Chair of Polymer Processing

Master's Thesis

Investigating Hydrogen Diffusion in Filled Polymers with Simple Geometries: An ABAQUS Simulation Study

Alexander Lukas Graf, BSc

May 2024

MONTANUNIVERSITÄT LEOBEN
www.unileoben.ac.at

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, den Einsatz von generativen Methoden und Modellen der künstlichen Intelligenz vollständig und wahrheitsgetreu ausgewiesen habe, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich den Satzungsteil „Gute wissenschaftliche Praxis“ der Montanuniversität Leoben gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 22.04.2024

Unterschrift Verfasser/in
Alexander Lukas Graf

Acknowledgements

I would like to use these lines to express my gratitude to all those who helped me complete this work.

First of all, I would like to thank Assoc. Prof. Dr. Thomas Lucyshyn for supervising and reviewing this thesis. I am very grateful to him for taking the time out of his busy schedule to supervise this work.

I would also like to thank the PCCL for giving me the opportunity to write this thesis. I sincerely appreciate the help of my supervisor, DI Dr. Johannes Macher, who always gave me advice and support. I would also like to take this opportunity to thank DI Dr. Peter Fuchs, MBA for his support.

I would also like to express my heartfelt gratitude to my mother Jutta, my brother Max, my sister Katrin and my brother-in-law Stefan. They have always actively supported me and made it possible for me to get this far on the path I have taken.

Last but not least, I would like to express my gratitude to my girlfriend Selina, who has always been there for me and helped me to get back on my feet even in the most difficult times.

The research work was performed within the COMET module "Polymers4Hydrogen" (project no.: 872165) at the Polymer Competence Center Leoben GmbH (PCCL, Austria), within the framework of the COMET program of the Federal Ministry for Transport, Innovation and Technology and the Federal Ministry for Digital and Economic Affairs. The COMET module is funded by the Austrian Government and the State Government of Styria.

Alexander Graf, B.Sc
May 2024

Abstract

This work was carried out in the context of the module "Polymers 4 Hydrogen" at the Polymer Competence Center Leoben GmbH. The aim of this thesis was to create a Finite Element Method (FEM) model for the simulation of hydrogen diffusion through particle filled membranes with an interface zone around the filler particles. This FEM model was based on an extended Nielsen model and was implemented in ABAQUS. At the end of this thesis, a comparison was made between the interface model, a standard FEM model without interface zone and the analytical solution of the extended Nielsen model.

For all simulations performed in this thesis, it was assumed that the polymer matrix is a homogeneous material and the filler particles act as absolute barriers which were therefore implemented as holes in the matrix. Each filler particle was modelled with a thin interface around its boundary edges that separates the particle from the matrix. In this interface zone, an orientation is applied to the mesh nodes that allows the diffusivity of the interface zone to be changed depending on the direction of flow and the adhesion coefficient. The purpose of this interface zone was to represent the true interfacial diffusion behavior between a filler particle and the matrix material.

Several ways to implement such an interface zone in an FEM model were evaluated. In the final version of the model, the interface zone was implemented at the mesh node level through the use of ABAQUS subroutines. In these models, the filler particles were regularly and periodically arranged in the membrane according to the assumptions of the Nielsen model. The evaluation and comparison of the analytical model results with the results of the FEM simulations with and without the interface zone showed that the FEM simulations with an interface zone were in better agreement with the analytical data than the simulations without the zone.

Kurzfassung

Diese Arbeit wurde im Rahmen des Moduls "Polymers 4 Hydrogen" am Polymer Competence Center Leoben GmbH durchgeführt. Ziel dieser Arbeit war die Erstellung eines FEM-Modells zur Simulation der Wasserstoffdiffusion durch partikelgefüllte Membranen mit einer Interface-Zone um die Partikel. Dieses FEM-Modell basiert auf einem erweiterten Nielsen-Modell und wurde in ABAQUS implementiert. Am Ende dieser Arbeit wurde ein Vergleich zwischen dem Interface-Modell, einem Standard-FEM-Modell ohne Interface-Zone und der analytischen Lösung des erweiterten Nielsen-Modells durchgeführt.

Für alle in dieser Arbeit durchgeführten Simulationen wurde angenommen, dass die Polymermatrix ein homogenes Material ist und die Füllstoffpartikel als absolute Barrieren wirken, welche daher als Löcher in der Matrix implementiert wurden. Jeder Füllstoffpartikel wurde an seinen Rändern mit einer dünnen Interface-Zone modelliert, die das Partikel von der Matrix trennt. In dieser Interface-Zone wird eine Materialorientierung auf die Netzknoten angewendet, die es ermöglicht, die Diffusivität der Interface-Zone in Abhängigkeit von der Flussrichtung und dem Adhäsionskoeffizienten zu bestimmen. Der Zweck dieser Interface-Zone ist es, das tatsächliche Grenzflächendiffusionsverhalten zwischen einem Füllstoffpartikel und dem Matrixmaterial darzustellen.

Es wurden mehrere Möglichkeiten zur Implementierung einer solchen Interface-Zone in ein FEM-Modell untersucht. In der endgültigen Version des Modells wurde die Interface-Zone auf der Netzknotenebene mit Hilfe von ABAQUS-Subroutinen implementiert. In diesen Modellen wurden die Füllstoffpartikel nach den Annahmen des Nielsen-Modells regelmäßig und periodisch in der Membran angeordnet. Die Auswertung und der Vergleich der Ergebnisse des analytischen Modells mit den Ergebnissen der FEM-Simulationen mit und ohne Interface-Zone zeigten, dass die FEM-Simulationen mit Interface-Zone besser mit den analytischen Daten übereinstimmen als die Simulationen ohne Interface-Zone.

Table of contents

Chapter 1

Introduction

1.1 Motivation

Applications in mobility and transport such as hydrogen powered vehicles make light-weight tank solutions necessary, which consist partially or completely of fiber-reinforced polymers [39]. Hydrogen, as a small molecule, easily permeates comparably dense materials. This is even more the case if the molecular structure of the dense material is not latticed, as is the case in polymers. Consequently, a polymer hydrogen tank will lose its content faster than a hydrogen tank made from metal. Therefore a reinforcement is sought to increase the diffusion barrier properties of polymers. [20]

While comparatively simple permeation models are sufficient to describe permeation in homogeneous materials, complex material systems with e.g. voids, different crystal structures, fillers, fibers, different layers, etc. are not fully understood yet and can for this reason only be modeled with simplifications. Therefore, a simulation model which is specifically designed for a more complex material system could help with predicting the diffusion processes around fillers and fibers as barrier materials in polymeric membranes.

The objective of this work is to develop a new numerical model for the simulation of diffusion through fiber-reinforced polymer composites using the Finite Element Method (FEM) solver ABAQUS [42]. The model will be built from three different phases:

- The matrix, the homogeneous polymer volume of the membrane in which diffusion occurs according to the Fickian laws and independently of direction.

- The filler particles, which are embedded in the matrix and are assumed to have ideal barrier properties. Consequently, no mass flow can occur into or out of the filler.
- The interface between the filler and the matrix. The dimension of this interface is one less than that of the entire filler-matrix geometry. This filler-matrix-interface (FMI) does not alter mass flow along its thickness, but the diffusion rate parallel to the outer edge of the filler will be assumed to be dependent on the adhesion coefficient between filler and matrix.

With this model, the barrier properties of the simulated composite can be correlated to the mechanical adhesion between the fillers and the matrix. Therefore, this model should provide a better understanding of diffusion processes in reinforced polymers.

1.2 State of the Art

The fundamentals of permeation and diffusion in polymers have been known for decades as shown by Crank [3], Klopffer et al. [18] and Philibert [36]. The resulting analytical models derived are still in use today and have benefited greatly from modern computing power as shown by Macher et al. [25]. However, with the increasing use of reinforced polymer composites in all areas of engineering, the need for a suitable simulation model for diffusion and permeation has arisen especially in the fields of transportation and energy. For highly simplified composite models, in which each phase is assumed to be homogeneous and all phases are separated from each other, the established models are still usable, as shown by Macher et al. [24] and Monsalve-Bravo et al. [29]. However, for more advanced and detailed simulations of filler or fiber-reinforced composites, these models must be revised. This is shown by Schultheiss [41] in his work where he used the analytical models based on Fick's laws and extended them so that they could better describe the properties of an interface layer. He uses a custom made finite difference method (FDM) solver to simulate the barrier properties of a three-phase fiber-reinforced polymer model.

Duncan et al. [5] give an overview about the basics of diffusion and permeation processes of hydrogen in polymers and describe how to model these processes. They also roughly deal with the topic of FEM simulation of diffusion processes. Furthermore, Alhijazi et al. [1] wrote a review about the possibilities of FEM analysis for natural fiber composites.

There are a few papers in the literature that use ABAQUS to simulate hydrogen diffusion through metals. Most of them use ABAQUS to simulate the mechanical weakening effects

of hydrogen diffusion through metals. These works used the heat transfer / mass diffusion analogy and included user-defined subroutines to couple the diffusion process with the weakening of the material, e.g. [2, 4, 17, 31]. Olden et al. [32] simulated the diffusion of hydrogen through duplex stainless steels. Their model was built on two different phases (austenite and ferrite) and was used in three different setups: fine, coarse, and elongated phases. In addition, the effects of strain and stress near an embedded flaw on the hydrogen concentration were evaluated using a user-defined subroutine. Similarly, in the work of Zhang et al. [47], the effects of tensile and compressive stresses on the diffusion of hydrogen through steel pipes were simulated. A multi-phase ABAQUS simulation was performed by Pu et al. [37], on the topic of stress-assisted diffusion process along grain boundaries and the mechanical response of the grain boundary in a general polycrystalline material. Their model consists of seven different crystalline zones and the grain boundary interfaces between them.

In most of the reviewed literature references, the FEM models for the simulation of diffusion are built from only one phase and the composite behavior is added by using experimentally determined parameters, e.g. [7, 8, 13, 19, 34]. In contrast, Gholami et al. [11] used ABAQUS to simulate the hygrothermal degradation of the mechanical properties of fiber reinforced composites by performing a micro-scale analysis on a two-phase model (matrix and fibers). Li et al. [21] and Zhao et al. [48], simulated the diffusion of chloride ions through cement and they used three-phase and five-phase models, respectively. Papathanasiou and Tsiantis [35] dealt with the barrier properties of flake-filled polymers with a two-phase model in OpenFOAM [33].

1.3 Contribution of this Work

The novelty of this work is based on the following points:

1. Few papers have been found in the literature that focus on the use of ABAQUS or other commercial FEM solvers to simulate diffusion processes in reinforced polymers.

2. In almost all of the literature found, simplifications were made in the models used in one way or another, e.g. [7, 8, 11, 13, 19, 34, 35]. Most of the simulation models consist of only one material whose permeation parameters were determined in advance in experiments. Some simulations use a material model with two phases but these phases are depicted as homogeneous and separated from each other. These models are described in more detail in Section 1.2. The simulation model proposed in this work uses a three-phase model, which connects the matrix and the filler with the FMI.

3. The proposed adhesion coefficient could allow more accurate modeling of complex reinforced polymer components. A model with this coefficient was not found in any of the literature reviewed. With this coefficient, it would be possible to translate the real filler matrix adhesion into an FEM model for permeation simulation. This could lead to a better understanding of the barrier properties of reinforced polymers.

Chapter 2

Theory

2.1 Permeation

Permeation is a process in which the equalization of concentration differences takes place without external influences, resulting in an increase in entropy. This physical process arises from the undirected random movements of particles as result of their thermal energy. If there are different particle concentrations in an area, these random particle movements tend to cause particles to move from higher concentrations to lower concentrations with higher probability. This results in a macroscopic mass transport which leads to an equalization of the particle concentrations.

Since all the simulations performed in this thesis are purely two-dimensional, the following equations and coefficients are also assumed to be two-dimensional. The random movements of the particles are mathematically represented by the diffusion coefficient D ($\mathrm{m}^2 \cdot \mathrm{s}^{-1}$). This coefficient comes from the Einstein-Smoluchowski relationship [6, 45] which links D with the mobility of the particles μ ($\mathrm{s} \cdot \mathrm{kg}^{-1}$):

$$D = \mu \cdot k_b \cdot T \tag{2.1}$$

where k_b ($\mathrm{J} \cdot \mathrm{K}^{-1}$) is the Boltzmann constant and T (K) the absolute temperature. Another important factor is the solubility coefficient S ($\mathrm{mol} \cdot \mathrm{m}^{-2} \cdot \mathrm{Pa}^{-1}$) which establishes the relationship between the concentration C ($\mathrm{mol} \cdot \mathrm{m}^{-2}$) and partial pressure p (Pa) of a gas in the polymer membrane:

$$C = S(T,p) \cdot p \tag{2.2}$$

S can be dependent on T, p or both. If the diffusion coefficient D is multiplied with the solubility coefficient S the permeation coefficient P ($\mathrm{mol} \cdot \mathrm{s}^{-1} \cdot \mathrm{Pa}^{-1}$) is obtained:

$$P = D \cdot S \tag{2.3}$$

This coefficient is the link between the kinetic and thermodynamic aspects of diffusion and is a measure of how well a gas can permeate through a solid at a given pressure. [10, 18]

2.2 Sorption Models

In Table 2.1, five established sorption models are listed. The first three are explained in detail below as they are relevant for the diffusion of gases through polymer membranes. The last two models are mainly applied on the diffusion of vapors which tend to condense, and therefore cause swelling in polymers, and are only listed for the sake of completeness.

Table 2.1 Different models of sorption and typical associated interactions. [18]

Sorption model	Main component interactions
Henry	polymer-polymer
Langmuir	polymer-penetrant
Dual mode	combination of Henry and Langmuir models
Flory-Huggins	penetrant-penetrant
BET	combination of Langmuir and Flory-Huggins models

- Henry's law sorption: This is the simplest sorption case where the gas can be assumed to be ideal and the relationship between the concentration and the pressure of the gas can be described as linear. This model works best at low pressures where the polymer-polymer interactions are stronger than the polymer-penetrant or penetrant-penetrant interactions. The solubility coefficient S (Eq. 2.2) is constant for this model. Thereby, the equation for Henry's law can be written as followed: [18]

$$C = S \cdot p \tag{2.4}$$

- Langmuir mode sorption: In this sorption model, a linear relationship between pressure and concentration in the membrane at low pressures is assumed. Above a certain

concentration, this sorption model depicts a saturation of the membrane. Once the saturation is reached, the concentration in the membrane will stay constant, even if the pressure is increased further. The saturation concentration of the membrane is described by C'_H ($\mathrm{mol} \cdot \mathrm{m}^{-2}$) and b_H (Pa^{-1}) describes the gradient of the linear segment at the start of the model.

$$C = \frac{C'_H \cdot b_H \cdot p}{1 + b_H \cdot p} = S(p) \cdot p \tag{2.5}$$

- Dual mode sorption: The dual mode model is a combination of Henry's law model and the Langmuir model. It was developed to describe the sorption of non-reactive gases in glassy polymers. This model considers the presence of two different diffusing molecule types: one trapped and one freely moving. [18]

In this work, the Henry's law sorption model is applied, because it is assumed that the simulations operate in a region in which a linear relationship between gas pressure and concentration is valid. Furthermore, hydrogen can be assumed to behave ideally at such low pressures and the membrane has no trapping sites.

2.3 Basic Diffusion Models

In 1855, Adolf Fick empirically derived the two basic laws of diffusion, the so-called Fick's laws. These were theoretically derived from thermodynamics, and thereby proven, by Albert Einstein at the beginning of the 20th century. [6]

Fick's first law describes the relationship between the diffusive flow density F ($\mathrm{mol} \cdot \mathrm{m}^{-1} \cdot \mathrm{s}^{-1}$) and the gradient of the concentration ∇C ($\mathrm{mol} \cdot \mathrm{m}^{-3}$). In this work, it is assumed that D is independent of C and is therefore not derived. This assumption is reasoned in Section 2.4. This results in the equation for Fick's first law:

$$F = -D \cdot \nabla C \tag{2.6}$$

Fick's second law is described by a partial differential equation relating the change in concentration over time $\frac{\partial C}{\partial t}$ ($\mathrm{mol} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$) to the Laplace operator of the concentration C. This results in the following formula for Fick's second law:

$$\frac{\partial C}{\partial t} = D \cdot \Delta C \tag{2.7}$$

2.3.1 Analytical Solution of Fick's Laws for Diffusion through a Membrane

This section gives a brief overview of the analytical simulation of diffusion. For a detailed description of the mathematical derivation of the equation shown below, see Crank [3] or Macher et al. [25].

At the start of the permeation process the initial state is in effect. There is no concentration of the permeate in the membrane and the diffusion process is just beginning. As soon as the diffusion process has started, it is in the transient state. Here the concentration of the permeate in the membrane increases with time. The permeate is deposited in the membrane, so to speak. The saturation of the increase in concentration of the permeate in the membrane depends on the boundary conditions of the membrane. For example, if the membrane is immersed in the medium. Then each side of the membrane would be exposed to the same level of medium concentration C_{sub}. In this case, saturation of the membrane is reached when C_{sub} is present at every point in the membrane. Another case would be that the membrane acts as a barrier of some kind. Then on one side of the membrane the concentration of the permeate would be high (C_{high}) and on the other side of the membrane the concentration of the permeate would be low (C_{low}). In this case, saturation of the membrane would be reached when a linear gradient of concentration across the membrane is reached. When the membrane has reached such a saturated state, the transient state is over and the steady state is reached. At steady state, the rate of diffusion into the membrane is equal to the rate of diffusion out of the membrane. Therefore, the concentration of the permeate in the membrane does not change in this state. This means that reaching steady state does not mean that the membrane is completely saturated, but that it has reached a saturation state according to the boundary conditions. The initial and boundary conditions shown in Eq. 2.8 allow the evaluation by an analytical method. Only if the concentration at the inlet and outlet surface remains constant, an analytical calculation of the diffusion can be derived.

$$\begin{aligned} C(x,t=0) &= 0 \quad \text{for} \quad 0 < x < L \\ C(x=0,t) &= C_1 \quad \text{for} \quad 0 < t < \infty \\ C(x=L,t) &= C_2 \quad \text{for} \quad 0 < t < \infty \end{aligned} \tag{2.8}$$

If Eq. 2.7 is solved as a Sturm-Liouville problem with the initial and boundary conditions shown in Eq. 2.8 by using eigenfunctions, the result for the concentration is

$$C(x,t) = (C_2 - C_1)\frac{x}{L} + C_1 + \frac{2}{\pi} \cdot \sum_{n=1}^{\infty} [(-1)^n \cdot C_2 - C_1] \cdot \sin(n\pi \frac{x}{L}) \cdot e^{-\frac{n^2\pi^2 Dt}{L^2}} \tag{2.9}$$

In order to calculate the flow out of the membrane as a function of time, the analytical solution of the concentration (Eq. 2.9) is substituted into Fick's first law (Eq. 2.6) with differentiation at $x = L$, so that the result is

$$F(t) = D \cdot \frac{C_1 - C_2}{L} + \frac{2D}{L} \cdot \sum_{n=1}^{\infty} [(-1)^n \cdot C_1 - C_2] \cdot e^{-\frac{n^2\pi^2 Dt}{L^2}} \tag{2.10}$$

If Eq. 2.10 is integrated by t with limits 0 (s) and t (s), the following equation for the cumulative flow density Q ($\mathrm{mol} \cdot \mathrm{m}^{-1}$) out of the membrane for time t is obtained:

$$Q(t) = D \cdot \frac{C_1 - C_2}{L} \cdot t - \frac{(C_1 + 2C_2)L}{6} - \frac{2L}{\pi^2} \cdot \sum_{n=1}^{\infty} \frac{[(-1)^n \cdot C_1 - C_2]}{n^2} \cdot e^{-\frac{n^2\pi^2 Dt}{L^2}} \tag{2.11}$$

In general, it is advantageous for the application of an analytical solution in a numerical calculation that its quantities are changed into a dimensionless form because this reduces rounding errors and simplifies fitting algorithms, as the dimensionless equations are easily scalable. The following equations show an example of such a transformation for the concentration C, the distance x and the time t.

$$\hat{C} = \frac{C}{C_1}, \quad \hat{x} = \frac{x}{L}, \quad \hat{t} = \frac{D \cdot t}{L^2} \tag{2.12}$$

where C_1 is the boundary condition of the concentration at the upstream face of the membrane and L is the total thickness of the membrane. $\hat{C}$, $\hat{x}$ and τ are the respective dimensionless variables.

