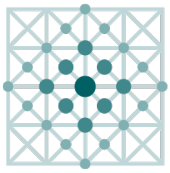


RWTH AACHEN UNIVERSITY

The present work was submitted to the
Chair of Applied and Computational Mathematics (ACoM)

Communicated by Manuel Torrilhon

A Model-Adaptive Discontinuous Galerkin Scheme for Hierarchical Moment Equations in Kinetic Gas Theory

Master Thesis
Computational Engineering Science

Aachen, April 25, 2025

Presented by	Matthias Daniel Geratz 390973 Matthias.Geratz@rwth-aachen.de
Supervisor & First examiner	Prof. Dr. Manuel Torrilhon Chair of Applied and Computational Mathematics (ACoM) RWTH Aachen University
Second examiner	Dr. Lambert Theisen Chair of Applied and Computational Mathematics (ACoM) RWTH Aachen University

Affidavit

Affidavit / Statutory Declaration

I hereby declare in lieu of an oath that I have completed the present thesis entitled *A Model-Adaptive Discontinuous Galerkin Scheme for Hierarchical Moment Equations in Kinetic Gas Theory* independently and without illegitimate assistance from third parties. I have used no other than the specified sources and aids. Furthermore I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, April 25, 2025

Matthias Daniel Geratz

Abstract

In this thesis, we present a model- and *hp*-adaptive high-order DG formulation on unstructured meshes. The numerical scheme is further equipped with a general boundary condition formulation and a higher order curved geometry representation, by means of either a Lagrange or NURBS basis for the element geometry. A mortar method is employed to handle the fluxes at non-conforming interfaces. The methods are implemented in Julia [1], yielding a prototype research code aimed at facilitating adaptive simulations of steady-state linearized hierarchical moment equations in kinetic gas theory. Representative model problems are presented and their well-posedness is discussed. Utilizing these, the framework is verified based on empirical convergence studies. High-order convergence rates are obtained, in particular for unstructured meshes and curved boundaries with general boundary conditions. It is found that a curved geometry representation is beneficial for the stability and accuracy of the numerical scheme. The capabilities of the numerical formulation and its implementation are demonstrated by simulation of a Knudsen pump microflow. The scheme is observed to be stable and converge in the presence of local mesh, DG polynomial degree and in particular model refinement.

Keywords: discontinuous Galerkin method, high-order, model-adaptivity, hp-adaptivity, curved geometry, moment equations, kinetic gas theory, microflows

Acknowledgments

I would like to thank my supervisor Prof. Manuel Torrilhon for his mentoring, advice and generous support up to and during the course this thesis. I highly appreciate the freedom and flexibility, in choice of topic and execution, that was afforded to me and the guidance I received along the way.

Further, I appreciate the fruitful discussions with colleagues at the Institute for Applied and Computational Mathematics.

Finally, my thanks go to my family and friends who have supported me throughout my studies and the writing of this thesis.

Contents

1. Introduction

The classical equations of Navier-Stokes and Fourier modelling gas dynamics become inaccurate when the gas under consideration is rarefied [2]. This is indicated by the Knudsen number $\mathrm{Kn} = \lambda/L$, relating the gas molecules mean free path between collisions λ to the characteristic length scale L of a problem [2, 3]. In this case, the number of particle collisions becomes insufficient to maintain thermal equilibrium, necessitating the use of extended models derived from kinetic gas theory [3]. Of particular interest is the so-called transition regime ($0.01 \leq \mathrm{Kn} \leq 10$), where classical models are insufficient, but the full kinetic description by means of the Boltzmann equation is too computationally expensive [3, 4]. A promising approach to close this gap in modelling capabilities are so-called moment equations, which are relaxation systems resulting from a Hermite-discretization of the Boltzmann equation in velocity space [5]. Initially introduced by Grad in 1949 [6], the study of moment equations is an active research field with increasing relevance, e.g. high altitude or microchannel flows, and many recent contributions, see [3, 4] for an overview.

A beneficial property of the moment method is that it leads to a hierarchy of systems with increased accuracy with respect to the true Boltzmann solution and thus successively larger systems can be used to estimate the model error and refine the model as needed [5]. We enumerate the model hierarchy and associate a model level m to each moment system, with higher levels corresponding to larger, more accurate moment systems. However, even the first extension beyond Navier-Stokes-Fourier induces a significant increase in computational cost, due to the larger set of moment variables leading to an elevated number of numerical degrees of freedom [7]. Fortunately, higher model levels are often only required in small regions where local flow or geometric features, such as curved walls, necessitate it [5]. Therefore, to maintain a manageable computational cost, which correlates with the number of degrees of freedom, it suggests itself to only refine the model in regions where it is required to achieve a target accuracy. This leads to the concept of model-adaptive (m-adaptive) moment methods, proposed in [3, 5]. Initial investigations performed in [8] and [9] on one-dimensional problems show promising results, justifying a more extensive exploration of the topic. The central goal of this thesis is to formulate and implement a proof-of-concept numerical framework capable of m-adaptive discretizations, with the aim of facilitating future work. The focus thereby lies on the methods themselves, not on a high performance implementation.

Besides allowing for model adaptivity, several other requirements are imposed on the numerical scheme: The scheme should be (a) higher order, (b) rely on unstructured meshes, (c) capable of representing curved geometries, (d) equipped with a general boundary condition formulation, and (e) adaptive in the discretization. The motivation behind these requirements is summarized as follows: Higher order discretizations were found to yield significant increases in accuracy in the context of moment methods [7], especially for larger Knudsen numbers [5]. Unstructured meshes enable the handling of complex geometries, as motivated by [4]. Curved geometry representations are mandatory to obtain higher order convergence in the presence of curved boundaries [10], which frequently occur in engineering applications. Furthermore, due to rarefied gas effects near the boundary, such as Knudsen layers [4], an accurate approximation near boundaries is important. Moreover, the boundary conditions of the moment systems need to be implemented in a stable manner, in particular on curved domains [5]. Lastly, discretization and model adaptivity should be considered together, motivated by [8], as to not squander any potential gains obtained from model adaptivity on a uniformly refined discretization.

Discontinuous Galerkin (DG) methods are able to handle locally refined irregular meshes of complex geometries and are well suited for the use with adaptive methods [11]. As a consequence, h- and p-adaptivity is commonplace in modern DG frameworks, see for example [12, 13, 14]. The advantages of DG extend to the m-adaptive case, where the number of variables may change from element to element, making it the method of choice for the scheme herein.

Several simplifications are made to keep the scope of the project manageable. We only consider two-dimensional linear first order hyperbolic equations and seek a steady state solution. These kinds of problems arise in the context of slow, microchannel flows, where the linearized moment equations suffice [3]. Lastly, for these linear models, wall boundary conditions are well established [5, 15], thus only wall boundaries are used throughout this work.

The remainder of this thesis is structured in the following way: In Chapter 2, model problems are introduced and their well-posedness is discussed by means of the energy method. To numerically solve these model problems, a discontinuous Galerkin scheme on unstructured curved meshes is derived in Chapter 3. In doing so, particular emphasis is placed on the treatment of boundary conditions and curved geometry approximations. Moreover, the scheme is extended to the *hpm*-adaptive case. Chapter 4 deals with the verification of the numerical methods and their implementation based on convergence studies. Thereafter, the adaptive framework is employed to compute the solution of an example application cases in Chapter 5. Lastly, conclusions are drawn in Chapter 6 and possible directions for future work are given.

2. Model problems

In this chapter, several model problems are introduced and their well-posedness is briefly discussed. While considerably simplified, they serve to represent more complex sets of equations modelling nonequilibrium gas dynamics. The purpose of these model problems is to guide the development of the numerical methods and their implementation presented in the following chapters. Importantly, they provide test cases for verification and analysis, ensuring correctness of the methods. Particular emphasis is put on model cascades with increasing levels of fidelity and accuracy, which form the basis for the model adaptivity pursued in this work.

2.1. General setup

Most generally, the problems treated herein are first order partial differential equations augmented with suitable initial and boundary conditions, yielding an initial-boundary value problem:

Definition 2.1.1 (Initial-boundary value problem)**.** Find the vector-valued function $u : (\mathbf{x}, t) \in \mathbb{R}^d \times \mathbb{R}_{\geq 0} \to u(\mathbf{x}, t) = [u_1, u_2, ..., u_{n_v}]^T \in \mathbb{R}^{n_v}$ that satisfies

$$\partial_t u + \nabla \cdot F(u) = Pu + S, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad t \geq 0, \tag{2.1a}$$

$$Bu = g, \quad \mathbf{x} \in \partial\Omega, \tag{2.1b}$$

$$u(\mathbf{x}, 0) = u_0, \quad \mathbf{x} \in \Omega, \tag{2.1c}$$

where $d \in \mathbb{N}$ is the spatial dimension, Ω the physical domain with boundary $\partial\Omega$, $P : \mathbb{R}^{n_v} \to \mathbb{R}^{n_v}$ is a production or relaxation operator, $S : \mathbf{x} \in \mathbb{R}^d \to S(\mathbf{x}) \in \mathbb{R}^{n_v}$ a spatially varying source term, $B : \mathbb{R}^{n_v} \to \mathbb{R}^{n_{bc}}$ is the boundary condition operator possibly dependent on the position $\mathbf{x} \in \mathbb{R}^d$ and outward facing unit normal $\mathbf{n} \in \mathbb{R}^d$, $g : \mathbf{x} \in \mathbb{R}^d \to g(\mathbf{x}) \in \mathbb{R}^{n_{bc}}$ the boundary data and $u_0 : \mathbf{x} \in \mathbb{R}^d \to u_0(\mathbf{x}) \in \mathbb{R}^{n_v}$ are given initial conditions. The positive natural numbers $n_v \in \mathbb{N}$ and $n_{bc} \in \mathbb{N}$ denote the number of solution variables n_v and the number of boundary conditions n_{bc} respectively. The flux function $F : \mathbb{R}^{n_v} \to \mathbb{R}^{n_v \times d}$ is a matrix-valued function, where each column $F_j(u)$ gives the flux of all variables in direction $j \in \{1, .., d\}$.

In practical applications, the boundary is often split up into several distinct sections Γ_i with $\bigcup_i \Gamma_i = \partial\Omega$ and $\Gamma_i \cap \Gamma_j = \emptyset$ for $i \neq j$. Each of these sections then has their own boundary condition operator $B_i(\mathbf{n}, \mathbf{x})$ and data $g_i(\mathbf{x})$. Furthermore, note that in Definition 2.1.1 we have abbreviated $\partial/\partial t$ with ∂_t and the arguments of all functions and operators are often suppressed for the sake of brevity.

The evolution equation (2.1a) is a system of conservation laws, for which an extensive body of literature exists and the interested reader is referred to any standard textbook, such as [16, 17, 18]. Herein, we only review, based on the book [16], the key properties required to discuss the well-posedness of the present model problems and to construct the numerical scheme for them. Firstly, equation (2.1a) can be written in quasi-linear form:

$$\partial_t u + \sum_{j=1}^{d} A_j \frac{\partial u}{\partial x_j} = Pu + S, \tag{2.2}$$

where the system matrices A_j are Jacobian matrices of the sufficiently smooth flux functions F_j, defined by

$$A_j(u) = \frac{\partial F_j}{\partial u}. \tag{2.3}$$

Based on properties of the flux Jacobian (2.3) the partial differential equation (2.1a) is classified has hyperbolic, which is defined as [16]:

Definition 2.1.2 (Hyperbolic)**.** The system (2.1a) is called *hyperbolic* if, for any $u \in \mathbb{R}^{n_v}$ and any $\boldsymbol{\omega} = [\omega_1, ..., \omega_d]^T \in \mathbb{R}^d$, $\boldsymbol{\omega} \neq 0$, the matrix

$$A(u, \boldsymbol{\omega}) = \sum_{j=1}^{d} \omega_j A_j(u) \tag{2.4}$$

has a full set of linearly independent right eigenvectors v_i with corresponding real eigenvalues $\lambda_i \in \mathbb{R}$, $i \in \{1, ..., n_v\}$.

A consequence of hyperbolicity is that all flux Jacobians A_j, or a linear combination thereof, can be diagonalized by means of an eigendecomposition:

$$A = V\Lambda V^{-1}, \quad \text{where} \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_{n_v}),\ V = [v_1 \,|\, v_2 \,|\, ... \,|\, v_{n_v}]. \tag{2.5}$$

Hence, given a direction j, any solution state at a given point in space and time can be represented in a basis of the eigenvectors of A_j, yielding so-called characteristic variables [16]. A useful intuition behind the eigenvectors is that they represent characteristic modes (i.e. waves) and the corresponding eigenvalues give the wave speeds that these modes are transported with along the direction j. Of particular importance for the formulation of fluxes along interfaces and boundaries is the normal-direction flux Jacobian, denoted as $A_n = A(u, \boldsymbol{\omega} = \mathbf{n})$.

For the well-posedness analysis in Section 2.3, the system (2.1a) is required to be symmetrizable, which is defined as [16]:

Definition 2.1.3 (Symmetrizable)**.** The system of equations (2.1a) is called *symmetrizable* if there exists a symmetric positive definite matrix $\hat{S}(u) \in \mathbb{R}^{n_v \times n_v}$, such that the transformed flux Jacobians

$$\hat{S}(u)A_j(u), \quad j \in \{1, .., d\} \tag{2.6}$$

are symmetric.

Note that a system is symmetrizable if it admits an entropy function [19], which is the case for most physical systems [17]. Alternatively to simultaneously symmetrizing the flux Jacobians with $\hat{S}$, a similarity transformation can be employed:

Proposition 2.1.1. *In the context of Definition 2.1.3 a symmetrizable system can also be symmetrized by means of the similarity transformation*

$$SA_jS^{-1}, \quad \textit{where} \quad S \coloneqq \hat{S}^{1/2}\,. \tag{2.7}$$

Proof. Since $\hat{S}$ is symmetric positive definite, S is well defined by $S = \hat{S}^{1/2} = V\Lambda^{1/2}V^T$, where the columns of V are the orthogonal eigenvectors of $\hat{S}$ and the square root is applied to the eigenvalues Λ_{ii} individually. In particular S is symmetric, which implies that SAS^{-1} is symmetric:

$$\begin{aligned} \implies SAS^{-1} = S^{-1}SSAS^{-1} &= S^{-1}\left(\hat{S}A\right)S^{-1} && (2.8)\\ &= S^{-T}\left(\hat{S}A\right)^T S^{-T} && (2.9)\\ &= \left(S^{-1}\left(\hat{S}A\right)S^{-1}\right)^T = (SAS^{-1})^T\,. && (2.10) \end{aligned}$$

□

The benefit of this approach is that the result of a similarity transformation has the same characteristic polynomial and thus the same eigenvalues [20].

Throughout this work, we consider only two-dimensional ($d = 2$, $\mathbf{x} = [x, y]^T$) linear constant coefficient problems, which constitutes a significant simplification of the general initial-boundary value problem 2.1.1. Consequently, the flux Jacobians, and production operator are constant matrices and the flux function is given by

$$F(u) = [A_x\, u \,|\, A_y\, u], \quad \text{where} \quad A_j = const. \quad \text{for} \quad j \in \{x, y\}. \tag{2.11}$$

The boundary condition operator $B(\mathbf{n}, \mathbf{x})$ may still depend on $\mathbf{n}$ and $\mathbf{x}$, but not on u. Additionally, all problems under consideration are assumed to be non-dimensionalized. Lastly, we seek the steady-state solution of the problem, meaning $\partial_t u = 0$, which, for a suitable choice of boundary conditions, is independent of the initial conditions.

2.2. Heat systems

A fundamental process in engineering applications is heat conduction, where the relevant quantities are the (non-dimensional) temperature θ and heat flux vector $q = [q_x, q_y]^T$. In the following, the classical heat equation and its extension to the thermal nonequilibrium regime are introduced.

2.2.1. The advection-diffusion equation

The starting point is the two-dimensional unsteady advection-diffusion equation recast as an equation for the temperature distribution θ:

$$\partial_t \theta + \mathbf{a} \cdot \nabla \theta - k \Delta \theta = s(\mathbf{x}), \tag{2.12}$$

with advection velocity $\mathbf{a} = [a_x, a_y]^T \in \mathbb{R}^{d=2}$, thermal conductivity $k \in \mathbb{R}_{\geq 0}$ and heat source $s(\mathbf{x})$. To make Equation (2.12) fit into the present framework, we eliminate the second derivative term by introducing the scaled heat flux $q \coloneqq -\kappa \nabla \theta$, where $\kappa = \sqrt{k}$, as a variable and rewriting the system as

$$\partial_t \theta + \nabla \cdot (\mathbf{a}\theta + \kappa q) = s(x, y), \tag{2.13a}$$

$$\partial_t q + \kappa \nabla \theta = -q, \tag{2.13b}$$

where a time derivative of the heat flux is added. Although this changes the dynamics of the system, the steady state remains consistent with Equation (2.12). From Equation (2.13a) the system matrices A_j, production matrix P and source term S are readily identified as

$$A_x = \begin{bmatrix} a_x & \kappa & 0 \\ \kappa & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; A_y = \begin{bmatrix} a_y & 0 & \kappa \\ 0 & 0 & 0 \\ \kappa & 0 & 0 \end{bmatrix}, \; P = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \; S = \begin{bmatrix} s(x,y) \\ 0 \\ 0 \end{bmatrix}. \tag{2.14}$$

The problem is completed by choosing suitable initial and boundary conditions. To obtain a steady-state solution of the time-dependent problem that is identical to the solution of the corresponding time-independent problem, the solution should be independent of the initial conditions. Note that conserved quantities will maintain their volume average over the entire domain Ω, if no exchange occurs at the boundary $\partial\Omega$. Therefore, all variables without a relaxation term need to appear in the boundary conditions on at least part of the boundary.

The boundary condition employed is a classical temperature jump condition [5]:

$$\tilde{\chi} \left(\theta - \theta^w \right) = q \cdot \mathbf{n}, \quad \tilde{\chi} \geq 0, \quad \text{on} \quad \Gamma \subseteq \partial\Omega, \tag{2.15}$$

with prescribed wall temperature θ^w and a wall modelling parameter $\tilde{\chi}$, yielding the boundary condition operator and data

$$B = \begin{bmatrix} \tilde{\chi} & -n_x & -n_y \end{bmatrix}, \quad g(x,y) = \tilde{\chi}\theta^w(x,y), \tag{2.16}$$

where n_x and n_y are the components of the outward unit normal vector $\mathbf{n}$. Temperature Dirichlet and heat flux Neumann conditions are obtained from the limits $\tilde{\chi} \to \infty$ and $\tilde{\chi} \to 0$ respectively.

Note that the coordinate system can be rotated without changing the underlying physics of the problem, as long as $\mathbf{a}$ is rotated accordingly. It follows that, instead of considering a general normal direction $\mathbf{n}$ it can often suffice to only examine $\mathbf{n} = [1, 0]$. All other cases are then assumed to be treatable by rotating the system and its solution variables into the fixed x-direction normal system. Lastly, note that for $\mathbf{a} = 0$ the classical heat equation arising from Fourier's law [21] is obtained and the system becomes isotropic.

2.2.2. The extended heat system

While Fourier's law is sufficient for a broad range of engineering applications, it becomes inaccurate for a gas in thermal nonequilibrium [2, 21]. The cause of this are multiscale dissipation effects, which become relevant when the number of particle collisions in a gas become insufficient to maintain thermal equilibrium [3]. As mentioned in Chapter 1, the magnitude of these effects is quantified by the Knudsen number, computed as the ratio between the gas molecules mean free path between collisions λ and the characteristic length scale of the problem L [3]:

$$\mathrm{Kn} = \frac{\lambda}{L}. \tag{2.17}$$

For $\mathrm{Kn} \gtrsim 0.01$ a gas is called rarefied [2] and exhibits rarefied gas phenomenon, which are not predicted by classical continuum models, see the review article [4] for an overview. This necessitates extended models, derived from kinetic gas theory, to accurately capture these effects. One of the most successful extended models to date are the regularized 13-moment (R13) equations [3, 4, 22]. From these, under considerable simplification, the (linearized) extended heat system is obtained [7]:

$$\partial_t \theta + \nabla \cdot q = s(x,y), \tag{2.18a}$$

$$\partial_t q + \frac{1}{2}\nabla \cdot R + \frac{5}{2}\nabla\theta = -\frac{2}{3\,\mathrm{Kn}} q, \tag{2.18b}$$

$$\partial_t R + \frac{24}{5}\left(\nabla q\right)_{STF} = -\frac{1}{\mathrm{Kn}} R, \tag{2.18c}$$

which constitutes an extension of the classical heat equation to the rarefied regime.

Note that we have again added a time derivative to the last equation to bring the system into a first order form. The subscript $(\cdot)_{STF}$ denotes the symmetric trace-free part of a tensor, which for a rank 2 tensor $A \in \mathbb{R}^{d\times d}$ is constructed as

$$A_{STF} = \frac{1}{2}\left(A + A^T\right) + \frac{1}{3}\mathrm{tr}(A), \tag{2.19}$$

where $\mathrm{tr}(\cdot)$ denotes the trace operator. Hence, the additional variable R is a symmetric trace-free rank 2 tensor, which, in two-dimensions, has three unique entries R_{xx}, R_{xy} and R_{yy}.

The boundary conditions considered herein are wall boundary conditions given by [7, 15]:

$$2\tilde{\chi}\left(\theta - \theta^w + \frac{1}{5}R_{nn}\right) = q_n, \tag{2.20a}$$

$$\frac{11}{5}\tilde{\chi}\left(q_t - q_t^w\right) = R_{nt}, \tag{2.20b}$$

with specified tangential wall heat flux q_t^w, wall temperature θ^w and free parameter $\tilde{\chi} \geq 0$. Note that as these conditions are quite general slip boundary conditions, their implementation in a numerical scheme requires special care.

From Equations (2.18) and (2.20) the system matrices, production matrix, boundary condition operator and boundary data are readily obtained:

$$A_x = \left[\begin{array}{c|cc|ccc} 0 & 1 & 0 & 0 & 0 & 0 \\ \hline \frac{5}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \hline 0 & \frac{16}{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{12}{5} & 0 & 0 & 0 \\ 0 & -\frac{8}{5} & 0 & 0 & 0 & 0 \end{array}\right], \quad A_y = \left[\begin{array}{c|cc|ccc} 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{5}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \hline 0 & 0 & -\frac{8}{5} & 0 & 0 & 0 \\ 0 & \frac{12}{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{16}{5} & 0 & 0 & 0 \end{array}\right], \tag{2.21a}$$

$$P = \mathrm{diag}\left(0, -\frac{2}{3\,\mathrm{Kn}}, -\frac{2}{3\,\mathrm{Kn}}, -\frac{1}{\mathrm{Kn}}, -\frac{1}{\mathrm{Kn}}, -\frac{1}{\mathrm{Kn}}\right), \tag{2.21b}$$

$$B = \left[\begin{array}{c|cc|ccc} 2\tilde{\chi} & -n_x & -n_y & \frac{2}{5}\tilde{\chi}n_x^2 & \frac{4}{5}\tilde{\chi}n_x n_y & \frac{2}{5}\tilde{\chi}n_y^2 \\ \hline 0 & \frac{11}{5}\tilde{\chi}n_y & \frac{11}{5}\tilde{\chi}n_x & n_x n_y & n_y^2 - n_x^2 & -n_x n_y \end{array}\right], \quad g = \left[\begin{array}{c} 2\tilde{\chi}\theta^w \\ \hline \frac{11}{5}\tilde{\chi}q_t^w \end{array}\right]. \tag{2.21c}$$

The matrices are structured in terms of the variables of the system. Neglecting the blocks associated with the last variable, which corresponds to setting $R = 0$, reduces the equations to the (non-symmetric) heat equations. Analogously, the boundary conditions reduce to those of the heat system (2.16). Thus the heat system is nested inside the extended heat system, forming a simple model cascade over two model levels.

Due to the increase in variables and accompanying computational effort, it makes sense to only employ the extended system in regions of the domain where it is necessary to maintain model accuracy. Hence, both heat systems together form a basic model-adaptive system. It is for this reason, that the extended heat system is studied in this work. The goal being a numerical scheme, which facilitates dynamically switching between different model levels in different parts of the domain.

2.3. Well-posedness of continuous problems

The reliability of any numerical scheme used to find approximate solutions of the continuous initial-boundary value problem 2.1.1 depends on the well-posedness of the continuous problem [23]. For (symmetric) hyperbolic problems, the pure initial value problem, i.e. the Cauchy problem, is well-posed [19, 24]. Therefore, the well-posedness of the initial-boundary value problem often hinges on the choice of admissible boundary conditions. Moreover, well-posed boundary conditions and their stable implementation is especially important for moment equations [5].

Motivated by the above, we briefly discuss the well-posedness of the model problems under consideration. Herein only the essentials are mentioned. The interested reader is referred to the textbook [24] for a rigorous introduction and to [23, 25] for a more applied examination of well-posedness. In accordance with the references, we define well-posedness in the sense of [23, 24]:

Definition 2.3.1 (Well-posed)**.** The initial-boundary value problem 2.1.1 is called *strongly well-posed* if for smooth and compatible data S, u_0, g there exists a unique smooth solution u such that

$$\|u(\cdot,t)\|^2_{L_2(\Omega)} \leq K_T \left\{ \|u_0\|^2_{L_2(\Omega)} + \int_0^t \|S(\cdot,\tau)\|^2_{L_2(\Omega)} + \|g(\cdot,\tau)\|^2_{L_2(\partial\Omega)} d\tau \right\} \tag{2.22}$$

holds for every finite time interval $0 \leq t \leq T \in \mathbb{R}_{\geq 0}$ with the constant $K_T \in \mathbb{R}_{\geq 0}$ being independent of the data S, u_0, g. If instead $g = 0$ and an analogous estimate to (2.22) without boundary terms holds, the problem is called *well-posed*.

