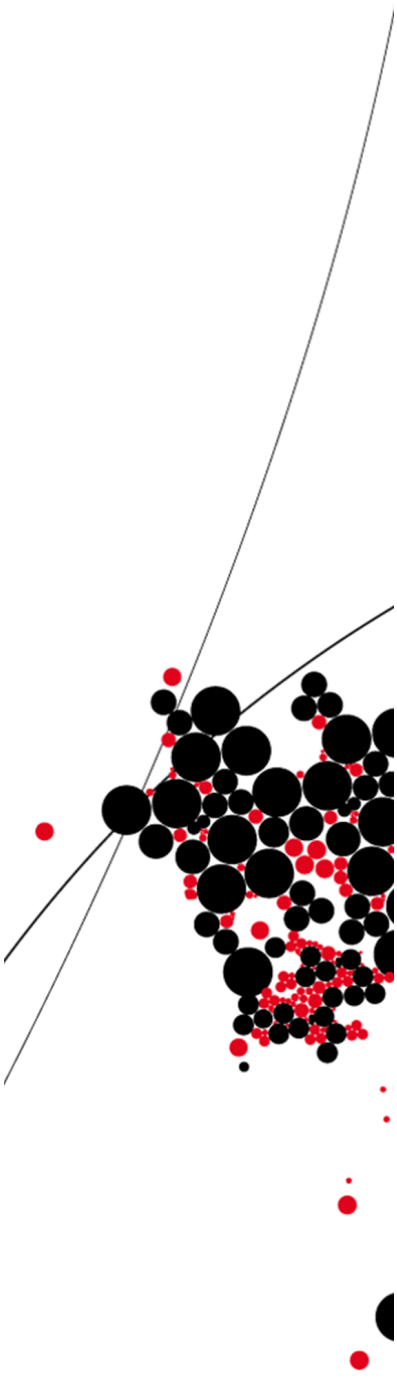
UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**



CFD modeling of a gas exchange membrane in OpenFOAM

Marissa H.C. Frijns

M.Sc. Thesis
February 2017

Assessment committee:
Prof. dr. Bernard J. Geurts
Paolo Cifani, MSc
Dr. Mike A. Bochev

Multiscale Modelling and Simulation
Research Group
Department of Applied Mathematics

Abstract

The Wyss Institute at Harvard is developing organs-on-a-chip which are able to mimic organ functions and could even become person-specific. The largest benefit for this kind of chips is the model which is based on humans instead of animals. The predictions made are realistic and drugs can be tested for toxicity. The goal of this thesis is to build a basis for the computation of gas exchange between two fluids, separated by a membrane, to support the development of such a chip. In addition, the effect of membrane degradation on the gas exchange is implemented and reviewed to provide a starting point in expanding the model properties.

The open-source CFD software OpenFOAM is used for mathematical simulations of the convective and diffusive flows of a generic gas species through a membrane. The model is verified with both spatial and temporal grid refinement studies and their convergence.

Membrane degradation is implemented in the model to apply both time and concentration dependent diffusion coefficient decline. An equation was created and implemented to simulate the degradation and partial healing of a membrane due to a concentration peak. Due to the adjustable parameters in this equation, the profile of the concentration dependent diffusion coefficient can be manipulated to obtain the desired outcome. This provides proof of the applicability of OpenFOAM as a CFD software to model gas exchange and contributes a basis to apply concentration dependent membranes in a model.

Acknowledgement

Writing this thesis has been an intense period for me, not only in the scientific sense, but also on a personal level. It has had a big impact on me and I would like to reflect on the people who have supported and helped me so much throughout this time.

At first, I would like to thank my thesis supervisor Bernard Geurts of the Multiscale Modeling and Simulation group at the University of Twente. He consistently allowed this thesis to be my own work but steered me in the right direction whenever he thought I needed it.

I would also like to acknowledge Paolo Cifani of the Multiscale Modeling and Simulation group at the University of Twente as my second adviser of this thesis, and I am grateful for his very valuable comments on this thesis. The door to Paolo's office was always open whenever I ran into a trouble spot or had a question about my research. We have spent several hours looking at the code in OpenFOAM and without his input this thesis would not have reached its full potential.

At last, I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. Furthermore, I want to thank Robin for his love, help and support. I am indebted to him for the discussions which helped me to get a good overview of the problems at hand and how to solve them. Finally, there are my friends. We were not only able to support each other by deliberating over our problems and findings, but also happily by talking about things other than just our research work. This accomplishment would not have been possible without them. Thank you.

Table of Contents

Chapter 1

Introduction

Typically the process of developing new drugs is a sequential and lengthy process. Pharmaceutical companies have several moments where the status of the drug is reviewed and decisions are made on whether to continue with its development. After the new drug is synthesized, it will be screened for toxicity in vitro [1] and subsequently tested on animals. Animal lives are lost in this process, and more importantly, animal testing often fails to predict human responses to the drug because traditional animal models do not accurately mimic human pathophysiology [2, 3]. One of the downsides of animal testing is that when the results are unsuccessful in animals, promising drugs are abandoned while it may work for humans. For example, the widely used drugs aspirin and penicillin would not have been available if the current animal testing proceedings were in place then [2]. Besides that, the failure rate of drugs which are successful in preclinical tests and do not proceed to market (including animal testing) is estimated by the FDA to be close to 96 percent [2]. If the drug is still considered a promising candidate for further development after the animal testing, the pharmaceutical company may begin human clinical trials. These clinical trials take years to complete and testing a single drug can cost millions of dollars [4]. The costs of a clinical trial keep increasing every year, which has significant implications for the public health, since the willingness of pharmaceutical companies for clinical trials declines, decreasing the output of new drugs. The reduction in willingness results in fewer applicable drugs and treatments for patients [5]. Also, human clinical studies have revealed considerable variability in responses to the drugs between patients. The results are dependent on the participant and their medical history. The whole concept of drug testing nowadays misses its mark, and for this reason, a new human-based alternative is required to generate safe and reliable predictions of drug efficiency.

The Wyss Institute at Harvard University is developing specialized lab-on-a-chip systems which can mimic some of the complicated mechanical and biochemical behavior of human organs. These chips present much more realistic models of the human organs than flat layers of cells grown in petri dishes [6]. Some examples of these micro-engineered models are based on the liver, heart, lung, intestine, kidney, brain and bone [3]. The team at the Wyss Institute is even exploring the potential of combining different organ systems. An example is the combination of a “breathing” lung-on-a-chip with a “beating” heart-on-a-chip. By linking several chips together, researchers can study the effect of one drug in several organs. A drug that gives beneficial results in treating lung disease, for instance, might have toxic results on a different organ in the human body. Combining chips could help find anomalies earlier in the process, reducing the research and money invested in a, for example, toxic drug, which will never go to market. The combination of organs-on-a-chip may lead to a personalized linked human-on-a-chip someday, on which drugs can be tested. With this perspective, the necessity for animal testing would be reduced, and clinical trials would be safer due to the better predictions of the drugs.

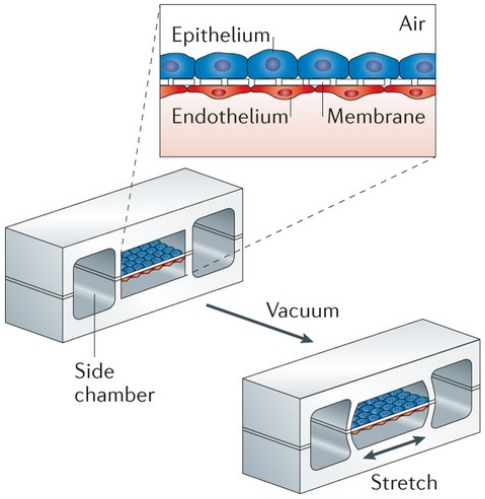


Figure 1.1: A schematic representation of the lung-on-a-chip developed at the Wyss Institute [3].

The work in this thesis is motivated by a lung-on-a-chip invented by the Wyss Institute team. This microchip consists of a three-dimensional system based on a living, breathing human lung. This detailed microchip has two layers of living cell tissue, containing cells of the lung's alveoli and the blood vessels that surround them, opposite of each other on a porous, flexible polymer [7]. Air is delivered to the upper part of the chip where the lung cells are, and a fluid which mimics blood is delivered to the lower part of the chip (Figure 1.1). Breathing is mimicked by generating a vacuum which is applied to the chambers adjacent to the cell channels which results in stretching and relaxing of the membrane. This movement of the membrane is important since it is experimentally shown that there is a considerable increase in the diffusion of nanoparticles through the membrane when the chip is mechanically stimulated [3, 7]. From experiments, it became apparent that a lung-on-a-chip gave a better representation of real life lungs than a static conventional petri dish model.

The lung-on-a-chip at the Wyss Institute is well developed. However, the created lung membrane has a flat structure. The membrane of an alveoli has a more complex shape, more like small lobes which are better approximated by a curved geometry. Additional research on a lung-on-a-chip is being performed at the University of Twente in collaboration with the University of Leiden and Rotterdam to extend the properties of the membrane in the lung-on-a-chip [8]. In this collaboration, research is being performed to achieve the same functions but with a curved geometry which is much closer to reality. Changing the geometry sounds easy. However, the flow of the fluids will change for example and may become turbulent instead of laminar.

1.1 Problem Statement

The development towards a complete organ-on-a-chip continues by expanding the functionalities and structure to mimic a realistic organ. To support the development towards an improved chip, the goal was set to create a mathematical model to understand and predict the working processes that take place inside a lung-on-a-chip in more detail. This model ranges from the fluid-mechanical transport to the passage of species through the membrane structure separating the external from the internal lungs, respectively air and blood. A mathematical model can be very helpful to explain the behavior of a complex system such as the gas exchange in the alveoli of the human respiratory system. Simulations can serve as a substitute to traditional experiments and has the benefit that experiments which can not be done traditionally, impractical or too expensive, can be performed with the use of simulations. When a simulation model is developed, it is easy to try new ideas to increase our understanding of

the system at hand or to try and simulate different experimental conditions. The computational fluid dynamics software used in this thesis is the open-source program OpenFOAM. This software was chosen as knowledge about this program was already present in the research group.

During the development of the mathematical model in OpenFOAM, it became apparent that due to the complexity of the system and required computational time, the magnitude of the project was too large to generate a full model within the given time for this thesis. Therefore, the goal was adjusted during the research to build a basis for the computation of gas exchange between two fluids separated by a membrane and the effect of membrane degradation on the gas exchange.

1.2 Structure of this thesis

This thesis is divided into six chapters and is organized as follows. Chapter 1 gives an outline of the background aspects behind the motivation for this thesis and defines the problem statement. In chapter 2 some literature and background material is covered which is used in subsequent chapters. The respiratory system is briefly described which forms the basis of the simulation, followed by an introduction of the CFD software OpenFOAM. Chapter 3 describes the development of a model of the gas exchange between the alveoli and the capillaries through a membrane. The chapter provides the verification of the solver for the velocity and the concentration equations via grid refinements and looking at the convergence rates for used the numerical schemes. Chapter 4 investigates the behavior of the diffusion through the membrane when it becomes a function of time and concentration of species. With the parameter study indicates how the function for the diffusion coefficient can be manipulated. Chapter 5 and 6 summarizes the main features of the resulting model and presents the overall conclusions of the research study. Several suggestions are made for future research work since only the surface is scratched.

Chapter 2

Theoretical Framework

2.1 Physiology

2.1.1 Respiratory system

Oxygen is a key fuel for the operation of the human body. Oxygen has to be taken from the air around us and separated from other gasses, bacteria, and other impurities. The organ system which is responsible for this process is the respiratory system, shown in Figure 2.1. The respiratory system is defined as a group of organs involved in drawing oxygen into and removing carbon dioxide out of the body (and cells). Oxygen is required to enable living cells to retrieve energy stored in carbohydrates, fats, and proteins [9]. Carbon dioxide is the waste product of the metabolism of the cell.

Respiration starts when the diaphragm contracts, the lungs expand and air enters through the oral cavity or nasal passage and moves through the nasal cavity to the lungs due to compression of the diaphragm [10]. The nose and the other upper respiratory passages rapidly warm and moisturize the relatively cold and dry air and filter large particulates from the air by cilia and mucous. Air flow continues into the bronchi and moves down into an alveolus where oxygen diffuses into the blood via the capillary beds. Gas exchange between the alveolus and the blood flow is selective since not everything which enters the lungs should end up in our blood flow. Oxygen is allowed to diffuse into the blood, where it attaches to the hemoglobin (Hb) in red blood cells. Carbon dioxide diffuses out of the blood and back into the air in the alveolus where it is expelled out of the body by the depression of the diaphragm [10].

In the respiratory system, the alveolus is the primary area of gas exchange. Alveoli are hemispheric structures with diameters that range from 75 to 300 μm [10]. The $\sim$300 million alveoli have a combined surface area of 50 to 100 m^2 and an aggregated maximal volume of 5 to 6 L [10]. In humans, the lung surface is so large and so efficient that O_2 and CO_2 transport across the alveolar wall is $\sim$3 fold faster than necessary when the person is resting at sea level [10]. This indicates that the membrane is not the limiting factor in oxygen uptake rate, but rather the supply of oxygen.

2.1.2 Gas exchange in the lungs

The exchange of both O_2 and CO_2 across the alveolar blood-gas barrier occurs by diffusion. The concentration gradients by which O_2 and CO_2 diffuse through the membrane are created by the ventilation and circulation of air. The membrane is covered with a film of water on the alveolar side since diffusion of oxygen through the membrane occurs in the liquid phase. The membrane will separate a volume filled with moist air from a volume of blood plasma at 37°C [10].

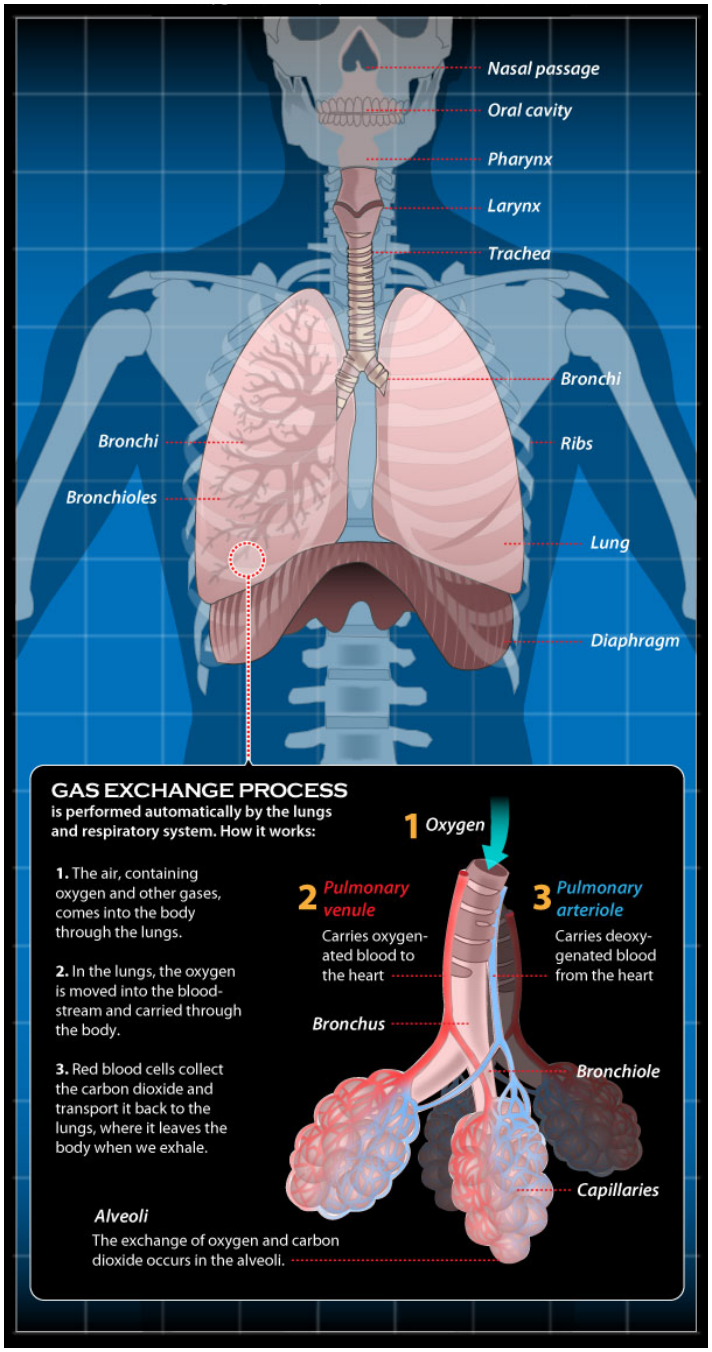


Figure 2.1: An overview of the relevant parts of the respiratory system and a closer look at the gas exchange process in the alveoli [11].

Fick's law of diffusion states that the rate of transfer of gas through a sheet of tissue is proportional to the tissue area (A) and the difference in gas partial pressure (P_i) between the two sides and inversely proportional to the thickness (d) [10]:

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \frac{A}{d}\mathscr{D}(P_A - P_C), \tag{2.1.1}$$

where $\frac{dV}{dt}$ is the amount of gas that diffuses through the tissue, $\mathscr{D}$ the diffusion coefficient, P_A stands for the partial pressure of the alveolar side of the membrane, P_C for the partial pressure of the capillary side. Dalton's law [12] expresses the fact that the total pressure of a mixture of gasses is equal to the sum of the partial pressures of the individual gasses in the mixture. Gasses dissolve, diffuse, and react according to their partial pressures, and not according to their concentrations in gas mixtures or liquids.

An important parameter in Fick's law is the diffusion coefficient of gas, which describes the rate of diffusion of a specific gas through a particular medium. Two properties of the gas contribute to the diffusion coefficient, namely the molecular weight (MW) and solubility of the gas in water (s). Graham's law [12] states that diffusion is inversely proportional to the square root of the molecular weight, and according to Henry's law [12], the concentration of the gas dissolved in water is proportional to the partial pressure in the gas phase. There is also a proportionality constant (k) that describes the interaction of the gas with the membrane. Fick's law of diffusion then becomes

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \underbrace{\frac{Aks}{d\sqrt{MW}}}_{\mathscr{D}_L}(P_A - P_C), \tag{2.1.2}$$

where $\mathscr{D}_L$ is the diffusing capacity for the lung [mL/(min · mm Hg)]. This equation describes the diffusion of gas between two compartments whose properties are uniform both spatially and temporally. However, a closer examination of the parameters in real life reveals that this approximation is more complex than described above. The parameters may vary in the following ways [10]:

- Lung expansion causes the surface area to increase and the thickness of the membrane to decrease.
- The thickness of the membrane is not equal everywhere.
- The alveolar partial pressure changes over time due to the respiration.
- The alveolar partial pressure varies among alveoli due to variations in the resistance of the conducting airways, and the compliance of the alveoli.
- The diffusion of oxygen is maximal at the beginning of the pulmonary capillary and gradually falls to zero along the capillary.

These complications which arise from Fick's law with O_2 diffusion also apply to CO_2 diffusion.

Apparently, a single set of fixed values for $\mathscr{D}_L$, P_A and P_C cannot be inserted into the whole system, but it can for a single piece of alveolar wall at a single time during the respiratory cycle. Each cell has an individual diffusion coefficient, and the overall diffusion can be compiled by taking the sum for all pieces and times.