2.4 Classification of Diffusion Modes for Polymers

Diffusion is a process at the molecular level and therefore both the type of diffusing gas and the structure of the polymer membrane are crucial. As shown in Table 2.2, the glass transition temperature T_g of the polymer membrane and the critical temperature T_c of the gas are of crucial importance.

Table 2.2 General behavior observed for the transport of small molecules in polymers.[16, 26]

T value compared to a characteristic temperature of the system	**Gases with T > T_c** **H_2, He, O, N_2**	**More condensable gases or vapors (T < T_c)** **CO_2, SO_2, NH_3, hydrocarbons**
T > T_g **Rubbery polymers**	Fickian diffusion constant D Henry's mode sorption constant S increases slightly with T P decreases slightly with pressure (hydrostatic pressure effect)	Fickian diffusion D function of C: $D(C)$ Single mode sorption S decreases with T P increases with pressure (plasticization effect)
T < T_g **Glassy polymers**	Dual mode sorption $S(p)$ Free volume diffusion	Dual mode sorption $S(p)$ Non-Fickian and anomalous diffusion

Depending on the ambient temperature T, the following four cases can be distinguished [18, 40]:

- T is higher than T_c and T_g: This case is most easily represented analytically or by computer simulation. Diffusion proceeds strictly according to Fick and it is reasonable to calculate with constants D, S and P values. Henry's mode sorption applies, whereby the pressure and concentration of the gas have a linear relationship at low gas pressures. This model reaches its limits if the compressibility of the gas has a significant influence.

- T is lower than T_c and higher than T_g: This case can still be described by purely using Fick's equations. As a rule, S can still be assumed to be constant. On the other hand, D shows a dependence on C, which means that P can also be assumed to be variable.

- T is higher than T_c and lower than T_g: In this case, Fick's laws can no longer be applied; instead, free volume diffusion methods are used. Furthermore, dual mode diffusion can be assumed in this case. This means that there is a Henry diffusive flux and a Langmuir diffusive flux. The resulting two diffusion coefficients can be combined

into a p-dependent diffusion coefficient. S is also p dependent, but S and D are not interdependent. P can also be assumed to be variable.

- T is lower than T_c and T_g : As in the previous case, all three coefficients are variable. However, this area is absolutely Non-Fickian and the methods of abnormal diffusion are used.

As mentioned earlier in Section 2.2, Henrys' law sorption and Fickian diffusion are assumed for this work. Furthermore, this work focuses on the diffusion of hydrogen gas. Therefore, of the four cases shown in Table 2.2, only the upper left is applicable to this work.

2.5 Diffusion in Particle Filled Polymer Systems

The focus of this work is on particle-filled polymer membranes. Therefore, this section serves as a brief introduction to the theoretical models that have been formulated for this type of multi-component material system. An overview of the structure of such a filler-polymer system is given in Fig. 2.1. The particles shown in this figure are regularly and periodically arranged as described by Nielsen in his work [30]. Furthermore, the distances between the fillers and the rows of fillers are defined by s and d as described by Macher et. al. [24] in their work, which serves as an extension of the model proposed by Nielsen. This extension of the model also allows it to be implemented in numerical simulations. This in turn allows a direct comparison between the analytical and numerical data obtained.

2.5.1 Nielsen Model

Fick's First law (Eq. 2.6) is used to calculate the flow of the permeate. In this work, it is assumed that the sorption behaves according to Henry's law (Eq. 2.4). By combining these two laws, the following equation is obtained:

$$F = -D \cdot S \cdot \frac{dp}{dx} \tag{2.13}$$

with $\frac{dp}{dx}$ ($\mathrm{Pa} \cdot \mathrm{m}^{-1}$) as the pressure gradient along the thickness of the membrane. Furthermore, since this work assumes steady-state behavior for all diffusion processes, the following equation can be derived.

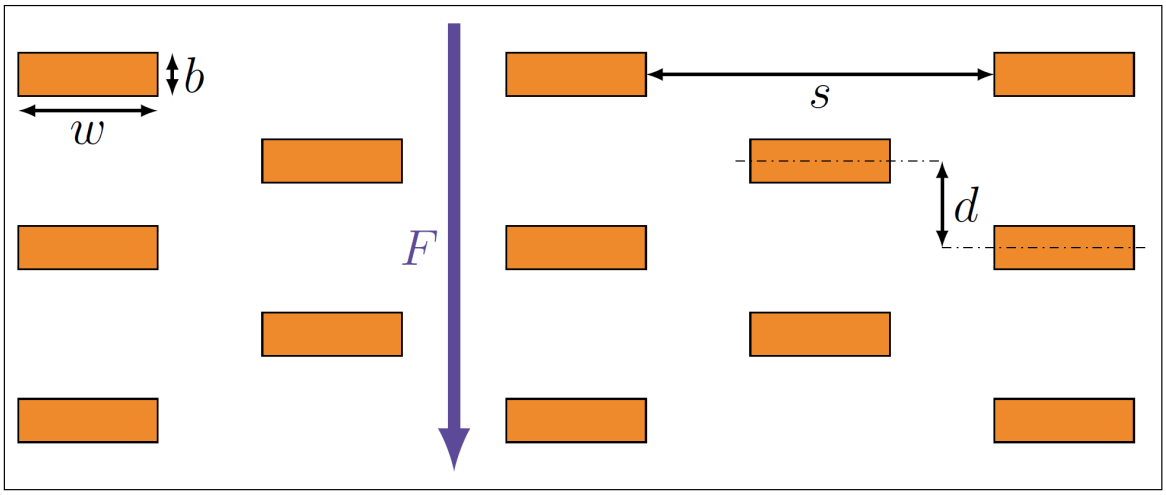


Fig. 2.1 Schematic drawing of a particle-filled polymer membrane, F represents the diffusion flow density and the arrow indicates the direction, w (m) is the width of a filler particle perpendicular to F and b (m) is the thickness of a filler particle parallel to F, s (m) is the slit shape which describes the distance between two filler particles in a row and d (m) is the filler distance which describes the distance from one row of filler particles to another. [24]

$$F = -D \cdot S \cdot \frac{p_1 - p_2}{L} \tag{2.14}$$

Where p_1 (Pa) and p_2 (Pa) are the pressures on the side of the membrane with the high and low concentration of the permeate, respectively, and L (m) is the thickness of the membrane between these two sides. Nielsen described filler particles as impermeable barriers in the membrane [30]. With this assumption, Eq. 2.15 describes the effective length L_{eff} (m) that the permeate has to travel through the membrane. τ (-) is the tortuosity factor that proportionally describes the longer tortuous path of the permeate due to the impermeable filler particle assumption.

$$L_{eff} = \tau \cdot L \tag{2.15}$$

On average, a filler particle in the membrane can be expected to add $\frac{w}{2}$ to the length of L_{eff} [30]. Each diffusing particle encounters on average a number of filler particles $\langle N \rangle$, which is calculated as described in Eq. 2.16. b is the thickness of the filler particles as shown in Fig. 2.1 and ϕ_f is the volume fraction of the filler particles in the membrane.

$$\langle N \rangle = \frac{L}{b} \cdot \phi_f \tag{2.16}$$

With these two assumptions, the effective length of the path of the permeate can be calculated as

$$L_{eff} = L \cdot (1 + \frac{w}{2b} \cdot \phi_f) \tag{2.17}$$

Where w is the width of the filler particle as shown in Fig. 2.1. By comparing Eq. 2.15 and Eq. 2.17, the tortuosity factor of the filled membrane can be described as

$$\tau = 1 + \frac{\alpha}{2} \cdot \phi_f \tag{2.18}$$

with α being the aspect ratio of the filler particles defined as $\frac{w}{b}$.

For convenience, it is suggested not to change the external measurable parameters such as L, p_1 or p_2. Instead, an effective diffusion coefficient D_{eff} can be defined according to Eq. 2.14 and Eq. 2.17:

$$D_{eff} = \frac{D_0}{1 + \frac{\alpha}{2} \cdot \phi_f} \tag{2.19}$$

where D_0 is the diffusion coefficient of an unfilled membrane.

Furthermore, since the filler particles are assumed to be impermeable, they reduce the available volume for permeation. The effective solubility coefficient is calculated in Eq. 2.20, in which S_0 is the solubility coefficient of an unfilled membrane.

$$S_{eff} = S_0 \cdot (1 - \phi_f) \tag{2.20}$$

In essence, the Nielsen model can be written as the ratio of the effective flow F_{eff} through a filled membrane to the flow F_0 through an unfilled membrane of the same external dimensions. By combining this statement with Eqs. 2.14, 2.19 and 2.20, the following equation can be derived.

$$\frac{F_{eff}}{F_0} = \frac{D_{eff} \cdot S_{eff}}{D_0 \cdot S_0} = \frac{P_{eff}}{P_0} = \frac{1 - \phi_f}{1 + \frac{\alpha}{2} \cdot \phi_f} \tag{2.21}$$

2.5.2 Extension of the Nielsen Model

As shown in Eq. 2.21, the Nielsen model depends on only two variables, α and ϕ_f. As mentioned above, this is not enough information about the particle filled membrane to model a numerical simulation with it. An additional specification of the geometric dimensions of the filled membrane is necessary to directly compare Nielsen's model with the results of a numerical simulation. As shown in Fig. 2.1, Macher's extension of Nielsen's model introduces the slit shape s and the filler distance d. With these two additional variables, it is possible to reproduce the same model in a numerical simulation. Furthermore, this figure shows that there are channels with unobstructed flow paths between the filler particles, which was also predicted by Nielsen in his work [30]. This means that the flow of the permeate through the filled membrane consists of two partial flows. One is the unobstructed flow through the channel and the other is the tortuous flow that is obstructed by parts of the fillers. This configuration of the filled membrane can be reduced to a unit cell model shown in Fig. 2.2. There is also a representation of the paths of the two types of flow through the filled membrane. This approach is similar to the one used by Minelli et. al. [27] in their work.

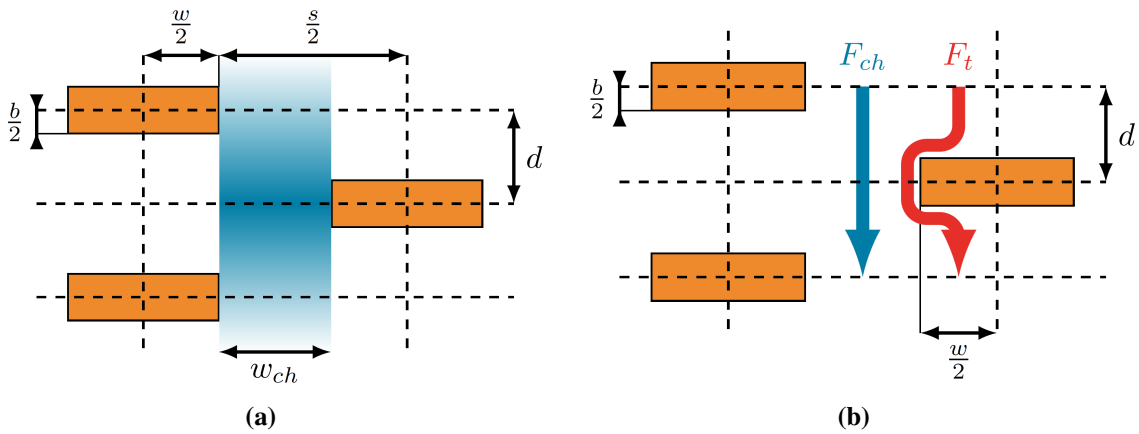


Fig. 2.2 Schematic representation of a unit cell in a filled membrane structured as described by Nielsen [30]. The unit cell is enclosed by the dashed lines. (a) Dimensions of the components in the unit cell. The blue area represents a channel with a width of w_{ch} (m) where the permeate is not hindered in its flow. (b) The two accumulating flows through the unit cell are shown. F_{ch} in blue is the unobstructed flow as described in (a) and F_t in red is the tortuous flow around the filler particles. [24]

The area A_u (m^2) of such a unit cell can then be calculated as shown in Eq. 2.22. Furthermore, the area A_f (m^2) of the filler particle in this unit cell is then defined according to Eq. 2.23.

$$A_u = \frac{w+s}{2} \cdot d \tag{2.22}$$

$$A_f = \frac{w \cdot b}{2} \tag{2.23}$$

Due to the two-dimensionality of this model, ϕ_f (-) is reintroduced as the filler area fraction, which can be calculated as

$$\phi_f = \frac{A_f}{A_u} = \frac{w \cdot b}{d \cdot (w+s)} \tag{2.24}$$

By rearranging Eq. 2.24, the filler distance can then be calculated as

$$d = \frac{w \cdot b}{\phi_f \cdot (w+s)} \tag{2.25}$$

As shown in Eq. 2.25, the filler distance depends on the slit shape s of the filled membrane.

To calculate s, it is necessary to determine the ratio between the flows through a filled and an unfilled membrane. As shown in Fig. 2.2, the flow through the filled membrane consists of the two flows F_{ch} and F_t. Assuming that the channels in the filled membrane are constant along the thickness of the membrane, the flow through them depends only on their width. The width of these channels can be calculated as

$$w_{ch} = \frac{s-w}{2} \tag{2.26}$$

If the width of the channels in the filled membrane is related to the width of the whole unit cell, the channel flow ratio f_{ch} (-) can be calculated as the fraction of the flow through the channel F_{ch} and the flow through the unfilled membrane F_0.

$$f_{ch} = \frac{F_{ch}}{F_0} = \frac{s-w}{w+s} \tag{2.27}$$

To calculate the tortuous flow F_t, it is necessary to calculate the effective length L_t (m) of this flow through the unit cell as shown in Eq. 2.28. The solution of the integral in this equation results in the average additional path length x (m) caused by the filler particle in the unit cell.

$$L_t = d + \frac{\int_0^{\frac{w}{2}} x\,dx}{\frac{w}{2}} = d + \frac{w}{4} \tag{2.28}$$

By combining Eq. 2.15 with Eq. 2.28, the following equation can be derived for the tortuosity of the effective path length.

$$\tau_t = 1 + \frac{w}{4d} \tag{2.29}$$

Before the tortuous ratio f_t (-) of the flow through the filled membrane can be calculated, another assumption must be made. As shown in Fig. 2.2, the tortuous flow is not calculated per unit cell, but per two unit cells connected in the direction of flow. With this arrangement of two unit cells it is possible to calculate the tortuous flow ratio with only one equation:

$$f_t = \frac{F_t}{F_0} = \frac{1}{2\tau} \cdot \frac{w}{w+s} \tag{2.30}$$

Combining Eq. 2.30 with Eqs. 2.25 and 2.29 results in the final equation for the tortuous ratio of flow through the filled membrane as

$$f_t = \frac{2w \cdot b}{\phi_f \cdot (w+s)^2 + 4b \cdot (w+s)} \tag{2.31}$$

The sum of f_{ch} and f_t is equal to the relative flow through the filled membrane $\frac{F_{eff}}{F_0}$ and can be substituted into Eq. 2.21. This results in

$$\frac{F_{eff}}{F_0} = \frac{1-\phi_f}{1+\frac{\alpha}{2} \cdot \phi_f} = f_{ch} + f_t \tag{2.32}$$

Substituting Eqs. 2.27 and 2.31 in Eq. 2.32 defines the following equation for the slit shape.

$$s = w \cdot \left[-\frac{2\alpha \cdot \phi_f + 4}{\phi_f \cdot \alpha \cdot (\alpha+2)} + \frac{\sqrt{\alpha^4 \cdot \phi_f^2 + 6\alpha^3 \cdot \phi_f + 12\alpha^2 + 24\alpha + 16}}{\phi_f \cdot \alpha \cdot (\alpha+2)} \right] \tag{2.33}$$

With Eqs. 2.25 and 2.33 in place, the missing parameters s and d of the Nielsen model are determined and the now extended model can be compared to numerical simulations. It is important that one of the following two statements in Eq. 2.34 is true so that there is no collision between filler particles.

$$\begin{gathered} s > w \\ \text{or} \\ d > b \end{gathered} \tag{2.34}$$

2.6 Interface Layer

Every multi-component material contains interfacial layers formed by chemical and physical processes. These processes include interdiffusion of atoms or molecules, immobilization, crystallization of thermoplastics, and crosslinking. The interfacial layer is a thin transition zone between the matrix and a filler, with a thickness in the nanometer range. The structural, physical, chemical and mechanical properties of the interfacial layer may differ from those of the matrix and filler, depending on the type of filler and matrix and the physical and chemical processes involved. The existence of this interfacial layer has been proven by Fourier transform infrared (FTIR) spectroscopy [12]. Each interfacial zone has different properties due to the many influences acting on it. In general, the interface is considered the weakest part of a multi-component material, but it also provides the opportunity to tailor the material to provide the required properties [16].

Because of their variability, but also because of the numerous factors that influence their properties, there have been many studies on interface layers. A lot of research has been done on carbon fibers due to their popularity in the industry, (e.g. [9, 22, 23, 46, 49]). But there is also literature for various other material combinations such as polymer-natural fiber, polymer blends, nanocomposites, etc. [14, 28, 38, 43, 50].

Chapter 3

Simulation

For this work, several simulations were carried out for three different simulation models. The specifyers of these models are:

- Membrane with a circular filler particle, shown in Fig. 3.1a
- Membrane with a rectangular filler particle, shown in Fig. 3.1b
- Membrane with multiple rectangular filler particles, shown in Fig. 3.1c

The first two models, each with just a single filler particle in the center, were primarily used to enable the development of the interface zone simulation technique. This technique is described in more detail in Section 3.1. These simple models were also used to gather data for parameters such as interface zone thickness, interface zone mesh size, et cetera. These parameters were necessary for the successful implementation of the interface zone in the more complex third model. Several tests were carried out with this model using different filler particle aspect ratios and filler particle area ratios. These were then compared with the results of the same simulations without the interface zone and the results of the extended Nielsen model, which was described in Section 2.5.2.

