KARLSRUHE INSTITUTE OF TECHNOLOGY

DOCTORAL THESIS

Realizability-preserving discretization strategies for hyperbolic and kinetic equations with uncertainty

Author:
Jonas KUSCH

Supervisors:
Prof. Dr. Martin FRANK
Prof. Dr. Ryan MCCLARREN
Prof. Dr. Christian WIENERS

A thesis submitted in fulfillment of the requirements for the degree of Dr. rer. nat.

in the

Department of Mathematics

Karlsruhe Institute of Technology

Date of oral exam: 06.05.2020

Realizability-preserving discretization strategies for hyperbolic and kinetic equations with uncertainty

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN

von der KIT-Fakultät für Mathematik des
Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von

Jonas Kusch

Tag der mündlichen Prüfung: 06.05.2020

1. Referent: Prof. Dr. Martin FRANK
2. Referent: Prof. Dr. Christian WIENERS
3. Referent: Prof. Dr. Ryan MCCLARREN

Acknowledgements

Foremost, I would like to thank my advisor Martin Frank, who gave me the opportunity and freedom to work on various fascinating research topics. Discussions with Martin improved my understanding of different research areas significantly and helped me stay focused as well as motivated throughout my time as a PhD student. He provided me with the right amount of guidance, introduced me to various researchers in my field and created a creative and productive work environment. One of the people Martin introduced me to is Ryan McClarren, who gave me a great amount of new ideas for several projects. The great hospitality of Ryan and his wife Katie created a comfortable and welcoming atmosphere during my research stay at Notre Dame university. Ryan's ability to directly give helpful advise for almost every upcoming problem, ranging from coding issues to questions on certain details in physical models as well as numerical methods, prevented me from getting stuck quite a few times. For all that I am very thankful. Furthermore, I would like to thank Professor Wieners for fruitful discussions concerning my work on uncertainty quantification and for spending time reading my thesis. His comments and suggestions gave me a more theoretical viewpoint on several aspects of my work. Also, I wish to thank Graham Kaland, who greatly helped me to improve writing papers with his critical questions, comments and suggestions. Our long discussions concerning entropy closures enriched my understanding of this method and his ideas on realizability in connection with filters greatly improved this part of my thesis. Furthermore, I want to thank Cory Hauck for sharing his ideas on ray-effect mitigation and his patience when presenting and explaining various concepts from transport theory. Also, I wish to thank Nicolas Gauger and Stephan Schmidt who encouraged me to continue working in the field of shape optimization. Though this work is not part of my thesis, it helped me when deriving acceleration techniques for uncertainty quantification. I always enjoyed visiting Nicolas Gauger's research group in Kaiserslautern since it provided me with refreshing input and prevented me from getting stuck in my own scientific bubble. I would like to thank my colleagues in Karlsruhe for creating a pleasant work atmosphere. Especially, I would like to thank Jannick Wolters and Thomas Camminady for countless discussions on various interesting projects, which greatly improved the results of this thesis. Furthermore, I want to thank Lisa Kusch for reading and discussing my thesis as well as her hospitality whenever I visited Kaiserslautern. Lastly, I want to thank Hanna Schulz for her moral support, which has been an essential factor during the last years.

Preface

A vast amount of physical and engineering applications require the solution of partial differential equations (PDEs), which describe the dynamics of various physical phenomena. In applications of practical interest, these PDEs cannot be solved analytically, meaning that the solution has to be approximated through a computer simulation. Approximating the solution requires the construction of efficient numerical methods, which broadly speaking boils down to finding a satisfactory, finite-dimensional description of the solution on its phase space (i.e. the domain on which the solution is defined), as well as a set of equations describing the behavior of this finite-dimensional solution representation. This work studies two types of problems with high-dimensional phase spaces which require a carefully chosen discretization: First, hyperbolic problems which are subject to uncertainties are considered. Uncertainties, which can for example arise through measurement errors in input data, are commonly modeled by incorporating additional parameters in the solution's phase space. Second, the discretization of kinetic equations, which in addition to space and time depend on angular variables is studied. Deriving efficient discretization techniques which preserve physical solution bounds while not showing spurious solution artifacts poses a challenge for both of these problems.

A frequently used discretization strategy is to represent the solution by a finite number of polynomials with corresponding expansion coefficients. A discretization method that makes use of such polynomial representations is called *modal* method. In the context of uncertainty quantification, a popular modal discretization strategy is the *stochastic-Galerkin* (SG) method, whereas kinetic theory uses the name *spherical harmonics* (P_N) method. If the solution possesses sufficient regularity, polynomial approximations lead to satisfactory results, i.e. a small number of expansion coefficients suffices to obtain an adequate representation of the solution. However, when the solution lacks this regularity, polynomial approximations show spurious oscillations, which do not only yield poor approximation results, but also destroy fundamental solution properties. For instance, when the solution leaves an admissible set —which in the case of fluid dynamics equations could mean that the fluid has a negative density— one speaks of the loss of realizability or admissibility. Methods, which commonly preserve realizability, are *nodal* methods, which rely on a pointwise description of the solution. In the field of uncertainty quantification, a popular nodal method, which represents the solution on a quadrature set, is *Stochastic Collocation* (SC), whereas kinetic theory uses the name *discrete ordinates* (S_N) method. Despite preserving realizability, nodal methods tend to show imprints of the chosen quadrature set on the solution, leading to an unsatisfactory approximation.

The aim of this thesis is to investigate and develop discretization methods, which preserve realizability while yielding an adequate solution approximation. A modal method, which aims at preserving realizability, is the Intrusive Polynomial Moment (IPM) method [112] for uncertainty quantification and M_N method [95, 80] for kinetic theory. The main idea is to perform the polynomial approximation on the so-called entropy variables instead of the conserved solution quantities. For scalar problems, the IPM method can be used to force the solution to stay within prescribed bounds. Furthermore, the resulting evolution equations inherit important properties of the original, non-discretized problem. The main challenge of these methods is computational, since they require repeatedly solving an optimization problem. Furthermore, common discretizations of the IPM method can violate realizability

conditions, meaning that the computed moment vectors do not belong to a realizable solution, which is an often occurring problem of minimal entropy methods. In addition to that, the discretized solution can still oscillate within the bounds enforced by the method. We tackle these challenges in Chapter 2, which is based on [67, 69]. Here, we propose two strategies to construct an IPM algorithm which ensures realizability for first- and second order methods. Additionally, we study the effects the chosen entropy has on the solution quality and propose using an entropy which ensures a maximum-principle while yielding non-oscillatory solution approximations. To tackle the computational costs of minimal entropy methods, we propose to adaptively choose between cheap but inaccurate nodal and expensive but accurate modal minimal entropy updates in every spatial cell in Chapter 3, based on [68]. In this chapter, we also introduce a new basis to represent the numerical solution, which facilitates solving the IPM optimization problem. Chapter 4, which is based on [71], extends the realizability preserving IPM algorithm to systems. To cope with increased numerical costs, we propose acceleration techniques for intrusive methods and give a comparison with non-intrusive collocation methods. Moving from scalar equations to systems complicates the choice of admissible entropies, yielding oscillatory solutions for both, the IPM and SG method. To dampen oscillations for SG, we propose to apply filters in Chapter 5, based on [70]. Despite satisfactory solutions of the filtered SG method, we no longer maintain realizability, which is why filters are combined with the IPM method in Chapter 6, yielding non-oscillatory and realizable solution approximations. This chapter is based on unpublished work that has been developed together with Graham Kaland, Martin Frank and Ryan McClarren. In Chapter 7, we start looking at problems from transport theory which are prone to uncertain input parameters. Here, we apply the P_N method to discretize angular variables and then use the IPM method to obtain a finite-dimensional description of the uncertain domain of the derived P_N system. This chapter is based on unpublished work, which has been conducted together with Ryan McClarren. As previously mentioned, a method preserving realizablity of kinetic equations is the S_N method, which however yields non-physical artifacts, commonly called ray-effects. In Chapter 8, we investigate two strategies in order to mitigate ray-effects, based on [17, 36]. The first method, called rotated S_N (rS_N) proposes to add a random rotation of the chosen quadrature set to the S_N method. The second method, called artificial scattering S_N (asS_N) adds an artificial, forward-peaked scattering operator to the original S_N equations. The discussion of the rS_N method mainly focuses on a modified equation analysis. For the asS_N method we will propose an implicit discretization strategy. Chapter 9 uses the discussed modal approximation of the entropy variables to discretize the spatial domain. Here, we aim at preserving the entropy dissipation property on a discrete level by choosing implicit time discretizations. This chapter is based on unpublished work with Graham Kaland and Martin Frank. In Chapter 10, we summarize our results, discuss open problems and give an outlook on future work.

The previously mentioned chapters require certain mathematical concepts and fundamentals, which we will summarize in Chapter 1. We start with a discussion of hyperbolic conservation laws in Section 1.1, which serve as model problems for the discretization techniques used throughout this work. A special focus lies on the notion of entropy, which is an important concept used in this thesis: In particular, the IPM method (used from Chapter 2 to Chapter 7) as well as the DG scheme discussed in Chapter 9 are based on a discretization of the entropy variables, which are introduced in Section 1.1.3. Hyperbolicity, which is introduced in Section 1.1.3 will become an important concept for IPM since this method is constructed to yield a

hyperbolic moment system. Section 1.2 discusses space and time discretizations of hyperbolic conservation laws, which will be used for numerical methods throughout this work. The property of monotonicity, presented in Section 1.2.1, will become important when constructing a realizability-preserving scheme in Chapter 2. Bound-preserving limiters, which are presented in Section 1.2.4, are used to extend this scheme to higher order. The derivation of a DG scheme applied to the entropy variables in Chapter 9 relies on entropy dissipation properties presented in Section 1.2.2. Additionally, these properties are used to derive a stable sweeping method for radiation transport in Chapter 8. In Section 1.3, the radiative transfer equation is presented, which lays the foundation of Chapter 7 and Chapter 8. Here, the P_N method from Section 1.3.1 will be used in Chapter 7 and Chapter 8 presentimg ray-effect mitigation techniques for the S_N method as shown in Section 1.3.3. Section 1.4 discusses methods to propagate uncertainties through given hyperbolic equations, which will be the main aim of Chapters 2 to 7. Here, the focus lies on collocation methods (Section 1.4.4) which will especially be used for a comparison between intrusive and non-intrusive methods in Chapter 4. Furthermore, presented tensorized and sparse grid quadratures in Section 1.4.4 will be used for nodal discretizations of intrusive methods as presented in Section 2.3.3 and used for high-dimensional uncertainties in Chapters 4 and 7. Intrusive methods for uncertainty quantification which are presented in Section 1.4.5 play a key role from Chapters 2 to 7.

Contents

Chapter 1

Introduction

1.1 Hyperbolic conservation laws

Hyperbolic conservation laws are the cornerstone of this work. They frequently arise in physical and engineering applications and their tendency to form discontinuities poses various issues, especially for high-dimensional phase spaces. Before introducing such high-dimensional settings, the discussion focuses on conservation laws and their numerical discretization. A special focus will lie on theoretical and numerical concepts that become important throughout this work. Especially the notion of entropy will be introduced in both, continuous and discrete frameworks.

1.1.1 Balance laws and conservation equations

Systems of hyperbolic equations play a key role in various research areas, including Euler equations in fluid dynamics [135], magnetohydrodynamics (MHD) equations in plasma physics [127], and radiation-hydrodynamics in astrophysics [85, 93]. All these applications are based on balance laws, which are frequently used to determine the time evolution of physical quantities. These quantities are governed by an underlying set of particles. To derive balance equations, one starts by choosing a spatial volume $V_0 \subset \mathbb{R}^d$ composed of a certain number of particles. The movement of this volume in time and space is then determined by the movement of these particles in the spatial domain $D \subset \mathbb{R}^d$. Hence if a particle with velocity $\boldsymbol{v} \in \mathbb{R}^d$ moves along a trajectory $\boldsymbol{\Phi}(t, \boldsymbol{x})$, meaning that

$$\frac{d}{dt}\boldsymbol{\Phi}(t, \boldsymbol{x}) = \boldsymbol{v}(\boldsymbol{\Phi}(t, \boldsymbol{x})) \ ,$$

the time evolution of the so-called material volume V_t is

$$V_t = \{\boldsymbol{\Phi}_t(\boldsymbol{x}) = \boldsymbol{\Phi}(t, \boldsymbol{x}) : \boldsymbol{x} \in V_0\} \ .$$

To understand certain physical phenomena, one wishes to see how a quantity $\boldsymbol{u}$ inside this material volume changes over time. Such a change can result from fluxes over the boundary of the material volume ∂V_t, called surface forces, as well as forces inside the material volume, called volume forces which are divided into distant effects and production terms. We denote surface forces by $\boldsymbol{J}^{(\boldsymbol{u})}$, distant effects by $\boldsymbol{\ell}^{(\boldsymbol{u})}$ and the production term by $\boldsymbol{\sigma}^{(\boldsymbol{u})}$. Then, in its general form, a balance law for a vector of quantities $\boldsymbol{u}$ reads

$$\frac{d}{dt}\int_{V_t} \boldsymbol{u} \, dV = \int_{\partial V_t} \boldsymbol{J}^{(\boldsymbol{u})} \cdot \boldsymbol{n} \, dA + \int_{V_t} \boldsymbol{\ell}^{(\boldsymbol{u})} \, dV + \int_{V_t} \boldsymbol{\sigma}^{(\boldsymbol{u})} \, dV \ ,$$

where $\boldsymbol{n}$ is the unit normal of ∂V_t. Commonly, this balance law is studied in its local form, which is given by

$$\partial_t \boldsymbol{u} + \operatorname{div}(\boldsymbol{v}\boldsymbol{u}^T) + \operatorname{div}\boldsymbol{J}^{(\boldsymbol{u})} - \boldsymbol{\ell}^{(\boldsymbol{u})} - \boldsymbol{\sigma}^{(\boldsymbol{u})} = 0\,.$$

There are several examples for physical phenomena which are governed by equations of this form:

Example 1.

- *The Navier-Stokes equations describe the time evolution of fluids when neglecting volume forces. The variables used to quantify the behavior of the gas are usually density ρ, momentum $\rho\boldsymbol{v}$ and total energy ρE, where $E = e + \frac{1}{2}\boldsymbol{v}^T\boldsymbol{v}$. Here, e denotes the internal energy. The surface and volume forces then become*

$$\begin{aligned}
u_1 = \rho: &\qquad \boldsymbol{J}^{(\rho)} = \boldsymbol{0},\ \ \ell^{(\rho)} = 0,\ \ \sigma^{(\rho)} = 0\,,\\
u_2 = \rho\boldsymbol{v}: &\qquad \boldsymbol{J}^{(\rho\boldsymbol{v})} = \boldsymbol{p}\boldsymbol{I} - \boldsymbol{\sigma},\ \ \ell^{(\rho\boldsymbol{v})} = 0,\ \ \sigma^{(\rho\boldsymbol{v})} = 0\,,\\
u_3 = \rho E: &\qquad \boldsymbol{J}^{(\rho E)} = \boldsymbol{v}^T(p\boldsymbol{I} - \boldsymbol{\sigma}) + \boldsymbol{q},\ \ \ell^{(\rho E)} = 0,\ \ \sigma^{(\rho E)} = 0\,,
\end{aligned}$$

where p is the pressure and q is the heat flux. Since we now have $d+4$ unknowns and only $d+2$ equations, we need to close this system. Commonly, this is done by prescribing a thermal equation of state $p = p(\rho, e)$ and a caloric equation of state $e = e(T)$. For a caloric perfect gas, we for example have

$$\textit{material}\, p = \rho e(\gamma - 1),\ \textit{with the heat capacity ratio}\ \gamma = \frac{C_P}{C_V},$$

where C_V, C_P is the heat capacity at constant volume/pressure. Furthermore, we have

$$e = C_V T.$$

- *When neglecting viscosity and heat conduction, one gets the Euler equations, i.e. we have*

$$\begin{aligned}
u_1 = \rho: &\qquad \boldsymbol{J}^{(\rho)} = \boldsymbol{0},\ \ \ell^{(\rho)} = 0,\ \ \sigma^{(\rho)} = 0\,,\\
u_2 = \rho\boldsymbol{v}: &\qquad \boldsymbol{J}^{(\rho\boldsymbol{v})} = \boldsymbol{p}\boldsymbol{I},\ \ \ell^{(\rho\boldsymbol{v})} = 0,\ \ \sigma^{(\rho\boldsymbol{v})} = 0\,,\\
u_3 = \rho E: &\qquad \boldsymbol{J}^{(\rho E)} = \boldsymbol{v}^T p\boldsymbol{I},\ \ \ell^{(\rho E)} = 0,\ \ \sigma^{(\rho E)} = 0\,.
\end{aligned}$$

The discussion of analytic solution properties and numerical methods will neglect distant effects and production, which is why these terms are set to zero in the following. To simplify notation, we combine the streaming term $\operatorname{div}(\boldsymbol{v}\boldsymbol{u}^T)$ and the volume term in a so-called *physical flux* $\boldsymbol{f}_j : \mathbb{R}^m \to \mathbb{R}^m$, where $j = 1, \cdots, d$ is the spatial coordinate direction. The time evolution of a vector of conserved quantities $\boldsymbol{u} : D \times \mathbb{R}^+ \to \mathcal{D} \subset \mathbb{R}^m$ can then be expressed in its *conservative form* as

$$\partial_t \boldsymbol{u} + \operatorname{div}(\boldsymbol{f}_1(\boldsymbol{u}), \cdots, \boldsymbol{f}_d(\boldsymbol{u})) = 0\,, \tag{1.1}$$

$$\boldsymbol{u}(t = 0, \boldsymbol{x}) = \boldsymbol{u}_{\mathrm{IC}}(\boldsymbol{x})\,, \tag{1.2}$$

where $\boldsymbol{u}_{\mathrm{IC}}$ is the initial condition. Usually, (1.1) is supplemented with boundary conditions, which we will specify later on.

1.1.2 Classical, weak and entropy solutions

Defining the flux Jacobians $\boldsymbol{A}_j := \nabla_{\boldsymbol{u}} \boldsymbol{f}_j \in \mathbb{R}^{m\times m}$, the system (1.1) can be rewritten in its *quasi-conservative form*

$$\partial_t \boldsymbol{u} + \sum_{j=1}^{d} \boldsymbol{A}_j(\boldsymbol{u}) \partial_{x_j} \boldsymbol{u} = \boldsymbol{0}, \tag{1.3}$$

which can be used to determine the type of equation (1.1): Defining the flux Jacobian into direction $\boldsymbol{w} \in \mathbb{R}^d$ as

$$\boldsymbol{A}(\boldsymbol{u}, \boldsymbol{w}) := \sum_{j=1}^{d} \boldsymbol{A}_j(\boldsymbol{u}) w_j,$$

enables the characterization of the conservation equation (1.1) with the following definition:

Definition 1. *A system is called hyperbolic, if the flux Jacobian $\boldsymbol{A}(\boldsymbol{u}, \boldsymbol{w})$ has only real eigenvalues $\lambda_k(\boldsymbol{u}, \boldsymbol{w})$ for $k = 1, \cdots, m$ with a complete family of eigenvectors $\boldsymbol{r}_k(\boldsymbol{u}, \boldsymbol{w})$ for all states $\boldsymbol{u} \in \mathcal{D}$ and every direction $\boldsymbol{w} \in \mathbb{R}^d$ with $\|\boldsymbol{w}\| = 1$. I.e. for $k = 1, \cdots, m$ we have*

$$\boldsymbol{A}(\boldsymbol{u}, \boldsymbol{w}) \boldsymbol{r}_k(\boldsymbol{u}, \boldsymbol{w}) = \lambda_k(\boldsymbol{u}, \boldsymbol{w}) \boldsymbol{r}_k(\boldsymbol{u}, \boldsymbol{w})$$

with $\lambda_k \in \mathbb{R}$. If all eigenvalues are pairwise distinct for all states $\boldsymbol{u} \in \mathcal{D}$, then equation (1.1) is called strictly hyperbolic.

Note that for one spatial dimension, i.e. $d = 1$, the direction is $w = 1$ and therefore hyperbolicity holds, if the flux Jacobian $\nabla_{\boldsymbol{u}} \boldsymbol{f}$ is diagonalisable with real eigenvalues.

Conservation equations can be solved analytically with the method of characteristics. When assuming that equation (1.1) is scalar[1], the characteristics are curves

$$\mathcal{C} = \{(t(s), \boldsymbol{x}(s))\}$$

with

$$\frac{dt(s)}{ds} = 1, \quad \frac{dx_j(s)}{ds} = \nabla_u f_j(u(t(s), \boldsymbol{x}(s))). \tag{1.4}$$

The solution u is constant along characteristic curves, since

$$\frac{du(t(s), \boldsymbol{x}(s))}{ds} = \partial_t u \frac{dt}{ds} + \sum_{j=1}^{d} \partial_{x_j} u \frac{dx_j}{ds} \overset{(1.4),(1.3)}{=} 0. \tag{1.5}$$

Furthermore, characteristics are straight lines, i.e. $\frac{d\boldsymbol{x}(s)}{ds}$ is constant, which can be seen by checking that

$$\frac{dx_j(s)}{ds} = \nabla_u f_j(u(t(s), \boldsymbol{x}(s))) \overset{(1.5)}{=} \nabla_u f_j(u(0, \boldsymbol{x}(0))) = const.$$

When tracing characteristics back in time to $t = 0$, the solution can be determined by evaluating the known initial condition. However, the applicability of the method of

[1] The method of characteristics can also be derived for systems, which however requires a more complicated notation.

characteristics is limited: If the physical flux f is non-linear and the initial condition is non-constant, characteristics intersect after a finite time. Intersecting characteristics indicate that the solution takes on two different values of the initial condition on the same spatial point at the same time, meaning that discontinuities form. Indeed, physical applications described with the help of conservation laws tend to show discontinuous shocks. In this case, the solution is no longer differentiable and characteristic curves cannot be defined. Hence, a new concept allowing discontinuous solutions is needed. This concept leads to so-called *weak solutions*, whereas the solution given by characteristics is called *classical solution*.

The main idea of a weak solution is to move spatial and time derivatives in the original equation (1.1) onto a smooth test function. Hence, after multiplying the original conservation equation (1.1) with test functions $\phi \in C_0^\infty(\mathbb{R}_+ \times \mathbb{R}^d)$, and applying integration by parts, one gets

$$\int_{\mathbb{R}_+\times\mathbb{R}} \boldsymbol{u}\phi_t + \sum_{j=1}^{d} \boldsymbol{f}_j(\boldsymbol{u})\partial_{x_j}\phi \, d\boldsymbol{x}dt + \int_{\mathbb{R}} \boldsymbol{u}_{\mathrm{IC}}(\boldsymbol{x})\phi(0,\boldsymbol{x}) \, d\boldsymbol{x} = 0, \quad \forall \phi \in C_0^\infty(\mathbb{R}_+ \times \mathbb{R}^d). \tag{1.6}$$

Consequently, the solution $\boldsymbol{u}$ does not need to be differentiable in the spatial and time variables. One can now study properties of shocks in the weak solution concept. Let us assume that the shock moves along a curve Σ^* with normal vector $\boldsymbol{n} = (n_t, n_1, \cdots, n_d)^T$ in the space-time domain. A space-time domain with finite support, which we call D_0, is now split into two volumes $D^\pm \subset \mathbb{R}^{1+d}$ by Σ^*, which one plugs into (1.6) to obtain the following conditions for discontinuous solutions, which are called *Rankine-Hugoniot conditions*: If the solution is differentiable inside each region $D^\pm \subset \mathbb{R}^{1+d}$, discontinuous jumps from the state $\boldsymbol{u}_L$ to $\boldsymbol{u}_R$ are admissible weak solutions, if

$$(\boldsymbol{u}_L - \boldsymbol{u}_R)n_t + \sum_{j=1}^{d} (\boldsymbol{f}_j(\boldsymbol{u}_L) - \boldsymbol{f}_j(\boldsymbol{u}_R))n_j = \boldsymbol{0} \tag{1.7}$$

holds. Unfortunately, the notion of weak solutions does not yield uniqueness, since multiple solutions fulfilling the Rankine-Hugoniot condition exist. Hence, to ensure uniqueness, one needs to introduce a further concept, called the entropy solution [78, Chapter 3.8.1]. Let us first introduce the entropy:

Definition 2. *Let $\mathcal{D}$ be convex. Then a convex function $s : \mathcal{D} \to \mathbb{R}$ is called an entropy for the conservation equations (1.1) if there exist d functions $\tilde{F}_j : \mathcal{D} \to \mathbb{R}$, called entropy fluxes, which fulfill the integrability condition*

$$\nabla_{\boldsymbol{u}} s(\boldsymbol{u}) \nabla_{\boldsymbol{u}} \boldsymbol{f}_j(\boldsymbol{u}) = \nabla_{\boldsymbol{u}} \tilde{F}_j(\boldsymbol{u}), \qquad j = 1, \cdots, d. \tag{1.8}$$

For classical solutions, the integrability condition ensures conservation of entropy: By multiplying $\nabla_{\boldsymbol{u}} s(\boldsymbol{u})$ from the left with the original equation (1.1), we get with the integrability condition (1.8) that

$$\partial_t s(\boldsymbol{u}) + \sum_{j=0}^{d} \partial_{x_j} \tilde{F}_j(\boldsymbol{u}) = 0. \tag{1.9}$$

Since (1.9) is again in conservation form, the entropy is conserved at smooth solutions and the flux functions $\tilde{F}_j$ are the flux functions of the entropy balance law (1.9).

