Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

A Discontinuous Galerkin Model of the Channel Flow and Gas Diffusion Layer in a PEMFC

Michel Takken

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

A Discontinuous Galerkin Model of the Channel Flow and Gas Diffusion Layer in a PEMFC

Author:	Michel Takken
1st examiner:	Univ.-Prof. Dr. Michael Bader
Assistant advisor:	Lukas Krenz, M.Sc.
Submission Date:	April 16th, 2021

Declaration for the master's thesis

Name: Michel Takken

Student ID Number (Matrikelnummer): 03711781

Supervisor: Lukas Krenz, M.Sc.

Examiner: Univ.-Prof. Dr. Michael Bader

Title: A Discontinuous Galerkin Model of the Channel Flow and Gas Diffusion Layer in a PEMFC

I, Michel Takken, confirm that the version of my thesis I send to coordinators@cse.tum.de on April 16th, 2021 is the final version of my thesis that is to be graded.

With the submission of my master's thesis I confirm that this thesis is my own work and I have documented all sources and material used.

April 16th, 2021 Michel Takken

Acknowledgments

This thesis is the product of years of studying and hard work that have been made possible through the support and insiration that I have received during this period. My special thanks go out to Professor Shohji Tsushima of Osaka University, who greatly inspired me to develop a passion for the computational modeling of polymer electrolyte membrane fuel cells during a university exchange program, and to Professor Pim Groen of the TU Delft, who supervised and supported me during my Bachelor End Project to follow my ambition to study Computational Science and Engineering. This thesis could have never taken the form it has now without the supervision of Lukas Krenz, M.Sc. and I am very thankful for his support. Finally, I would like to thank my friends and family, who supported me through my studies, and everyone who has proofread or given comments in any form on my thesis. It would have been a different journey without any of you.

Abstract

Due to a global shift towards renewable and emission-free energy sources, the need for electrical energy systems is increasing. Fuel cells are important energy systems, as they can convert chemical energy of hydrogen and oxygen into electrical energy, without expelling any environmentally harmful emissions. Especially in electric vehicles and electric aircraft, the combination of a fuel cell system with a hydrogen tank has the potential to become more relevant than batteries, as battery-systems carry a high gravimetric density and scale poorly volumetrically. For a specialized desing of a fuel cell system for vehicles or aircraft, a good understanding of the underlying physics is required. This understanding can be achieved through accurate computational modeling of such systems. The goal of this thesis is to simulate the free flow and porous flow regions of a Polymer Electrolyte Membrane Fuel Cell (PEMFC) using a discontinuous Galerkin approach in Julia. Starting with an already existing implementation of a discontinuous Galerkin simulation of the Euler equations, the computation of the compressible Navier Stokes equations (NSE) is included. These equations require an additional numerical treatment to ensure a stable simulation. Hence, the interior penalty method is implemented for the computation of elliptic partial differential problems. Afterwards, the compressible NSE are coupled with the species equation to obtain the distribution of a chemical species in the flow. To obtain the Darcy-Brinkman-Frorchheimer model for a reactive flow through a porous medium, the simulation is then extended with source terms. The implementation of the compressible NSE is verified for the Taylor-Green problem and the lid-driven cavity flow problem, after which the simulation is used for the evaluation of a gas diffusion layer configuration in a PEMFC.

Contents

Part I.

Introduction and Background Theory

1. Introduction

Renewable energy sources, such as solar panels and wind turbines, are a green alternative to the more conventional energy sources, such as gas and oil, to provide energy to our homes and offices. Governments and companies worldwide are working towards a greener energy economy, and even our cars and most of public transportation have shifted towards greener energy sources, such as batteries [1]. Batteries, however, have a limited range of application, because volumetrically they scale poorly with the amount of energy that is stored and they are heavy in weight. This can surely be overcome by improved battery technologies, which could be developed, but the reality is that it is hard to, e.g. store energy peaks, that are produced by wind farms or solar parks, in batteries. Electric vehicles need a significant amount of time to recharge, before they have a decent range. The development of fully electrical aircraft is still in the early stages and the use of batteries to power an aircraft is challenging due to the density of battery-systems [5].
The problems that are introduced by batteries are relatively easily solved by using a fuel cell system that runs on, e.g. hydrogen. Fuel cells are electrochemical cells that work almost like batteries, but their volumetric and gravimetric scaling is significantly better [26], not to mention that they consume a fuel which can be replenished quickly. Furthermore, energy peaks from renewable energy sources can be stored in hydrogen through electrolysis, making fuel cells a relevant alternative to batteries in stationary applications such as solar farms and wind parks. A fuel cell can be seen as an "electrical energy factory", just like a battery, but the energy source is an external fuel, instead of internal chemicals with electric potential.
In aircraft applications, fuel cells are an especially interesting energy source, because all harmfull emissions of an aircraft could theoretically be removed. This is because fuel cells can be designed to only produce water during the conversion of chemical energy into electrical energy, provided that hydrogen is used as a fuel. There is still room for improvement in fuel cell design, when it comes to specialized applications. In aviation applications, fuel cells can still be improved considering the operating conditions and material science, such that they operate in harmony with the rest of the aircraft subsystems. To have a functional aircraft flying on electricity, a fuel cell's energy production must be on the same level as the aircrafts energy consumption. On the other hand, to have a functional fuel cell system, the operating conditions of the fuel cell (e.g. operating temperature and pressure) and the supply of hydrogen and air, must be on a level that is required by the fuel cell. Research must still be done to get a better understanding of this interface of a fuel cell subsystem and the rest of an aircraft, and to get a better understanding of the potential of fuel cell systems in aircraft. Research that is not only costly and time-consuming, but often also

very difficult. The inner workings of a fuel cell are complex and not easily predicted. A computational model could provide us with answers that experimental research often cannot.
Good simulations of a fuel cell are thus necessary for a properly functional fuel cell system that provides energy to an aircraft. Recently developed fuel cell models rely on finite element or finite volume discretization methods [12]. The underlying discontinuous properties of a fuel cells computational domain make it difficult to get convergent computational solutions [30]. This underlines the importance of the exploration of various computational methods for the simulation of fuel cells.
In this thesis, the discontinuous Galerkin approach for the modeling of a fuel cell will be discussed. The discontinuous Galerkin method (DGM) has revived a lot of interest in many computational scientists, after it was long neglected since its first application in the 70s by Reed & Hill [27]. We will make use of the discontinuous basis of the method and the benefits of high-order basis functions, to construct a fuel cell model. The goal of this thesis is to provide the reader with the proper numerical means to realise this model. In chapter 2, the fundamentals of a fuel cell and the mathematical model that is commonly used will be explained. Chapter 3 will present the mathematical tools that are necessary for the DGM. Afterwards, the discretization of the compressible Navier-Stokes equations and fuel cell model using the DGM will be examined in chapters 4 and 5 respectively. The discretization of the compressible Navier-Stokes equations is treated separately from the fuel cell model, because our numerical solution of the Navier-Stokes equations will be verified to a certain extend. The computational model will be programmed in Julia and the numerical implementation is discussed in chapter 6, after which a verification of the implementation for the compressible Navier-Stokes equations is given in chapter 7. In chapter 8 we will conclude with results and we will reflect on recommendations that can be made for any future work done on the model, after which this thesis is concluded in chapter 9.

2. Fuel Cell Theory

A fuel cell can be described as an electrochemical system that converts chemical energy into electrical energy, similar to a battery. The difference being that a battery is a closed system, with internal chemicals, and a fuel cell uses external chemicals, i.e. a fuel, to generate electricity. This external fuel, is stored in a tank and can be refilled whenever necessary. The main advantages of chosing fuel cells over conventional combustion engines are that water is the only by-product of the energy conversion (assuming that hydrogen is used as a fuel) and that the energy conversion is more efficient. This efficiency is due to the fact that a combustion engine creates a lot of useless energy in the form of heat when it converts internal energy of chemicals to useful work (in the thermodynamic sense). Fuel cells, on the other hand, generate useful work (in the form of electric energy) directly from the internal energy of chemicals, leading to an energy conversion that can be as efficient as up to 90%. This is significantly better than the 25-50% for combustion engines [26].
More importantly, since fuel cells can convert energy without generating harmful emissions, like carbon-/nitrogen-oxides, they can be a key element of the solution to the climate change problem.
In this chapter, we will discuss the background knowledge of fuel cells that is required to understand the principles of the model that was programmed during this thesis. Fuel cells are complicated systems and there are many variations to them. In section 2.1 we will go over the different types of fuel cells, explain why the polymer electrolyte membrane fuel cell (PEMFC) was chosen to be modelled and analyse the internal workings of a PEMFC in more detail. Afterwards, the commonly used mathematical model of a PEMFC is presented in section 2.2. Finally, we will discuss the performance analysis of a PEMFC in section 2.3.

2.1. PEMFC

Before we start discussing the possible fuel cell types, we should have a look at the basic mechanism of a fuel cell. A fuel cell essentially consists of the following four components: An electrolyte, electrodes, reactants and a wire. In fig. 2.1 a basin is depicted containing an electrolyte in liquid form. The two red rods are the electrodes. The reactants are hydrogen (H_2) on the left and oxygen (O_2) on the right (here represented by gas bubbles attached to the electrodes), and the wire is the line on top that connects the two electrodes. The complete electrochemical reaction that happens in a simple fuel cell can be divided into two half-reactions. These two half-reactions are the hydrogen *oxidation* reaction (HOR), where

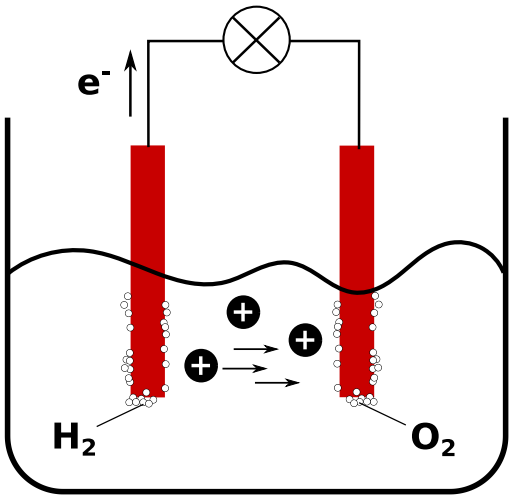

Figure 2.1.: A simplified schematic representation of a fuel cell inspired by an image by O'Hayre et al. [26]. The liquid in the basin is the electrolyte, and the two rods are the anode (left) and cathode (right) electrodes. The reactants, hydrogen and oxygen, are present as gas bubbles near the electrodes. Electrons flow through the wire connecting the two electrodes on top. The crossed circle represents a lamp that uses the electrical energy. The protons flow through the electrolyte.

electrons are removed from the reactant, and the oxygen *reduction* reaction (ORR), where electrons are added to the reactant. These half-reactions occur at the two electrodes that are spatially separated by the electrolyte. This spatial separation is necessary to force the electrons to flow through the connecting wire, such that the energy carried by the electrons can be harvested by, e.g. a lamp (represented by the crossed circle in fig. 2.1). To complete the full reaction, the transport of protons (or another protonic charge carrier) from one electrode to the other is also needed. The electrolyte allows a flow of protons from one electrode to the other, but enforces the electrons to flow through the wire. The electrode of the oxidation half-reaction, is called the *anode*, and the electrode of the reduction half-reaction is called the *cathode*.
The two half-reactions are in our case a split version of the hydrogen combustion reaction:

$$\text{Oxidation reaction:} \qquad H_2 \rightleftharpoons 2H^+ + 2e^-$$

$$\text{Reduction reaction:} \qquad \frac{1}{2}O_2 + 2H^+ + 2e^- \rightleftharpoons H_2O$$

In these reactions the reactants are hydrogen and oxygen, and the product is water. As long as there are no carbon or nitrogen parts present in the reaction, no carbon- or nitrogenoxides (greenhouse gasses) are produced. It must be noted that this not always a given for the combustion of hydrogen, because the ambient air contains nitrogen and car-

Table 2.1.: Operation conditions of major fuel cell types as given by O'Hayre et al. [26].

	PEMFC	PAFC	AFC	MCFC	SOFC
Operating temperature (°C)	80	200	60-220	650	600-1000
Fuel compatibility	H, methanol	H	H	H,CH	H, CH, CO

bon. For a set of combustion stoichiometries, hydrogen may not be the only reactant in the combustion and environmentally harmful gasses can be produced. This is not the case for fuel cells, although the electrolyte itself might get polluted. This leads to the degradation of the fuel cell performance [26]. Electrolyte pollution can be avoided to a certain extend by proper fuel cell design.

2.1.1. Fuel Cell Choice

Fuel cells can have different structures and materials, but the main idea is the same. The choice for the electrodes, electrolyte and reactants has a huge impact on the performance of a fuel cell and its required operating conditions. This thesis will cover the model of a polymer electrolyte membrane fuel cell (PEMFC), which uses a polymer membrane as an electrolyte.
The PEMFC is chosen for this thesis, because it is a prevailing fuel cell type in mobile applications, such as vehicles and aircraft [9, 10]. This is mainly due to its efficiency and relatively low operating temperature range of 80-100 °C. For sake of completeness, four other major fuel cell types given by O'Hayre et al. [26] are listed below with the PEMFC:

- Polymer electrolyte membrane fuel cell (PEMFC)
- Phosporic acid fuel cell (PAFC)
- Alkaline fuel cell (AFC)
- Molten carbonate fuel cell (MCFC)
- Solid-oxide fuel cell (SOFC)

From table 2.1 it is clear that the operating temperature of the PEMFC makes it attractive for mobile applications. Another advantage of PEMFCs is their high power density and their low weight, making it the most interesting among fuel cells for aircraft application. Altough methanol can also be used as a fuel in a PEMFC, we will assume hydrogen as a fuel.

2.1.2. Inner Workings of a PEMFC

The schematic inner working of a PEMFC is illustrated in fig. 2.2. From left to right, the PEMFC consists of the following seven layers:

- Anode flow channel
- Anode gas diffusion layer (GDL)
- Anode atalyst layer (CL)
- The electrolyte membrane
- Cathode gas diffusion layer (GDL)
- Cathode catalyst layer (CL)
- Cathode flow channel

Together with the CLs on both anode and cathode side, the electrolyte membrane forms the membrane electrode assembly (MEA). Chemical half-reactions only occur at the triple points, where the reactant meets the catalysator and electrolyte. The function of each of these layers will now be examined.

Flow Channels

In fig. 2.1 the reactants are simply illustrated by gas bubbles, collected at the electrodes. This is not a realistic illustration, because no source for these reactants is illustrated. The flow channels act as a riverbed for the flow of (mainly gaseous) fluids that contain the reactants. In fig. 2.2 the flow channels are the two outmost layers that contain reactants and products, characterized by the illustrated inflows on the bottom and outflows on the top. The main function of the flow channel is to convect the reactants, with the objective that the reactants be spread out over the electrolyte sheet as uniformly as possible. During the electrochemical reaction, reactants are depleted and the reaction product, H_2O in our case, can "choke" the fuel. Both the depletion of reactants and choking of the fuel cell affect the performance of a fuel cell drastically. This leaves an important job for the flow channels. Namely, the guiding of the reactants and products such that neither depletion nor choking happens. Pressure differences at the beginning and ending of the anode and cathode flow channels, and the choice of the pattern of the flow channels greatly influences the convection of reactants and products. Examples of flow patterns are given in fig. 2.3.

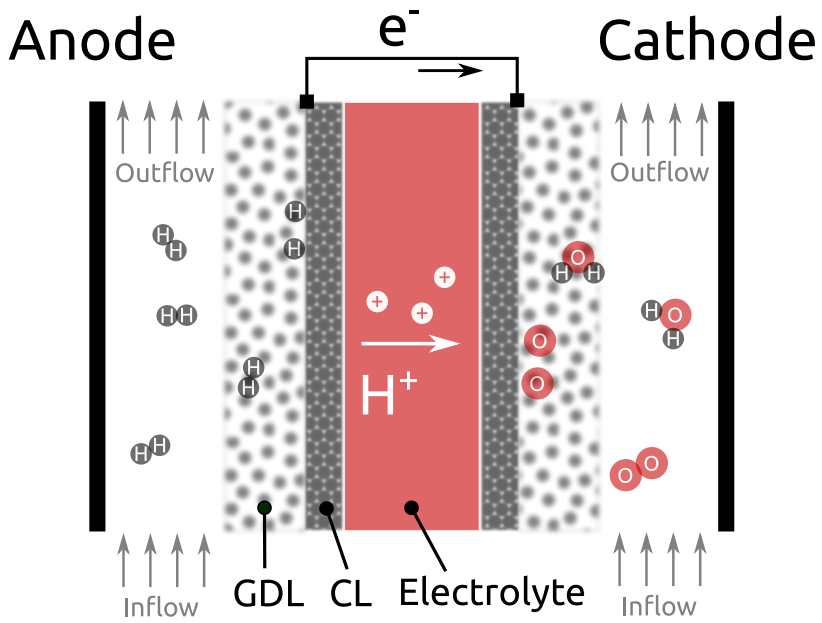

Figure 2.2.: Schematic 2D representation of a PEMFC. From the sides to the center, this figure portrays the flow channels, gas diffusion layers (GDL), catalyst layers (CL) and polymer electrolyte membrane are portrayed. The anode side is on the left and the cathode side is on the right. The electrodes are connected by a wire on top.

Gas Diffusion Layers

The GDLs are layers of porous material that transport reactants from the flow channels to the MEA, and products from the MEA back to the flow channel. This layer is essential, because in a regular flow channel the velocity profile shows a stagnant flow at the walls and the highest flow velocities are reached at the biggest distance from the walls. This is especially true for a laminar flow, which is the dominating flow type inside a fuel cell [26]. To prevent the reactants from flowing past the MEA, without any reactions occurring, the GDL is introduced. In this region, the diffusion of the flow dominates. This means that reactants are transported to regions where they are in low concentration near the electrolyte, where the reactants are depleted in electrochemical reactions and products are transported from the electrolyte, where it is high in concentration, to the gas channel, where it is low in concentration.

Catalyst Layers

The CL is a very thin layer of catalyst material applied to the GDL to encourage the electrochemical reaction by lowering the activation potential of the half-reactions. The most

common and most effective catalyst for a PEMFC is platinum [26]. Its presence lowers the activation energy for the electrochemical reaction, thus it improves the efficiency of the fuel cell (see section 2.3 for activation losses).

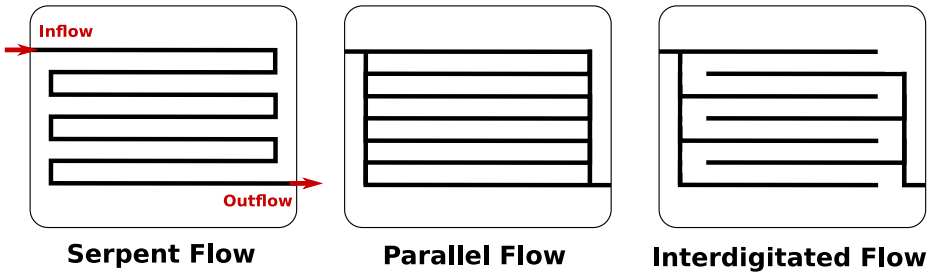

Figure 2.3.: Common flow pattern examples for PEMFCs are: Serpent, parallel and interdigitated flow [26]. Fluids flow along the black lines. For the interdigitated flow, the fluid will "cross" out of plane through the GDL, which can be imagined to be stacked on top of the flow pattern. The fluids flow in at the upper left corner and out at the bottom right corner.

Polymer Electrolyte Membrane

The polymer electrolyte membrane is the electrolyte for the PEMFC. For the polymer membrane usually Nafion-117 is used. Its function is to transport protons from the anode side to the cathode side of the fuel cell. The performance of Nafion-117 is greatly affected by the operating temperature and the water saturation level of the membrane [26].

2.2. M2 Model

The physics of a PEMFC involve many complicated phenomena, such as electrochemical reactions, current distribution, two-phase flow transport and heat transfer. These simultaneously occurring processes can be described by a mathematical model. Recently, the multiphase mixture (M^2) model has been widely used for modeling PEMFCs [30]. Although we will be using a simplified version of the M^2 model, it is best to explain the complete model such that the simplifications made later will be better understood.
A disadvantage of using the M^2 model is that a discontinuous diffusion coefficient is introduced in the water concentration equation due to the existence of two distinct regions of single-phase and two-phase flow, resulting in a non-convergent, oscillatory, nonlinear iteration for continuous Galerkin discretizations [30]. Just as for the finite volume method, a motivation for choosing a discontinuous Galerkin approach for the discretization of the model is that this discontinuity in the diffusion coefficient can be captured and handled properly due to the discontinuous nature of this method. Additionally, most benefits of the continuous Galerkin method, such as the use of high-order basis functions are kept.

The model presented below is based on the reference paper by Sun [30]. This paper considers a steady-state case, where the back-diffusion of water in the membrane is neglected. We also choose to neglect the back-diffusion of water, but in contrast to Sun [30], we will be considering a time-dependent problem. We do this because the model could then be developed further for complex situations such as cold-start simulation [24].
The M^2 model is based on the Navier-Stokes equations for flow, with some modifications known as the Darcy-Brinkman-Forchheimer model [20, 30]. For a two-phase model, the physical parameters and coefficients are defined through two-phase mixture relations. Therefore, properties like Prandtl's number, the heat capacity ratio and the thermal conductivity coefficient are defined as functions of the fraction of open pore space occupied by liquid and gas phases. The explicit definitions of the two-phase relations are given by Sun [30]. For us, it is important to realise that these values, which are usually regarded as constants, are actually variables.
The model will first be presented as it was given by Sun [30], without the time derivative. The time derivative will be added to the model later in this thesis.

Mass Equation

The mass equation is the same as for the standard compressible Navier-Stokes equations and is given by

$$\nabla \cdot (\rho \boldsymbol{v}) = 0. \tag{2.1}$$

Here, the density is given by ρ and the velocity vector is given by $\boldsymbol{v}$.

Momentum Equation

The momentum equation is defined as

$$\frac{1}{\varepsilon^2}\nabla \cdot (\rho \boldsymbol{v}\boldsymbol{v}) = -\nabla p + \nabla \cdot (\mu \nabla \boldsymbol{v}) + S_u, \tag{2.2}$$

where ε is the porosity of the porous media. The porosity of a media is the ratio of open pore space divided by the total space. In the flow channels, porosity is equal to one. The pressure and viscosity are denoted by p and μ respectively. The first term on the left-hand side is the advection term, the first term on the right-hand side is the pressure term and the second term on the right-hand side is the diffusion term. The source term for the momentum equation is denoted by S_u. It is zero in the flow channels and it is equal to the Darcy force in the porous regions. The source term functions as a sink to the momentum of the flow due to the porosity of the medium. Notice that this equation becomes the compressible Navier-Stokes momentum equation in the flow channels.