Each model in this work consists of a two-dimensional membrane containing one or more two-dimensional filler particles. The filler particles are either circular or rectangular in shape and have an interface zone and a mesh transition zone around them. A simplified schematic of such a membrane with a rectangular filler particle in the center is shown in Fig. 3.2. The purpose of the interface zone is to create a thin layer around the filler particle that increases or decreases the diffusion rate of the membrane along the filler edge. Therefore, the diffusion rate normal to the filler particle edge remains unchanged from the diffusion

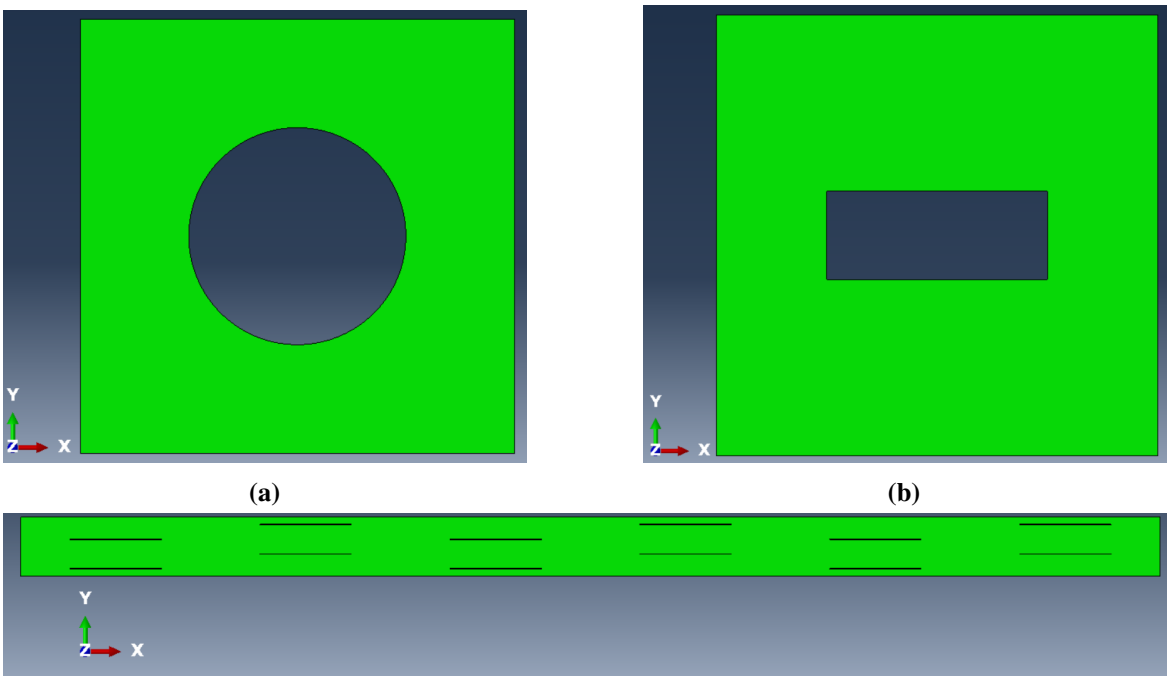


Fig. 3.1 Geometries of membranes with a) a circular, b) a rectangular and c) multiple rectangular filler particles. The inlets of these membranes are always at the top edge, the outlets are always at the bottom edge, and the left and right edges of the membranes are connected by a periodic boundary condition.

rate of the membrane and only along the edges of the filler a change is applied. The mesh transition zone is only used to provide a more gradual transition of the mesh for a stable mesh generation process.

3.1 Interface Zone

The core of this work is the implementation of the interface zone. Therefore, the process leading to the final configuration of the interface zone is discussed in detail in this section. As described in Section 2.6, a real membrane, reinforced with filler particles would show a change in the properties of the membrane in the transition area from filler to membrane. This effect occurs most strongly directly at the edge of the filler and bleeds into the membrane material. Since the significant portion of the effect occurs directly at the filler edge, it was initially decided to define the interface zone as a one-dimension lower element, in comparison to the rest of the model. That is, if the geometry of the membrane and the filler particle are assumed to be two-dimensional, the interface zone would have had to be implemented as a

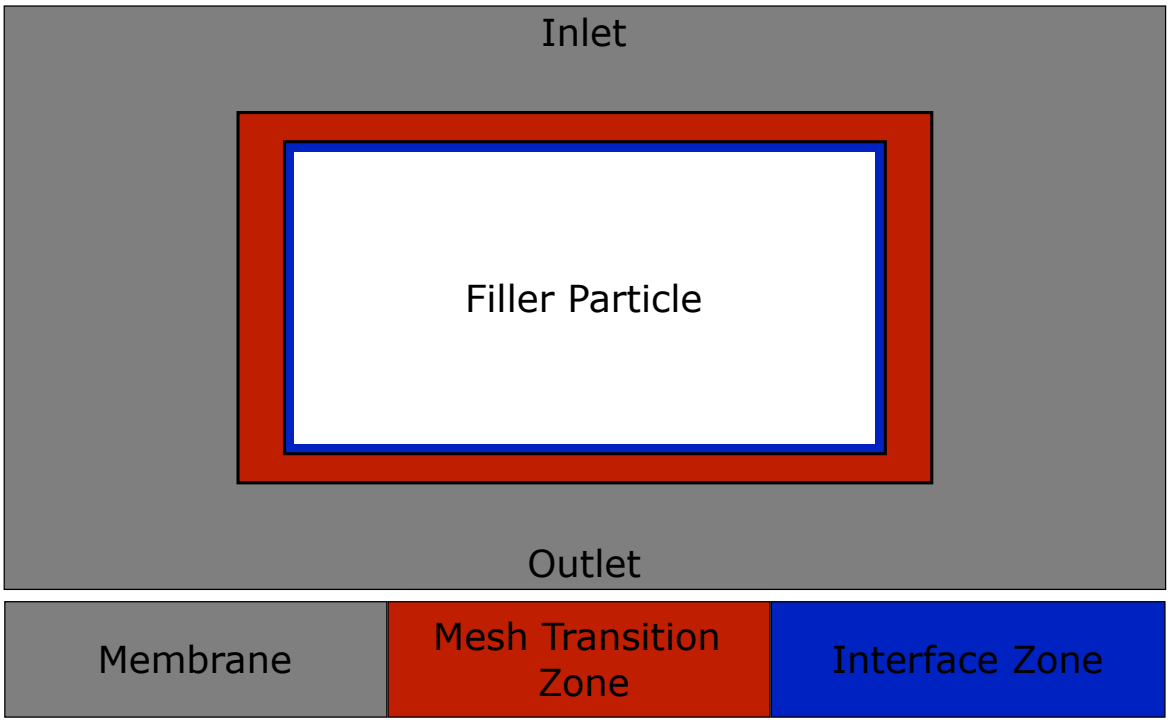


Fig. 3.2 Simple overview of the membrane with a rectangular filler particle, the mesh transition zone and the interface zone.

one-dimensional element. This would have allowed for the simplest possible model setup and kept the simulation time low. In ABAQUS, however, it is not possible to implement a one-dimensional element, i.e. an edge, with its own material properties in a two-dimensional diffusion analysis. There are so-called STRINGER elements in ABAQUS which allow to assign a material to an edge, but these also require the specification of a cross-sectional area for this edge. This means that the one-dimensional behavior is only superficial, and behind a black box the simulation operates in two dimensions. This was unacceptable because it would lead to results that were difficult or impossible to evaluate. Therefore, it was decided to implement a custom two-dimensional solution that could best represent the real behavior of such an interface zone. This process would eliminate the black box and allow more control over the simulation process.

The first approach in this variant of implementation was to partition the membrane in such a way that a separate area is created around the filler particle. This area was then assigned its own direction-dependent material. Furthermore, the local orientation of the coordinate system in this area was rotated in such a way that the above-described effect of a changed flow around the filler was created. If the thickness of this area was small enough, the results of these simulations were a step in the right direction. However, due to the sharp transition

of the diffusion coefficients at the boundaries of the interface zone, undesirable edge effects occurred which significantly degraded the simulation quality.

In order to keep the interface zone as thin as possible and still ensure that the materials show a gradual transition, it was decided to change the material properties at the mesh node level. For this purpose it was necessary to integrate ABAQUS subroutines into the simulation. ABAQUS provides more than 100 different subroutines for different applications. Each of these subroutines has access to different systems or data from ABAQUS and has its own methods to interact with ABAQUS. To achieve the desired effect, three different subroutines, ORIENT, UFIELD, and USDFLD, were programmed and implemented in Fortran [15] as described in Section 3.2.

In the first iteration the handling of these subroutines was mostly unknown so the subroutines were powered by a Python [44] script which calculated the necessary field data. In this Python script, the positions and sizes of the filler particles as well as the positions of the mesh nodes were read from the ABAQUS input file. Then the distance of the nodes to the filler particles was calculated and if their distance to the particle edge was below a threshold value, they were assigned a scalar value representing the relative position between the filler particle edge and the threshold value. Furthermore, in the script the interface zone was divided into the areas shown in Fig. 3.3 and for each node the corresponding area was evaluated. This enabled the calculation of the necessary rotation of the local coordinate system for each node. The values for distance and rotation calculated in this script were then written to CSV files. Of the three subroutines used, UFIELD and USDFLD were responsible for adjusting the diffusion coefficients and ORIENT for rotating the local coordinate systems of the nodes. For this purpose, the relevant CSV file was read in each case and the correct value was imported and applied. This procedure had the disadvantage that a lot of file calls had to be made which increased the simulation time significantly. For this reason, the membrane was first divided into two materials, the interface material and the membrane material. The interface material is linked to the subroutines, i.e. the subroutines are only called for mesh nodes and mesh elements for which this material has been assigned. The membrane material is a homogeneous material with constant values that is assigned everywhere except in the interface zone. This minimized the number of subroutine calls. However, simulations with many filler particles still resulted in long simulation times due to the mesh density in the interface zone. Therefore, the Python script tasks were shifted to the subroutines, reducing the number of file calls to one per type of subroutine. The exact structure and details of the functions of the final subroutines will be discussed in the next section.

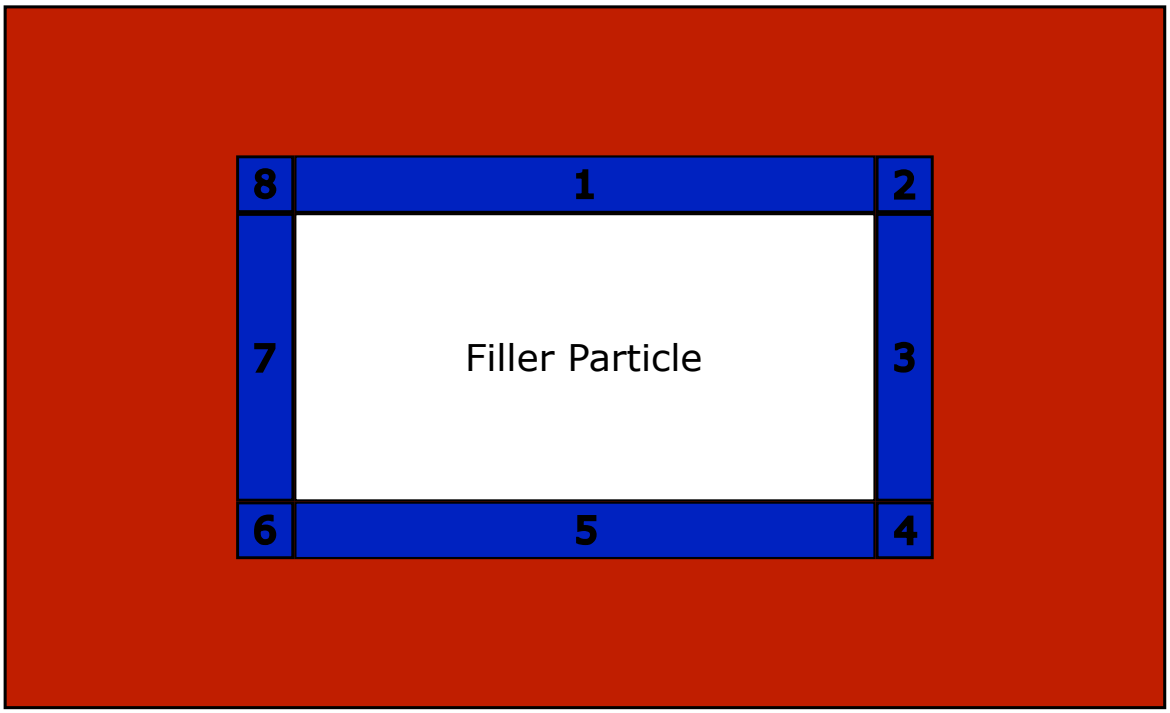


Fig. 3.3 Simple sketch of the interfacial zone and its partitions around a rectangular filler particle. Partitions 1 and 5 change the diffusive flow along the vertical axis, partitions 3 and 7 along the horizontal axis, and partitions 2, 4, 6, and 8 always at a 45 degree angle between the two neighboring partitions.

3.2 Fortran Subroutines

The subroutine ORIENT is responsible for rotating the local coordinate system of each mesh node to achive the desired directional behavior of the interface zone. This means that during the simulation this subroutine is called for each mesh node in the interface zone. Two different sequences exist for this subroutine, depending on whether the filler particle is circular or rectangular. For the variant used for rectangular fillers, an example is given in Algorithm 1 in the form of pseudocode. The code for a filler particle with a circular shape differs in that it is not necessary to divide the interface zone into separate orientation partitions since the direction vector from the filler center to the node can be calculated and the local coordinate system can be rotated based on it. The final result of this subroutine is shown in Fig. 3.4. By comparing the circular and rectangular filler particles there, it is easy to see that in the case of the circular filler particle, there is a continuous change of orientation along the edge of the filler, while in the case of the rectangular filler particle, different zones are clearly visible. The actual code used for the subroutine is provided in the appendix in Section A.1.1.

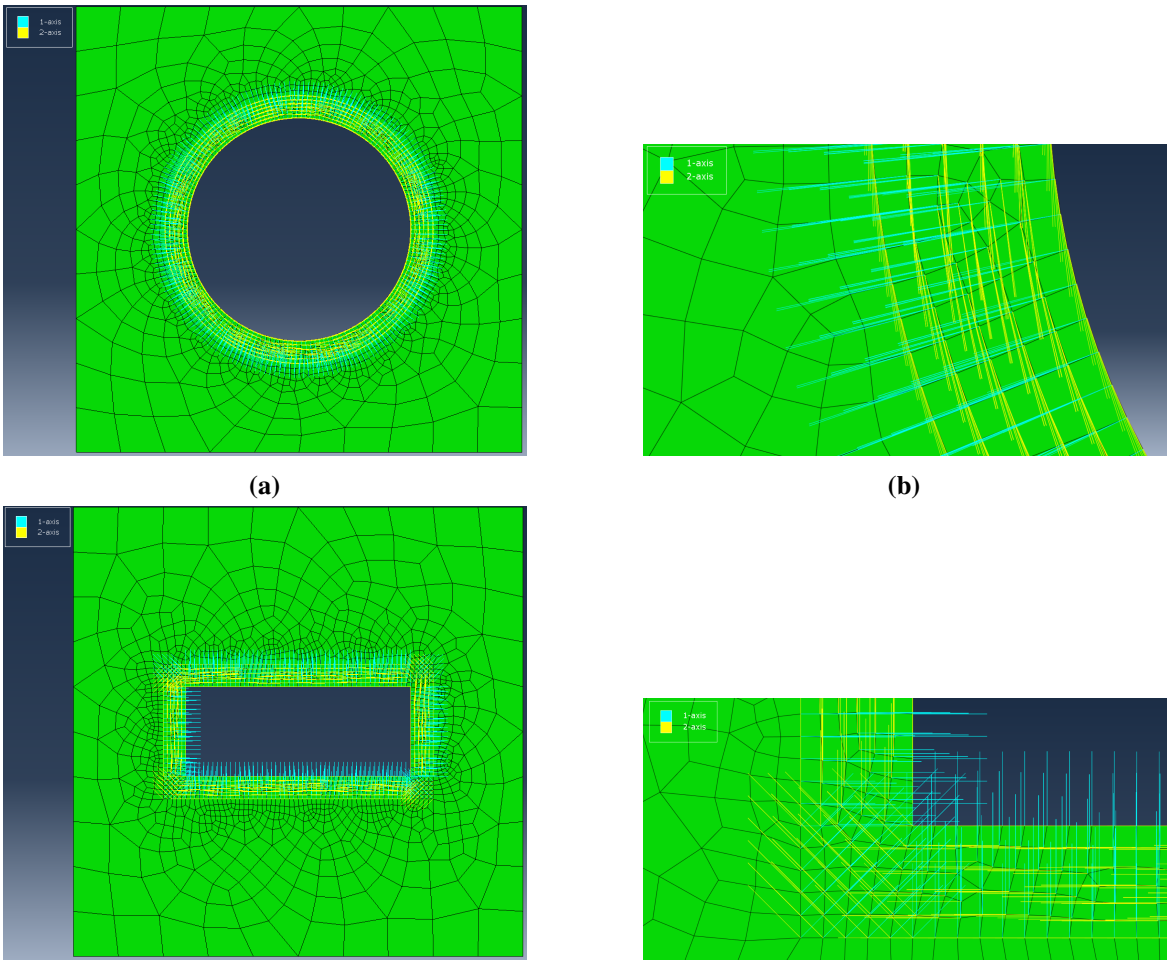
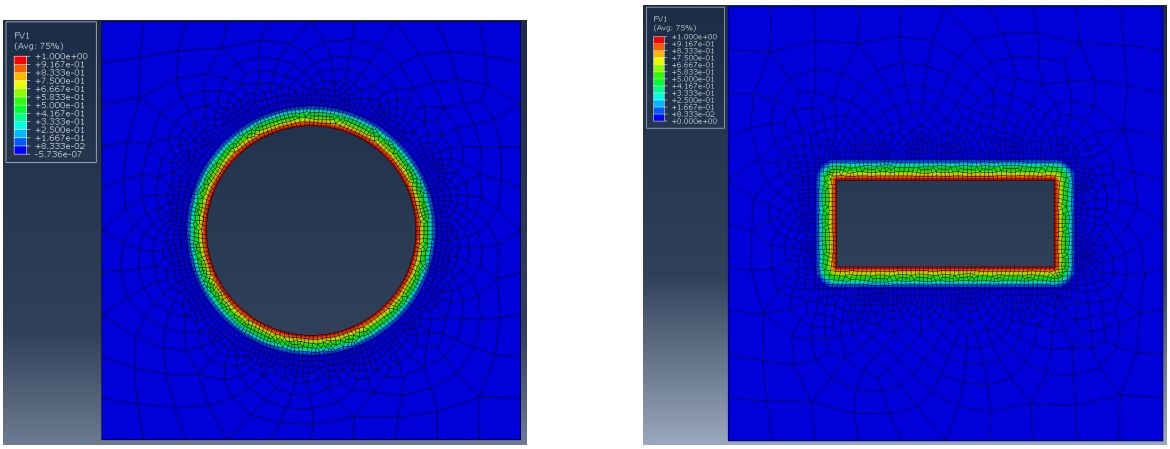


Fig. 3.4 Oriention of the loacal node coordinate system in the interface zone for a circular filler particle: (a) complete membrane and (b) zoomed view. Oriention of the loacal node coordinate system in the interface zone for a rectangular filler particle: (c) complete membrane and (d) zoomed view. In this work, the 2-axis shown in yellow is the one with the variable diffusion rate.

The subroutine UFIELD is responsible for generating a field over the interface zone around the filler particles. This field contains a scalar value for each node in the interface, which indicates the relative position of the node with respect to the thickness of the interface zone. This means that if a node is located directly on the edge of a filler particle, the value of this field for this node is equal to one. On the other hand, if a node is located directly on the outer edge of the interface zone, the field has a value of zero for this node. An example of this subroutine is given in Algorithm 2 in the form of pseudocode. The final result of this subroutine can be seen in Fig. 3.5. The actual code used for the subroutine is provided in the appendix in Section A.1.2.

