure file to use (*.txt): C:\Users\Niko\Sync\Research Sync\Results\Raw

Flow rate file to use (*.txt): C:\Users\Niko\Sync\Research Sync\Results\Raw

Presssure/temperature sensor file to use (*.mat): C:\Users\Niko\Sync\Research Sync\Results\Raw

○ De-ionized Water
○ De-ionized Water and Glycerol Ratio
Viscosity (mPa*s): 0 Temperature (°C): 0
● Power Law Fluid (Non-Newtonain)
K Constant: 0 n Constant: 0

Scale Factor: 1.512129723 Number of Open Channels: 16
Inlet: C Channel Letter - [um]: H - 40 um

Status: Program has finished plotting the already analyzed file.

Figure 117. Graphical user interface (GUI) for computing the apparent viscosity and main components of the interface.

3) In “File Selection”, locate and select the input pressure, flow rate and pressure and temperature data files using the respective “Select File” buttons on the interface.
4) In “Fluid Parameter Input”, select the radio button to the left of “Power Law Fluid (Non-Newtonian)”.
5) Input the reference/theoretical value of the K and n constant. These constants will be used to calculate the theoretical apparent viscosity.
6) In “Chip Selection and Experimental Factors”, input the:
    a. Scale factor for the fluid, refer to “Pressure-Flow Control System SOP” for instructions on how to obtain the scale factor.
    b. Number of channels open when using the design, obtained by visually inspecting the channels during the experiment.
    c. The inlet used on the chip, which affects the length between the inlet port and first pressure tap on the design.
    d. The channel design designated letter, which affects which set of manually measured dimensions of the channels.
7) It is recommended to save the settings at this stage using the “Save Settings” button in “Buttons to Run GUI”. A prompt will appear with the option to select the name and location of the settings file.
8) Press the “Run” button in “Buttons to Run GUI”. A series of plots will appear on the screen, followed by a prompt to save the analyzed data. The file will have the default name of the pressure and temperature sensor file name, which can be changed. Select a name and location for the file before saving and viewing the plots.
9) Plots will appear that show the:
    a. Input pressure, measured pressure, flow rate, apparent viscosity and theoretical viscosity, as shown in Figure 118.

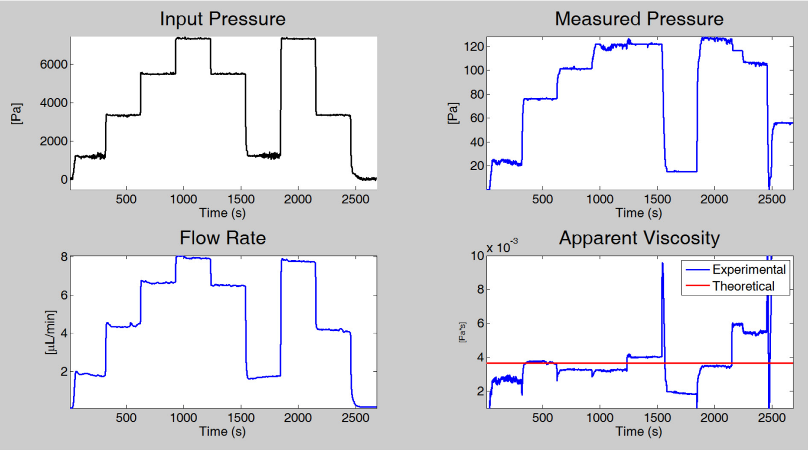


Figure 118. Example plot of signal processed and analyzed data with the input pressure (top left), measured pressure (top right), flow rate (bottom left), and experimental and theoretical apparent viscosities (bottom right).

b. Apparent viscosity with error curves indicating potential error with an error of ±3 µm in the measurement of the channel dimensions, in comparison to the theoretical apparent viscosity, as shown in Figure 119.

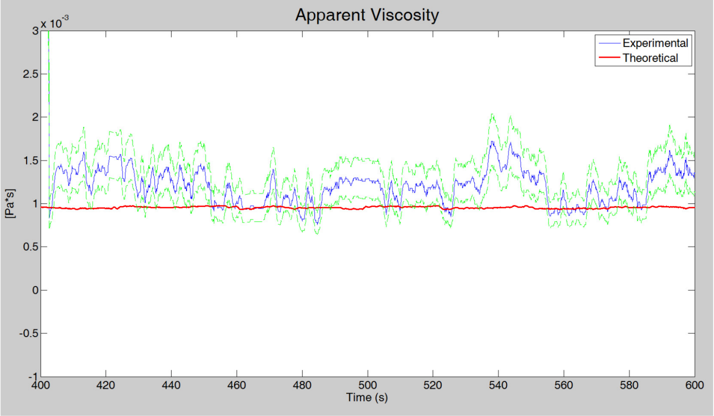


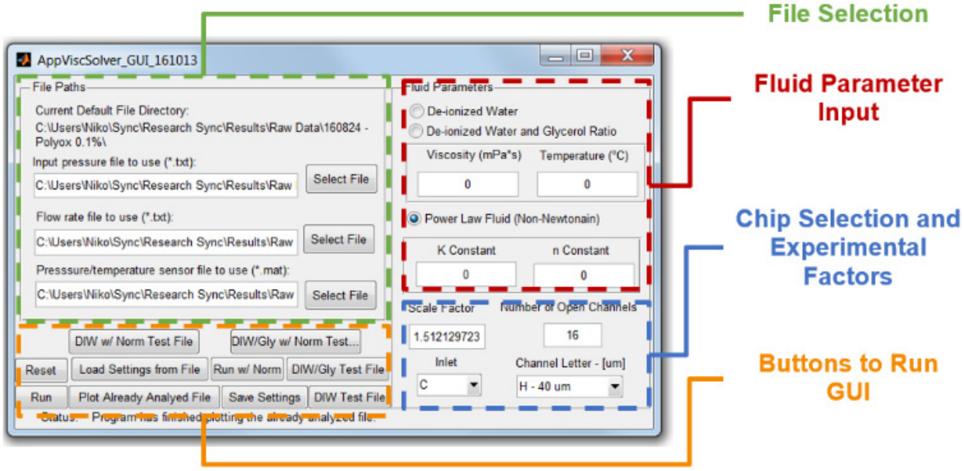
Figure 119. Example of a plot indicating the (blue line) experimental apparent viscosity, (red line) theoretical experimental viscosity, and (green lines) maximum and minimum error curves for an error of ±3 µm in the measurement of the channel dimensions.

10) Once finished analyzing the plots and GUI, close the plot windows and MATLAB.

D.8.3. Loading Saved Settings

The settings that were used to analyze an experiment can be saved as explained in Newtonian Fluids Section D.8.1or Non-Newtonian Power Law Fluids Section D.8.2. These settings can be loaded from the saved settings file to run the experiment again, or modify the settings that were used.

1) Locate and open "AppViscSolver_GUI_170802.m" within MATLAB.
   Note: The file called "AppViscSolver_GUI_170802.fig" is also required to be in the same folder as "AppViscSolver_GUI_170802.m".
2) Running the program will bring up an interface. The interface has been visually grouped in Figure 120by the functionality of that section.


AppViscSolver_GUI_161013

File Paths

Current Default File Directory:
C:\Users\Niko\Sync\Research Sync\Results\Raw Data\160824 - Polyox 0.1%\

Input pressure file to use (*.txt): C:\Users\Niko\Sync\Research Sync\Results\Raw

Flow rate file to use (*.txt): C:\Users\Niko\Sync\Research Sync\Results\Raw

Presssure/temperature sensor file to use (*.mat): C:\Users\Niko\Sync\Research Sync\Results\Raw

Fluid Parameters

○ De-ionized Water
○ De-ionized Water and Glycerol Ratio

Viscosity (mPa*s): 0
Temperature (°C): 0

● Power Law Fluid (Non-Newtonain)

K Constant: 0
n Constant: 0

Scale Factor: 1.512129723
Number of Open Channels: 16
Inlet: C
Channel Letter - [um]: H - 40 um

Status: Program has finished plotting the already analyzed file.

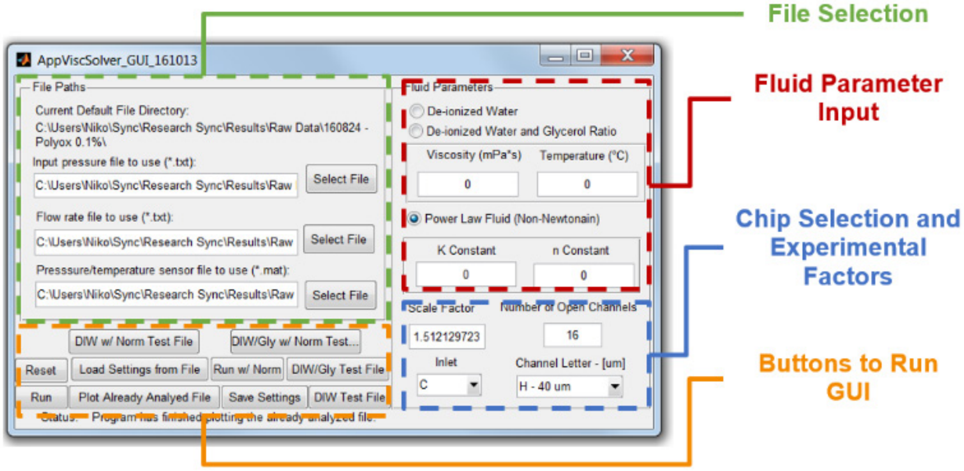
Figure 120. Graphical user interface (GUI) for computing the apparent viscosity and main components of the interface.

3) Press the "Load Settings from File" button. A prompt will appear to select the settings file.
4) Locate and select the saved settings file.
5) The saved settings will load, ready to be modified and/or analyzed with the "Run" button.

D.8.4. Loading Pre-Analyzed Data

If an experiment's files have already been analyzed, the data can be re-plotted without having to load the settings again. This option uses the data file of analyzed data as saved in Newtonian Fluids Section D.8.1 or Non-Newtonian Power Law Fluids Section D.8.2.

1) Locate and open "AppViscSolver_GUI_170802.m" within MATLAB.
   Note: The file called "AppViscSolver_GUI_170802.fig" is also required to be in the same folder as "AppViscSolver_GUI_170802.m".
2) Running the program will bring up an interface. The interface has been visually grouped in Figure 121by the functionality of that section.



AppViscSolver_GUI_161013

File Paths

Current Default File Directory: C:\Users\Niko\Sync\Research Sync\Results\Raw Data\160824 - Polyox 0.1%\

Input pressure file to use (*.txt): C:\Users\Niko\Sync\Research Sync\Results\Raw

Flow rate file to use (*.txt): C:\Users\Niko\Sync\Research Sync\Results\Raw

Presssure/temperature sensor file to use (*.mat): C:\Users\Niko\Sync\Research Sync\Results\Raw

Fluid Parameters

○ De-ionized Water

○ De-ionized Water and Glycerol Ratio

Viscosity (mPa*s): 0

Temperature (°C): 0

● Power Law Fluid (Non-Newtonain)

K Constant: 0

n Constant: 0

Scale Factor: 1.512129723

Number of Open Channels: 16

Inlet: C

Channel Letter - [um]: H - 40 um

Status: Program has finished plotting the already analyzed file.

Figure 121. Graphical user interface (GUI) for computing the apparent viscosity and main components of the interface.

3) Press the "Plot Already Analyzed File" button. A prompt will appear to select the data file with the analyzed data.
4) Plots will appear that show the:
    a. Input pressure, measured pressure, flow rate, apparent viscosity and theoretical viscosity, as shown in Figure 122.

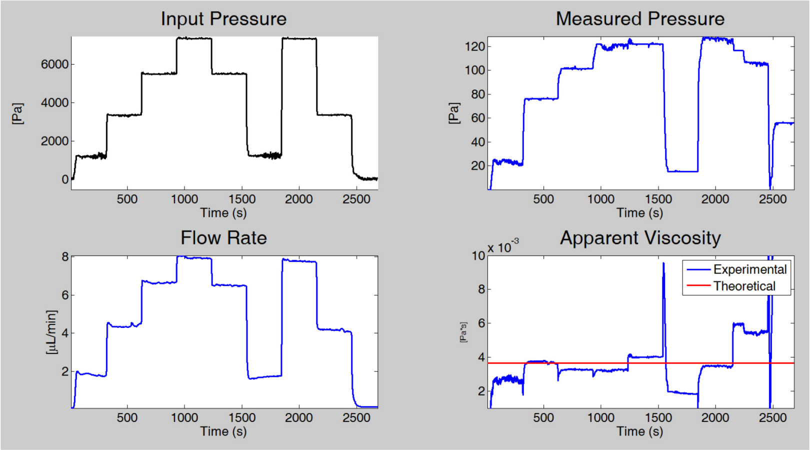


Figure 122. Example plot of signal processed and analyzed data with the input pressure (top left), measured pressure (top right), flow rate (bottom left), and experimental and theoretical apparent viscosities (bottom right).

    b. Apparent viscosity with error curves indicating potential error with an error of ±3 μm in the measurement of the channel dimensions, in comparison to the theoretical apparent viscosity, as shown in Figure 123.

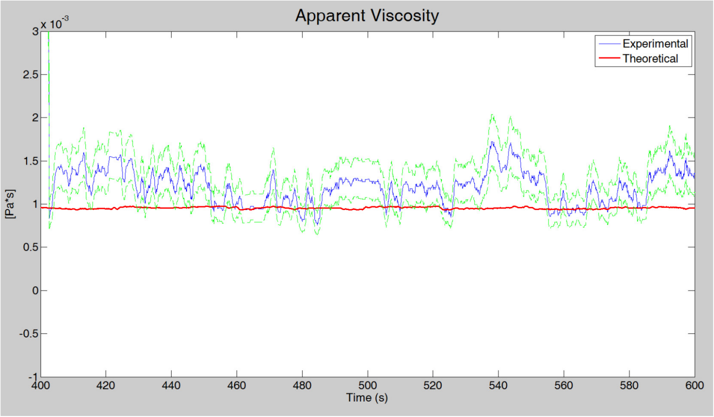


Figure 123. Example of a plot indicating the (blue line) experimental apparent viscosity, (red line) theoretical experimental viscosity, and (green lines) maximum and minimum error curves for an error of ±3 µm in the measurement of the channel dimensions.

5) Once finished analyzing the plots and GUI, close the plot windows and MATLAB.

D.8.5. Other GUI Buttons

The GUI contains other buttons in "Buttons to Run GUI" which are in progress:

a) "Reset"

   Resets the interface to a freshly-opened state.

b) "Run w/ Norm"

   Runs the file using a function to normalize the data.

c) "DIW Test File", and "DIW/Gly Test File"

   Testing the "Run" function with a test file (avoids having to re-enter the data while testing the functions).

d) "DIW w/ Norm Test File", and "DIW/Gly w/ Norm Test File"

   Testing the "Run w/ Norm" function with a test file (avoids having to re-enter the data while testing the functions).