Species Equation

The general species equation must be solved for each species $k \in \{H_2, O_2, H_2O\}$, resulting in three separate equations. The general species transport equation is given by

$$\nabla \cdot (\gamma_c \boldsymbol{v} C^k) = \nabla \cdot (D_g^{\mathrm{k,eff}} \nabla C_g^k) - \nabla \cdot \left[\left(\frac{m f_l^k}{M^k} - \frac{C_g^k}{\rho_g} \right) \boldsymbol{j}_l \right] + S_k. \tag{2.3}$$

This equation describes the movement of a chemical species (H_2, O_2 or H_2O) in the flow. The total mixture molar concentration of a species is given by C^k. An advection correction factor, γ_c, is necessary to correct the advection for two-phase flow regions. The effective gas diffusion coefficient is given by $D_g^{\mathrm{k,eff}}$, M^k is the molar weight and f_l^k is the liquid mass fraction. Solving this equation is necessary to obtain the concentration of a species in the catalyst region where the electrochemical reaction happens. The term on the left-hand side describes the advection of a species along the flow. The first term on the right-hand side models the diffusivity of a species in the flow. The diffusivity becomes dominant in the diffusivity region, so it should not be neglected. The second term on the right-hand side describes the relative motion of species due to the capillary action in the porous regions [30]. The evaluation of the capillary-diffusional flux of the liquid phase, $\boldsymbol{j}_l$, is quite complicated, but can be summarized as a function of the capillary pressure of the porous material, which is in turn evaluated by the Leverett function [32]. For a more explicit definition of $\boldsymbol{j}_l$, the reader is referred to the reference paper by Sun [30]. The source term of the species equation, S_k, describes the creation and depletion of a species due to electrochemical reactions and for water it additionally models the change of concentration due to electro-osmotic drag [30].

Energy Equation

The energy equation is given by

$$\nabla \cdot (\gamma_T \rho c_p \boldsymbol{v} T) = \nabla \cdot (k \nabla T) + S_T, \tag{2.4}$$

where the advection heat-transfer correction is denoted by γ_T, c_p is the heat capacity at constant pressure, T is the temperature and k is the thermal conductivity. The term on the left-hand side describes the transfer of energy by convection, the first term on the right-hand side describes the energy transfer due to heat conduction and the second term on the right-hand side is the energy source term S_T. The energy source term includes a few components that may affect the energy in the system. These are: Energy change due to electrochemical reactions, heat change due to water condensation and evaporation, and heat from electronic or protonic resistances.

Proton and Electron Transport

With the proton and electron transport equations, the proton potential Φ_e and electron potential Φ_s can be solved, and the model is concluded. These potentials are the last unknowns that need to be solved, in order to predict the reaction rates of the electrochemical reactions in the system. The proton transport equation is given by

$$\nabla \cdot (\kappa \nabla \Phi_e) + S_{\Phi_e} = 0 \tag{2.5}$$

and the electron transport equation is given by

$$\nabla \cdot (\sigma \nabla \Phi_s) + S_{\Phi_s} = 0. \tag{2.6}$$

The proton transport equation exists in the MEA region, where κ is the protonic conductivity of the material and the source term S_{Φ_e} is defined by the amount of electrochemical reactions. The electron transport equation has the same shape and it is present in the catalyst layers and the current collectors (wire and lamp). The conductivity σ is the electronic conductivity in the material, and the source term $S_{\Phi_s} = -S_{\Phi_e}$ is also regulated by the electrochemical reactions.

2.3. Current Density Graph

The complexity of a fuel cell follows from the model that was given in section 2.2. For engineering applications, it is important that the fuel cell performance can be evaluated simply. A common way to describe the performance of a fuel cell is through the current-density. The fuel cell performance scales linearly with the area of the electrolyte, meaning that a fuel cell with a twice as large area can provide a twice as big current, provided that there is an adequate supply of reactants. Therefore, to compare various fuel cells, the performance is usually measured in current-density. The current-density is the generated current in ampères divided by the electrolyte surface

$$\text{Current-density} = \frac{\text{Generated current (A)}}{\text{Electrolyte surface (cm}^2\text{)}}. \tag{2.7}$$

The generated current-density of a fuel cell depends on the voltage at which the fuel cell operates. This dependency is usually illustrated by a current-density curve and an example is given in fig. 2.4. Multiplying the current with the corresponding voltage results in the power-density curve (not illustrated), which can be used to determine the required fuel cell configuration and to obtain the maximum available power, a value that is relevant for the design of electronic systems. The characteristic shape of the current-density curve originates from three separate losses that occur during operation. These losses are the activation loss, ohmic loss and concentration loss. The height of the current-density curve is determined by the thermodynamically defined optimal energy conversion (E_0 in fig. 2.4),

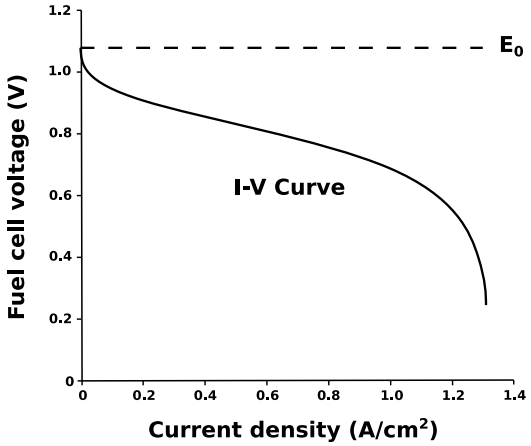

Figure 2.4.: Simplified reference current-density (i-V) curve for a PEM fuel cell inspired by an image by O'Hayre et al. [26].

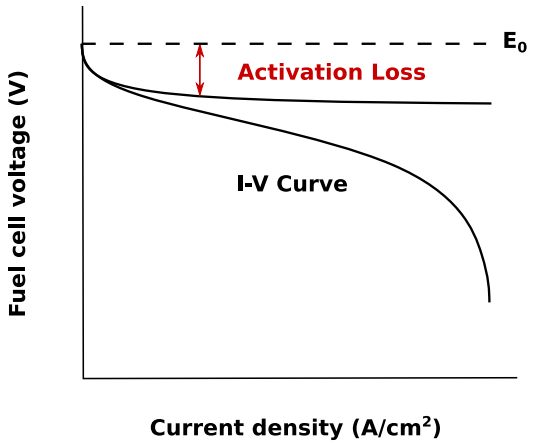

Figure 2.5.: The activation loss of a PEM fuel cell inspired by an image by O'Hayre et al. [26].

based on the change in internal energy of the reactants due to the electrochemical reaction. The complete current-density curve is given by

$$\text{Current-density} = E_0 - \text{Activation Loss} - \text{Ohmic Loss} - \text{Concentration Loss}.$$

The thermodynamic optimum and the three losses can be predicted using a fuel cell model. In this section we will discuss why each of the losses occur and give the equations to predict the thermodynamic optimal voltage and the three current losses.

2.3.1. Thermodynamic Optimum

With thermodynamics, the maximum amount of energy that can be obtained from an electrochemical reaction can be defined, as it yields the theoretical boundaries of what is possible with a fuel cell [26]. The Gibbs free energy describes the maximum electrical energy that can be obtained from an electrochemical reaction and it is calculated using the follownng equation

$$E_0 = -\frac{\Delta g}{nF}. \tag{2.8}$$

Here, Δg is the difference in Gibbs free energies of the reactants and products of a chemical reaction, n is the amount of electrons that are transferred in a chemical reaction and F is Faraday's constant. The difference in Gibbs free energies is a function of temperature and pressure, so the available potential energy depends on the operating pressures and temperature of a fuel cell. A fuel cell operating at room temperature on 3 atm pure H_2 and 5 atm air, has a predicted reversible cell voltage of 1.254 V [26].

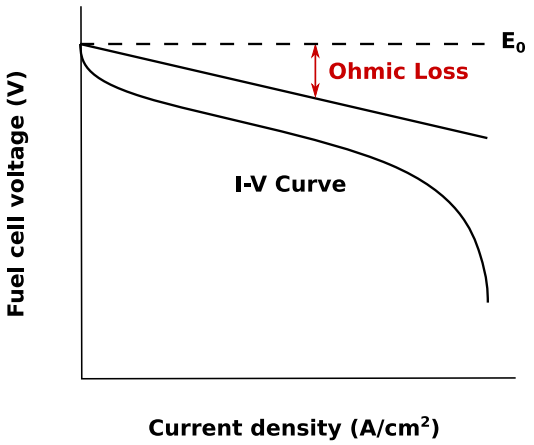

Figure 2.6.: The ohmic loss of a PEM fuel cell inspired by an image by O'Hayre et al. [26].

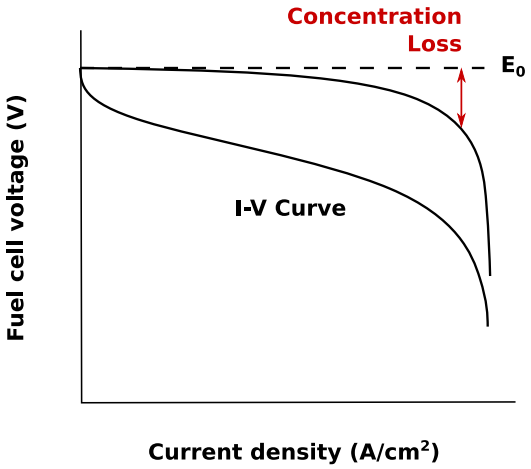

Figure 2.7.: The concentration loss of a PEM fuel cell inspired by an image by O'Hayre et al. [26].

2.3.2. Activation Loss

For the electrochemical reaction to happen, a certain amount of activation overpotentail is needed to "ignite" a reaction. This required overpotential is lowered by the presence of a catalyst. The activation overpotential is given by the Butler-Volmer equation for the anode and cathode half-reactions, but it is not given here as the equation is rather lengthy and complicated (see chapter 3 [26]) and the effect of the activation loss is illustrated in fig. 2.5 instead. The activation loss is the dominant loss in the lower current density regions.

2.3.3. Ohmic Loss

The ohmic loss occurs due to the fact that energy is required to move the protons through the electrolyte. It depends on the resistance of the material through which the protons flow. In our case the water saturation profile of Nafion-117 influences the resistance of the membrane, so the ohmic loss is highly dependable on the water content of the membrane. There are two physical phenomena that regulate the water saturation along the membrane: The electro-osmotic drag and the back-diffusion of water [26]. The electro-osmotic drag is the effect of protons dragging water molecules along as they move from the anode to the cathode side. Back diffusion of water occurs due to a high concentration of water at the cathode side, which diffuses water back to the anode side. The electron transport is usually negligible compared to proton transport. The effect of the ohmic loss is illustrated in fig. 2.6 and it is the dominant loss in the middle regions of the current density.

2.3.4. Concentration Loss

The concentration loss is the loss that happens due to the depletion of reactants at both electrodes and a "choking" effect of water mainly at the cathode. Reactants must at all times be delivered to the membrane and the surplus of product must be carried away. This loss is easiest to regulate for a fuel cell through the design choices, such as the flow channel pattern and GDL configuration and porosity. Computational fluid dynamics (CFD) methods are used to predict this loss and to optimize the fuel cell for high power output. This loss is the focus of this thesis, as we will introduce a discontinuous Galerkin discretization approach for simulating the flow of species through non-porous and porous media.
The effect of the concentration loss on the current-density curve is most dominant in the higher current-density regions and it is illustrated in fig. 2.7. For a high power output of a fuel cell, the objective is to find a point on the current-density graph where both the voltage and current are high. This can be done by translating the concentration loss curve to the right, such that the overall current-density curve can be smeared out to higher current-density regions than illustrated.

3. Discontinuous Galerkin Method: Theory

The discontinuous Galerkin method (DGM) was first introduced in the 70s by Reed & Hill [27] to simulate neuron transport and has ever since been used as a method for solving hyperbolic problems. The local conservation properties make it an interesting method for shocks and the localized formulation makes it a method that is straightforward to parallelize. The discontinuous Galerkin method emerged from the implementation of weakly enforced boundary conditions for the continuous Galerkin method [4] and it was realised that the weakly enforcing of boundary conditions could also be used as a flux on a boundary between two gridcells. This allows for the use of discontinuous test function spaces, which makes it an interesting method for simulating problems where discontinuities may arise. High-order basis functions can be used to capture physical behaviour that cannot be captured by a low order interpolant, allowing for an increased understanding of a physical system.
The use of DG methods for elliptic problems finds its origin in the introduction of the interior penalty methods [3, 13] and an excellent overview of various discontinuous Galerkin methods for elliptic problems is provided by D. N. Arnold et al. [4]. The given solver can already solve problems of hyperbolic nature, using a discontinuous Galerkin discretization. To be able to solve the flow of a PEMFC, however, the solver must be extended with additional discretization methods that are used for solving elliptic and parabolic partial differential equations (PDEs) of higher order. High-order (2^{nd} order or higher) spatial derivatives in a PDE require an additional numerical flux on the boundary between cells for a stable solution. For the solver we will choose to implement the symmetric interior penalty discontinuous Galerkin (SIPG) method, a method from the class of interior penalty methods.
The most important notation with respect to DG methods will be given in section 3.1. Secondly, some background information will be given on the functionality of the basis functions in section 3.2. Thirdly, the internal penalty DG methods will be treated in section 3.3. Lastly, the treatment of time-dependent problems is explained in section 3.4.

3.1. Notation

Before the symmetric interior penalty method for elliptic PDEs can be explained, the notation that will be used for the DG methods will be explained here. This notation will be supplemented throughout the thesis and a complete list can be found at the end of this thesis in the list of symbols.

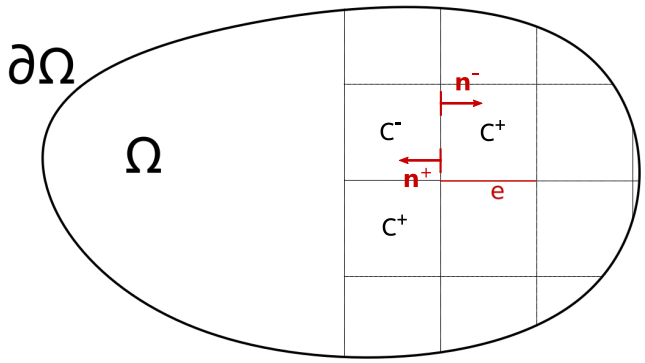

Figure 3.1.: An arbitrarily shaped domain Ω with boundary $\partial\Omega$. The domain is divided by a regular grid of cells C. The normal of a cell is indicated by the outward normal $\boldsymbol{n}^-$ and the normal of a neighbouring cell is indicated by $\boldsymbol{n}^+$. An edge e is the interface of the two neighbouring cells.

In fig. 3.1 an arbitrarily shaped domain Ω with boundary $\partial\Omega$ is shown. It consists of elements, which can have any arbitrary shape, but will be assumed to be square in this thesis for simplicity. It must be noted that in this case Ω and $\partial\Omega$ are actually computational approximations of the physical domain and boundary respectively, meaning that $\partial\Omega$ is a piecewise linear representation and each line borders one cell element. The inner cell is denoted by C^- and a neighbouring cell is denoted by C^+. With this nomenclature, the normal vectors $\boldsymbol{n}^-$ and $\boldsymbol{n}^+$, which point outward for cells C^- and C^+, can be distinguished. Lastly, the edge e is defined as the interface connecting two neighbouring cells.
The degrees of freedom are denoted by the vector $\boldsymbol{u}$, which shouldn't be confused with its scalar counterpart u, the flow velocity in x-direction, and a flux is be denoted by $\boldsymbol{F}$.
Additional to the previously defined elements, there are a few sets that need to be defined. These sets are listed below:

- Set of all cells: $\mathcal{T}_h$
- Set of all edges: $\mathcal{E}_h$
- Set of all internal edges: Γ
- Set of all boundary edges: $\partial\Omega$

Operators

The approximation of the degrees of freedom is continuous within each cell, but a discontinuity exists across an edge. This means that on each edge, two values for the degrees of

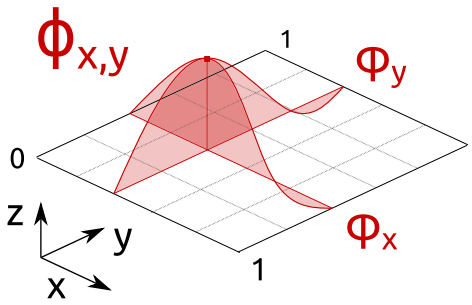

Figure 3.2.: A 2D scalar basis function $\phi_{x,y}$ on a cell C

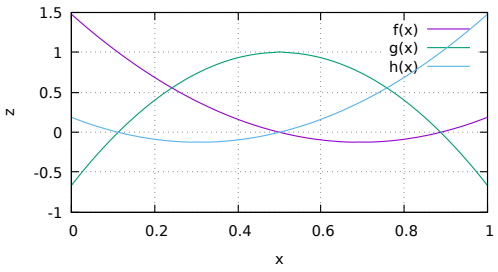

Figure 3.3.: Second order Lagrange polynomials

freedom ($\boldsymbol{u}^-$ and $\boldsymbol{u}^+$) can be defined. We introduce the following operators to denote th average and jump across an edge. The operator $\{\!\{\cdot\}\!\}$ will be used to denote the average, and $[\![\cdot]\!]$ will denote the jump of scalars or vectors across an edge e. Apart from that, the operator $[\![\cdot]\!]_s$ is used to define a scalar jump of a vector. These scalar and vector operators are listed below:

Vector:

$$\begin{aligned} \{\!\{\boldsymbol{v}\}\!\} &:= \frac{1}{2}\left(\boldsymbol{v}^+ + \boldsymbol{v}^-\right) \\ [\![\boldsymbol{v}]\!] &:= \boldsymbol{v}^+ \otimes \boldsymbol{n}^+ + \boldsymbol{v}^- \otimes \boldsymbol{n}^- \\ [\![\boldsymbol{v}]\!]_s &:= \boldsymbol{v}^+ \cdot \boldsymbol{n}^+ + \boldsymbol{v}^- \cdot \boldsymbol{n} \end{aligned}$$

Scalar:

$$\begin{aligned} \{\!\{p\}\!\} &:= \frac{1}{2}\left(p^+ + p^-\right) \\ [\![p]\!] &:= p^+ \boldsymbol{n}^+ + p^- \boldsymbol{n}^- \end{aligned}$$

Basis Functions

To explain the notation for the basis functions, fig. 3.2 is used. This figure shows a single cell C with a single two-dimensional basis function $\phi_{x,y}$ that consists of a combination of one-dimensional basis functions: φ_x and φ_y. It is worth noting that one- and two-dimensional basis functions are denoted by different versions of the small Greek letter ϕ. From here on, each basis function mentioned in this thesis can be assumed to be two-dimensional, unless stated otherwise. Furthermore, the Lagrange polynomials will be used as basis functions in this thesis.

3.2. Lagrange Basis

The basis is used for the approximation of the physical solution and the approximation of the solution $\boldsymbol{u}$ is denoted by $\boldsymbol{u}_h$. The Lagrange basis function, of polynomial order p leads

to $(p+1)^2$ basis functions in 2D and is given by

$$\hat{\varphi}_i(x) = \prod_{\substack{0 \le j \le p \\ i \neq j}} \frac{x - \hat{x}_j}{x_i - \hat{x}_j}. \tag{3.1}$$

for the reference cell $\hat{C}$, where the reference cell is defined as $\hat{C}(\hat{x}, \hat{y})$ with $\hat{x},\ \hat{y} \in [0,1]^2$. The reference cell is used to simplify the computation by first approximating the solution on the reference cell and afterwards map this solution to the physical cell. This mapping procedure is explained in more detail in appendix B. Functions and values with respect to the reference element are denoted by a hat "ˆ".
Furthermore, the nodes are chosen such that we have co-located Gaussian quadrature and the evaluation of integrals will therefore be relatively cheap. Mapped to the interval $[0,1]$ the quadrature points, quadrature weights and the corresponding basis functions φ_i and their derivatives are given below for $p = 2$:

$$\hat{x} \in \left\{ 0.5 - \sqrt{\frac{3}{20}},\ 0.5,\ 0.5 + \sqrt{\frac{3}{20}} \right\}, \qquad w \in \left\{ \frac{5}{18},\ \frac{8}{18},\ \frac{5}{18} \right\}$$

$$\begin{aligned}
\hat{\varphi}_1(x) &= 3\frac{1}{3}x^2 - 4.62433x + 1.47883 & \hat{\varphi}_1(x)' &= 6\frac{2}{3}x - 4.62433 \\
\hat{\varphi}_2(x) &= -\frac{20}{3}x^2 + 6\frac{2}{3}x - \frac{2}{3} & \hat{\varphi}_2(x)' &= -\frac{40}{3}x + 6\frac{2}{3} \\
\hat{\varphi}_3(x) &= 3\frac{1}{3}x^2 - 2.04234x + 0.187836 & \hat{\varphi}_3(x)' &= 6\frac{2}{3}x - 2.04234
\end{aligned}$$

The basis functions $\hat{\varphi}_{\{1,2,3\}}$ are plotted in fig. 3.3 as $f(x)$, $g(x)$ and $h(x)$ respectively. Here it can clearly seen that the other Lagrange functions are equal to zero, where one of them has its maximum, meaning that the functions are orthogonal with respect to each other. The sum of the lagrange polynomials is an interpolating function going through the nodes of the lagrange polynomials. In fig. 3.3, the result will be the line $y = 1$.