Algorithm 1: Pseudocode example for an ABAQUS ORIENT subroutine.

```
SUBROUTINE ORIENT
    /* The data of the node for wich this subroutine is called. */
    Input: nodeData;
    /* Create a global list, which can be called from every
       subroutine call, in which the data of all the filler
       particles is stored. */
    Create Global List: fillerPartikelList = EMPTY;
    /* Create variables to hold numerical data. */
    Create Variable: minDistanceToFillerParticle = INFINITY;
    /* Create variables to hold numerical data. */
    Create Variable: idClosestFillerParticle = NULL;
    /* If the filler particle list is empty, load the necessary
       data from a csv file.  This is only necessary for the first
       time this subroutine is called. */
    if fillerPartikelList == Empty then
        load fillerParticleData into fillerPartikelList;
    /* Search for the closest filler particle to the node and save
       its id. */
    for fillerParticle in fillerPartikelList do
        calculate distance from node.position to fillerParticle;
        if distance < minDistanceToFillerParticle then
            minDistanceToFillerParticle = distance;
            idClosestFillerParticle = fillerParticle.id;
    /* Evaluate the orientation zone id for the current node based
       on its position. */
    calculate orientationZone;
    /* Apply a rotation to the local coordinatesystem of the node
       based on the orientation zone. */
    if orientationZone == 1 then
        rotate localCoordinateSystem LEFT 90 DEGREES;
    else if orientationZone == 2 or 6 then
        rotate localCoordinateSystem LEFT 45 DEGREES;
    else if orientationZone == 3 or 7 then
        Pass;
    else if orientationZone == 4 or 8 then
        rotate localCoordinateSystem RIGHT 45 DEGREES;
    else if orientationZone == 5 then
        rotate localCoordinateSystem RIGHT 90 DEGREES;
```



(a) **(b)**

Fig. 3.5 Diffusion coefficient field in the interface zone for (a) a circular filler particle and (b) a rectangular filler particle.

The subroutine USDFLD is significantly different from the two previously described. This subroutine is called for all mesh elements in the interface zone and not for the mesh nodes. This is because this subroutine is integrated in the material behavior and thus directly accesses the integration points of the elements. It follows that the task of this subroutine in this work is to interpolate the values of the field generated by the UFIELD subroutine to the integration points of the mesh elements. Since this is the basic function of this subroutine, it was implemented without any additional code. It was not possible to output the result of this subroutine graphically by ABAQUS, because it is only a subfunction and not an independent field. The actual code used for the subroutine is provided in the appendix in Section A.1.3.

The field generated by the USDFLD subroutine is then used by the ABAQUS material itself to assign a diffusion coefficient to the integration points based on this exponential function:

$$D_{IP}(x) = D_{Imax} \cdot e^{\Psi \cdot (x-1)} + D_M \cdot (1 - e^{\Psi \cdot (x-1)}) \tag{3.1}$$

In this equation, D_{IP} is the diffusivity coefficient in the interfacial zone for a flow parallel to the edge of the filler as a function of the distance to the outer edge of the interface zone x, D_{Imax} is the maximum of the diffusivity in the interfacial zone, and D_M is the diffusivity in the matrix. x is a control variable that represents different relative distances from the outer edge of the interface at which the diffusivity is calculated. This is necessary because in ABAQUS the material data can only be provided as a table and not as a function. In order to generate the desired course of the diffusion rate change, a prefactor Ψ was introduced. This type of function was chosen because it allows for a large gradient without creating jumps

Algorithm 2: Pseudocode example for an ABAQUS UFIELD subroutine.

```
SUBROUTINE UFIELD
    /* The data of the node for wich this subroutine is called. */
    Input: nodeData;
    /* Create a global list, which can be called from every
       subroutine call, in which the data of all the filler
       particles is stored.                                     */
    Create Global List: fillerPartikelList = EMPTY;
    /* Create a variable to hold numerical data.                */
    Create Variable: minDistanceToFillerParticle = INFINITY;
    /* If the filler particle list is empty, load the necessary
       data from a csv file.  This is only necessary for the first
       time this subroutine is called.                          */
    if fillerPaticelList == Empty then
        load fillerParticleData into fillerPartikelList;
    /* Search for the distance of the closest filler particle to
       the node.                                                */
    for fillerParticle in fillerPartikelList do
        calculate distance from node.position to fillerParticle;
        if distance < minDistanceToFillerParticle then
            minDistanceToFillerParticle = distance;
    /* Calculate the relative position of the node in relation to
       the thickness of the interface zone.                     */
    calculate relativePositionValue;
    /* Save the relative position as a field variable.          */
    Save relativePositionValue in FIELD;
```

in the curve. Various experiments were then carried out to find a Ψ for which the curve has no jumps at the beginning, a large rate of change while still maintaining the bleeding effect, which was explained in Section 3.1. Such curves are shown in Fig. 3.6. After evaluating these experiments, it was decided to use a Ψ of 20 for the simulations in this thesis.

Furthermore, for future projects, the adhesion coefficient mentioned in the introduction can be represented by the two variables D_{Imax} and Ψ.

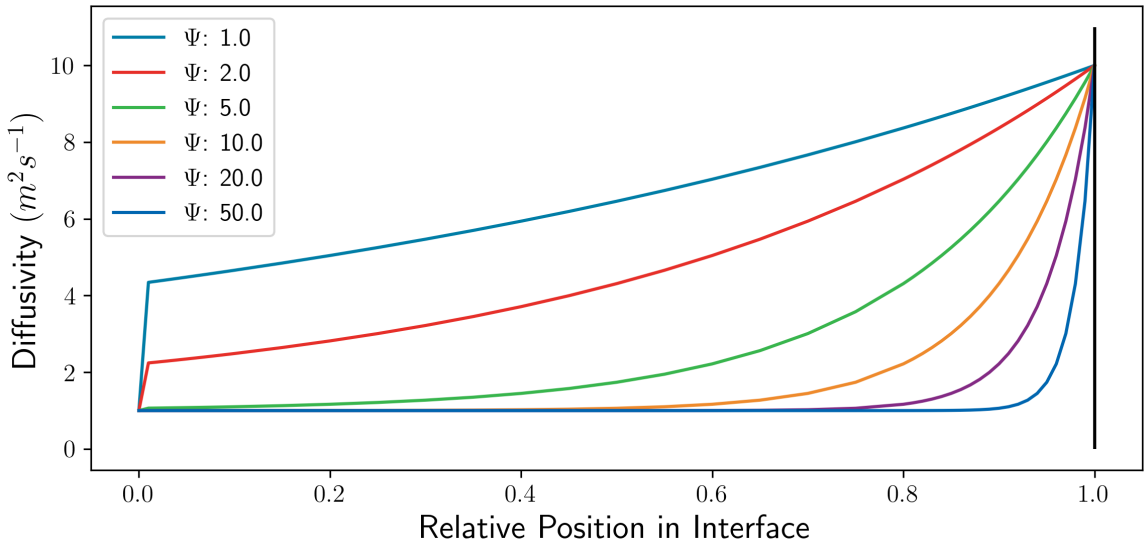


Fig. 3.6 Curve shapes of diffusion rates in the interface zone at different prefactors Ψ. A value of 0 on the x-axis represents the position at the outer edge of the interface zone and a value of 1 represents the position at the outer edge of the filler particle.

3.3 Simulation Workflow

3.3.1 Overview

Since the used simulation models differ only in geometry and the resulting relevant code differences have already been explained in Section 3.1, the proposed simulation workflow is generally valid for this work. First of all, each simulation has the following basic configuration:

- The inlet of the membrane is at the top edge. There the concentration of the permeate is constant at a value of $1\,\mathrm{mol} \cdot \mathrm{m}^{-2}$.
- The outlet of the membrane is at the bottom edge. There the concentration of the permeate is constant at a value of $0\,\mathrm{mol} \cdot \mathrm{m}^{-2}$.
- The initial concentration of the permeate in the membrane is set to $0\,\mathrm{mol} \cdot \mathrm{m}^{-2}$.
- The left and right edges of the membrane are connected with a periodic boundary condition.
- The filler particle is always implemented as a hole in the membrane. The permeate can therefore not penetrate the particle.

- Every filler particle has an interface zone around it, as described in Section 3.1.

The general workflow of the simulations used in this work consists of multiple steps, which are controlled by one central script. First of all, the following simulation parameters are defined in this central script:

- Filler particle thickness, b
- Filler particle aspect ratio, α
- Filler particle area ratio, ϕ_f
- Number of filler particles in the membrane
- Mesh seeding sizes of the membrane, mesh transition zone and the interface zone
- Inlet and outlet concentration of the diffusion medium, C_1 and C_2, respectively
- Diffusion coefficients for both materials, D_M for the matrix material, D_{IP} and D_{IN} for the interface material
- Solubility coefficient for both materials, S_M for the matrix and S_I for the interface

Based on these simulation parameters, the positions of all fillers in the membrane are then calculated, inserted into a list and saved together with the simulation parameters in CSV files. These files are then automatically read in during the simulation steps in which they are required. Next, the following steps are performed:

1. Generation of the ABAQUS input file based on the parameters from the CSV file.
2. Running the ABAQUS interface simulation, with the subroutines.
3. Extracting data out of the ABAQUS ODB file.
4. Running the ABAQUS standard simulation.

These steps are explained in detail in the following subsections. At the end of a simulation series, the result files were read in again with the help of a subsequent script, the data contained therein was transformed into the desired form and plotted in various plots. The actual code used in this Python script is provided in the appendix in Section A.2.4.

3.3.2 Creation of Input File

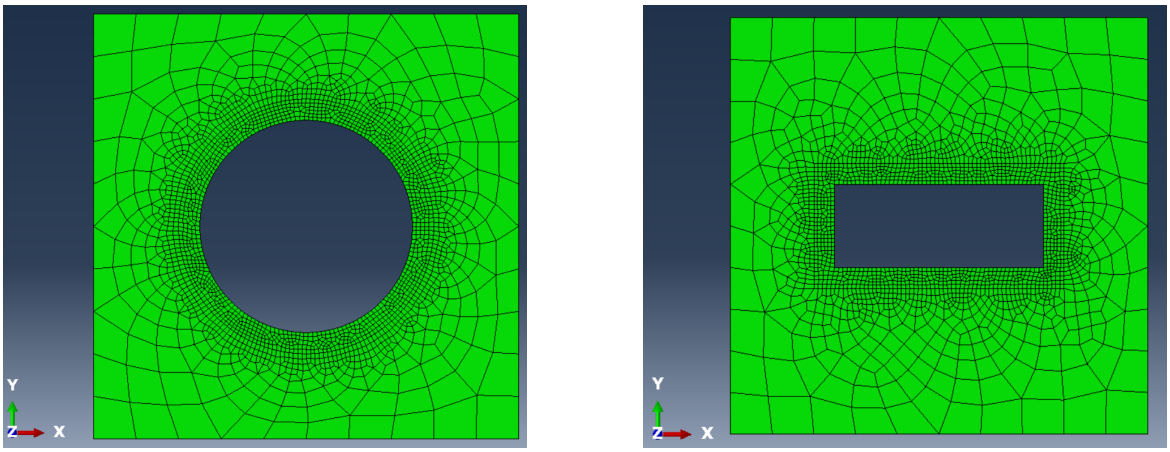
First, the ABAQUS input file is created on the basis of the simulation parameters defined in the input. For this purpose ABAQUS is automated by a Python script which runs the following sequence:

1. Geometry creation: The membrane is drawn as a rectangle. The filler particles are simultaneously cut into this rectangle as holes. The area around these holes is then partitioned twice. Once at a very small distance from the perimeter of the hole and once at ten times that distance.
2. Material creation: The interface and membrane materials are created and assigned to the corresponding areas of the model.
3. Mesh creation: The membrane mesh is kept coarse, while the edges of the filler particles are seeded with a finer mesh size. The outer edges of the interface zone are also seeded with the same fine mesh size. The outer edges of the mesh transition zone are seeded with an intermediate mesh size between the size of the membrane and the interface zone mesh. This results in a mesh configuration as shown in Fig. 3.7. This mesh configuration is necessary for the above mentioned subroutines to work with sufficient accuracy.
4. Field creation: The ABAQUS field for the UFIELD Subroutine is created and assigned to the interface zone.
5. Set creation: The necessary ABAQUS node sets on the left, right and bottom edge are created. The sets on the left and right edge are necessary for the periodic boundary condition which connects these two edges. The set on the bottom edge is necessary for collecting the output data.
6. Time step creation: A time step is the framework for the boundary conditions. That means in this step the inlet and outlet boundary conditions as well as the periodic boundary are defined.

The actual code used in the Python script is provided in the appendix in Section A.2.1.

3.3.3 ABAQUS Interface Simulation

After the ABAQUS input file has been created, the simulation is started by a batch file. This batch file is needed to load the input file into the simulations and to connect them with the



(a) (b)

Fig. 3.7 Mesh of membranes with (a) a circular filler particle and (b) a rectangular filler particle.

subroutines. When the simulation is finished, ABAQUS is started again automated by a Python script and the ODB file of the simulation is opened. Afterwards, the script extracts all relevant data and saves them to a CSV file. This Python script is provided in the appendix in Section A.2.2.

3.3.4 ABAQUS Standard Simulation

For the standard simulation, another simulation is performed with the previously defined simulation parameters, but without an interface zone material in the model. However, the mesh is generated in the same way as for the simulation with an interface zone to avoid a significant difference in the results. Since no subroutines are required for this simulation, all three steps, i.e. model creation, simulation and data extraction, can be automated using a single Python script. This simulation is necessary to compare the results of the interface simulations. The actual code used in the Python script is provided in the appendix in Section A.2.3.

Chapter 4

Results

This chapter presents the results of the simulations performed in the course of this thesis. Two types of plots have been chosen to highlight different aspects of the results. First, the raw results of the different simulations are discussed to show the influence of the different parameters on the resulting outflow over the width of the membrane. Then, the outflow values are integrated over the individual membranes to obtain the total outflow for a given membrane. These integrated values are better suited for comparison with one another. Furthermore, some of the results obtained from ABAQUS are shown directly as examples.

4.1 Raw Results - One Filler Particle

In Fig. 4.1 the permeate concentration is shown for the simulations with only one central filler particle in the membrane, as provided by ABAQUS. Comparing Fig. 4.1a with Fig. 4.1b and Fig. 4.1c with Fig. 4.1d it can be observed that the implementation of an interface zone leads to a better distribution of the permeate around the filler. It can be clearly seen that the interface zone increases the flow of the medium through the membrane.

In Fig. 4.2, the raw results of simulations with a circular and a rectangular filler particle are shown. These simulations were carried out with the following parameters:

- Diffusivity of the matrix material, $D_M = 1.0\,\mathrm{m}^2 \cdot \mathrm{s}^{-1}$
- Diffusivity of the interface material parallel to the filler particle edges, $D_{IP} = 10.0\,\mathrm{m}^2 \cdot \mathrm{s}^{-1}$
- Diffusivity of the interface material normal to the filler particle edges, $D_{IN} = 1.0\,\mathrm{m}^2 \cdot \mathrm{s}^{-1}$

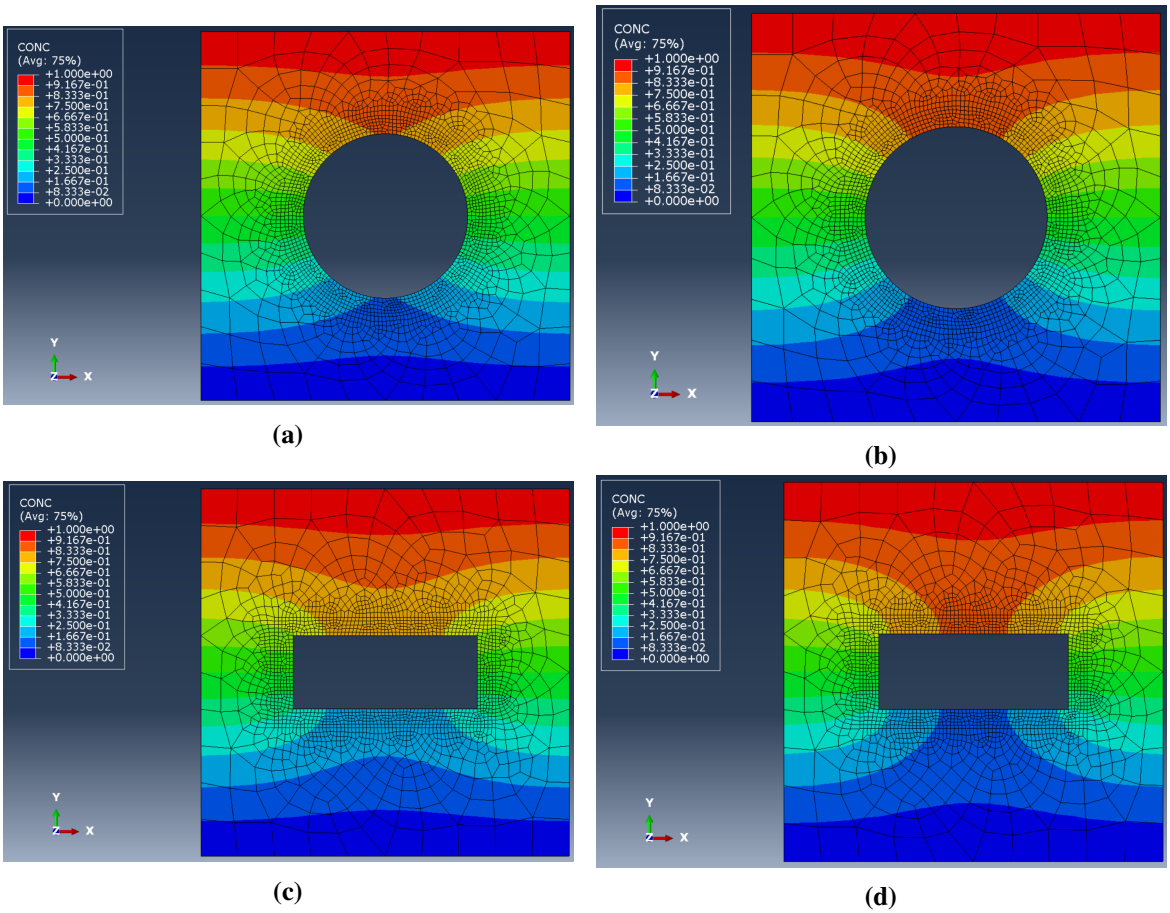
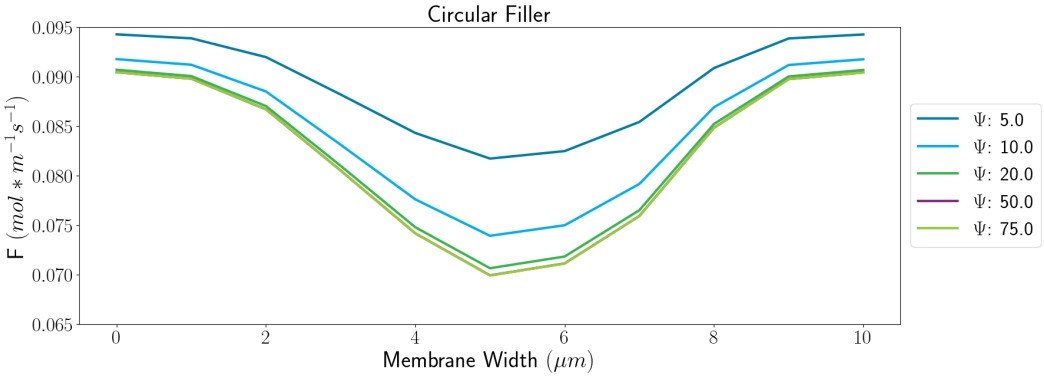


Fig. 4.1 Concentration results of the simulations of a membrane with (a) a circular filler in the center and an interface zone, (b) a circular filler in the center and no interface zone, (c) a rectangular filler in the center and an interface zone and (d) a rectangular filler in the center and no interface zone.