D.9. MATLAB Code

D.9.1. Pressure and Temperature Measurement MATLAB Code

```
Model {
  Name            "Pressure_Temp_Measurement_160509"
  Version         7.8
  MdlSubVersion       0
  GraphicalInterface {
    NumRootInports      0
    NumRootOutports     0
    ParameterArgumentNames  ""
    ComputedModelVersion    "1.62"
    NumModelReferences      0
    NumTestPointedSignals   0
  }
  SavedCharacterEncoding  "windows-1252"
  SaveDefaultBlockParams  on
  ScopeRefreshTime    0.035000
  OverrideScopeRefreshTime on
  DisableAllScopes    off
  DataTypeOverride    "UseLocalSettings"
  DataTypeOverrideAppliesTo "AllNumericTypes"
  MinMaxOverflowLogging   "UseLocalSettings"
  MinMaxOverflowArchiveMode "Overwrite"
  FPTRunName          "Run 1"
  MaxMDLFileLineLength    120
  StartFcn        "start_time = clock"
  StopFcn         "uisave()"
  Created         "Mon Sep 30 15:43:40 2013"
  Creator         "Fenech_Grads"
  UpdateHistory       "UpdateHistoryNever"
  ModifiedByFormat    "%<Auto>"
  LastModifiedBy      "Niko"
  ModifiedDateFormat      "%<Auto>"
  LastModifiedDate    "Thu May 12 14:49:04 2016"
  RTWModifiedTimeStamp    384690407
  ModelVersionFormat      "1.%<AutoIncrement:62>"
  ConfigurationManager    "None"
  SampleTimeColors    off
  SampleTimeAnnotations   off
  LibraryLinkDisplay      "none"
  WideLines       off
  ShowLineDimensions      off
  ShowPortDataTypes   off
  ShowDesignRanges    off
  ShowLoopsOnError    on
  IgnoreBidirectionalLines off
  ShowStorageClass    off
  ShowTestPointIcons      on
  ShowSignalResolutionIcons on
  ShowViewerIcons     on
  SortedOrder         off
  ExecutionContextIcon    off
  ShowLinearizationAnnotations on
  BlockNameDataTip    off
  BlockParametersDataTip  off
  BlockDescriptionStringDataTip off
  ToolBar         on
  StatusBar       on
  BrowserShowLibraryLinks off
  BrowserLookUnderMasks   off
  SimulationMode      "normal"
  LinearizationMsg    "none"
  Profile         off
  ParamWorkspaceSource    "MATLABWorkspace"
  AccelSystemTargetFile   "accel.tlc"
  AccelTemplateMakefile   "accel_default_tmf"
  AccelMakeCommand    "make_rtw"
  TryForcingSFcnDF    off
  RecordCoverage      off
  CovPath         "/"
  CovSaveName         "covdata"
  CovMetricSettings   "dw"
  CovNameIncrementing     off
  CovHtmlReporting    on
  CovForceBlockReductionOff on
  covSaveCumulativeToWorkspaceVar on
  CovSaveSingleToWorkspaceVar on
  CovCumulativeVarName    "covCumulativeData"
  CovCumulativeReport     off
  CovReportOnPause    on
  CovModelRefEnable   "Off"
  CovExternalEMLEnable    off
  ExtModeBatchMode    off
  ExtModeEnableFloating   on
  ExtModeTrigType     "manual"
  ExtModeTrigMode     "normal"
  ExtModeTrigPort     "1"
  ExtModeTrigElement      "any"
  ExtModeTrigDuration     1000
  ExtModeTrigDurationFloating "auto"
  ExtModeTrigHoldOff      0
  ExtModeTrigDelay    0
  ExtModeTrigDirection    "rising"
  ExtModeTrigLevel    0
  ExtModeArchiveMode      "off"
  ExtModeAutoIncOneShot   off
  ExtModeIncDirWhenArm    off
  ExtModeAddSuffixToVar   off
  ExtModeWriteAllDataToWs off
  ExtModeArmWhenConnect   on
  ExtModeSkipDownloadWhenConnect off
  ExtModeLogAll       on
  ExtModeAutoUpdateStatusClock on
  BufferReuse         on
  ShowModelReferenceBlockVersion off
```

```
ShowModelReferenceBlockIO off
Array {
  Type            "Handle"
  Dimension           1
  Simulink.ConfigSet {
    $ObjectID           1
    Version             "1.11.1"
    Array {
  Type            "Handle"
  Dimension       8
  Simulink.SolverCC {
    $ObjectID       2
    Version         "1.11.1"
    StartTime       "0.0"
    StopTime        "3600"
    AbsTol          "auto"
    FixedStep       "auto"
    InitialStep         "auto"
    MaxNumMinSteps      "-1"
    MaxOrder        5
    ZcThreshold         "auto"
    ConsecutiveZCsStepRelTol "10*128*eps"
    MaxConsecutiveZCs   "1000"
    ExtrapolationOrder      4
    NumberNewtonIterations  1
    MaxStep         "auto"
    MinStep         "auto"
    MaxConsecutiveMinStep   "1"
    RelTol          "1e-3"
    SolverMode          "Auto"
    ConcurrentTasks     off
    Solver          "ode45"
    SolverName          "ode45"
    SolverJacobianMethodControl "auto"
    ShapePreserveControl    "DisableAll"
    ZeroCrossControl    "UseLocalSettings"
    ZeroCrossAlgorithm      "Nonadaptive"
    AlgebraicLoopSolver     "TrustRegion"
    SolverResetMethod   "Fast"
    PositivePriorityOrder   off
    AutoInsertRateTranBlk   off
    SampleTimeConstraint    "Unconstrained"
    InsertRTBMode       "Whenever possible"
  }
  Simulink.DataIOCC {
    $ObjectID       3
    Version         "1.11.1"
    Decimation          "1"
    ExternalInput       "[t, u]"
    FinalStateName      "xFinal"
    InitialState        "xInitial"
    LimitDataPoints     off
    MaxDataPoints       "1000"
    LoadExternalInput   off
    LoadInitialState    off
    SaveFinalState      off
    SaveCompleteFinalSimState off
    SaveFormat          "Array"
    SignalLoggingSaveFormat "ModelDataLogs"
    SaveOutput          on
    SaveState       off
    SignalLogging       on
    DSMLogging          on
    InspectSignalLogs   off
    SaveTime        on
    ReturnWorkspaceOutputs  off
    StateSaveName       "xout"
    TimeSaveName        "tout"
    OutputSaveName      "yout"
    SignalLoggingName   "logsout"
    DSMLoggingName      "dsmout"
    OutputOption        "RefineOutputTimes"
    OutputTimes         "[]"
    ReturnWorkspaceOutputsName "out"
    Refine          "1"
  }
  Simulink.OptimizationCC {
    $ObjectID       4
    Version         "1.11.1"
    Array {
      Type            "Cell"
      Dimension           8
      Cell            "BooleansAsBitfields"
      Cell            "PassReuseOutputArgsAs"
      Cell            "PassReuseOutputArgsThreshold"
      Cell            "ZeroExternalMemoryAtStartup"
      Cell            "ZeroInternalMemoryAtStartup"
      Cell            "OptimizeModelRefInitCode"
      Cell            "NoFixptDivByZeroProtection"
      Cell            "UseSpecifiedMinMax"
      PropName            "DisabledProps"
    }
    BlockReduction      on
    BooleanDataType     on
    ConditionallyExecuteInputs on
    InlineParams        off
    UseIntDivNetSlope   off
    UseFloatMulNetSlope     off
    UseSpecifiedMinMax      off
    InlineInvariantSignals  off
    OptimizeBlockIOStorage  on
    BufferReuse         on
    EnhancedBackFolding     off
    StrengthReduction   off
    ExpressionFolding   on
    BooleansAsBitfields     off
    BitfieldContainerType   "uint_T"
    EnableMemcpy        on
    MemcpyThreshold     64
    PassReuseOutputArgsAs   "Structure reference"
    ExpressionDepthLimit    2147483647
    FoldNonRolledExpr   on
    LocalBlockOutputs   on
    RollThreshold       5
    SystemCodeInlineAuto    off
    StateBitsets        off
    DataBitsets         off
    UseTempVars         off
    ZeroExternalMemoryAtStartup on
    ZeroInternalMemoryAtStartup on
    InitFltsAndDblsToZero   off
    NoFixptDivByZeroProtection off
    EfficientFloat2IntCast  off
    EfficientMapNaN2IntZero on
    OptimizeModelRefInitCode off
    LifeSpan        "inf"
    MaxStackSize        "Inherit from target"
    BufferReusableBoundary  on
    SimCompilerOptimization "Off"
    AccelVerboseBuild   off
    AccelParallelForEachSubsystem on
  }
  Simulink.DebuggingCC {
    $ObjectID       5
    Version         "1.11.1"
    RTPrefix        "error"
    ConsistencyChecking     "none"
    ArrayBoundsChecking     "none"
    SignalInfNanChecking    "none"
    SignalRangeChecking     "none"
    ReadBeforeWriteMsg      "UseLocalSettings"
    WriteAfterWriteMsg      "UseLocalSettings"
    WriteAfterReadMsg   "UseLocalSettings"
    AlgebraicLoopMsg    "warning"
    ArtificialAlgebraicLoopMsg "warning"
    SaveWithDisabledLinksMsg "warning"
    SaveWithParameterizedLinksMsg "warning"
    CheckSSInitialOutputMsg on
    UnderspecifiedInitializationDetection "Classic"
    MergeDetectMultiDrivingBlocksExec "none"
    CheckExecutionContextPreStartOutputMsg off
    CheckExecutionContextRuntimeOutputMsg off
    SignalResolutionControl "UseLocalSettings"
    BlockPriorityViolationMsg "warning"
    MinStepSizeMsg      "warning"
    TimeAdjustmentMsg   "none"
    MaxConsecutiveZCsMsg    "error"
    MaskedZcDiagnostic      "warning"
    IgnoredZcDiagnostic     "warning"
    SolverPrmCheckMsg   "warning"
    InheritedTsInSrcMsg     "warning"
    DiscreteInheritContinuousMsg "warning"
    MultiTaskDSMMsg     "error"
    MultiTaskCondExecSysMsg "error"
    MultiTaskRateTransMsg   "error"
    SingleTaskRateTransMsg  "none"
    TasksWithSamePriorityMsg "warning"
    SigSpecEnsureSampleTimeMsg "warning"
    CheckMatrixSingularityMsg "none"
    IntegerOverflowMsg      "warning"
    Int32ToFloatConvMsg     "warning"
    ParameterDowncastMsg    "error"
    ParameterOverflowMsg    "error"
    ParameterUnderflowMsg   "none"
    ParameterPrecisionLossMsg "warning"
    ParameterTunabilityLossMsg "warning"
    FixptConstUnderflowMsg  "none"
    FixptConstOverflowMsg   "none"
    FixptConstPrecisionLossMsg "none"
    UnderSpecifiedDataTypeMsg "none"
    UnnecessaryDatatypeConvMsg "none"
    VectorMatrixConversionMsg "none"
    InvalidFcnCallConnMsg   "error"
    FcnCallInpInsideContextMsg "Use local settings"
    SignalLabelMismatchMsg  "none"
    UnconnectedInputMsg     "warning"
    UnconnectedOutputMsg    "warning"
    UnconnectedLineMsg      "warning"
    SFcnCompatibilityMsg    "none"
    FrameProcessingCompatibilityMsg "warning"
    UniqueDataStoreMsg      "none"
    BusObjectLabelMismatch  "warning"
    RootOutportRequireBusObject "warning"
    AssertControl       "UseLocalSettings"
    EnableOverflowDetection off
    ModelReferenceIOMsg     "none"
    ModelReferenceMultiInstanceNormalModeStructChecksumCheck "error"
    ModelReferenceVersionMismatchMessage "none"
    ModelReferenceIOMismatchMessage "none"
    ModelReferenceCSMismatchMessage "none"
    UnknownTsInhSupMsg      "warning"
    ModelReferenceDataLoggingMessage "warning"
    ModelReferenceSymbolNameMessage "warning"
    ModelReferenceExtraNoncontSigs "error"
    StateNameClashWarn      "warning"
    SimStateInterfaceChecksumMismatchMsg "warning"
    SimStateOlderReleaseMsg "error"
    InitInArrayFormatMsg    "warning"
    StrictBusMsg        "ErrorLevel1"
    BusNameAdapt        "WarnAndRepair"
    NonBusSignalsTreatedAsBus "none"
    LoggingUnavailableSignals "error"
    BlockIODiagnostic   "none"
    SFUnusedDataAndEventsDiag "warning"
    SFUnexpectedBacktrackingDiag "warning"
    SFInvalidInputDataAccessInChartInitDiag "warning"
    SFNoUnconditionalDefaultTransitionDiag "warning"
    SFTransitionOutsideNaturalParentDiag "warning"
    SFUnconditionalTransitionShadowingDiag "warning"
  }
  Simulink.HardwareCC {
    $ObjectID       6
    Version         "1.11.1"
    ProdBitPerChar      8
    ProdBitPerShort     16
    ProdBitPerInt       32
    ProdBitPerLong      32
    ProdBitPerFloat     32
    ProdBitPerDouble    64
    ProdBitPerPointer   32
    ProdLargestAtomicInteger "Char"
    ProdLargestAtomicFloat  "None"
    ProdIntDivRoundTo   "Undefined"
    ProdEndianess       "Unspecified"
    ProdWordSize        32
    ProdShiftRightIntArith  on
    ProdHWDeviceType    "32-bit Generic"
    TargetBitPerChar    8
    TargetBitPerShort   16
    TargetBitPerInt     32
    TargetBitPerLong    32
    TargetBitPerFloat   32
    TargetBitPerDouble      64
    TargetBitPerPointer     32
    TargetLargestAtomicInteger "Char"
    TargetLargestAtomicFloat "None"
    TargetShiftRightIntArith on
    TargetIntDivRoundTo     "Undefined"
    TargetEndianess     "Unspecified"
    TargetWordSize      32
    TargetTypeEmulationWarnSuppressLevel 0
    TargetPreprocMaxBitsSint 32
    TargetPreprocMaxBitsUint 32
    TargetHWDeviceType      "Specified"
    TargetUnknown       off
    ProdEqTarget        on
  }
  Simulink.ModelReferenceCC {
    $ObjectID       7
    Version         "1.11.1"
    UpdateModelReferenceTargets "IfOutOfDateOrStructuralChange"
    CheckModelReferenceTargetMessage "error"
    EnableParallelModelReferenceBuilds off
    ParallelModelReferenceErrorOnInvalidPool on
    ParallelModelReferenceMATLABWorkerInit "None"
    ModelReferenceNumInstancesAllowed "Multi"
    PropagateVarSize    "Infer from blocks in model"
    ModelReferencePassRootInputsByReference on
    ModelReferenceMinAlgLoopOccurrences off
    PropagateSignalLabelsOutOfModel off
    SupportModelReferenceSimTargetCustomCode off
  }
  Simulink.SFSimCC {
    $ObjectID       8
    Version         "1.11.1"
    SFSimEnableDebug    on
    SFSimOverflowDetection  on
    SFSimEcho       on
    SimBlas         on
    SimCtrlC        on
    SimExtrinsic        on
    SimIntegrity        on
    SimUseLocalCustomCode   off
    SimParseCustomCode      on
    SimBuildMode        "sf_incremental_build"
  }
  Simulink.RTWCC {
    $BackupClass        "Simulink.RTWCC"
    $ObjectID       9
    Version         "1.11.1"
    Array {
      Type            "Cell"
      Dimension           7
      Cell            "IncludeHyperlinkInReport"
      Cell            "GenerateTraceInfo"
      Cell            "GenerateTraceReport"
      Cell            "GenerateTraceReportSl"
      Cell            "GenerateTraceReportSf"
      Cell            "GenerateTraceReportEml"
      Cell            "GenerateSLWebview"
      PropName            "DisabledProps"
    }
    SystemTargetFile    "grt.tlc"
    GenCodeOnly         off
    MakeCommand         "make_rtw"
    GenerateMakefile    on
    TemplateMakefile    "grt_default_tmf"
    GenerateReport      off
    SaveLog         off
    RTWVerbose          on
    RetainRTWFile       off
    ProfileTLC          off
    TLCDebug        off
    TLCCoverage         off
    TLCAssert       off
    ProcessScriptMode   "Default"
    ConfigurationMode   "Optimized"
    ConfigAtBuild       off
    RTWUseLocalCustomCode   off
    RTWUseSimCustomCode     off
    IncludeHyperlinkInReport off
    LaunchReport        off
    TargetLang          "C"
    IncludeBusHierarchyInRTWFileBlockHierarchyMap off
    IncludeERTFirstTime     off
    GenerateTraceInfo   off
    GenerateTraceReport     off
    GenerateTraceReportSl   off
    GenerateTraceReportSf   off
    GenerateTraceReportEml  off
    GenerateCodeInfo    off
    GenerateSLWebview   off
    GenerateCodeMetricsReport off
    RTWCompilerOptimization "Off"
    CheckMdlBeforeBuild     "Off"
    CustomRebuildMode   "OnUpdate"
    Array {
      Type            "Handle"
      Dimension           2
      Simulink.CodeAppCC {
        $ObjectID           10
        Version             "1.11.1"
        Array {
      Type            "Cell"
      Dimension       21
      Cell            "IgnoreCustomStorageClasses"
      Cell            "IgnoreTestpoints"
      Cell            "InsertBlockDesc"
      Cell            "InsertPolySpaceComments"
      Cell            "SFDataObjDesc"
      Cell            "MATLABFcnDesc"
      Cell            "SimulinkDataObjDesc"
      Cell            "DefineNamingRule"
      Cell            "SignalNamingRule"
      Cell            "ParamNamingRule"
      Cell            "InlinedPrmAccess"
      Cell            "CustomSymbolStr"
      Cell            "CustomSymbolStrGlobalVar"
      Cell            "CustomSymbolStrType"
      Cell            "CustomSymbolStrField"
      Cell            "CustomSymbolStrFcn"
      Cell            "CustomSymbolStrFcnArg"
      Cell            "CustomSymbolStrBlkIO"
      Cell            "CustomSymbolStrTmpVar"
      Cell            "CustomSymbolStrMacro"
      Cell            "ReqsInCode"
      PropName        "DisabledProps"
        }
        ForceParamTrailComments off
        GenerateComments        on
        IgnoreCustomStorageClasses on
        IgnoreTestpoints        off
        IncHierarchyInIds       off
        MaxIdLength         31
        PreserveName        off
        PreserveNameWithParent  off
        ShowEliminatedStatement off
        IncAutoGenComments      off
        SimulinkDataObjDesc     off
        SFDataObjDesc       off
        MATLABFcnDesc       off
        IncDataTypeInIds        off
        MangleLength        1
        CustomSymbolStrGlobalVar "$R$N$M"
        CustomSymbolStrType     "$N$R$M"
        CustomSymbolStrField    "$N$M"
        CustomSymbolStrFcn      "$R$N$M$F"
        CustomSymbolStrFcnArg   "rt$I$N$M"
