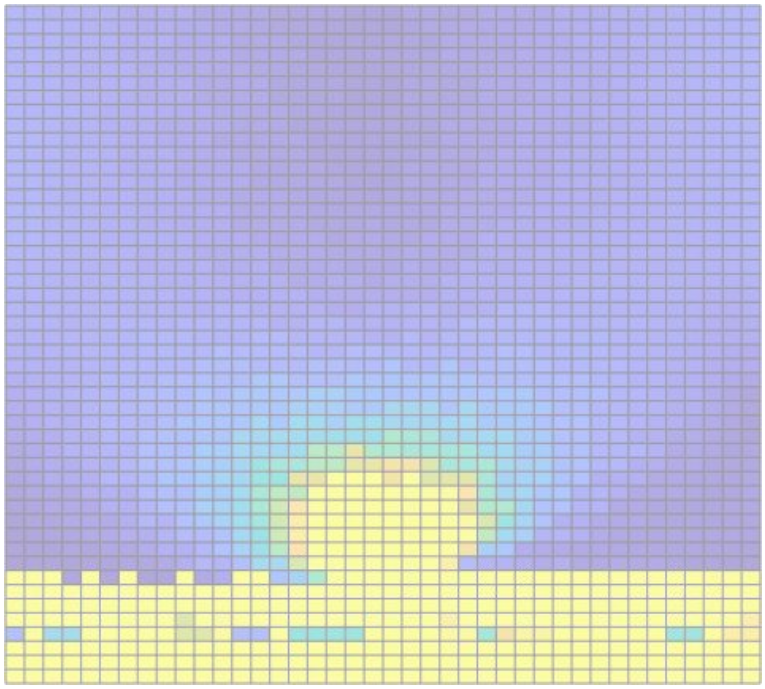
Master Thesis

RWTH AACHEN UNIVERSITY

Lothar Brixius

Fluid Mechanical Aspects of Hemostasis



LEHRSTUHL FÜR MEDIZINISCHE INFORMATIONSTECHNIK
Fakultät für Elektrotechnik und Informationstechnik, RWTH Aachen
Univ.-Prof. Dr.-Ing. Dr. med. Dr. h.c. Steffen Leonhardt
Supervisor: Univ.-Prof. Dr.-Ing. Dr. med. Dr. h.c. Steffen Leonhardt

Date: 19.10.22

mediT

Acknowledgements

I want to thank my research-supervisors Martin and Dario for their support and scientific input. Also, I want to thank Prof. Leonhardt and Dr. Müller for making this thesis possible. I'm grateful to Dr. Schmidt and Prof. Müller for the provided scripts and the answering of questions related to them. Additional gratitude goes to Mr. van den Heuvel, Mr. Sahin, Prof. Hayden and Mr. Grab for the assistance with the microfluidic model. I'm in debt to Ms. Sailer for her help with "DefocusTracker". For their assistance with specific questions, I must thank Mrs. Öri (Ibidi), Dr. Zbarsky (COMSOL) and Dr. von Rücker. A big thank you is also due to my other colleagues in Munich, with whom I had such a great time!

Personally, I'm in debt to my friends in Ratingen, Aachen and Munich. Without their friendship, the last years would have been quite exhausting. Last but not least, I want to thank the countless doctors and surgeons who helped me the last two decades. Especially Dr. Sabbagh and Dr. Rüggeberg have to be mentioned.

"A long journey comes to an end. The prize are the memories we made on the way."

Munich 08.10.22

Lothar Brixius

Eidesstattliche Versicherung

Name, Vorname

Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum

Unterschrift

Abstract

Investigating the concept of hemostasis may result in new insights into preventing strokes and ensuring proper wound closure of blood vessels after hemorrhages. Platelets are the "anucleus cells" mainly involved in preventing blood loss and in maintaining the blood vessel's integrity and therefore a key factor. Upon injury they form an "emergency plug" named thrombus, to cut off the flow of blood to the outside tissue of the vessel. However, under pathological conditions, unregulated platelet activation can induce intravascular thrombosis, ultimately leading to vascular damages, inflammation, but also to ischemia and stroke, which are a leading cause of death worldwide.

When treating a stroke patient, there is a trade-off to be made between dissolving the unwanted thrombus with specific medication and keeping the protection mechanism of the platelets intact, in order to avert uncontrolled blood loss. Therefore, optimizing anti-thrombotic therapies require good models of that process and insight into hemostasis. From a medical perspective, mice or rats can be used as models for human hemostasis. However, measuring fluid mechanical parameters (e.g., shear rates) in such models is very difficult, but at the same time these parameters, are believed to be of vital importance/play a key role for triggering the platelet response. Another aspect is that any vessel of an animal is different in terms of tissue composition and geometry. The experiments require the animal to be killed as well, making large scale testing and the reproducibility of such tests difficult.

This thesis, therefore, aims to extract the physical variables present during hemostasis, e.g., by performing a computer simulation, and correlate them to a given animal experiment. In the beginning, the theory of fluid mechanics and hemostasis is briefly laid out. Afterwards, a mathematical analysis of blood as a fluid is performed. The goal of this is to get knowledge on how to set up the next step: a computer simulation of a blood vessel with a wound. The results of the computer simulation can be used to lay out the design of a corresponding microfluidic chip. The latter aims at overcoming difficulties in standardizing animal experiments. As a last step, a way to map together data from animal experiments and the computer simulation or microfluidic model is presented. It will be discussed how, in the end, a full biomechanical simulation of the closure of a wound in a blood vessel could be achieved by splitting the problem up into its fluid-mechanical and biomedical parts.

Because this master thesis stands very much at the beginning of investigating the task described above, many aspects are for now rather proof-of-concepts than fully developed models and methods. Still, they will provide a good stepping-stone for further development. In detail, the thesis has four aspects: 1) fluid mechanical and mathematical modelling of blood, 2) a computer simulation, numerically calculating fluid-mechanical properties of a given vessel and wound set-up; in particular COMSOL will be used for that, 3) designing and manufacturing a microfluidic chip for standardized vessel and wound sizes and large-scale experiments,

4) bringing the results of these experiments together in a special data structure implemented in MATLAB, which in the future could be used to perform a complete virtual simulation of the process of thrombus formation.

This thesis is a cooperation between various scientific fields, mostly fluid mechanics and medicine. Hence, it becomes clear that knowledge from various experts was required. For example, animal experiments were performed by Mr. Rossaro, medical department of the LMU, and used as a reference for validation and discussion of the presented models. Concepts taken from colleagues are marked as such. For details, please refer to the "Acknowledgments" section and the corresponding chapters of this thesis.

Zusammenfassung

Untersuchungen zum Thema Hämostase (englisch: hemostasis) können Erkenntnisse zur Prävention von Schlaganfällen oder zur Unterstützung des Wundheilungsprozesses liefern. Sogenannte Thrombozyten (englisch: platelets) sind der Schlüssel, um einen weiteren Blutverlust nach einer kleinen Verletzung an einem Blutgefäß zu verhindern. Sie formen einen „Notfallstöpsel“, einen Thrombus, um einen weiteren Blutfluss in die Außenwelt zu stoppen. Unglücklicherweise ist der gleiche Mechanismus für die Entstehung von Schlaganfällen verantwortlich. Vereinfacht gesprochen bildet sich fälschlicherweise ein Thrombus in einem eigentlich intakten Gefäß und verstopft dies. Schlaganfälle sind eine der Haupttodesursachen in der Welt. Die Behandlung eines Schlaganfallpatienten erzwingt deswegen einen Zielkonflikt zwischen der schnellen Auflösung des ungewollten Thrombus auf der einen und dem Erhalt der Thrombozytenfunktionalität auf der anderen Seite. Insbesondere wenn zusätzlich chirurgische Eingriffe am Herzen oder anderen Gefäßen notwendig sind, wird es wichtig, einen unkontrollierten Blutverlust zu vermeiden. Die richtige Dosierung anti-thrombotischer Medikamente erfordert daher akkurate Modelle dieses Prozesses und weitreichende Einsichten in das Thema Hämostase.

Aus einer medizinischen Perspektive können Mäuse und Ratten als Modelle für den Ablauf der Hämostase im Menschen verwendet werden. Jedoch haben Tierversuche einige Nachteile. Zum Beispiel kann es schwer oder sogar unmöglich sein, gleichzeitig die Thronbusbildung und die dabei auftretenden fluidmechanischen Größen aufzuzeichnen. Dies wird später genau erläutert. Allerdings wird vermutet, dass genau diese Größen entscheidend für das Verhalten der Thrombozyten sind. Weiterhin sind Tierversuche schwer zu skalieren oder zu reproduzieren, da jedes Tier und jedes Blutgefäß leicht unterschiedlich sind. Hinzu kommen ethische Überlegungen, da die Tiere bei oder nach den Versuchen getötet werden.

Diese Arbeit will daher Methoden präsentieren, mit denen die physikalischen Parameter solcher Experimente ergänzt werden können. Als Ausgangspunkt dient eine kurze Einleitung in die theoretischen Hintergründe zur Fluidmechanik und zu relevanten biologischen Aspekten. Im Anschluss wird eine Möglichkeit diskutiert, Blut als Flüssigkeit mathematisch korrekt zu beschreiben. Dies soll die Grundlage für den nächsten Schritt bilden: ein Wundmodell, einmal als Computersimulation und einmal als realer mikrofluidischer Kanal. Letzteres könnte eine Perspektive sein, um eine Standardisierung und Skalierung der Tierversuche durchzuführen. Zum Schluss wird eine Möglichkeit vorgestellt, die Ergebnisse aus den oben beschriebenen Experimenten in einer speziellen Datenstruktur zusammenzubringen und so eine Grundlage für eine mögliche biomechanische Simulation des gesamten Hämostase-Prozesses zu bilden. Da diese Arbeit nur den Auftakt zur Bearbeitung dieser Problemstellung bildet, wurde der Fokus auf das Ausarbeiten von Konzepten gelegt und nicht darauf, jeden einzelnen Aspekt bis zum Ende durchzuexerzieren. Diese Arbeit soll einen soliden Grundstein für eine weitere Untersuchung der Problemstellung dieser Thesis bilden.

Nochmal im Detail sind Bestandteile dieser Arbeit: 1) Die fluidmechanische und mathematische Beschreibung von Blut, 2) ein Computersimulationsmodell, welches numerische Werte für z.B. die Flussgeschwindigkeit eines gegebenen Gefäß-Wund-Aufbaus liefert; hierfür wird COMSOL genutzt werden, 3) Entwurf und Herstellung von Mikrofluidikchips, die ein Standardmodell für Wund-Gefäß-Anordnungen darstellen und 4) Einführung eines speziellen Datentyps, implementiert in MATLAB, der eine Synthese der verschiedenen Experimentaldaten und eine Grundlage für eine biomechanische Simulation des gesamten Hämostase-Prozesses ermöglicht.

In dieser Arbeit kommen verschiedene Teilaspekte diverser wissenschaftlicher Disziplinen zusammen. Deswegen war es notwendig, die Expertise verschiedener Kolleginnen und Kollegen einzuholen. So wurden zum Beispiel die Tierversuche zum Vergleich und zur Validierung der Modelle von Herrn Rossaro durchgeführt. Weitere Hilfestellungen durch Dritte finden sich in den „Acknowledgements“ und den jeweiligen Kapiteln.

Contents

List of Symbols and Abbreviations

Abbreviations

AT3	Antithrombin III
AV	Artifical vessel
BM	Basement membrane
C	Compliance of a vessel
CF	Continuum flow
EC	Endothelial cell
ECM	Extracellular matrix
EEL	External elastic lamina
FE	Fluid element
FLE	Fahraeus-Lindqvist effect
IEL	Internal elastic lamina
LMU	Ludwig Maximilian University of Munich
PC	Pulmanory Circulation
PDMS	Polydimethylsiloxane
PF3	Platelet factor 3
PR	Photo resistive
PVAT	Perivascular adipose tissue
RBC	Red blood cell
RWTH	Rheinisch-Westfälische Technische Hochschule Aachen
SC	Systematic Circulation
SF	Slip flow
TFP	Tissue factor protein
tPA	Tissue plasminogen activator
TUM	Technical University of Munich
VSM	Vascular smooth muscle cells
vWBF	von Willebrand factor, sometimes also abbreviated as vWF

Physical Quantities

A	Cross-section, (2D-)Area	m^2
β	Deformation angle	$^\circ$
c_{pulse}	Pulse wave velocity	$\frac{m}{s}$
d	Distance	m
d_{hydr}	Hydraulic diameter	m
δp	Pressure loss	Pa
F	Force	N
K_{cc}	Consistency coefficient	$\frac{kg \cdot s^{n-2}}{m}$
k_s	Average surface roughness	μm
Kn	Knudsen number	$-$
l	Length	m
l_{chr}	Characteristic dimension	m
λ_{mfp}	Mean free path	m
λ_{pfc}	Pipe friction coefficient	$\frac{kg}{m^3}$
P_{tm}	Transmural pressure	Pa
ϕ	Shear rate	$\frac{1}{s}$
ρ	Density	$\frac{kg}{m^3}$
σ_{tw}	Tangential wall stress	$\frac{N}{m^2}$
t	Time	s
t_0	Temporal starting point	s
T	Temperature	$^\circ C$
τ	Shear stress	Pa
τ_0	Yield stress	Pa
$\bar{\bar{\tau}}$	Stress tensor	Pa
u	Velocity	$\frac{m}{s}$
U	Absolute velocity	$\frac{m}{s}$
$\tilde{u}$	Velocity vector	$\frac{m}{s}$
μ	Dynamic viscosity	$Pa \cdot s$
v	Viscosity	$Pa \cdot s$
$\tilde{x}$	Spatial position	m
$\tilde{x}_0$	Spatial starting position	m

Mathematical Quantities

n_{Power}	Power law index
n_{HB}	Herschel-Bulkey parameter
Φ	Arbitrary fluid property
$\tilde{v}, \mathbf{v}$	Vector v, bold letters denote vectors in COMSOL related equations
$\otimes$	Outer product of two vectors
$\|\| \cdot \|\|_2, \|\| \cdot \|\|_{euclidean}$	Euclidean norm
$\|\| \cdot \|\|$	Generic norm. If not specified, the Euclidean norm

1 Introduction

In this chapter, the structure of this thesis will be explained in more detail. From an organizational point-of-view, this work is a cooperation between medicine and engineering departments. Involved were the Innere Medizin (internal medicine) of the Ludwig-Maximilians-University of Munich (LMU) and the Heinz-Nixdorf-Chair of the Electrical Engineering Department of the Technical University of Munich (TUM). Written as an external thesis, it was submitted to the Chair for Medical Information Technology of the RWTH University of Aachen.

Figure 1.1 helps to explain how the different parts of this work are connected to each other. While the abstract explained the motivation for researching the fluid-mechanical aspects of hemostasis, here the interaction of the various sub-disciplines will be shown. Depicted in Figure 1.1 are the actual parts of the thesis (light blue) which use animal experiments as a reference and mean of validation (green). Yellow denotes possible real-world applications with a chance for commercialization. At this stage, commercialization is merely an idea, meant to hint at further potential research.

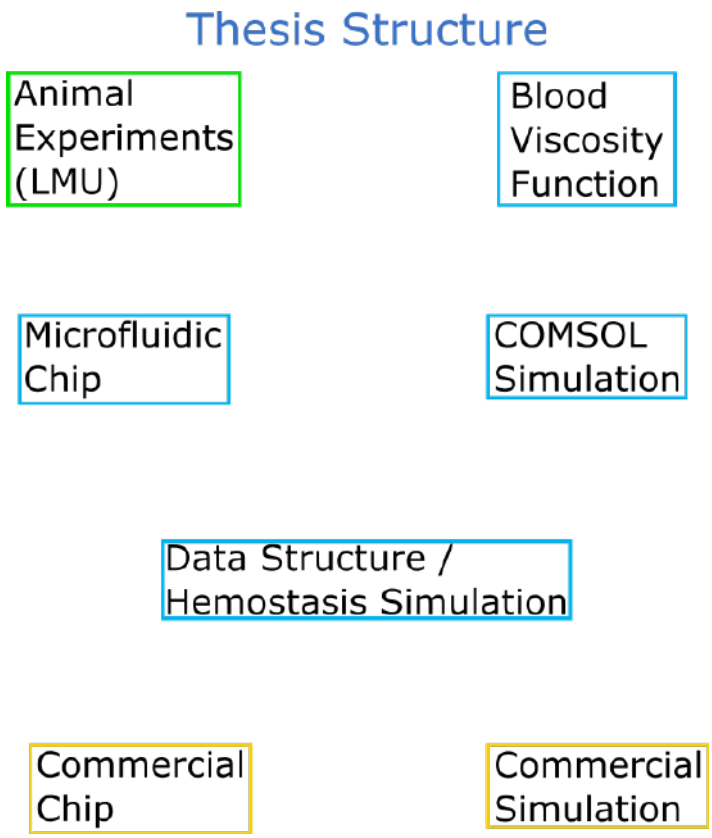


Figure 1.1: General structure of the thesis. The relationship between the various fields can be seen in the figures 1.2 to 1.6.

All animal experiments were conducted by Mr. Rossaro from the LMU. An explanation of the procedure is provided in Chapter 4.1. For now, it is sufficient to say that after anaesthetizing, the guts of mice were extracted, then individual blood vessels loosened from the wall of the gut and punctured with a needle. Using a microscope and a camera, the process of thrombus formation can be recorded. Images and videos are the resulting kind of data. An example can be seen in Figure 1.2, while Figure 1.3 conveys the notion of medical validation of models through these experiments.

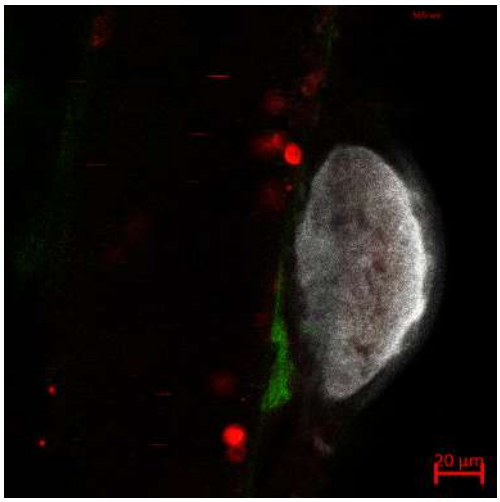


Figure 1.2: Thrombus formation in a blood vessel of a mouse, after a needle injury. White denotes the formed thrombus. Images taken by Mr. Rossaro.

Every model that is later constructed, aims at replicating these experiments (in parts) to make quantities measurable that stay hidden otherwise. Most important is the depiction of the shear rate present during the thrombus formation. Chapter 2.1.3 explains the meaning of the term shear rate, but as of right now it is sufficient to say it can be interpreted as a change of velocity over a certain distance inside a fluid. Two main concepts are introduced for this task: a computer simulation of a blood vessel with a wound (Chapter 4) and a microfluidic channel depicting the same set-up (Chapter 5). Digital and real-world replica can be used to cross-check conclusions. Further, they provide numerical values of fluid variables like shear rate, pressure and velocity as an input for a data structure that maps them back to the animal experiments (Chapter 6). The same structure can form a basis for a biomechanical simulation of hemostasis, as will be also shown in Chapter 6.

It is of upmost importance to depict the fluid-mechanical properties of blood accurately in these models. For this, thorough literature research was performed and will be presented in Chapter 3 together with a way to derive analytical expressions for modelling blood viscosity. Because blood is a non-Newtonian fluid, describing its viscosity is difficult but again highly relevant. Non-Newtonian fluids get introduced in Chapter 2.1.3. Knowledge won about blood viscosity can be used to feed mathematical expressions and numerical values into computer simulations and, e.g., for mixing a test liquid for microfluidic experiments. These connections between the corresponding topics are depicted in Figure 1.4.

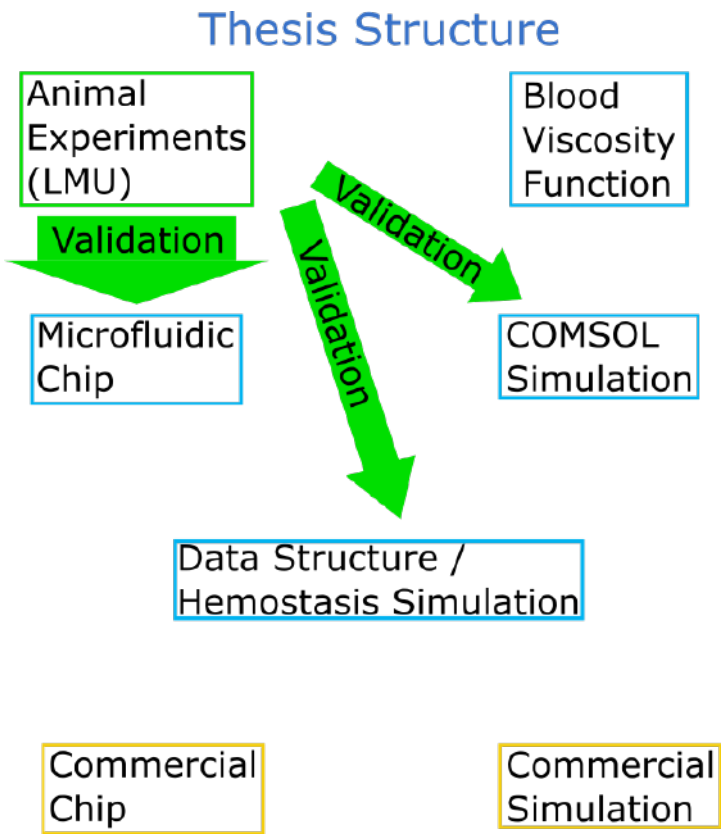


Figure 1.3: Animal experiments make up the backbone of all presented models and are used as a mean of validating their quality.

The presented research is of practical value as well. Two scenarios come to mind: First, using a standardized microfluidic chip, producible in large quantities for drug research or patient diagnosis. Animal experiments are not without issues, as was already explained in the abstract. They require a long turn-over cycle of breeding the required animals with higher maintenance costs and legal regulations compared to store some silicon or PDMS chips in a warehouse. Every animal is a little different in its biological properties, which can potentially influence the experiment. Take for example genetic mutations or vessel diameter. Even more so, as each wound induced by a needle or a laser will have a slightly different diameter.

Ex Vivo chips offer large scale, standardized testing at presumably lower costs. In addition, without chips, each experiment requires the animals to be killed and financially efficient raising of test subjects most likely means a loss of life quality for the mammals until a certain legal minimum is met. For testing the drug dosage of an already hospitalized patient, in vivo experiments might even be impossible or medically dangerous. Chips operated with a small blood sample of the patient might provide a feasible alternative. Effects of drugs that decrease the activity of platelet cells or other key components of hemostasis could be investigated with this method and a minimum functionality of thrombus formation insured.

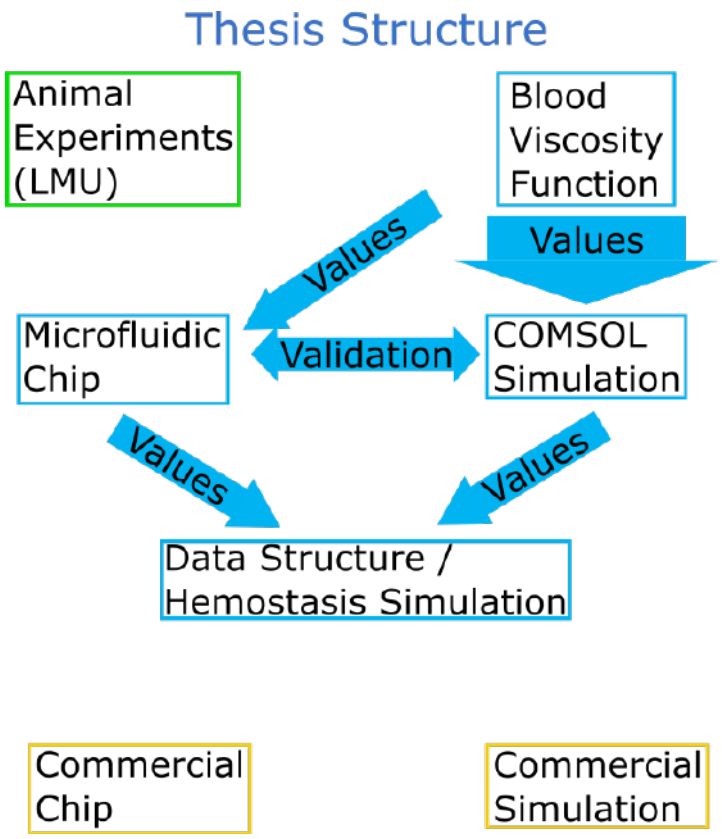


Figure 1.4: Relation between the four sub-topics of this master thesis.

Although it seems far from possible right now to replace clinical studies with computer simulations, a commercial biomedical simulation of the thrombus formation process could allow for rapid, low-cost drug prototyping. If the effects on certain participants in the hemostasis reaction chain are known or can be approximated, the resulting implications for sufficient wound healing can be visualized. For example, let's say it is known that a new painkiller will decrease the activity of platelet cells by 20 %. Now the corresponding parameter in the simulation is changed accordingly. The question now is, if a hole that can occur in a vessel during a cardiac surgery is closed fast enough to ensure only a small amount of blood loss. Simulation results could show if it makes sense to perform further, more expensive, clinical research on this drug, or if the drug interferes too much with thrombus formation. Figure 1.5 depicts this last train of thought, the application and commercialization of the presented research.

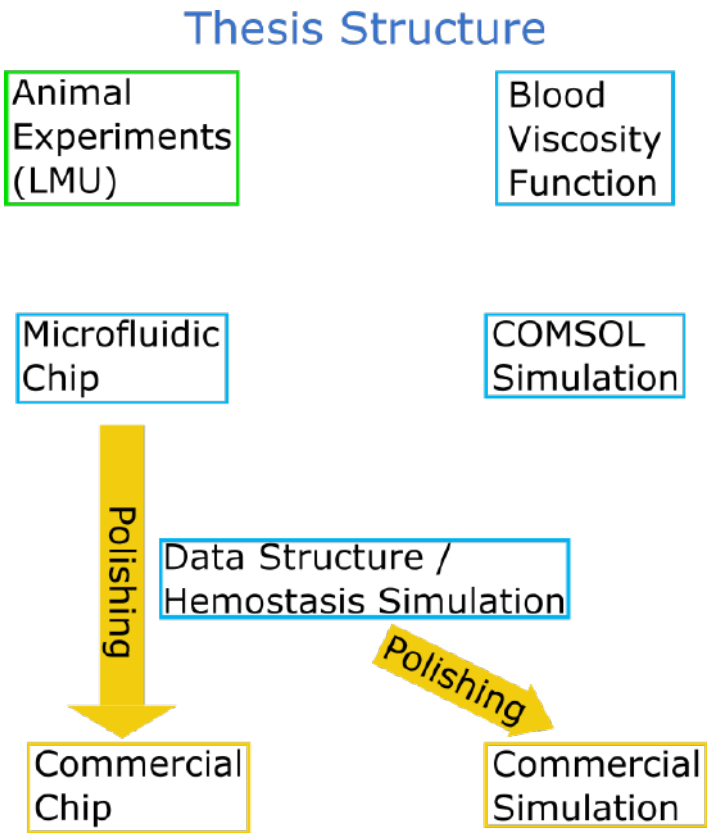


Figure 1.5: Possibilities of commercializing the research topics of this thesis.

2 Theory

This thesis will be governed by two main aspects. One is the medical/biological field of hemostasis with a focus on thrombus formation. The other is fluid mechanics. In this chapter, a brief introduction to both will be given, presenting the theoretical background to model the process of a blood vessel being injured and a consequential closure of this hole by a thrombus.

As of now, it is commonly accepted that the **shear rate** is one of the driving causes for platelets to change their behaviour, which leads to thrombus formation [NWa09] [FF05]. Since measuring these alterations in vivo is not possible, a fluid mechanical simulation of a blood vessel before, during and after a little breach occurs in the wall, must be developed. To set up the simulation properly, especially the boundary problems, knowledge about on the physical laws and the resulting differential equations has to be gathered. Chapter 2.1 will be the basis of modelling hemostasis from a fluid mechanical perspective.

The biological basis for that are the topics of blood circulation, blood composition, blood vessel structure and finally hemostasis itself. All of this can be found in Chapter 2.2. Restricting a simulation to only provide the physical parameters present before, during and after the inflicted injury occurs, does not mean biological aspects can be ignored. They determine the fluid mechanical behaviour of blood and vessel. Ultimately, humans are the animal of interest, so everything will be discussed on the basis of human physiology.

2.1 Fluid Mechanics

The discussion of fluid mechanics in this thesis closely follows the script for the lecture "Fluid Mechanics 1" from Professor Adams of the Chair for Aerodynamics and Fluid Mechanics at the Technical University of Munich [Ada18].

2.1.1 Fluid Elements

Fluid mechanics in general can be applied to both liquids and gases. However, only liquids are of interest in this thesis, more specific the properties of blood. Therefore, the terms fluid and liquid are used interchangeable here. To start off, a **fluid element** (FE) of an arbitrary small volume dV is introduced. Inside this volume, a homogeneous density ρ is assumed. On average, all the molecules inside dV move into direction $\vec{u}$ in an again arbitrary small time-interval dt. Fluid molecules mostly interact with each other because of electromagnetic forces that result from the nuclei and shell electrons [Ada18]. In Figure 2.1 a schematic of a FE can be seen.

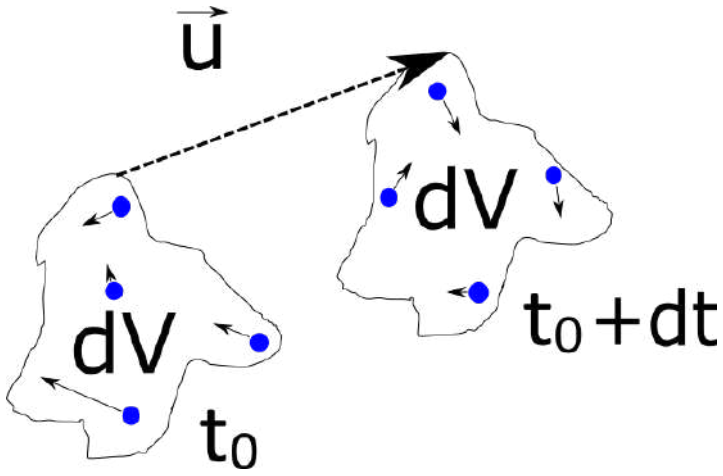


Figure 2.1: Schematic of the concept of an FE, idea taken from [Ada18].

An important quantity in fluid mechanics is the **mean free path** λ_{mfp}. It expresses how far a molecule can travel inside a fluid before hitting another molecule. Comparing this variable to the **characteristic dimension** l_{chr} of a fluidic set-up, e.g., the diameter of a pipe under observation, lead to the **Knudsen number** Kn:

$$Kn = \frac{\lambda_{mfp}}{l_{chr}} \tag{2.1}$$

No literature values for the mean free path of water or blood could be found in literature. This might be due to the fact that the mean free path is usually only considered for gases. Liquids should be so dense that molecules barely have space to move. The mean free path of water can be used as an approximation for blood plasma, which consists mostly of water. The blood vessels and the microfluidic chambers under consideration have a diameter of around $l_{chr} = 100-250$ $[um]$. As a result, Kn could not be determined. But it is reasonable to say that it might be smaller than 0.01, arguing that a water molecule or a RBC cannot move for a longer distance than 1 $[um]$ without hitting another particle. According to [Ada18] in case of Kn < 0.01, the mathematical simplifications for a **continuum flow** (CF) are valid. For $0.01 < Kn < 0.1$ one has to consider more complicated formulas valid for **slip flow** (SF).

The characteristic length is the result of the minimal vessel size to be considered in this thesis, decided by Mr. Rossaro, medical department LMU, and the hole size after applying an injury to a vessel during the animal experiments. Even if a 30-gauge needle with a 300 $[um]$ diameter is used, the injury will have a diameter of $100 - 200$ $[um]$, as can be seen in figure 1 of [TMPFa18].

2.1.2 Continuum Principle

It is important to notice that the following descriptions are only valid for a CF. Especially the use of continuous field quantities when discussing Chapter 2.1.4 is not correct for SFs [Ada18].

A fluid is approximated by filling its entire volume with an infinite number of FEs, which in return take up an invitively small amount of space. No space is left in between two FEs. The physical quantities of the flow will be described by every FEs individual quantities. These quantities are position $\vec{x}_i$, velocity $\vec{u}_i$, pressure p_i, temperature T_i and density ρ_i. This idea of filling a fluid with individual FEs is called continuum principle [Ada18]. Figure 2.2 depicts this concept.

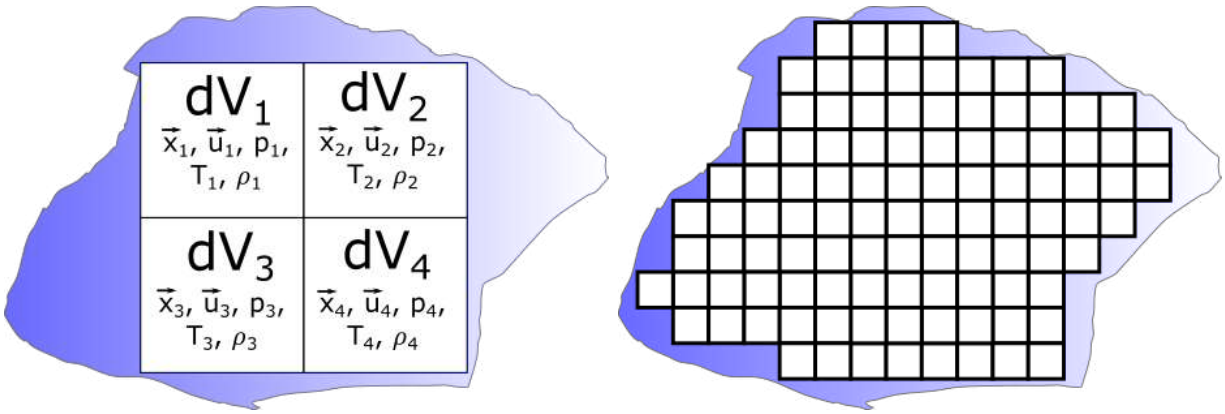


Figure 2.2: Visualization of the Continuum Principle. On the left is illustrated that each FE has its own properties, which in total make up the overall properties of the liquid. On the right, it can see that choosing the FEs small enough can lead to any desired accuracy when approximating the volume of the fluid.

It might seem intuitive that one can get as close as desired to a "casual" fluidic volume, by just taking many small FEs. But it can be even mathematically proven! The volume of the cubes will converge to zero if the amount of FEs goes to infinity, but keep in mind that FEs were defined to have a volume and thereby a size as close to zero as desired in that situation. No overlap between cubes is required for approximating the fluid.

A "casual" volume in this context means a volume that can be approximated by a mathematically open area A. An open area is a subset of $\mathbb{R}^3$ so that for each point $x \in A$ a ball with a radius $r > 0$ can be found that still is fully covered by $A(B_{(r(x))} \subseteq A)$. r may depend on the point x under consideration (Def. 7.14 [Mül20a]). $d(x, a)$ from Def. 7.14 (a) [Mül20a] is the Euclidean distance function, which is a mathematical formulation of what in everyday life is understood when talking about the distance of two points in space. Each fluid that is not some ionized plasma or a similarly complicated phenomenon will have a sharp border to the outside world and a finite expansion.

This means that aside from the hull, for each point inside the volume, a ball small enough to still be fully inside the fluid can always be found. While this is clear from a naive concept of space, where everything can be made as small as desired, mathematically this is a result of the uncountable infinity of $\mathbb{R}^3$ and the properties of the Euclidean norm $||\cdot||_{euclidean} \hat{=} d(x,a)$ from above.

The physical hull, mathematically corresponds to a boundary B of A (Def. 7.21 (a) [Mül20a]). Taking this boundary away, turns $A \setminus B$ into an open area. Since the boundary can be picked as thin as desired, one always neglects only an arbitrarily small part of the volume. The same applies for the number of molecules neglected. Since no quantity of interest $(\vec{x}, \vec{u}, p, T, \rho)$ is surface dependent but rather a volumetric property, the neglection of B is justified. Mathematically speaking, neglecting the boundary of A, which is a Lebesgue-Null-Set with regard to the three-dimensional Lebesgue-Measure λ^3 (Def. 11.34 [Mül20b], in combination with Sentence 11.37 [Mül20b]). The boundary, thereby, will not influence any volume-integrals. With this said, Theorem 2.1 can be formulated, which backs up the idea of the continuum principle.

Theorem 2.1: Every volume of a fluid $V \subseteq \mathbb{R}^3$ that can approximated arbitrarily well with an open area $A \subseteq \mathbb{R}^3$ is a countably infinite disjunct unification of cubic FEs.

This follows directly from Lemma 11.16 in [Mül20b]. In the notation of the proof provided in [Mül20b] the FEs are the $Q_k^{(n)}$. Each cube has a volume of $\left(\frac{1}{2^n}\right)^3 = \frac{1}{2^{3n}}$. For $n \longrightarrow \infty$ the volume converges to zero, as mentioned above. U_n is the set of all cubes that fill up A and thereby V. The idea of this proof is the concept depicted in Figure 2.2. For details, please refer to [Mül20b].

The concept of fluid elements is used for deriving the Kinematics of Flow (Chapter 2.1.4). It further gives a natural introduction to the concept of discretization of liquids for numerical simulations, Chapter 2.1.4 and 2.1.6.

2.1.3 Viscosity

Viscosity is an important concept in fluid mechanics. Analogous to friction, it can be derived in the following way: Moving a plate with the surface area A over the surface of a fluid with an absolute velocity U requires the force $F = \tau A$. It is assumed that the entire geometric area A is in contact with the fluid. Notice that A has nothing to do with the mathematical area A used in the chapter above. τ is called the **shear stress** and F is the force required to overcome the viscosity of the fluid.

If one now further assumes the fluid to be placed on a floor with distance d to the plate that is moving on the liquid, one gets a set-up like the one depicted in Figure 2.3. β denotes the angle between the vector of the plate's starting point

and its current position, called the deformation angle. In the schematic below, it is assumed that the plate started moving from position $\vec{x}_0 = \begin{pmatrix} 0 \\ d \end{pmatrix}$. In general,

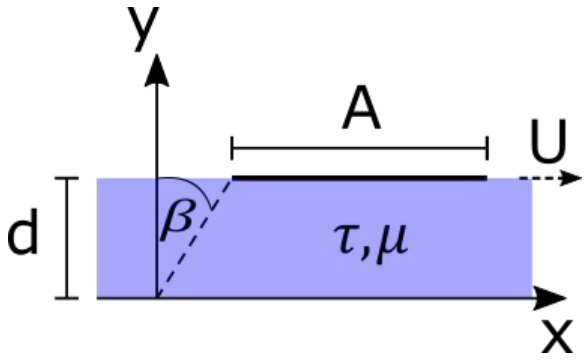


Figure 2.3: Concept for derivation of the viscosity. Plate A is moved on the surface of a fluid, represented by a blue box, in direction U. Idea taken from [Ada18].

$\tau(\frac{d\beta}{dt})$ is a function of the deformation rate $\frac{d\beta}{dt}$. A **Newtonian fluid** is defined by this function being linear and therefore equal to the **shear rate** $\varphi = \frac{du_x}{dy}$, see Equation 2.2:

$$\tau_{Newtonian}\left(\frac{d\beta}{dt}\right) = \mu\frac{d\beta}{dt} = \mu\frac{du_x}{dy} = \mu\varphi \tag{2.2}$$

where u_x is the Cartesian x-component of U. μ is the **dynamic viscosity** with a unit of $\left[\frac{Ns}{m^2} = \frac{kg}{ms}\right]$, which in general depends on temperature and pressure. For liquids, in opposition to gases, μ declines with increasing temperature since the intermolecular attractive forces decrease [Ada18]. In general, the shear rate is a tensor of second order [Alt12], meaning a matrix of the following form:

$$\varphi = \begin{pmatrix} \frac{du_x}{dx} & \frac{du_x}{dy} & \frac{du_x}{dz} \\ \frac{du_y}{dx} & \frac{du_y}{dy} & \frac{du_y}{dz} \\ \frac{du_z}{dx} & \frac{du_z}{dy} & \frac{du_z}{dz} \end{pmatrix} \tag{2.3}$$

Often the tensor gets simplified by arguing that velocity only changes in a certain direction so the derivates into the other directions must be zero, e.g., $\frac{du_x}{dx} = 0$ or that a flow only goes into a certain direction, meaning the velocity components are zero in the other directions, and their derivatives as well. In the example above, it is assumed that $u_y = u_z = 0$. To have a valid expression for talking about shear rate as a scalar value for every possible form of φ, one could for example use the matrix norm [BZ22] of (2.3).

Non-Newtonian fluids have a non-linear viscosity function $\tau\left(\frac{d\beta}{dt}\right)$. They are problematic in the sense that there exists no closed mathematical expression to describe them [Bai08]. This means especially that no exact formula for the viscosity of blood is known [MHa95], since blood is a non-Newtonian fluid [RL64],[NS19]. Based on experimental observations, [THJ04] asserts that blood has a viscoelastic behaviour, which means it behaves like a combination of a liquid (viscous) and an elastic solid. This is attributed to the elasticity of red blood cells, which make up about 40 [%] vol. of human blood [Das11].

Generally speaking, it can be said that the dynamic viscosity of a Newtonian fluid always remains constant, whereas for non-Newtonian fluids the viscosity changes with different shear rates [AAa20]. Different models are proposed to describe the phenomena, of which the Power-Law model and Herschel-Bulkey model will be introduced. For the Power-Law model, Equation 2.4 is used to determine the shear stress [MHa95]:

$$\tau_{Power}\left(\frac{d\beta}{dt}\right) = K_{cc}\left(\frac{du_x}{dy}\right)^{n_{Power}} = \mu_a \frac{du_x}{dy} = \mu_a \varphi \tag{2.4}$$

where $K_{cc}\left[\frac{kg \cdot n^{n-2}}{m}\right]$ is a consistency coefficient and n_{Power} the **Power-Law index**. For $n_{Power} = 1$, a Newtonian fluid is described and $K_c c = \mu$ from (2.2). $\mu_a := K\left(\frac{du_x}{dy}\right)^{n_{Power}-1}$ where K depends on the fluid. μ_a has a unit of $\left[\frac{kg}{ms}\right]$. If $n_{Power} < 1$, the equation describes a **pseudo-plastic** of **shear thinning fluid**, whereas $n_{Power} > 1$ is used for **dilatant fluids** [MHa95]. Blood is a shear-thinning fluid with n_{Power} close to, but smaller than 1 [MHa95]. According to [NN12], $n_{Power} = 0.9$. The Power-Law model is also known under the name **Ostwalde-de Waele model** [AAa20].

Another way to describe a non-Newtonian fluid is the **Herschel-Bulkey model** [Das11]:

$$\tau_{HP}\left(\frac{d\beta}{dt}\right) = \mu_{HB}\left(\frac{du_x}{dy}\right)^{n_{HB}} + \tau_0 = \mu_{HB} \cdot \varphi^{n_{HB}} + \tau_0 \quad for \quad \tau_{HP} \geq \tau_0 \tag{2.5}$$

τ_0 is the **yield stress** and n_{HB} the Herschel-Bulkey parameter. Notice the similarities to the Power-Law model. $n_{HB} = 1$ describes a "Bringham plastic" [Das11]. Yield stress is the amount of shear stress for which a material changes from elastic to plastic behaviour [Die86]. Additional non-Newtonian models can be found in [AAa20], for example : Carreau-Yasuda [Irg13], Kundu and Cohen [KC02] and Tropea et al. [TYF07].

[Que83] and [Rod83] claim that for large vessels, say arteries, blood can still be treated as a Newtonian fluid. It must be noted that [Que83] and [Rod83] are taken from the same book. Modelling the viscosity μ of blood provides an issue that is addressed in Chapter 3: Human Blood Viscosity. Especially for slow flow rates, near quasi steady conditions, the transition area between vessels of different sizes and especially for small blood vessels like capillaries, blood must be treated as non-Newtonian fluid [MHa95].

2.1.4 Apparent Viscosity

Not only does the viscosity of blood depend on the shear rate, but also the ratio of plasma and red blood cells. Also, factors like temperature play a role, see Chapters 2.2 and Chapter 3. For this reason, the expression **apparent viscosity** is used to make clear that μ is not fixed throughout the blood circulatory system but a local value [AMAa11]. It also alludes to the idea that viscosity in vivo can only be measured close to the vessel wall. Later will be explained that even inside the same blood vessel, viscosity is strongly

depending on the distance from the centre axis **(Fahraeus Lindquist effect)**, see Chapter 2.2. This underlines the complexity and importance of modelling the fluidic properties of blood accurately.