Let us now move to a situation where characteristics intersect, i.e. no classical solution is available and we need to pick a physical meaningful, unique solution from a set of different solution candidates. To include some physical intuition, let us remember that a physical system includes friction, which is modeled by diffusion terms on the right hand side of the original equation (1.1). Hence, a more realistic setting would be

$$\partial_t \boldsymbol{u}^{(\varepsilon)} + \sum_{j=1}^{d} \partial_{x_j} \boldsymbol{f}_j(\boldsymbol{u}^{(\varepsilon)}) = \varepsilon \Delta \boldsymbol{u}^{(\varepsilon)}, \tag{1.10}$$

where $\varepsilon > 0$ models the strengths of friction. The solution to this problem is called the *viscous solution*. An interesting question is how the solution $\boldsymbol{u}^{(\varepsilon)}$ behaves when ε tends to zero, i.e. when we turn off friction and fall back to a balance law of the form (1.1). Indeed, under certain conditions [45], the limit of $\boldsymbol{u}^{(\varepsilon)}$ is unique as well as a weak solution of the original hyperbolic equations. That is good news, however we would like to find an easily verifiable condition to check if a solution belongs to such a limit. For this, let us check whether the solution to (1.10) dissipates the entropy and how the limit $\varepsilon \to 0$ affects this dissipation. Similar to the derivation of (1.9), multiplying (1.10) with $\nabla_{\boldsymbol{u}} s(\boldsymbol{u})$ yields

$$\partial_t s(\boldsymbol{u}) + \sum_{j=0}^{d} \partial_{x_j} \tilde{F}_j(\boldsymbol{u}) = \varepsilon \nabla_{\boldsymbol{u}} s(\boldsymbol{u}) \Delta \boldsymbol{u}^{(\varepsilon)}.$$

Note that with

$$\nabla \left((\nabla \boldsymbol{u})^T \nabla_{\boldsymbol{u}} s(\boldsymbol{u}) \right) = (\nabla \boldsymbol{u})^T \nabla_{\boldsymbol{u}\boldsymbol{u}} s(\boldsymbol{u}) \nabla \boldsymbol{u} + \nabla_{\boldsymbol{u}} s(\boldsymbol{u}) \Delta \boldsymbol{u},$$

we get

$$\partial_t s(\boldsymbol{u}) + \sum_{j=0}^{d} \partial_{x_j} \left(\tilde{F}_j(\boldsymbol{u}) - \varepsilon \partial_{x_j} \left((\nabla \boldsymbol{u})^T \nabla_{\boldsymbol{u}} s(\boldsymbol{u}) \right) \right) = -\varepsilon \left(\nabla \boldsymbol{u} \right)^T \nabla_{\boldsymbol{u}\boldsymbol{u}} s(\boldsymbol{u}) \nabla \boldsymbol{u}. \tag{1.11}$$

Hence, we derived a balance equation for the entropy with a production term on the right hand side. In order to ensure entropy dissipation, we need to ensure negativity of the right hand side, which is obviously the case, since the entropy s is convex and the friction strength ε is positive. Hence, the viscous solution dissipates the entropy for all ε. Furthermore, it can be shown that if the sequence $\boldsymbol{u}^{(\varepsilon)}$ converges, the limit will still fulfill (1.11). This motivates the definition of the *entropy solution*.

Definition 3. *If $\boldsymbol{u}$ is a weak solution, which fulfills*

$$\partial_t s(\boldsymbol{u}) + \sum_{j=0}^{d} \partial_{x_j} \tilde{F}_j(\boldsymbol{u}) \leq 0 \tag{1.12}$$

in a weak sense for all admissible entropies s, then $\boldsymbol{u}$ is called an entropy solution.

Indeed, it can be shown that the entropy solution and the limit of the viscous problem coincide under certain conditions [45], meaning that the entropy solution is unique. Scalar hyperbolic problems admit a unique entropy solution $u(t, \cdot) \in L^1(\mathbb{R}^d)$ for every $t \in \mathbb{R}^+$ if the initial condition lies in $L^\infty(\mathbb{R}^d)$, see e.g. [45, 123]. Note that opposed to the entropy used in thermodynamics, the mathematical entropy s is dissipated in time: By integrating (1.12) over the spatial domain while assuming that

the entropy fluxes are zero at the boundary, we obtain

$$\frac{d}{dt}\int_D s(\boldsymbol{u})\,d\boldsymbol{x} \leq 0. \tag{1.13}$$

1.1.3 Entropy variables and hyperbolicity

The notion of entropy is closely related to hyperbolicity, which can be shown with the help of the *entropy variables*

$$\boldsymbol{v} = \nabla_{\boldsymbol{u}} s(\boldsymbol{u})^T \in \mathbb{R}^m. \tag{1.14}$$

If s is strictly convex, the mapping $\boldsymbol{v}(\boldsymbol{u})$ is one-to-one and the solution $\boldsymbol{u}$ can be represented in terms of entropy variables as $\boldsymbol{u} : \mathbb{R}^m \to \mathbb{R}^m$ with $\boldsymbol{u}(\boldsymbol{v}) = (\nabla_{\boldsymbol{u}} s)^{-1}(\boldsymbol{v})$.[2] A change from the conserved quantities $\boldsymbol{u}$ to their corresponding entropy variables can be performed to put (1.1) in its *symmetric form*

$$\partial_t \boldsymbol{u}(\boldsymbol{v}) + \sum_{j=1}^{d} \partial_{x_j} \boldsymbol{g}_j(\boldsymbol{v}) = \boldsymbol{0}, \tag{1.15}$$

where the flux with respect to the entropy variables has been denoted by $\boldsymbol{g}_j$, i.e.

$$\boldsymbol{g}_j(\boldsymbol{v}) := \boldsymbol{f}_j(\boldsymbol{u}(\boldsymbol{v})) \qquad \text{with } j = 1, \cdots, d. \tag{1.16}$$

Our goal is to check hyperbolicity, i.e. (1.15) needs to be brought into its quasi-conservative form (1.3). Applying the chain rule results in

$$\boldsymbol{H}(\boldsymbol{v})\partial_t \boldsymbol{v} + \sum_{j=1}^{d} \boldsymbol{B}_j(\boldsymbol{v})\partial_{x_j}\boldsymbol{v} = \boldsymbol{0}, \tag{1.17}$$

with

$$\boldsymbol{H}(\boldsymbol{v}) = \nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v}) \qquad \text{and} \qquad \boldsymbol{B}_j(\boldsymbol{v}) = \nabla_{\boldsymbol{v}}\boldsymbol{g}_j(\boldsymbol{v}). \tag{1.18}$$

Note that $\boldsymbol{H}(\boldsymbol{v}) = \left(\nabla^2_{\boldsymbol{u}} s(\boldsymbol{u})\right)^{-1}$, which can be checked by differentiating $\nabla_{\boldsymbol{u}} s(\boldsymbol{u}(\boldsymbol{v})) = \boldsymbol{v}$ with respect to $\boldsymbol{v}$. Here, the notation $\nabla^2_{\boldsymbol{u}}\cdot$ represents the Hessian matrix with respect to $\boldsymbol{u}$. Consequently, the first term in (1.17) is symmetric positive definite and can therefore be rewritten as $\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, where $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is orthogonal and $\boldsymbol{\Lambda} \in \mathbb{R}^{m\times m}$ is a diagonal matrix with positive entries. Consequently, the regular, symmetric matrix $\boldsymbol{H}^{1/2} := \boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}\boldsymbol{Q}^T$ exists. Multiplying (1.17) with $\boldsymbol{H}^{-1} = \boldsymbol{H}^{-1/2}\boldsymbol{H}^{-1/2}$ from the left results in the system

$$\partial_t \boldsymbol{v} + \sum_{j=1}^{d} \boldsymbol{H}^{-1/2}\boldsymbol{H}^{-1/2}\nabla_{\boldsymbol{v}}\boldsymbol{g}_j(\boldsymbol{v})\boldsymbol{H}^{-1/2}\boldsymbol{H}^{1/2}\partial_{x_j}\boldsymbol{v} = \boldsymbol{0}\,. \tag{1.19}$$

It remains to check under which conditions the flux Jacobian of this system is diagonalizable with real eigenvalues. Since $\boldsymbol{H}^{-1/2}$ is symmetric, symmetry of $\boldsymbol{B}_j$ suffices to show symmetry (and thereby diagonalizability with real eigenvalues) of $\boldsymbol{H}^{-1/2}\nabla_{\boldsymbol{v}}\boldsymbol{g}_j(\boldsymbol{v})\boldsymbol{H}^{-1/2}$. Multiplying this matrix with $\boldsymbol{H}^{-1/2}$ from the left and $\boldsymbol{H}^{1/2}$

[2] Note that we have prescribed $\boldsymbol{u}$ to be in $\mathbb{R}^m$, i.e. strictly speaking we have $\boldsymbol{u}(\boldsymbol{v}) = (\nabla_{\boldsymbol{u}} s)^{-T}(\boldsymbol{v})$.

from the right is a similarity transformation and therefore does not change eigenvalues. Hence when $\boldsymbol{B}_j$ is symmetric, the system (1.19) is diagonalizable with real eigenvalues and therefore hyperbolic[3]. Note that this ensures hyperbolicity of the original system (1.1), since the flux Jacobian of (1.19) can be written as

$$\boldsymbol{H}^{-1}\nabla_{\boldsymbol{v}}\boldsymbol{g}_j(\boldsymbol{v}) = \nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{u}}\boldsymbol{f}_j(\boldsymbol{u})\nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v}).$$

Thus, the matrix on the left is a similarity transform of the flux Jacobian of the original problem (1.1). It remains to check symmetry of $\boldsymbol{B}_j$ which can be ensured via the concept of entropy.

Theorem 1. *The matrices $\boldsymbol{B}_j$ are symmetric iff the integrability condition (1.8) holds.*

Proof. In the following proof, index notation and Einstein's sum convention are used. We assume that the integrability condition

$$\partial_{u_i} s(\boldsymbol{u})\partial_{u_\ell}(\boldsymbol{f}_j)_i(\boldsymbol{u}) = \partial_{u_\ell}\tilde{F}_j(\boldsymbol{u}), \qquad j = 1, \cdots, d$$

holds. Differentiating these equations with respect to $\boldsymbol{u}$ yields

$$\partial_{u_\ell u_k}\tilde{F}_j(\boldsymbol{u}) = \partial_{u_i u_k} s(\boldsymbol{u})\partial_{u_\ell}(\boldsymbol{f}_j)_i(\boldsymbol{u}) + \partial_{u_i} s(\boldsymbol{u})\partial_{u_\ell u_k}(\boldsymbol{f}_j)_i(\boldsymbol{u}), \qquad j = 1, \cdots, d. \tag{1.20}$$

Since the left-hand side is symmetric, so is the right hand side. Due to symmetry of the second term on the right-hand side, the matrices

$$\partial_{u_i u_k} s(\boldsymbol{u})\partial_{u_\ell}(\boldsymbol{f}_j)_i(\boldsymbol{u}), \qquad \text{with } j = 1, \cdots, d$$

must be symmetric. Multiplication with $\nabla^2_{\boldsymbol{u}} s(\boldsymbol{u})^{-1} \equiv \nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v})$ from both sides gives symmetric matrices

$$\partial_{u_\ell}(\boldsymbol{f}_j)_i(\boldsymbol{u})\partial_{v_i} u_\ell(\boldsymbol{v}) \qquad \text{with } j = 1, \cdots, d, \tag{1.21}$$

which is equivalent to the flux Jacobian of the flux $\boldsymbol{g}_j$ defined in (1.16). Hence, the matrices $\boldsymbol{B}_j$ in (1.19) are symmetric and the system can be diagonalized with real eigenvectors. Vice versa, symmetry of $\boldsymbol{B}_j$ implies symmetry of the right-hand side of (1.20), meaning that $\partial_{u_\ell}\tilde{F}_j(\boldsymbol{u})$ has a potential function $\tilde{F}_j$ which satisfies the integrability condition. □

Broadly speaking, the proof shows the existence of the flux potentials $\psi_j : \mathbb{R}^m \to \mathbb{R}$ where $j = 1, \cdots, d$. The potentials fulfill $\nabla_{\boldsymbol{v}}\psi_j(\boldsymbol{v}) = \boldsymbol{g}_j(\boldsymbol{v})$. Therefore, when bringing (1.1) into its symmetric form (1.15), the resulting flux Jacobians read

$$\partial_{u_\ell}(\boldsymbol{f}_j)_i(\boldsymbol{u})\partial_{v_i} u_\ell(\boldsymbol{v}) = \nabla^2_{\boldsymbol{v}}\psi_j(\boldsymbol{v}) \qquad \text{with } j = 1, \cdots, d,$$

which are obviously symmetric. Note that this resembles the result (1.21), which showed symmetry of the flux Jacobian. The flux potentials are given by

$$\psi_j(\boldsymbol{v}) = \boldsymbol{v}^T\boldsymbol{g}_j(\boldsymbol{v}) - \tilde{F}_j(\boldsymbol{u}(\boldsymbol{v}))$$

[3] If $\boldsymbol{B}_j$ is symmetric for $j = 1, \cdots, d$, the flux Jacobian into any direction $\boldsymbol{w} \in \mathbb{R}^d$, namely $\boldsymbol{B}(\boldsymbol{v}, \boldsymbol{w}) := \sum_{j=1}^d w_j \boldsymbol{B}_j(\boldsymbol{v})$, is symmetric as well, i.e. the system is hyperbolic, cf. Definition 1.

since

$$\begin{aligned}\nabla_{\boldsymbol{v}}\psi_j(\boldsymbol{v}) =& \boldsymbol{g}_j(\boldsymbol{v}) + \boldsymbol{v}^T\nabla_{\boldsymbol{v}}\boldsymbol{g}_j(\boldsymbol{v}) - \nabla_{\boldsymbol{u}}\tilde{F}_j(\boldsymbol{u}(\boldsymbol{v}))\nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v})\\ \overset{(1.8)}{=}& \boldsymbol{g}_j(\boldsymbol{v}) + \boldsymbol{v}^T\nabla_{\boldsymbol{v}}\boldsymbol{g}_j(\boldsymbol{v}) - \nabla_{\boldsymbol{u}}s(\boldsymbol{u})\nabla_{\boldsymbol{u}}\boldsymbol{f}_j(\boldsymbol{u})\nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v})\\ =& \boldsymbol{g}_j(\boldsymbol{v}) + \boldsymbol{v}^T\nabla_{\boldsymbol{v}}\boldsymbol{g}_j(\boldsymbol{v}) - \boldsymbol{v}^T\nabla_{\boldsymbol{v}}\boldsymbol{g}_j(\boldsymbol{v}).\end{aligned}$$

Hence, the entropy fluxes are given by

$$\tilde{F}_j(\boldsymbol{u}(\boldsymbol{v})) = \boldsymbol{v}^T\boldsymbol{g}_j(\boldsymbol{v}) - \psi_j(\boldsymbol{v}). \tag{1.22}$$

In the case of scalar equations, i.e. when

$$\partial_t u + \sum_{j=1}^{d}\partial_{x_j} f_j(u) = 0, \tag{1.23a}$$

$$u(t=0,x) = u_{\mathrm{IC}}(x), \tag{1.23b}$$

all convex functions can be used as entropies. In particular, a family of entropies, which is also called the Kružkov entropy [66], given by

$$s(u) = |u-k| \qquad \text{for all } k \in \mathbb{R} \tag{1.24}$$

fulfills the integrability condition for the entropy flux

$$\tilde{F}_j(u) = \operatorname{sign}(u-k)(f_j(u) - f_j(k)).$$

This family of entropies can be employed to derive several solution properties for scalar equations. Two important stability properties are

$$\|u(t,\cdot) - u^*(t,\cdot)\|_{L^1(D)} \leq \|u_{\mathrm{IC}} - u^*_{\mathrm{IC}}\|_{L^1(D)}, \tag{1.25a}$$

$$\|u(t,\cdot)\|_{L^\infty(D)} \leq \|u_{\mathrm{IC}}\|_{L^\infty(D)}, \tag{1.25b}$$

where u^* is an entropy solution to (1.23a) with initial condition u^*_{IC} [58, Chapter 2.4]. Property (1.25a), known as L^1 stability, guarantees boundedness of oscillations, playing a key role in the construction of numerical schemes for solving scalar problems (1.23) as this ensures convergence to a unique, physically meaningful solution (see [78, Chapter 15.3]). Property (1.25b) is known as the maximum–principle or infinity stability [58, Chapter 2.4] and ensures bounds on the solution.

An important concept of hyperbolic conservation laws is the *total variation*, which is defined as

$$\mathrm{TV}(u(t,\cdot)) := \sup\left\{\int_D u(t,x)\operatorname{div}\phi(\boldsymbol{x})\,d\boldsymbol{x} : \phi \in C_0^1(D), \|\phi\|_{L^\infty(D)} \leq 1\right\} \tag{1.26}$$

where $C_0^1(D)$ are the continuously differentiable functions with compact support. When u is sufficiently smooth for all t, namely $u \in W^{1,1}(D)$, then (1.26) can be written as

$$\mathrm{TV}(u) := \int_D |\partial_x u(t,x)|\,dx.$$

Choosing $u^*_{\mathrm{IC}}(x) = u_{\mathrm{IC}}(x+h)$, multiplying (1.25a) with $1/h$ and letting h go to zero yields

$$\int_D |\partial_x u(t,x)|\, dx \leq \int_D |\partial_x u_{\mathrm{IC}}(x)|\, dx, \tag{1.27}$$

which shows that oscillations w.r.t. x are bounded by oscillations of the initial condition. We can reformulate this as

$$\mathrm{TV}(u(t,\cdot)) \leq TV(u_{IC}) \qquad \forall t \geq 0. \tag{1.28}$$

This property is called the *Total Variation Diminishing* (TVD) property, which will play an important role in the construction of numerical schemes to solve hyperbolic conservation laws.

1.2 Finite volume methods

Most applications do not allow the derivation of an analytic solution, which is why the hyperbolic equation (1.1) needs to be solved numerically. Commonly, numerical methods for hyperbolic problems are derived for scalar equations in one spatial dimension and are then extended to multidimensional, non-scalar settings. Therefore, the following section will discuss how to approximate a solution to

$$\partial_t u(t,x) + \partial_x f(u(t,x)) = 0 \tag{1.29}$$

numerically. First, the solution u needs to be discretized, i.e. represented by a finite number of coefficients. The main strategy is to divide the physical domain $x \in [a,b]$ into N_x cells $I_j = [x_j, x_{j+1}]$ with $a = x_0 < \cdots < x_{N_x} = b$ as well as the time domain into N_t discrete values $0 = t_0 < \cdots < t_{N_t-1} = t_{\mathrm{end}}$. The solution is then represented in every spatial cell for each time step. A commonly used class of numerical schemes are *finite volume methods*, which represent the solution as an average in each cell. In cell I_j at time step t_n, the solution is denoted by u_j^n, i.e. with $\Delta x_j := x_{j+1} - x_j$ we have

$$u_j^n := \frac{1}{\Delta x_j} \int_{x_j}^{x_{j+1}} u(t_n, x)\, dx. \tag{1.30}$$

In order to derive an evolution equation for this cell wise approximation, the original equation (1.23) is averaged over a space-time cell $[t_n, t_{n+1}] \times [x_j, x_{j+1}]$, which yields

$$\frac{1}{\Delta t_n \Delta x_j} \int_{x_j}^{x_{j+1}} \int_{t_n}^{t_{n+1}} \partial_t u(t,x)\, dt dx + \frac{1}{\Delta t_n \Delta x_j} \int_{t_n}^{t_{n+1}} \int_{x_j}^{x_{j+1}} \partial_x f(u)\, dx dt = 0 \tag{1.31}$$

with $\Delta t_n := t_{n+1} - t_n$. Solving the two inner integrals then gives

$$\frac{1}{\Delta t_n}(u_j^{n+1} - u_j^n) + \frac{1}{\Delta t_n \Delta x_j} \int_{t_n}^{t_{n+1}} f(u(t,x_j)) - f(u(t,x_{j+1}))\, dt = 0.$$

Using this expression to compute the time evolution of the discrete solution values is not possible, since the integral term depends on the solution u evaluated at the cell interfaces between cells I_{j-1}, I_j and I_{j+1} at continuous times t. At this point, an approximation of the integral term needs to be made, where one commonly uses an

ansatz of the form

$$f^*(u_{j-1}^n, u_j^n) \approx \frac{1}{\Delta t_n} \int_{t_n}^{t_{n+1}} f(u(t, x_j))\, dt. \tag{1.32}$$

Hence, the averaged flux between cells I_{j-1} and I_j between time t_n and t_{n+1}, is approximated by using the function values u_{j-1}^n and u_j^n. The approximation function f^* is called the *numerical flux*. In the following, an equidistant spatial and time grid with spacing Δt and Δx is assumed in order to simplify notation. When plugging the approximation (1.32) into (1.31) one obtains the update formula for the discrete solution

$$u_j^{n+1} = h(u_{j-1}^n, u_j^n, u_{j+1}^n) \tag{1.33}$$

with $h : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ given by

$$h(u, v, w) = v - \frac{\Delta t}{\Delta x} \left(f^*(v, w) - f^*(u, v) \right). \tag{1.34}$$

The numerical scheme (1.33) is said to be a *finite volume scheme*. When assuming that the spatial domain is the entire real line, we can write the discrete solution as a sequence $u_\Delta := (u_j)_{j \in \mathbb{Z}}$. The time update function for this sequence is then denoted by $h_\Delta(u_\Delta) := \left(h(u_{j-1}, u_j, u_{j+1}) \right)_{j \in \mathbb{Z}}$. Note that this scheme fulfills a conservation property

$$\sum_{j \in \mathbb{Z}} h(u_{j-1}^n, u_j^n, u_{j+1}^n) = \sum_{j \in \mathbb{Z}} u_j^n, \tag{1.35}$$

i.e. the numerical scheme conserves the total mass $\int_{-\infty}^{\infty} u(t, x)\, dx$ in time on a discrete level. Therefore, schemes making use of an update function which takes the form (1.34) are said to be in *conservation form*.

1.2.1 Consistency and monotonicity

Now the question arises how to pick the numerical flux f^*. Indeed, choosing f^* is a crucial step and should fulfill several important properties, which will be discussed in the following. To make sure the scheme approximates the original equation (1.29), the numerical flux needs to be *consistent*:

Definition 4. *A numerical flux f^* is said to be consistent if $f^*(u, u) = f(u)$. In this case, the scheme (1.33) is said to be consistent with the original equation (1.29).*

To quantify the consistency, the *order of consistency* is introduced:

Definition 5. *A scheme of the form (1.33) has a consistency order $\bar{p}$ if for $\lambda = \Delta t / \Delta x = const$ and $\Delta t \to 0$ the update function h fulfills*

$$u(t + \Delta t, x) - h(u(t, x - \Delta x), u(t, x), u(t, x + \Delta x)) = O(\Delta t^{\bar{p}+1})$$

when u is a smooth solution to (1.29).