        CustomSymbolStrBlkIO    "rtb_$N$M"
        CustomSymbolStrTmpVar   "$N$M"
        CustomSymbolStrMacro    "$R$N$M"
        DefineNamingRule        "None"
        ParamNamingRule         "None"
        SignalNamingRule        "None"
        InsertBlockDesc         off
        InsertPolySpaceComments off
        SimulinkBlockComments   on
        MATLABSourceComments    off
        EnableCustomComments    off
        InlinedPrmAccess        "Literals"
        ReqsInCode          off
        UseSimReservedNames     off
      }
      Simulink.GRTTargetCC {
        $BackupClass        "Simulink.TargetCC"
        $ObjectID           11
        Version             "1.11.1"
        Array {
      Type            "Cell"
      Dimension       16
      Cell            "GeneratePreprocessorConditionals"
      Cell            "IncludeMdlTerminateFcn"
      Cell            "CombineOutputUpdateFcns"
      Cell            "SuppressErrorStatus"
      Cell            "ERTCustomFileBanners"
      Cell            "GenerateSampleERTMain"
      Cell            "GenerateTestInterfaces"
      Cell            "ModelStepFunctionPrototypeControlCompliant"
      Cell            "CPPClassGenCompliant"
      Cell            "MultiInstanceERTCode"
      Cell            "PurelyIntegerCode"
      Cell            "SupportComplex"
      Cell            "SupportAbsoluteTime"
      Cell            "SupportContinuousTime"
      Cell            "SupportNonInlinedSFcns"
      Cell            "PortableWordSizes"
      PropName        "DisabledProps"
        }
        TargetFcnLib        "ansi_tfl_table_tmw.mat"
        TargetLibSuffix         ""
        TargetPreCompLibLocation ""
        TargetFunctionLibrary   "ANSI_C"
        UtilityFuncGeneration   "Auto"
        ERTMultiwordTypeDef     "System defined"
        CodeExecutionProfiling  off
        ERTMultiwordLength      256
        MultiwordLength         2048
        GenerateFullHeader      on
        GenerateSampleERTMain   off
        GenerateTestInterfaces  off
        IsPILTarget         off
        ModelReferenceCompliant on
        ParMdlRefBuildCompliant on
        CompOptLevelCompliant   on
        ConcurrentExecutionCompliant on
        IncludeMdlTerminateFcn  on
        GeneratePreprocessorConditionals "Disable all"
        CombineOutputUpdateFcns off
        CombineSignalStateStructs off
        SuppressErrorStatus     off
        ERTFirstTimeCompliant   off
        IncludeFileDelimiter    "Auto"
        ERTCustomFileBanners    off
        SupportAbsoluteTime     on
        LogVarNameModifier      "rt_"
        MatFileLogging          on
```

```
        MultiInstanceERTCode    off
        SupportNonFinite        on
        SupportComplex          on
        PurelyIntegerCode       off
        SupportContinuousTime   on
        SupportNonInlinedSFcns  on
        SupportVariableSizeSignals off
        EnableShiftOperators    on
        ParenthesesLevel        "Nominal"
        PortableWordSizes       off
        ModelStepFunctionPrototypeControlCompliant off
        CPPClassGenCompliant    off
        AutosarCompliant        off
        UseMalloc           off
        ExtMode             off
        ExtModeStaticAlloc      off
        ExtModeTesting          off
        ExtModeStaticAllocSize  1000000
        ExtModeTransport        0
        ExtModeMexFile          "ext_comm"
        ExtModeIntrfLevel       "Level1"
        RTWCAPISignals          off
        RTWCAPIParams       off
        RTWCAPIStates       off
        RTWCAPIRootIO       off
        GenerateASAP2       off
      }
      PropName            "Components"
    }
  }
  PropName        "Components"
    }
    Name            "Configuration"
    CurrentDlgPage          "Data Import//Export"
    ConfigPrmDlgPosition     [ 64, 39, 960, 707 ]
  }
  PropName            "ConfigurationSets"
}
Simulink.ConfigSet {
  $PropName           "ActiveConfigurationSet"
  $ObjectID           1
}
BlockDefaults {
  ForegroundColor     "black"
  BackgroundColor     "white"
  DropShadow          off
  NamePlacement       "normal"
  FontName            "Helvetica"
  FontSize            10
  FontWeight          "normal"
  FontAngle           "normal"
  ShowName            on
  BlockRotation       0
  BlockMirror         off
}
AnnotationDefaults {
  HorizontalAlignment     "center"
  VerticalAlignment       "middle"
  ForegroundColor     "black"
  BackgroundColor     "white"
  DropShadow          off
  FontName            "Helvetica"
  FontSize            10
  FontWeight          "normal"
  FontAngle           "normal"
  UseDisplayTextAsClickCallback off
}
LineDefaults {
  FontName            "Helvetica"
  FontSize            9
  FontWeight          "normal"
  FontAngle           "normal"
}
BlockParameterDefaults {
  Block {
    BlockType           Constant
    Value           "1"
    VectorParams1D          on
    SamplingMode        "Sample based"
    OutMin              "[]"
    OutMax              "[]"
    OutDataTypeStr          "Inherit: Inherit from 'Constant value'"
    LockScale           off
    SampleTime          "inf"
    FramePeriod         "inf"
    PreserveConstantTs      off
  }
  Block {
    BlockType           Gain
    Gain            "1"
    Multiplication          "Element-wise(K.*u)"
    ParamMin            "[]"
    ParamMax            "[]"
    ParamDataTypeStr        "Inherit: Same as input"
    OutMin              "[]"
    OutMax              "[]"
    OutDataTypeStr          "Inherit: Same as input"
    LockScale           off
    RndMeth             "Floor"
    SaturateOnIntegerOverflow on
    SampleTime          "-1"
  }
  Block {
    BlockType           Reshape
    OutputDimensionality    "1-D array"
    OutputDimensions        "[1,1]"
  }
  Block {
    BlockType           Scope
    ModelBased          off
    TickLabels          "OneTimeTick"
    ZoomMode            "on"
    Grid            "on"
    TimeRange           "auto"
    YMin            "-5"
    YMax            "5"
    SaveToWorkspace         off
    SaveName            "ScopeData"
    LimitDataPoints         on
    MaxDataPoints       "5000"
    Decimation          "1"
    SampleInput         off
    SampleTime          "-1"
  }
  Block {
    BlockType           Sum
    IconShape           "rectangular"
    Inputs              "++"
    CollapseMode        "All dimensions"
    CollapseDim         "1"
    InputSameDT         on
    AccumDataTypeStr        "Inherit: Inherit via internal rule"
    OutMin              "[]"
    OutMax              "[]"
    OutDataTypeStr          "Inherit: Same as first input"
    LockScale           off
    RndMeth             "Floor"
    SaturateOnIntegerOverflow on
    SampleTime          "-1"
  }
  Block {
    BlockType           ToWorkspace
    VariableName        "simulink_output"
    MaxDataPoints       "1000"
    Decimation          "1"
    SampleTime          "0"
    FixptAsFi           off
    NumInputs           "1"
  }
}
System {
  Name            "Pressure_Temp_Measurement_160509"
  Location            [2, 82, 1331, 725]
  Open            on
  ModelBrowserVisibility  off
  ModelBrowserWidth       200
  ScreenColor         "white"
  PaperOrientation        "landscape"
  PaperPositionMode       "auto"
  PaperType           "usletter"
  PaperUnits          "inches"
  TiledPaperMargins       [0.500000, 0.500000, 0.500000, 0.500000]
  TiledPageScale      1
  ShowPageBoundaries      off
  ZoomFactor          "100"
  ReportName          "simulink-default.rpt"
  SIDHighWatermark        "154"
  Block {
    BlockType           Constant
    Name            "Correction\nConstant (T_A_A)"
    SID             "79"
    Position            [625, 334, 690, 366]
    Value           "-22.64"
  }
  Block {
    BlockType           Constant
    Name            "Correction\nConstant (T_A_B)"
    SID             "80"
    Position            [630, 491, 690, 529]
    Value           "-22"
  }
  Block {
    BlockType           Constant
    Name            "Correction\nConstant (T_B_A)"
    SID             "49"
    Position            [630, 715, 690, 745]
    Value           "-21.57"
  }
  Block {
    BlockType           Constant
    Name            "Correction\nConstant (T_B_B)"
    SID             "69"
    Position            [630, 873, 690, 907]
    Value           "-22.82"
  }
  Block {
    BlockType           Constant
    Name            "Correction\nConstant (V_LR)"
    SID             "10"
    Position            [780, 95, 810, 125]
    Value           "0"
  }
  Block {
    BlockType           Constant
    Name            "Correction\nConstant (V_LR)1"
    SID             "146"
    Position            [455, 210, 485, 240]
    Value           "-1287"
  }
  Block {
    BlockType           Gain
    Name            "Correction\nGain (P_right)"
    SID             "22"
    Position            [725, 55, 755, 85]
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Gain
    Name            "Correction\nGain (P_right)1"
    SID             "147"
    Position            [400, 170, 430, 200]
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
```

```
  BlockType           Reference
  Name            "Pressure Sensor\nAnalog Input"
  SID             "3"
  Ports           [0, 8]
  Position            [60, 132, 195, 698]
  LibraryVersion          "1.14"
  SourceBlock         "daqlib/Analog Input"
  SourceType          "Analog Input"
  AcqMode             "0"
  DeviceMenu          "nidaq Dev1 (USB-6008)"
  Device              "nidaq Dev1 (USB-6008)"
  ObjConstructor          "analoginput('nidaq','Dev1')"
  SampleRate          "100"
  ActualRate          "100"
  SamplesPerFrame         "2"
  InputType           "SingleEnded"
  NChannelsSelected       "8"
  Channels            "1$0$$3$#1$1$$3$#1$2$$8$#1$3$$8$#1$4$$3$#1$5$$3$#1$6$$8$#1$7$$8$#"
  NPorts              "1 per hardware channel"
  SignalType          "Sample-based"
  DataType            "double"
}
Block {
  BlockType           Reshape
  Name            "Reshape\n(P_LR)"
  SID             "13"
  Ports           [1, 1]
  Position            [725, 243, 755, 267]
}
Block {
  BlockType           Reshape
  Name            "Reshape\n(T_A_A)"
  SID             "83"
  Ports           [1, 1]
  Position            [765, 363, 795, 387]
}
Block {
  BlockType           Reshape
  Name            "Reshape\n(T_A_B)"
  SID             "84"
  Ports           [1, 1]
  Position            [765, 523, 795, 547]
}
Block {
  BlockType           Reshape
  Name            "Reshape\n(T_B_B)"
  SID             "71"
  Ports           [1, 1]
  Position            [765, 903, 795, 927]
}
Block {
  BlockType           Reshape
  Name            "Reshape\n(right)1"
  SID             "51"
  Ports           [1, 1]
  Position            [765, 743, 795, 767]
}
Block {
  BlockType           Scope
  Name            "Scope\n(P_LR)"
  SID             "2"
  Ports           [1]
  Position            [725, 169, 755, 201]
  Floating            off
  Location            [692, 58, 1359, 731]
  Open            on
  NumInputPorts       "1"
  List {
ListType        AxesTitles
axes1           "P_right Pressure (Pa)"
  }
  List {
ListType        ScopeGraphics
FigureColor     "[0.5 0.5 0.5]"
AxesColor       "[0 0 0]"
AxesTickColor       "[1 1 1]"
LineColors      "[1 1 0;1 0 1;0 1 1;1 0 0;0 1 0;0 0 1]"
LineStyles      "-|-|-|-|-|-"
LineWidths      "[0.5 0.5 0.5 0.5 0.5 0.5]"
MarkerStyles        "none|none|none|none|none|none"
  }
  TimeRange           "3600"
  YMin            "0"
  YMax            "200"
  DataFormat          "StructureWithTime"
  MaxDataPoints       "18000"
  SampleTime          "0"
}
Block {
  BlockType           Scope
  Name            "Scope\n(P_LR)1"
  SID             "153"
  Ports           [1]
  Position            [870, 114, 900, 146]
  Floating            off
  Location            [788, 248, 1338, 703]
  Open            off
  NumInputPorts       "1"
  List {
ListType        AxesTitles
axes1           "P_right Pressure (Pa)"
  }
  List {
ListType        ScopeGraphics
FigureColor     "[0.5 0.5 0.5]"
AxesColor       "[0 0 0]"
AxesTickColor       "[1 1 1]"
LineColors      "[1 1 0;1 0 1;0 1 1;1 0 0;0 1 0;0 0 1]"
LineStyles      "-|-|-|-|-|-"
LineWidths      "[0.5 0.5 0.5 0.5 0.5 0.5]"
MarkerStyles        "none|none|none|none|none|none"
  }
  TimeRange           "3600            "
  YMin            "61"
  YMax            "72"
  SaveName            "ScopeData2"
  DataFormat          "StructureWithTime"
  MaxDataPoints       "18000"
  SampleTime          "0"
}
Block {
  BlockType           Scope
  Name            "Scope\n(T_A_A)"
  SID             "85"
  Ports           [1]
  Position            [765, 294, 795, 326]
  Floating            off
  Location            [63, 72, 538, 459]
  Open            off
  NumInputPorts       "1"
  ZoomMode            "yonly"
  List {
ListType        AxesTitles
axes1           "%<SignalLabel>"
  }
  List {
ListType        ScopeGraphics
FigureColor     "[0.5 0.5 0.5]"
AxesColor       "[0 0 0]"
AxesTickColor       "[1 1 1]"
LineColors      "[1 1 0;1 0 1;0 1 1;1 0 0;0 1 0;0 0 1]"
LineStyles      "-|-|-|-|-|-"
LineWidths      "[0.5 0.5 0.5 0.5 0.5 0.5]"
MarkerStyles        "none|none|none|none|none|none"
  }
  TimeRange           "500"
  YMin            "42"
  YMax            "48.5"
  SaveName            "ScopeData4"
  DataFormat          "StructureWithTime"
  MaxDataPoints       "18000"
  SampleTime          "0"
}
Block {
  BlockType           Scope
  Name            "Scope\n(T_A_B)"
  SID             "86"
  Ports           [1]
  Position            [765, 454, 795, 486]
  Floating            off
  Location            [1, 49, 1367, 737]
  Open            off
  NumInputPorts       "1"
  List {
ListType        AxesTitles
axes1           "%<SignalLabel>"
  }
  List {
ListType        ScopeGraphics
FigureColor     "[0.5 0.5 0.5]"
AxesColor       "[0 0 0]"
AxesTickColor       "[1 1 1]"
LineColors      "[1 1 0;1 0 1;0 1 1;1 0 0;0 1 0;0 0 1]"
LineStyles      "-|-|-|-|-|-"
LineWidths      "[0.5 0.5 0.5 0.5 0.5 0.5]"
MarkerStyles        "none|none|none|none|none|none"
  }
  TimeRange           "500"
  YMin            "41.5"
  YMax            "47.5"
  SaveName            "ScopeData5"
  DataFormat          "StructureWithTime"
  MaxDataPoints       "18000"
  SampleTime          "0"
}
Block {
  BlockType           Scope
  Name            "Scope\n(T_A_B)1"
  SID             "99"
  Ports           [1]
  Position            [485, 599, 515, 631]
  Floating            off
  Location            [28, 307, 575, 714]
  Open            off
  NumInputPorts       "1"
  List {
ListType        AxesTitles
axes1           "%<SignalLabel>"
  }
  List {
ListType        ScopeGraphics
FigureColor     "[0.5 0.5 0.5]"
AxesColor       "[0 0 0]"
AxesTickColor       "[1 1 1]"
LineColors      "[1 1 0;1 0 1;0 1 1;1 0 0;0 1 0;0 0 1]"
LineStyles      "-|-|-|-|-|-"
LineWidths      "[0.5 0.5 0.5 0.5 0.5 0.5]"
MarkerStyles        "none|none|none|none|none|none"
  }
  TimeRange           "500"
  YMin            "0.265"
  YMax            "0.35"
  SaveName            "ScopeData6"
  DataFormat          "StructureWithTime"
  MaxDataPoints       "18000"
  SampleTime          "0"
}
Block {
  BlockType           Scope
  Name            "Scope\n(T_A_B)2"
  SID             "100"
  Ports           [1]
  Position            [625, 569, 655, 601]
  Floating            off
  Location            [380, 141, 927, 548]
  Open            off
  NumInputPorts       "1"
  List {
```

```
ListType        AxesTitles
axes1           "%<SignalLabel>"
  }
  List {
ListType        ScopeGraphics
FigureColor     "[0.5 0.5 0.5]"
AxesColor       "[0 0 0]"
AxesTickColor       "[1 1 1]"
LineColors      "[1 1 0;1 0 1;0 1 1;1 0 0;0 1 0;0 0 1]"
LineStyles      "-|-|-|-|-|-"
LineWidths      "[0.5 0.5 0.5 0.5 0.5 0.5]"
MarkerStyles        "none|none|none|none|none|none"
  }
  TimeRange           "500"
  YMin            "51"
  YMax            "68"
  SaveName            "ScopeData7"
  DataFormat          "StructureWithTime"
  MaxDataPoints       "18000"
  SampleTime          "0"
}
Block {
  BlockType           Scope
  Name            "Scope\n(T_B_A)"
  SID             "52"
  Ports           [1]
  Position            [765, 674, 795, 706]
  Floating            off
  Location            [88, 384, 544, 746]
  Open            off
  NumInputPorts       "1"
  List {
ListType        AxesTitles
axes1           "%<SignalLabel>"
  }
  List {
ListType        ScopeGraphics