2.1.5 Kinematics of Flow

FEs are used to describe the kinematic of flows. Either a single FE is tracked the whole time and the observer moves along with it (Lagrange description) or the observer is at a fixed position and watches multiple FEs passing through this location (Euler description). Figure 2.4 illustrates the difference between the two descriptions. A flow is said to be **stationary**, if all of its properties Φ aside from the location do not change over time at any given (but fixed) location, $\vec{x}_0$ meaning: $\frac{d\Phi(\vec{x}_0)}{dt} = 0$. Φ can be vectorial. These considerations provide the mathematical tools for formulating equations like the Reynolds-Transport-Theorem or the **Navier-Stokes equations**. Physical laws taken from classical

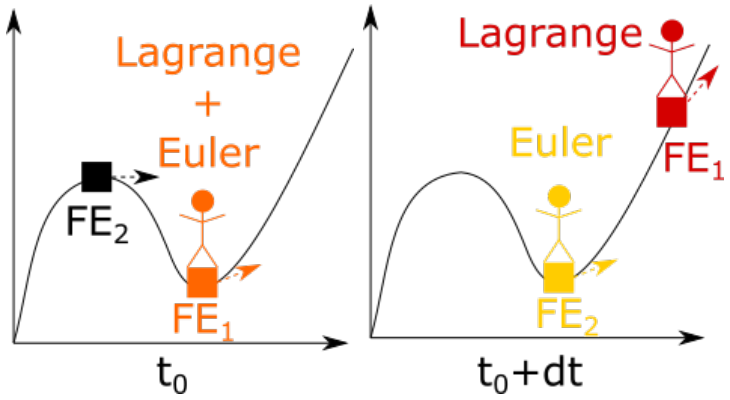


Figure 2.4: Illustration of the difference between the Lagrange description and the Euler description. At the starting point t_0, both refer to the same position and the same fluid element (left side). But as time dt passes (right side), the Lagrange description follows FE_1, whereas the Euler description stays at the same place where in the meantime FE_2 has arrived. Idea taken from [Ada18]

.

Newtonian mechanics can now be formulated for liquids, like the conservation of mass, energy and momentum. For details, please refer to chapter 3.3, 4 and 5 of [Ada18]. Utilizing these equations, the mathematical Boundary Value Problems of a fluidic set-up can be solved. The boundary value problems organically describe the experimental situation in real life: Only a few variables are known and only at certain points measurements can be taken.

The simulation software used in this thesis, COMSOL (COMSOL AB, Sweden, Version 5.6), uses the Navier-Stokes equations together with the continuity equation and the equation for compressible fluids [Com22a]. Therefore, they are explained in more detail in Chapters 2.1.6 and 2.1.7.

2.1.6 Continuity Equation and Compressible Fluids

The derivation of the continuity equation is done by examining the physical law of the conservation of mass. For a given fluid volume, the mass m always stays the same over the time, meaning $\frac{dm}{dt} = 0$. In general, the mass of an object is the volume-integral over its local density $\rho(\vec{x}, t)$. This means $\frac{dm}{dt} = \int_V \rho(\vec{x}, t) d\tilde{V}$. Together with the Reynolds-Transport theorem and other equations, this leads to the **continuity equation** (2.6). For a detailed derivation, especially an explanation of the Reynolds-Transport-Theorem, please refer to [Ada18].

$$\frac{d\rho(\vec{x}, t)}{dt} + \nabla \cdot (\rho(\vec{x}, t)\vec{u}) = 0, \quad \nabla = \left(\frac{d}{dx}, \frac{d}{dy}, \frac{d}{dz}\right)^T \tag{2.6}$$

An additional important concept in fluid mechanics is the **material derivative**. Let Φ be an arbitrary, vectorial or scalar fluid property. The material derivative $\frac{D\Phi}{Dt}$ is defined as [Ada18]:

$$\frac{D\Phi(\vec{x}, t)}{Dt} := \frac{d\Phi(\vec{x}, t)}{dt} + \vec{u} \cdot \nabla\Phi(\vec{x}, t) \tag{2.7}$$

A fluid is said to be **incompressible** if the material derivative of the density ρ is close or equal to zero $\frac{D\rho(\vec{x},t)}{Dt} = 0$. In contrast, this condition is not met by a **compressible fluid**. For an incompressible fluid, subtracting (2.7) from (2.6) and using the product rule for derivations on (2.6) results in:

$$\rho(\vec{x}, t)(\nabla \cdot \vec{u}) = 0 \tag{2.8}$$

The authors of [NWa09] treat blood as incompressible in their simulations as can be taken from their "Methods" subsection. As already mentioned above, others claim that blood is a compressible fluid due to the deformability of red blood cells [Bai08]. Again, blood cells make up around 40 [%] of the human blood in terms of volume [Das11].

2.1.7 Navier-Stokes Equations

From the differential form of the impulse conservation equation, the **Navier-Stokes equations** (2.9) can be derived. For details, please refer to chapter 4.2 and 6 in [Ada18]. Notice that (2.9) consists of three equations, one for each Cartesian coordinate.

$$\frac{d}{dt}(\rho\vec{u}) + \nabla \cdot (\rho\vec{u} \otimes \vec{u}) = -\nabla p + \nabla \cdot \bar{\bar{\tau}} + \rho\vec{F} \tag{2.9}$$

$\otimes$ denotes the outer product of two vectors. Again, ρ represents the density and $\vec{u}$ the velocity vector. $\bar{\bar{\tau}}$ is the stress tensor and $\vec{F}$ forces acting on every FE, for example gravity.

COMSOL's GUI shows all equations used by the numerical solver. The Navier-Stokes equations is used in the form of (2.10), which has some minor differences to (2.9):

$$\rho\frac{d\mathbf{u}}{dt} + \nabla[-p\mathbf{I} + \mathbf{K}] + \mathbf{F} \tag{2.10}$$

Formally, COMSOL uses bold letters to denote vectors. Important differences are that ρ is assumed to be time and spatially independent and presumably included in the F term. I represents the unity matrix, K the viscose forces and F the external forces [Com22d]. More details on this matter could not be found in the documentation.

2.1.8 Reynolds Number

Another important number when it comes to fluid mechanics, is the **Reynolds number** Re, defined as [Ada18]:

$$Re = \frac{\|\vec{u}\| l_{chr} \rho}{\mu} \geq 0 \tag{2.11}$$

Assuming $Re << 1$, the Navier-Stokes equation can be simplified. This simplification is called **Stokes flow** or **creeping flow**. If a stationary, incompressible flow is also assumed, one receives the set of equations (2.12)-(2.14). Equation 2.12 states the simplification that can be made regarding the gradient of p [Ada18]:

$$\nabla p = \mu \Delta \vec{u} \tag{2.12}$$

$$\nabla \cdot \vec{u} = 0 \tag{2.13}$$

$$\Delta(\nabla \times \vec{u}) = 0 \tag{2.14}$$

Keep in mind that the use of μ implies the assumption of a Newtonian fluid. In the case of blood, one obtains the following values: $\|\vec{u}\| = 0.03 - 20 \left[\frac{cm}{s}\right]$, $l = 100 - 250\ [um]$, see Subchapter 2.1.1, $\rho = 990 - 1060\ \left[\frac{kg}{m^3}\right]$ and $\mu = 1.2 - 4[mPa \cdot s]$ which leads to $Re = 0.007 - 44$. In conclusion, the simplification can not always be made for the calculations when considering blood flow. Re can even reach values above 2000 in some cases, see page 228 in [BLSa19].

The average velocity of blood depends on the location in the circulatory system. It ranges from $0.03 \left[\frac{cm}{s}\right]$ in the capillaries to $20 \left[\frac{cm}{s}\right]$ in big vessels like the aorta, see fig. 19.4 in [BLSa19], corresponding to Fig. 2.7 in this thesis. For big vessels with a rapid flow, the viscosity is around $3 - 4\ [mPa \cdot s]$, while plasma only has a value of $1.2\ [mPa \cdot s]$ pp. 223 ff. [BLSa19]. Water at body temperature ($37\ [°C]$) has a density of $\rho = 990\ \left[\frac{kg}{m^3}\right]$ [USG22] and can be used as an approximation for blood plasma, which mostly consists of water. At the same temperature blood has a density of $\rho = 1060\ \left[\frac{kg}{m^3}\right]$[CJ98]. The density of the RBC layer, see Fahraeus-Lindqvist effect Chapter 2.2 and 3, might be even higher.

[AMAa11] states that in the proximal segments of the aorta and the a. pulmonalis, the Latin name for the lung artery, during systole, the Reynolds number can be significantly higher than 2000, resulting in a turbulent flow, as described in the next subchapter. Systole simply describes the phase of heart muscle contraction when blood is pushed from the chamber into the aorta. Malfunctions like vascular stenosis and anaemia can also result in a turbulent flow in arteries far away from the heart.

2.1.9 Laminar and Turbulent flow

To know if a flow is laminar or turbulent is important, because it influences the equations that need to be solved for a fluid mechanical boundary problem. For example, the rotation of the velocity vector for a laminar flow is close to or equal to zero, but not so for a turbulent flow. It also influences the **pipe friction coefficient** λ_{pfc}. To determine if a flow is laminar or not, the Reynolds number is used again.

In general, λ_{pfc} is defined as [Ada18]:

$$\lambda_{pfc} \coloneqq \delta p \frac{2d}{\rho \bar{U}^2 l} \tag{2.15}$$

where δp is the pressure loss over the pipe length l with d being the pipe diameter and $\bar{U}$ the velocity averaged over the pipe bisection. In case of a turbulent flow, the velocity is also averaged.

A flow is said to be laminar if $Re < Re_{crit} = 2040$ [AMAa11]. In this case, λ_{pfc} can be calculated as [Ada18]:

$$\lambda_{pfc} = \frac{64\mu}{\bar{U} d \rho} \tag{2.16}$$

For turbulent flows, the average surface roughness k_s needs to be considered. This is an experimental value that can be found in engineering sheets for various materials, manufacturing methods and for surfaces of different quality [Ada18]. Depending on the value of k_s, the distinction between **hydraulically even** and **hydraulically rough** valves must be made. This influences how λ_{pfc} can be determined.

A valve is hydraulically even, if [Ada18]:

$$k_s \leq \frac{5\mu}{\sqrt{\rho \tau}} \tag{2.17}$$

Now λ_{pfc} can be taken from a corresponding table or by solving (2.18) numerically [Ada18]:

$$\sqrt{\lambda_{pfc}} \left(ln \left(Re_d \sqrt{\lambda_{pfc}} \right) - 0.4 \right) = 0.5, \quad Re_d \coloneqq \frac{\rho \bar{U} d}{\mu} \tag{2.18}$$

For $Re_{crit} \leq Re \leq 10^5$ the simpler expression (2.19) can be used [Ada18]:

$$\lambda_{pfc} = \frac{0.3164}{\sqrt[4]{Re_d}} \tag{2.19}$$

A valve is considered hydraulically rough, if [Ada18]:

$$k_s \geq \frac{70\mu}{\sqrt{\rho \tau}} \tag{2.20}$$

λ_{pfc} can then only be determined experimentally. Results can be found in corresponding Moody diagrams. If k_s is in the **hydraulic transition period**, meaning $\frac{70\mu}{\sqrt{\rho\tau}} < k_s < \frac{5\mu}{\sqrt{\rho\tau}}$, one can decide between using a table of experimental values or the two empirical relations below, named Colebrook-White (2.21) and Karman-Nikurade (2.22) [Ada18]:

$$\sqrt{\lambda_{pfc}} log \left(\frac{2.51}{\sqrt{\lambda_{pfc}} Re_d} + \frac{0.27 k_s}{d} \right) = -0.5 \tag{2.21}$$

$$\lambda_{pfc} = \left(1.14 - 2ln \left(\frac{k_s}{d} \right) \right)^{-2} \tag{2.22}$$

For λ_{pfc} and k_s no values could be found in literature.

2.1.10 Hydraulic Diameter

A noteworthy concept is the **hydraulic diameter** d_{hydr}, that allows to amend the formulas from above for valves with non-circular cross-sections. d_{hydr} is calculated with the help of the cross-section of the fluid A_F and the length of the valve's wall in contact with the liquid l_F [Ada18].

$$d_{hydr} = \frac{4 A_F}{l_F} \tag{2.23}$$

Figure 2.5 shows the idea behind this formula. Given the fact that blood vessels are assumed to be perfectly symmetrical valves and the blood always occupies the full cross-section of a vessel throughout this thesis. These assumptions imply that there is always enough blood to compensate the losses of the wound. This is done for the sake of simplicity and can be reasoned by considering the total blood volume of a human of around 5 L compared to the losses of a needle injury, for example at the tip of a finger, which will not exceed a few mL. No hint in literature was found to suggest this is not plausible. Still, d_{hydr} bears significance for the microfluidic chips, see Chapter 5. Here, a rectangular cross-section has to be chosen for manufacturing reasons.

2.1.11 Hagen-Poiseuille Equation

Given the geometry of an idealized blood vessel, it seems like an analytical expression should exist for all properties of the flow, for example pressure, velocity etc. While the geometry is indeed simple, with or without the hole, the complex behaviour of blood is the issue at hand. If blood was laminar, stationary and homogeneous (Newtonian) one could make use of the **Hagen-Poiseuille equations** [BLSa19] to analytically solve the flow conditions for a vessel without a wound. The fluid volume $\frac{dV}{dt}$ flowing through any cross-section of the vessel from inlet to outlet would in this case corresponds to:

$$\frac{dV}{dt} = \frac{r_i^4 \pi (p_1 - p_2)}{8 \mu l} = const. \tag{2.24}$$

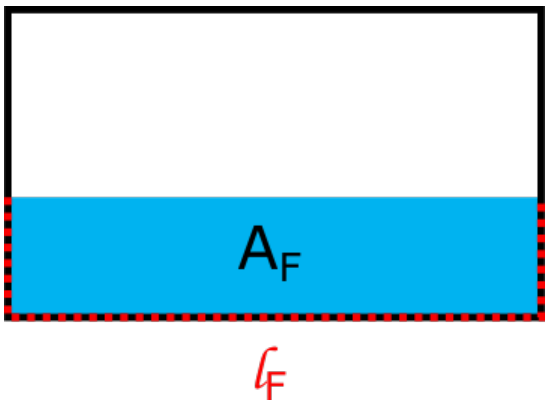


Figure 2.5: Concept of the hydraulic diameter. Idea taken from [Ada18].

Where p_1 is the pressure at inlet and p_2 at the outlet, l the length, r_i the inner diameter of the valve. With this, Equation 2.24 can be derived, which states the closed analytical expression for the velocity profile [Ada18]. Please refer to chapter 6.2.2 in [Ada18] for a detailed derivation. Ideally, such an expression would also exist for the same set-up with blood and wound included, but it seems that a numerical evaluation is more promising, see Chapters 3 and 4.

$$u_x(r) = \frac{r^2 - r_i^2}{4\mu} \cdot \frac{dp}{dx} \tag{2.25}$$

For many reasons, the assumptions used for the derivation of (2.24) are not true for blood as a fluid. E.g., in most blood vessels flow is not stationary, but pulsating because of the heartbeat. The Hagen Poiseuille equations cannot simply be applied [BLSa19]. Still Figure 2.6 provides a view of how the velocity profile of a simple fluid like water would look like. It can be noticed that the velocity increases towards the centre of the valve.

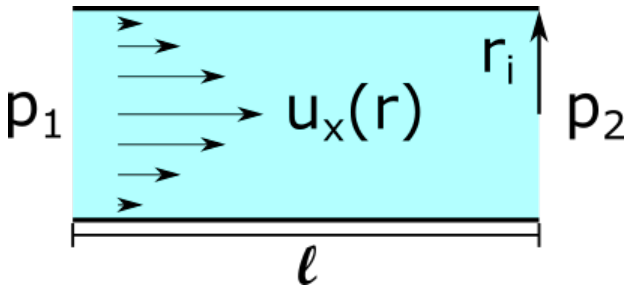


Figure 2.6: Velocity profile of a Newtonian fluid for a cylindrical valve with laminar and stationary flow conditions. Concept taken [BLSa19].

2.1.12 Summary

Like in many sub-disciplines of mechanical engineering, formulas derived with analytical physical considerations must be simplified to keep the expressions manageable. It can already be seen that, especially for defining the fluidic conditions in blood, formulas become quite complex. This issue is further discussed in Chapter 3 “Human Blood Viscosity”.

2.2 Biology

The second part of scientific theory needed in this thesis is a biological description of blood circulation, blood vessels and blood itself. Furthermore, the process of **hemostasias** is briefly introduced from a medical perspective. Because only a small part of the complex circulatory system is of interest, no attention is paid to the heart, lung or any other organs. Also, for hemostasis there will be a strong focus on the most important factors like **platelets**, also called thrombocytes. This chapter will closely follow the books "Physiologie des Menschen" (Human Physiology) written by Ralf Brandes, Florian Lang, Robert Schmidt and others [BLSa19] as well as "Anatomie Physiologie" (Anatomy Physiology) by Christoph Zalpour [Zal10].

2.2.1 Circulatory System

As with any fluid, blood flows along a gradient of high to low pressure. Vessel types along this way are distinct from each other in structure and size. The local pressure and flow resistance changes accordingly and in response to determine the local velocity, see Figure 2.7. In addition, the flow is influenced by the fact that blood is not emitted in a continuous way by the heart; see Chapter 2.2.5 "Pulse Waves".

The main purpose of the circulatory system is to deliver oxygen to the organs from the lung. CO_2 is emitted in reverse direction. Inside the lung, capillaries and pulmonary alveoli exchange carbon-dioxide from the body for oxygen from the inhaled air. Exhaling allows for a final removal of CO_2 from the human body. Also, nutrition can be delivered to all organs by the blood flow and exchanged for waste products via the corresponding capillaries [BLSa19].

From the left heart ventricle with the highest pressure, blood passes through the aorta, the arteries, the arterioles to the capillaries on to the venules via the veins and the vena cava to the right heart atrium, the point of lowest blood pressure. Vessel diameters decrease from Aorta to capillaries and they increase again till the vena cava is reached. All of this is illustrated in Figure 2.7 upper part.

In addition to the separation by pressure gradient, the circulatory system gets divided into the **systemic circulation** (SC) and the **pulmonary circulation** (PC). Besides the vessels themselves, which are connected both in parallel and in series, the two heart halves can also be understood as two pumps connected in series [BLSa19]. A difference in pressure causes the blood to flow; **the heart works as a pressure operated pump**. From the left ventricle of the heart to its right atrium, the SC can be found. It connects the heart to all the organs, but the lung. Correspondingly, the PC goes from the right ventricle to the left atrium and connects to the lung. Figure 2.7 provides an overview of the pressure gradient plotted over the system.

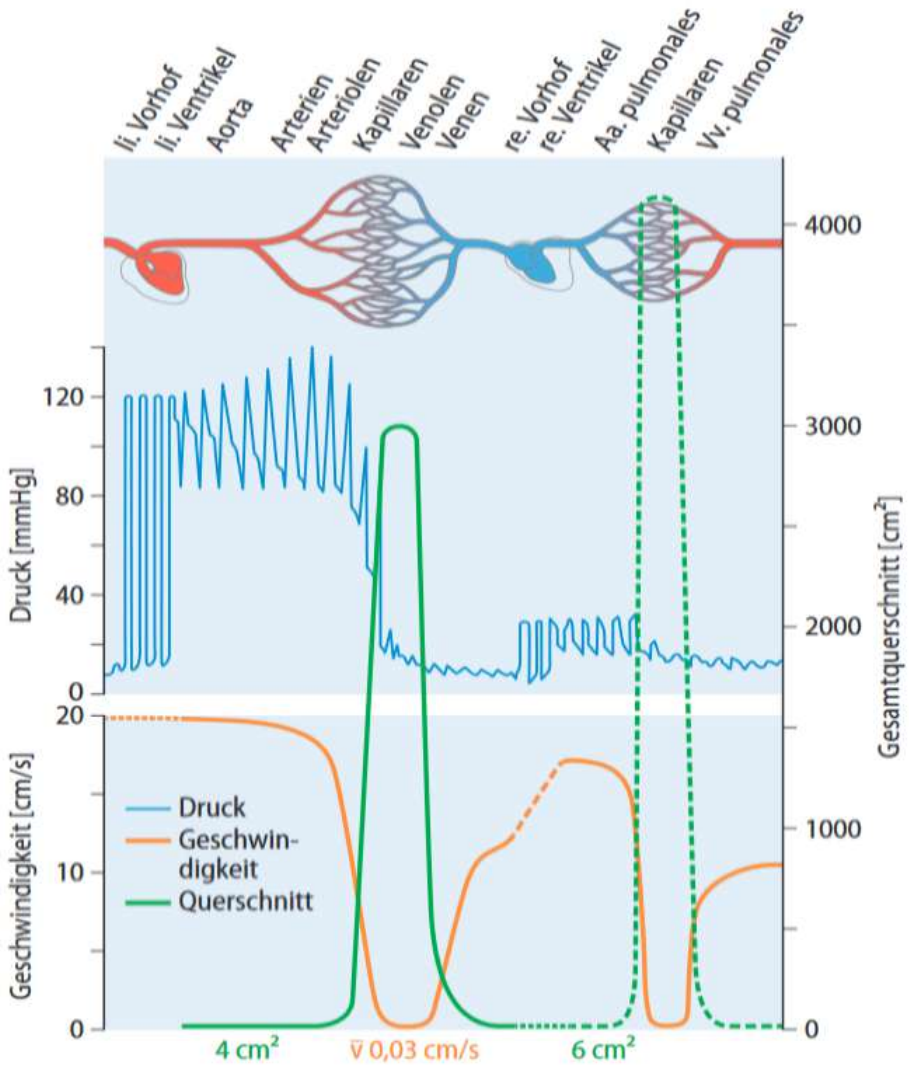


Figure 2.7: Blood pressure (German: Druck), -velocity (Geschwindigkeit) and vessel diameter (Gesamtquerschnitt) throughout the blood circle. Taken from [BLSa19].

Roughly speaking, the "pipes" of the circulatory system can be divided into three types:

- **Arteries** and **arterioles**: capable of operating at high pressure
- **Capillaries**: enable metabolism (molecule exchange) between blood and organs
- **Veins** and **venules**: transport blood back to the heart, operate at low pressure and can store a certain amount of blood because of their flexible structure

It becomes clear that creating a wound model needs to account for the vessel type. Besides variables like pressure and velocity, also the structure of the vessel wall changes, which will be the next topic.

2.2.2 Vessel Wall

Given the wide range of vessel types, one might already expect their anatomy to differ. Indeed, the composition of their walls changes accordingly, as can be seen in Figure 2.8 below. This influences the elasticity of a vessel wall. In general, the elasticity is determined by material properties and the **transmural pressure** P_{tm}, defined as the difference of the pressure of the tissue surrounding the vessel and the pressure inside of it. P_{tm} can be set equal to the inside pressure in case of most arteries because the pressure of the surrounding tissue is close to zero. This simplification however cannot be made for vessels close to the heart or skeletal muscle. Veins also experience a non-zero tissue pressure [BLSa19].

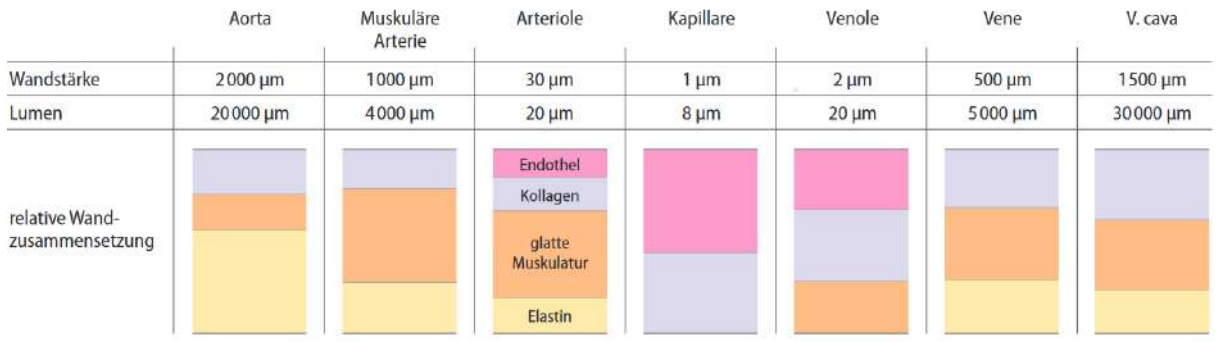


Figure 2.8: Relative amount of wall components for different vessels. First entry denotes vessel, thickness (German: Wandstärke), then lumen and then the relative wall composition (relative Wandzusammensetzung). Pink represents endothelium, purple collagen, orange smooth muscle tissue and yellow elastin. . Taken from [BLSa19].

The transmural pressure can be used to calculate the tangential wall stress σ_{tw} with the help of the inner diameter of the valve r_i and wall thickness h [BLSa19]:

$$\sigma_{tw} = \frac{r_i P_{tm}}{h} \tag{2.26}$$

Any vessel wall must be able to resist this stress. When describing the elasticity for medical purposes, both for a single vessel or the entire circulatory system, the compliance C is used. It depends on the patient's age and is defined as the change in volume over the change in pressure [BLSa19]:

$$C = \frac{\Delta V}{\Delta P} \tag{2.27}$$

Vessels can change their diameter and thereby influence the pressure drastically, especially for small vessels like arterioles or capillaries, see fig. 19.11, p. 233 in [BLSa19]. For arterioles, changes from above 50 to below 25 $[mmHg]$ are possible. This means that a fluid mechanical simulation must deal with a variety of boundary conditions when small vessels are considered. Especially the influence of capillaries on the blood flow tends to get underestimated.

Next, the composition of the wall of a blood vessel is discussed. Figure 2.9 provides a schematic of the general structure. Intima denotes the most inner part, consisting of a single layer of **endothelial cells** (ECS) upon the basement membrane (BM). ECs control the transfer of molecules into and out of the blood stream and prevent blood clothing when they are intact [MGOVa19]. Sequential follows "media", separated from the BM by the internal elastic lamina (IEL). The media part of the vessel is mostly made of vascular smooth muscle cells (VSMC) with additional extracellular matrix (ECM). VSMC can contract and relax the vessel [MGOVa19]. ECM is composed of collagen and von Willebrand factor (vWBF), amongst other things.

External elastic lamina (EEL) denotes the consequential layer, a barrier between the outer coat (adventitia) and media. Adventitia is tasked with both immune and inflammatory responses, in addition with vascular development and cell signalling. On the most outside layer, the vasa vasorum, perivascular adipose tissue (PVAT), nerve endings, lymph vessels and fibroblast can be found. Fibroblasts produce ECM and provide a first vascular response to, e.g., inflammation [MGOVa19].

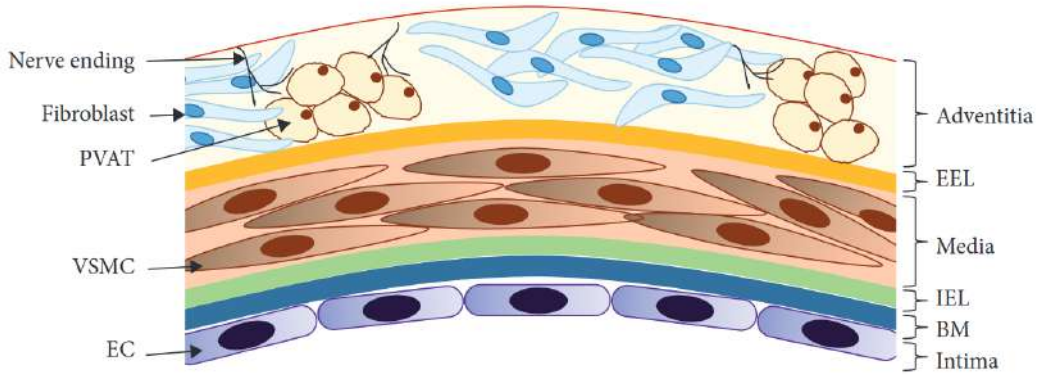


Figure 2.9: Structure of the vascular wall. The different layers are listed on the side of the picture. Taken from [MGOVa19].

In conclusion, vessel wall is an active component of the circulatory system, something that is hard to model in a computer simulation or a silica-based vessel replica.

2.2.3 Blood

Blood is the fluid of interest for this thesis. Its complex fluid mechanical properties were already discussed in the last chapter and will be further corroborated in the next. Reasons for this complexity is its complicated composition and the dynamic behaviour of **red blood cells** (RBC). In total, blood is difficult to model in a computer simulation or even just to describe by analytical mathematical expressions.

Being the transport vehicle for respiratory gases, hormones, nutrition and heat throughout the body, is the main task of blood. A grown-up has a blood volume of around 5 litres. Almost half of this is **blood plasma**, a fluid mostly composed of water, some dissolved ions and organic materials like glucose. RBCs are the other major contributor to blood volume, making up 40 % of its volume [Das11]. Other important cells flowing inside the blood are **white blood cells** and platelets. Additionally, the plasma contains certain proteins, like **fibrinogen**, that contribute to hemostasias significantly. In total, more than 1000 different proteins can be found in human plasma [PSZa19].

One of the tasks of plasma is to exchange molecules with the surrounding tissue. RBCs transport O_2 and CO_2. White blood cells, also known as leucocytes, are an important part of the immune system and fend off bacteria etc. They can migrate through the wall of a vessel. Last but not least are the platelet cells that seal off wounds to prevent further blood loss in case of an injury, see below [BLSa19].

The non-Newtonian behaviour of blood was already mentioned in the previous chapter. Since plasma consists mostly of water, red blood cells are mainly responsible for this phenomenon [Fun93]. This is due to the aggregation of erythrocytes cell, known as **rouleaux**. Their existence requires the proteins fibrinogen and globulin [Fun93]. They become prevalent for low shear rates and velocities until the RBCs form single cell lines within capillaries. It is speculated, that when the shear rate tends towards zero, blood behaves like solid [Fun93]. Increasing the shear rate leads to a break-up of the rouleaux and hence to a reduction of the blood's viscosity. Figure 2.10 shows pictures of various RBC rouleaux formations.

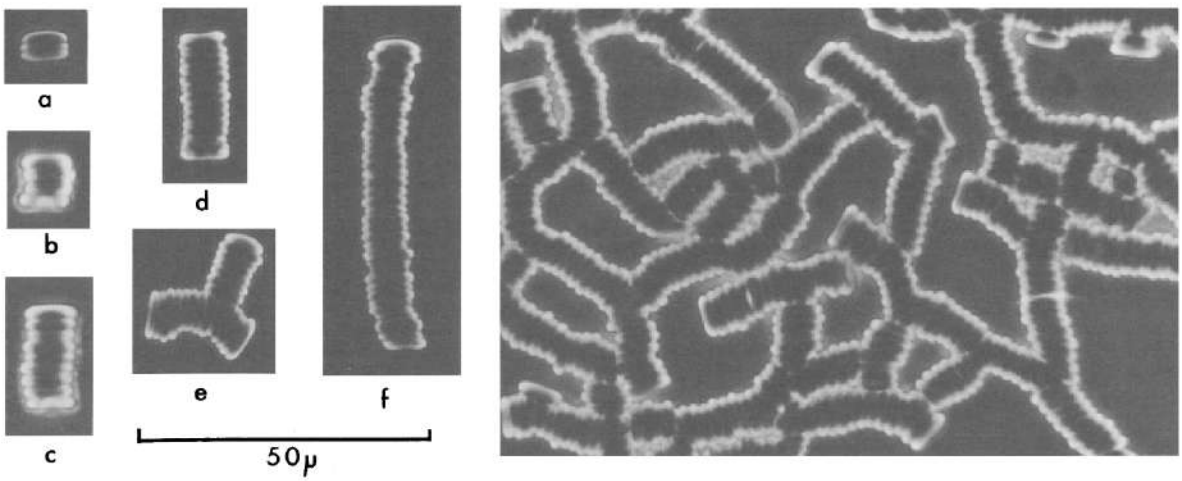


Figure 2.10: Pictures of various RBC rouleaux formations. The number of connected cells is 2, 4, 9, 15 and 36 for a, b, c, and f respectively. For e the number is not specified. On the right, a network of rouleaux can be seen. Taken from [Fun93].

Another explanation for the non-linear behaviour is the deformability of RBCs. For higher shear rates, blood behaves rather like an emulsion of droplets inside a surrounding fluid instead of small solid particles. With the help of this deformation, the blood can flow more easily and the apparent viscosity drops.

Modelling all these phenomena into a fluid mechanical simulation will be quite a challenge. Probably some simplifications must be made, e.g., not modelling individual RBCs but a two-phase flow. A dispersion of droplets inside a surrounding fluid could maybe represent the biological observations close enough to get an accurate description on a macroscopic, meaning cell-group, level.

2.2.4 Fahraeus-Lindqvist Effect

For vessels with a diameter of 4 to 300 $[um]$, a decline in apparent viscosity can be observed. Together with the RBC deformability mentioned above, this is due to the **Fahraeus-Lindqvist effect** (FLE), which describes the accumulation of erythrocytes in the middle of such vessels. Close to the wall, only a layer of plasma remains. Because of this separation, hematocrit has rather little influence on viscosity inside small vessels [BLSa19].

The outside layer encounters higher shear rates than the layer on the inside. Biologically speaking, the outer layer serves to decrease friction for the red blood cells moving in the middle. This means for the affected vessels, blood must be modelled as a multi-layer fluid, see Chapter 3.2. For diameter sizes of 5 to 10 $[um]$, apparent viscosity drops to plasma or water like levels. RBCs go through the vessel in a single file and experience extreme deformation. They end up looking like a droplet or parachute. For even smaller diameters, RBCs reach the limit of possible deformation, leading to a sharp increase in blood viscosity [BLSa19].

2.2.5 Pulse Waves

Another aspect to keep in mind when modelling a circulatory blood flow system are the **pulse waves**, resulting from the non-continuous operation of the heart, the heartbeat. The blood ejected by the heart is more accelerated than the blood in the aorta. This leads to an increase of pressure because of the inertia of the blood. Furthermore, an increased cross-section of the vessel can be observed, see Figure 2.11. The periodic increase of local blood volume induces a rhythmical increase and decrease of pressure gradient moving along the artery and synchronized with heart systole and diastole [BLSa19].

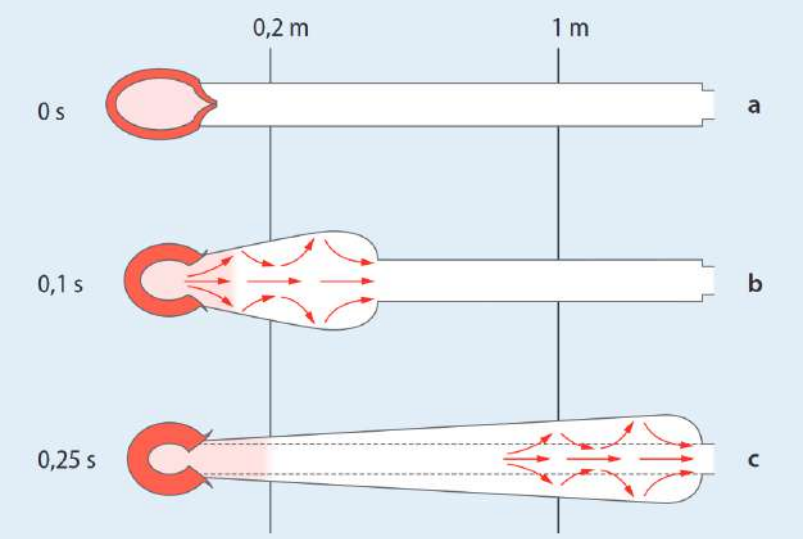


Figure 2.11: Schematics of a pulse wave propagation from the heart (left) into an arterial system. Taken from [BLSa19].

Three phenomena occur because of pulse waves: flow pulse, pressure pulse and volume pulse. For a system with no wave reflections, these three pulses would look identical in shape. However, wave-resistance, -reflection and -velocity are additional phenomena to account for in a biological system. A detailed description can be found in chapter 19.3.2 ff. in [BLSa19]. Here, it shall be sufficient to only state the formula for the pulse wave velocity c_{pulse} in Equation 2.27. For teenagers $c_pulse = 4 - 6,7$ and $9 - 10 \left[\frac{m}{s}\right]$ for aorta, a. femoralis and a. tibialis respectively. An increase in average blood pressure of 10 $[mmHg]$ increases c_{pulse} by around $0.4 - 0.8 \left[\frac{m}{s}\right]$ [BLSa19].

$$c_{pulse} = \sqrt{\frac{V_{local}}{\rho C}} \tag{2.28}$$

where V_{local} is the local blood volume.

An implementation of pulse waves into a computer simulation or a microfluidic model will be difficult to achieve. A pulse wave implies a time transient density, a compressible fluid, local turbulences and most problematic of all: a time dependent geometry of the vessel.

2.2.6 Influence of Gravity

It is to be expected that **gravity influences the parameters of the blood flow**, like it would influence a non-living hydraulic system. When a human is lying down, they are negligible. This is not the case for a standing person. For a grown-up, the difference between a stretched-out arm and the feet in standing position can be -30 to 90 and 35 to 180 $[mmHg]$ for venules and arteries respectively [BLSa19].

2.2.7 Microcirculation

The pressure inside small vessels like capillaries is essential for special parts of nutrition exchange between blood and organs. For an analysis of this exchange, the **Starling equation** is used, which must be amended for the effect of the lymph vessels. This equation describes how much fluid is passing through the vessel wall and therefore also influences the fluid mechanical description of capillaries. A detailed explanation can be found in chapter 20 of [Ada18].

2.2.8 Hemostasis

For every animal with a blood circulatory system, it is vital that the circuit's functionality is upheld. A lack of oxygen supply to certain organs like the brain or the heart can lead to a rapid death. Two of the key parameters for maintaining system functionality are the structural integrity and the amount of fluid passing through the body.

In this thesis the focus is on mechanisms used to fix smaller injuries to a vessel like a smaller cut in the finger or a needle penetrating a blood vessel. The protection mechanism for such injuries is called hemostasias. It consists of four steps: vessel contraction, short-term stop of blood loss, long-term of blood loss and a final "cleaning-up" step [Zal10]:

A. **Vessel contraction**: The damaged vessel reduces its diameter, hence limiting blood flow and consequentially -loss.

B. **Short-term stop of blood loss**: As a quick but temporary solution, a plug called thrombus clogs the wound. This is the key phenomenon from a medical perspective for this thesis.

C. **Long-term stop of blood loss**: For a long-term solution, a fibrin network forms to stabilize the thrombus. Cells called fibroblasts close-up the hole depositing extracellular matrix, finally creating the so-called scar.

D. **Cleaning-up**: Remains of the thrombus and fibrin network are removed to again allow an unimpeded flow of blood.

Especially smaller damages can occur all the time to the blood circulatory system, through, i.e., traumatic events or inflammatory processes/damages. Aside from blood loss to the outside world, internal bleeding or just superficial damage to the vessel hull also need to be fixed by one's coagulation system regularly.

Immediately after the vessel injury, a **shrinkage of the vessel** can be observed. This reduces the amount of blood flow and thereby the loss of this fluid. Furthermore, the ECs rolls up and gets sticky [Zal10]. Granules, a form of storage inside cells, from the thrombocytes release communication chemicals. Most important is the chemical thromboxane A2, which increases the level of vessel contraction. Aside from that also platelet factor 3 (PF3) is released to enhance the long-term stop of blood loss.

As a next step, platelet cells connect to the connective tissue fibres on the edges of the wound. With more and more platelets arriving at the sight, eventually the hole gets closed in 1-3 minutes if the wound is not too severe. The plug clogging up the wound is called the **thrombus**. It grows from the edges to the centre. Platelets are activated and expand their volume, see Figure 2.12. Up to now, only platelets are part of the thrombus and therefore it is called a **white thrombus**.

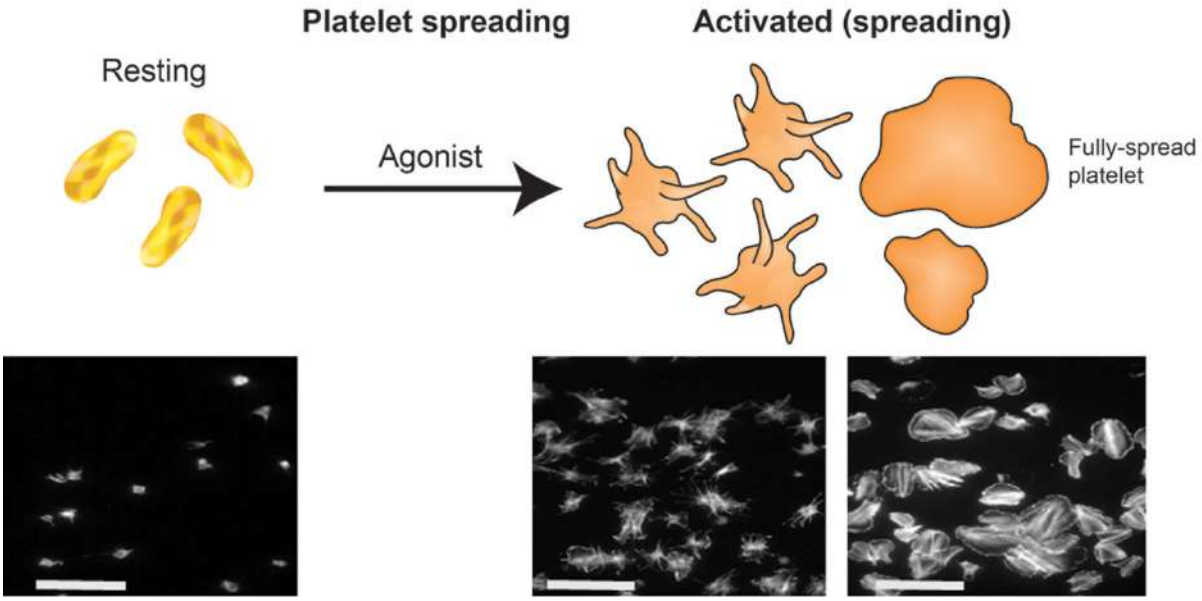


Figure 2.12: Schematic and images of platelets in resting (left) and activated state (right). The white bar corresponds to a length of 20 [*um*]. Taken from [MLLG20].

Such a situation may also occur in an intact vessel initiated by an irregular structure inside the vessel, e.g., endothelial destruction, atherosclerotic plaques or misfunction of the venous valves. If a thrombus forms by mistake inside an intact vessel, also RBCs can become part of the structure, leading to a red thrombus [Zal10]. White thrombi and red thrombi are two different processes. For this thesis, only white thrombi are considered. After an initial grow-phase, the thrombus starts to shrink again, 20 minutes after the occurrence of the injury [STWa13]. In detail, injury matrix proteins (e.g., collagen and vWBF) are exposed to the cell-surface receptors of the platelet [EM13] leading to a concentration of platelets around the wound, where they firmly connect with the vessel wall. Platelet activation is induced by an abnormal high shear rate present in the proximity of the wound hole [WA17]. Platelets already stuck, will recruit new ones by releasing agonists, such as ADP [EM13].

Subsequentially, a **network of fibrin** gets attached to the thrombus, which would otherwise disintegrate. Fibrin can be viewed as a bunch of stripes made out of glue. After a couple of minutes, with respect to the infliction of the injury, the vessel loosens up again and without fibrin as a connective glue, the platelet cells would be washed away. A long-term solution is needed. Asides from stabilizing the thrombus, the fibrin network pulls itself together and, in this manner, reduces the diameter of the wound hole. Now fibroblasts migrate into the thrombus, reorganize its structure and close up the wound for good with a scar [Zal10].

Days or weeks after the successful closure, thrombus and fibrin network get dismantled in a multistep process. Blood can now again flow normally through the vessel. This process is called **fibrinolysis** and is started by the enzyme plasmin. Under normal condition, only a precursor of plasmin called plasminogen can be found moving inside the blood stream. Its transformation is induced by, amongst others, urokinase and tissue plasminogen activator (tPA) [Zal10].

Similar to plasmin, fibrin only can be found in its deactivated state called **fibrinogen** in normal blood. Unwanted thrombi could form otherwise inside a healthy vessel. The activation of fibrin is a complex chain of reactions, involving many so-called **factors**. **Thrombin** marks the final step of this chain reaction, activating fibrinogen on the surface of the wound. But again, normally thrombin is deactivated and, in this state, called **prothrombin**. Thrombin itself gets activated by factors V to XIII and **calcium**. Calcium also plays an important role in the other steps of the reaction chain, as will be seen in the next few paragraphs. The odd numbering of the factors is due to historical reasons, meaning the order in which they were discovered. To be precise, the chain reaction mentioned just now is called **hemostasis**. Hemostasis can follow two different pathways, the **extrinsic** and the **intrinsic pathway** [Zal10].

Damage to the vessel, that penetrates the wall and not just scratches its inner surface, will be fixed via the extrinsic pathway. As soon as blood comes in contact with the tissue outside the vessel, factor III, called tissue factor, is released and in return activates factor VII, which now activates factor X with the help of calcium. In total, the extrinsic pathway only takes a couple of seconds. From factor X onwards, extrinsic and intrinsic pathway come to a common end track, discussed below. If the damage only scratches the inner surface of a vessel, the intrinsic pathway is activated. Factor XII gets released by the damaged ECs, forming the most inner layer, as already mentioned. Factor XII activates factor XI which in return activates factor IX. As a final step for this pathway, factor VII, IX together with calcium ions and PF3 activate factor X. With 1-6 minutes, the intrinsic is a lot slower than the extrinsic pathway.

Factor X and Factor V together with calcium, transmute prothrombin into thrombin as a common end track to both pathways. Finally, fibrin gets synthesized from circulating fibrinogen and a fibrin network can form [Zal10]. Take note that the two pathways are not strictly distinguishable, e.g., factor IX of the intrinsic system can also be activated by the extrinsic cascade.