A further property, which will be used to ensure stability of proposed schemes is monotonicity:

Definition 6. *A scheme of the form (1.33) is called monotone if the update function h given in (1.34) is monotonically increasing in every argument. I.e. when h is differentiable, a*

monotone scheme fulfills

$$\frac{\partial h(u_1,u_2,u_3)}{\partial u_i} \geq 0 \qquad \text{for } i = 1,2,3. \tag{1.36}$$

Already the numerical flux $f^*(u_\ell, u_r)$ shows whether the resulting scheme will be monotone: By computing the derivatives of h and prescribing positivity, one readily obtains

$$\frac{\partial f^*}{\partial u_\ell} \geq 0, \tag{1.37a}$$

$$1 - \frac{\Delta t}{\Delta x}\left(\frac{\partial f^*}{\partial u_\ell} - \frac{\partial f^*}{\partial u_r}\right) \geq 0, \tag{1.37b}$$

$$\frac{\partial f^*}{\partial u_r} \leq 0. \tag{1.37c}$$

Monotone schemes posses desirable properties. Before going into details, it is worth mentioning that by Godunov's order barrier theorem [141, Chapter 9.2] monotone schemes are commonly of first-order (if $h \in C^3$). However, they yield properties that copy the behavior of the entropy solution on a discrete level. To show this one uses the following (semi-) norms

$$\begin{aligned}
\|u_\Delta\|_{L^1(\Delta)} &:= \Delta x \sum_{j\in\mathbb{Z}} |u_j|,\\
\|u_\Delta\|_{L^\infty(\Delta)} &:= \max_{j\in\mathbb{Z}} |u_j|,\\
\mathrm{TV}_\Delta(u_\Delta) &:= \sum_{j\in\mathbb{Z}} |u_{j+1} - u_j|.
\end{aligned}$$

A monotone scheme possesses the nice property of fulfilling a maximum–principle. Furthermore, it is L^1 contractive while fulfilling the TVD property on a discrete level:

Theorem 2. *If a scheme is monotone, the solution fulfills the discrete maximum principle*

$$\min_{\ell\in\mathbb{Z}^0} u_\ell^0 \leq u_j^n \leq \max_{\ell\in\mathbb{Z}} u_\ell^0 \qquad \text{for all } n \in \mathbb{N}, j \in \mathbb{Z}. \tag{1.39}$$

Furthermore, the scheme is L^1 contractive, i.e.

$$\|h_\Delta(w_\Delta) - h_\Delta(u_\Delta)\|_{L^1(\Delta)} \leq \|w_\Delta - u_\Delta\|_{L^1(\Delta)} \qquad \forall u_\Delta, w_\Delta \tag{1.40}$$

holds and the scheme is total variation diminishing, i.e.

$$TV_\Delta(h_\Delta(u_\Delta)) \leq TV_\Delta(u_\Delta). \tag{1.41}$$

Proof. Let us first derive the maximum-principle, then the L^1 contraction and with this result the TVD property.

- We define a constant solution sequence $(w_j)_{j\in\mathbb{Z}}$ which takes the maximal value of the solution sequence u_Δ. Hence we have $w_j = \max_{j\in\mathbb{Z}} u_j^n$. Now due to monotonicity, we have

$$u_j^{n+1} = h(u_{j-1}^n, u_j^n, u_{j+1}^n) \overset{(1.36)}{\leq} h(w_{j-1}, w_j, w_{j+1}).$$

Since h is in conservation form and $(w_j)_{j\in\mathbb{Z}}$ is constant we have

$$u_j^{n+1} \leq h(w_{j-1}, w_j, w_{j+1}) = \max_{j\in\mathbb{Z}} u_j^n.$$

Successively applying this strategy yields $u_j^n \leq \max_{\ell\in\mathbb{Z}} u_\ell^0$. To show the remaining direction, i.e. $\min_{\ell\in\mathbb{Z}^0} u_\ell^0 \leq u_\ell^0$ one picks a constant sequence $(w_j)_{j\in\mathbb{Z}}$ with $w_j = \min_{j\in\mathbb{Z}} u_j^n$ and performs the steps taken before analogously.

- For showing L^1 contraction, we make use of the notation $x_+ := \max(x, 0)$. When defining $y_j := \max(u_j, w_j)$, we have

$$\begin{aligned}\sum_{j\in\mathbb{Z}} \Big((h_\Delta(u_\Delta))_j - (h_\Delta(w_\Delta))_j \Big)_+ &\overset{(1.36)}{\leq} \sum_{j\in\mathbb{Z}} \Big((h_\Delta(y_\Delta))_j - (h_\Delta(w_\Delta))_j \Big) \\ &\overset{(1.35)}{=} \sum_{j\in\mathbb{Z}} (y_j - w_j) = \sum_{j\in\mathbb{Z}} (u_j - w_j)_+ . \end{aligned} \tag{1.42}$$

Now we use $|u - w| = (u - w)_+ + (w - u)_+$, i.e. the x_+ notation is used to represent absolute values. Hence, we get

$$\begin{aligned}\sum_{j\in\mathbb{Z}} |h_\Delta(u_\Delta)_j - h_\Delta(w_\Delta)_j| &= \sum_{j\in\mathbb{Z}} (h_\Delta(u_\Delta)_j - h_\Delta(w_\Delta)_j)_+ + (h_\Delta(w_\Delta)_j - h_\Delta(u_\Delta)_j)_+ \\ &\overset{(1.42)}{\leq} \sum_{j\in\mathbb{Z}} (u_j - w_j)_+ + (w_j - u_j)_+ = \sum_{j\in\mathbb{Z}} |u_j - w_j|. \end{aligned} \tag{1.43}$$

Now, with the definition of the discrete L^1 norm, one obtains the L^1 contraction property.

- It remains to show the TVD property. Similar to the continuous case, the proof uses the L^1 stability shown above. Choosing the solution sequence u_Δ and the shifted solution sequence w_Δ with $w_j := u_{j+1}$, the L^1 contraction property (1.40) yields

$$\begin{aligned}\mathrm{TV}_\Delta(h_\Delta(u_\Delta)) = \sum_{j\in\mathbb{Z}} |h_\Delta(u_\Delta)_{j+1} - h_\Delta(u_\Delta)_j| &= \sum_{j\in\mathbb{Z}} |h_\Delta(w_\Delta)_j - h_\Delta(u_\Delta)_j| \\ &\overset{(1.43)}{\leq} \sum_{j\in\mathbb{Z}} |w_j - u_j| = \sum_{j\in\mathbb{Z}} |u_{j+1} - u_j| = \mathrm{TV}_\Delta(u_\Delta). \end{aligned}$$

□

Note that these discrete properties mimic the analytically derived behavior of the entropy solution. To be more precise:

- The maximum-principle (1.39) can be rewritten as

$$\|u_\Delta^n\|_{L^\infty(\Delta)} \leq \|u_\Delta^n\|_{L^\infty(\Delta)} \qquad \forall n \in \mathbb{N},$$

which resembles infinity stability (1.25b).

- L^1 stability (1.40) has its continuous counterpart (1.25a).

- The TVD property (1.41) of the numerical scheme resembles the continuous behavior (1.28).

1.2.2 Discrete entropy dissipation

Despite the previously discussed resemblance with the continuous entropy solution, the finite volume scheme (1.33) equipped with a monotone flux will not dissipate all admissible entropies in time. At its best, a scheme should fulfill the discrete entropy dissipation

$$\sum_{j\in\mathbb{Z}} s(u_j^{n+1}) \leq \sum_{j\in\mathbb{Z}} s(u_j^n), \tag{1.44}$$

which mimics the continuous dissipation property (1.13). To check if a numerical scheme fulfills (1.44), we rewrite the scheme in terms of entropy variables. Recall, that for a given solution value u, the corresponding entropy variable v can be computed by $v = s'(u)$. Vice-versa, the entropy variable uniquely defines the solution value by $u = (s')^{-1}(v)$. Hence, for a solution sequence u_Δ, the corresponding sequence of entropy variables $v_\Delta = (v_j)_{j\in\mathbb{Z}}$ with $v_j := s'(u_j)$ can be defined. This allows writing a numerical scheme (1.33) in its entropy variables

$$u(v_j^{n+1}) = u(v_j^n) - \frac{\Delta t}{\Delta x}\left(g^*(v_j^n, v_{j+1}^n) - g^*(v_{j-1}^n, v_j^n)\right).$$

Here, we defined the numerical flux written in its entropy variables as $g^*(v_\ell, v_r) := f^*(u(v_\ell), u(v_r))$. Before investigating the entropy dissipation of this scheme, we take one step back and assume that the time variable is still continuous. This yields the semi-discrete scheme

$$\frac{d}{dt}u(v_j(t)) = -\frac{1}{\Delta x}\left(g^*(v_j(t), v_{j+1}(t)) - g^*(v_{j-1}(t), v_j(t))\right).$$

Omitting the time discretization allows us to first investigate the entropy dissipation property of the spatial discretization, i.e. the dissipation property of the numerical flux. For ease of exposition, we will make use of the notation $g^*_{j+1/2} := g^*(v_j(t), v_{j+1}(t))$. Now to derive an equation describing the time evolution of the entropy, we multiply with $v_j = s'(u_j)$, since this gives

$$\frac{d}{dt}s(u(v_j(t))) = \frac{d}{dt}s(u_j(t)) = -\frac{1}{\Delta x}v_j\left[g^*_{j+1/2} - g^*_{j-1/2}\right]. \tag{1.45}$$

We rewrite the right hand side as

$$\begin{aligned} v_j\left[g^*_{j+1/2} - g^*_{j-1/2}\right] =& \frac{1}{2}(v_j + v_{j+1})g^*_{j+1/2} - \frac{1}{2}(v_{j+1} - v_j)g^*_{j+1/2} \\ &- \frac{1}{2}(v_j + v_{j-1})g^*_{j-1/2} + \frac{1}{2}(v_{j-1} - v_j)g^*_{j-1/2} \\ =& \frac{1}{2}(v_j + v_{j+1})g^*_{j+1/2} - \frac{1}{2}(v_j + v_{j-1})g^*_{j-1/2} \qquad (1.46a) \\ &- \frac{1}{2}(v_{j+1} - v_j)g^*_{j+1/2} - \frac{1}{2}(v_j - v_{j-1})g^*_{j-1/2}. \qquad (1.46b) \end{aligned}$$

Note that (1.46a) is conservative, i.e. it can be written as $\bar{F}_{j+1/2} - \bar{F}_{j-1/2}$ with $\bar{F}(v_\ell, v_r) := \frac{1}{2}(v_\ell + v_r)g^*(v_\ell, v_r)$. This is not the case for (1.46b), i.e. the entropy is not conserved.

We now must ensure entropy dissipation by the choice of the numerical flux. In order to gain consistency with the continuous entropy inequality (1.12), we first define the discrete flux potential $\psi_j := \psi(v_j)$ and add

$$-\frac{1}{2}(\psi_j + \psi_{j+1}) + \frac{1}{2}(\psi_{j-1} + \psi_j) + \frac{1}{2}(\psi_j - \psi_{j+1}) + \frac{1}{2}(\psi_{j-1} - \psi_j) = 0$$

to (1.46). Hence, we get that the term $v_j \left[g^*_{j+1/2} - g^*_{j-1/2}\right]$ equals

$$\begin{aligned}&\frac{1}{2}(v_j + v_{j+1})g^*_{j+1/2} - \frac{1}{2}(\psi_j + \psi_{j+1}) - \frac{1}{2}(v_j + v_{j-1})g^*_{j-1/2} + \frac{1}{2}(\psi_{j-1} + \psi_j)\\ &- \frac{1}{2}(v_{j+1} - v_j)g^*_{j+1/2} + \frac{1}{2}(\psi_j - \psi_{j+1}) - \frac{1}{2}(v_j - v_{j-1})g^*_{j-1/2} + \frac{1}{2}(\psi_{j-1} - \psi_j).\end{aligned}$$

Defining $\tilde{F}_{j+1/2} := \tilde{F}^*(v_j, v_{j+1})$ with

$$\tilde{F}^*(v_j, v_{j+1}) = \frac{1}{2}(v_j + v_{j+1})g^*_{j+1/2} - \frac{1}{2}(\psi_j + \psi_{j+1}) \tag{1.47}$$

we then get

$$\begin{aligned}\frac{d}{dt}s(u_j) = &-\frac{1}{\Delta x}\left[\tilde{F}_{j+1/2} - \tilde{F}_{j-1/2}\right]\\ &+ \frac{1}{\Delta x}\Big[\frac{1}{2}(v_{j+1} - v_j)g^*_{j+1/2} - \frac{1}{2}(\psi_j - \psi_{j+1})\\ &\quad + \frac{1}{2}(v_j - v_{j-1})g^*_{j-1/2} - \frac{1}{2}(\psi_{j-1} - \psi_j)\Big].\end{aligned} \tag{1.48}$$

To ensure a cell entropy inequality, we must choose the numerical flux f^* such that

$$(v_{j+1} - v_j)g^*_{j+1/2} \leq \psi_{j+1} - \psi_j, \tag{1.49}$$

which is the so-called e-scheme property [105, 131]. Note that the conservative flux (1.47) is consistent with $F(u(v)) = vg(v) - \psi(v)$, i.e. we have derived a consistent discrete counterpart of the continuous entropy inequality (1.12). To quantify the entropy dissipation, we define the dissipation term

$$D_{j+1/2} = \frac{1}{2}(v_{j+1} - v_j)g^*_{j+1/2} - \frac{1}{2}(\psi_j - \psi_{j+1})$$

and rewrite (1.48) as

$$\frac{d}{dt}s(u_j) = -\frac{1}{\Delta x}\left[\tilde{F}_{j+1/2} - \tilde{F}_{j-1/2}\right] + \frac{1}{\Delta x}\left[D_{j+1/2} + D_{j-1/2}\right].$$

The e-scheme property (1.49) ensures positivity of D, i.e. summing over all cells gives a global entropy inequality (1.44). Note that if equality holds in (1.49), we obtain $D = 0$ and thereby entropy conservation.

So far, we kept a continuous time variable. When choosing a time discretization, one needs to study the effect on the entropy dissipation. We start with the previously chosen explicit Euler scheme

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}\left(g^*(v_j^n, v_{j+1}^n) - g^*(v_{j-1}^n, v_j^n)\right).$$

Multiplication with v_j^n yields

$$v_j^n u_j^{n+1} = v_j^n u_j^n - \frac{\Delta t}{\Delta x}\left[\tilde{F}_{j+1/2}^n - \tilde{F}_{j-1/2}^n\right] + \frac{\Delta t}{\Delta x}\left[D_{j+1/2}^n + D_{j-1/2}^n\right], \tag{1.50}$$

where we can reuse most terms of the semi-discrete analysis. To relate the terms $v_j^n u_j^{n+1}$ and $u_j^n v_j^n$ to the entropy, we use the identity

$$s(u(v_j^{n+1})) - s(u(v_j^n)) = \int_{-1/2}^{1/2} \frac{d}{d\beta} s(u(v_j^{n+1/2}(\beta)))\, d\beta$$

where

$$v_j^{n+1/2}(\beta) := \frac{1}{2}(v_j^{n+1} + v_j^n) + \beta \Delta v_j^{n+1/2}, \quad \text{with} \quad \Delta v_j^{n+1/2} := v_j^{n+1} - v_j^n. \tag{1.51}$$

For more general $\boldsymbol{u} \in \mathbb{R}^m$, i.e. if we switch from scalar equations to systems, we get

$$\begin{aligned}\int_{-1/2}^{1/2} \frac{d}{d\beta} s(\boldsymbol{u}(\boldsymbol{v}_j^{n+1/2}(\beta)))\, d\beta &= \int \frac{\partial s}{\partial u_i} \frac{\partial u_i(\boldsymbol{v}_j^{n+1/2}(\beta))}{\partial v_l} \frac{dv_{l,j}^{n+1/2}}{d\beta}\, d\beta \\ &= \int_{-1/2}^{1/2} \langle \boldsymbol{v}_j^{n+1/2}, \boldsymbol{H}(\boldsymbol{v}_j^{n+1/2}) \Delta \boldsymbol{v}_j^{n+1/2} \rangle\, d\beta,\end{aligned}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product for vectors. Here we again use the matrix $\boldsymbol{H}(\boldsymbol{v}) = \nabla_{\boldsymbol{v}} \boldsymbol{u}(\boldsymbol{v})$ as defined in (1.18). With

$$\boldsymbol{v}_j^{n+1/2}(\beta) = \boldsymbol{v}_j^n + (\beta + 1/2)\Delta \boldsymbol{v}_j^{n+1/2},$$

we can rearrange this to

$$\begin{aligned}\int \langle \boldsymbol{v}_j^{n+1/2}, \boldsymbol{H}(\boldsymbol{v}_j^{n+1/2}) \Delta \boldsymbol{v}_j^{n+1/2} \rangle\, d\beta =& \boldsymbol{v}_j^n \int \boldsymbol{H}(\boldsymbol{v}_j^{n+1/2}(\beta))\, d\beta (\boldsymbol{v}_j^{n+1} - \boldsymbol{v}_j^n) \\ &+ \int (\beta + 1/2) \langle \Delta \boldsymbol{v}_j^{n+1/2}, \boldsymbol{H}(\boldsymbol{v}_j^{n+1/2}(\beta)) \Delta \boldsymbol{v}_j^{n+1/2} \rangle\, d\beta.\end{aligned}$$

Substituting $\eta = \boldsymbol{v}_j^{n+1/2}(\beta)$ with $\frac{d\eta}{d\beta} = \Delta \boldsymbol{v}_j^{n+1/2}$ into the first term yields

$$\boldsymbol{v}_j^n \int \boldsymbol{H}(\boldsymbol{v}_j^{n+1/2}(\beta))\, d\beta (\boldsymbol{v}_j^{n+1} - \boldsymbol{v}_j^n) = \boldsymbol{v}_j^n \int \boldsymbol{H}(\eta) \frac{1}{\Delta \boldsymbol{v}_j^{n+1/2}} d\eta \Delta \boldsymbol{v}_j^{n+1/2} = \boldsymbol{v}_j^n (\boldsymbol{u}_j^{n+1} - \boldsymbol{u}_j^n).$$

Hence, we get

$$s(\boldsymbol{u}(\boldsymbol{v}_j^{n+1})) - s(\boldsymbol{u}(\boldsymbol{v}_j^n)) = \boldsymbol{v}_j^n \cdot (\boldsymbol{u}_j^{n+1} - \boldsymbol{u}_j^n) - \mathcal{E}_j$$

with the dissipation term

$$\mathcal{E}_j := -\int_{-1/2}^{1/2} (\beta + 1/2) \langle \Delta \boldsymbol{v}_j^{n+1/2}, \boldsymbol{H}(\boldsymbol{v}_j^{n+1/2}(\beta)) \Delta \boldsymbol{v}_j^{n+1/2} \rangle\, d\beta.$$

Note that since $\boldsymbol{H}$ is symmetric positive definite, the dissipation term will be negative. Switching back to our scalar representation and plugging the result into (1.50)

yields

$$v_j^{n+1}u_j^{n+1} - v_j^{n+1}u_j^n = s(u(v_j^{n+1})) - s(u(v_j^n)) + \mathcal{E}_j \tag{1.52}$$

$$\overset{(1.50)}{=} -\frac{\Delta t}{\Delta x}\left[\tilde{F}_{j+1/2}^n - \tilde{F}_{j-1/2}^n\right] + \frac{\Delta t}{\Delta x}\left[D_{j+1/2}^n + D_{j-1/2}^n\right], \tag{1.53}$$

Hence, we have

$$s(u(v_j^{n+1})) = s(u(v_j^n)) - \frac{\Delta t}{\Delta x}\left[\tilde{F}_{j+1/2}^{n+1} - \tilde{F}_{j-1/2}^{n+1}\right] + \frac{\Delta t}{\Delta x}\left[D_{j+1/2}^{n+1} + D_{j-1/2}^{n+1}\right] - \mathcal{E}_j,$$

which shows that an explicit Euler discretization will add entropy, since the dissipation term $\mathcal{E}_j$ is negative. However, when using an implicit Euler discretization, one obtains the dissipation term

$$\mathcal{E}_j^{IE} := \int_{-1/2}^{1/2} (1/2 - \beta)\langle \Delta \boldsymbol{v}_j^{n+1/2}, \boldsymbol{H}(\boldsymbol{v}_j^{n+1/2}(\beta))\Delta \boldsymbol{v}_j^{n+1/2}\rangle \, d\beta > 0, \tag{1.54}$$

i.e. the entropy will be dissipated. For further information on entropy dissipation properties for time discretizations we refer to [130, Chapter 7].