FigureColor     "[0.5 0.5 0.5]"
AxesColor       "[0 0 0]"
AxesTickColor       "[1 1 1]"
LineColors      "[1 1 0;1 0 1;0 1 1;1 0 0;0 1 0;0 0 1]"
LineStyles      "-|-|-|-|-|-"
LineWidths      "[0.5 0.5 0.5 0.5 0.5 0.5]"
MarkerStyles        "none|none|none|none|none|none"
  }
  TimeRange           "200"
  YMin            "43"
  YMax            "47.25"
  SaveName            "ScopeData1"
  DataFormat          "StructureWithTime"
  MaxDataPoints       "18000"
  SampleTime          "0"
}
Block {
  BlockType           Scope
  Name            "Scope\n(T_B_B)"
  SID             "72"
  Ports           [1]
  Position            [765, 834, 795, 866]
  Floating            off
  Location            [812, 349, 1307, 732]
  Open            off
  NumInputPorts       "1"
  ZoomMode            "yonly"
  List {
ListType        AxesTitles
axes1           "%<SignalLabel>"
  }
  List {
ListType        ScopeGraphics
FigureColor     "[0.501960784313725 0.501960784313725 0.501960784313725]"
AxesColor       "[0 0 0]"
AxesTickColor       "[1 1 1]"
LineColors      "[1 1 0;1 0 1;0 1 1;1 0 0;0 1 0;0 0 1]"
LineStyles      "-|-|-|-|-|-"
LineWidths      "[0.5 0.5 0.5 0.5 0.5 0.5]"
MarkerStyles        "none|none|none|none|none|none"
  }
  TimeRange           "500"
  YMin            "42.5"
  YMax            "49"
  SaveName            "ScopeData3"
  DataFormat          "StructureWithTime"
  MaxDataPoints       "18000"
  SampleTime          "0"
}
Block {
  BlockType           Sum
  Name            "Sum"
  SID             "8"
  Ports           [2, 1]
  Position            [810, 60, 830, 80]
  ShowName            off
  IconShape           "round"
  Inputs              "|++"
  InputSameDT         off
  OutDataTypeStr          "Inherit: Inherit via internal rule"
  SaturateOnIntegerOverflow off
}
Block {
  BlockType           Sum
  Name            "Sum2"
  SID             "148"
  Ports           [2, 1]
  Position            [485, 175, 505, 195]
  ShowName            off
  IconShape           "round"
  Inputs              "|++"
  InputSameDT         off
  OutDataTypeStr          "Inherit: Inherit via internal rule"
  SaturateOnIntegerOverflow off
}
Block {
  BlockType           Sum
```

```
    Name            "Sum4"
    SID             "53"
    Ports           [2, 1]
    Position            [690, 680, 710, 700]
    ShowName            off
    IconShape           "round"
    Inputs              "|++"
    InputSameDT         off
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Sum
    Name            "Sum5"
    SID             "73"
    Ports           [2, 1]
    Position            [690, 840, 710, 860]
    ShowName            off
    IconShape           "round"
    Inputs              "|++"
    InputSameDT         off
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Sum
    Name            "Sum6"
    SID             "87"
    Ports           [2, 1]
    Position            [690, 300, 710, 320]
    ShowName            off
    IconShape           "round"
    Inputs              "|++"
    InputSameDT         off
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Sum
    Name            "Sum7"
    SID             "88"
    Ports           [2, 1]
    Position            [690, 460, 710, 480]
    ShowName            off
    IconShape           "round"
    Inputs              "|++"
    InputSameDT         off
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(P_LR)"
    SID             "12"
    Ports           [1]
    Position            [795, 240, 855, 270]
    VariableName        "P_right"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(T_A_A)"
    SID             "91"
    Ports           [1]
    Position            [835, 360, 895, 390]
    VariableName        "T_A_A"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(T_A_B)"
    SID             "92"
    Ports           [1]
    Position            [835, 520, 895, 550]
    VariableName        "T_A_B"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(T_B_A)"
    SID             "55"
    Ports           [1]
    Position            [835, 740, 895, 770]
    VariableName        "T_B_A"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(T_B_B)"
    SID             "75"
    Ports           [1]
    Position            [835, 900, 895, 930]
    VariableName        "T_B_B"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(V_A_A)"
    SID             "89"
    Ports           [1]
    Position            [520, 370, 580, 400]
    VariableName        "V_A_A"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(V_A_B)"
    SID             "90"
    Ports           [1]
    Position            [520, 530, 580, 560]
    VariableName        "V_A_B"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(V_B_A)"
    SID             "54"
    Ports           [1]
    Position            [520, 750, 580, 780]
    VariableName        "V_B_A"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(V_B_B)"
    SID             "74"
    Ports           [1]
    Position            [520, 910, 580, 940]
    VariableName        "V_B_B"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(V_LR"
    SID             "33"
    Ports           [1]
    Position            [640, 125, 700, 155]
    VariableName        "V_left"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           ToWorkspace
    Name            "To Workspace\n(V_LR1"
    SID             "149"
    Ports           [1]
    Position            [315, 245, 375, 275]
    VariableName        "V_left1"
    MaxDataPoints       "inf"
    SampleTime          "-1"
    SaveFormat          "Array"
  }
  Block {
    BlockType           Reshape
    Name            "V_reshape\n(V_A_A)"
    SID             "93"
    Ports           [1, 1]
    Position            [445, 373, 475, 397]
  }
  Block {
    BlockType           Reshape
    Name            "V_reshape\n(V_A_B)"
    SID             "94"
    Ports           [1, 1]
    Position            [445, 533, 475, 557]
  }
  Block {
    BlockType           Reshape
    Name            "V_reshape\n(V_B_A)"
    SID             "56"
    Ports           [1, 1]
    Position            [445, 753, 475, 777]
  }
  Block {
    BlockType           Reshape
    Name            "V_reshape\n(V_B_B)"
    SID             "76"
    Ports           [1, 1]
    Position            [445, 913, 475, 937]
  }
  Block {
    BlockType           Reshape
    Name            "V_reshape\n(V_LR)"
    SID             "32"
    Ports           [1, 1]
    Position            [565, 128, 595, 152]
  }
  Block {
    BlockType           Reshape
    Name            "V_reshape\n(V_LR)1"
    SID             "150"
    Ports           [1, 1]
    Position            [240, 248, 270, 272]
  }
  Block {
    BlockType           Gain
    Name            "Volts to inH2O\n(V_LR)"
    SID             "21"
    Position            [565, 55, 595, 85]
    Gain            "2"
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Gain
    Name            "Volts to inH2O\n(V_LR)1"
    SID             "151"
    Position            [240, 170, 270, 200]
    Gain            "2"
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Gain
    Name            "inH2O to Pa\n(T_A_B)"
    SID             "98"
    Position            [510, 454, 585, 486]
    Gain            "194.22"
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Gain
    Name            "inH2O to Pa\n(T_B_A)"
    SID             "58"
    Position            [510, 674, 585, 706]
    Gain            "191.79"
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Gain
    Name            "inH2O to Pa\n(T_B_B)"
    SID             "78"
    Position            [510, 834, 585, 866]
    Gain            "200.11"
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Gain
    Name            "inH2O to Pa\n(V_LR)"
    SID             "5"
    Position            [630, 54, 705, 86]
    Gain            "249.082"
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Gain
    Name            "inH2O to Pa\n(V_LR)1"
    SID             "152"
    Position            [305, 169, 380, 201]
    Gain            "249.082"
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Block {
    BlockType           Gain
    Name            "mV to Celsius\n(T_A_A)"
    SID             "97"
    Position            [510, 294, 585, 326]
    Gain            "198.16"
    ParamDataTypeStr        "Inherit: Inherit via internal rule"
    OutDataTypeStr          "Inherit: Inherit via internal rule"
    SaturateOnIntegerOverflow off
  }
  Line {
    SrcBlock            "Correction\nConstant (V_LR)"
    SrcPort             1
    Points              [5, 0]
    DstBlock            "Sum"
    DstPort             2
  }
  Line {
    SrcBlock            "Reshape\n(P_LR)"
    SrcPort             1
    DstBlock            "To Workspace\n(P_LR)"
    DstPort             1
  }
  Line {
    SrcBlock            "Volts to inH2O\n(V_LR)"
    SrcPort             1
    DstBlock            "inH2O to Pa\n(V_LR)"
    DstPort             1
  }
  Line {
    SrcBlock            "inH2O to Pa\n(V_LR)"
    SrcPort             1
    DstBlock            "Correction\nGain (P_right)"
    DstPort             1
  }
  Line {
    SrcBlock            "Correction\nGain (P_right)"
    SrcPort             1
    DstBlock            "Sum"
    DstPort             1
  }
  Line {
    SrcBlock            "V_reshape\n(V_LR)"
    SrcPort             1
    DstBlock            "To Workspace\n(V_LR"
    DstPort             1
  }
  Line {
    SrcBlock            "Correction\nConstant (T_B_A)"
    SrcPort             1
    Points              [5, 0]
    DstBlock            "Sum4"
    DstPort             2
  }
  Line {
    SrcBlock            "Sum4"
    SrcPort             1
    Points              [15, 0]
    Branch {
```

```
DstBlock        "Scope\n(T_B_A)"
DstPort         1
  }
  Branch {
Points          [0, 65]
DstBlock        "Reshape\n(right)1"
DstPort         1
  }
}
Line {
  SrcBlock              "Reshape\n(right)1"
  SrcPort               1
  DstBlock              "To Workspace\n(T_B_A)"
  DstPort               1
}
Line {
  SrcBlock              "V_reshape\n(V_B_A)"
  SrcPort               1
  DstBlock              "To Workspace\n(V_B_A)"
  DstPort               1
}
Line {
  SrcBlock              "Pressure Sensor\nAnalog Input"
  SrcPort               7
  Points                [30, 0; 0, 100; 185, 0]
  Branch {
Points          [0, 75]
DstBlock        "V_reshape\n(V_B_A)"
DstPort         1
  }
  Branch {
DstBlock        "inH2O to Pa\n(T_B_A)"
DstPort         1
  }
}
Line {
  SrcBlock              "Correction\nConstant (T_B_B)"
  SrcPort               1
  Points                [5, 0]
  DstBlock              "Sum5"
  DstPort               2
}
Line {
  SrcBlock              "Sum5"
  SrcPort               1
  Points                [15, 0]
  Branch {
DstBlock        "Scope\n(T_B_B)"
DstPort         1
  }
  Branch {
Points          [0, 65]
DstBlock        "Reshape\n(T_B_B)"
DstPort         1
  }
}
Line {
  SrcBlock              "Reshape\n(T_B_B)"
  SrcPort               1
  DstBlock              "To Workspace\n(T_B_B)"
  DstPort               1
}
Line {
  SrcBlock              "inH2O to Pa\n(T_B_B)"
  SrcPort               1
  DstBlock              "Sum5"
  DstPort               1
}
Line {
  SrcBlock              "V_reshape\n(V_B_B)"
  SrcPort               1
  DstBlock              "To Workspace\n(V_B_B)"
  DstPort               1
}
Line {
  SrcBlock              "Pressure Sensor\nAnalog Input"
  SrcPort               8
  Points                [5, 0; 0, 190; 210, 0]
  Branch {
Points          [0, 75]
DstBlock        "V_reshape\n(V_B_B)"
DstPort         1
  }
  Branch {
DstBlock        "inH2O to Pa\n(T_B_B)"
DstPort         1
  }
}
Line {
  SrcBlock              "Correction\nConstant (T_A_A)"
  SrcPort               1
  Points                [5, 0]
  DstBlock              "Sum6"
  DstPort               2
}
Line {
  SrcBlock              "Sum6"
  SrcPort               1
  Points                [15, 0]
  Branch {
DstBlock        "Scope\n(T_A_A)"
DstPort         1
  }
  Branch {
Points          [0, 65]
DstBlock        "Reshape\n(T_A_A)"
DstPort         1
  }
}
Line {
  SrcBlock              "Reshape\n(T_A_A)"
  SrcPort               1
  DstBlock              "To Workspace\n(T_A_A)"
  DstPort               1
```

```
}
Line {
  SrcBlock            "mV to Celsius\n(T_A_A)"
  SrcPort             1
  DstBlock            "Sum6"
  DstPort             1
}
Line {
  SrcBlock            "V_reshape\n(V_A_A)"
  SrcPort             1
  DstBlock            "To Workspace\n(V_A_A)"
  DstPort             1
}
Line {
  SrcBlock            "Pressure Sensor\nAnalog Input"
  SrcPort             3
  Points              [215, 0]
  Branch {
Points          [0, 75]
DstBlock        "V_reshape\n(V_A_A)"
DstPort         1
  }
  Branch {
DstBlock        "mV to Celsius\n(T_A_A)"
DstPort         1
  }
}
Line {
  SrcBlock            "Correction\nConstant (T_A_B)"
  SrcPort             1
  Points              [5, 0]
  DstBlock            "Sum7"
  DstPort             2
}
Line {
  SrcBlock            "Sum7"
  SrcPort             1
  Points              [15, 0]
  Branch {
Points          [0, 65]
DstBlock        "Reshape\n(T_A_B)"
DstPort         1
  }
  Branch {
DstBlock        "Scope\n(T_A_B)"
DstPort         1
  }
}
Line {
  SrcBlock            "Reshape\n(T_A_B)"
  SrcPort             1
  DstBlock            "To Workspace\n(T_A_B)"
  DstPort             1
}
Line {
  SrcBlock            "inH2O to Pa\n(T_A_B)"
  SrcPort             1
  Points              [10, 0]
  Branch {
DstBlock        "Sum7"
DstPort         1
  }
  Branch {
Points          [0, 115]
DstBlock        "Scope\n(T_A_B)2"
DstPort         1
  }
}
Line {
  SrcBlock            "V_reshape\n(V_A_B)"
  SrcPort             1
  DstBlock            "To Workspace\n(V_A_B)"
  DstPort             1
}
Line {
  SrcBlock            "Pressure Sensor\nAnalog Input"
  SrcPort             4
  Points              [50, 0; 0, 90; 165, 0]
  Branch {
DstBlock        "inH2O to Pa\n(T_A_B)"
DstPort         1
  }
  Branch {
Points          [0, 70]
Branch {
  Points          [0, 5]
  DstBlock        "V_reshape\n(V_A_B)"
  DstPort         1
}
Branch {
  Points          [0, 75]
  DstBlock        "Scope\n(T_A_B)1"
  DstPort         1
}
  }
}
Line {
  SrcBlock            "inH2O to Pa\n(T_B_A)"
  SrcPort             1
  DstBlock            "Sum4"
  DstPort             1
}
Line {
  SrcBlock            "Pressure Sensor\nAnalog Input"
  SrcPort             1
  Points              [0, -100; 305, 0]
  Branch {
DstBlock        "Volts to inH2O\n(V_LR)"
DstPort         1
  }
  Branch {
Points          [0, 70]
DstBlock        "V_reshape\n(V_LR)"
DstPort         1
```

```
      }
    }
    Line {
      SrcBlock            "Pressure Sensor\nAnalog Input"
      SrcPort             2
      Points              [25, 0]
      Branch {
    DstBlock        "Volts to inH2O\n(V_LR)1"
    DstPort         1
      }
      Branch {
    DstBlock        "V_reshape\n(V_LR)1"
    DstPort         1
      }
    }
    Line {
      SrcBlock            "Correction\nConstant (V_LR)1"
      SrcPort             1
      Points              [5, 0]
      DstBlock            "Sum2"
      DstPort             2
    }
    Line {
      SrcBlock            "Volts to inH2O\n(V_LR)1"
      SrcPort             1
      DstBlock            "inH2O to Pa\n(V_LR)1"
      DstPort             1
    }
    Line {
      SrcBlock            "inH2O to Pa\n(V_LR)1"
      SrcPort             1
      DstBlock            "Correction\nGain (P_right)1"
      DstPort             1
    }
    Line {
      SrcBlock            "Correction\nGain (P_right)1"
      SrcPort             1
      DstBlock            "Sum2"
      DstPort             1
    }
    Line {
      SrcBlock            "V_reshape\n(V_LR)1"
      SrcPort             1
      DstBlock            "To Workspace\n(V_LR1"
      DstPort             1
    }
    Line {
      SrcBlock            "Sum"
      SrcPort             1
      Points              [10, 0; 0, 60]
      DstBlock            "Scope\n(P_LR)1"
      DstPort             1
    }
    Line {
      SrcBlock            "Sum2"
      SrcPort             1
      Points              [100, 0]
      Branch {
    DstBlock        "Scope\n(P_LR)"
    DstPort         1
      }
      Branch {
    Points          [0, 70]
    DstBlock        "Reshape\n(P_LR)"
    DstPort         1
      }
    }
    Annotation {
      Position            [65, 325]
    }
  }
}
```