Note that the estimate for strongly well-posedness in [24] contains an additional solution boundary term on the left-hand side, which is not present in the definition found in [23] and also omitted here. We neglect the compatibility between the initial and boundary conditions for practical simulations to the steady state, where zero initial conditions are chosen even for inhomogeneous boundary conditions. Lastly, note that the following analysis only applies to problems with smooth data and smooth solutions [23].

2.3.1. The energy method

In this section, the energy method is used to derive sufficient conditions on the boundary conditions (2.1b) leading to a well-posed problem. The presented derivation is heavily based on [23, 25], in which further details can be found. The starting point is a homogeneous, meaning zero source term ($S = 0$), constant coefficient problem with symmetric coefficient matrices. We additionally restrict ourselves to hyperbolic problems in the form of Equation (2.1a). The quasi-linear form of the conservation law (2.2) is then multiplied with u^T from the left and integrated over the domain Ω, yielding:

$$\int_\Omega u^T \partial_t u \, dx + \int_\Omega u^T \sum_{j=1}^{d} A_j \partial_{x_j} u \, dx = \int_\Omega u^T P u \, dx. \tag{2.23}$$

Utilizing that all A_j are constant and symmetric, the chain rule of differentiation is applied and the resulting system rearranged:

$$\int_\Omega \frac{1}{2} \partial_t \left(u^T u \right) \, dx + \int_\Omega \frac{1}{2} \sum_{j=1}^{d} \partial_{x_j} \left(u^T A_j \, u \right) \, dx = \int_\Omega u^T P u \, dx \tag{2.24}$$

$$\iff \qquad \partial_t \int_\Omega u^T u \, dx + \int_\Omega \sum_{j=1}^{d} \partial_{x_j} \left(u^T A_j u \right) \, dx = 2 \int_\Omega u^T P u \, dx \tag{2.25}$$

$$\iff \qquad \partial_t \|u\|^2_{L_2(\Omega)} + \oint_{\partial\Omega} u^T A_n u \, ds = 2 \int_\Omega u^T P u \, dx, \tag{2.26}$$

where Gauss' divergence theorem is applied in the last step. A bounded derivative of the energy $\|u\|^2_{L_2(\Omega)}$ implies a bounded energy and consequently a bounded solution u [25]. Therefore, the two other terms in (2.26) need to be either bounded exclusively by the data g or they need to lead to a reduction of the energy. Since the production term on the right-hand side depends on the solution, we require that P be negative semi-definite, i.e. $u^T P u \leq 0$, which has a dampening effect on the energy of the system. Otherwise the energy may be amplified, causing to exponential growth that cannot be bounded independent of the solution. The above derivation yields the *energy estimate* [25]:

$$\partial_t \|u\|^2_{L_2(\Omega)} + \oint_{\partial\Omega} u^T A_n u \, ds \leq 0. \tag{2.27}$$

Clearly, to bound the energy it remains to estimate the term $u^T A_n u$ by imposing suitable boundary conditions on u. By diagonalizing the symmetric real matrix $A_n = V \Lambda V^T$, the modes of u contributing to energy increase and decrease can be identified. Sorting the entries $\lambda_i \in \mathbb{R}$ of the diagonal matrix Λ by their sign, the matrix V is organized into blocks $[V_+ \,|\, V_- \,|\, V_0]$ corresponding to blocks of $\Lambda = \operatorname{diag}(\Lambda_+, \Lambda_-, \Lambda_0)$. Defining and decomposing $w = V^T u = [w_+ \,|\, w_- \,|\, w_0]^T$ the boundary term integrand

of the energy estimate (2.27) is written as [23]:

$$u^T A_n u = \begin{bmatrix} w_+ \\ w_- \end{bmatrix} \begin{bmatrix} \Lambda_+ & 0 \\ 0 & \Lambda_- \end{bmatrix} \begin{bmatrix} w_+ \\ w_- \end{bmatrix}. \tag{2.28}$$

Let $n_- \in \mathbb{N}$ and $n_+ \in \mathbb{N}$ be the number of negative and positive λ_i respectively. A solution-independent estimate of (2.28) can only be obtained by constraining all negative (i.e. incoming) modes w_-. For this reason the minimum number of boundary conditions is n_-. Imposing more than the minimum number of boundary conditions jeopardizes the existence of the solution [24]. This means that the number of boundary conditions leading to well-posedness is $n_{bc} = n_-$ [23]. Each condition then specifies one negative mode by means of the other variables present, yielding the form of the boundary conditions given by [23]:

$$w_- = Rw_+ + g, \quad R \in \mathbb{R}^{n_- \times n_+}. \tag{2.29}$$

Note that the prescription of w_- cannot contain parts of w_0, meaning that the zero modes w_0 must not appear on the right-hand side of (2.29), since $w_0^T \Lambda_- w_0 \leq 0$ cannot be estimated in terms of the data and therefore an energy estimate cannot be established. Formally, the nullspace of the boundary condition operator needs to contain the nullspace of A_n, spanned by w_0, which is referred to as the *nullspace condition* [15]:

$$\ker(A_n) \subseteq \ker(B). \tag{2.30}$$

For a hyperbolic problem, using an eigendecomposition for the diagonalization makes the columns of V the eigenvectors, λ_i the eigenvalues and w the characteristic variables corresponding to individual wave modes. In general, providing values for only the incoming characteristics and leaving the outgoing ones untouched leads to well-posed problems [24]. However, in practice this overly restricts the space of possible boundary conditions. Therefore, we use so-called characteristic boundary conditions, where the incoming waves are specified as a linear combination of the outgoing waves [23]. This is exactly what (2.29) does.

The last step in the procedure is to derive conditions on the coefficient matrix R, such that the boundary integral of the energy estimate (2.27) is bounded only by the data g. In the simplest case, $R = 0$ and the integrand is estimated as

$$u^T A_n u = w_+^T \Lambda_+ w_+ + g^T \Lambda_- g \geq g^T \Lambda_- g, \tag{2.31}$$

giving a lower bound on the boundary integral term and thus an upper bound on the growth of the energy. For $R \neq 0$ the situation is a bit more complex. Roughly speaking, the prescription of incoming modes w_- by outgoing modes w_+ reflects

the energy of the outgoing modes back into the domain. Care needs to be taken to not reflect more into the domain than what is going out. Hence, the combined energy of incoming modes cannot be more that the combined energy of all outgoing modes, otherwise solution-dependent amplification may occur. Concrete conditions are obtained by inserting Equation (2.29) into Equation (2.28), resulting in:

$$u^T A_n u = \begin{bmatrix} w_+ \\ g \end{bmatrix} \begin{bmatrix} \Lambda_+ + R^T \Lambda_- R & \Lambda_- g \\ \Lambda_- g & \Lambda_- \end{bmatrix} \begin{bmatrix} w_+ \\ g \end{bmatrix}, \tag{2.32}$$

which, setting $g = 0$, yields the condition [23]:

$$\Lambda_+ + R^T \Lambda_- R \geq 0. \tag{2.33}$$

Satisfaction of the condition (2.33) leads to a well-posed problem [23]. Strongly well-posedness for inhomogeneous boundary conditions ($g \neq 0$) requires either the bound to be sharpened to $\Lambda_+ + R^T \Lambda_- R > 0$ [23] or for

$$R^T \Lambda_- g \in \text{range}\left(\Lambda_+ + R^T \Lambda_- R\right) \tag{2.34}$$

to hold in addition to (2.33) [15].

2.3.2. Well-posed boundary conditions for hyperbolic systems

Both number of boundary conditions n_{bc} and the form of the boundary conditions (2.29), constrained by (2.33), yielding a well-posed problem are now known. To show well-posedness of the model problems it remains to relate the matrix R to the boundary condition operator B. To expose the contributions of each mode w_i to the individual boundary conditions, we multiply the operator B with V from the right and split the result into three blocks:

$$BV = [B_+ \,|\, B_- \,|\, B_0], \quad \text{where} \quad B_+ = BV_+,\ B_- = BV_-,\ B_0 = BV_0. \tag{2.35}$$

For all incoming modes to be uniquely specified, the matrix $B_- \in \mathbb{R}^{n_{bc} \times n_-}$ needs to be full rank and square. Thus the boundary condition (2.1b) can be transformed by multiplication with B_-^{-1} from the left:

$$B_-^{-1} B u = B_-^{-1} g =: \hat{g}, \tag{2.36}$$

where we define the transformed boundary data $\hat{g}$. Note that the condition (2.34) now needs to be evaluated with $\hat{g}$. Equation (2.35) becomes

$$B_-^{-1} BV = \left[B_-^{-1} B_+ \,|\, I_{n_{bc} \times n_{bc}} \,|\, B_0\right], \tag{2.37}$$

where $I_{n_{bc}\times n_{bc}}$ is the $n_{bc} \times n_{bc}$ identity matrix, indicating that each incoming mode is uniquely associated with a single transformed boundary condition.

Equating the transformed boundary conditions (2.36) with the form of the boundary conditions (2.29) obtained from the energy method and rearranging yields:

$$B_-^{-1}Bu = \hat{g} = w_- - Rw_+ = \left(V_-^T - RV_+^T\right) u \tag{2.38}$$

$$\implies B_-^{-1}B = \left(V_-^T - RV_+^T\right) \tag{2.39}$$

$$\implies B_-^{-1}BV = \left(V_-^T - RV_+^T\right) V = \left[R \,|\, I_{n_{bc}\times n_-} \,|\, 0_{n_{bc}\times n_0}\right], \tag{2.40}$$

where $n_0 \in \mathbb{N}$ is the number of zero modes and $0_{n_{bc}\times n_0}$ is a $n_{bc} \times n_0$ matrix filled with zeros. By comparing the blocks of (2.40) with (2.37) the nullspace condition $B_0 = 0$ and the coefficient matrix

$$R \coloneqq B_-^{-1}B_+ \tag{2.41}$$

are identified. Note that a similar derivation to the one above can be found in [5, 15].

Given B and symmetric flux Jacobians A_j the well-posedness of the problem 2.1.1 can be verified by diagonalizing $A_n = V\Lambda V^T$, calculating R and testing the condition (2.33). In general, A_n and B depend on the normal direction $\mathbf{n}$, hence the well-posedness needs to be verified algebraically for all possible directions. This can be circumvented by only considering a single fixed direction, for example $\mathbf{n} = [1, 0]^T$, if the equations are rotationally invariant. For any arbitrary direction the problem can be rotated to the chosen reference direction, yielding the same matrices A_n and $B_n = B(\mathbf{n})$ and therefore the same conditions for well-posedness apply.

Practical simulations are often based on a non-symmetric form of the problem and employ an eigendecomposition of the local non-symmetric flux Jacobian A_n to prescribe characteristic boundary conditions, see for example [26]. The boundary conditions enforced in this way are the same as those of the symmetrized system, up to a change of variables. Additionally, when the decomposition is explicitly available, it suffices to look at the non-symmetric system to verify well-posedness:

Proposition 2.3.1. *Let B_n and A_n be the boundary operator and the flux Jacobian with respect to the direction $\mathbf{n}$ of a non-symmetric hyperbolic problem. Further let $A_n = Q\Lambda Q^{-1}$ denote the eigendecomposition of A_n. If the problem is symmetrizable, the well-posedness can be determined on the basis of B_+, B_- and B_0 obtained from B_nQ and the eigenvalues Λ_{ii}, without needing to symmetrize the problem.*

Proof. Since the problem is symmetrizable, by proposition 2.1.1 there exists a matrix S such that $SA_nS^{-1} = \hat{A}_n$ is symmetric. The boundary conditions of the symmetric problem are given by

$$B_nu = B_nS^{-1}Su = \hat{B}_n\hat{u} = g, \tag{2.42}$$

where $\hat{B}_n \coloneqq B_n S^{-1}$ and $\hat{u} \coloneqq Su$ are the boundary condition operator and the solution variables of the symmetric problem. Furthermore, the eigenvectors q_i of the non-symmetric flux Jacobian satisfy

$$A_n q_i = \lambda_i q_i \implies S A_n S^{-1} S q_i = \lambda_i S q_i \implies \hat{A}_n v_i = \lambda_i v_i, \quad \text{with} \quad v_i = S q_i. \tag{2.43}$$

In particular, we see that $v_i = Sq_i$ satisfies the defining equation for the eigenvectors of the symmetric flux Jacobian. Let Q and V be the eigenvector matrices corresponding to q_i and v_i. The decomposition of the symmetric boundary condition operator can therefore be computed as

$$B_n Q = B_n S^{-1} V = \hat{B}_n V = [B_+ \,|\, B_- \,|\, B_0]\,. \tag{2.44}$$

Furthermore, since $\hat{A}_n$ is similar to A_n, the eigenvalues $\lambda_i = \Lambda_{ii}$ are the same [20]. Consequently, conditions (2.33), (2.30), (2.34) and the number of boundary conditions coincide for the symmetrized and non-symmetric system. □

2.3.3. Well-posedness of the heat systems

To exemplify the derived conditions and to ensure well-posedness of the considered model problems, the well-posedness of the heat systems presented in Section 2.2 is verified in the following. Note that, due to rotational invariance, it suffices to only consider the x-direction.

Example (Heat system)**.** Taking the system matrices (2.14) with $\mathbf{a} = 0$ we find:

$$A_x = \begin{bmatrix} 0 & \kappa & 0 \\ \kappa & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \implies \Lambda = \begin{bmatrix} \kappa & 0 & 0 \\ 0 & -\kappa & 0 \\ 0 & 0 & 0 \end{bmatrix}, V = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.45}$$

$$B_x = \begin{bmatrix} \tilde{\chi} & -1 & 0 \end{bmatrix} \implies BV = \begin{bmatrix} \tilde{\chi} - 1 & -(\tilde{\chi} + 1) & 0 \end{bmatrix}, \tag{2.46}$$

from which it follows that the nullspace condition is satisfied and condition (2.33) becomes

$$\kappa - \kappa \underbrace{\left(\frac{-(\tilde{\chi}+1)}{\tilde{\chi}-1} \right)^2}_{=R} \geq 0 \implies (\tilde{\chi}+1)^2 \geq (\tilde{\chi}-1)^2\,. \tag{2.47}$$

Hence, the heat system is well-posed for $\tilde{\chi} \geq 0$. As mentioned in Section 2.2.1, the boundary conditions (2.16) are a linear combination of Dirichlet and Neumann boundary conditions. Clearly, the conditions need to be added together such that the incoming wave has equal or less amplitude than the outgoing wave, which is only the case for $\tilde{\chi} \geq 0$.

Example (Extended heat system)**.** From the system matrices (2.21) the eigenvalues and eigenvectors of A_x and the boundary condition operator in characteristic variables are determined as:

$$\Lambda = \operatorname{diag}\left(\sqrt{\frac{41}{10}}, \sqrt{\frac{6}{5}}, -\sqrt{\frac{41}{10}}, -\sqrt{\frac{6}{5}}, 0, 0\right), \tag{2.48}$$

$$Q = \begin{bmatrix} -\frac{5}{8} & -\frac{5}{8} & 0 & 0 & 0 & -\frac{1}{5} \\ \sqrt{\frac{205}{128}} & -\sqrt{\frac{205}{128}} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\sqrt{\frac{5}{24}} & \sqrt{\frac{5}{24}} & 0 & 0 \\ -2 & -2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.49}$$

$$B_x Q = \begin{bmatrix} \frac{5\sqrt{410}-164\tilde{\chi}}{80} & 0 & \frac{-5\sqrt{410}-164\tilde{\chi}}{80} & 0 & 0 & 0 \\ 0 & \frac{11}{2\sqrt{30}}\tilde{\chi} - 1 & 0 & -\frac{11}{2\sqrt{30}}\tilde{\chi} - 1 & 0 & 0 \end{bmatrix}, \tag{2.50}$$

from which it follows that the nullspace condition is satisfied. Note that the first boundary condition must contain the contribution from the variable R, otherwise the boundary condition would violate the nullspace condition [5]. For hyperbolic problems, specifying characteristic variables corresponding to zero speeds is not allowed [24], meaning that the nullspace condition is necessary for well-posedness. Therefore, a simple temperature jump condition may not be prescribed.

For the extended heat system, condition (2.33) becomes

$$\begin{gathered} \underbrace{\begin{bmatrix} \sqrt{\frac{41}{10}} & 0 \\ 0 & \sqrt{\frac{6}{5}} \end{bmatrix}}_{\Lambda_+} + R^T \underbrace{\begin{bmatrix} -\sqrt{\frac{41}{10}} & 0 \\ 0 & -\sqrt{\frac{6}{5}} \end{bmatrix}}_{\Lambda_-} R \geq 0, \\ \text{where} \quad R = \begin{bmatrix} 1 - \frac{50}{25+2\sqrt{410}\tilde{\chi}} & 0 \\ 0 & \frac{120}{60+11\sqrt{30}\tilde{\chi}} - 1 \end{bmatrix}, \end{gathered} \tag{2.51}$$

which yields the two conditions:

$$\frac{\tilde{\chi}}{\left(25 + 2\sqrt{410}\tilde{\chi}\right)^2} \geq 0 \quad \text{and} \quad \frac{\tilde{\chi}}{\left(60 + 11\sqrt{30}\tilde{\chi}\right)^2} \geq 0. \tag{2.52}$$

Consequently, the extended heat system is well-posed for $\tilde{\chi} \geq 0$.

2.4. Hierarchical moment approximations of the Boltzmann equation

As mentioned in Chapter 1, the moment method is one way to derive extended models in kinetic gas theory describing thermal nonequilibrium flows. One example for moment equations is the extended heat system presented in Section 2.2.2, describing the temperature and heat flux in a rarefied or microscopic setting. More general sets of equations containing further macroscopic quantities of interest for engineering applications, such as density, velocity and stress tensor, are also readily derived by means of the moment method.

In the following, based on the reference [5], the derivation of a particular set of moment equations is briefly outlined. Similar derivations can be found in [27, 28]. For a general introduction into moment equations and their derivation we refer to the textbook [2], the reviews [3, 4] and the references cited therein. The presented approach yields a hierarchy of models which are suitable for a model-adaptive discretization scheme and are used in Chapter 5 to demonstrate the capabilities of the numerical scheme discussed in this work. We focus on a microscopic setting, where the Knudsen number is small due to the small length scale of the problem, such as the flow side of a microchannel. In these cases, linearized equations are often sufficient, because the considered processes are slow [3].

2.4.1. Linearized moment equations

As mentioned above, this section follows the derivation presented in [5] with additions from other sources. The starting point is a monatomic ideal gas which obeys the governing equation of kinetic gas theory, the Boltzmann equation, given by [2]:

$$\partial_t f + c_i \partial_{x_i} f = S(f), \tag{2.53}$$

where $f(\mathbf{c}; \rho, \mathbf{v}, \theta)$ is the particle distribution function, dependent on the particle velocity $\mathbf{c} \in \mathbb{R}^d$, the macroscopic velocity $\mathbf{v} \in \mathbb{R}^d$, density $\rho \in \mathbb{R}$ and temperature $\theta \in \mathbb{R}$. Vectors are denoted in bold and their components by means of free indices, e.g. c_i are the components of the particle velocity $\mathbf{c}$. Note that external forces are neglected and local source terms may be added to the right hand side. The collision operator $S(f)$ drives the distribution function towards thermal equilibrium, at which $S(f) = 0$ holds and the equilibrium distribution function is given by a Maxwellian [2]:

$$f_M(\mathbf{c}; \rho, \mathbf{v}, \theta) = \frac{\rho}{m\sqrt{2\pi\theta}^3} \exp\left(-\frac{(c_i - v_i)^2}{2\theta}\right), \tag{2.54}$$

where m is the particle mass.

For slow flows, the distribution function can be split into a constant equilibrium background state $f_0(\mathbf{c}) = f_M(\mathbf{c}; \rho_0, 0, \theta_0)$ and a perturbation $\tilde{f}$, such that $f = f_0 + \varepsilon\tilde{f}$. The constant background state is assumed to be at rest ($\mathbf{v} = 0$) with a background density ρ_0 and temperature θ_0, such that the macroscopic fields satisfy [27]:

$$\rho = \rho_0 + \varepsilon\tilde{\rho}, \quad v_i = \varepsilon\tilde{v}_i, \quad \theta = \theta_0 + \varepsilon\tilde{\theta}. \tag{2.55}$$

The variables of interest are the perturbations, for which an equation is obtained by inserting the perturbation ansatz into the Boltzmann equation (2.53), exploiting linearity to cancel out the equation for the ground state and rescaling the remaining equation by $1/\varepsilon$:

$$\partial_t(f_0 + \varepsilon\tilde{f}) + c_i\partial_{x_i} f(f_0 + \varepsilon\tilde{f}) = S(f_0 + \varepsilon\tilde{f}) \tag{2.56}$$

$$\implies \partial_t\tilde{f} + c_i\partial_{x_i}\tilde{f} = S(\tilde{f}). \tag{2.57}$$

While the Boltzmann equation for the perturbation $\tilde{f}$ has the same structure as the Boltzmann equation for the full distribution function, the advantage is that subsequent approximations yield linear equations that are linearized versions of the nonlinear equations derived without a perturbation approach.

Due to the distribution function's dependence on the position $\mathbf{x} \in \mathbb{R}^3$ and the velocity $\mathbf{c} \in \mathbb{R}^3$, meaning the problem is six-dimensional, the solution of the Boltzmann equation by means of numerical methods is prohibitively expensive [3]. To remedy this, the dependence on the velocity is eliminated by a suitable discretization in velocity space, resulting in a system of moment equations whose solution approximates that of the Boltzmann equation. The distribution function is therefore approximated by [27]:

$$f_N(\mathbf{x}, t, \mathbf{c}) = (1 + \varepsilon\varphi(\mathbf{x}, t, \mathbf{c}))\, f_0(\mathbf{c}) = f_0 + \varepsilon\tilde{f}_N, \tag{2.58}$$

where φ is a polynomial in the velocity given by the spectral expansion [5]

$$\varphi(\mathbf{x}, t, \mathbf{c}) = \sum_{n=0}^{N_d}\sum_{s=0}^{M_n} w^{(s)}_{i_1 i_2 \cdots i_n}(\mathbf{x}, t)\psi^{(s)}_{i_1 i_2 \cdots i_n}(\mathbf{c}), \tag{2.59}$$

with fully symmetric trace-free coefficients $w^{(s)}_{i_1 i_2 \cdots i_n}$ and polynomial basis functions $\psi^{(s)}_{i_1 i_2 \cdots i_n}$. Summation over repeated indices $i_k \in 1, 2, 3$, i.e. summation convention, is implied in the above formula and in the following. The obtained moment system depends on the choice of the highest tensor degree N_d and the number of trace coefficients M_n on each level n [5].

The basis functions are orthogonal velocity polynomials given by [5, 27]:

$$\psi^{(s)}_{i_1 i_2 \cdots i_n}(\mathbf{c}) = L(\mathbf{c}/\theta^{1/2}), \quad \text{where} \quad L(\xi) \coloneqq \xi_{\langle i_1}\xi_{i_2}\cdots\xi_{i_n\rangle} p^{(n)}_s(\xi^2/2). \tag{2.60}$$

Details on the construction of the fully symmetric trace-free tensor $\xi_{\langle i_1}\xi_{i_2}\cdots\xi_{i_n\rangle}$ of degree n and the polynomials $p_s^{(n)}$ are defined in [5].

Multiplying the distribution function $\tilde{f}_N$ with the basis functions scaled by the particle mass m, and integrating over the velocity space yields moments of the distribution function $\tilde{w}^{(s)}_{i_1 i_2\cdots i_n}(\mathbf{x},t)$, given by [5]

$$\tilde{w}^{(s)}_{i_1 i_2\cdots i_n}(\mathbf{x},t) := \alpha^{(s,n)} w^{(s)}_{i_1 i_2\cdots i_n}(\mathbf{x},t) = m\int_{\mathbb{R}^3} \psi^{(s)}_{i_1 i_2\cdots i_n}(\mathbf{c})\tilde{f}_N(\mathbf{x},t,\mathbf{c})\,d\mathbf{c}. \tag{2.61}$$

Due to the orthogonality of the basis functions, the moments are the corresponding coefficients $w^{(s)}_{i_1 i_2\cdots i_n}(\mathbf{x},t)$ scaled by normalization factors $\alpha^{(s,n)}$, see [5]. The moments depend only on space and time and are related to fluid dynamic quantities [5, 27]:

$$\tilde{w}^0 = \frac{\tilde{\rho}}{\rho_0},\ \tilde{w}^0_i = \rho_0\frac{\tilde{v}_i}{\theta_0^{1/2}},\ \tilde{w}^1 = -\rho_0\frac{3\tilde{\theta}}{2\theta_0},\ \tilde{w}^0_{ij} = \frac{\tilde{\sigma}_{ij}}{\theta_0},\ \tilde{w}^1_i = -\frac{\tilde{q}_i}{\theta_0^{3/2}}, \tag{2.62}$$

where $\tilde{q}_i$ and σ_{ij} are the perturbations of the heat flux and stress tensor respectively. It remains to derive a set of partial differential equations for which the moments are the solution variables. This is done by inserting the distribution function $\tilde{f}_N$ into the Boltzmann equation (2.56), multiplying my the basis functions scaled with the particle mass m and integrating over the velocity space, yielding

$$\partial_t \underbrace{m\int_{\mathbb{R}^3}\psi^{(s)}_{i_1 i_2\cdots i_n}\tilde{f}_N\,d\mathbf{c}}_{=\tilde{w}^{(s)}_{i_1 i_2\cdots i_n}(\mathbf{x},t)=:u} + \partial_{x_j}\underbrace{m\int_{\mathbb{R}^3} c_j\psi^{(s)}_{i_1 i_2\cdots i_n}\tilde{f}_N\,d\mathbf{c}}_{=:A_j u} = \underbrace{m\int_{\mathbb{R}^3}\psi^{(s)}_{i_1 i_2\cdots i_n}S(\tilde{f}_N)\,d\mathbf{c}}_{=:P^{(s)}_{i_1 i_2\cdots i_n}}, \tag{2.63}$$

where the moments are collected into a variable vector $u(\mathbf{x},t)$, resulting in a hyperbolic, symmetrizable and rotationally invariant relaxation system [5] fitting into the present framework given by Definition 2.1.1. The system matrices A_j are computed using computer algebra software, along the lines of [29]. The production term on the right hand side of Equation (2.63) depends on the approximation of the collision operator [5]. The simplest case is given by the linearized Bhatnagar-Gross-Krook (BGK) collision operator [2]:

$$S(\tilde{f}) = -\frac{1}{\tau}(\tilde{f} - \tilde{f}_M), \tag{2.64}$$

where τ is a relaxation time usually taken to be synonymous with the Knudsen number Kn. The resulting production operator reads as [5]

$$P^{(s)}_{i_1 i_2\cdots i_n} = -\frac{1}{\tau}a^{(s,n)}\tilde{w}^{(s)}_{i_1 i_2\cdots i_n}, \tag{2.65}$$

where $a^{(s,n)} = 1$ for all (s,n) except for $(s,n)\in\{(0,0),(0,1),(1,0)\}$ [27], meaning that the moments related to mass, momentum and energy are conserved while the higher moments relax towards a thermal equilibrium, where they are zero.