1.2.3 High-order schemes

As already discussed, monotone schemes are only first-order accurate, which is why the construction of higher-order schemes often focuses on preserving only the TVD property (1.41). Note that this property is sufficient to guarantee convergence to a weak solution in the scalar case [21]. To check whether a scheme is TVD, it is put into its *increment form*: Denoting the numerical fluxes at the boundaries of cell j by $f_{j\pm1/2}^*$, we can write the time update as

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}\left[f_{j+1/2}^* - f_{j-1/2}^*\right]. \tag{1.55}$$

When adding and substracting $\frac{\Delta t}{\Delta x} f_j$, where $f_j := f(u_j^n)$ we can rewrite this as

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{\Delta x}\frac{f_j - f_{j+1/2}^*}{u_{j+1}^n - u_j^n}(u_{j+1}^n - u_j^n) - \frac{\Delta t}{\Delta x}\frac{f_j - f_{j-1/2}^*}{u_j^n - u_{j-1}^n}(u_j^n - u_{j-1}^n).$$

Hence, with

$$C_{j+1/2}^+ := \frac{\Delta t}{\Delta x}\frac{f_j - f_{j+1/2}^*}{u_{j+1}^n - u_j^n}, \quad C_{j+1/2}^- := \frac{\Delta t}{\Delta x}\frac{f_{j+1} - f_{j+1/2}^*}{u_{j+1}^n - u_j^n}$$

and $\Delta u_{j+1/2}^n := u_{j+1}^n - u_j^n$, we can write the numerical scheme in its *increment form*

$$u_j^{n+1} = u_j^n + C_{j+1/2}^+ \Delta u_{j+1/2}^n - C_{j-1/2}^- \Delta u_{j-1/2}^n. \tag{1.56}$$

Now, the TVD property is ensured by the following

Theorem 3. *The scheme* (1.55) *is TVD, if*

$$C_{j+1/2}^+ \geq 0, \qquad C_{j+1/2}^- \geq 0, \qquad 1 - C_{j+1/2}^+ - C_{j+1/2}^- \geq 0. \tag{1.57}$$

Proof. Subtracting the to states

$$u_j^{n+1} = u_j^n + C_{j+1/2}^+ \Delta u_{j+1/2}^n - C_{j-1/2}^- \Delta u_{j-1/2}^n,$$
$$u_{j+1}^{n+1} = u_{j+1}^n + C_{j+3/2}^+ \Delta u_{j+3/2}^n - C_{j+1/2}^- \Delta u_{j+1/2}^n,$$

leads to

$$u_{j+1}^{n+1} - u_j^{n+1} = C_{j+3/2}^+ \Delta u_{j+3/2}^n + (1 - C_{j+1/2}^- - C_{j+1/2}^+) \Delta u_{j+1/2}^n + C_{j-1/2}^- \Delta u_{j-1/2}^n.$$

With this, we can write

$$\begin{aligned}
\mathrm{TV}(u_\Delta^{n+1}) &= \sum_{j\in\mathbb{Z}} \left| u_{j+1}^{n+1} - u_j^{n+1} \right| \\
&= \sum_{j\in\mathbb{Z}} \left| C_{j+3/2}^+ \Delta u_{j+3/2}^n + (1 - C_{j+1/2}^- - C_{j+1/2}^+) \Delta u_{j+1/2}^n + C_{j-1/2}^- \Delta u_{j-1/2}^n \right| \\
&\overset{(1.57)}{\leq} \sum_{j\in\mathbb{Z}} C_{j+3/2}^+ \left| \Delta u_{j+3/2}^n \right| + \sum_{j\in\mathbb{Z}} (1 - C_{j+1/2}^+ - C_{j-1/2}^-) \left| \Delta u_{j+1/2}^n \right| \\
&\quad + \sum_{j\in\mathbb{Z}} C_{j-1/2}^- \left| \Delta u_{j-1/2}^n \right| \\
&= \sum_{j\in\mathbb{Z}} C_{j+1/2}^+ \left| \Delta u_{j+1/2}^n \right| + \sum_{j\in\mathbb{Z}} (1 - C_{j+1/2}^+ - C_{j+1/2}^-) \left| \Delta u_{j+1/2}^n \right| \\
&\quad + \sum_{j\in\mathbb{Z}} C_{j+1/2}^- \left| \Delta u_{j+1/2}^n \right| = \sum_{j\in\mathbb{Z}} \left| \Delta u_{j+1/2}^n \right| = \mathrm{TV}(u_\Delta^n).
\end{aligned}$$

Hence, the scheme is TVD. □

Let us now construct a higher-order scheme and then check if the TVD property is fulfilled. We will start with a linear advection equation (i.e. $f(u) = au$). The key idea to construct higher-order schemes is to replace the solution inside the numerical flux $f^*(u_j, u_{j+1})$ by a higher-order approximation at the cell interface between cell j and cell $j+1$. This is done by defining a piece wise linear solution approximation

$$\tilde{u}(t_n, x) = u_j^n + \sigma_j^n (x - x_{j+1/2}) \qquad \text{for cell } [x_j, x_{j+1}].$$

Now σ_j^n is a slope that needs to be determined by neighboring cells. When $a > 0$ (which we will assume in the following), a natural choice would for example be $\sigma_j^n = \frac{1}{\Delta x}(u_{j+1}^n - u_j^n)$. Using this continuous description of our solution, the linear advection equation can be solved analytically by moving the solution to the right by a factor of $a\Delta t$ and then averaging over cells. With $\lambda_a := \frac{a\Delta t}{\Delta x}$, this yields

$$u_j^{n+1} = u_j^n - \lambda_a (u_j^n - u_{j-1}^n) - \frac{\Delta x}{2} \lambda_a (1 - \lambda_a) (\sigma_j^n - \sigma_{j-1}^n). \tag{1.58}$$

To check under which conditions on the slope σ this scheme fulfills the TVD property, we introduce

$$\phi_j := \frac{\Delta x}{u_{j+1} - u_j} \sigma_j, \qquad \theta_j := \frac{u_j - u_{j-1}}{u_{j+1} - u_j}.$$

Then we can rewrite (1.58) as

$$\begin{aligned} u_j^{n+1} =& u_j^n - \lambda_a (u_j^n - u_{j-1}^n) - \frac{1}{2}\lambda_a (1-\lambda_a) \left((u_{j+1} - u_j)\phi_j^n - (u_j - u_{j-1})\phi_{j-1}^n \right) \\ =& u_j^n - \left(\lambda_a - \frac{1}{2}\lambda_a (1-\lambda_a) \left(\frac{\phi_j^n}{\theta_j} - \phi_{j-1}^n \right) \right) (u_j^n - u_{j-1}^n) \, . \end{aligned}$$

Note that we have derived an increment for the second-order scheme where

$$\begin{aligned} C_{j-1/2}^- &= \lambda_a \left(1 - \frac{1-\lambda_a}{2} \left(\frac{\phi_j^n}{\theta_j} - \phi_{j-1}^n \right) \right) \, , \\ C_{j+1/2}^+ &= 0 \, . \end{aligned}$$

To check the TVD property, we need to make sure conditions (1.57) are fulfilled. Since obviously $C_{j+1/2}^+ \geq 0$, we need to ensure that $0 \leq C_{j+1/2}^- \leq 1$, which we have to enforce by the construction of ϕ as well as the choice of λ_a. Let us assume that ϕ is a function of θ, i.e. $\phi_j = \phi(\theta_j)$. Consequently, the corresponding slope can be written as $\sigma_j = \sigma(u_{j-1}, u_j, u_{j+1})$. With the choice of the so-called Courant-Friedrichs-Lewy (CFL) condition $|\lambda_a| \leq 1$ as well as

$$\left| \frac{\phi(\theta_j)}{\theta_j} - \phi(\theta_{j-1}) \right| \leq 2 \qquad \text{for all } \theta_j \text{ and } \theta_{j-1}, \tag{1.59}$$

we obtain $C_{j+1/2}^- \in [0,1]$, meaning that a scheme fulfilling (1.59) will be TVD. In order to obtain conditions that are easily verified, we take

$$0 \leq \frac{\phi(\theta)}{\theta} \leq 2, \qquad 0 \leq \phi(\theta) \leq 2, \quad \text{and } \phi(\theta) = 0 \quad \text{for } \theta \leq 0 \, .$$

Let us now interpret the different variables that have been introduced to obtain this result. To understand the role of ϕ, we write down the numerical flux that belongs to our initial scheme (1.58): With the definition of ϕ we get

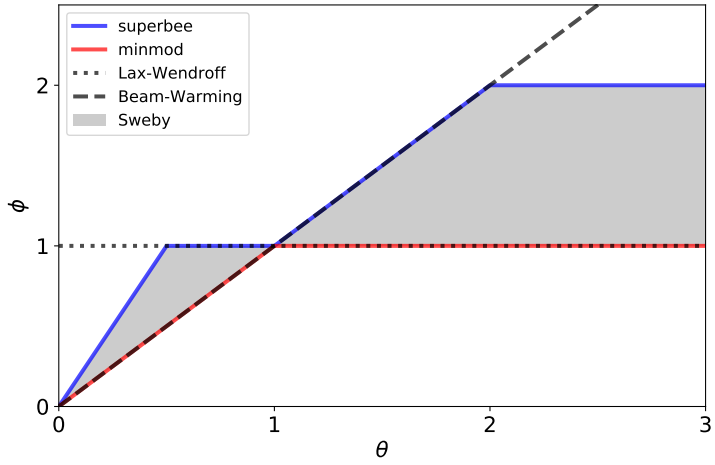
$$\begin{aligned} f^* =& a \left(u_j + \frac{\Delta x}{2} (1 - \lambda_a) \sigma_j \right) \\ =& a u_j + \frac{a}{2} (1 - \lambda_a) (u_{j+1} - u_j) \cdot \phi(\theta_j) \, . \end{aligned}$$

When $\phi(\theta) = 1$, this gives the second-order Lax-Wendroff flux [76] and $\phi(\theta) = \theta$ yields the second-order Beam-Warming flux [10]. It is important to note that here, second-order means second-order in space and time. It has been shown in [128], that a second-order method should lie inside the region defined by these two definitions of $\phi(\theta)$. A prominent choice of ϕ is the *minmod* limiter, which is given by

$$\phi_{\text{mm}}(\theta) := \max\{0, \min\{1, \theta\}\}.$$

The corresponding slope can be written as

$$\sigma(u, v, w) = \frac{1}{\Delta x} \text{minmod}(w - v, v - u) \tag{1.60}$$



with the minmod function

$$\text{minmod}(a,b) = \begin{cases} a & \text{if } |a| < |b|, ab > 0 \\ b & \text{if } |b| < |a|, ab > 0 \\ 0 & \text{else} \end{cases}.$$

This result can be extended to non-linear equations [78, Chapter 16.3.2]. The idea is to compute a piece-wise linear reconstruction of $f(u)$ in which the non-linear equation reduces to an advection equation at the cell boundaries. Commonly, this is done by replacing the term u_j inside the numerical flux by a higher-order approximation

$$u^-_{j+1/2} := u_j + \frac{\Delta x}{2}\sigma_j.$$

Similarly, the term u_{j+1} is replaced by

$$u^+_{j+1/2} := u_{j+1} - \frac{\Delta x}{2}\sigma_{j+1}.$$

The choice of the slope σ follows the idea of the linear analysis, i.e. one uses slopes, which lie in the Sweby region.

1.2.4 Bound preserving schemes

Note that the derived second-order scheme does not necessarily maintain desirable properties such as the discrete maximum principle. Recall that the maximum principle (1.39) is a key feature of scalar conservation laws and numerical methods aim at maintaining this characteristic, see for example [11, 20, 82, 148, 48]. Based on [149], the following section gives a brief overview on the construction of higher-order maximum-principle satisfying schemes. To construct higher-order schemes, the solution can be represented by a polynomial in each spatial cell. When denoting the polynomial in cell j by $p_j(x)$, the numerical scheme uses this solution reconstruction at the cell interfaces to evaluate the numerical flux. Recall that cell j is given by $I_j = [x_j, x_{j+1}]$. Defining $u^-_{j+1/2} := p_j(x_{j+1})$ and $u^+_{j+1/2} := p_{j+1}(x_{j+1})$, the higher-order scheme replaces the numerical flux $f^*(u_j, u_{j+1})$ in (1.33) by $f^*(u^-_{j+1/2}, u^+_{j+1/2})$. Now with $\lambda := \Delta t/\Delta x$ we modify the original first-order finite volume scheme, which

was

$$u_j^{n+1} = u_j^n - \lambda(f^*(u_j^n, u_{j+1}^n) - f^*(u_{j-1}^n, u_j^n)) =: h_\lambda(u_{j-1}^n, u_j^n, u_{j+1}^n).$$

Replacing the values in the numerical fluxes by the solution reconstructions at the cell interfaces yields

$$\begin{aligned} u_j^{n+1} =& u_j^n - \lambda(f^*(u_{j+1/2}^-, u_{j+1/2}^+) - f^*(u_{j-1/2}^-, u_{j-1/2}^+)) \\ =:& h_\lambda^{(2)}(u_j^n, u_{j+1/2}^-, u_{j+1/2}^+, u_{j-1/2}^-, u_{j-1/2}^+). \end{aligned} \tag{1.61}$$

Now we wish to derive conditions such that (1.61) fulfills a maximum principle. Similar to first-order schemes, we wish to show that $h_\lambda^{(2)}$ increases monotonically in every argument. A main idea to check monotonicity is expressing the cell average u_j^n in (1.61) by using a quadrature rule. When the polynomial p_j is of degree N, a Gauss-Lobatto quadrature rule with $Q = \lfloor (N+3)/2 \rfloor$ quadrature points $\hat{x}_j^k$, i.e. $x_j = \hat{x}_j^1 < \hat{x}_j^2 < \cdots < \hat{x}_j^Q = x_{j+1}$, and weights w_k can be used to express the cell averages by

$$u_j^n = \frac{1}{\Delta x}\int_{I_j} p_j(x)dx = \sum_{k=1}^{Q} w_k p_j(\hat{x}_j^k) = \sum_{k=2}^{Q-1} w_k p_j(\hat{x}_j^k) + w_1 p_j(\hat{x}_j^1) + w_Q p_j(\hat{x}_j^Q).$$

Now plugging this into the scheme (1.61) and adding

$$0 = \lambda\left(f^*(u_{j-1/2}^+, u_{j+1/2}^-) - f^*(u_{j-1/2}^+, u_{j+1/2}^-)\right)$$

yields

$$\begin{aligned} u_j^{n+1} =& \sum_{k=2}^{Q-1} w_k p_j(\hat{x}_j^k) + w_Q\left(u_{j+1/2}^- - \frac{\lambda}{w_Q}\left[f^*(u_{j+1/2}^-, u_{j+1/2}^+) - f^*(u_{j-1/2}^+, u_{j+1/2}^-)\right]\right) \\ &+ w_1\left(u_{j-1/2}^+ - \frac{\lambda}{w_1}\left[f^*(u_{j-1/2}^+, u_{j+1/2}^-) - f^*(u_{j-1/2}^-, u_{j-1/2}^+)\right]\right) \\ =& \sum_{k=2}^{Q-1} w_k p_j(\hat{x}_j^k) + w_Q h_{\lambda/w_Q}(u_{j-1/2}^+, u_{j+1/2}^-, u_{j+1/2}^+) \\ &+ w_1 h_{\lambda/w_1}(u_{j-1/2}^-, u_{j-1/2}^+, u_{j+1/2}^-). \end{aligned} \tag{1.62}$$

With $w := w_{1,Q} = \frac{1}{Q(Q-1)}$, the scheme $h_{\lambda/w}$ is monotone if the CFL condition

$$\max_u |f'(u)| \frac{\Delta t}{\Delta x} \le w_{1,Q} = \frac{1}{Q(Q-1)} \tag{1.63}$$

holds. Therefore, the same CFL condition ensures monotonicity of the high-order scheme $h_\lambda^{(2)}$. Using the same arguments as in Theorem 2 shows that the scheme $h_\lambda^{(2)}$ is bounded by u_- from below and by $u_+ > u_-$ from above if all input arguments in (1.62) have the same bounds. Therefore, it must be ensured that

$$p_j(\hat{x}_j^k), u_{j-1/2}^-, u_{j-1/2}^+, u_{j+1/2}^-, u_{j+1/2}^+ \in [u_-, u_+].$$

The bounds can be imposed on all input arguments by making use of a bound-preserving limiter

$$\tilde{p}_j(x) = \theta[p_j(x) - u_j] + u_j, \qquad \theta = \min\left\{1, \left|\frac{u_+ - u_j}{u_j^+ - u_j}\right|, \left|\frac{u_- - u_j}{u_j^- - u_j}\right|\right\},$$

with

$$u_j^+ = \max_{x \in I_j} p_j(x), \quad u_j^- = \min_{x \in I_j} p_j(x).$$

Besides imposing bounds on the numerical solution when the CFL condition (1.63) is used, the limiter does not destroy accuracy. For further details, see [149].

1.3 Kinetic equations

The *radiation transport* (or *linear Boltzmann*) equation is a linear integro-differential equation describing the movement of particles traveling through a background medium, through which particles can be absorbed or scattered. It plays a key role in various physics applications such as nuclear engineering [28, 55], high-energy astrophysics [85, 93], supernovae [37, 129] and fusion [88, 86]. The equation is given by

$$\partial_t \psi(t, \boldsymbol{x}, \boldsymbol{\Omega}) + \boldsymbol{\Omega} \cdot \nabla_{\boldsymbol{x}} \psi(t, \boldsymbol{x}, \boldsymbol{\Omega}) + \sigma_a(\boldsymbol{x}) \psi(t, \boldsymbol{x}, \boldsymbol{\Omega}) = \sigma_s(\boldsymbol{x})(S\psi)(t, \boldsymbol{x}, \boldsymbol{\Omega}) + q(t, \boldsymbol{x}, \boldsymbol{\Omega}). \tag{1.64}$$

In this equation, ψ is the *angular flux* which depends on time $t \in \mathbb{R}^+$, spatial position $\boldsymbol{x} = (x, y, z)^T \in \mathbb{R}^3$ and direction of travel $\boldsymbol{\Omega} \in \mathbb{S}^2$. The units are chosen so that particles travel with unit speed. The first two terms in (1.64) describe streaming, i.e. particles move in the direction $\boldsymbol{\Omega}$ without any interaction with the background material. The function $\sigma_a(\boldsymbol{x})$ is the *absorption cross-section* which describes the loss due to absorption by the material. The strength of scattering with the background medium is determined by the *scattering cross-section* σ_s. The operator describing scattering events is given by

$$(S\psi)(t, \boldsymbol{x}, \boldsymbol{\Omega}) = \left(\int_{\mathbb{S}^2} \sigma(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}') \psi(t, \boldsymbol{x}, \boldsymbol{\Omega}') \, d\boldsymbol{\Omega}' - \psi(t, \boldsymbol{x}, \boldsymbol{\Omega}) \right), \tag{1.65}$$

where $\sigma(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}')$ is the scattering kernel. The first term in the scattering operator (1.65) describes the loss and the second term describes the gain of particles with direction $\boldsymbol{\Omega}$ due to incoming scattering. Note that the mentioned cross-sections are also known as opacities. More generally, the cross-sections, and thereby the angular flux itself, can depend on frequencies, which we neglect in this work.

Numerically solving this equation is challenging, since the phase space of the solution (the angular flux) is at least six-dimensional, consisting of three spatial dimensions, two directional (angular) parameters and time. To reduce the phase space dimension, the angular dependency is discretized. Hence the task is to describe the dependency on the direction of travel

$$\boldsymbol{\Omega} = \left(\sqrt{1 - \mu^2} \cos(\varphi), \sqrt{1 - \mu^2} \sin(\varphi), \mu \right)^T \tag{1.66}$$

where $\mu \in [-1,1]$ and $\varphi \in [0,2\pi)$ with a finite number of parameters[4]. Various angular discretization strategies exist (cf. [16] for a comparison). Three of these strategies will be discussed in the following.

1.3.1 The spherical harmonics method

The spherical harmonics (P_N) method [18, 113, 81] expands the solution in terms of angular variables with finitely many spherical harmonics basis functions. For degree $\ell \geq 0$, order $k \in [-\ell,\ell]$ (where $\ell,k \in \mathbb{Z}$) and the associated Legendre function $P_\ell^k \in \mathbb{P}_\ell$, the spherical harmonics basis is given by

$$Y_\ell^k(\boldsymbol{\Omega}) = \sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-k)!}{(\ell+k)!}}e^{ik\varphi}P_\ell^k(\mu).$$

Defining the spherical bracket operator

$$\langle\cdot\rangle_{\boldsymbol{\Omega}} := \int_{\mathbb{S}^2} \cdot \, d\boldsymbol{\Omega},$$

we have that

$$\left\langle Y_\ell^k \overline{Y_{\ell'}^{k'}} \right\rangle_{\boldsymbol{\Omega}} = \delta_{\ell,\ell'}\delta_{k,k'},$$

i.e. the spherical harmonics are orthonormal. Furthermore, they are eigenfunctions of the scattering operator S: Expanding the scattering kernel in terms of Legendre polynomials $P_\ell := P_\ell^0$, where ℓ is the degree and $\int_{-1}^1 P_\ell(\mu)^2 \, d\mu = \frac{2}{2\ell+1}$ holds, gives the expansion

$$\sigma(\boldsymbol{x},\mu) = \sum_{\ell=0}^{\infty}\frac{2\ell+1}{4\pi}\sigma_\ell(\boldsymbol{x})P_\ell(\mu) \tag{1.67}$$

with $\sigma_\ell(\boldsymbol{x}) = 2\pi\int_{-1}^1 \sigma(\boldsymbol{x},\mu)P_\ell(\mu)\,d\mu$. Additionally, the Legendre polynomials can be expressed in the spherical harmonics basis as

$$P_\ell(\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}') = \frac{4\pi}{2\ell+1}\sum_{k=-\ell}^{\ell}\overline{Y_\ell^k}(\boldsymbol{\Omega}')Y_\ell^k(\boldsymbol{\Omega}). \tag{1.68}$$

The expansions (1.67) and (1.68) can be used to determine the eigenvalues of the scattering operator:

$$\begin{aligned}\int_{\mathbb{S}^2}\sigma(\boldsymbol{x},\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}')Y_\ell^k(\boldsymbol{\Omega}')\,d\boldsymbol{\Omega}' &= \int_{\mathbb{S}^2}\sum_{\ell'=0}^{\infty}\frac{2\ell'+1}{4\pi}\sigma_{\ell'}(\boldsymbol{x})P_{\ell'}(\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}')Y_\ell^k(\boldsymbol{\Omega}')\,d\boldsymbol{\Omega}' \\ &= \sum_{\ell'=0}^{\infty}\int_{\mathbb{S}^2}\sigma_{\ell'}(\boldsymbol{x})\sum_{k'=-\ell'}^{\ell'}\overline{Y_\ell^{k'}}(\boldsymbol{\Omega}')Y_{\ell'}^{k'}(\boldsymbol{\Omega})Y_\ell^k(\boldsymbol{\Omega}')\,d\boldsymbol{\Omega}' \\ &= \sum_{\ell'=0}^{\infty}\sum_{k'=-\ell'}^{\ell'}\sigma_{\ell'}(\boldsymbol{x})\delta_{\ell,\ell'}\delta_{k,k'}Y_{\ell'}^{k'}(\boldsymbol{\Omega}) = \sigma_\ell(\boldsymbol{x})Y_\ell^k(\boldsymbol{\Omega}).\end{aligned}$$

[4]Note that in the context of kinetic equations, φ denotes an angular variable, whereas in the context of uncertainty quantification, φ is used to denote polynomial basis functions.

Hence, the eigenvalues that correspond to the eigenfunctions Y_ℓ^k of the scattering operator S are given by $\sigma_s(1-\sigma_\ell)$. In the following, the real-valued spherical harmonics

$$m_\ell^k = \begin{cases} \frac{(-1)^k}{\sqrt{2}}(Y_\ell^k + (-1)^k Y_\ell^{-k}), & k < 0 \\ Y_\ell^0 & k = 0 \\ -\frac{(-1)^k i}{\sqrt{2}}(Y_\ell^{-k} + (-1)^k Y_\ell^k), & k > 0 \end{cases} \tag{1.69}$$

are used. The $n_\ell := 2\ell + 1$ real harmonic basis functions of degree ℓ are collected in a vector $\boldsymbol{m}_\ell$. All basis functions up to degree N are then stored in

$$\boldsymbol{m} = (\boldsymbol{m}_0, \boldsymbol{m}_1^T, \cdots, \boldsymbol{m}_N^T)^T \in \mathbb{R}^{(N+1)^2}.$$

These basis functions are used to span the spherical phase space of the solution, i.e. for a given set of expansion coefficients (or moments) $\hat{\boldsymbol{u}}(t,\boldsymbol{x}) \in \mathbb{R}^{(N+1)^2}$, the solution is approximated by

$$\psi(t,\boldsymbol{x},\boldsymbol{\Omega}) \approx \psi_{\mathrm{P}_N}(\hat{\boldsymbol{u}}(t,\boldsymbol{x});\boldsymbol{\Omega}) := \sum_{\ell=0}^{N}\sum_{k=-\ell}^{\ell} \hat{u}_\ell^k(t,\boldsymbol{x}) m_\ell^k(\boldsymbol{\Omega}) \tag{1.70}$$

$$= \hat{\boldsymbol{u}}(t,\boldsymbol{x})^T \boldsymbol{m}(\boldsymbol{\Omega}). \tag{1.71}$$

To gain an L^2 optimal solution representation, the moments should equal the Fourier coefficients of the angular flux, i.e. $u_\ell^k \equiv \langle \psi m_\ell^k \rangle_{\boldsymbol{\Omega}}$. A time evolution equation for the moments can then be derived by multiplying the radiation transport equation with the real harmonics (1.69) and integrating over the angular domain, yielding

$$\partial_t \hat{\boldsymbol{u}} + \sum_{j=1}^{d} \partial_{x_j} \langle \Omega_j \psi \boldsymbol{m} \rangle_{\boldsymbol{\Omega}} + \sigma_a \hat{\boldsymbol{u}} + \sigma_s \boldsymbol{G} \hat{\boldsymbol{u}} = \hat{\boldsymbol{q}}. \tag{1.72}$$

Note that we omit the dependency on the phase space for better readability. Here, $\hat{\boldsymbol{q}} = \langle q\boldsymbol{m} \rangle_{\boldsymbol{\Omega}}$ is the moment vector of the source and $\boldsymbol{G}$ is a diagonal matrix with entries $G_{(\ell,k),(\ell,k)} = 1 - \sigma_\ell$ resulting from the eigenvector relation of the scattering operator. Unfortunately, the derived moment system (1.72) is not closed, meaning that it does not solely depend on the moment vector $\hat{\boldsymbol{u}}$. Therefore, the angular flux showing up in the flux term is replaced by its polynomial representation (1.70). Plugging this solution ansatz into (1.72) yields the hyperbolic P_N *equations*

$$\partial_t \hat{\boldsymbol{u}} + \sum_{j=1}^{d} \boldsymbol{A}_j \partial_{x_j} \hat{\boldsymbol{u}} + \sigma_a \hat{\boldsymbol{u}} + \sigma_s \boldsymbol{G} \hat{\boldsymbol{u}} = \hat{\boldsymbol{q}}. \tag{1.73}$$