D.9.2. GUI for Calculating Apparent Viscosities MATLAB Code

```
function varargout = AppViscSolver_GUI_170802(varargin)
% APPVISCSOLVER_GUI_170802 MATLAB code for AppViscSolver_GUI_170802.fig
%      APPVISCSOLVER_GUI_170802, by itself, creates a new APPVISCSOLVER_GUI_170802 or raises the existing
%      singleton*.
%
%      H = APPVISCSOLVER_GUI_170802 returns the handle to a new APPVISCSOLVER_GUI_170802 or the handle to
%      the existing singleton*.
%
%      APPVISCSOLVER_GUI_170802('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in APPVISCSOLVER_GUI_170802.M with the given input arguments.
%
%      APPVISCSOLVER_GUI_170802('Property','Value',...) creates a new APPVISCSOLVER_GUI_170802 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AppViscSolver_GUI_170802_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AppViscSolver_GUI_170802_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AppViscSolver_GUI_170802

% Last Modified by GUIDE v2.5 18-Aug-2017 22:14:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AppViscSolver_GUI_170802_OpeningFcn, ...
                   'gui_OutputFcn',  @AppViscSolver_GUI_170802_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

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

% UIWAIT makes AppViscSolver_GUI_170802 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% Hints: get(hObject,'String') returns contents of edit_open_channels as text
%        str2double(get(hObject,'String')) returns contents of edit_open_channels as a double
% Hint: edit controls usually have a white background on Windows. for CreateFcn
%       See ISPC and COMPUTER.

% --- Executes just before AppViscSolver_GUI_170802 is made visible.
function AppViscSolver_GUI_170802_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AppViscSolver_GUI_170802 (see VARARGIN)

% Choose default command line output for AppViscSolver_GUI_170802
handles.output = hObject;

%%% Setting custom default file paths %%%
handles.default_file_path = 'C:\Users\Niko\Sync\Research Sync\Results\Raw Data\';
handles.input_pres_path_and_name = '';
handles.flow_rate_path_and_name = '';
handles.pres_temp_path_and_name = '';

%%% Setting up the default fluid parameters and updating their fields %%%
handles.value_radiobutton_diw = 1; % Will select this radiobutton
handles.value_radiobutton_diw_and_gly = 0;
handles.value_radiobutton_power_law = 0;

handles.fluid_visc = '0';
handles.fluid_visc_temp = '0';

handles.K_constant = '0';
handles.n_constant = '0';

handles.open_channels = '16';
handles.scale_factor = '1';

% Setting the pop-up menu option #
handles.popup_sel_inlet_index = 1;
handles.popup_sel_channel_name_index = 1;
handles.nominal_width_index = 1;

update_gui(hObject, eventdata, handles)

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = AppViscSolver_GUI_170802_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%%%%% ----- User Functions ----- %%%%%

function read_files(hObject, eventdata, handles)

global synced_data

set(handles.text_status, 'String', 'Program has started reading the files.');

% Checks if all files have been selected
if exist(handles.input_pres_path_and_name,'file') == 0 || ...
        exist(handles.flow_rate_path_and_name,'file') == 0 || ...
        exist(handles.pres_temp_path_and_name,'file') == 0
    set(handles.text_status, 'String', 'Missing an input file.');

else
    % Reading in the date from the file
    input_pres_full_path = handles.input_pres_path_and_name;
    fid = fopen(input_pres_full_path);
    read_start_1 = 1;
    %Reading in line with format of "File Created: 01/22/2016  01:52:21 PM"
    pres_scan_data_1 = textscan(fid, '%s %s %f/%f/%f  %f:%f:%f %s','HeaderLines',read_start_1);
    fclose(fid);

    % Reading in line with format of "13:52:21.267  0.000   0.000   0.000   0.000   -0.029  0.000   -0.029  0.000   -0.029  3.051   -0.029  3.051   0.054"
    % read line after 13
    fid = fopen(input_pres_full_path);
    read_start_2 = 13 - read_start_1;
    pres_scan_data_2 = textscan(fid, '%s\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f','HeaderLines',read_start_2);
    fclose(fid);

    % Converts the date vectors to a date number
    pres_scan_data_3 = textscan(pres_scan_data_2{1,1}{1}, '%f:%f:%f');

    input_pres_datenum = datenum(pres_scan_data_1{1,5}, pres_scan_data_1{1,3}, pres_scan_data_1{1,4}, ...
        pres_scan_data_3{1,1}, pres_scan_data_3{1,2}, pres_scan_data_3{1,3});
    input_pres_start_time = datevec(input_pres_datenum);

    % Converting string time to number time
    input_pres_time = str2double(pres_scan_data_2{1,1}(1:end));
    input_pres_time(1,1) = 0;

    raw_input_pres_date_vec = datevec(input_pres_time/60/60/24);
    input_pres_datenum = datenum(raw_input_pres_date_vec) + datenum(input_pres_start_time);
    input_pres_date_vec = datevec(input_pres_datenum);

    % Taking the pressures from the channels from the textscan
    ch_input_pres(:,1) = pres_scan_data_2{1,7};
    ch_input_pres(:,2) = pres_scan_data_2{1,9};
    ch_input_pres(:,3) = pres_scan_data_2{1,11};
    ch_input_pres(:,4) = pres_scan_data_2{1,13};

    input_pres = ch_input_pres * 100; % Converting [mbar] to [Pa]
    flow_rate_full_path = handles.flow_rate_path_and_name;
    fid = fopen(flow_rate_full_path);
    read_start = 9;
    flow_scan_data = textscan(fid, '%f/%f/%f %f:%f:%f:%f\t%f\t%f\r\n','HeaderLines',read_start);
    fclose(fid);

    %Converts the date vectors to a date number
    flow_datenum = datenum(2000 + flow_scan_data{1,3}, flow_scan_data{1,1}, flow_scan_data{1,2}, ...
        flow_scan_data{1,4}, flow_scan_data{1,5}, flow_scan_data{1,6} + flow_scan_data{1,7});
    flow_datevec = datevec(flow_datenum);
    raw_flow_rate = flow_scan_data{1,8};

    % Opens data file and reads in the pressure, time and temperature variables
    pres_temp_full_path = handles.pres_temp_path_and_name;

    pres_var = 'P_right';
    time_var = 'tout';
    temp_var = 'T_A_A';

    % Checks if start_time exists in pressure/temperature sensor
    vars = whos('-file',pres_temp_full_path);
    start_time_exists = ismember('start_time', {vars.name});

    load(pres_temp_full_path);

    if not(start_time_exists)
        start_time = flow_datenum(1);
    end

    raw_pres_temp_date_vec = datevec(tout/60/60/24);
    pres_temp_datenum = datenum(raw_pres_temp_date_vec) + datenum(start_time);
    pres_temp_date_vec = datevec(pres_temp_datenum);

    % Adjusts data to sync times
    flow_start_time = flow_datenum(1);
    flow_end_time = flow_datenum(end);
    pres_start_time = pres_temp_datenum(1);
    pres_end_time = pres_temp_datenum(end);
    input_pres_start_time = input_pres_datenum(1);
    input_pres_end_time = input_pres_datenum(end);

    start_time_temporary = [flow_start_time, pres_start_time, input_pres_start_time];
    end_time_temporary = [flow_end_time, pres_end_time, input_pres_end_time];

    % Finds the closest value by splitting data into bins of averaged values
    sync_start_time_datenum = max(start_time_temporary);
    sync_end_time_datenum = min(end_time_temporary);

    input_pres_time_edges = [-inf; (input_pres_datenum(1:end-1) + input_pres_datenum(2:end))/2; +inf];
    pres_temp_time_edges = [-inf; (pres_temp_datenum(1:end-1) + pres_temp_datenum(2:end))/2; +inf];
    flow_time_edges = [-inf; (flow_datenum(1:end-1) + flow_datenum(2:end))/2; +inf];

    % [~, pres_temp_snyc_start_time_index] = histc(snyc_start_time_datenum, pres_temp_time_edges);
    % [~, pres_temp_snyc_end_time_index] = histc(snyc_end_time_datenum, pres_temp_time_edges);
    [~, flow_snyc_start_time_index] = histc(sync_start_time_datenum, flow_time_edges);
    [~, flow_snyc_end_time_index] = histc(sync_end_time_datenum, flow_time_edges);
    flow_time_num_points = flow_snyc_end_time_index - flow_snyc_start_time_index + 1;

    % Finds the closest value by splitting data into bins of averaged values
    [~, input_pres_time_index] = histc(flow_datenum(flow_snyc_start_time_index:flow_snyc_end_time_index), input_pres_time_edges);
    [~, pres_temp_time_index] = histc(flow_datenum(flow_snyc_start_time_index:flow_snyc_end_time_index), pres_temp_time_edges);

    % Finds the elapsed time since the start of the run
    start_vec = zeros(flow_time_num_points,6);
    start_vec = repmat(datevec(flow_datenum(flow_snyc_start_time_index)),[flow_time_num_points 1]);
    flow_time = etime(datevec(flow_datenum(flow_snyc_start_time_index:flow_snyc_end_time_index)),start_vec);
    input_pres_time = etime(datevec(input_pres_datenum(input_pres_time_index)),start_vec);
    pres_temp_time = etime(datevec(pres_temp_datenum(pres_temp_time_index)),start_vec);
    flow_rate = raw_flow_rate(flow_snyc_start_time_index:flow_snyc_end_time_index)*str2num(handles.scale_factor);

    % Test to ensure closest time value was taken
    %test_diff = [ flow_time, pres_temp_time, input_pres_time, flow_time - pres_temp_time];

    % Zeroing the data based on the first 10 seconds of the test (at 0 Pa)
    zeroing_time = 10;
    sampling_rate_tout = 0.02;

    raw_mean_pressure = mean(P_right, 2);
    mean_temperature_raw = mean(T_A_A, 2);

    mean_pressure = ...
        abs(raw_mean_pressure - mean(raw_mean_pressure(1:zeroing_time/sampling_rate_tout)));

    % time [s], measured pressure [Pa], flow rate [uL/min], input pressure [Pa]
    synced_data = [ ...
        pres_temp_time, ...
        mean_pressure(pres_temp_time_index), ...
        flow_rate, ...
        mean_temperature_raw(pres_temp_time_index), ...
        input_pres(input_pres_time_index,1), ...
        input_pres(input_pres_time_index,2), ...
        input_pres(input_pres_time_index,3), ...
        input_pres(input_pres_time_index,4)];

    % Test plot to make the data
    % plot(mean_pressure(time_index(:,1)) .* flow_rate(1:end_index))

    set(handles.text_status, 'String', 'Program has finished reading in the files.');
    guidata(hObject, handles);
end

function run_analysis(hObject, eventdata, handles)
set(handles.text_status, 'String', 'Program has started analyzing the data.');

global synced_data
global input_pressures
global analyzed_data
global min_app_visc_error
global max_app_visc_error

%% User defined values
% TODO add configuration file for these values

% Values before 160807
% actual_h_channels = mean([92.535, 96.733, 93.753, 97.659, 102.081]); % height of channels [um]
% actual_open_channels = 16;
% actual_20_um_width = 20;
% actual_40_um_width = mean([44.015, 44.532, 43.578, 43.450, 41.450]);
% actual_60_um_width = 60;
% actual_100_um_width = 100;

actual_20_um_width = 0;
actual_40_um_width = 0;
actual_60_um_width = 0;
actual_100_um_width = 0;

actual_open_channels = str2num(handles.open_channels);

% After 160807, width is average of smallest and widest part of channel
switch handles.selected_channel_name
    case 'A - 20 um' % This design did not work
        chosen_channel_width = 20;
        actual_h_channels = mean([]);
        actual_20_um_width = mean([]);
    case 'B - 40 um'
        chosen_channel_width = 40;
        actual_h_channels = ...
            mean([108.464, 111.662, 117.6, 120.448, 115.35, 113.77, 98.364, 101.072, 106.196, 104.442, 104.187, 102.756]);
        actual_40_um_width = ...
            mean([ ...
            mean([38.6, 36.44, 38.196, 32.99, 31.639, 34.8, 32.926, 32.926, 31.201, 32.379, 32.787, 31.773]), ...
            mean([70.567, 69.879, 75.105, 71.622, 75.295, 68.391, 72.757, 71.843, 71.54, 75.549, 73.54, 70.963]) ...
            ]);
    case 'C - 100 um'
        chosen_channel_width = 100;
        actual_h_channels = ...
            mean([117.18, 122.955, 121.192, 123.307, 107.641, 107.171, 114.403, 115.334, 114.403, 118.749, 117.834, 119.061]);
        actual_100_um_width = ...
            mean([ ...
            mean([92.61, 96.231, 105.902, 91.272, 103.876, 98.266, 97.495, 95.628, 101.817, 98.884, 103.496, 97.466]), ...
            mean([137.238, 155.522, 116.952, 107.264, 122.983, 122.882, 138.533, 142.965, 149.158, 139.809, 140.067, 136.327]) ...
            ]);
    case 'D - 60 um'
        chosen_channel_width = 60;
        actual_h_channels = ...
            mean([133.607, 130.7577, 112.053, 115.465, 111.099, 113.745, 107.93, 110.856, 104.674, 109.169, 108.342, 105.04]);
        actual_60_um_width = ...
            mean([ ...
            mean([61.482, 62.878, 57.773, 61.741, 63.456, 61.159, 60.814, 60.449, 61.588, 62.419, 60.779, 61.118]), ...
            mean([90.05, 90.163, 83.468, 86.888, 82.298, 85.739, 83.047, 83.04, 86.76, 86.029, 86.791, 90.064]) ...
            ]);
    case 'E - 20 um'
        chosen_channel_width = 20;
        actual_h_channels = ...
            mean([103.905, 109.47, 107.727, 115.229, 104.389, 107.955, 110.284, 106.728]);
        actual_20_um_width = ...
            mean([ ...
            mean([17.89, 16.745, 17.306, 23.553, 17.946, 15.496, 18.789, 17.655]), ...
            mean([44.269, 45.437, 41.899, 44.861, 44.665, 39.361, 45.599, 41.101]) ...
            ]);
    case 'F - 20 um'
        chosen_channel_width = 20;
        actual_h_channels = ...
            mean([98.769, 100.604, 99.766, 104.193, 100.714, 99.305, 100.475, 102.22, 101.273, 100.348, 101.772]);
        actual_20_um_width = ...
            mean([ ...
            mean([22.249, 29.559, 18.09, 24.937, 21.951, 20.78, 24.292, 26.345, 24.013, 19.902, 25.503]), ...
            mean([27.242, 30.734, 25.663, 29.025, 28.387, 28.094, 26.949, 27.231, 26.646, 25.768, 29.067]) ...
            ]);
    case 'G - 100 um'
        chosen_channel_width = 100;
        actual_h_channels = ...
            mean([110.673, 112.034, 110.928, 106.491, 115.361, 110.524, 107.482, 110.588, 106.969, 105.857, 104.677, 106.979]);
        actual_100_um_width = ...
            mean([ ...
            mean([97.608, 94.724, 98.922, 102.094, 99.963, 97.683, 97.916, 98.584, 100.408, 97.669, 98.338, 97.735]), ...
            mean([112.547, 112.644, 118.999, 120.096, 120.897, 123.052, 113.352, 113.021, 112.522, 115.289, 112.587, 114.56]) ...
            ]);
    case 'H - 40 um'
        chosen_channel_width = 40;
        actual_h_channels = ...
            mean([107.311, 103.313, 106.974, 108.018, 104.456, 99.043, 98.652, 99.898, 91.619, 94.852, 99.328, 99.876]);
        actual_40_um_width = ...
            mean([ ...
            mean([57.341, 45.265, 47.764, 45.784, 43.07, 44.353, 43.508, 39.497, 39.773, 41.22, 40.573, 38.916]), ...
            mean([69.873, 69.335, 69.544, 67.338, 71.306, 68.981, 67.83, 67.82, 58.661, 64.159, 60.662, 64.954]) ...
            ]);
    case 'I - 60 um'
        chosen_channel_width = 60;
        actual_h_channels = ...
            mean([104.667, 109.653, 107.308, 110.915, 111.798, 108.592, 108.306, 106.819, 104.763, 104.257, 101.514, 104.306]);
        actual_60_um_width = ...
            mean([ ...
            mean([59.032, 61.478, 57.574, 57.671, 54.838, 60.613, 61.189, 60.005, 64.957, 59.93, 63.117, 58.994]), ...
            mean([87.64, 83.339, 86.111, 83.154, 90.842, 86.392, 85.764, 84.901, 91.089, 87.357, 91.471, 88.805]) ...
            ]);
end

time = ...
    synced_data(:,1);
pressures = ...
    synced_data(1:end,2);
flow_rates = ...
    synced_data(1:end,3);
input_pressures = ...
    [synced_data(1:end,5), ...
    synced_data(1:end,6), ...
    synced_data(1:end,7), ...
    synced_data(1:end,8)];

% Setting up indexes of the end matrix
index_R_hyd = 1;
index_V_int = 2;

num_data_points = max(size(flow_rates)); % Finds how many values there are
num_indiv_resistances = 3; % Number of components in the measured system

% Loop to find error if the height and/or width are measured wrong

% Errors for width and height [um]
w_plus_minus = 3;
h_plus_minus = 3;
w_intervals = 3;
h_intervals = 3;
```

```
handles.value_radiobutton_diw = get(handles.radiobutton_diw, 'Value');
handles.value_radiobutton_diw_and_gly = get(handles.radiobutton_diw_and_gly, 'Value');

% From the min to max of range for width
for w_err = -w_plus_minus : w_intervals : w_plus_minus

    % The width is + or - the error value, based on the index
    w_ch_20_um = actual_20_um_width + w_err;
    w_ch_40_um = actual_40_um_width + w_err;
    w_ch_60_um = actual_60_um_width + w_err;
    w_ch_100_um = actual_100_um_width + w_err;

    for h_err = -h_plus_minus : h_intervals : h_plus_minus

        % The height is + or - the error value, based on the index
        h_channels = actual_h_channels + h_err;

        %% Channel Dimensions
        dims_chamber_1 = ...
            [500,   3740,    h_channels,    1591,   50, 100; ...    % Design with 20 um width channels
            500,    3100,    h_channels,    1271,   50, 100; ...    % Design with 40 um width channels
            500,    3100,    h_channels,    1271,   50, 100; ...    % Design with 60 um width channels
            500,    3100,    h_channels,    1271,   50, 100];       % Design with 100 um width channels
        % entrance width, exit width, height, length, post radius, post separation
        % distance ***

        dims_parallel_channels = ...
            [h_channels,    w_ch_20_um,     1200,   actual_open_channels; ... % Design with 20 um width channels
            h_channels,     w_ch_40_um,     4000,   actual_open_channels; ...   % Design with 40 um width channels
            h_channels,     w_ch_60_um,     4000,   actual_open_channels; ...   % Design with 60 um width channels
            h_channels,     w_ch_100_um,    4000,   actual_open_channels];      % Design with 100 um width channels
        % width, height, length, number of channels
        % individual channels [um]

        dims_chamber_2 = ...
            [3740,  500,    h_channels, 1591,   50, 100; ...    % Design with 20 um width channels
            3100,   500,    h_channels, 1271,   50, 100; ...    % Design with 40 um width channels
            3100,   500,    h_channels, 1271,   50, 100; ...    % Design with 60 um width channels
            3100,   500,    h_channels, 1271,   50, 100];       % Design with 100 um width channels
        % entrance width, exit width, height, length, post radius, post separation
        % distance

        %% Used for initializing matrices
        num_w_ch = size(dims_parallel_channels, 1);

        %% Resistance Functions

        % #1 pressure system component
        R_hyd_cham_1_pillar_constant = ...
            cham_pillar_constant_160115 (dims_chamber_1);
        % Resistance constant for rectangular channels
        % between first pressure sensor and parallel channels

        % #2 pressure system component
        R_hyd_parallel_individual_constant = rec_constant_160115 (dims_parallel_channels);
        % Resistance and internal volume for rectangular channels
        % individual parallel channels

        % #3 pressure system component
        R_hyd_eq_parallel_constant = ...
            R_hyd_parallel_individual_constant./dims_parallel_channels(:,4);
        % Equivalent hydraulic resistance and internal volume of
        % the parallel channels

        % #4 pressure system component
        R_hyd_cham_2_pillar_constant = ...
            cham_pillar_constant_160115 (dims_chamber_2);
        % Resistance and internal volume for rectangular channels
        % between first pressure sensor and parallel channels

        w_ch_index = handles.nominal_width_index;
        % [w_ch_index, ~, ~] = find(dims_parallel_channels(:,2) == chosen_channel_width + w_err);
        inlet_index = 69 - handles.inlet;

        % Initializing a column matrix of hydraulic resistance constants
        R_hyd_const = cat(1, ...
            R_hyd_cham_1_pillar_constant(w_ch_index), ...
            R_hyd_eq_parallel_constant(w_ch_index), ...
            R_hyd_cham_2_pillar_constant(w_ch_index));

        R_hyd_tot_const = single(sum(R_hyd_const));
        flow_rates_SI = single(flow_rates./(60*1e6*1e3));

        Q_parallel_individual = flow_rates./dims_parallel_channels(w_ch_index,4);
        % Flow rate in an individual channel

        app_shear_rate = app_shear_rate_rec_newtonian_160228(Q_parallel_individual, dims_parallel_channels, w_ch_index);