Parallel to the whole process described so far, an inflammatory response is triggered, like attracting monocytes to the wound, to deal with possible intruders like bacteria [MGOVa19]. Last but not least, it shall be mentioned that so-called inhibitors circulate constantly in the blood to keep the vessel operational. For example, they deactivate fibrin that got detached from the thrombus or activated by accident. Most important to mention are the inhibitors antithrombin III (AT III) and protein C and S.

The sheer number of molecules involved, suggest that a biomedical simulation of hemostasias is most likely quite difficult to achieve. Making neglections that result in tolerable errors would be one of the key tasks. However, without accounting for hemostasis, a fluid mechanical simulation will most likely only be valid until the point in time at which the injury occurs. The reason for this is the change in geometry due to the forming thrombus. Again, describing the shear rates at the side of thrombus formation is key for this thesis, since it aims to provide numeric values during this process to support medical and biological research into this topic. An abnormal shear rate is the main non-biological factor for platelet activation [WA17]. Many simplifications were already made in the description of hemostasis above, a list of more factors can be found in Table 2.1 below.

2.2.9 Thrombus Structure

Thrombi are heterogenous and dynamic structures. It is not yet fully understood how they are shaped and according to which factors. However, recent studies like [SJTa14] defined at least two regions with different properties, stability and composition in mice thrombi in vivo; the so-called core and shell.

In vivo experiments show that cores are formed by dense, stably adherent and irreversibly activated platelets, enriched in thrombin, fibrin and p-selectin. It is surrounded by an outer layer of less adherent, non-fully activated platelets, forming a downstream tail. These separate layers can still be observed, even one hour after inflicting the injury [STWa13]. A graphical representation of core and shell can be found in Figure 2.13. For each layer different signalling pathways exist, “with adenosine 5´-diphosphate/P2Y12 signalling being critical for platelet recruitment and retention in the shell, whereas thrombin signalling drives full platelet activation and firm adhesion in the core” [SJTa14].

Clotting Factor	Synonym	Annotations	Normal Plasma Concentration $\left[\frac{\mu g}{ml}\right]$
Intrinsic (Residing in Blood Plasma)			
Factor VIII:C	Antihemophilic factor		0.1
Factor IX	Christmas factor [Fun93] / haemophilia B [Zal10]		4
Factor XI	Plasma thromboplastin antecedent [Fun93] / Rosenthal factor [Zal10]		4
Factor XII	Hageman or contact factor	Starting point for the intrinsic path	29
High-molekular-weight Kininogen	Fitzgerald factor		70
Prekallikrein	Fletcher factor		50
Von Willebrand factor	vWBF		7
Extrinsic (Residing in Cells)			
Factor III	Tissue factor /Tissue thromboplastin	Starting point of the extrinsic path	1
Factor VII	Proconvertin		1
Common Pathway			
Factor I	Fibrinogen		2500
Factor II	Prothrombin		100
Factor IV	Calcium		80-100 [Yee15]
Factor V	Proaccelerin		5-12
Factor VI	Accelerin, activated Proaccelerin [Zal10]	[Fun93] does not associate Factor VI with anything	-
Factor X	Stuart-Prower factor		5
Factor XIII	Fibrin Stabilizing factor		10

Table 2.1: Properties of Human Clotting Factors, taken from [Fun93], complemented with [Zal10].

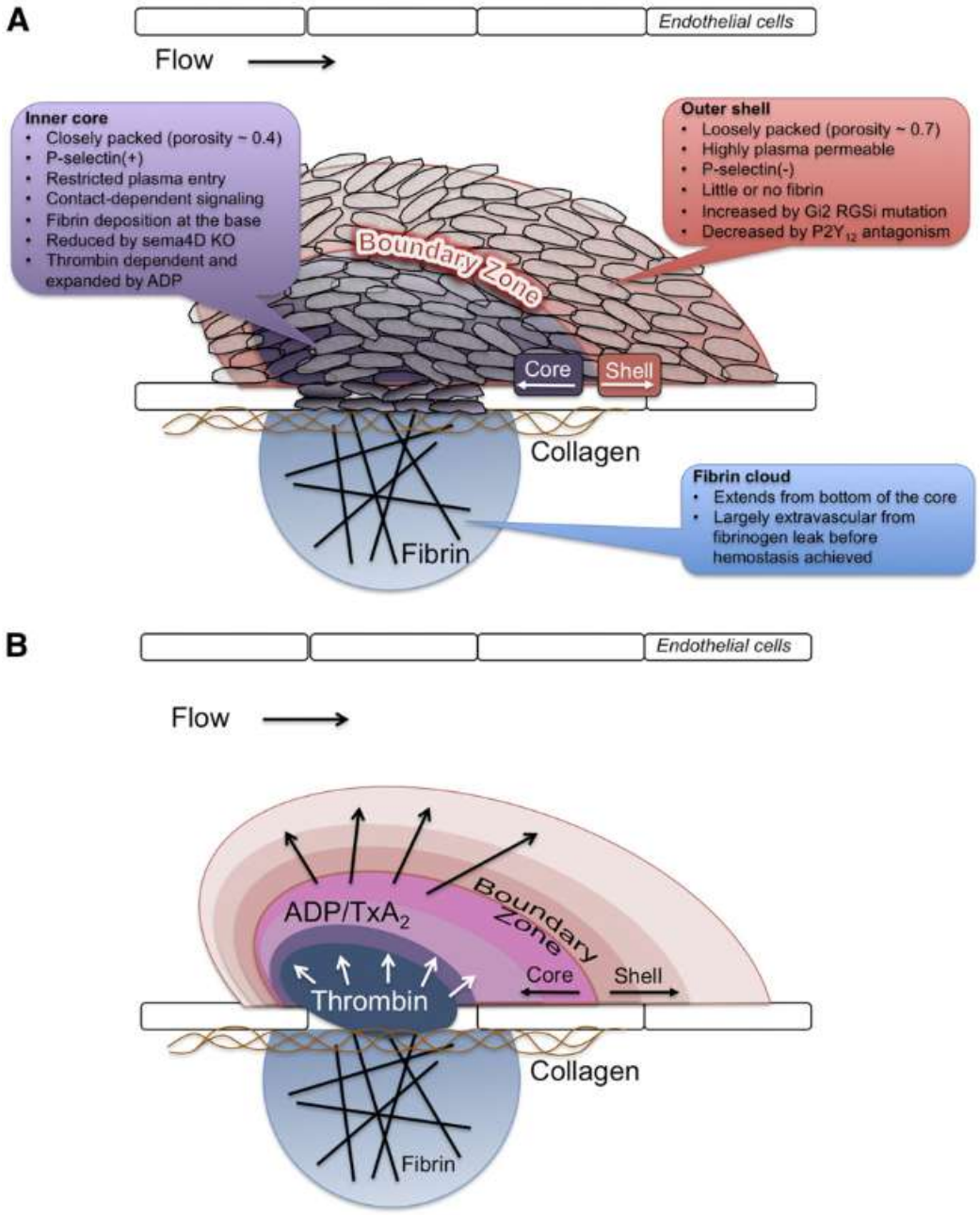


Figure 2.13: Structure of a thrombus. The difference between the core, directly at the wound, and the shell can be seen. The shell is shaped by the flow, with a "tail" forming in flow direction. Taken from [STWa13].

2.2.10 Hemostatic Components

This chapter closes with a list of the involved components for hemostasis and their brief description.

Platelets: Platelets originate from the megakaryocytes in the bone marrow. They have no nucleus and are disc-shaped with a diameter of $1 - 4$ $[um]$ [EM13] [Zal10]. Flowing freely in the blood, platelets not only play a role in hemostasias and thrombosis but also in inflammation, innate immunity and tumour metastasis. Their activation time is in the area of ms. Depending on gender and age, about 150 to $400 \cdot 10^9$ of them can be found in a litre of blood. Megakaryocytes in the bone marrow release new platelets into the blood stream. After circulating 7 to 10 days, mostly resident cells of the liver and spleen decide to take them out of service [MLLG20] [Zal10]. Although they have no nucleus, they contain RNA, ribosomes, mitochondria and a number of storage organelles and granules. The shape of platelets can flatten rapidly if they are exposed to collagen, laminin, fibrinogen and vWBF [MLLG20]. Platelets get activated by high local shear rates, even if encountered for only for a few milliseconds [WA17]. After activation, they will stick to exposed ECM and form a plug to close the wound, called thrombus. Furthermore, they contract forcefully to make the thrombus denser and more stable [TFTa19].

Fibrin: Fibrin molecules work as "glue stripes" to keep the thrombus together. In a process called clot retraction, clot density and size are increased by an interaction of fibrin and platelets. Thrombus stability and a secure cut-off of unwanted blood flow to the outside are the desired result. For mice, it was shown that the retraction process takes 1-2 hours [SAMa17].

Red Blood Cells: The key responsibility of RBCs is to ensure a supply of oxygen for the entire body and to transport the carbon dioxide "waste" to the lung to be exhaled. Normally RBCs do not occur in a thrombus that is meant to close an injury of the vessel walls (white thrombi). But thrombi can also form as a malfunction and clog up an otherwise functional vessel. In this scenario, RBCs become part of a thrombus and are then referred to as "red thrombi" [Zal10].

White Blood Cells: White Blood Cells arrive at an injury site after platelets started binding in the proximity. They attach to the platelets to tighten the platelet structure and kill bacteria that might have entered through the wound [MLLG20].

von Willebrand Factor: vWBF together with collagen leads to the activation of platelets [TFTa19]. It reacts to increasing shear stress or immobilization by changing its tertiary structure "including the molecule to unfold and expose sites within the A1 domain of vWBF that directly bind to the GPIbα subunit of GPIb-IX-V. This generates signalling events that trigger platelet aggregation" [MLLG20].

Collagen: Together with vWBF, collagen triggers the mobilization of calcium inside of platelets. As a result, the shape of the platelet changes. Furthermore, ADP and thromboxane A2 (TxA2) are released by the platelet, which trigger other platelets in the proximity [TFTa19].

Calcium: Calcium plays a key role during multiple phases of hemostasis. Its removal can be used to conserve donated blood [Zal10].

2.2.11 Summary

Table 2.1 and Subchapter 2.2.10 provide a compact overview of how many components are involved in the biochemical process of hemostasis. A simulation of these aspects most likely could only work with simplifications. For example, a reduction to the most important components like platelets and fibrin could be a way. Dividing blood as a fluid into discrete elements, see FEs from Chapter 2.1, a program could simply keep track of their concentration instead of simulating single molecules or cells.

It is to be noted that mice were used instead of humans for experiments serving as a reference for this thesis. All experiments were performed by the corresponding medical team from LMU, especially Mr. Rossaro. This is also a common mammal model in literature, see for example [SJTa14] and [NWa09]. Pigs are also used in literature [WA17]. For a detailed discussion on similarities/differences to human hemostasis, please refer to corresponding literature. For the scope of this thesis, data from animal experiments was treated interchangeably with human data.

In the remaining part of this thesis, most of the aspects from above will be neglected for the sake of simplicity and time-management. It is always assumed that a stationary flow has been established; pulse waves originating from the heart are not depicted. Therefore, only a local part of the vessel, where the injury occurs, will be modelled. From this stationary situation, the next step is an instantaneous injury, modelled as a simple hole in the wall. The wall itself is a rigid and static object, neglecting amongst other things its tightening during hemostasis. Air is presumed to be immediately beyond the vessel wall without surrounding tissue etc. Gravity is also not taken into account as it might also be the least influential neglection since rats do not provide enough height difference for gravity to influence blood flow in any significant way. Blood will be modelled as a homogeneous fluid, ignoring amongst other things the theory of FLE. Since Mr. Rossaros experiments always involved vessels larger than capillaries, micro-circulatory effects like the Starling equation can be left out.

After describing most of the relevant theory, Chapter 4 and 5 introduce two separate ways to obtain numerical values for the shear rate present at the site of an injury. Chapter 4 will use computer simulation and Chapter 5 will show a potential laboratory set-up.

3 Human Blood Viscosity

The viscosity of human blood cannot be treated as a scalar value. It rather depends on many factors such as temperature, hematocrit (volume-percent of red blood cells compared to whole blood) or shear rate [RL64] [NS19]. Additional influence can be attributed to the diameter of the vessel [Das11] [MHa95]. It is even reported that electromagnetic fields influence hematocrit and therefore viscosity [Bai08]. Running computer simulations of blood vessels or creating artificial blood for an ex-vivo model, requires an analytical expression of the viscosity. As input variables, this function must have the factors mentioned above. Here, the following approach is taken to solve this task: With the help of a dataset from [RL64], curve-fitting is performed.

As a first step, one has to find the mathematical characteristics of functions that correspond to experimental data. This can mean, for example, to choose between a polynomic- or trigonometric function. Next, the coefficients of these blank functions must be calculated. When a numerical expression of the function has been found, the difference between data-points and function is determined. Lastly, it is discussed how to improve this approach to receive more accurate results. [RL64] measured human blood viscosity at different temperatures, shear rates and hematocrits. It is important to notice that the shear rate was only indirectly determined from the rounds per-minute (rpm) of the centrifuge used in the experiment.

3.1 Viscosity for Wide Blood Vessels

It was already discussed in Chapter 2.2 that blood viscosity changes for different parts of the circulatory system. For big vessels with a rapid flow, the viscosity is around $3 - 4$ $[mPa \cdot s]$, while plasma only has a value of 1.2 $[mPa \cdot s]$. Water at 4 $[°C]$ has 1 $[mPa \cdot s]$ [BLSa19]. Figure 3.1 shows the change of relative blood viscosity (compared to plasma) for different vessel sizes.

3.2 Viscosity for Narrow Blood Vessels

For a narrow blood vessel, blood cannot be treated as a Newtonian fluid [MHa95]. To specify, [MHa95] talks about a “narrow arterial tube” without providing concrete numbers. Smaller vessels like capillaries are of course part of this category [MHa95]. When looking at the blood flow through such vessels, multilayer models have to be considered. The idea is shown in Figure 3.2.

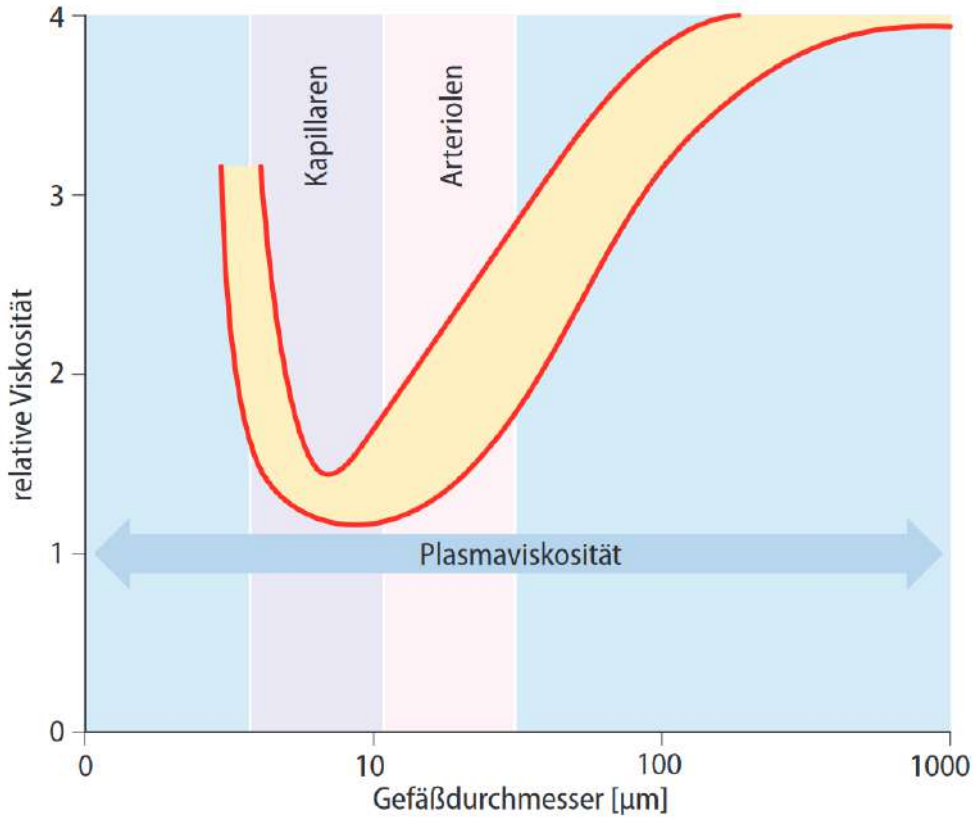


Figure 3.1: Relative blood viscosity for various vessel sizes. On the x-axis, the vessel diameter $[um]$ is plotted against the relative blood viscosity compared to plasma ($100\% \coloneqq 1.2\ [mPa \cdot s]$) on the y-axis. Taken from [BLSa19].

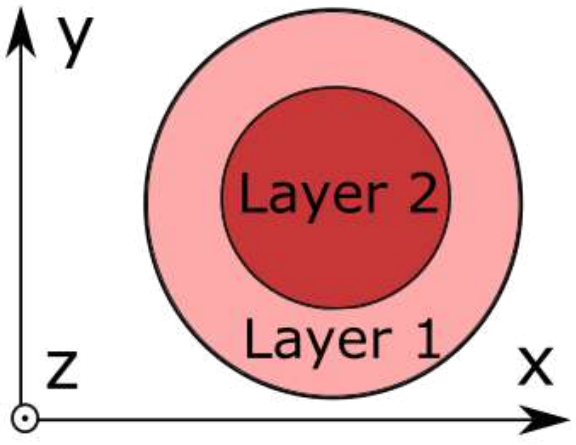


Figure 3.2: Splitting the flow through a vessel in different layers. Layer 1 is composed mostly of plasma and layer 2 of RBCs. Idea taken from [MHa95].

The reason for dividing the flow into layers is that red blood cells mostly flow inside the middle of the channel (layer 2 in Fig. 3.2), while towards the vessel hull only a thin [MHa95] layer of plasma (layer 1) remains, which moves slower than the layer in the middle [FL31]. This idea is backed up by imaging blood flow in animal vessels [Blo62]. According to [MHa95], some authors approximate the plasma layer to behave like a Newtonian fluid, while [8] points out fibrinogen causes a non-Newtonian behaviour even for the plasma. Other things to consider when modelling blood flow in a narrow vessel in steady state are the blunting of the velocity profile, the **Fahraeus-Lindqvist effect** [MHa95] [Sue78] and its inversion the **sigma effect** [Das11] [BS70] [Din67].

[MHa95] proposes a two-layer model, in which each layer as treated as an individual power law fluid, see Chapter 2.13 Eq. 2.3. [Das11] proposes a division of blood into three layers and the use of a Herschel-Bulkey model. Also, [Das11] provides references to many more published models. All the proposed models are quite complex and not further used in this thesis. For details, please refer to [Das11] and [MHa95]. It has to be noted that [Das11] has taken parts of his introduction from [MHa95]. Especially [Das11] p.1, bottom, is very similar to p.1 of [MHa95]. Structure and sources are very similar. Although [MHa95] is quoted in [Das11] at a later stage, it is only with reference to [MHa95]'s mathematical model. To the best of the author's knowledge, the model is [Das11]'s own creation.

3.3 Shear Rate dependency of Blood Viscosity

Platelets get activated when encountering abnormal high shear rates [NS19]. This means that the shear rate is the most important fluidic parameter for the purpose of this thesis. Ergo, it might be sufficient to model the viscosity of human blood only depending on the shear rate φ. [Bai08] proposes the following analytical function for the apparent viscosity μ:

$$\mu(\varphi) = \mu_\infty + (\mu_0 - \mu_\infty)\frac{1 + log(1 + \Lambda\varphi)}{1 + \Lambda\varphi} \tag{3.1}$$

Λ $[s]$ is a material constant, μ_0, μ_∞ $[cP] := centri\ pose = mPa \cdot s$ denotes the limes for $\varphi \longrightarrow 0$ or ∞ respectively. This function is based on general regression analysis performed by [PGa02], leading to the following numerical values: $\mu_0 = 180$ $[cP]$, $\mu_\infty = 3.96$ $[cP]$ and $\Lambda = 0$ or 53.22 $[s]$. Another expression can be found in [Fun93] and is stated in the equation below:

$$\mu(\varphi) = \frac{(\sqrt{\tau_y} + \sqrt{\eta|\varphi|}\,)^2}{|\varphi|} \tag{3.2}$$

where τ_y denotes the yield stress η denotes a material constant respectively. $\tau_y \approx 0.05$ $\left[\frac{dyn}{cm^2}\right]$ and almost constant for temperatures between 10 to 37 $[°C]$ [Fun93]. Numerical values for η could not be found. In order to demonstrate the procedure of determining a function with multiple arguments, blood viscosity is not only modelled depending on shear rate, but also on temperature and hematocrit as well in the rest of this chapter.

3.4 Experimental Blood Rheology

It might be desirable to recreate the experiments performed in [RL64], or do similar ones, to attain more data. As mentioned above, [RL64] does not measure the applied shear rate directly, but converts this value from the rpm of the used centrifuge. More specifically, a micro cone-plate viscometer was used. Multiplying the rpm with a fixed factor taken from the American National Bureau of Standards, provides the corresponding shear rate [RL64]. For more details, please take a look at the "Methods" section of the paper. Still, there exist other means to determine the viscosity of non-Newtonian fluids. In this paragraph, they will be quickly summarized, based on the overview presented in [AAa20].

A **shear rheometer** can be used to determine the shear rate dependency of the viscosity of a liquid. Unfortunately for microfluidic set-ups, especially the "flow in biological systems", this seems not to be advisable according to [AAa20] [YJUZ06]. Another possibility is combining **ultra-sound-Doppler velocity profiling** with **pressure difference technology**, called UVP-PD [Bra76] [WSS07]. The next option is **particle shadow velocimetry** (PSV) [Bru95] [Jen04] [Klo98] [RWK07]. [AAa20] used **shadowgraph particle image velocimetry** (PIV) for their own experiments. For more details on these methods, please refer to [AAa20] or its references.

As a last point for this subchapter, it should be noted that when performing experiments with blood, biocompatibility of the used materials plays an important role. A more detailed discussion of this topic can be found in Chapter 5. No measurements of blood viscosity were performed as part of this thesis. As mentioned, data provided in [RL64] was used. This is due to the additional workload of such experiments, which additionally have a more of a medical than an engineering character.

3.5 Analytical Functions for Blood Viscosity

Blood viscosity depends on many environmental factors. For simplicity, in this approach it is assumed that viscosity μ $[cP]$ only depends on temperature t $[°C]$, hematocrit h $[volume - \%]$ and shear rate φ $[s^{-1}]$. Furthermore, all three variables are treated as if they were independent of each other. One obtains a simple linear combination of three terms, all depending on only one variable. v must be a "physically meaningful" expression. To ensure this and limit the number of possibilities, the functions are assumed to be continuous and infinitely often differentiable. This is reasonable, as the law of energy conservation for Newtonian mechanics does not allow for non-continuous changes. Quantized energy states can be neglected here, because quantum physics not yet play a role on a scale of at least $1[nm]$:

$$\mu(t, h, \varphi) = f_1(t) \ + f_2(h) \ + f_3(\varphi) \in C^\infty \tag{3.3}$$

C^∞ denotes the vector space of all infinitely often continuously differentiable functions in mathematic literature. f_1, f_2 and $f_3 \in C^\infty$ for the same reasons stated above. Now the remaining question is: How do these three subfunctions look like?

One way to determine this, is to plot the experimental data and judge what kind of dependencies seem likely for $f_1 - f_3$. Given the biological complexity of blood, see Chapter 2.2, no bioanalytical analysis is performed here. For an example of such an approach, please refer to chapter 3 in [Fun93]. It is important to keep in mind, that scaling can have a huge impact on the presentation of a graph.

3.5.1 Determining Temperature Dependency

Since no graphs for the temperature–velocity relationship are provided in [RL64], they were created from the provided data. A first assumption for the sake of simplicity would be that the relationship is linear. Yet if one takes a look at Figure 3.3 and b, this seems unlikely. Judging by the created plots, $f_1(t)$ also seems to be an exponential decaying

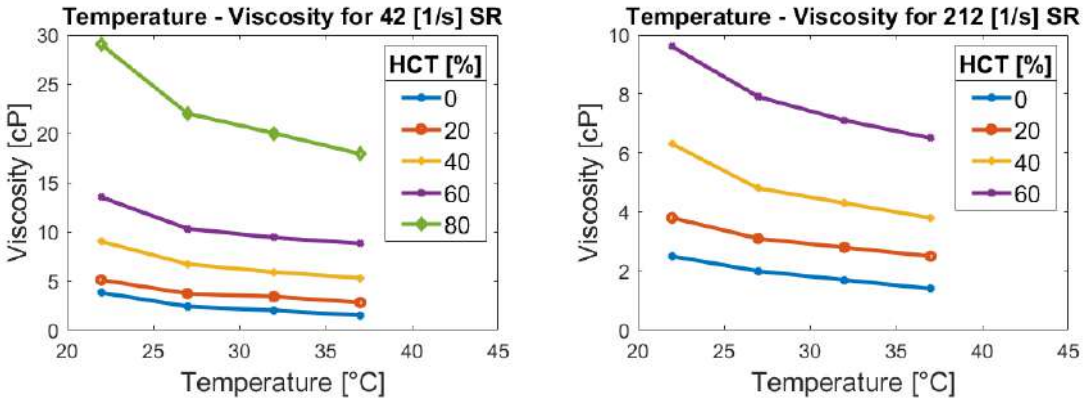


Figure 3.3: Temperature-viscosity relation for fixed shear rates and multiple hematocrits. Shear rate increases from left to right image. An almost linear decay can be observed, the first and second data point indicate a very weak exponential decay. Data taken from [RL64].

($b1 < 0$) function. The linear option is kept as a possibility to see which provides the better fit in the end. One can observe that especially in Figure 3.4 the slope close to the intercept of f_1 with the y-axis seems to be shear-rate dependent. This already implies that an independency of the three variables is not given. As the complexity of the problem would massively increase with interdependent variables, the simplification made, is kept in place.

$$f_1(t) = \begin{cases} a_1 e^{b_1 t} + c_1 \\ a_1 t + c_1 \end{cases} \tag{3.4}$$

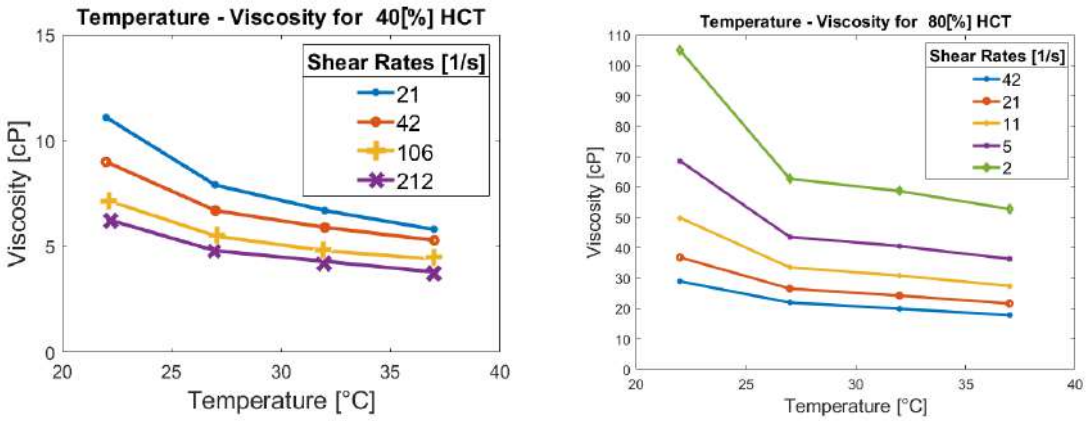


Figure 3.4: Temperature-viscosity relation for fixed hematocrits and multiple shear rates. Hematocrit increases from left to right image. An exponential decay can be observed for the first to points, afterwards it seems more like a linear decay. Data taken from [RL64].

3.5.2 Determining Hematocrit Dependency

According to literature, "whole blood viscosity is linear related to hematocrit" [NS19]. But looking at Figure 3.5 and 3.6, the relationship between hematocrit and viscosity might be rather a polynomial of higher order or exponential. Because in [RL64] only the different shear rates were fixed, also the hematocrit–viscosity relation for fixed temperatures is plotted in Figure 3.6. A quadratic function is most similar to a linear fit, in the sense that both are polynomials and its degree is closed to the degree of a linear polynomial. In order not to increase the number of combinatorial possibilities, an exponential relation is not considered. The option of a linear function is kept for research purposes. Thus, the following equations are possible for $f_2(h)$:

$$f_2(h) = \begin{cases} a_2h^2 + b_2h + c_2 \\ a_2h + c_2 \end{cases} \tag{3.5}$$

3.5.3 Determining Shear Rate Dependency

"Blood viscosity decreases exponentially with increasing shear rates"[NS19]. [Bai08] backs this up. Looking at Figures 3.7 and 3.8 seems to back up this claim. $f_3(\varphi)$ indeed seems to be an exponential decaying ($b_3 < 0$) function. The rapid declining could also be caused by a rational function, e.g., $\frac{1}{x-a}$. A pole, close to or at $\varphi = 0$ would then be the cause of the non-linearity:

$$f_3(\varphi) = \begin{cases} a_3 e^{b_3 \varphi} + c_3 \\ a_3 \frac{1}{\varphi - b_3} + c_3 \end{cases} \tag{3.6}$$

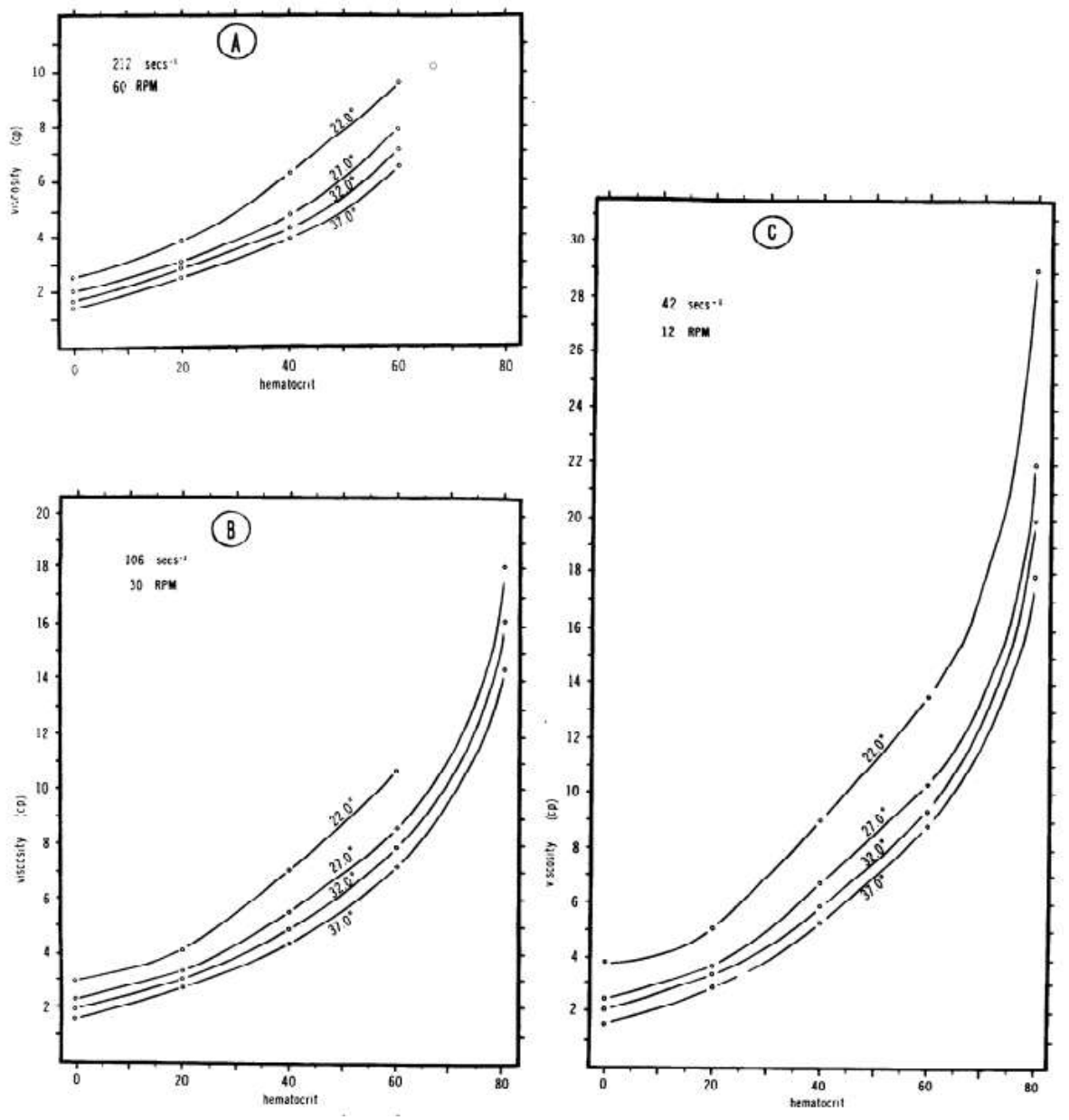


Figure 3.5: Hematocrit–viscosity relation for fixed shear rates at different temperatures. A, B and C denote a fixation of the shear rate to 212, 106 and 42 $\left[\frac{1}{2}\right]$ respectively. An exponential growth can be observed. Bad resolution is due to the quality of the original document. Taken from [RL64].

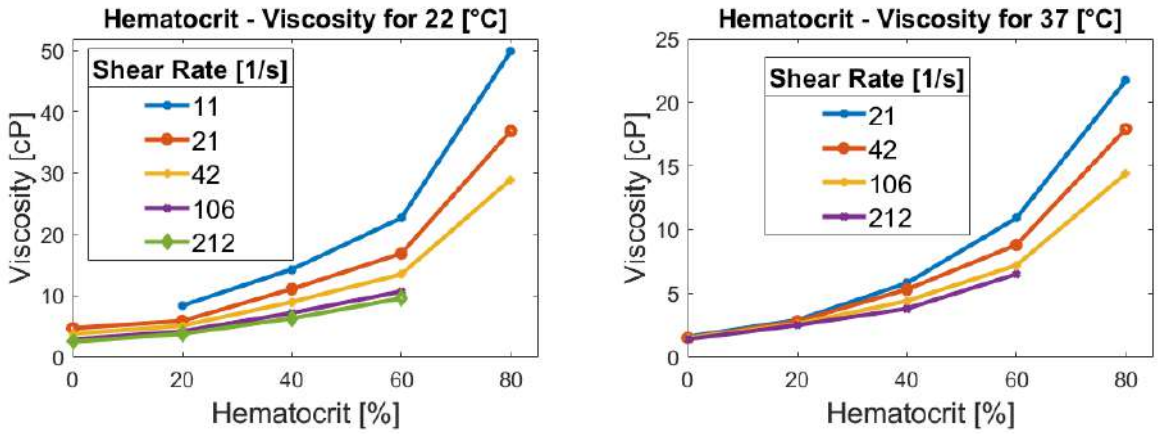


Figure 3.6: Hematocrit–viscosity relationship for fixed temperatures and varying shear rates. On the left $T = 22$ [°C] and on the right $T = 37$ [°C]. Data taken from [RL64].

3.5.4 Subfunction Combination

Taking into account Equations 3.4 - 3.6, Equation 3.7 states all eight possible combinations to create a function $\mu(t, h, \varphi)$ out of the subfunctions f_1, f_2 and f_3. Uniting all interceptions with the y-axis into a single variable $c := c_1 + c_2 + c_3$ leads to better readability. The Roman number indicates the number of the function. As a next step, the numerical values for each coefficient will be determined.

$$\mu(t,h,\varphi) = \begin{cases} a_1 e^{b_1 t} + a_2 h^2 + b_2 h + a_3 e^{b_3 \varphi} + c & I \\ a_1 e^{b_1 t} + a_2 h^2 + b_2 h + a_3 \frac{1}{\varphi - b_3} + c & II \\ a_1 e^{b_1 t} + a_2 h + a_3 e^{b_3 \varphi} + c & III \\ a_1 e^{b_1 t} + a_2 h + a_3 \frac{1}{\varphi - b_3} + c & IV \\ a_1 t + a_2 h^2 + b_2 h + a_3 e^{b_3 \varphi} + c & V \\ a_1 t + a_2 h^2 + b_2 h + a_3 \frac{1}{\varphi - b_3} + c & VI \\ a_1 t + a_2 h + a_3 e^{b_3 \varphi} + c & VII \\ a_1 t + a_2 h + a_3 \frac{1}{\varphi - b_3} + c & VIII \end{cases} \tag{3.7}$$

3.6 Coefficient Optimization

Numerical values for the coefficients of an equation that describes experimental observations is done via curve-fitting. Since μ has multiple input variables, multidimensional curve-fitting is needed at this point. There are many algorithms for multidimensional curve-fitting, e.g., the least square method. The quality of their implementation will influence the accuracy of the functions. Because of that, already existing implementations will be used. A table with the data used, can be found in [RL64].

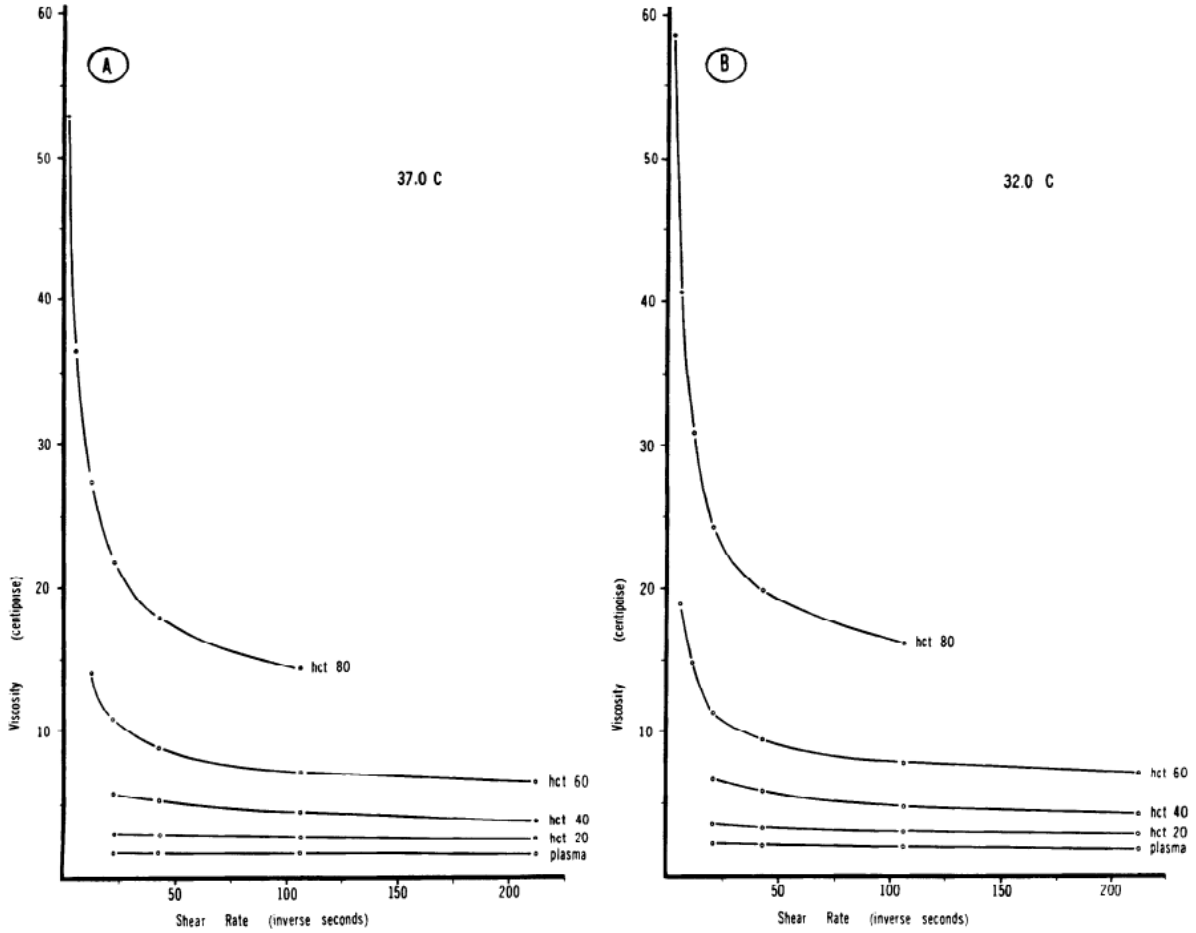


Figure 3.7: Shear rate–viscosity relationship for fixed temperatures and varying hematocrits. On the left, marked with an A the graph corresponds to data taken an 37 [$°C$], on the right (B) to 32 [$°C$].Bad quality due to the quality of the original document. Taken from [RL64].

Before starting to program the optimization process, it makes sense to formulate the mathematical problem precisely. Coefficients a_1, b_1, a_2, b_2, a_3, b_3, c shall be found, so that the sum of all deviations of a function, evaluated at a measurement point, and the actual measurement becomes minimal. Given that all functions describe a real-world process, the coefficients must be real numbers. Formalizing this leads to Equation 3.8 which is equal to Equation 3.9:

$$\vec{a}_i \in \mathbb{R}^7 : \sum_{(t,h,\varphi)\in M} |\hat{\mu} - \mu_i(\vec{a}_i, t, h, \varphi)| \leq \sum_{(t,h,\varphi)\in M} |\hat{\mu} - \mu_i(\vec{b}_i, t, h, \varphi)| \; \forall \vec{b}_i \in \mathbb{R}^7 \tag{3.8}$$

$$\inf_{\vec{a}_i \in \mathbb{R}^7} \sum_{(t,h,\varphi)\in M} |\hat{\mu} - \mu_i(\vec{a}_i, t, h, \varphi)| \tag{3.9}$$

M denotes the set of all 99 measurement points, so $|M| = 99$. $\hat{\mu}$ denotes the measured viscosity at the corresponding temperature t, hematocrit h and shear rate φ. For readability reasons, they are not labelled with an additional index in the equations above, but they

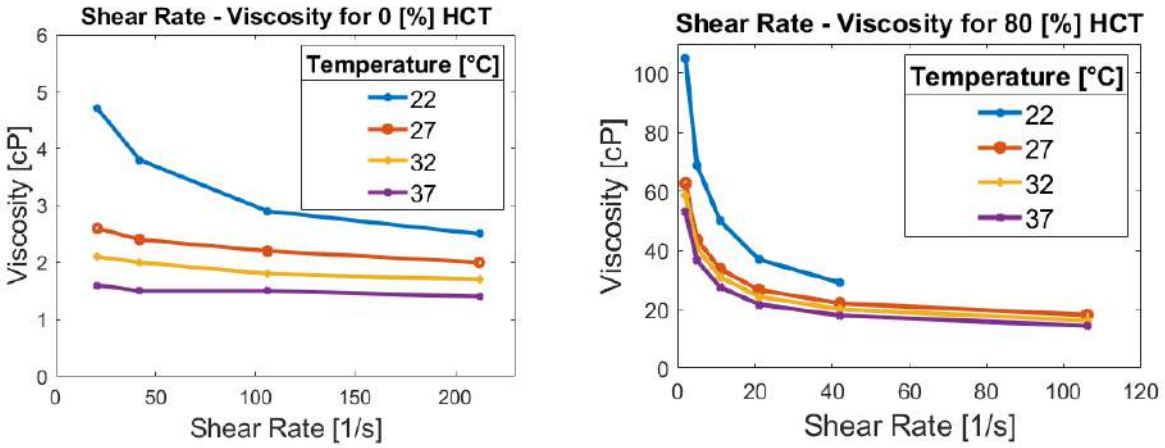


Figure 3.8: Shear rate–viscosity relationship for fixed hematocrits and varying temperatures. Data taken from [RL64].

could for example be labelled with a j going from 1 to 99. $\vec{a}_i := (a_1,\ b_1,\ a_2,\ b_2,\ a_3,\ b_3,\ c)$ $i \in \{1, \cdots, 8\}$ represents the coefficients of the i-th function. In case a coefficient is not needed in (3.7), it is set to zero to make a unique solution possible. It is not obvious that a solution for this problem exists or is uniquely defined. Analysing this would require a detailed mathematical discussion. Since for this thesis only numerical results are of interest, this discussion left out.

For an implementation of the formulated problem, a python script based on the function curve_fit() from the open-source NumPy library is used [sci22] as was suggested in [sta22b]. "Non-linear least squares" (NLLSA) is the optimization method used by this function [sci22]. This algorithm will be discussed in the next subchapter. For traceability the python script used, is given in Appendix A. For the source code of the curve_fit() function, please take a look at [git22b]. Although they should not influence the results, computer hardware, OS and IDE used for running the code are provided for transparency in Appendix B.

As (3.9) suggest, the problem just formulated is a multidimensional minimizing problem. A necessary condition for finding local minima for functions with more than one input variable, is $grad\ \mu_i = 0$. Notice that the derivates with respect to $\vec{a}_i$ have to be formed; not with respect to (t, h, φ). One way would be to use the resulting equations and solve them analytically or numerically, for example with Mathematica. For simple functions, this may even be possible on paper. However, as functions get more complicated, an analytical expression of the derivatives might not be possible. In this case the non-linear least squares algorithm remains which was used in this thesis.