2.1.3 Capillaries

Blood vessels consist of a system of tubes which transfer blood to and from the heart to all the parts of the body. The diameter of these blood vessels varies enormously, from a diameter of ∼25 mm in the aorta to only ∼9 μm in the capillaries [13]. The velocity of blood through the vessels changes with diameter, and it is naturally the fastest in the aorta. In Figure 2.2 the velocity of the blood flow

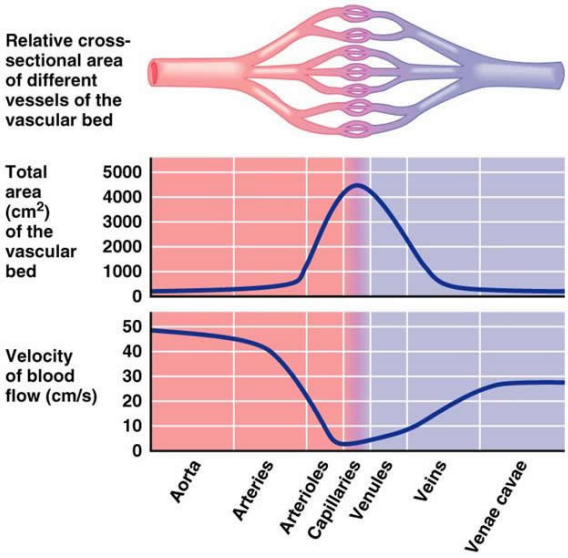


Figure 2.2: The cross-sectional area of different vessels of the vascular bed compared to the velocity of the blood flow [13].

for each type of blood vessel can be found. The Reynolds number of blood flow in the body varies from 1 in small arterioles (a small branch of an artery leading into capillaries) to approximately 4000 in the aorta [14].

The capillaries are necessary for the gas exchange in the lungs. These tiny blood vessels allow for the exchange of substances between blood and the fluids outside of them. The most important factor for gas exchange is the high density of capillaries around an alveolus which gives a large area for diffusion.

2.1.4 Membrane

For oxygen to end up in the bloodstream, it has to diffuse through the membrane into the capillaries. In this work, the diffusion is assumed to be over a homogeneous one layer membrane. However, the actual membrane is a three-ply structure comprising an alveolar epithelial cell, a capillary endothelial cell, and the intervening interstitial space containing extracellular matrix [10]. Without going into the biological background of these structures, each structure contributes a resistance to the oxygen diffusion rate.

As O_2 diffuses from the alveolar air to the Hb inside an erythrocyte (red blood cell), it must cross 12 discrete mini-barriers (Figure 2.3) and contributes to a so-called membrane diffusing capacity, $\mathscr{D}_M$. The 12 diffusive steps are analogous to 12 resistors in series. Electrical current (I) in $I = \frac{1}{R} \cdot \Delta V$ corresponds to the flow of gas; the reciprocal of resistance corresponds to the diffusing capacity; and the voltage difference corresponds to the pressure difference. The membrane diffusing capacity $\mathscr{D}_M$ can be determined via

$$\frac{1}{\mathscr{D}_M} = \frac{1}{\mathscr{D}_1} + \frac{1}{\mathscr{D}_2} + \frac{1}{\mathscr{D}_3} \ldots + \frac{1}{\mathscr{D}_{11}} + \frac{1}{\mathscr{D}_{12}}. \tag{2.1.3}$$

where $\mathscr{D}_i$ (for $i = 1..12$) corresponds to the diffusion coefficients of all twelve diffusion steps in Figure 2.3. These parameters vary with location in the lung and position in the respiratory cycle, as discussed in section 2.1.2.

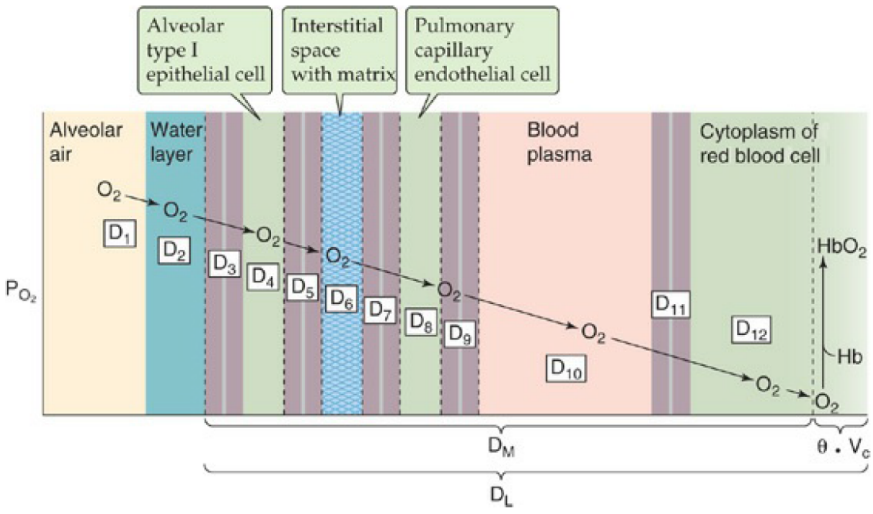


Figure 2.3: Transport of O_2 from the alveoli to the capillary where it binds to Hb [10].

The final step for most of the O_2 entering the bloodstream is binding to the Hb. This occurs at the finite rate $(\theta \cdot V_C)$, where θ is a rate constant that describes how many mL of gas binds to the Hb in 1 mL of blood each minute. V_C is the volume of the blood in the pulmonary arteries. The diffusing capacity of the membrane in an alveoli can then be expressed by

$$\frac{1}{\mathscr{D}_L} = \frac{1}{\mathscr{D}_M} + \frac{1}{\theta \cdot V_C}. \tag{2.1.4}$$

2.2 Toolbox

The research presented in this thesis is concerned with the development of a simulation model of the gas exchange between the alveoli and the capillaries designed in computational fluid dynamics software. Computational fluid dynamics (CFD) [15, 16] is the analysis of systems involving fluid flow, heat transfer and so on by utilizing computer-based simulations. In CFD the three fundamental governing equations of the fluid dynamics are used: the continuity, momentum, and energy equations. These equations are discretized and solved iteratively till the defined tolerance for the residual is small enough. To date, no closed form solution to these governing equations is known. Therefore computers are used to perform the complex calculations required to simulate the interaction between liquids and/or gasses with surfaces defined by boundary conditions. Matrices with numbers are the result of a CFD simulation and not an analytic solution. After the simulation is performed, some post processing can be done to analyze and visualize the solution.

2.2.1 OpenFOAM

OpenFOAM is a free and open source 3D computational fluid dynamics software package. It can be used for a wide variety of features to solve anything from complex flows involving chemical reactions to turbulent flows, heat transfer, acoustics and electromagnetism [17]. The programming language in which this toolbox is written is C++ although it has its own syntax. For example, the equation

$$\frac{\partial u}{\partial t} + \nabla \cdot \phi u - \nabla \cdot (\nu \nabla u) \quad = \quad -\nabla P \tag{2.2.1}$$

is written in OpenFOAM as

```
fvVectorMatrix UEqn
(
  fvm::ddt(U)
+ fvm::div(phi, U)
- fvm::laplacian(nu, U)
);

solve(UEqn == -fvc::grad(p));
```

where `fvc::` will calculate the derivative using the current values and `fvm::` will discretize the term into a matrix equation.

Due to the wide range of applications of OpenFOAM, the appropriate available solver has to be chosen for a particular application. Standard solvers can be found in the OpenFOAM User Guide [18] and are divided into several categories, e.g. incompressible flow, multiphase flow, combustion, and particle-tracking flows. The solvers in OpenFOAM use numerical schemes for derivatives in equations that are calculated during a simulation. To be able to use these numerical schemes, each computational domain has to be divided into grid cells and each center of the grid cell is used for calculations.

Five of the numerical schemes used in this thesis are explained below. For the spatial discretization, two schemes are considered: Gauss Upwind and Central Difference [19]. The upwind scheme can be written in a forward and a backward upwind scheme, depending on the sign of the characteristic speed. The accuracy of the upwind scheme is first order, as can be seen in the equations below. The central difference scheme is given by the third equation and is second order accurate in space.

$$\begin{aligned}
\frac{\partial u}{\partial x} &= \frac{u(x_{i+1},t_j)-u(x_i,t_j)}{\Delta x}+\mathcal{O}(h)\\
\frac{\partial u}{\partial x} &= \frac{u(x_i,t_j)-u(x_{i-1},t_j)}{\Delta x}+\mathcal{O}(h)\\
\frac{\partial u}{\partial x} &= \frac{u(x_{i+1},t_j)-u(x_{i-1},t_j)}{2\Delta x}+\mathcal{O}(h^2)
\end{aligned}$$

The temporal grid refinement makes use of three types of numerical schemes, namely Euler, Crank-Nicolson and Backward Difference (BDF2). The first numerical scheme is a first order scheme while the other two are second order schemes. The schemes are defined as follows:

- Implicit Euler [19]:

$$\frac{u(x_i,t_{j+1})+u(x_i,t_j)}{\Delta t} = F(u(x_i,t_{j+1})) \tag{2.2.2}$$

- Crank-Nicolson [19]:

$$\frac{u(x_i,t_{j+1})+u(x_i,t_j)}{\Delta t} = \frac{1}{2}\left[F(u(x_i,t_{j+1}))+F(u(x_i,t_j))\right] \tag{2.2.3}$$

- BDF2 [20]:

$$\frac{u(x_i,t_{j+2})-\frac{4}{3}u(x_i,t_{j+1})+\frac{1}{3}u(x_i,t_j)}{\Delta t} = \frac{2}{3}F(u(x_{i+2},t_{j+2})) \tag{2.2.4}$$

where F is a function dependent on the ODE system $u'(t)=F(u(t))$.

In this work, an extension of an existing solver of OpenFOAM was made (named `chipFoam`) since creating a new system would take a tremendous amount of additional time. The current solver is called `conjugateHeatFoam` which uses two regions. The original solver simulates the heat conduction from a fluid to a solid. To be able to exchange information between the two regions in the system a

unique boundary condition named `regionCouple` is used. This boundary condition is applied where the interaction between two domains occurs. The same principal is used in the `chipFoam` solver, however, a minimum of three layers is required (fluid-solid-fluid). To reduce the complexity of fluid flows, only two dimensions are considered. Because OpenFOAM is a 3D CFD modeling program, the length/depth for the third dimension was chosen to be 1/10 of the height of the channel.

An important component of the solver is the way it handles the coupling of the separate regions. The diffusion coefficient at the interface is computed by an harmonic mean. From Patankar's book [21] it is known that when there could be abrupt changes in the diffusion coefficient (or any other parameter) the harmonic interpolation scheme behaves much better than the linear one. The expression for the linear and harmonic interpolation between the membrane and a channel are respectively presented below:

$$\mathscr{D} = \frac{1}{2}\left(\mathscr{D}_M + \mathscr{D}_{Ch}\right) \tag{2.2.5}$$

$$\mathscr{D} = \frac{2\,\mathscr{D}_M\mathscr{D}_{Ch}}{\mathscr{D}_M + \mathscr{D}_{Ch}} = \frac{2}{\dfrac{1}{\mathscr{D}_{Ch}} + \dfrac{1}{\mathscr{D}_M}} \tag{2.2.6}$$

To see the behavior of these two interpolation functions, two limiting cases are studied:

- Let $\mathscr{D}_{Ch} \to 0$. Then, from (2.2.5) it is clear that $\mathscr{D} \to \frac{1}{2}\mathscr{D}_M$ and (2.2.6) gives $\mathscr{D} \to 0$. This last one implies that the flux at the face of the membrane becomes zero, where the membrane is fully stuck, and nothing goes in or out anymore. This would be different if the linear average were used where a leak from the membrane would always be present.

- Let $\mathscr{D}_M \gg \mathscr{D}_{Ch}$. This indicates that interface diffusion coefficient is not dependent on $\mathscr{D}_M$ for the harmonic interpolation, however, for the linear interpolation, $\mathscr{D}_M$ would have maintained an effect on $\mathscr{D}_{Ch}$.

With the use of the harmonic interpolation, abrupt changes can be handled without requiring an excessively fine grid in the vicinity of the change.

To get a better understanding of all components which are needed for a simulation to run, a small tutorial can be found in Appendix A. In this section, the process of setup, simulation and post-processing is explained.

2.2.2 PISO algorithm

There are two widely used algorithms in CFD computing, namely the SIMPLE and PISO algorithm. SIMPLE stands for Semi-Implicit Method for Pressure Linked Equations and is a guess-and-correct procedure of the calculation of the pressure and velocity equations on a staggered grid [16]. The other algorithm is PISO and stands for Pressure Implicit with Splitting of Operators. PISO can be seen as an extension of SIMPLE with an extra corrector step to enhance its performance per iteration step [16]. For the solver `chipFoam` the algorithm PISO is used and the computing steps are explained in Appendix A.2.

Chapter 3

Numerical Verification

An important part of computational mathematics is the verification of a model. To verify whether the solver `chipFoam` is working properly, a comparison is made between the output of the model and an analytical solution. First, the equation for the velocity is considered, and secondly, the profile is analyzed. Subsequently, the analysis of the system is extended by the addition of a species transport equation.

Both spatial and temporal mesh refinements and their convergence are studied. Examination of the spatial convergence of a simulation involves performing the simulation on two or more successively finer grids. As the grid is refined (grid cells become smaller and the number of cells in the computational domain increases) and the time step is refined (decreased) the spatial and temporal discretization errors, respectively, should decrease algebraically, creating a straight line in a loglog plot of error versus resolution. The slope of the line is dependent on the discretization scheme used.

3.1 Velocity

A system is chosen to investigate the behavior of spatial and temporal grid refinements on the velocity profile. To approach the shape and interactions in a lung-on-a-chip, the system is made from two co-current flowing channels separated by a membrane (Figure 3.1). Both channels are modeled with laminar flow considering the Reynolds number in the capillaries. At the inlet of each channel a block profile is considered, and at the walls, a no-slip condition is imposed. This block profile will converge to a Poiseuille profile and will be fully developed beyond the entrance length of the channel.

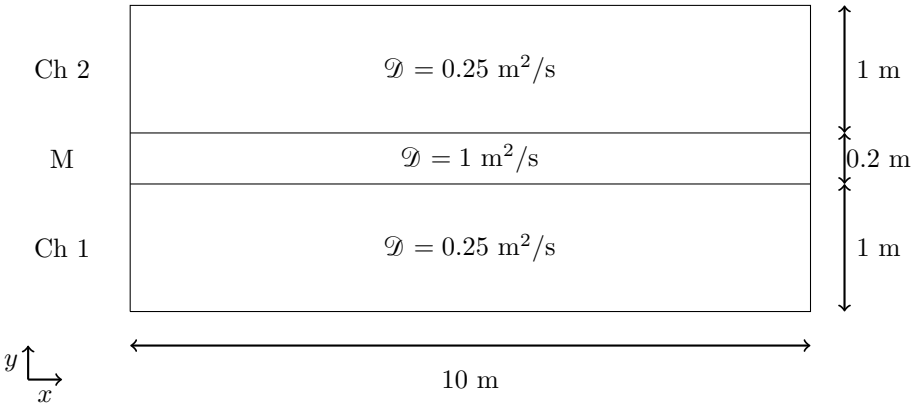


Figure 3.1: The computational domain of two channels (Ch) separated by a membrane (M).

3.1.1 Analytic solution

The flow in the channel is assumed to be two-dimensional, incompressible and to have constant properties. With these assumptions the continuity and momentum equations [22] for steady flow are

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3.1.1}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{3.1.2}$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{3.1.3}$$

where u and v are velocity components in respectively the x- and y-direction, ν is the kinematic viscosity, ρ is the density and P stands for the gas partial pressure.

The flow in both channels is assumed to be a fully developed laminar flow, constrained by flat parallel walls on the top and bottom of the channel. Each fluid particle moves at a constant velocity along a streamline and the velocity profile remains unchanged in the flow direction. No velocity component in the y-direction is possible due to impermeability of the walls which indicates that $v = 0$ across the system. The continuity equation simplifies to

$$\frac{\partial u}{\partial x} = 0, \tag{3.1.4}$$

where $u = u(y)$. The momentum equations simplify to:

$$0 = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu\frac{\partial^2 u}{\partial y^2} \tag{3.1.5}$$

$$0 = -\frac{1}{\rho}\frac{\partial P}{\partial y} \tag{3.1.6}$$

Equation (3.1.4) is used to simplify (3.1.2). The only equation left after the simplification is (3.1.5), which is a linear second-order ODE and this is easily integrated twice to yield

$$u(y) = \frac{y^2}{2\mu}\frac{\mathrm{d}P}{\mathrm{d}x} + C_1 y + C_2, \tag{3.1.7}$$

where C_1 and C_2 are integration constants, and μ is the dynamic viscosity. In order to determine the integration constants, the no-slip boundary condition is applied at both walls of the channel.

$$y = 0, y = h \qquad u = 0$$

The final solution then becomes

$$u(y) = \frac{1}{2\mu}\frac{\mathrm{d}P}{\mathrm{d}x}\left[y^2 - hy\right], \tag{3.1.8}$$

where h is the height of the channel. The velocity distribution in the channel is parabolic and symmetrical about the center. This exact solution is also called the Poiseuille flow.

This parabolic profile can also be explained physically. When fluid enters a channel at a uniform velocity across the channel, friction will have an influence on the velocity profile along the channel. The fluid particles of the layer closest to the surface of the channel will come to a complete stop due to the no-slip condition imposed at the surface. Due to momentum transfer and the viscosity of the fluid, the adjacent layers will also be slowed down. However, due to the conservation of mass to hold, the velocity of the middle layer should increase resulting in a velocity gradient across the cross section of the channel [23].

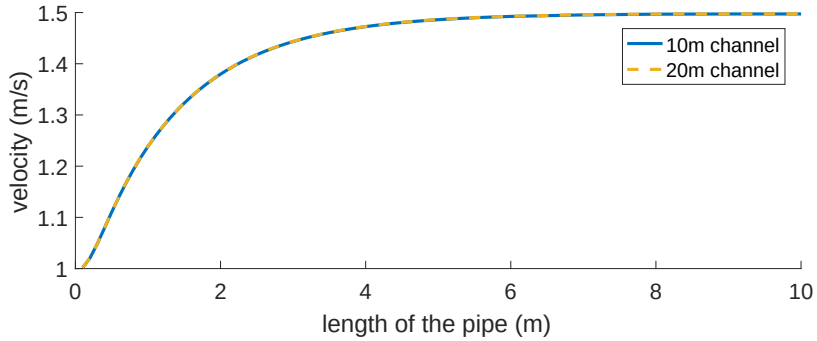


Figure 3.2: Determination of the entrance length to obtain a fully developed Poiseuille flow from a block profile.

3.1.2 Entrance length

When a block profile is imposed at the beginning of the channel, it will take some space for the Poiseuille profile to be reached. The length it takes to reach this profile and become fully developed is called the entrance length. The entrance length is important because only after this the comparison can be made to the Poiseuille flow.