The flux Jacobians $\boldsymbol{A}_j$ are given by the recursion relation of the spherical harmonics. With

$$\Omega_j \boldsymbol{m}_\ell = \left(\boldsymbol{a}_\ell^{(j)}\right)^T \boldsymbol{m}_{\ell-1} + \boldsymbol{a}_{\ell+1}^{(j)} \boldsymbol{m}_{\ell+1},$$

the flux Jacobians read

$$\boldsymbol{A}_j = \begin{pmatrix} 0 & \boldsymbol{a}_1^{(j)} & & \\ \left(\boldsymbol{a}_1^{(j)}\right)^T & \boldsymbol{0} & \boldsymbol{a}_2^{(j)} & \\ & \ddots & \ddots & \ddots \\ & & \left(\boldsymbol{a}_N^{(j)}\right)^T & \boldsymbol{0} \end{pmatrix}.$$

The submatrices $\boldsymbol{a}_\ell^{(j)}$ are given by

$$\boldsymbol{\Omega} m_\ell^k = \frac{1}{2} \begin{pmatrix} (1-\delta_{k,-1})(\tilde{c}_{\ell-1}^{|k|-1} m_{\ell-1}^{k^-} - \tilde{d}_{\ell+1}^{|k|-1} m_{\ell+1}^{k^-}) - \tilde{e}_{\ell-1}^{|k|+1} m_{\ell-1}^{k^+} + \tilde{f}_{\ell+1}^{|k|+1} m_{\ell+1}^{k^+} \\ \operatorname{sign}(k) \left((1-\delta_{k,1})(-\tilde{c}_{\ell-1}^{|k|-1} m_{\ell-1}^{-k^-} + \tilde{d}_{\ell+1}^{|k|-1} m_{\ell+1}^{-k^-}) - \tilde{e}_{\ell-1}^{|k|+1} m_{\ell-1}^{-k^+} + \tilde{f}_{\ell+1}^{|k|+1} m_{\ell+1}^{-k^+} \right) \\ 2(a_{\ell-1}^k m_{\ell-1}^k + b_{\ell+1}^k m_{\ell+1}^k) \end{pmatrix}, \tag{1.74}$$

where we modify the sign notation such that $\operatorname{sign}(0) = 1$ holds. Furthermore, we use the notation

$$k^+ = k + \operatorname{sign}(k), \qquad k^- = k - \operatorname{sign}(k).$$

Coefficients used in (1.74) are

$$\tilde{c}_\ell^k = \begin{cases} 0, & k < 0, \\ \sqrt{2} c_\ell^k, & k = 0, \\ c_\ell^k, & k > 0, \end{cases} \qquad \tilde{d}_\ell^k = \begin{cases} 0, & k < 0, \\ \sqrt{2} d_\ell^k, & k = 0, \\ d_\ell^k, & k > 0, \end{cases}$$
$$\tilde{e}_\ell^k = \begin{cases} \sqrt{2} e_\ell^k, & k = 1, \\ e_\ell^k, & k > 1, \end{cases} \qquad \tilde{f}_\ell^k = \begin{cases} \sqrt{2} f_\ell^k, & k = 1, \\ f_\ell^k, & k > 1, \end{cases}$$

with

$$a_\ell^k = \sqrt{\frac{(\ell-k+1)(\ell+k+1)}{(2\ell+3)(2\ell+1)}}, \qquad , b_\ell^k = \sqrt{\frac{(\ell-k)(\ell+k)}{(2\ell+1)(2\ell-1)}},$$
$$c_\ell^k = \sqrt{\frac{(\ell+k+1)(\ell+k+2)}{(2\ell+3)(2\ell+1)}}, \qquad d_\ell^k = \sqrt{\frac{(\ell-k)(\ell-k-1)}{(2\ell+1)(2\ell-1)}},$$
$$e_\ell^k = \sqrt{\frac{(\ell-k+1)(\ell-k+2)}{(2\ell+3)(2\ell+1)}}, \qquad f_\ell^k = \sqrt{\frac{(\ell+k)(\ell+k-1)}{(2\ell+1)(2\ell-1)}}.$$

P_N shows spectral convergence while maintaining rotational invariance. Unfortunately, the P_N method suffers from oscillations in non-smooth regimes, which can lead to non-physical, negative values of the solution, see Figure 1.1.

1.3.2 The minimal entropy method

To circumvent the issue of negative solution values, the *minimal entropy* (M_N) [95, 80] method has been introduced, which we briefly mention in the following. Its main idea is to not represent the solution by a polynomial ansatz of the form (1.70), but

with a constraint optimization problem

$$\psi_{\mathrm{M}_N} := \arg\min_{\psi} \eta(\psi) \qquad \text{subject to } \langle \boldsymbol{m}\psi \rangle_{\boldsymbol{\Omega}} = \hat{\boldsymbol{u}}. \tag{1.75}$$

Here, an entropy, which is commonly denoted by η in the context of kinetic theory, is minimized under a moment constraint. The constraint states that the minimizer must have the same moments as $\hat{\boldsymbol{u}}$. It can be shown that the resulting minimizer uniquely depends on the moment vector $\hat{\boldsymbol{u}}$, i.e. we have created a closure, which can be used to close the moment system (1.72). Several properties of minimal entropy methods will be derived for the uncertainty quantification framework, see Sections 1.4.5 and 1.4.6. Therefore, we only briefly mention several properties of the M_N method and leave the derivation to later sections. First, when choosing the Boltzmann entropy function $\eta(\psi) = \psi \log(\psi) - \psi$, the minimizer (1.75) takes the form $\psi_{\mathrm{M}_N} = \exp(\boldsymbol{\lambda}^T \boldsymbol{m})$, where $\boldsymbol{\lambda} \in \mathbb{R}^{(N+1)^2}$ are the Lagrange multipliers belonging to (1.75). Therefore, the resulting solution reconstruction remains positive. Furthermore, the resulting moment system will be hyperbolic: When plugging the M_N solution ansatz into (1.72) and applying the chain rule, one gets

$$\boldsymbol{H}(\boldsymbol{\lambda})\partial_t \boldsymbol{\lambda} + \sum_{j=1}^{d} \partial_{x_j} \boldsymbol{B}_j(\boldsymbol{\lambda})\boldsymbol{\lambda} = \boldsymbol{0}.$$

Here,

$$\boldsymbol{H}(\boldsymbol{\lambda}) = \langle \exp(\boldsymbol{\lambda}^T \boldsymbol{m}) \boldsymbol{m}\boldsymbol{m}^T \rangle \quad \text{and } \boldsymbol{B}_j(\boldsymbol{\lambda}) = \langle \Omega_j \exp(\boldsymbol{\lambda}^T \boldsymbol{m}) \boldsymbol{m}\boldsymbol{m}^T \rangle,$$

where $\boldsymbol{H}$ is symmetric positive definite and the flux Jacobians $\boldsymbol{B}_j$ are symmetric. Hence, following (1.19), hyperbolicity of the moment system holds. For an efficient way of finding the minimizer (1.75), we refer to Sections 1.4.5 and 1.4.6.

1.3.3 The discrete ordinates method

A frequently used method, which ensures positivity of the angular flux is the discrete ordinates (S_N) method [81]. The core strategy to discretize the angular domain is to use a discrete nodal set of $Q \in \mathbb{N}$ possible directions (also called ordinates)

$$\{\boldsymbol{\Omega}_1, \cdots, \boldsymbol{\Omega}_Q\} \subset \mathbb{S}^2.$$

Then, a nodal representation of the solution is obtained by

$$\psi_q(t, \boldsymbol{x}) := \psi(t, \boldsymbol{x}, \boldsymbol{\Omega}_q) \quad \text{with } q = 1, \cdots, Q.$$

Commonly, these ordinates are chosen according to a quadrature rule, i.e. with adequate quadrature weights w_q, we have

$$\int_{\mathbb{S}^2} \sigma(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}')\psi(t, \boldsymbol{x}, \boldsymbol{\Omega}')\, d\boldsymbol{\Omega}' \approx \sum_{q=1}^{Q} w_q \sigma(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}_q)\psi_q(t, \boldsymbol{x}).$$

Evaluated at the different ordinates, the radiative transfer equation (1.64) when omitting the source term becomes

$$\partial_t \psi_q(t,\boldsymbol{x}) + \boldsymbol{\Omega}_q \cdot \nabla_{\boldsymbol{x}} \psi_q(t,\boldsymbol{x}) + \sigma_t(\boldsymbol{x})\psi_q(t,\boldsymbol{x}) = \sigma_s(\boldsymbol{x}) \sum_{p=1}^{Q} w_p \sigma(\boldsymbol{x}, \boldsymbol{\Omega}_q \cdot \boldsymbol{\Omega}_p)\psi_p(t,\boldsymbol{x})$$

with $q = 1, \cdots, Q$ and $\sigma_t(\boldsymbol{x}) := \sigma_a(\boldsymbol{x}) + \sigma_s(\boldsymbol{x})$. Hence, we obtain a set of equations, which describe the time evolution of the angular flux on the chosen set of quadrature points. The equations for the individual ordinates couple through the discretized scattering term. A common choice to discretize the angular components (1.66) uses a tensorized quadrature on the sphere with Gauss quadrature for μ and equally weighted and spaced points for φ. Hence, the discrete points for φ_i read

$$\varphi_i = i\Delta\varphi \quad \text{for} \quad i = 1, \ldots, N_q \quad \text{and} \quad \Delta\varphi = \frac{2\pi}{N_q}. \tag{1.76}$$

If the Gauss quadrature for μ uses N_q points, then we obtain a total of $Q = N_q^2$ possible directions. Note that here and throughout this thesis we denote the total number of quadrature points in all dimensions by Q and the number of points for one dimension by N_q. A main disadvantage of S_N methods are so called ray-effects [74, 100, 87], which are spurious artifacts that stem from the limited number of directions in which particles can travel. Figure 1.1 shows the scalar flux

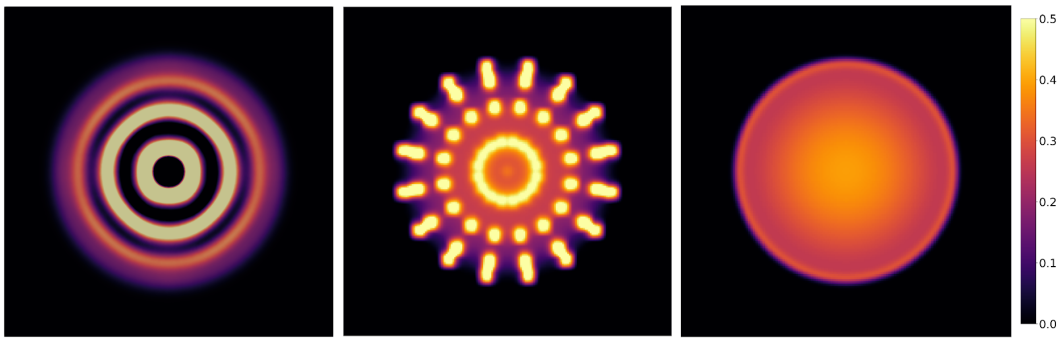


FIGURE 1.1: Comparison of the scalar flux ϕ with different methods for the line-source problem. The line-source problem assumes a high particle distribution in the center of the spatial domain, surrounded by a void. As time increases, the particles travel into the void region. The reference solution as well as the P_N and S_N approximations are shown at time $t = 1$. From left to right: P_7, S_8, reference solution. For more information on the chosen parameters and initial condition, see Section 8.1.3.

$$\Phi(t,\boldsymbol{x}) = \int_{S^2} \psi(t,\boldsymbol{x},\boldsymbol{\Omega}')\, d\boldsymbol{\Omega}'$$

computed with P_N and S_N for the line-source test case, which puts a large number of particles in the center of the spatial domain, which then travel in all directions. For more information regarding the line-source problem, see Section 8.1.3. It can be seen that while P_N oscillates, the S_N solution shows ray-effects. Note, that the chosen scale only depicts positive solution values. While the S_N solution remains positive, the oscillations of the P_N solution yield negative, non-physical values of the scalar flux.

1.4 Uncertainty Quantification

Given that accurate numerical solutions to the presented equations exist, computational scientists are increasingly concerned with how uncertainties in the "input" data, such as initial/boundary conditions, constitutive relations, and other parameters affect the conclusions drawn from computer simulations. Answering such questions are the purview of the field of uncertainty quantification (UQ) [89]. In addition to time, space and (in the case of kinetic equations) angular variables, *random hyperbolic equations* additionally depend on uncertainties, which further extend the phase space. Before stating hyperbolic equations containing uncertainty, let us briefly summarize certain fundamentals of probability theory based on [126, Chapter 2]:

Definition 7. *A triplet $(\Omega, \mathcal{F}, P)$ with Ω being a set of possible outcomes of a random experiment, $\mathcal{F}$ being its σ-algebra and P being a probability measure is called a probability space. A probability measure fulfills*

1. $0 \leq P(A) \leq 1 \qquad \forall A \in \mathcal{F}$,
2. $P(\Omega) = 1$,
3. $P(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$ *for a sequence* $A_1, A_2, \cdots$ *with* $A_i \cap A_j = \emptyset$.

To quantify the outcome of a random experiment, one commonly makes use of mappings which have a probability space as domain:

Definition 8. *Assume that $(\Omega, \mathcal{F}, P)$ is a probability space and $(E, \mathcal{G})$ is a measurable space. Let us define a function $f : \Omega \to E$, which generates a σ-algebra*

$$\sigma(f) := \sigma\left(\{\{\omega \in \Omega : f(\omega) \in Y\} : Y \in \mathcal{G}\}\right)$$

on Ω. Now f is called a measurable function if $\sigma(f) \subset \mathcal{F}$. A measurable function whose domain is a probability space is called a random variable (or E-valued random variable).

This rather abstract definition of a random variable will later become useful, since we will be interested in random variables which map onto function spaces. For now, let us assume a scalar random variable $X : \Omega \to \mathbb{R}$. The *cumulative distribution function* and the *probability density function* of such a random variable are given by the following definition:

Definition 9. *The function*

$$F_X(x) := P\left(\{\omega \in \Omega : X(\omega) \leq x\}\right)$$

is called a cumulative distribution function. It is given by the area under the so-called probability density function $f_X : \Omega \to \mathbb{R}$, i.e.

$$F_X(x) = \int_{-\infty}^{x} f_X(\bar{x}) d\bar{x}.$$

Let us now assume that the outcome of a random experiment $\omega \in \Omega$ defines a hyperbolic conservation law, which can be described by a *random hyperbolic equation*

$$\partial_t w(t, \boldsymbol{x}, \omega) + \nabla \cdot \boldsymbol{f}(w(t, \boldsymbol{x}, \omega)) = 0 \quad \text{in } D, \tag{1.77a}$$

$$w(t = 0, \boldsymbol{x}, \omega) = w_{\mathrm{IC}}(\boldsymbol{x}, \omega). \tag{1.77b}$$

For ease of presentation, we assume that the randomness enters through the initial condition. Furthermore, we assume a scalar conservation law. Several concepts of hyperbolic equations apply for such problems. Following [97], we assume that the initial condition is an $L^1(D)$-random variable, i.e.

$$w_{IC} : (\Omega, \mathcal{F}) \to (L^1(D), \mathcal{B}(L^1(D))),$$

where $\mathcal{B}$ denotes the Borel σ-field, which roughly speaking applies the concept of the Borel σ-algebra to function spaces. For more information, see [97]. Furthermore, let

$$P(\{\omega \in \Omega : w_{IC}(\cdot, \omega) \in (L^\infty \cap BV)(D)\}) = 1,$$

which we denote by

$$w_{IC}(\cdot, \omega) \in L^\infty(D) \cap BV(D) \;\; P\text{-a.s.}\,.$$

Note that *P-almost surely* (P-a.s.) can be interpreted as the notion of *almost everywhere* applied to probability spaces. Then, the previously presented notion of the entropy solution can be adopted for random hyperbolic equations: A function

$$w : (\Omega, \mathcal{F}) \to L^\infty([0, T]; L^1(D)),$$

i.e. a random field $\omega \to w(t, x, \omega)$ is called a *random entropy solution* if it fulfills the weak solution and entropy condition for random hyperbolic equations, which are

1. A solution fulfilling

$$\int_{\mathbb{R}_+ \times \mathbb{R}} w(t, \boldsymbol{x}, \omega)\phi_t(t, \boldsymbol{x}) + \sum_{j=1}^{d} f_j(w(t, \boldsymbol{x}, \omega))\partial_{x_j}\phi(t, \boldsymbol{x})\, d\boldsymbol{x}dt + \int_{\mathbb{R}} w_{\mathrm{IC}}(\boldsymbol{x}, \omega)\phi(0, \boldsymbol{x})\, d\boldsymbol{x} = 0, \;\; \forall \phi \in C_0^\infty(\mathbb{R}_+ \times \mathbb{R}^d)$$

   P-a.s. for $\omega \in \Omega$ is called a weak solution in the context of random hyperbolic equations.

2. A weak solution fulfilling the entropy inequality P-a.s. for all admissible entropies in a weak sense, i.e.

$$\int_{\mathbb{R}_+ \times \mathbb{R}} s(w(t, \boldsymbol{x}, \omega))\phi_t(t, \boldsymbol{x}) + \sum_{j=1}^{d} \tilde{F}_j(w(t, \boldsymbol{x}, \omega))\partial_{x_j}\phi(t, \boldsymbol{x})\, d\boldsymbol{x}dt \geq 0,$$

   for all $0 \leq \phi \in C_0^\infty(\mathbb{R}_+ \times \mathbb{R}^d)$ is called a random entropy solution.

For scalar equations it can be shown that if the initial condition is strongly measurable, the L^1 stability (1.25a) ensures strong measurability of the random entropy solution. If the randomness only enters through the initial condition, the standard stability results of hyperbolic equations, i.e. (1.25a), (1.25b) and (1.28) hold P-a.s. for $\omega \in \Omega$. Furthermore, when defining the norm

$$\|w\|_{L^p(\Omega; L^\infty([0,T]; L^1(D)))} := \left(\int_\Omega \|w\|^p_{L^\infty([0,T]; L^1(D))}\, dP(\omega) \right)^{\frac{1}{p}}$$

we have that $\|w\|_{L^p(\Omega;L^\infty([0,T];L^1(D)))} < \infty$, i.e. together with strong measurability, we know that the solution is Bochner integrable. A more thorough discussion can for example be found in [116]. Since we interpret uncertainties as parameters which we propagate through a given mathematical model, we will not go into further details in this work. For more information on strong measurability and Bochner spaces we refer to [139, Chapter 1]. To propagate uncertainties through a given (not necessarily scalar) differential equation, many authors treat random equations of the form (1.77) as *parametric hyperbolic equations*, see e.g. [27, 112, 110, 46]. For a vector of p parameters $\boldsymbol{\xi} \in \Theta \subseteq \mathbb{R}^p$ we wish to solve

$$\partial_t \boldsymbol{u}(t,\boldsymbol{x},\boldsymbol{\xi}) + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}(t,\boldsymbol{x},\boldsymbol{\xi})) = \boldsymbol{0} \quad \text{in } D, \tag{1.78a}$$

$$\boldsymbol{u}(t=0,\boldsymbol{x},\boldsymbol{\xi}) = \boldsymbol{u}_{\mathrm{IC}}(\boldsymbol{x},\boldsymbol{\xi}). \tag{1.78b}$$

Note that now the solution depends on the parameter vector $\boldsymbol{\xi}$ instead of the outcome of a random experiment $\omega \in \Omega$. By assigning a probability density function $f_{\Xi,i}(\xi_i)$ to each parameter ξ_i, we can interpret the parameter vector $\boldsymbol{\xi}$ as a vector of independently distributed random variables $\boldsymbol{\xi} : \Omega \to \Theta$ with probability density function $f_\Xi(\boldsymbol{\xi}) := \prod_{i=1}^p f_{\Xi,i}(\xi_i)$. Therefore, we will refer to $\boldsymbol{\xi}$ as a vector of random variables. If the uncertainty in the solution to (1.77) can be described by this vector of random variables, i.e. when assuming that the (not necessarily scalar) solution to (1.77) can be written as

$$\boldsymbol{w}(t,\boldsymbol{x},\omega) = \boldsymbol{u}(t,\boldsymbol{x},\boldsymbol{\xi}(\omega)),$$

parametric equations of the form (1.78) can be seen as random hyperbolic equations when dropping the dependency on ω. This approach is motivated by the gPC expansion, which states that if a quantity $w(\omega)$ is a second-order random field, i.e. if

$$\int_\Omega w(\omega)^2 dP(\omega) < \infty,$$

it can be represented by an infinite expansion

$$w(\omega) = \sum_{i=0}^{\infty} \hat{w}_i \varphi_i(\boldsymbol{\xi}(\omega))$$

with suitable basis functions φ_i. Not that in general, the parameter vector $\boldsymbol{\xi}$ can have an infinite length. The gPC expansion will be discussed in greater detail in Section 1.4.2. For a more detailed discussion of the representation of random fields by a gPC expansion, we refer to [43, Chapter 2.4]. Statistical quantities such as the *expectation value* or the *variance* can then be obtained by

$$\mathrm{E}[\boldsymbol{u}](t,\boldsymbol{x}) = \langle \boldsymbol{u}(t,\boldsymbol{x},\cdot)\rangle, \qquad \mathrm{Var}[\boldsymbol{u}](t,\boldsymbol{x}) = \langle (\boldsymbol{u}(t,\boldsymbol{x},\cdot) - \mathrm{E}[\boldsymbol{u}](t,\boldsymbol{x}))^2\rangle,$$

where we use the bracket operator to define an integration weighted by the probability density function f_Ξ, i.e.

$$\langle g\rangle := \int_\Omega g(\boldsymbol{\xi}(\omega)) dP(\omega) = \int_\Theta g(\boldsymbol{\xi}) f_\Xi(\boldsymbol{\xi}) d\xi_1 \cdots d\xi_p. \tag{1.79}$$

Note that the expectation value is often referred to as *expectation* or *expected value* in the literature, however we will make use of the name expectation value (though strictly speaking this is not the correct expression) throughout this thesis.

1.4.1 Examples of hyperbolic equations with uncertainty

To give a clearer picture of the effects uncertainties have on the solution to hyperbolic conservation laws, let us state two of those equations and investigate how uncertainties are propagated through these models. A very simple example of a scalar hyperbolic equation is Burgers' equation, which reads

$$\partial_t u(t,x,\xi) + \partial_x \frac{u(t,x,\xi)^2}{2} = 0, \tag{1.80a}$$

$$u(t=0,x,\xi) = u_{\mathrm{IC}}(x,\xi). \tag{1.80b}$$

As in [112], we choose the random initial condition

$$u_{\mathrm{IC}}(x,\xi) := \begin{cases} u_L, & \text{if } x < x_0 + \sigma\xi \\ u_L + \frac{u_R - u_L}{x_0 - x_1}(x_0 + \sigma\xi - x), & \text{if } x \in [x_0 + \sigma\xi, x_1 + \sigma\xi] \\ u_R, & \text{else} \end{cases}, \tag{1.81}$$

which for $\xi = 0$ is a linear connection from x_0 to x_1. To determine the time evolution of this system, one can use characteristics, which show that the solution moves with a velocity of $f'(u) = u$. As time increases, solution values with a value of $u_L > u_R$, will move faster to the right than solution values of u_R, meaning that a shock will form at time $t_s = \frac{x_1 - x_0}{u_L - u_R}$. The speed of this shock can be determined by the Rankine-Hugoniot condition (1.7), which gives $v_{shock} = \frac{1}{2}(u_L - u_R)$. In our case, ξ is uniformly distributed on the interval $[-1,1]$ with $\sigma = 0.2$, leading to an uncertain initial position of the linear connection, given by $x_0 + \sigma\xi$ and $x_1 + \sigma\xi$. As time increases, shocks will form at the deterministic time t_s at different spatial positions for different values of ξ. The corresponding expectation value and variance can be found in Figure 1.2. Once the shocks have formed, the expectation value is a linear connection between the shock positions for $\xi \in \{-1,1\}$. The support of the corresponding variance lies inside these two shock positions and attains a maximum in the center.