Lagrange Basis in 2D

The Lagrange basis for a two-dimensional problem is the tensor-product basis given by

$$\hat{\phi}_n(x,y) = \hat{\varphi}_{n_x}(x) \cdot \hat{\varphi}_{n_y}(y), \tag{3.2}$$

where $\hat{\phi}_n$ denotes the 2D basis functions and $\hat{\varphi}_{n_x}$ and $\hat{\varphi}_{n_y}$ are the 1D basis functions. The partial derivatives in x-direction and in y-direction are given by equations 3.3 and 3.4, respectively

$$\frac{\partial \hat{\phi}_n(x,y)}{\partial x} = \frac{\partial \hat{\varphi}_{n_x}(x)}{\partial x} \cdot \hat{\varphi}_{n_y}(y), \tag{3.3}$$

$$\frac{\partial \hat{\phi}_n(x,y)}{\partial y} = \hat{\varphi}_{n_x}(x) \cdot \frac{\partial \hat{\varphi}_{n_y}(y)}{\partial y}. \tag{3.4}$$

3.3. Interior Penalty Discontinuous Galerkin Method

The problem of a discontinuity in the approximation across an edge is that there is no uniquely defined approximation on the edge. This can be illustrated by fig. 3.2, by imagining two adjacent cells with different approximations for the solution at a shared edge. To solve this problem, methods have been developed to approximate a numerical flux value at the edge and common examples for the numerical flux are the central flux, Rusanov flux or the Lax-Friedrichs flux. The characteristics of a discontinuous Galerkin discretization, such as consistency and stability, strongly depend on the chosen flux. For a more extensive explanation, the reader is referred to LeVeque [23].
Since the discontinuous Galerkin method was first introduced in the early 1970s, there have been many approaches to apply this method to PDEs with high-order derivative terms. A generalized approach to discretizing high-order PDEs using the DGM is presented by D. N. Arnold et al. [4]. In this section we will discuss the general approach of arriving at the primal formulation of an equation. For this, a few tools will be needed, which are given in the green formula boxes.

Green's Theorem

$$\int_C \boldsymbol{u}\nabla^2\boldsymbol{v}\, d\boldsymbol{x} = -\int_C \nabla\boldsymbol{u}\cdot\nabla\boldsymbol{v}\, d\boldsymbol{x} + \int_{\partial C} \boldsymbol{u}\nabla\boldsymbol{v}\cdot\boldsymbol{n}\, ds$$

The first step is to define the PDE as a system of first-order differential equations. Each term is then multiplied with a corresponding test function from the test function space and the result is integrated over the domain, resulting in the weak form of the equation. If the test function space is equal to the basis function space, the problem is better known as a Galerkin problem. Since the basis function space is a space of discontinuous piecewise polynomials, the test space will also be a space of discontinuous piecewise polynomial functions, resulting in a discontinuous Galerkin problem. The problem is then defined as a sum of cells with a continuous test function space and then, Green's theorem is applied to the system of first-order differential equations, to obtain an integral evaluated over the cell and an integral defined over the cell boundary ∂C.

The unifying formula

The unifying formula is given as

$$\sum_{C\in\mathcal{T}_h}\int_{\partial C} v\boldsymbol{\tau}\cdot\boldsymbol{n}\,ds = \sum_{e\in\mathcal{E}_h}\int_e [\![v]\!]\,\{\!\{\boldsymbol{\tau}\}\!\}\,ds + \sum_{e\in\Gamma}\int_e \{\!\{v\}\!\}\,[\![\boldsymbol{\tau}]\!]_s\,ds.$$

This formula is based on the following identity:

$$a_1b_1 - a_2b_2 = \frac{1}{2}(a_1+a_2)(b_1-b_2) + \frac{1}{2}(b_1+b_2)(a_1-a_2).$$

So far, the steps taken have been along the lines of those for the continuous Galerkin method, but now the sum of all cells is taken to form the domain Ω. The unifying formula is applied to rewrite the integrals defined over the cell boundaries $\sum \partial C$ into an integral defined over the edges Γ. This essentially means that it is used to eliminate duplicates of cell-boundary terms and replaces them with a unique edge definition, where the average ($\{\!\{\cdot\}\!\}$) and jump ($[\![\cdot]\!]$) operators that were introduced in section 3.1 are used. Defining the problem as an integral over the domain Ω and an integral of edges, gives the *flux formulation*.
As an example, the flux formulation for the Poisson equation is given by D. N. Arnold et al. [4] as

$$\int_\Omega \nabla u_h\cdot\nabla v\,d\boldsymbol{x} + \int_{\mathcal{E}_h}([\![\hat{u}_h - u_h]\!]\cdot\{\!\{\nabla v\}\!\} - \{\!\{\hat{\sigma}\}\!\}\cdot[\![v]\!])\,ds + \int_\Gamma(\{\!\{\hat{u}_h-u_h\}\!\}[\![\nabla v]\!] - [\![\hat{\sigma}]\!]\{\!\{v\}\!\})\,ds = \int_\Omega fv\,d\boldsymbol{x}.$$

Here, the variables denoted by a hat represent the numerical flux. Various numerical fluxes are given in table 3.1, where $\delta_{r,j}$ denote the penalty coefficients for the respective methods and β is a vector valued function which is constant on each edge [4]. For the solver we will be implementing the interior penalty fluxes, due to its consistency and stability characteristics [28]. Substituting the numerical fluxes in the flux formulation by the definitions given for the interior penalty method will result in the interior penalty method for the poisson equation. The following identities are used to simplify the expression

Averages and Jumps identities

$$\{\!\{a+b\}\!\} = \{\!\{a\}\!\} + \{\!\{b\}\!\} \qquad \{\!\{[\![a]\!]\}\!\} = [\![a]\!]$$
$$[\![a+b]\!] = [\![a]\!] + [\![b]\!] \qquad \{\!\{\{\!\{a\}\!\}\}\!\} = \{\!\{a\}\!\}$$
$$[\![\{\!\{a\}\!\}]\!] = 0 \qquad [\![[\![a]\!]]\!] = 0$$

Table 3.1.: Some numerical fluxes for the DGM as given by D. N. Arnold et al. [4].

Method	$\hat{u}_h$	$\hat{\sigma}_h$
Bassi & Rebay [6]	$\{\!\{u_h\}\!\}$	$\{\!\{\sigma_h\}\!\}$
Brezzi et al. [7]	$\{\!\{u_h\}\!\}$	$\{\!\{\sigma_h\}\!\} - \delta_r([\![u_h]\!])$
LDG [8]	$\{\!\{u_h\}\!\} - \beta \cdot [\![u_h]\!]$	$\{\!\{\sigma_h\}\!\} + \beta[\![\sigma_h]\!] - \delta_j([\![u_h]\!])$
IP [13]	$\{\!\{u_h\}\!\}$	$\{\!\{\nabla u_h\}\!\} - \delta_j([\![u_h]\!])$

The primal formulation for the interior penalty method is given by D. N. Arnold et al. [4] as

$$\int_\Omega \nabla u_h \cdot \nabla v \, d\boldsymbol{x} - \int_\Gamma ([\![u_h]\!] \cdot \{\!\{\nabla v\}\!\} + \epsilon \{\!\{\nabla u_h\}\!\} \cdot [\![v]\!]) \, ds + \int_\Gamma \delta [\![u_h]\!] \cdot [\![v]\!] \, ds. \tag{3.5}$$

Here, last term is the stabilization term with the penalty coefficient δ. Its value is mesh dependent and is given by D. N. Arnold et al. [4] as $\frac{\eta_e}{h_e}$ on each edge e with η_e a positive number. An alternative definition is given by J. Hesthaven & Warburton [18], who provide $\delta \geq C\frac{p^2}{h}$, with $C \geq 1$, where the lower bound on C is derived by Shahbazi [29].
For edges that coincide with the domain boundary $\partial\Omega$ the interior penalty flux is given by

$$\hat{u}_h = 0, \qquad \hat{\sigma} = \{\!\{\nabla u_h\}\!\} - \delta [\![u_h]\!].$$

The final linear system that is described by equation 3.5 is a symmetric system for $\epsilon = 1$. This special form of the interior penalty method is the SIPG method. Other common variations include $\epsilon = -1$ for the nonsymmetric interior penalty Galerkin (NIPG) method [28], and $\epsilon = 0$ for the incomplete interior penalty Galerkin (IIPG) method [11].

3.4. Time-Stepping Methods

Apart from the spatial accuracy, the order of accuracy of the time dimension also affects the total order of approximation for time dependent problems.
For simplicity we will be considering one-step methods for solving the ordinary differential equation (ODE) and the maximum timestep size usually depends on the eigenvalue characteristics of the flow in computational fluid dynamics. For the SIPG method the penalty parameter can theoretically be chosen as large as possible, as long as the lower bound, given by Shahbazi [29], is exceeded. A high penalty parameter, however, leads to an increased condition number of the resulting linear system and, thus, degrades the convergence for an iterative solving method for steady state problems [18]. For time-dependent problems, a similar performance degradation can be expected on the maxi-

mum timestep size. A loss of about 50% of the maximum timestep size is observed for an increase of the penalty parameter from $\delta = 0.67$ to $\delta = 2$ by Alhawwary & Wang [2] for the BR2 scheme with RK3 and RK4 time discretizations. The BR2 scheme is numerically similar to the SIPG method, so this means that the maximum allowed timestep size for a time-dependent problem could also depend on the size of the penalty parameter [2].

Part II.

Model Discretization

4. Compressible Navier-Stokes Equations

There are a few physical phenomena that need to be modeled for an accurate representation of a PEMFC. Before a number of these phenomena is considered, the most important question that must be answered is whether the flow is to be seen as compressible or incompressible, since both flow representations have a significant effect on the modeling of the remaining physics. In most fuel cell literature, it is chosen to represent the flow in a PEMFC as a compressible flow (see [26], [30]). The reason for this is that the pressure and density of the fluid play a significant role in the fuel cell performance. The fluid flow speed is perhaps not very high (definitely below Mach 0.3, the conventional bound for considering airflow as compressible), but a fuel cell can be designed to operate at pressures higher than atmospheric pressure. More importantly, the density plays an important role when we consider a flow that has more than one chemical species. Therefore, the compressible Navier-Stokes equations are an obvious starting point in the process of developing a PEMFC model.
Apart from its physically more accurate representation, the compressible Navier-Stokes equations are relatively simple to implement with the discontinuous Galerkin method. The incompressible Navier-Stokes equations require a projection of the velocity to a divergence-free subspace and usually also a time-splitting scheme. Neither are necessary for the compressible Navier-Stokes equations [18]. On the other hand, compressible Navier-Stokes equations can pose some computing difficulties, since very small time steps in explicit time-stepping methods are required for computational stability. This is due to the diffusion, and the dependence of the eigenvalues on the speed of sound [18]. An implicit time-stepping method can be chosen, which would be necessary to accurately solve a non-linear PDE (diffusion), but it requires a big non-linear system of equations to be solved. A good compromise could be a combination of the two.
In this chapter the derivation of the compressible Navier-Stokes equations in SIPG form will be explained. The compressible Navier-Stokes equations will be given together with the equations of state in section 4.1. The discretization with the interior penalty DG method is explained in section 4.2 and section 4.3 will explain how to obtain the linear system from the discretization and their implementation. Afterwards, the boundary conditions will be discussed in section 4.4 and, finally, this chapter will end with a description of the time-stepping method in section 4.5.

4.1. The Compressible Navier-Stokes Equations

For the discretization of the Navier-Stokes equations we consider a two-dimensional time-dependent problem. The conservative system is given by

$$\frac{\partial}{\partial t}\boldsymbol{u} + \nabla \cdot \boldsymbol{F}^h(\boldsymbol{u}) - \nabla \cdot \boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{S}, \tag{4.1}$$

where $\boldsymbol{u} \in \mathbb{R}^4$ are the conservative variables, referred to as the *degrees of freedom*, and $\boldsymbol{F}^h, \boldsymbol{F}^v \in \mathbb{R}^{4\times 2}$ are respectively the hyperbolic and viscous fluxes of the Navier-Stokes equations, and $\boldsymbol{S}$ is the source term vector of the Navier-Stokes equations. The numerical treatment of the source terms is explained in chapter 5, so for now a homogeneous problem will be assumed.
The explicit definitions of the degrees of freedom $\boldsymbol{u}$, hyperbolic flux $\boldsymbol{F}^h$ and viscous flux $\boldsymbol{F}^v$ are

$$\begin{aligned} \boldsymbol{u} &= \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \qquad \boldsymbol{F}^h(\boldsymbol{u}) = \begin{pmatrix} \rho u & \rho v \\ \rho u^2 + p & \rho uv \\ \rho uv & \rho v^2 + p \\ u(\rho E + p) & v(\rho E + p) \end{pmatrix}, \\ \boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u}) &= \begin{pmatrix} 0 & 0 \\ \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \\ \sigma_{11}u + \sigma_{12}v + k^{\text{eff}}T_x & \sigma_{21}u + \sigma_{22}v + k^{\text{eff}}T_y \end{pmatrix}. \end{aligned} \tag{4.2}$$

Here ρ, $\boldsymbol{v} = (u, v)^T$, p, E and T are the density, velocity vector, pressure, specific total energy and temperature respectively. The thermal conductivity is given by k. The viscous stress tensor σ is defined as

$$\begin{aligned} \sigma(\boldsymbol{v}, \nabla \boldsymbol{v}) &= \mu \left(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T - \frac{2}{3}\boldsymbol{I}\,(\nabla \cdot \boldsymbol{v}) \right) \\ &= \mu \begin{pmatrix} \frac{4}{3}u_x - \frac{2}{3}v_y & v_x + u_y \\ u_y + v_x & \frac{4}{3}v_y - \frac{2}{3}u_x \end{pmatrix}. \end{aligned} \tag{4.3}$$

To close the system, we still need an equation for pressure and temperature. For this we use the definition of total enthalpy H, which is given by

$$H = E + \frac{p}{\rho} = e + \frac{1}{2}\boldsymbol{v}^2 + \frac{p}{\rho},$$

where e is the specific internal energy. This definition of the total enthalpy is used to find

the definition of pressure, which is given by the equation of state of an ideal gas

$$\begin{aligned} p &= (\gamma - 1)\rho e \\ &= (\gamma - 1)(\rho E - \frac{1}{2}\rho u^2 - \frac{1}{2}\rho v^2). \end{aligned} \tag{4.4}$$

where γ is the heat capacity ratio c_p/c_v, which has the value $\gamma = 1.4$ for air. The temperature can be found using the relation

$$T = \frac{e}{c_v} = \frac{\gamma e}{c_p} = \frac{\mu\gamma}{Prk} e,$$

meaning that

$$kT = \frac{\mu\gamma}{Pr}(E - \frac{1}{2}u^2 - \frac{1}{2}v^2), \tag{4.5}$$

where we take the Prandtl number $Pr = 0.72$ for air. Let it be clear that values such as Pr, γ and k, which can generally be treated as constants, will be variable in the context of a fuel cell simulation since the flow will be a mixture of chemical species.

4.2. Discontinuous Galerkin Discretization with Interior Penalty

The derivation of the discontinuous Galerkin discretization with interior penalty for the compressible NSE is based on the paper by Hartmann and Houston [16] and on the lecture notes of Hartmann and Leicht [17]. The compressible NSE in equation 4.1 can, for the purpose of discretization, be rewritten in the following equivalent form

$$\frac{\partial}{\partial t}\boldsymbol{u} + \frac{\partial}{\partial x_n}\left(\boldsymbol{F}^h_{x_n}(\boldsymbol{u}) - \underbrace{G_{nm}(\boldsymbol{u})\frac{\partial \boldsymbol{u}}{\partial x_m}}_{\sigma} \right) = 0. \tag{4.6}$$

Here the matrix G is the homogeneity tensor of the viscous Flux and its explicit definition is given in appendix A. The einstein notation is used to denote the summation over subscripts. In this 2D example, the subscripts have the following values: $n, m \in \{1, 2\}$. We will be using σ as a auxiliary variable in the ensuing derivation and it is not to be confused with the viscous stress tensor from equation 4.3.
The discretization of the compressible NSE will be done according to the steps explained in section 3.3. The equations are split into a system of first-order equations, the weak form is formulated and afterwards integration by parts is applied, giving us the *flux formulation* of the compressible NSE. The goal of this section will be to arrive at the *primal formulation* of the compressible NSE, to eliminate the need for the auxiliary variable σ.

Weak Form

Equation 4.6 is re-written as a semidiscrete system of first-order partial differential equations in equations 4.7a and 4.7b. For the hyperbolic flux, the Rusanov flux is used (see J. S. Hesthaven [19] for the discontinuous Galerkin discretization for hyperbolic problems). The viscous flux is considered, as

$$\sigma = \begin{pmatrix} G_{1m}(\boldsymbol{u})\frac{\partial \boldsymbol{u}}{\partial x_m} & G_{2m}(\boldsymbol{u})\frac{\partial \boldsymbol{u}}{\partial x_m} \end{pmatrix}, \tag{4.7a}$$

$$-\nabla \cdot \sigma = 0. \tag{4.7b}$$

Here $\sigma \in \mathbb{R}^{4\times 2}$ functions as the auxiliary variable for splitting 4.6 to a system of first-order PDEs. The next step will be to take the weak form of the system. For that, the test function spaces need to be defined,

$$\Sigma_h = \{\boldsymbol{\sigma}_h \in [L_2(\Omega)]^{4\times 2} : \boldsymbol{\sigma}_h|_C \in [Q_p(C)]^{4\times 2}, C \in T\},$$
$$V_h = \{\boldsymbol{v}_h \in [L_2(\Omega)]^4 : \boldsymbol{v}_h|_C \in [Q_p(C)]^4, C \in T\}.$$

Here, $Q_p(C)$ denotes the space of tensor product polynomials on C of degree p in each coordinate direction. Multiplying 4.7a and 4.7b by the corresponding test functions, respectively $\tau \in \Sigma_h$ and $\boldsymbol{v} \in V_h$, and applying integrating by parts, we arrive at the following system:

$$\int_C \sigma \cdot \tau \, d\boldsymbol{x} = \int_{\partial C} u_j \left[\left((G(\boldsymbol{u})_{nm})_{:j} \, \tau_{:n} \right) n_m \right] \, ds - \int_C u_j \frac{\partial}{\partial x_m} \left((G(\boldsymbol{u})_{nm})_{:j} \, \tau_{:n} \right) \, d\boldsymbol{x}, \tag{4.8a}$$

$$-\int_C (\nabla \cdot \sigma)\boldsymbol{v} \, d\boldsymbol{x} = -\int_{\partial C} (\sigma \cdot \boldsymbol{n})\boldsymbol{v} \, ds + \int_C \sigma \cdot (\nabla \boldsymbol{v}) \, d\boldsymbol{x} = 0. \tag{4.8b}$$

Where $\boldsymbol{n}$ is the normal unit vector of a face, facing outward of the cell and n_m are the components of the normal unit vector. Here we used the following relation:

$$\begin{aligned} \int_C \sigma \cdot \tau \, d\boldsymbol{x} = \int_C \sigma_{:n} \tau_{:n} \, d\boldsymbol{x} &= \int_C (G(\boldsymbol{u})_{nm})_{:j} \frac{\partial u_j}{\partial x_m} \tau_{:n} \, d\boldsymbol{x} \\ &= \int_C \frac{\partial u_j}{\partial x_m} (G(\boldsymbol{u})_{nm})_{:j} \, \tau_{:n} \, d\boldsymbol{x} \end{aligned}$$

Here, the weak form of the system of PDEs is shown and the colon-notation is used (analogous to the use of the colon syntax in julia) to denote that there is no summation implied for the respective index. Instead, the operation is applied uniquely for each element of the

set that is described by the colon-notation. We deviate here from the derivation by Hartmann & Houston [16], since we are not considering a bilinear approach. In this example, the result will be a vector of length four, since $\tau \in \mathbb{R}^{4\times 2}$ and $G_{nm} \in \mathbb{R}^{4\times 4}$.

Flux Formulation

Our task is to find approximations of the solution, given by $\boldsymbol{u}_h \in V_h$ and $\sigma_h \in \Sigma_h$, such that for all $C \in T_h$ we have

$$\int_C \sigma_h \cdot \tau \; d\boldsymbol{x} = \int_{\partial C} (\hat{u}_h)_j \left[((G(\boldsymbol{u}_h)_{nm})_{:j}\tau_{:n})n_m\right] \; ds - \int_C (u_h)_j \frac{\partial}{\partial x_m}\left[(G(\boldsymbol{u}_h)_{nm})_{:j}\tau_{:n}\right] \; d\boldsymbol{x}, \tag{4.9a}$$

$$- \int_{\partial C} (\hat{\sigma}_h \cdot \boldsymbol{n})\boldsymbol{v} \; ds + \int_C \sigma_h \cdot (\nabla \boldsymbol{v}) \; d\boldsymbol{x} = 0. \tag{4.9b}$$

This is called the *flux formulation* of the method. The fluxes $\hat{\boldsymbol{u}}_h$ and $\hat{\sigma}_h$ represent the numerical fluxes for the viscous flux along the face ∂C. In the following derivation, the explicit notation of the dependence of G_{nm} on $\boldsymbol{u}_h$ will be omitted.