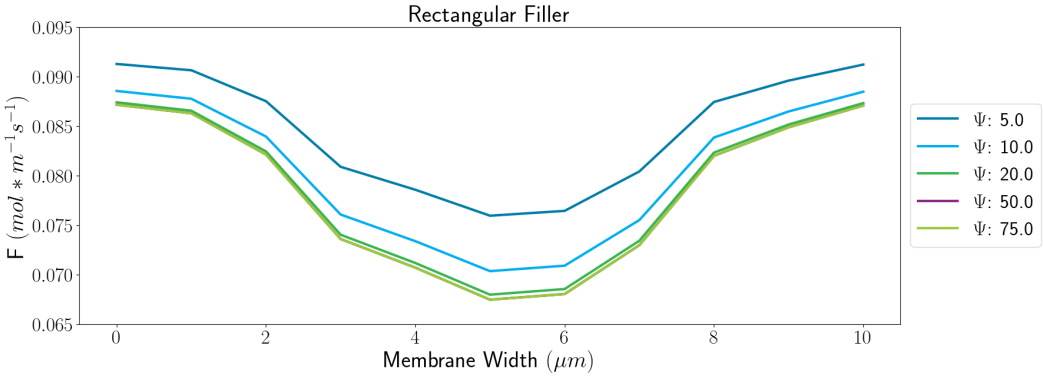
- Interface thickness, $w_I = 0.5\,\mu\text{m}$

The purpose of these simulations was to get an overview of the influence of Ψ on the simulation results. This was necessary as the value of Ψ was only assigned based on the results in Fig. 3.6 and a validation was required. As can be seen from these figures, a Ψ of 50 or more results in a complete overlap of the curves. The reason for this is that only the diffusivity of the mesh nodes closest to the filler particle is affected. However, since it was desired to emulate a bleeding effect as described in Section 3.1, it was decided to leave the value of Ψ at 20. These figures also show the effect of the central filler particle on the outflow from the membrane. In both cases (circle and rectangle), there is a significant drop in the outflow at the center of the membrane. However, the width of this drop in the graph is greater for the rectangular filler particle. This is due to the fact that the dimensions of the round and

rectangular fillers were chosen so that the areas are equal, and since the rectangular particle is not square, the result is a particle that is wider perpendicular to the direction of diffusion.



(a)



(b)

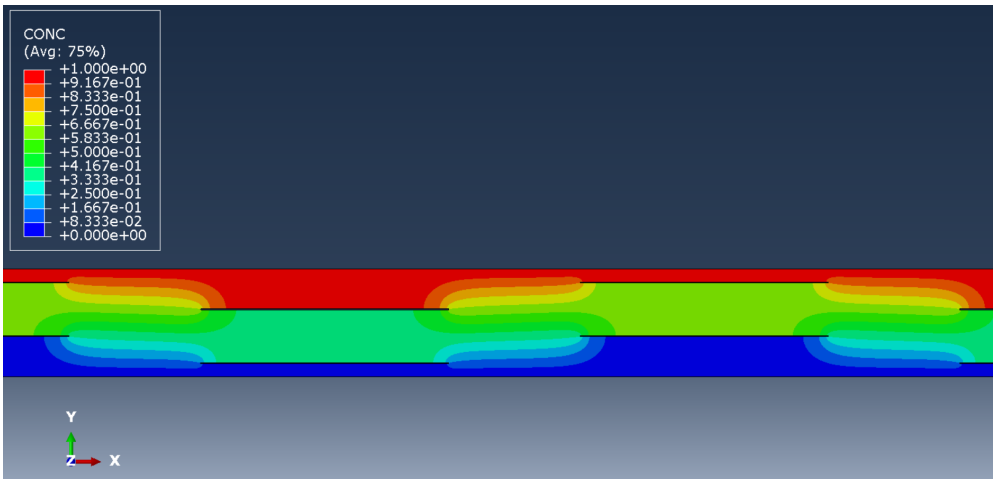
Fig. 4.2 (a) Outflow F over the width of a membrane with one circular filler in the middle, (b) outflow over the width of a membrane with one rectangular filler in the middle.

4.2 Raw Results - Multiple Filler Particles

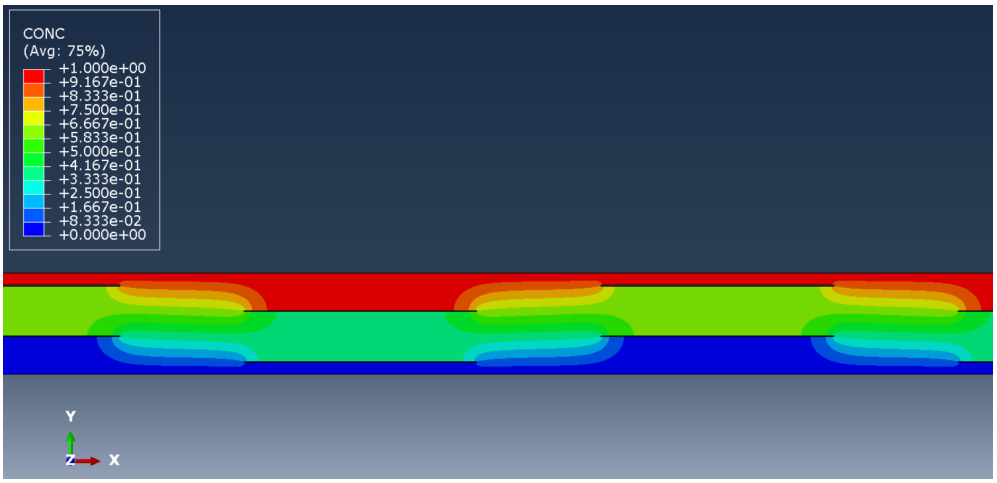
In Fig. 4.3, the concentration results are shown for the simulations with multiple rectangular filler particles in the membrane, as provided by ABAQUS. Comparing Fig. 4.3a with Fig. 4.3b shows that the size of the filler particles is too small compared to the membrane to produce any visible differences. This proves the previous point that it is necessary to perform preliminary simulations on simpler systems to gather information for the more complex simulation.

Fig. 4.4 shows the raw results of simulations with multiple filler particles in the membrane. These simulations were carried out with the following parameters:

- $\phi_f = 0.01$



(a)



(b)

Fig. 4.3 Concentration results of simulating a membrane with (a) multiple rectangular filler particles and an interface zone around each filler, (b) multiple rectangular filler particles without the interface zone.

- $w_I = 0.25\,\mu\text{m}$
- $D_M = 1.0\,\text{m}^2 \cdot \text{s}^{-1}$
- $D_{IP} = 10.0\,\text{m}^2 \cdot \text{s}^{-1}$
- $D_{IN} = 1.0\,\text{m}^2 \cdot \text{s}^{-1}$
- Interface Prefactor (Ψ) = 20.0

The results shown here are an excerpt from the final results of this work, which are shown and described in Fig. 4.6. The purpose of this presentation is to show how the outflow of a

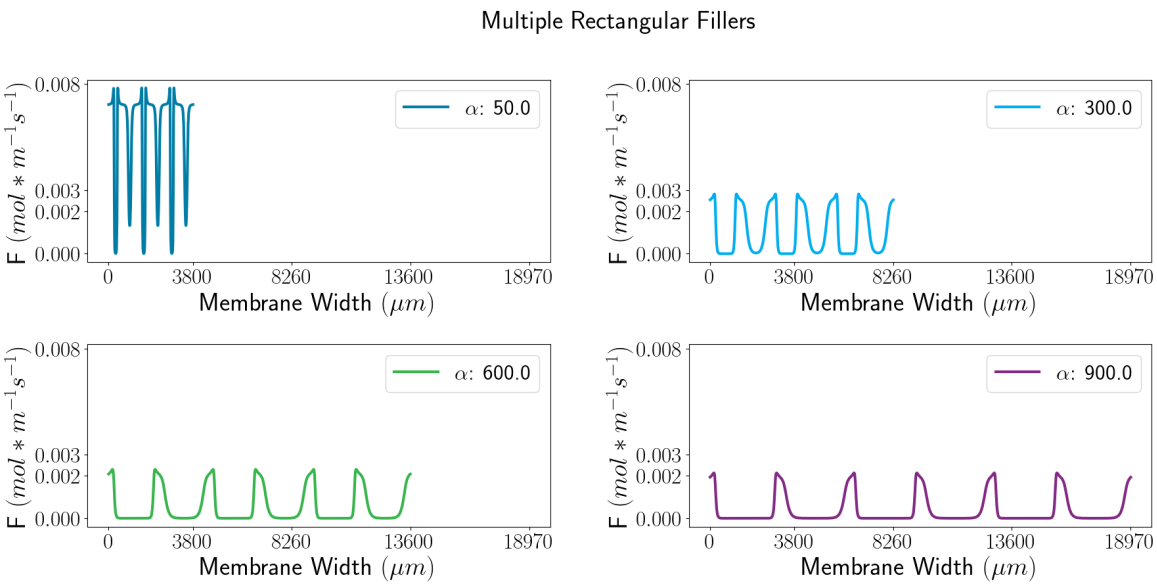


Fig. 4.4 Outflows across the width of various membranes with multiple rectangular fillers. The membranes differ by the α values of the filler particles, as shown in the legends of the plots.

membrane with multiple filler particles behaves. The width of the membrane depends on the number of filler particles, the aspect ratio α of the particles, and ϕ_f of the particles in the membrane. Therefore, as the aspect ratio increases, the width of the membrane also increases, as shown in Fig. 4.6. The thickness of the filler particles, i.e. their expansion in the diffusion direction, is constant, so α only affects the width of the filler particles, i.e. their expansion perpendicular to the diffusion direction. This is the reason why the valleys in the curves in Fig. 4.6 become wider as α increases. Furthermore, this behavior of the filler particles also results in a stronger barrier effect, which also reduces the total amount of outflow Q as α increases.

4.3 Accumulated Permeate - One Filler Particle

In Fig. 4.5 the processed results of several simulations with either a round or a rectangular filler particle are shown. The results were processed by integrating the outflow values over the width of each membrane and accumulating the outflow values over the duration of the simulation to obtain the total outflow over time Q_{total} from the membrane. These results show the effect of Ψ and D_{IP} on the amount of total outflow from the membrane.

When comparing the two subfigures in Fig. 4.5, it is noticeable that the integrated outflow values of the membrane with a round filler particle are higher than those of the membrane

with a rectangular filler particle, although both particle types have the same area. This difference is due to the fact that the rectangular particle is wider than the round particle. To achieve a higher barrier effect for the same filler area fraction, particles with a large width but small thickness should be used. In the case of the definition used in this work, this means that a high aspect ratio results in better barrier properties for the same filler area fraction. Under this condition, a significant barrier effect can be achieved, even with a low particle filler area fraction. However, highly aspected filler particles have the disadvantage that the orientation of the filler particles has a major influence on the barrier effect.

The resulting value of a simulation without interface zone around the filler particle is shown in both subfigures of Fig. 4.5 as a red line. If D_{IP} is set to 1, the resulting cumulative outflow values lie on the red line. If D_{IP} is greater than 1, the result is above the red line, indicating that more outflow has occurred. If D_{IP} is less than 1, the result is below the red line, indicating less outflow.

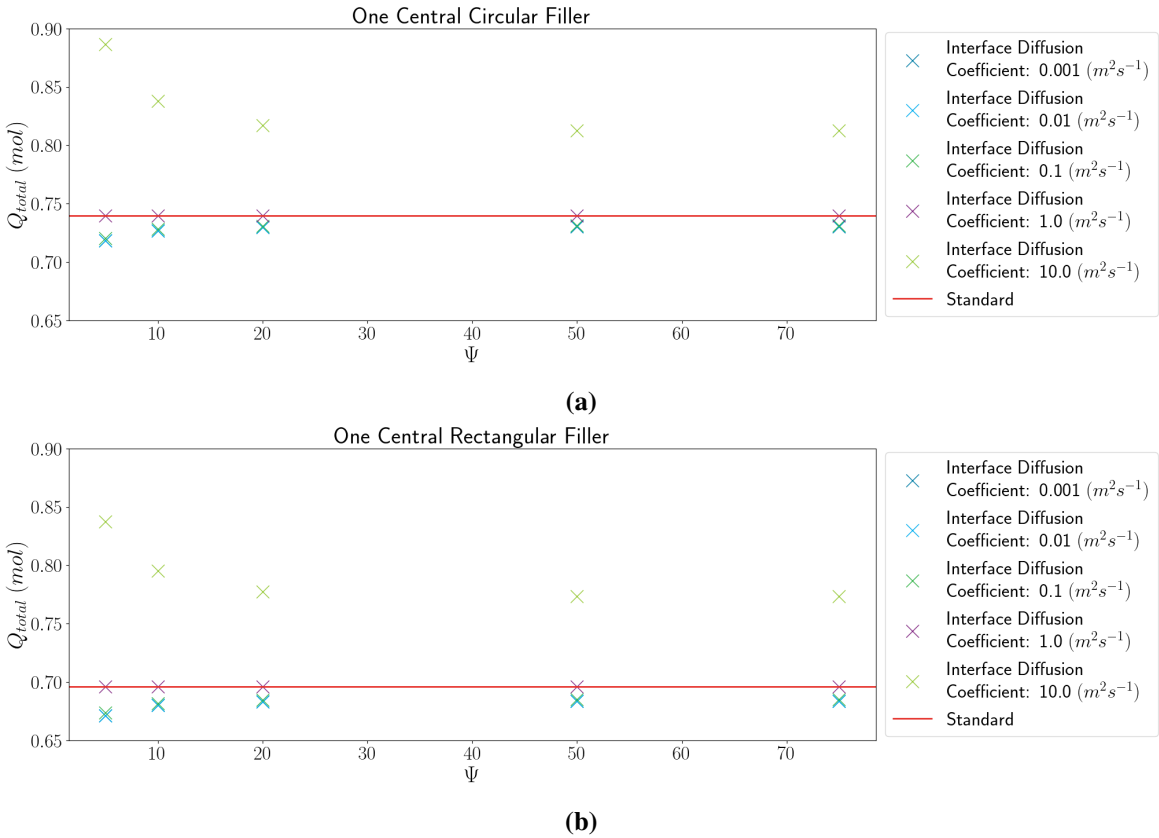


Fig. 4.5 (a) Integrated outflow of membranes with one circular filler in the middle, (b) integrated outflows of membranes with one rectangular filler in the middle.

4.4 Accumulated Permeate - Multiple Filler Particles

One aim of this work was to compare the following three types of data and assess the influence of an interface zone on the diffusivity of reinforced membranes:

- Analytically obtained data from the Nielsen model described in detail in Section 2.5
- FEM simulated result data with interface zones, which was described in detail in Section 3.3.3
- FEM simulated result data without interface zones, which was described in detail in Section 3.3.4

This comparison is shown in Fig. 4.6, where the analytical data are plotted as continuous lines, the FEM data without interface zone are labeled "o" and the FEM data with interface zone are labeled "x". As described in Section 2.5, the filler particles in the Nielsen model are arranged in a regular pattern. This creates channels in which unhindered diffusion flow is possible. As ϕ_f and α increase, the width of these channels relative to the membrane width decreases. Above a certain ϕ_f value, this ratio remains more or less constant. The ϕ_f value at which this plateau occurs, depends on α. The higher the α value, the earlier the plateau occurs. This structure of the membrane causes the outflow to decrease with increasing ϕ_f as well as with increasing α. This result was expected because increasing the number of filler particles or increasing the size of the filler particles should result in an increased barrier effect.

Since the differences between the FEM data and the analytical data are difficult to see in Fig. 4.6, the absolute deviations of the FEM data from the analytical data were calculated and are shown in Fig. 4.7. The standard FEM model agrees quite well with the analytical model for filler area fractions between 0.06 and 0.09. When ϕ_f is below this range, the results of the FEM model are higher than those of the analytical model. When ϕ_f is above this range, the results are lower than those of the analytical model. The results of the FEM model with interface zone are consistently higher than those of the standard FEM model, since the interfacial zone facilitates diffusion flow through the membrane. The advanced FEM model agrees quite well with the analytical model for filler area fractions between 0.06 and 0.15.

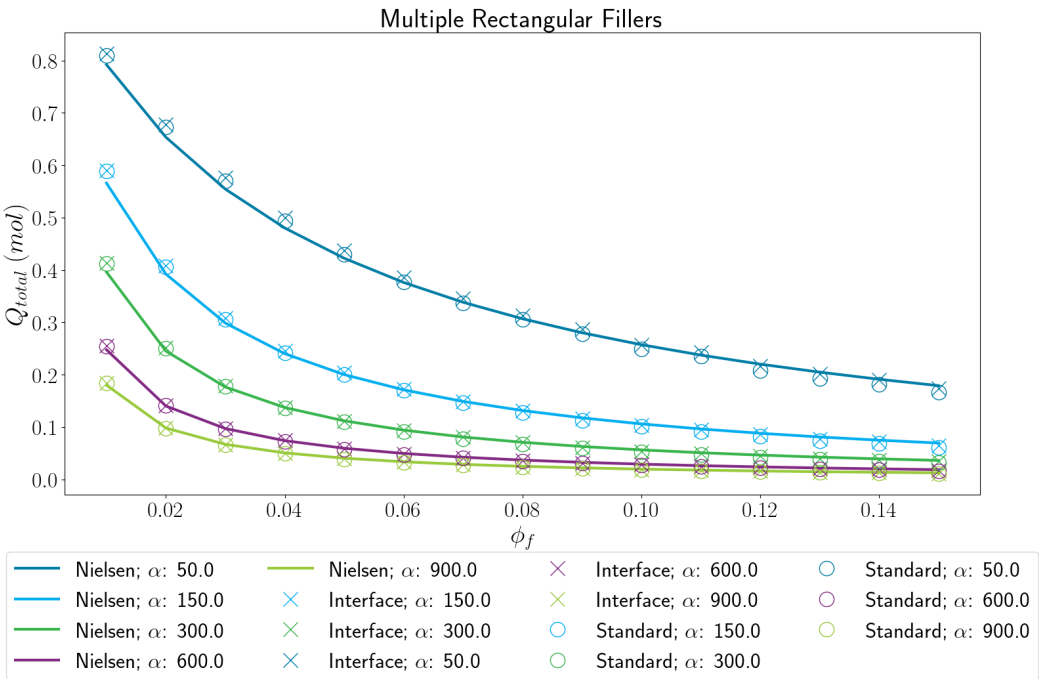


Fig. 4.6 Accumulated outflows of membranes with multiple rectangular fillers.

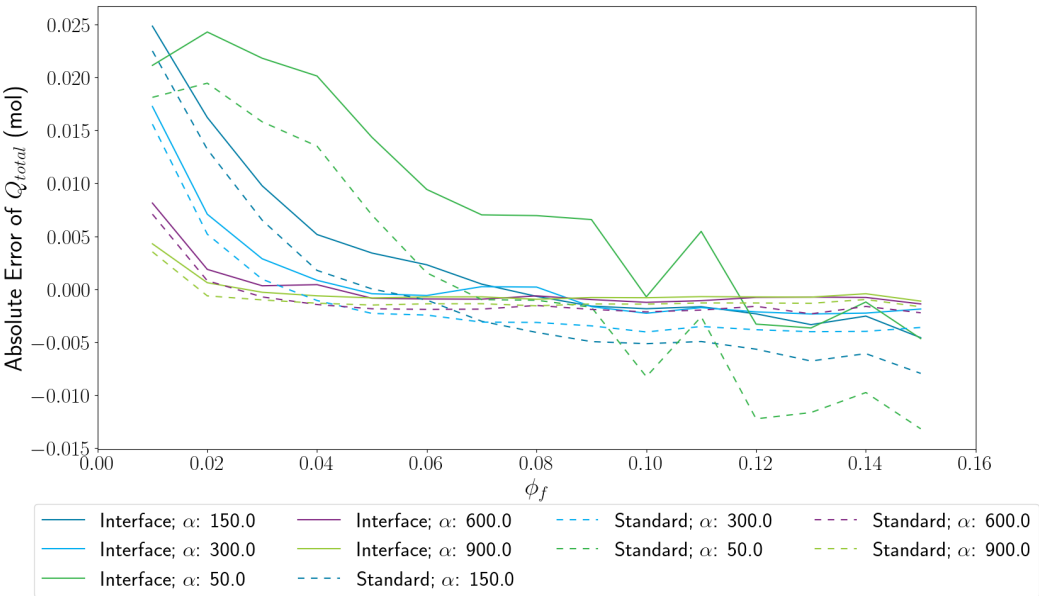


Fig. 4.7 Absolute error of the total outflow from the membranes in FEM models in respect to the analytically obtained data.