The second equations we consider describe the time evolution of a gas, which can be described by the Euler equations, see Example 1. The Euler equations read

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix} + \partial_x \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(\rho E + p) \end{pmatrix} = \mathbf{0}, \tag{1.82}$$

where ρ is the gas density, ρu is momentum and ρE is the total energy. One can determine the pressure p from

$$p = (\gamma - 1)\rho\left(E - \frac{1}{2}u^2\right).$$

The heat capacity ratio is γ and has a value of 1.4 for air. For this system, we consider Sod's shock tube experiment [124] with an uncertain shock position, see for example [112, 117, 70]. This test case describes the evolution of the gas which initially shows

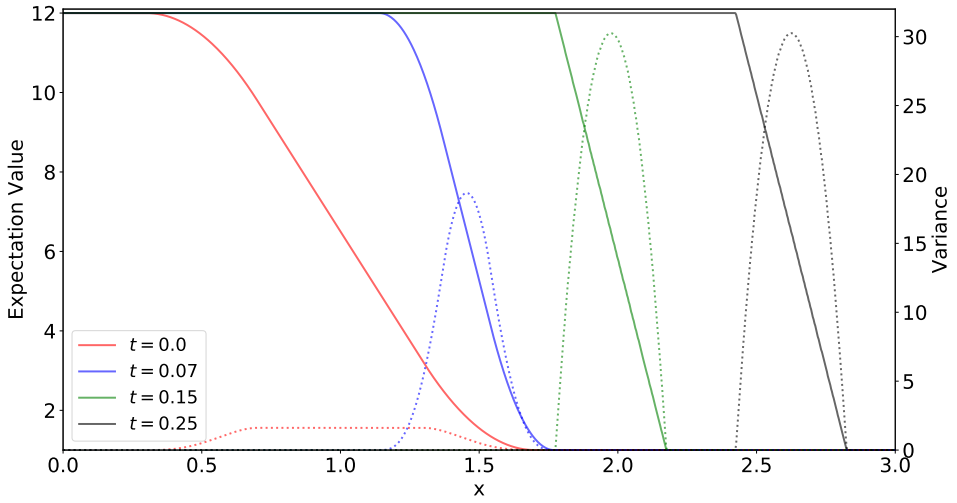


FIGURE 1.2: Expectation value (solid line) and variance (dotted lines) of Burgers' test case for different times. Note that the variance does not have the same units as the expectation value and one often uses the standard deviation $\sqrt{\mathrm{Var}[u]}$ to draw conclusions regarding the uncertainty of the solution.

a shock profile, i.e. we have the initial condition

$$\rho_{\mathrm{IC}} = \begin{cases} \rho_L & \text{if } x < x_{\mathrm{interface}}(\xi) \\ \rho_R & \text{else} \end{cases},$$
$$(\rho u)_{\mathrm{IC}} = 0,$$
$$(\rho E)_{\mathrm{IC}} = \begin{cases} (\rho E)_L & \text{if } x < x_{\mathrm{interface}}(\xi) \\ (\rho E)_R & \text{else} \end{cases}.$$

In this example, the initial shock position $x_{\mathrm{interface}} \in D = [0,1]$ is assumed to be uncertain and we use $x_{\mathrm{interface}}(\xi) = x_0 + \sigma\xi$, where ξ is uniformly distributed in the interval $[-1,1]$ and $\sigma = 0.05$. The initial states of Sod's shock tube experiment are $\rho_L = 1.0, (\rho E)_L = 2.5$ and $\rho_R = 0.125, (\rho E)_R = 0.25$. For the deterministic case, the system will form three solution characteristics, which travel with different speeds, namely a rarefaction wave, a contact discontinuity and a shock. When including the uncertainty inside the initial shock position, the resulting expectation value and variance can be found in Figure 1.3. The expectation value of the contact discontinuity and the shock will be linear connections between the respective shock position for $\xi \in \{-1,1\}$, while the expectation value of the rarefaction wave shows a smooth profile, traveling to the left of the spatial domain. All three regions show an increased variance.

1.4.2 Discretization of the random dimension

Numerical methods aim at recovering the previously discussed results and the discretization strategy for the random domain crucially affects the resulting solution quality. In the following, let us discuss how to represent the dependency on the

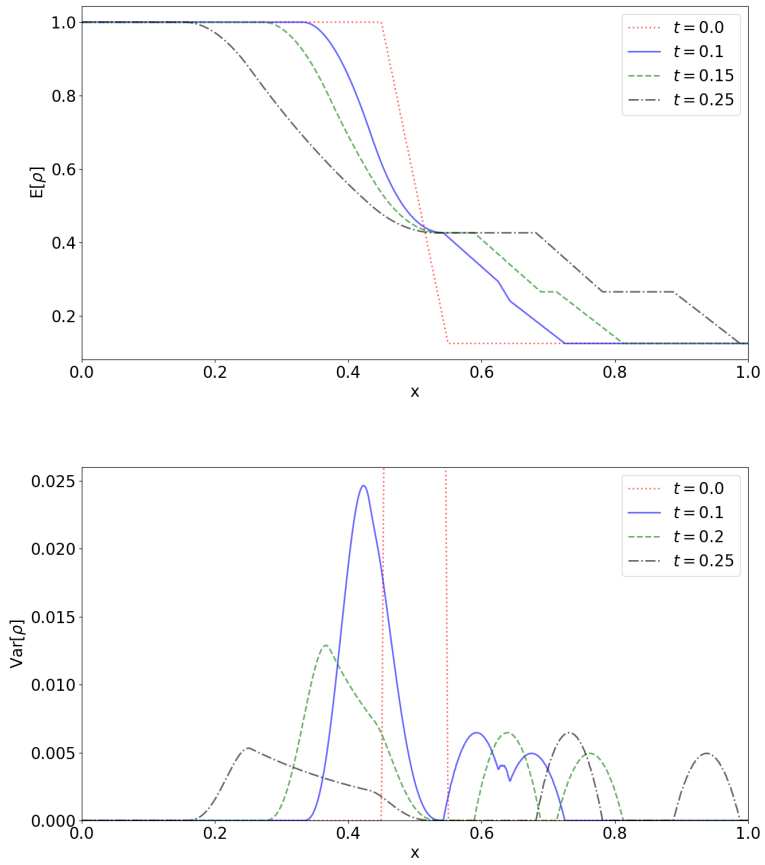


FIGURE 1.3: Expectation value (top) and variance (bottom) of Sod's shock tube problem for different times. The three solution regions belong to the rarefaction wave (left), the contact discontinuity (center) and the shock (right).

random domain Θ with finitely many unknowns. Therefore, let us for now assume that the solution $\boldsymbol{u}$ only depends on a vector of random variables and no longer on space and time. A popular discretization approach is to span the solution with a set of polynomial basis functions $\varphi_i : \Theta \to \mathbb{R}$ such that for the multi-index $i = (i_1, \cdots, i_p) \in \mathbb{N}_0^p$ we have $|i| \leq N$. Note that this yields

$$M := \binom{N+p}{p} \tag{1.83}$$

basis functions when defining $|i| := \sum_{n=1}^{p} |i_n|$. Commonly, these functions are chosen to be orthonormal polynomials [142] with respect to the probability function, i.e. $\langle \varphi_i \varphi_j \rangle = \prod_{n=1}^{p} \delta_{i_n j_n}$. The *generalized polynomial chaos* (gPC) expansion [147] approximates the solution by

$$\mathcal{U}(\hat{\boldsymbol{u}}; \boldsymbol{\xi}) := \sum_{|i| \leq N} \hat{\boldsymbol{u}}_i \varphi_i(\boldsymbol{\xi}) = \hat{\boldsymbol{u}}^T \boldsymbol{\varphi}(\boldsymbol{\xi}), \tag{1.84}$$

where the deterministic expansion coefficients $\hat{\boldsymbol{u}}_i \in \mathbb{R}^m$ are called *moments* or *gPC coefficients*. To allow a more compact notation, we collect the M moments for which $|i| \leq N$ holds in the moment matrix $\hat{\boldsymbol{u}} := (\hat{\boldsymbol{u}}_i)_{|i| \leq N} \in \mathbb{R}^{M \times m}$ and the corresponding

basis functions in $\boldsymbol{\varphi} := (\varphi_i)_{|i|\leq N} \in \mathbb{R}^M$. In the following, the dependency of $\mathcal{U}$ on $\boldsymbol{\xi}$ will occasionally be omitted for sake of readability. The solution ansatz (1.84) is L^2-optimal if the moments are chosen to be the Fourier coefficients

$$\hat{\boldsymbol{u}}_i \equiv \langle \boldsymbol{u}\varphi_i \rangle \in \mathbb{R}^m. \tag{1.85}$$

Optimality means that the expansion (1.84) minimizes the weighted L^2 error, i.e. we use the probability density function f_Ξ as a weight inside the norm. Then, we have

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^{M\times m}}{\arg\min} \left\langle \|\mathcal{U}(\boldsymbol{\alpha}) - \boldsymbol{u}\|_2^2 \right\rangle,$$

where $\|\cdot\|_2$ denotes the Euclidean norm.[5] The ansatz (1.84) can be used to compute the quantities of interest. Indeed, we have that

$$\mathrm{E}[\mathcal{U}(\hat{\boldsymbol{u}})] = \hat{\boldsymbol{u}}_0, \quad \mathrm{Var}[\mathcal{U}(\hat{\boldsymbol{u}})] = \mathrm{E}[\mathcal{U}(\hat{\boldsymbol{u}})^2] - \mathrm{E}[\mathcal{U}(\hat{\boldsymbol{u}})]^2 = \left(\sum_{i=1}^{M} \hat{u}_{i\ell}^2 \right)_{\ell=1,\cdots,m}.$$

The gPC reconstruction (1.84) shows good accuracy when $\boldsymbol{u}$ is sufficiently smooth with respect to the random dimension. However, when representing non-smooth solutions, the gPC expansion yields oscillatory approximations which can potentially violate important properties of hyperbolic conservation laws.

To obtain a mathematical expression for the smoothness of a function, let us define a the Sobolev space $H_\Xi^{q,\infty}$: For a multi-index i, a function $u : \Theta \subset \mathbb{R}^p \to \mathbb{R}$ has mixed derivatives

$$\partial^{(i)} u := \frac{\partial^{i_1}}{\partial \xi_1^{i_1}} \cdots \frac{\partial^{i_p}}{\partial \xi_p^{i_p}} u. \tag{1.86}$$

Then, with the norm

$$\|u\|_{H_\Xi^{q,\infty}} := \left(\sum_{|i|_\infty \leq q} \|\partial^{(i)} u\|_{L_\Xi^2}^2 \right)^{1/2},$$

we have $u \in H_\Xi^{q,\infty}$ if $\partial^{(i)} u \in L_\Xi^2$ for $|i|_\infty := \max_{i_1,\cdots,i_p} |i| \leq q$. Here, the space L_Ξ^2 is the L^2 space weighted with the probability density function f_Ξ. We have that since $|i|_\infty \leq q$, an index q requires all derivatives up to a *maximal* order q to lie in L_Ξ^2. This requirement is also needed to prove the convergence rate of sparse grids (as for example remarked in [136]), which we briefly discuss in Section 1.4.4. The Sobolev index q now reflects how accurately the gPC expansion (1.84) approximates the function u which can be seen by the following theorem:

Theorem 4. *Let $u \in H_\Xi^{q,\infty}([-1,1]^p)$, i.e. $\boldsymbol{\xi}$ is uniformly distributed in $\Theta = [-1,1]^p$. Then, there exists a constant C such that*

$$\|\mathcal{U}(\hat{\boldsymbol{u}}) - u\|_{L_\Xi^2([-1,1]^p)} \leq C N^{-q} \|u\|_{H_\Xi^{q,\infty}([-1,1]^p)}.$$

[5] Note that, in order to simplify notation, we recycle the notation $\hat{\boldsymbol{u}}_i$ for different methods in the remainder. In general, whenever a method makes use of $\hat{\boldsymbol{u}}_i$, this notation indicates the gPC coefficients of some approximation to the exact solution $\boldsymbol{u}$.

In terms of the number of moments needed to achieve a total degree of N, given by (1.83), this reads

$$\|\mathcal{U}(\hat{\boldsymbol{u}}) - u\|_{L^2_\Xi([-1,1]^p)} \leq \tilde{C}\,(M \cdot p!)^{-q/p}\,\|u\|_{H^{q,\infty}_\Xi([-1,1]^p)}.$$

Proof. In the following, we assume that the Sobolev index q is an even number. The proof will be shown for Legendre polynomials which are eigenfunctions of the operator

$$Lu := \frac{d}{d\xi}\left[(1-\xi)^2\frac{d}{d\xi}\right]u, \tag{1.87}$$

meaning that

$$L\varphi_i = \lambda_i\varphi_i \tag{1.88}$$

with $\lambda_i := -i(i+1)$. Let us begin with a one-dimensional uncertainty, i.e. $\xi \in [-1,1]$. The operator L is self-adjoint, since

$$\begin{aligned}
\langle (Lu)v\rangle &= \int_{-1}^{1}(Lu)(\xi)v(\xi)f_\Xi(\xi)\,d\xi \\
&= \int_{-1}^{1}\frac{d}{d\xi}\left[(1-\xi)^2\frac{d}{d\xi}\right]u(\xi)v(\xi)f_\Xi(\xi)\,d\xi \\
&= -\int_{-1}^{1}(1-\xi^2)\frac{d}{d\xi}u(\xi)\frac{d}{d\xi}v(\xi)f_\Xi(\xi)\,d\xi + (1-\xi^2)\frac{d}{d\xi}u(\xi)v(\xi)f_\Xi(\xi)|_{-1}^{1} \\
&= -\int_{-1}^{1}u(\xi)\frac{d}{d\xi}\left[(1-\xi^2)\frac{d}{d\xi}v(x)\right]f_\Xi(\xi)d\xi = \langle u(Lv)\rangle.
\end{aligned} \tag{1.89}$$

Therefore, the relation

$$\langle u\varphi_i\rangle \overset{(1.88)}{=} \frac{1}{\lambda_i}\langle uL\varphi_i\rangle \overset{(1.89)}{=} \frac{1}{\lambda_i}\langle Lu\varphi_i\rangle \tag{1.90}$$

holds, which when being applied k times yields

$$\langle u\varphi_i\rangle = \frac{1}{\lambda_i^k}\langle L^k u\varphi_i\rangle.$$

Extending this result to p dimensions means that i becomes a multi-index $i = (i_1, \cdots, i_p)$. In this case, we use the self-adjoint operator

$$L := \prod_{\ell=1}^{p}\frac{d}{d\xi_\ell}\left[(1-\xi_\ell)^2\frac{d}{d\xi_\ell}\right]$$

for which the eigenvalues become

$$\lambda_i := \prod_{\ell=1}^{p}(-i_\ell)\cdot(i_\ell+1).$$

Note that for $|i| \geq N+1$ we have that

$$\frac{1}{\lambda_i^2} \leq \frac{1}{\lambda^2_{(N+1,0,\cdots,0)}} < \frac{1}{N^4}. \tag{1.91}$$

The theorem can now be proven with

$$\left\langle \left(\mathcal{U}(\hat{\boldsymbol{u}}) - u\right)^2 \right\rangle = \left\langle \left(\sum_{|i|=0}^{\infty} \hat{u}_i \varphi_i - \sum_{|i|\leq N} \hat{u}_i \varphi_i \right)^2 \right\rangle = \left\langle \left(\sum_{|i|\geq N+1} \hat{u}_i \varphi_i \right)^2 \right\rangle.$$

Using Parseval's identity (PI) as well as orthonormality of the basis functions yields

$$\begin{aligned}\left\langle \left(\sum_{|i|\geq N+1} \hat{u}_i \varphi_i \right)^2 \right\rangle &\overset{\text{PI}}{=} \sum_{|i|\geq N+1} \hat{u}_i^2 = \sum_{|i|\geq N+1} \langle u\varphi_i \rangle^2 = \sum_{|i|\geq N+1} \lambda_i^{-2m} \langle L^m u \varphi_i \rangle^2 \\ &\overset{(1.91)}{\leq} N^{-4m} \sum_{|i|\geq N+1} \langle L^m u \varphi_i \rangle^2 \leq N^{-4m} \sum_{|i|=0}^{\infty} \langle L^m u \varphi_i \rangle^2 \overset{\text{PI}}{=} N^{-4m} \langle L^m u \rangle^2.\end{aligned}$$

With $q := 2m$, we get

$$\left\langle \left(\mathcal{U}(\hat{\boldsymbol{u}}) - u\right)^2 \right\rangle \leq cN^{-2q} \|L^m u\|_{L^2_\Xi}^2 \leq \left(CN^{-q} \|u\|_{H^{q,\infty}_\Xi} \right)^2.$$

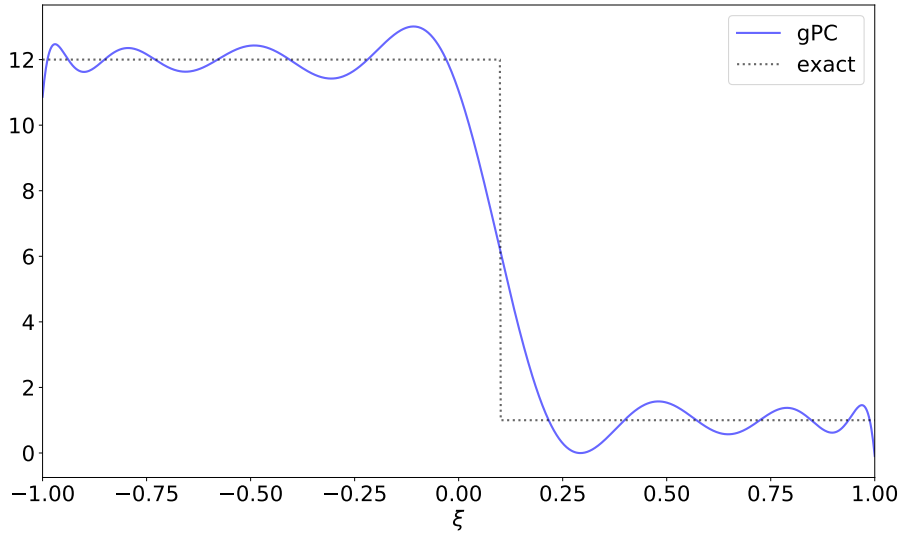
□

Example 2. *In the following example, we investigate the gPC expansion of a given function u for a scalar random variable $\xi \sim U([-1,1])$, i.e. ξ is uniformly distributed in the interval $\theta = [-1,1]$. The function u is a shock in ξ, i.e.*

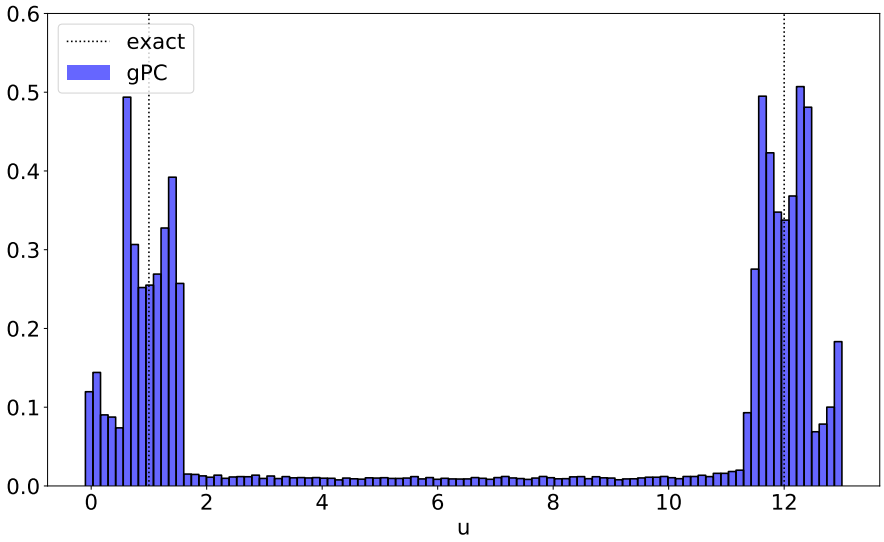
$$u(\xi) = \begin{cases} 12 & \text{if } \xi > 0.1 \\ 1 & \text{else} \end{cases}. \tag{1.92}$$

The exact solution as well as the gPC reconstruction (1.84) with $N = 16$ can be found in Figure 1.4A. Typical characteristics of the gPC expansion at shocks stick out and point to potential shortcomings of methods based on gPC: First, the gPC reconstruction oscillates, thereby violating bounds of the exact solution. Especially in the context of hyperbolic conservation laws, which fulfill the maximum-principle (1.39), this can potentially destroy important solution properties. Second, the gPC reconstruction shows a large distance to the exact solution. When taking a look at the corresponding histogram in Figure 1.4B, one sees that the gPC approximation smears out the exact distribution, leading to a large number of function values which do not appear in the exact solution.

For now, we studied approximation techniques for a given function which only depends on a vector of random variables $\boldsymbol{\xi}$. However, recall that we are interested in determining the moments that correspond to the solution of (1.78). Therefore, one needs to include time and space into the solution's phase space and propose methods to propagate the uncertainty through the given equations. Approximating the moments will again add an error to our final solution. Numerical methods for determining the moments $\hat{\boldsymbol{u}}$ can be divided into intrusive and non-intrusive techniques. Broadly speaking, non-intrusive methods use a given code framework for the deterministic problem as a black-box whereas intrusive methods modify such a code framework or even require writing an entirely new code.



(A)



(B)

FIGURE 1.4: (A) Exact solution and gPC reconstruction with $N = 16$. The gPC coefficients of (1.92) have been computed with a Gauss quadrature using 500 nodes. A comparison of the resulting L^2 error for different truncation orders can be found in Figure 1.5. (B) Histogram of the gPC approximation.

1.4.3 Monte-Carlo methods

A popular non-intrusive approach are *Monte-Carlo* (MC) methods [126, Chapter 9.5], which randomly sample input unceratinties to compute quantities of interest. Note that computing quantities of interest such as expectation value or moments boils down to computing an integral over the uncertain domain Θ. Given a sequence of independently identically distributed random variables $\boldsymbol{\xi}^{(k)}$, the Monte-Carlo method

approximates a given integral by

$$Qg := \int_\Theta g(\boldsymbol{\xi}) f_\Xi(\boldsymbol{\xi})\, d\boldsymbol{\xi} \approx Q_{MC}g := \frac{1}{N_{MC}} \sum_{k=1}^{N_{MC}} g(\boldsymbol{\xi}^{(k)}),$$

i.e. MC computes N_{MC} samples $\boldsymbol{\xi}^{(k)}$, evaluates the integrand $g : \Theta \to \mathbb{R}$ at these samples and then averages over the resulting function values. When the integrand is the solution to a hyperbolic conservation law at a fixed time t (i.e. we compute the expectation value of $\boldsymbol{u}$), this procedure translates into running a given deterministic solver at N_{MC} samples to compute $\boldsymbol{u}(t, \boldsymbol{x}, \boldsymbol{\xi}^{(k)})$ and averaging over these results. Further quantities of interest can be computed in the same manner. The error of the MC method is given by

$$|Qg - Q_{MC}g| = \sqrt{\frac{\mathrm{Var}(g)}{N_{MC}}}.$$

Note that opposed to the convergence rate of the gPC expansion, given in Theorem 4, the convergence of Monte-Carlo does not directly depend on the smoothness of the solution, but is fixed at order $1/2$, which appears to be quite slow. However, the convergence rate does not depend on the dimension of the random space p. Thus, going to higher-dimensional random spaces does not affect the convergence speed (if the variance remains constant). Variants of Monte-Carlo, which aim at accelerating MC are Quasi-Monte-Carlo methods [101], which use quasi-random sequences that generate more evenly spaced samples as well as Multilevel-Monte-Carlo methods, which sample on differently refined spatial meshes (different levels) to reduce the corresponding variance [54, 98, 97, 99].