The moment equations (2.63) are a projection of the Boltzmann equation (2.56) onto the basis functions (2.60) [5]. This indicates a useful perspective for understanding the moment method: The moment equations can be interpreted as a velocity space Galerkin approximation of the Boltzmann equation [9, 30]. In this context, the discrete subspace is spanned by the basis functions (2.60) which act as test functions. The approximate distribution function (2.58) given by the expansion (2.59) is seen as a linear combination of trial functions with the coefficients being the degrees of freedom of the Galerkin discretization. Equation (2.63) can then be interpreted as a Galerkin weak form. Differences compared to a conventional Galerkin approximation are the presence of the ground state in the approximate distribution function, the choice of global polynomials and that the integrals of the weak form are evaluated algebraically.

To use the derived moment equations in a model-adaptive context, it remains to specify a hierarchy of models and a projection operator to project solution variables from one level to another. For the model hierarchy, we consider the so-called *ordered theories* [27], starting at the Grad 13 equations ($N_d = 2$), where M_n is selected as $M_n = \lceil (N_d - n)/2 \rceil$ with the ceiling function $\lceil \cdot \rceil$. Every $N_d \geq 2$ constitutes one set of moment equations and is associated with a corresponding model level $m \in \mathbb{N}$. The two-dimensional system matrices A_x of the first three model levels are depicted in Figure 2.1. Each system matrix is nested withing the matrices of the higher levels, meaning that each successive model level extends the equations and the set of solution variables.

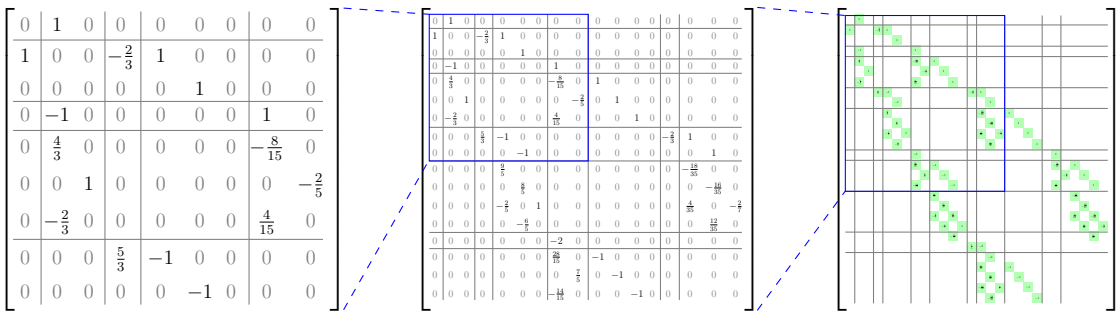


Figure 2.1. System matrices A_x of the linearized moment equations stemming from three successive ordered moment theories. Each matrix is nested within the higher level matrices, forming a hierarchical model. Row and column delimiters indicate the structure of the tensor variables. For the rightmost matrix, zeros are omitted and the sparsity pattern is highlighted. Inspired by [5].

For model-adaptivity to work, we assume convergence of the model hierarchy to the solution of the Boltzmann equation, as done in [5]. Further, note that any ordered tuple $(N_{d,1}, \ldots, N_{d,m})$ describes a valid model hierarchy.

Lastly, when adapting the model level, a projection operator between model levels is required. This is accomplished by means of the definition of the moments (2.61), where the target moments are computed by means of the target basis functions and the source distribution function, constructed from the source moments. We define a projection operator $\boldsymbol{P}^{m^s \to m^t}$ projecting from a source level m^s to a target level m^t and simply write

$$u^t = \boldsymbol{P}^{m^s \to m^t} u^s \tag{2.66}$$

to denote the projection of the variables u^s into u^t. Due to the orthogonality of the basis functions, the projection between model levels is equivalent to a extension of the variable vector with zeros when $m^s > m^t$, or discarding the higher level variables when $m^s < m^t$.

2.4.2. Wall boundary conditions

It remains to specify suitable boundary conditions for the moment equations. We consider only wall boundary conditions and refer to [31] for the case of inflow and outflow conditions. In the kinetic picture, all particles entering the domain need to be prescribed, thus the incoming part of the distribution function f needs to be specified at the boundaries [32]. This is accomplished by means of the Maxwell accommodation model [5, 32]:

$$f(\mathbf{c})|_{\partial\Omega} = \begin{cases} \chi f^w(\mathbf{c}) + (1-\chi) f(\mathbf{c}^\star) & c_n < 0, \\ f(\mathbf{c}) & c_n \geq 0, \end{cases} \tag{2.67}$$

where $c_n = \mathbf{n} \cdot \mathbf{c}$ is the velocity component in direction of the outward wall normal and $c^\star$ is the reflected particle velocity. The accommodation coefficient χ gives the ratio of accommodated to specularly reflected particles that make up the incoming part of the distribution function at the wall $f|_{\partial\Omega}$ [32]. Accommodated particles take on the wall distribution function $f^w = f_M(\mathbf{c}; \rho^w, 0, \theta^w)$, where the wall temperature θ^w needs to be specified and the wall density ρ_w follows from mass conservation [32].

By splitting the distribution function into even and odd parts with respect to the normal velocity component c_n the accommodation model (2.67) can be written as [5]

$$\int_{c_n<0} \psi^{(BC)} f_N^{(odd)} \, d\mathbf{c} = \frac{\chi}{2-\chi} \int_{c_n<0} \psi^{(BC)} f^w - f_N^{(even)} \, d\mathbf{c}, \tag{2.68}$$

where $\psi^{(BC)}$ are the basis functions corresponding to the odd moments, which are prescribed by the boundary conditions. By inserting the distribution function $\tilde{f}_N$ and the basis polynomials into the above expression, the coefficients of the linear

boundary condition operator B_n are derived [5]. Lastly, similarly to [32], we define the modified accommodation coefficient $\tilde{\chi}$ as

$$\tilde{\chi} \coloneqq \frac{\sqrt{2\pi}\chi}{2-\chi} \tag{2.69}$$

and note that the accommodation coefficient can be chosen differently for each boundary condition [3].

In general, boundary conditions derived in the above manner do not lead to well-posed problems [15, 33]. This is because they violate the nullspace condition [15] introduced in Section 2.3. To remedy this issue, the coefficients of the highest order moments are adjusted to obtain well-posedness [8, 15]. The resulting boundary condition operator in x-direction for the linearized G13 equations ($N_d = 2$) reads as

$$B_x = \left[\begin{array}{c|cc|c|ccc|cc} -\tilde{\chi} & 1 & 0 & \frac{\tilde{\chi}}{3} & -\frac{4\tilde{\chi}}{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\tilde{\chi} & 0 & 0 & 1 & 0 & 0 & \frac{2\tilde{\chi}}{5} \\ -\frac{\tilde{\chi}}{2} & 0 & 0 & -\frac{7\tilde{\chi}}{6} & \frac{2\tilde{\chi}}{5} & 0 & 0 & 1 & 0 \end{array}\right], \tag{2.70}$$

(2.71)

where the vertical delimiters indicate the structure of the tensor variables as done in Figure 2.1. The next highest model level boundary condition operator is

$$B_x = \left[\begin{array}{c|cc|c|ccc|ccc|cc|ccc} -\tilde{\chi} & 1 & 0 & \frac{\tilde{\chi}}{3} & -\frac{\tilde{\chi}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2\tilde{\chi}}{15} & -\frac{2\tilde{\chi}}{7} & 0 & 0 \\ 0 & 0 & -\tilde{\chi} & 0 & 0 & 1 & 0 & 0 & \frac{\tilde{\chi}}{5} & 0 & -\frac{6\tilde{\chi}}{7} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{\tilde{\chi}}{2} & 0 & 0 & -\frac{7\tilde{\chi}}{6} & \frac{\tilde{\chi}}{4} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \frac{3\tilde{\chi}}{5} & -\frac{6\tilde{\chi}}{7} & 0 & 0 \\ \hline \frac{2\tilde{\chi}}{5} & 0 & 0 & -\frac{2\tilde{\chi}}{5} & -\frac{6\tilde{\chi}}{5} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{4\tilde{\chi}}{75} & \frac{2\tilde{\chi}}{5} & 0 & 0 \\ -\frac{\tilde{\chi}}{5} & 0 & 0 & \frac{\tilde{\chi}}{5} & \frac{\tilde{\chi}}{10} & 0 & -\tilde{\chi} & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{2\tilde{\chi}}{75} & -\frac{2\tilde{\chi}}{35} & 0 & \frac{2\tilde{\chi}}{7} \\ 0 & 0 & -\frac{\tilde{\chi}}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{11\tilde{\chi}}{10} & 0 & \frac{3\tilde{\chi}}{7} & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{2.72}$$

By comparing coefficients it can be seen, that the coefficients corresponding to the highest moments in each boundary condition of the smaller system do not match their respective counterparts in the larger system's boundary condition operator. This is due to the aforementioned adjustment to obtain well-posedness. However, this entails a breaking of the hierarchical structure, since the boundary condition operators are not fully nested anymore. This may introduce problems when refining from one model level to another, due to mismatching boundary conditions. An investigation of this potential issue is left for future work.

3. Discontinuous Galerkin formulation

To compute approximate solutions of the model problems presented in the previous chapter, an appropriate numerical scheme is required. In this chapter a discontinuous Galerkin (DG) finite element scheme along the lines of [34, 35, 36, 37] is formulated. DG methods are capable of higher order discretizations, allow for unstructured meshes with complex geometries and are ideal for adaptivity [11, 38]. This makes a discontinuous Galerkin scheme well suited for the model problems, as mentioned in Chapter 1. In particular, the presented scheme is readily extended to handle discretization and model adaptivity, which is discussed in Section 3.5. Note that often a major part of a DG formulation is the treatment of discontinuities in the solutions of nonlinear equations, see for example [39]. Since only linear problems are considered in this work, methods for shock-capturing and limiting are not required and thus omitted.

The present DG formulation is a method of lines method, in which first space is discretized by means of a discontinuous Galerkin approach and then the resulting system of ordinary differential equations (ODE) is discretized in time [36]. Therefore, this chapter with deals first with the spatial discretization and then the temporal one. Additionally, special attention is given to the incorporation of boundary conditions into the scheme and the representation of the geometry of domains with curved boundaries. The chapter ends with some details on the implementation.

3.1. Spatial discretization

The key ingredients of a discontinuous Galerkin formulation are the Galerkin weak form, discontinuous approximations and the numerical fluxes [11]. In the following, we first derive the Galerkin weak form in Section 3.1.1 and transform it into a form more amenable to computation in Sections 3.1.2 and 3.1.3. Thereafter, we introduce the concrete choice of DG basis functions and numerical fluxes in Section 3.1.4 and Section 3.1.6 respectively. Modern higher order DG codes exploit the tensor product structure of the DG basis functions to increase computational efficiency and throughput, see for example [40, 41]. Such a basis formulations carries nontrivial implications on the design of the assembly algorithms and their implementation and is therefore briefly discussed in Section 3.1.5.

3.1.1. Discontinuous Galerkin weak formulation

Starting point of the DG finite element method is the weak form of the problem, which is obtained by multiplying the conservation law (2.1a) with a sufficiently smooth test function $\varphi \in V \subseteq L^2(\mathbb{R}^d)$, integrating over the domain and performing integration by parts, yielding

$$\int_\Omega \partial_t u(\mathbf{x},t)\varphi(\mathbf{x})\,d\mathbf{x} = \int_\Omega F(u(\mathbf{x},t)) \cdot \nabla\varphi(\mathbf{x})\,d\mathbf{x} \tag{3.1a}$$

$$- \int_{\partial\Omega} F(u(\mathbf{x},t)) \cdot \mathbf{n}\varphi(\mathbf{x})\,d\Gamma \tag{3.1b}$$

$$+ \int_\Omega \left(Pu(\mathbf{x},t) + S(\mathbf{x})\right)\varphi(\mathbf{x})\,d\mathbf{x} \quad \forall\varphi \in V. \tag{3.1c}$$

Note that for systems of equations the test function is formally vector-valued. Selecting a standard basis for the test functions then extracts each individual equation of the system of equations from the weak form. In the present formulation all test function components are considered to be from the same function space. To keep the notation simple, the solution variables are considered to be embedded inside the solution, which then acts as a scalar. All arithmetic operations performed with the solution u and the integration of resulting terms are performed component-wise. The vector-valued production and source terms are treated in a similar fashion and the dot products with the matrix-valued flux F evaluate as matrix-vector multiplications. Consequently, we interpret the weak form (3.1) as a vector-valued weak form for each component of the solution vector and corresponding conservation law.

The computational domain Ω is partitioned into non-overlapping quadrilateral elements Ω^e, such that $\bigcup_e \Omega^e = \Omega$ and $\Omega^{e_1} \cap \Omega^{e_2} = \emptyset$ for $e_1 \neq e_2$. Following the notation of [42], the elements are enumerated by e with $1 \leq e \leq n_{el}$, where $n_{el} \in \mathbb{N}$ is the number of elements. The collection of all elements constitutes an unstructured quadrilateral mesh, which is the basis of the spatial discretization of the problem 2.1.1.

The weak form (3.1) is discretized by means of a Galerkin finite element method by selecting a finite dimensional subspace $V_h^p \subset V$, spanned by finitely many test functions $\varphi_i \in V_h^p$ and trial functions $\psi_j \in V_h^p$. A discontinuous Galerkin method is obtained by choosing a piecewise polynomial function space [34, 43]:

$$V_h^p := \{f \in L^2(\mathbb{R}^d) : f|_{\Omega^e} \in \Pi_p\}, \tag{3.2}$$

where Π_p is the space of polynomials of degree $\leq p$. Consequently, the solution is allowed to jump at the interfaces between elements and is represented by a polynomial within each element. The test and trial functions are split over the elements and we write φ_i^e and ψ_j^e to denote the element-wise test and trial functions. The approximate

solution $u_h(\cdot, t) \in V_h^p$ is then given by a basis expansion, see [43]:

$$u_h(\mathbf{x}, t) \coloneqq \sum_{j=1}^{k} \psi_j^e(\mathbf{x}) u_j^e(t), \quad \mathbf{x} \in \Omega^e, \tag{3.3}$$

where the time-dependent polynomial coefficients $u_j^e(t)$ are the degrees of freedom of the discretization. The index $j \in \{1, \dots, k\}$ now enumerates all $k = (p+1)^d$ basis functions of a given element. For example, in one dimension we have $1 \leq j \leq p+1$. The concrete form of the DG basis functions ψ and φ is elaborated in Section 3.1.4. Note that for vector-valued solutions the coefficients are implied to be vector-valued themselves, i .e . $u_j^e = (u_\nu)_j^e$, where $\nu \in \{1, \dots, n_v\}$.

The Galerkin weak form is derived by restricting the test functions of the weak form (3.1) to the finite dimensional subspace and replacing the solution with the approximation (3.3). Thereafter, the integrals are split into element contributions, yielding the Galerkin element weak form

$$\sum_{j=1}^{k} \partial_t u_j^e \int_{\Omega^e} \psi_j^e \varphi_i^e \, d\mathbf{x} = \int_{\Omega^e} F(u_h^e) \cdot \nabla \varphi_i^e \, d\mathbf{x} \tag{3.4a}$$

$$- \int_{\partial\Omega^e} F_n^*(u_h^-, u_h^+) \varphi_i^e \, d\Gamma \tag{3.4b}$$

$$+ \int_{\Omega^e} \left(P u_h^e + S\right) \varphi_i^e \, d\mathbf{x} \quad \forall i \in \{1, \dots, k\} \tag{3.4c}$$

which constitute $k n_v$ equations for the $k n_v$ element degrees of freedom. Note that the arguments are suppressed for clarity. The element interior solution is denoted by $u_h^e = u_h|_\Omega^e$. At the element boundary $\partial\Omega^e$, the solution is discontinuous. For any point on an interface between two elements, let u_h^- and u_h^+ respectively be the solution values of the element interior solution and the neighboring element's solution at that point. The connection between different elements is then established by means of numerical fluxes [44]. The computation of the normal direction flux through a suitable two-point numerical flux function F_n^* is therefore a crucial component the DG scheme, as it links the otherwise discontinuous and independent element solutions together.

3.1.2. Reference transformation and quadrature

The basis functions φ_i^e and ψ_j^e and the integration domains in the element weak form (3.4) depend on the element e. This makes the weak form tricky to compute in practice, especially for complex element geometries. Therefore, as is standard in finite element methods, see for example [42], coordinate transformations and quadrature rules are utilized to make the formulation more amenable to numerical computation.

In the following only the two-dimensional case ($d = 2$) is considered. A reduction to the one-dimensional case is straightforward. Let $\boldsymbol{\xi} = [\xi_1, \xi_2]^T = [\xi, \eta]^T$ be the coordinate of the reference space containing the reference element domain $\hat{\Omega}^e = [-1, 1]^d$. We define the diffeomorphism $\Phi^e : \hat{\Omega}^e \to \Omega^e$ as a mapping of the reference element onto the element e in physical space. Note that the chosen form of the mapping determines the accuracy of the representation of the physical geometry by the mesh and numerical scheme. The form of the mapping Φ^e determines what geometries can be represented accurately by the mesh and numerical scheme. Specification of suitable mapping functions is deferred to Section 3.3. A sketch of the setup is depicted in Figure 3.1. Note in particular the counterclockwise enumeration of the elements corners and edges, also called interfaces, which is required to correctly link neighboring elements interfaces together.

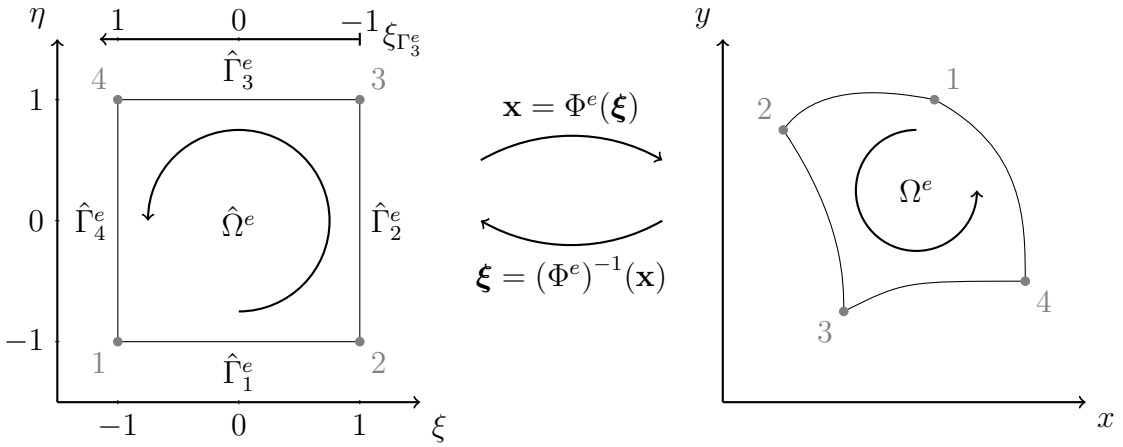


Figure 3.1. Sketch of the mapping between an element in the computational domain (right) and the two-dimensional reference element $\hat{\Omega}^e = [-1, 1]^2$ (left). The corner vertices and edges of the element are enumerated in counter clockwise order. For every reference edge Γ^e_i, $i \in \{1, 2, 3, 4\}$ a local coordinate system aligned with the order of its vertices is defined. For example, $\xi_{\Gamma^e_3}$ denotes the local reference coordinate of the third edge.

Applying a coordinate transformation to change the domain of integration of the left-hand side term of the weak form (3.4) yields:

$$M^e_{ij} := \int_{\Omega^e} \psi^e_j \varphi^e_i \, d\mathbf{x} = \int_{\hat{\Omega}^e} \hat{\psi}_j(\boldsymbol{\xi}) \hat{\varphi}_i(\boldsymbol{\xi}) \left|\det J^e(\boldsymbol{\xi})\right| \, d\boldsymbol{\xi}, \tag{3.5}$$

where $J^e := D\Phi^e$ is the Jacobian of the geometry mapping. The matrix M^e is called the (element) mass matrix. The basis functions are defined on the reference element, which is the same for all elements, meaning that expressions for $\hat{\psi}$ and $\hat{\varphi}$ are explicitly available. Their physical space counterparts are evaluated by inverting the geometry mapping. However, this is not required in the course of the computation, as all operations are performed on the reference element. Since all elements share the same

set of basis functions, the superscript $(\cdot)^e$ is dropped. Note that for a p-adaptive scheme, the set of basis functions may vary from element to element. In that case, the element weak form is evaluated with the set of basis functions corresponding to the element's polynomial degree p.

In general, the integrals cannot be evaluated analytically and are thus approximated by a quadrature rule. Let $\boldsymbol{\xi}_q$ and w_q be the quadrature nodes and weights on the reference element. The element mass matrix is then computed as

$$M_{ij}^e \approx \sum_{q=1}^{n_q} \hat{\psi}_j(\boldsymbol{\xi}_q)\hat{\varphi}_i(\boldsymbol{\xi}_q) \left|\det J(\boldsymbol{\xi}_q)\right| w_q = \sum_{q=1}^{n_q} \hat{\psi}_{jq}\hat{\varphi}_{iq} \left|\det J_q^e\right| w_q, \tag{3.6}$$

where $n_q \in \mathbb{N}$ is the number of quadrature nodes and the index $q = \{1, \dots, n_q\}$ is used to indicate evaluation at the corresponding quadrature node. The production and source term volume integral (3.4c) is transformed in a similar manner, yielding

$$V_i^e :\approx \sum_{q=1}^{n_q} \left(P u_{h,q}^e + S(\Phi^e(\boldsymbol{\xi}_q))\right) \hat{\varphi}_i(\boldsymbol{\xi}_q) \left|\det J_q^e\right| w_q, \tag{3.7}$$

where $u_{h,q}^e$ is the approximate solution at the quadrature nodes, obtained from evaluating the basis expansion (3.3):

$$u_{h,q}^e = \sum_{j=1}^{k} \hat{\psi}_j(\boldsymbol{\xi}_q) u_j^e. \tag{3.8}$$

For the transformation of the volume flux term on the right-hand side of Equation (3.4a), we note the chain rule of differentiation

$$\frac{\partial \hat{\varphi}_i}{\partial x_j} = \frac{\partial \hat{\varphi}_i}{\partial \xi_k}\frac{\partial \xi_k}{\partial x_j} \implies \nabla_x \hat{\varphi}_i(\boldsymbol{\xi}) = (J^e)^{-T} \nabla_\xi \hat{\varphi}_i(\boldsymbol{\xi}), \tag{3.9}$$

where summation over the repeated index k is implied. Using the above, the volume flux term is then transformed to the reference element and the quadrature rule is applied:

$$\begin{aligned} G_i^e &:= \int_{\Omega^e} F(u_h^e(\mathbf{x}, t)) \cdot \nabla \varphi_i^e \, d\mathbf{x} && (3.10)\\ &= \int_{\hat{\Omega}^e} F(u_h^e(\boldsymbol{\xi}, t)) \cdot \nabla_x \hat{\varphi}_i \left|\det J^e(\boldsymbol{\xi})\right| \, d\boldsymbol{\xi} && (3.11)\\ &= \int_{\hat{\Omega}^e} (J^e)^{-1} F(u_h^e(\boldsymbol{\xi}, t)) \cdot \nabla \hat{\varphi}_i \left|\det J^e(\boldsymbol{\xi})\right| \, d\boldsymbol{\xi} && (3.12)\\ &\approx \sum_{q=1}^{n_q} \left(w_q \left|\det J_q^e\right| (J_q^e)^{-1} F_q^e\right) \cdot \nabla \hat{\varphi}_{iq}. && (3.13) \end{aligned}$$

In the last step the flux matrix at the quadrature node q is abbreviated as $F_q^e = F(u_{h,q}^e) \in \mathbb{R}^{n_v \times d}$ and the summand is separated into a solution and geometry dependent integrand term and the test function derivative. In practice, guided by [41], the sum is evaluated by first preparing the integrand for all quadrature points, followed by an integration step that evaluates the product with the test functions.