If the total channel length is long compared to the entrance length the effect of the entrance length can be neglected and the total channel length can be used in calculations. If the total channel length is relatively short in comparison with the entrance length the developing stage becomes much more significant and has to be analyzed separately. For the calculation of the entrance length, the Reynolds number and the diameter of the channel are required. The Reynolds number (Re) is defined as follows

$$Re = \frac{u_{max} d}{\nu}, \tag{3.1.9}$$

where u_{max} is the maximum velocity of the parabola and d is the characteristic length of the channel. These values are chosen such that the Reynolds number becomes 150 and this results in the following entrance length [22, 23]:

$$L_{laminar} = 0.06 \cdot Re \cdot d = 9 \text{ m}. \tag{3.1.10}$$

In Figure 3.2 it is shown that the flow is fully developed after 9m, since the velocity profile does not change anymore with downstream distance. The entrance length is also checked for a longer channel (20m, with same grid spacing as before) to confirm that this length stays the same; with the max values respectively $[1.4972; 1.4975]$. For most calculations it is important to know when the flow is fully developed; otherwise, the computed data is not reliable.

To compare the velocity profile computed in OpenFOAM to the analytical solution, the flow should be fully developed. When a block profile is imposed at the inlet of this system, almost the whole length of the channel is required to obtain a developed laminar flow and only in the last meter, the flow can be considered as fully developed. For this reason, a parabolic profile is imposed directly at the inlet. It takes less than 0.1s to settle to the final numerical solution which differs a little bit from the analytic solution due to the discretization error (Figure 3.3). All further calculations should be done after 0.1s to make sure the flow is fully developed.

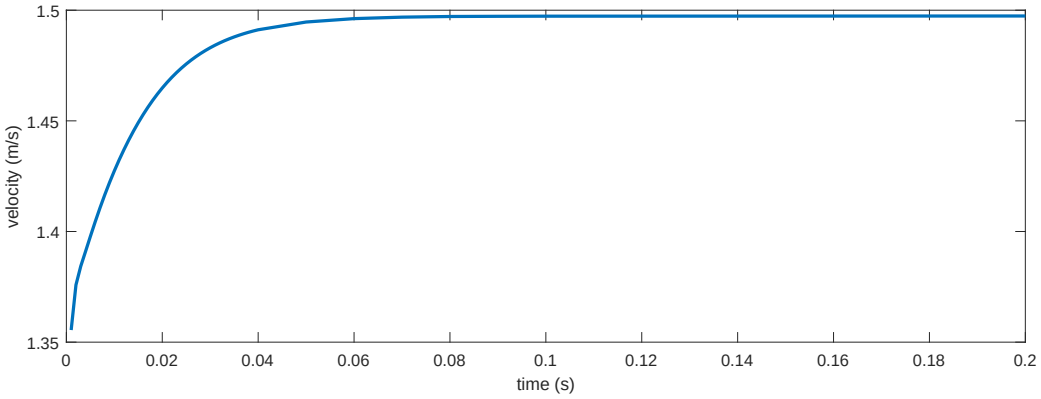


Figure 3.3: Required time to obtain a fully developed profile by imposing a Poiseuille flow at the inlet. The maximum velocity requested is 1.5 m/s.

3.1.3 CFL condition

While solving partial differential equations numerically by the method of finite differences the Courant-Friedrichs-Lewy (CFL) condition is required for the convergence. The CFL condition [19] is defined as:

$$\frac{u\Delta t}{\Delta x} \leq C_{\max}, \tag{3.1.11}$$

where u is the local flow velocity, Δt the simulation time step and Δx the grid size. The value of $C_{\max}$ changes with the method used to solve the discretized equation, especially depending on whether the method is explicit or implicit. If an explicit solver is used then typically $C_{\max} = 1$. Implicit solvers are usually less sensitive to numerical instability and so larger values of $C_{\max}$ are tolerated. Generally, $C_{\max} < 1.0$ is a good rule of thumb. With the CFL condition, the ratio between $\Delta t/\Delta x$ can be determined for several Reynolds numbers. The higher the Reynolds number, the smaller the cells have to be to resolve the boundary layer. The smaller the cells and the higher the speed, the smaller the time step that allows for a stable solution. This is a desirable property, otherwise, the approximation errors may be magnified and the solution explodes.

3.1.4 Conservation of mass

The principal of conservation of mass is used to check if the velocity is the same at different points in the channel when there is no diffusion possible through the membrane. The conservation of mass implies that mass can neither be created nor destroyed, although it may be rearranged in space. Due to the conservation of mass, in the steady state, the flow passing through $x = \frac{L}{4}$ should be equal to the flow passing through, e.g., $x = \frac{L}{2}$ and $x = \frac{3L}{4}$. The conservation of mass can be determined by

$$Q_N \quad = \quad \int_0^h u(\hat{x}, y)\, dy, \tag{3.1.12}$$

where h is the height and $\hat{x}$ is one of the above specified locations in the channel. A grid of 800×160 is chosen, which is the finest grid that was simulated in OpenFOAM during this research. The finest grid is selected to approximate the best solution. As an example, two cases are shown in Figure 3.4 in which a blob of species is covered by a different grid. It can be seen that a finer grid covers the curves in the boundaries more accurately and that a calculated average concentration would be approximated better.

All three places in the channel have a value close to 1, which indicates that the conservation law holds, since the velocity block profile is imposed at the boundary with a value of one.

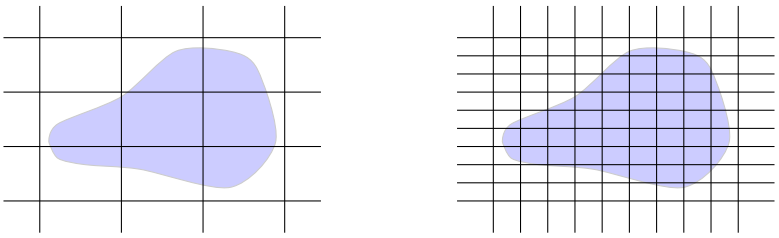


Figure 3.4: A blob of species divided by a coarse (left) and a fine (right) grid.

3.1.5 Spatial convergence

To visualize the effect of both mesh quality and size on the results, initial simulations were carried out with three mesh resolutions, 3×7, 5×15 and 10×30 and with Gauss upwind as the spatial discretization and Euler as the temporal discretization. Figure 3.5a shows the velocity profile in a channel for these three different mesh resolutions. The measured values can be found in the grid cell centers, and therefore the velocity profile can only be determined up till a certain distance from the wall equal to the distance from the center of the grid to the grid wall. The boundary condition on the wall is no-slip and there is no velocity in the membrane since exclusively diffusion is present. The coarsest mesh is not an accurate approximation of the analytic solution. However, when making the mesh finer, the approximation becomes better. To check whether or not the approximations converges with the right theoretical order to the analytic solution, the rate of convergence for discretization methods should be determined. This rate of convergence is defined as:

$$|f_n - f| \sim n^{-p}. \tag{3.1.13}$$

A sequence f_n is said to converge to f with order p [19]. The error $\|f\|$ is measured in the so-called L_1 norm defined as:

$$\frac{1}{N} \sum_i |u_i^N - u_i^{\text{ref}}|, \tag{3.1.14}$$

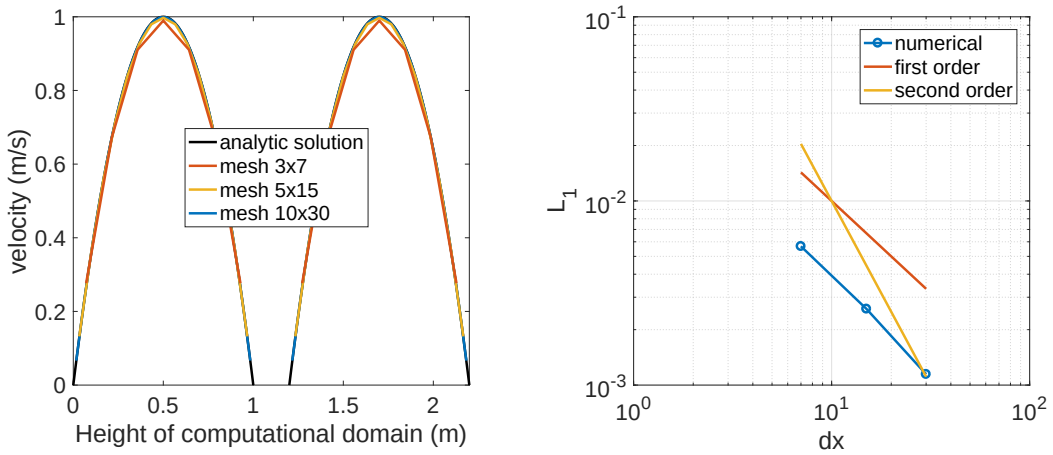
where u_i^N is the velocity field computed on a mesh size with N grid points and u_i^{ref} is the analytical reference solution or a sufficiently finer grid. It is also possible to determine the L_1 norm at one position in the channel and is then defined as:

$$\sum_i |u_i^N - u_i^{\text{ref}}|. \tag{3.1.15}$$

The convergence rate of the solution on three grid refinements is plotted in Figure 3.5b. This indicates first order convergence since the numerical data follows the first order line. This is also in agreement with the theoretical order of the Gauss upwind scheme that was adopted for the simulations.

The convergence of the solver `chipFoam` is checked with the discretization schemes Gauss upwind and central difference. Both methods are checked at three positions in the channel ($x = 2.5$m, $x = 5$m, and $x = 7.5$m). These three points in the channel are chosen as such that they are not too close to the beginning or the end of the channel, otherwise, the inlet and outlet conditions could have an influence on the flow. To check the correctness of the implementation of the convective term a uniform velocity field is imposed at the inlet of the channel. This leads to a region near the inlet where $u \cdot \nabla u$ is not zero. In Figure 3.6 only one position in the channel is plotted as an example. Gauss upwind shows first order convergence and central difference shows second order convergence in the graph, which is expected from the theory.

Another way to establish, quantitatively, the order of accuracy of the numerical scheme is to estimate the truncation error for successive grid refinements [24]. When the exact solution u is not



(a) (b)

Figure 3.5: (a) Result of three grid refinements converging towards the analytical solution. Two parabola representing the first and second channel, separated by the membrane (no flow). (b) Convergence plot of the respective grid refinements at $x = 0.5$ and $t = 2s$. The variable dx indicates the size of the grid cell and L_1 is the norm defined in (3.1.15) to measure the error.

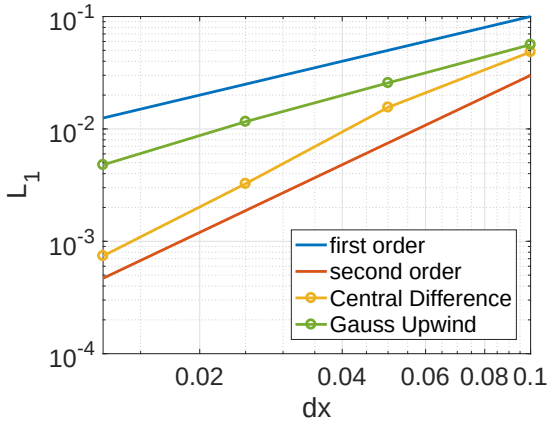


Figure 3.6: Convergence measured in the middle of a channel for two difference methods, respectively Gauss upwind and central difference. L_1 is the norm defined in (3.1.14).

	Central Difference			Gauss Upwind		
	x = 1/4	x = 1/2	x = 3/4	x = 1/4	x = 1/2	x = 3/4
$u_{\Delta x}$	5.0218	4.0046	3.8615	2.0994	1.9758	1.9778
$u_{\Delta x/2}$	3.8429	3.9423	3.9720	1.8222	1.9107	1.9761
$u_{\Delta x/4}$	3.8443	3.9919	4.1429	1.3815	1.7447	2.0946

Table 3.1: The convergence rate for two difference schemes at three locations in the channel. The values in the first column indicate the term $u_{\Delta x}$ in the expression of (3.1.17).

known there are two main approaches. The first one is to compute a numerical reference solution with a very small Δx and subtract this from the solution for a grid of size Δx and from one of size $\frac{\Delta x}{2}$:

$$\frac{(u_{\Delta x})^n - (u_{\Delta x_{\text{ref}}})^n}{\left(u_{\Delta x/2}\right)^n - (u_{\Delta x_{\text{ref}}})^n} \approx \frac{\mathcal{O}\left((\Delta x)^n\right)}{\mathcal{O}\left((\frac{\Delta x}{2})^n\right)} \approx 2^n, \tag{3.1.16}$$

where $u_{\Delta x}$ represents the solution at a specified grid and n indicates the order of convergence. The reference grid ($u_{\Delta x_{\text{ref}}}$) should be sufficiently finer than the used grid in order to have an accurate estimate of the convergence rate. This can be quite an expensive numerical strategy. The second approach is to look at ratios of differences between the numerical solutions computed for different Δx. Most commonly the solutions are compared where Δx is successively halved:

$$\frac{(u_{\Delta x})^n - \left(u_{\Delta x/2}\right)^n}{\left(u_{\Delta x/2}\right)^n - \left(u_{\Delta x/4}\right)^n} \approx \frac{\mathcal{O}\left(1 - 2^{-n}\right)}{\mathcal{O}\left(2^{-n} - 2^{-2n}\right)} \approx 2^n. \tag{3.1.17}$$

When the difference scheme is first order ($n = 1$), the ratio should be close to 2 and for second order the ratio should be close to 4. The convergence ratios for the two difference schemes mentioned above can be found in Table 3.1. The ratios found in Table 3.1 are similar to the convergence rates found in Figure 3.6.

3.1.6 Temporal convergence

The temporal grid refinement makes use of three types of `ddtSchemes`, namely Euler, Crank-Nicolson and Backward Difference. For the temporal grid refinement, the same system set-up is used as before. The spatial mesh size is chosen to be 800×160 and the spatial discretization scheme is central due to higher accuracy. This very fine mesh is required to measure the convergence of the temporal error independently. For the temporal grid refinement study, the flow should still change in time when the data for the convergence rate is taken, since the derivative in time is considered. To check for which time steps this is true, six locations in the channel ($L_x = \left[\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\right]$ and $L_y = \left[\frac{1}{4}, \frac{1}{2}\right]$) are investigated in terms of an still unsteady velocity profile. In Figure 3.7 the value of u is shown as a function of time for the given six locations. It can be seen that at 2 seconds the lines have not settled yet, indicating that the velocity is not established up until then. In OpenFOAM, the simulations for the temporal grid refinement are run up to 2 seconds and with this data, the convergence rate can be calculated using (3.1.15). In Figure 3.8 the convergence rates are plotted for two locations in the channel. There is a clear difference between the convergence rate in the first part of the channel (entrance length) and when the velocity profile is fully developed. It is expected that this difference is due to the change in the flow since a block profile was imposed at the inlet which gradually converges to a Poiseuille flow. When the flow is spatially developed the convergence rate corresponds to the theoretical order.

It is also possible to integrate over the difference of the velocity obtained with a coarse Δt and a reference value Δt_{ref}.

$$\frac{1}{\tilde{t}} \int_0^{\tilde{t}} \left| u_N^E(x^*, t; \Delta t) - u_N^E(x^*, t; \Delta t_{\text{ref}}) \right| \mathrm{d}t. \tag{3.1.18}$$

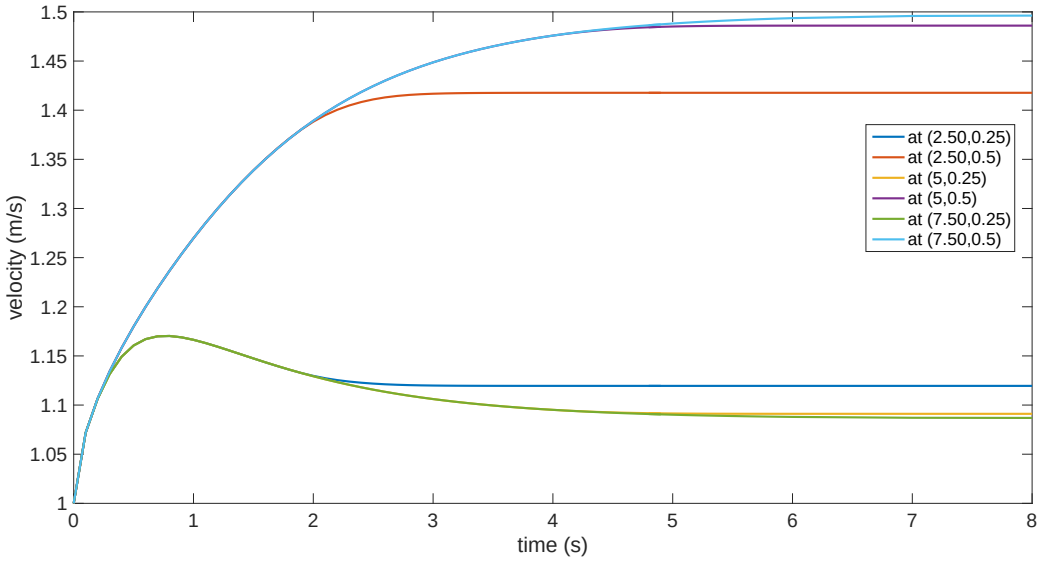


Figure 3.7: Convergence time to obtain a steady state velocity profile at six location in the channel.

Temporal	Time (s)
Euler	7208,43
BDF2	7036,25
Crank-Nicolson	8006,03

Table 3.2: Computation times for all temporal schemes where 800×160 was the grid used.

The E represents the method which is used, $\tilde{t}$ is the time until which the solution is monitored and x^* is the position where the value is measured. In Figure 3.9 it can be seen that Euler halves the value for each mesh refinement in the fully developed part of the channel, while for BDF2 and Crank-Nicolson the value becomes four times smaller. From the figure, it is clear that the methods correspond to the theoretically expected convergence rate and is similar to Figure 3.8.

As previously stated, the spatial and temporal grid refinement convergence to their theoretical order. From this, it is concluded that the solver for the Navier-Stokes equations is implemented correctly in OpenFOAM.

To determine which difference schemes will be used for further calculations of the velocity derivatives, the accuracy of the schemes is compared. For the spatial part it is quite clear, central difference is best applicable since the accuracy is much higher. The only case where central difference could fail is when $Pe > 2$, but this is not verified in this simulation. For the temporal part, the two second order schemes are considered. Since the error for both second-order schemes is roughly the same, the most appropriate scheme is therefore controlled by the computation time. In this case, there is a clear difference in computation time (Table 3.2), therefore the BDF2 scheme is chosen.