1.4.4 Collocation methods

A popular non-intrusive method is the *Stochastic Collocation* (SC) method, see e.g. [146, 7, 84]. When looking at the literature, the name Stochastic Collocation has been used in different contexts. The idea of collocation methods in general is to choose a set of Q points $\boldsymbol{\xi}_k$ on which the forward model J is then evaluated, i.e. we have $J_k := J(\xi_k)$. Following [126], collocation methods are equipped with a general interpolation procedure, meaning that from the solution of the forward model at the chosen point set one can retrieve a continuous solution approximation $\tilde{J}(\boldsymbol{\xi})$. At the point set, the approximation fulfills $\tilde{J}(\boldsymbol{\xi}_k) = J_k$ and in [126] the name Stochastic Collocation denotes any collocation method applied to the random domain. In [33], the name collocation refers to methods which specifically use interpolating Lagrange polynomials to represent the solution. For a scalar random variable, these functions read

$$L_j(\xi) := \prod_{k=1, k\neq j} \frac{\xi - \xi_k}{\xi_j - \xi_k}.$$

Then, the forward model can be interpolated at different values for ξ by

$$\tilde{J}(\xi) = \sum_{k=1}^{Q} J(\xi_k) L_j(\xi). \tag{1.93}$$

When using roots of the $(Q+1)_{\text{st}}$ orthonormal polynomial as collocation points, this collocation strategy is called Stochastic Collocation in [33]. Note that in this case, computing the integral of (1.93) yields a Gauss quadrature rule [23, Chapter 10.3].

Recall, that our main goal was to approximate the PC expansion coefficients (1.85), which will be done by a Gauss-Quadrature rule and we denote this method as Stochastic Collocation . For a given set of Q quadrature weights w_k and quadrature points $\boldsymbol{\xi}_k$, the moments are approximated by

$$\hat{\boldsymbol{u}}_i = \langle \boldsymbol{u}\varphi_i \rangle \approx \langle \boldsymbol{u}\varphi_i \rangle_Q := \sum_{k=1}^{Q} w_k \boldsymbol{u}(t, \boldsymbol{x}, \boldsymbol{\xi}_k)\varphi_i(\boldsymbol{\xi}_k) f_\Xi(\boldsymbol{\xi}_k). \tag{1.94}$$

Since the solution at a fixed quadrature point (i.e. $\boldsymbol{u}(t, \boldsymbol{x}, \boldsymbol{\xi}_k)$) can be computed by a standard deterministic solver, the SC method does not require a significant implementation effort. Furthermore, SC is embarrassingly parallel, since the required computations can be carried out in parallel on different cores. A downside of collocation methods are aliasing effects, which stem from the inexact approximation of integrals. Consequently, the final solution approximation does not only show the error from the gPC approximation given by Theorem 4, but will also depend on the quadrature error. Let us denote the gPC expansion (1.84) of total degree N by

$$P_N \boldsymbol{u} := \sum_{|i| \leq N} \langle \boldsymbol{u}\varphi_i \rangle \varphi_i(\boldsymbol{\xi}), \tag{1.95}$$

and the corresponding collocation approximation by

$$I_N \boldsymbol{u} := \sum_{|i| \leq N} \langle \boldsymbol{u}\varphi_i \rangle_Q \varphi_i(\boldsymbol{\xi}). \tag{1.96}$$

With the triangle inequality with $\|\cdot\|$ being a norm in ξ, we have that

$$\|\boldsymbol{u} - I_N \boldsymbol{u}\| \leq \|\boldsymbol{u} - P_N \boldsymbol{u}\| + \|P_N \boldsymbol{u} - I_N \boldsymbol{u}\|.$$

Hence, the inexact computation of the moments adds a quadrature (or *aliasing*) error to the overall distance to the exact solution. When denoting the moments of $P_N \boldsymbol{u}$ by $\hat{\boldsymbol{u}}$ and the moments of $I_N \boldsymbol{u}$ by $\hat{\boldsymbol{w}}$, the aliasing error can be expressed as

$$\|P_N \boldsymbol{u} - I_N \boldsymbol{u}\| = \left(\sum_{|i| \leq N} (\hat{u}_i - \hat{w}_i)^2 \right)^{1/2}. \tag{1.97}$$

Note that not only the projection error $\|\boldsymbol{u} - P_N \boldsymbol{u}\|$ converges with $O(N^{-\alpha})$ (as shown in Theorem 4, where α is the smoothness of $\boldsymbol{u}$). Also the aliasing error (1.97) will show this convergence rate, see [144]. To visualize the effects of the aliasing error, we again look at the setting from Example 2, where we chose a sufficiently fine number of quadrature points $N_q = 500$ to compute expansion coefficients. Now we modify the truncation order N and check the L^2 error of the resulting approximation (1.96) when using $N_q = 500$ as well as $N_q = N + 1$ quadrature points. The choice $N_q = 500$ will yield a small aliasing error and we hope to recover the error from the pPC approximation (1.95), which is why we denote this error as *projection error*. The error when using $N_q = N + 1$ collocation points will be denoted as *collocation error*. The results for this investigation can be found in Figure 1.5. Since shocks lie in the fractional Sobolev space $H_\Xi^{1/2}$, we expect to see a convergence rate of $O(N^{-1/2})$. Indeed, both approximations show this convergence rate, however the error of the

collocation solution is affected by the aliasing error (1.97), which does not destroy the overall convergence rate, but yields oscillations, which can lead to a significant increase of the error.

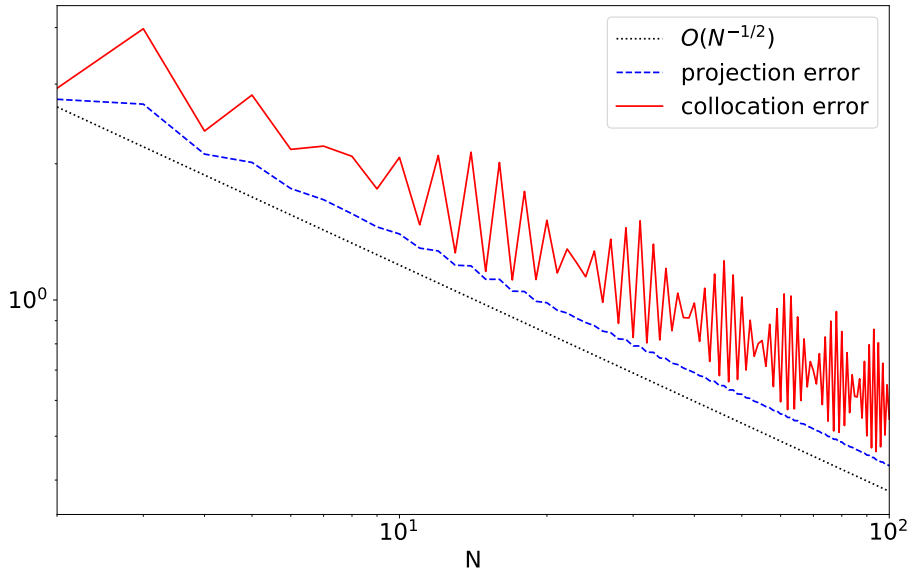


FIGURE 1.5: L^2 error of approximations to the shock (1.92) from Example 2 for different truncation orders 10. The chosen approximation strategies are the gPC expansion (1.95), denoted as *projection error*, as well as collocation according to (1.96) with $N_q = N + 1$ quadrature points, denoted as *collocation error*.

Tensorized Grids

A crucial step which determines the effectiveness of the collocation method is the construction of a quadrature rule for the potentially high-dimensional domain $\Theta \subset \mathbb{R}^p$. Our goal here will be to construct a quadrature rule which yields a high accuracy for a low number of quadrature points Q. Assuming that a quadrature rule is given on the one-dimensional interval $[-1, 1]$ by

$$\int_{-1}^{1} g(\xi) f_\Xi(\xi) d\xi \approx Q^{(1)} g := \sum_{k=1}^{N_q} g(\tilde{\xi}_k) \tilde{w}_k,$$

a p-dimensional quadrature rule can be constructed with tensorization: Using a multi-index $k = (k_1, \cdots, k_p)$ as well as $|k|_\infty := \max_{\ell=1\cdots p} k_\ell$, we get

$$\int_{-1}^{1} \cdots \int_{-1}^{1} g(\boldsymbol{\xi}) f_\Xi(\boldsymbol{\xi}) d\xi_1 \cdots d\xi_p \approx \sum_{|k|_\infty \leq N_q} w_k g(\tilde{\xi}_{k_1}, \cdots, \tilde{\xi}_{k_p}), \tag{1.98}$$

with weights $w_k := \tilde{w}_{k_1} \cdot \ldots \cdot \tilde{w}_{k_p}$. When the one-dimensional quadrature rule yields an error of $O(N_q^{-\alpha})$, its tensorized version has the same error while requiring $Q = N_q^p$ quadrature points. Hence the error behaves like $O(Q^{-\alpha/p})$, i.e. the number of dimensions slows down the convergence rate. This is a key issue in the field of uncertainty quantification and when dealing with high-dimensional phase spaces in

general and is often referred to as the *curse of dimensionality*. Similarly, the gPC expansion suffers from the curse as well: As discussed previously, the error of the gPC expansion behaves like $O(N^{-\alpha})$, and the total number of moments M (when using polynomials up to a total degree of N) was given by (1.83). Note that the increase of M when going to higher dimension p is significantly slower than the increase of the total number of quadrature points Q. Therefore, the curse of dimensionality affects tensorized quadrature rules more heavily. Sparse grids aim at mitigating the effect of the dimension on the error behavior for quadrature rules.

Sparse Grids

In the following, we provide a brief introduction of sparse grids. More information can for example be found in [122, Chapter 11.1.3]. Commonly, the accuracy of quadrature formulas is determined by the polynomial degree which can be integrated exactly. While tensorized grids of the form (1.98) integrate polynomials of maximal degree N exactly, sparse grids aim at guaranteeing the exact integration of total degree N, while in turn requiring less quadrature points. For this, a nested quadrature (e.g. Clenshaw Curtis (CC) [19]) is used. Nested means that a quadrature rule at level ℓ recycles all quadrature points from level $\ell - 1$. We define a one-dimensional quadrature rule at level ℓ for a function g by

$$Q_\ell^{(1)} g := \sum_{k=1}^{R_\ell} g(\xi_\ell^k) w_\ell^k,$$

where R_ℓ is the number of quadrature points at level ℓ and ξ_ℓ^k, w_ℓ^k denote the quadrature points and weights at level ℓ. A difference relation at level ℓ can be defined by

$$\Delta_\ell^{(1)} g := \left(Q_\ell^{(1)} - Q_{\ell-1}^{(1)} \right) g,$$

with $Q_0^{(1)} \equiv 0$. The operator $\Delta_\ell^{(1)}$ essentially acts like a quadrature rule, where the quadrature points equal the nodes of $Q_\ell^{(1)}$, however the weights are given by the difference of the corresponding weights at levels ℓ and $\ell - 1$. Going to an p-dimensional domain of integration, the sparse grid quadrature at level ℓ can be defined by

$$Q_\ell^{(p)} g = \sum_{|\ell'| \leq \ell + p - 1} \left(\Delta_{\ell_1}^{(1)} \otimes \cdots \otimes \Delta_{\ell_p}^{(1)} \right) g, \tag{1.99}$$

where again, $|\ell'|$ denotes the total degree of the multi-index $\ell' \in \mathbb{N}^p$. Note that the tensor product formulation can be brought into similar form

$$Q_\ell^{(p)} g = \sum_{|\ell'|_\infty \leq \ell} \left(\Delta_{\ell_1}^{(1)} \otimes \cdots \otimes \Delta_{\ell_p}^{(1)} \right) g. \tag{1.100}$$

Hence, the only difference between sparse grids and tensorized grids is that the outer sum in (1.99) and (1.100) runs over two different index sets

$$\mathbb{I}_{\text{sparse}} := \left\{ \ell' \in \mathbb{N}^p \,\middle|\, \sum_{i=1}^{p} \ell_i \leq \ell + p - 1 \right\},$$
$$\mathbb{I}_{\text{tensorized}} := \left\{ \ell' \in \mathbb{N}^p \,|\, \ell_i \leq \ell, \text{ for } i = 1, \cdots, p \right\}.$$

Note, that numerically computing integrals using sparse grids will not follow (1.99), i.e. the sum of tensorized quadrature rules is not computed. Instead, one determines the quadrature points at the highest level as well as the corresponding quadrature weights. Compared to the tensorized index set, which has $O(N^p)$ quadrature points, sparse grids use $O(N(\log_2(N)^{p-1}))$ quadrature points to integrate polynomials of total degree N exactly, see e.g. [136]. Therefore, the curse of dimensionality is mitigated. Note that the number of points in a sparse grid to integrate polynomials of total degree N exactly is still bigger than the number of moments in a gPC expansion with the same total degree. Therefore, an intriguing idea is to derive a method which uses the moments as unknowns and does not rely on quadrature rules. Methods presented in the next section use the moments as unknowns, but unfortunately, we will see that these methods generally still rely on quadrature rules.

1.4.5 Intrusive methods

Intrusive methods are in general more difficult to implement and come along with higher numerical costs. The main idea of these methods is to directly derive a system of equations for the moments and then implementing a numerical solver for this system.

The stochastic-Galerkin method

A commonly used intrusive method follows the idea of a standard Galerkin method, applied to the stochastic dimension. Plugging the polynomial solution ansatz (1.84) into the original hyperbolic problem (1.78) yields

$$\partial_t \sum_{|i|\leq N} \hat{\boldsymbol{u}}_i(t,\boldsymbol{x})\varphi_i(\boldsymbol{\xi}) + \nabla \cdot \boldsymbol{f}\left(\sum_{|i|\leq N} \hat{\boldsymbol{u}}_i(t,\boldsymbol{x})\varphi_i(\boldsymbol{\xi})\right) = \boldsymbol{r}(t,\boldsymbol{x},\boldsymbol{\xi}), \tag{1.101}$$

where $\boldsymbol{r}(t,\boldsymbol{x},\boldsymbol{\xi})$ is the residual resulting from the inaccurate solution ansatz. The expansion coefficients are then chosen such that this residual is orthogonal to our finite-dimensional solution space. Hence, when multiplying the system (1.101) with φ_i where again $|i| \leq N$ and performing a weighted integration over the random domain according to (1.79), we obtain the so-called *SG moment system*

$$\partial_t \hat{\boldsymbol{u}}_i(t,\boldsymbol{x}) + \nabla \cdot \langle \boldsymbol{f}(\mathcal{U}(\hat{\boldsymbol{u}}(t,\boldsymbol{x})))\varphi_i \rangle = \boldsymbol{0} \tag{1.102}$$

with $|i| \leq N$. I.e. projecting the residual $\boldsymbol{r}$ to zero yields a closed system, which consists of M time evolution equations for the first M moments. Commonly, this system is supplemented with adequate boundary- and initial conditions, which we will specify for the individual problems whenever necessary. This technique, which is called the stochastic-Galerkin (SG) [43] method is intrusive, in that it requires the implementation of new code. If all integrals showing up in (1.102) can be computed analytically (and if the moment system is stable), we can essentially replace the quadrature error from collocation methods by the truncation error of our ansatz, which now affects the time evolution of the moment vector. For scalar hyperbolic equations, this is reflected by the following theorem presented in [27]:

Theorem 5. *Assume that the solution to a random hyperbolic differential equation of the form* (1.78) *with* $p = 1$ *is given by* $u(t,x,\xi)$ *and the corresponding SG approximation of*

(1.102) *is given by* $\mathcal{U}(\hat{\boldsymbol{u}}(t,\boldsymbol{x});\xi)$. *Furthermore, let*

$$u \in L^\infty(\mathbb{R}^+ \times D \times \Theta) \cap L^\infty(\mathbb{R}^+ \times D; H_\Xi^{q,\infty}(\Theta)).$$

Then, the stochastic-Galerkin error is given by

$$\|u(t,\cdot,\cdot) - \mathcal{U}(\hat{\boldsymbol{u}}(t,\cdot);\cdot)\|_{L^2(D\times\Theta)} \leq C\left(\|u_{IC} - \mathcal{U}(\hat{\boldsymbol{u}}(0,\cdot);\cdot)\|_{L^2(D\times\Theta)} + \frac{1}{N^q}\right).$$

A similar result for advection equations with random advection speed has been shown in [46] and in [62] for kinetic equations. However, showing stability for general hyperbolic systems is a difficult task and so far not always possible. As discussed previously, the solutions to hyperbolic problems are generally nonsmooth, and thus the SG method converges slowly and exhibits the oscillations of Gibbs phenomenon. Furthermore, the moment system of SG is not necessarily hyperbolic [112]. A clearer image of the effects this has in applications can be seen when for example quantifying uncertainties for gas dynamics equations with SG: The finite-dimensional projection of density or energy can lead to oscillatory undershoots, which can yield non-physical, negative values, see Figure 1.6.

The Intrusive Polynomial Moment method

A generalization of stochastic-Galerkin, which ensures hyperbolicity is the Intrusive Polynomial Moment (IPM) method [112]. Instead of expanding the conserved variables $\boldsymbol{u}$ with polynomials, the IPM method performs such an expansion on the entropy variables, which have been introduced in (1.14). Hence, substituting the entropy variables $\boldsymbol{v} = \nabla_{\boldsymbol{u}} s(\boldsymbol{u})^T$ into (1.78) yields

$$\partial_t \boldsymbol{u}(\boldsymbol{v}(t,\boldsymbol{x},\boldsymbol{\xi})) + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}(\boldsymbol{v}(t,\boldsymbol{x},\boldsymbol{\xi}))) = \boldsymbol{0}, \tag{1.103}$$

where again $\boldsymbol{u} : \mathbb{R}^m \to \mathbb{R}^m$ with $\boldsymbol{u}(\boldsymbol{v}) = (\nabla_{\boldsymbol{u}} s)^{-1}(\boldsymbol{u})$. Now, a finite-dimensional representation of the entropy variables is obtained by an expansion in terms of gPC polynomials, i.e.

$$\boldsymbol{v}(t,\boldsymbol{x},\boldsymbol{\xi}) \approx \boldsymbol{v}_N(t,\boldsymbol{x},\boldsymbol{\xi}) := \sum_{|i|\leq N} \hat{\boldsymbol{v}}_i(t,\boldsymbol{x})\varphi_i(\boldsymbol{\xi}) = \hat{\boldsymbol{v}}(t,\boldsymbol{x})^T \boldsymbol{\varphi}(\boldsymbol{\xi}) \tag{1.104}$$

where the *entropic expansion coefficients* (also called *dual variables*) $\hat{\boldsymbol{v}}_i \in \mathbb{R}^m$ are collected in the matrix $\hat{\boldsymbol{v}} := (\hat{\boldsymbol{v}}_i)_{|i|\leq N} \in \mathbb{R}^{M\times m}$. Replacing the exact entropy variables inside the original problem (1.103) by this expansion, we obtain

$$\partial_t \boldsymbol{u}\left(\hat{\boldsymbol{v}}(t,\boldsymbol{x})^T\boldsymbol{\varphi}(\boldsymbol{\xi})\right) + \nabla \cdot \boldsymbol{f}\left(\boldsymbol{u}\left(\hat{\boldsymbol{v}}(t,\boldsymbol{x})^T\boldsymbol{\varphi}(\boldsymbol{\xi})\right)\right) = \tilde{\boldsymbol{r}}(t,\boldsymbol{x},\boldsymbol{\xi}). \tag{1.105}$$

As done for (1.101), the residual $\tilde{\boldsymbol{r}}$ is again projected to zero, yielding

$$\partial_t \left\langle \boldsymbol{u}\left(\hat{\boldsymbol{v}}(t,\boldsymbol{x})^T\boldsymbol{\varphi}\right)\varphi_i \right\rangle + \nabla \cdot \left\langle \boldsymbol{f}\left(\boldsymbol{u}\left(\hat{\boldsymbol{v}}(t,\boldsymbol{x})^T\boldsymbol{\varphi}\right)\right)\varphi_i \right\rangle = \boldsymbol{0} \tag{1.106}$$

for $|i| \leq N$. The moments belonging to the dual variables $\hat{\boldsymbol{v}}$ are now given by

$$\hat{\boldsymbol{u}}_i(\hat{\boldsymbol{v}}) = \left\langle \boldsymbol{u}\left(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi}\right)\varphi_i \right\rangle \qquad \text{for } |i| \leq N. \tag{1.107}$$

This mapping, i.e $\hat{\boldsymbol{u}} : \mathbb{R}^{M\times m} \to \mathcal{R} \subset \mathbb{R}^{M\times m}$ is one-to-one, meaning that similar to $\boldsymbol{v}(\boldsymbol{u})$, we can define a function $\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}})$ with $\hat{\boldsymbol{v}} : \mathcal{R} \to \mathbb{R}^{M\times m}$. Making use of this mapping as well as the definition of the moments in (1.106) yields the IPM system

$$\partial_t \hat{\boldsymbol{u}}_i + \nabla \cdot \left\langle \boldsymbol{f}\left(\boldsymbol{u}\left(\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}})^T\boldsymbol{\varphi}\right)\right)\varphi_i \right\rangle = \boldsymbol{0}. \tag{1.108}$$

Let us first note some remarkable features of IPM:

Theorem 6. *If the entropy $s(\boldsymbol{u})$ fulfills the integrability condition (1.8), the IPM system when applied to the viscous equations (1.10) dissipates the expectation value of the original entropy. I.e. when $\hat{\boldsymbol{u}}(t,\boldsymbol{x})$ is the solution to the IPM system for the viscous problem (1.10) and $\hat{\boldsymbol{v}}(t,\boldsymbol{x})$ are the corresponding dual variables, we have*

$$\frac{d}{dt}\left\langle \int_D s\left(\boldsymbol{u}\left(\hat{\boldsymbol{v}}(t,\cdot)^T\boldsymbol{\varphi}\right)\right)\, d\boldsymbol{x}\right\rangle \leq 0. \tag{1.109}$$

Proof. Performing the Galerkin projection for the viscous problem yields

$$\partial_t \left\langle \boldsymbol{u}\left(\boldsymbol{v}_N^{(\varepsilon)}\right)\varphi_i\right\rangle + \sum_{j=1}^{N_x}\partial_{x_j}\left\langle \boldsymbol{f}_j\left(\boldsymbol{u}\left(\boldsymbol{v}_N^{(\varepsilon)}\right)\right)\varphi_i\right\rangle = \varepsilon\left\langle \Delta\boldsymbol{u}\left(\boldsymbol{v}_N^{(\varepsilon)}\right)\varphi_i\right\rangle, \tag{1.110}$$

where $\boldsymbol{v}_N^{(\varepsilon)}$ is the gPC expansion for the viscous problem. In the following, we omit the superscript ε for ease of presentation. To derive a time evolution equation for the entropy, we multiply (1.110) from the left by $\hat{\boldsymbol{v}}_i^T$ and sum over i with $|i| \leq N$. Then the first term of (1.110) becomes

$$\begin{aligned}\left\langle \left(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi}\right)^T \partial_t\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right\rangle &= \left\langle \nabla_{\boldsymbol{u}} s\left(\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right)\partial_t\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right\rangle \\ &= \partial_t\left\langle s\left(\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right)\right\rangle.\end{aligned}$$

For the flux term, we get

$$\begin{aligned}\left\langle \left(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi}\right)^T\sum_{j=1}^{d}\partial_{x_j}\boldsymbol{f}_j\left(\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right)\right\rangle &= \left\langle \nabla_{\boldsymbol{u}} s\left(\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right)\sum_{j=1}^{d}\nabla_{\boldsymbol{u}}\boldsymbol{f}_j\left(\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right)\partial_{x_j}\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right\rangle \\ &\overset{(1.8)}{=} \left\langle \sum_{j=1}^{d}\nabla_{\boldsymbol{u}}\tilde{F}_j(\boldsymbol{u}(\boldsymbol{v}_N))\partial_{x_j}\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right\rangle \\ &= \sum_{j=1}^{d}\partial_{x_j}\left\langle \tilde{F}_j(\boldsymbol{u}(\boldsymbol{v}_N))\right\rangle.\end{aligned}$$

Again, using the inverted product rule for the right hand side gives

$$\begin{aligned}\left\langle \left(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi}\right)^T\Delta\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right\rangle &= \left\langle \nabla_{\boldsymbol{u}} s(\boldsymbol{u}\left(\boldsymbol{v}_N\right))\Delta\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right\rangle \\ &= \left\langle \nabla\cdot\left(\left(\nabla\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right)^T\nabla_{\boldsymbol{u}} s(\boldsymbol{u}\left(\boldsymbol{v}_N\right))\right)\right\rangle \\ &\quad - \left\langle \left(\nabla\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right)^T\nabla_{\boldsymbol{u}\boldsymbol{u}} s(\boldsymbol{u}\left(\boldsymbol{v}_N\right))\nabla\boldsymbol{u}\left(\boldsymbol{v}_N\right)\right\rangle.\end{aligned}$$

Here, the first term of the right hand side can be pulled into the entropy flux and the remaining term on the right is positive due to convexity of the entropy $s(\boldsymbol{u})$.