To compute the surface contribution (3.4b) to the element weak form, a representation of the element edge geometry is required. Note that every geometric edge is associated with an interface, either between two elements or with the boundary. The reference element boundary $\partial\hat{\Omega}$ is divided into its four interfaces $\hat{\Gamma}_l^e$ with $l \in \{1,2,3,4\}$, enumerated in counterclockwise order, as seen in Figure 3.1. The mapping between each physical edge and the corresponding reference interface can be expressed in terms of the element geometry mapping. The interface geometry mapping

$$\tilde{\Phi}^{e,l}(\xi) := \Phi^e(\boldsymbol{\xi}|_{\hat{\Gamma}_l^e}(\xi)), \quad \xi \in [-1,1], \tag{3.14}$$

for the interface l of the element e is defined by means of the coordinate transformation $\boldsymbol{\xi}|_{\hat{\Gamma}_l^e}(\xi)$, which maps the interface-local reference coordinate to the two-dimensional element reference coordinates. For example, the third interface reference coordinate, shown in Figure 3.1, yields the transformation $\boldsymbol{\xi}|_{\hat{\Gamma}_l^e}(\xi_{\Gamma_3^e}) = [-\xi_{\Gamma_3^e}, 1]^T$. Since each edge is a one-dimensional curve immersed in two-dimensional reference space, we introduce the metric tensor g, given by [45]:

$$g^{e,l}(\xi) = \left(D\tilde{\Phi}^{e,l}(\xi)\right)^T D\tilde{\Phi}^{e,l}(\xi), \tag{3.15}$$

to be used for the coordinate transformation of surface integrals.

Utilizing the above, the surface integral (3.4b) of the weak form is split over the interfaces of the element, transformed to the reference interface and approximated by a quadrature rule, yielding

$$I_i^e := \sum_{l=1}^{4} \int_{\Gamma_l^e} -F_n^*(u_h^-, u_h^+)\varphi_i^e \, d\Gamma \tag{3.16}$$

$$= \sum_{l=1}^{4} \int_{\hat{\Gamma}_l^e} -F_n^*(u_h^-, u_h^+)(\xi)\hat{\varphi}_i(\xi)\sqrt{\det g^{e,l}(\xi)}\, d\xi \tag{3.17}$$

$$\approx \sum_{l=1}^{4}\sum_{q=1}^{n_q} -F_n^*(u_{h,q}^-, u_{h,q}^+)\hat{\varphi}_{iq}\sqrt{\det g_q^{e,l}}\, w_q, \tag{3.18}$$

where n_q denotes the number of quadrature nodes on each interface. The basis functions on the reference interface are obtained by the mapping of the reference coordinates to the interface-local coordinate system described above. The numerical flux F_n^* evaluated at each quadrature point implicitly depends on the outward normal direction $\mathbf{n}(\xi) = [-t_1, t_2]^T/\|t\|$, which is obtained from the components of the tangent vector $\mathbf{t}(\xi) = \partial_\xi \tilde{\Phi}^{e,l}(\xi)$ along the curved interface.

The internal solution $u_{h,q}^-$ evaluated at the interface quadrature nodes is readily obtained by a basis evaluation of the current element's basis and solution coefficients. In contrast, the external solution $u_{h,q}^+$ needs to be either computed from the boundary conditions as $u_q^+ = u_{bc}(u_q^-)$, elaborated on in Section 3.2, or provided by the

corresponding neighboring element. For a conforming interface between two elements employing the same quadrature rule, there is a one-to-one correspondence between the quadrature nodes of the two sides of the interface. Due to the counterclockwise ordering and use of interface-local coordinate systems, the quadrature nodes on one side are always in reverse order with respect to the quadrature nodes on the other side. Therefore, in practice, when iterating over the quadrature nodes of an interface, one side is simply iterated in reverse to pair up the nodes correctly. It is for this reason that a counterclockwise ordering and interface-local coordinate systems are employed.

3.1.3. Residual assembly

Recombining all the transformed terms derived in the previous section, the element weak form (3.4) reads as

$$M^e \dot{u}^e = G_i^e(u^e) + I_i^e(u^e) + V_i^e(u^e), \tag{3.19}$$

where $(u^e)_j = u_j^e$, the time derivative of the coefficients is abbreviated as $\dot{u}^e = \partial_t u^e$ and is isolated by inverting the mass matrix to the right-hand side, yielding the element semidiscretization

$$\dot{u}^e = (M^e)^{-1} \left(G_i^e(u^e) + I_i^e(u^e) + V_i^e(u^e)\right) =: R_i^e(u^e), \tag{3.20}$$

also called the element residual herein, since it measures the deviation from the steady state $\dot{u}^e = 0$. Finally, the splitting of the weak form (3.1) is reversed by collecting the degrees of freedoms of all elements into a single vector $\mathbf{u}$ and the element residuals into a single residual operator R, resulting in

$$\dot{\mathbf{u}} = R(\mathbf{u}), \tag{3.21}$$

which is a system of ordinary differential equations for all solution coefficients. As the evaluation of the residual (3.21) is the core part of the numerical scheme, we note some practical considerations in the following.

A fundamental optimization is to precompute the evaluation of the basis functions at the quadrature nodes and store the results in shape function matrices, see [46]:

$$(S^\varphi)_{qi} \coloneqq \hat{\varphi}_i(\xi_q), \quad (S^\psi)_{qj} \coloneqq \hat{\psi}_j(\xi_q). \tag{3.22}$$

Similarly, interface quadrature node shape function matrices can be defined and precomputed. Utilizing the shape function matrices, many of the operations to evaluate the element residuals can be expressed in terms of basic linear algebra

subprograms (BLAS). For example, the interpolation of the solution to the element quadrature points is expressed as a matrix-vector product

$$u^e_{h,q} = \left(S^\psi u^e\right)_q, \tag{3.23}$$

which is handled efficiently by any BLAS implementation. Furthermore, when the number of basis functions is equal to the number of quadrature points, the evaluation of a matrix-vector product with the mass matrix inverse can be computed via [46]:

$$(M^e)^{-1}u^e = \left((S^\varphi)^T W^e S^\psi\right)^{-1} u^e = (S^\psi)^{-1}(W^e)^{-1}(S^\varphi)^{-T}u^e, \tag{3.24}$$

where $(W^e)_{qq} \coloneqq w_q|\det J^e_q|$ is a diagonal matrix and the inverse shape function matrices are precomputed. Clearly, instead of explicitly inverting the mass matrix, it is cheaper to invert only the diagonal matrix W^e and then evaluate (3.24) for all solution variables to compute the element residual (3.20). An alternative to the consistent mass matrix approach is to approximate the mass matrix, for example by row-sum lumping. However, this may lead to a degradation of order of accuracy, see for example [45, 47]. For this reason we primarily use the consistent mass matrix formulation (3.24).

Lastly, note that that the numerical flux is conservative and therefore the computed normal fluxes of an interface's left and right elements are related by $F^*_n(u^-_{h,q}, u^+_{h,q}) = -F^*_n(u^+_{h,q}, u^-_{h,q})$. Consequently, at every quadrature node of every interface the normal flux is only computed once and then stored for use during the left and right elements weak form evaluation. This necessitates that the interpolation of the solution data to the interface quadrature nodes is done beforehand. A rough sketch of the resulting procedure is shown in Algorithm 1. Since neighboring elements are only coupled through the numerical fluxes at the interfaces, the extension of the assembly algorithm to handle adaptivity is performed by modifying step (ii) of the procedure, which is done in Section 3.5. As long as the implemented data structures support varying number of basis functions, quadrature points and number of equations, the required modifications to the other steps are straightforward and only restricted to the implementation. Conceptually they remain the same as depicted in Algorithm 1.

Algorithm 1 Assembly of the global DG residual (3.21)

(i) loop over elements: compute $u_{h,q}|_{\hat{\Gamma}^e_l}$ for $q \in \{1, \dots, n_q\}$ and $l \in \{1, \dots, 4\}$

(ii) loop over interfaces: compute $F^*_n(u^{left}_{h,q}, u^{right}_{h,q})$ for $q \in \{1, \dots, n_q\}$

(iii) loop over elements: compute the element residual (3.20)

3.1.4. DG basis functions and quadrature rules

In the following the form of the trial and test basis functions and the quadrature rule is clarified. Note that the basis functions are polynomials defined on the reference element, which for quadrilaterals is a square. Therefore, given a set of one-dimensional basis functions, defined on the reference interval $\xi \in [-1, 1]$, the two-dimensional basis functions are constructed by means of a tensor product:

$$\hat{\varphi}_i(\boldsymbol{\xi}) \coloneqq \hat{\varphi}_{i_1}(\xi_1)\hat{\varphi}_{i_2}(\xi_2), \quad i = i_2 + i_1(p+1), \quad i_1, i_2 \in \{1, \dots, p+1\}, \tag{3.25a}$$

$$\hat{\psi}_j(\boldsymbol{\xi}) \coloneqq \hat{\psi}_{j_1}(\xi_1)\hat{\psi}_{j_2}(\xi_2), \quad j = j_2 + j_1(p+1), \quad j_1, j_2 \in \{1, \dots, p+1\}, \tag{3.25b}$$

where the indices i and j are constructed such that the order of the basis functions matches the order resulting from the tensor product, e.g. $\hat{\varphi}_i(\boldsymbol{\xi}) = (\hat{\varphi}(\xi_1) \otimes \hat{\varphi}(\xi_2))_i$, meaning that the index of the second dimension varies the fastest. Further, note that we distinguish between the one- and two-dimensional basis functions through their argument. The maximum polynomial degree p of the basis determines the spatial convergence order of the numerical scheme. Analogously, the quadrature nodes and weights are also constructed by means of a one-dimensional quadrature rule:

$$\boldsymbol{\xi}_q \coloneqq (\xi_{q_1}, \xi_{q_2}), \quad w_q \coloneqq w_{q_1} w_{q_2}, \quad q = q_2 + q_1 n_q, \quad q_1, q_2 \in \{1, \dots, n_q\} \tag{3.26}$$

where n_q now denotes the number of one-dimensional quadrature points. Due to the tensor product nature of the basis, it suffices to specify only the one-dimensional basis functions and quadrature rule. Two typical choices are a modal basis based on Legendre polynomials, see [43], or a nodal basis based on Lagrange polynomials, see [48].

The Legendre polynomials are commonly defined by the recursion [48]

$$L_{k+1}(\xi) = \frac{2k+1}{k+1}\xi L_k(\xi) - \frac{k}{k+1}L_{k-1}(\xi), \quad \text{with} \quad L_0(\xi) = 1,\ L_1(\xi) = \xi \tag{3.27}$$

and subsequently normalized to obtain orthogonal basis functions that satisfy

$$\int_{\hat{\Omega}^e} \hat{\psi}_j(\xi)\hat{\varphi}_i(\xi)\, d\xi = \delta_{ij}. \tag{3.28}$$

Following [43], we normalize the trial function with respect to L_∞, i.e. $\hat{\psi}_j = L_j$, meaning that the test function $\hat{\varphi}_i = 0.5(2p+1)L_i$ is the L_1 normalized dual function. Note that this is the default basis that is chosen for the simulations presented in this work. Alternatively, in practice a normalization with respect to L_2 is often performed, yielding coinciding test and trial functions [43]. To complete the modal basis, a sufficiently accurate Gauss-Legendre quadrature rule is selected, such that the quadrature error is of the same or higher order as the order of the numerical

scheme [43]. Since the mass matrix inverse formula (3.24) relies on square shape function matrices, we choose $n_q = p + 1$, which also results in an exact integration of the mass matrix integrand for affine element geometries.

A nodal DG basis is obtained by utilizing Lagrange polynomials, defined as

$$\ell_k(\xi) = \prod_{i=1, i\neq k}^{p+1} \frac{\xi - \xi_i}{\xi_k - \xi_i}, \tag{3.29}$$

where ξ_k with $k \in \{1, \dots, p+1\}$ are the interpolation nodes, as test and trial functions, i.e. $\hat{\varphi}_i = \hat{\psi}_i = \ell_i$. The nodes are selected to be the nodes of the employed quadrature rule, leading to a collocation approximation [12, 48]. This not only eliminates the need for the interpolation of the basis to the quadrature nodes, but also yields a diagonal mass matrix due to the discrete orthogonality of the basis functions [12]. A popular choice of nodes are the Gauss-Lobatto quadrature nodes, see for example [12, 14, 48].

Preliminary analysis reveals that in cases where two edges of a single element degenerate into a single curve, the nodal DG scheme may develop spurious oscillations inside of the element. While this can be remedied by avoiding too coarse meshes, herein the issue is avoided by selecting a modal basis. A detailed analysis of the influence of the choice of basis on the approximation quality of the present DG scheme applied to moment equations in curved domains is left for future work.

3.1.5. Tensor product formulation

The tensor product structure of the DG basis presented in the previous section allows for the evaluation of the element weak form integrals by means of sum factorization [40, 49]. For high-order bases this significantly reduces the computational cost [41] and is a common optimization present in modern DG codes, see for example [13, 50]. While our implementation of the presented DG scheme is not focused on high performance, we consider the tensor product formulation an important enough property for high-order schemes to be included in the design of the solver. In the following, we summarize the employed tensor product formulation, which is based on [41].

For a tensor product basis, the interpolation of the solution values to the quadrature nodes, given in Equation (3.23), can be factorized as

$$u_h^e = S^\psi u^e = \sum_{i_1=1}^{k} \sum_{i_2=1}^{k} \hat{\psi}_{i_1}(\xi_{q_1}) \hat{\psi}_{i_2}(\xi_{q_2}) u_{ij}^e = \left(S_1^\psi \otimes S_2^\psi \right) u^e, \tag{3.30}$$

where S_i denotes the shape function matrix of the one-dimensional basis in reference direction i. Since the same basis and quadrature rule is used for all directions, we

have $S_1 = S_2$. Furthermore, the two-dimensional shape function matrix $S = S_1 \otimes S_2$ can be expressed as a tensor product of the one-dimensional matrices. Crucially, the tensor product in Equation (3.30) does not need to be evaluated directly, but can be efficiently computed by means of the so-called "vec trick", which we define as [46, 51, 52]:

Definition 3.1.1 (Roth's column lemma; "Vec Trick")**.** For the matrices $S_2 \in \mathbb{R}^{a\times b}$, $U \in \mathbb{R}^{b\times c}$ and $S_1 \in \mathbb{R}^{c\times d}$ it holds, that

$$(S_1^T \otimes S_2)\text{vec}(U) = \text{vec}(S_2 U S_1), \tag{3.31}$$

where $\text{vec}(\cdot)$ converts a matrix to a vector by stacking the columns of the matrix. In particular, for the vector $u = \text{vec}(U) \in \mathbb{R}^{bc}$ with $\text{mat}(u) = U$ we have

$$(S_1 \otimes S_2)u = \text{vec}(S_2 \text{mat}(u) S_1^T). \tag{3.32}$$

Consequently, it suffices to store only the one-dimensional shape function matrices S_1^ψ and S_1^φ. Moreover, the two-dimensional shape function matrix is never constructed explicitly and all matrix-vector products with it are evaluated by means of Equation (3.32). Note that in practice, each entry of the coefficients u_j^e and quadrature point solution u_q^e is a vector of n_v solution variables. Therefore, the current implementation of Equation (3.32) also carries all solution variables as innermost dimension.

Using the tensor product formulation of the basis, the application of the mass matrix inverse (3.24) is written as [46]:

$$(M^e)^{-1}u^e = \left((S_1^\psi)^{-1} \otimes (S_1^\psi)^{-1}\right)(W^e)^{-1}\left((S_1^\varphi)^{-T} \otimes (S_1^\varphi)^{-T}\right)u^e, \tag{3.33}$$

assuming the number of basis functions $p+1$ is equal to the number of quadrature points n_q. Furthermore, the volume flux integral (3.13) is computed via [41]

$$G^e = \begin{bmatrix} D_1^\varphi \otimes S_2^\varphi \\ S_1^\varphi \otimes D_2^\varphi \end{bmatrix}^T \mathbb{I} = [D_1^\varphi \otimes S_2^\varphi]\,\mathbb{I}_1 + [S_1^\varphi \otimes D_2^\varphi]\,\mathbb{I}_2, \tag{3.34}$$

where the integrand $\mathbb{I} = [\mathbb{I}_1, \mathbb{I}_2]^T$ is expressed as a vector and (D_i^φ) are precomputed one-dimensional test function derivatives in direction i, given herein by $(D_1^\varphi)_{qj} = (D_2^\varphi)_{qj} = \partial_\xi \hat{\varphi}_j(\xi_q)$. Lastly, for the interpolation of the solution to the interfaces we introduce the $1 \times k = p+1$ matrix S_f, which evaluates the basis functions at the appropriate reference interval boundaries. For example, using $(S_f)_{1j} = \hat{\psi}_j(-1)$ the solution is interpolated to the first interface's quadrature nodes by means of [41]:

$$S_1\,[I_1 \otimes S_f]\,u^e = S_1 \text{vec}(S_f \text{mat}(u^e) I^T). \tag{3.35}$$

In the evaluation of Equation (3.35) the identity matrix is simply omitted and the $\text{vec}(\cdot)$ operation reinterprets the input row vector as a column vector. The integration step, i.e. the multiplication with the test function and subsequent summation, is performed in a transposed manner.

3.1.6. Numerical fluxes

The DG scheme relies on the selection of a suitable numerical flux function F_n^*. In general, the numerical flux is an exact or approximate Riemann solver [35, 39]. Furthermore, guided by [35, 36, 39] a suitable numerical flux is a two-point Lipschitz continuous function that is conservative, consistent and monotone. A conservative numerical flux satisfies [35]

$$F_n^*(u^-, u^+) = -F_{-n}^*(u^+, u^-), \tag{3.36}$$

where F_{-n}^* denotes the numerical flux in the opposite direction as F_n^*. This yields in conservation of the solution quantities of the conservation law by the semidiscretization. Consistency is defined as [36]

$$F_n^*(u, u) = F_n(u) \coloneqq F(u) \cdot \mathbf{n}, \tag{3.37}$$

where $F_n(u)$ is the analytical flux in the direction $\mathbf{n}$. Thus, as the discretization is refined and the solution values $u^\pm$ get closer together, the numerical flux approaches the analytical one. Lastly, monotonicity means that $F_n^*(u^-, u^+)$ is non-decreasing in u^- and non-increasing in u^+ [36], which is motivated by the stability of monotone first order schemes [39].

A common choice of numerical flux is the local Lax-Friedrichs (LLF) flux, given by [17, 35, 53]:

$$F_n^*(u^-, u^+) = \frac{1}{2}\left[F_n(u^-) + F_n(u^+) - \lambda_{max}\left(u^+ - u^-\right)\right], \tag{3.38}$$

where λ_{max} is an estimate of the maximum wave speed, given by the maximum absolute eigenvalue of the flux Jacobian $A(u)$, for all u between u^- and u^+, see [17]. Note that for a convex flux the maximum wave speed is achieved at one or both of the bounding states u^- and u^+ [43]. In the case of a linear rotationally invariant system of equations, the maximum wave speed is constant and can be computed as the maximum eigenvalue of the system matrix A_x.

Another fundamental numerical flux is the Godunov flux, see [17], which is based on the exact solution of a Riemann problem at the interface. For a constant coefficient problem, the Godunov flux is constructed as follows: Let $A_n = R\Lambda L$ be the eigendecomposition of the normal direction flux Jacobian with eigenvectors r_i and eigenvalues λ_i. The characteristic variables of the left and right states are computed as $\alpha^\pm = Lu^\pm$. The Godunov flux is then given by [17]

$$F_n^*(u^-, u^+) = F_n\left(\sum_{i:\lambda_i<0} r_i\alpha_i^- + \sum_{i:\lambda_i>0} r_i\alpha_i^+\right). \tag{3.39}$$

Note that waves $r_i\alpha_i$ corresponding to zero eigenvalues are in the nullspace of the matrix A_n and therefore do not contribute to the flux. Further, note that in the linear case the Godunov flux coincides with the Roe flux and constitutes an upwind scheme for the characteristic variables, which for a scalar equation is also called the Courant-Isaacson-Rees flux, see [17, 43].

In practice, we use the simple LLF flux for most simulations, as it was shown to yield satisfactory results in [26], and use the Godunov flux only as a point of comparison.

3.2. Implementation of boundary conditions

For interfaces that lie on the boundary of the computational domain, the external solution state $u^+ \equiv u_{bc}(u^-)$ at each quadrature point needs to be computed from the internal state u^- such that it satisfies the boundary conditions $B_n u_{bc} = g$ (2.1b), with B_n being the boundary condition matrix in the outward facing wall normal direction. In doing so, special care needs to be taken, so that the resulting numerical boundary conditions yield a stable scheme. Even for well-posed analytical boundary conditions, their realization in the numerical scheme by an improper formula for u_{bc} may lead to an unstable method, see for example [54]. While discrete energy stability may be verified for certain families of discretizations, see [23, 55], the analysis of energy stability of the present scheme is out of scope of this work. Instead, we rely on conventional boundary condition implementations that mimic the key properties leading to well-posedness of the continuous problem, discussed in Section 2.3 and have been shown to yield good results in practice, see [5, 26].

3.2.1. Characteristic boundary conditions

The concept of characteristic boundary conditions, mentioned in Section 2.3.1, carries over to the discrete case, where the boundary state u_{bc} is constructed by specifying the incoming waves by means of a known external solution state u_{ext}, yielding characteristic boundary conditions [26]:

$$u_{bc} = R\Lambda^- L u_{\text{ext}} + R\Lambda^+ L u^-. \tag{3.40}$$

where R and $L = R^{-1}$ are obtained from the characteristic decomposition of the normal direction flux Jacobian $A_n = R\Lambda L$. The eigenvalues $\lambda_i = \Lambda_{ii}$ are used to split the solution into outgoing and incoming modes using the diagonal indicator matrices

$$\Lambda^+_{ii} := \begin{cases} 1 & \lambda_i \geq 0, \\ 0 & \lambda_i < 0, \end{cases} \qquad \Lambda^-_{ii} := \begin{cases} 0 & \lambda_i \geq 0, \\ 1 & \lambda_i < 0, \end{cases} \tag{3.41}$$

where the waves corresponding to zero wave speeds are attributed to the internal solution to remain consistent with the nullspace condition derived in Section 2.3. Equation (3.40) then gives the boundary state by taking the internal solution and replacing the incoming wave modes with the corresponding modes of the provided external solution.

The above method requires that either the exact solution at the boundary or an explicit expression for u_{ext} in terms of u^- is known. However, for the model problems considered herein, the boundary conditions are given in terms of the boundary condition operator B_n and data g. Thus, we follow the procedure given in [5] to derive a general expression for the boundary state u_{bc}. We define a selector matrix X and transformed variables v, such that

$$X = [X_{bc} \,|\, X_{\text{rest}}] \in \mathbb{R}^{n_v \times n_v}, \quad X_{bc} \in \mathbb{R}^{n_v \times n_{bc}}, \quad v = Xu = [v_{bc} \,|\, v_{\text{rest}}]^T, \tag{3.42}$$

where, as defined in Chapter 2, n_v is the number of variables, meaning $u, v \in \mathbb{R}^{n_v}$ and n_{bc} is the number of boundary conditions. The selector matrix splits the state u into a set of boundary variables v_{bc} for which values are supplied via the boundary conditions and remaining variables v_{rest} onto which no conditions are imposed. Note that when choosing $X = R$ as the eigenvector matrix of A_n, the transformed variables v are the characteristic variables. An expression for the boundary variables is obtained by writing the boundary in terms of the transformed variables and rearranging:

$$B_n u = B_n X v = B_n X_{bc} v_{bc} + B_n X_{\text{rest}} v_{\text{rest}} = g \tag{3.43}$$

$$\implies v_{bc} = -\left(B_n X_{bc}\right)^{-1} \left(B_n X_{\text{rest}} v_{\text{rest}} - g\right), \tag{3.44}$$

where it is assumed that the product is $B_n X_{bc}$ invertible, which necessitates that X_{bc} has full rank and none of its columns are in the nullspace of B_n.

An expression for the difference between the internal and external states can then be derived [5]:

$$\begin{aligned}
u^- - u^+ &= X(v^- - v^+) && \text{(3.45a)}\\
&= X_{bc}(v_{bc}^- - v_{bc}^+) + X_{\text{rest}}(v_{\text{rest}}^- - v_{\text{rest}}^+) && \text{(3.45b)}\\
&= X_{bc}(v_{bc}^- + \left(B_n X_{bc}\right)^{-1} \left(B_n X_{\text{rest}} v_{\text{rest}} - g\right)) && \text{(3.45c)}\\
&= X_{bc} \left(B_n X_{bc}\right)^{-1} \left(B_n X_{\text{rest}} v_{\text{rest}}^- + B_n X_{bc} v_{bc}^- - g\right) && \text{(3.45d)}\\
&= X_{bc} \left(B_n X_{bc}\right)^{-1} \left(B_n u^- - g\right), && \text{(3.45e)}
\end{aligned}$$

where in the first step the decomposition of the selector matrix is applied. Thereafter, the external boundary variables v_{bc}^+ are replaced according to Equation (3.44) and the remaining transformed variables are extrapolated, meaning $v_{\text{rest}}^- = v_{\text{rest}}^+$. This enforces the boundary condition on the external transformed variables. Lastly, the internal

transformed variables v_{bc}^- are pulled inside the parenthesis, yielding the boundary conditions residual of the internal state. Consequently, the boundary condition state $u_{bc} = u^+$ can be computed from the internal state u^- as

$$u_{bc} = u^- - X_n(B_n u^- - g), \quad \text{where} \quad X_n \coloneqq X_{bc}\left(B_n X_{bc}\right)^{-1}, \tag{3.46}$$

which is the formulation used in the present numerical scheme to enforce boundary conditions. Presently, we choose $X_{bc} = \Lambda^- R$, which yields characteristic boundary conditions, since setting $u_{\text{ext}} = u_{bc}$ satisfies Equation (3.40). Intuitively, Equation (3.46) constructs u_{bc} by taking u^- and modifying only the incoming wave modes such that the boundary conditions hold.