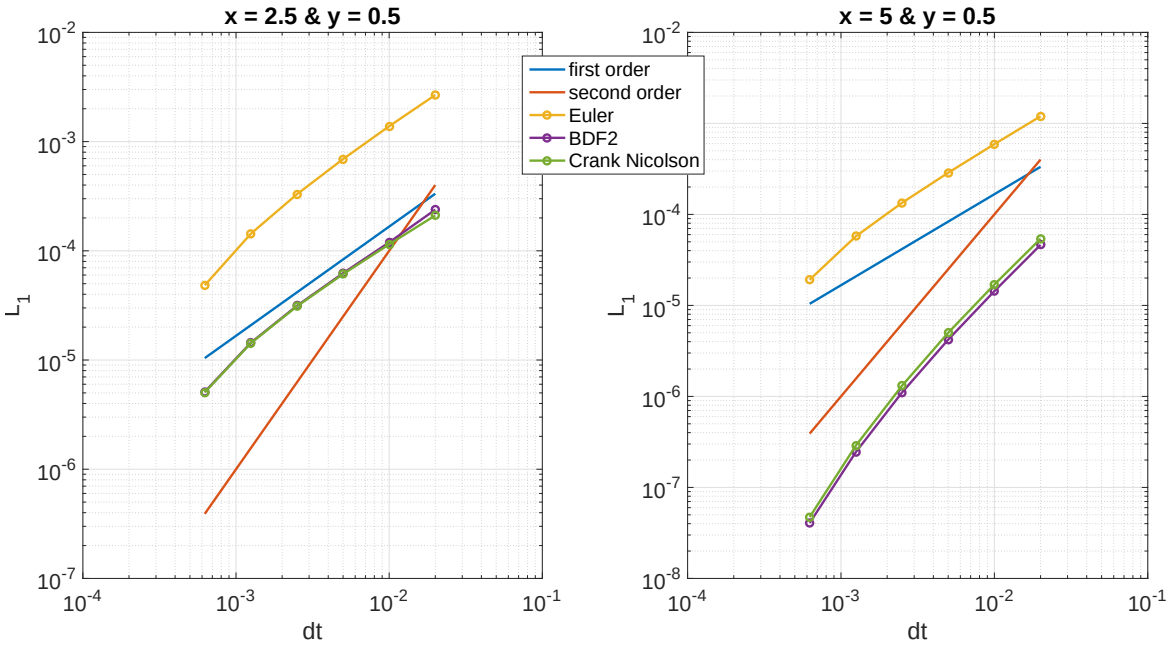


Figure 3.8: Three time-dependent differential schemes compared to first and second order convergence.

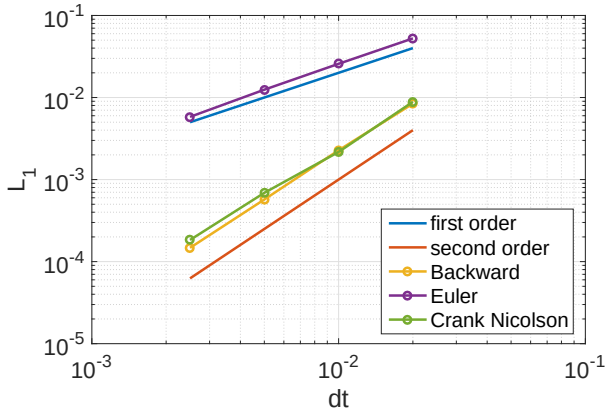


Figure 3.9: Evaluation of equation (3.1.18) for three different time integration schemes.

3.2 Concentration

In this section, the behavior of a generic species dissolved in the fluid medium is evaluated in OpenFOAM. Similar conditions and dimensions are chosen as in the previous section. However, the system model used (Figure 3.1) was the first design and was mainly focused on verification. The dimensions of the system are not in proportion with a lung-on-a-chip application. More importantly, the value of the viscosity is unrealistic and resembles a thick syrup instead of a more blood like viscosity. For this reason, a new system is chosen and to sustain comparability, the amount of change has to be kept small; the Reynolds and Péclet numbers are fixed and the dimension of the channel, the viscosity and diffusivity of the equations are adjusted. The input profile for the concentration is a Gaussian profile, which will result in a blob of species. The blob will be beneficial to track where the particles end up.

3.2.1 Non-dimensionalized equations

The problem is rewritten to reduce the parameter space in dimensionless form. The advantage of this is that only the dimensionless numbers need to be varied instead of each parameter itself.

The incompressible Navier-Stokes momentum equation is written as

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -\frac{1}{\rho}\nabla P + \nu \nabla^2 \mathbf{u}. \tag{3.2.1}$$

The equation above can be non-dimensionalized through selection of appropriate scales which are defined as follows:

$$\begin{aligned} \mathbf{x}^* &= \frac{\mathbf{x}}{L} \\ \mathbf{u}^* &= \frac{\mathbf{u}}{V} \\ t^* &= \frac{t}{L/V} \\ P^* &= \frac{PL}{\mu V} \end{aligned}$$

where L is a characteristic length, V is a characteristic velocity and P is a characteristic pressure. Substituting the scales in (3.2.1) and rearranging the constants, the non-dimensionalized Navier-Stokes momentum equation obtained is:

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla)\,\mathbf{u}^* = -\nabla P^* + \underbrace{\frac{\nu}{VL}}_{1/Re} \nabla^2 \mathbf{u}^*. \tag{3.2.2}$$

The Reynolds number is defined as the ratio of inertial forces to viscous forces and consequently quantifies the relative importance of each type of force for given flow conditions [22].

The species transport equation is written as

$$\frac{\partial c}{\partial t} + (\mathbf{u} \cdot \nabla)\, c = \mathscr{D} \nabla^2 c. \tag{3.2.3}$$

The only extra scale for this equation is defined as follows:

$$c^* = \frac{c}{c_0}$$

where c_0 is the initial concentration. Substituting the scales in (3.2.4), the non-dimensionalized species equation obtained is:

$$\frac{\partial c^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla)\, c^* = \underbrace{\frac{\mathscr{D}}{VL}}_{1/Pe} \nabla^2 c^*. \tag{3.2.4}$$

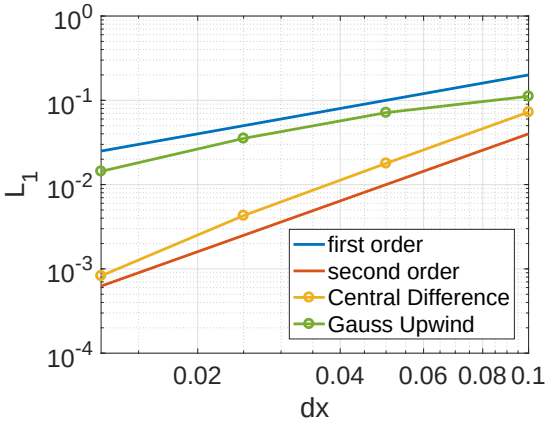


Figure 3.10: Convergence measured in the middle of the channel for two types of difference methods.

The Péclet number (Pe) is the ratio of the rate of advection of a physical quantity by the flow to the rate of diffusion of the same quantity driven by an appropriate gradient [22]. It is well known that the cell Péclet number must be smaller than 2.0 to maintain numerical stability when using the central difference scheme for the convective term in a linear convective-diffusive problem [19].

3.2.2 Convergence

With the equations dimensionless, the dimensions of the system can easily be changed. The system should still behave the same as before and the dimensionless numbers play a convenient role in doing this. For the case in section 3.1 the dimensionless Péclet and Reynolds numbers are determined as follows:

$$Pe = \frac{d \cdot u}{\mathscr{D}} = \frac{1 \cdot 1}{0.25} = 4$$

$$Re = \frac{d \cdot u}{\nu} = \frac{1 \cdot 1}{1} = 1$$

In these dimensionless numbers the diameter d is used instead of the length L of the channel, since a channel flow is assumed. These dimensionless numbers can also be determined for the grid cells themselves:

$$Pe_{\text{cell}} = \frac{\sqrt{(10/50)^2 + (1/10)^2} \cdot 1}{0.25} \approx 0.89$$

$$Re_{\text{cell}} = \frac{\sqrt{(10/50)^2 + (1/10)^2} \cdot 1}{1} \approx 0.22$$

For a convection-diffusion problem, both Pe and Re of the cell should be lower than 2 to ensure stability when using the central difference scheme. The coarsest grid is used, which results into the highest Péclet and Reynolds numbers.

The dimensionless numbers have to stay in each system to keep it comparable. Indicating that ν and $\mathscr{D}$ should decrease in value when Δx is decreased. With these new values, grid refinement studies are conducted again for both the spatial and temporal part. The system defined in section 3.2 shows the same convergence rate as the system defined in section 3.1 as can be seen from the convergence rates in Figure 3.10 and 3.11.

A scalar quantity (species) can now be added and verified. The input profile for the concentration is a Gaussian profile given by $A \exp\left(kt^2 + qx^2\right)$, which will result in a blob of species. A blob of species is chosen over a uniform distribution of species to track where the species end up. The constant values k and q are chosen as such that the blob does not move too fast or too slow through the channel.

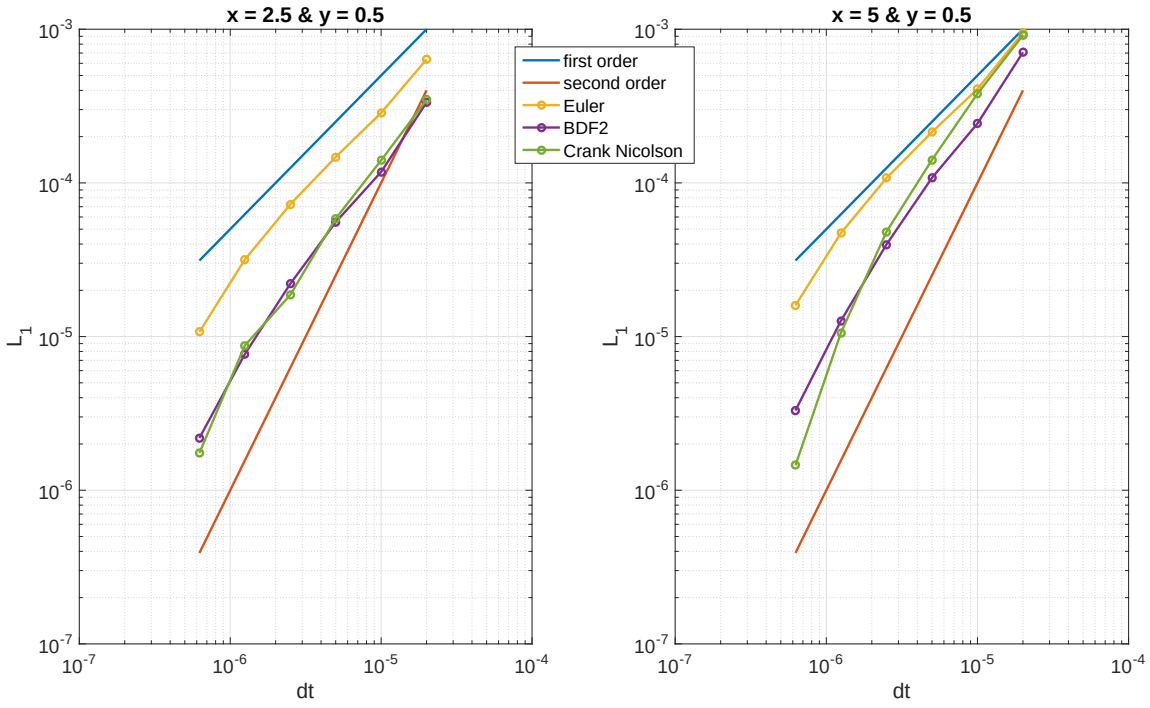


Figure 3.11: Three time-dependent differential schemes compared to first and second order convergence. Taken at time equal to 0.002.

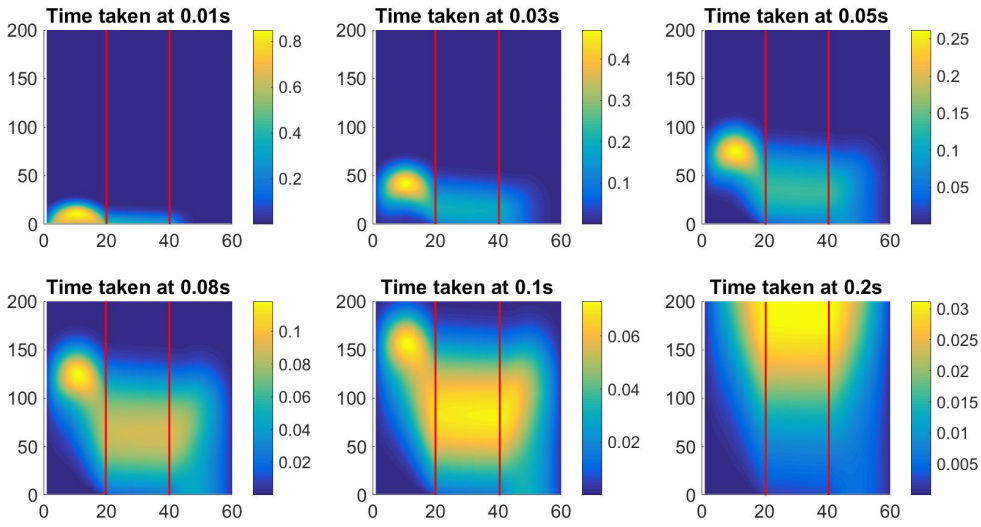


Figure 3.12: Concentration distribution at 6 points in time. Maximum velocity of Poiseuille flow is set at 1 m/s. Diffusion constants for each channel is 2.5e-4 m^2/s and for the membrane 1e-3 m^2/s.

The simulation is run with Poiseuille flow in both channels, with a maximum velocity of 1 m/s. The diffusion coefficient in the membrane is four times higher than in the channels. The result of this simulation is shown in Figure 3.12. The species enters channel one and starts diffusing towards and into the membrane to the second channel. The scale of the concentration is adjusted in each graph to visualize the changes in the overall concentration profile, rather than the actual concentration at each specific location. After 0.2 seconds, the concentration is distributed almost evenly over the system. It 's hard to see whether the convection is larger than the diffusion. Neither force seems to dominate when looking at the moving concentration blob in channel one. This can be verified by reviewing the Sherwood number [22] in channel one. The Sherwood number shows the ratio of the convective and diffusive forces. For particular simulation the Sherwood number is 1.067, indicating that the convection is only slightly higher than the diffusion. There is no velocity in the membrane itself which explains the slower behavior of the species seen in Figure 3.12. Due to the tendency of concentration distributing evenly over a system, it can be explained that the highest concentration will agglomerate in the center of the membrane. Due to these observations, it is concluded that the system behaves as expected.

Spatial and temporal grid refinements are applied to the new system, and the convergence rates in Figure 3.13 correspond to the theoretical rate for almost all schemes. When looking at Figure 3.13 it can be seen that for the Crank-Nicolson scheme the slope deviates from the declining trend when Δt becomes too small. To check if something else is interfering with the convergence, the tolerance of the solver is adjusted to see the convergence at several tolerances. Both the BDF2 and Crank-Nicolson schemes are run with ascending tolerances for the BiCG solver with a Cholesky preconditioner in Figure 3.14. It becomes clear that decreasing the tolerance has a positive effect on the convergence rate of the scheme. However, the anomaly for the Crank-Nicolson scheme stays to exist and due to the positive results from the previous section, BDF2 is adopted as the difference scheme from now on. For the spatial part, the central difference scheme is best applicable since the solution is more accurate. The only case where central difference could fail is when $Pe > 2$, but this is not verified for this simulation.

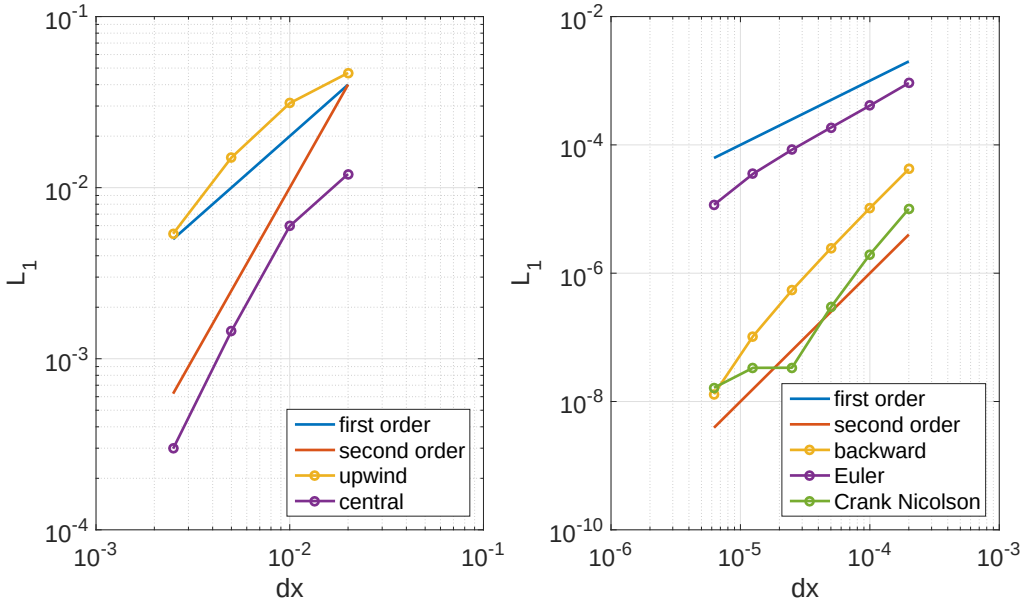


Figure 3.13: Grid refinement for both the spatial (left) and temporal (right) parts of the species equation.

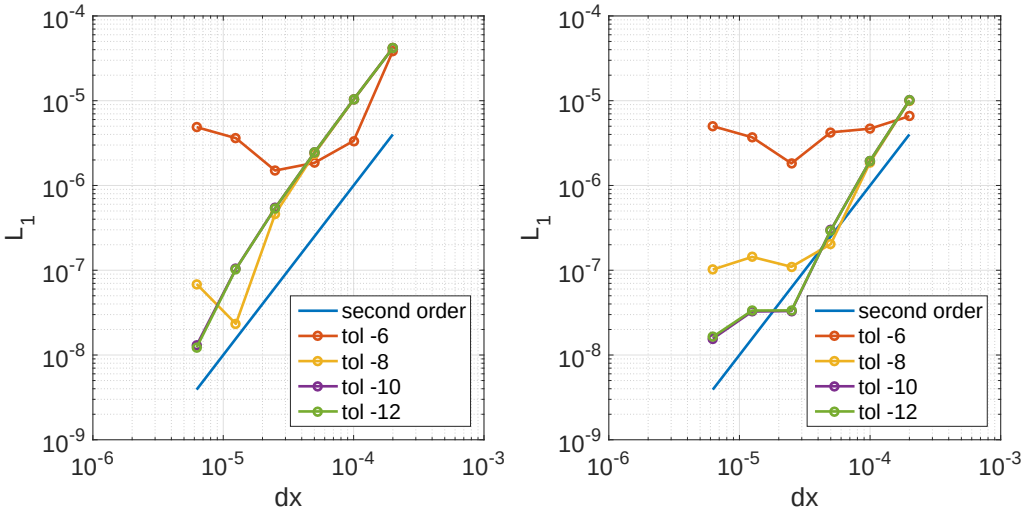


Figure 3.14: Comparison of simulations run with ascending tolerance for two schemes BDF2 (left) and Crank-Nicolson (right).

Chapter 4

Membrane Fouling

The International Union of Pure and Applied Chemistry defines fouling as follows [25]: *"The process that results in a decrease in performance of a membrane, caused by the deposition of suspended or dissolved solids on the external membrane surface, on the membrane pores, or within the membrane pores."* Membrane fouling can cause severe flux decline and affect the quality of the membrane itself. Flux decline is the decrease of the permeation through a membrane.