Primal Formulation

To get the primal formulation, the auxiliary variable in equations 4.9a and 4.9b is eliminated. We do this by performing a second integration by parts on equations 4.9a and we set $\tau = \nabla \boldsymbol{v}$, which gives

$$\begin{aligned}\int_C \sigma_h \cdot \nabla \boldsymbol{v} \; d\boldsymbol{x} = &\int_C \left(G_{nm}\frac{\partial \boldsymbol{u}_h}{\partial x_m}\right)\frac{\partial \boldsymbol{v}}{\partial x_n} \; d\boldsymbol{x} \\ &+ \int_{\partial C} (\hat{u}_h - u_h)_j \left[\left((G_{nm})_{:j}\frac{\partial \boldsymbol{v}}{\partial x_n}\right) n_m\right] \; ds.\end{aligned} \tag{4.10}$$

Substituting equation 4.10 into 4.9b, and summing over all elements, yields the following *primal formulation*

$$\begin{aligned}\int_\Omega \left(G_{nm}\frac{\partial \boldsymbol{u}_h}{\partial x_m}\right)\frac{\partial \boldsymbol{v}}{\partial x_n} \; d\boldsymbol{x} - \sum_{C\in T_h}\int_{\partial C} (\hat{\sigma}_h \cdot \boldsymbol{n})\, \boldsymbol{v} \; ds \\ + \sum_{C\in T_h}\int_{\partial C} (\hat{\boldsymbol{u}}_h - \boldsymbol{u}_h)\left(\left((G_{nm})_{:,j}\frac{\partial \boldsymbol{v}}{\partial x_n}\right) n_m\right) \; ds = 0.\end{aligned} \tag{4.11}$$

Using the definitions from section 3.1, we define the primal formulation in a face-based manner. For this, we make use of the generalised form of the *unifying formula* given in

section 3.3. This results in the following *primal formulation*

$$\begin{aligned}\int_\Omega \left(G_{nm}\frac{\partial \boldsymbol{u}_h}{\partial x_m}\right)\frac{\partial \boldsymbol{v}}{\partial x_n}\,d\boldsymbol{x} + \int_\Gamma [\![\hat{\boldsymbol{u}}_h - \boldsymbol{u}_h]\!]_j \cdot \{\!\{(G_{nm})_{:j}\frac{\partial \boldsymbol{v}}{\partial x_n}\}\!\} - [\![\boldsymbol{v}]\!]\cdot\{\!\{\hat{\sigma}_h\}\!\}\,ds \\ \int_{\Gamma_I} \{\!\{\hat{\boldsymbol{u}}_h - \boldsymbol{u}_h\}\!\}_j [\![(G_{nm})_{:j}\frac{\partial \boldsymbol{v}}{\partial x_n}]\!]_s - \{\!\{\boldsymbol{v}\}\!\}[\![\hat{\sigma}_h]\!]_s\,ds = 0.\end{aligned} \tag{4.12}$$

Numerical Flux

For the symmetric interior penalty for the compressible NSE, the numerical flux is given as follows

Numerical Flux

$$\hat{\boldsymbol{u}}_h = \{\!\{\boldsymbol{u}_h\}\!\} \qquad \hat{\sigma}_h = \{\!\{G(\boldsymbol{u}_h)\cdot\nabla \boldsymbol{u}_h\}\!\} - \delta[\![\boldsymbol{u}_h]\!]$$

We use the identities for jumps and averages presented in section 3.3 and arrive at the final *primal formulation* for the symmetric interior penalty DGM, given by

$$\begin{aligned}\int_\Omega \underbrace{\left(G_{nm}\frac{\partial \boldsymbol{u}_h}{\partial x_m}\right)}_{\boldsymbol{F}^v(\boldsymbol{u}_h,\nabla\boldsymbol{u}_h)}\frac{\partial \boldsymbol{v}}{\partial x_n}\,d\boldsymbol{x} - \int_\Gamma [\![\boldsymbol{u}_h]\!]_j \cdot \{\!\{(G_{nm})_{:,j}\frac{\partial \boldsymbol{v}}{\partial x_n}\}\!\}\,ds \\ - \int_\Gamma [\![\boldsymbol{v}]\!]\cdot\{\!\{\underbrace{G_{nm}\frac{\partial \boldsymbol{u}_h}{\partial x_m}}_{\boldsymbol{F}^v(\boldsymbol{u}_h,\nabla\boldsymbol{u}_h)}\}\!\}\,ds + \delta\int_\Gamma [\![\boldsymbol{u}_h]\!]\cdot[\![\boldsymbol{v}]\!]\,ds = 0.\end{aligned} \tag{4.13}$$

In a more compact notation this becomes

$$\begin{aligned}\int_\Omega \nabla\boldsymbol{v}\cdot(\boldsymbol{F}^v(\boldsymbol{u}_h,\nabla\boldsymbol{u}_h))\,d\boldsymbol{x} - \underbrace{\int_\Gamma [\![\boldsymbol{u}_h]\!]_j\cdot\{\!\{G_{:j}\cdot\nabla\boldsymbol{v}\}\!\}\,ds}_{\text{symmetry term}} \\ \underbrace{-\int_\Gamma [\![\boldsymbol{v}]\!]\cdot\{\!\{\boldsymbol{F}^v(\boldsymbol{u}_h,\nabla\boldsymbol{u}_h)\}\!\}\,ds}_{\text{flux term}} + \underbrace{\delta\int_\Gamma [\![\boldsymbol{u}_h]\!]\cdot[\![\boldsymbol{v}]\!]\,ds}_{\text{penalty term}} = 0.\end{aligned} \tag{4.14}$$

Here we have denoted the three parts of the numerical flux as *symmetry term*, *flux term* and *penalty term*. Setting this terminology will be useful to understand the explanation of the discretization and implementation.

4.3. Linear System

In this section we show the discretization of the primal formulation of the Navier-Stokes equations, using the discontinuous Galerkin method. For that, we take the primal formulation of equation 4.14 and add the discontinuous Galerkin discretization of the hyperbolic flux as given by Krenz & Reinarz [22] for a cell C:

$$\begin{aligned}\underbrace{\int_C \phi_i \phi_j \, d\boldsymbol{x}}_{M} \frac{\partial \underline{u}_i}{\partial t} - \underbrace{\int_C \boldsymbol{F}^h(\boldsymbol{u}) \cdot \nabla \boldsymbol{v}_j \, d\boldsymbol{x}}_{v^h} + \underbrace{\int_{\partial C} (\boldsymbol{F}^h(\boldsymbol{u}) \cdot \boldsymbol{n}) \cdot \boldsymbol{v}_j \, ds}_{r} \\ + \underbrace{\int_C \nabla \boldsymbol{v}_j \cdot (G(\boldsymbol{u}) \cdot \nabla \boldsymbol{u}_h) \, d\boldsymbol{x}}_{v^v} - \underbrace{\int_e \{\!\{ G(\boldsymbol{u}) \cdot \nabla \boldsymbol{u}_h \}\!\} \cdot [\![\boldsymbol{v}_j]\!] \, ds}_{e^f} \\ - \underbrace{\int_e \{\!\{ G_{:,l} \cdot \nabla \boldsymbol{v}_j \}\!\} \cdot [\![\boldsymbol{u}_h]\!]_l \, ds}_{e^s} + \delta \underbrace{\int_e [\![\boldsymbol{u}_h]\!] \cdot [\![\boldsymbol{v}_j]\!] \, ds}_{e^p} \\ = 0.\end{aligned} \tag{4.15}$$

Here e indicates the integral over the edges of a cell C. The subscripts i, j here are used to indicate the index of the basis and test functions respectively. The subscript l is used to indicate the summation over the multiplication of matrix G and $\boldsymbol{u_h}$, resulting from the matrix-vector multiplication from equation 4.6. The dot products with the G matrices are done such that for a dot product with a test function we sum over the first index n and for a dot product with the approximation $\boldsymbol{u}$ we sum over the second index m of the G matrix. Matrix-vector multiplications with matrix G are always with the vector $\boldsymbol{u}$. Equation 4.15 can be written in the following more compact form

$$M \frac{\partial \underline{u}}{\partial t} = v^h - r - v^v + e^f + e^s - \delta e^p. \tag{4.16}$$

Here the edge terms e^f, e^s and e^p indicate the *flux term*, *symmetry term* and *penalty term* respectively. The volume terms for the hyperbolic and viscous flux are indicated by v^h and v^v. The Rusanov flux term, solving the numerical flux for the hyperbolic flux, is denoted by r. The mass-matrix is indicated by M. The mass-matrix M, hyperbolic volume term v^h and the hyperbolic numerical flux term r were already implemented in the solver. In this section we will focus on the numerical treatment of the viscous volume term v^v and the three edge terms e^f, e^s and e^p.
Before we start with the numerical treatment, we must define the problem on the reference cell $\hat{C}$. The problem is solved for the reference cell and the solution is then mapped to the physical cell C. This means that scaling coefficients will appear during the transformation from the physical cell to the reference cell. For a more detailed explanation of the mapping procedure, please see appendix B.

To get the linear system representation of equation 4.15, the unknown $\boldsymbol{u}$ and its fluxes must be represented by their approximation in the Lagrange basis, denoted by $\boldsymbol{u}_h$. The integrals are then calculated by using Gaussian quadratures and a global linear system can be built, which will be solved for $\boldsymbol{u}_h$, giving a discrete approximation for the density, velocity and temperature fields. In the following subsections we will explicitly define the terms v^v, e^f, e^s and e^p, such that they are ready to be implemented in the code.

4.3.1. Viscous Volume Term

Terms that are integrated over the entire cell (the volume of the cell) are called volume terms. In the compressible NSE scheme there are two volume terms. Namely, one for the hyperbolic flux, and one for the viscous flux. These terms are essentially derived by firstly expanding the expression in the Lagrange basis and then computing the integral using Gaussian quadrature. The result will be of a form that is ready to be implemented in the code.
For the viscous volume term, the same strategy is followed as for the hyperbolic volume term discretization, which is given by Krenz and Reinarz. [22] The viscous volume term is discretized as

$$\begin{aligned} v_j^v &= \int_C \boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u}) \cdot \nabla \boldsymbol{v}_j \; d\boldsymbol{x} \\ &= det(J) \int_{\hat{C}} \boldsymbol{F}^v(\boldsymbol{u}(\mathcal{M}(\hat{\boldsymbol{x}})), J^{-1}\hat{\nabla}\boldsymbol{u}) \cdot J^{-1}\hat{\nabla}\hat{\phi}_j \; d\hat{\boldsymbol{x}} \\ &= A \int_{\hat{C}} \sum_i \underline{\boldsymbol{F}}_i^v \hat{\phi}_i(\hat{\boldsymbol{x}}) \cdot \hat{\nabla}\hat{\phi}_j(\hat{\boldsymbol{x}}) \; d\hat{\boldsymbol{x}} \\ &= A \sum_k \sum_i \underline{\boldsymbol{F}}_i^v \underbrace{\hat{\phi}_i(\hat{\boldsymbol{x}}_k)}_{\delta_{ik}} \cdot \hat{\nabla}\hat{\phi}_j(\hat{\boldsymbol{x}}_k) \; \omega_k \\ &= A \sum_k \underline{\boldsymbol{F}}_k^v \cdot \hat{\nabla}\hat{\phi}_j(\hat{\boldsymbol{x}}_k) \; \omega_k. \end{aligned} \tag{4.17}$$

Here, J denotes the jacobian matrix of the mapping operator $\mathcal{M}$ from the reference cell to the physical cell. The determinant of the Jacobian of the transformation and the inverse Jacobian appear after the mapping procedure. Because the physical cells are squares, the inverse Jacobian can be taken out of the integral. The value of A is then Δx ($= \Delta y$). The viscous flux coefficients, $\underline{\boldsymbol{F}}^v$, are the evaluations of the viscous flux on the quadpoints in the physical cell. Since the quadpoints are defined on the reference cell, the arguments for the viscous flux must be transformed before they can be used.
The steps that are taken to discretize this volume term can be summarized in the following way:

- First the integral is transformed to the reference cell.

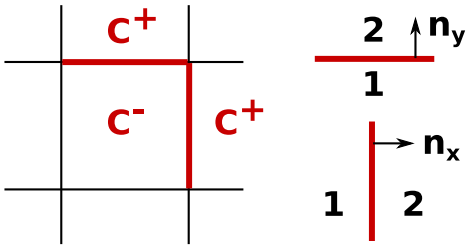

Figure 4.1.: Left: Cell nomenclature relative to neighbour cells. Right: Convention for nomenclature in the context of a single edge. The normal directions for an edge is indicated. The left and bottom sides are indicated by 1 and the right and top side by 2.

- The viscous flux is expanded by the basis functions $\hat{\phi}_i$.
- The integral is approximated using the Gaussian quadrature rule. The integrand is evaluated at the quadpoints $\hat{\boldsymbol{x}}_k$ and weighted by the quadweights ω_k. The evaluations for distinct quadpoints are then summed up.
- The last step is to simplify the expression if possible. In this case, it is possible to eliminate a summation, because we use co-located quadrature.

This summarized discretization procedure will also be leading for our discretization of the edge terms. Although the edge terms are based on one-dimensional integrals, much of the concepts used for the volume terms are still applicable.

4.3.2. Flux Edge Term

The discretization of the edge terms mainly follows the discretization procedure that was given for the volume term. In our approach we select an edge $e \in \Gamma$, which is always an inner edge because we assume a case with periodic boundary conditions. The definitions of the average $\{\!\{\cdot\}\!\}$ and jump $[\![\cdot]\!]$ operators are expanded. Here the - and + superscripts denote the inner cell, and the neighbouring cell respectively. In the context of an edge, it is simpler to define the two neighbouring cells relative to an edge. For this we set the following convention: For vertical edges, the left side is denoted by 1 and the right side is denoted by 2, and for horizontal edges the top cell is denoted by 1 and the bottom cell is denoted by 2. This concept is visualised in fig. 4.1.
The definition of e^f will first be derived for the reference edge $\hat{e}$, which is simply the edge

of a reference cell and has a length of one,

$$\begin{aligned} e^f &= \int_e \{\!\{ \boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u}) \}\!\} \cdot [\![\boldsymbol{v}]\!] \, ds \\ &= \int_e \frac{1}{2} \left(\boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u})^- + \boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u})^+ \right) \cdot \left(\boldsymbol{v}^- - \boldsymbol{v}^+ \right) \otimes \boldsymbol{n} \, ds \\ &= \int_{\hat{e}} \frac{1}{2} \left(\boldsymbol{F}^v \left(\boldsymbol{u}(\mathcal{M})(\hat{\boldsymbol{x}}), J^{-1} \hat{\nabla} \boldsymbol{u} \right)^- + \boldsymbol{F}^v \left(\mathcal{M}(\hat{\boldsymbol{x}}), J^{-1} \hat{\nabla} \hat{\boldsymbol{x}} \right)^+ \right) \\ &\qquad \cdot \left(\hat{\phi}_j^- - \hat{\phi}_j^+ \right) \otimes \boldsymbol{n} \, \Delta e \, d\hat{s} \\ &= \int_{\hat{e}} \frac{1}{2} \left(\sum_i \underline{F}_i^{v\,-} \hat{\phi}_i^- + \sum_i \underline{F}_i^{v\,+} \hat{\phi}_i^+ \right) \cdot \left(\hat{\phi}_j^- - \hat{\phi}_j^+ \right) \otimes \boldsymbol{n} \, \Delta e \, d\hat{s}. \end{aligned} \tag{4.18}$$

Here, $\hat{\nabla}$ is the gradient defined on the coordinates of the reference cell. It is multiplied by the inverse Jacobian, J^{-1}, to get the derivatives on the physical edge. Because we consider a reference cell, the integral is multiplied by the edge length Δe to get the integral over the physical edge and we can simplify certain terms in equation 4.18.
We will continue our derivation for the edge γ (the right edge of a cell), to demonstrate the simplifications:

$$\begin{aligned} \gamma^f &= \int_\gamma \frac{1}{2} \left(\sum_i \underline{F}_i^{v\,-} \hat{\phi}_i(1, y) + \sum_i \underline{F}_i^{v\,+} \hat{\phi}_i(0, y) \right) \cdot \left(\hat{\phi}_j(1, y) \right) \Delta y \, d\hat{s} \\ &= \Delta y \sum_k \frac{1}{2} \left(\sum_i \underline{F}_i^{v\,-} \hat{\phi}_i(1, y_k) + \sum_i \underline{F}_i^{v\,+} \hat{\phi}_i(0, y_k) \right) \cdot \left(\hat{\phi}_j(1, y_k) \right) \, \omega_k \\ &= \Delta y \Lambda^\gamma \left(\frac{1}{2} P^{\text{right}} \underline{F}^{v\,-} + \frac{1}{2} P^{\text{left}} \underline{F}^{v\,+} \right). \end{aligned} \tag{4.19}$$

Here, the test function of the outer edge is eliminated because it does not project onto the cell C^-. Instead, it projects to the cell C^+ and will be considered as the projection from the left edge in the next cell to the right. For the other edges we get the following definitions:

$$\delta^f = \Delta x \, \Lambda^\delta \left(\frac{1}{2} P^{\text{Bottom}} \underline{F}^{v\,-} + \frac{1}{2} P^{\text{Top}} \underline{F}^{v\,+} \right) \tag{4.20}$$

$$\alpha^f = \Delta y \, \Lambda^\alpha \left(\frac{1}{2} P^{\text{Left}} \underline{F}^{v\,-} + \frac{1}{2} P^{\text{Right}} \underline{F}^{v\,+} \right) \tag{4.21}$$

$$\beta^f = \Delta x \, \Lambda^\beta \left(\frac{1}{2} P^{\text{Top}} \underline{F}^{v\,-} + \frac{1}{2} P^{\text{Bottom}} \underline{F}^{v\,+} \right). \tag{4.22}$$

Here the P-matrices are the projection matrices for the cell-data to the corresponding edge. $\underline{F}^v$ are the coefficients of the viscous flux on the 2D quadpoints. The edges are here named

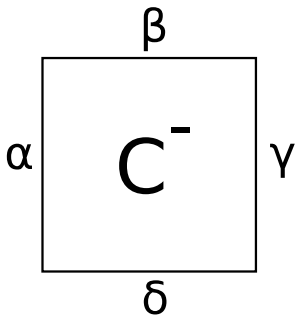

Figure 4.2.: The edge nomenclature with respect to the inner cell.

according to fig. 4.2. For the evaluation of $\underline{F^v}$ we have to make sure to use the transformed arguments $\mathcal{M}(\hat{\boldsymbol{x}})$ and $J^{-1}\hat{\nabla}\hat{\boldsymbol{x}}$.

The Λ matrices are defined based on the test functions for each edge as

$$\begin{aligned}
\Lambda^{\alpha}_{jk} &= -\left(\hat{\phi}_j(0, y_k)\right)\ \omega_k \\
\Lambda^{\beta}_{jk} &= \left(\hat{\phi}_j(1, x_k)\right)\ \omega_k \\
\Lambda^{\gamma}_{jk} &= \left(\hat{\phi}_j(1, y_k)\right)\ \omega_k \\
\Lambda^{\delta}_{jk} &= -\left(\hat{\phi}_j(0, x_k)\right)\ \omega_k.
\end{aligned}$$

These matrices will have the shape $order^d \times order$ and have the function of mapping the results, projected on the edge, back to the cell.

4.3.3. Symmetry Term

For the symmetry term we can follow the same logic as for the flux term. The G-matrices are a function of $\boldsymbol{u}$ and can therefore not be treated as a constant and taken out of the average operator. The explicit notation of the dependency on $\boldsymbol{u}$ is omitted in the following

derivation:

$$\begin{aligned}
e^s &= \int_e \{\!\{G(\boldsymbol{u}_h)\cdot\nabla\boldsymbol{v}\}\!\}\cdot[\![\boldsymbol{u}_h]\!]\,ds \\
&= \int_e \frac{1}{2}\left(G^-\cdot\nabla\boldsymbol{v}^- + G^+\cdot\nabla\boldsymbol{v}^+\right)\cdot(\boldsymbol{u}_h^- - \boldsymbol{u}_h^+)\otimes\boldsymbol{n}\,ds \\
&= \int_{\hat{e}} \frac{1}{2}\left(G^-\cdot J^{-1}(\hat{\nabla}\hat{\phi}_j^-) + G^+\cdot J^{-1}(\hat{\nabla}\hat{\phi}_j^+)\right)\cdot\left(\boldsymbol{u}_h^- - \boldsymbol{u}_h^+\right)\otimes\boldsymbol{n}\,\Delta e\,d\hat{s} \\
&= \int_{\hat{e}} \frac{1}{2}\left(G^-\cdot J^{-1}(\hat{\nabla}\hat{\phi}_j^-) + G^+\cdot J^{-1}(\hat{\nabla}\hat{\phi}_j^+)\right) \\
&\qquad\qquad \cdot\left(\sum_i \hat{\phi}_i^-\underline{\boldsymbol{u}}_i^- - \sum_i \hat{\phi}_i^+\underline{\boldsymbol{u}}_i^+\right)\otimes\boldsymbol{n}\,\Delta e\,d\hat{s}
\end{aligned} \tag{4.23}$$

As for the flux term, the derivation is continued for the edge γ, given by

$$\begin{aligned}
\gamma^s &= \int_\gamma \frac{1}{2}\left(G^-\cdot J^{-1}\hat{\nabla}\hat{\phi}_j(1,y)\right)\left(\sum_i \underline{\boldsymbol{u}}_i^-\hat{\phi}(1,y) - \sum_i \underline{\boldsymbol{u}}_i^+\hat{\phi}(0,y)\Delta y\,d\hat{s}\right) \\
&= \Delta y\sum_k \frac{1}{2}\left(G^-\cdot J^{-1}\hat{\nabla}\hat{\phi}_j(1,y_k)\right) \\
&\qquad\qquad \left(\sum_i \underline{\boldsymbol{u}}_i^-\underbrace{\hat{\phi}(1,y_k)}_{\delta_{ik}} - \sum_i \underline{\boldsymbol{u}}_i^+\underbrace{\hat{\phi}(0,y_k)}_{\delta_{ik}}\right)\omega_k \\
&= \Delta y\sum_k \frac{1}{2}\left(G^-\cdot J^{-1}\hat{\nabla}\hat{\phi}_j(1,y_k)\right)\left(P^{\text{right}}\underline{\boldsymbol{u}}^- - P^{\text{left}}\underline{\boldsymbol{u}}^+\right)\omega_k \\
&= \Delta y\sum_k \frac{1}{2}\left(G^-\cdot\Delta_{j,k}^{\gamma}\right)\left(P^{\text{right}}\underline{\boldsymbol{u}}^- - P^{\text{left}}\underline{\boldsymbol{u}}^+\right)
\end{aligned} \tag{4.24}$$

For the other three edges we get the following similar results

$$\alpha^s = \Delta y\sum_k \frac{1}{2}\left(G^-\cdot\Delta_{j,k}^{\alpha}\right)\left(P^{\text{Left}}\underline{\boldsymbol{u}}^- - P^{\text{Right}}\underline{\boldsymbol{u}}^+\right) \tag{4.25}$$

$$\beta^s = \Delta x\sum_k \frac{1}{2}\left(G^-\cdot\Delta_{j,k}^{\beta}\right)\left(P^{\text{Top}}\underline{\boldsymbol{u}}^- - P^{\text{Bottom}}\underline{\boldsymbol{u}}^+\right) \tag{4.26}$$

$$\delta^s = \Delta x\sum_k \frac{1}{2}\left(G^-\cdot\Delta_{j,k}^{\delta}\right)\left(P^{\text{Bottom}}\underline{\boldsymbol{u}}^- - P^{\text{Top}}\underline{\boldsymbol{u}}^+\right). \tag{4.27}$$

Here we have defined the new Δ-matrices. The have a similar shape to the Λ matrices and

they also carry the function of projecting the results defined on an edge back to the cell.

$$\begin{aligned}
\Delta^{\alpha}_{j,k} &= J^{-1}\hat{\nabla}\hat{\phi}_j(0, y_k)\ \omega_k \\
\Delta^{\beta}_{j,k} &= J^{-1}\hat{\nabla}\hat{\phi}_j(x_k, 1)\ \omega_k \\
\Delta^{\gamma}_{j,k} &= J^{-1}\hat{\nabla}\hat{\phi}_j(1, y_k)\ \omega_k \\
\Delta^{\delta}_{j,k} &= J^{-1}\hat{\nabla}\hat{\phi}_j(x_k, 0)\ \omega_k
\end{aligned} \tag{4.28}$$

The transpose of these matrices can be used to project the gradients of the degrees of freedom onto the edge.