3.6.1 Non-Linear Least Squares Algorithm

In this subchapter, a set of mathematical equations is derived that can be used by a curve-fitting algorithm. The derivation is taken from [Els17]. Finding an analytical expression fit to describe a given set of data is one of the classical use-cases for the least squares algorithm. If it is assumed that the function of the input variable is not simply linear, a more general approach called non-linear least squares (NLLSA) is needed.

For a given function μ and measurement set M, the coefficients $\vec{a}$ of μ shall be chosen in a way that the average error between function and experimental values becomes minimal. Since the following considerations are the same for all eight functions, the index i from (3.8) and (3.9) is neglected here. Consider the residual $r_j := \hat{\mu}_j - \mu(\vec{a}, t_j, h_j, \varphi_j)$ for the j-th measurement point. $\hat{\mu}_j$ is the observed viscosity at t_j, h_j, and φ_j. Now the problem can be formulated as minimizing the sum $S := \sum_{j=1}^{m} r_j^2$ where $m = |M|$. S assumes a minimal value if the gradient for each coefficient in $\vec{a}$ is zero:

$$\frac{dS}{da} = 2\sum_{j=1}^{m} r_j \frac{dr_j}{da} \stackrel{!}{=} 0 \quad \forall a \in \{a_1,\ b_1,\ a_2,\ b_2,\ a_3,\ b_3,\ c\} \tag{3.10}$$

$\frac{dr_j}{da}$ depends on (t_j, h_j, φ_j). Therfore (3.10) might not possess a closed-form solution. But if a start-solution $\vec{a}_0$ is provided, an approximation can be calculated recursively in k steps by using a correction term $\Delta\vec{a}_k$ each time:

$$\vec{a} \approx \vec{a}_k = \vec{a}_{k-1} + \Delta\vec{a}_k \tag{3.11}$$

To derive an expression for $\Delta\vec{a}_k$ an approximation for μ shall be found. For readability, the input variables of μ with an index j are neglected, since this procedure is always the same. Notice that they are fixed values here, depending on the taken measurement. A first-order Taylor polynomial is chosen around the point $\vec{a}_{k-1}$ for each j.

$$\mu(\vec{a}) \approx T_1(\mu, (\vec{a}_k) = \mu(\vec{a}_{k-1}) + \sum_{a \in \{a_1,\, b_1,\, a_2,\, b_2,\, a_3,\, b_3,\, c\}} \frac{d\mu(\vec{a})}{da}\bigg|_{\vec{a}=\vec{a}_{k-1}} (a - a_{k-1}) \tag{3.12}$$

Notice that the derivative $\frac{d\mu(\vec{a})}{da}$ must not be known analytically. It is sufficient to find a numerical approximation for the point $\vec{a}_{k-1}$. One way to do this would be to use the multidimensional difference quotient [BZ22]. No need for an analytical derivation is the advantage of the NLLSA over a classical curve discussion. Now the correction therm for the k-th step $\Delta\vec{a}_k := \vec{a} - \vec{a}_{k-1}$ can be defined. To obtain numerical values for the correction therm, Equation 3.13 is solved. This can for example be done with the Gauss-Newton algorithm [MM11].

$$(J^T J)\Delta\vec{a}_k = J^T \vec{r}_{k-1} \tag{3.13}$$

J is the **Jacobi matrix** [Mül20a] of μ evaluated at $\vec{a}_{k-1}$. Using $\{a_1, \cdots, a_7\} := \{a_1,\ b_1,\ a_2,\ b_2,\ a_3,\ b_3,\ c\}$ the jl-th entry of J can be defined as:

$$J_{jl} = \frac{d\mu(\vec{a}, t_j, h_j, \varphi_j)}{da_l}\bigg|_{\vec{a}=\vec{a}_{k-1}} \tag{3.14}$$

To derive (3.13), consider the following: for each measurement point j, the residual r shall become zero. The residual can be expanded by adding and subtracting $\mu(\vec{a}_{k-1})$ at the same time. This allows for an approximation of the residual with the first order Taylor expansion from above:

$$r = [\hat{\mu} - \mu(\vec{a}_{k-1})] + [\mu(\vec{a}_{k-1}) - \mu(\vec{a})] \approx r_{k-1} - \sum_{l=1}^{7} J_{jl}(a_l - a_{l,k}) \stackrel{!}{=} 0 \tag{3.15}$$

with $r_{k-1} \coloneqq \hat{\mu} - \mu(\vec{a}_{k-1})$ beeing a known numerical value from the previous step. Expanding r_{k-1} into a vector with the j-th residual as the j-th entry and multiplying with J^T provides a set of equations (3.13), with one equation for each coefficient in $\vec{a}$. J_{jl} can be determined by numerical means without actually calculating the corresponding derivative of μ. Hence, (3.13) can be solved for the correction term and the approximation for $\vec{a}$ calculated via (3.11).

Similar to the case of a single input variable, where after $f'(x) = 0$, $f''(x) > 0$ must be fulfilled for x to be a local minimum, the **Hesse-matrix** of $\mu(\vec{a})$ must be positive definite [BZ22]. However, determining the Hesse-matrix will yield the same problems discussed for (3.10). An alternative is to simply try out all possibilities for $\vec{a}$ by evaluating (3.9). Also notice that $Df = J$ requires **total differentiability** of f, see Sentence 5.23 in [BZ22]. This condition is met for most analytical function encountered in engineering.

3.6.2 Numerical Calculations

curve_fit(), like many algorithms of that sort, requires a starting solution from which it starts the optimization process, see last subchapter. Ideally, no matter the starting solution, the algorithm would always return the same result in the end, with different amounts of time needed, if a solution exists. But it can happen that the algorithm gets stuck on a local maximum or minimum, or cannot reach a solution at all in the maximum number of calculation steps set by the user. This means in practice that a good starting solution is key in order to reach good results.

Unfortunately, there is no way of knowing a good starting solution a priori, since this would mean one would already know the answer to the question that is to be answered. If a starting solution shall not be guessed randomly, literature using similar functions or experimental values are needed. Since this could not be done here, a random approach was chosen.

To cover a large interval of numbers with few computation steps, all coefficients were set to the same value. Values were picked from an interval of $\pm 1,000,000$ to ± 0.0000001 in logarithmic steps. Then the curve_fit() function was called with the chosen starting coefficients. It should be noted that this could lead to "jumping over" a good starting solution. Also, it would be better not to set all coefficients to the same value, but to try

out all possible combinations. Notice how this increases the required number of calculations exponentially. E.g., if one tries out the set {-1000, -100, -10, -1, -0.1, 0.1, 1, 10, 100, 1000}, ten steps are required. But checking every possible combination for seven coefficients would require 107 steps. It is easy to see that the formula for the number of combinations is m^n, where m is the number of steps one wants to try out and n representing the number of coefficients of the function to optimize.

Under the assumption that every calculation takes 1 ms and all calculations are performed in a row without further delays in-between leads to $\frac{10^7}{10^3 \cdot 3600} = 2.7\ [h]$ of calculation time for the example of 10 possibilities for 7 coefficients. Since the example is already restricted to one function and a low number of steps, it becomes clear that some smarter approach is needed. One possibility would be to use the concept of the Quicksort-algorithm, where the variable of interest recursively gets compared to the middle and the terminal values of an interval and then sorted in the half that fits best [24]. Another way to think about this would be halving the interval iteratively and logarithmically, similar as described in 3.5.3.

Since the curve_fit() function shows a behaviour that very heavily depends on the starting position, the results were tracked manually in a "Convergence Table". Appendix C shows the "convergence table" for function 1 from (3.7). After finding a solution for four out of seven coefficients, the algorithm seemed not to work properly anymore and gave out random responses for various start values of the remaining coefficients. The rest was determined in MATLAB by brute-forcing different combinations both logarithmically and linearly; the corresponding code can be found in Appendix D. The idea was to look for the best fitting combinations of coefficients in terms of their relative error, as described by (3.9).

It is most likely, that the presented result is only a local solution for the four coefficients and the solver got stuck at that solution. Analysing if a solution is a global maximum or minimum would be non-trivial, since it would require a curve discussion in n dimensions corresponding to the number n of coefficients. Therefore, this is not done here. One possibility could be to solve $grad\ \mu(\vec{a}) = 0$ in, e.g., Mathematica. Since the expressions used so far are not too complicated this approach might work, but looking at the functions proposed by literature, see subchapter 3.3, this might not always work. Using the NLLSA is a workaround to analytically determine the derivations of μ with respect to the coefficients because of the use of a first-order Taylor approximation.

In practice, curve_fit() barely worked, so it was only used for function 1, resulting in a much higher relative error than for the rest of the functions. Using other implementations like the lsqnonlin() function [MAT22] from MATLAB did not work either. For this reason, the remaining coefficients were determined with **"logarithmic iterative interval bisection"** (LIIB), explained in the next subchapter.

3.6.3 Logarithmic Iterative Interval Bisection

Because an algorithmic approach bore no fruits, a brute-force/trial-and-error method was chosen next. Since the issue of required computation time from above still holds true, a "smart" way of trial-and-error must be chosen. Possible coefficients must be checked efficiently, so consider the following:

Every number can be understood as a floating-point number with a sign S, a coefficient C and an exponent B. As a base, 10 is chosen, which leads to $S \cdot C \cdot 10^B$. For example, one could have $+1.47 \cdot 10^{-5}$ with $S = +1$, $C = 1.47$ and $B = -5$. The concept of logarithmic interval bisection is now to cover an interval as large as possible, and then iteratively close in on the best solution. To do so, one should first try to determine S, then B and finally C because this is the fastest way to cut down the size of the interval one needs to consider. This approach defines the LIIB.

To give an example for the LIIB, the optimization of a single coefficient a is considered. $a \in \{-10^9, -1, -10^{-9}, +10^{-9}, +1, +10^9\}$ would be a possible set of starting guesses. Whatever provides the smallest relative error, see (3.9), is then further looked into. If more than one coefficient is considered, the procedure stays the same, but the number of possibilities increases. Assume that -1 delivered the least relative error, then a new combination to try out could be $a \in \{-10^6, -10^3, -1, -10^{-3}, -10^{-6}\}$. If -1 delivers the best rest results again, $a \in \{-50, -25, -1, -0.5, -0.25, -0.1\}$ could be tried next.

This procedure comes with many drawbacks, unfortunately. Most importantly, a local solution could be overlooked. Secondly, trying out more options means exponential grow in possible coefficient combinations and therefore computation time, see above. Thirdly, in practice, it proved to be not always clear which coefficient might be promising. There were functions for which both a positive and a negative number of the same magnitude showed equal relative errors, although one might assume they lead to totally opposite results given these points are opposite of each other on a number scale.

Mathematically speaking, this procedure assumes continuity and monotony of the function. Notice that this must not apply to μ with respect to t, h and φ as input variables, but rather as functions of the coefficients a_1, b_1, a_2, b_2, a_3, b_3 and c. To make any statement on the quality and finite termination of this algorithm, a mathematical analysis would be required. This is however outside the scope of this thesis. Practice showed that this procedure is far from optimal.

Table 3.1 shows the numerical results for the coefficients of each function. Function I was analysed with curve_fit(), functions II-VIII with LIIB. A "-" denotes where a coefficient wasn't part of a function, see (3.7):

Function Number	a_1	b_1	a_2	b_2	a_3	b_2	c
I	-7	9E-03	1.55 E-3	-0.435	-9.9	7	31.04
II	1E-8	-1E17	3.2E-3	8E-3	-9	-1	9.5E-2
III	0.65	-1E-10	0.1	-	1.1	-0.0125	0.7
IV	16	-0.09	0.1	-	30	1.25	0.8
V	-9E-6	-	25E-4	31E-4	0.65	45E-7	1.1
VI	1E-8	-	15E-4	43E-3	75	-1	0.72
VII	-1E-10	-	0.11	-	11.5	-77E-3	1.4
VIII	77E-4	-	0.09	-	-9E-3	12E5	0.95

Table 3.1: Resulting coefficients. Function numbers and coefficients correspond to (3.7). "-" denotes a coefficient that was not part of the function.

3.7 Validation

There are various aspects to validating the results. When an algorithmic method like curve_fit() is chosen, one can enter the resulting coefficients as a new initial guess into the algorithm. The result should not change, otherwise the algorithm did not terminate correctly, for example because too many steps were required. Next, change the coefficients by $\pm 1 - 1000\%$ from the initial value and see if the algorithm returns the initial solution. Ideally, it should do so, no matter how far one sets the coefficients off. In practice, there will be a limit on how far the starting guess can be different from the correct results before the algorithms starts to give different answers. This can happen because it terminates at a different local minimum than before.

Another thing to investigate are mathematical and physical plausibility of the results. Are the resulting poles of the function inside the domain of definition? Do the limits towards infinity and zero match experimental observations or theoretical results from the fluid mechanical perspective? In the case of curve_fit() setting off the solution coefficients even by a little led to completely different results. This was one of the reasons why another method was needed. Given that every function with coefficients from Table 3.1 returns finite values for every t, h and φ combination they provide physically meaningful values. Next task is to evaluate the quality of the results.

An important aspect from a mathematical point, is the type of convergence. For functions, uniform or pointwise convergence is possible. An explanation of the difference can be found for example in [Mül19]. Although the chosen function and determined coefficients may result in a small relative error, it still can happen that the actual values of blood viscosity differ greatly from the result provided by a function determined with the methods of this chapter. This might even happen in a close proximity to points at which an experimental value was taken for which the relative error is close to zero. One can visualize that problem when looking at the Gibbs phenomenon of a non-continuous function, e.g., the saw tooth function, which is to be approximated by a finite Fourier series.

Function Number	Average Relative Error [%]	Maximum Relative Error [%]	Minimum Relative Error [%]
I	182	639	0.64
II	51	120	0.48
III	36	90	<0.1
IV	29	80	1
V	30	83	0.3
VI	27	158	0.12
VII	33	130	<0.1
VIII	38	91	0.3

Table 3.2: Relative errors of the functions. The average relative error is high for all. This is especially true for function I which was determined with curve_fit() function instead of LIIB.

To solve this issue, an analytical discussion of the type of convergence, pointwise or unanimous, would be needed. However, this is impossible without an idea of what the actual function describing human blood viscosity perfectly for all relevant inputs looks like. For this thesis it shall suffice to assume that if the relative error becomes small enough, the function becomes close enough to the real one in the sense that the points in which they considerably differ, becomes a Lebesgue-null-set and as such at least not influences integrals over the function. Again, this is only a mathematical assumption and by no ways certain.

3.8 Evaluation

Once all the coefficients are numerically determined, every combination of t_j, h_j and φ_j at which a measurement was taken in [RL64], will be inserted into each function μ_i with coefficients from Table 3.1. The calculated values $\mu_{ij} := \mu_i(t_j, h_j, \varphi_j)$ are subtracted from the corresponding measurement $\hat{\mu}_j$ and then normalized by diving the result through $\hat{\mu}_j$. The error e of μ_j is then the defined as the sum of the absolute value of all individual errors divided by the number of measurements n. In this case $1 \leq j \leq 99 = n$. The viscosity and thereby $\hat{\mu}_j$ is always greater or equal to 0.

$$e(\mu_i) := \sum_{j=1}^{n} \frac{|\hat{\mu}_j - \mu_{ij}|}{\hat{\mu}_j} \tag{3.16}$$

In Table 3.2 the resulting relative errors $\frac{e(\mu_i)}{n}$ for all functions are provided. Figure 3.9 shows the corresponding box plots. It can be seen how much the error depends on the given data points. Keep in mind that the functions μ_i are four dimensional, three input- and one output dimension, so plotting them in a conventional way makes no sense.

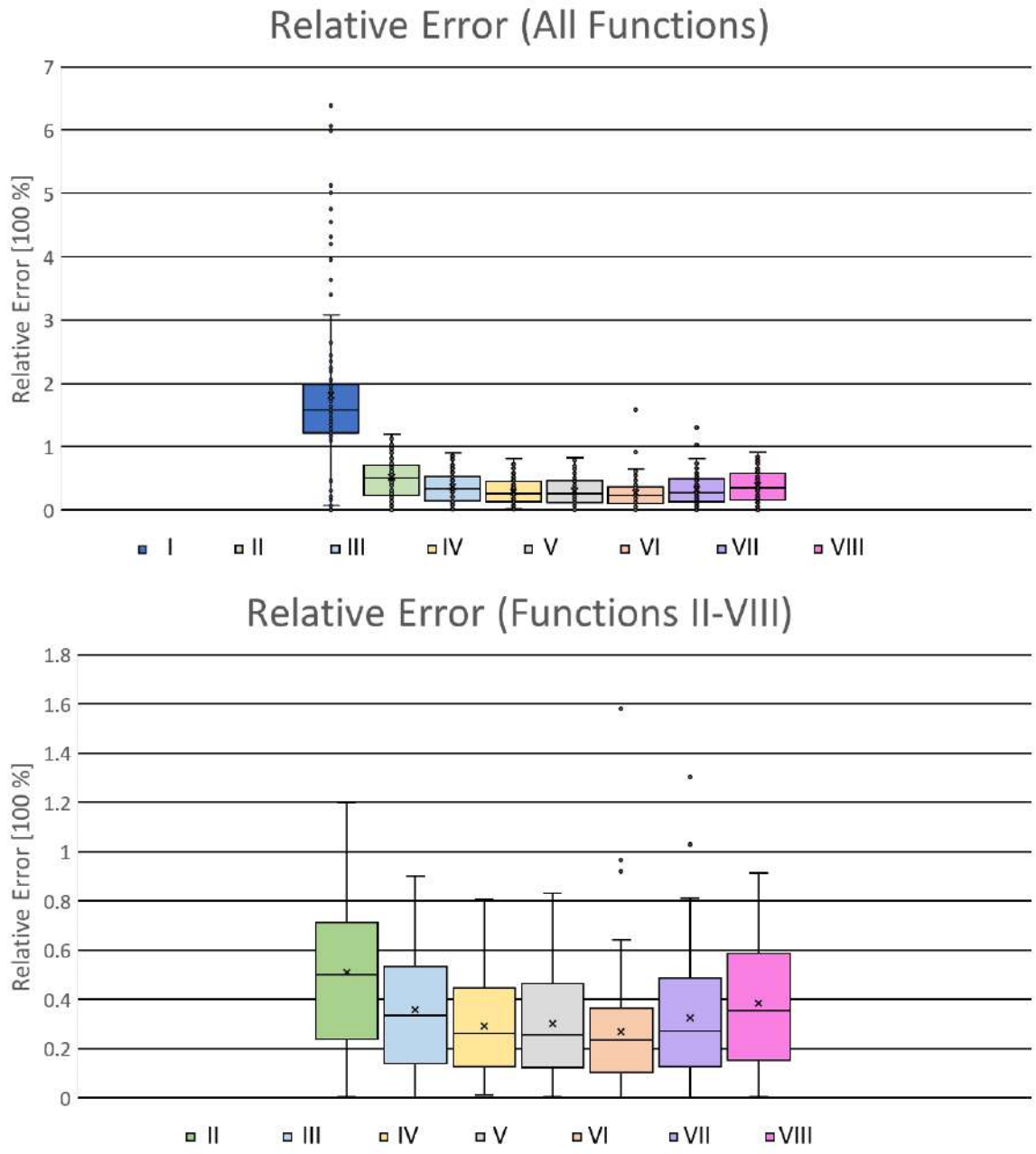


Figure 3.9: Distribution of the relative error of all functions. Function I has higher relative errors due to the different way of calculating the coefficients.

3.9 Discussion

It becomes clear that modelling the viscosity of blood is a highly non-trivial matter and depends on many factors. Most likely it is a good idea to take an approach depending on the application of interest. For the process shown here, many improvements can be made.

3.9.1 Results

Overall, the relative error proved to be quite high. Especially for medical applications, the tolerable error might be 1 % or less, e.g., to ensure a safe treatment. Using the curve_fit() function presented a disadvantage to the manual "logarithmic iterative interval bisection". Reasons for this might be the difficulty in finding good starting guesses for the coefficient. Another reason might be that the functions (with the coefficients considered as input variables) are highly non-linear and not monotone. This might interfere with the algorithm. It can get stuck in local solutions or not even find them at all if the slopes are too steep.

To improve the results, one could use more calculation power to try out more start-solutions and iterate over smaller step-sizes. However, every function has a natural lower boundary on how good the curve can match the measurement points. This lower bound could be higher than the tolerable relative error. This because the type of the viscosity function is unknown, and the assumed type might be off. Figuratively speaking, $ax + b$ can only match $c \cdot sin(dx) + e$ so well.

From the function considered in this thesis, IV seems the most promising in terms of its relative error. This would mean that temperature dependency is linear, hematocrit quadratic and the shear rate behaves like a single-pole expression.

3.9.2 Dataset

Interpolated functions can only be as good as the data used for optimizing its coefficients. So, the user of a function must trust that the data was generated under standardized conditions. This would allow judging if the conditions are similar enough for the application at hand, or if the results can be used after applying a correcting factor. The calculated coefficients are only valid for the conditions at which the data was generated, if a change in these conditions cannot be proven to be irrelevant. This is not a critic of the data set provided in [RL64], but a general warning. It is always important to lay out under which environmental settings the data set was collected.

To achieve better results, the most obvious approach would be to acquire more data points. The set used here had around 100 data points, which is not much because a function depending on three parameters must be interpolated. The degrees of freedom are the number of coefficients, not the number of input variables. Also, the step size in the data set is quite high for temperature ($5 - 10$ [$^\circ C$]) and hematocrit (20 [%]). Smaller step sizes (e.g., 1 [$^\circ C$] and 5 [%]) and therefore more data points, would be desirable. [RL64] is from the year 1964. Progress made in the construction of measurement devices and sensors might deliver more accurate measurements of the viscosity. A direct method of measuring φ might prove to represent the in-vitro behaviour of blood more accurately.

The mean errors of the measured viscosity in [RL64] are extremely high in some cases. In fact, the mean error itself seems to be depending on t, h and φ. One could use the same process again to define functions with mean errors as an output. A dataset with the highest mean error around 10 % of the original value would be desirable. Then the additional function for the mean error might be negligible and each function could be said to be accurate in a range of ± 10 %. This would imply that the error gets transferred linearly in the optimization process, which also would have to be mathematically proven.

A way of overall improvement would be to search for more data in the literature. If the data already used is not to be replaced but to be amended, one must look out for compatibility of the data sets, meaning, where the laboratorial conditions close enough to each other to make the data comparable. The other approach would be to conduct the experiments on one's own, which would take time and resources not available during this thesis.

3.9.3 Assumed Function Types

Besides shear rate, hematocrit and temperature, the viscosity of blood depends also on other factors, as discussed in the introduction to this chapter. These factors would also need to be considered in the formula of the viscosity functions in (3.7) to achieve a more precise model. Trying to implement more variables also means that the experiments for generating data become more complex, as more sensors are needed. Also, a bigger data set is required to optimize the coefficients properly.

Instead of trying to determine an analytical function for the viscosity a priori, and then try to optimize its coefficients, one could also use tools like machine learning to determine the functions a posteriori. Plotting the graphs and judging by eyesight what function might fit, can lead to grave errors. The scaling of the axis alone can render any interpretation of the graphs useless. Not determining the functions a priori might have the advantage that a computer could detect interdependencies between the variables that are not known to the user or numerically find a better fitting function to represent a dependency.

Many simple analytical functions, limited to a specific domain, can assume similar values, although their expressions are very different. For example, the exponential decay regarding the shear rate can be modelled either by e^{-x} or $\frac{1}{x}$. Since negative shear rates have not been included in the data set, we cannot investigate this by simply looking at some negative values. In other cases, negative values etc. might have no physical meaning and therefore cannot be measured in the first place.

Comparing the numerically determined function with data points used to calculate the coefficients is a bad approach from a scientific level. In practice, all the data collected for optimization will be used, assuming the function gets better that way. Then the calculated model will be used to make predictions for which no data exists. For scientific validity, it would be better to take out some data points a priori and then use them for comparison to

judge how accurate a function is. Because the dataset was already small, no points were taken out.

One of the main issues is assuming that the influence of each input variable is independent of the other ones. A viscosity function cannot simply be constructed by adding three independent subfunctions. This can be seen in Fig. 3.3 and Fig. 3.8, where the y-axis intersection does not scale linearly with shear rate and temperature, respectively. E.g., for a fixed temperature, the decay of viscosity by increasing shear is hematocrit dependent, meaning that $f_3(t)$ should rather be modelled as $a_3 e^{b_3(h)\varphi} + c_3$ where the coefficient b_3 is a function of h. But using more complicated functions will most likely drastically increase the size of the dataset needed for determining accurate coefficients. Computation time will increase and numerical calculations will be prone to more errors. Therefore, this was not done in this thesis.

Different functions might provide a better fit. Especially the chosen terms for the shear rate are rather simple compared to what can be found in literature. In addition, their coefficients were the last ones to converge when using the "logarithmic iterative interval bisection". Using (3.1) and (3.2) might give better results. Using multiple algorithms, different implementations of those and various datasets would make for an interesting approach to dive deeper into the topic. Multilayer models or a posteriori machine learning based methods could lead to more accurate results. This remains however a research topic for another thesis as it is now time to move on and not only model blood but an entire vessel.

4 Fluid Mechanical Computer Simulation

Measuring physical parameters like shear rate, pressure or velocity of blood in vivo is not possible. Still these quantities are needed for a medical or biological analysis of experiments related to thrombus formation, performed on human or non-human blood vessels. One approach to attain these values is simulating the experiment on a computer and combine it with the other data, e.g., images and videos recorded with a microscope.

The usual procedure of the experiments to emulate is as follows : The gut of a rat or mice is extracted from the anesthetised (but living) animal. The body fat is removed from gut and blood vessels connected to it. Individual blood vessels are loosened from the outside of the gut. They are placed on a microscope chip and fixated with tape or glue. The vessel is injured manually by a person using a clinical needle, e.g., from syringes. In literature this injury is sometimes also conducted with lasers [STWa13]. Since the vessel is under a microscope and the components of the blood have been coloured with antibodies, one can now observe the bleeding out of the small hole in the vessel wall. For documentation and analysis, everything is recorded as images or videos with a fluorescence confocal microscope. In this way, one can observe platelets forming the thrombus to stop the blood flow of the injury.

From this it becomes clear what the simulation needs to depict: the vessel and the blood flow before the injury, as a reference, and the appearance of a hole inside the vessel wall and the flow conditions afterwards.

Take note that at this stage, it is not the goal to simulate the biological process of thrombus creation. Rather, a fluid-mechanical description of the situation is desired. As was discussed in Chapter 2.2 this implies that the results are only reliable for the immediate aftermath of an injury, since the vessel changes is geometry by contracting and closing the hole bit by bit with a thrombus. Please refer to Chapter 4.8 for a way on how to take the thrombus into account, nonetheless.

As a first step, a software designed for simulating fluid mechanics only is required. **COMSOL** Multiphysics by the company COMSOL [Com22b] was chosen for its usability and transparency concerning which equations are solved for a given model. Other alternatives would be SIMULIA from the company DASSAULT Systems (also commercial) [3DS22], or the open-source project SPlisHSPlasH [GIT22a].

Since commercial products have always some intransparencies about, e.g., their numerical solver methods, open-source projects are preferable from an academic point of view. Also, it makes the results better reproduceable for the scientific community since no paywall exits to recreate the simulations. Their back-draw might be a lesser overall quality of the software, e.g., usability or the necessary hardware power for acceptable computation time. For the reason just stated, a commercial environment was chosen.

4.1 About COMSOL

COMSOL Multiphysics, from now on referred to simply as COMSOL, is a graphics-based software tool for simulating various physics on a system or device level. As mentioned above, it was chosen because it displays the equations solved for a given problem and its easy-to-understand GUI. For a discussion of the equations solved in this set-up, please refer to Chapter 2.1. Furthermore, COMSOL allows for an automatic mesh generation, which saves a lot of time when doing rapid prototyping. To reproduce the simulations described below, a license for COMSOL Multiphysics and their Microfluidics expansion pack is needed. The simulation software is an important detail, given how it decides over the solved equations and the implementation of the numerical methods used. Version 5.6 of COMSOL together with the already mentioned expansion pack were used. For a detailed discussion of version history and their numerical implementation, please refer to their website. Although it should not influence the results, Appendix B states the OS and PC hardware used.

4.2 Model Specification Sheet

Before starting to develop a simulation model, it should be discussed what the model is supposed to achieve and how this is supposed to do that. This is not specifically needed for this thesis but rather always advisable in terms of project management. Some of the questions that need answering are, divided by category:

A. Description of the Problem / Goal of the Simulation

   1. What is the system / observation / experiment in the real world to be simulated?

   2. What is to be achieved with the data generated by the simulation?

   3. What will be the input data for the simulation and what (data-) format is required for the output?

   4. What parameters of the simulation will be fixated, which need to stay easily changeable?

   5. What are the key factors for achieving these goals, what must not be taken as seriously?

   6. What results are we expecting and why?

   7. What numerical values for both input and output already exist in literature?

B. Usability

1. Who is supposed to use the simulation?
2. What kind of background knowledge is required of the user?
3. How is the interface / GUI supposed to look?
4. Is the simulation to be published or just for internal use?

C. Translation of the experimental into a mathematical problem

1. What equations describe the system of interest?
2. What kind of aspects from natural sciences play a role for the problem? How can they be translated into mathematical equations?
3. Which fields of mathematics apply here?
4. Are these equations valid for the given physical circumstances? Especially are some specific mathematical theorems, e.g., the divergence theorem, used and their respective requirements met?
5. Does the problem possess a solution at all? Is this solution unique?
6. What are the border values?
7. Does the uniqueness or existence of a solution get lost if parts of the problem change?

D. Hardware

1. What are the minimum hardware requirements for the simulation?
2. How much money will a device with that hardware specifications cost?
3. How much computational time should one simulation take?
4. Shall the simulation run on a local PC or a cluster?

E. Numerical Accuracy

1. What range of numerical error is acceptable for which results / input variables?

2. Can the chosen algorithm achieve this accuracy for the given hardware and time restrictions?

3. How does numerical accuracy correlate to computation time exactly? Quadratic, logarithmic, exponential. . . ?

F. Algorithms

1. What algorithms are to be used for solving the defined mathematical problem?

2. Does the algorithm converge theoretically and practically?

3. How can these algorithms be implemented?

4. What will be the converging criterion for the algorithm, e.g., for which norm and pointwise or uniform convergence?

5. How can the result of the algorithm be validated?

6. What alternatives exist to the chosen algorithm? What are the differences? Why choose a specific algorithm?

7. Which tests must the algorithms pass to be considered valid? Construct such tests.

G. Software

1. What software offers an implementation of these algorithms?

2. Can one use pre-existing software to fully simulate our problems, or can one tweak some existing software into doing what one wants?

3. What alternative exists to the software one wants to use? What are the differences? Why choose a specific software?

4. Which tests must a Software pass to be considered validate? Construct such tests.

H. Development

1. How many people are tasked with creating the simulation?

2. How will they be organized? Who is project lead? Who is responsible for which aspects of the simulation?

3. What are their educational backgrounds?

4. Has something equal / similar already been done? Can the source code or something similar be apprehended? Can related literature be found? How did they approach the task?

5. How long is development supposed to take? What is the deadline? What can be useful milestones? What are the deadlines for the milestones?

6. Is the simulation developed internally/externally or in a hybrid form?

I. Verification and Validation

1. How and by whom will be decided if the results of our simulation are satisfactory?

2. Can the simulation be cross-checked with real world experiments? How will these experiments look like and how accurate will they be?

3. How do the actual results differ from the expectations and why?

4. Which tests must a simulation pass to be considered valid? How can such tests be constructed?

5. Document the in vivo experiments that are to be recreated with the simulation.

6. When is the deadline? How much time shall the entire project take? How can it be achieved?

7. What are the technical limitations of the software used, and why can the goal still be achieved goal?

Many aspects are to be considered additionally when managing a similar project on a larger scale. This thesis is in a way a "commissioned work". It serves the purpose of enabling biomedical research into thrombus formation. An accurate description of the present shear rate is the key variable to be depicted by the simulation. No further adjustments to COMSOL were made and it met any of the requirements mentioned above anyhow. Still, discussing this point is important in case a self-implementation of the experiment is desired at a later stage of this project, of which this thesis only marks the beginning. Below a list is provided with answer to the questions from above. Many questions are irrelevant for this thesis.

A. Description of the Problem / Goal of the Simulation

1. Fluid mechanical aspects of the biomedical process of hemostasis.

2. Numerical values at given locations inside a vessel that can be referenced to a corresponding animal experiment.

3. Input: vessel geometry, diameter of the wound, pressure inside the vessel. Output: velocity and pressure profile throughout the vessel, but most importantly the local change in shear rate around the wound.

4. The physics / equations to be solved always stay the same. So do the numerical coefficients in these equations, see for example Chapter 4.4. The parameters stated under point 3. above need to be easily interchangeable as they will be different for every animal that serves as a reference.

5. The shear rate is key. However, its order of magnitude is more important than a precise value, so errors of up to $\pm 900[\%]$ are acceptable.

6. See Chapter 4.6.

7. See Chapter 2 and 4.6.

B. Usability

1. PhD students with a degree in the area of biology, medicine or a related subject.

2. Once simulation can be confirmed to be finished, little knowledge in the area of mathematics, physics or fluid-mechanics shall be required. But at this stage the simulation has many errors, see below. Hence, for now an understanding of the mathematical fundamentals (engineering bachelor level or equivalent) and the fluid-mechanical theory explained in Chapter 2.1 are mandatory.

3. Does not apply here. COMSOL has a quite user-friendly GUI; even for people without a technical background, it is easily accessible.

4. Internal use only, especially given the complications that still occur at this stage.

C. Translation of the Experimental- into a Mathematical Problem

1. Most importantly the Navier-Stokes equations and the formulars for non-Newtonian fluids. For details, please refer to Chapter 2.1 and the beginning of Chapter 3.

2. Physics, especially fluid-mechanics. Biology, especially the anatomy of blood vessels and hemostais. Please refer to Chapter 2 for a more detailed explanation.

3. Mostly what is covered in the subjects Analysis 1 to 3 in the German mathematics bachelor curriculum, which is pretty much the same for any German

university. In the Anglo-Saxon system, they are referred to as Calculus.

4. Yes, because this thesis does in no way enter the field of quantum mechanics or any other area where the classical Newtonian physics are no longer valid. Everything can be described as real valued object inside the finite dimensional and well explored $\mathbb{R}$-vector space $\mathbb{R}^3$. Especially a discussion of convergence was neglected for this reason in Chapter 2.1. Norms are equivalent here. Furthermore, all functions were said to be continuous and infinitely often differentiable. This can be reasoned with every function needing to follow the law of energy and momentum conservation. In case of Chapter 3, a discussion of pointwise vs. uniform convergence would be desirable, but was left out due to keeping the discussion on a more practical level. The point can be made that for a finite measure space uniform convergence aside from a Lebesgue null set is sufficient since all variables of interest, for example pressure, are integral properties.

5. There is no reason to believe that a solution either does not exist or is not unique, as long as all input variables are chosen in a physical meaningful way and the simulation is running correctly.

6. See Chapter 4.4 below.

7. No indication to assume this can happen. Of course, the simulation aims only to depict meaningful values for the input variable, e.g., not a blood vessel with a diameter of 1 km.

D. Hardware

1. Any ordinary laptop that is capable of running more than just a web browser should suffice. That might however require putting the simulation on a cluster system at a later stage of the project.

2. A device with the equivalent power of a laptop costing $1000 - 2000\,€$ in winter 2021 should be capable of running the simulation locally. Keep in mind that laptops tend to be more expensive than PCs with similar hardware power. A laptop costing $500 - 1000\,€$ should be able to run it with the help of a cluster. It is assumed since PCs/laptops are globally available that regional price differences in general are negligible.

3. One to fifteen minutes. Once the model reaches a more sophisticated state and is implemented in 3D instead of 2D, even 1-3 days on a cluster should be reckoned with.

4. This entire thesis was done on local hardware. At a later stage, using a cluster would be advisable.

E. Numerical Accuracy

1. See Chapter 4.6.

2. Since the details of the solving algorithms and its implementation are a company secret of COMSOL this question cannot be answered. For the scope of this thesis, COMSOL is trusted to provide accurate results if everything is set up correctly.

3. See answer to the question above.

F. Algorithms

1. See above and Chapter 4.4.

2. See above.

3. See above.

4. See above.

5. See above.

6. See above.

7. See above.

G. Software

1. This thesis uses COMSOL.

2. This discussion is the subject of this chapter.

3. Alternatives to COMSOL were listed at the end of Chapter 4.1.

4. They must match the animal experiments already mentioned. Another way of cross-checking would be to use the micro-fluidic experiments proposed in the next chapter.

H. Development

1. Does not apply for this thesis. Only the author was working and responsible for the model presented here. Of course, this happed on the basis of regular discussion and input with colleagues both from LMU and TUM.

2. See above.

3. Electrical engineering. It might be desirable to create a more interdisciplinary team of mathematicians with specialization in differential equations and numeric, physicists or mechanical engineers with specialization in fluid-mechanics and biologists or medical professionals with advanced knowledge on the field of hemostasis. At least the latter was present in the form of Mr. Rossaro and the LMU team. Informatic students with advanced knowledge in implementing numerical solvers for differential equations would be useful if one decides to create a simulation tool that can also account for the thrombus formation etc.

4. Not to the knowledge of the author. Papers related to the subject tend to not explain precisely how their simulation was set up. Examples for this are [SHBa18] [TFTa19] [SJTa14] which do not mention if blood was modelled as a non-Newtonian or Newtonian fluid. But this seems to be one of the key factors when simulating blood accurately. An example of a more transparent description can be found in [JGWa16].

5. A master thesis in general in the German university system takes 6 months. The task was not fulfilled in a satisfactory manner, see Chapter 4.8 Discussion.

6. Does not apply here.

I. Verification and Validation

1. The medical department of the LMU is the controlling instance here.

2. As already mentioned, animal and microfluidic experiments can be used for validation. Not done here.

3. See Chapter 4.8.

4. Does not apply here.

5. See Chapter 2.2.

6. As mentioned, the practical work on this thesis was conducted in 6 months, excluding the process of writing. Internal milestones were used but are not outlined to keep this chapter compact.

7. As was discussed in Chapter 2.2 biological aspects relevant to the process of hemostasis, like the contraction of the vessel, can barely be modelled in COMSOL which is intended to simulate inorganic materials.

4.3 Expectations

Purpose of this thesis is to model the biomedical process of haemostasis. Thus, the benchmark are the expectations and observations of the medical department of the LMU. Mr Rossaro provided Figure 4.1 below displaying the expected velocity, shear rate and pressure. He also stated that one should "keep in mind that we do not know exactly which exact numerical value we should expect, but most likely there will be a huge variance in vivo. That is why approximations or ranges of values are fine." To be precise, the issue is the impossibility to measure these variables in vivo precisely, see also "apparent viscosity" in Chapter 2.2. This is exactly the reason for conducting this thesis in the first place. However, from an engineering perspective this is problematic since a numerical validation of the experimental results remains difficult.

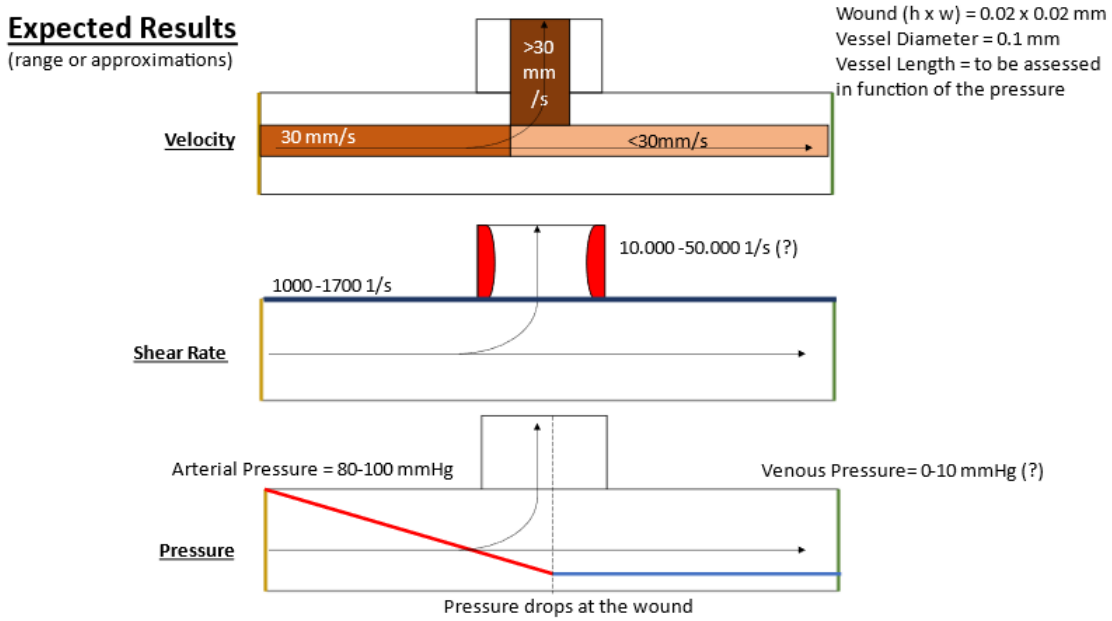


Figure 4.1: Expected velocity, shear rate and pressure results. Graphic provided by Mr. Rossaro.

4.4 Model Description

As COMSOL was chosen as a simulation environment, it makes sense to follow the workflow imbedded in the software.

4.4.1 Physics Interface

After choosing a simplified model setup, a 2D space domain is chosen. Next creeping flow is chosen as a physics interface. This was reasoned in Chapter 2.1 with the evaluation of the Knudsen Number. COMSOL additionally states that "very low Reynolds numbers" are necessary. This condition might not necessarily be true throughout the entire circulatory system, again please look at Chapter 2.1 for the corresponding discussion of the Reynolds number. On the other hand COMSOL recommends this physics interface for "small geometrical length scales (for example, in microfluidic and MEMS devices". Finally, a time-dependent study was picked. All of the quotes in this paragraph stem from the GUI of COMSOL 5.6.

4.4.2 Geometry

Next, let's discuss the geometry of the problem. It is quite simple: a main channel, representing the blood vessel, is connected via a smaller side channel, representing the wound, to a big reservoir representing the environment. The side channel is connected orthogonally with respect to the direction of flow. As already mentioned, no connective tissue is considered. Instead, the height of the side channel represents the distance from ECs to air. In-between would also be, e.g., skin.

The vessel is modelled with a finite length, but should be big compared to the vessel diameter. Also, one has to make sure that the length is long enough so that a fully developed flow is present at the side of the injury. Concrete numerical values will be provided later; for now, a general description shall be enough. Ideally, the "reservoir" representing the environment outside the patient would be infinitely large; in practice, each side should be significantly longer than the vessel length.

For the sake of simplicity, all walls are assumed to be perfectly even and not moving. Vessel and wound are depicted as ideal cylinders; the reservoir as a very large cuboid. The set-up has many symmetrical aspects, for example the z-axis symmetry for the vessel if the wound is closed. To further simplify the problem and save computation time, it is transformed into a 2D problem by cutting along the xy-plane. This implies that the vessel is centred around the z-axis and the wound-cylinder around the x-axis. A Schematic of the 2D set-up is provided below in Figure 4.2. Next to discuss are the geometric variables that remain to be determined. Vessel length and height were already mentioned. $(0, 0)$ is arbitrarily chosen as a position for the lower left corner of the vessel rectangle. They uniquely determine the position of the wound, which is said to have its centre at half the vessel's length. The same applies to the position of the reservoir, this time with respect to half the wound width. Again, height and width are needed for the wound and the reservoir. All the needed heights and widths are depicted in Figure 4.3:

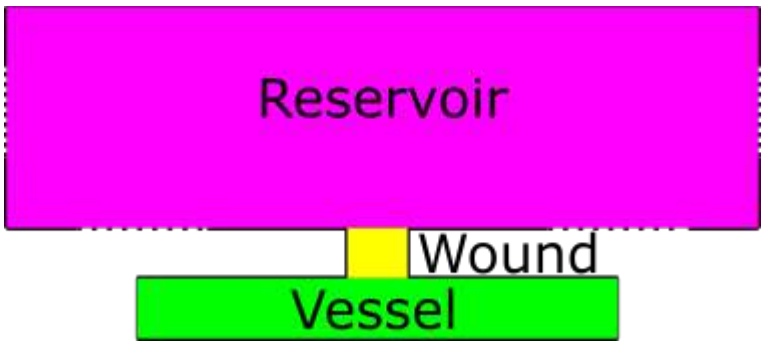


Figure 4.2: Schematic of the separate geometric domains used for setting up the simulation. Green denotes the vessel domain, yellow the wound and pink the reservoir. The dotted lines hint at the fact that the reservoir's width and height are significantly longer than portrayed here.