Fouling is also possible in the lung membrane, think for example about smoking which causes fine particles and tar to "stick" to the membrane and possibly block the pores through which the oxygen should diffuse. Because this affects the overall performance, it is an important aspect to investigate. The mathematical description of membrane fouling can be based on different fouling mechanisms. However, the membrane is often seen as a black box in literature [26, 27, 28] where only the overall diffusion is affected regarding the flux, rather than localized fouling by actual particulates. In this thesis, fouling is first approached by a mathematically described decay of the diffusion coefficient which goes to zero. However, it is not favorable for the diffusion to stop completely which resulted in a bounded diffusion coefficient. The chapter concludes with a function for the diffusion coefficient which is dependent on the concentration diffusing through the membrane. The membrane degrades when concentration is present in the membrane and partially recovers after the concentration moves away again.

4.1 Declining diffusion coefficient

First, a linearly decaying diffusion coefficient is implemented. It is a good way to verify that the diffusion coefficient is behaving as expected and hence can be implemented correctly. A second and more realistic function to approach fouling in the membrane is an exponentially decaying function. As the flux is affected by both the diffusion coefficient and the driving force due to concentration differences, an exponential decay would be more applicable. The functions are defined as

$$\mathscr{D}_{\text{lin}} = \mathscr{D}_0 \left(1 - \frac{t}{\tau}\right) \tag{4.1.1}$$

$$\mathscr{D}_{\text{exp}} = \mathscr{D}_0 \exp\left(\frac{-\alpha t}{\tau}\right) \tag{4.1.2}$$

where $\mathscr{D}_0$ is the initial diffusion coefficient, α is a constant and τ the time when the diffusion coefficient becomes zero ($t = \tau$ means $\mathscr{D} = 0$). The simulation does not stop when $t = \tau$, to see the effect over time on the concentration of species.

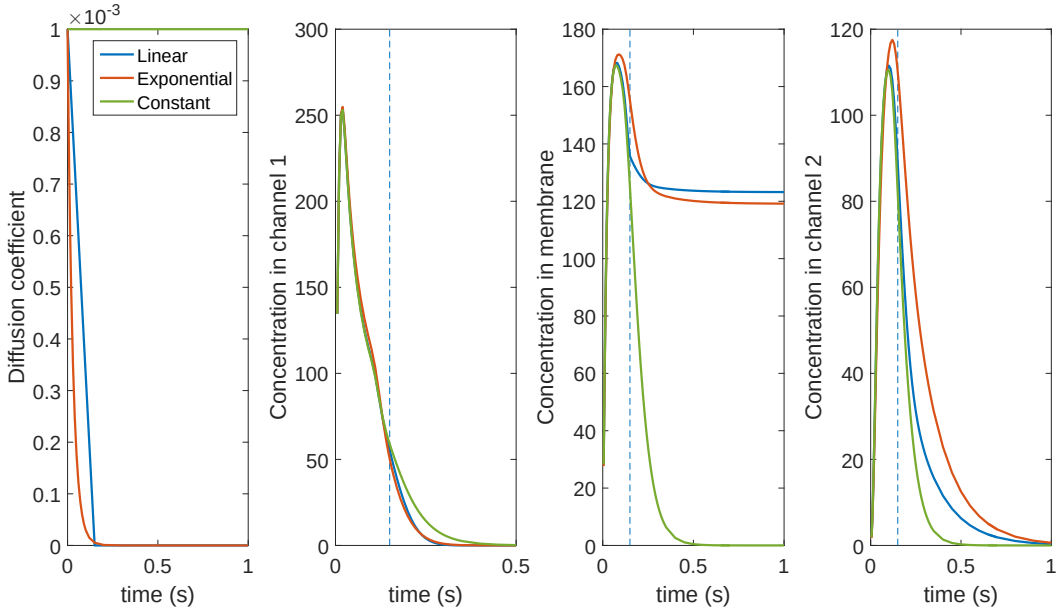


Figure 4.1: The effect of the decline in diffusion coefficient on the concentration in both channels and the membrane. The dashed line indicates $t = 0.15$s when the diffusion coefficient becomes zero for the linear decline.

Both the linear and exponential function of the diffusion coefficient are implemented and compared to a system with a constant diffusion coefficient which is taken as a reference. The result of this comparison is given in Figure 4.1 for each part of the system. In the left figure, the change in the diffusion coefficient of three functions are given, with at $t = 0.15$s an approximately zero diffusion coefficient for the linear and exponential decay. This point in time is chosen since the blob of species has not yet left the channel, when there would be no diffusion through the membrane, and the diffusion coefficient is not instant zero which leaves us with some change in the concentration. The effect of the changing diffusion coefficient on the concentration profile can be seen in the next three figures in Figure 4.1 where the concentration is plotted respectively in channel one, the membrane and channel two. The sum is taken of the concentration of the species in each part of the system for each time step. Each result is observed and compared with what would be theoretically expected to happen to determine whether the simulation is correct. As an indication, the time at which the diffusion is zero for the linear decay (and near zero for exponential) is shown with the striped blue vertical line.

With a decaying diffusion coefficient, the concentration slightly builds up in channel one. This increase is expected as less concentration can exit channel one via the membrane before the outlet of the channel is reached. This effect is only minor for the exponential decay and negligible in case of the linear decay. The opposite happens later in time where the concentration for the constant diffusion (reference) is higher. Since the diffusion has become (almost) zero, only convection is present and the species will leave the channel. As a confirmation of this behavior, a comparison between constant and zero diffusion through the membrane is made in Figure 4.2. The (s-)shape of the red curve is mainly driven by the concentration blob entering and exiting the channel due to convection.

In view of the decreasing diffusion coefficient in the membrane, more and more of the species slows down and is "trapped", which results in the higher peak of concentration. This increase is more significant for the exponential than for the linear decay. A significant part of this "trapped" species reaches the other end of the membrane and diffuses to channel two. The concentration profiles stagnate at a particular value as $\mathscr{D}$ goes to zero as is visible from the graph. The exponential decay moves much

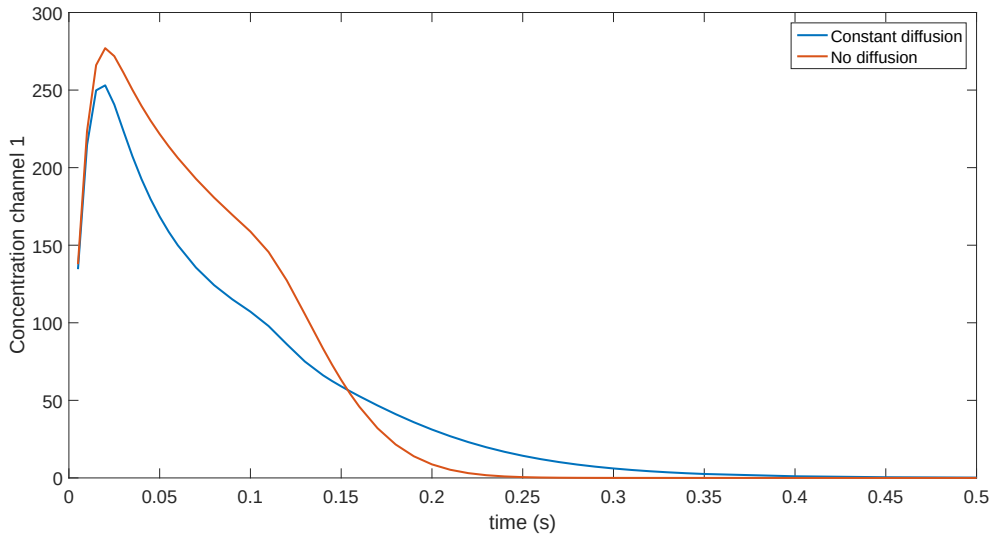


Figure 4.2: The sum of the concentration at each time step for two diffusion cases: constant diffusion and no diffusion.

faster towards zero in channel one compared to the reference, the change in the concentration profile in contrast to the reference is much more significant. However, the exponential diffusion coefficient does not actually reach zero which results in some leakage of the species in channel two. When the linear diffusion coefficient becomes (near) zero in the membrane, it is expected that the concentration in the membrane shows a sharp turn towards a particular value, as the species should not be able to move anymore. The line for the linear decay indicates that there is a significant change in the profile, however, the concentration moves slowly towards a specific value and most importantly only starts after the diffusion coefficient becomes zero. The concentration field at $t = 0.15$s is compared to the steady-state field at $t = 1$s and only at the boundaries, a concentration difference is present, indicating leakage to both channels. To see what happens at the boundaries of the membrane, the central discretization around the boundary is studied and a sketch can be found in Figure 4.3. The flux ($\vec{j} = -\mathscr{D}\nabla c$) at the boundary is determined by the diffusion coefficient and the derivative is discretized by the central discretization scheme. When looking at the left node from the membrane, the flux will always be zero due to the $\mathscr{D}$, however, the right node of the membrane will create a flux since both nodes have concentration. Due to this discontinuity concentration is generated and to bypass this generation a one-sided discretization should be applied only at the first node in each channel. Correcting this in OpenFOAM requires a lot of changes in the solver and is therefore not favorable. In addition, it is assumed that the flux will not be equal to zero and therefore the case of $\mathscr{D} \to 0$ is not realistic.

The concentration of both linear and exponential decay in channel two are expected to go towards zero, just as the reference concentration profile for a constant diffusion coefficient. Either nothing comes through the membrane anymore or the species is held inside the membrane and all the species exit at the outlet. The higher peak in species for the exponential function is caused by a faster decay of the diffusion in the membrane. The decline of both profiles is less steep after about 0.15 seconds than for the reference concentration profile. When looking at Figure 4.4 the center of the concentration blob present in the membrane is located much further along the length in the case of the constant and the linear diffusion coefficient compared to the exponential diffusion coefficient. Therefore, most of the species enters channel two earlier in the channel for the exponential diffusion coefficient, explaining the shift in time and reduced decay of the concentration in the channel.

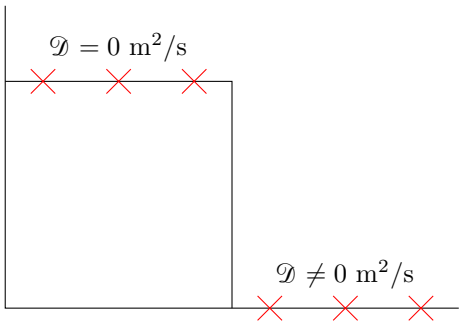


Figure 4.3: The discretization around the right boundary of the membrane.

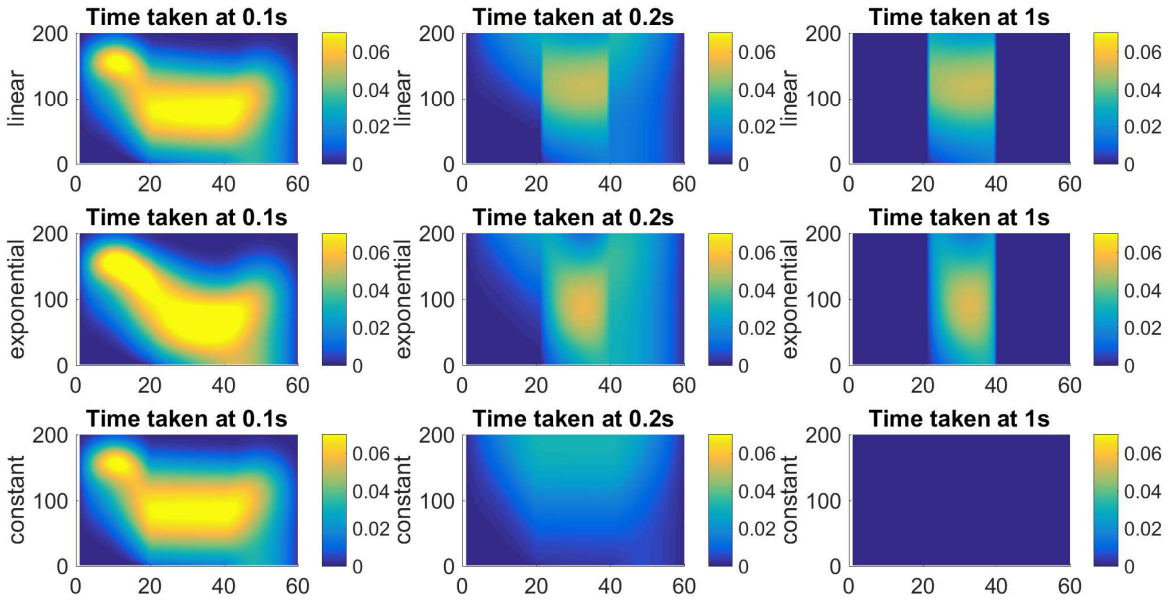


Figure 4.4: The effect of the linear, exponential and constant diffusion coefficient profiles on the concentration distribution over time. Each graph consists of channel one, the membrane and channel two from left to right. The concentration enters the system in the lower left corner.

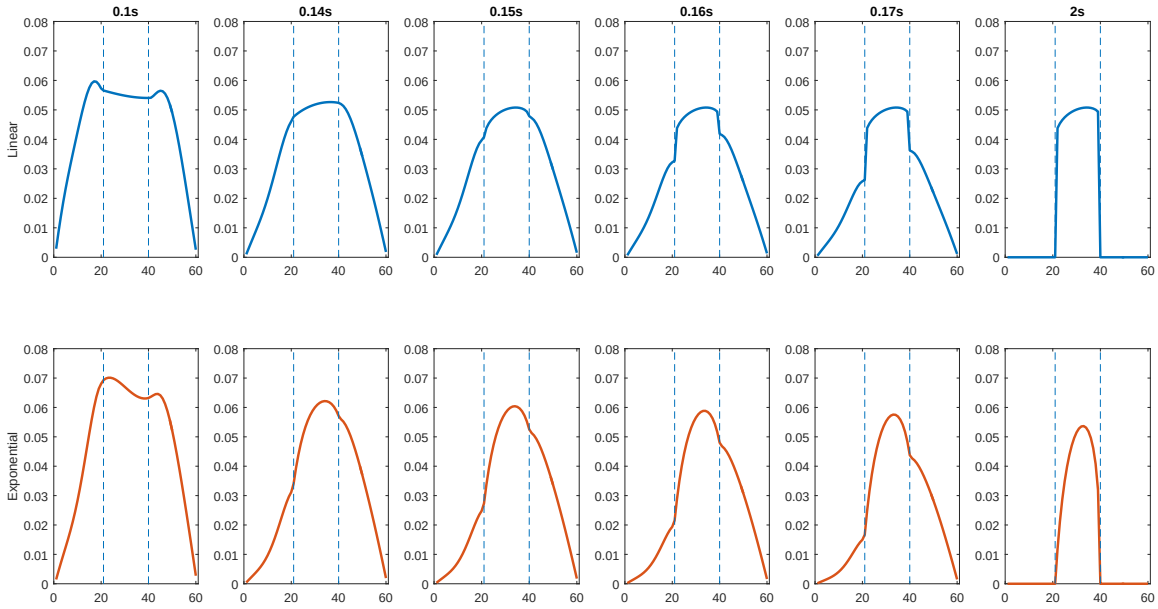


Figure 4.5: Cross-sections of the concentration profiles taken at $y = 100$ from Figure 4.4. Concentration profiles plotted for the linear (blue) and exponential (red) function at different times.

When looking at Figure 4.4 at the concentration profiles for the linear and exponential diffusion functions, the shapes are different but not very clear from the top. For this reason, the concentration profiles are plotted over time in Figure 4.5. The linear function reaches faster the steady state solution than the exponential function. When the diffusion coefficient becomes (near) zero, the concentration in the membrane "freezes" and the species in both channels exit through the outlet without continued exchange with the membrane. This results in two different profiles inside the membrane. It is evident that the solver needs more time to reach the steady-state solution in the second channel than for the first channel. For this case, the computation time should be over 1.5s to reach steady state. The concentration in the channels goes to zero with time, as expected.

The leakage at the boundaries of the membrane is a numerical error and should not happen. Besides that, a zero diffusion coefficient in the lungs is very deadly and therefore, another approximation of the membrane should be considered, e.g. a bounded linear and exponential function for the diffusion coefficient.

4.2 Bounded diffusion coefficient

To prevent the diffusion coefficient from going to zero, a bounding parameter is required. Both the linear and exponential function need to be bounded by this function and are defined as

$$\mathscr{D}_{\text{lin}} = \max\left\{\mathscr{D}_0\left(1-\frac{t}{\tau}\right), \mathscr{D}_1\right\} \tag{4.2.1}$$

$$\mathscr{D}_{\text{exp}} = \mathscr{D}_1 + (\mathscr{D}_0 - \mathscr{D}_1)\exp\left(\frac{-\alpha t}{\tau}\right) \tag{4.2.2}$$

with

$$\mathscr{D}_1 = \mathscr{D}_0 \exp(-\alpha). \tag{4.2.3}$$

The parameter $\mathscr{D}_1$ is the bounding parameter and α is a constant determined by the negative log of the ratio $\mathscr{D}_1/\mathscr{D}_0$. This ratio determines at which percentage of the initial diffusion coefficient the diffusion becomes constant. For now the ratio is chosen to be $\frac{1}{10}$ so the behavior of the species can be examined when the diffusion coefficient becomes small.

The results of the bounded functions are given in Figure 4.6. In the figure it can be seen that the amount of concentration in each part of the system behaves similarly compared to the reference constant diffusion coefficient. When making τ smaller, the diffusion coefficient goes faster to zero and the behavior of the species changes. In Figure 4.6a, τ is taken the same as in Figure 4.1. Again the exponential function has a higher concentration in the membrane and in channel two than the reference diffusion coefficient. The concentration of species in all parts of the system goes to zero as expected. However, comparing to Figure 4.1 the decline in concentration is a bit faster for the linear and exponential function in the membrane and in channel two, due to the concentration going to zero in the membrane. When τ is halved, the linear and exponential function behave the same. When τ is halved again, the exponential function reaches almost instant $\mathscr{D}_1$ which results in a smaller concentration of species diffusing into the membrane. In contrast, the linear decay has a higher concentration in the membrane compared to the previous case. This concludes that the tipping point of τ lies at the point where the concentration is highest in the membrane. Where the low diffusion coefficient at τ before the concentration peak in the membrane has a trapping effect for the linear equation and a blocking effect for the exponential.

A spatial grid refinement is utilized for both functions and they behave according to their theoretical convergence rate (Figure 4.7). The temporal grid refinement is not shown since the error becomes smaller than the machine error and will not have a meaning other than that the used time step is guaranteed to be small enough.

A parameter study is done for the three parameters which have an influence on the function of the diffusion coefficient. Only the exponential function is considered since the function is more general than the linear case. The parameter study is worked out in Appendix B.

4.3 Coupled function for the diffusion coefficient

In this section, a model for the diffusion coefficient is developed to include the dependency on the concentration. The developed model in this section is not based on an actual case but rather builds a platform and basis for a coupled system. The goal is to model the degradation of a membrane which is locally affected by a generic gas species. The diffusion of the membrane should directly be affected by this species and more importantly progress back towards its original value, mimicking the partial healing of the membrane.