4.3.4. Penalty Term

Now for the last edge term, the penalty term, we apply the same procedure

$$\begin{aligned}
e^p &= \int_e [\![\boldsymbol{u}_h]\!] \cdot [\![\boldsymbol{v}]\!]\ ds \\
&= \int_e (\boldsymbol{u}_h^- - \boldsymbol{u}_h^+) \otimes \boldsymbol{n} \cdot (\boldsymbol{v}^- - \boldsymbol{v}^+) \otimes \boldsymbol{n}\ ds \\
&= \int_{\hat{e}} \left(\sum_i \hat{\phi}_i^- \underline{\boldsymbol{u}}_i^- - \sum_i \hat{\phi}_i^+ \underline{\boldsymbol{u}}_i^+ \right) \otimes \boldsymbol{n} \cdot \left(\hat{\phi}_j^- - \hat{\phi}_j^+ \right) \otimes \boldsymbol{n}\ \Delta e\ d\hat{s}
\end{aligned} \tag{4.29}$$

Evaluated for the edge γ

$$\begin{aligned}
\gamma^p &= \int_{\hat{\gamma}} \left(\sum_i \hat{\phi}_i(1, y) \underline{\boldsymbol{u}}_i^- - \sum_i \hat{\phi}_i(0, y) \underline{\boldsymbol{u}}_i^+ \right) \left(\hat{\phi}_j(1, y) \right)\ \Delta y\ d\hat{s} \\
&= \Delta y \sum_k \Bigg(\sum_i \underbrace{\hat{\phi}_i(1, y_k)}_{P^{\text{right}}_{k,i}} \underline{\boldsymbol{u}}_i^- - \sum_i \underbrace{\hat{\phi}_i(0, y_k)}_{P^{\text{left}}_{k,i}} \underline{\boldsymbol{u}}_i^+ \Bigg) \underbrace{\left(\hat{\phi}_j(1, y_k) \right) \omega_k}_{\Lambda^{\gamma}_{j,k}} \\
&= \Delta y\, \Lambda^{\gamma} \left(P^{\text{right}} \underline{\boldsymbol{u}}^- - P^{\text{left}} \underline{\boldsymbol{u}}^+ \right)
\end{aligned} \tag{4.30}$$

For the other edges likewise

$$\delta^p = \Delta x \Lambda^{\delta} \left(P^{\text{bottom}} \underline{\boldsymbol{u}}^- - P^{\text{top}} \underline{\boldsymbol{u}}^+ \right) \tag{4.31}$$

$$\alpha^p = \Delta y \Lambda^{\alpha} \left(P^{\text{left}} \underline{\boldsymbol{u}}^- - P^{\text{right}} \underline{\boldsymbol{u}}^+ \right) \tag{4.32}$$

$$\beta^p = \Delta x \Lambda^{\beta} \left(P^{\text{top}} \underline{\boldsymbol{u}}^- - P^{\text{bottom}} \underline{\boldsymbol{u}}^+ \right) \tag{4.33}$$

The penalty parameter δ depends on the mesh and is given by Hartmann & Houston [16]

as the following equation

$$\delta = C\frac{\mu p^2}{\tilde{h}}. \tag{4.34}$$

where μ is the fluid viscosity, p is the polynomial order of the Lagrange polynomials, and $\tilde{h}$ is given as the smallest size of an adjacent cell to an edge, divided by the size of the edge. As we have square cells, $\tilde{h}$ will simple have the value of the edge length. The penalty coefficient C is a numerical value that depends on the mesh and it regulates the severity of the penalty. It must be above a treshold value to guarantuee stability. Hartmann & Houston [16] have empirically chosen $C = 10$ for the compressible Navier-Stokes discretization.

$$\begin{aligned} \delta_e &= \max(C_{k^+}, C_{k^-}) \\ C_k &= \frac{(p+1)(p+d)}{d}\frac{\mathcal{A}(\partial C/\ \partial\Omega)/2 + \mathcal{A}(\partial C \cap \partial\Omega)}{\mathcal{V}(C)}, \qquad C \in T_h \end{aligned} \tag{4.35}$$

Here d is the dimensionality of the simulation, in our case $d = 2$. The area and volume of a cell are given here by $\mathcal{A}(C)$ and $\mathcal{V}(C)$ respectively. Since we work in 2D, these are respectively given by the total length of all edges of the cell, and the area of the cell. The penalty parameter is here locally defined for an edge e, to get the global penalty parameter, we simply take the maximum values over all edges. The more attentive eye will notice that the viscosity is missing in the latter definition. This would be because the latter definition is a general expression for the penalty parameter.

4.4. Boundary Conditions

For the approach discussed until no, we assumed periodic boundary conditions, where the domain theoretically stretches into infinity and all edges could be considered to be internal edges. To obtain a model that approximates a fuel cell more realistically, we have to define a finite domain, surround the domain with a boundary and set bondary conditions. This section introduces the boundary conditions and the discretization thereof, following Hartmann & Leicht [17].
We define the following four types of boundary types:

Periodic - This is the boundary condition that was assumed so far. No explicit boundary state is defined and the state of the ghost layers of the simulation is equal to the corresponding inner cell at the opposite side of the domain.

Adiabatic, no-slip wall - An adiabatic, no-slip wall is chosen for the walls in the fuel cell solver. For an adiabatic wall, no heat transfer across the wall occurs, whereas in reality, heat transfer will occur between the channel wall and the flow, where there is a difference in temperature. This would call for a solver that approximates the wall temperature distribution based on the heat generated during the electrochemical process and is coupled to ours. Since this extension is out of the scope of this thesis, we will assume the walls to

have the same temperature as the flow and are, therefore, assumed to be adiabatic.
The no-slip condition is in this situation justified for the walls, as there is generally a stagnant flow directly at surfaces. The boundary conditions given below, are given for subsonic flow.

Inflow - In order to specify the velocity, density and pressure of the flow entering the computational domain, inflow boundary conditions must be set. The inflow conditions can be set by freestream flow profiles, but we will assume a uniform distribution along the inflow boundary, so a single Dirichlet boundary value for the inflow suffices.

Outflow - The state of the outflow of a flow channel cannot be predicted and if a Dirichlet boundary condition were to be set on the outflow, it would result in either an ever increasing or decreasing density, because mass must be conserved. However, in the subsonic flow velocity regions, we want to prevent any discontinuities, such as shocks, within the compuational domain or near the outlet outside the computational domain. Therefore, we set Neumann boundary conditions at the outflow to enforce no change in density, velocity or pressure after the flow leaves the computational domain.
The boundary conditions are given as a function of degrees of freedom in the neighbouring inner cell and are given by

$$\boldsymbol{u}_{\Gamma,\text{adia}}(\boldsymbol{u}) = \begin{pmatrix} \rho \\ 0 \\ 0 \\ \rho E \end{pmatrix}, \quad \boldsymbol{u}_{\Gamma,\text{in}}(\boldsymbol{u}) = \begin{pmatrix} \rho_{\text{in}} \\ \rho u_{\text{in}} \\ \rho v_{\text{in}} \\ \frac{p_{\text{in}}}{\gamma-1} + \frac{u_{\text{in}}^2+v_{\text{in}}^2}{2\rho_{\text{in}}} \end{pmatrix}, \quad \boldsymbol{u}_{\Gamma,\text{out}}(\boldsymbol{u}) = \begin{pmatrix} \rho \\ u \\ v \\ \rho E \end{pmatrix}$$

4.4.1. Discretization at the Wall

The wall boundary can be discretized, based on the interior numerical fluxes or on normal boundary fluxes. The former being the more stable, and the latter being the more accurate option [17]. We will choose the latter boundary fluxes, which are given as

$$\begin{aligned} \hat{\boldsymbol{u}}_h &= \boldsymbol{u}_\Gamma(\boldsymbol{u}_h^-), \\ \hat{\sigma}_{\Gamma,h} &= \boldsymbol{F}^v_{\text{adia}}(\boldsymbol{u}_\Gamma(\boldsymbol{u}_h^-), \nabla \boldsymbol{u}_h^-) - \delta_\Gamma(\boldsymbol{u}_h^-). \end{aligned} \tag{4.36}$$

In the viscous flux and the corresponding homogeneity tensor G Γ_{adia}, the temperature derivatives are set to zero. The wall edge terms are given by

$$\begin{aligned} \int_{\Gamma_W} \boldsymbol{n} \cdot (\boldsymbol{F}^c(\boldsymbol{u}_\Gamma) - \boldsymbol{F}^v_{\text{adia}}(\boldsymbol{u}_\Gamma, \nabla \boldsymbol{u}_h^-) + \delta_\Gamma(\boldsymbol{u}_h^-) \cdot \boldsymbol{v}^- \, ds \\ - \int_{\Gamma_W} (\boldsymbol{u}_h^- - \boldsymbol{u}_\Gamma) \otimes \boldsymbol{n} \cdot (G^T(\boldsymbol{u}_h^-) \nabla \boldsymbol{v}_h^-) \, ds. \end{aligned} \tag{4.37}$$

4.4.2. Discretization at the Farfield Boundary

The numerical fluxes for the inflow or outflow boundaries are given as

$$\begin{aligned} \hat{\boldsymbol{u}}_h &= \boldsymbol{u}_\Gamma(\boldsymbol{u}_h^-), \\ \hat{\sigma}_{\Gamma,h} &= \boldsymbol{F}^v(\boldsymbol{u}_\Gamma(\boldsymbol{u}_h^-), \nabla \boldsymbol{u}_h^-) - \delta_\Gamma(\boldsymbol{u}_h^-). \end{aligned} \tag{4.38}$$

The corresponding boundary edge terms are then defined as

$$\begin{aligned} \int_{\Gamma_{I,O}} \boldsymbol{n} \cdot (\boldsymbol{F}^c(\boldsymbol{u}_\Gamma) - \boldsymbol{F}^v(\boldsymbol{u}_\Gamma, \nabla \boldsymbol{u}_h^-) + \delta_\Gamma(\boldsymbol{u}_h^-) \cdot \boldsymbol{v}^- \, ds \\ - \int_{\Gamma_{I,O}} (\boldsymbol{u}_h^- - \boldsymbol{u}_\Gamma) \otimes \boldsymbol{n} \cdot (G^T(\boldsymbol{u}_h^-) \nabla \boldsymbol{v}_h^-) \, ds. \end{aligned} \tag{4.39}$$

4.5. Time-Stepping Method

We are mainly using the explicit euler method for timestepping. The explicit Euler scheme is given by

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \, \frac{\partial \boldsymbol{u}}{\partial t} \tag{4.40}$$

For the explicit Euler, the timestepsize is defined by the following formula

$$\Delta t \leq \mathrm{CFL} \, \alpha(p) \, h \sum_{i=1}^{d} \frac{1}{|\lambda_c^{max}|_i + |\lambda_v^{max}|_i \frac{1}{\alpha(p) h}}, \tag{4.41}$$

where $\alpha(p) \leq (2p+1)^{-1}$ [18, 14]. Here, p denotes the polynomial order of the basis functions. The maximum eigenvalues of the Jacobian of the hyperbolic and viscous flux are denoted by $_c$ and λ_v. The maximum eigenvalues are defined for the compressible NSE as

$$\begin{aligned} |\lambda_c^{max}| &= |\boldsymbol{v}_n| + c, \\ |\lambda_v^{max}| &= \max\left(\frac{4\mu}{3\rho}, \frac{\gamma\mu}{\mathrm{Pr}\rho}\right). \end{aligned} \tag{4.42}$$

Here, c is the speed of sound given by

$$c = \sqrt{(\gamma R T)}, \tag{4.43}$$

where $R = 8.314$ is the universal gas constant and γ and T are the heat capacity ratio and temperature respectively. There are two maximum eigenvalues because there are two different fluxes. For the implementation of the penalty term, the maximum timestepsize must be adapted as is described in section 3.4. As there is no sharply defined boundary on

the maximum timestep due to the penalty coefficient, it will be set empirically through the CFL condition.

5. Extension to System of Equations

In the previous chapter we discussed the compressible Navier-Stokes equations and we went over the numerical treatment of the viscous flux. Now it is time to extend the system to describe the behaviour of species in the flow and to include porosity effects based on Sun [30]. The M^2 model, which was explained in section 2.2, is a rather complicated model, which means that it leaves room for a lot of mistakes in the implementation thereof. The goal of this chapter is to present an extension to the compressible NSE that functions as a simplified model of the flow regions in a PEMFC, while still capturing the most relevant phenomena.
A few simplifying assumptions will be introduced in section 5.1 and the numerical treatment of source-terms in a discontinuous Galerkin model is discussed in section 5.2. The extensions of the NSE will be given in section 5.3. This section will be concluded with a complete system for a compressible reactive flow in porous regions.

5.1. Simplifying Assumptions

The M^2 model presented in 2.2 is an extensive model, that captures most of the physicsal phenomena that occur in a PEMFC during operation. For the initial approach towards a discontinuous Galerkin discretization of this model, it is deemed too complex to include all physical activities that are described by the M^2 model. Including all the physical phenomena would introduce too much room for mistakes in the code for this thesis, and including everything would divert us from the initial intend of this thesis to suggest an approach for the use of the discontinuous Galerkin discretization for a PEMFC model. For future work, it is recommended that the simplifications mentioned in this section, be replaced by their physically correct counterparts.
Each assumption made will be clarified and validated below. The assumptions made are shortly summarized as:

- In the Species Equation, the relative motion due to condensation or evaporation is neglected.
- The effective diffusivity $D_g^{k,\text{eff}}$ is assumed to be a function of the diffusivity D_g^k and porosity ϵ only.
- The source term S_k only depends on the presence of electrochemical reactions.
- A uniformly distributed overpotential η is assumed for electrochemical reactions.

- The advection correction factor γ_c is equal to unity.

Neglected Relative Motion

The Species Equation was given in section 2.2 as

$$\nabla \cdot (\gamma_c \boldsymbol{v} C^k) = \nabla \cdot (D_g^{\mathrm{k,eff}} \nabla C_g^k) - \nabla \cdot \left[\left(\frac{m f_l^k}{M^k} - \frac{C_g^k}{\rho_g} \right) \boldsymbol{j}_l \right] + S_k.$$

The term on the left-hand side describes the advection of a species within the flow. The first term on the right-hand side is a diffusion term describing the diffusion of gases in the flow. This term is regarded highly relevant, as it may demonstrate a significant difference between the flow behaviour in the gas channel and in the porous region. The second term on the right-hand side describes the flow of a species due to the relative motion of liquid to gas phase under capillary action in the porous regions [30]. Although it is a term that will also illustrate a difference in flow behaviour between the porous and non-porous regions, it is driven by the transformation of water from a liquid into a vapor and vice-versa.
Firstly, on the anode side, only the presence of hydrogen and nitrogen are assumed. In reality there is also water present in both liquid and vapor form, due to the wetting of the membrane, but the capturing of this phenomenon is not the intend of this thesis. This term will be neglected on the anode side. Secondly, the complexity of this term, which lies mainly in its nonlinearity and dependence on many variables, makes it preferable to ignore this term for the time being.

Effective Diffusivity

The effective gas diffusivity coefficient $D_g^{k,\mathrm{eff}}$ is described by Sun [30] as a function of the gas diffusivity coefficient, the porosity of the medium and the volumetric fraction occupied by water in either the liquid or the gas phase. For simplicity, the effective gas diffusivity coefficient is chosen to depend only on the gas diffusivity coefficient of a species, and the porosity of a medium, because these are quantities that can be predefined. The volumetric fraction of water in liquid or vapor form requires an extra calculation step, so the volumetric fraction is simply assumed to be one ($s = 1$). This means that we assume that water, if present at all, is only present in vapor form in this step.

The Source Term S_k

The source term S_k is assumed to only be a function of electrochemical reactions. This means that the electro-osmotic drag, which is the second source term that is used in the species equation, is neglected. The electro-osmotic drag describes the movement of water through a polymer elektrolyte due to the protonic current. Essentially, the transfer of hydrogen and oxygen are not directly affected by this term, but they are indirectly affected

by this term because the movement of water also excites the movement of hydrogen and oxygen. This term depends on the derivative of the proton flux through the polymer membrane, and would excessively complicate the system. The intend of this thesis is to capture the movement of species due to advection and diffusion, and the effect of electro-osmotic drag will therefore be neglected.

The Overpotential

The model of a PEMFC is multiphysical, and as we focus on the flow regions of the fuel cell, we do not model the electrochemical reactions themselves, nor do we model the electronic potentials of the electronic or protonic phases. The reaction rate of electrochemical reactions does, however, depend on the overpotential between regions. For now, a uniformly distributed overpotential η will be introduced, to investigate the depletion rate of reactants for different overpotentials.

The Advection Correction Factor

The advection correction factor γ_c is introduced to take the difference of flow-fields of the liquid and gas phases into account. This variable is equal to one in the gas channels, and in our case it will also be assumed to be one in the porous regions. It should model the effect of liquid water filling the pores and thereby blocking multiple pathways for the gas to flow through the medium, but as was already mention in the assumption for *Effective Diffusivity*, water is assumed to be present purely in vapor form to simplify the model. This means that the advection correction factor is equal to one, also in the porous regions.

Justification of the Simplifications

The simplifications mentioned will make the model less accurate, or at least less accurate than currently existing models of a PEMFC. However, the intend of the solver is to present a discontinuous Galerkin approach for a PEMFC model. The terms or effects that have been neglected can of course be implemented in any later work done on the solver, but they are momentarily not necessary for the illustration of the approach.

5.2. Godunov Splitting Method

Before we start discussing the extensions that are made to the Navier-Stokes system, the methods for implementing source terms should be considered. In our consideration we will mainly focus on the splitting methods. The main idea for splitting methods is that the equations are first solved homogeneously and then the heterogenous part is solved as an ODE. For the analysis of splitting methods, we mainly focus on the examples given by LeVeque [23].

Godunov Splitting

A standard approach for solving source terms, is to use the *fractional-step* or *operator-splitting method*, where we alternate between solving the homogeneous problem $\boldsymbol{u}_t^* + \boldsymbol{F}(\boldsymbol{u})_x = 0$ and the ODE $\boldsymbol{u}_t = \boldsymbol{\psi}(\boldsymbol{u}^*)$ to solve the full problem

$$\boldsymbol{u}_t + \boldsymbol{F}(\boldsymbol{u})_x = \boldsymbol{\psi}(\boldsymbol{u}), \tag{5.1}$$

where ψ is used to denote the heterogenous part of the equation. Note that ψ does not necessarily have to depend on $\boldsymbol{u}$ only, but it can also depend on its derivatives or other variables. To solve the ODE, we can use standard one-step methods such as the Explicit Euler method.
For a problem with the two operators $\mathcal{A}$ and $\mathcal{B}$, the *commutator* is defined as $\mathcal{AB} - \mathcal{BA}$. In a situation where the two operators commute, the commutator is equal to zero. This means that it does not matter in which order the operators are applied to the variables, and that there is no splitting error. If the two operators do not commute, there is a difference in the result when we use the splitting method, compared to when we do not use the splitting method (the unsplit method here being the original heterogenous equation). This difference in result is called the *splitting error*. The splitting error can be analysed by using the Taylor series expansions. This error is first order accurate in time for the Godunov splitting, but it is often smaller than the discretization error [23].

Strang Splitting

The Strang splitting method is an alternative to the Godunov splitting method and it is second order accurate in time. The idea is to first solve the first subproblem over only half a timestep. Then we use that result as data for a full time step on the second subproblem. Then the entire sequence is repeated. This would formally result in a splitting scheme with a second order accuracy in the splitting error, in case the two operators do not commute. The Godunov scheme is formally only first-order accurate, but the accuracy between the Godunov and Strang splitting is in practice often indistinguishable [23].

Pittfalls

There are a few pittfalls that we should be aware off before continuing. Firstly, in the case that two operators do not commute, a splitting error arises. This in itself is not a problem at the moment, when one is satisfied with an accuracy of first order. Secondly, in problems with *stiff* source terms, it might be necessary to use an implicit method for the ODE problem. The necessity for an implicit method can arise in the case of a reactive flow, when the reaction happens on a much smaller timescale than the time scale of the fluis dynamics [23]. Since we are considering a reactive flow in our case, it is important to keep this possibility in mind.

5.3. Extended System of Equations

Now that we have examined all the necessary tools for the discontinuous Galerkin method, the extensions to the Navier-Stokes system can be discussed. Some modifications to the current homogeneous compressible NSE system will need to be made to simulate flow through porous regions and to compute the concentration of species in the flow. In short, the extensions that are made are summarised as:

- Adding the possibility for porous flow by adding the source term S_u to the momentum equations.
- Adding a species equation for each species that is relevant in the flow region and simulate electrochemical reactions by adding source/sink terms to the species equations.
- Adding a source term to the energy equation to account for the change in energy following from reactions.

The Momentum Equation

For the addition of porous regions the momentum equations have been adapted, following the example set by following the example set by Sun [30], by adding a porosity fraction ε in front of the first term on the right-hand side and by adding the source term S_u, which describes the Darcy force, giving

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho u\\ \rho v\end{pmatrix} + \nabla\cdot\begin{pmatrix}\frac{1}{\varepsilon^2}\rho u^2 & \frac{1}{\varepsilon^2}\rho uv\\ \frac{1}{\varepsilon^2}\rho uv & \frac{1}{\varepsilon^2}\rho v^2\end{pmatrix} - \nabla\cdot\begin{pmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{21} & \sigma_{22}\end{pmatrix} = \boldsymbol{S_u}, \tag{5.2}$$

where ε is equal to 1 for the gas channels, and ranges between 0.4 and 0.6 for the gas diffusion layers and catalyst layers. The source term is defined as

$$\boldsymbol{S_u} = -\mu K^{-1}\boldsymbol{v} = -\mu\begin{pmatrix}\frac{1}{K_{xx}}u\\ \frac{1}{K_{yy}}v\end{pmatrix}. \tag{5.3}$$

Here the *through-plane* and *in-plane* permeability functions are given as $8.69\cdot 10^{-12}$ m^2 and $1.9\cdot 10^{-12}$ m^2 respectively in the porous regions [30]. Depending on the orientation of the porous region in the simulation, K_{xx} and K_{yy} receive their value accordingly. The *through-plane* and *in-plane* flow directions are in our case given by the flow in x-direction and the y-direction respectively. Since the permeability tensor K has such small values, the source terms will lead to a rather stiff ODE in the splitting method. Therefore, an implicit timestepping method will need to be used to solve the hetrogeneous part of the equation.