        if isfield(handles, 'value_radiobutton_power_law')
            if handles.value_radiobutton_power_law == 1
                % K and n calculated using 1/s and mPa*s, so there needs to be
                % divided by 1000
                app_shear_rate = app_shear_rate_rec_non_newtonian_170730(Q_parallel_individual, dims_parallel_channels, w_ch_index, handles.n_constant);
            end

        % Shear rate in each of the parallel channels
        end

        % End matrix will have:
        % col 1 = time [s]
        % col 2 = pressure [Pa]
        % col 3 = flow rate [uL/min]
        % col 4 = apparent viscosity [mPa*s]
        % col 5 = shear rate of individual channel [1/s]

        % Indices are incremented
        w_err_index = w_err + w_plus_minus + 1;
        h_err_index = h_err + h_plus_minus + 1;

        % Apparent viscosities are calculated
        calculated_app_viscosities (w_err_index,h_err_index,:) = ...
            pressures./(flow_rates_SI.*R_hyd_tot_const);

    end
end

% Handles the power law fluid separately to get the value
if isfield(handles, 'value_radiobutton_power_law')
    handles.value_radiobutton_power_law = get(handles.radiobutton_power_law, 'Value');
```

```
end

% Handles the DIW fluid separately to get the theoretical value
if handles.value_radiobutton_diw == 1
    % Water viscosity over temperature
    water_visc_20C = 1.002e-3;
    temp_minus_20 = 20-synced_data(:,4);
    exponent_water = (temp_minus_20./(synced_data(:,4)+96)).* ...
        (1.2378-1.303e-3.*temp_minus_20+3.06e-6.*temp_minus_20.^2+2.55e-8.*temp_minus_20.^3);
    theo_deionized_water_visc = 10.^(exponent_water).*water_visc_20C;

    theo_visc = theo_deionized_water_visc;
end

% Calculates the theoretical viscosity from the input reference viscosity
if handles.value_radiobutton_diw_and_gly == 1
    % Measured glycerol viscosity, converting from mPa*s to Pa*s
    theo_visc (1:num_data_points,1) = str2num(handles.fluid_visc)/1000;
end

% Calculates the theoretical viscosity from the input power law constants
if isfield(handles, 'value_radiobutton_power_law')
    if handles.value_radiobutton_power_law == 1
        % K and n calculated using 1/s and mPa*s, so there needs to be
        % divided by 1000
        theo_visc = ( ...
        str2num(handles.K_constant)*(app_shear_rate.^(str2num(handles.n_constant)-1)))*1e-3;
    end
end

% Initializing analyzed data matrix before filling with data
analyzed_data = zeros(num_data_points,5);

app_viscosities (:,1) = calculated_app_viscosities(w_plus_minus+1,h_plus_minus+1,:);
columnizing_min_app_visc_error (:,1) = calculated_app_viscosities(1,1,:);
min_app_visc_error = columnizing_min_app_visc_error;
columnizing_max_app_visc_error (:,1) = calculated_app_viscosities(w_plus_minus*2 + 1,h_plus_minus*2 + 1,:);
max_app_visc_error = columnizing_max_app_visc_error;

analyzed_data = ...
    [time, ...
    pressures, ...
    flow_rates, ...
    app_viscosities, ...
    theo_visc, ...
    app_shear_rate];

% Calculating the ratio between the theoretical viscosities and the
% calculated ones
ratio = theo_visc./analyzed_data(:,4);

set(handles.text_status, 'String', 'Program has finished analyzing the data and will prompt to save analyzed data.');

% Filename values
w_ch_chosen = chosen_channel_width;
h_ch_chosen = 100;

[~,pres_temp_file_name,~] = fileparts(handles.pres_temp_path_and_name);

default_save_full_path = ...
    strcat(handles.default_file_path, pres_temp_file_name, ...
    '_chip ', handles.selected_channel_name, ...
    '_inlet ', handles.inlet, ...
    '_h ', num2str(h_ch_chosen), ...
    ' - Analyzed');
[save_file_name,save_file_path,save_file_ext] = ...
    uiputfile('*.mat', 'Please choose file path and name to save variables to.', default_save_full_path);

save([save_file_path save_file_name '.mat']);

% Future check for file name
% if isequal(filename,0) || isequal(pathname,0)
%    disp('User selected Cancel')
% else
%    disp(['User selected ',fullfile(pathname,filename)])
% end

guidata(hObject, handles);

function run_analysis_with_norm(hObject, eventdata, handles)
set(handles.text_status, 'String', 'Program has started analyzing the data.');

% Incomplete normalization analysis function

global synced_data
global input_pressures
global analyzed_data
global min_app_visc_error
global max_app_visc_error

%% User defined values
% TODO add configuration file for these values

% Values before 160807
% actual_h_channels = mean([92.535, 96.733, 93.753, 97.659, 102.081]); % height of channels [um]
% actual_open_channels = 16;
% actual_20_um_width = 20;
% actual_40_um_width = mean([44.015, 44.532, 43.578, 43.450, 41.450]);
% actual_60_um_width = 60;
% actual_100_um_width = 100;

actual_20_um_width = 0;
actual_40_um_width = 0;
actual_60_um_width = 0;
actual_100_um_width = 0;

actual_open_channels = str2num(handles.open_channels);

% After 160807, width is average of smallest and widest part of channel
switch handles.selected_channel_name
    case 'A - 20 um' % This design did not work
        chosen_channel_width = 20;
        actual_h_channels = mean([]);
        actual_20_um_width = mean([]);
    case 'B - 40 um'
        chosen_channel_width = 40;
        actual_h_channels = ...
            mean([108.464, 111.662, 117.6, 120.448, 115.35, 113.77, 98.364, 101.072, 106.196, 104.442, 104.187, 102.756]);
        actual_40_um_width = ...
```

```
            mean([ ...
            mean([38.6, 36.44, 38.196, 32.99, 31.639, 34.8, 32.926, 32.926, 31.201, 32.379, 32.787, 31.773]), ...
            mean([70.567, 69.879, 75.105, 71.622, 75.295, 68.391, 72.757, 71.843, 71.54, 75.549, 73.54, 70.963]) ...
            ]);
    case 'C - 100 um'
        chosen_channel_width = 100;
        actual_h_channels = ...
            mean([117.18, 122.955, 121.192, 123.307, 107.641, 107.171, 114.403, 115.334, 114.403, 118.749, 117.834, 119.061]);
        actual_100_um_width = ...
            mean([ ...
            mean([92.61, 96.231, 105.902, 91.272, 103.876, 98.266, 97.495, 95.628, 101.817, 98.884, 103.496, 97.466]), ...
            mean([137.238, 155.522, 116.952, 107.264, 122.983, 122.882, 138.533, 142.965, 149.158, 139.809, 140.067, 136.327]) ...
            ]);
    case 'D - 60 um'
        chosen_channel_width = 60;
        actual_h_channels = ...
            mean([133.607, 130.7577, 112.053, 115.465, 111.099, 113.745, 107.93, 110.856, 104.674, 109.169, 108.342, 105.04]);
        actual_60_um_width = ...
            mean([ ...
            mean([61.482, 62.878, 57.773, 61.741, 63.456, 61.159, 60.814, 60.449, 61.588, 62.419, 60.779, 61.118]), ...
            mean([90.05, 90.163, 83.468, 86.888, 82.298, 85.739, 83.047, 83.04, 86.76, 86.029, 86.791, 90.064]) ...
            ]);
    case 'E - 20 um'
        chosen_channel_width = 20;
        actual_h_channels = ...
            mean([103.905, 109.47, 107.727, 115.229, 104.389, 107.955, 110.284, 106.728]);
        actual_20_um_width = ...
            mean([ ...
            mean([17.89, 16.745, 17.306, 23.553, 17.946, 15.496, 18.789, 17.655]), ...
            mean([44.269, 45.437, 41.899, 44.861, 44.665, 39.361, 45.599, 41.101]) ...
            ]);
    case 'F - 20 um'
        chosen_channel_width = 20;
        actual_h_channels = ...
            mean([98.769, 100.604, 99.766, 104.193, 100.714, 99.305, 100.475, 102.22, 101.273, 100.348, 101.772]);
        actual_20_um_width = ...
            mean([ ...
            mean([22.249, 29.559, 18.09, 24.937, 21.951, 20.78, 24.292, 26.345, 24.013, 19.902, 25.503]), ...
            mean([27.242, 30.734, 25.663, 29.025, 28.387, 28.094, 26.949, 27.231, 26.646, 25.768, 29.067]) ...
            ]);
    case 'G - 100 um'
        chosen_channel_width = 100;
        actual_h_channels = ...
            mean([110.673, 112.034, 110.928, 106.491, 115.361, 110.524, 107.482, 110.588, 106.969, 105.857, 104.677, 106.979]);
        actual_100_um_width = ...
            mean([ ...
            mean([97.608, 94.724, 98.922, 102.094, 99.963, 97.683, 97.916, 98.584, 100.408, 97.669, 98.338, 97.735]), ...
            mean([112.547, 112.644, 118.999, 120.096, 120.897, 123.052, 113.352, 113.021, 112.522, 115.289, 112.587, 114.56]) ...
            ]);
    case 'H - 40 um'
        chosen_channel_width = 40;
        actual_h_channels = ...
            mean([107.311, 103.313, 106.974, 108.018, 104.456, 99.043, 98.652, 99.898, 91.619, 94.852, 99.328, 99.876]);
        actual_40_um_width = ...
            mean([ ...
            mean([57.341, 45.265, 47.764, 45.784, 43.07, 44.353, 43.508, 39.497, 39.773, 41.22, 40.573, 38.916]), ...
            mean([69.873, 69.335, 69.544, 67.338, 71.306, 68.981, 67.83, 67.82, 58.661, 64.159, 60.662, 64.954]) ...
            ]);
    case 'I - 60 um'
        chosen_channel_width = 60;
        actual_h_channels = ...
            mean([104.667, 109.653, 107.308, 110.915, 111.798, 108.592, 108.306, 106.819, 104.763, 104.257, 101.514, 104.306]);
        actual_60_um_width = ...
            mean([ ...
            mean([59.032, 61.478, 57.574, 57.671, 54.838, 60.613, 61.189, 60.005, 64.957, 59.93, 63.117, 58.994]), ...
            mean([87.64, 83.339, 86.111, 83.154, 90.842, 86.392, 85.764, 84.901, 91.089, 87.357, 91.471, 88.805]) ...
            ]);
end

% Flow rates for the liquid [uL/min]
% *** test values
%end_index = 10000;

time = ...
    synced_data(:,1);
pressures = ...
    synced_data(1:end,2);
flow_rates = ...
    synced_data(1:end,3);
input_pressures = ...
    [synced_data(1:end,5), ...
    synced_data(1:end,6), ...
    synced_data(1:end,7), ...
    synced_data(1:end,8)];

% Normalizes the data based on best linear fit between measured pressure
% and input
handles.normalize = true;

%Finding the linear regression, b1 = y-int and b2 = slope
y = pressures;
x = input_pressures(:,1);
X = [ones(length(pressures),1) x];
b = X\y;
y_int_norm = b(1);
slope_norm = b(2);

norm_pressures = y_int_norm + slope_norm.*input_pressures(:,1);

% Setting up indexes of the end matrix
index_R_hyd = 1;
index_V_int = 2;

num_data_points = max(size(flow_rates)); % Finds how many values there are
num_indiv_resistances = 3; % Number of components in the measured system

% Loop to find error if the height and/or width are measured
% wrong

% Errors for width and height [um]
w_plus_minus = 3;
h_plus_minus = 3;
w_intervals = 3;
h_intervals = 3;

% From the min to max of range for width
```

```
for w_err = -w_plus_minus : w_intervals : w_plus_minus

    w_ch_20_um = actual_20_um_width + w_err;
    w_ch_40_um = actual_40_um_width + w_err;
    w_ch_60_um = actual_60_um_width + w_err;
    w_ch_100_um = actual_100_um_width + w_err;

    for h_err = -h_plus_minus : h_intervals : h_plus_minus

        h_channels = actual_h_channels + h_err;

        %% Channel Dimensions
        dims_chamber_1 = ...
            [500,   3740,    h_channels,    1591,   50, 100; ...    % Design with 20 um width channels
            500,    3100,    h_channels,    1271,   50, 100; ...    % Design with 40 um width channels
            500,    3100,    h_channels,    1271,   50, 100; ...    % Design with 60 um width channels
            500,    3100,    h_channels,    1271,   50, 100];       % Design with 100 um width channels
        % entrance width, exit width, height, length, post radius, post separation
        % distance ***

        dims_parallel_channels = ...
            [h_channels,    w_ch_20_um,     1200,   actual_open_channels; ... % Design with 20 um width channels
            h_channels,     w_ch_40_um,     4000,   actual_open_channels; ...   % Design with 40 um width channels
            h_channels,     w_ch_60_um,     4000,   actual_open_channels; ...   % Design with 60 um width channels
            h_channels,     w_ch_100_um,    4000,   actual_open_channels];      % Design with 100 um width channels
        % width, height, length, number of channels
        % individual channels [um]

        dims_chamber_2 = ...
            [3740,  500,    h_channels, 1591,   50, 100; ...    % Design with 20 um width channels
            3100,   500,    h_channels, 1271,   50, 100; ...    % Design with 40 um width channels
            3100,   500,    h_channels, 1271,   50, 100; ...    % Design with 60 um width channels
            3100,   500,    h_channels, 1271,   50, 100];       % Design with 100 um width channels
        % entrance width, exit width, height, length, post radius, post separation
        % distance

        %% Used for initializing matrices
        num_w_ch = size(dims_parallel_channels, 1);

        %% Resistance Functions

        % #1 pressure system component
        R_hyd_cham_1_pillar_constant = ...
            cham_pillar_constant_160115 (dims_chamber_1);
        % Resistance constant for rectangular channels
        % between first pressure sensor and parallel channels

        % #2 pressure system component
        R_hyd_parallel_individual_constant = rec_constant_160115 (dims_parallel_channels);
        % Resistance and internal volume for rectangular channels
        % individual parallel channels

        % #3 pressure system component
        R_hyd_eq_parallel_constant = ...
            R_hyd_parallel_individual_constant./dims_parallel_channels(:,4);
        % Equivalent hydraulic resistance and internal volume of
        % the parallel channels

        % #4 pressure system component
        R_hyd_cham_2_pillar_constant = ...
            cham_pillar_constant_160115 (dims_chamber_2);
        % Resistance and internal volume for rectangular channels
        % between first pressure sensor and parallel channels

        w_ch_index = handles.nominal_width_index;
        % [w_ch_index, ~, ~] = find(dims_parallel_channels(:,2) == chosen_channel_width + w_err);
        inlet_index = 69 - handles.inlet;

        % Initializing a column matrix of hydraulic resistance constants
        R_hyd_const = cat(1, ...
            R_hyd_cham_1_pillar_constant(w_ch_index), ...
            R_hyd_eq_parallel_constant(w_ch_index), ...
            R_hyd_cham_2_pillar_constant(w_ch_index));

        R_hyd_tot_const = single(sum(R_hyd_const));
        flow_rates_SI = single(flow_rates./(60*1e6*1e3));

        Q_parallel_individual = flow_rates./dims_parallel_channels(w_ch_index,4);
        % Flow rate in an individual channel

        app_shear_rate = app_shear_rate_rec_newtonian_160228(Q_parallel_individual, dims_parallel_channels, w_ch_index);
        % Shear rate in each of the parallel channels

        % End matrix will have:
        % col 1 = time [s]
        % col 2 = pressure [Pa]
        % col 3 = flow rate [uL/min]
        % col 4 = apparent viscosity [mPa*s]
        % col 5 = shear rate of individual channel [1/s]

        w_err_index = w_err + w_plus_minus + 1;
        h_err_index = h_err + h_plus_minus + 1;

        calculated_app_viscosities (w_err_index,h_err_index,:) = ...
            norm_pressures./(flow_rates_SI.*R_hyd_tot_const);

    end
end

handles.value_radiobutton_diw = get(handles.radiobutton_diw, 'Value');
handles.value_radiobutton_diw_and_gly = get(handles.radiobutton_diw_and_gly, 'Value');
if isfield(handles, 'value_radiobutton_power_law')
    handles.value_radiobutton_power_law = get(handles.radiobutton_power_law, 'Value');
end

if handles.value_radiobutton_diw == 1
    % Water viscosity over temperature
    water_visc_20C = 1.002e-3;
    temp_minus_20 = 20-synced_data(:,4);
    exponent_water = (temp_minus_20./(synced_data(:,4)+96)).* ...
        (1.2378-1.303e-3.*temp_minus_20+3.06e-6.*temp_minus_20.^2+2.55e-8.*temp_minus_20.^3);
    theo_deionized_water_visc = 10.^(exponent_water).*water_visc_20C;

    theo_visc = theo_deionized_water_visc;
end

if handles.value_radiobutton_diw_and_gly == 1
    % Measured glycerol viscosity, converting from mPa*s to Pa*s
    theo_visc (1:num_data_points,1) = str2num(handles.fluid_visc)/1000;
    % str2num(handles.fluid_visc_temp)
end

if isfield(handles, 'value_radiobutton_power_law')
    if handles.value_radiobutton_power_law == 1
        'testing power law'
    end
end

analyzed_data = zeros(num_data_points,5);

app_viscosities (:,1) = calculated_app_viscosities(w_plus_minus+1,h_plus_minus+1,:);
columnizing_min_app_visc_error (:,1) = calculated_app_viscosities(1,1,:);
min_app_visc_error = columnizing_min_app_visc_error;
columnizing_max_app_visc_error (:,1) = calculated_app_viscosities(w_plus_minus*2 + 1,h_plus_minus*2 + 1,:);
max_app_visc_error = columnizing_max_app_visc_error;

analyzed_data = ...
    [time, ...
    norm_pressures, ...
    flow_rates, ...
    app_viscosities, ...
    theo_visc, ...
    app_shear_rate];

% Calculating the ratio between the theoretical viscosities and the
% calculated ones
ratio = theo_visc./analyzed_data(:,4);

set(handles.text_status, 'String', 'Program has finished analyzing the data and will prompt to save analyzed data.');

% Filename values
w_ch_chosen = chosen_channel_width;
h_ch_chosen = 100;

[~,pres_temp_file_name,~] = fileparts(handles.pres_temp_path_and_name);

default_save_full_path = ...
    strcat(handles.default_file_path, pres_temp_file_name, ...
    '_chip ', handles.selected_channel_name, ...
    '_inlet ', handles.inlet, ...
    '_h ', num2str(h_ch_chosen), ...
    '_norm - Analyzed');
[save_file_name,save_file_path,save_file_ext] = ...
    uiputfile('*.mat', 'Please choose file path and name to save variables to.', default_save_full_path);

save([save_file_path save_file_name '.mat']);

% if isequal(filename,0) || isequal(pathname,0)
%    disp('User selected Cancel')
% else
%    disp(['User selected ',fullfile(pathname,filename)])
% end

guidata(hObject, handles);

function plot_analyzed_data(hObject, eventdata, handles)

%%% TODO center y-axis on theoretical viscosity

global input_pressures
global analyzed_data
global min_app_visc_error
global max_app_visc_error

%% Graphical Analysis
graph_start_time = 10;

time = analyzed_data(:,1);
pressures = analyzed_data(:,2);
flow_rates = analyzed_data(:,3);
app_viscosities = analyzed_data(:,4);
theo_visc = analyzed_data(:,5);
app_shear_rate = analyzed_data(:,6);

[~,pres_temp_file_name,~] = fileparts(handles.pres_temp_path_and_name);

time_edges = [-inf; (time(1:end-1) + time(2:end))/2; +inf];
[~, time_start] = histc(graph_start_time, time_edges);

app_visc = figure('Name','Error Estimates due to Dimension','NumberTitle','off');

% Apparent viscosity single plot
app_visc_samples_per_second = 1/0.1;
app_visc_avg_sec = 1;
coeff_app_visc_plot = ones(1, app_visc_avg_sec*app_visc_samples_per_second)/ ...
    (app_visc_samples_per_second*app_visc_avg_sec);
app_visc_fDelay = (length(coeff_app_visc_plot)-1)/2;
avg_app_visc_plot = filter(coeff_app_visc_plot, 1, app_viscosities);
app_visc_err_min_plot = filter(coeff_app_visc_plot, 1, min_app_visc_error);
app_visc_err_max_plot = filter(coeff_app_visc_plot, 1, max_app_visc_error);

app_visc_diff_err_min_plot = (avg_app_visc_plot-app_visc_err_min_plot);
app_visc_diff_err_max_plot = (app_visc_err_max_plot-avg_app_visc_plot);

plot(time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    app_visc_diff_err_min_plot(time_start:end),'--g', ...
    time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    app_visc_diff_err_max_plot(time_start:end),'--g'), ...
    axis tight;
title(strcat('Apparent Viscosity [', pres_temp_file_name, ']'))
xlabel('Time (s)') % x-axis label
ylabel('[Pa*s]') % y-axis label

app_visc = figure('Name','Percent Error Estimates due to Dimension','NumberTitle','off');

% Apparent viscosity single plot
app_visc_samples_per_second = 1/0.1;
app_visc_avg_sec = 1;
coeff_app_visc_plot = ones(1, app_visc_avg_sec*app_visc_samples_per_second)/ ...
    (app_visc_samples_per_second*app_visc_avg_sec);
app_visc_fDelay = (length(coeff_app_visc_plot)-1)/2;
avg_app_visc_plot = filter(coeff_app_visc_plot, 1, app_viscosities);
app_visc_err_min_plot = filter(coeff_app_visc_plot, 1, min_app_visc_error);
app_visc_err_max_plot = filter(coeff_app_visc_plot, 1, max_app_visc_error);

app_visc_percent_err_min_plot = (avg_app_visc_plot-app_visc_err_min_plot)./avg_app_visc_plot;