Chapter 5

Conclusions

The aim of this thesis was to numerically simulate the diffusion of hydrogen through different particle filled membranes with the introduction of a filler-matrix interface. The difference in the membranes was based on the area fraction of the particles in the membrane and the width of the filler particles. ABAQUS was chosen as the numerical solver for this work because of the software's feature set and in-house knowledge of it. The extended Nielsen model was selected as the basis for the simulation model because it has the advantage that it can be used in both analytical and numerical simulations, making the results directly comparable.

Implementing the interface zone was difficult for two reasons. Firstly, the implementation of an interface zone in ABAQUS was not easy. Instead of the ABAQUS GUI, Fortran subroutines had to be used, which had to be learned first. Furthermore, a lot of research had to be done to find the right subroutines for the problem at hand, as there are over 100 different subroutines avaiable in ABAQUS. Each of these subroutines has its own usecase, comes with different methods and functions and has access to different systems of ABAQUS. Secondly, the implementation of the Python modules and the subroutines had to be improved, as a first, non-optimized attempt showed very poor performance due to too many file calls for each calculation step. All calculations necessary for the subroutines to function were carried out using Python scripts and the results were saved on the hard disk. These result files were then opened several times by the Fortran subroutines in each simulation step.

In its final implementation, the interface zone was integrated into the model at a mesh node and integration point level. Three subroutines were used for this setup. The first subroutine had the sole purpose of calculating the orientation of each node in the interface zone, based on its position in this zone, relative to the global coordinate system. The second subroutine was used to calculate the position of each node in the interface zone relative to the thickness

of the interface zone. Therefore, if the node is exactly at the edge of the filler particle, the relative position value is equal to one, and if the node is at the outer edge of the interface zone, the relative position value is equal to zero. The results of these calculations are stored in a scalar field attached to all interface mesh nodes. The third subroutine is necessary because the relative position values stored in the scalar field cannot be used directly by ABAQUS to adjust the material data. This third subroutine takes the scalar field values at the mesh node positions and interpolates them to the integration points of the corresponding mesh elements. With this sophisticated system in place, it was possible to create the desired behavior of the interface zone and perform the necessary simulations for this work.

The results of this work not only show that such a FEM model with interface zone is possible, but also provide plausible data with respect to the standard FEM and the analytical model. However, both numerical models show larger deviations from the analytical model for membranes with low filler area fraction and small filler particle widths. The analytical model predicts better barrier properties for such filled membranes than the numerical simulations. Due to the lack of data from real experiments, it was not possible to quantitatively compare the two numerical models. After completing the work, it can be said that ABAQUS has a steep learning curve, but the simulation results and the adaptability of the simulation models with the subroutines are convincing.

Chapter 6

Outlook

The next step to improve this model should be to optimize the code to achieve better simulation runtime and stability. Before extending the simulation setup further, consideration should be given to comparing the results not only with analytical data, but also with real experimental results. These comparisons are essential for the evaluation of the model. Below are some suggestions about additions to the model that could lead to an improvement in accuracy.

For example, the filler particles could be placed randomly. This method of membrane generation would allow for more naturally built models. Another way to generate more realistic data would be to develop a three-dimensional system. This setup would make it easier to replicate real world experiments. Since the Nielsen model, which was heavily referenced in this thesis, only works in two dimensions, either a new model for the structure of the membrane would have to be developed, the filler particles would have to be placed randomly or the model could be based on a CT scan of a real reinforced polymer membrane.

For a different simulation model in this project, an attempt was made to implement a random-based distribution of filler particles in a two-dimensional model. The implementation was successful, but required a lot of optimization to achieve a reasonable runtime of the membrane generation. It can be concluded that it would be even more difficult to achieve a reasonable implementation in a three-dimensional model. [26]

7 References

[1] Alhijazi, M., Zeeshan, Q., Qin, Z., Safaei, B., and Asmael, M. (2020). Finite element analysis of natural fibers composites: A review. *Nanotechnology Reviews*, 9(1):853–875.

[2] Barrera, O., Tarleton, E., Tang, H. W., and Cocks, A. (2016). Modelling the coupling between hydrogen diffusion and the mechanical behaviour of metals. *Computational Materials Science*, 122:219–228.

[3] Crank, J. (1975). *The Mathematics of Diffusion*. Clarendon Press, Oxford, second edition edition.

[4] Díaz, A., Alegre, J. M., and Cuesta, I. I. (2016). Coupled hydrogen diffusion simulation using a heat transfer analogy. *International Journal of Mechanical Sciences*, 115-116:360–369.

[5] Duncan, B., Urquhart, J., and Roberts, S. (2005). *Review of Measurement and Modelling of Permeation and Diffusion in Polymers*. Report, National Physical Laboratory.

[6] Einstein, A. (1905). Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Annalen der Physik*, (322(8)):549–560.

[7] Eslami, S., Honarbakhsh-Raouf, A., and Eslami, S. (2015). Effects of moisture absorption on degradation of E-glass fiber reinforced Vinyl Ester composite pipes and modelling of transient moisture diffusion using finite element analysis. *Corrosion Science*, 90:168–175.

[8] Eslami, S., Rauf-Honarbakhsh, A., and Eslami, S. (2014). *Durability of E-glass/Vinyl Ester Composite Structures and Their Modeling in ABAQUS*. Proceedings of the American Society for Composites 2014-Twenty-ninth, University of California San Diego.

[9] Gao, S.-L., Mäder, E., and Zhandarov, S. F. (2004). Carbon fibers and composites with epoxy resins: Topography, fractography and interphases. *Carbon*, 42(3):515–529.

[10] George, S. C. and Thomas, S. (2001). Transport phenomena through polymeric systems. *Progress in Polymer Science*, 26(6):985–1017.

[11] Gholami, M., Afrasiab, H., Baghestani, A. M., and Fathi, A. (2021). Hygrothermal degradation of elastic properties of fiber reinforced composites: A micro-scale finite element analysis. *Composite Structures*, 266:113819.

[12] Hatsuo Ishida and Jack L Koenig (1978). Fourier transform infrared spectroscopic study of the silane coupling agent/porous silica interface. *Journal of Colloid and Interface Science*, 64(3):555–564.

[13] Huan-Chang, T., Chia-Hsiang, H., and Rong-Yeu, C. (2017). Long Fiber Orientation and Structural Analysis Using Moldex3D, Digimat and ABAQUS Simulations: Conference Paper - SPE ANTEC® Anaheim 2017.

[14] Kabir, M. M., Wang, H., Lau, K. T., and Cardona, F. (2012). Chemical treatments on plant-based natural fibre reinforced polymer composites: An overview. *Composites Part B*, 43(7):2883–2892.

[15] Kedward, L. J., Aradi, B., Čertík, O., Curcic, M., Ehlert, S., Engel, P., Goswami, R., Hirsch, M., Lozada-Blanco, A., Magnin, V., Markus, A., Pagone, E., Pribec, I., Richardson, B., Snyder, H., Urban, J., and Vandenplas, J. (2022). The State of Fortran. *Computing in Science & Engineering*, 24(2):63–72.

[16] Kim, J.-K. and Hodzic, A. (2003). Nanoscale characterisation of thickness and properties of interphase in polymer matrix composites. *The Journal of Adhesion*, 79(4):383–414.

[17] Kim, N.-H., OH, C.-S., KIM, Y.-J., YOON, K.-B., and Ma, Y.-W. (2012). Hydrogen-assisted stress corrosion cracking simulation using the stress-modified fracture strain model. *Journal of Mechanical Science and Technology*, 26(8):2631–2638.

[18] Klopffer, M. H. and Flaconneche, B. (2001). Transport Properties of Gases in Polymers: Bibliographic Review. *Oil & Gas Science and Technology*, 56(3):223–244.

[19] Li, H., Wang, N., Han, X., Fan, B., Feng, Z., and Guo, S. (2020). Simulation of Thermal Behavior of Glass Fiber/Phenolic Composites Exposed to Heat Flux on One Side. *Materials*, 13(2).

[20] Li, H.-W. and Onoue, K. (2016). Compressed Hydrogen: High-Pressure Hydrogen Tanks. In Sasaki, K., Li, H.-W., Hayashi, A., Yamabe, J., Ogura, T., and Lyth, S. M., editors, *Hydrogen Energy Engineering*, Green Energy and Technology, pages 273–278. Springer Japan, Tokyo.

[21] Li, Y., Chen, X., Jin, L., and Zhang, R. (2016). Experimental and numerical study on chloride transmission in cracked concrete. *Construction and Building Materials*, 127:425–435.

[22] Liu, F., Wang, D., Liu, J., Wei, H., Zhang, H., Xu, J., Li, S., Qin, Z., Wang, R., Jia, H., and Zhang, J. (2020). Reviews on Interfacial Properties of the Carbon Fiber Reinforced Polymer Composites. *Journal of Physics: Conference Series*, 1637(1):012027.

[23] Liu, L., Jia, C., He, J., Zhao, F., Fan, D., Xing, L., Wang, M., Wang, F., Jiang, Z., and Huang, Y. (2015). Interfacial characterization, control and modification of carbon fiber reinforced polymer composites. *Composites Science and Technology*, 121:56–72.

[24] Macher, J., Golestaneh, P., Macher, A. E., and Hausberger, A. (2022a). Filler Models Revisited: Extension of the Nielson Model with Respect to the Geometric Arrangements of Fillers. *Polymers*.

[25] Macher, J., Hausberger, A., Macher, A. E., Morak, M., and Schrittesser, B. (2020). Critical Review of Models for H2-Permeation through Polymers with Focus on the Differential Pressure Method. *International Journal of Hydrogen Energy*.

[26] Macher, J., Morak, M., and Hausberger, A. (2022b). *Hydrogen Permeation in Filled Polymers: Correlation of Permeation Properties and Filler Arrangements*. Report, Polymer Competence Center Leoben GmbH, Leoben.

[27] Minelli, M., Baschetti, M. G., and Doghieri, F. (2009). Analysis of modeling results for barrier properties in ordered nanocomposite systems. *Journal of Membrane Science*, 327(1-2):208–215.

[28] Mishin, Y., Herzig, C., Bernardini, J., and Gust, W. (1997). Grain boundary diffusion: fundamentals to recent developments. *International Materials Reviews*, 42(4):155–178.

[29] Monsalve-Bravo, G. and Bhatia, S. (2018). Modeling Permeation through Mixed-Matrix Membranes: A Review. *Processes*, 6(9):172.

[30] Nielsen, L. E. (1967). Models for the Permeability of Filled Polymer Systems. *Journal of Macromolecular Science: Part A - Chemistry*, 1(5):929–942.

[31] OH, C.-S., KIM, Y.-J., and YOON, K.-B. (2010). Coupled Analysis of Hydrogen Transport using ABAQUS. *Journal of Solid Mechanics and Materials Engineering*, 4(7):908–917.

[32] Olden, V., Saai, A., Jemblie, L., and Johnsen, R. (2014). FE simulation of hydrogen diffusion in duplex stainless steel. *International Journal of Hydrogen Energy*, 39(2):1156–1163.

[33] OpenCFD Ltd. (2011 - 2023). OpenFOAM.

[34] Ounaies, M., Harchay, M., Dammak, F., and Daly, H. B. (2018). Prediction of hygrothermal behavior of polyester/glass fiber composite in dissymmetric absorption. *Journal of Composite Materials*, 52(29):4001–4007.

[35] Papathanasiou, T. D. and Tsiantis, A. (2017). Orientational randomness and its influence on the barrier properties of flake-filled composite films. *Journal of Plastic Film & Sheeting*, 33(4):438–456.

[36] Philibert, J. (1991). *Atom movements - diffusion and mass transport in solids*. Les Editions de Physique.

[37] Pu, C., Gao, Y., Wang, Y., and Sham, T.-L. (2017). Diffusion-coupled cohesive interface simulations of stress corrosion intergranular cracking in polycrystalline materials. *Acta Materialia*, 136:21–31.

[38] Pukánszky, B. (2005). Interfaces and interphases in multicomponent materials: past, present, future. *European Polymer Journal*, 41(4):645–662.

[39] Sasaki, K., Li, H.-W., Hayashi, A., Yamabe, J., Ogura, T., and Lyth, S. M., editors (2016). *Hydrogen Energy Engineering*. Green Energy and Technology. Springer Japan, Tokyo.

[40] Scheichl, R., Klopffer, M., Benjelloundabaghi, Z., and Flaconneche, B. (2005). Permeation of gases in polymers: parameter identification and nonlinear regression analysis. *Journal of Membrane Science*, 254(1-2):275–293.

[41] Schultheiß, D. (2007). *Permeation Barrier for Lightweight Liquid Hydrogen Tanks*. Master's Thesis, Universität Augsburg, Augsburg.

[42] Smith, M. (2009). ABAQUS/Standard User's Manual, Version 6.9.

[43] Sukjoo, C. (2005). *Micromechanics, Fracture Mechanics and Gas Permeability of Composite Laminates for Cryogenic Storage Systems*. Dissertation, University of Florida.

[44] van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[45] von Smoluchowski, M. (1906). Zur kinetischen Theorie der Brownschen Molekularbewegung und der Suspensionen. *Annalen der Physik*, 326(14):756–780.

[46] Wu, Q., Li, M., Gu, Y., Li, Y., and Zhang, Z. (2014). Nano-analysis on the structure and chemical composition of the interphase region in carbon fiber composite. *Composites Part A: Applied Science and Manufacturing*, 56:143–149.

[47] Zhang, S., Wang, H., Hou, F., Chen, H., and Tan, P. (2015). Influence of the Types of Stress on Hydrogen Induced Damage By Simulating Hydrogen Diffusion. *International Conference on Materials, Environmental and Biological Engineering*.

[48] Zhao, R., Wang, M., and Guan, X. (2023). Exploring Exact Effects of Various Factors on Chloride Diffusion in Cracked Concrete: ABAQUS-Based Mesoscale Simulations. *Materials*, 16(7).

[49] Zheng, H., Zhang, W., Li, B., Zhu, J., Wang, C., Song, G., Wu, G., Yang, X., Huang, Y., and Ma, L. (2022). Recent advances of interphases in carbon fiber-reinforced polymer composites: A review. *Composites Part B: Engineering*, 233:109639.

[50] Zhou, Y., Fan, M., and Chen, L. (2016). Interface and bonding mechanisms of plant fibre composites: An overview. *Composites Part B: Engineering*, 101:31–45.

8 List of figures

9 List of tables

10 List of symbols

α	Aspect ratio of the filler particles.	(dimensionless)
$\frac{\partial C}{\partial t}$	Change of the concentration in the membrane over time. (2D)	$[\frac{\partial C}{\partial t}] = \mathrm{mol} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
$\frac{\partial p}{\partial x}$	Pressure gradient along the thickness of the membrane. (2D)	$[\frac{\partial p}{\partial x}] = \mathrm{Pa} \cdot \mathrm{m}^{-1}$
$\hat{C}$	Dimensionless concentration of the gas in the membrane.	(dimensionless)
$\hat{t}$	Dimensionless time.	(dimensionless)
$\hat{x}$	Dimensionless distance.	(dimensionless)
$\langle N \rangle$	Number of filler particles in the membrane.	(dimensionless)
μ	Mobility of the particles.	$[\mu] = \mathrm{s} \cdot \mathrm{kg}^{-1}$
∇C	Gradient of the concentration along the thickness of the membrane. (2D)	$[\nabla C] = \mathrm{mol} \cdot \mathrm{m}^{-3}$
ϕ_f	Area fraction of the filler particles in the membrane.	(dimensionless)
Ψ	Diffusion prefactor, necessary to calculate the interface diffusion coefficent in respect to the relative position in the interface zone.	(dimensionless)
τ	Tortuousity factor, describes the longer tortuous path of the diffusing medium due to the impermeable filler particles.	(dimensionless)
A_f	Area of the filler particle in an unit cell.	$[A_f] = \mathrm{m}^2$
A_u	Area of a unit cell in the membrane.	$[A_u] = \mathrm{m}^2$
b	Thickness of the filler particle.	$[b] = \mathrm{m}$

b_H	Describes the gradient of the linear segment at the start of the Langmuir model.	$[b_H] = \mathrm{Pa}^{-1}$
C	Concentration of the gas in the membrane. (2D)	$[C] = \mathrm{mol} \cdot \mathrm{m}^{-2}$
C'_H	Membrane saturation constant in the Langmuir model. (2D)	$[C'_H] = \mathrm{mol} \cdot \mathrm{m}^{-2}$
C_1	Concentration of the gas at the upstream border of the membrane. (2D)	$[C_1] = \mathrm{mol} \cdot \mathrm{m}^{-2}$
C_2	Concentration of the gas at the downstream border of the membrane. (2D)	$[C_2] = \mathrm{mol} \cdot \mathrm{m}^{-2}$
D	Diffusion coefficient of the membrane. (2D)	$[D] = \mathrm{m}^2 \cdot \mathrm{s}^{-1}$
d	Distance between two rows of filler particles.	$[d] = \mathrm{m}$
D_0	Diffusion coefficient of the unfilled membrane. (2D)	$[D_0] = \mathrm{m}^2 \cdot \mathrm{s}^{-1}$
D_{eff}	Effective diffusion coefficient of the filled membrane. (2D)	$[D_{eff}] = \mathrm{m}^2 \cdot \mathrm{s}^{-1}$
D_{Imax}	Maximum diffusion coefficient in the interface zone. (2D)	$[D_{Imax}] = \mathrm{m}^2 \cdot \mathrm{s}^{-1}$
D_{IN}	Diffusion coefficient in the interface zone, if flow is normal to the filler particle edge. (2D)	$[D_{IN}] = \mathrm{m}^2 \cdot \mathrm{s}^{-1}$
D_{IP}	Diffusion coefficient in the interface zone, if flow is parallel to the filler particle edge. (2D)	$[D_{IP}] = \mathrm{m}^2 \cdot \mathrm{s}^{-1}$
D_M	Diffusion coefficient of the membrane outside of the interface zone. (2D)	$[D_M] = \mathrm{m}^2 \cdot \mathrm{s}^{-1}$
F	Diffusive flow density through the membrane. (2D)	$[F] = \mathrm{mol} \cdot \mathrm{s}^{-1} \cdot \mathrm{m}^{-1}$
F_0	Flow density through the unfilled membrane. (2D)	$[F_0] = \mathrm{mol} \cdot \mathrm{s}^{-1} \cdot \mathrm{m}^{-1}$
F_{ch}	Unhindered flow density through the channels of the membrane. (2D)	$[F_{ch}] = \mathrm{mol} \cdot \mathrm{s}^{-1} \cdot \mathrm{m}^{-1}$
f_{ch}	Ratio of the flow through the unobstructed channel in respect to the total flow.	(dimensionless)