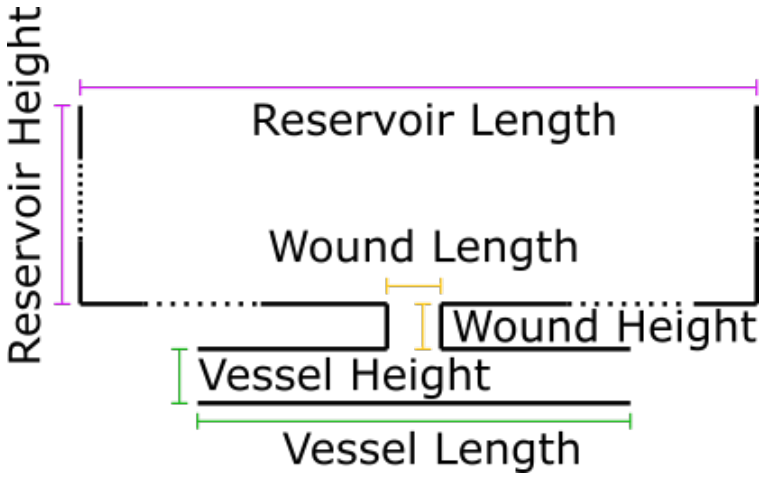


Figure 4.3: Geometric degrees of freedom: the widths and lengths of vessel, wound and reservoir. Colouring chosen to match Figure 4.2.

For a fluid-mechanical simulation there is one remaining thing to consider on a fundamental level: which edges function as inlet, outlet or rigid wall of the set-up. Inlet and outlet in terms of graph theory can also be understood as source and sink. For the source, it is assumed that an infinite amount of fluid is available at any given time instance. Similarly, the sink has an infinitely large fluid storage capacity without effects like backlog. How much fluid actually flows is determined by the border conditions, discussed later.

The set-up is as follows: one source at the left edge of the vessel rectangle, symbolizing the incoming blood flow. This flow is denoted as inlet in Figure 4.4 below and assumed to be ideal. Everything that biologically happens before the short vessel section represented by the vessel rectangle gets neglected, e.g., the heart. This simplifies working on the model quite a lot and saves required computational power and -time. Analogue two sinks can be found at the top, left and right edge of the reservoir and the right edge of the vessel.

When considering the case of an intact vessel, every edge not mentioned so for is considered impermeable. All incoming fluid disappears in the sink on the right side of the vessel. Once the wound is simulated to be open, the lower and top edge of the wound rectangle become permeable. As a last step before talking about the numerical values used for the simulation, a discussion of the fluidic domains is needed. Please again take a look at Figure 4.2. The geometry is separated into three domains: vessel (green), wound (yellow) and reservoir (pink). They correspond to how the set-up of fluid domains was chosen, each with its own viscosity, see Chapter 2.1. Inside the vessel domain, a power-law model simulates the shear-dependent viscosity of blood. The reservoir is modelled to have similar properties to air, so a constant viscosity. The vessel domain is set up as a non-Newtonian Power-Law fluid, on the other hand the environment is modelled as a Newtonian fluid which is supposed to simulate air. For the setup of the wound domain, please take a look at the following paragraph.

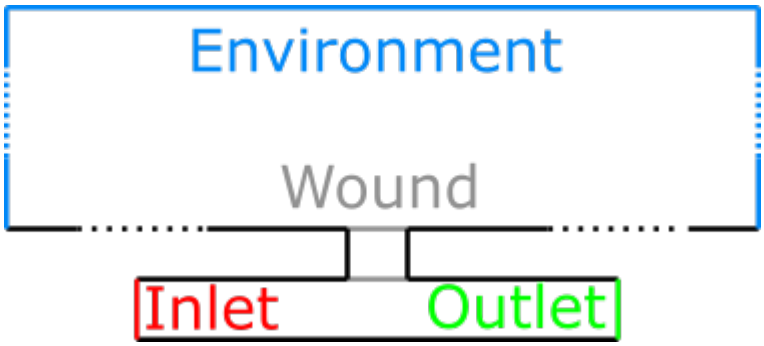


Figure 4.4: Source, sinks and walls of the simulated experiment. The walls of the wound are coloured grey to denote their permeability once the wound is open.

4.4.3 Wound Opening

To depict the opening of the wound, a trick suggested by COMSOL was used [Com22c]. At the beginning of the simulation, t=0, the viscosity of the wound was set as high as possible to make the wound impermeable since a high viscosity means that no fluid can pass through a certain area, at least not with very high pressure or force pushing it, which of course is not applicable here. After a certain amount of time has passed, $t = t_0$, the viscosity of the wound domain becomes the same as for the vessel domain, simulating an (instantaneous) blood flow through the wound. The mathematical function used for this was: $1000 \cdot step(t[s] - 0.5[s]) + 4\ [cP]$, where step() is 0 for $t \geq 0$ and 1 for $t < 0$. In COMSOL this function was said to have two continuous derivatives and a transition period of 0.001 $[s]$. These to additions to "smoothen" the edges of the step function make calculations easier for the numerical solver.

4.4.4 Input Values

After describing all the fundamental aspects of the simulation, this subchapter lists the numerical values used while running the simulation:

- Vessel Length: 1 $[mm]$
- Vessel Height: 200 $[um]$
- Wound Length: 60 $[um]$
- Wound Height: 1 $[um]$
- Reservoir Length: 20 $[cm]$
- Reservoir Height: 20 $[cm]$
- Inlet Pressure: 126 $[mmHg]$
- Outlet Pressure: 125 $[mmHg]$
- Environmental Pressure: 1 $[bar] = 750$ $[mmHg]$
- Viscosity Vessel Domain (Blood): 4 $[cP]$, p.223 [BLSa19]
- Density Vessel Domain (Blood): 1060 $\left[\frac{kg}{m^3}\right]$ [CJ98]
- Power-Law Index (Vessel Domain): 0.9 [NN12]
- Viscosity Environmental Domain: 0.0186 $[cP]$ [Edg22]
- Viscosity Wound Domain: $1000 \cdot step(t[s] - 0.5[s]) + 4$ $[cP]$, where step() is 0 for $t \geq 0$ and 1 for $t < 0$.
- Temperature in Every Domain: 20 $[°C]$

The pressure difference between Inlet and outlet was chosen small because only a small section of a vessel is simulated. Even with a wound, there is no reason to believe that pressure would drop significantly. A similar set-up could be found on the COMSOL demo library [COM22e]. Every length must be chosen finite. Ideally, the Reservoir would be a lot bigger than the dimensions of the vessel. Looking at the numbers above, the vessel length is bigger than the width by a factor of 200. The reservoir-to-vessel length ratio

is hence 1000. It is important to place the centre of the wound away far enough from the inlet and the outlet to have a fully developed flow. This was achieved by placing the wound was at half the vessel length.

4.4.5 Solver

A time duration of 2 $[s]$ was simulated in steps of 0.05 $[s]$. No details on how the solver operates could be found.

4.5 Additional Equations

COMSOL uses a couple of additional equations at the inlet, outlet or on the edges marked as a rigid wall. They shall be briefly mentioned here. At the inlet and outlet of the vessel, a fully developed flow with an average pressure is set as a boundary condition. The value of the average pressure can be found above. The following two equations are solved here:

$$\mathbf{u} \cdot \mathbf{t} = 0 \tag{4.1}$$

which denotes that the velocity vector has no tangential component at the inlet. The second one is:

$$[-p\mathbf{I} + \mathbf{K}]n = -p_{grad}\mathbf{n} \tag{4.2}$$

Same applies for the outlet of the reservoir. For the wall, "no slip" is set as a boundary condition. The velocity becomes zero, meaning $\mathbf{u} = \mathbf{0}$. The initial velocity and pressure values for the entire domain are set to 0. For the fluidic properties, the following equations are solved:

$$\rho\frac{d\mathbf{u}}{dt} = \nabla \cdot [-p\mathbf{I} + \mathbf{K}] + \mathbf{F} \tag{4.3}$$

which corresponds to the Navier-Stokes equations, see Chapter 2.1. Next is:

$$\rho\nabla \cdot \mathbf{u} = 0 \tag{4.4}$$

which corresponds to the equation for a compressible fluid. This is a simplification for the blood domain. Finally, the following equation is solved:

$$\mathbf{K} = \mu_{app}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right) \tag{4.5}$$

Which seems to be a transition term for the velocity vectors going from the non-Newtonian into the Newtonian fluid domain. For the Power-Law model, COMSOL solves:

$$\mu_{app} = \mu_{blood}\left(\frac{\varphi}{\varphi_{ref}}\right)^{n-1} \tag{4.6}$$

$$\mathbf{S} = 0.5\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right] \tag{4.7}$$

$$\varphi = max(\sqrt{2\mathbf{S} : \mathbf{S}}, \varphi_{min}) \tag{4.8}$$

The lower shear rate limit φ_{min} was set to 0.01 $[s^{-1}]$ the reference shear rate φ_{ref} to 1 $[s^{-1}]$

4.6 Mesh Validation

When performing a fluid mechanical simulation, one should make sure that the results are independent of the chosen mesh. This can be done by choosing ever finer meshes and check if certain values of the system do or do not change significantly. For example, one could look at the total energy of the system. Because COMSOL assigns the mesh automatically, it is assumed here that the chosen mesh size is good enough.

4.7 Results

The results of the simulation can be seen in the Figures 4.5-4.8 below. Screenshots were taken for $t = 0.45$ $[s]$ and $t = 0.55$ $[s]$ so shortly before and after the wound opening. Pressure, velocity, velocity direction and shear rate can be seen. It can be seen that the results are completely off. The reason for this is not exactly clear. The main issue seems to be the small pressure gradient, which causes virtually no flow in the vessel. On the other hand, there is no reason to assume that over the distance of 1 $[mm]$, even with a wound, the pressure would change drastically. When setting flow rates as an inlet condition, with values as to expect for a vessel of a similar size, COMSOL calculates pressure values at the inlet which are so high they would normally damage the vessel. As a result of the pressure inside the vessel being considerable lower than environmental pressure, fluid was flowing into the wound, as can be seen on the bottom of Fig. 4.7. This is nonsensical, but on the other hand the pressure conditions are representing in vivo conditions, with the atmospheric pressure close to the earth surface being 1 $[bar]$. Below are stated some possible improvements, which might lead to better results.

4.8 Summary

Many improvements need to be made in the COMSOL model. One of them would be to use a two- or multi-phase flow model to simulate blood leaving the vessel and flowing into the environment. Additionally, the Fahraeus-Lindqvist effect could be included in this way. For larger blood vessels, RBC rouleaux could be modelled as gas bubbles. Connective tissue, especially muscle tissue, could be included to depict the vessel contraction after the appearance of an injury. Blood should also be modelled as a compressible fluid. Please refer to Chapter 2.2 for all the additional neglections made. A way to include the thrombus influencing the flow could the **Brinkman equations**, p. 1827 [SJTa14].

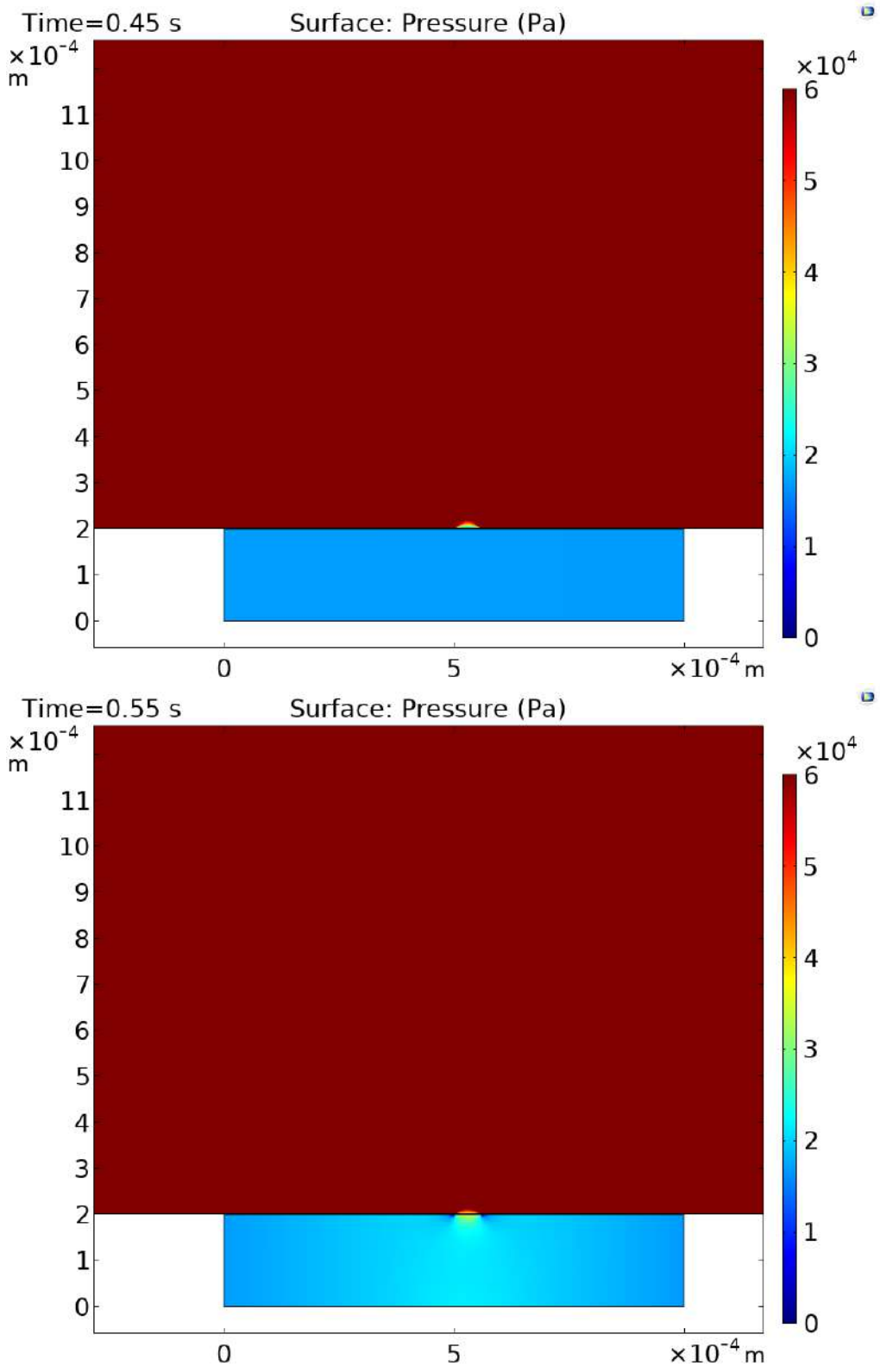


Figure 4.5: Results of the COMSOL simulation. Pressure throughout the system is plotted for $t = 0.45$ $[s]$ (top) and $t = 0.55$ $[s]$ (bottom). Opening of the wound happens at $t = 0.5$ $[s]$.

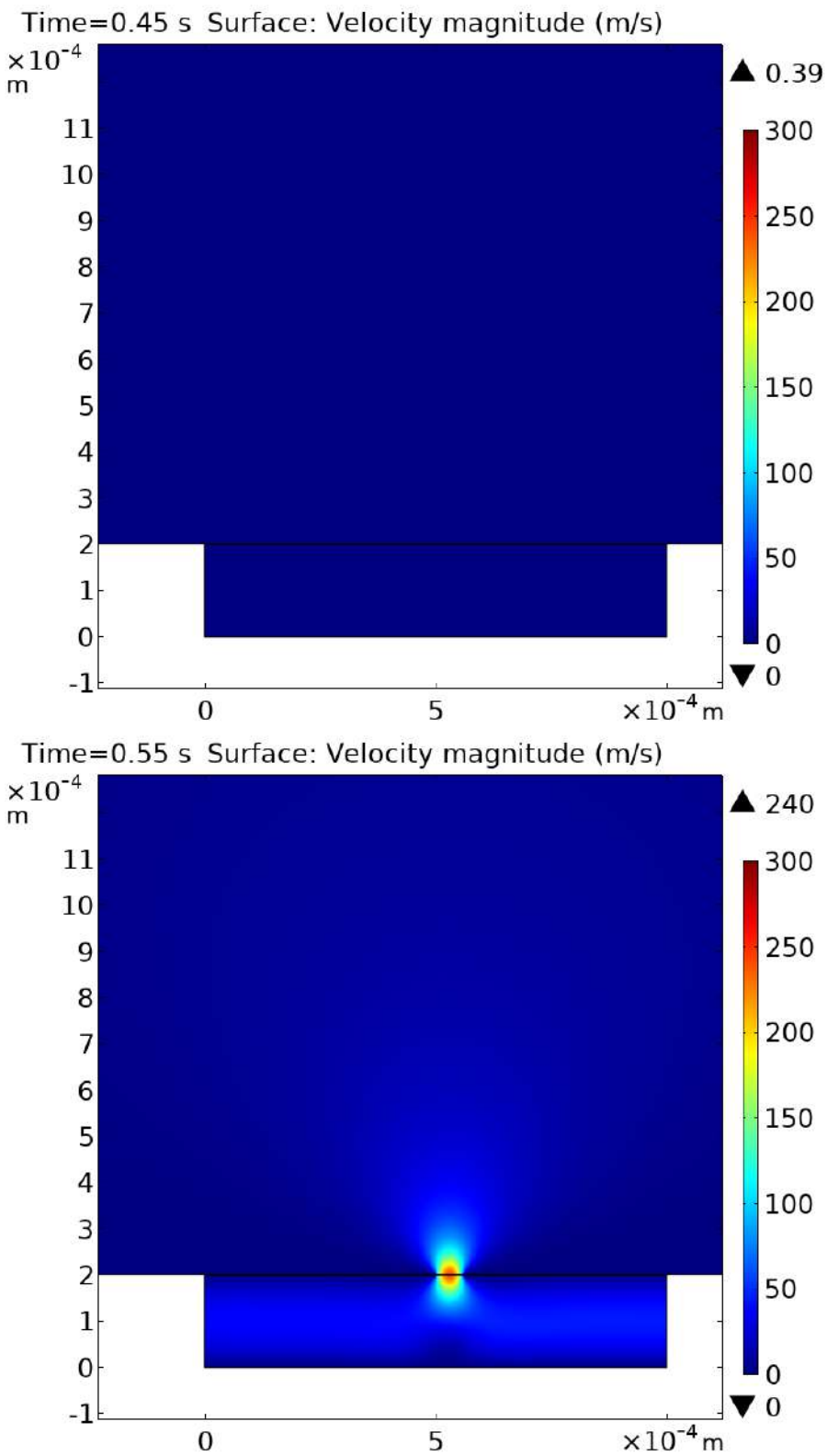


Figure 4.6: Results of the COMSOL simulation. Velocity throughout the system is plotted for $t = 0.45$ $[s]$ (top) and $t = 0.55$ $[s]$ (bottom). Opening of the wound happens at $t = 0.5$ $[s]$.

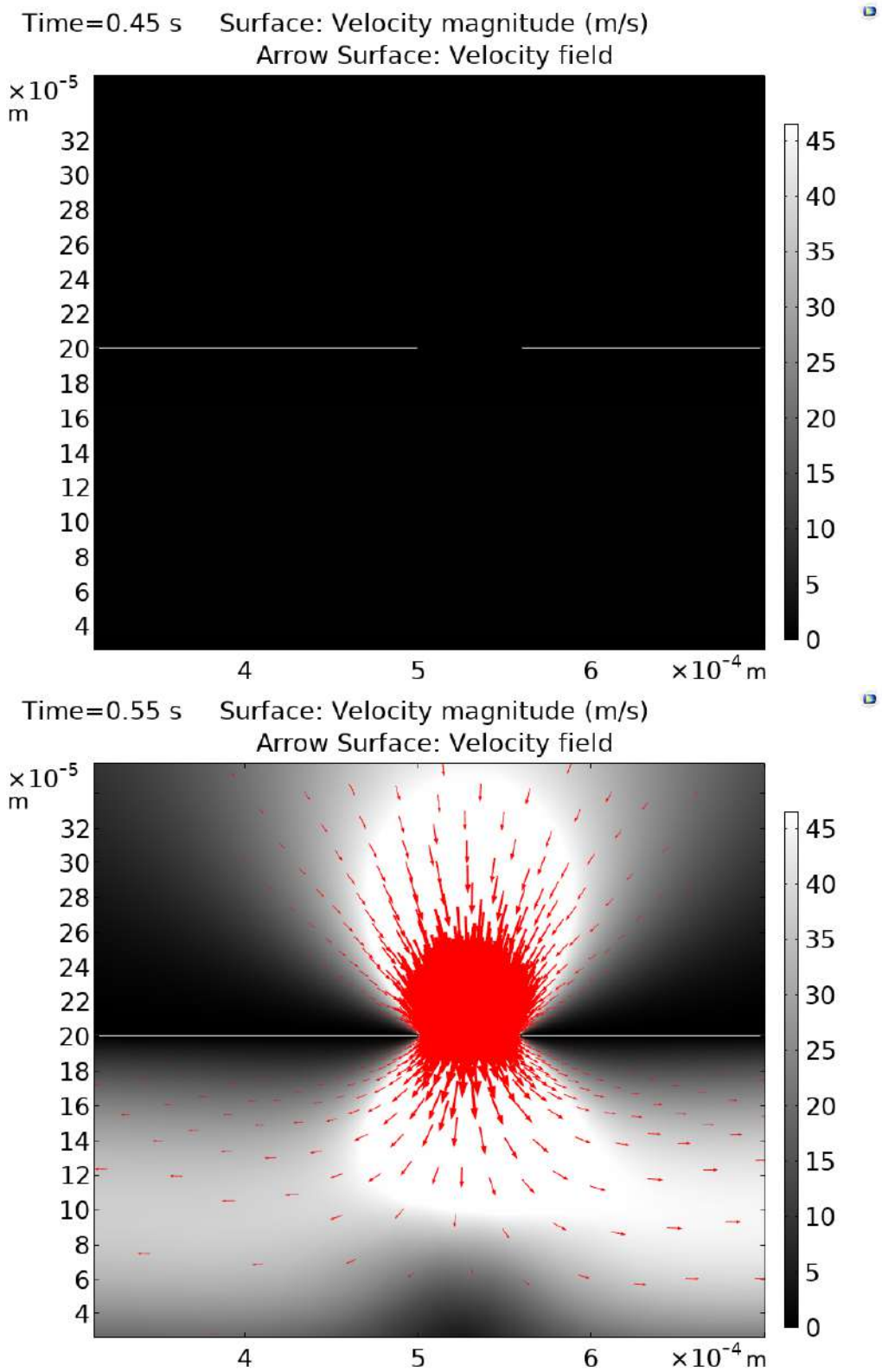


Figure 4.7: Results of the COMSOL simulation. Velocity field throughout the system is plotted for $t = 0.45$ $[s]$ (top) and $t = 0.55$ $[s]$ (bottom). Opening of the wound happens at $t = 0.5$ $[s]$.

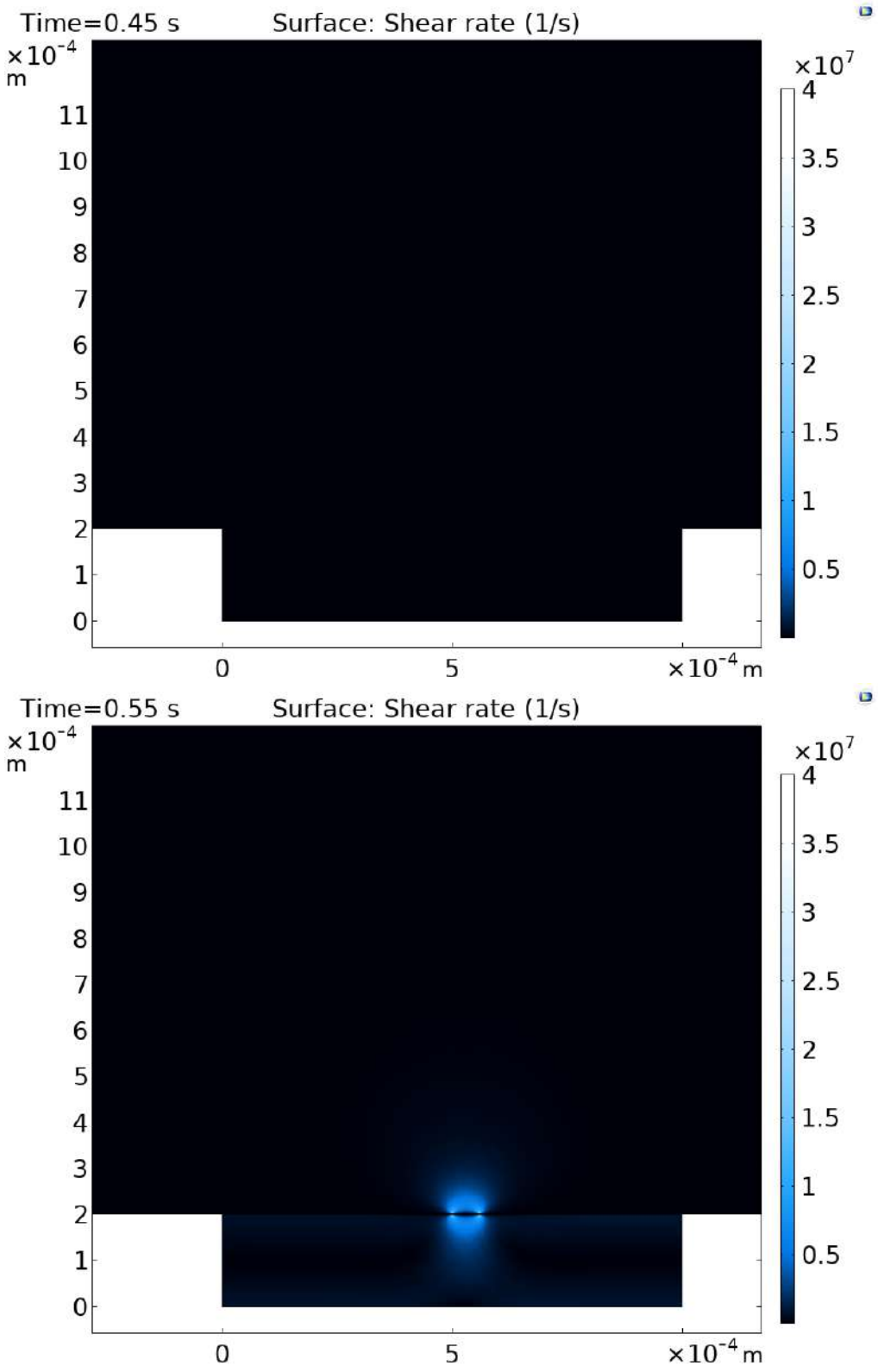


Figure 4.8: Results of the COMSOL simulation. Shear rate throughout the system is plotted for $t = 0.45$ $[s]$ (top) and $t = 0.55$ $[s]$ (bottom). Opening of the wound happens at $t = 0.5$ $[s]$.

5 Microfluidic Wound Model

In the last chapter, a simulative approach for determining the shear rate was presented. Another method that comes to mind are lab experiments, meaning creating artificial vessels (AV) and connecting them to a pump that operates at conditions similar to in vivo. The advantage would be controllable parameters, like vessel size and flow velocity. Furthermore, in this way experiments become better reproducible: every animal, every vessel will be a little different. Different in a way that can be hard to quantify. By creating a transparent microfluidic chip with an AV inside, fluid movement through vessel and wound can be recorded. This is done by tracking certain particles in the fluid, for example, small metal balls placed inside water. Particle movement can be used to calculate the fluid's viscosity or shear rate.

Another aspect to consider is scalability: medical papers often work with a low number of samples, only a dozen or a few dozen animals. Aside from cost factors, also ethical problems arise since animals have no chance to survive these experiments. If AV models can be designed accurately enough, this can be avoided. This leads to a new issue: making the experiments similar to the in-vivo situation. Simply using whole blood as a fluid will not work because of coagulation; an issue that will be addressed briefly in Subchapter 5.1.10. In Chapter 2.2 an extensive overview of all the biological processes happening during hemostasis was provided, for example the contraction of the vessel. As with the computer simulation, one must think about how, or if, these effects can be included in a microfluidic experiment. At a later stage, microfluidic experiments and computer simulation could be used to cross-check each other.

Here, only a proof-of-concept can be provided for conducting the experiments just mentioned. On the other hand, this allows implementing every necessary step at least as a demonstration. Additional work will be needed to work out the details, but the processes stay the same. For a description of the animal experiments serving as a blueprint, please refer to the beginning of Chapter 4. Generally speaking, the experiment can be separated into three stages. The chapter will be structured accordingly: design and manufacturing of the microfluidic chip, conducting the experiment and data recording and finally data evaluation.

5.1 Design and Manufacturing

If a method for manufacturing AVs has been established, a large quantity of probes with almost the same geometry can be created. Standard derivations for a given manufacturing process most likely can be found in literature exist or experimentally determined. Still, in terms of reproducibility, this beats animal experiments. An AV will always consist of four parts: a tube connecting the pump with the chip, called inlet, a main channel representing

the blood vessel, a wound channel simulating the injury of the vessel wall and two outlets for transporting fluid from the main- and wound channel to a bin. The most interesting part when designing a chip for hemostasis experiments is the implementation of a wound. In this thesis three possible approaches will be presented: a predetermined breaking point, a silicon flap operated by negative pressure and a H-shaped set-up working with air under counter pressure.

5.1.1 Predetermined Breaking Point

Separating main- and wound channel with a physical barrier that can be torn in is an intuitive choice. This wall however creates three issues: It has to be strong enough not to break accidentally during manufacturing or pre-wound-opening operation. But at the same time, it must be fragile enough to reliably give way once the wound gets "activated". A third issue are the remains of such a wall. They may clog up the channels and disturb the flow.

The latter aspect advocates a "wound activation" by applying negative pressure over a short period of time at the outlet of the wound channel. A separating wall could be torn in and sucked out of the experimental set-up into the fluid waste. Another option could be to install a needle hole in the chip that is aimed at the wall. After rapturing the wall, the head of the needle needs gets pulled back quickly to not interfere with the experiment. Potential fluid leakage through the needle hole also needs to be considered. If the material can be made transparent, at least in parts, a laser could be used to melt the wall down. This however may damage the rest of the structure.

It was planned to use the method described above with a classical 3D printing manufacturing process, discussed below in Chapter 5.1.5. In practice, it proved already difficult to manufacture a main channel with ca. 100 $[um]$ small dimensions. The wall would need to be even thinner. Even 3D printers used by private manufacturing companies, costing around 10 to 20 $[k€]$, could not potentially create such small structures according to the author's research at the time of this thesis (October 2021 to April 2022).

A last consideration concerns the production process of the chip: should it be created in two parts or one? If the chip can be created in one step, manufacturing time and -effort can be reduced. Additionally, fewer leakage problems will occur. On the other hand, if the chip is created as two parts, one containing the main- and the other one the wound channel, a small foil could be integrated into the structure as a wall. This foil could be thinner than what can normally be printed and might more reliably give way as a result.

5.1.2 Silicon Flap

This idea was presented in [SHBa18]. Fig. 5.1 illustrates the idea: main channel and wound are separated by an impenetrable wall that would be easier to manufacture than a predetermined breaking point. The bottom of both channels works as a flap that can be moved by applying negative pressure to a chamber below, see Fig. 5.1 b and c. Another advantage would be that no remains of the wall can clog up the channel. However, the method leads to a wound model whose geometry is not the same as in vivo, see Fig. 5.1 c, top row. Also, manufacturing requires a multistep process and might not allow for an easy chip production in large quantities.

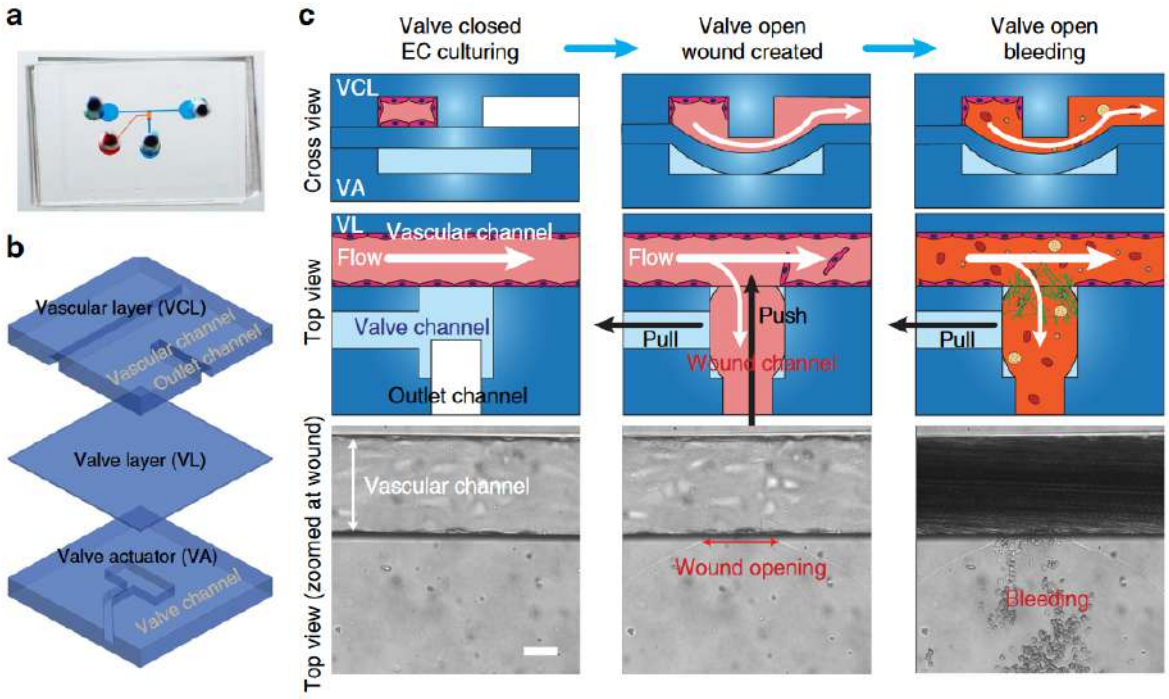


Figure 5.1: Microfluidic wound model using a silicon flap. On the right, the operation principle can be seen. Image taken from [SHBa18].

5.1.3 Counter Pressure

A way to tackle issues from both previous examples, the potential clogging of channels and the non-ideal geometry of the wound, will be presented as a last example. The idea was proposed by Prof. Hayden from the Heinz-Nixdorf Chair, TUM. A graphical representation of this idea can be found in Figure 5.2. Initially, the fluid is circulated through the vessel inlet and -outlet. As of now, no counter pressure is applied, so liquid can also flow into the wound channel, although the experiment has not started. Now air gets pumped into the wound channel, the lower part of the tilted "H" that can be seen in Figure 5.2.

Air pressure gets increased until the fluid is pushed back into the main channel. For "activating" the wound, counter pressure drops to zero and fluid can flow through the wound channel. This design has the advantage that no separating wall is needed, avoiding the complications discussed for the first proposal. It is also closer to the in vivo situation compared to the second proposal. As a drawback, it must be noted that endothelial cells grown on the chip walls for biocompatibility reasons, see below, might be destroyed when applying the initial air pressure. Once a concept for implementing the wound is chosen, the question of manufacturing method remains to be solved. Here, four possible options come to mind: classical 3D printing, silicon moulding and PDMS laser printing. These three options are suited for self-manufactured chips. The last option is to buy an already existing solution.

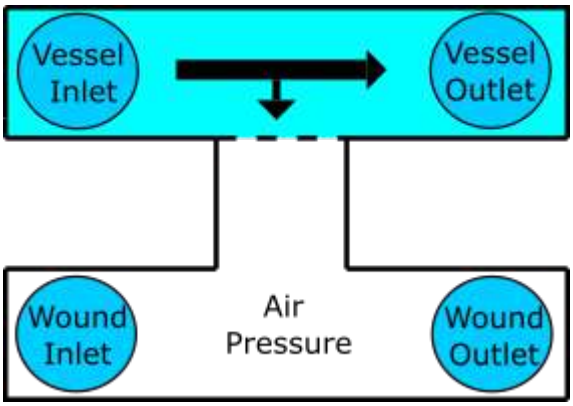


Figure 5.2: Schematic of the "H" shaped air pressure wound model. Air is pumped through the wound channel to make sure liquid flows only through the vessel channel. To "open the wound", the pressure is set to environmental pressure. Idea proposed by Prof. Hayden, TUM.

5.1.4 3D Printing

3D printing has become increasingly popular in recent years. Machines became cheaper, making them available even as consumer products. Many different types of 3D printing exist. In fact, one could argue that the PDMS laser writer presented below is also a 3D printer. In principle a printing material, like ink in a conventional printer, gets placed into a certain position in a liquid state and then hardened, for example, by a photochemical reaction induced by a laser writer. This writer would be the analogy of a conventional printhead.

For the project, of which this thesis was a part, Mr. van den Heuvel from LMU was tasked with designing the CAD and printing a prototype. The results can be seen in Figure 5.3. "Miicraft – Ultra 50" was the used printer. It features a xy-resolution of 30 $[um]$ and a z-layer resolution of 5 to 500 $[um]$ [Mii22].

"Autodesk – Fusion 360" was used for the CAD model. For the CAD model, the dimensions were as follows: 5 [mm] length for the main channel, 1.25 [mm] width for both main- and side channel. From the opposite wall to the tip of the side channel. 1.5 [mm] was the height of the print. The actual printout was magnified 3 times in size. Even then, all channels were clogged. It is assumed that the printer could not deal with the small scales, although according to the data sheet this should not be an issue.

After printing the protype, a glass lid must be glued to the bottom. This seals off the channels and allows for microscopy since the printer material is non-transparent. Unfortunately, the prototype was not functional. The printer was not able to leave the space for the channels empty. While the inlets and outlets could possibly be freed with a drill, the main channel and wound channel, which is the small side channel in Figure 5.3, should be printed correctly for the experiment to run smoothly. As mentioned above, even very expensive commercial 3D printers might have problems to print a structure in the desired dimensions. One has to think about using an upscaled version if it can still provide some relevant insights. Otherwise, another manufacturing method must be chosen.

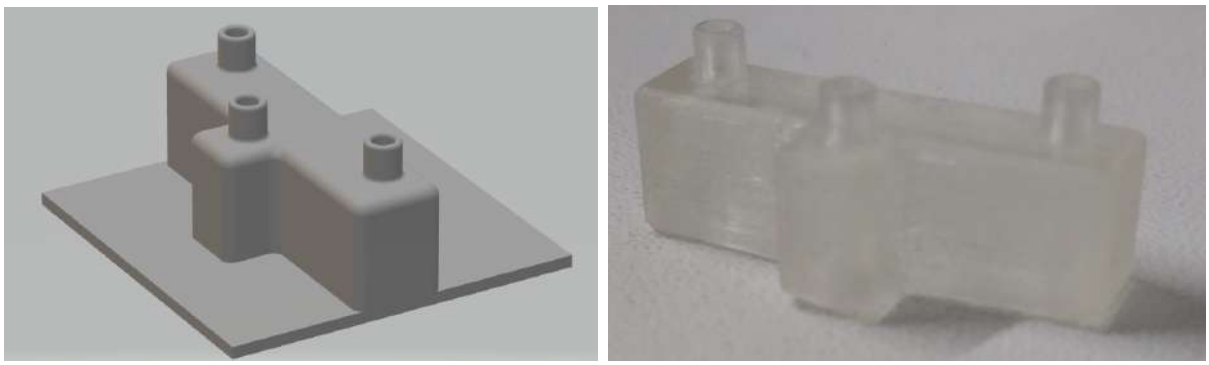


Figure 5.3: CAD model (left) and print (right) of the prototype chip done by Mr. van den Heuvel, LMU.

5.1.5 Silicon Moulding

Looking at the chip presented in [SHBa18], the idea of silicon moulding comes to mind as an alternative manufacturing method. Although according [SHBa18]'s method part soft lithography was used, silicon moulding might allow for a faster way of manufacturing. After a chip is designed as a CAD model, a negative could be created as a metal mould, similar to a biscuit cutter. The manufacturing of the metal negative must be done special machines. It might be expensive if done by a private company. Teaming up with a chair from the mechanical engineering department or the material science department could be a workaround. Once a mould has been obtained, silica could be poured in and hardened, for example by baking. Finally, the chip must be carefully extracted from the mould and glued onto a microscope dish. This option was not practically explored during this thesis.

5.1.6 PDMS Laser Printing

Polydimethylsiloxane (PDMS) laser printing in principle is another form of 3D printing. In short, it works as follows: A photoresist (PR) gel or fluid is hardened by a laser. The photo resistive has the property to dry up if light with a certain wavelength is shone on it. This is done by a laser moving along rails in the printer. In this first manufacturing step, the "negative" of the chip is printed. The negative is what later will be an empty channel through which liquid flows. As a next step, non-hardened PR is washed away and the negative placed in a Petri dish filled with liquid PDMS. Baking the PDMS and removing the negative now provides the actual chip. This process is explained in more detail below.

As a second and last prototype constructed as part of this thesis, was a PDMS chip following the concept of the "H" layout mentioned above. CAD design was performed in AutoCAD. Figure 5.4 shows an image of the CAD design. The length of the outer bars of the H was 2 $[cm]$, the width 100 $[um]$. For the middle bar, a length of 200 $[um]$ was chosen together with a width of 50 $[um]$. The circles on the tips have a radius of 100 $[um]$. Ideally, the wound channel should be placed at half the channel height to represent the in vivo situation. Unfortunately, the printer is not capable of placing a structure on non-hardened part of the photo resistive, so at least two printing steps would be necessary to create a chip with a wound-channel in the middle. The two halves need to be glued together in an airtight way to prevent fluid leakage in the chip. To safe time, the wound channel was placed on the bottom of the chip, hence requiring only one production step. An illustration can be seen in Figure 5.5. For the same reason, the cross-section of the channel is rectangular and not circular.

Figure 5.4: CAD design of the PDMS prototype done in AutoCAD.

The manufacturing took place in a cleanroom with yellow foils attached to the window. The cleanroom has a security level S1 and a bio-level 2 but no official cleanroom level. While a cleanroom ensures a low amount of dust particles that could contaminate the chip, the yellow foils filter out parts of the sunlight with a wavelength like that used to harden the photo resistive by the printer. To use the PR, a solid bottom is required. It should not chemically interact with the PR and endure temperatures of up to 200 $[°C]$. In case of this thesis, a silicon wafer was used since they are cheap and easy to come by. As a first step of manufacturing, the wafer was placed inside an oven and got backed for 90 $[min]$ at 150 $[°C]$. This serves to remove potential humidity from the wafer. Inside the oven, the wafer should be placed on some kind of pedestal with its bottom facing down. No special requirements for oven or pedestal exist.

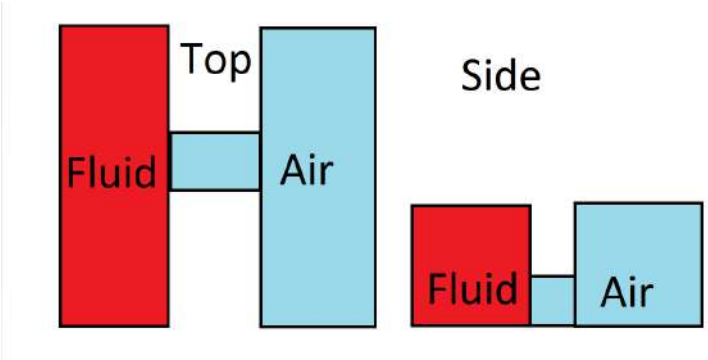


Figure 5.5: Scheme of the channel printed with PDMS. The wound model chosen for this chip is the same as in Subchapter 5.1.3.

To make sure the wafer is still dry, the next step needs to be done immediately afterwards. Applying the PR to the wafer on a special workbench prevents hairs and dust from getting on the chip. Aside from an exhaust duct, a glass lid separates the workspace from the operator, only hands and arms can be moved freely through an empty space at the bottom. The set-up can be seen in Figure 5.6. As a next step the wafer is placed in the centre of the centrifuge, see Figure 5.6 B, blue rectangle. The centrifuge has a hole on the top which can be used to spray acetone on the wafer to clean its surface. This can also be done while the centrifuge is spinning at 5000 $[RPM]$ to distribute the acetone equally. One should wait a few minutes until the acetone has vaporized and additionally use the nitrogen blow-dryer to dry the surface before applying the PR. Nitrogen is used because compressed air can sometimes contain unwanted particles. 1 $[ml]$ of PR should be used for every inch of wafer diameter. Specifically, the PR "SU 8 3050" was used. For details, please refer to the data sheet of the manufacturer [Kay22].

To distribute the PR on the surface of the wafer evenly, the centrifuge is used. The RPM and operating time depend on the PR and the height of the structure one wants to print. It is advisable to have a "ramping-up time" in the beginning, where a lower RPM is applied, to ensure an even distribution of the PR. For this thesis, ramping-up was done at 500 $[RPM]$ for $5 - 10$ $[s]$ with an acceleration of 100 $\left[\frac{RPM}{s}\right]$. Then 3000 $[RPM]$ with an acceleration of 300 $\left[\frac{RPM}{s}\right]$ was applied for 30 $[s]$. According to the manufacturer, this should lead to a thickness of 50 $[um]$. Consult with manufacturers for possible error margins. After taking out the wafer, the centrifuge might require cleaning with, e.g., acetone to remove PR rests.