app_visc_percent_err_max_plot = (app_visc_err_max_plot-avg_app_visc_plot)./avg_app_visc_plot;
```

```
plot(time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    app_visc_percent_err_min_plot(time_start:end),'--g', ...
    time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    app_visc_percent_err_max_plot(time_start:end),'--g'), ...
    axis tight;
title(strcat('Apparent Viscosity [', pres_temp_file_name, ']'))
xlabel('Time (s)') % x-axis label
ylabel('[Pa*s]') % y-axis label

app_visc = figure('Name',pres_temp_file_name,'NumberTitle','off');

% Apparent viscosity single plot
app_visc_samples_per_second = 1/0.1;
app_visc_avg_sec = 1;
coeff_app_visc_plot = ones(1, app_visc_avg_sec*app_visc_samples_per_second)/ ...
    (app_visc_samples_per_second*app_visc_avg_sec);
app_visc_fDelay = (length(coeff_app_visc_plot)-1)/2;
avg_app_visc_plot = filter(coeff_app_visc_plot, 1, app_viscosities);
avg_theo_visc_plot = filter(coeff_app_visc_plot, 1, theo_visc);
app_visc_err_min_plot = filter(coeff_app_visc_plot, 1, min_app_visc_error);
app_visc_err_max_plot = filter(coeff_app_visc_plot, 1, max_app_visc_error);

plot(time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    avg_app_visc_plot(time_start:end),'b', ...
    time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    avg_theo_visc_plot(time_start:end),'r', ...
    time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    app_visc_err_min_plot(time_start:end),'--g', ...
    time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    app_visc_err_max_plot(time_start:end),'--g'), ...
    axis tight;
title(strcat('Apparent Viscosity [', pres_temp_file_name, ']'))
xlabel('Time (s)') % x-axis label
ylabel('[Pa*s]') % y-axis label
legend('Experimental','Theoretical')
%axis([-inf,inf,-1e-3,3e-3])

combined_figure = figure('Name',pres_temp_file_name,'NumberTitle','off');
%             b = uicontrol('Parent',combined_figure,'Style','slider','Position',[81,54,419,23],...
%                 'value',zeta, 'min',0, 'max',5);
%             b.Callback = @(es,ed) updateSystem(h,tf(wn^2,[1,2*(es.Value)*wn,wn^2]));
title('test')

% Input pressure subplot
input_pres_samples_per_second = 1/0.1;
input_pres_avg_sec = 3;
coeff_input_pres_plot = ones(1, input_pres_avg_sec*input_pres_samples_per_second)/ ...
    (input_pres_samples_per_second*input_pres_avg_sec);
input_pres_fDelay = (length(coeff_input_pres_plot)-1)/2;
avg_input_pres_plot = filter(coeff_input_pres_plot, 1, input_pressures);
%             app_visc.chk1 = uicontrol(app_visc.figure,'Style','checkbox','Position',[5 5 200 20],'String','Some text','Value',1);
%     app_visc.chk2 = uicontrol(app_visc.figure,'Style','checkbox','Position',[5 25 200 20],'String','Other text','Value',1);
%     app_visc.btn1 = uicontrol(app_visc.figure,'Style','pushbutton','Position',[5 45 200 20],'String','plot','Callback',@btnCallback);
subplot(2,2,1);
colour = ['k';'b';'r';'g'];
hold on
for i = 1:4
    plot (time(time_start:end) - input_pres_fDelay/input_pres_samples_per_second, ...
        avg_input_pres_plot(time_start:end,i), colour(i)), axis tight;
end
hold off
title('Input Pressure')
xlabel('Time (s)') % x-axis label
ylabel('[Pa]') % y-axis label
legend('Ch 1','Ch 2','Ch 3','Ch 4')

% Measured pressure subplot
pres_samples_per_second = 1/0.1;
coeff_pres_plot = ones(1, pres_samples_per_second)/pres_samples_per_second;
pres_fDelay = (length(coeff_pres_plot)-1)/2;
avg_pres_plot = filter(coeff_pres_plot, 1, pressures);

subplot(2,2,2);
plot (time(time_start:end)  - pres_fDelay/pres_samples_per_second, ...
    avg_pres_plot(time_start:end),'b'), axis tight;
title('Measured Pressure')
xlabel('Time (s)') % x-axis label
ylabel('[Pa]') % y-axis label

% Flow rate subplot
flow_samples_per_second = 1/0.02;
coeff_flow_plot = ones(1, flow_samples_per_second)/flow_samples_per_second;
flow_fDelay = (length(coeff_flow_plot)-1)/2;
avg_flow_plot = filter(coeff_flow_plot, 1, flow_rates);

subplot(2,2,3);
plot (time(time_start:end) - flow_fDelay/flow_samples_per_second, ...
    avg_flow_plot(time_start:end),'b'), axis tight;
title('Flow Rate')
xlabel('Time (s)') % x-axis label
ylabel('[\muL/min]') % y-axis label

% Apparent viscosity subplot
subplot(2,2,4);
plot(time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    avg_app_visc_plot(time_start:end),'b', ...
    time(time_start:end) - app_visc_fDelay/app_visc_samples_per_second, ...
    avg_theo_visc_plot(time_start:end),'r'), axis tight;
title('Apparent Viscosity')
xlabel('Time (s)') % x-axis label
ylabel('[Pa*s]') % y-axis label
legend('Experimental','Theoretical')
%axis([-inf,inf,-1e-3,3e-3])

% Test plotting input vs. measured
% figure(3);
% plot (avg_input_pres_plot(time_start:20000,1), ...
%     avg_pres_plot(time_start:20000),'*'), axis tight;

function [R_hyd_const] = rec_constant_160115 ...
    (dims_temp)
% Calculates the hydraulic resistance and internal volume of a rectangular channel using the:
% channel width (w) from dim_temp(:,1) *** needs to be larger than the height
% channel height (h) from dim_temp(:,2) *** needs to be smaller than the width
% channel length (L) from dim_temp(:,3)
% viscosity (visc)
```

```
w = dims_temp(:,1).*1e-6;   % converting um to [m]
h = dims_temp(:,2).*1e-6;   % converting um to [m]
L = dims_temp(:,3).*1e-6;   % converting um to [m]

num_widths = size(w);
for i = 1 : 1 : num_widths
    if sum(w(i) < h(i)) > 0
        temp = w(i);
        w(i) = h(i);
        h(i) = temp;
        warning('Program switched %.2d and %.2s for width and height.', w(i), h(i))
    end
end

syms n b_n
beta_n = ...
    (2*n - 1) .* pi./w;
n_iterations = 50;

sum_series = symsum( ...
    1./((beta_n).^4).* ...
    (1 - 2./(beta_n.*h) .* ...
    tanh(beta_n.*h./2)), ...
    n, 1, n_iterations);

%Initial test approximations
% R_hyd_const_approx1 = (12.*L)./(1-0.63.*(h./w))./((h.^3).*w);
% R_hyd_const_approx2 = (12.*L)./(w.*h.^3)./(1 - (6.*2.^5.*h)./(pi.^5.*w));

% Gets the hydraulic resistance constant in [Pa*s/m^3]
R_hyd_const = (L.*w)./(8.*h.*double(sum_series));

function [R_hyd_const] = cham_pillar_constant_160115 ...
    (dims_temp)
% Calculates the hydraulic resistance constant (without the viscosity) for a chamber filled with cylindrical pillars/posts using the:
% entrance width (b1)               from dim_temp(:,1)
% exit width (b2)                   from dim_temp(:,2)
% channel height (h)                from dim_temp(:,3)
% channel length (L)                from dim_temp(:,4)
% post radius (r_post)              from dim_temp(:,5)
% post separation distance (L_sep)  from dim_temp(:,6)

b1 =     dims_temp(:,1).*1e-6;      % converting [um] to [m]
b2 =     dims_temp(:,2).*1e-6;      % converting [um] to [m]
h =      dims_temp(:,3).*1e-6;      % converting [um] to [m]
L =      dims_temp(:,4).*1e-6;      % converting [um] to [m]
r_post = dims_temp(:,5).*1e-6;      % converting [um] to [m]
L_sep =  dims_temp(:,6).*1e-6;      % converting [um] to [m]

delta_sep = L_sep./2;

% Constants for quadratic fibre arrangement
C1_quad = 16/(9*pi*sqrt(2));
V_f_max_quad = pi/4;

% Constants for hexagonal fibre arrangement
% C1_hex = 16/(9*pi*sqrt(6));
% V_f_max_hex = pi/(2*sqrt(3));

% Choosing which fibre arrangement to use
C1 = C1_quad;
V_f_max = V_f_max_quad;

V_f = (pi*r_post.^2)./((2*r_post+2*delta_sep).^2);
% Fraction of pillar area
V_f_ratio = ((V_f_max./V_f).^(1/2))-1;
% Check for volume fraction ratio
K_Darcy_perp = C1.*((V_f_ratio).^(5/2)).*(r_post.^2);
% Darcy's law for perpendicular flow

area_flow_avg = ((b1+b2)/2).*h;
% Calculates the average flow area through the chamber as if there were
% no pillars

% hydralic resistance constant through the pillars
R_hyd_const=(L)./(K_Darcy_perp.*area_flow_avg);

function [shear_rate] = app_shear_rate_rec_newtonian_160228 ...
    (Q_temp, dims_temp, w_ch_index)
% Calculates the apparent shear rate in a rectangular channel using the:
% channel width (w) from dim_temp(:,1) *** needs to be larger than the height
% channel height (h) from dim_temp(:,2) *** needs to be smaller than the width
% channel length (L) from dim_temp(:,3)
% viscosity (visc)

w = dims_temp(w_ch_index,1).*1e-6;      % converting [um] to [m]
h = dims_temp(w_ch_index,2).*1e-6;      % converting [um] to [m];
Q = Q_temp./1000./1000000./60;  % converting [uL] to [m^3];

if sum(w(:) < h(:)) > 0
    error('Program error with width being smaller than the height')
end

w_h_ratio = h./w;

syms f_i
f_x_iterations = 16;

sum_series = single( ...
    symsum( ...
    tanh(pi./2.*(2*f_i-1).*w_h_ratio)./ ...
    (2*f_i-1).^5, ...
    f_i, 1, f_x_iterations));

f_star = ( ...
    1./((1+1./w_h_ratio).^2.* ...
    (1-192./(pi.^5.*w_h_ratio).*sum_series) ...
    ));

shear_rate = ...
    (6.*Q)./(w.*(h.^2)).* ...
    (1+w_h_ratio).* ...
    f_star;

shear_rate = shear_rate;

function [shear_rate] = app_shear_rate_rec_non_newtonian_170730 ...
    (Q_temp, dims_temp, w_ch_index, n_constant)
% Calculates the apparent shear rate in a rectangular channel using the:
```

```
% channel width (w) from dim_temp(:,1) *** needs to be larger than the height
% channel height (h) from dim_temp(:,2) *** needs to be smaller than the width
% channel length (L) from dim_temp(:,3)
% viscosity (visc)

w = dims_temp(w_ch_index,1).*1e-6;      % converting [um] to [m]
h = dims_temp(w_ch_index,2).*1e-6;      % converting [um] to [m];
Q = Q_temp./1000./1000000./60;  % converting [uL] to [m^3];

if sum(w(:) < h(:)) > 0
    error('Program error with width being smaller than the height')
end

w_h_ratio = h./w;

syms f_i a_i a_b_i
f_x_iterations = 16;
a_x_iterations = 100;
a_b_x_iterations = 100;

sum_series = single( ...
    symsum( ...
    tanh(pi./2.*(2*f_i-1).*w_h_ratio)./ ...
    (2*f_i-1).^5, ...
    f_i, 1, f_x_iterations));

f_star = ( ...
    1./((1+1./w_h_ratio).^2.* ...
    (1-192./(pi.^5.*w_h_ratio).*sum_series) ...
    ))

a_sum_series = single( ...
    symsum( ...
    ((-1).^(a_i+1))./ ...
    (((2*a_i+1)./2.*pi).^3).* ...
    1./ ...
    cosh(((2*a_i+1)*pi)./(2.*w_h_ratio)), ...
    a_i, 0, a_x_iterations));

a_star = 1./ ...
    (2.*(1+w_h_ratio).^2.* ...
    (1+4.*a_sum_series))

a_b_sum_series = single( ...
    symsum( ...
    tanh(((2.*a_b_i+1).*pi)./(2.*w_h_ratio))./ ...
    (2*a_b_i+1).^5, ...
    a_b_i, 0, a_b_x_iterations));

a_and_b_star = 3./ ...
    (2.* ...
    (1+w_h_ratio).^2.* ...
    (1-192./pi.^5.*w_h_ratio.*a_b_sum_series));

b_star = a_and_b_star - a_star

app_shear_rate = ...
    (6.*Q)./(w.*(h.^2)).* ...
    (1+w_h_ratio).* ...
    f_star;

shear_rate = app_shear_rate*(2/3)*(b_star/f_star + a_star/f_star*1/str2num(n_constant));

function reset_gui(hObject, eventdata, handles)
% Resets the values in the GUI

%%% Setting custom default file path %%%
handles.default_file_path = 'C:\Users\Niko\Sync\Research Sync\Results\Raw Data\';
set(handles.text_current_directory_path, 'String', handles.default_file_path)

%%% Setting up default path names %%%
handles.input_pres_path_and_name = '';
set(handles.edit_input_pres_path, 'String', 'Please select a file')
handles.flow_rate_path_and_name = '';
set(handles.edit_flow_rate_path, 'String', 'Please select a file')
handles.pres_temp_path_and_name = '';
set(handles.edit_pres_temp_path, 'String', 'Please select a file')

%%% Setting up the default fluid parameters and updating their fields %%%
handles.nominal_width_index = 1;
handles.inlet = 'A';

set(handles.radiobutton_diw,'Value',1);
handles.K_constant = '0';
set(handles.edit_K_constant,'String',handles.K_constant);
handles.n_constant = '0';
set(handles.edit_n_constant,'String',handles.n_constant);

handles.fluid_visc = '0';
set(handles.edit_fluid_visc,'String',handles.fluid_visc);
handles.fluid_visc_temp = '0';
set(handles.edit_fluid_visc_temp,'String',handles.fluid_visc_temp);

handles.scale_factor = '1';
set(handles.edit_scale_factor,'String',handles.scale_factor);
handles.open_channels = '16';
set(handles.edit_open_channels,'String',handles.open_channels);

% Setting the pop-up menu option #
handles.selected_channel_name = 'A - 20 um';
set(handles.popupmenu_channel_name, 'Value',1);
handles.nominal_width_index = 1;
set(handles.popupmenu_inlet,'Value',handles.nominal_width_index);

guidata(hObject, handles);

function update_gui(hObject, eventdata, handles)

set(handles.text_current_directory_path, 'String', handles.default_file_path);
set(handles.edit_input_pres_path, 'String', handles.input_pres_path_and_name);
set(handles.edit_flow_rate_path, 'String', handles.flow_rate_path_and_name);
set(handles.edit_pres_temp_path, 'String', handles.pres_temp_path_and_name);

set(handles.radiobutton_diw,'Value',handles.value_radiobutton_diw);
set(handles.radiobutton_diw_and_gly,'Value',handles.value_radiobutton_diw_and_gly);
if isfield(handles, 'value_radiobutton_power_law')
    set(handles.radiobutton_power_law,'Value',handles.value_radiobutton_power_law);
    set(handles.edit_K_constant,'String',handles.K_constant);
    set(handles.edit_n_constant,'String',handles.n_constant);
```

```
end
set(handles.edit_fluid_visc, 'String', handles.fluid_visc);
set(handles.edit_fluid_visc_temp, 'String', handles.fluid_visc_temp);
set(handles.edit_open_channels, 'String', handles.open_channels);
set(handles.edit_scale_factor, 'String', handles.scale_factor);

set(handles.popupmenu_inlet, 'Value', handles.popup_sel_inlet_index)
set(handles.popupmenu_channel_name, 'Value', handles.popup_sel_channel_name_index);

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.

%%%%% ----- Call Backs ----- %%%%%

function pushbutton_run_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_run.
% hObject    handle to pushbutton_run (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
read_files(hObject, eventdata, handles);
run_analysis(hObject, eventdata, handles);
plot_analyzed_data(hObject, eventdata, handles);

% --- Executes on button press in pushbutton_run_with_norm.
function pushbutton_run_with_norm_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_run_with_norm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
read_files(hObject, eventdata, handles);
run_analysis_with_norm(hObject, eventdata, handles);
plot_analyzed_data(hObject, eventdata, handles);

function pushbutton_diw_test_file_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_diw_test_file.
% hObject    handle to pushbutton_diw_test_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%% Updating GUI with the file path information %%%
% File path and names
handles.default_file_path = 'C:\Users\Niko\Sync\Research Sync\Results\Raw Data\160318 - DI Water Cal\';
set(handles.text_current_directory_path, 'String', handles.default_file_path);

input_pres_file_name = '160318 - Exp 2 DIW_Default_measurements_2016-03-18_1601.txt';
handles.input_pres_path_and_name = strcat(handles.default_file_path,input_pres_file_name);
set(handles.edit_input_pres_path, 'String', handles.input_pres_path_and_name, 'HorizontalAlignment','right');

flow_rate_file_name = '160318 - Exp 2 DIW_FRPData_20160318_1601.txt';
handles.flow_rate_path_and_name = strcat(handles.default_file_path, flow_rate_file_name);
set(handles.edit_flow_rate_path, 'String', handles.flow_rate_path_and_name);

pres_temp_file_name = '160318 - Exp 2 DIW.mat';
handles.pres_temp_path_and_name = strcat(handles.default_file_path, pres_temp_file_name);
set(handles.edit_pres_temp_path, 'String', handles.pres_temp_path_and_name);

% Fluid paramters
set(handles.radiobutton_diw,'Value',1);
handles.fluid_visc = '0'; % In mPa*s
set(handles.edit_fluid_visc, 'String', handles.fluid_visc);
handles.fluid_visc_temp = '0';
set(handles.edit_fluid_visc_temp, 'String', handles.fluid_visc_temp);

handles.open_channels = '16';
set(handles.edit_open_channels, 'String', handles.open_channels);

handles.scale_factor = '1';
set(handles.edit_scale_factor, 'String', handles.scale_factor);

handles.inlet = 'C';
set(handles.popupmenu_inlet, 'Value', 3);

handles.selected_channel_name = 'H - 40 um';
set(handles.popupmenu_channel_name, 'Value', 8);
handles.nominal_width_index = 2;

guidata(hObject, handles);
pushbutton_run_Callback(hObject, eventdata, handles)

% --- Executes on button press in pushbutton_diw_norm_test_file.
function pushbutton_diw_norm_test_file_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_diw_norm_test_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%% Updating GUI with the file path information %%%
% File path and names
handles.default_file_path = 'C:\Users\Niko\Sync\Research Sync\Results\Raw Data\160318 - DI Water Cal\';
set(handles.text_current_directory_path, 'String', handles.default_file_path);

input_pres_file_name = '160318 - Exp 2 DIW_Default_measurements_2016-03-18_1601.txt';
handles.input_pres_path_and_name = strcat(handles.default_file_path,input_pres_file_name);
set(handles.edit_input_pres_path, 'String', handles.input_pres_path_and_name, 'HorizontalAlignment','right');

flow_rate_file_name = '160318 - Exp 2 DIW_FRPData_20160318_1601.txt';
handles.flow_rate_path_and_name = strcat(handles.default_file_path, flow_rate_file_name);
set(handles.edit_flow_rate_path, 'String', handles.flow_rate_path_and_name);

pres_temp_file_name = '160318 - Exp 2 DIW.mat';
handles.pres_temp_path_and_name = strcat(handles.default_file_path, pres_temp_file_name);
set(handles.edit_pres_temp_path, 'String', handles.pres_temp_path_and_name);

% Fluid paramters
set(handles.radiobutton_diw,'Value',1);
handles.fluid_visc = '0'; % In mPa*s
set(handles.edit_fluid_visc, 'String', handles.fluid_visc);
handles.fluid_visc_temp = '0';
set(handles.edit_fluid_visc_temp, 'String', handles.fluid_visc_temp);

handles.open_channels = '16';
set(handles.edit_open_channels, 'String', handles.open_channels);

handles.scale_factor = '1';
set(handles.edit_scale_factor, 'String', handles.scale_factor);

handles.inlet = 'C';
set(handles.popupmenu_inlet, 'Value', 3);
```

```
handles.selected_channel_name = 'H - 40 um';
set(handles.popupmenu_channel_name, 'Value', 8);
handles.nominal_width_index = 2;

guidata(hObject, handles);
pushbutton_run_with_norm_Callback(hObject, eventdata, handles)

%%%% Test file for diw/gly 10-08-16
% --- Executes on button press in pushbutton_diw_gly_test_file.
function pushbutton_diw_gly_test_file_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_diw_gly_test_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%% Updating GUI with the file path information %%%
% File path and names
handles.default_file_path = 'C:\Users\Niko\Sync\Research Sync\Results\Raw Data\160810 - Gly-DIW 20 Pas\';
set(handles.text_current_directory_path, 'String', handles.default_file_path);

input_pres_file_name = '160810 - Exp 5 Gly-DIW 20 Pas_Default_measurements_2016-08-10_1639.txt';
handles.input_pres_path_and_name = strcat(handles.default_file_path,input_pres_file_name);
set(handles.edit_input_pres_path, 'String', handles.input_pres_path_and_name, 'HorizontalAlignment','right');

flow_rate_file_name = '160810 - Exp 5 Gly-DIW 20 Pas_FRPData_20160810_1639.txt';
handles.flow_rate_path_and_name = strcat(handles.default_file_path, flow_rate_file_name);
set(handles.edit_flow_rate_path, 'String', handles.flow_rate_path_and_name);

pres_temp_file_name = '160810 - Exp 5 Gly-DIW 20 Pas.mat';
handles.pres_temp_path_and_name = strcat(handles.default_file_path, pres_temp_file_name);
set(handles.edit_pres_temp_path, 'String', handles.pres_temp_path_and_name);

% Fluid paramters
set(handles.radiobutton_diw_and_gly,'Value',1);
handles.fluid_visc = '17.5383'; % In mPa*s
set(handles.edit_fluid_visc, 'String', handles.fluid_visc);
handles.fluid_visc_temp = '22.6217';
set(handles.edit_fluid_visc_temp, 'String', handles.fluid_visc_temp);

handles.open_channels = '16';
set(handles.edit_open_channels, 'String', handles.open_channels);

handles.scale_factor = '1.5121';
set(handles.edit_scale_factor, 'String', handles.scale_factor);

handles.inlet = 'C';
set(handles.popupmenu_inlet, 'Value', 3);

handles.selected_channel_name = 'H - 40 um';
set(handles.popupmenu_channel_name, 'Value', 8);
handles.nominal_width_index = 2;

guidata(hObject, handles);
pushbutton_run_Callback(hObject, eventdata, handles)