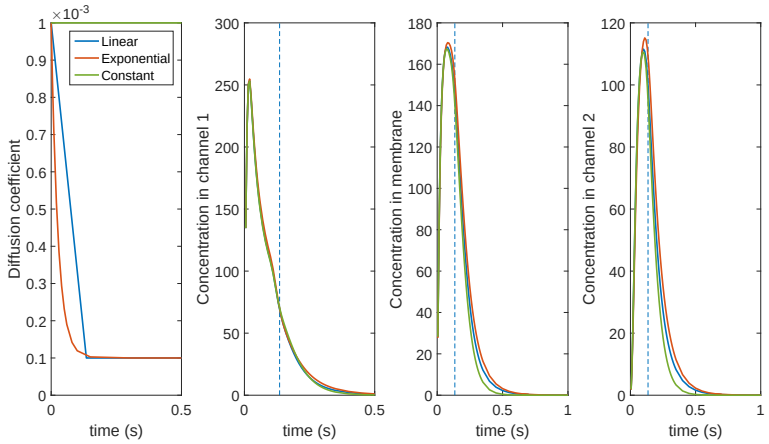
This model can be created by applying a specific equation with a dependency on local species concentration in the membrane. The membrane would be negatively affected by this species and when the concentration decreases the diffusion coefficient should move towards a set lower concentration, representing permanent damage to the membrane.

The bounded exponential function in the previous section is a solution of

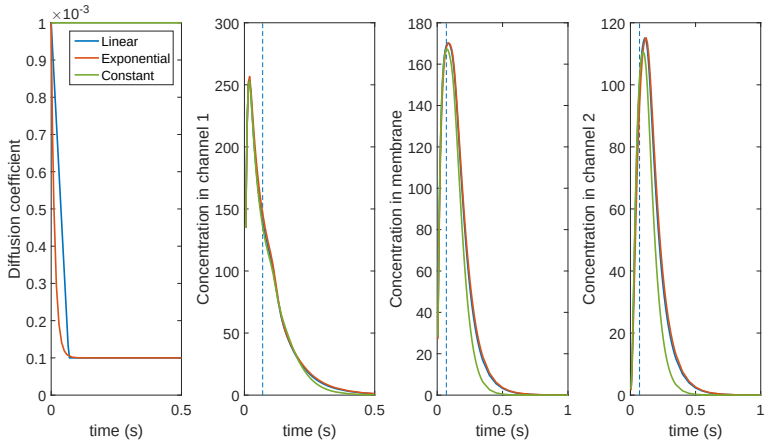
$$\frac{d\mathscr{D}}{dt} = -\frac{\alpha}{\tau}\left(\mathscr{D} - \mathscr{D}_1\right) \tag{4.3.1}$$

with $\mathscr{D}(0) = \mathscr{D}_0$ as initial condition. This equation is implemented in OpenFOAM and the species exchange is simulated. The result of this model is compared to see if the result matches the decay of the exponential diffusion function defined in (4.2.2). The results of both functions match perfectly, indicating a correct implementation of (4.3.1). The only difference was that the simulation employing (4.3.1) had a smaller Δt, otherwise the residual for the species equation did not become small enough.

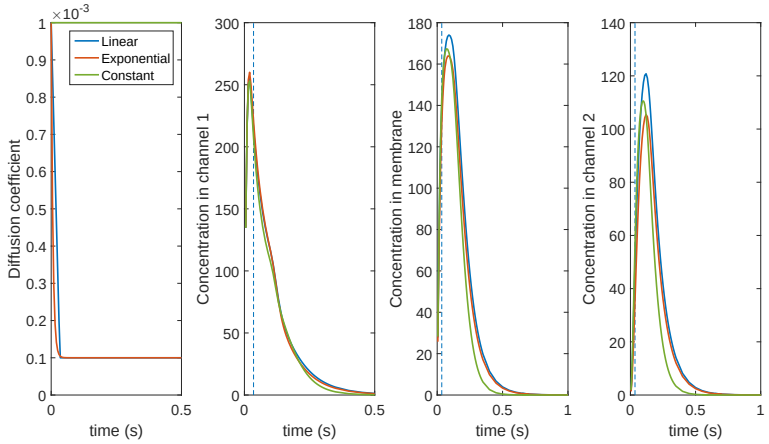
To create a diffusion coefficient in the membrane which is dependent on both time and concentration, equation (4.3.1) is tailored to include the concentration of species. The constants α and $\mathscr{D}_1$ are



(a)



(b)



(c)

Figure 4.6: Concentration profile of two functions for the diffusion constant, bounded by $\mathscr{D}_1$, respectively linear and exponential decay compared to a constant diffusion. The sum is taken over the concentration in each part of the system. The parameter $\tau = 0.15$ (infliction point by striped line) is halved consecutively in (b) and (c).

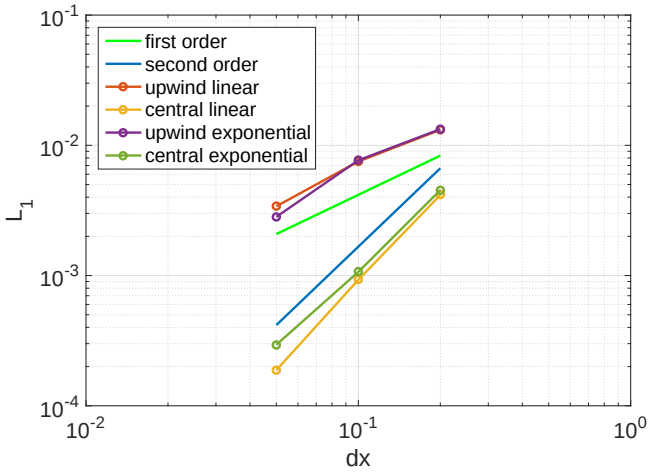


Figure 4.7: Spatial grid refinement of the linear and exponential function of the diffusion coefficient.

replaced by a linear equation, such that:

$$\begin{aligned} \frac{d\mathscr{D}}{dt} &= -\frac{\theta(c)}{\tau}\left[\mathscr{D} - \mathscr{D}_2(c)\right] \\ \theta(c) &= \alpha + \beta \cdot c \\ \mathscr{D}_2(c) &= \gamma + \varepsilon \cdot c \\ \mathscr{D}(0) &= \mathscr{D}_0 \end{aligned} \tag{4.3.2}$$

where α, β, γ, ε and τ are parameters. The values of the parameters have to be determined, to obtain the requested behavior of the diffusion coefficient while with varying concentration. To determine the magnitude of α and γ, a simplified model is studied first, by setting β and ϵ to zero in (4.3.2). In addition, both parameters have restrictions:

$$\alpha > 0$$

$$0 < \gamma < \mathscr{D}_0$$

The parameter α can not be negative as an increase of the diffusion coefficient is not considered. The parameter γ should not be larger than $\mathscr{D}_0$ as again the diffusion coefficient would increase over its maximum value $\mathscr{D}_0$. In addition, γ can not be smaller than zero since the diffusion can not physically be negative. To determine the values of the parameters for the first simulation, the values for α and γ are taken such that the decay of the diffusion coefficient is equal to the exponential function in the previous case. For simplicity, the parameter τ is also taken the same as before.

To obtain values for both β and ε in the right order of magnitude, the values are taken equal to α and γ respectively. The order of magnitude is sufficient to assess the effect of the diffusion coefficient and the behavior of the affected concentration distribution. Comparing the results for each simulation in a 2D form would be difficult and for this reason, the average of one specific row in the matrix of the membrane is chosen and compared to the same row in another calculation with different parameter values. A row near the inlet of the channel is preferred to apply the effect of a high concentration on the diffusion coefficient via the parameters. The effect of each parameter is therefore increased in magnitude in each evaluation to make comparison more effective. Figure 4.8 compares the result of three chosen grid rows near the inlet of the channel. Mind that only the diffusion in the membrane is evaluated and shown. The row of the grid cells at $x = 25$ is chosen as an arbitrary row as the results are quite similar.

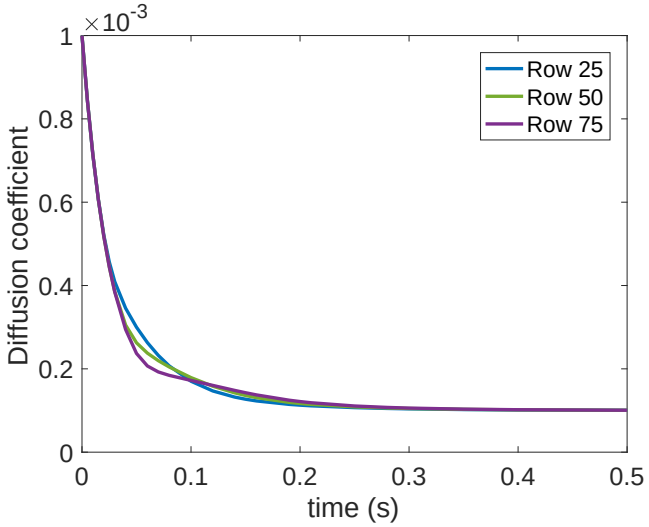


Figure 4.8: Comparison of the average diffusion coefficient over time of a specific grid cell row in the membrane

To confirm the difficulty of assessing a 2D plot the results for a simulation is shown in Figure 4.9 with the parameters $\beta = \alpha$ and $\varepsilon = -\gamma$. Although difficult to compare, this form of assessment does show that the diffusion coefficient is affected by the main concentration blob moving along the channel. After some time it becomes visible that the diffusion coefficient starts to increase again and that the damage done is partly being reversed (see the color bars). The simulation is run till 1 second, however, to recover the entire membrane more time is needed.

The parameters β and ε are both simulated with a positive and negative value and the curves are visible in Figure 4.10a. When one of the parameters is considered, the other one is taken as zero to exclude interference on the behavior. These curves for β and ε are compared to the function in which the parameters are both zero. All curves end up at the asymptotic value γ, however, the curve with $\varepsilon < 0$ has a faster decline than the curve with $\varepsilon > 0$ which is more evident in Figure 4.10b. Both curves for β deviate from the reference line and $\beta > 0$ will reach the asymptotic value ahead of $\beta < 0$.

It is not immediately clear what the effect of these parameters is on the diffusion coefficient and for this reason, the values for both parameters are increased. In Figure 4.11 the parameter β is increased significantly and the inflection point appears earlier in time. However, when the value of β becomes negative, an increase in diffusion coefficient can be seen, which can be explained by looking at the analytic solution of equation (4.3.2).

$$\mathscr{D}(c,t) \quad = \quad (\gamma + \varepsilon \cdot c) + [\mathscr{D}_0 - (\gamma + \varepsilon \cdot c)] \exp\left(-\frac{(\alpha + \beta \cdot c)\, t}{\tau}\right) \tag{4.3.3}$$

When the term $\alpha + \beta \cdot c$ becomes negative due to β, the exponent becomes positive which results in an increasing diffusion coefficient. When the concentration increases, the term could become big and the simulation explodes.

The values for ε are increased to see the behavior in more detail, see Figure 4.11. This figure shows that ε causes a kind of damping effect of the diffusion coefficient, a clear overshoot is seen when moving towards the bounded value. When ε becomes negative, the function becomes underdamped and approaches zero faster than in the case of critical damping (the other lines), but has an overshoot and then approaches the bounding function. The diffusion coefficient comes close to the zero, so increasing the value of ε could make for an interesting case. The value for ε is made twice as big and this results in a negative diffusion coefficient. In Figure 4.12 the diffusion coefficient profile and the

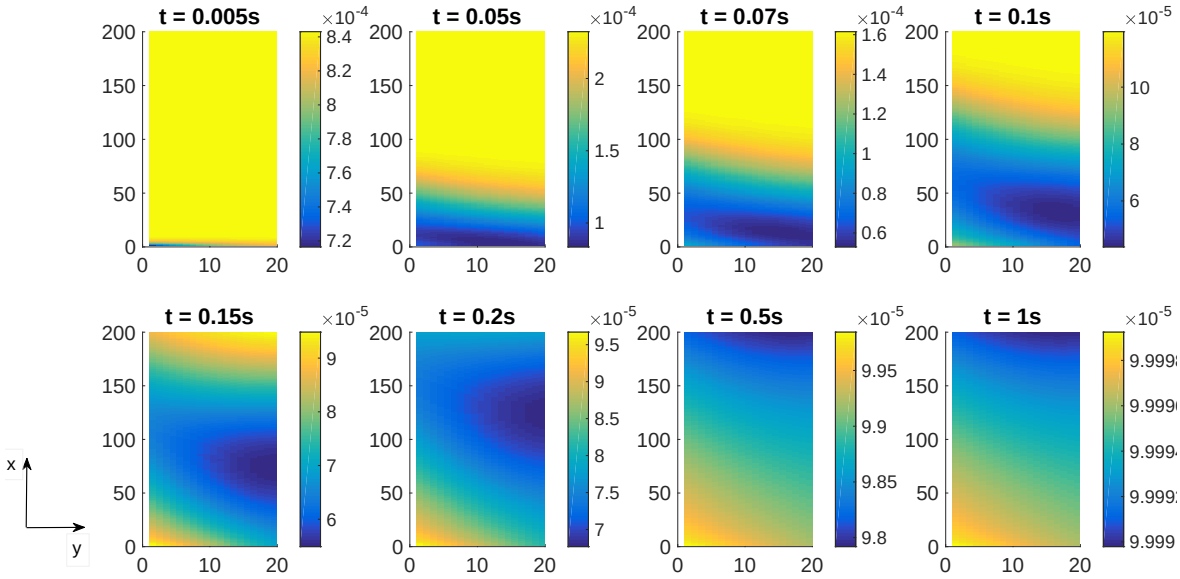
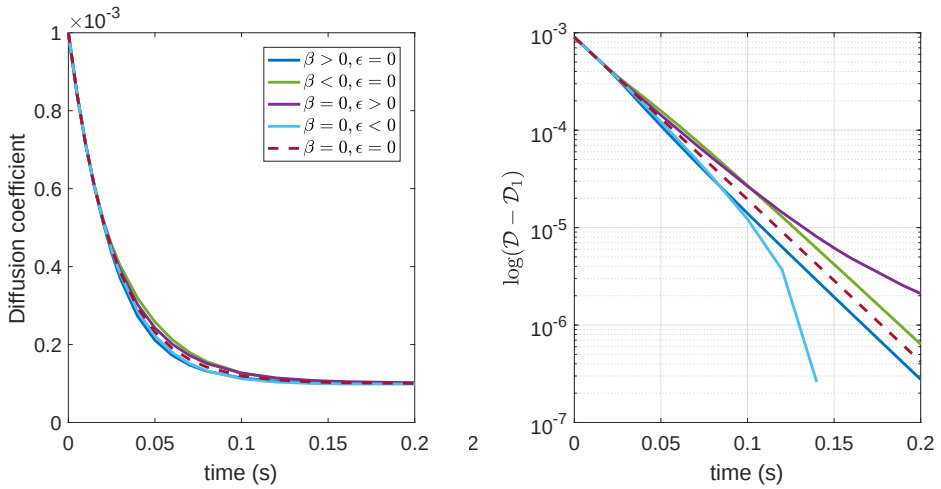
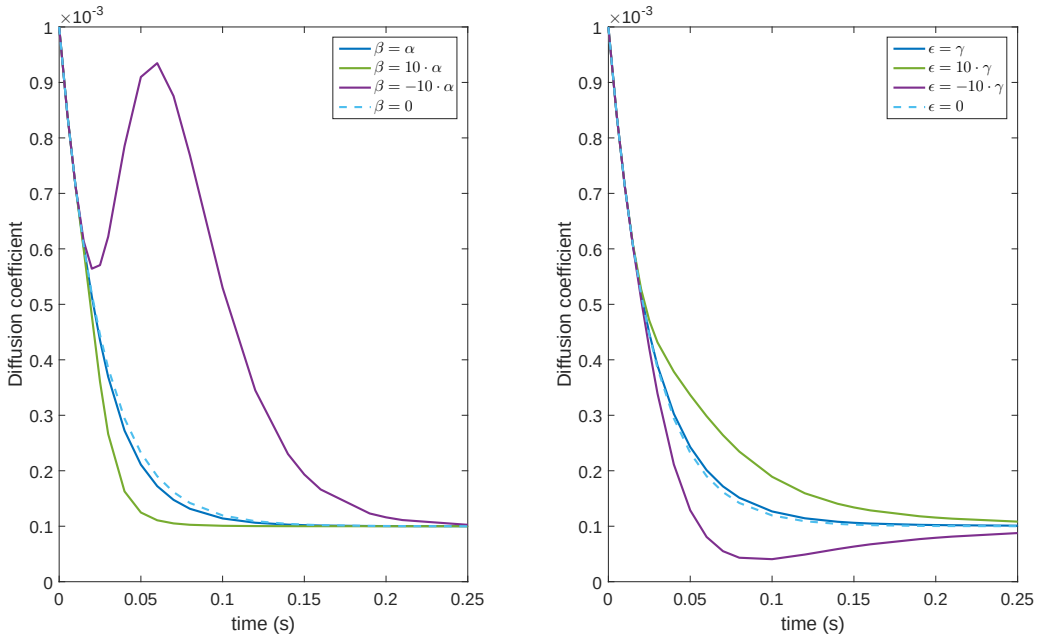


Figure 4.9: The diffusion coefficient profile as a function of concentration over the membrane. With $\beta = \alpha$ and $\varepsilon = -\gamma$.



(a) (b)

Figure 4.10: (a) Average diffusion coefficient on row 25 of the membrane over time with different parameter settings. (b) Log of the diffusion coefficient minus the bounding parameter over the time with different parameter settings.



(a) (b)

Figure 4.11: The diffusion coefficient over time for different parameter values of β and $\varepsilon = 0$ (left). The diffusion coefficient over time for different parameter values of ε and $\beta = 0$ (right). Both with a striped reference line where $\beta = \varepsilon = 0$.

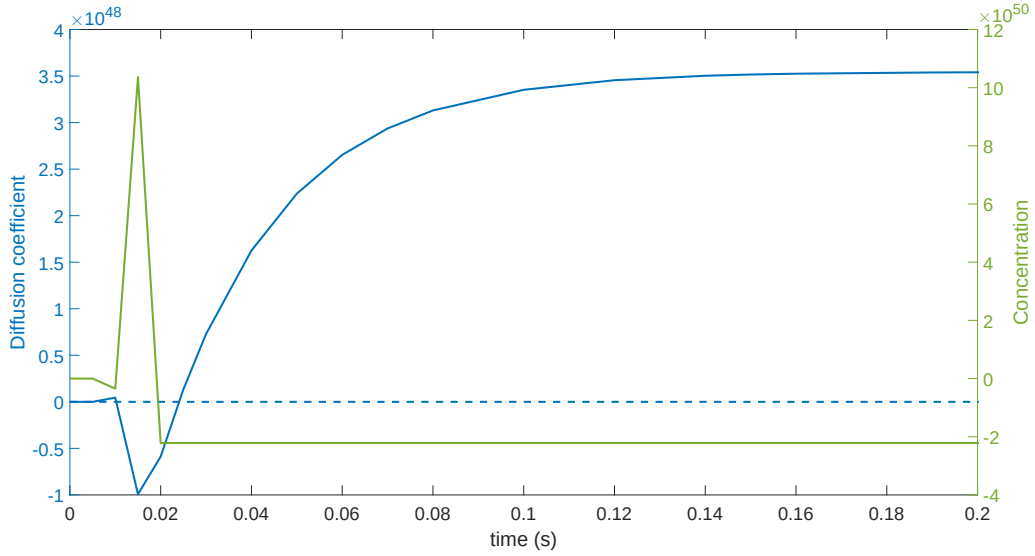


Figure 4.12: The diffusion coefficient over time for $\varepsilon = -20\gamma$ and $\beta = 0$ with the corresponding concentration. Showing unrealistic values due to a too negative value for ε.

associated concentration profile are depicted. When the diffusion coefficient becomes negative, the concentration increases drastically and with this increasing concentration, the diffusion coefficient rises far beyond the reference diffusion coefficient. It is clear that the final concentration in the membrane can never be negative, so the diffusion coefficient should not become smaller (or equal) to zero.