F_{eff}	Effective flow through the filled membrane. (2D)	$[F_{eff}] = \mathrm{mol} \cdot \mathrm{s}^{-1} \cdot \mathrm{m}^{-1}$
F_t	Tortuous flow density through the channels of the membrane. (2D)	$[F_t] = \mathrm{mol} \cdot \mathrm{s}^{-1} \cdot \mathrm{m}^{-1}$
f_t	Ratio of the tortuous flow in respect to the total flow.	(dimensionless)
k_b	Boltzmann constant.	$[k_b] = \mathrm{J} \cdot \mathrm{K}^{-1}$
L	Membrane width.	$[L] = \mathrm{m}$
L_{eff}	Effective length of the tortuous path of the permeate through the filled membrane.	$[L_{eff}] = \mathrm{m}$
L_t	Length of the tortuous path through the unit cell.	$[L_t] = \mathrm{m}$
P	Permeation coefficient of the membrane.	$[P] = \mathrm{mol} \cdot \mathrm{s}^{-1} \cdot \mathrm{Pa}^{-1}$
p	Partial pressure of the gas in the membrane.	$[p] = \mathrm{Pa}$
p_1	Pressure of the gas at the upstream boundary of the membrane.	$[p_1] = \mathrm{Pa}$
p_2	Pressure of the gas at the downstream boundary of the membrane.	$[p_2] = \mathrm{Pa}$
Q	Cumulative diffusive flow density through the membrane. (2D)	$[Q] = \mathrm{mol} \cdot \mathrm{m}^{-1}$
Q_{total}	Cumulative diffusive flow through the membrane.	$[Q_{total}] = \mathrm{mol}$
S	Solubility coefficient of the membrane. (2D)	$[S] = \mathrm{mol} \cdot \mathrm{m}^{-2} \cdot \mathrm{Pa}^{-1}$
s	Slit shape, distance between two filler particles in the same row.	$[s] = \mathrm{m}$
S_0	Solubility coefficient of the unfilled membrane. (2D)	$[S_0] = \mathrm{mol} \cdot \mathrm{m}^{-2} \cdot \mathrm{Pa}^{-1}$
S_{eff}	Effective solubility coefficient of the filled membrane. (2D)	$[S_{eff}] = \mathrm{mol} \cdot \mathrm{m}^{-2} \cdot \mathrm{Pa}^{-1}$
S_I	Solubility coefficient in the interface zone. (2D)	$[S_I] = \mathrm{mol} \cdot \mathrm{m}^{-2} \cdot \mathrm{Pa}^{-1}$
S_M	Solubility coefficient in the undisturbed membrane. (2D)	$[S_M] = \mathrm{mol} \cdot \mathrm{m}^{-2} \cdot \mathrm{Pa}^{-1}$
T	Temperature.	$[T] = \mathrm{K}$

T_c	Critical temperature of a gas.	$[T_c]$ = K
T_g	Glass transition temperature of a polymer.	$[T_g]$ = K
w	Width of the filler particle.	$[w]$ = m
w_I	Thickness of the interface zone.	$[w_I]$ = m
w_{ch}	Width of a channel in a filler particle arrangement.	$[w_{ch}]$ = m

Appendix A

Codes

A.1 Fortran - ABAQUS Subroutines

A.1.1 Subroutine ORIENT

```
      SUBROUTINE ORIENT(T,NOEL,NPT,LAYER,KSPT,COORDS,BASIS,
     * ORNAME,NNODES,CNODES,JNNUM)
C
        INCLUDE 'ABA_PARAM.INC'
C
        CHARACTER*80 ORNAME
C
        DIMENSION T(3,3),COORDS(3),BASIS(3,3),CNODES(3,NNODES)
        DIMENSION JNNUM(NNODES)

        user coding to define T

        RETURN
        END
```

A.1.2 Subroutine UFIELD

```
      SUBROUTINE UFIELD(FIELD,KFIELD,NSECPT,KSTEP,KINC,TIME,NODE,
     * COORDS,TEMP,DTEMP,NFIELD)
C
      INCLUDE 'ABA_PARAM.INC'
C
      DIMENSION FIELD(NSECPT,NFIELD), TIME(2), COORDS(3),
     1 TEMP(NSECPT), DTEMP(NSECPT)
C


      user coding to define FIELD


      RETURN
      END
```

A.1.3 Subroutine USDFLD

```
      SUBROUTINE USDFLD(FIELD,STATEV,PNEWDT,DIRECT,T,CELENT,
     * TIME,DTIME,CMNAME,ORNAME,NFIELD,NSTATV,NOEL,NPT,LAYER,
     * KSPT,KSTEP,KINC,NDI,NSHR,COORD,JMAC,JMATYP,MATLAYO,LACCFLA)
C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*80 CMNAME,ORNAME
      CHARACTER*3  FLGRAY(15)
      DIMENSION FIELD(NFIELD),STATEV(NSTATV),DIRECT(3,3),
     * T(3,3),TIME(2)
      DIMENSION ARRAY(15),JARRAY(15),JMAC(*),JMATYP(*),COORD(*)


      user coding to define FIELD and, if necessary, STATEV and PNEWDT


      RETURN
      END
```

A.2 Python - ABAQUS Control Scripts and General Utility Scripts

A.2.1 ABAQUS Input File Creation

```
# Imports
from abaqus import *
from abaqusConstants import *
from caeModules import *

import csv
import math
import numpy as np
import re
import sys


def log(msg):
    # Helper-function to print logs to the python cosole, the abaqus
    console and save them in a log file
    print(str(msg))
    print >> sys.__stdout__, str(msg)
    with open("../Logs/log.txt", "a") as log_file:
        log_file.write(str(msg))
        log_file.write("\n")


# Variables
case = None
with open("../DATA_FILES/actual_case.txt") as file:
    reader = csv.reader(file, delimiter=",")
    for row in reader:
        if row:
            case = row[0].replace("\\", "/").replace("DATA_FILES/", "")


case_path = "../DATA_FILES/" + case

geometric_params = []
with open(case_path + "/Geometric_Data.csv") as file:
    reader = csv.reader(file, delimiter=",")
    for row in reader:
```

```
        for row_elem in row:
            str_floats = re.findall(r"[-+]?\d*\.\d+|\d+", row_elem)
            tmp = []
            for str_elem in str_floats:
                tmp.append(float(str_elem))
            geometric_params.append(tmp)

name_params = None
with open(case_path + "/Output_Data.csv") as file:
    reader = csv.DictReader(file, delimiter=",")
    for row in reader:
        name_params = row

filler_radius = geometric_params[0][0]
interface_thickness = geometric_params[1][0]
membrane_thickness = geometric_params[2][0]
membrane_width = geometric_params[3][0]

solubility = geometric_params[4][0]
diffusivity_matrix = geometric_params[5][0]
diffusivity_interface = geometric_params[6][0]
interface_prefactor = geometric_params[7][0]

matrix_mesh_size = geometric_params[8][0]
interface_mesh_size = geometric_params[9][0]

filler_center = [geometric_params[10][0], geometric_params[11][0]]

interface_distances = [
    0.05,
    0.1,
    0.15,
    0.2,
    0.25,
    0.3,
    0.35,
    0.4,
    0.45,
    0.5,
    0.55,
    0.6,
    0.65,
    0.7,
    0.75,
```

```
    0.85,
    0.9,
    0.95,
    0.98,
    0.99,
]
diffusivity = [(1.0, diffusivity_matrix, 0.0, 0.0, 0.0)]

for dist in interface_distances:
    diff_value = diffusivity_interface * math.exp(interface_prefactor * (dist - 1)) + diffusivity_matrix * (
        1 - math.exp(interface_prefactor * (dist - 1))
    )
    diffusivity.append((1.0, diff_value, 0.0, 0.0, dist))

diffusivity.append((1.0, diffusivity_interface, 0.0, 0.0, 1.0))
diffusivity = tuple(diffusivity)

# ------------
# Abaqus Setup
# ------------

Mdb()
my_session = session
my_mdb = mdb

my_mdb.models.changeKey(fromName="Model-1", toName=name_params["model"])
my_model = my_mdb.models[name_params["model"]]

# -------------
# Part Creation
# -------------

sketch = my_model.ConstrainedSketch(name="__profile__", sheetSize=200.0)
sketch.rectangle(point1=(0, 0), point2=(membrane_width, membrane_thickness))
p1 = (filler_center[0] + filler_radius, filler_center[1])
sketch.CircleByCenterPerimeter(center=filler_center, point1=p1)
my_part = my_model.Part(name=name_params["part"], dimensionality=TWO_D_PLANAR, type=DEFORMABLE_BODY)
my_part.BaseShell(sketch=sketch)

sketch = my_model.ConstrainedSketch(name="__profile__", sheetSize=200.0)
```

```
p2 = (filler_center[0] + filler_radius + interface_thickness,
    filler_center[1])
sketch.CircleByCenterPerimeter(center=filler_center, point1=p2)
my_part.PartitionFaceBySketch(faces=my_part.faces, sketch=sketch)

# --------------
# Material Setup
# --------------

my_material = my_model.Material(name=name_params["material_matrix"])
my_material.Diffusivity(law=GENERAL, table=((diffusivity_matrix,),))
my_material.Solubility(table=((solubility,),))

my_material = my_model.Material(name=name_params["material_interface"])
my_material.Diffusivity(type=ORTHOTROPIC, law=GENERAL, dependencies=1,
    table=diffusivity)
my_material.Solubility(table=((solubility,),))
my_material.UserDefinedField()

# -------------
# Section Setup
# -------------

my_model.HomogeneousSolidSection(
    name=name_params["section_matrix"], material=name_params["
    material_matrix"], thickness=None
)
my_model.HomogeneousSolidSection(
    name=name_params["section_interface"], material=name_params["
    material_interface"], thickness=None
)

region_matrix = regionToolset.Region(
    faces=my_part.faces.findAt(
        ((filler_center[0] + filler_radius + interface_thickness + 1e-3,
     filler_center[1], 0),),
    )
)
region_interface = regionToolset.Region(
    faces=my_part.faces.findAt(
        ((filler_center[0] + filler_radius + interface_thickness - 1e-3,
     filler_center[1], 0),),
    )
)
```

```
my_part.SectionAssignment(
    region=region_matrix,
    sectionName=name_params["section_matrix"],
    offset=0.0,
    offsetType=MIDDLE_SURFACE,
    offsetField="",
    thicknessAssignment=FROM_SECTION,
)
my_part.SectionAssignment(
    region=region_interface,
    sectionName=name_params["section_interface"],
    offset=0.0,
    offsetType=MIDDLE_SURFACE,
    offsetField="",
    thicknessAssignment=FROM_SECTION,
)

# ----------
# Mesh Setup
# ----------

elemType1 = mesh.ElemType(elemCode=DC2D4, elemLibrary=STANDARD)
elemType2 = mesh.ElemType(elemCode=DC2D3, elemLibrary=STANDARD)

my_part.setMeshControls(regions=my_part.faces, elemShape=QUAD, algorithm
    =ADVANCING_FRONT)
my_part.setElementType(
    regions=regionToolset.Region(faces=my_part.faces),
    elemTypes=(elemType1, elemType2),
)
my_part.seedPart(size=matrix_mesh_size, deviationFactor=0.1,
    minSizeFactor=0.1)

interface_edges = my_part.edges.getByBoundingBox(
    xMin=(filler_center[0] - filler_radius - interface_thickness - 1e-3)
    ,
    xMax=(filler_center[0] + filler_radius + interface_thickness + 1e-3)
    ,
    yMin=(filler_center[1] - filler_radius - interface_thickness - 1e-3)
    ,
    yMax=(filler_center[1] + filler_radius + interface_thickness + 1e-3)
    ,
)
my_part.Set(edges=interface_edges, name="SeedByEdge")
```

```
my_part.seedEdgeBySize(edges=interface_edges, size=interface_mesh_size)
my_part.generateMesh()

# ----------------
# Oriantation Setup
# ----------------

my_part.MaterialOrientation(region=region_interface, orientationType=
    USER)

# --------------
# Assembly Setup
# --------------

my_assembly = my_model.rootAssembly
my_assembly.DatumCsysByDefault(CARTESIAN)

my_instance = my_assembly.Instance(name=name_params["part"] + "-1", part
    =my_part, dependent=ON)

# ---------
# Set Setup
# ---------

all_nodes = my_instance.nodes
top_nodes = my_instance.nodes.getByBoundingBox(
    xMin=-1e-3,
    yMin=membrane_thickness - 1e-3,
    zMin=0.0,
    xMax=membrane_width + 1e-3,
    yMax=membrane_thickness + 1e-3,
    zMax=0.0,
)
bot_nodes = my_instance.nodes.getByBoundingBox(
    xMin=-1e-3, yMin=-1e-3, zMin=0.0, xMax=membrane_width + 1e-3, yMax=1
    e-3, zMax=0.0
)

interface_nodes = my_instance.nodes.getByBoundingCylinder(
    center1=(filler_center[0], filler_center[1], 0 - 1e-3),
    center2=(filler_center[0], filler_center[1], 0 + 1e-3),
    radius=filler_radius + interface_thickness + 1e-3,
)

```

```
interface_elements = my_instance.elements.getByBoundingCylinder(
    center1=(filler_center[0], filler_center[1], 0 - 1e-3),
    center2=(filler_center[0], filler_center[1], 0 + 1e-3),
    radius=filler_radius + interface_thickness + 1e-3,
)

all_nodes_set = my_assembly.Set(nodes=all_nodes, name=name_params["
    set_All"])
interface_nodes_set = my_assembly.Set(nodes=interface_nodes, name=
    name_params["set_Interface_Nodes"])
interface_element_set = my_assembly.Set(nodes=interface_elements, name=
    name_params["set_Interface_Elements"])
inlet_set = my_assembly.Set(nodes=top_nodes, name=name_params["set_In"])
outlet_set = my_assembly.Set(nodes=bot_nodes, name=name_params["set_Out"
    ])


all_left_nodes = my_instance.nodes.getByBoundingBox(
    xMin=-1e-3,
    yMin=-1e-3,
    zMin=0.0,
    xMax=1e-3,
    yMax=membrane_thickness + 1e-3,
    zMax=0.0,
)
all_left_nodes_y_coord = []

all_right_nodes = my_instance.nodes.getByBoundingBox(
    xMin=membrane_width - 1e-3,
    yMin=-1e-3,
    zMin=0.0,
    xMax=membrane_width + 1e-3,
    yMax=membrane_thickness + 1e-3,
    zMax=0.0,
)
all_right_nodes_y_coord = []

for node in all_left_nodes:
    all_left_nodes_y_coord.append(node.coordinates[1])

left_idx = np.argsort(all_left_nodes_y_coord)

left_nodes = []
for idx in left_idx:
```

```
    for node in all_left_nodes:
        if node.coordinates[1] == all_left_nodes_y_coord[idx]:
            left_nodes.append(node.label)

for node in all_right_nodes:
    all_right_nodes_y_coord.append(node.coordinates[1])

right_idx = np.argsort(all_right_nodes_y_coord)

right_nodes = []
for idx in right_idx:
    for node in all_right_nodes:
        if node.coordinates[1] == all_right_nodes_y_coord[idx]:
            right_nodes.append(node.label)


node_sets = []
for ii in range(len(left_nodes)):
    right_name = "NR" + str(ii)
    left_name = "NL" + str(ii)

    my_assembly.SetFromNodeLabels(
        nodeLabels=(
            (
                name_params["part"] + "-1",
                (left_nodes[ii],),
            ),
        ),
        name=left_name,
    )
    my_assembly.SetFromNodeLabels(
        nodeLabels=(
            (
                name_params["part"] + "-1",
                (right_nodes[ii],),
            ),
        ),
        name=right_name,
    )
    node_sets.append([right_name, left_name])

# --------------
# Timestep Setup
# --------------
```

```

my_model.MassDiffusionStep(
    name=name_params["time_step"],
    previous="Initial",
    response=STEADY_STATE,
    amplitude=RAMP,
)
del my_model.historyOutputRequests["H-Output-1"]
my_model.fieldOutputRequests["F-Output-1"].setValues(variables=("MFL", "
    CONC", "COORD", "FV"))

# ---------
# Setup BCs
# ---------

inlet_region = regionToolset.Region(nodes=top_nodes)
my_model.ConcentrationBC(
    name=name_params["bc_In"],
    createStepName=name_params["time_step"],
    region=inlet_region,
    fixed=OFF,
    distributionType=UNIFORM,
    fieldName="",
    magnitude=1.0,
    amplitude=UNSET,
)

outlet_region = regionToolset.Region(nodes=bot_nodes)
my_model.ConcentrationBC(
    name=name_params["bc_Out"],
    createStepName=name_params["time_step"],
    region=outlet_region,
    fixed=OFF,
    distributionType=UNIFORM,
    fieldName="",
    magnitude=0.0,
    amplitude=UNSET,
)

# ------------
# Setup Fields
# ------------

field_region = regionToolset.Region(nodes=interface_nodes_set.nodes)
```

```
my_model.Field(
    name=name_params["diff_field"],
    createStepName=name_params["time_step"],
    region=field_region,
    distributionType=USER_DEFINED,
    fieldVariableNum=1,
)

# ---------
# Setup Constraints
# ---------

for ii, node_set in enumerate(node_sets):
    right_name = node_set[0].upper()
    left_name = node_set[1].upper()
    my_model.Equation(
        name="eq" + str(ii),
        terms=((1.0, right_name, 11), (-1.0, left_name, 11)),
    )

# ---------
# Setup Job
# ---------

my_job = my_mdb.Job(
    name=case,
    model=name_params["model"],
    description="",
    type=ANALYSIS,
    atTime=None,
    waitMinutes=0,
    waitHours=0,
    queue=None,
    memory=90,
    memoryUnits=PERCENTAGE,
    getMemoryFromAnalysis=True,
    explicitPrecision=SINGLE,
    nodalOutputPrecision=SINGLE,
    echoPrint=OFF,
    modelPrint=OFF,
    contactPrint=OFF,
    historyPrint=OFF,
    userSubroutine="",
    scratch="",
```

```
    resultsFormat=ODB,
    numThreadsPerMpiProcess=1,
    multiprocessingMode=DEFAULT,
    numCpus=2,
    numDomains=2,
    numGPUs=1,
)

my_job.writeInput()
```

A.2.2 ABAQUS Result Extraction

```
# Imports
from odbAccess import *
import numpy as np
import os
import csv


def log(msg):
    # Helper-function to print logs to the python cosole, the abaqus console and save them in a log file
    print(str(msg))
    print >> sys.__stdout__, str(msg)
    with open("../Logs/export_log.txt", "a") as log_file:
        log_file.write(str(msg))
        log_file.write("\n")


case = None
with open("../DATA_FILES/actual_case.txt") as file:
    reader = csv.reader(file, delimiter=",")
    for row in reader:
        if row:
            case = row[0].replace("\\", "/").replace("DATA_FILES/", "")

case_path = "../DATA_FILES/" + case

name_params = None
with open(case_path + "/Output_Data.csv") as file:
    reader = csv.DictReader(file, delimiter=",")
    for row in reader:
```

```
        name_params = row

# Export Data
my_session = session
my_odb = my_session.openOdb(name=case + ".odb")
frame = my_odb.steps["time_step"].frames[1]
instance = my_odb.rootAssembly.instances[name_params["part"].upper() + "-1"]

set_region = my_odb.rootAssembly.nodeSets[name_params["set_Out"].upper()]
mfl_data = frame.fieldOutputs["MFL"].getSubset(region=set_region, position=ELEMENT_NODAL).values

data_list = []
for index in range(len(mfl_data)):
    data_list.append(
        [
            instance.getNodeFromLabel(mfl_data[index].nodeLabel).coordinates[0],
            instance.getNodeFromLabel(mfl_data[index].nodeLabel).coordinates[1],
            mfl_data[index].data[1],
        ]
    )
np.savetxt(
    "../RESULTS/Result_" + case + ".csv",
    data_list,
    delimiter=", ",
    fmt="%s",
    header="x, y, mfl_2",
)
```