% --- Executes on button press in pushbuttondiw_gly_norm_test_file.
function pushbuttondiw_gly_norm_test_file_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttondiw_gly_norm_test_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%% Updating GUI with the file path information %%%
% File path and names
handles.default_file_path = 'C:\Users\Niko\Sync\Research Sync\Results\Raw Data\160810 - Gly-DIW 20 Pas\';
set(handles.text_current_directory_path, 'String', handles.default_file_path);

input_pres_file_name = '160810 - Exp 5 Gly-DIW 20 Pas_Default_measurements_2016-08-10_1639.txt';
handles.input_pres_path_and_name = strcat(handles.default_file_path,input_pres_file_name);
set(handles.edit_input_pres_path, 'String', handles.input_pres_path_and_name, 'HorizontalAlignment','right');

flow_rate_file_name = '160810 - Exp 5 Gly-DIW 20 Pas_FRPData_20160810_1639.txt';
handles.flow_rate_path_and_name = strcat(handles.default_file_path, flow_rate_file_name);
set(handles.edit_flow_rate_path, 'String', handles.flow_rate_path_and_name);

pres_temp_file_name = '160810 - Exp 5 Gly-DIW 20 Pas.mat';
handles.pres_temp_path_and_name = strcat(handles.default_file_path, pres_temp_file_name);
set(handles.edit_pres_temp_path, 'String', handles.pres_temp_path_and_name);

% Fluid paramters
set(handles.radiobutton_diw_and_gly,'Value',1);
handles.fluid_visc = '17.5383'; % In mPa*s
set(handles.edit_fluid_visc, 'String', handles.fluid_visc);
handles.fluid_visc_temp = '22.6217';
set(handles.edit_fluid_visc_temp, 'String', handles.fluid_visc_temp);

handles.open_channels = '16';
set(handles.edit_open_channels, 'String', handles.open_channels);

handles.scale_factor = '1.5121';
set(handles.edit_scale_factor, 'String', handles.scale_factor);

handles.inlet = 'C';
set(handles.popupmenu_inlet, 'Value', 3);

handles.selected_channel_name = 'H - 40 um';
set(handles.popupmenu_channel_name, 'Value', 8);
handles.nominal_width_index = 2;

guidata(hObject, handles);
pushbutton_run_with_norm_Callback(hObject, eventdata, handles)

function pushbutton_input_pres_path_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_input_pres_path.
% hObject    handle to pushbutton_input_pres_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%% Asking for the input pressure file and setting the default directory %%%
[input_pres_file_name,input_pres_path,~] = uigetfile(strcat(handles.default_file_path, '*.txt'), ...
    'Please select the input pressure file to use (*.txt)');

%%% Inputting path and file name into handles structure %%%
if input_pres_file_name ~= 0
    handles.input_pres_path_and_name = strcat(input_pres_path, input_pres_file_name);
    handles.default_file_path = input_pres_path;
    guidata(hObject, handles);

    %%% Updating GUI with the information %%%
    set(handles.edit_input_pres_path, 'String', handles.input_pres_path_and_name)
    set(handles.text_current_directory_path, 'String', handles.default_file_path)
```

```
end

function pushbutton_flow_rate_path_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_flow_rate_path.
% hObject    handle to pushbutton_flow_rate_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%% Asking for the input pressure file and setting the default directory %%%
[flow_rate_file_name,flow_rate_path,~] = uigetfile(strcat(handles.default_file_path, '*.txt'), ...
            'Please select the flow rate file to use (*.txt)');

%%% Inputting path and file name into handles structure %%%.
if flow_rate_file_name ~= 0
    handles.flow_rate_path_and_name = strcat(flow_rate_path, flow_rate_file_name);
    handles.default_file_path = flow_rate_path;
    guidata(hObject, handles);

    %%% Updating GUI with the information %%%
    set(handles.edit_flow_rate_path, 'String', handles.flow_rate_path_and_name)
    set(handles.text_current_directory_path, 'String', handles.default_file_path)
end

function pushbutton_pres_temp_path_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_pres_temp_path.
% hObject    handle to pushbutton_pres_temp_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%% Asking for the input pressure file and setting the default directory %%%
[pres_temp_file_name,pres_temp_path,~] = uigetfile(strcat(handles.default_file_path, '*.mat'), ...
                'Please select the presssure/temperature sensor file to use (*.mat)');

%%% Inputting path and file name into handles structure %%%
if pres_temp_file_name ~= 0
    handles.pres_temp_path_and_name = strcat(pres_temp_path, pres_temp_file_name);
    handles.default_file_path = pres_temp_path;
    guidata(hObject, handles);

    %%% Updating GUI with the information %%%
    set(handles.edit_pres_temp_path, 'String', handles.pres_temp_path_and_name)
    set(handles.text_current_directory_path, 'String', handles.default_file_path)
end

function pushbutton_reset_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_reset.
% hObject    handle to pushbutton_reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

reset_gui(hObject, eventdata, handles);
set(handles.text_status, 'String', 'Program was reset.');
guidata(hObject, handles);

function pushbutton_load_settings_from_file_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_load_settings_from_file.
% hObject    handle to pushbutton_load_settings_from_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%% Asking for the input pressure file and setting the default directory %%%
[load_settings_file_name,load_settings_path,~] = uigetfile(strcat(handles.default_file_path, '*.mat'), ...
    'Please select the presssure/temperature sensor file to use (*.mat)');

%%% Inputting path and file name into handles structure %%%
if load_settings_file_name ~= 0
    handles.full_settings_load_path = strcat(load_settings_path, load_settings_file_name);
    load(handles.full_settings_load_path);

    %% Updating GUI with the file path information %%%
    % File path and names
    handles.default_file_path = gui_param.default_file_path;
    handles.input_pres_path_and_name = gui_param.input_pres_path_and_name;
    handles.flow_rate_path_and_name = gui_param.flow_rate_path_and_name;
    handles.pres_temp_path_and_name = gui_param.pres_temp_path_and_name;

    % Fluid paramters
    handles.value_radiobutton_diw = gui_param.value_radiobutton_diw;
    handles.value_radiobutton_diw_and_gly = gui_param.value_radiobutton_diw_and_gly;
    if isfield(gui_param, 'value_radiobutton_power_law') == 1
        handles.value_radiobutton_power_law = gui_param.value_radiobutton_power_law;
        handles.K_constant = gui_param.K_constant;
        handles.n_constant = gui_param.n_constant;
    end

    handles.fluid_visc = gui_param.fluid_visc; % In mPa*s
    handles.fluid_visc_temp = gui_param.fluid_visc_temp;
    handles.open_channels = gui_param.open_channels;
    handles.scale_factor = gui_param.scale_factor;

    handles.inlet = gui_param.inlet;
    switch handles.inlet
        case 'A'
            handles.popup_sel_inlet_index = 1;
        case 'B'
            handles.popup_sel_inlet_index = 2;
        case 'C'
            handles.popup_sel_inlet_index = 3;
        case 'D'
            handles.popup_sel_inlet_index = 4;
    end

    handles.selected_channel_name = gui_param.selected_channel_name;
    handles.nominal_width_index = gui_param.nominal_width_index;

    switch handles.selected_channel_name
        case 'A - 20 um'
            handles.popup_sel_channel_name_index = 1;
        case 'B - 40 um'
            handles.popup_sel_channel_name_index = 2;
        case 'C - 100 um'
            handles.popup_sel_channel_name_index = 3;
        case 'D - 60 um'
            handles.popup_sel_channel_name_index = 4;
        case 'E - 20 um'
            handles.popup_sel_channel_name_index = 5;
        case 'F - 20 um'
            handles.popup_sel_channel_name_index = 6;
        case 'G - 100 um'
            handles.popup_sel_channel_name_index = 7;
        case 'H - 40 um'
            handles.popup_sel_channel_name_index = 8;
        case 'I - 60 um'
            handles.popup_sel_channel_name_index = 9;
    end
    guidata(hObject, handles);

    update_gui(hObject, eventdata, handles);
end

function pushbutton_save_settings_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_save_settings.
% hObject    handle to pushbutton_save_settings (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%% TODO get save file name
%%% TODO save file

%%% Setting file paths %%%
gui_param.default_file_path = handles.default_file_path;

gui_param.input_pres_path_and_name = handles.input_pres_path_and_name;
gui_param.flow_rate_path_and_name = handles.flow_rate_path_and_name;
gui_param.pres_temp_path_and_name = handles.pres_temp_path_and_name;

% Fluid paramters
gui_param.value_radiobutton_diw = get(handles.radiobutton_diw,'Value');
gui_param.value_radiobutton_diw_and_gly = get(handles.radiobutton_diw_and_gly,'Value');
gui_param.value_radiobutton_power_law = get(handles.radiobutton_power_law,'Value');
gui_param.fluid_visc = get(handles.edit_fluid_visc,'String');
gui_param.fluid_visc_temp = get(handles.edit_fluid_visc_temp,'String');
gui_param.K_constant = get(handles.edit_K_constant,'String');
gui_param.n_constant = get(handles.edit_n_constant,'String');

gui_param.open_channels = get(handles.edit_open_channels,'String');
gui_param.scale_factor = get(handles.edit_scale_factor,'String');

gui_param.inlet = handles.inlet;
gui_param.selected_channel_name = handles.selected_channel_name;
gui_param.nominal_width_index = handles.nominal_width_index;

%%%
[~,pres_temp_file_name,~] = fileparts(handles.pres_temp_path_and_name);

default_save_full_path = ...
    strcat(handles.default_file_path, pres_temp_file_name, ...
    '_chip ', handles.selected_channel_name, ...
    '_inlet ', handles.inlet, ...
    ' - Settings');
[save_file_name,save_file_path,save_file_ext] = ...
    uiputfile('*.mat', 'Please choose file path and name to save settings to.',default_save_full_path);

save([save_file_path save_file_name '.mat'], 'gui_param');
%%%

% myMatFullFileName = fullfile(handles.default_file_path, handles.pres_temp_path_and_name);
% save(myMatFullFileName, handles);
% uiputfile

function pushbutton_plot_analyzed_file_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_plot_analyzed_file.
% hObject    handle to pushbutton_plot_analyzed_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global input_pressures
global analyzed_data
global min_app_visc_error
global max_app_visc_error

[load_analyzed_file_name,load_settings_path,~] = uigetfile(strcat(handles.default_file_path, '*.mat'), ...
    'Please select the analyzed file to use (*.mat)');

%%% Inputting path and file name into handles structure %%%
if load_analyzed_file_name ~= 0
    full_load_analyzed_path = strcat(load_settings_path, load_analyzed_file_name);
    analyzed_data_file = load(full_load_analyzed_path);

    input_pressures = analyzed_data_file.input_pressures;
    analyzed_data = analyzed_data_file.analyzed_data;
    min_app_visc_error = analyzed_data_file.min_app_visc_error;
    max_app_visc_error = analyzed_data_file.max_app_visc_error;

    handles.default_file_path = analyzed_data_file.handles.default_file_path;
    handles.input_pres_path_and_name = analyzed_data_file.handles.input_pres_path_and_name;
    handles.flow_rate_path_and_name = analyzed_data_file.handles.flow_rate_path_and_name;
    handles.pres_temp_path_and_name = analyzed_data_file.handles.pres_temp_path_and_name;

    handles.value_radiobutton_diw = analyzed_data_file.handles.value_radiobutton_diw;
    handles.value_radiobutton_diw_and_gly = analyzed_data_file.handles.value_radiobutton_diw_and_gly;
    if isfield(handles, 'value_radiobutton_power_law')
        handles.value_radiobutton_power_law = analyzed_data_file.handles.value_radiobutton_power_law;
    end
    handles.fluid_visc = analyzed_data_file.handles.fluid_visc;
    handles.value_radiobutton_diw_and_gly = analyzed_data_file.handles.value_radiobutton_diw_and_gly;
    if isfield(handles, 'value_radiobutton_power_law')
        handles.value_radiobutton_power_law = analyzed_data_file.handles.value_radiobutton_power_law;
    end
    handles.fluid_visc = analyzed_data_file.handles.fluid_visc;
    handles.fluid_visc_temp = analyzed_data_file.handles.fluid_visc_temp;
    handles.open_channels = analyzed_data_file.handles.open_channels;
    handles.scale_factor = analyzed_data_file.handles.scale_factor;
    handles.popup_sel_inlet_index = analyzed_data_file.handles.popup_sel_inlet_index;
    handles.popup_sel_channel_name_index = analyzed_data_file.handles.popup_sel_channel_name_index;

    update_gui(hObject, eventdata, handles);
    plot_analyzed_data(hObject, eventdata, handles);
    set(handles.text_status, 'String', 'Program has finished plotting the already analyzed file.');
else
    set(handles.text_status, 'String', 'Program could not open the file.');
end

function pushbutton_plot_analyzed_file_Callback(hObject, eventdata, handles)
% Executes on button press in pushbutton_plot_analyzed_file.
% hObject    handle to pushbutton_plot_analyzed_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[load_analyzed_file_name,load_settings_path,~] = uigetfile(strcat(handles.default_file_path, '*.mat'), ...
    'Please select the analyzed file to use (*.mat)');

%%% Inputting path and file name into handles structure %%%
if load_analyzed_file_name ~= 0
    full_load_analyzed_path = strcat(load_settings_path, load_analyzed_file_name);
    analyzed_data_file = load(full_load_analyzed_path);

    set(handles.text_current_directory_path, 'String', analyzed_data_file.handles.default_file_path);
    set(handles.edit_input_pres_path, 'String', analyzed_data_file.handles.input_pres_path_and_name);
    set(handles.edit_flow_rate_path, 'String', analyzed_data_file.handles.flow_rate_path_and_name);
    set(handles.edit_pres_temp_path, 'String', analyzed_data_file.handles.pres_temp_path_and_name);

    set(handles.radiobutton_diw,'Value',analyzed_data_file.handles.value_radiobutton_diw);
    set(handles.radiobutton_diw_and_gly,'Value',analyzed_data_file.handles.value_radiobutton_diw_and_gly);
    if exist('handles.value_radiobutton_power_law')
    set(handles.radiobutton_power_law,'Value',analyzed_data_file.handles.value_radiobutton_power_law);
    end
    set(handles.edit_fluid_visc, 'String', analyzed_data_file.handles.fluid_visc);
    set(handles.edit_fluid_visc_temp, 'String', analyzed_data_file.handles.fluid_visc_temp);
    set(handles.edit_open_channels, 'String', analyzed_data_file.handles.open_channels);
    set(handles.edit_scale_factor, 'String', analyzed_data_file.handles.scale_factor);

    set(handles.popupmenu_inlet, 'Value', analyzed_data_file.handles.popup_sel_inlet_index)
    set(handles.popupmenu_channel_name, 'Value', analyzed_data_file.handles.popup_sel_channel_name_index);
end

function popupmenu_inlet_Callback(hObject, eventdata, handles)
% Executes on selection change in popupmenu_inlet.
% hObject    handle to popupmenu_inlet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.popup_sel_inlet_index = get(handles.popupmenu_inlet, 'Value');

switch handles.popup_sel_inlet_index
    case 1
        handles.inlet = 'A';
    case 2
        handles.inlet = 'B';
    case 3
        handles.inlet = 'C';
    case 4
        handles.inlet = 'D';
end
guidata(hObject, handles);

function popupmenu_channel_name_Callback(hObject, eventdata, handles)
% Executes on selection change in popupmenu_channel_name.
% hObject    handle to popupmenu_channel_name (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.popup_sel_channel_name_index = get(handles.popupmenu_channel_name, 'Value');

switch handles.popup_sel_channel_name_index
    case 1
        handles.selected_channel_name = 'A - 20 um';
        handles.nominal_width_index = 1;
    case 2
        handles.selected_channel_name = 'B - 40 um';
        handles.nominal_width_index = 2;
    case 3
        handles.selected_channel_name = 'C - 100 um';
        handles.nominal_width_index = 4;
    case 4
        handles.selected_channel_name = 'D - 60 um';
        handles.nominal_width_index = 3;
    case 5
        handles.selected_channel_name = 'E - 20 um';
        handles.nominal_width_index = 1;
    case 6
        handles.selected_channel_name = 'F - 20 um';
        handles.nominal_width_index = 1;
    case 7
        handles.selected_channel_name = 'G - 100 um';
        handles.nominal_width_index = 4;
    case 8
        handles.selected_channel_name = 'H - 40 um';
        handles.nominal_width_index = 2;
    case 9
        handles.selected_channel_name = 'I - 60 um';
        handles.nominal_width_index = 3;
end
guidata(hObject, handles);

function edit_open_channels_Callback(hObject, eventdata, handles)
% hObject    handle to edit_open_channels (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%%TODO check input value
user_input_open_channels = get(handles.edit_open_channels, 'Value')
%handles.open_channels = user_input_open_channels;
guidata(hObject, handles);

function edit_scale_factor_Callback(hObject, eventdata, handles)
% hObject    handle to edit_scale_factor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit_fluid_visc_temp_Callback(hObject, eventdata, handles)
% hObject    handle to edit_fluid_visc_temp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit_fluid_visc_Callback(hObject, eventdata, handles)
% hObject    handle to edit_fluid_visc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit_input_pres_path_Callback(hObject, eventdata, handles)
% hObject    handle to edit_input_pres_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
function edit_flow_rate_path_Callback(hObject, eventdata, handles)
% hObject    handle to edit_flow_rate_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit_pres_temp_path_Callback(hObject, eventdata, handles)
% hObject    handle to edit_pres_temp_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit_K_constant_Callback(hObject, eventdata, handles)
% hObject    handle to edit_K_constant (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit_n_constant_Callback(hObject, eventdata, handles)
% hObject    handle to edit_n_constant (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%%%%% ----- Create Functions ----- %%%%%
function popupmenu_inlet_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu_inlet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', {'A', 'B', 'C', 'D'});

function popupmenu_channel_name_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu_channel_name (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', { ...
    'A - 20 um', ...
    'B - 40 um', ...
    'C - 100 um', ...
    'D - 60 um', ...
    'E - 20 um', ...
    'F - 20 um', ...
    'G - 100 um', ...
    'H - 40 um', ...
    'I - 60 um'});

function edit_open_channels_CreateFcn(hObject, eventdata, handles)
% Executes during object creation, after setting all properties.
% hObject    handle to edit_open_channels (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_scale_factor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_scale_factor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_fluid_visc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_fluid_visc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_fluid_visc_temp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_fluid_visc_temp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_input_pres_path_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_input_pres_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_flow_rate_path_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_flow_rate_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_pres_temp_path_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_pres_temp_path (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
function edit_K_constant_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_K_constant (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function edit_n_constant_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_n_constant (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%%% Unsorted functions
```