Note that the boundary integrals of the element weak form could use the analytical flux, evaluated with u_{bc}, instead of the numerical flux. However, this may lead to stability issues [26]. Therefore, the boundary integrals are computed employing the numerical flux. Lastly, note that the present formulation yields weakly imposed boundary conditions [5, 23].

3.2.2. Rotation of symmetric 2D tensors

The evaluation of the boundary state (3.46) requires the calculation of X_n, which entails the eigendecomposition of the normal direction flux Jacobian A_n, at every boundary quadrature node for every evaluation of the residual. Both precomputing and recomputing X_n at every node becomes prohibitively expensive, in terms of memory or computational resources, for larger moment systems. For linear rotationally invariant systems of equations, the excessive cost can be avoided, by applying the boundary conditions in a normal-aligned coordinate system. In practice, the boundary state is computed by rotating the internal solution u^- into a fixed reference coordinate system, i.e. $\hat{\mathbf{n}} = [1, 0]^T$, calculating $u_{bc}^{(\hat{\mathbf{n}})}$ by means of Equation (3.46) with $B_n = B_{\hat{\mathbf{n}}}$ and $X_n = X_{\hat{\mathbf{n}}}$ and rotating the result back into the original coordinate system to obtain u_{bc}. The matrices $X_{\hat{\mathbf{n}}}$ and $B_{\hat{\mathbf{n}}}$ are constant and thus computed once for a given system of equations. While this approach eliminates the cost associated with computing X_n and B_n, it introduces two rotation steps into the boundary condition procedure. General moment systems can contain tensor variables with arbitrarily high rank depending on the selected moment theory. In particular, each successive ordered theory model, introduced in Section 2.4, contains a tensor variable of one rank higher than the previous model. For this reason, we briefly outline a procedure to efficiently rotate two-dimensional symmetric tensors.

Using index notation, where summation over repeated indices is implied, the rotation of a two-dimensional tensor is written as

$$\hat{w}_{i_1 i_2 \cdots i_n} = R_{i_1 j_1} R_{i_2 j_2} \cdots R_{i_n j_n} w_{j_1 j_2 \cdots j_n}, \tag{3.47}$$

where R_{ij} is the rotation matrix defined as

$$R \coloneqq \begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix}, \tag{3.48}$$

with rotation angle γ being the counterclockwise angle between $\mathbf{n}$ and $\hat{\mathbf{n}}$. The rank $n \in \mathbb{N}$ tensors $\hat{w}_{i_1 i_2 \cdots i_n}$ and $w_{j_1 j_2 \cdots j_n}$ are considered to be fully symmetric. Thus, all permutations of a fixed set of indices $j_k \in \{1, 2\}$ or $i_k \in \{1, 2\}$ share the same value and the tensors have $n + 1$ unique entries that can be enumerated by

$$w_\ell = w_{\underbrace{1 \cdots 1}_{n-\ell} \underbrace{2 \cdots 2}_{\ell}}, \quad \ell \in \{0, \ldots, n\}. \tag{3.49}$$

We now construct each entry $\hat{w}_m$, $m \in \{0, \ldots, n\}$ of the rotated tensor by evaluating the sum of products (3.47) with the fixed set of target indices $i_1 \cdots i_n$, given by the enumeration (3.49) with m, and all possible sets of source indices $j_1 \cdots j_n$. The summation over all components of $w_{j_1 j_2 \cdots j_n}$ can be avoided by considering only the unique components w_ℓ and for each component the sum of unique factors $R_{i_1 j_1} R_{i_2 j_2} \cdots R_{i_n j_n}$, dependent on $i_1 \cdots i_n$ and all $j_1 \cdots j_n$ that are permutations of the canonical order (3.49). The unique factors can be iterated based on the number if indices k, for which $i_l = j_l = 2$ holds, called here the overlap of twos. The product (3.47) can then be written as

$$\hat{w}_m = \sum_{\ell=0}^{n} w_l \sum_{k=\max\{0, \ell+m-n\}}^{\min\{\ell, m\}} \binom{n-m}{c} \binom{m}{d} \left(R_{11}^{k+a} b R_{21}^{c+d} \right), \quad m \in \{0, \ldots, n\}, \tag{3.50}$$

where $c = \ell - k$ is the number of indices such that $i_l = 1 \wedge j_l = 2$, $d = m - k$ is the number of indices such that $i_l = 2 \wedge j_l = 1$ and $a = (n - m) - c$ is the number of overlapping ones. Lastly, $b = 1$ if d is even and $b = -1$ otherwise, ensuring that $R_{21}^c R_{12}^d = b R_{21}^{c+d}$, since $R_{21} = -R_{12}$. In practice, the evaluations of the binomial coefficient and the exponents of the sine and cosine of the angle of rotation are computed at the beginning of the rotation procedure and stored in tables, such that the sum (3.50) can be efficiently evaluated for all tensors and their components.

3.3. Geometry representation

The coordinate transformation of the Galerkin element weak form to a reference element, performed in Section 3.1.2, relies on the specification of a suitable element geometry mapping $\Phi^e(\boldsymbol{\xi})$ and its Jacobian $J^e(\boldsymbol{\xi})$. Similar to the DG basis, the element geometry is represented by an expansion in basis functions with associated element-specific geometry coefficients. Moreover, the two-dimensional geometry basis will be based on the product of one-dimensional basis functions defined on the reference element. The employed geometry basis determines what element geometries and thus domain geometries can be represented exactly or approximately. To obtain a convergence order above second order, a high order curved boundary approximation is required [10, 12]. In the context of moment equations, the local curvature of the boundary can be related to a local Knudsen number [5]. This motivates the use of higher order geometry representations to accurately capture thermal nonequilibrium effects at and near curved walls. In the following, we introduce two possible higher order representations of the geometry, based on either Lagrange polynomials or non-uniform rational B-splines (NURBS).

3.3.1. Lagrange element geometry

The standard approach in classical finite elements methods is to use Lagrange polynomials to represent the element geometry, see for example [42]. In analogy to the nodal DG basis in Section 3.1.4, we define the geometric basis functions

$$N_i(\boldsymbol{\xi}) \coloneqq N_{i_1}(\xi_1) N_{i_2}(\xi_2), \quad \text{with} \quad N_{i_j}(\xi_j) \coloneqq \ell_j(\xi_j),\ j \in \{1, \dots, p+1\}, \tag{3.51}$$

as a product of one-dimensional Lagrange polynomials $\ell_j(\xi)$ defined in Equation (3.29). The total number of basis functions is given by $k = (p+1)^d$ with $p \geq 1$ being the degree of the geometry basis and $d = 2$ the spatial dimension of the problem. Every basis function is equipped with an associated geometry coefficient $\mathrm{x}_i^e \in \mathbb{R}^d$, which are the vertex coordinates in physical space of the corresponding interpolation node on the reference element. In particular, for a linear approximation ($p = 1$) the coefficients are the corner points of the element. Higher order geometry approximations are obtained by increasing the number of interpolation nodes in each direction, i.e. taking $p > 1$. Presently, we consider only the simplest case, in which the higher order interpolation nodes are taken to be uniformly spaced along the reference interval. The element geometry mapping reads as

$$\Phi^e(\boldsymbol{\xi}) = \sum_{i_1=1}^{p+1} \sum_{i_2=1}^{p+1} \hat{N}_{i_1}(\xi_1) \hat{N}_{i_2}(\xi_2) \mathrm{x}^e_{i_1 i_2} = \sum_{i=1}^{k} N_i(\boldsymbol{\xi}) \mathrm{x}_i^e, \quad i = i_2 + i_1(p+1) \tag{3.52}$$

and its Jacobian is given by the sum of dyadic products between the coefficients and the basis function derivatives:

$$J^e(\boldsymbol{\xi}) = \frac{\partial \Phi^e(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \sum_{i=1}^{k} \mathrm{x}_i^e \left(\nabla \hat{N}_i(\boldsymbol{\xi})\right)^T. \tag{3.53}$$

Both the mapping and its Jacobian are evaluated by means of shape function matrices $(S^N)_{qi} = N_i(\boldsymbol{\xi}_q)$. A tensor product formulation can once again be employed to reduce the computational effort. However, this optimization is currently neglected in favor of simplicity.

In practice, the starting point is often a quadrilateral mesh using a linear geometry approximation, generated for example with Gmsh [56]. To obtain a mesh with curved boundary representation, the mesh is supplemented with boundary curve parameterizations, which are interpolated to obtain the higher order geometry coefficients. The interpolation nodes inside the element are then adjusted to be linearly spaced in physical space between the nodes on the element interfaces. While this suffices for constant meshes, in the case of h-adaptivity, subdivision of a boundary element requires a re-interpolation of the boundary curve to increase the accuracy of the geometry.

3.3.2. NURBS element geometry

Another method to represent the geometry is by means of non-rational uniform B-splines (NURBS) as geometric basis functions. One advantage of NURBS over a Lagrangian description is their capability to represent conic sections exactly [45]. Furthermore, since NURBS are industry standard in computer-aided design (CAD) [45], the resulting geometric description can be encoded exactly by a NURBS-based geometry representation, allowing for easier mesh refinement. The present formulation of NURBS basis functions is based on the textbook [45], to which the reader is referred for details. Note that the present approach employs NURBS only to represent the geometry, not the solution. It is therefore in spirit similar to a NURBS-enhanced finite element method, see for example [57, 58]. However, we do not augment the element geometry with NURBS, but completely represent it by NURBS. Furthermore, each element's NURBS representation is independent of its neighbors, meaning that at construction of the mesh, a single NURBS curve along a boundary is broken down into element-wise sections.

Once again, the geometry mapping is given as a linear combination of basis functions and coefficients:

$$\Phi^e(\boldsymbol{\xi}) = \sum_{i=1}^{k} R_i(\boldsymbol{\xi}) \mathbf{B}_i^e, \tag{3.54}$$

where the coefficients $\mathbf{B}_i^e$ are the control point coordinates and the NURBS basis functions R_i are defined as [45]

$$R_i(\boldsymbol{\xi}) = \frac{N_{i_1}(\xi_1)N_{i_2}(\xi_2)w^e_{i_1 i_2}}{\sum_{\hat{i}_1}^{p+1}\sum_{\hat{i}_2}^{p+1} N_{\hat{i}_1}(\xi_1)N_{\hat{i}_2}(\xi_2)w^e_{\hat{i}_1\hat{i}_2}} = \frac{N_i(\boldsymbol{\xi})w_i^e}{\sum_{\hat{i}=1}^{k} N_{\hat{i}}(\boldsymbol{\xi})w_{\hat{i}}^e}, \quad i = i_2 + i_1(p+1). \tag{3.55}$$

Note that the weights w_i^e are element-specific and together with the control point coordinates constitute the geometry coefficients of an element. The one-dimensional B-spline basis functions $N_i(\xi)$ are constructed by means of the Cox-de Boor recursion formula [45]:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi), \quad \text{for} \quad p \geq 1 \tag{3.56}$$

and the base case ($p = 0$) is given by

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.57}$$

For $\xi_i = \xi_{i+1}$ we find $N_{i,0} = 0$ and furthermore consider terms in (3.56) with zero denominator as zero, since the corresponding basis function is also zero in this case [45]. The knots $\xi_i \in \mathbb{R}$ are specified in terms of a knot vector $[\xi_1, \ldots, \xi_{n+p+1}]^T$, where n is the number of B-spline basis functions. To construct the basis for the element geometry, we use a uniform and open knot vector on the reference interval with $n = p + 1$, which has the form $[-1, \ldots, -1, 1, \ldots, 1]^T$, where the first and last entry are repeated $p + 1$ times.

The element Jacobians are computed using Equation (3.53) with derivatives of the NURBS basis functions obtained from the quotient rule as [45]

$$\frac{\partial R_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = w_i^e \frac{W(\boldsymbol{\xi})N_i'(\boldsymbol{\xi}) - W'(\boldsymbol{\xi})N_i(\boldsymbol{\xi})}{W(\boldsymbol{\xi})^2}, \quad \text{where} \quad W(\boldsymbol{\xi}) = \sum_{i=1}^{k} N_i(\boldsymbol{\xi})w_i^e \tag{3.58}$$

and $N_i'(\boldsymbol{\xi}) = \partial_{\boldsymbol{\xi}} N_i(\boldsymbol{\xi})$.

When utilizing a NURBS basis for the element geometry, the geometry coefficients of an element are the weights w_i^e and control point coordinates $\mathbf{B}_i^e$. Therefore, the basis functions (3.55) depend on the element and their values at quadrature nodes cannot be precomputed and stored in shape function matrices. Instead, B-spline shape function matrices $(S^N)_{qi} = N_i(\boldsymbol{\xi}_q)$ are used to efficiently evaluate the NURBS basis functions (3.55) and derivatives (3.58).

In general, to impose a parameterized curve on a mesh boundary, all element geometry coefficients need to be determined by means of a curve fitting procedure. If the parameterization is given as a NURBS function, this may be avoided. Furthermore, for circle sections simple formulas to obtain the weights and control points for a quadratic NURBS function are available, see [45].

3.4. Time integration

To complete the numerical scheme, the spatial semidiscretization (3.21) is discretized in time. The model problems presented in Chapter 2 are in the context of slow and steady microchannel flows, for which a steady-state solution is sought. A solution $\boldsymbol{u}$ is considered to be steady if it satisfies

$$\dot{\boldsymbol{u}} = R(\boldsymbol{u}) = \boldsymbol{0}. \tag{3.59}$$

Note that we consider the steady state to be independent of the initial solution, which is the case for suitable boundary conditions, see Section 2.2.1. Consequently, the solution $\boldsymbol{u}$ of Equation (3.59) is unique, if the model is well-posed. In practice, a residual of zero is never achieved due to machine precision rounding errors. Instead, an approximate solution, that satisfies Equation (3.59) up to a specified tolerance, is sought.

The steady-state solution is obtained by integrating Equation (3.59) in time. This can be accomplished by advancing an initial solution guess in time by means of a time-stepping scheme until the solution is sufficiently steady, which is discussed in Section 3.4.1. Alternatively, the linear equation system implied by Equation (3.59) can be directly solved to obtain the steady-state solution, see Section 3.4.2.

3.4.1. Runge-Kutta time discretization

The time-marching approach, in which one solves a steady-state problem by means of a time evolution of a time-dependent problem, is a standard approach in engineering practice [18]. In this section, we discretize time using an explicit Runge-Kutta method, leading to so-called Runge-Kutta discontinuous Galerkin methods, see for example [35]. For an explicit time-stepping scheme, the time-step Δt is restricted by the well known Courant-Friedrichs-Lewy (CFL) condition [12, 17, 18, 43, 44]:

$$\Delta t = C_{cfl} \frac{h_{min}}{\lambda_{max}(2p+1)}, \tag{3.60}$$

where $C_{cfl} > 0$ is the Courant number, p is the polynomial degree of the DG basis, h_{min} is a measure of the minimum element size and λ_{max} is an estimate of the maximum wave speed, i.e. the maximum absolute eigenvalue of the flux Jacobian, throughout the domain. While for simple problems the CFL condition is an exact stability criterion obtained from linear stability analysis, see for example [18], it can be considered more as a heuristic for complex problems. While in principle, a Courant number $C_{cfl} \in (0, 1]$ yields a stable scheme, due to the involved estimates a more practical choice is $C_{cfl} = 0.9$ [59]. Herein, the minimum element size h_{min}

is taken as the minimum diagonal length over all elements. However, note that this measure may prove insufficient for cases with large aspect ratio elements. For linear problems the maximum wave speed estimate λ_{max} is constant and computed as the sum of the maximum absolute eigenvalues of the x- and y-direction system matrices, which is two times the maximum wave speed for a rotationally invariant problem. While the additional factor two might be unnecessary, it is included to be consistent with the two-dimensional CFL conditions given in [17, 35]. Lastly, the condition (3.60) does not account for potential stiffness of the system of ordinary differential equations induced by the relaxation term. We assume that the system is not stiff for the Knudsen numbers of interest.

Up to third order, higher order DG schemes require a higher order time discretization to remain stable for a fixed Courant number [44]. Therefore, time discretization is performed using a third order strong stability preserving Runge-Kutta scheme (SSPRK3), given by [43, 60]:

$$\boldsymbol{u}^{(1)} = \boldsymbol{u}^{(0)} + \Delta t\, R(\boldsymbol{u}^{(0)}) \tag{3.61a}$$

$$\boldsymbol{u}^{(2)} = \boldsymbol{u}^{(0)} + \Delta t \left(\frac{1}{4}\boldsymbol{u}^{(0)} + \frac{1}{4} R(\boldsymbol{u}^{(1)}) \right) \tag{3.61b}$$

$$\boldsymbol{u}^{(3)} = \boldsymbol{u}^{(0)} + \Delta t \left(\frac{1}{6}\boldsymbol{u}^{(0)} + \frac{1}{6} R(\boldsymbol{u}^{(1)}) + \frac{2}{3} R(\boldsymbol{u}^{(2)}) \right) , \tag{3.61c}$$

where R is the residual operator (3.21), the solution of the previous time step is denoted by $\boldsymbol{u}^{(0)}$ and the result of the current time step by $\boldsymbol{u}^{(3)}$. For a spatial discretization higher than third order, the Courant number needs to be reduced accordingly, see [44].

Starting at an initial guess, i.e. $\boldsymbol{u}^{(0)} = \boldsymbol{0}$, the time stepping scheme (3.61) is applied until the steady state is reached. A suitable stopping criterion is required to complete the time-marching procedure, since the residual does not reach zero due to rounding errors. Instead of reaching zero, we therefore require that some measure of magnitude of the residual vector $\boldsymbol{r} = R(\boldsymbol{u})$ lies below a predefined threshold ε. Discrete counterparts of the L_2 and L_∞ norms [43] are chosen as a measure, yielding the respective stopping criterions

$$\frac{1}{\sqrt{n}} \|\boldsymbol{r}\|_2 = \sqrt{\frac{1}{n} \textstyle\sum_{i=1}^{n} r_i^2} < \varepsilon \quad \text{and} \quad \|\boldsymbol{r}\|_\infty = \max_{1 \le i \le n} \{|r_i|\} < \varepsilon, \tag{3.62}$$

where n is the number of degrees of freedom, which is equal to the size of the residual vector. The choice of a suitable value for the threshold depends on the specific problem considered. Moreover, depending on the solution magnitude, it may be more suitable to use relative versions of the aforementioned norms. For the present simulations, using the discrete L_∞ norm and picking a value moderately close to machine precision, e.g. $\varepsilon = 10^{-12}$, performs adequately.

3.4.2. Solution by means of linear solvers

Another common approach is to directly solve the set of algebraic equations resulting from Equation (3.59). The residual operator (3.21) of a linear problem is linear and can therefore be written as

$$R(\boldsymbol{u}) = \boldsymbol{A}\boldsymbol{u} - \boldsymbol{b} \overset{!}{=} \boldsymbol{0} \implies \boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}, \tag{3.63}$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the system matrix and $\boldsymbol{b} \in \mathbb{R}^n$ the right-hand side vector of the linear system of equations. The total number of degrees of freedom n gives the size of the linear system. Clearly, the steady-state solution can be obtained by solving (3.63) using any available linear solver. For this purpose, the right-hand side vector $\boldsymbol{b} = -R(\boldsymbol{0})$ is obtained by a single evaluation of the residual. Furthermore, a matrix-multiplication with the system matrix can be evaluated in a matrix-free way by means of a single evaluation of the residual:

$$\boldsymbol{A}\boldsymbol{u} = R(\boldsymbol{u}) + \boldsymbol{b}. \tag{3.64}$$

Alternatively, the entire matrix can be collected by either modifying the assembly algorithms or by evaluating element residuals with standard basis vectors to extract column slices of the system matrix. However, in this work we employ the matrix-free formulation, as it requires less memory and is able to deliver higher performance than matrix-based methods [40]. A downside of this approach is that it restricts the applicable solution methods and preconditioners.

To solve the linear system (3.63) in a matrix-free way, the BiCGstab(l) [61] iterative Krylov subspace method is employed. Furthermore, convergence is accelerated by using preconditioning, in which the linear system is rewritten as

$$\boldsymbol{P}_l \boldsymbol{A}\boldsymbol{z} = \boldsymbol{P}_l \boldsymbol{b}, \tag{3.65}$$

with the left preconditioner $\boldsymbol{P}_l$ being an approximate inverse of $\boldsymbol{A}$, such that the resulting system (3.65) is better conditioned [18, 62]. Herein, we consider only two basic choices for the left preconditioner. Firstly, a diagonal preconditioner, where $\boldsymbol{P}_l$ is the inverse of the diagonal of $\boldsymbol{A}$. However, note that the system matrix diagonal entries may be zero, depending on the solved equations and the DG basis. In these cases the diagonal preconditioner may not be used. The second option is a block-diagonal preconditioner, where the diagonal blocks of $\boldsymbol{P}_l$ contain the inverses of the diagonal blocks of the system matrix. Each diagonal block corresponds to the equations and solution variables of a single element and can therefore be extracted from the element residuals.

Preliminary simulations reveal, that in the inspected cases the solution by means of the linear solver outperforms the time-marching approach of the previous section. For this reason, simulations performed in the following chapters employ the linear solver to obtain the steady-state solution.

3.5. Model- and *hp*-adaptivity

In this section, the discontinuous Galerkin formulation presented in the previous Sections is extended to allow for local refinement of the mesh, polynomial degree and model level. The local refinement of the mesh, leading to an h-adaptive scheme, and the necessary data structures are briefly outlined in Section 3.5.1. Of particular interest in this work is the capability to dynamically adjust the model level in each element. This is accomplished by an implementation that allows for a variable amount of degrees of freedom in each element in combination with a suitable method to compute the interface fluxes between elements with differing model levels. This is handled by a mortar method [63], which facilitates the exchange of information between the non-conforming elements and is descibed in Section 3.5.2.

Note that for any adaptive procedure, a suitable refinement criterion is required. In this work, we focus solely on the capability of the numerical scheme to allow for adaptivity and leave the development and implementation of adaptive procedures for future work.

3.5.1. Adaptive mesh refinement

Inspired by p4est [64], the elements of the spatial discretization are organized into a forest of trees. Given a coarse base mesh, each element is taken to be the root of a quadtree of elements. To allow for different meshes with different resolution, e.g. to compute errors on finer meshes while maintaining the coarse mesh, a list of active elements is used to index the elements of the forest mesh. In the simplest case, the leaf nodes of all element trees are taken to be the active elements.

An adaptive discretization is obtained by refining only some of the elements, yielding local differences in resolution. Each element can be refined individually into four children by splitting the parent element along the axes of the reference domain. Note that we only consider isotropic refinement, meaning that an element is either a leaf node, or has four children. Furthermore, we restrict the allowed resolution difference between neighboring active elements to one, as is done in [64].

When refining an element, the geometry coefficients of the child elements need to be computed from the parent element's coefficients. For a Lagrange geometry representation this is done by simply interpolating the parent element's geometry. Refinement of a NURBS element is achieved by knot insertion [45]: First, the two-dimensional NURBS geometry coefficients are tranformed into three-dimensional B-spline control points. Thereafter, knot insertion is performed on the B-spline control points, followed by transforming the resulting points back to NURBS weights and control points, see [45] for details.

3.5.2. The mortar method

The numerical fluxes on non-conforming interfaces are computed by means of the mortar element method. Roughly speaking, the interface solutions of both sides are projected onto so-called mortars, on which the numerical fluxes are computed analogously to the conforming case, followed by projecting the fluxes back to the left and right interfaces [63]. In the following, we will first specify how the numerical fluxes at the quadrature node of model-non-conforming interfaces are computed. Thereafter, the employed mortar method for handling *hp*-non-conforming interfaces, given in [63], is briefly outlined.

On each mortar interface, the left and right solutions Φ^L and Φ^R are represented by means of Lagrange polynomials interpolating the conforming mortar quadrature node values [63]:

$$\Phi^{L,R}(\xi) = \sum_{q=1}^{k^\Sigma} \Phi_q^{L,R} \ell_q^\Sigma(\xi) \tag{3.66}$$

where $\Phi_q^{L,R}$ denotes the (vector-valued) nodal coefficients, i.e. quadrature node values, of the left and right solution polynomials and the number of basis functions on the mortar is denoted by k^Σ. The fluxes are defined on each model level, thus to evaluate the numerical fluxes at each quadrature point, one of the solutions needs to be projected to the model level of the other solution. In the same way that the *hp*-mortar is defined on the finest resolution of the contributing interfaces, see [63], we project the coarser solution to the level of the finer one. This is justified by the condition that a projection to the mortar and back should yield the original solution, which is also called the outflow condition [63]. Therefore, using the model projection operator (2.66) defined in Section 2.4, the scaled normal interface flux Ψ is evaluated as

$$\Psi_q^{L,R} = \boldsymbol{P}^{m^\Sigma \to m^{L,R}} F^*(\boldsymbol{P}^{m^L \to m^\Sigma} \Phi_q^L, \boldsymbol{P}^{m^R \to m^\Sigma} \Phi_q^R) \sqrt{\det g_q}. \tag{3.67}$$

The mortar model level $m^\Sigma = \max\{m^L, m^R\}$ is chosen as the highest of the two model levels. Consequently, one of the projection operators is the identity. The projection of the flux back to a lower model level is a simple truncation of the higher moment fluxes. Note also, that the geometry scaling of the reference coordinate transformation is included in the quadrature point flux evaluations. This is important for the *hp*-mortar projections presented in the following, as these are performed in reference space.