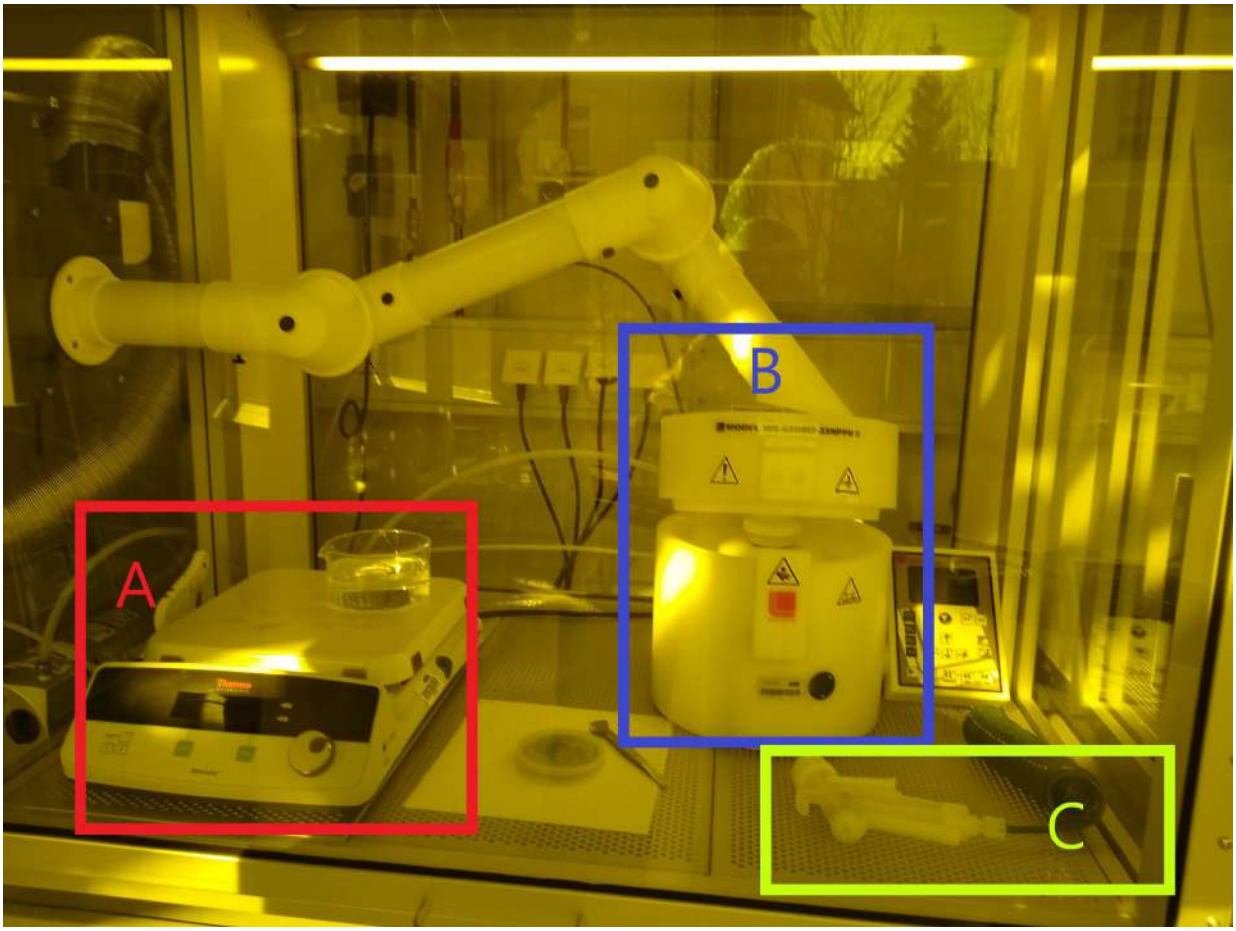


Figure 5.6: Workbench for applying the PR. A (red): Heating plate, B (blue): Centrifuge and C (green): blow-dryer.

To continue the process, a step called "soft backing" is needed. It will again depend on the PR and the desired height. Again, it is advisable to use a ramping-up and later a ramping-down step to reduce the thermal stress on the material. Here, the heat plate was set to 65 $[°C]$ for 5 $[min]$ and then 95 $[°C]$ for 15 $[min]$. An additional cooling-down step was not performed, but could for example be done at 65 $[°C]$ for 5 $[min]$.

Now all preparations have taken place to print the negative of the channel. For this, a laser writer "Dilase 250" from the company "KLOÉ" was used. For detailed information on this device, please refer to the company's website [KLO22]. The printer was operated via a connected PC and a software provided by the manufacturer called "Dilase Soft". Please refer to the manufacturer for details of how to use the software. The imported CAD structure was filled with polygons with a 2 $[um]$ point size. The laser must be set to a zero level and corrected for potential differences between the focal points of the internal camera and the laser. These are experimental values and have to be figured out individually. "Modulation" and "Writing Speed" were set to 15 and 20 $\left[\frac{mm}{s}\right]$ respectively. Both are experience values suggested by colleges, found by means of trial and error. A spot size of 4 $[um]$ was chosen. Finally, the printing process can start.

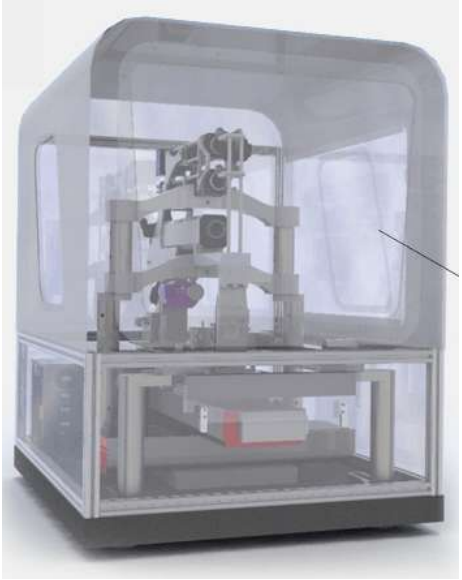


Figure 5.7: CAD model of the interior of the printer from the manufacture's website [KLO22]. The rails in the middle of the cube can move the laser around.

It may take up to several hours. A promotional CAD model of the printer's interior can be found in Figure 5.7. Once the printing is finished, the "post exposure backing time" must be done on the same heat plate seen in Figure 5.6. It will again depend on the set-up. Here, 1 minute at 65 $[°C]$ and 5 minutes at 95 $[°C]$ were applied. To obtain the negative of the channel, the rest of the non-hardened PR has to be washed away. For this, a "developer fluid" is used. "mr-Dev 600" developer fluid from "micro resist technology" was filled in a bowl so that the wafer would be fully covered in this liquid. Leaving the wafer in the bath for too long can lead to the destruction of the printed structure. The unnecessary PR can be washed off by shaking the bowl carefully from time to time.

In practice, more time was needed for this step than the PR manufacturer suggested. After the specified time of 8 minutes had passed, the wafer is cleaned with isopropanol. if white stains appear on the wafer, it should be placed back again. This process is illustrated in Figure 5.8. If little or no white stains remain, a nitrogen blow-dryer (Figure 5.6 C) is used to clean the wafer. Acetone can also be used additionally. Please note that the developer fluid has to be deposed properly because of its chemical properties. The wafer now needs to dry in an appropriate box for one day.

On the next day, one can start to mix the PDMS. In this thesis, a 27 $[g]$: 3 $[g]$ $\hat{=}$ 9 : 1 relation of base and curing agent was used. As a base, "SYLGARD 184 Silicone Elastomer Base" was picked with a corresponding curing agent from the company "The Dow Chemical Company". Once the wafer is placed inside a plastic Petri dish, pour over the PDMS. The Petri dish will be destroyed in the process, so make sure not to use one made from glass. Next, the air bubbles must be extracted from the PDMS. For this, use a vacuum chamber and -pump set-up, as the one visible in Figure 5.9.

When only a few bubbles are left, the Petri dish can be taken out of the vacuum chamber and the wafer should be rested for 10 minutes until all bubbles are gone. Next, bake the PDMS for 1 $[h]$ at a temperature of 65 $[°C]$. To attain the actual channel or at least its upper-half, the PDMS needs to be carefully separated from the dish and the wafer. It proved to be useful to cut the plastic Petri dish on the side with a razor blade and break off the resulting parts separately. As a final step the majority of the PDMS can

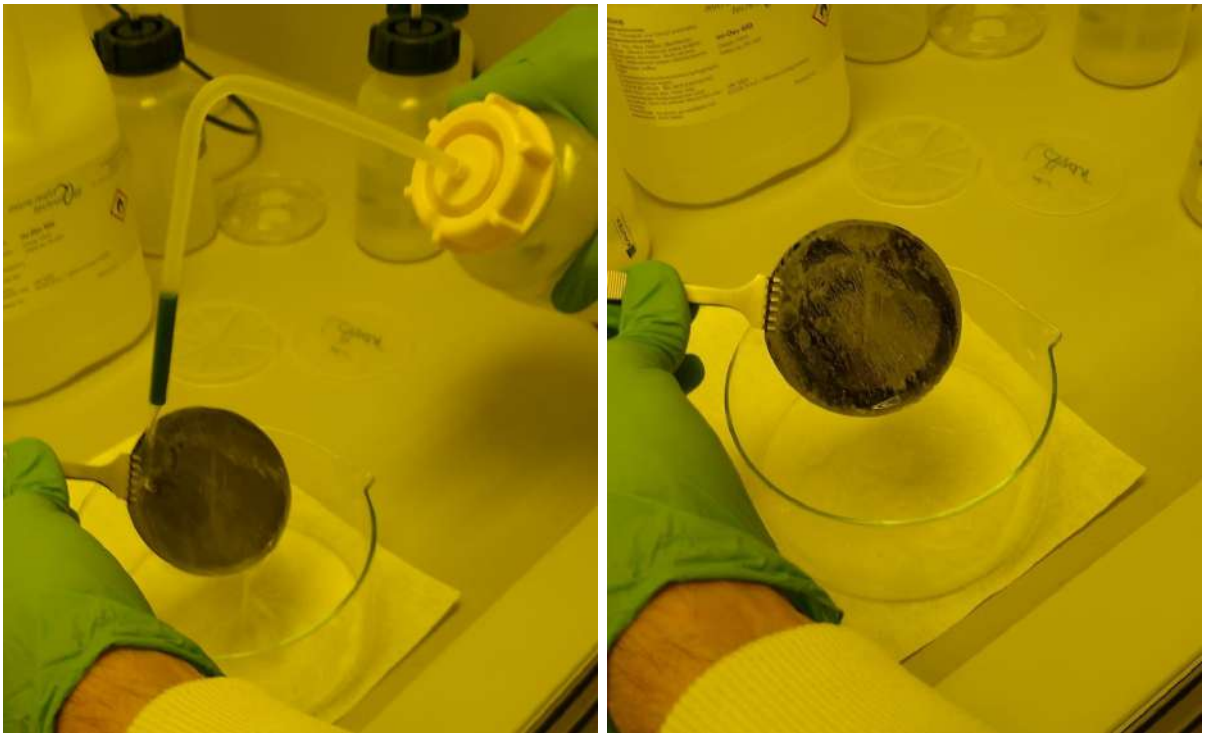


Figure 5.8: Cleaning off the rest of the PR with isopropanol (left) until no more white strains can be seen (right).

be cut away. One must be careful not to apply too much stress. Even small cracks can lead to fluid leakages in the final chip. At this point the channel in the PDMS can again be cleaned with isopropanol and a nitrogen blow dryer. Using a special hole puncher, a needle with a mechanical actuator, the hole for the connecting tubes can be made. The tubes need to be carefully connected and the PDMS glued to a microscope slide. This is illustrated in Figure 5.10 below. The chip that can be seen here was unfortunately not operational due to an error in the CAD design. Holes for the connecting tubes were not placed far enough away from each other so that they raptured the channel when punching the necessary holes into the PDMS. Adjusting it was not possible and a new chip could not be manufactured for time reasons.

Among other things, the time requirements mentioned in this subchapter mean that this procedure is not suited for large scale testing. Many steps are necessary that have to be performed for each chip. In practice, chips tend to malfunction from the start or after a short period of operation, about one week. An alternative might be to buy already existing chips, which is discussed in the next subchapter.

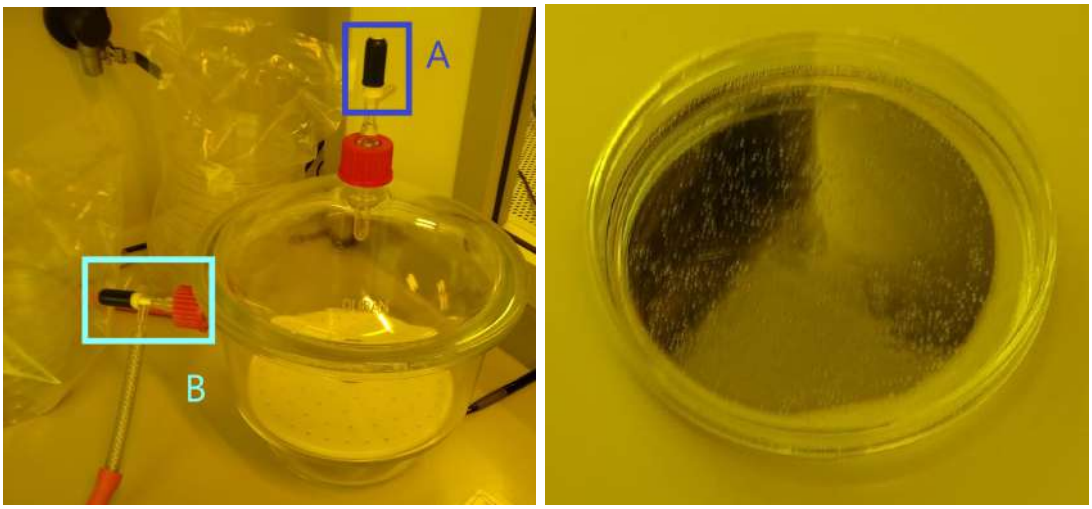


Figure 5.9: Vacuum chamber (left) with two valve for connecting the vacuum pump (B) and releasing the negative pressure (A). On the right the wafer placed in a plastic Petri dish, buried in PDMS can be seen.

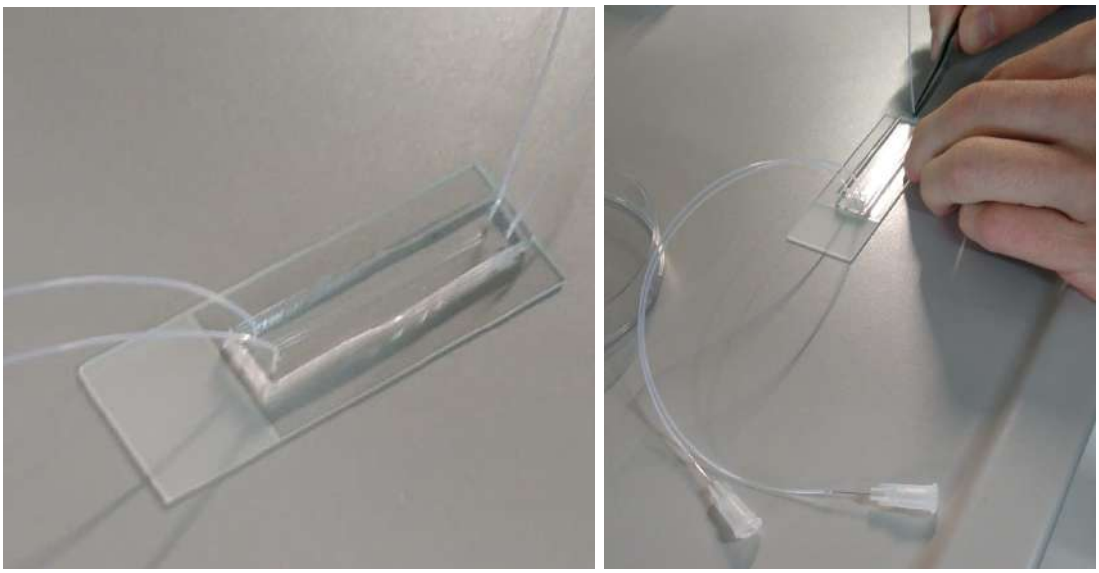


Figure 5.10: Connecting the tubes to the PDMS chip (left) and placing the final chip on a microscope slide (right).

5.1.7 Outsourcing

As a last resort, one can buy already existing chips from third party manufacturers like [chi22] or [Ibi22a]. Although the design might not be exactly as desired, the advantages would be the availability and better quality of the chip. Also, time for the manufacturing process can be saved. If financially possible, one could also think about handing out a development contract for an individual chip. However, the cost for this might be quite high, starting at 50 [k€], as one company responded to a request.

None of the self-manufactured prototypes reached an operational state. For this reason, an already existing chip was used. The chip includes no wound model but allowed a time saving conduction of a dummy experiment and showing a possible way of evaluating the resulting data.

5.1.8 Used Chip

A pre-manufactured chip from the company Ibidi was used for the experiment described later in Subchapter 5.2. Figure 5.11 shows an advertising image of the chip. The model named "u-Slide I Luer" consists of a simple design: inlet and outlet are connected in a straight line, embedded into a microscope slide. No wound model is part of this chip. The height of the channel is 250 [um], the width 500 um and the length 50 [mm] [Ibi22b], hence the cross-section is a rectangle. According to a response from Ibidi technical support, the channel height has a manufacturing error of ± 5 [%]. The manufacturing error for the channel width was not revealed because of company policies. As a last point before coming to the actual experiment, a discussion of the test fluid will be lead.

Figure 5.11: Advertising image of the "u-slide I Luer", taken from [Ibi22b].

5.1.9 Test Fluid

Ideally, human full blood should be used as a test fluid because of the complexity of its fluid mechanical properties, discussed in Chapter 2. Colouring platelets with matching antibodies could be used in combination with a fluorescence microscope to track their movement through the chip. In Subchapter 5.3 the method for evaluating video-recordings of such movements will be discussed. The major drawback of using full blood is the requirement of a biocompatible chip. If this condition is not fulfilled, unwanted platelet triggering can occur, clogging the main channel and other parts with red thrombi. Biocompatibility will be addressed in the next subchapter.

An alternative would be to use half-blood or specially prepared blood. For example, depriving blood of calcium is used in the case of blood donations [Zal10]. A similar procedure could be an alternative to a biocompatible chip. Other drawbacks are legal- and storage requirements for blood. Fortunately, only small amounts of blood will be needed per experiment. Although no full- or half blood was used for this thesis, less than 10 $[ml]$ were necessary for a multiple minute operation of the microfluidic chip.

For time and simplicity reasons, no blood or blood derivate was used for the experiments. Distilled water was mixed with a metal flake solution 10 $[mL]$: 100 $[uL]\hat{=}100:1$. The flakes have an albumin hull. While probably many similar products exist, here "micromer -M albumnin (BSA)" with an average diameter of 12 $[um]$ were used, produced by the company "micromod". They are already sold as a flake-water mixture with a concentration of 25 $\left[\frac{mg}{ml}\right]$. Although the specific product seems to be discontinued, for a detailed description of similar products, please refer to the website of the manufacturer [Mic22]. After adding more water, the mixture should be stirred again before being pulled into the syringe. Below in Figure 5.15 and 5.17 images of the particles under a microscope can be found.

5.1.10 Biocompability

Given the complex non-Newtonian behaviour of blood, it seems a good idea to run the experiments with whole blood if possible. Even then effects like the initial contraction of a vessel after the occurrence of the wound, see Chapter 2.2 still have to be either neglected or modelled differently. The problem is: Using blood with no extra preparations leads to red thrombus formations in the chip or channels. As mentioned, using blood requires hemocompatible materials [KWTG21]. Achieving this is non-trivial and could not be done in this thesis. Still an overview is presented in this subchapter, based on the review article "Control of Blood Coagulation by Hemocompatible Material Surfaces" by Jana Kuchinka, Christian Willems, Dimitry Telyshev and Thomas Groth [KWTG21].

Many materials can be used in the context of medical devices, for example ceramics, polymers and iron alloys. But none of them reach enough blood compatibility [KWTG21] [SKM12]. For a detailed explanation of how the unwanted thrombi appear in medical devices, please refer to [KWTG21]. To summarize, protein adsorption on a device surface triggers the activation of the coagulation system. "In particular, the adsorption of coagulation factors, such as FXII, HMKW and fibrinogen, but also complement immune factors, immune globulins and adhesive proteins (vWBF, FN, VN), must be suppressed or controlled" [KWTG21]. In Chapter 2.2 it was already described that thrombus activation can be separated into an intrinsic and extrinsic pathway. For medical technology, the intrinsic pathway causes the most problems [SVH98]. Although not known until the early 2000s, the extrinsic pathway also must be considered for blood transport [KWTG21] [Gor04].

To prevent blood coagulations, titanium [EL03] or diamond-like carbon coating [FMMM09] can be used. Other methods include "[...] repulsive surface changes, making superhydrophilic surfaces, as well as layers of hydrophilic macromolecules to exploit hydration forces and steric repulsion [Bri17]" [KWTG21]. Further ideas are surface modifications with organic molecules [CBS04], brush-forming polymers [LOSC15], polyethylene oxide [ADCH21], zwitterionic coatings [JFHW15], 2-methacryloyloxyethyl phosphorylcholine (MPC) polymer coating [FMW18], albumin layers, textured surfaces [DCSa87], heparin [LCW03] or hirudin [GW08]. Finally, one can also grow endothelial cells in a microfluidic chip [SHBa18]. While an endothelial coating seems to bring the chip closer to the in vivo situation, it might also be very difficult to realize. Other ideas could be implemented by integration into the manufacturing process or performing an additional step. This needs to be further explored and is beyond the scope of this thesis.

5.1.11 Conduction of the Experiment

After discussing the necessary preparatory steps, this subchapter describes the procedure of the experiment. First a microfluidic chip gets placed on a microscope, then a pump and a waste are connected to the inlet and outlet of the chip via tubes. The pump starts and fluid flows from here through connecting tubes into the chip and then into a waste container. A camera mounted on the microscope films a part of the chip so that the movement of the particles can be recorded. Stopping the pump ends the experiment. Later on, the video data is analysed to determine the particle velocity, see Chapter 5.3.

The next paragraph explains this in more detail. At this stage, a microfluidic chip has been chosen. First, one needs to add the connective tubes to the chip. Figure 5.12 provides an overview of all the necessary components for that. Notice that the chip in the photo is different from the one seen above, but since both are from the same company, connecting them works the same. In practice, it proved to be practical to fixate the chip on the microscope table only after the connective tubes had been attached. Only once the chip is no longer able to move, pump and waste should be added.

The chip can be fixated, for example, with tape. Placing the tube connected to the chip outlet in a small measurement glass provides a bin for the fluid. Throughout running the experiment, the tip of the tube may not be covered by fluid or the pressure at the outlet changes! This would correspond to an abrupt change of border conditions in COMSOL, see last chapter, and is unwanted. Figure 5.13 shows a close-up of the chip connected to the bin. As a following step, the syringe needs to be filled with the test fluid described in the last subchapter. One might need to remove the adapter on the syringe tip. Fortunately, it can be taken off easily. The syringe is placed in a syringe pump and start the pump. Two things are to be kept in mind here. It takes a certain amount of time until fluid is filling up the entire chip and tubes. This is necessary to reach a stationary condition.

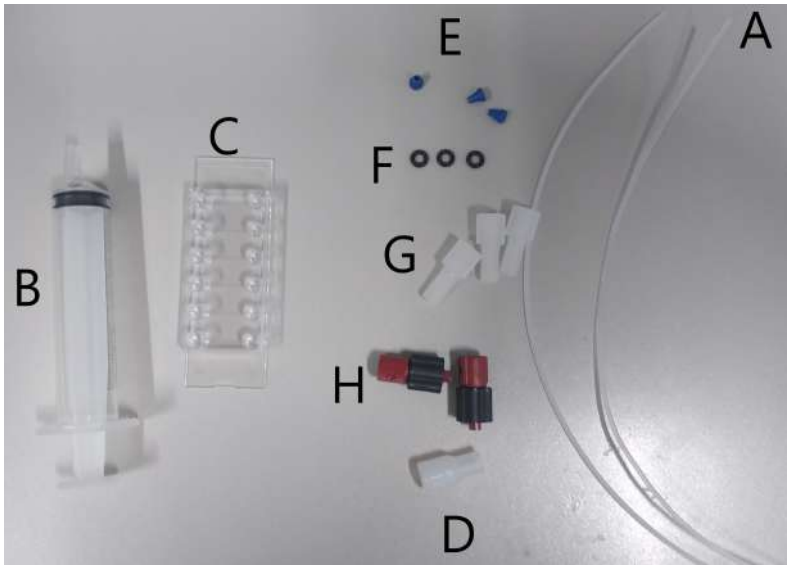


Figure 5.12: Necessary parts for adding connective tubes to a microfluidic chip. A denotes the connective tubes, B the syringe later used with a syringe pump. C is the microfluidic chip, D an adapter for the tip of the syringe. E and F are rubber cones and -rings to prevent fluid leakage at both the syringe connector and the chip-connectors H. G denotes the screws necessary to fixate the tubes to the three adapters.

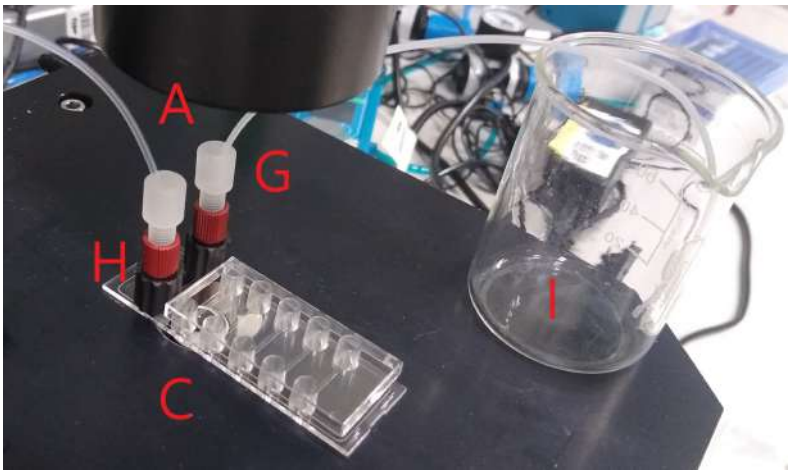


Figure 5.13: Close-up of the chip on the microscope table connected to the bin. A denotes the microscope lens, C the microfluidic chip, G and H the connective adapters for tubes and chip and I the fluid waste bin.

A "Nikon ECLIPSE TS100" microscope was used with a 40x magnification together with a camera mount. The second thing is that pumps usually operate with flow rates. A flow rate of 0.001 $\left[\frac{ml}{min}\right]$ was chosen. Faster flow rates were not possible since particles would not be visible on the videos. A camera with a higher FPS could fix that. "FUSION 4000-X DUAL INDEPENDENT CHANNELS" by the company CHEMYX was used as a pump. For technical details, please refer to the manufacturer [Che22].

An alternative to a pump operating at a certain flow rate would be a pump that operates at a certain pressure. In this way, a simulation with pressure values as border conditions can be implemented as a lab experiment. An example of such a pump is the "P2C S" from the company "biophysical tools" [Def22]. Talking about pressure, another problem should be mentioned: the connective tubes and the adapters have a different diameter than the chip. Hence, at the intersections, the velocity of the fluid will change according to the law of volume flow conversation. Kinks in the tubes and especially the situation at the outlet tube can also influence the experiment in a bad way.

Finally, one can start the recording of the experiment by activating the camera. Since the software used for tracking the particles, see next subchapter, can only handle black and white recordings, a camera without colour capabilities is enough. For this thesis, videos were recorded with colour channels present and later converted. The camera "MC 170 HD" by the company Leica was used together with the corresponding software "LAS EZ" for image or video recordings.

5.2 Data Evaluation

The output data type of the experiments will be video files. Two components are key: the clearness of the images making up the videos, called frames, and a precise tracking of the time passed. If all frames are recorded at the same speed, as they usually are, dividing recording time through the number of frames allows an assignment of each frame a time duration value. Most camera manufactures already provide the inverse of this number, the so-called "frames per second" (FPS).

A single reference length is needed to determine the distance travelled by a particle, assuming the camera is not moving. This reference length can be dived by the number of pixels that make up that length, so each pixel in an image can be assigned a physical length. This can be understood as the resolution of an image. One possible way to establish a reference length would be to use the width of the channel. Notice that manufacturing uncertainties directly translate to uncertainties of the final velocity values.

How many particles can be seen per frame will strongly depend on the particle concentration. A compromise has to be made between interference with the fluidic properties and more velocity sample data. For each frame, the position of a particle must be evaluated and the time value of the frame assigned to this position. Then, in the next frame one must determine where the particle has moved, meaning which particles in two consecutive frames are the same. Tracking how many pixels the centre of a particle has moved allows to determine what distance it has travelled. This together with the FPS allows determining the velocity of a particle.

Intuitively, the shear rate φ can be understood as the spatial derivation of the velocity. Since no analytic function will be available for the velocity vector field, an approximation can be found by taking the velocity difference $\vec{u}_1 - \vec{u}_2$ of two particles close to each other and dividing by their distance $\vec{x}_1 - \vec{x}_2$. Remember that in general φ is a second-order tensor, see Chapter 2.1, Equation 2.3. This means the ij-th matrix entry corresponds to the j-th spatial derivative of the i-th component of the velocity vector:

$$\varphi_{ij} = \frac{du_i}{dx_j} \approx \frac{u_{i1} - u_{i2}}{x_{j1} - x_{j2}} \tag{5.1}$$

Even at low particle concentrations, tracking will be close to impossible to do by a human and prone to error. An alternative comes to mind: machine learning. If the computer can be taught what a particle looks like, each frame can be automatically analysed. The only thing left to do now, is to find an algorithm to determine which particles belong together in two subsequential frames. Image recognition and -evaluation has been a well-established machine learning application, for example to drive autonomous cars. In the last years, finished implementations have become increasingly available. For a deduction of the necessary theory etc., please refer to literature. An already existing implementation shall suffice here.

"DefocusTracker" is an open-source MATLAB tool that was specifically built for the task at hand: tracking particles in a video recording of a fluidic experiments [Tra22], hence making it an obvious choice for this thesis. It is an implementation of a method referred to as "general defocus particle tracking". Please refer to the website of the creators for details. An algorithmic alternative could be convolutional neural networks [Tra22]. A possible alternative is "Object-Tracking-Deepsort" [Nan22]. MATLAB R2018b and the add-ons "Curve Fitting Toolbox", "Image Toolbox" and "Statistics Toolbox" are necessary requirements for using DefocusTracker [Def22]. Having performed the experiments as described in the last subchapter, the next step is to create a reference particle image set to teach the program what it should track. For details on how to operate the software, please refer to [Def22] and especially to the work-through examples.

To take the training pictures, the pump is stopped. When the particles stop to move, a single particle is picked out that ideally has no close neighbours and can later easily be cut out of the image. Additionally, the particle should be "sharp", meaning it is floating in the focus level of the microscope. Now a "z-scan" needs to be performed, a process explained in the following paragraphs.

Two options are possible: either one wants to correlate particle "sharpness" to the z-position of the particle inside the channel, or one simply wants to average particle velocity over the entire channel height. The first option is quite complicated. One needs to determine the z-position of the optical focal point relative to the coordinate system of the channel. In practice, that will be close to impossible without a digital movable focal point and precise knowledge of the chip's dimensions and manufacturing error margins.

In both cases, the next step is to zoom the microscope out so far that the particle is barely visible. A picture is taken. Afterwards, one must zoom in a little more and take another picture. If the sharpness of the particle should translate to its z-coordinate, one needs to keep track of how far the focal point is moved up and down the entire time. Repeating the process of zooming in and taking a photo until one reaches the point where the particle becomes sharp again gives the first half of the data set. This process continues by increasing zoom so much that the particle is not sharp anymore until eventually it becomes barely visible.

For image recognition, as with any form of machine learning, the size of the data set is key. Too few samples lead to a bad particle detection. Too many may result in "overfitting". DefocusTracker offers an option for virtual image interpolation, meaning a virtual upscaling of the training data set. Please look at their website for details. Twenty images of a single particle were taken at different focal positions. But the resulting particle recognition was not satisfactory; it might be necessary to take 100 to 1000 images. Figure 5.14 illustrates the concept of the z-scan. Once the images are made, the particle that was chosen as a reference needs to be cut out. DefocusTracker can only handle black and white images, so they also might need to be converted.

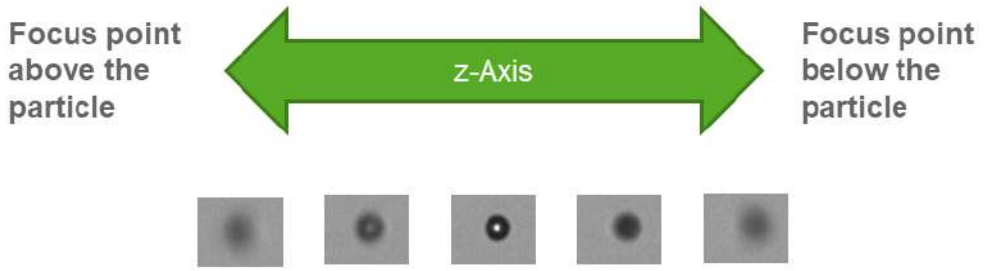


Figure 5.14: Illustration of the "scan". On the bottom one can see the particle recorded with the microscope zoomed out, in plane and zoomed in from left to right.

After training the model, DefocusTracker offers a control window to control particle detection in the training images. It draws a green circle around what it assumes to be the particle. Figure 5.15 shows two examples of good, and two of bad particle recognition. Keep in mind that the training images will in a way provide an upper bound of quality for the particle recognition. Except for the images where the particle is barely visible, one should make sure to have good results before proceeding.

Regrettably, DefocusTracker cannot handle videos per se. For that reason, one must decompose each video into its individual frames and then convert those frames to the .tif format. On Linux, this can be done via the command line [Sta22a]. Be advised that this can lead to a lot of files and blocked hard drive storage. Figure 5.16 shows an example analysis of a single image. Green circles where the software assumes particles to be. It can be seen that many errors occur: a lot of particles do not get recognized and, even worse, many "particles" are detected in empty spaces. Especially the latter is unacceptable for a reliable experiment evaluation.

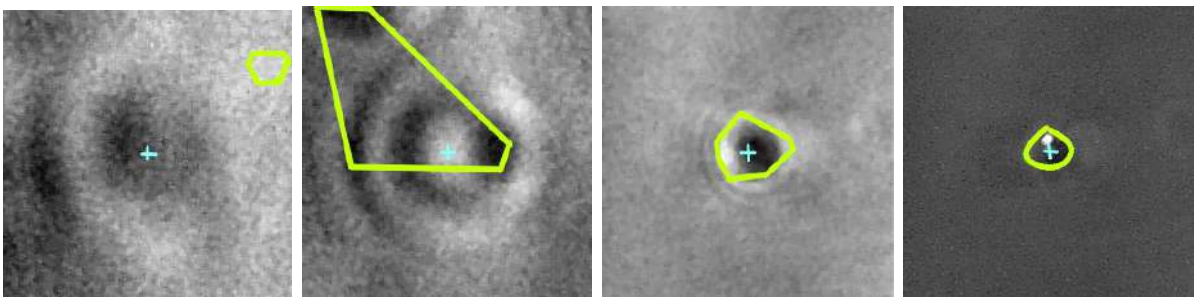


Figure 5.15: DefocusTracker control window for evaluating the particle recognition of the training set. The blue cross marks the middle of the particle (set by the user) and the green circle, where the program assumes the particle. On the left, two images with bad recognition can be seen. On the right, two particles were detected perfectly. Original lines plotted by DefocusTracker, redrawn with image editing tools for visibility.

Probably the training image set is not good enough; more images might be needed. DefocusTracker is also capable of calculating each particle's velocity. A visual cross-check is possible by a built-in function to plot the velocity vector as an orange arrow, see Figure 5.17.

There are many obvious sources for measurement errors. The capabilities of the human eye, when setting up the experiment and creating the training image set, for example. Others might be the optics of the camera or the microscope (resolution, frame rate, cleanness of the lens, manufacturing tolerances) or numerical inaccuracy of software used. Last but not least, the errors when determining the pixel-to-length ratio in x and y direction and especially the correct values of the z-scan are important. A higher channel might be more difficult to evaluate than a narrow one, since the particles can only be sharp for a certain amount of height. Also, if no numerical z-scan is performed, the z-component of the velocity vector gets neglected.

Finally, the evaluated velocity data together with the pressure and flow-rate border conditions, determined by the experimental set-up, can be fed into the next processing step: the MATLAB "checkerboard" for bringing together simulation- or experimental data with corresponding animal experiments. The same process can also be used to compare simulations with lab experiments. Implementation of such a data structure is the topic of the next and final chapter for this thesis.

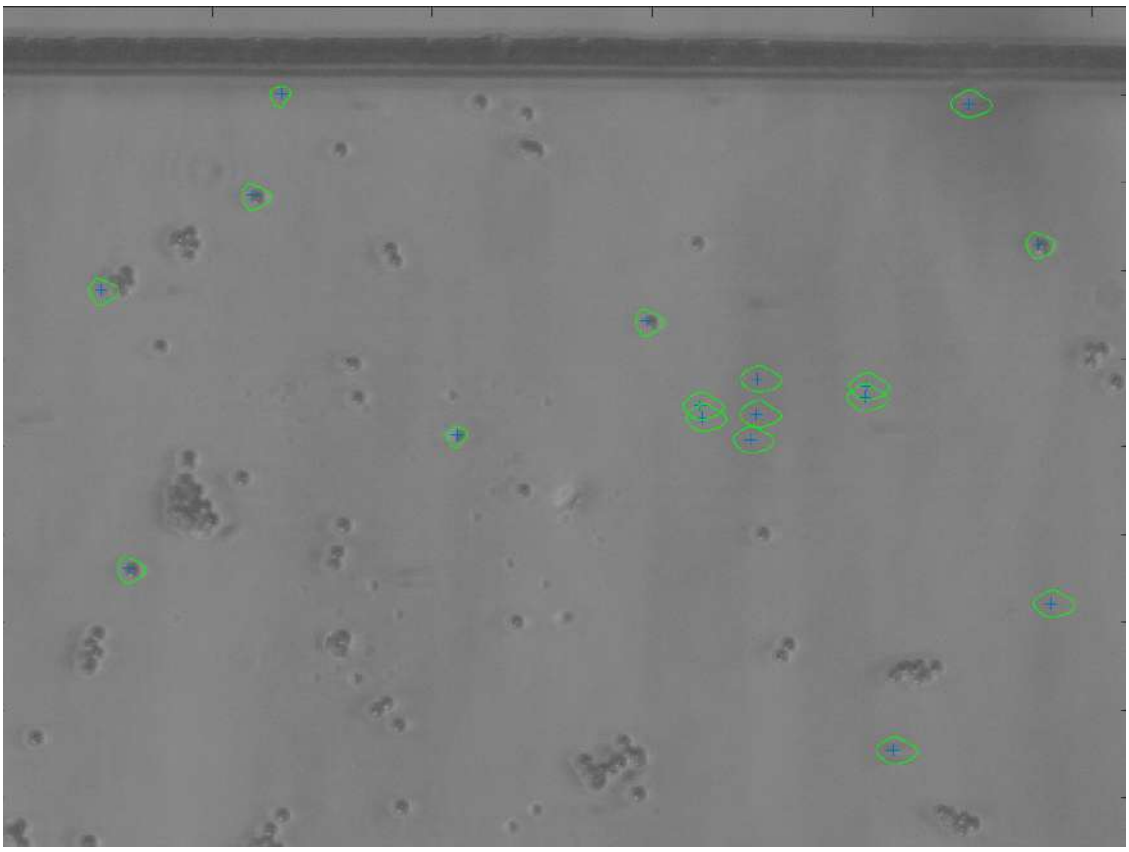


Figure 5.16: Dummy evaluation of a stationary image with particles dissolved in distilled water in a microfluidic channel. Notice especially the large group of non-existent particles on the right, while actual particles throughout the image get ignored. Lines drawn by DefocusTracker.

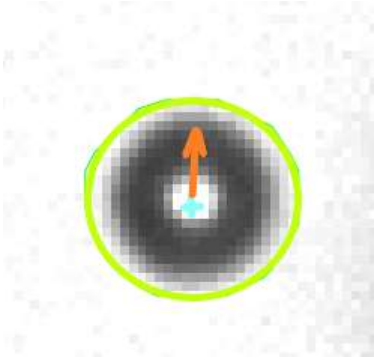


Figure 5.17: Exemplary image of a particle with an orange velocity arrow. Original lines plotted by DefocusTracker, redrawn with image editing tools for visibility.

6 Combining Biological and Fluid Mechanical Data

COMSOL model and microfluidic experiments of the two previous chapters served to generate fluid related physical parameters not available from conducting animal experiments alone. If the simulation or the microfluidic chip can be developed into a "market-ready" state, they can provide more insight into corresponding animal experiments. A relationship between formation of the thrombus and numerical shear rate values could be investigated.

Assuming this has been achieved, the question remains: How to bring the data together? To find an answer, first consider which type of output data each kind of experiment generates. The animal experiments are documented by taking images or videos of the blood vessel before, during and after inflicting the injury. A video is not a time-continuous signal, but rather pieced together by single images called frames. Hence, from now on, the terms "image", "frame" and "video" get used interchangeably when talking about data evaluation.

If the platelets have been marked with antibodies, the forming of the thrombus can be recorded. Additionally, objects such as the vessel wall, platelets RBCs etc. can be made visible. Each pixel will correspond to a physical length, depending on the spatial resolution of the image. To determine the resolution, a reference length could be used, but larger microscopes usually provide this information in the recording software. In conclusion, every pixel represents the information which biological object is present at a certain spatial position.

COMSOL simulations and microfluidic experiments will be a combination of location values and local physical parameters like the velocity vector, pressure or shear rate. While COMSOL will store spatial coordinates of the results automatically, for a microfluidic experiment the same rules apply, as just discussed for the animal experiments. In general, each data set will have its own zero-point, hence the difference vector between the two zero-points must be established. One way to do this would be to pick out a district location, like the middle of the wound. Then, coordinates can be matched according to the physical distance from the zero-point. If no exact match can be found, data can be taken from the proximate area. Each point now stores both biological and fluid mechanical data.

This chapter will show an implementation of a data structure capable of handling the tasks discussed above. MATLAB was chosen as an environment since it already comes with a lot of necessary subfunctions. The structure is named **"map"** because matching the data sets is like matching two maps of the same area but each containing different information. If both maps are tilted in the same way, share the same relative coordinate system, have the same orientation and use the same scaling, the task will be much easier. Ideally, this would also apply to the output of the conducted experiments.

Such a data structure can also be used for performing a biological computer simulation of hemostasis. Unfortunately, this could not be achieved during this thesis. Nevertheless, an idea for realizing a "thrombus prediction tool" based on machine learning trained with "maps" from real experiments will be presented in the outlook.

6.1 Structure of a Map

A geographic map is often separated into squares along longitude and latitude lines. Each square is uniquely identified by, for example, the bottom left corner coordinates. The same idea is used in the "map" structure. On an actual map each square contains for example a little bit of forest, roads and houses. One can discretize this information by assigning each square a singular value. For example, if the majority of an area is covered by forest the entire square is treated as a forest, as if no roads or houses were present. After such a discretization has taken place, one obtains a map looking like a checkerboard with a loss in spatial resolution but requiring less storage capacity.

In a way, the same concept is used by the "map". While technically each coordinate assigned to a square is just a singular point, its properties are assigned to the entire area of the square. A "map" is an array containing a tile-id, coordinates, fluidic properties and biological properties. Tile ID and coordinate are redundant information because all "map" tiles will be equidistant to each other. When a new "map" is initialized, no a-priori fluidic or biological information is assigned to each tile. To initialize a map, a user must provide its size and resolution. The size corresponds to the physical length of the images. The resolution can be chosen arbitrarily. A class constructor divides size by resolution and determines how many equidistant rectangles can fit in. Each rectangle is assigned its index and bottom left x- and y-coordinate. Figure 6.1 illustrates this process. A video can be understood as an ordered list of images. Each image is assigned a time value. The same is done to represent time transient data with "maps". In detail an object of the class "map" contains an array with time stamps, a storage of the current time stamp, and an array with checkerboards each one storing the information represented in Figure 6.1. A lot of different ways exist to implement a similar structure using various data types in numerous programming languages. Surely many improvements to the one presented here are possible. Appendix E states the code of the "map" class. One obvious way of improvement would be to relabel the class "map.m" as "atlas.m" and the variable listOfTables as listOfMaps.

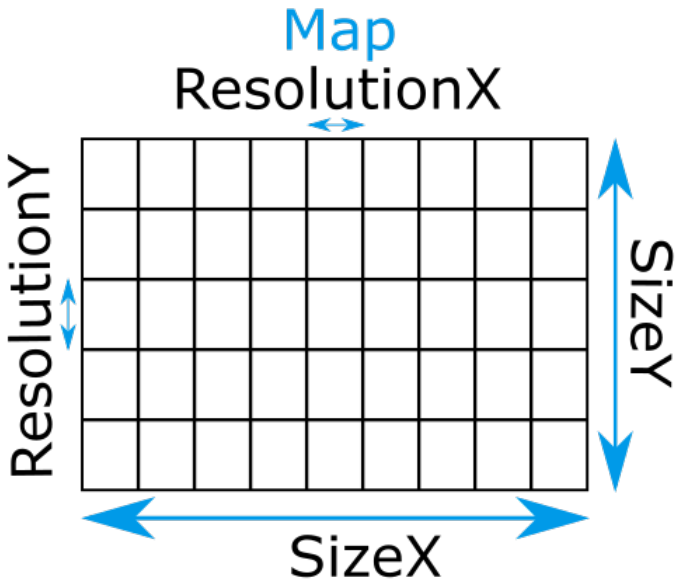


Figure 6.1: Construction of the "map" data type. It can be understood as a checkerboard, or a grid overlay of the pictures and simulation data. Each tile's size is determined by the picked resolution. The text in black corresponds to the variable names in the code and is therefore written as one word.