The Species Equations

To be able to capture the behaviour of the different species (H_2, O_2 and H_2O), a species function is added to the system for each species that is present in a region. This means that there is a hydrogen function added on the anode side, and on the cathode side the oxygen and water functions are added. The depletion and creation of species is captured in the corresponding source term S_k, which only describes the depletion and creation due to electrochemical reactions. The species equation for hydrogen is

$$\frac{\partial}{\partial t}\rho C_{H_2} + \nabla \cdot (\rho \boldsymbol{v} C_{H_2}) = \nabla \cdot \left(D_g^{H_2,\text{eff}} \nabla C_g^{H_2} \right) + S_{H_2}. \tag{5.4}$$

The effective gas diffusivity coefficient is given according to Sun [30] as

$$D_g^{H_2,\text{eff}} = D_g^{H_2} f(\varepsilon), \tag{5.5}$$

$$f(\varepsilon) = \begin{cases} \varepsilon \left(\frac{\varepsilon - \varepsilon_p}{1 - \varepsilon_p} \right)^{\alpha} & \text{in GDLs} \\ \varepsilon^{1.5} & \text{in CLs} \end{cases}, \qquad \alpha = \begin{cases} 0.521 & \text{in-plane} \\ 0.785 & \text{through-plane} \\ 0 & \text{in gas-channel} \end{cases}.$$

Here, $D_g^{H_2} = 1.1028 \cdot 10^{-4}$ m^2/s is the gas diffusivity for hydrogen and $\varepsilon_p = 0.11$ is the percolation threshold. The source term for the hydrogen equation is

$$S_{H_2} = \begin{cases} -\frac{j_a}{2F} & \text{in CLs} \\ 0 & \text{otherwise} \end{cases}, \qquad j_a = a\, i_{0,a} \sqrt{\frac{C_{H_2}}{C_{H_2}^{\text{ref}}}} \left(\frac{\alpha_a + \alpha_c}{RT} F \eta \right). \tag{5.6}$$

Here the volumetric transfer current density of hydrogen on the anode side j_a is derived from the Butler-Volmer equations, F is the Faraday constant and η describes the overpotential and is to be set explicitly at a constant value, before running the solver. Furthermore, $a\, i_{0,a} = 10^9$, $C_{H_2}^{\text{ref}} = 44.58$ mol/m^3 and $a_\alpha + a_c = 2$ [30]. These constants are given for each species for the cathode or anode side. The explicit definitions of the oxygen and water equations, including their source terms, can be found in Sun [30].

The Energy Equation

Apart from influencing the composition of reactants in the flow, electrochemical reactions generally also have an effect on the total energy of a system. To account for this, we have to add a source term to the energy equation that describes the change in energy in the system due to electrochemical reactions

$$\frac{d}{dt}\rho E + \nabla \cdot \begin{pmatrix} \rho u E & \rho v E \end{pmatrix} - \nabla \cdot \begin{pmatrix} \sigma_{11} u + \sigma_{12} v + k T_x & \sigma_{11} u + \sigma_{12} v + k T_x \end{pmatrix} = S_T. \tag{5.7}$$

The change in energy due to evaporation and condensation of water has been neglected in our solver and the ohmic effects (energy increase due to electricity resistance) have been omitted, since the solver does not solve for the potential distribution. The source term is

$$S_T = \begin{cases} j_a(\eta) & \text{in the anode CL} \\ j_c(\eta + T\frac{dU_0}{dT}) & \text{in the cathode CL} \\ 0 & \text{otherwise} \end{cases}. \tag{5.8}$$

Here, U_0 is the thermodynamic equilibrium potential of the reaction and $\frac{dU_0}{dT} = -0.9 \cdot 10^{-3}$ in the cathode, and zero in the anode [26]. The volumetric transfer current density the cathode side j_c is given by

$$j_c = -a\, i_{0,c}\, e^{-16456\left(\frac{1}{T} - \frac{1}{353.15}\right)} \frac{C_{O_2}}{C_{O_2}^{\text{ref}}} e^{-\frac{\alpha_c F}{RT}\eta}. \tag{5.9}$$

This equation is also derived from the Butler-Volmer equation, and $ai_{0,c} = 3.5 \cdot 10^4$ in the cathode CL, $\alpha_c = 1$ and $C_{O_2}^{\text{ref}} = 40.88\ \text{mol/m}^3$ [30]

The Extended System

Starting from the Navier-Stokes system in equation 4.1, and including the above definitions to the system, we get the following result for the extended system on the anode side

$$\frac{\partial}{\partial t}\boldsymbol{u} + \nabla \cdot \tilde{\boldsymbol{F}}^h(\boldsymbol{u}) - \nabla \cdot \tilde{\boldsymbol{F}}^v(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{S}, \tag{5.10}$$

where the degrees of freedom, hyperbolic and viscous fluxes have the following definitions:

$$\begin{aligned} \boldsymbol{u} &= \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \\ \rho C_{H_2} \end{pmatrix}, \qquad \tilde{\boldsymbol{F}}^h(\boldsymbol{u}) = \begin{pmatrix} \rho u & \rho v \\ \frac{1}{\varepsilon^2}\rho u^2 + p & \frac{1}{\varepsilon^2}\rho uv \\ \frac{1}{\varepsilon^2}\rho uv & \frac{1}{\varepsilon^2}\rho v^2 + p \\ u(\rho E + p) & v(\rho E + p) \\ \rho u C_{H_2} & \rho v C_{H_2} \end{pmatrix}, \\ \tilde{\boldsymbol{F}}^v(\boldsymbol{u}, \nabla \boldsymbol{u}) &= \begin{pmatrix} 0 & 0 \\ \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \\ \sigma_{11}u + \sigma_{12}v + kT_x & \sigma_{21}u + \sigma_{22}v + kT_y \\ D_g^{H_2,\text{eff}} C_x^{H_2} & D_g^{H_2,\text{eff}} C_y^{H_2} \end{pmatrix}. \end{aligned} \tag{5.11}$$

The pressure still follows from the ideal gas law in equation 4.4.

Discretization of the Source Terms

The source term is defined as

$$\boldsymbol{S} = \begin{pmatrix} 0 \\ S_u^x \\ S_u^y \\ S_T \\ S_{H_2} \end{pmatrix} = \begin{pmatrix} 0 \\ -\mu K_{xx} u \\ -\mu K_{yy} v \\ j_a(\eta_a + T\frac{dU_0}{dT}) \\ -\frac{j_a}{2F} \end{pmatrix}. \tag{5.12}$$

For the evaluation of the source terms, the Godunov splitting method is applied. The values for K_{xx} and K_{yy} are in the order of magnitude of 10^{-12} and very small compared to the degrees of freedom. This means that the source term might be stiff and, therefore, lead to very small maximum timestep sizes for the explicit Euler scheme for the heterogenous part of the extended system. To circumvent this issue, we choose the implicit Euler method for the source term of the momentum equation $\boldsymbol{S_u}$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t\, \boldsymbol{\psi}(\boldsymbol{u})^{n+1}, \tag{5.13}$$

where n indicates the current timestep and Δt is the same stepsize as for the full step. The momentum source term does not depend on the temperature or species concentration, so it can be considered separately. The step for the momentum source term can be written as

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho u \\ \rho v \end{pmatrix} = -\frac{\mu}{\rho^n}\begin{pmatrix} K_{xx} & 0 \\ 0 & K_{yy} \end{pmatrix}\begin{pmatrix} \rho u \\ \rho v \end{pmatrix}^{n+1} = A(\boldsymbol{u}^n)\boldsymbol{u}^{n+1}. \tag{5.14}$$

The system to be solved for the implicit Euler scheme is

$$(\mathcal{I} - \Delta t A(\boldsymbol{u}^n))\boldsymbol{u}^{n+1} = \boldsymbol{u}^n. \tag{5.15}$$

The matrix $A(\boldsymbol{u})$ is a diagonal matrix, hence the inverse of the system is trivial

$$\begin{pmatrix} \rho u \\ \rho v \end{pmatrix}^{n+1} = \begin{pmatrix} \frac{1}{1+\Delta t \frac{\mu}{\rho^n} K_{xx}} & 0 \\ 0 & \frac{1}{1+\Delta t \frac{\mu}{\rho^n} K_{yy}} \end{pmatrix}\begin{pmatrix} \rho u \\ \rho v \end{pmatrix}^{n} \tag{5.16}$$

The source terms for the energy and species equations do not have a significantly higher or lower order of magnitude with respect to the degrees of freedom. These terms will therefore be evaluated explicitly. For the evaluation of the energy and species source terms, the newly updated values for ρu and ρv will be used.

Discontinuous Galerkin Discretization of the Extended System

The extension of the compressible NSE will call for some adjustments to the discretization process presented in section 4.2 and section 4.3. The fluxes $\boldsymbol{F}(\boldsymbol{u})$ and $\boldsymbol{F}(\boldsymbol{u}, \nabla \boldsymbol{u})$ are

replaced by their extended counterparts $\tilde{\boldsymbol{F}}(\boldsymbol{u})$ and $\tilde{\boldsymbol{F}}(\boldsymbol{u}, \nabla \boldsymbol{u})$. The same goes for the homogeneity tensor G, which is replaced by its extended counterpart $\tilde{G}$ and given in appendix A. The numerical flux for the SIPG method is adjusted to

$$\hat{\boldsymbol{u}}_h = \{\!\{\boldsymbol{u}_h\}\!\}, \qquad \hat{\sigma}_h = \{\!\{\tilde{G}(\boldsymbol{u}_h)\}\!\} - \delta [\![u_h]\!]. \tag{5.17}$$

Time-Stepping

From the Jacobian matrix of the extended system it follows that the maximum eigenvalue for the viscous flux has to be chosen according to the following definition

$$|\lambda_v^{\max}| = \max\left(\frac{4\mu}{3\rho}, \frac{\gamma\mu}{\mathrm{Pr}\rho}, \frac{D_k}{\mu}\right). \tag{5.18}$$

Part III.

Implementation and Verification

6. Code Implementation

The solver was originally used in the course "M.Sc. Praktikum: Modern Wave Propagation - Discontinuous Galerkin Julia" by Krenz & Reinarz [22] at the Chair of Scientific Computing of the Technische Universität München. The solver is implemented in the programming language Julia (v1.6). Julia is programming language with properties that make it easy to use for experimental numerical implementations. There is a significant community working with Julia in academia and Julia packages are usually distributed with an open source license.
The already existing version of the solver could be used to solve hyperbolic conservation equations using the discontinuous Galerkin method with solvers for the hyperbolic numerical flux. This means that, for example, the Euler equations could already be solved.
In this chapters we will go over the implementation steps that were done to arrive at a PEMFC solver. We will start with an introduction to the construction of the solver in section 6.1, where we include some code listings to illustrate the structure. With this knowledge we will go into the adjustments that were made during this thesis, starting with the grid in section 6.2. Then the addition of the viscous volume term is explained in section 6.3, after which the implementation of the edge terms will be discussd in section 6.4. This chapter ends with an explanation of the source term implementation in section 6.5.

6.1. Structure of the Solver

The basic working of the solver can be understood from the pseudocode given in listing 6.1. The main function needs a configuration file as an argument. This configuration file contains information on the settings for the simulation, such as the grid size, end time, timestepping scheme, output frequency, etc. Based on these settings, we can build a grid, initialize a time integrator for the timestepping scheme and define global matrices. This is all done at the very beginning of the code.
Then the initial condition is set for all cells in the domain. The initial conditions are defined in a function that is unique for each combination of system of equations and scenario that the solver can solve.
A vtk plotter is initiated to regulate the output of the solver. The output is saved in vtk files, that can be opened with a post-processing tool.
The main part of the runtime will be spent inside the while loop in lines 11-22 in listing 6.1. One iteration of this time loop is a complete timestep evaluation. The maximum timestep size is evaluated during every iteration and for the initial timestep the eigenvalues of the

initial conditions are used to set the maximum timestepsize. In line 16, the right-hand side of equation 4.16 is evaluated. The degrees of freedom are updated afterwards by multiplying the solution of the right-hand side evaluation with the inverse of the mass-matrix of the system, giving the solution of the homogeneous problem for a timestep. Finally, the degrees of freedom are updated to the solution of the heterogeneous problem in line 17, where the source terms are evaluated according to the splitting scheme.

Listing 6.1: Pseudocode of the main structure of the solver.

```
1  function main(configfile)
2          load_configuration;
3          build_grid;
4          make_time_integrator;
5          define_global_matrices;
6          for all C ∈ T_h
7                  set_initial_values;
8          end
9          initiate_vtk_plotter;
10         t = 0, dt = 0;
11         while t < t_end
12                 if t = 0
13                         compute_initial_eigenvalues;
14                 else
15                         evaluate_dt;
16                         evaluate_RHS;
17                         evaluate_Source;
18                 end
19                 if (t ≈ t_plot)
20                         plot;
21                 end
22         end
23 end
```

6.1.1. Timestepping

The iteration that is listed in lines 11-22 in listing 6.1 shows the steps that are taken for a single timestep. However, for a higher-order timestepping scheme, such as Runge-Kutta methods, intermediate steps are computed before a complete step is taken. Therefore, we use a timestep integrator function, which is listed for the explicit Euler scheme in listing 6.2. This integrator is "wrapped" around the evaluations of the right-hand side and the source term. If a higher-order time stepping scheme is chosen, the time integrator will

control the intermediate evaluations of the right-hand side and the source terms, to get a more accurate approximation of the solution in time, after a single timestep. Here the function `f(...)` is the closure that evaluates the right-hand side or the source terms. The function `step` takes here another (higher-order) function as an argument.

Listing 6.2: Function for explicit Euler update

```
1 function step(f, integrator::ExplicitEuler, grid, dt)
2     integrator.dofsupdate .= 0.0
3     f(integrator.dofsupdate, grid.dofs, grid.time)
4     grid.dofs .+= dt .* integrator.dofsupdate
5 end
```

6.1.2. Global Matrices

The advantage of defining the discontinuous Galerkin approach on a reference cell is that many matrices and functions can be precomputed on the reference cell and then mapped to the physical cell. This mapping is rather cheap for affine transformations, thus it saves runtime and memory space. In the solver we use the struct `global_matrices` to hold all these precomputed matrices and the struct is initiated and defined in line 5 of listing 6.1. The construction of the global matrices depends on the shape and size of the reference cell and on the basis that is used for the approximation of the solution.

Listing 6.3: Basis struct

```
1 struct Basis
2     quadpoints::Array{Float64,1}
3     quadweights::Array{Float64,1}
4     order::Int64
5     dimensions::Int64
6 end
```

In the solver, the Lagrange basis is implemented with a `basis` object, defined with the basis struct given in listing 6.3. This object stores the nodal points and quadrature weights of the Gaussian quadrature for a specified dimensionality (we generally work with 2D problems) and polynomial order. These points are used to define a set of Lagrange polynomials and their corresponding derivatives. With the basis set, we can calculate global matrices such as the mass-matrix, derivative matrices and face projection matrices which are subsequently stored in a `global_matrices` object.

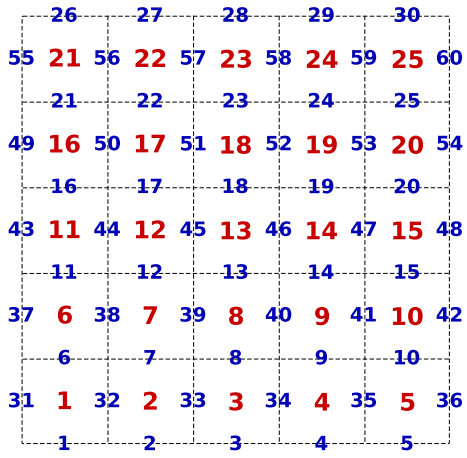

Figure 6.1.: The numbering of cells and edges in a cartesian grid. The red numbers indicate the IDs of the cells, the blue numbers the IDs of the edges.

6.2. Grid

In numerical simulations, the grid describes the computational domain that is used to solve the problem. During a simulation run, the `grid` object holds much of the general information about the simulation itself, such as the current time, the global max eigenvalues and a set of arrays of cells and edges. Apart form that, the grid object also holds all the data of the degrees of freedom (`dofs`) and fluxes, using the *Structure of Arrays* approach. This means that the simulation data is not saved in a `cell` object, as one might expect, but all the simulation data is actually held by the `grid` object. The shape of `dofs` and `flux` arrays is given by

$$\texttt{dofs} = [\texttt{order}^2, \texttt{ndofs}, \texttt{gridsize}]$$
$$\texttt{flux} = [2 * \texttt{order}^2, \texttt{ndofs}, \texttt{gridsize}].$$

Here `order` is the polynomial order of the basis function, `ndofs` is the number of degrees of freedom, and gridsize is total amount of cells in the grid. The `flux` array takes twice the amount of space as `dofs`, as it needs to store information at each point for two directions - Fx and Fy in a 2D cartesian grid. The grid struct is given in listing 6.4
In our case we take a regular grid, so the grid-cells are rectangular. The cells and edges are stored in a row-wise fashion, from top to bottom, as shown for a 5 by 5 uniform grid in fig. 6.1. A `cell` object stores general information such as its global position, size and cell ID. Additionally, a cell stores the IDs of the enclosing edges and the IDs of the neighbouring cells. Its struct is given in listing 6.5.

Listing 6.4: Grid struct

```
mutable struct Grid
    basis::Basis
    cells::Array{Cell,1}
    edges::Array{Edge,1}
    size::Array{Float64,1}
    dofs::Array{Float64,3}
    flux::Array{Float64,3}
    flux_visc::Array{Float64,3}
    maxeigenval::Float64
    time::Float64
end
```

The edges are stored such that the horizontal edges are stored before the vertical edges (see the ordering given in fig. 6.1). An `Edge` object stores information on the orientation of an edge, its length, its ID, and the IDs of the neighbouring two cells. In listing 6.6 the normal is set to (1,0) and (0,1) for vertical and horizontal edges respectively. This corresponds to the orientation of the edge that is defined in fig. 4.1. The `neighborsidx` array also stores the IDs of the neighbouring cells according to fig. 4.1. The left or top cell is stored first and the right or bottom cell is stored next. Apart from that, the edge object also contains a buffer, the content of which may be changed during runtime, making it a mutable object. The function of this buffer is to hold edge term values for a timestep, meaning that for the edges, we are using the *Array of Struct* approach. The `Edge` struct is given in listing 6.6.

Listing 6.5: Cell struct

```
struct Cell
    center::Array{Float64,1}
    size::Array{Float64,1}
    edges::Array{Int64,1}
    neighbors::Array{Cell,1}
    facetypes::Array{FaceType,1}
    dataidx::Int64
end
```

Listing 6.6: Edge struct

```
mutable struct Edge
    normal::Tuple{Int64, Int64}
    FluxBuffer::Array{Float64,3}
    GBuffer::Array{Float64,4}
    dofsBuffer::Array{Float64,3}
    neighborsidx::Array{Int64,1}
    dataidx::Int64
    Inner::Boolean
    Outer::Boolean
    boundary::Array{Int64,1}
    length::Float64
end
```

The `boundary` array stores information on the boundary type of an edge on either of its two sides. Internal edges hold `[0,0]` for the `boundary` array and the `Inner` is set to one

to identify the edge as an inner edge quicker. If an edge is part of the domain boundary, then the corresponding value of `boundary` is set to `1`, `2` or `3` for inflow boundary, outflow boundary or a boundary condition respectively. The value for `Outer` is set to one to identify an edge as an outer edge quickly.

6.3. Viscous Flux Volume Term

The evaluations of the numerical hyperbolic flux and the corresponding volume term in the right-hand side (v^h and r in equation 4.16) were already implemented. The solver needs to be extended, such that the viscous flux from equation 4.2 is implemented. To achieve this, we need to implement the remaining four terms in the right-hand side of equation 4.16. In this section we will discuss the implementation of the volume term of the viscous flux (v^v in equation 4.16). Since the viscous flux $\boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u})$ is also a function of the derivatives of the degrees of freedom, the numerical treatment for v^v is slightly different from v^h.
The viscous flux $\boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u})$ depends on the derivatives of velocity and temperature. These values can be obtained from $\boldsymbol{u}$ by using the chain rule for derivatives. This can be a time consuming operation and it would be more efficient to compute the viscous flux as $\boldsymbol{F}^v(\boldsymbol{u}, \nabla \boldsymbol{u}) = G(\boldsymbol{u}) \cdot \nabla \boldsymbol{u}$. This means that for each cell we must evaluate the derivatives of the degrees of freedom and the matrix G at each node. This is done by having a "volume kernel" iterate over all cells. This kernel is an object that holds memory for buffers that can be used to store intermediate data. The buffer for the volume kernel is given in listing 6.7

Listing 6.7: Struct for the volume kernel: BuffersVolume

```
1 struct BuffersVolume
2     derivativeBuffer::Array{Float64,3}
3     GBuffer::Array{Float64,3}
4     scaled_fluxcoeff::Array{Float64,2}
5     X::Diagonal{Float64, Array{Float64, 1}}
6 end
```

To obtain the derivatives of the degrees of freedom $\nabla \boldsymbol{u}$, we multiply the degrees of freedom $\boldsymbol{u}$ with the derivative matrices that are predefined on the reference cell. The result of this matrix-vector multiplication is multiplied with the inverse jacobian matrix of the mapping operator $\mathcal{M}$ to get the derivatives on the nodes of the physical cell. This result is then stored in `derivativeBuffer` of the volume kernel. The matrix G is evaluated with the function `evaluate_G(G, dofs)` and as arguments to the function, a pointer to `GBuffer` of the volume kernel and the degrees of freedom of the respective cell are given. The function then computes the entries of the G matrix and stores it in `GBuffer`.
Finally, the kernel evaluates the viscous flux by applying a dot product to the matrix G

and the derivatives of the degrees of freedom. This operation can be seen as a generalized matrix vector multiplication `gemv`. Important to point out is that the iteration of the kernel over all cells can be done in parallel, since there are no couplings between the cells here.