A.2.3 ABAQUS Standard Simulation

```
# Imports
from abaqus import *
from abaqusConstants import *
from caeModules import *

import csv
import math
import numpy as np
import re
import sys


def log(msg):
    # Helper-function to print logs to the python cosole, the abaqus
    console and save them in a log file
    print(str(msg))
    print >> sys.__stdout__, str(msg)
    with open("../Logs/log.txt", "a") as log_file:
        log_file.write(str(msg))
        log_file.write("\n")


# Variables
case = None
with open("../DATA_FILES/actual_case.txt") as file:
    reader = csv.reader(file, delimiter=",")
    for row in reader:
        if row:
            case = row[0].replace("\\", "/").replace("DATA_FILES/", "")


case_path = "../DATA_FILES/" + case

geometric_params = []
with open(case_path + "/Geometric_Data.csv") as file:
    reader = csv.reader(file, delimiter=",")
    for row in reader:
        for row_elem in row:
            str_floats = re.findall(r"[-+]?\d*\.\d+|\d+", row_elem)
            tmp = []
            for str_elem in str_floats:
                tmp.append(float(str_elem))
```

```
            geometric_params.append(tmp)

name_params = None
with open(case_path + "/Output_Data.csv") as file:
    reader = csv.DictReader(file, delimiter=",")
    for row in reader:
        name_params = row

filler_radius = geometric_params[0][0]
interface_thickness = geometric_params[1][0]
membrane_thickness = geometric_params[2][0]
membrane_width = geometric_params[3][0]

solubility = geometric_params[4][0]
diffusivity_matrix = geometric_params[5][0]
diffusivity_interface = geometric_params[6][0]
interface_prefactor = geometric_params[7][0]

matrix_mesh_size = geometric_params[8][0]
interface_mesh_size = geometric_params[9][0]

filler_center = [geometric_params[10][0], geometric_params[11][0]]

# ------------
# Abaqus Setup
# ------------

Mdb()
my_session = session
my_mdb = mdb

my_mdb.models.changeKey(fromName="Model-1", toName=name_params["model"])
my_model = my_mdb.models[name_params["model"]]

# -------------
# Part Creation
# -------------

sketch = my_model.ConstrainedSketch(name="__profile__", sheetSize=200.0)
sketch.rectangle(point1=(0, 0), point2=(membrane_width,
    membrane_thickness))
p1 = (filler_center[0] + filler_radius, filler_center[1])
sketch.CircleByCenterPerimeter(center=filler_center, point1=p1)
```

```
my_part = my_model.Part(name=name_params["part"], dimensionality=
    TWO_D_PLANAR, type=DEFORMABLE_BODY)
my_part.BaseShell(sketch=sketch)

sketch = my_model.ConstrainedSketch(name="__profile__", sheetSize=200.0)
p2 = (filler_center[0] + filler_radius + interface_thickness,
    filler_center[1])
sketch.CircleByCenterPerimeter(center=filler_center, point1=p2)
my_part.PartitionFaceBySketch(faces=my_part.faces, sketch=sketch)

# --------------
# Material Setup
# --------------

my_material = my_model.Material(name=name_params["material_matrix"])
my_material.Diffusivity(law=GENERAL, table=((diffusivity_matrix,),))
my_material.Solubility(table=((solubility,),))

# -------------
# Section Setup
# -------------

my_model.HomogeneousSolidSection(
    name=name_params["section_matrix"], material=name_params["
    material_matrix"], thickness=None
)

region_matrix = regionToolset.Region(faces=my_part.faces)

my_part.SectionAssignment(
    region=region_matrix,
    sectionName=name_params["section_matrix"],
    offset=0.0,
    offsetType=MIDDLE_SURFACE,
    offsetField="",
    thicknessAssignment=FROM_SECTION,
)

# ----------
# Mesh Setup
# ----------

elemType1 = mesh.ElemType(elemCode=DC2D4, elemLibrary=STANDARD)
elemType2 = mesh.ElemType(elemCode=DC2D3, elemLibrary=STANDARD)
```

```
my_part.setMeshControls(regions=my_part.faces, elemShape=QUAD, algorithm=ADVANCING_FRONT)
my_part.setElementType(
    regions=regionToolset.Region(faces=my_part.faces),
    elemTypes=(elemType1, elemType2),
)
my_part.seedPart(size=matrix_mesh_size, deviationFactor=0.1, minSizeFactor=0.1)

interface_edges = my_part.edges.getByBoundingBox(
    xMin=(filler_center[0] - filler_radius - interface_thickness - 1e-3),
    xMax=(filler_center[0] + filler_radius + interface_thickness + 1e-3),
    yMin=(filler_center[1] - filler_radius - interface_thickness - 1e-3),
    yMax=(filler_center[1] + filler_radius + interface_thickness + 1e-3),
)
my_part.Set(edges=interface_edges, name="SeedByEdge")
my_part.seedEdgeBySize(edges=interface_edges, size=interface_mesh_size)
my_part.generateMesh()

# --------------
# Assembly Setup
# --------------

my_assembly = my_model.rootAssembly
my_assembly.DatumCsysByDefault(CARTESIAN)

my_instance = my_assembly.Instance(name=name_params["part"] + "-1", part=my_part, dependent=ON)

# ---------
# Set Setup
# ---------

all_nodes = my_instance.nodes
top_nodes = my_instance.nodes.getByBoundingBox(
    xMin=-1e-3,
    yMin=membrane_thickness - 1e-3,
    zMin=0.0,
    xMax=membrane_width + 1e-3,
```

```
    yMax=membrane_thickness + 1e-3,
    zMax=0.0,
)
bot_nodes = my_instance.nodes.getByBoundingBox(
    xMin=-1e-3, yMin=-1e-3, zMin=0.0, xMax=membrane_width + 1e-3, yMax=1
    e-3, zMax=0.0
)

interface_nodes = my_instance.nodes.getByBoundingCylinder(
    center1=(filler_center[0], filler_center[1], 0 - 1e-3),
    center2=(filler_center[0], filler_center[1], 0 + 1e-3),
    radius=filler_radius + interface_thickness + 1e-3,
)

interface_elements = my_instance.elements.getByBoundingCylinder(
    center1=(filler_center[0], filler_center[1], 0 - 1e-3),
    center2=(filler_center[0], filler_center[1], 0 + 1e-3),
    radius=filler_radius + interface_thickness + 1e-3,
)

all_nodes_set = my_assembly.Set(nodes=all_nodes, name=name_params["
    set_All"])
interface_nodes_set = my_assembly.Set(nodes=interface_nodes, name=
    name_params["set_Interface_Nodes"])
interface_element_set = my_assembly.Set(nodes=interface_elements, name=
    name_params["set_Interface_Elements"])
inlet_set = my_assembly.Set(nodes=top_nodes, name=name_params["set_In"])
outlet_set = my_assembly.Set(nodes=bot_nodes, name=name_params["set_Out"
    ])


all_left_nodes = my_instance.nodes.getByBoundingBox(
    xMin=-1e-3,
    yMin=-1e-3,
    zMin=0.0,
    xMax=1e-3,
    yMax=membrane_thickness + 1e-3,
    zMax=0.0,
)
all_left_nodes_y_coord = []

all_right_nodes = my_instance.nodes.getByBoundingBox(
    xMin=membrane_width - 1e-3,
    yMin=-1e-3,
```

```
    zMin=0.0,
    xMax=membrane_width + 1e-3,
    yMax=membrane_thickness + 1e-3,
    zMax=0.0,
)
all_right_nodes_y_coord = []

for node in all_left_nodes:
    all_left_nodes_y_coord.append(node.coordinates[1])

left_idx = np.argsort(all_left_nodes_y_coord)

left_nodes = []
for idx in left_idx:
    for node in all_left_nodes:
        if node.coordinates[1] == all_left_nodes_y_coord[idx]:
            left_nodes.append(node.label)

for node in all_right_nodes:
    all_right_nodes_y_coord.append(node.coordinates[1])

right_idx = np.argsort(all_right_nodes_y_coord)

right_nodes = []
for idx in right_idx:
    for node in all_right_nodes:
        if node.coordinates[1] == all_right_nodes_y_coord[idx]:
            right_nodes.append(node.label)


node_sets = []
for ii in range(len(left_nodes)):
    right_name = "NR" + str(ii)
    left_name = "NL" + str(ii)

    my_assembly.SetFromNodeLabels(
        nodeLabels=(
            (
                name_params["part"] + "-1",
                (left_nodes[ii],),
            ),
        ),
        name=left_name,
    )
```

```
    my_assembly.SetFromNodeLabels(
        nodeLabels=(
            (
                name_params["part"] + "-1",
                (right_nodes[ii],),
            ),
        ),
        name=right_name,
    )
    node_sets.append([right_name, left_name])

# --------------
# Timestep Setup
# --------------

my_model.MassDiffusionStep(
    name=name_params["time_step"],
    previous="Initial",
    response=STEADY_STATE,
    amplitude=RAMP,
)
del my_model.historyOutputRequests["H-Output-1"]
my_model.fieldOutputRequests["F-Output-1"].setValues(variables=("MFL", "
    CONC", "COORD", "FV"))

# ---------
# Setup BCs
# ---------

inlet_region = regionToolset.Region(nodes=top_nodes)
my_model.ConcentrationBC(
    name=name_params["bc_In"],
    createStepName=name_params["time_step"],
    region=inlet_region,
    fixed=OFF,
    distributionType=UNIFORM,
    fieldName="",
    magnitude=1.0,
    amplitude=UNSET,
)

outlet_region = regionToolset.Region(nodes=bot_nodes)
my_model.ConcentrationBC(
    name=name_params["bc_Out"],
```

```
    createStepName=name_params["time_step"],
    region=outlet_region ,
    fixed=OFF,
    distributionType=UNIFORM,
    fieldName="",
    magnitude=0.0,
    amplitude=UNSET,
)

# ---------
# Setup Constraints
# ---------

for ii, node_set in enumerate(node_sets):
    right_name = node_set[0].upper()
    left_name = node_set[1].upper()
    my_model.Equation(
        name="eq" + str(ii),
        terms=((1.0, right_name, 11), (-1.0, left_name, 11)),
    )


# ---------
# Setup Job
# ---------

job_name = "Standart__" + case
my_job = my_mdb.Job(
    name=job_name,
    model=name_params["model"],
    description="",
    type=ANALYSIS,
    atTime=None,
    waitMinutes=0,
    waitHours=0,
    queue=None,
    memory=90,
    memoryUnits=PERCENTAGE,
    getMemoryFromAnalysis=True,
    explicitPrecision=SINGLE,
    nodalOutputPrecision=SINGLE,
    echoPrint=OFF,
    modelPrint=OFF,
    contactPrint=OFF,
```

```
    historyPrint=OFF,
    userSubroutine="",
    scratch="",
    resultsFormat=ODB,
    numThreadsPerMpiProcess=1,
    multiprocessingMode=DEFAULT,
    numCpus=2,
    numDomains=2,
    numGPUs=1,
)

my_job.submit()
my_job.waitForCompletion()

# ---------
# Output
# ---------

my_session = session
my_odb = my_session.openOdb(name=job_name + ".odb")
frame = my_odb.steps[name_params["time_step"]].frames[1]
instance = my_odb.rootAssembly.instances[name_params["part"].upper() + "-1"]

set_region = my_odb.rootAssembly.nodeSets[name_params["set_Out"].upper()]
mfl_data = frame.fieldOutputs["MFL"].getSubset(region=set_region, position=ELEMENT_NODAL).values

data_list = []
for index in range(len(mfl_data)):
    data_list.append(
        [
            instance.getNodeFromLabel(mfl_data[index].nodeLabel).coordinates[0],
            instance.getNodeFromLabel(mfl_data[index].nodeLabel).coordinates[1],
            mfl_data[index].data[1],
        ]
    )
np.savetxt(
    "../RESULTS/Result_" + job_name + ".csv",
    data_list,
    delimiter=", ",
```

```
    fmt="% s",
    header="x, y, mfl_2",
)
```

A.2.4 Postprocessing

```
# Imports
import os
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as sci

plt.rc("font", size=18)  # controls default text sizes
plt.rc("legend", fontsize=10)  # legend fontsize
plt.rc("figure", titlesize=18)  # fontsize of the figure title
plt.rc("axes", labelsize=18)  # fontsize of the x and y labels

# PCCL-Colors
PCCL_main = tuple(np.array([0.0, 125.0, 166.0]) / 255.0)
PCCL_lightblue = tuple(np.array([1.0, 174.0, 240.0]) / 255.0)
PCCL_darkgreen = tuple(np.array([56.0, 181.0, 77.0]) / 255.0)
PCCL_lightgreen = tuple(np.array([153.0, 202.0, 59.0]) / 255.0)
PCCL_yellow = tuple(np.array([232.0, 196.0, 56.0]) / 255.0)
PCCL_orange = tuple(np.array([238.0, 138.0, 44.0]) / 255.0)
PCCL_red = tuple(np.array([230.0, 47.0, 41.0]) / 255.0)
PCCL_purple = tuple(np.array([131.0, 42.0, 132.0]) / 255.0)
PCCL_violet = tuple(np.array([92.0, 73.0, 152.0]) / 255.0)
PCCL_darkblue = tuple(np.array([0.0, 102.0, 176.0]) / 255.0)

pccl_colors = [
    PCCL_main,
    PCCL_red,
    PCCL_darkgreen,
    PCCL_orange,
    PCCL_purple,
    PCCL_darkblue,
]

linestyles = {
    "loosely dotted": (0, (1, 10)),
    "dotted": (0, (1, 1)),
    "densely dotted": (0, (1, 1)),
```

```
    "long dash with offset": (5, (10, 3)),
    "loosely dashed": (0, (5, 10)),
    "dashed": (0, (5, 5)),
    "densely dashed": (0, (5, 1)),
    "loosely dashdotted": (0, (3, 10, 1, 10)),
    "dashdotted": (0, (3, 5, 1, 5)),
    "densely dashdotted": (0, (3, 1, 1, 1)),
    "dashdotdotted": (0, (3, 5, 1, 5, 1, 5)),
    "loosely dashdotdotted": (0, (3, 10, 1, 10, 1, 10)),
    "densely dashdotdotted": (0, (3, 1, 1, 1, 1, 1)),
}

result_directory = "RESULTS/"
file_names = []
for file in os.listdir(result_directory):
    if file.endswith(".csv"):
        if "Outflow" not in file:
            file_names.append(file)

data_arr = []
for file_name in file_names:
    data_arr.append(np.genfromtxt(result_directory + file_name,
    delimiter=","))

plot_list = []
outflow_data = []
for idx, data in enumerate(data_arr):
    pos_arr = np.array([item[0] for item in data])
    mfl_2_arr = np.array([item[2] for item in data])

    sorted_pos_arr, sort_idx = np.unique(pos_arr, return_index=True)
    sorted_mfl_2_arr = np.zeros(sorted_pos_arr.size)
    for unique_idx, unique_pos in enumerate(sorted_pos_arr):
        value_idx = np.where(pos_arr == unique_pos)
        sorted_mfl_2_arr[unique_idx] = np.mean(mfl_2_arr[value_idx])

    integrated_value = sci.simpson(sorted_mfl_2_arr, sorted_pos_arr)
    membrane_length = sorted_pos_arr[-1] - sorted_pos_arr[0]
    outflow_value = integrated_value / membrane_length * -1.0

    plot_label = (
        file_names[idx]
        .replace(".csv", "")
        .replace("Result_Job__", "")
```

```
        .replace("FS_5-0__2-0__IT_0-5__DM_1-0__", "")
        .replace("PF_", "psi: ")
        .replace("DI_", "D: ")
        .replace("-", ".")
        .replace("__", ", ")
    )
    if "Standart" in plot_label:
        plot_label = "No Interface"

    outflow_data.append([plot_label, outflow_value])
    plot_list.append([plot_label, sorted_pos_arr, sorted_mfl_2_arr])

plt.figure()
plt.xlabel("Membrane Width")
plt.ylabel("Outflow over Membrane Width")

for plot_data in plot_list:
    plot_label = plot_data[0]
    plot_color = pccl_colors[1]
    plot_linestyle = "-"

    if "psi: 75" in plot_label:
        plot_linestyle = linestyles["dashed"]
    elif "psi: 50" in plot_label:
        plot_linestyle = linestyles["dashdotted"]
    elif "psi: 20" in plot_label:
        plot_linestyle = linestyles["dotted"]
    elif "psi: 5" in plot_label:
        plot_linestyle = linestyles["dashdotdotted"]

    if "D: 10.0," in plot_label:
        plot_color = pccl_colors[0]
    elif "D: 1.0," in plot_label:
        plot_color = pccl_colors[2]
    elif "D: 0.1," in plot_label:
        plot_color = pccl_colors[3]
    elif "D: 0.01," in plot_label:
        plot_color = pccl_colors[4]
    elif "D: 0.001," in plot_label:
        plot_color = pccl_colors[5]

    plt.plot(
        plot_data[1],
        plot_data[2],
```

```
        label=plot_label,
        color=plot_color,
        linestyle=plot_linestyle,
    )

plt.legend()
# plt.tight_layout()

plt.figure()
plt.xlabel("Membrane Width")
plt.ylabel("Outflow over Membrane Width")
for plot_data in plot_list:
    if "No Interface" in plot_data[0] or "psi: 20" in plot_data[0]:
        plot_label = plot_data[0]
        plot_color = pccl_colors[1]
        plot_linestyle = "-"

        if "psi: 75" in plot_label:
            plot_linestyle = linestyles["dashed"]
        elif "psi: 50" in plot_label:
            plot_linestyle = linestyles["dashdotted"]
        elif "psi: 20" in plot_label:
            plot_linestyle = linestyles["dotted"]
        elif "psi: 5" in plot_label:
            plot_linestyle = linestyles["dashdotdotted"]

        if "D: 10.0," in plot_label:
            plot_color = pccl_colors[0]
        elif "D: 1.0," in plot_label:
            plot_color = pccl_colors[2]
        elif "D: 0.1," in plot_label:
            plot_color = pccl_colors[3]
        elif "D: 0.01," in plot_label:
            plot_color = pccl_colors[4]
        elif "D: 0.001," in plot_label:
            plot_color = pccl_colors[5]

        plt.plot(
            plot_data[1],
            plot_data[2],
            label=plot_label,
            color=plot_color,
            linestyle=plot_linestyle,
        )
```

```python

plt.legend()
# plt.tight_layout()
plt.savefig("Outflows_Rect_Small_New.png", bbox_inches="tight")


plt.figure()

x_positions = [1, 2, 3, 4]
x_axis_labels = ["5", "20", "50", "75"]
plt.xlabel("psi")
plt.xticks(x_positions, x_axis_labels)

plt.ylabel("Sum of Outflow over Membrane Width")

for elem in outflow_data:
    plot_label = elem[0]
    mark_style = "o"
    plot_color = pccl_colors[1]
    x_pos = 0

    if "psi: 75" in plot_label:
        x_pos = 4
    elif "psi: 50" in plot_label:
        x_pos = 3
    elif "psi: 20" in plot_label:
        x_pos = 2
    elif "psi: 5" in plot_label:
        x_pos = 1

    if "D: 10.0," in plot_label:
        plot_color = pccl_colors[0]
    elif "D: 1.0," in plot_label:
        plot_color = pccl_colors[2]
    elif "D: 0.1," in plot_label:
        plot_color = pccl_colors[3]
    elif "D: 0.01," in plot_label:
        plot_color = pccl_colors[4]
    elif "D: 0.001," in plot_label:
        plot_color = pccl_colors[5]

    if not "No Interface" in plot_label:
        plt.plot(x_pos, elem[1], color=plot_color, markersize=4, label=
    plot_label, marker=mark_style, linewidth=0)
```

```
    else:
        plt.hlines(xmin=1, xmax=4, y=elem[1], color=plot_color, label="No Interface")


plt.legend()
# plt.tight_layout()
plt.show()
```