6.2 Including Fluid Data

In this subchapter, a concept for reading in output data of a fluid mechanical simulation will be presented. For microfluidic experiments, the results can be processed similarly. This thesis worked with 2D simulations, but simply adding a third Cartesian coordinate to the "map" tables allows processing 3D data in the same manner.

A fluid mechanical simulation will in one way or another discretize its geometry, for example by spanning a **mesh** filling up the simulation domain. Similar to graph theory, **nodes** and **edges** exist. Edges could be compared to strings connecting two nodes together. Like roads or highways, they might be uni- or bidirectional, with a corresponding directional orientation. Nodes are the spatial locations where a numeric evaluation of the differential equations take place. Edges represent which nodes are physically connected, meaning their differential equations are mathematically coupled. Figure 6.2 shows a screenshot for COMSOL with a mesh inside a defined geometry. One can already see the issue at hand: the nodes are not evenly distributed and most certainly not in an equidistant rectangular matter.

Many ways exist for matching a simulation mesh and the grid of a "map". Here the closest neighbour approach is chosen. Alternatives would be averaging over the closest n neighbour nodes or some form of weighted averaging, where each node influences an intersection of the grid more if they are closer together.

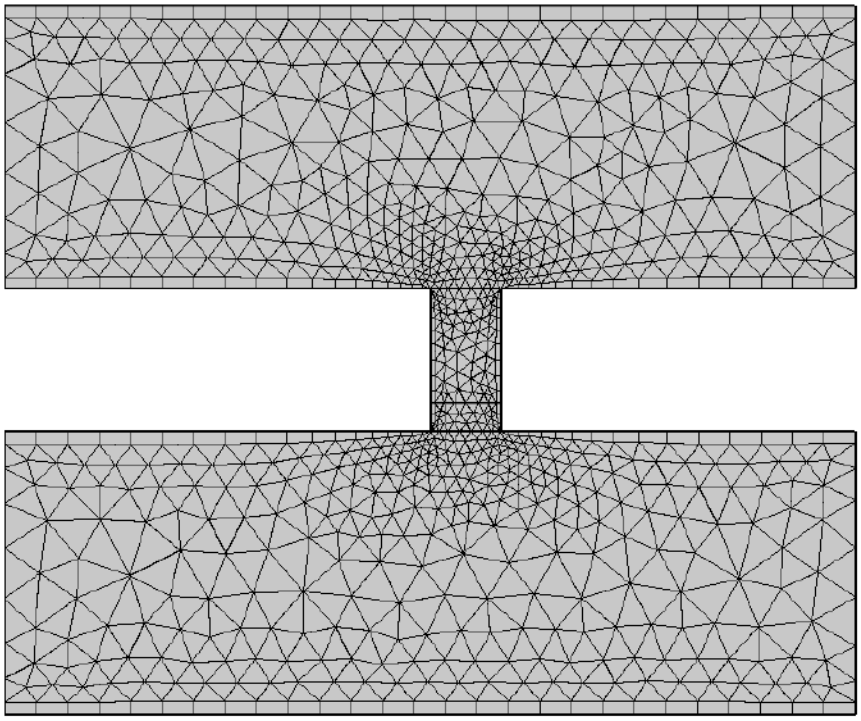


Figure 6.2: Example picture of a simulation mesh, here for COMSOL. It can be seen that the distribution of nodes is not equdistant but increases at the small channel connecting the two big ones. Screenshot taken from a COMSOL model of the H shaped microfluidic channel discussed in Chapter 5.1.3.

As a name for this program, **"discretizeMap.m"** was chosen. Figure 6.3 illustrates the algorithmic process of the code. For each grid intersection, represented by a small square, the program looks for a mesh nod in a vicinity of ε. This is done based on the coordinates of each intersection or node. A circle of radius ε is drawn around the interception. This corresponds mathematically to using the Euclidean norm $\|\cdot\|_2$, which is the most intuitive to use for this application. If two or more nodes can be found inside the circle, the closest one is chosen. This scenario is depicted by the upper right square in Figure 6.3. If a nod lies within two or more circles, and is the closest one available each time, it will get chosen by more than one intersection, illustrated by the two left squares in Figure 6.3. If no node exists within the radius, the intersection choses none. After a node has been chosen, its fluid mechanical properties get copied to the intersection. Appendix F contains the code of "discretizeMap.m".

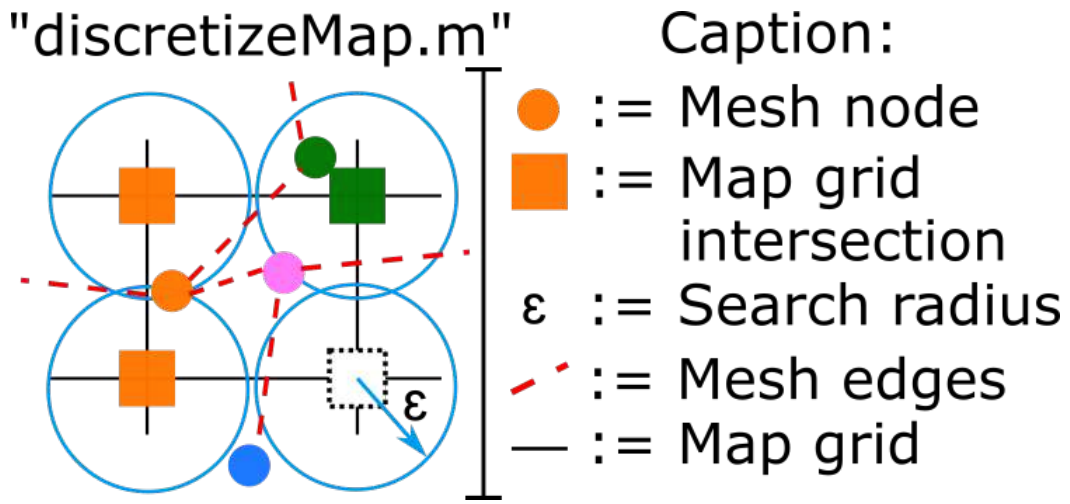


Figure 6.3: Visualisation of applying the closest neighbour approach to the problem of matching "map" intersections with mesh nodes. Different matching scenarios are shown on the right, the caption on the left.

It was already explained that there are certain areas for which a high density of mesh nodes exist. These tend to be the most critical areas of a simulation, for example the wound channel. As a result, the resolution of a "map" will in general be worse than the output data of the fluid simulation. To solve this, one would have to work with the mesh of the simulation software instead of an arbitrary grid approximation. In return, this might lead to problems later, when reading in image data from animal experiments, see next subchapter. Still, it becomes obvious that the number of "map" tiles is of major importance for accurately matching simulation- and experimental data. This can be seen in Figure 6.4. As a side note, it shall be mentioned that the code runs very slow. Better implementation of the idea presented in Figure 6.3 or the usage of parallel for-loops might be a solution.

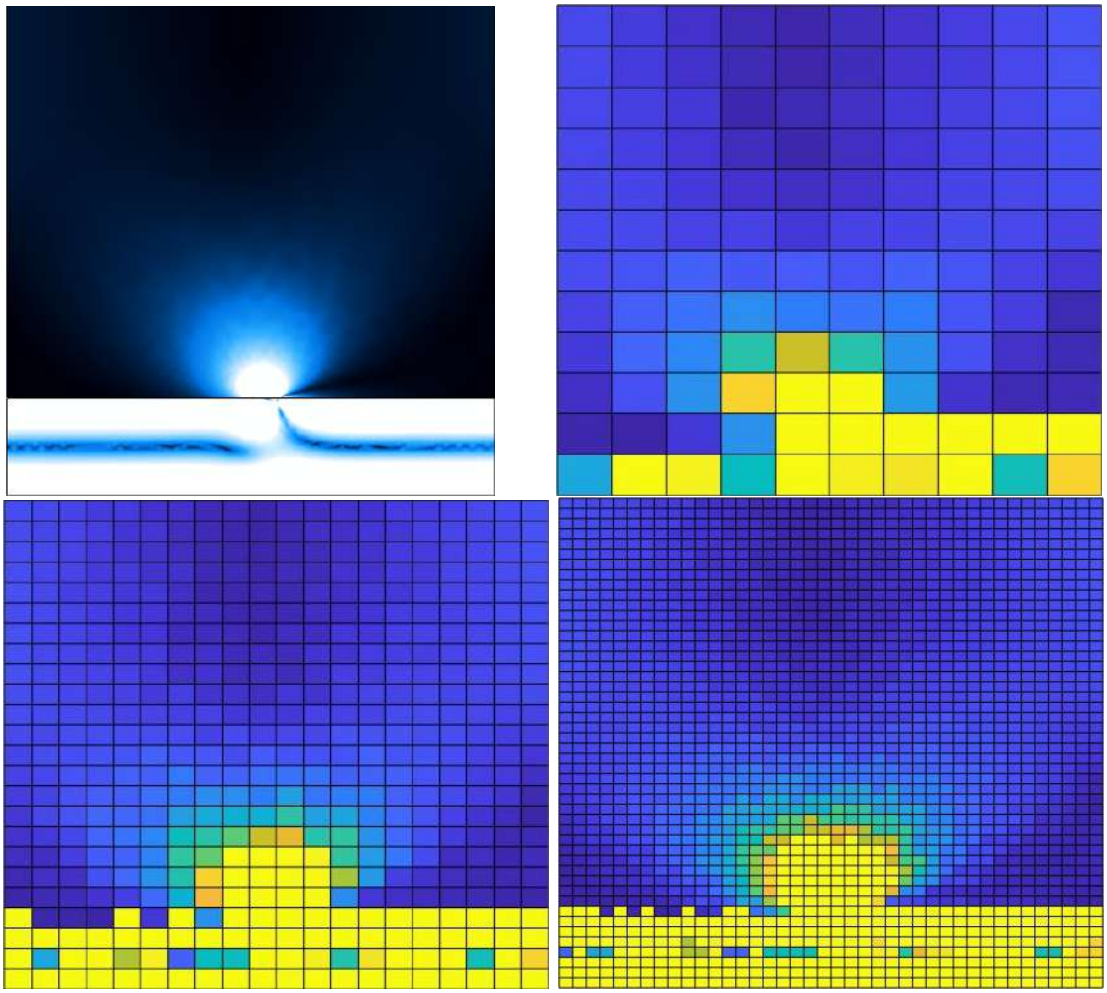


Figure 6.4: Pictures of an COMSOL simulation (top left) and its "discretization" to four "maps" with increasing resolution of 10x10, 20x20 and 40x40 tiles from top right to bottom left. Keep in mind that COMSOL uses a more sophisticated graphical representation, not representative of the actual resolution. Still, the resolution of the simulation will be better before loading it into a "map". The simulation results in this picture were taken from an earlier iteration of the COMSOL wound model.

6.3 Including Biological Data

While a mesh provides a better "resolution" of fluid mechanical data, having an equidistant rectangular grid makes reading in image data much easier. Every video can be understood as a sequence of images, as mentioned already, hence in suffices to show a way to read in a single image as a demonstration. For this a picture was kindly provided by Mr. Rossaro, medical department, LMU. It was taken during the performance of an in vivo experiment on a mouse whose blood vessel was injured with a needle.

It is assumed the user provides the spatial dimensions of the image and the bottom-left-corner coordinates with reference to the coordinate system of the "map". The next step is to cut the image along the edges of the grid and assign each tile the corresponding part of the image. Each cropped image is evaluated and the result stored in the corresponding tile. An alternative approach would be to evaluate the image first, assign each pixel or group of pixels a value and then project the data onto the "map". Both methods will have individual pros and cons in terms of image processing. The latter is not chosen here.

Cutting the image along the "map" grid works by determining the physical length each pixel represents. Since the user provides the dimensions of the image, this ratio can be easily calculated, since most images store their number of pixels in their metadata. One has to watch out for floating point errors during this and the next few steps, it makes sense to use appropriate scaling, for example using $[um]$ as a length unit instead of $[m]$. Since the distance of two grid intersection is known, the number of pixels that lie in between can be determined. Except for the border tiles, this number is always the same. At the borders, special attention must be paid to perform the cuts correctly. A demonstration of the image cutting can be seen in Figure 6.5. The processing steps considered so far are handled by the **"readInImage.m"** function. For an evaluation of the cropped images, the **"evaluateThrombus.m"** function gets called inside **"readInImage.m"**. This allows for a more flexible evaluation of the cropped images because the evaluation function can be exchanged in a simple manner. To save computation time, one should make sure to only call the evaluation function for tiles covered by at least parts of the image.

How the evaluated step looks like depends on what kind of information one wants to extract from the image. For this thesis, a simple approach shall suffice: a pixel is said to be "occupied by a thrombus" if it is white. Notice that via the "pixel to length" ratio, each pixel can also be understood as a coordinate. Coloured images usually use three or four values to store the colour of each pixel. The first three values represent the intensity of red, green or blue. This is done by assigning each a value from 0 to 255 representing no or full intensity. By the laws of colour mixing, each colour visible to the human eye can be represented. For more details, please refer to corresponding literature. To additionally store transparency, some image formats add a fourth value, describing a fully transparent (0) or fully coloured (255) pixel.

This fourth channel is not part of the evaluation. If a pixel has values of 255 for each colour channel, it is counted as "white". Alternatively, values from 245 to 255 for each channel could also be acceptable. As a next step, the number of white pixels is divided by the total number of pixels of the cropped image. No border effects appear for tiles only covered in part by the image, because "readInImage.m" always hands over the same number of pixels. Pixels not part of the image get set to zero for all colour channel, making them black technically. How many pixels fit in a tile is depending on the physical length depicted by a pixel and a tile, respectively.

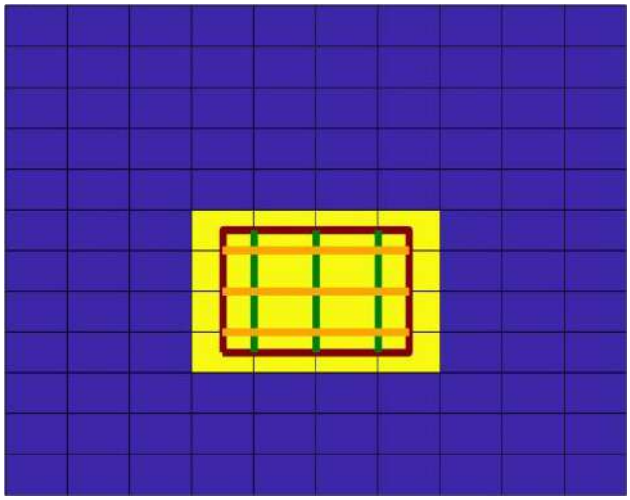


Figure 6.5: Demonstration of assigning parts of the image to corresponding "map" tiles. The image is drawn as a red square. Tiles that are at least partially affected by the image are highlighted in yellow, non-affected tiles are blue. Cuts in x- and y-direction are coloured green and orange, respectively.

The evaluation performed in this manner is very simplistic. The quote of white to non-white pixels gets multiplied by thousand, since scaling provided better results for graphical representation later. This value represents "how much thrombus can be found in a certain tile", by assuming that a denser thrombus structure leads to more white pixels. It shows the transition areas also, where a tile is partially occupied by a thrombus. Using a graphical output method, based on existing MATLAB functions, a comparison between an image projection and the results of reading in the image can be made. The result can be seen in Figure 6.6.

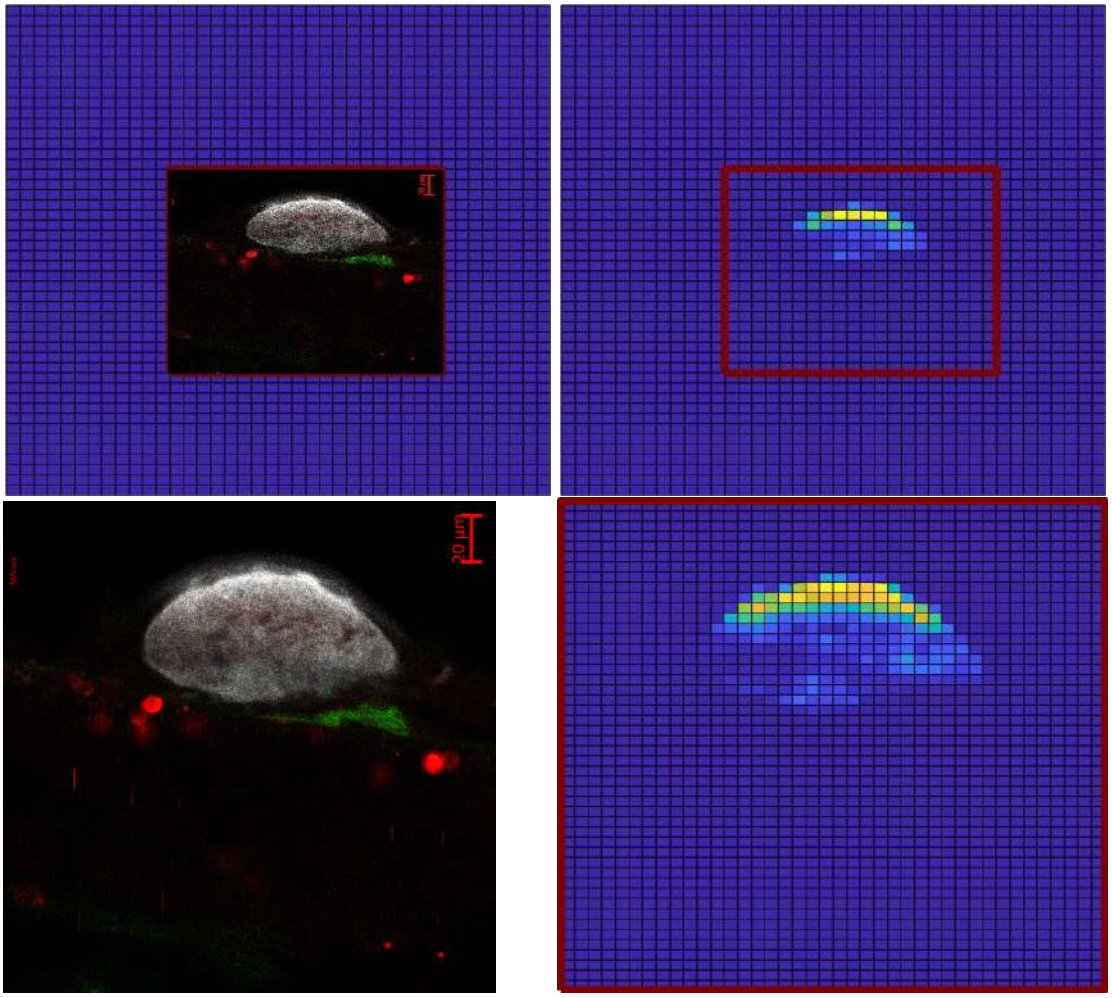


Figure 6.6: Comparison of plotting a thrombus image on a „map“ via a photo editing program (left) and the result of reading in the image with the method presented in this subchapter (right) for two different images scales (top and bottom).

6.4 Thrombus Simulation

The methods laid out above, a mere proof-of-concepts. While correlating thrombus location and local shear rates is possible, this was not done for an actual experiment, only with sample data. Originally, it was also planned to implement a form of superficial simulation of the thrombus formation. For this, mainly three components would be needed, presented in the next subchapters.

6.4.1 Fluid Mechanical Enhancements

Implementation of physical barriers like vessel walls, inlets and outlets are a next obvious step. A movement function for moving objects located on one tile at a certain time frame to another tile in the next time frame could also be included. It could be based on the velocity vectors of the original tile and the time between two frames. Particles must be able to stop at a wall tile or bounce off. For a physically correct movement, they also need to store their prior velocity to have a continuity of motion. If they reach an outlet, the particles must disappear. At every time instance, a new supply of particles must be generated at the inlet. Examples for such particles could be platelets or fibrinogen.

6.4.2 Biochemical Enhancements

Because of the required computation time, it is nonsensical to simulate individual particles. An alternative might be to store particle concentrations for each tile. A certain percentage of particles stays on a given tile or moves according to the local velocity vectors each time instance based on experimental statistics. Same applies for the concertation levels "spawned" at inlet and "deleted" at the outlet, representing the particles coming in from, or flowing out to the connected circulatory system. Under certain conditions, "chemical reactions" could take place on a tile, for example if the software predicts a thrombus will form here. To model the reaction, a "chemicalReaction.m" function could be written to evaluate each tile at each time instant and if a local Boolean variable is set to "true", the concertation of the products gets lowered and increased for the educts according to experimental statistics. Graphical output methods, like the one used for the creation of Figure 6.6, can then be used to visualize the thrombus forming over time.

6.4.3 Thrombus Prediction

The next milestone of the simulation would be to predict how a thrombus will form in terms of time and space for a given vessel-wound scenario. Without the involvement of animal or human experiments, one shall be able to predict how, for given boundary conditions, the thrombus formation process will look like. For this, a two-step process would be needed, if no solver for the fluid mechanical differential equations is to be implemented on the "map" structure. For example, this could be done by first performing a fluid simulation in COMSOL, and then loading the data on a "map". The fluid mechanical solution now serves as a basis for performing the biological and chemical simulations.

To make such predictions possible, using "maps" filled with animal experimental data and corresponding fluidic values could be used as a training set for a machine learning algorithm, similar to the training images of a single particle used in the previous chapter.

Implementing this idea will be difficult and time-consuming. Hundreds if not thousands of experiments might be needed for the generation of a training-set and this will not be the only problem. Such a large scale of animal experiments will be not doable for many reasons. But if the wound model of the microfluidic chip can be made accurate enough, a data set based on microfluidic experiments might be an alternative.

Use-case of such a "thrombus prediction software", might be an answer to medical questions of drug dosage and development. It was already outlined how chemical reactions could be implemented into the software. If one now has experimental data or predictions how a certain drug will influence these reactions, "drugged blood" can be included by changing the "chances" or "rates" inside the "chemicalReaction.m" code. This will provide insight into how certain drugs influence thrombus formation. Simulations might be set up in a way to depict typical medical scenarios, from accidentally cutting a finger to side injuries when performing heart surgery. For such surgeries, sometimes instruments need to be shoved through the patient's circulatory systems with long instruments to reach the heart and damage vessel tissue along the way. Such a procedure prevents the necessity to opening the torso.

6.5 Summary

The ideas presented in subchapter 6.4 remain an ambitious vision at this moment. Implementing them might be a task rather suited for one or multiple PhDs than for a single master thesis. Still, achieving at least some level of thrombus formation prediction might be worth the while. For now, only a correlation of thrombus formation and local shear rate is possible with the "map" data structure presented in this chapter. This chapter concludes the results of this master thesis; a brief final summary of all the methods for investigating fluid mechanical aspects of hemostasis will be given in the final chapter.

7 Outlook

In this final chapter a quick recapitulation of the theory, methods and applications of the **fluid mechanical aspects of hemostasis** will be provided. The starting-point of this thesis were animal experiments investigating **thrombus formation** for small blood vessel injuries. The most relevant physical factor for **platelets** to form a thrombus is the **shear rate** φ, the spatial derivation of the velocity vector. When performing these experiments, the shear rate cannot be measured. So complementary experiments or simulations must be performed. **Fluid mechanical simulations** and **microfluidic experiments** are two ways to do this. Setting them up properly requires an in-depth understanding of fluid mechanics and related biological processes. Chapter 2 thoroughly explored both.

Equipped with the theoretical foundation, Chapter 3 presented another necessary cornerstone: determining the **viscosity of human blood** by matching experimental data to blank analytical functions via **curve fitting**. The analytical functions need to be deduced from theory or experimental observations. Once a mathematical description of blood viscosity has been found, it can be used for the next step: the computer simulation or to create a mixture resembling blood for the microfluidic experiments. Determining the shear rate by means of simulation was shown in Chapter 4. In particular, the software **COMSOL** was used. Chapter 5 continued by presenting a **microfluidic artificial vessel** and provided ideas on how to include a wound model. **Tracking particles** moving in the fluid allows determining local velocities and hence shear rates. Additionally, ways for manufacturing concepts for microfluidic chips were discussed, together with **"DefocusTracker"**, a machine learning based approach to evaluate the video recordings of the experiments.

As a last step, Chapter 6 presented a data structure called **"map"** for bringing together animal experiment recordings with shear rate data generated by the means just mentioned. It was explained how this task can be understood as bringing together two geographical maps of the same location, but each map containing different information. For this, a reference coordinate-system and an accurate scale conversion of image length, meaning the physical lengths depicted in the image, are needed. This can be done by translating pixels into physical length. Last but not least, it was discussed how the same data structure could be used to provide a computer simulation of hemostasis to allow for rapid testing of drug effects on thrombus formation.

Each chapter already included discussions of possible improvements and what could be achieved by investing more time and resources into the respective topic. While the theory covers a lot of the relevant aspects already, describing the blood viscosity could use three major improvements: a bigger data set for the optimisation of the function coefficients, finding a better implementation of the **non-linear least squares algorithm** than **curve_fit()** from the NumPy library and deducing more accurate analytical blank functions from literature. Coupling of the influence of haematocrit, shear rate and temperature instead of just adding three terms together seems like a good idea.

Neither the COMSOL simulation nor the microfluidic experiments could achieve an acceptable accuracy in the time-scope of this thesis. For the fluid mechanic software, accurate modelling of the blood is key. Including the **Fahraeus-Lindqvist effect** for small vessels, or model RBCs flowing in blood as a liquid-gas emulsion for bigger vessels, is necessary. To achieve two- or multiphase flow models seems the most promising approach. Such a model would also allow the simulated blood to flow out of the vessel. The next step for the microfluidic experiments is to build a chip that includes a functional wound model. Ideas on how to construct a artificial wound were laid out. Another improvement would be to use blood instead of a metal-flake-water solution. Maybe simply taking away the calcium from blood is sufficient to allow its usage for the experiments. Then one could track platelets labelled by corresponding antibodies. "DefocusTracker" might still be suited for the evaluation of the video recordings, if a larger set of training images is created.

MATLAB has the advantage of providing a well-documented library of existing functions, especially graphical output methods. Hence, the "map" structure was implemented in this environment. Alternatives could be Python, Java or C++ each coming with an individual trade-off between complexity and runtime. All these languages also come with open-source functions, e.g., for graphical output methods. One should use them to save time and increase code stability. How chemical reactions and biological processes, could be implemented was discussed as a last point. Usage of **stochastic means** and assigning each tile an **average particle concentration** instead of simulating individual particles seems like the most promising approach.

Once all these methods reach a market-ready state, the applications would be numerous. Imagine for example a patient with a red thrombus inside a critical vessel. Urgent treatment with an anti-thrombotic drug is needed. At the same time, heart surgery needs to be performed. A long medical instrument needs to be pushed through the big veins inside the patient's legs to his or her heart. During such a procedure, damages to the vessel wall can occur. Will a thrombus form in time so that the blood loss is not critical? Answers to that could either be provided by taking a blood sample, adding the drug and running it through a microfluidic chip, or by inserting the influence of the drug as parameters in the "map" structure and running a simulation.

At this point, all of this is mere a glimpse into the future, but could be a powerful application worth devoting further research projects to.

A Python Script for Curve Fitting

Python script for determining the Coefficients of the functions in Chapter 3. For the code of the curve_fit() function, please refer to [git22b]. Colours are chosen in correspondence with the Spyder IDE. The code is only valid for one function as well as one starting guess and must be changed accordingly.

```
# -*- coding: utf-8 -*-
"""Title: Blood Viscosity Function Coefficient Optimization (Python)
Author: Lothar Brixius
Date: 08.03.22"""
```

```
#Import necessary libraries
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit

#Inialize variables with zeros, 99 is the number of data points.
t = np.zeros(99) #array of float64
h = np.zeros(99)
p = np.zeros(99)

#Define function to optimize
def func(X, a, b, c, d, e, f, g):
    t,h,p = X
    return a*np.exp(b*t) + c*h**2 + d*h + e * np.exp(f*p) + g
# Read in reference data
data = pd.read_excel("Table-human-blood-viscosity.xlsx")
t = data["Temperature"].to_numpy(float)
h = data["Hematocrit"].to_numpy(float)
p = data["Shear-Rate"].to_numpy(float)
v = data["Viscosity"].to_numpy(float)

# Initial guesses for the coefficients a, b, c, d, e, f, g:
p0 = 901, -147, 0.00155, -0.435, -1.31e-6, -100000, 31.04
print(curve_fit(func, (t,p,h), v, p0))
```

B PC Specs

Hardware, OS and IDE used for running the code from Appendix B and E in Chapter 3. Theoretically this should not influence the results.

Manufacturer: HP

System Model: OMEN Laptop 15-en1xxx

System Type: System Type x64-based PC

Operating System: Microsoft Windows 10 Pro, Version 10.0.19043 Build 19043 CPU: Processor: AMD Ryzen 5 5600H with Radeon Graphics, 3301 MHz, 6 Core(s), 12 Logical Processor(s)

GPU: NVIDIA GeForce RTX 3060 Laptop GPU, Driver Version: 472.47, Driver Type: DCH

RAM: DDR4, 16 GB, Channel: 2 x 64-bit, DRAM Frequency: 1597.3 MHz

Hard Drive: SAMSUNG MZVLB512HBJQ-000H1 477 GB SSD

IDE: Spyder IDE 5.1.5, Python 3.9.7 64-bit, Qt 5.9.7, PyQt5, 5.9.2, Windows 10, MATLAB R2021b

C Convergence Table

"Convergence table" for function 1 from Chapter 3. Upper row denotes starting vector, lower row denotes resulting vector. Results come from the code in Appendix B. Red denotes a dead end, no solution could be determined or another error. Green denotes a step with successful convergence of a coefficient or a coefficient that has been already successfully determined. Light blue denotes the coefficient we look at next. Yellow denotes a non-determined coefficient. Pink denotes the end results:

a	b	c	d	e	f	g
1. Step: Logarithmic Search						
1	1	1	1	1	1	1
5505	-147	5170	-19332	-1.30E-17	1	3.00E+07
10	10	10	10	10	10	10
100	100	100	100	100	100	100
0.1	0.1	0.1	0.1	0.1	0.1	0.1
1.74	-13.8	0.0012	-0.32	-0.278	0.058	15.1
0.01	0.01	0.01	0.01	0.01	0.01	0.01
-1	-1	-1	-1	-1	-1	-1
-10	-10	-10	-10	-10	-10	-10
-10	-10	0.00155	-0.435	-9.9	-10	31.04
-100	-100	-100	-100	-100	-100	-100
-100	-100	0.0015	-0.435	-9.9	-100	31.04
-1000	-1000	-1000	-1000	-1000	-1000	-1000
-1000	-1000	0.0015	-0.435	-9.9	-1000	31.04
-0.1	-0.1	-0.1	-0.1	-0.1	-0.1	-0.1
901	-439	0.00155	-0.435	-9.9	-668	31.04
-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
7.23	-24.9	0.00155	-0.435	-9.9	-15.02	31.04
2. Step Confirming Coefficients c, d, e, g						
a	b	c	d	e	f	g
5500	-147	0.00155	-0.435	-9.9	-15.02	31.04
1.7	-13.8	0.0012	-0.32	-0.278	-668	15.1
901	-439	5170	-1.90E+07	-1.31E-17	0.058	3.00E+07
7.23	-24.9					

Figure C.1: Convergence table, page 1.

5500	b all combination tried. Input = output	0.00155	-0.435	-9.9	-15.02	31.04
901	-147	0.0012	-0.435	-9.9	-15.02	31.04
901	-147	0.00155	-0.435	-9.9	-15.02	31.04
901	-147	5170	-0.435	-9.9	-15.02	31.04
901	-147	0.00155	-0.435	-9.9	-15.02	31.04
901	-147	0.00155	-0.435	-9.9	-15.02	15.1
901	-147	0.00155	-0.435	-9.9	-15.02	31.04
901	-147	0.00155	-0.435	-9.9	-15.02	3.00E+07
901	-147	0.00155	-0.435	-9.9	-15.02	31.04
901	-147	0.00155	-0.32	-9.9	-15.02	31.04
901	-147	0.00155	-0.435	-9.9	-15.02	31.04
901	-147	0.00155	-1.90E+07	-9.9	-15.02	31.04
901	-147	0.00155	-0.435	-9.9	-15.02	31.04
901	-147	0.00155	-0.435	-1.31E-17	-15.02	31.04
901	-147	0.00155	-0.435	-1.31E-17	-15.02	31.04
901	-147	0.00155	-0.435	-1.31E-06	-15.02	31.04
901	-147	0.00155	-0.435	-9.9	-15.02	31.04
3. Try to determine coefficient f						
a	b	c	d	e	f	g
901	-147	0.00155	-0.435	-9.9	-668	31.04
901	-147	0.00155	-0.435	-9.9	-668	31.04
901	-147	0.00155	-0.435	-9.9	0.058	31.04
901	-147	0.00155	-0.435	-9.9	-417.1	31.04
901	-147	0.00155	-0.435	-9.9	-300	31.04
901	-147	0.00155	-0.435	-9.9	-300	31.04

Figure C.2: Convergence table, page 2.

901	-147	0.00155	-0.435	-9.9	-400	31.04
901	-147	0.00155	-0.435	-9.9	-400	31.04
901	-147	0.00155	-0.435	-9.9	-417	31.04
901	-147	0.00155	-0.435	-9.9	-417	31.04
f:	1	-1	-10	-100	-1000	-10,000
	-100,000	-0.1	-0.01	-0.001	-0.0001	
901	-147	0.00155	-0.435	-9.9	-0.01	31.04
901	-147	0.00155	-0.435	-9.9	-80652	31.04
901	-147	0.00155	-0.435	-9.9	-0.001	31.04
901	-147	0.00155	-0.435	-9.9	-21761	31.04
901	-147	0.00155	-0.435	-9.9	-80652	31.04
901	-147	0.00155	-0.435	-9.9	-80652	31.04
901	-147	0.00155	-0.435	-9.9	-21761	31.04
901	-147	0.00155	-0.435	-9.9	-21761	31.04
4. Try to determine coefficient a						
a	b	c	d	e	f	g
5500	-147	0.00155	-0.435	-9.9	-668	31.04
5500	-147	0.00155	-0.435	-9.9	-668	31.04
1.7	-147	0.00155	-0.435	-9.9	-668	31.04
1.7	-147	0.00155	-0.435	-9.9	-668	31.04
7.23	-147	0.00155	-0.435	-9.9	-668	31.04
7.23	-147	0.00155	-0.435	-9.9	-668	31.04
1.00E-09	-147	0.00155	-0.435	-9.9	-668	31.04
1.00E-09	-147	0.00155	-0.435	-9.9	-668	31.04
a:	1.00E-08	1.00E-07	1.00E-06	1.00E-05	1.00E-04	1.00E-03
	1.00E-02	1.00E-01	1	10	1.00E+02	1.00E+03
	1.00E+04	1.00E+05	1.00E+06	1.00E+07	1.00E+08	1.00E+09
5. Try MATLAB Brute-Force (based on the average relative error)						

Figure C.3: Convergence table, page 3.

Possibilities:								
a	b	c	d	e	f	g		Error:
5500	-147	0.00155	-0.435	-9.9	-15.02	31.04		
1.7	-13.8				-668			
901	-439				0.058			
7.23	-24.9							
1.00E-09	1.00E-09				1.00E-09			
to	to				to			
1.00E+09	1.00E+09				1.00E+09			
-10	1.00E-04	0.00155	-0.435	-9.9	1.00E-04	31.04		183%
-7	0.009	0.00155	-0.435	-9.9	7.00E-04	31.04		182%

Figure C.4: Convergence table, page 4.

D LIIB MATLAB Source-Code

MATLAB Code to brute-force the coefficients that could not be determined by Python for Function 1. Same code for implementing LIIB. Colouring of the code according to MATLAB R2021b. Code is just an example for Function 1 and for a given interval and needs to be changed accordingly. Also, ". . . " denotes missing entries in the code to increase readability.

```
% 2.1: Data read-in and Variable creation
inputData = readtable("Table-human-blood-viscosity.xlsx");
v = zeros(height(inputData),1);
v1 = zeros(height(inputData),1);
t = zeros(height(inputData),1);
h = zeros(height(inputData),1);
sR = zeros(height(inputData),1);
error = zeros(99,1);
arrayA = [-1000000,…, -0.0001, 0.0001,…, 1000000];
arrayB = [-1000000,…, -0.0001, 0.0001,…, 1000000];
…
arrayF = [-1000000,…, -0.0001, 0.0001,…, 1000000];
solutionTable = cell(max(size(arrayA))*max(size(arrayB))*…*max(size(arrayF)),5);
counter = 1;

% 2.2: Variable fill-in From Data
for j = 1 : 1 : height(inputData)
    v(j) = table2array(inputData(j, 5));
    t(j) = table2array(inputData(j, 2));
    h(j) = table2array(inputData(j, 4));
    sR(j) = table2array(inputData(j, 3));
end

%Calculations
for j = 1 : 1 : max(size(arrayA))
    a = arrayA(j);
    for k = 1 : 1 : max(size(arrayB))
        b = arrayB(k);
        …
        for l = 1 : 1 : max(size(arrayF))
            f = arrayF(l);
            for i = 1 : 1 : 99
                v1(i) = a*exp(b*t(i)) + c*h(i).^2 + d*h(i) + e*exp(f*sR(i)) + g;
                error(i) = abs((v1(i)-v(i))/v(i));
            end
            errorSum = sum(error(:))/99;
            solutionTable{counter,1} = a;
            solutionTable{counter,2} = b;
            …
            solutionTable{counter,3} = f;
            solutionTable{counter,4} = error;
            solutionTable{counter,5} = errorSum;
            counter = counter + 1;
            if counter > (max(size(arrayA))*max(size(arrayB))*max(size(arrayF)))
                warning("check counter");
            end
        end
      …
    end
end
```

Figure D.1: LIIB implementation code.

E Map.m Source-Code

Code for “Map.m”. Colouring of the code in accordance with MATLAB R2022a.

```
%Title: Map Class; Author Lothar Brixius; Date: 29.12.21
%--------------------------------------------------------------------------
classdef Map
  properties
    nrOfTimeStamps = 0; %Stores how many timestamps there are for the simulation.
    timeStampArray double; %Stores all timestamps as doubles in [s] and their corresponding Nr.
    currentTimeStamp double; %Stores the current timestamp, where we are
    listOfTables cell; %Reads in all coordinate-tables from COMSOL, depicts the "physic simulation"
    sizeX double; %Size of the simulated area in x direction in [m].
    sizeY double; %Size of the simulated area in x direction in [m].
    resolutionX double; %Resolution in X direction in [m]
    resolutionY double; %Resolution in X direction in [m]
    nrOfMapTilesX int64; %Used when discretizing the map. round() to make it an integer
    nrOfMapTilesY int64; %Used, when discretizing the map. round() to make it an integer
  end

  methods
    function outputMap = Map(inputSizeX, inputSizeY, inputResolutionX, inputResolutionY)
      %Enforce inputs to be doubles:
      arguments
        inputSizeX (1,1) double
        inputSizeY (1,1) double
        inputResolutionX (1,1) double
        inputResolutionY (1,1) double
      end
      if nargin < 4
      % Presenting no arguments makes no sense aside from debugging
      error("Enter the correct number of variables. Size & resolution are needed for both x and y direction.");
      elseif inputSizeX <= 0 || inputSizeY <= 0 || inputResolutionX <= 0 || inputResolutionY <= 0
        error("Please enter positive values for all input parameters.");
      else
        outputMap.sizeX = round(inputSizeX, nrOfDecimalDigits(inputSizeX) + 3);
        outputMap.sizeY = round(inputSizeY, nrOfDecimalDigits(inputSizeY) + 3);
        outputMap.resolutionX = round(inputResolutionX, nrOfDecimalDigits(inputResolutionX) + 3);
        outputMap.resolutionY = round(inputResolutionY, nrOfDecimalDigits(inputResolutionY) + 3);
        %Determine checkerboard dimensions +1 for 0 row/column
        outputMap.nrOfMapTilesX = round(outputMap.sizeX/outputMap.resolutionX) +1;
        outputMap.nrOfMapTilesY = round(outputMap.sizeY/outputMap.resolutionY) +1;

        %Inform the user if one of the input variables was rounded.
        if ~isequal(outputMap.sizeX, inputSizeX)
          warning("Map size in x direction was rounded to prevent errors when using functions of the map.");
        elseif ~isequal(outputMap.sizeY, inputSizeY)
          warning("Map size in y direction was rounded to prevent errors when using functions of the map.");
        elseif ~isequal(outputMap.resolutionX, inputResolutionX)
          warning("Map resolution in x direction was rounded to prevent errors when using map functions.");
        elseif ~isequal(outputMap.resolutionY, inputResolutionY)
          warning("Map resolution in y direction was rounded to prevent errors when using map functions.");
        end
      end
    end
  end
end
```

Figure E.1: Map.m Source-Code

F DiscretizeMap.m and findClosestPoint.m Source-Code

Code for “discretizeMap.m” and “findClosestPoint.m”. Colouring of the code in accordance with MATLAB R2022a.

```
function inputMap = discretizeMap(inputMap, inputTable, mapNr)
    c = width(inputTable) + 1; %stores number of columns, +1 for the storing the amount of thrombus

    %Check if the columns have the correct names
    stringStorage = inputTable.Properties.VariableNames;
    if ~isequal(stringStorage, ["x","y","sR","Velocity","Pressure","u","v"])
       error("Column names of the input table are not correct.");
    end
    stringStorage = ["x","y","sR","Velocity","Pressure","u","v","Thrombus"];

    %Initialize blank storage cell array
    inputMap.listOfTables{mapNr} = cell(inputMap.nrOfMapTilesX, inputMap.nrOfMapTilesY);
    %At each checkerboard field create a table to store coordinates and physical values.

    %Conveinence stuff, necessary because of Debugging
    varTypesStorage = string(zeros(1,c));
    for i = 1:c
       varTypesStorage(1,i) = "double";
    end

    %Fill the tables with Coordinates and NaN values
    for i = 1:inputMap.nrOfMapTilesX
       for j = 1:inputMap.nrOfMapTilesY
          inputMap.listOfTables{mapNr}{i,j} = table('Size', [1,
c],'VariableTypes',varTypesStorage,'VariableNames', stringStorage);

          %Store x and y coordinates at 1. and 2. position
          inputMap.listOfTables{mapNr}{i,j}(1,1) = array2table(double(i-1)*inputMap.resolutionX);
          inputMap.listOfTables{mapNr}{i,j}(1,2) = array2table(double(j-1)*inputMap.resolutionY);
          %Fill in NaN values for how many variables/columns the inputTable has.
          for k = 3:1:c
             inputMap.listOfTables{mapNr}{i,j}(1,k) = array2table(NaN);
          end
       end
    end

%Now try to find a close point and store its values. A lot of improvement can be done here. E.g., don't just take
%the closest point but create an average value over the closed n points and/or weight them by their distance
    for i = 1:1:inputMap.nrOfMapTilesX
       for j = 1:1:inputMap.nrOfMapTilesY
          a = double(i)*inputMap.resolutionX;
          b = double(j)*inputMap.resolutionY;
          l = findClosestPoint(inputTable, a, b); %returns NaN if no closest point was found in a certain radius.
          if ~isnan(l) %if a meaningful point was found / l is NOT NaN
             %fill the array with the values stored at l
             for k = 3:1:(c-1) %c-1 because thrombus cannot be read from COMSOL
                inputMap.listOfTables{mapNr}{i,j}(1,k) = inputTable(l,k);
             end
          end
       end
    end
end
```

Figure F.1: discretizeMap.m Source-Code

```
function positionInInputTable = findClosestPoint(inputTable, x, y)
  %findClosestPoint Finds the closest Point in a Table for given x and y coordinates. Assumes that the first
  %columns of inputTable hold the coordinates. Expects these coordinates to be in [m]. Returns the index of
  %the best match found inside a radius of epsilon or NaN if no match is found inside this radius.

  %Required variables
  positionInInputTable = NaN;
  storage = [NaN, NaN]; %store best result so far
  [rows, columns] = size(inputTable);
  epsilon = 0.01; %search radius in [m]

  %Main loop
  for i = 1:rows %loop through each row of inputTable
    %Dummy storage, makes code more readable
    x1 = table2array(inputTable(i,1));
    y1 = table2array(inputTable(i,2));

    if sqrt((x-x1)^2 + (y-y1)^2) < epsilon
    %Only consider a point in a one cm radius. The fields outside the simulated plane must not be filled!
    a = storage(1,1);
    b = storage(1,2);
      if isnan(a) || isnan(b) %Can't check storage in 1 step. First time a good point is found, storage is empty
        positionInInputTable = i;
        storage = [x1, y1];
      else %Notice that elseif can't be used because storage could be NaN
        if abs(x-x1) + abs(y-y1) < abs(x-storage(1)) + abs(y-storage(2))
%If we found a better solution use this one. Abs is used instead of explicit definition above without reason
          positionInInputTable = i;
          storage = [x1, y1];
        end
      end
    end
  end
end
```