For an *hp*-conforming interface, the mortar solution function coefficients coincide with the corresponding quadrature node evaluations of the solution functions. In the case of different polynomial degrees $p^{L,R}$, the solutions need to be projected to the mortar to obtain a matching set of nodes to compute the fluxes at. Note that we only consider the case in which the number of quadrature nodes equals the number of basis functions. The solutions on the left and right reference interfaces are represented by

a basis expansion in Lagrange polynomials with the interpolation nodes being the quadrature nodes on the interface:

$$u^{L,R}(\xi) = \sum_{j=1}^{k^{L,R}} u_j^{L,R} \ell_j^{L,R}(\xi), \quad \xi \in [-1, 1], \tag{3.68}$$

where $k^{L,R} = p^{L,R} + 1$ is the number of one-dimensional basis functions of the left and right elements respectively. Note that the quadrature point solution values $u_j^{L,R}$ are readily available from the interface interpolation step of the assembly algorithm. The coefficients of the mortar solution functions are then obtained by means of an L_2-projection [63]:

$$\min_{\Phi^{L,R}} \int_{-1}^{1} \left(\Phi^{L,R} - u^{L,R}\right)^2 d\xi. \tag{3.69}$$

Since the L_2-projection is an orthogonal projection, it holds that [63, 65]

$$\int_{-1}^{1} \left(\Phi^{L,R} - u^{L,R}\right) \varphi_i^{\Sigma} \, d\xi = 0, \quad \varphi_i^{\Sigma} \in \Pi_{p^{\Sigma}}, \tag{3.70}$$

where the test functions φ_i^{Σ} are a basis for the polynomials of degree $\leq p^{\Sigma}$. To satisfy the outflow condition p^{Σ} is chosen as the maximum between the left and right polynomial degrees [63]. Inserting the definitions of the solution functions (3.66) and (3.68), yields [63]

$$\sum_{q=1}^{k^{\Sigma}} \Phi_q^{L,R} \underbrace{\int_{-1}^{1} \ell_q^{\Sigma} \varphi_i^{\Sigma} \, d\xi}_{=:M_{iq}} = \sum_{j=1}^{k^{L,R}} \psi_q^{L,R} \underbrace{\int_{-1}^{1} \ell_j^{L,R} \varphi_i^{\Sigma} \, d\xi}_{=:S_{jq}^{L,R}}, \tag{3.71}$$

from which the projection operators $\boldsymbol{P}^{L,R\to\Sigma} = \boldsymbol{M}^{-1}\boldsymbol{S}^{L,R}$ are obtained by evaluating the integrals exactly via a quadrature rule. Note that the matrices $\boldsymbol{M}$ and $\boldsymbol{S}^{L,R}$ depend on the choice of test function for the L_2-projection, but the resulting projection operator does not, since the result of the projection is unique.

The projection of the scaled numerical fluxes from the mortar back to the left and right interfaces is performed analogously. The normal fluxes Ψ and the fluxes $\tilde{F}^{L,R}$ at the left and right interfaces are represented by means of a Lagrange basis on the corresponding quadrature nodes. Inserting the basis expansions of the fluxes into the L_2-projection [63]

$$\int_{-1}^{1} \left(\tilde{F}^{L,R} - \Psi\right) \varphi_i^{L,R} \, d\xi = 0, \quad \varphi_i^{L,R} \in \Pi_{p^{L,R}}, \tag{3.72}$$

yields the projection matrices $\boldsymbol{P}^{\Sigma\to L,R} = (\boldsymbol{M}^{L,R})^{-1}\boldsymbol{S}$, with [63]

$$M_{ij}^{L,R} = \int_{-1}^{1} \ell_j^{L,R} \varphi_i^{L,R} \, d\xi, \quad i, j \in \{1, \ldots, k^{L,R}\}, \tag{3.73}$$

$$S_{iq} = \int_{-1}^{1} \ell_q^{\Sigma} \varphi_i^{L,R} \, d\xi, \quad i \in \{1, \ldots, k^{L,R}\}, q \in \{1, \ldots, k^{\Sigma}\}. \tag{3.74}$$

The final mortar case is that of a spatially non-conforming interface. Herein, we only consider interfaces where one side is a single resolution finer than the other side and the subdivision is in the middle of the interface. In the following, without loss of generality, we consider the left interface to be the coarser one. Two mortars, corresponding to the two fine-side interfaces are employed for the evaluation of the interface fluxes. The projection of the finer interface quadrature point solution values to the mortars is a simple identity mapping.

Once again, an L_2-projection is employed to interpolate the solution of the coarse interface to the quadrature nodes on the mortar interfaces [63]:

$$\int_{-1}^{1} \left(\Phi^{\Sigma}(z) - u^{L}(\xi(z)) \right) \varphi_i^{\Sigma}(z)\, dz = 0, \quad \varphi_i^{\Sigma} \in \Pi_{p^{\Sigma}}, \tag{3.75}$$

where $z = 2\xi \pm 1$ is the local coordiante of the mortars and Φ^{Σ} belongs either to the top Σ^T or bottom Σ^B mortar. Projection matrices are then obtained analogously to the p-mortar case above. For the projection of the two mortar flux polynomials back to the single coarse interface, we define the piecewise function $\Psi^L(\xi)$ as [63]

$$\Psi^{L}(\xi) = \begin{cases} \Psi^{\Sigma^T}(2\xi + 1), & 0 \leq \xi \leq 1, \\ \Psi^{\Sigma^B}(2\xi - 1), & -1 \leq \xi \leq 0. \end{cases} \tag{3.76}$$

Note that the ambiguity at the midpoint $\xi = 0$ does not matter, because the function is only evaluated inside of integrals. Using the above function, the L_2-projection flux (3.72) now reads as [63]

$$\int_{-1}^{1} \tilde{F}^{L} \varphi_i^{L}\, d\xi = \int_{-1}^{0} \Psi^{\Sigma^B} \varphi_i^{L}\, d\xi + \int_{0}^{1} \Psi^{\Sigma^T} \varphi_i^{L}\, d\xi, \quad \varphi_i^{L} \in \Pi_{p^{L}}. \tag{3.77}$$

After transforming the right-hand side integrals to the mortar interface coordinates, the two projection matrices are readily obtained by computing the basis function integrals as done above.

In practice, since all integrals are on the fixed reference domain, the projection matrices are precomputed once at the beginning of the simulation. Furthermore, fully non-conforming interfaces are handled by first projecting to h-mortars, then p-mortars and lastly performing the model projection at the quadrature nodes. After the computation of the numerical fluxes, the back projection is performed in the reverse order. While the hp-mortars can be combined into a single projection step, the present implementation keeps them separated for the sake of simplicity. Lastly, note that only the second step of the assembly Algorithm 1 needs to be modified to accommodate the mortar method. In this way, the presented DG formulation is extended to handle adaptive spatial and model discretization.

3.6. Implementation details

A major part of this work is the implementation of the DG formulation described in this chapter in Julia [1]. The implementation is focused on providing a proof-of-concept research code to explore model adaptive simulation techniques for moment equations. In doing so, the code relies on several external dependencies, briefly outlined in the following. The extended heat system matrices are obtained from the PDE using `Symbolics.jl` [66], which is also used to compile expressions for the production, source and boundary terms into efficient Julia functions. Quadrature weights and nodes for the DG basis are computed using `FastGaussQuadrature.jl` [67]. To achieve acceptable performance, `LoopVectorization.jl` [68] and `Polyester.jl` [69] are employed to speed up the assembly kernels. Most assembly routines are dynamically sized, to simplify the model and polynomial adaptivity, which require different amounts of element degrees of freedom for each element. However, statically sized vectors from `StaticArrays.jl` [70] are used wherever applicable, such for the geometry information. The solution of the linear system is performed by means of wrapping the residual operator via `LinearMaps.jl` [71] and using an iterative Krylov solver from `IterativeSolvers.jl` [72]. Lastly, the simulation results are written to ParaView [73] files by means of `WriteVTK.jl` [74].

General unstructured meshes can be created using Gmsh [56] and loaded into the developed simulation tool. Alternatively, structured mesh generators for simple geometries are implemented. The `forest mesh` contains a `geometry basis`, which allows for the representation of curved geometry. In a processing step, circular geometry can be imposed on the boundary interfaces and elements of the mesh by means of a curve parameterization. A `problem` structure is used to collect the equations and boundary conditions of the initial-boundary value problem. Legendre and Lagrange polynomials of arbitrary degree are used to automatically construct a `DG basis` of a required order. A `quadrature` object is used to store precomputed shape function matrices of the DG and geometry bases and provide easy to use wrappers around the implemented tensor product kernels, discussed in Section 3.1.5. An `assembler` structure is used to manage geometry and DG quadrature objects and corresponding assembly kernels. To allow for an arbitrary distribution of degrees of freedoms over the elements, relevant indexing information is stored in a `cache` object. Together, the `forest mesh`, `problem`, `DG basis`, `quadrature`, `assembler` and `cache` objects form a `semidiscretization`, which is used to evaluate the residual. Lastly, in the case of a hierarchy of problems (m-adaptivity) and DG bases (p-adaptivity), hierarchical versions of the aforementioned code structures are used to encapsulate objects for each model and polynomial level.

Importantly, the implementation outlined above is structured in a way to allow a variable amount of solution variables and equations in each element, which is a necessity for model-adaptivity. Notably, many existing codes use statically sized objects to allow for significant compiler optimization of the assembly kernels. When using statically sized kernels, the executed code is different for elements with different model and polynomial levels. When the elements are ordered haphazardly, this may incur significant switching overhead. Therefore, to maintain a good throughput of elements, the order of elements can become important, especially for larger problems. An efficient ordering technique that is suitable for adaptive refinement may thus be required to reach high performance. The current implementation relies mostly on dynamically sized objects for the sake of simplicity and leaves a high performance considerations for future work.

4. Convergence studies

A crucial step in the development of any numerical software is its verification, which ensures the correctness of the implementation and the suitability of the numerical scheme to accurately solve the model problems. In this chapter, convergence studies are performed based on test cases of the model problems introduced in Chapter 2, for which analytical solutions are available. The goal is to ensure that the DG formulation presented in the previous chapter is capable of solving the model problems introduced in Chapter 2. In doing so, the correctness of the implementation is also asserted. Moreover, the conducted numerical experiments serve to investigate whether the solver satisfies the design goals outlined in Chapter 1. For this reason, emphasis is put on higher order convergence, in particular in cases with curved domains, which require a higher order representation of the geometry.

In the following sections, the convergence behavior is systematically investigated using progressively more complex test cases. First, in Section 4.1 convergence of the approximate solution is shown for a heat system test case on structured meshes. Thereafter, we switch to the extended heat system in Section 4.2, employing structured meshes and unstructured meshes. Of particular interest is the influence of the Knudsen number on the convergence of the scheme and the stability of the boundary conditions in the presence of curved geometry. Lastly, for a model-adaptive scheme to make sense, the model hierarchy needs to converge, which is briefly discussed in Section 4.3.

All simulations in this work are performed using a modal DG basis with L_∞ normalized trial functions and the LLF flux, as mentioned in Sections 3.1.4 and 3.1.6 respectively. Preliminary computations have shown that the LLF flux is sufficient and that the modal basis seems more robust in cases with highly deformed elements. However, a detailed investigation of the choice of DG basis and numerical flux on the robustness and accuracy of the scheme is left for future work.

Figures and plots were created using gnuplot [75] and ParaView [73].

4.1. Heat system convergence studies

The advection diffusion equation (2.12) is reduced to the diffusion equation by setting the advection velocity $\boldsymbol{a}$ to zero. Furthermore, let the diffusivity be given by $k = 1$. The steady state of two-dimensional the heat system (2.13a) then coincides with the solution of the Poisson problem

$$-\Delta\theta(x, y) = s(x, y), \tag{4.1}$$

with heat source s. We consider an annulus (double cylinder) geometry parameterized by the inner cylinder radius R_0 and outer radius R_1 as computational domain, in which an approximate solution to equation (4.1) is sought. The choice of this geometry yields a non-trivial test case, due to the curved boundaries [7]. The manufactured solution

$$\theta(r) = 1 + (q_n^w - 2R_0)r + r^2, \tag{4.2}$$

depending only on the radius $r = \sqrt{x^2 + y^2}$, is inserted into Equation (4.1) to obtain the source term

$$s(x, y) = -(4 + \frac{q_n^w - 2R_0}{r}). \tag{4.3}$$

Dirichlet and Neumann boundary conditions are prescribed at the outer and inner cylinders respectively. The corresponding boundary condition operators and data read as

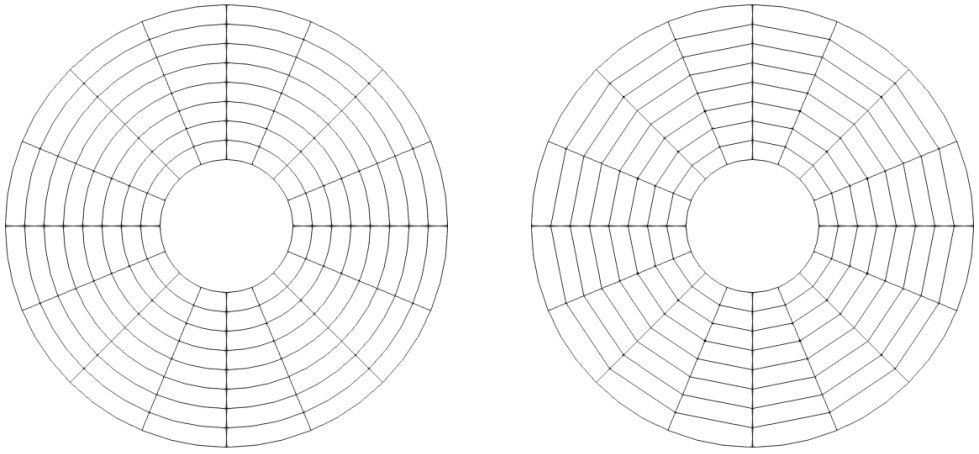
$$B(n)|_{R_1} = [1, 0, 0]\,, \qquad g(n, x)|_{R_1} = \theta(x, y), \tag{4.4}$$

$$B(n)|_{R_0} = [0, -n_x, -n_y]\,, \quad g(n, x)|_{R_0} = -q_n^w, \tag{4.5}$$

where the inner cylinder heat flux is taken as $q_n^w = 0.1$.

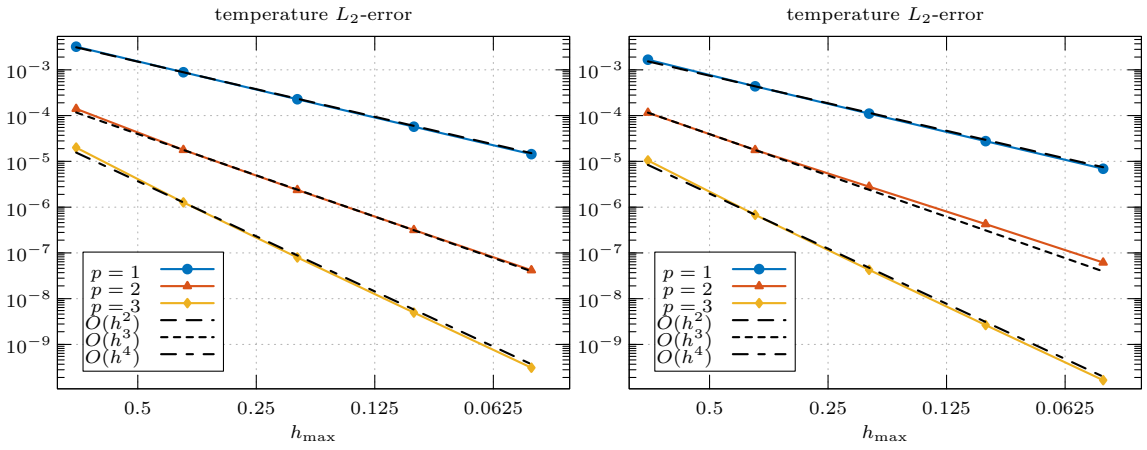
The computational domain is spatially discretized by structured meshes, depicted in Figure 4.1. In the case of a structured mesh all elements are readily curved, as seen in Figure 4.1a, yielding a uniformly smooth approximation of the geometry throughout the domain. In practice, especially for unstructured meshes, propagating the curvature into the elements inside the domain can be challenging. Therefore, in practice the geometry is only imposed on the boundary elements, as shown in 4.1b.

Figures 4.2 and 4.3 show the L_2-error of the temperature field approximation computed on successively refined structured meshes. A quadrature rule rule with two more nodes is used to compute the error between the approximate and analytical solutions. The characteristic mesh length $h_{\max}$ is taken to be the maximum element diagonal throughout the mesh. In all cases the obtained convergence rates roughly match the theoretical convergence rates. In particular, higher order convergence is obtained in spite of the curved geometry, due to the higher order geometry approximations.



a) Mesh with all elements curved. **b)** Mesh with only boundary elements curved.

Figure 4.1. Structured meshes for the annulus geometry. The curved geometry is imposed on all elements in (a), while in (b) only the boundary elements exhibit curved edges. Both meshes contain 128 elements and correspond to the second refinement level of all structured mesh convergence studies presented herein.

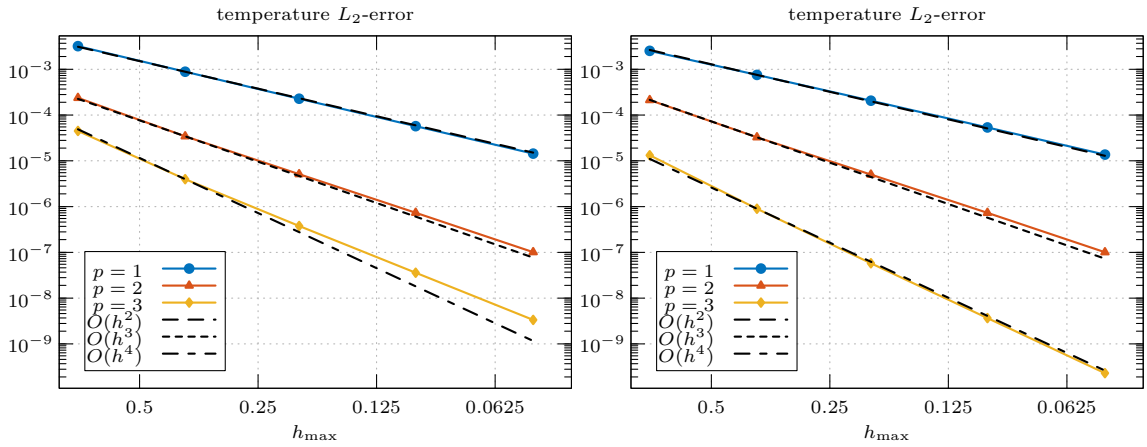


a) Lagrange geometry basis. **b)** NURBS geometry basis.

Figure 4.2. L_2-errors of the temperature using fully curved structured annulus meshes for various DG basis polynomial degrees p. The degree of the Lagrange geometry representation (a) is taken to be equal to the DG polynomial degree, while the NURBS (b) degree is fixed at second order. The black reference lines show the theoretical convergence order of the DG schemes.

The convergence rates of the L_2-error for the fully curved mesh are plotted in Figure 4.2. Also shown are three reference lines, corresponding to second, third and fourth order convergence. Figure 4.2a shows the error for the DG formulation with a Lagrange geometry basis. Clearly, optimal convergence rates are obtained. In comparison, Figure 4.2b shows the same simulations using a NURBS geometry basis. The convergence rate of the third order scheme is observed to be slightly below the theoretical optimum. Nonetheless higher order is still achieved. Comparing the two figures, comparable errors are obtained.

We switch from the fully curved mesh to the mesh with only curved boundary elements, for which convergence rates are plotted in Figure 4.3. Using a Lagrange geometry representation, depicted in Figure 4.3a, shows that once again higher order convergence is obtained. However, the fourth order errors deviate significantly from the reference line, indicating a degradation of the convergence rate. In comparison, the NURBS geometry representation maintains its fourth order accuracy, as shown in Figure 4.3b. Furthermore, the overall error of the fourth order NURBS scheme is lower than that of the corresponding Lagrange cases.



a) Lagrange geometry basis

b) NURBS geometry basis

Figure 4.3. L_2-errors of the temperature using structured annulus meshes, where only the boundary interfaces are curved, for various DG basis polynomial degrees p. The degree of the Lagrange geometry representation (a) is taken to be equal to the DG polynomial degree, while the NURBS (b) degree is fixed at second order. The black reference lines show the theoretical convergence order of the DG schemes.

4.2. Extended heat system convergence studies

In this section, we broaden the scope to the extended heat system (2.21) discussed in Section 2.2.2, which acts as a substitute for more general moment systems. In particular the boundary conditions constitute a meaningful generalization and require careful treatment, see [5]. The computational domain remains the same annulus geometry as presented in the previous section and the errors are also computed in the same way. A benefit of this geometry is that there are analytical solutions available for the extended heat system [7]. The boundary conditions of the considered problem read as

$$\theta^w|_{R_0} = 0, \quad \theta^w_{R_1} = 0.2 \quad \text{and} \quad q_t^w|_{R_0} = q_t^w|_{R_1} = 0, \tag{4.6}$$

and a constant heat source $s(x, y) = 1$ is applied to the temperature equation of the extended heat system. For this problem, an analytical solution can be derived [5, 76, 77]:

$$\theta(r) = c_1 - \frac{1}{15\mathrm{Kn}} \left(r^2 + 4c_2 \ln\left(\frac{r}{\mathrm{Kn}}\right)\right), \tag{4.7}$$

where c_1 and c_2 are integration constants dependent on the boundary conditions and Knudsen number Kn. The relevant values are given in Table 4.1.

	Kn = 0.001	Kn = 0.01	Kn = 0.1	Kn = 1	Kn = 1 ($\theta^w_{R_1} = 0$)
c_1	$-2.81698981865 \times 10^{2}$	$c_2 \to -1.65281349961 \times 10^{1}$	$-3.39949695671 \times 10^{-2}$	$6.98759334357 \times 10^{-1}$	$4.91214883239 \times 10^{-1}$
c_2	$-1.89309689621 \times 10^{-4}$	$-1.91787286568 \times 10^{-3}$	$-1.78273408219 \times 10^{-2}$	$-4.80275384579 \times 10^{-2}$	$-3.16265577665 \times 10^{-2}$

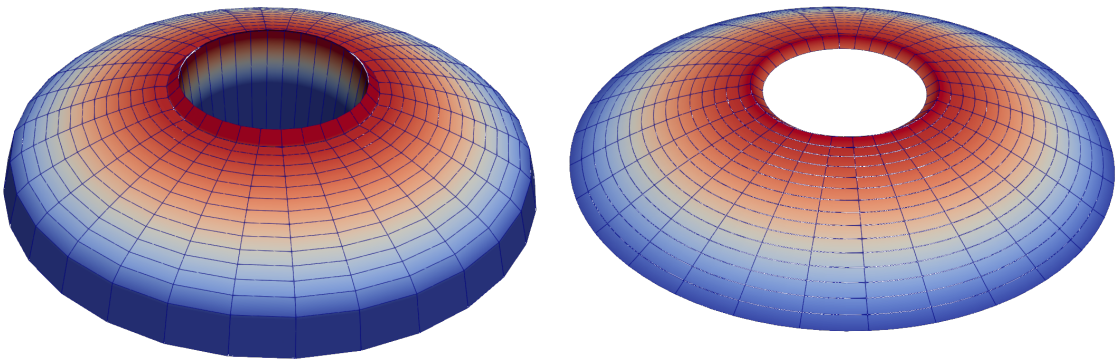
Table 4.1. Integration constants for the analytical solution of the extended heat system.

4.2.1. Approximation quality at curved boundaries

Many engineering applications require an accurate description of the interaction of a fluid with the boundary. In the case of moment equations, motivated by investigations in [5, 78], particular emphasis needs to be placed on the accuracy of the approximation near boundaries. Therefore, in this section, we consider the influence of the geometry representation on the solution near the boundary. Especially, the differences between linear and higher order approximations of a curved boundary are highlighted. In the following, to simplify the setup, we consider only a wall temperature of zero at both boundaries, i.e. $\theta^w_{R_1} = 0$, and a Knudsen number of $\mathrm{Kn} = 1$.

Figure 4.4 depicts the temperature profiles of the extended heat system test case on structured meshes obtain with a linear DG basis and either a linear Lagrange or quadratic NURBS geometry basis. In the case of a linear Lagrange geometry approximation, shown in Figure 4.4a, the solution exhibits an unphysical kink at the

outer boundary and a single spurious oscillation at the inner boundary. Oscillations at the boundary were previously observed for the extended heat system at high Knudsen numbers in [5]. Reducing the Knudsen number alleviates the issue. Furthermore, the heat flux seems smooth and unaffected by the problems in the temperature profile. A possible explanation may be Babuška's paradox, which can occur when specifying a normal direction Dirichlet boundary condition together with a tangential direction Neumann boundary condition [78]. Moreover, no issues where encountered in [79], which used a stabilized finite element formulation. A detailed investigation of this phenomenon is left for future work. A curved representation of the boundary geometry also seems to mostly eliminate the flaws in the temperature profile, as can be seen on the basis of Figure 4.4b, where only a small kink at the inner boundary is observed.