6.4. Edge Terms

The edge terms have been discretized in section 4.3 and are defined by equations 4.19, 4.24 and 4.30 for the γ-edge. As the edge terms are defined for one of the four edges of a cell, it is clear that the value of the edge term depends on the orientation and position of the edge with respect to its neighbouring cells. So, for each cell, the edge terms of four edges need to be computed. If we iterate over all cells, however, this will lead to a set of duplicate calculations for a single edge.
We implement, therefore, an edge kernel that iterates over all edges. This kernel is listed in listing 6.8. It evaluates the degrees of freedom $\boldsymbol{u}^1$ and $\boldsymbol{u}^2$ projected onto the edge and stores the result in `dofsBuffer`. The superscripts 1 and 2 indicate here the side of an edge according to fig. 4.1. In a similar fashion the gradients of the degrees of freedom are projected on both sides of an edge by multiplying the degrees of freedom on both sides with the corresponding Δ matrix (see section 4.3) and inverse Jacobian, and stored in `dofsGradBuffer`. With the projections in `textBuffer` the G matrix can be evaluated for the nodes on the edge. The result is stored in `GBuffer`.

Listing 6.8: Struct for the edge kernel: BuffersEdge

```
1 struct BuffersEdge
2     dofsBuffer::Array{Float64,3}
3     dofsGradBuffer::Array{Float64,4}
4     GBuffer::Array{Float64,4}
5 end
```

Afterwards, the kernel uses all the stored data to compute the viscous flux on both sides of the edge, and stores the result in the `FluxBuffer` of the corresponding `edge` object (see listing 6.6). The projected degrees of freedom are also stored inside the `edge` object in `dofsBuffer`. Additionally, the dot-product of the G matrix with the unit normal vectore of the edge is stored in `GBuffer` of the `edge` object. Storing all this data in buffers that are uniquely allocated for each edge costs a lot of memory, but it is necessary in this approach, because in the next step we will iterate over the cells again to compute all edge terms on each cell. This iteration over all edges could be done in parallel.
In the subsequent iteration over all cells the edge terms are evaluated by multiplying the projected viscous flux, projected G matrix and projected degrees of freedom with the right Λ matrix for the flux and penalty edge term. For the symmetry edge term, a multiplication with a Δ matrix is necessary. After a multiplication with the edge length, due to the map-

ping to the physical cell, the result can be added to the update value `du`, which is used to update the degrees of freedom for the homogeneous problem.

6.5. Source Terms

The implementation of the source terms is given in listing 6.9. During a step, the degrees of freedom are updates according to the gomogeneous part of 5.10 for timestep `dt`. This evaluation is then used for the update of the degrees of freedom according to the heterogenous part of equation 5.10 for a timestep `dt`. The function `evaluate_source` computes `du` for the heterogenous part, following the narative in section 5.3.

Listing 6.9: Source term evaluation

```
function evaluate_sources(eq, scenario, du, dofs, grid)
    for i in eachindex(grid.cells)
        @views cell = grid.cells[i]
        @views evaluate_source(eq, scenario, dofs, du, cell)
        elem_massmatrix = volume(cell) * reference_massmatrix
        inv_massmatrix = inv(elem_massmatrix)
        @views du = inv_massmatrix * @views du
    end
end
```

7. Model Verification

To understand and prove the accuracy of a computational model, there are some procedures that must be taken. Two of these procedures are: Verification and validation. The verification of a computational model is usually done by comparing the computational results to previous, possibly more accurate, computational results. In case an analytical solution exists, one can verify a model by comparing it to the analytical solution. This can be useful to prove that parts of the code, that solve standardized problems, are correct.
Validating a computational model means that one compares the computational solution to real solutions that are obtained experimentally. Taking into account any measurement inaccuracies, this procedure can be used to prove that code is not only properly implemented, but also the underlying mathematical model is physically correct. During the validation procedure, it may turn out that certain physical phenomena in the flow were misunderstood or even neglected.
During this thesis, the verification step was conducted for the discontinuous Galerkin implementation of the Navier-Stokes equations by solving the Taylor-Green case and a lid-driven cavity problem. The Taylor-Green case verification is described in section 7.1, and is meant to demonstrate that the basic Navier-Stokes system is implemented correctly. In section 7.2, the cavity case will be discussed. The cavity case will give some insight into the correctness of implementation of the boundary conditions.

7.1. Taylor-Green Vortex

The Taylor-Green vortex is usually used for the verification of time dependent incompressible Navier-Stokes simulations, to demonstrate that the time-stepping scheme and the spatial discretization are correct. Since we are implementing the time-dependent compressible Navier-Stokes equations, there will be some discrepancies between the model and the analytical solution, because our model has a variable density. Nonetheless, the Taylor-Green case is a good verification example for a compressible flow with negligible density gradients. As long as the flow stays in the incompressibe flow regions, so with Mach numbers below 0.3 for air, the compressible Navier-Stokes equations give nearly the same solutions as the incompressible Navier-Stokes equations.

Table 7.1.: Maximum L^2-error of the velocities u and v compared to the analytical solution at time $t = 1$ for viscosity $\mu = 0.1$. The explicit Euler scheme with CFL $= 0.5$ was used for timestepping. The errors are given for a domain containing $K \times K$ cells with polynomial basis order p. The penalty coefficient δ is set to zero because no instabilities were observed.

p \ K	5	10	20
3	0.2377916	0.1766741	0.1518713
4	0.2140415	0.1672278	0.1479784
5	0.1875251	0.1599287	0.1454245

The Taylor-Green case is, following Taylor & Green [31], given by

$$\begin{aligned} \rho(x,y,t) &= 1, \\ u(x,y,t) &= \cos(x)\sin(y)\, e^{-2\nu t}, \\ v(x,y,t) &= -\sin(x)\cos(y)\, e^{-2\nu t}, \\ p(x,y,t) &= \frac{1}{4}(\cos(2x)+\cos(2y))\, e^{-4\nu t} + C. \end{aligned} \tag{7.1}$$

Here we consider periodic boundary conditions and the initial condition of the Taylor-Green case is given for $t = 0$. Equation 7.1 gives the analytic solution of the Taylor-Green case at time t. The kinematic viscosity ν is given by $\nu = \mu/\rho$. The constant C governs the speed of sound and is in our case $C = 100/\gamma$, giving a Mach numer of 0.1. The problem is solved on the domain $[0, 2\pi]^2$.

Computational Approximation

The verification of the Taylor-Green problem is given in fig. 7.1 for a fifth order approximation for a domain of 10×10 elements. The size of the domain is $2\pi \times 2\pi$ and fig. 7.1 describes the approximation of the solution of u along the line $(0.5, y)$ and the approximation of the solution of v along the line $(x, 0.5)$ at time $t = 1$. The penalty coefficient was set to zero, as there were no instabilities observed for high-order spatial approximations of the Taylor-Green flow. An introduction of a penalty parameter of 0.1, 1.0 or 5.0 did not have a noticable impact on the computational results.
In table 7.1 the L^2-error of the numerical approximation of the velocity vectors is shown for a set of mesh refinements and polynomial orders for the basis functions. The Taylor-Green case was run with the explicit-Euler timestepping scheme, which means that the results are first order accurate in time. Nonetheless, a convergence for the approximation of the solution can be observed for finer meshes and high-order basis functions.

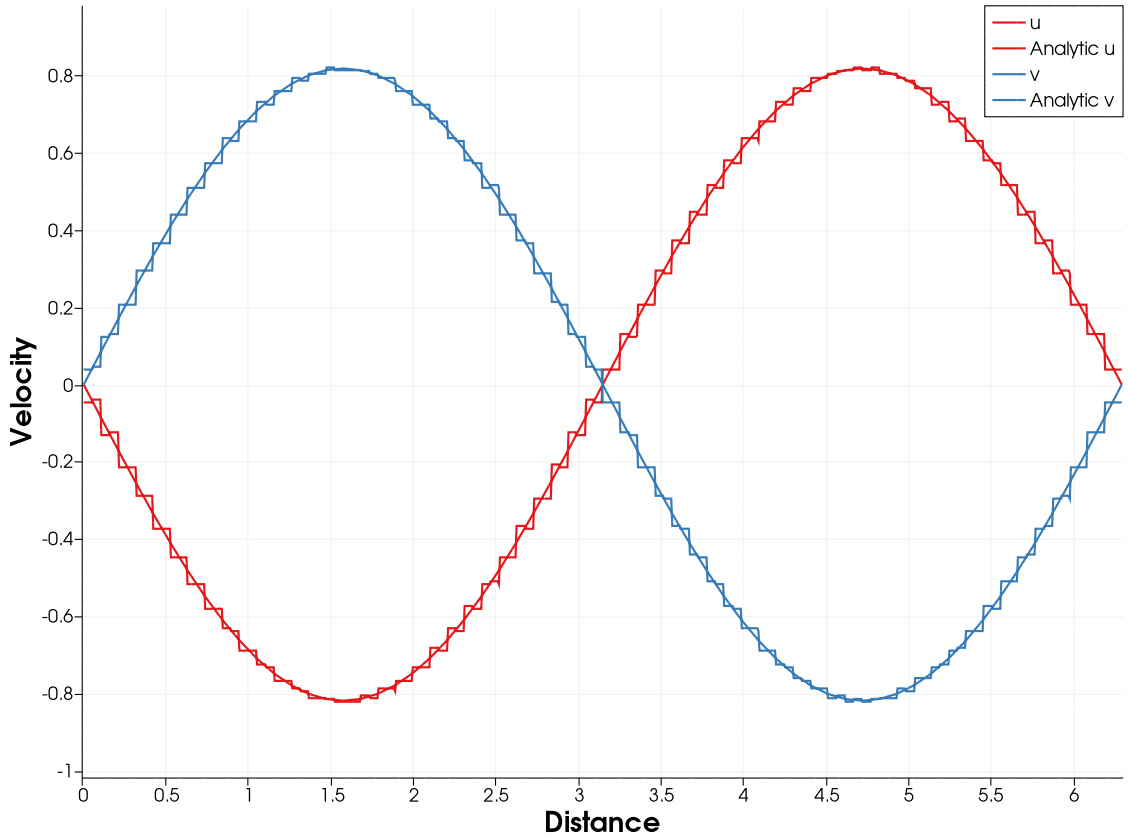

Figure 7.1.: A section plot of the velocity profile of v at the line $x = 0.5$ and of u at the line $y = 0.5$ at time $t = 1$. The analytical solution is given by the continuous lines. The computed solution is for 5^{th} order, $K = 10$ with $\mu = 0.1$. The penalty coefficient was set to zero.

7.2. Cavity Flow

The lid-driven cavity case is used as a verification method for the implementation of the boundary conditions. The lid-driven cavity is a flow problem on the domain $[0, 1]^2$, where the left, right and bottom boundaries of the domain are adiabatic walls and the top boundary has a flow boundary condition of flow moving to the right with a speed of 1. This problem simulates a case of a 2D cavity with a lid on top that moves to the right with a speed of 1, hence the name "lid-driven cavity". A depiction of the case is given in fig. 7.2.

Table 7.2.: Reference data for the cavity flow problem as given by Ghia et al. [15] along the line $x = 0.5$ for u (left two columns) and $y = 0.5$ for v (right two columns).

y	u	x	v
1.0	1.0	1.0	0.0
0.9766	0.84123	0.9688	-0.05906
0.9688	0.78871	0.9609	-0.07391
0.9609	0.73722	0.9531	-0.08864
0.9531	0.68717	0.9453	-0.10313
0.8516	0.23151	0.9063	-0.16914
0.7344	0.00332	0.8594	-0.22445
0.6172	-0.13641	0.8047	-0.24533
0.5000	-0.20581	0.5000	0.05454
0.4531	-0.21090	0.2344	0.17527
0.2813	-0.15662	0.2266	0.17507
0.1719	-0.10150	0.1563	0.16077
0.1016	-0.06434	0.0938	0.12317
0.0703	-0.04775	0.0781	0.10890
0.0625	-0.04192	0.0703	0.10091
0.0547	-0.03717	0.0625	0.09233
0.0	0.0	0.0	0.0

Computational Approximation

No analytical solution exists for the problem, but it is a well-known case for testing computational fluid models. Verification was done by using data from Ghia et al. [15] along the lines $x = 0.5$ and $y = 0.5$ at $t = 1$ with $\mu = 0.1$. The verification data is given in table 7.2, where the left two columns give the data for u along the line $x = 0.5$ and the two columns on the right give the data for v along the line $y = 0.5$. The computational approximation and the reference data are given in fig. 7.3, where the red line represents the approximation of u and the blue line the approximation of v. The basis functions are of 4^{th} order at time $t = 1$ and the domain is divided into 16×16 elements. As for the Taylor-Green case, the timestepping scheme is the explicit Euler scheme and the viscosity was set to $\mu = 0.1$.
The penalty coefficient was set to zero for the approximation given in fig. 7.3 because there was no need to increase the stability of the solver. As for the Taylor-Green case, the introduction of a non-zero penalty parameter did not have a noticable effect on the computational approximation for the lid-driven cavity flow.

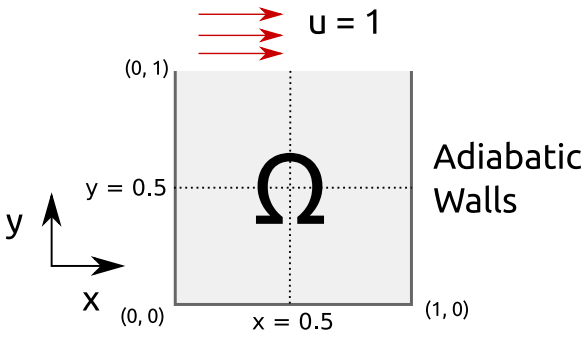

Figure 7.2.: The lid-driven cavity flow problem. The domain Ω with size $[0, 1]^2$ is illustrated by the gray field, with black edges on the bottom,left and right side as the adiabatic walls. On top, there is a flow $\boldsymbol{v} = (1, 0)^T$, which simulates the moving lid.

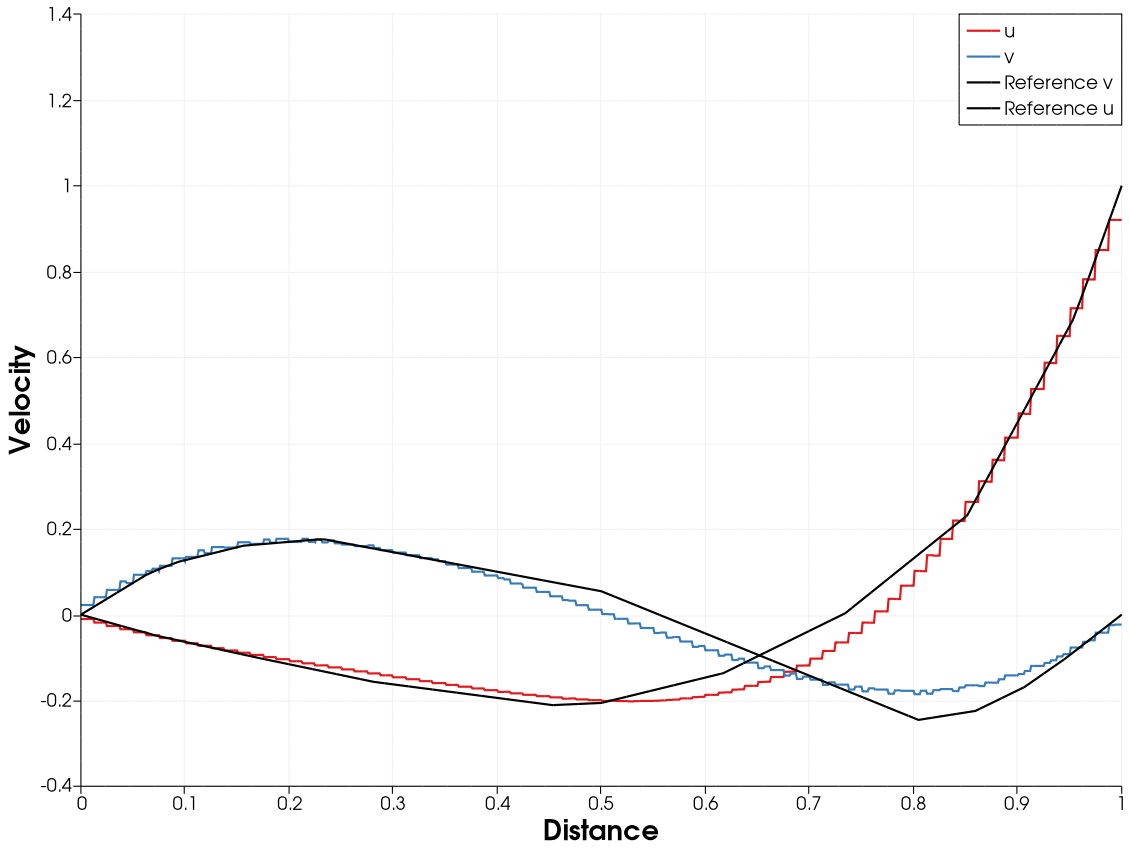

Figure 7.3.: A section plot of the velocity profile at the lines $x = 0.5$ and $y = 0.5$. The computed velocity in x-direction is indicated by u and the velocity in y-direction is indicated by v. The black lines denote the reference data from table 7.2 as given by Ghia et al. [15]. The computed solution is for 4^{th} order basis functions, K = 16 and viscosity $\mu = 0.1$. The penalty coefficient was set to zero.

Part IV.

Results and Conclusion

8. Results

The verification procedure in chapter 7 demonstrates that the compressible NSE are implemented such that the solver produces acceptable results for the Taylor-Green flow and a lid-driven cavity flow. The verification of the implementation of the extended system with porosity and source terms was not carried out. Nonetheless, it is interesting to have a look at the result for a PEMFC flow with a free flow domain and a porosity domain.
In section 8.1, a PEMFC flow case is described and the results produced by the solver are presented. For a PEMFC solver that is high-order accurate in time, work is still needed to be done and some recommendations for future work are given in section 8.2

8.1. PEMFC Flow

For the PEMFC simulation that is described below, the extended system of equations as described in chapter 5 is used. The overpotential that regulates the simulation of the electrochemical reactions is set to zero, as a nonzero overpotential would lead to instabilities. This effectively means that the energy equation and the species equation are homogeneous. The PEMFC simulation is carried out for the anode flow channel and GDL.

Anode Flow Case

The case that is simulated for the anode flow channel and GDL of a PEMFC is illustrated in fig. 8.1. The problem is solved on a domain of shape $[0, 4] \times [0, 1]$, where light gray domain in fig. 8.1 indicates the free flow region of the domain and the dark gray region represents the porous region of the domain. The height of the GDL is set to 0.4 and the porosity to $\varepsilon = 0.4$. The penalty coefficient was set according to equation 4.34 with the constant $C = 10$, which is the value that was empirically found to be stable for the compressible NSE by Hartmann & Houston [16].
The initial condition for the flow velocity is $\boldsymbol{v} = (0, 0)^T$ and we determine a pressure of $10/\gamma$, which would set the Mach number in the computational domain to 0.01. Additionally, the concentration of H_2 is set to 0.9. For the walls, adiabatic, no-slip boundary conditions are set and for the inflow at the left boundary, a flow velocity of $\boldsymbol{v} = (1, 0)^T$ is set.

Results

The solver is unstable for the problem described above. In fig. 8.2 and fig. 8.3, the results for the anode flow case are illustrated at time $t = 0.3$ and $t = 0.4$ respectively. The white lines are the streamlines of the flow and they can be seen to interact with the porous region, indicated by a dotted box. The density is illustrated by the colour scheme and it is visible that as the front of the flow reaches the outflow boundary, the density increases drastically. This is not the expected behaviour and the density keeps increasing until the flow becomes unstable. This increase of density at the outflow boundary suggests that the outflow boundary conditions might not be implemented correctly and this cause for instability is not solved for a higher penalty coefficient. This problem did not arise for the lid-driven cavity case, because no outflow boundaries were set.

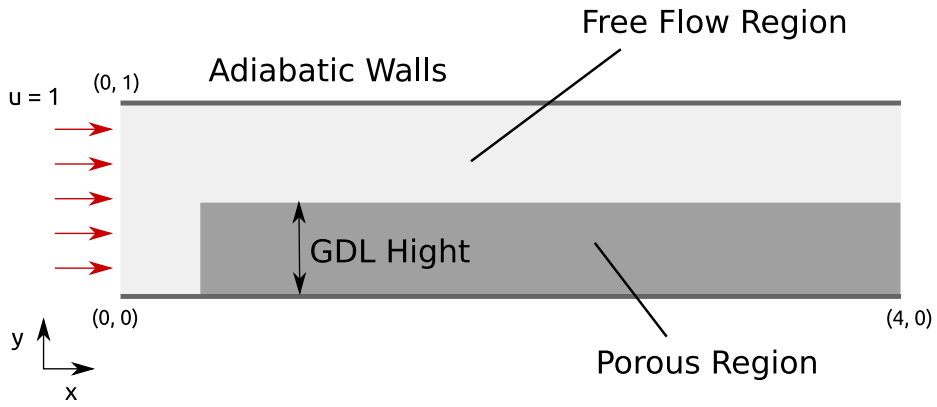

Figure 8.1.: The flow case for the anode flow channel and GDL of a PEMFC. The domain Ω with size $[0, 4] \times [0, 1]$ is illustrated by the light gray field, with black edges at the bottom and top sides to indicate the adiabatic walls. On the left, there is an inflow $\boldsymbol{v} = (1, 0)^T$. On the right side we have outflow boundary conditions. The dark gray area represents the porous region of the domain. For this area a porosity $\varepsilon = 0.4$ is set and the flow viscosity is set to $\mu = 0.000181$.

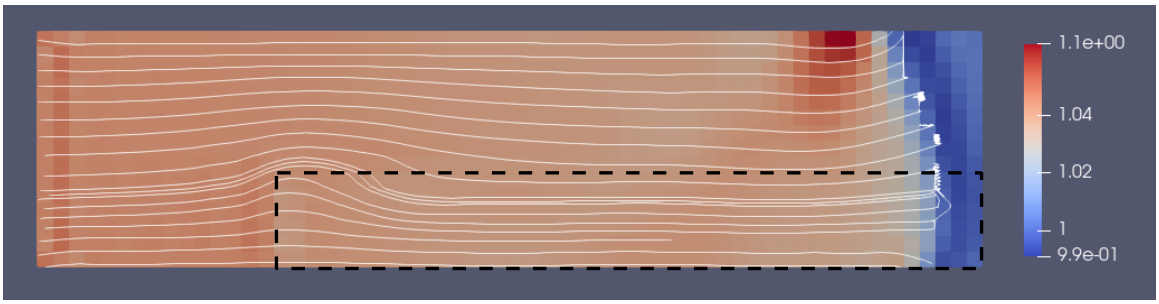

Figure 8.2.: The result of the anode flow case at time $t = 0.3$. The density of the domain is illustrated by the colour scheme and the white lines represent the streamlines of the flow. An interaction between the flow and the porous domain, indicated by the dottet box, can be observed.