Figure F.2: findClosestPoint.m Source-Code

G ReadInPicture.m and evaluateThrombusInImage.m Source-Code

Code for “readInPicture.m” and “evaluateThrombusInImage.m”. Colouring of the code in accordance with MATLAB R2022a.

```
function outputMap =
readInPicture(inputMap,inputTimestamp,inputImageFilename,inputImageCoordinates,inputImageFormat)
%READINPICTURE Load an image of a thrombus into a map. Please insert the input image coordinates in the
%following way: inputImageCoordinates = [x1, y1, x2, y2], where x1 = bottom-left corner x-coordinate y1 =
%bottom-left corner y-coordinate x2 = top-right corner x-coordinate y2 = top-right corner x-coordinate Author:
%Lothar Brixius Date: 16.02 I. Check input parameters for correctness 1. Enforce correct input format
arguments
    inputMap Map; %Insert default map
    inputTimestamp int8 = 1; inputImageFilename string = "test.txt";
    inputImageCoordinates (1,4) double = [0,0,0,0];
    inputImageFormat string = "PNG";
end
% 1.1 Split up the input image coordinates for code readability. Round the image size to three digits after the
first significant digit of the map resolution to prevent rounding errors further down.
bottomLeftImageXCoordinate=rnd(inputImageCoordinates(1,1),nrOfDecimalDigits(inputMap.resolutionX) + 3);
bottomLeftImageYCoordinate=rnd(inputImageCoordinates(1,2),nrOfDecimalDigits(inputMap.resolutionY)+3);
topRightImageXCoordinate = inputImageCoordinates(1,3);
topRightImageYCoordinate = inputImageCoordinates(1,4);
% 2. Check if the image positions are correct. 2.1 Check if image is properly stretched
if bottomLeftImageXCoordinate >= topRightImageXCoordinate
  error("Please enter bottom-left x-coordinate for the image that is smaller than the top-right x-coordinate.");
elseif bottomLeftImageYCoordinate >= topRightImageYCoordinate
  error("Please enter a bottom-left y-coordinate for the image that is < the top-right y-coordinate.");
end
% 2.2 Check if all coordinates are >= 0
if inputImageCoordinates(1,1) < 0 || inputImageCoordinates(1,2) < 0 || inputImageCoordinates(1,3) < 0 ||
inputImageCoordinates(1,4) < 0
    error("All image coordinates must be greater than 0");
end
% 2.3 Check if image lies inside the map
if bottomLeftImageXCoordinate > inputMap.sizeX || topRightImageXCoordinate > inputMap.sizeX
    error("All X Coordinates of the Picture must lay inside the borders of the Map.");
end
if bottomLeftImageYCoordinate > inputMap.sizeY || topRightImageYCoordinate > inputMap.sizeY
    error("All Y Coordinates of the Picture must lay inside the borders of the Map.");
end
% 3. Check input map
if isempty(inputMap.listOfTables)
    error("Please read in a COMSOL data set first.");
end
% Separate call to prevent error of a "void call". We assume that if {1,1} is filled all {k,l} are filled
if isempty(inputMap.listOfTables{1,1})
    error("Please read in a COMSOL dataset first.");
end
% II. Create Additional Help Variables and Read in Image 1. Create Map Frame storage for III. %WARNING:
%floating point numbers require special treatment when using <=, == or >=.
  bottomLeftMapXIndex = int64(1); bottomLeftMapYIndex = int64(1);
  topRightMapXIndex = inputMap.nrOfMapTilesX; topRightMapYIndex = inputMap.nrOfMapTilesY;
% 2. Create cut index variables for IV / V. int64 needed as typecasting for additions with other index variables.
  currentCutIndexX = int64(1); currentCutIndexY = int64(1);
% 3. Read in the image and create image processing variables. Please notice that [x,y]=size() causes a bug for
%images with more than one layer, e.g., RGB images. Also [x,y] needs to be typecasted to int64 for index
%calculation further below. The image needs to be 90° clockwise (mathematically negative)
```

Figure G.1: readInPicture.m Source-Code, page 1.

```
    inputImage = imread(inputImageFilename,inputImageFormat);
    inputImage = imrotate(inputImage, -90);
    [nrOfPixelsInputImageX, nrOfPixelsInputImageY, nrOfColorLayersInputImage] = size(inputImage);
    nrOfPixelsInputImageX = int64(nrOfPixelsInputImageX);
    nrOfPixelsInputImageY = int64(nrOfPixelsInputImageY);
    nrOfColorLayersInputImage = int64(nrOfColorLayersInputImage); %for debugging
lengthPerPixelRatioX=(topRightImageXCoordinate-bottomLeftImageXCoordinate)/dbl(nrOfPixelsInputImageX);
lengthPerPixelRatioY=(topRightImageYCoordinate-bottomLeftImageYCoordinate)/ dbl(nrOfPixelsInputImageY);
% III. Determine the Closest Map Tiles. In order to save computation time, we look in advance which tiles will
%be affected by image, meaning which tiles are covered entirely or in parts with the image. For this reason,
%we determine the upper left and bottom rightmost tile, that still cover parts of the image. Dynamic indexing
%is needed, depending how many edges of the picture lie on edges of the map.
%isSmallerOrEqualFloatingPointNumber() and isBiggerOrEqualFloatingPointNumber() are my own creation.
%Make sure MATLAB can access the corresponding files. 1. Bottom-left x index of the first tile in contact with
%the image. array2table() necessary because MATLAB hasn`t implemented soft type casting properly. Check if
%one more tile will cut into the image. Assumes all rows of a map have the same distance:
while
isSmallerOrEqualFloatingPointNumber(table2array(inputMap.listOfTables{inputTimestamp}{bottomLeftMapXI
ndex + 1, 1}(1,1)),bottomLeftImageXCoordinate)
    bottomLeftMapXIndex = bottomLeftMapXIndex + 1;
    if bottomLeftMapXIndex > inputMap.nrOfMapTilesX
      error("Trying to evaluate outside the map.");
    end
  end
% 2. Bottom-left y index of first tile in contact with image. Assumes all columns of map have same distance:
while isSmallerOrEqualFloatingPointNumber(table2array(inputMap.listOfTables{inputTimestamp}{1,
bottomLeftMapYIndex + 1}(1,2)), bottomLeftImageYCoordinate)
    bottomLeftMapYIndex = bottomLeftMapYIndex + 1;
    if bottomLeftMapYIndex > inputMap.nrOfMapTilesY
      error("Trying to evaluate outside the map.");
    end
end
% 3. Upper-right x index of last tile in contact with image. Check if subtracting 1 more tile will cut into image
while
isBiggerOrEqualFloatingPointNumber(table2array(inputMap.listOfTables{inputTimestamp}{topRightMapXInde
x - 1, 1}(1,1)),topRightImageXCoordinate)
    topRightMapXIndex = topRightMapXIndex - 1;
end
% 3.1 The while loop stops one instance too early. For this reason, we need to subtract 1 more tile.
topRightMapXCoordinate = table2array(inputMap.listOfTables{inputTimestamp}{topRightMapXIndex,
topRightMapYIndex}(1,1));
if topRightMapXIndex > 1 && isBiggerOrEqualFloatingPointNumber(topRightMapXCoordinate,
topRightImageXCoordinate)
    topRightMapXIndex = topRightMapXIndex - 1;
end
% 4. Upper-right y index of the last tile in contact with the image.
while isBiggerOrEqualFloatingPointNumber(table2array(inputMap.listOfTables{inputTimestamp}{1,
topRightMapYIndex - 1}(1,2)),topRightImageYCoordinate)
    topRightMapYIndex = topRightMapYIndex - 1;
end
% 4.1 The while loop stops one instance too early again. We need to subtract one more tile.
topRightMapYCoordinate = table2array(inputMap.listOfTables{inputTimestamp}{topRightMapXIndex,
topRightMapYIndex}(1,2));
```

Figure G.2: readInPicture.m Source-Code, page 2.

```
if (topRightMapYIndex > 1) && isBiggerOrEqualFloatingPointNumber(topRightMapYCoordinate,
topRightImageYCoordinate)
    topRightMapYIndex = topRightMapYIndex - 1;
end
% 5. Update the map tile coordinates. table2array() because of typecasting.+1 needed because we always have
%the zero-coordinate row/column at index 1,i or j,1 in a map. Please refer map class and discretizeMap.m.
  bottomLeftMapXCoordinate = table2array(inputMap.listOfTables{inputTimestamp}{bottomLeftMapXIndex,
bottomLeftMapYIndex}(1,1));
  bottomLeftMapYCoordinate = table2array(inputMap.listOfTables{inputTimestamp}{bottomLeftMapXIndex,
bottomLeftMapYIndex}(1,2));
  topRightMapXCoordinate = table2array(inputMap.listOfTables{inputTimestamp}{topRightMapXIndex,
topRightMapYIndex}(1,1));
  topRightMapYCoordinate = table2array(inputMap.listOfTables{inputTimestamp}{topRightMapXIndex,
topRightMapYIndex}(1,2));
% 6. Determine pixel offset for V. Use ceil to make sure a pixel is lost rather than having a void call.
  leftPixelOffset = int64(ceil((bottomLeftImageXCoordinate - bottomLeftMapXCoordinate) /
lengthPerPixelRatioX));
  rightPixelOffset = int64(ceil((topRightMapXCoordinate - topRightImageXCoordinate) /
lengthPerPixelRatioX));
  bottomPixelOffset = int64(ceil((bottomLeftImageYCoordinate - bottomLeftMapYCoordinate) /
lengthPerPixelRatioY));
  topPixelOffset = int64(ceil((topRightMapYCoordinate - topRightImageYCoordinate) / lengthPerPixelRatioY));
% IV. Pre-Assigned Pixels to tiles. To save computation time, we want to slice the image into the parts that
%corresponds to a tile. To save space and time only store the "cuts" in x and y direction as separate arrays.
%Want to evaluate coherent structures, I suggest pre-processing image and not trying to modify this. 1. Need
%to cut after every tile covered (in parts) by image minus very last tile in x/y direction. if statements make sure
%don't get a bug where last cut is performed on last pixel of image. Make sure no negative values occur.
  amountOfImageCutsX =  topRightMapXIndex - bottomLeftMapXIndex - 1;
  amountOfImageCutsY =  topRightMapYIndex - bottomLeftMapYIndex - 1;
% 1.1 Special case where the edge of the las tile is the edge of the picture, check III 5.5.
if table2array(inputMap.listOfTables{inputTimestamp}{topRightMapXIndex, topRightMapYIndex}(1,1)) <
topRightImageXCoordinate
    amountOfImageCutsX = amountOfImageCutsX + 1;
end
if table2array(inputMap.listOfTables{inputTimestamp}{topRightMapXIndex, topRightMapYIndex}(1,2)) <
topRightImageYCoordinate
    amountOfImageCutsY = amountOfImageCutsY + 1;
end
% 2. Create blank arrays to store the indices of the cuts. double() typecasting is needed for the integer
%values. -1 at beginnings of a for loop, because cutting at last pixel of an image makes no sense. If image lies
%completely inside one tile (amountOfCuts == 0) need special treatment. Since reading out and converting
%tables is extremely time consuming for the PC (because MATLAB calls a lot of subroutines) it MUST be done
%outside for loop or only if the if statement is triggered and a new determination is strictly necessary.
if amountOfImageCutsX ~= 0
    imageCutsXIndexList = zeros(1, amountOfImageCutsX);
    for i = 1:1:(nrOfPixelsInputImageX - 1)
      if isBiggerOrEqualFloatingPointNumber((bottomLeftImageXCoordinate + (double(i) *
lengthPerPixelRatioX)), table2array(inputMap.listOfTables{inputTimestamp}{bottomLeftMapXIndex +
currentCutIndexX, 1}(1,1))) && (currentCutIndexX <= amountOfImageCutsX)
          imageCutsXIndexList(1, currentCutIndexX) = i; currentCutIndexX = currentCutIndexX + 1;
      end
    end
  else
```

Figure G.3: readInPicture.m Source-Code, page 3.

```
        amountOfImageCutsX = NaN; imageCutsXIndexList = NaN;
        warning("Image lies completly in one coulmn of the map.");
    end
    if amountOfImageCutsY ~= 0
        imageCutsYIndexList = zeros(1, amountOfImageCutsY);
        for i = 1:1:(nrOfPixelsInputImageY - 1)
            if isBiggerOrEqualFloatingPointNumber((bottomLeftImageYCoordinate + (double(i) *
lengthPerPixelRatioY)), table2array(inputMap.listOfTables{inputTimestamp}{1, bottomLeftMapYIndex +
currentCutIndexY}(1,2))) && (currentCutIndexY <= amountOfImageCutsY)
                imageCutsYIndexList(1, currentCutIndexY) = i;
                currentCutIndexY = currentCutIndexY + 1;
            end
        end
    else
        amountOfImageCutsY = NaN; imageCutsYIndexList = NaN;
        warning("Image lies completly in one row of the map.");
    end
% V. Evaluate Tiles for Thrombus 1. Create an Empty Evaluation Tile. Another typecasting to in64 needed
% because size() is not implementet well by MATHWORKS... Keep in mind the size() bug described above so [x,
%y] = size(image) will cause problems when reading in an RGB image.
evaluationTile = uint8(zeros(floor(inputMap.resolutionX / lengthPerPixelRatioX),floor(inputMap.resolutionY /
lengthPerPixelRatioY),3));
    [nrOfPixelsEvaluationTileX, nrOfPixelsEvaluationTileY, nrOfColorLayersEvaluationTile] = size(evaluationTile);
    nrOfPixelsEvaluationTileX = int64(nrOfPixelsEvaluationTileX);
    nrOfPixelsEvaluationTileY = int64(nrOfPixelsEvaluationTileY);
% 2. Reset the currentCutIndex counters since they are already used in section IV. The reset for y is technically
%not necessary because it needs to get reset anyways in for loop. Left it here to make logic of code clearer.
    currentCutIndexX = 1; currentCutIndexY = 1;
% 2. Evaluate the Image for every tile. Notice the -1 is necessary because each map index represents the top
%right rectangle on map. Right top map tile is not touching the image. Please refer to the map class.
for i = bottomLeftMapXIndex : 1 : topRightMapXIndex - 1
        currentCutIndexY = 1; %needs to be reset for every new loop over j.
        for j = bottomLeftMapYIndex : 1 : topRightMapYIndex - 1
% 2.1 Read in proper part of image from evaluation tile. Preparation of start and end indices for filling up
%evaluation tile. Default case start indices. Defined already above:
            startPixelIndexXEvaluationTile = 1;
            endPixelIndexXEvaluationTile = nrOfPixelsEvaluationTileX;
            startPixelIndexYEvaluationTile = 1; endPixelIndexYEvaluationTile = nrOfPixelsEvaluationTileY;
% 2.1.1 Stadard handling if cuts in x direction need to be made.
            if amountOfImageCutsX ~= 0
                startPixelIndexXImage = int64(imageCutsXIndexList(currentCutIndexX));
                if max(size(imageCutsXIndexList)) > 1
                    endPixelIndexXImage = int64(imageCutsXIndexList(currentCutIndexX + 1));
                else
                    endPixelIndexXImage = nrOfPixelsInputImageX;
                end
            else
                startPixelIndexXImage = 1;
                endPixelIndexXImage = nrOfPixelsInputImageX;
            end
            if amountOfImageCutsY ~= 0
                startPixelIndexYImage = int64(imageCutsYIndexList(currentCutIndexY));
                if max(size(imageCutsXIndexList)) > 1
```

Figure G.4: readInPicture.m Source-Code, page 4.

```
            endPixelIndexYImage = int64(imageCutsYIndexList(currentCutIndexY + 1));
          else
            endPixelIndexYImage = nrOfPixelsInputImageY;
          end
        else
          startPixelIndexYImage = 1; endPixelIndexYImage = nrOfPixelsInputImageY;
        end
% 2.2 Special handling of the most left/right/bottom/top tiles due to an eventual pixel offset when filling up
%the evaluation tile. startPixelIndex and endPixelIndex are given inside the relative coordinate system of each
%induvial tile. Overwrite default values from above.
        if i == bottomLeftMapXIndex && ~isequal(amountOfImageCutsX, 0) && ~isequal(amountOfImageCutsX,
1) && ~isequal(leftPixelOffset,0)
          startPixelIndexXEvaluationTile = leftPixelOffset;
          endPixelIndexXEvaluationTile = int64(imageCutsXIndexList(1));
          startPixelIndexXImage = int64(1);
          endPixelIndexXImage = int64(imageCutsXIndexList(1));
        elseif i == (topRightMapXIndex - 1) && ~isequal(amountOfImageCutsX, 0) &&
~isequal(amountOfImageCutsX, 1)
          startPixelIndexXEvaluationTile = imageCutsXIndexList(amountOfImageCutsX);
          endPixelIndexXEvaluationTile = nrOfPixelsEvaluationTileX - rightPixelOffset;
          startPixelIndexXImage = int64(imageCutsXIndexList(amountOfImageCutsX));
          endPixelIndexXImage = nrOfPixelsInputImageX;
        end
        if j == bottomLeftMapYIndex && ~(amountOfImageCutsY <= 1) && ~isequal(bottomPixelOffset,0)
          startPixelIndexYEvaluationTile = bottomPixelOffset;
          endPixelIndexYEvaluationTile = int64(imageCutsYIndexList(1));
          startPixelIndexYImage = int64(1); endPixelIndexYImage = imageCutsYIndexList(1);
        elseif j == topRightMapYIndex - 1 && ~isequal(amountOfImageCutsY, 0) &&
~isequal(amountOfImageCutsY, 1)
          startPixelIndexYEvaluationTile = int64(imageCutsYIndexList(amountOfImageCutsY));
          endPixelIndexYEvaluationTile = nrOfPixelsEvaluationTileY - topPixelOffset;
          startPixelIndexYImage = int64(imageCutsYIndexList(amountOfImageCutsY));
          endPixelIndexYImage = nrOfPixelsInputImageY;
        end
% 2.1.3 Actuall Reading in of the evaluation tile from image. if statement for error handling. linespace doesn't
%work with int64 because MATHWORKS didn't implement proper typecasting, so conversion to double is
%necessary.  linespace is set to do 100 steps by default so this needs to be specified as well.
        if startPixelIndexXImage == endPixelIndexXImage || startPixelIndexYImage == endPixelIndexYImage
          warning("Unnecissary Evaluation Tile created.");
        else
          evaluationTile2InputImagePixelIndexTranslationArrayX =
linspace(double(startPixelIndexXImage),double(endPixelIndexXImage), double(endPixelIndexXImage -
(startPixelIndexXImage - 1)));
          evaluationTile2InputImagePixelIndexTranslationArrayY =
linspace(double(startPixelIndexYImage),double(endPixelIndexYImage), double(endPixelIndexYImage -
(startPixelIndexYImage - 1)));
          for k = startPixelIndexXEvaluationTile : 1 : endPixelIndexXEvaluationTile
            for l = startPixelIndexYEvaluationTile : 1 : endPixelIndexYEvaluationTile
              evaluationTile(k,l,:) =
inputImage(evaluationTile2InputImagePixelIndexTranslationArrayX(k),evaluationTile2InputImagePixelIndexTra
nslationArrayY(l),:);
            end
          end
```

Figure G.5: readInPicture.m Source-Code, page 5.

```
% 2.1.4 Call the actual thrombus evaluation function evaluation tile. ONLY CHANGE THIS LINE OF CODE TO
%YOUR SPECIFIC IMAGE EVALUATION FUNCTION!!!
  inputMap.listOfTables{inputTimestamp}{i,j}(1,8) = evaluateThrombusInImage(evaluationTile, 255, 255, 255);
      end
      if currentCutIndexY + 1 < max(size(imageCutsYIndexList))
        currentCutIndexY = currentCutIndexY + 1;
      else
        warning("Unnecessary iteration in section V.");
      end
    end
    if currentCutIndexX + 1 < max(size(imageCutsXIndexList))
      currentCutIndexX = currentCutIndexX + 1;
    else
      warning("Unnecessary iteration in section V.");
    end
  end
% VI. Graphical Output for Debugging. It makes sense to provide a function that graphically represents which
% tiles were used in the program and where the picture actually lies. When readInPicture is normally used, this
%function should be turned off. 1. Create storage arrays
[sX, sY] = size(inputMap.listOfTables{inputTimestamp}); %all maps have the same size
A = zeros(sX,sY); %Storage for X Coordinates, since the "surface()" function won't work with tables or cells
B = zeros(sX,sY); %Storage for Y Coordinates
C = zeros(sX,sY); %Storage for information. % 2. Read in the data needed for plotting
  for i = 1:1:sX
    for j = 1:1:sY
      A(i,j) = table2array(inputMap.listOfTables{1}{i,j}(1,1));
      B(i,j) = table2array(inputMap.listOfTables{1}{i,j}(1,2));
      if i >= bottomLeftMapXIndex && i <= topRightMapXIndex && j >= bottomLeftMapYIndex && j <=
topRightMapYIndex
        C(i,j) = 1;
      end
    end
  end
% 3. Create rectangle to paint the image frame
x1 = [bottomLeftImageXCoordinate, bottomLeftImageXCoordinate, topRightImageXCoordinate,
topRightImageXCoordinate, bottomLeftImageXCoordinate];
y1 = [bottomLeftImageYCoordinate, topRightImageYCoordinate, topRightImageYCoordinate,
bottomLeftImageYCoordinate, bottomLeftImageYCoordinate];
z1 = [2, 2, 2, 2, 2];
% 4. Create the cuts in x direction
if ~(isnan(imageCutsXIndexList))
  x2 = zeros(2,max(size(imageCutsXIndexList))); y2 = zeros(2,max(size(imageCutsXIndexList)));
  z2 = zeros(2, max(size(imageCutsXIndexList)));
  for i = 1 : 1 : max(size(z2))
    z2(1,i) = 2; z2(2,i) = 2;
  end
  for i = 1 : 1 : max(size(x2))
    x2(1,i) = (bottomLeftImageXCoordinate + (double(imageCutsXIndexList(i)) * lengthPerPixelRatioX));
    x2(2,i) = (bottomLeftImageXCoordinate + (double(imageCutsXIndexList(i)) * lengthPerPixelRatioX));
  end
  for i = 1 : 1 : max(size(y2))
    y2(1,i) = bottomLeftImageYCoordinate; y2(2,i) = topRightImageYCoordinate;
  end
```

Figure G.6: readInPicture.m Source-Code, page 6.

```
end
% 5. Create the cuts in y direction
if ~(isnan(imageCutsYIndexList))
   x3 = zeros(2,max(size(imageCutsYIndexList))); y3 = zeros(2,max(size(imageCutsYIndexList)));
   z3 = zeros(2, max(size(imageCutsYIndexList)));
   for i = 1 : 1 : max(size(z3))
      z3(1,i) = 2; z3(2,i) = 2;
   end
   for i = 1 : 1 : max(size(y3))
      y3(1,i) = (bottomLeftImageYCoordinate + (double(imageCutsYIndexList(i)) * lengthPerPixelRatioY));
      y3(2,i) = (bottomLeftImageYCoordinate + (double(imageCutsYIndexList(i)) * lengthPerPixelRatioY));
   end
   for i = 1 : 1 : max(size(x3))
      x3(1,i) = bottomLeftImageXCoordinate; x3(2,i) = topRightImageXCoordinate;
   end
end
% 6. Create the actual plot (simple grid with x and y coordinates):
   surface(A,B,C);
   hold on
   plot3(x1, y1, z1, "Color", [.5, 0, 0], "LineWidth", 4);
   %{
   if ~(isnan(imageCutsXIndexList))
      for i = 1 : 1 : max(size(x2))
         x2neu = zeros(1,2); x2neu(1,1) = x2(1,i); x2neu(1,2) = x2(2,i); y2neu = zeros(1,2); y2neu(1,1) = y2(1,i);
         y2neu(1,2) = y2(2,i); z2neu = zeros(1, 2); z2neu(1,1) = z2(1,i); z2neu(1,2) = z2(2,i);
         plot3(x2neu, y2neu, z2neu, "Color", [0, 0.5, 0], "LineWidth", 4);
      end
   end
   if ~(isnan(imageCutsYIndexList))
      for i = 1 : 1 : max(size(x3))
         x3neu = zeros(1,2);
         x3neu(1,1) = x3(1,i); x3neu(1,2) = x3(2,i); y3neu = zeros(1,2); y3neu(1,1) = y3(1,i);
         y3neu(1,2) = y3(2,i); z3neu = zeros(1, 2); z3neu(1,1) = z3(1,i); z3neu(1,2) = z3(2,i);
         plot3(x3neu, y3neu, z3neu, "Color", [1, 0.65, 0], "LineWidth", 4);
      end
   end
   %}
   hold off
   title("Grid"); xlabel("x-Cooedinates [m]"); ylabel("y-Coordinates [m]");
%}
% VII. Return Output
outputMap = inputMap;
end
```

Figure G.7: readInPicture.m Source-Code, page 7.

Bibliography

[3DS22] 3DS: *Simulia Website*. https://www.3ds.com/products-services/simulia/, accessed on 07.03.2022

[AAa20] Ansari, Shadi ; Ashker, Rashid ; al. et: Measurement of the Flow Behaviour Index of Newtonian and Shear-Thinning Fluids via Analysis of the Flow Velocity Characteristics in a Mini-Channel. In: *SpringerNature Applied Sciences Journal* (2020), October. https://doi.org/10.1007/s42452-020-03561-w

[Ada18] Adams, N.: *Fluidmechanik 1-Einführung in die Dynamik der Fluide*. Summer Semester 2018. Technical University of Munich (TUM), 2018. – Script to the lecture „Fluid Mechanics 1“

[ADCH21] Ashcraft, M. ; Douglass, M. ; Chen, Y. ; Handa, H.: Combination strategies for antithrombotic biomaterials: An emerging trend towards hemocompatibility. In: *Biomater. Sci.* 9 (2021), 2413-2423. https://pubs.rsc.org/en/content/articlelanding/2021/BM/D0BM02154G

[Alt12] Altenbach, H.: *Kontinuumsmechanik (Continuum Mechanics)*. Springer, 2012. – ISBN 9783642241185

[AMAa11] Avila, Kerstin ; Moxey, David ; Avila, Marc ; al. et: Onset of Turbulence in Pipe Flow. (2011)

[Bai08] Baieth, H.E.: Physical Parameters of Blood as a Non-Newtonian Fluid. In: *International Journal of Biomedical Science* 4 (2008), December

[Blo62] Bloch, E. H.: A Quantitative Study of the Hemodynamics in the Living Microvascular System. In: *American Journal of Anatomy* 110 (1962), S. 125–153

[BLSa19] Brandes, Ralf ; Lang, Florian ; Schmidt, Ralf ; al. et: *Physiologie des Menschen (Human Physiology)*. 32. Springer, 2019. – ISBN 9783662564677

[Bra76] Bradestini, M.: A Digital 128-Channel Transcutaneous Blood Flow Meter. In: *Biomed. Tech.* 21(s1) (1976), S. 291–293

[Bri17] Brian, R.: Heparin Coatings for Improving Blood Compatibility of Medical Devices. In: *Adv. Drug Deliver. Rev.* 112 (2017), 12-13. https://pubmed.ncbi.nlm.nih.gov/28042080/

[Bru95] Bruun, H. H.: Hot-Wire Anemometry: Principles and Signal Analysis. In: *Meas. Sci. Technol.,* https://doi.org/10.1088/0957-0233/7/10/024 7(10) (1995), S. 532

[BS70] Buggliarello, G. ; Sevilla, J.: Velocity Distribution and Other Characteristics of Steady and Pulsatile Blood Flow in Fine Glass Tubes. In: *Biorheology* 7 (1970), S. 85–107

[BZ22] Brokate, Martin ; Zimmer, Johannes: *Vorlesungsskript zur Analysis 2 (Lecture Notes for Analysis 2).* Summer Semester 2022. Technical University of Munich (TUM), 2022

[CBS04] Chen, H. ; Brook, M. ; Sheardown, H.: Silicone elastomers for reduced protein adsorption. In: *Biomaterials* 25 (2004), 2273-2282. https://www.sciencedirect.com/science/article/abs/pii/S0142961203007579?via%3Dihub

[Che22] Chemyx: *Syringe Pump.* https://www.chemyx.com/syringe-pumps/fusion-4000-x/, accessed on 22.09.2022

[chi22] chipshop microfluidic: *Microfluidic Chip Shop.* https://www.microfluidic-chipshop.com/catalogue/, accessed on 15.09.2022

[CJ98] Cutnell, John ; Johnson, Cutnell: *Physics.* 4th Edition. 1998. – 308 S.

[Com22a] Company, COMSOL: *Navier-Stokes Equations: Conservation Laws.* https://www.comsol.com/multiphysics/navier-stokes-equations?parent=modeling-conservation-mass-energy-momentum-0402-432-302, accessed on 06.01.2022

[Com22b] Company, COMSOL: *COMSOL Website.* https://www.comsol.com/comsol-multiphysics, accessed on 07.03.2022

[Com22c] Company, COMSOL: *COMSOL Tutorial Valve.* https://www.comsol.de/video/simulating-closing-gate-valve-part-1, accessed on 08.09.2022

[Com22d] Company, COMSOL: *Navier-Stokes equations in COMSOL.* https://www.comsol.com/multiphysics/navier-stokes-equations, accessed on 22.04.2022

[COM22e] COMSOL: *COMSOL Blood VEssel Demo.* https://www.comsol.com/model/fluid-structure-interaction-in-a-network-of-blood-vessels-660, accessed on 17.10.2022

[Das11] Das, K.: A Mathematical Model on the Consistency Coefficient of the

Herschel-Bulkey Flow of Blood Through Narrow Vessel. In: *Arabian Journal for Science and Engineering* 36 (2011), April. – ISSN 13198025. – No. 3

[DCSa87] Dasse, K. ; Chipman, S. ; Sherman, C. ; al. et: Clinical experience with textured blood contacting surfaces in ventricular assist devices. In: *ASAIO J.* 33 (1987), July, 418-425. https://europepmc.org/article/MED/3314931

[Def22] DefocusTracker: *Defocus Tracker Github.* https://gitlab.com/defocustracking/defocustracker-matlab, accessed on 22.09.2022

[Die86] Dieter, G.: *Mechanical Metallurgy.* 1986

[Din67] Dintenfass, L.: An Inversion of the Fahraeus-Linqvist Phenomenon in Blood Flow Through Capillaries of Diminishing Radius. In: *Nature* 215 (1967), S. 1099–1100

[Edg22] Edge, Engineers: *Numerical Values for Air Viscosity.* https://www.engineersedge.com/physics/viscosity_of_air_dynamic_and_kinematic_14483.htm, accessed on 17.10.2022

[EL03] Ellingsen, J. ; Lyngstadaas, S.: Increasing Biocompatibility by Chemical Modification of Titanium Surfaces. In: *Bioimplant Interface* (2003), 323-340. https://onlinelibrary.wiley.com/doi/10.1002/jbm.b.31291

[Els17] Elsevier, B. V.: *Advanced Data Analysis and Modelling in Chemical Engineering.* 2017 http://dx.doi.org/10.1016/B978-0-444-59485-3.00009-6. – Chapter 9.1

[EM13] Engelmann, Bernd ; Massberg, Steffen: Thrombosis as an intravascular effector of innate immunity. In: *Nature Reviews Immunology, www.nature.com/reviews/immunol* 13 (2013), January

[FF05] Furie, Bruce ; Furie, Barbara: Thrombus Formation In Vivo. In: *The Journal of Clinical Investigation* 115 (2005), December. http://www.jci.org. – No. 12

[FL31] Fahraeus, R. ; Lindqvist, T.: The Viscosity of the Blood in Narrow Capillary Tubes. In: *American Journal of Physiology* 96 (1931), S. 562–568

[FMMM09] Fedel, M. ; Motta, A. ; Maniglo, D. ; Migliaresi, C.: Surface properties and blood compatibility of commercially available diamond-like carbon coatings for cardiovascular devices. In: *J. Biomed. Mater. Res.* 90 (2009), S. 338–349

[FMW18] Fischer, M. ; Maitz, M. ; Werner, C.: Coatings for Biomaterials to Im-

prove Hemocompatibility. In: *Hemocompatibility of Biomaterials for Clinical Applications* (2018), S. 163–190

[Fun93] Fung, Y. C.: *Biomechanics: Mechanical Properties of Living Tissues.* 2. Edition. 1993. – ISBN 9780387979472

[GIT22a] GITHUB: *SPlisHSPlasH.* https://github.com/InteractiveComputerGraphics/SPlisHSPlasH,accessedonthe07.03.22, accessed on 07.03.2022

[git22b] github: *curve_fit documentation3.* https://github.com/scipy/scipy/blob/v1.8.0/scipy/optimize/_minpack_py.py#L533-L839, accessed on 11.03.2022

[Gor04] Gorbert, M.: Biomaterial-Associated Thrombosis: Roles of Coagulation Factors, Complement, Platelets and Leukocytes. In: *Biomaterials* (2004)

[GW08] Greinacher, A. ; Warkentin, T.: The Direct Thrombin Inhibitor Hirudin. In: *Thromb. Haemost.* 99 (2008), S. 819–829

[Ibi22a] Ibidi: *Channel Slides.* https://ibidi.com/19-channel-slides, accessed on 15.09.2022

[Ibi22b] Ibidi: *u-Slides luer.* https://ibidi.com/channel-slides/50--slide-i-luer.html#/25-surface_modification-ibitreat_15_polymer_coverslip_tissue_culture_treated_sterilized/33-pcs_box-15_individually_packed/56-channel_height-02_mm, accessed on 15.09.2022

[Irg13] Irgens, F.: *Rheology and non-Newtonian fluids.* Springer, 2013 https://doi.org/10.1007/978-3-319-01053-3

[Jen04] Jensen, K. D.: Flow Measurements Techniques. In: *J. Braz. Soc. Mech. Sci. Eng., https://doi.org/10.1590/S1678-58782004000400006* 26(4) (2004), S. 400–419

[JFHW15] Jaffer, I. ; Fredenburgh, J. ; Hirsh, J. ; Weitz, J.: Medical device-induced thrombosis: What causes it and how can we prevent it? In: *J. Thromb. Haemost* 13 (2015), 72-81. https://pubmed.ncbi.nlm.nih.gov/26149053/

[JGWa16] Jain, Abhishek ; Graveline, Arnada ; Waterhouse, Anna ; al. et: A shear gradient-activated microfluidic device for automated monitoring of whole blood hemostasis and platelet function. In: *NATURE COMMUNICATIONS* (2016), January

[Kay22] KAYAKUAM: *SU8-Photo-Resist*. https://kayakuam.com/products/su-8-photoresists/, accessed on 19.09.2022

[KC02] KUNDU, P. K. ; COHEN, I. M.: *Fluid Mechanics*. 2nd Edition. Academic Press Cambridge, 2002. – ISBN 9780123813992

[Klo98] KLOPFENSTEIN, R. J.: Air Velocity and Flow Measurement Using a Pitot Tube. In: *ISA Trans., https://doi.org/10.1016/S0019-0578(98)00036-6* 37(4) (1998), S. 257–263

[KLO22] KLOE: *Dilase-250*. https://www.kloe-france.com/en/photolithography-equipment/direct-laser-writing/direct-laser-writer-dilase-250, accessed on 19.09.2022

[KWTG21] KUCHINKA, Jana ; WILLEMS, Christian ; TELYSHEV, Diminitry ; GROWTH, Thomas: Control of Blood Coagulation by Hemocompatible Material Surfaces – A Review. In: *Bioengineering* (2021), December. https://doi.org/10.3390/bioengineering8120215

[LCW03] LAVAUD, S. ; CANIVET, E. ; WUILLAI, A.: Optimal Anticoagulation Strategy in Haemodialysis with Heparin-Coated Polyacrylonitrile Membrane. In: *Nephrol. Dial. Transplant.* 18 (2003), 2097-2104. https://academic.oup.com/ndt/article/18/10/2097/1807496

[LOSC15] LOWE, S. ; O'BRIEN-SIMPSON, N. ; CONNAL, L.: Antibiofouling polymer interfaces: Poly (ethylene glycol) and other promising candidates. In: *Polym. Chem.* 6 (2015), 198-212. https://pubs.rsc.org/en/content/articlelanding/2015/PY/C4PY01356E

[MAT22] MATLAB: *curve_fit documentation4*. https://www.mathworks.com/help/optim/ug/lsqnonlin.html, accessed on 28.06.2022

[MGOVa19] MARCHIO, Patricia ; GUERRA-OJEDA, Sol ; VILA, Jose ; AL. et: Targeting Early Atherosclerosis: A Focus on Oxidative Stress and Inflammation. In: *Oxidative Medicine and Cellular Longevity, https://www.hindawi.com/journals/omcl/2019/8563845/* 2019 (2019), July

[MHa95] MAZUMDAR, H.P. ; HABISHVASI, U.N. ; AL. et: On the Consistency Coefficient of a Power-Law Flow of Blood Through the Narrow Vessel. In: *Engineering Transactions* 43 (1995), S. 373–382

[Mic22] MICROMOD: *Metal Flakes*. https://micromod.de/en/product-category/magnetic-particles/micromer-m-magnetic-particles/, accessed on 18.09.2022

[Mii22] MiiCraft: *Ultra Series 3D Printer*. https://miicraft.com/product-2/, accessed on 21.09.2022

[MLLG20] Montague, Samantha ; Lim, Yean ; Lee, Woei ; Gardiner, Elizabeth: Imaging Platelet Process and Function – Current Emerging Approaches for Imaging in vitro and in vivo. In: *Frontiers in Immunology, https://www.frontiersin.org/* 11 (2020), January

[MM11] Meloun, Milan ; Miliky, Jiri: *Statistical Data Analysis*. 2011. – ISBN 978–0–85709–109–3. – Chapter 8.5.2.1

[Mül19] Müller, Peter: *Analysis 1 – Analysis einer Variable*. Winter Semester 2019/2020. Ludwig Maximilian University of Munich (LMU), 2019. – Script to the lecture „Analysis 1“

[Mül20a] Müller, Peter: *Analysis 2 – Topologie und Differentialrechnung mehrerer Variablen*. Summer Semester 2020. Ludwig Maximilian University of Munich (LMU), 2020. – Script to the lecture „Analysis 2“

[Mül20b] Müller, Peter: *Analysis 3 – Maßtheorie und Integralrechnung mehrerer Variablen*. Winter Semester 2020/2021. Ludwig Maximilian University of Munich (LMU), 2020. – Script to the lecture „Analysis 3“

[Nan22] Nanonets: *Object Tracking Deepsort*. https://nanonets.com/blog/object-tracking-deepsort, accessed on 30.01.2022

[NN12] Nguyen, Quoc-Hung ; Nguyen, Ngoc-Diep: Incompressible Non-Newtonian Fluid Flows. In: *Continuum Mechanics – Progress in Fundamentals and Engineering Applications* (2012), March. http://www.intechopen.com/books/continuum-mechanics-progress-in-fundamentals-and-engineering-applications non-newtonian-fluid-flows. ISBN 9789535104476

[NS19] Nader, Elie ; Skinner, Sarah: Blood Rheology: Key Parameters, Impact on Blood Flow, Role in Sickle Cell Disease and Effects of Exercise. In: *Frontiers in Physiology* 10 (2019), October. www.frontiersin.org. – Art. No. 1329

[NWa09] Nesbitt, Warwick ; Weststein, Erik ; al. et: A shear gradient-dependent platelet aggregation mechanism drives thrombus formation. In: *Nature Medicine* 15 (2009), June. – No. 6

[PGa02] Pontrelli, G. ; Grigon, M. ; al. et: An Inversion of the Fahraeus-Linqvist Phenomenon in Blood Flow Through Capillaries of Diminishing Radius. In: *Biomedical Res.* (2002)

[PSZa19] Pernemalm, M. ; Sandberg, A. ; Zhu, Y. ; al. et: In-depth Human Plasma Proteome Analysis Captures Tissue Proteins and Transfer of Protein Variants Across the Placenta. In: *Elife* (2019). https://pubmed.ncbi.nlm.nih.gov/30958262/

[Que83] Quemada, D.: Blood Rheology and its Implication in Flow of Blood. In: *Arteries and Arterial Blood Flow* (1983), S. 1–127

[RL64] Rand, Peter ; Lacombe, Eleanor: Viscosity of normal human blood under normothermic and hypothermic conditions. In: *Journal of Applied Physiology* (1964). www.physiology.org/journal

[Rod83] Rodkiewicz, C. M.: Flow in Large Arteries. In: *Arteries and Arterial Blood Flow* (1983)

[RWK07] Raffel, M. ; Willert, C. E. ; Kompenhans, J.: *211 Fluid mechanical properties*. Bd. 37(4). 2007. – 15–16 S.

[SAMa17] Samson, Andre ; Alwis, Imala ; Maclean, Jessica ; al. et: Contractile forces in platelet aggregates under microfluidic shear gradients reflect platelet inhibition and bleeding risk. In: *American Society of Hematology, www.bloodjournal.org* (2017)

[sci22] scipy: *curve_fit documentation1*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html, accessed on 11.03.2022

[SHBa18] Sakurai, Yumiko ; Hardy, Elaissa ; Byungwook, Ahn ; al. et: A microengineered vascularized bleeding model that integrates the principal components of hemostasis. In: *NATURE COMMUNICATIONS* (2018)

[SJTa14] Stalker, Timothy ; John, Welsh ; Tomaiuolo, Maurizio ; al. et: A Systems approach to haemostasis: 3. Thrombus consolidation regulates intrathrombus solute transport and local thrombin activity. In: *BLOOD* 124 (2014), June

[SKM12] Sin, D. ; Kei, H. ; Miao, X.: Surface Coatings for Ventricular Assist Devices. In: *Coatings for Biomedical Applications* (2012), S. 264–283

[Sta22a] Stackoverflow: *Split AVI into Frames*. https://stackoverflow.com/questions/3917601/ffmpeg-split-avi-into-frames-with-known-frame-rate, accessed on 03.02.2022

[sta22b] stackoverflow: *curve_fit documentation2*.

https://stackoverflow.com/questions/28372597/python-curve-fit-with-multiple-independent-variables, accessed on 11.03.2022

[STWa13] Stalker, Timothy ; Traxler, Elizabeth ; Wu, Jie ; al. et: Hierarchical organisation in the haemostatic response and its relationship to the platelet-signalling network. In: *BLOOD* 121 (2013), March

[Sue78] Suetra, S. P.: Red Cells Motion and Deformation in the Microcirculation. In: *Cardiovascular and Pulmonary Dynamics* (1978), S. 221–242

[SVH98] Scheerder, L. de ; Verbeken, E. ; Humbeeck, J. van: Metallic Surface Modification. In: *Seminars in Interventional Cardiology* (1998), S. 139–144

[TFTa19] Ting, Lucas ; Feghhi, Shirin ; Taparia, Nikita ; al. et: Contractile forces in platelet aggregates under microfluidic shear gradients reflect platelet inhibition and bleeding risk. In: *Nature Communications, https://doi.org/10.1038/s41467-019-09150-9* (2019), October

[THJ04] Thurston, G. B. ; Henderson, N. M. ; Jeng, M.: Viscoelastic Properties of Blood on Analoges. Advances in Hemodynamics and Hemorheology. (2004)

[TMPFa18] Tomaiuolo, Maurizio ; Matzko, Chelsea ; Poventud-Fuentes, Izmarie ; al. et: Interrelationships between structure and function during the hemostatic response to injury. In: *PNAS* (2018), December. https://www.pnas.org/doi/full/10.1073/pnas.1813642116

[Tra22] Tracker, DeFocus: *Fundamentals of Defocus Tracker*. https://defocustracking.com/fundamentals/, accessed on 30.01.2022

[TYF07] Tropea, E. C. ; Yarin, A. L. ; Foss, J. F.: *Springer Handbook of Experimental Fluid Mechanics*. Springer, 2007 https://doi.org/10.1007/978-3-540-30299-5

[USG22] USGS: *Water Density*. https://www.usgs.gov/special-topics/water-science-school/science/water-density, accessed on 16.10.2022

[WA17] Wooton, David ; Alevriadou, Rita: The Shear Stress of Busting Blood Clots. In: *The New England Journal of Medicine* (2017), October

[WSS07] Wiklund, J. ; Shahram, M. ; Stading, M.: Methodology for In-Line Rheology by Ultrasound Doppler Velocity Profiling and Pressure Difference Techniques. In: *Chem. Eng. Sci., https://doi.org/10.1016/j.ces.2007.05.007* 62 (2007), S. 4277–4293

[Yee15] Yee, J. A.: *Hypercalcemia*. 2015. – ISBN 9780128012383

[YJUZ06] Yang, S. ; Ji, B. ; Undar, A. ; Zahn, J.: Microfluidic Devices for Continuous Blood Plasma Separation and Analysis During Pediatric Cardiopulmonary Bypass Procedures. In: *ASAIO J.*, *https://doi.org/10.1097/01.mat.0000249015.76446.40* 52(6) (2006), S. 698–704

[Zal10] Zalpour, Christoph: *Anatomie Physiologie (Anatomy Physiology)*. 3rd Edition. Urban Fischer Verlag, 2010. – ISBN 9783437453021