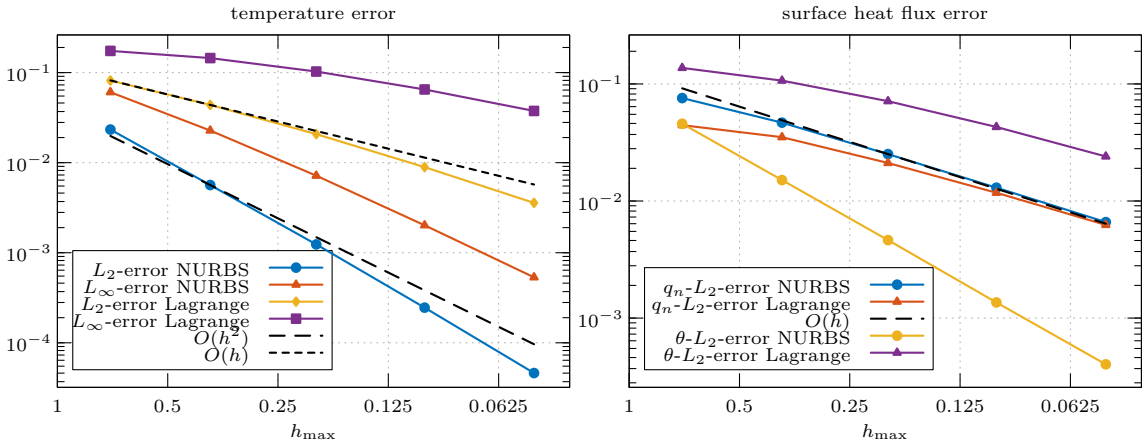


a) Lagrange geometry basis. **b)** NURBS geometry basis.

Figure 4.4. Temperature profile of the extended heat system solution on a structured mesh with 512 elements using either a linear Lagrange (a) or quadratic NURBS (b) basis for the geometry.

To quantify the effect of the observed numerical issues, we consider the convergence rates of the temperature profile inside the domain and along the inner boundary in Figure 4.5. While both Lagrange and NURBS geometry representations seem to yield a converging scheme, Figure 4.5a clearly shows a degradation in convergence rate of the Lagrange-based scheme. Only first order convergence is obtained for the L_2-error. Furthermore, the overall errors are significantly larger than the errors of the solutions employing a curved geometry description. In all cases, the L_∞-error, computed as the maximum absolute difference between the approximate and exact solution over all quadrature points, is larger than the respective L_2-error. The convergence rates of profiles along the inner boundary is shown in Figure 4.5b. Surprisingly, both geometry representations yield only first order convergence rate for the normal direction heat flux at the inner boundary. Furthermore, both cases yield comparable errors, which may

be due to the heat flux profiles being undisturbed by the issues in the temperature profile, as mentioned above. Lastly, the temperature distribution along the wall converges with the optimal second order in the case of a curved geometry description, while both the convergence rate and the magnitude of the errors are significantly worse in the case of a linear geometry approximation. Consequently, the higher order representation of curved boundaries is a crucial component of the numerical scheme. Thus, in the following sections, a quadratic NURBS representation is employed for all test cases.



a) Volume errors. **b)** Inner cylinder surface errors.

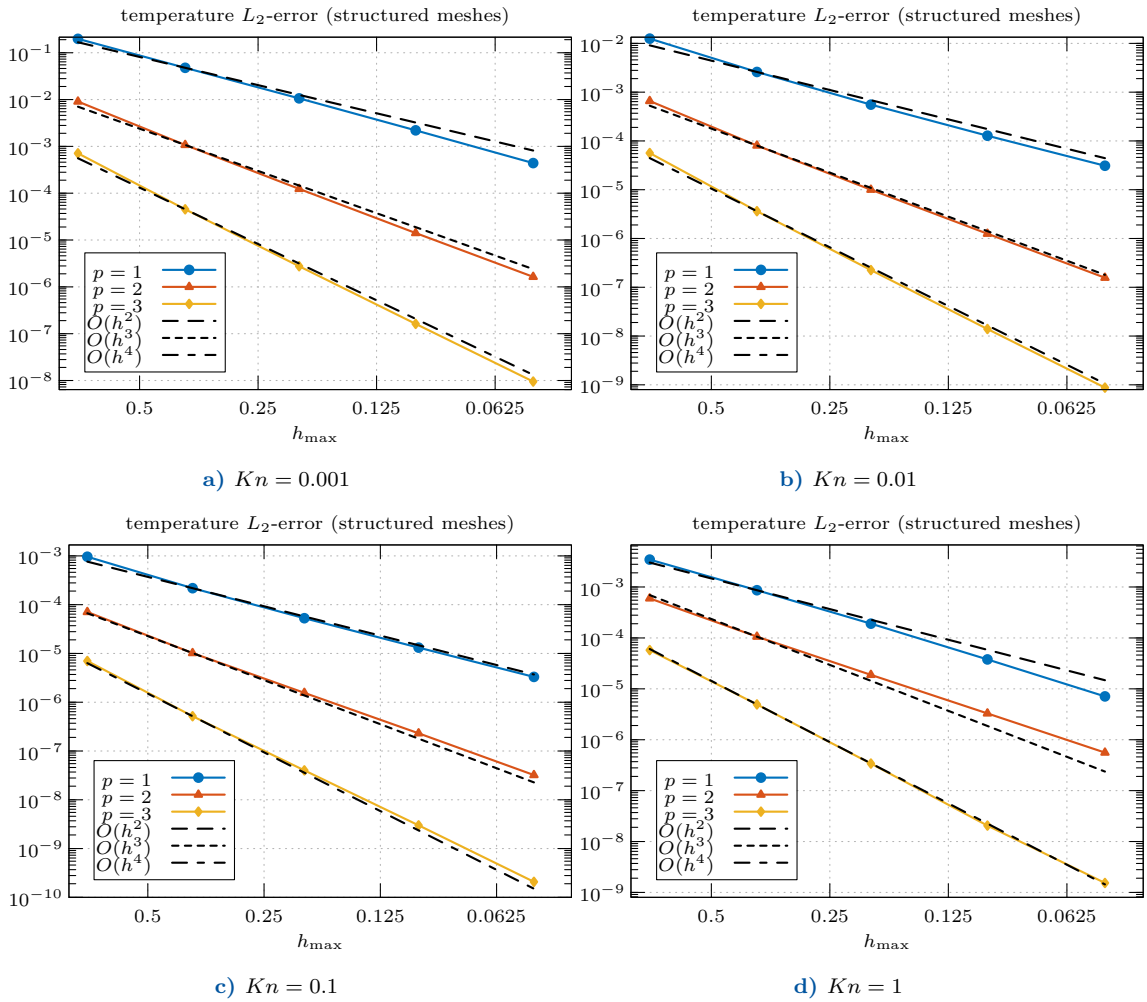
Figure 4.5. Convergence rates of the L_2- and L_∞-errors of the temperature inside the domain (a) and the temperature and normal direction heat flux along the inner cylinder boundary (b) using linear DG basis functions ($p = 1$) and the structured mesh. The geometry is represented by either a linear Lagrange basis, yielding a piecewise linear approximation of the boundary, or a quadratic NURBS basis, which represents the geometry exactly. Reference lines are plotted in black.

4.2.2. Higher order convergence

In this section, using the test case outlined at the start of Section 4.2, we investigate the convergence behavior of the numerical scheme, in particular with respect to higher order convergence rates and the effect of the Knudsen number thereon. This is done by first considering a sequence of fully curved structured meshes, as done in previous sections, to eliminate any potential errors induced by strongly skewed bad quality elements. Thereafter, a comparison with fully unstructured meshes is made to demonstrate the capability of the scheme to accurately resolve problems with curved boundaries and unstructured meshes.

Figure 4.6 plots the convergence rates of the L_2-error of the temperature distribution for various Knudsen numbers. Note that the differences in the magnitude of the

error can be attributed to the different solution magnitudes in the different cases. For relatively small Knudsen numbers, shown in Figures 4.6a and 4.6b, the observed convergence rates are slightly higher than the corresponding reference lines, indicating optimal convergence. As the Knudsen number increases, the convergence rates of the higher order discretizations slightly decrease, as depicted in 4.6c and 4.6d. The second order scheme performs well in all cases. However, a third order discretization yields a noticeably reduced convergence rate in the $Kn = 1$ case, while the fourth order scheme retains optimal convergence. Nonetheless, the obtained results validate the higher order capabilities of the numerical scheme in solving moment equations in the presence of curved boundaries.



a) $Kn = 0.001$ **b)** $Kn = 0.01$

c) $Kn = 0.1$ **d)** $Kn = 1$

Figure 4.6. Convergence rates of the L_2-errors of the temperature inside the domain for various Knudsen numbers and DG polynomial orders using fully curved structured meshes with a quadratic NURBS geometry representation. Reference lines are plotted in black.

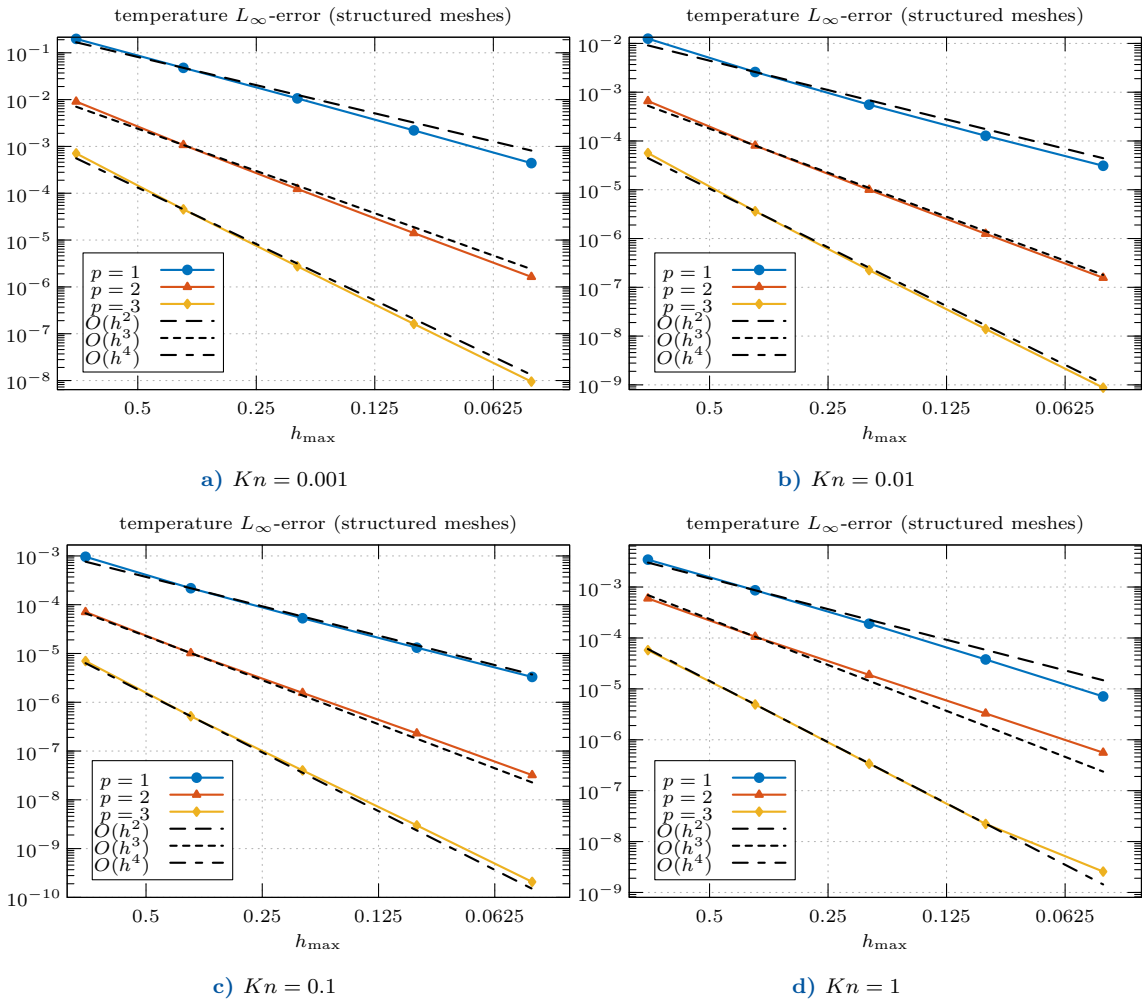
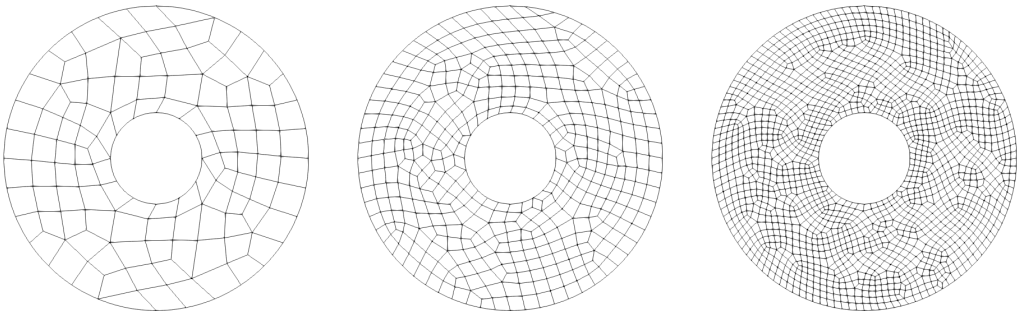


Figure 4.7. Convergence rates of the L_∞-errors of the temperature inside the domain for various Knudsen numbers and DG polynomial orders using fully curved structured meshes with a quadratic NURBS geometry representation. Reference lines are plotted in black.

Similar conclusions can be drawn from the empirical convergence studies of the L_∞-error shown in 4.7. For all cases, except the third order case in Figure 4.7d, the convergence rates are near optimal. Note that the simulation of higher Knudsen number problems took more computational time. Furthermore, for the combination of the higher order schemes and high Knudsen numbers, the linear solver, described in Section 3.4.2, exhibited difficulties converging to the desired tolerance. In particular the fourth order case, depicted in 4.7d, shows a deviation of the reference line in the last point. This is due to the error in the solution of the linear solver being larger than the discretization error of the spatial discretization. For this reason, the time marching approach is used to further lower the residual after the linear solve, in

order to obtain a better converged solution. However, this is computationally quite expensive. Due to the above considerations, the deviation of the convergence rate from the fourth order reference line is attributed to a lack of convergence of the time integration procedure. The obtained results stress the need for better preconditioners and iterative solvers.

To verify the numerical scheme for general unstructured meshes on curved domains, the structured meshes of the previous computations are replaced with fully unstructured quadrilateral meshes generated with Gmsh [56]. The first three of the four consecutively finer meshes used in the following are shown in Figure 4.8. Note that for the coarser meshes, depicted in Figures 4.8a and 4.8b some elements along the outer cylinder boundary have two edges on the boundary. As a result of the curved geometry approximation, both edges belong to the same curve. Preliminary numerical experiments showed that a nodal DG basis is more sensitive to the mesh quality than the employed modal basis, in particular with respect to these kinds of elements.

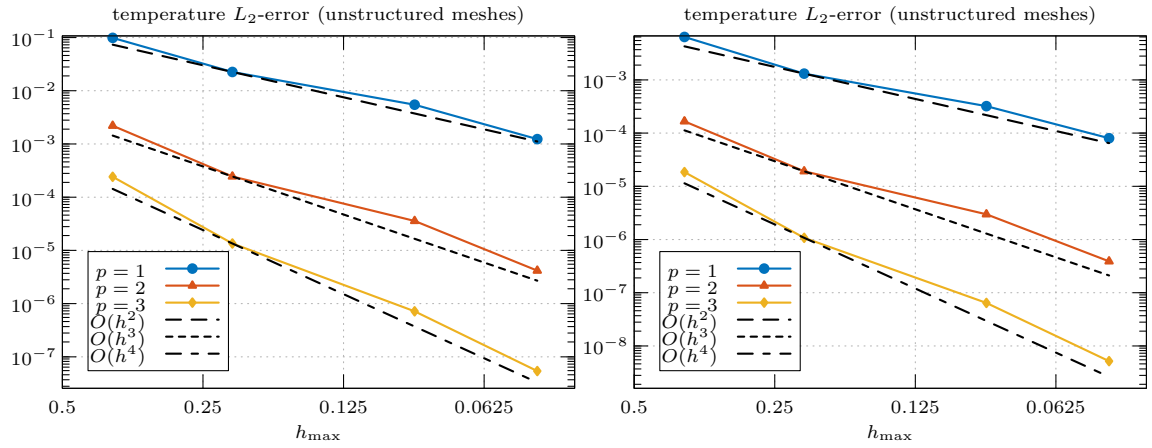


a) 110 elements **b)** 444 elements **c)** 1703 elements

Figure 4.8. Unstructured meshes for the double cylinder (annulus) test case. Generated with Gmsh.

The convergence rates of the L_2-error of the temperature distribution inside of the computational domain is graphed in Figure 4.9. The overall behavior of the convergence rates is similar to the one observed using the structured meshes. A noticeable difference is the kink in the lines caused by the third data point. Since this is present in the convergence rates of of all DG polynomial orders and Knudsen numbers, this can be attributed to changes in the meshes themselves. Also, note that the kink is more pronounced for the higher Knudsen number cases shown in Figures 4.9c and 4.9d, indicating a higher sensitivity with respect to the mesh quality in these cases. Comparing the magnitude of the errors in Figure 4.9 to those obtained with structured meshes, depicted in Figure 4.6, shows that an unstructured mesh entails significantly higher errors, which is well known, see for example [5]. Lastly, note that for h-adaptive simulations, a coarse initial mesh needs to be selected. It is therefore

of relevance to know how coarse the initial mesh can be without jeopardizing the overall solution accuracy at later refinement stages. A detailed investigation of this is left for future work.



a) $Kn = 0.001$ **b)** $Kn = 0.01$

c) $Kn = 0.1$ **d)** $Kn = 1$

Figure 4.9. Convergence rates of the L_2-errors of the temperature inside the domain for various Knudsen numbers and DG polynomial orders using unstructured meshes generated with Gmsh. The curved boundary geometry is imposed only on the boundary elements using a quadratic NURBS geometry representation. Reference lines are plotted in black.

In the context of the preceding convergence studies, the DG formulation itself and its implementation are considered to be verified. Furthermore, their suitability to solve moment equations on unstructured domains with curved boundaries while retaining higher order convergence rates is demonstrated, satisfying the design goals outlined in Chapter 1.

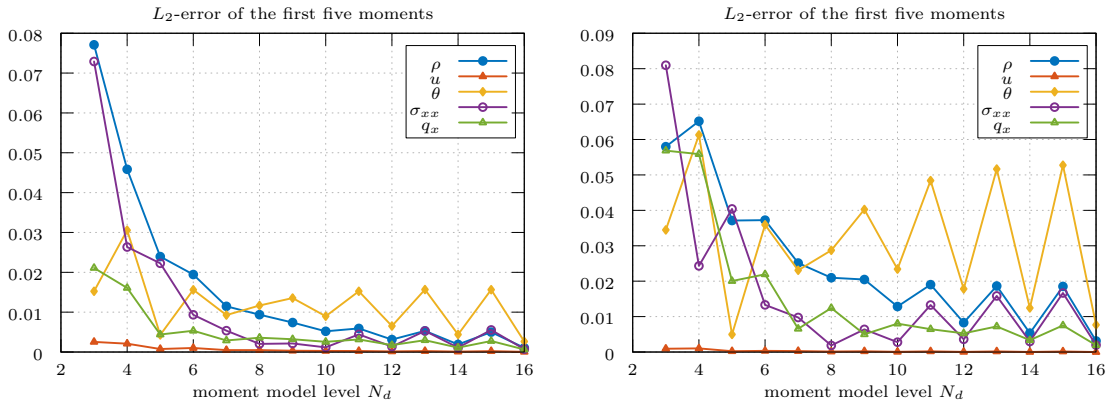
4.3. Model convergence

For model-adaptivity to work, the selected hierarchy of models needs to converge as the model level increases. Therefore, we briefly investigate the model error convergence to guide the appropriate selection of a moment hierarchy. More detailed investigations can be found in [8, 27]. The setup uses is the one-dimensional heat conduction between two plates, inspired by [27]. The solved equations are ordered theory moment equation, detailed in Section 2.4. The computational domain is $\Omega = [x_{\min}, x_{\max}] = [-0.5, 0.5]$ and the boundary condition parameters read as

$$\theta^w|_{x_{\min}} = -1, \quad \theta^w|_{x_{\max}} = 1, \quad \rho^w = u^w = 0, \tag{4.8}$$

where ρ^w and u^w are the boundary density and velocity respectively. The accommodation coefficient of the first boundary condition is chosen as $\tilde{\chi}_\rho = 0.1$ and $\tilde{\chi} = 1.0$ for the remaining boundary conditions. The solutions are computed using a fifth-order scheme ($p = 5$) and 64 elements.

Figure 4.10 plots the L_2-errors of the first five moments over the increasing model level. All moments show oscillatory convergence behavior, which matches the results obtained in [8, 27]. Furthermore, comparing Figures 4.10a and 4.10b shows that the convergence is worse for higher Knudsen numbers. Note that the efficiency of model-adaptive schemes relies on a significant reduction of error when increasing the model level. Additionally, the non-uniform convergence can interfere with error estimators. For this reason, one may choose to only employ a hierarchy consisting of every second model level, as done in [8].

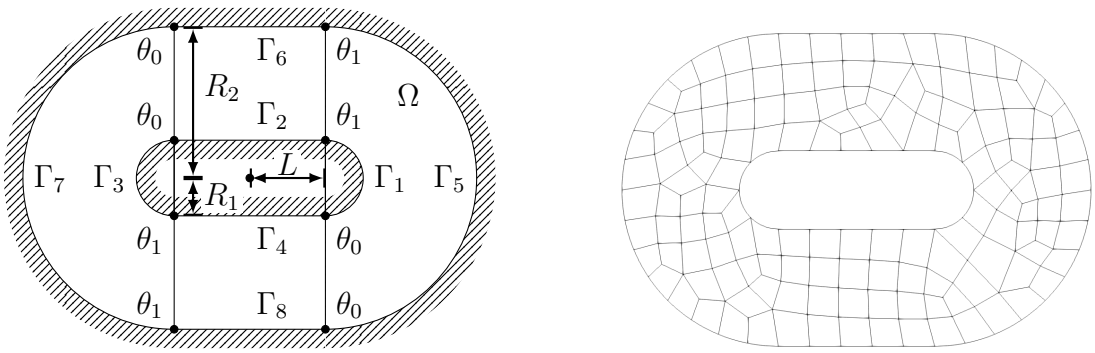


a) Kn = 0.1 **b)** Kn = 1

Figure 4.10. Model error of the first five moments for increasing model level N_d of ordered theory moment systems. The considered test case is the one-dimensional heat conduction between two plates. The errors are computed with respect to the $N_d = 20$ results. For convenience, the moments are denoted by their respective fluid dynamics quantities, see Equation (2.62). However, no scaling is applied.

5. Example: Thermal transpiration flow in a Knudsen pump

In this chapter, the *hpm*-adaptive capabilities of the present DG formulation and its implementation are demonstrated based on a simple numerical experiment in which an initial solution is iteratively refined. The selected example application is a thermal transpiration flow inside of a Knudsen pump along the lines of [79, 80]. A sketch of the computational domain and initial coarse mesh is depicted in Figure 5.1.



a) Knudsen pump geometry [79]. **b)** Initial mesh with 144 elements and $h_{\max} = 0.46$.

Figure 5.1. Sketch of the Knudsen pump geometry (a) and resulting coarse initial mesh (b) for the thermal transpiration flow test case. Linear temperature profiles from θ_0 to θ_1 are prescribed along all boundary segments Γ_i with $i \in \{1, \dots, 8\}$, yielding the same continuous temperature profile along the the inner and outer boundaries. The mesh employs a quadratic NURBS element geometry basis, which constitutes an exact representation of the curved geometry. The geometry sketch is taken from [79] and the overall application case is inspired by [79, 80].

Following [79], the geometric parameters are defined as $L = 1$, $R_1 = 1/2$ and $R_2 = 2$. Furthermore, expressions for the wall temperature $\theta^w(x, y)$, which yield linear profiles between the two temperatures $\theta_0 = 0.5$ and $\theta_1 = 1.5$, are given in Table 5.1. To model an impermeable wall, the other boundary data is set to zero and the accommodation coefficient $\tilde{\chi}$ is set to 0.001 for the first and to 1.0 for all other boundary conditions. The Knudsen number is chosen as $\mathrm{Kn} = 0.1$. The employed model hierarchy are the ordered theory equations introduced in Section 2.4. Due to the oscillating model convergence discussed in Section 4.3 only every second model is selected, meaning

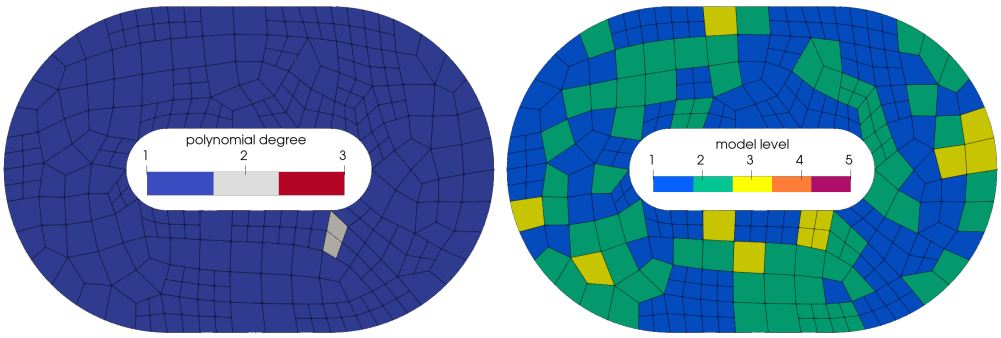
that the individual models are defined by $N_d = 2m + 1$ m, where $m \in \mathbb{N}$ is the model level.