The result in which ε is the negative of γ already shows the required effect of a "repairing" membrane which was introduced earlier. However, the diffusion coefficient which is reached over time is very low, almost 10%, which would result in nearly 90% irreversible damage to the membrane. It is assumed that irreversible damage would not be of such a significant amount and therefore about 20% of irreversible damage is chosen as a target. A new calculation is made in which the bounding function (parameter γ) is 80% of $\mathscr{D}_0$. The result of this base case can be found in Figure 4.13. The red line marks the line to which the curve should converge and it is visible that the diffusion coefficient decreases fast due to the concentration and passes the bounding parameter before recovering slowly. The results in this figure show that the goal of this section is reached as the diffusion coefficient in the membrane is affected by the concentration in a negative way after which the membrane "repairs" and is left with specific irreversible damage. To expand the application options of the equation the parameters can be adapted to manipulate the shape of the curve.

A parameter study is applied to find the effect of each parameter on the shape of the curve. The parameters and the shape of Figure 4.13 are taken as a reference. In Figure 4.13, the parameter β is taken as zero, however, during the following parameter study β will be given a value to see if the curve can be optimized. The parameters α and γ are taken for a specific value since the line to which the curve should converge is set to 80% of $\mathscr{D}_0$. For this reason, both ε and β can be adjusted based on respectively γ and α. The values are respectively multiplied and divided by 10 to determine which of the three manipulations can be used to adjust the shape. For instance, $\varepsilon = -\gamma$, $\varepsilon < -\gamma$, and $\varepsilon > -\gamma$ are compared to each other in Figure 4.14a. The striped line corresponds to the reference from Figure 4.13 and will be called the reference from here on out. For ε it is clear that the manipulation $\varepsilon > -\gamma$ can be used to produce the overshoot. The same is done for β, where the value is taken equal to α, divided by 10 and multiplied by 5. The parameter α is multiplied by 5 instead of 10 due to the large positive jump in diffusion which can not be compared to the other simulations. With increasing β, the minimum of the curve decreases which results in a steeper recovery of the diffusion coefficient

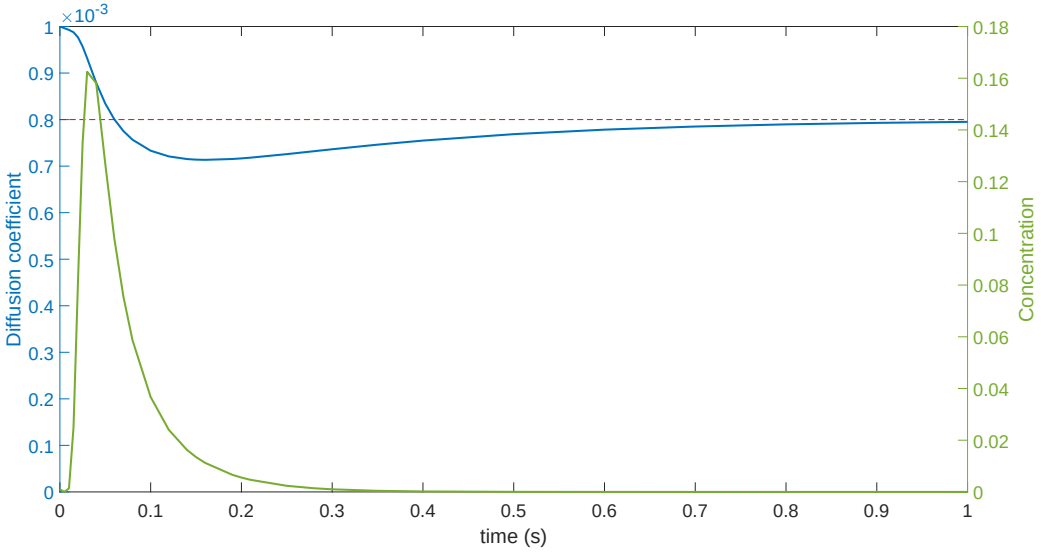


Figure 4.13: Base case diffusion coefficient for parameter study with corresponding concentration profile. Parameters: $\gamma = 0.8 \cdot \mathscr{D}_0$, $\alpha = -\log(\gamma/\mathscr{D}_0)$, $\beta = 0$, $\varepsilon = -8e-3$ and $\tau = 0.6$.

(Figure 4.14b). In addition, the parameter τ is halved consecutively and this has also an influence on the minimum of the curve as well as the width of the overshoot, as can be seen in Figure 4.14c. Therefore, the deeper the minimum, the faster the rise back to the bounding parameter. A gloss upon these restrictions is that the diffusion coefficient should never become zero, due to physically impossible results.

There are three things in Figure 4.13 which may be manipulated to get the result you want. How low the minimum goes is dependent on the parameters ε, β and τ, the width of this minimum is reliant on the parameter τ and the height of the bounding function is determined by γ.

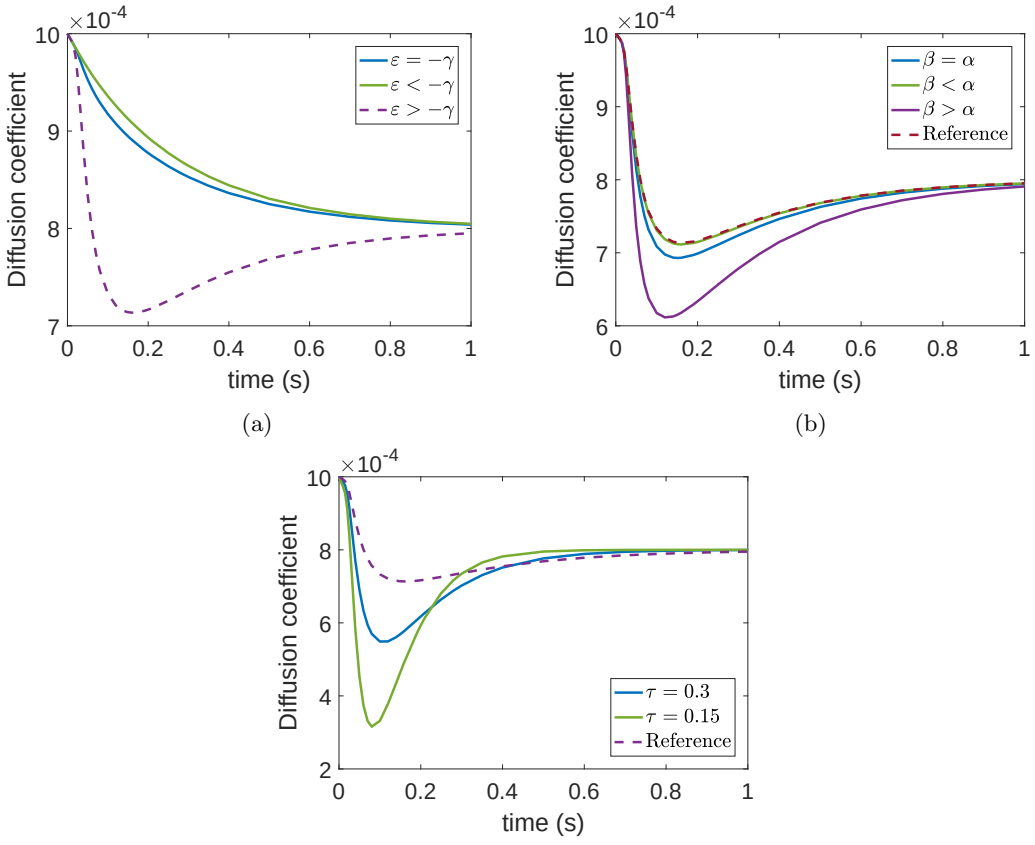


Figure 4.14: Results of the parameter study of (a) the parameter ε, (b) the parameter β and (c) the parameter τ.

Chapter 5

Conclusion

The goal of this thesis was to build a basis for the computation of gas exchange between two fluids separated by a membrane and the effect of membrane degradation on the gas exchange. The foundation of this thesis is based on the lung-on-a-chip technology developed by the Wyss Institute.

An existing solver in the open-source software OpenFoam was adjusted to create three regions (fluid-solid-fluid) to mimic the interior of the chip. A first basic (large) model containing a Poiseuille velocity profile in both channels was made and the correct theoretical order of accuracy was obtained. Subsequently, the dimensions of the system were reduced to centimeters to approach a more realistic size. The theoretical order of accuracy was determined for the new system and found equal to the previous case. Concluding that the system was indeed scalable and the same solutions could be obtained. The next step was the implementation of the species transport equation to include gas exchange in the model, by means of diffusion through the membrane. Based on each determination of order of accuracy, a difference scheme was successfully chosen for the spatial and temporal part of the velocity and concentration derivatives.

To extend the properties of the model, a basic form of membrane fouling was added to the simulation. The diffusion coefficient in the membrane was changed via a specific equation, in this case a linear and exponential decaying function. These equations were first set to be only dependent on time and to decay to zero. Results showed that with this decay towards zero, there was a numerical error at the boundaries of the membrane. For this reason the case of diffusion to zero was discarded.

A bounded decay equation, dependent on time, was considered to circumvent the previous problem. This bounded function was progressively expanded to include dependency on concentration. The final model consisted of a concentration and time dependent equation with adjustable parameters which could be manipulated to obtain a desired diffusion decay.

The goal of this thesis was reached by creating a basis for the computation of gas exchange and by providing a basis to simulate membrane degradation via a concentration dependent diffusion equation with adjustable parameters.

Chapter 6

Discussion and Recommendations

The application of OpenFOAM as a computational fluid dynamic simulator has shown to be time-consuming. The solver is far from working as the lab-on-a-chip designed by the Wyss Institute. There are several things which should be added/changed to make it more applicable:

- The control over the diffusion in the membrane is an important aspect of the model. Diffusion should only promote transport in the y-direction, while in the current application it also diffuses in the x-direction across the membrane. This is not representable for a lung-on-a-chip application as the diffusion through the membrane can be seen as small "vertical" portals through which the particles diffuse.
- The direction of flow through the channels are driven by the blood flow and air in the alveoli. Depending on the application in the lung-on-a-chip this could also be counter-current flow. When the blood and air channels flow counter-current of each other, a better overall mass transfer can be achieved. A higher concentration gradient between both channels is created as the oxygen-poor blood outlet is contacted with an oxygen-rich air inlet. An extension of the model would be the implementation of this counter-current flow and comparison with a co-current flow to evaluate the advantages.
- The concentration profile is a blob of species entering the channel at one time in the simulation. Breathing does not happen once and the condition should be changed to a periodic condition. To implement the behavior of exhaling and inhaling, the species inlet and profile should become a function of time. Cascaded simulations could mimic the actual respiration cycle.
- The exchange between oxygen and carbon dioxide should be implemented to evaluate the effect of multiple species in the model.
- The diffusion is dependent on the concentration of the air, while air is not toxic to the lung membrane. An extra specie should be introduced to mimic the tar in smoke to reduce the diffusion coefficient. The effect of the reduced diffusivity can then be studied for the oxygen concentration in the blood vessel. The other way around is also an option; medication can be added to an injured membrane.
- The shape of the simulated membrane is still flat, while the initial goal was to change the geometry of the membrane into a curved or a sine wave. A simple simulation of a flat membrane posed several challenges which will only be expanded when changing in shap of the membrane. Laminar flow might not be applicable (realistic) any more and different models may need to be applied.
- The membrane simulated in OpenFOAM consists of only one layer, while in Figure 2.3 it became apparent that there are three layers through which the concentration should diffuse in 12 diffusion

steps. A literature study should be done to check whether there is any information about these diffusion steps which can be simulated.

- The viscosity in the channels should be different from each other and should mimic air and blood in the channels. This will result in different flow rates and the species will be slower/faster in the channel.

Bibliography

[1] J.A. DiMasi, R.W. Hansen, H.G. Grabowski and L. Lasagna, "Cost of innovation in the pharmaceutical industry.," *Journal of Health Economics*, vol. 10, pp. 107–142, 1991.

[2] A. Akhtar, "The flaws and human harms of animal experimentation," *Cambridge Quarterly of Healthcare Ethics*, vol. 24, no. 4, pp. 407–419, 2015.

[3] E.W. Esch, A. Bahinski and D. Huh, "Organs-on-chips at the frontiers of drug discovery," *Nature Reviews Drug Discovery*, vol. 14, pp. 248–260, 2015.

[4] A. Sertkaya, H.H. Wong, A. Jessup, T. Beleche, "Key cost drivers of pharmaceutical clinical trials in the united states.," *Clinical Trials*, vol. 13, pp. 117–126, 2016.

[5] R. Collier, "Rapidly rising clinical trial costs worry researchers.," *CMAJ: Canadian Medical Association Journal.*, vol. 180, pp. 277–278, 2009.

[6] L. Levis, "Mimicking organs," *Harvard magazine*, 2016.

[7] D. Huh, B.D. Matthews, A. Mammoto, M. Montoya-Zavala, H.Y. Hsin and D.E. Ingber, "Reconstituting Organ-Level Lung Functions on a Chip," *Science*, vol. 328, pp. 1662–1668, 2010.

[8] Longfonds, "Long-op-een-chip." `https://www.longfonds.nl/lopende-onderzoeken/long-op-een-chip`, Accessed: 22-12-2016.

[9] B. Alberts, A. Johnson, J. Lewis, *Molecular Biology of the Cell*. New York: Garland Science, fourth ed., 2002.

[10] W.F. Boron and E.L. Boulpaep, *Medical Physiology: A Cellular and Molecular Approach*. Philadelphia: Saunders Elsevier, second ed., 2009.

[11] K. A. Zimmermann, "Respiratory system: Facts, function and diseases," October 2014. `http://www.livescience.com/22616-respiratory-system.html`, Accessed: 10-03-2016.

[12] R.H. Petrucci, F.G. Herring, J.D. Madura, C. Bissonnette, *General Chemistry: Principles and Modern Applications*. Pearson Prentice Hall, 10th ed., 2010.

[13] M. T. College, "The cardiovascular system: Blood vessels." `http://classes.midlandstech.edu/carterp/Courses/bio211/chap19/chap19.html`, Accessed: 9-07-2016.

[14] D.N. Ku, "Blood flow in arteries," *Annual Review of Fluid Mechanics*, vol. 29, pp. 399–434, 1997.

[15] J. Anderson, *Computational Fluid Dynamics: The Basics With Applications*. McGraw-Hill, 1995.

[16] H.K. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics. The finite volume method*. England: Longman Scientific & Technical, 1995.

[17] OpenCFD Ltd (ESI Group), "The open source CFD toolbox," 2004. Accessed: 15-11-2016.

[18] C. J. Greenshields, *OpenFOAM User Guide version 4.0*. OpenFOAM Foundation Ltd., June 2016.

[19] K. W. Morton & D. Mayers, *Numerical Solutions of Partial Differential Equations*. Cambridge: Cambridge University Press, second ed., 2005.

[20] E. Süli, D. Mayers, *An Introduction to Numerical Analysis*. Cambridge: Cambridge University Press, 1 ed., 2006.

[21] S.V. Patankar, *Numerical Heat Transfer and Fluid Flow (Series in computational methods in mechanics and thermal sciences)*. Washington: Taylor & Francis, 1980.

[22] R.B. Bird, W.E. Stewart and E.N. Lightfoot, *Transport Phenomena*. New York: John Wiley & Sons, Inc., revised second ed., 2007.

[23] Y.A. Cengel and J.M. Cimbala, *Fluid Mechanics. Fundamentals and applications.* New York: McGraw-Hill, 2006.

[24] Kungliga Tekniska högskolan, "Order of accuracy." `https://www.kth.se/social/upload/5287cd2bf27654543869d6c8/Ant-OrderofAccuracy.pdf`, Accessed: 24-05-2016.

[25] W.J. Koros, Y.H. Ma and T. Shimidzu, "Terminology for membranes and membrane processes (iupac recommendations 1996)," *Journal of Membrane Science*, vol. 120, pp. 149–159, 1996.

[26] W.R. Bowen, J.I. Calvo, A. Hermindez, "Steps of membrane blocking in flux decline during protein microfiltration," *Journal of Membrane Science*, vol. 101, pp. 153–165, 1994.

[27] G. Bolton, D. LaCasse and R. Kuriyel, "Combined models of membrane fouling: Development and application to microfiltration and ultrafiltration of biological fluids," *Journal of Membrane Science*, vol. 277, pp. 75–84, 2006.

[28] A.L. Lim and R. Bai, "Membrane fouling and cleaning in microfiltration of activated sludge wastewater," *Journal of Membrane Science*, vol. 216, pp. 279–290, 2003.

List of Symbols

A	tissue area	m^2
c	concentration of some species	mol
$C_{\max}$	maximum ratio for the Courant number	
d	thickness of the membrane	m
$\mathscr{D}$	diffusion coefficient	$\mathrm{m}^2\,\mathrm{s}^{-1}$
$\mathscr{D}_0$	initial diffusion coefficient	$\mathrm{m}^2\,\mathrm{s}^{-1}$
$\mathscr{D}_1$	bounded function for the diffusion coefficient	$\mathrm{m}^2\,\mathrm{s}^{-1}$
$\mathscr{D}_L$	diffusing capacity for the lung	mL/(min mm Hg)
$\mathscr{D}_M$	membrane diffusing capacity	$\mathrm{m}^2\,\mathrm{s}^{-1}$
h	height of the channel	m
$\vec{j}$	concentration flux	$\mathrm{mol}\,\mathrm{m}^{-3}$
L	length of the channel	m
n	order of convergence	
N	amount of grid points	
P	partial gas pressure	$\mathrm{kg}/(\mathrm{ms}^2)$
Pe	Péclet number	
Re	Reynolds number	
s	solubility of the gas in water	
Δt	time step	s
u	x-direction velocity	$\mathrm{m}\,\mathrm{s}^{-1}$
$u_{\Delta x}$	solution at a specified grid	
v	y-direction velocity	$\mathrm{m}\,\mathrm{s}^{-1}$
V_c	volume of blood in the pulmonary arteries	
x, y, z	coordinates	m
Δx	x-direction width of the control volume	m

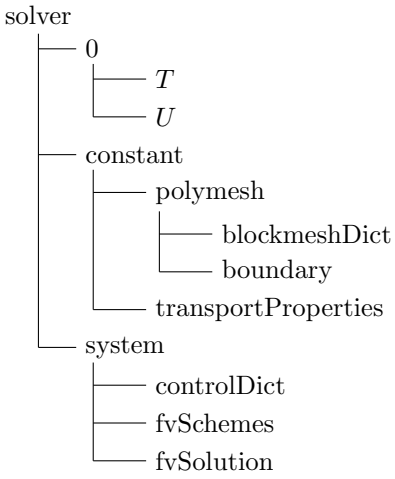
ν	kinematic viscosity	$\mathrm{m}^2\,\mathrm{s}^{-1}$
μ	dynamic viscosity	$\mathrm{kg}/(\mathrm{ms})$
ρ	density	$\mathrm{kg}\,\mathrm{m}^{-3}$
τ	time at which the diffusion coefficient becomes zero	s
θ	rate constant for the binding of O_2 to Hb	

Appendix A

Quick guide for OpenFOAM

A.1 Pre-processing

OpenFOAM has the following case structure which is used for the pre-processing:



In these files the data for the simulation can be found, e.g. mesh, fields, properties, control parameters etc. The structure is the same for most solvers and when a sufficient solver is chosen, the files in the case structure should be altered to match the new simulation. At first, the geometry should be modified in `blockMeshDict` to the block structure of the mesh. Then the initial, boundary and simulation parameters should be defined. Below the different files in the directories are explained in a bit more detail. This simplest thing to do, is explain the files based on a simple example. We have a closed box with some heat properties. When $t = 0$, the temperature is decreased on one side of the box which will result in a temperature gradient over time.