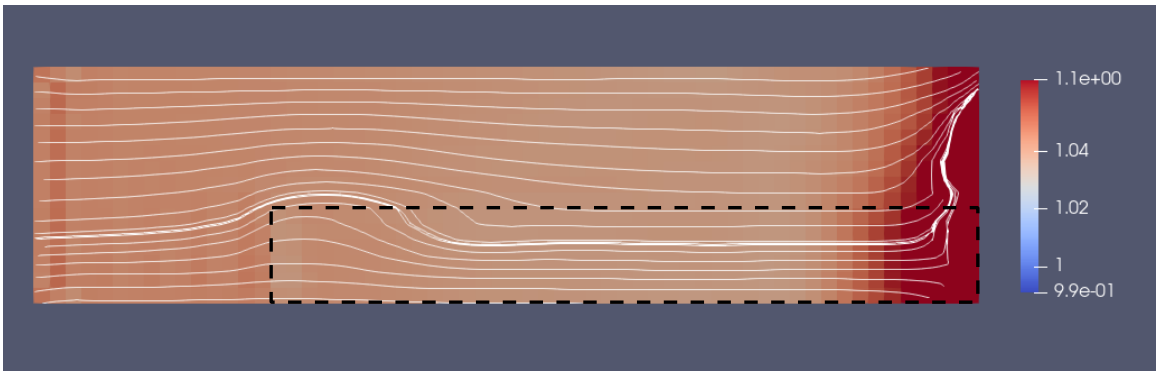

Figure 8.3.: The result of the anode flow case at time $t = 0.4$. The density of the domain is illustrated by the colour scheme and can be observed to increase as the flow has reached the outflow boundary. This increase in density will eventually lead to instabilities.

8.2. Recommendations for Future Work

The intention of this thesis was to present a discontinuous Galerkin approach for the simulation of the flow channel and porous layer in a PEMFC. In the process of getting a simulation, certain assumptions were made (see section 5.1) to simplify the simulation. For a more accurate representation of a PEMFC, a recommendation for future work would be to include the mathematical terms that were left out in this simplified model.

The following physical effects will need to be added:

- The relative motion of species due to the condensation or evaporation is to be added

to the species equation.

- The effective diffusivity is to be considered as a function of the volumetric fraction of water in liquid or vapor form.
- The effect of electro-osmotic drag should be included.
- The advection correction factor must be implemented.

The effects listed above are necessary for a more physically accurate representation of the flow in a PEMFC, following the description by Sun [30]. A good future project to extend this code, would be to introduce a coupling between this solver and a solver for the electronic potentials to get a more accurate simulation, using an external coupling library, such as preCICE. Alternatively, these phenomena could be directly implemented in this solver.

The verification procedure that is given in chapter 7 may suffice for the verification of the compressible NSE, but not for the implementation of the species equation, source terms and the porosity in the domain. The complexity of the problem calls for a verification method other than using analytical solutions to verify approximations of PDEs. The *method of manufactured solutions* can be used to create exact solutions for complex problems, to verify the implementation of the solver thereof. The method of manufactured solutions is explained in depth by Oberkampf & Roy [25]. The main concept of this idea is that one creates a solution for a "slightly modified equation". By applying the governing equations to the chosen solution, an additional analytic source term is generated. An approximation of the solution can then be computed for a problem with this additional source term. If the code is implemented correctly, the approximation will be close to the initially chosen solution.
For a high-order simulation of a PEMFC, high-order basis functions and timestepping methods need to be chosen. The polynomial order of the Lagrange basis functions can, theoretically, be chosen arbitrarily high and the timestepping method can be chosen to be any method with a high-order accuracy in time. The splitting method, however, limits the order of accuracy of the simulation. For a high-order method, the given Godunov splitting method must be replaced by a high-order splitting method. Possible alternative methods for higher-order splitting are given by Koch et al. [21].

9. Conclusion

The results that have been presented indicate that a discontinuous Galerkin approach can be used for the simulation of the flow channels and gas diffusion layers of PEMFCs. The advantages of the discontinuous Galerking approach are that it is a proper model for approximations for discontinuous problems, while maintaining the ability to scale to high-order simulations by implementing basis functions of high-order. The presented methods can be used for any high-order partial differential equation, provided a stable and consistent numerical flux is chosen for the fluxes with high-order derivative terms.
More work on the solver is necessary to guarantee reliable computational approximations, as the solver still shows some unwanted behaviour, such as instabilities, for certain configurations. Apart from that, for a high-order PEMFC solver, high-order approximation methods need to be implemented for the splitting method.
A reliable high-order discontinuous Galerkin solver for fuel cells in general is desirable for a better understanding of the inner workings of fuel cells, such that they can be designed for specialized applications, such as in vehicles or aircraft. The fuel cell system is not a modular system and must be specially designed for its application to ensure a higher efficiency of the entire system. A harmonious interaction between the fuel cell system and a complete vehicle or aircraft system is desired and can be attained through proper computational modeling. Investigation into reliable high-order simulations of fuel cells are therefore desired.
Throughout history, mankind has evolved the energy generation by burning various fuels, from wood, to coal, to gas. With each step, less carbon is present in the fuel and the logical next step would be to use a fuel that does not contain any carbon molecules at all: Hydrogen. The beauty of fuel cell systems lies in the fact that they can efficiently generate electrical energy from hydrogen and the only product of this process is water. It could be a solution to stop the emission of harmful greenhouse gases and, therefore, slow global warming.

Appendix

A. Matrix Definitions

Below the matrices $G_{ij}(\boldsymbol{u})$ are defined, as used in the compressible Navier Stokes Equations as given by

$$G_{11} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{4}{3}u & -\frac{4}{3} & 0 & 0 \\ v & 0 & -1 & 0 \\ (\frac{4}{3}u^2 + v^2 + \frac{\gamma}{Pr}(E - \boldsymbol{v}^2)) & -(\frac{4}{3} - \frac{\gamma}{Pr})u & -(1 - \frac{\gamma}{Pr})v & -\frac{\gamma}{Pr} \end{pmatrix} \tag{A.1}$$

$$G_{12} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -\frac{2}{3}v & 0 & \frac{2}{3} & 0 \\ u & -1 & 0 & 0 \\ \frac{1}{3}uv & -v & \frac{2}{3}u & 0 \end{pmatrix} \tag{A.2}$$

$$G_{21} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ v & 0 & -1 & 0 \\ -\frac{2}{3}u & \frac{2}{3} & 0 & 0 \\ \frac{1}{3}uv & \frac{2}{3}v & -u & 0 \end{pmatrix} \tag{A.3}$$

$$G_{22} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ u & -1 & 0 & 0 \\ \frac{4}{3}v & 0 & -\frac{4}{3} & 0 \\ (u^2 + \frac{4}{3}v^2 + \frac{\gamma}{Pr}(E - \boldsymbol{v}^2)) & -(1 - \frac{\gamma}{Pr})v & -(\frac{4}{3} - \frac{\gamma}{Pr})v & -\frac{\gamma}{Pr} \end{pmatrix} \tag{A.4}$$

The homogeneity matrices for the extended system $\tilde{G}_{ij}(\boldsymbol{u})$ are defined as

$$\tilde{G}_{11} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{4}{3}u & -\frac{4}{3} & 0 & 0 & 0 \\ v & 0 & -1 & 0 & 0 \\ (\frac{4}{3}u^2 + v^2 + \frac{\gamma}{Pr}(E - \boldsymbol{v}^2)) & -(\frac{4}{3} - \frac{\gamma}{Pr})u & -(1 - \frac{\gamma}{Pr})v & -\frac{\gamma}{Pr} & 0 \\ \frac{C_k D_k}{\mu} & 0 & 0 & 0 & \frac{D_k}{\mu} \end{pmatrix} \tag{A.5}$$

$$\tilde{G}_{12} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -\frac{2}{3}v & 0 & \frac{2}{3} & 0 & 0 \\ u & -1 & 0 & 0 & 0 \\ \frac{1}{3}uv & -v & \frac{2}{3}u & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.6}$$

$$\tilde{G}_{21} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ v & 0 & -1 & 0 & 0 \\ -\frac{2}{3}u & \frac{2}{3} & 0 & 0 & 0 \\ \frac{1}{3}uv & \frac{2}{3}v & -u & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.7}$$

$$\tilde{G}_{22} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ u & -1 & 0 & 0 & 0 \\ \frac{4}{3}v & 0 & -\frac{4}{3} & 0 & 0 \\ (u^2 + \frac{4}{3}v^2 + \frac{\gamma}{Pr}(E - \boldsymbol{v}^2)) & -(1 - \frac{\gamma}{Pr})v & -(\frac{4}{3} - \frac{\gamma}{Pr})v & -\frac{\gamma}{Pr} & 0 \\ \frac{C_k D_k}{\mu} & 0 & 0 & 0 & \frac{D_k}{\mu} \end{pmatrix} \tag{A.8}$$

B. Cell Transformation Mapping

Mapping from the Reference Cell

The referece cell is in described as a unit cell $\hat{C} = [0{\times}1]^2$ in 2D. We use the coordinates (x, y) for the physical domain, and $(\hat{x}, \hat{y})$ for the coordinates in the reference cell, also called the *Computational Domain*. The mapping function $\mathcal{M}$ maps the coordinates from the reference cell to the physical cell

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathcal{M}(\xi, \eta) = \text{cell-center}(C) + \begin{pmatrix} \Delta x & 0 \\ 0 & \Delta y \end{pmatrix} \begin{pmatrix} \hat{x} - 0.5 \\ \hat{y} - 0.5 \end{pmatrix}. \tag{B.1}$$

The problems will be derived in terms of the reference cell, which allows us to pre-calculate many values, making the computation faster. It must, however, be noted that this means that the problem equations will also need to be transformed to the reference cell case.

Transforming Derivatives and Integrals

Under mappings, the equations themselves are als transformed. Essentially, we follow the chain rule for this procedure. Formally, we find the effect of a mapping with the following definitions

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{\partial u}{\partial \hat{x}} \frac{\partial \hat{x}}{\partial x} + \frac{\partial u}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial x} \\ \frac{\partial u}{\partial y} &= \frac{\partial u}{\partial \hat{x}} \frac{\partial \hat{x}}{\partial y} + \frac{\partial u}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial y} \end{aligned} \tag{B.2}$$

We will also need to define the transformation of derivatives. Given in matrix-vector form it is

$$\begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\partial \hat{x}}{\partial x} & \frac{\partial \hat{y}}{\partial x} \\ \frac{\partial \hat{x}}{\partial y} & \frac{\partial \hat{y}}{\partial y} \end{pmatrix} \begin{pmatrix} \frac{\partial u}{\partial \hat{x}} \\ \frac{\partial u}{\partial \hat{y}} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{x}} \\ \frac{\partial x}{\partial \hat{y}} & \frac{\partial y}{\partial \hat{y}} \end{pmatrix}^{-1}}_{J} \begin{pmatrix} \frac{\partial u}{\partial \hat{x}} \\ \frac{\partial u}{\partial \hat{y}} \end{pmatrix} \tag{B.3}$$

The matrix that gives the transformation of the derivatives, is obtained indirectly from the inverse Jacobian of the transformation operator $\mathcal{M}$.
Finally, for the transformation of integrals, the transformation of differentials must also be

specified.

$$dx = \frac{\partial x}{\partial \hat{x}}\, d\hat{x}, \qquad dy = \frac{\partial y}{\partial \hat{y}}\, d\hat{y} \tag{B.4}$$

The transformation of an integral over a reference cell then becomes:

$$\int_C f(x,y)\; dxdy = \int_{\hat{C}} f(\mathcal{M}(\xi,\eta))\; \underbrace{\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta}}_{\det(J)}\; d\xi d\eta \tag{B.5}$$

If the computation includes derivatives, the Jacobian must be taken into account as well in the computation of the derivatives, giving

$$\begin{aligned} \int_C \nabla f(x,y)\; dxdy &= \int_C \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{pmatrix}\; dxdy \\ &= \int_{\hat{C}} \left(J^{-1} \begin{pmatrix} \frac{\partial f(\mathcal{M}(\xi,\eta))}{\partial \xi} \\ \frac{\partial f(\mathcal{M}(\xi,\eta))}{\partial \eta} \end{pmatrix} \right)^T \det(J)\; d\xi d\eta \end{aligned} \tag{B.6}$$

List of Abbreviations

AFC	Alkaline fuel cell
BR2	Second approach of Bassi and Rebay
CFD	Computational fluid dynamics
CL	Catalyst layer
DG	Discontinuous Galerkin
DGM	Discontinuous Galerkin method
GDL	Gas diffusion layer
HOR	Hydrogen oxidation reaction
ID	Identity
MCFC	Molten carbonate fuel cell
MEA	Membrane electrode assembly
ODE	Ordinary differential equation
ORR	Oxygen reduction reaction
PAFC	Phosphoric acid fuel cell
PDE	Partial differential equation
PEMFC	Proton exchange membrane fuel cell
RK3, RK4	3^{rd} / 4^{th} order Runge-Kutta scheme
SIPG	Symmetric interior penalty discontinuous galerkin (method)
SOFC	Solid-oxide fuel cell

List of Symbols

Latin Symbols

C	A cell element	k	Thermal conductivity coefficient
C^k	Molar concentration	M	Mass matrix
c	Speed of sound	M^k	Molar weight
D^k	Diffusion coefficient	$\mathcal{M}$	Mapping operator
d	Dimensionality of the domain	n	Amount of transferred electrons
E	Specific total energy	$\boldsymbol{n}$	Normal vector
E_0	Reversible cell voltage	P	Projection matrix
$\mathcal{E}_h$	Set of all edge elements	Pr	Prandtl's number
e	An edge element	p	Polynomial order
e	Edge flux term	p	Pressure
e	Specific internal energy	R	Universal gas constant
F	Faraday's constant	$\boldsymbol{S}$	Source term vector
$\boldsymbol{F}$	A flux	T	Temperature
f_l^k	Liquid mass fraction	$\mathcal{T}_h$	Set of all cell elements
G	Homogeneity tensor	t	Time
g	Gibbs energy	U_0	Thermodynamic equilibrium potential
H	Total enthalpy	u	Velocity in x-direction
h	Element size	$\boldsymbol{u}$	Degrees of freedom
J	Jacobian matrix	v	Velocity in y-direction
j_a	Volumetric transfer current density	v	Volume flux term
$\boldsymbol{j}_l$	Capillary-diffusional flux	$\boldsymbol{v}$	Test function from V_h
K	Permeability tensor	$\boldsymbol{v}$	Velocity vector

Greek Symbols

α	Left edge of a cell
β	Top edge of a cell
Γ	The set of all internal edges
γ	Heat capacity ratio
γ	Right edge of a cell
γ_c	Advection correction factor
γ_T	Advection heat-transfer correction factor
δ	Bottom edge of a cell
δ	Penalty coefficient
ε	Porosity of media
η	Overpotential
κ	Protonic conductivity
Λ	Jump matrix
λ	Eigenvalue
μ	Viscosity
ν	Kinematic viscosity
ρ	Density
σ	Auxiliary variable
σ	Electronic conductivity
σ	Viscous stress tensor
τ	Test function from Σ_h
ϕ	1D basis function
φ	2D basis function
$\boldsymbol{\psi}$	Heterogenous part of equation
Ω	The domain
ω	Quadrature weight

References

[1] Abe, J., Popoola, A., Ajenifuja, E., & Popoola, O. (2019). Hydrogen energy, economy and storage: Review and recommendation. *International Journal of Hydrogen Energy*, *44*(29), 15072-15086. Doi: https://doi.org/10.1016/j.ijhydene.2019.04.068

[2] Alhawwary, M., & Wang, Z. (2020, Jan). A combined-mode fourier analysis of dg methods for linear parabolic problems. *SIAM Journal on Scientific Computing*, *42*(6), A3825–A3858. Doi: 10.1137/20m1316962

[3] Arnold, D., & Douglas, N. (1982, 08). An interior penalty finite element method with discontinuous elements. *SIAM Journal on Numerical Analysis*, *19*. Doi: 10.1137/0719052

[4] Arnold, D. N., Brezzi, F., Cockburn, B., & Marini, L. D. (2002). Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, *39*(5), 1749-1779. Doi: 10.1137/S0036142901384162

[5] Barzkar, A., & Ghassemi, M. (2020). Electric power systems in more and all electric aircraft: A review. *IEEE Access*, *8*, 169314-169332. Doi: 10.1109/ACCESS.2020.3024168

[6] Bassi, F., & Rebay, S. (1997). A high-order accurate discontinuous finite element method for the numerical solution of the compressible navier–stokes equations. *Journal of Computational Physics*, *131*(2), 267-279. Doi: https://doi.org/10.1006/jcph.1996.5572

[7] Brezzi, F., Manzini, G., Marini, D., Pietra, P., & Russo, A. (2000). Discontinuous galerkin approximations for elliptic problems. *Numerical Methods for Partial Differential Equations*, *16*(4), 365-378. Doi: https://doi.org/10.1002/1098-2426(200007)16:4⟨365::AID-NUM2⟩3.0.CO;2-Y

[8] Cockburn, B., & Shu, C.-W. (1998). The local discontinuous galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, *35*(6), 2440-2463. Doi: 10.1137/S0036142997316712

[9] Costamagna, P., & Srinivasan, S. (2001a). Quantum jumps in the pemfc science and technology from the 1960s to the year 2000: Part i. fundamental scientific aspects. *Journal of Power Sources*, *102*(1), 242-252. Doi: https://doi.org/10.1016/S0378-7753(01)00807-2

[10] Costamagna, P., & Srinivasan, S. (2001b). Quantum jumps in the pemfc science and technology from the 1960s to the year 2000: Part ii. engineering, technology development and application aspects. *Journal of Power Sources*, *102*(1), 253-269. Doi: https://doi.org/10.1016/S0378-7753(01)00808-4

[11] Dawson, C., Sun, S., & Wheeler, M. F. (2004). Compatible algorithms for coupled flow and transport. *Computer Methods in Applied Mechanics and Engineering*, *193*(23), 2565-2580. Doi: https://doi.org/10.1016/j.cma.2003.12.059

[12] Demuren, A., & Edwards, R. L. (2020). Modeling proton exchange membrane fuel cells—a review. In *50 years of cfd in engineering sciences: A commemorative volume in memory of d. brian spalding* (pp. 513–547). Singapore: Springer Singapore. Retrieved from

[13] Douglas, J., & Dupont, T. (1976). Interior penalty procedures for elliptic and parabolic galerkin methods. In *Computing methods in applied sciences* (pp. 207–216). Springer.

[14] Dumbser, M., Balsara, D. S., Toro, E. F., & Munz, C.-D. (2008). A unified framework for the construction of one-step finite volume and discontinuous galerkin schemes on unstructured meshes. *Journal of Computational Physics*, *227*(18), 8209-8253. Doi: https://doi.org/10.1016/j.jcp.2008.05.025

[15] Ghia, U., Ghia, K., & Shin, C. (1982). High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, *48*(3), 387-411. Doi: https://doi.org/10.1016/0021-9991(82)90058-4

[16] Hartmann, R., & Houston, P. (2006, 01). Symmetric interior penalty dg methods for the compressible navier–stokes equations i: Method formulation. *International Journal of Numerical Analysis Modeling*, *3*, 1-20.

[17] Hartmann, R., & Leicht, T. (2014). *Higher order and adaptive dg methods for compressible flows* (Vol. 2014) (No. 3). Von Karman Institute for Fluid Dynamics. Retrieved from https://elib.dlr.de/90967/

[18] Hesthaven, J., & Warburton, T. (2007). *Nodal discontinuous galerkin methods: Algorithms, analysis, and applications*. Springer New York. Retrieved from https://books.google.de/books?id=APQkDOmwyksC

[19] Hesthaven, J. S. (2017). *Numerical methods for conservation laws: From analysis to algorithms*. SIAM.

[20] Kladias, N., & Prasad, V. (1991). Experimental verification of darcy-brinkman-forchheimer flow model for natural convection in porous media. *Journal of Thermophysics and Heat Transfer*, *5*(4), 560-576. Doi: 10.2514/3.301

[21] Koch, O., Auzinger, W., & Thalhammer, M. (2017). *Coefficients of various splitting methods.* Retrieved from [2021-04-14]https://www.asc.tuwien.ac.at/~winfried/splitting/index.php

[22] Krenz, L., & Reinarz, A. (2020). *Lecture notes: M.sc. praktikum: Modern wave propagation - discontinuous galerkin julia.*

[23] LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems*. Cambridge University Press. Doi: 10.1017/CBO9780511791253

[24] Meng, H. (2008). A pem fuel cell model for cold-start simulations. *Journal of Power Sources*, *178*(1), 141-150. Doi: https://doi.org/10.1016/j.jpowsour.2007.12.035

[25] Oberkampf, W. L., & Roy, C. J. (2010). *Verification and validation in scientific computing*. Cambridge University Press. Doi: 10.1017/CBO9780511760396

[26] O'Hayre, R., Cha, S., Colella, W., & Prinz, F. (2016). *Fuel cell fundamentals*. Wiley. Retrieved from https://books.google.de/books?id=02JYCwAAQBAJ

[27] Reed, W. H., & Hill, T. (1973). *Triangular mesh methods for the neutron transport equation* (Tech. Rep.). Los Alamos Scientific Lab., N. Mex.(USA).

[28] Rivière, B. (2008). *Discontinuous galerkin methods for solving elliptic and parabolic equations*. Society for Industrial and Applied Mathematics. Doi: 10.1137/1.9780898717440

[29] Shahbazi, K. (2005). An explicit expression for the penalty parameter of the interior penalty method. *Journal of Computational Physics*, *205*(2), 401 - 407. Doi: https://doi.org/10.1016/j.jcp.2004.11.017

[30] Sun, P. (2012, Apr 01). Efficient numerical methods for an anisotropic, nonisothermal, two-phase transport model of proton exchange membrane fuel cell. *Acta Applicandae Mathematicae*, *118*(1), 251-279. Doi: 10.1007/s10440-012-9688-0

[31] Taylor, G., & Green, A. (1937). Mechanism of the production of small eddies from large ones. *Proc. R. Soc. Lond. A*, *158*, 499-521.

[32] Wang, C., & Cheng, P. (1996). A multiphase mixture model for multiphase, multicomponent transport in capillary porous media—i. model development. *International Journal of Heat and Mass Transfer*, *39*(17), 3607-3618. Doi: https://doi.org/10.1016/0017-9310(96)00036-1