Γ_i |	Γ_1	Γ_2	Γ_3	Γ_4
$\theta(x,y)\vert_{\Gamma_i}$ |	$\frac{1}{2}\frac{2}{\pi}\,\mathrm{atan2}(x-1,y)+1$	$\frac{1}{2}x+1$	$-\frac{1}{2}\frac{2}{\pi}\,\mathrm{atan2}(1-x,y)+1$	$-\frac{1}{2}x+1$

Table 5.1. Knudsen pump temperature boundary condition expressions. Taken from [79].

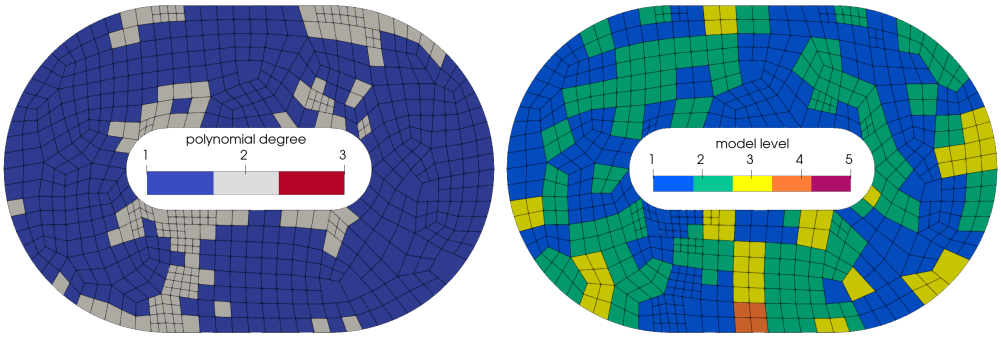
To complete the adaptive scheme, a refinement indicator and iteration procedure need to be specified. For the simulation performed in this section, a very rudimentary method is defined as follows. For any given solution, the element-wise discretization error in a quantity of interest, i.e. the temperature, is estimated by computing the difference between the current solution and a solution computed with the polynomial degree of all elements elevated by one. Analogously, the model error is computed by means of comparison to a once model-refined solution. Then in each refinement iteration step, the element-wise model and discretization errors are computed. Similarly to [8], the $\lfloor \epsilon N \rfloor$ elements with the highest combined error are either m-refined or hp-refined, depending on whether the model or discretization error is larger. Here, N is the total number of elements and ϵ is the percentage of elements refined in each step, chosen as $\epsilon = 0.2$. Furthermore, an artificial weighting of 0.4 of the element model errors is introduced, to induce a more balanced refinement between model and discretization in the herein considered case. For the refinement of the spatial discretization, the decision between h- and p-refinement is made based on whichever level is lower. Lastly, note that the above procedure is for demonstration purposes only and is neither an efficient nor effective method. More sophisticated procedures available in the literature have shown promising results, see for example [8]. The development and implementation of indicators and refinement algorithms is left for future work.

Figure 5.2 depicts the local polynomial degree and model level after every three steps of the adaptive refinement procedure applied to the Knudsen pump test case. Note also the adaptively refined mesh in every step and the resulting non-conforming interfaces. All local refinement levels do not exhibit an apparent structure, presumably do to the low quality of the employed indicator. Nonetheless, the figure clearly demonstrates the capability of the numerical scheme to handle fully hpm-adaptive discretizations on unstructured meshes with curved boundaries.



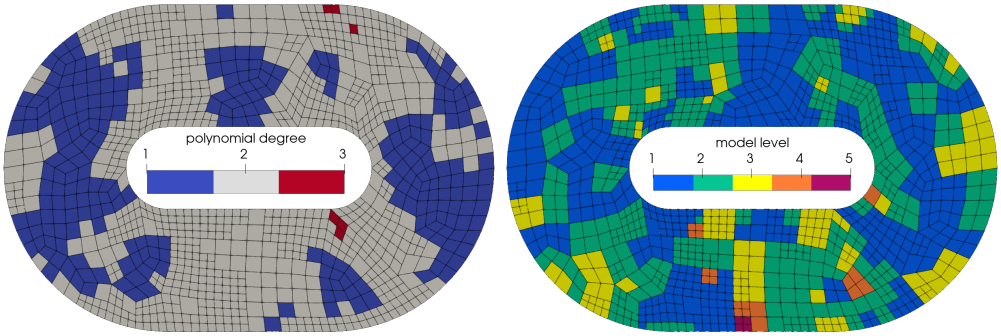
a) polynomial degree after 3 refinements

b) model level after 3 refinements



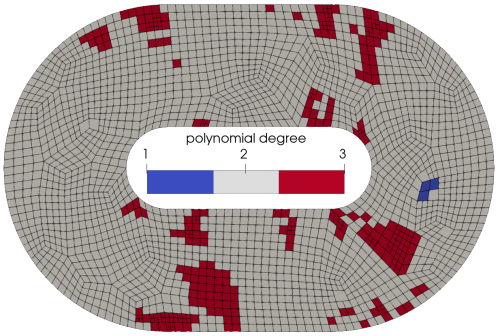
c) polynomial degree after 6 refinements

d) model level after 6 refinements

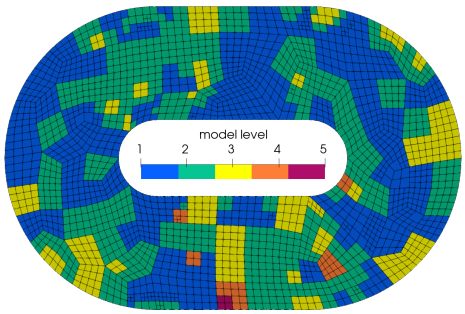


e) polynomial degree after 9 refinements

f) model level after 9 refinements



g) polynomial degree after 12 refinements



h) model level after 12 refinements

Figure 5.2. Element-wise DG basis polynomial degrees (left column) and model levels (right column) over the course of 12 refinement steps.

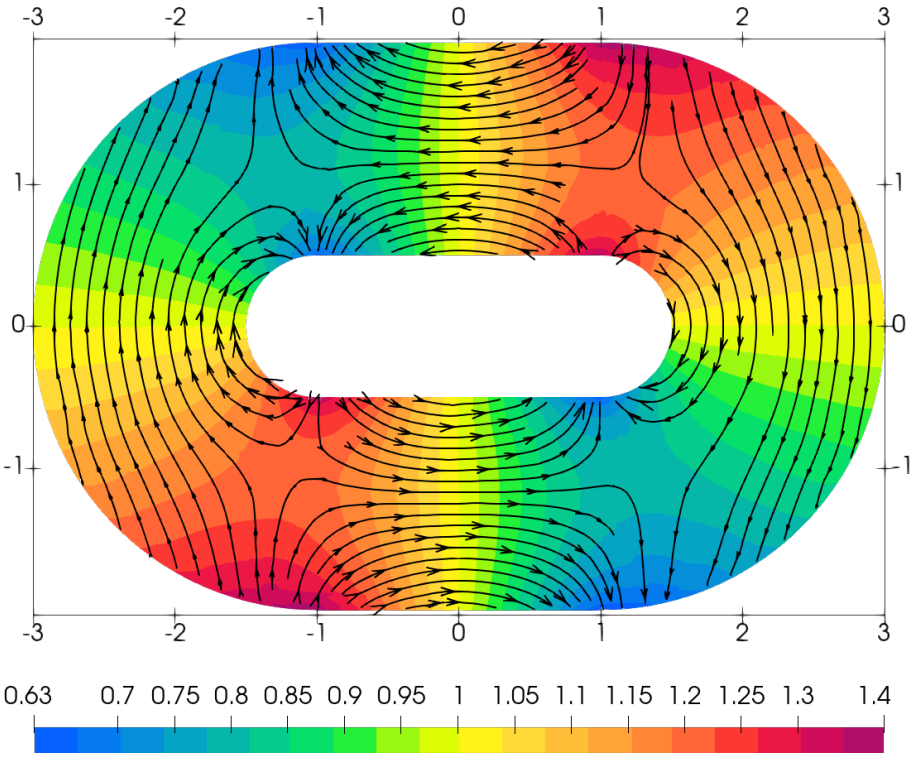


Figure 5.3. Temperature θ and heat flux q_i streamlines of the solution after 12 adaptive refinement steps.

Figure 5.3 shows the temperature distribution and the heat flux streamlines for the solution obtained after 12 steps of the adaptive procedure. Note that for convenience the temperature and heat flux are taken to be equal to their respective moments defined by Equation (2.62). While a detailed investigation of the error and resulting convergence rates of the adaptive scheme is left for future work, the obtained solution is in qualitative agreement with the one shown in [79]. Consequently, accurate results can be obtained by means of the model-adaptive DG formulation presented in this work.

6. Conclusion and future work

In this work, a model-adaptive high-order DG formulation on unstructured meshes was presented. Furthermore, resulting numerical scheme is equipped with a general boundary condition formulation, *hp*-adaptive capabilities by means of a mortar method and higher order curved geometry representations. All methods were implemented in Julia, yielding in a prototype research code, facilitating adaptive simulations of hierarchical moment equations. Model problems derived from kinetic gas theory were presented and their well-posedness discussed. Empirical convergence studies were conducted to verify the developed code. High-order convergence rates were obtained, in particular for unstructured meshes and curved boundaries with general boundary conditions. Emphasis was put on the representation of curved geometries by means of either a Lagrange or NURBS basis for the element geometry. It was found that even for a linear DG basis, employing a curved geometry representation is beneficial for the stability and accuracy of the scheme. Using a NURBS geometry representation was found to be a promising approach. The described capabilities of the numerical formulation were demonstrated by simulating the thermal transpiration flow inside of a Knudsen pump. The scheme was observed to be stable and converge in the presence of local mesh, DG polynomial degree and in particular model refinement.

Suggestions for future work are roughly divided into the areas of model, numerics and implementation development. For a model-adaptive scheme, the convergence behavior of the employed models is highly important. For this reason, building upon the literature [8, 27], the convergence behavior of the models employed herein could be further explored. An extension of the models to the nonlinear case or to regularized equations with parabolic terms is recommended to enable the simulation of a broader class of problems. With respect to the numerical scheme, the results obtained herein stress the need for robust, accurate and efficient solvers. Possible options are multigrid [62] and in particular matrix-free methods [40], both of which are well suited for the present matrix-free hierarchical framework. Refinement criterions are crucial for the success of the adaptive scheme. Consequently different methods, such as adjoint based adaptivity [8], may be explored. Considering the implementation, the integration of openly available high-performance libraries, such as p4est [64] and deal.II [13] into the current code is recommended for better performance, sustainability and shareability. We envision a automated solution framework to solve complex application cases where local adjustment of the model fidelity is paramount to obtain accurate results at reasonable computational cost.

Bibliography

[1] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1 (2017), pp. 65–98.

[2] H. Struchtrup. "Macroscopic transport equations for rarefied gas flows". In: *Macroscopic Transport Equations for Rarefied Gas Flows: Approximation Methods in Kinetic Theory*. Ed. by H. Struchtrup. Berlin, Heidelberg: Springer, 2005, pp. 145–160.

[3] M. Torrilhon. "Modeling Nonequilibrium Gas Flow Based on Moment Equations". In: *Annual Review of Fluid Mechanics* 48.Volume 48, 2016 (2016), pp. 429–458.

[4] H. Struchtrup and P. Taheri. "Macroscopic transport models for rarefied gas flows: a brief review". In: *IMA Journal of Applied Mathematics* 76.5 (2011), pp. 672–697.

[5] M. Torrilhon and N. Sarna. "Hierarchical Boltzmann simulations and model error estimation". In: *Journal of Computational Physics* 342 (2017), pp. 66–84.

[6] H. Grad. "On the kinetic theory of rarefied gases". In: *Communications on Pure and Applied Mathematics* 2.4 (1949), pp. 331–407.

[7] A. Westerkamp and M. Torrilhon. "Finite element methods for the linear regularized 13-moment equations describing slow rarefied gas flows". In: *Journal of Computational Physics* 389 (2019), pp. 1–21.

[8] N. Sarna. "Entropy stable hermite approximations of the Boltzmann equation". PhD Thesis. Dissertation, RWTH Aachen University, 2019, 2019.

[9] M. R. A. Abdelmalik and E. H. van Brummelen. "Error estimation and adaptive moment hierarchies for goal-oriented approximations of the Boltzmann equation". In: *Computer Methods in Applied Mechanics and Engineering* 325 (2017), pp. 219–239.

[10] F. Bassi and S. Rebay. "High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations". In: *Journal of Computational Physics* 138.2 (1997), pp. 251–285.

[11] B. Cockburn. "Discontinuous Galerkin methods". In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 83.11 (2003), pp. 731–754.

[12] N. Krais, A. Beck, T. Bolemann, H. Frank, D. Flad, G. Gassner, F. Hindenlang, M. Hoffmann, T. Kuhn, M. Sonntag, and C.-D. Munz. "FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws". In: *Computers & Mathematics with Applications*. Development and Application of Open-source Software for Problems with Numerical PDEs 81 (2021), pp. 186–219.

[13] D. Arndt, W. Bangerth, B. Blais, T. C. Clevenger, M. Fehling, A. V. Grayver, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, R. Rastak, I. Tomas, B. Turcksin, Z. Wang, and D. Wells. "The deal.II library, Version 9.2". In: *Journal of Numerical Mathematics* 28.3 (2020), pp. 131–146.

[14] H. Ranocha, M. Schlottke-Lakemper, A. R. Winters, E. Faulhaber, J. Chan, and G. J. Gassner. "Adaptive numerical simulations with Trixi.jl: A case study of Julia for scientific computing". In: *JuliaCon Proceedings* 1.1 (2022), p. 77.

[15] N. Sarna and M. Torrilhon. "On Stable Wall Boundary Conditions for the Hermite Discretization of the Linearised Boltzmann Equation". In: *Journal of Statistical Physics* 170.1 (2018), pp. 101–126.

[16] E. Godlewski and P.-A. Raviart. *Numerical Approximation of Hyperbolic Systems of Conservation Laws*. Vol. 118. Applied Mathematical Sciences. New York, NY: Springer, 2021.

[17] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. 2002.

[18] C. Hirsch. *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Elsevier, 2007.

[19] K. O. Friedrichs and P. D. Lax. "Systems of Conservation Equations with a Convex Extension". In: *Proceedings of the National Academy of Sciences* 68.8 (1971), pp. 1686–1688.

[20] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[21] I. Müller and W. Weiss. "Thermodynamics of irreversible processes — past and present". In: *The European Physical Journal H* 37.2 (2012), pp. 139–236.

[22] H. Struchtrup and M. Torrilhon. "Regularization of Grad's 13 moment equations: Derivation and linear analysis". In: *Physics of Fluids* 15.9 (2003), pp. 2668–2680.

[23] J. Nordström. "A Roadmap to Well Posed and Stable Problems in Computational Physics". In: *Journal of Scientific Computing* 71.1 (2017), pp. 365–385.

[24] H.-O. Kreiss and J. Lorenz. *Initial-Boundary Value Problems and the Navier-Stokes Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2004.

[25] J. Nordström and M. Svärd. "Well-Posed Boundary Conditions for the Navier–Stokes Equations". In: *SIAM Journal on Numerical Analysis* 43.3 (2005), pp. 1231–1255.

[26] N. Gerhard, S. Müller, and W. Dahmen. "An adaptive multiresolution discontinuous Galerkin scheme for conservation laws". PhD thesis. Aachen, 2017.

[27] M. Torrilhon. "Convergence Study of Moment Approximations for Boundary Value Problems of the Boltzmann-BGK Equation". In: *Communications in Computational Physics* 18.3 (2015), pp. 529–557.

[28] J. Bünger, E. Christhuraj, A. Hanke, and M. Torrilhon. "Structured derivation of moment equations and stable boundary conditions with an introduction to symmetric, trace-free tensors". In: *Kinetic and Related Models* 16.3 (2023), pp. 458–494.

[29] M. Torrilhon, J. Au, and H. Struchtrup. "Explicit fluxes and productions for large systems of the moment method based on extended thermodynamics". In: *Continuum Mechanics and Thermodynamics* 15.1 (2003), pp. 97–111.

[30] M. R. A. Abdelmalik and E. H. van Brummelen. "An entropy stable discontinuous Galerkin finite-element moment method for the Boltzmann equation". In: *Computers & Mathematics with Applications*. Finite Elements in Flow Problems 2015 72.8 (2016), pp. 1988–1999.

[31] N. Sarna and M. Torrilhon. "Entropy stable Hermite approximation of the linearised Boltzmann equation for inflow and outflow boundaries". In: *Journal of Computational Physics* 369 (2018), pp. 16–44.

[32] M. Torrilhon and H. Struchtrup. "Boundary conditions for regularized 13-moment-equations for micro-channel-flows". In: *Journal of Computational Physics* 227.3 (2008), pp. 1982–2011.

[33] A. S. Rana and H. Struchtrup. "Thermodynamically admissible boundary conditions for the regularized 13 moment equations". In: *Physics of Fluids* 28.2 (2016).

[34] B. Cockburn and C.-W. Shu. "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework". In: *Mathematics of Computation* 52.186 (1989), pp. 411–435.

[35] B. Cockburn and C.-W. Shu. "The Runge–Kutta Discontinuous Galerkin Method for Conservation Laws V: Multidimensional Systems". In: *Journal of Computational Physics* 141.2 (1998), pp. 199–224.

[36] C.-W. Shu. "Discontinuous Galerkin methods: general approach and stability". In: *Numerical solutions of partial differential equations* 201 (2009), pp. 1–44.

[37] B. Cockburn. "Discontinuous Galerkin Methods for Convection-Dominated Problems". In: *High-Order Methods for Computational Physics*. Ed. by T. J. Barth and H. Deconinck. Berlin, Heidelberg: Springer, 1999, pp. 69–224.

[38] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. "The Development of Discontinuous Galerkin Methods". In: *Discontinuous Galerkin Methods*. Ed. by M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick, B. Cockburn, G. E. Karniadakis, and C.-W. Shu. Vol. 11. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 3–50.

[39] B. Cockburn and C.-W. Shu. "Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems". In: *Journal of Scientific Computing* 16.3 (2001), pp. 173–261.

[40] D. Arndt, N. Fehn, G. Kanschat, K. Kormann, M. Kronbichler, P. Munch, W. A. Wall, and J. Witte. "ExaDG: High-Order Discontinuous Galerkin for the Exa-Scale". In: *Software for Exascale Computing - SPPEXA 2016-2019*. Ed. by H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. E. Nagel. Cham: Springer International Publishing, 2020, pp. 189–224.

[41] M. Kronbichler and K. Kormann. "Fast Matrix-Free Evaluation of Discontinuous Galerkin Finite Element Operators". In: *ACM Trans. Math. Softw.* 45.3 (2019), 29:1–29:40.

[42] J. Donea and A. Huerta. *Finite element methods for flow problems*. John Wiley & Sons, 2003.

[43] S. Müller. "Finite Volume and Finite Element Methods". Institut für Geometrie und Praktische Mathematik, 2024.

[44] B. Cockburn. "Discontinuous Galerkin Methods for Computational Fluid Dynamics". In: *Encyclopedia of Computational Mechanics*. John Wiley & Sons, Ltd, 2004.

[45] J. A. Cottrell, T. J. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[46] M. Kronbichler, S. Schoeder, C. Müller, and W. A. Wall. "Comparison of implicit and explicit hybridizable discontinuous Galerkin methods for the acoustic wave equation". In: *International Journal for Numerical Methods in Engineering* 106.9 (2016), pp. 712–739.

[47] S. Duczek and H. Gravenkamp. "Mass lumping techniques in the spectral element method: On the equivalence of the row-sum, nodal quadrature, and diagonal scaling methods". In: *Computer Methods in Applied Mechanics and Engineering* 353 (2019), pp. 516–569.

[48] D. A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*. Scientific Computation. Dordrecht: Springer Netherlands, 2009.

[49] S. A. Orszag. "Spectral methods for problems in complex geometries". In: *Journal of Computational Physics* 37.1 (1980), pp. 70–92.

[50] M. Kronbichler, K. Kormann, I. Pasichnyk, and M. Allalen. "Fast Matrix-Free Discontinuous Galerkin Kernels on Modern Computer Architectures". In: *High Performance Computing*. Ed. by J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes. Vol. 10266. Cham: Springer International Publishing, 2017, pp. 237–255.

[51] W. E. Roth. "On direct product matrices". In: *Bulletin of the American Mathematical Society* 40.6 (1934), pp. 461–468.

[52] A. Airola and T. Pahikkala. "Fast Kronecker Product Kernel Methods via Generalized Vec Trick". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.8 (2018), pp. 3374–3387.

[53] V. V. Rusanov. "The calculation of the interaction of non-stationary shock waves and obstacles". In: *USSR Computational Mathematics and Mathematical Physics* 1.2 (1962), pp. 304–320.

[54] J. Nordström. "The influence of open boundary conditions on the convergence to steady state for the Navier-Stokes equations". In: *Journal of Computational Physics* 85.1 (1989), pp. 210–244.

[55] L. Friedrich, A. R. Winters, D. C. Del Rey Fernández, G. J. Gassner, M. Parsani, and M. H. Carpenter. "An Entropy Stable h / p Non-Conforming Discontinuous Galerkin Method with the Summation-by-Parts Property". In: *Journal of Scientific Computing* 77.2 (2018), pp. 689–725.

[56] C. Geuzaine and J.-F. Remacle. "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities". In: *International Journal for Numerical Methods in Engineering* 79.11 (2009), pp. 1309–1331.

[57] R. Sevilla, S. Fernández-Méndez, and A. Huerta. "NURBS-Enhanced Finite Element Method (NEFEM)". In: *Archives of Computational Methods in Engineering* 18.4 (2011), pp. 441–484.

[58] M. Montanari, G. M. Santi, R. Sevilla, L. Alfredo, and N. Petrinic. "NURBS-enhanced finite element method (NEFEM) on quadrilateral meshes". In: *Finite Elements in Analysis and Design* 231 (2024), p. 104099.

[59] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[60] C.-W. Shu and S. Osher. "Efficient implementation of essentially non-oscillatory shock-capturing schemes". In: *Journal of Computational Physics* 77.2 (1988), pp. 439–471.

[61] G. L. Sleijpen and D. R. Fokkema. "BiCGstab (l) for linear equations involving unsymmetric matrices with complex spectrum". In: *Electronic Transactions on Numerical Analysis* 1.11 (1993), p. 2000.

[62] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid methods*. Academic press, 2001.

[63] D. A. Kopriva. "A Conservative Staggered-Grid Chebyshev Multidomain Method for Compressible Flows. II. A Semi-Structured Method". In: *Journal of Computational Physics* 128.2 (1996), pp. 475–488.

[64] C. Burstedde, L. C. Wilcox, and O. Ghattas. "p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees". In: *SIAM Journal on Scientific Computing* 33.3 (2011), pp. 1103–1133.

[65] W. Dahmen and A. Reusken. *Numerik für Ingenieure und Naturwissenschaftler: Methoden, Konzepte, Matlab-Demos, E-Learning*. Berlin, Heidelberg: Springer, 2022.

[66] S. Gowda, Y. Ma, A. Cheli, M. Gwóźdź, V. B. Shah, A. Edelman, and C. Rackauckas. "High-Performance Symbolic-Numerics via Multiple Dispatch". In: *ACM Commun. Comput. Algebra* 55.3 (2022), pp. 92–96.

[67] A. Townsend. *FastGaussQuadrature.jl*. Version 1.0.2. 2024.

[68] *LoopVectorization.jl*. Version 0.12.172. 2025.

[69] *Polyester.jl*. Version 0.7.16. 2024.

[70] *StaticArrays.jl*. Version 1.9.13. 2025.

[71] *LinearMaps.jl*. Version 3.11.4. 2025.

[72] *IterativeSolvers.jl*. Version 0.9.4. 2024.

[73] J. Ahrens, B. Geveci, and C. Law. "ParaView: An End-User Tool for Large Data Visualization". In: *Visualization Handbook*. Butterworth-Heinemann, 2005, pp. 717–731.

[74] J. I. Polanco. *WriteVTK.jl: a Julia package for writing VTK XML files*. Version 1.10.1. 2021.

[75] T. Williams, C. Kelley, C. Bersch, H.-B. Bröker, J. Campbell, R. Cunningham, D. Denholm, G. Elber, R. Fearick, C. Grammes, and many others. *gnuplot 6.0*. 2024.

[76] M. Torrilhon. "Slow gas microflow past a sphere: Analytical solution based on moment equations". In: *Physics of Fluids* 22.7 (2010), p. 072001.

[77] A. Westerkamp and M. Torrilhon. "Slow rarefied gas flow past a cylinder: Analytical solution in comparison to the sphere". In: *AIP Conference Proceedings* 1501.1 (2012), pp. 207–214.

[78] A. Westerkamp and M. Torrilhon. "Curvature-induced instability of a Stokes-like problem with non-standard boundary conditions". In: *Applied Numerical Mathematics* 121 (2017), pp. 96–114.

[79] L. Theisen and M. Torrilhon. "fenicsR13: A Tensorial Mixed Finite Element Solver for the Linear R13 Equations Using the FEniCS Computing Platform". In: *ACM Trans. Math. Softw.* 47.2 (2021), 17:1–17:29.

[80] K. Aoki, P. Degond, L. Mieussens, M. Nishioka, and S. Takata. "Numerical simulation of a Knudsen pump using the effect of curvature of the channel". In: *Rarefied Gas Dynamics. MS Ivanov and AK Rebrov, Eds. Novosibirsk* (2007), pp. 1079–1084.