A.1.1 blockMeshDict

In OpenFOAM, all geometries are generated in 3 dimensions, but since our case is in 2D we make the depth of the cube small compared to the length and the width (1m $\times$ 1m $\times$ 0.1m) with a uniform mesh of $20 \times 20 \times 1$. With this knowledge the file `blockMeshDict` can be adapted as in the code found below. In line 1-15 some general information about OpenFOAM is printed, and for the remainder of this document these lines will be removed from the quoting of the case files.

The `blockMeshDict` file starts with `vertices`, where the coordinates of each point of the outline

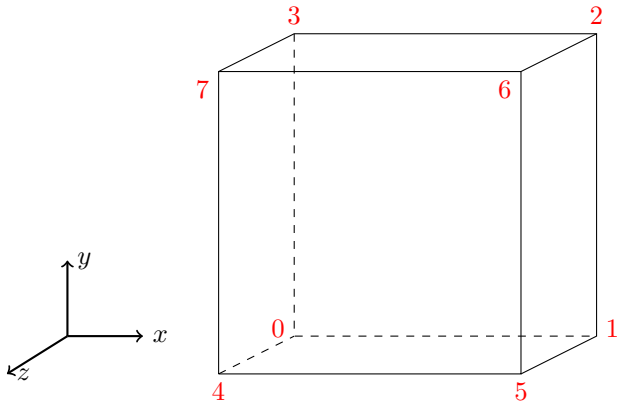


Figure A.1: Block structure of the mesh.

of the block are placed. The order in which you number the corners is free to choose, however, the common manner is depicted red in Figure A.1. In the brackets behind `blocks` the numbering of the points is written, behind that the mesh size in each direction and the cell expansion ratios.

In line 40-76 (`boundary` patch) four regions are defined and the name is the user's choice (keyword), for this case the keywords are chosen as depicted in Figure A.2. These names will be used as identifiers throughout other data files. The line below the keyword says `type` and here you have several options (with preprogrammed conditions) which can be found on page 139 of the OpenFOAM User Guide [18]. In this example, it is chosen to use `wall` since the box is closed. In the `faces` patch the corresponding side of the block structure is entered.

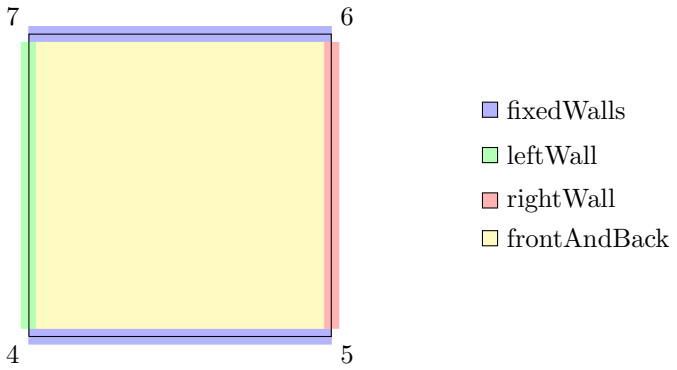


Figure A.2: 2D block mesh.

There are two input fields which are left open, `edges` and `mergePatchPairs`. If the geometry contains no curved edges, `edges` may be omitted. If the mesh is created by more than one block, the `mergePatchPairs` will "glue" the patch together, which is not necessary in this case.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.2.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     object      blockMeshDict;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
16
17  convertToMeters 1;
18
19  vertices
20  (
21      (0 0 0)
22      (1 0 0)
23      (1 1 0)
24      (0 1 0)
25      (0 0 0.1)
26      (1 0 0.1)
27      (1 1 0.1)
28      (0 1 0.1)
29  );
30
31  blocks
32  (
33      hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
34  );
35
36  edges
37  (
38  );
39
40  boundary
41  (
42      fixedWalls
43      {
44          type wall;
45          faces
46          (
47              (2 3 7 6)
48              (1 0 4 5)
49          );
50      }
51      leftWall
52      {
53          type wall;
54          faces
55          (
56              (0 3 7 4)
57          );
58      }
59      rightWall
60      {
61          type wall;
62          faces
63          (
64              (1 2 6 5)
65          );
66      }
67      frontAndBack
68      {
69          type empty;
70          faces
71          (
72              (0 1 2 3)
73              (4 5 6 7)
74          );
75      }
76  );
77
78  mergePatchPairs
79  (
80  );
81
82  // ************************************************************************* //
```

A.1.2 boundary

The boundary file contains the boundary specifications to construct the mesh. The number at the beginning of the file is the amount of patches defined. The types of the patches are the same as in `blockMeshDict`. The next entry is `nFaces` which is the number of faces in the patch, e.g. `fixedWalls` has 20 grid points at the top and 20 grid points at the bottom which result into 40 `nFaces`. The last entry is `startFace` which is the index into the face list in the patch. `startFace` is not necessary to determine and you can leave it blank. These values will be added during the simulation itself.

```
16  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18  4
19  (
20      fixedWalls
21      {
22          type            wall;
23          nFaces          40;
24          startFace       760;
25      }
26      leftWall
27      {
28          type            wall;
29          nFaces          20;
30          startFace       800;
31      }
```

```
32      rightWall
33      {
34          type            wall;
35          nFaces          20;
36          startFace       820;
37      }
38      frontAndBack
39      {
40          type            empty;
41          inGroups        1(empty);
42          nFaces          800;
43          startFace       840;
44      }
45  )
46
47  // ************************************************************************* //
```

A.1.3 transportProperties

Thermal diffusivity (in OpenFOAM defined as `DT` and in literature as α) is the thermal conductivity (k) divided by density (ρ) and specific heat capacity (c_p) at constant pressure:

$$\alpha = \frac{k}{\rho c_p}. \tag{A.1.1}$$

The unit of thermal diffusivity is m^2/s and this can be seen in the code in between the brackets. The format for this dimensionset is `[kg m s K mol A cd]` and each of the values corresponds to the power of each of the base units of measurement. The dimensions are important since they are checked for every run. If there is a dimension mismatch, the simulation will not run.

```
16  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18  DT              DT [ 0 2 -1 0 0 0 0 ] 0.01;
19
20
21  // ************************************************************************* //
```

A.1.4 T

The `internalField` is set to uniform 273K, which means that the initial temperature is uniformly distributed in the bulk. `fixedWalls` has type `zeroGradient`, which indicates that there is no flux to the surrounding. `leftWall` and `rightWall` have type `fixedValue` which allows to specify the temperature at that wall. For this case, the `leftWall` and the bulk have the same temperature, and the `rightWall` has a different temperature which will result in a gradient over time since the internalField will adjust.

```
15  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17  dimensions      [0 0 0 1 0 0 0];
18
19  internalField   uniform 273;
20
21  boundaryField
22  {
23      fixedWalls
24      {
25          type            zeroGradient;
26      }
27
28      leftWall
29      {
30          type            fixedValue;
31          value           uniform 273;
32      }
33
34      rightWall
35      {
36          type            fixedValue;
37          value           uniform 100;
38      }
39
40      frontAndBack
41      {
42          type            empty;
43      }
44  }
45
46  // ************************************************************************* //
```

A.1.5 U

There is no flow in this example only temperature is of the essence, so `internalField` and all walls are set to uniform (0 0 0). When flow is present, this file can be modified in the same manner as is done for the temperature.

```
15  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17  dimensions      [0 1 -1 0 0 0 0];
18
19  internalField   uniform (0 0 0);
20
21  boundaryField
22  {
23      fixedWalls
24      {
25          type            fixedValue;
26          value           uniform (0 0 0);
27      }
28
29      leftWall
30      {
31          type            fixedValue;
32          value           uniform (0 0 0);
33      }
34
35      rightWall
36      {
37          type            fixedValue;
38          value           uniform (0 0 0);
39      }
40
41      frontAndBack
42      {
43          type            empty;
44      }
45  }
46
47  // ************************************************************************* //
```

A.1.6 controlDict

When all files are modified, it is time to set the time control and data writing. The input data includes time information such as start time, end time and time step, and controls for reading and writing data such as write interval, format and compression. The write interval is the output of information to a file and the number 50 indicates that after 50 time steps a folder is built. This reduces the amount of printed data. Furthermore, during each simulation information is written into a log file where the residuals and Courant number can be found.

```
16  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18  application     scalarTransportFoam;
19
20  startFrom       startTime;
21
22  startTime       0;
23
24  stopAt          endTime;
25
26  endTime         0.1;
27
28  deltaT          0.0001;
29
30  writeControl    timeStep;
31
32  writeInterval   50;
33
34  purgeWrite      0;
35
36  writeFormat     ascii;
37
38  writePrecision  6;
39
40  writeCompression off;
41
42  timeFormat      general;
43
44  timePrecision   6;
45
46  runTimeModifiable true;
47
48
49  // ************************************************************************* //
```

A.1.7 fvSolution

The equation solvers, tolerances, and algorithms are controlled from the `fvSolution` dictionary in the system directory. The solvers defined below are used for each discretized equation.

```
15  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17  solvers
18  {
19      p
20      {
21          solver           PCG;
22          preconditioner   DIC;
23          tolerance        1e-10;
24          relTol           0;
25      }
26
27      U
28      {
29          solver           PBiCG;
30          preconditioner   DILU;
31          tolerance        1e-10;
32          relTol           0;
33      }
34
35      c
36      {
37          solver           BiCG;
38          preconditioner   Cholesky;
39
40          minIter          0;
41          maxIter          1000;
42          tolerance        1e-12;
43          relTol           0.0;
44      }
45  }
46
47  PISO
48  {
49      nCorrectors      2;
50      nNonOrthogonalCorrectors 0;
51      pRefCell         0;
52      pRefValue        0;
53  }
54
55  // ************************************************************************* //
```

PCG/PBiCG/BiCG stands for preconditioned (bi-)conjugate gradient, with PCG for symmetric matrices and PBiCG for asymmetric matrices. The solvers are iterative which means that they are based on reducing the equation residual over successive solutions. The residual is a measure of the error in the solution so that the smaller this becomes, the more accurate the solution becomes. It is also normalized to make it independent of the scale of the problem.

The preconditioners for the conjugate gradient solvers used in the simulations are DIC/DILU which stand for diagonal incomplete-Cholesky (symmetric) and incomplete-LU (asymmetric) respectively.

The tolerance represents the level at which the residual is small enough that the solution can be considered sufficiently accurate. The `relTol` is set to force the solution to convergence and `maxIter` is optional and defines the maximum amount of iteration loops.

The PISO algorithm will be explained in more detail in section A.2.

A.1.8 fvScheme

In this directory all numerical schemes used for the simulation are defined. Gauss linear represents the central difference scheme and backward is the BDF2 scheme for the time derivative.

```
15  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17  ddtSchemes
18  {
19      default          backward;
20  }
21
22  gradSchemes
23  {
24      default          Gauss linear;
25      grad(p)          Gauss linear;
26  }
27
28  divSchemes
29  {
30      default          none;
31      div(phi,U)       Gauss linear;
32      div(phi,T)       Gauss linear;
33  }
34
35  laplacianSchemes
```

```
36  {
37      default          Gauss linear corrected;
38  }
39
40  interpolationSchemes
41  {
42      default          linear;
43  }
44
45  snGradSchemes
46  {
47      default          corrected;
48  }
49
50  fluxRequired
51  {
52      default          no;
53      p;
54  }
55
56  // ************************************************************************* //
```

A.2 Simulation

Before the simulation can be started, some other steps need to be completed first. At first the previous data should be erased otherwise the results may be partially overwritten resulting in a incorrect solution. With cleaning, the parameters such as the mesh, boundary conditions and initial conditions are reset. The next step is to generate the mesh with `blockMesh` and then the simulation is ready to be started. The only thing left is to type in the solver you want to used and let the program solve the equations.

The simulation is run with a solver which is based on the PISO algorithm. The steps the algorithms takes are explained below:

1. Computation starts, initial U and p fields are guessed
2. Starting time loop
3. Boundary conditions are set
4. Linear discretized system is solved $\mathbf{M} \cdot U = -\nabla p$, where $\mathbf{M}$ is the momentum matrix
5. Obtained new U
6. Building matrix $\mathbf{A}$ and vector H
7. Mass flow over cell faces is computed
8. Pressure correction is applied
9. Obtained new p
10. Mass flow over cell faces is corrected
11. Momentum correction is applied
12. Obtained new corrected U on the basis of new pressure field
13. Boundary conditions are updated
14. Steps 6 - 13 are repeated with respect to nCorrectors value
15. Ending time loop

A.3 Post-processing

As soon as results are written to time directories, they can be viewed using ParaView or Matlab. Animations of several parameters can be seen in ParaView and profiles can be extracted from these animations. However, when you want to make some specific plots, it is advanced to use Matlab.

Appendix B

Parameter study $\mathscr{D} \nrightarrow 0$

This parameter study is done to see the behavior of these three parameters on the diffusion coefficient and how they influence the species concentration in each part of the system.

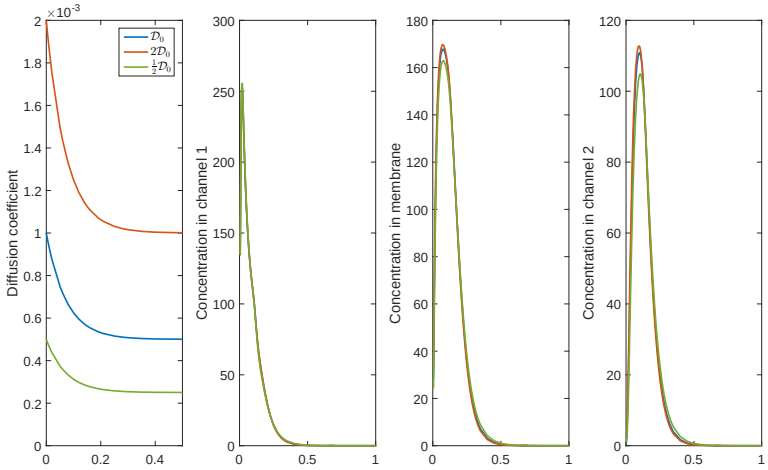
The first parameter is $\mathscr{D}_0$ called the reference value and is halved and doubled. The point where $\mathscr{D}$ reaches the bounding function $\mathscr{D}_1$ stays the same, thus when $\mathscr{D}_0$ is increased the function decreases faster. This can also be seen in the left graph in Figure B.1a. In the membrane and channel two, the concentration is highest corresponding to the highest diffusion coefficient and this is a logical consequence. The higher the diffusion coefficient, the more concentration is able to diffuse through the membrane.

The second parameter is $\mathscr{D}_1$ which is defined by the following ratio

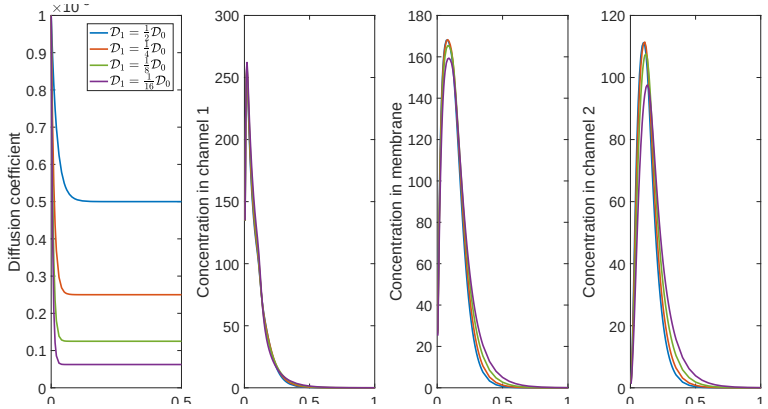
$$\frac{\mathscr{D}_1}{\mathscr{D}_0} = \left[\frac{1}{2}, \frac{1}{4}, \frac{1}{5}, \frac{1}{16}\right]. \tag{B.0.1}$$

This ratio tells for which diffusion coefficient the function is bounded. In the left graph in Figure B.1b the bounded function is lowered, however, τ stays the seem resulting in a steeper decline of the diffusion coefficient. There is a small difference in the concentration in channel one, where it is first higher, the lower and finally higher again. This behavior is the same as in section 4.1. The concentration of species decreases in the membrane and channel two with decreasing bounding function. Due to the fast decreasing diffusion coefficient, less time is available for the concentration diffuse through the membrane before the diffusion coefficient reaches its final value.

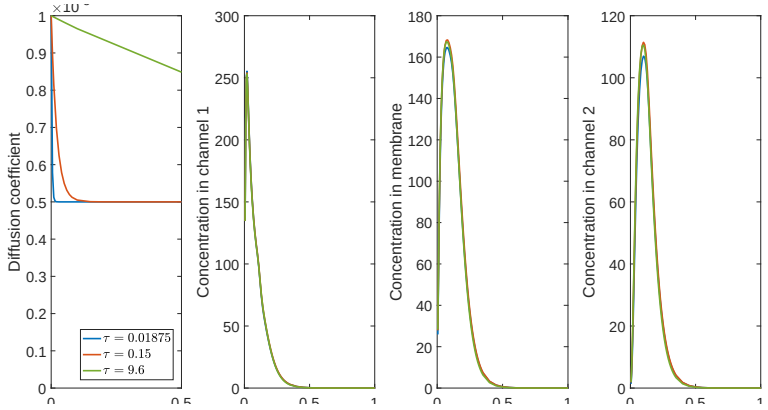
The last parameter which will be studied is τ. The value for τ is made a lot bigger and a lot smaller, which results in the point where $\mathscr{D}$ reaches $\mathscr{D}_1$ moving further away. It seems that when $\tau \to \infty$, the function behaves more and more like the constant diffusion coefficient as can be seen in Figure B.1c. The effect of the parameter on the concentration in the membrane and channel two is very small. Only making τ smaller results into a decrease of concentration due to diffusion.



(a)



(b)



(c)

Figure B.1: Parameter study for $\mathscr{D}_0, \mathscr{D}_1, \tau$.