Altogether, this yields

$$\partial_t \langle s(\boldsymbol{u}(\boldsymbol{v}_N))\rangle + \left\langle \nabla\cdot\left[\tilde{\boldsymbol{F}}(\boldsymbol{u}(\boldsymbol{v}_N)) + \varepsilon(\nabla\boldsymbol{u}(\boldsymbol{v}_N))^T\nabla_{\boldsymbol{u}}s(\boldsymbol{u}(\boldsymbol{v}_N))\right]\right\rangle = -\varepsilon\left\langle (\nabla\boldsymbol{u}(\boldsymbol{v}_N))^T\nabla_{\boldsymbol{u}\boldsymbol{u}}s(\boldsymbol{u}(\boldsymbol{v}_N))\nabla\boldsymbol{u}(\boldsymbol{v}_N)\right\rangle.$$

Integrating over the spatial and random domain when assuming that no external spatial fluxes enter the system yields (1.109). □

Thus, the IPM system of the viscous problem dissipates a specific entropy, which depends on the choice of the entropy variables and thereby on the choice of the deterministic entropy $s(\boldsymbol{u})$. In Chapter 2 we will study the effects of this choice in greater detail. For now, note that the IPM system of a scalar problem will not dissipate the expectation value of all admissible scalar entropies (namely all convex functions). Only the expectation value of the one chosen entropy will be dissipated in time by the viscous problem. Moreover, the IPM system is hyperbolic:

Theorem 7. *The IPM system can be brought into its symmetric form with symmetric positive definite temporal Jacobian and symmetric spatial Jacobian, if the entropy $s(\boldsymbol{u})$ fulfills the integrability condition (1.8).*

Proof. We can write the IPM system (1.106) in its symmetrized form

$$\hat{\boldsymbol{H}}(\hat{\boldsymbol{v}})\partial_t\hat{\boldsymbol{v}} + \sum_{j=1}^{d}\hat{\boldsymbol{B}}_j(\hat{\boldsymbol{v}})\partial_{x_j}\hat{\boldsymbol{v}} = \boldsymbol{0}, \tag{1.111a}$$

$$\text{with}\quad \hat{\boldsymbol{H}}(\hat{\boldsymbol{v}}) := \left\langle \nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v}_N)\otimes\boldsymbol{\varphi}\boldsymbol{\varphi}^T\right\rangle, \tag{1.111b}$$

$$\hat{\boldsymbol{B}}_j(\hat{\boldsymbol{v}}) := \left\langle \nabla_{\boldsymbol{u}}\boldsymbol{f}_j(\boldsymbol{u}(\boldsymbol{v}_N))\nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v}_N)\otimes\boldsymbol{\varphi}\boldsymbol{\varphi}^T\right\rangle. \tag{1.111c}$$

Here, we abuse notation by defining the multiplication of $\hat{\boldsymbol{H}}\in\mathbb{R}^{m\cdot M\times m\cdot M}$ with $\boldsymbol{y}\in\mathbb{R}^{M\times m}$ by

$$\left(\hat{\boldsymbol{H}}\cdot\boldsymbol{y}\right)_{li} := \sum_{l'=1}^{m}\sum_{i'=1}^{M}\hat{H}_{(l-1)m+i,(l'-1)m+i'}y_{l'i'}.$$

The same holds for the multiplication with $\hat{\boldsymbol{B}}_j$. As done for (1.17), if we can ensure $\hat{\boldsymbol{H}}$ being symmetric positive definite and $\hat{\boldsymbol{B}}_j$ symmetric, we know that the IPM system is hyperbolic. Obviously, $\hat{\boldsymbol{H}}$ is symmetric. Multiplication with $\hat{\boldsymbol{v}}\in\mathbb{R}^{M\times m}$ from both sides gives

$$\hat{\boldsymbol{v}}^T\hat{\boldsymbol{H}}\hat{\boldsymbol{v}} = \left\langle \boldsymbol{v}_N^T\nabla_{\boldsymbol{v}}\boldsymbol{u}(\boldsymbol{v}_N)\boldsymbol{v}_N\right\rangle \geq 0,$$

where we use that $\nabla_{\boldsymbol{v}}\boldsymbol{u} = \boldsymbol{H}$ is symmetric positive definite as done in (1.17). It remains to show symmetry of $\hat{\boldsymbol{B}}_j$ for all $j = 1,\cdots,d$. Using the definition of $\boldsymbol{B}_j$ from (1.18), we can rewrite (1.111c) as

$$\hat{\boldsymbol{B}}_j(\hat{\boldsymbol{v}}) := \left\langle \boldsymbol{B}_j(\boldsymbol{v}_N)\otimes\boldsymbol{\varphi}\boldsymbol{\varphi}^T\right\rangle.$$

By Theorem 1, we know that $\boldsymbol{B}_j$ is symmetric, from which we can conclude symmetry of $\hat{\boldsymbol{B}}_j$. □

Recall that solving the IPM system requires the mapping $\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}})$, i.e. a mapping from the moments to the dual variables. This mapping can be defined by inverting the dual variables to moments map (1.107). The inverse exists, since the Jacobian of $\hat{\boldsymbol{u}}(\hat{\boldsymbol{v}})$ is $\nabla_{\hat{\boldsymbol{v}}}\hat{\boldsymbol{u}}(\hat{\boldsymbol{v}}) = \hat{\boldsymbol{H}}(\hat{\boldsymbol{v}})$ which is positive definite, i.e. the dual variables to moments map is strictly monotonically increasing. Unfortunately, the inversion can generally not be performed analytically. In this case one needs to determine $\hat{\boldsymbol{v}}$ by solving the non-linear system of equations

$$\left\langle \boldsymbol{u}\left(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi}\right)\boldsymbol{\varphi}^T\right\rangle^T = \hat{\boldsymbol{u}} \tag{1.112}$$

for a given moment vector $\hat{\boldsymbol{u}}$ numerically. This task is commonly performed by reformulating (1.112) as a root-finding problem

$$\mathcal{G}(\hat{\boldsymbol{v}};\hat{\boldsymbol{u}}) \stackrel{!}{=} 0$$

with

$$\mathcal{G}(\boldsymbol{w};\hat{\boldsymbol{u}}) := \left\langle \boldsymbol{u}\left(\boldsymbol{w}^T\boldsymbol{\varphi}\right)\boldsymbol{\varphi}^T\right\rangle^T - \hat{\boldsymbol{u}} \tag{1.113}$$

and using Newton's method to determine the root of $\mathcal{G}$. Then, with $\nabla_{\boldsymbol{w}}\mathcal{G}(\boldsymbol{w};\hat{\boldsymbol{u}}) = \hat{\boldsymbol{H}}(\boldsymbol{w})^{-1}$ a Newton update takes the form $\boldsymbol{d} : \mathbb{R}^{M\times m}\times\mathbb{R}^{M\times m}\to\mathbb{R}^{M\times m}$ with

$$\boldsymbol{d}(\boldsymbol{w},\hat{\boldsymbol{u}}) := \boldsymbol{w} - \hat{\boldsymbol{H}}(\boldsymbol{v})^{-1}\cdot\mathcal{G}(\boldsymbol{w};\hat{\boldsymbol{u}}). \tag{1.114}$$

The function $\boldsymbol{d}$ will in the following be called dual iteration function. Now, the Newton iteration for an input moment vector $\hat{\boldsymbol{u}}$ is given by

$$\boldsymbol{w}^{(l+1)} = \boldsymbol{d}(\boldsymbol{w}^{(l)},\hat{\boldsymbol{u}}). \tag{1.115}$$

The exact dual state can then be obtained by computing the fixed point of $\boldsymbol{d}$. This fixed point can be computed by converging the iteration (1.115), i.e. $\hat{\boldsymbol{v}} := \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}) = \lim_{l\to\infty}\boldsymbol{d}(\boldsymbol{w}^{(l)},\hat{\boldsymbol{u}})$. To obtain a finite number of iterations, a stopping criterion

$$\left\|\mathcal{G}(\boldsymbol{w}^{(l)};\hat{\boldsymbol{u}})\right\| < \tau \tag{1.116}$$

is used, where $\tau > 0$ is a user determined parameter.

Besides yielding a hyperbolic moment system and dissipating a physically correct entropy, the IPM method has several advantages: Choosing the entropy $s(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u}$ yields the stochastic-Galerkin method, i.e. IPM generalizes different intrusive methods. Furthermore, at least for scalar problems, IPM is significantly less oscillatory compared to SG [67]. The main weakness of the IPM method is its computational effort, since it requires repeatedly solving the non-linear system (1.112). Hence, the desirable properties of IPM come along with significantly increased numerical costs. However, IPM and minimal entropy methods in general are well suited for modern HPC architecture, which can be used to reduce the run time [39].

To visualize the previously discussed issues of the stochastic-Galerkin method as well as the benefits of IPM, let us again look at Sod's shock tube from Section 1.4.1. I.e. we solve the Euler equations for a gas, which initially is at rest and shows a shock in density and energy. Let us assume that the shock position is uncertain and we wish to quantify the arising uncertainty of the solution at a later time t with SG

and IPM. First, we need to pick a suitable entropy for the IPM method. In this work, we choose the entropy

$$s(\rho, \rho u, \rho E) = -\rho \ln\left(\rho^{-\gamma}\left(\rho E - \frac{(\rho u)^2}{2\rho}\right)\right), \tag{1.117}$$

though more choices are possible. For more information on the chosen numerical discretization of the IPM and SG systems, see Section 4.1. When starting the computation, the SG method fails already during the first time update. The reason for this can be seen in Figure 1.6. Here, the SG and IPM reconstructions of the gas density ρ are depicted at $t = 0$ at a fixed spatial cell. While the IPM reconstruction maintains positivity, the Gibbs phenomena that result from the polynomial representation of SG lead to negative density values. A similar behavior can be seen for the energy e. Then, the eigenvalues of the Euler equations, which include $v \pm \sqrt{\gamma p/\rho}$ become complex, i.e. the system is no longer hyperbolic.

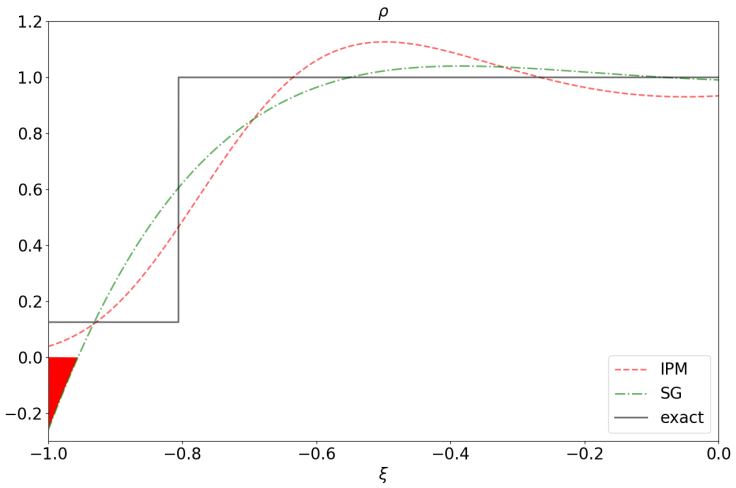


FIGURE 1.6: Initial gas density ρ for Sod's shock tube experiment with uncertain shock position. Shown are the SG and IPM results at fixed spatial position $x^* = 0.46$ when using 6 moments. More details on the chosen discretization of the respective moment systems can be found in Section 4.1. The view is zoomed to $\xi \in [-1, 0]$ and negative regions are marked in red.

1.4.6 Relation to kinetic theory

As for kinetic theory, the different discretization techniques previously discussed aim at representing the phase space with a finite number of unknowns. The core difference is an uncertain instead of an angular phase space. This is also reflected by the chosen methods, which show strong similarities [69]. Indeed, the nodal S_N discretization can be seen as a collocation method, where the main difference of S_N to Stochastic Collocation is the coupling of nodal equations through the collision operator. The same holds for intrusive methods in uncertainty quantification which can be interpreted as closures (similar to P_N and M_N, see Sections 1.3.1 and 1.3.2): Instead of directly choosing a finite-dimensional solution ansatz, let us derive a system which describes the exact time evolution of our moments. For this, we multiply the original equations (1.78) with the chosen set of basis functions $\varphi_i(\boldsymbol{\xi})$ with $|i| \leq N$

and integrate over the random domain Θ, which yields

$$\partial_t \hat{\boldsymbol{u}}_i(t,\boldsymbol{x}) + \nabla \cdot \langle \boldsymbol{f}\left(\boldsymbol{u}\left(t,\boldsymbol{x},\boldsymbol{\xi}\right)\right)\varphi_i\rangle = \boldsymbol{0}. \tag{1.118}$$

Ideally, we wish to determine the moments using this equation, since it describes the exact time evolution of $\hat{\boldsymbol{u}}$. Unfortunately, this system still depends on the unknown solution $\boldsymbol{u}\left(t,\boldsymbol{x},\boldsymbol{\xi}\right)$, which cannot be determined by our finite number of moments. Therefore, one needs to find a closure, i.e. we need to replace $\boldsymbol{u}$ by a function which only depends on the first M moments. As done for the gPC ansatz (1.84), we can for example choose a function

$$\mathcal{U}(\hat{\boldsymbol{u}};\boldsymbol{\xi}) = \sum_{|i|\leq N} \hat{\boldsymbol{u}}_i \varphi_i(\boldsymbol{\xi}), \tag{1.119}$$

i.e. a polynomial ansatz is used to reconstruct a solution in $\boldsymbol{\xi}$ when the moments are known. This ansatz is called a *closure* since it closes the moment system in that it yields a set of M equations for M unknowns. The polynomial closure ansatz of SG resembles the P_N method in the context of kinetic theory. Opposed to SG, the IPM closure is given by an optimization problem instead of a polynomial expansion (1.84). For a given convex entropy $s : \mathbb{R}^m \to \mathbb{R}$ of the original equations (1.78), this optimization problem reads[6]

$$\mathcal{U}(\hat{\boldsymbol{u}}) = \underset{\boldsymbol{u}\in L^1(\Theta)}{\arg\min} \langle s(\boldsymbol{u})\rangle \quad \text{subject to } \hat{\boldsymbol{u}}_i = \langle \boldsymbol{u}\varphi_i\rangle \text{ for } |i| \leq N. \tag{1.120}$$

As mentioned in Section 1.3.2 such a closure is called a minimum entropy closure in general or M_N closure in the kinetic context. To solve this constrained optimization problem, the saddle point of the Lagrangian L has to be determined, i.e.

$$\begin{aligned}
\max_{\boldsymbol{\lambda}} \min_{\boldsymbol{u}} L(\boldsymbol{u},\boldsymbol{\lambda}) &= \max_{\boldsymbol{\lambda}} \min_{\boldsymbol{u}} \left(\langle s(\boldsymbol{u})\rangle + \sum_{|i|\leq N} \boldsymbol{\lambda}_i^T \left(\hat{\boldsymbol{u}}_i - \langle \boldsymbol{u}\varphi_i\rangle\right)\right) \\
&= \max_{\boldsymbol{\lambda}} \left(\min_{\boldsymbol{u}} \langle s(\boldsymbol{u}) - \boldsymbol{u}^T\boldsymbol{\lambda}^T\boldsymbol{\varphi}\rangle + \sum_{|i|\leq N} \boldsymbol{\lambda}_i^T \hat{\boldsymbol{u}}_i \right) \\
&= \max_{\boldsymbol{\lambda}} \left(-\langle s_*(\boldsymbol{\lambda}^T\boldsymbol{\varphi})\rangle + \sum_{|i|\leq N} \boldsymbol{\lambda}_i^T \hat{\boldsymbol{u}}_i \right) \\
&= \min_{\boldsymbol{\lambda}} \left(\langle s_*(\boldsymbol{\lambda}^T\boldsymbol{\varphi})\rangle - \sum_{|i|\leq N} \boldsymbol{\lambda}_i^T \hat{\boldsymbol{u}}_i \right).
\end{aligned} \tag{1.121}$$

The Lagrange multipliers, or dual variables, are given by $\boldsymbol{\lambda}_i \in \mathbb{R}^m$ for $|i| \leq N$ and are collected in the matrix $\boldsymbol{\lambda} \in \mathbb{R}^{M\times m}$. Defining the dual state $\boldsymbol{\Lambda} := \boldsymbol{\lambda}^T\boldsymbol{\varphi}$, the Legendre transformation of the entropy is given by

$$\langle s_*(\boldsymbol{\Lambda})\rangle := \langle -s(\boldsymbol{u}(\boldsymbol{\Lambda})) + \boldsymbol{u}(\boldsymbol{\Lambda})^T\boldsymbol{\Lambda}\rangle$$

with $\boldsymbol{u}(\boldsymbol{\Lambda}) = \nabla_{\boldsymbol{\Lambda}} s_*(\boldsymbol{\Lambda})$. For more information on the dual approach see [15, Chapter 5] and for a more detailed derivation and discussion of the dual approach in kinetic theory, see [80]. Calculating the closure by solving a constrained optimization problem has now been reduced to finding the dual variables $\boldsymbol{\lambda}$ by solving the

[6] We assume that Θ is compact, in which case the ansatz must lie in $L^1(\Theta)$ to ensure the existence of a moment vector.

unconstrained dual problem (1.121). This problem is finite-dimensional as well as unconstrained. Plugging the resulting dual variables $\boldsymbol{\lambda}$ into the solution ansatz $\boldsymbol{u}(\boldsymbol{\Lambda})$ yields the closure

$$\mathcal{U}(\hat{\boldsymbol{u}}) = \boldsymbol{u}\left(\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}})^T\boldsymbol{\varphi}\right) \tag{1.122a}$$

$$\text{with } \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}) := \underset{\boldsymbol{\lambda}\in\mathbb{R}^{M\times m}}{\arg\min}\left\{\langle s_*(\boldsymbol{\lambda}^T\boldsymbol{\varphi})\rangle - \sum_{|i|\leq N}\boldsymbol{\lambda}_i^T\hat{\boldsymbol{u}}_i\right\}. \tag{1.122b}$$

Plugging this closure into the moment system (1.118) yields a closed set of equations, which is equivalent to the previously derived IPM system. Indeed, the Lagrange multipliers $\hat{\boldsymbol{\lambda}}$ play the role of the dual variables $\hat{\boldsymbol{v}}$ and the dual state $\boldsymbol{\Lambda}$ is equivalent to the gPC approximation of the entropy variables $\boldsymbol{v}_N$. Furthermore, the solution ansatz of the closure resembles the solution expressed in entropy variables, i.e.

$$\nabla_{\boldsymbol{\Lambda}} s_*(\boldsymbol{\Lambda}) = \left(\nabla_{\boldsymbol{\Lambda}} s\right)^{-1}(\boldsymbol{\Lambda}),$$

where we can either use $\boldsymbol{\Lambda}$ or $\boldsymbol{v}_N$ to denote the input. Then, the gradient of the dual problem (1.122b) equals the definition of $\mathcal{G}$, which we defined in (1.113). Hence, when using Newton's method to determine the root of $\mathcal{G}$, we are solving the dual problem (1.122b).

1.5 Recent work on hyperbolic problems with uncertainty

As discussed previously, hyperbolic problems tend to form shocks, which do not only appear in the physical, but also in random space. Unfortunately, this means that gPC expansions and quadrature rules have a slow rate of convergence, already for scalar uncertainties. Furthermore, solution approximations of different methods tend to show incorrect discontinuities in certain regions of the physical space, see e.g. [77, 70] for stochastic-Galerkin, [70, 110] for IPM and [8, 31] for Stochastic Collocation . These non-physical structures dissolve when the number of basis functions is increased [109, 103] or when artificial diffusion is added through either the spatial numerical method [103], filters [70] or limiters [118]. Also, a multi-element approach which divides the uncertain domain into cells and uses piece-wise polynomial basis functions to represent the solution has proven to mitigate non-physical discontinuities [140, 30]. Discontinuous structures commonly arise on a small portion of the space-time domain. Therefore, intrusive methods seem to be an adequate choice since they are well suited for adaptive strategies. By locally increasing the polynomial order [138, 65, 44, 71] or adding artificial viscosity [70] at certain spatial positions and time steps in which complex structures such as discontinuities occur, a given accuracy can be reached with significantly reduced numerical costs. As previously discussed, collocation methods show aliasing effects, which stem from the inexact approximation of integrals. Furthermore, collocation methods typically require a higher number of unknowns than intrusive methods to reach a given accuracy [145, 2, 71].

Chapter 2

Maximum-principle-satisfying second-order IPM scheme

In the following, we assume a scalar, hyperbolic conservation law which depends on a scalar random variable ξ and discuss if and how certain properties of the entropy solution are preserved by the discretization of the random space. While not all results presented in this chapter hold for systems, the remaining simplifications are imposed for ease of presentation. As discussed in Section 1.1.2, an important property of entropy solutions to scalar hyperbolic problems is the maximum principle, which states that for every $\xi \in \Theta$ we have

$$\min_{x \in D} u_{\mathrm{IC}}(x, \xi) \leq u(t, x, \xi) \leq \max_{x \in D} u_{\mathrm{IC}}(x, \xi)$$

for all $t \in \mathbb{R}^+$. Finite-volume schemes for deterministic problems, i.e. methods to discretize the spatial and time domain, are carefully constructed to satisfy this property, see for example [11, 20, 82, 148, 48]. The same should hold for the discretization of the random domain. However, the polynomial gPC ansatz (1.84) does not necessarily preserve the maximum principle, since oscillatory over- and undershoots can violate solution bounds, see e.g. Figure 1.4A.

The IPM method at least provides one with the ability to impose user-determined solution bounds u_- and u_+, since the entropy density s can be chosen such that the solution ansatz $s'(u)$ only takes values in (u_-, u_+). These bounds on the solution can be used to restrict the under- and overshoots of oscillations. In the original IPM paper [112] the log-barrier entropy density

$$s(u) = -\ln(u - u_-) - \ln(u_+ - u) \tag{2.1}$$

is used. Clearly this entropy does not allow an ansatz which takes on values outside the interval (u_-, u_+). Since we know that the solution should be bounded by

$$u_{\min} := \min_{x,\xi} u_{\mathrm{IC}}(x, \xi) \quad \text{and} \quad u_{\max} := \max_{x,\xi} u_{\mathrm{IC}}(x, \xi),$$

one can take $u_+ := u_{\max} + \Delta u$ and $u_- := u_{\min} - \Delta u$ with $\Delta u \in [0, \infty)$. When the solution to the primal problem (1.120) can only take values in (u_-, u_+), the problem is only feasible, if the moment vector $\hat{\boldsymbol{u}}$ lies in the set

$$\mathcal{R} := \left\{ \hat{\boldsymbol{u}} \in \mathbb{R}^{N+1} \,\middle|\, \exists u : \Theta \to (u_-, u_+) \text{ such that } \hat{\boldsymbol{u}} = \langle u \boldsymbol{\varphi} \rangle \right\}. \tag{2.2}$$

We call $\mathcal{R} \subseteq \mathbb{R}^{N+1}$ the *realizable set*. Note that here, we need to distinguish between realizability of moments and the solution itself. We call a solution realizable (or admissible), if it preserves certain mathematically or physically motivated bounds

(such as $u \in (u_-, u_+)$). A moment vector is called realizable, if it belongs to an underlying realizable solution. Realizability is important to keep in mind when designing numerical methods, because when the numerically computed moments leave the realizable set $\mathcal{R}$, the ansatz $u(\Lambda) = (s')^{-1}(\Lambda)$ is undefined, and so the IPM method crashes. Consequently, one of the main challenges facing the IPM method is the more complicated construction of high-order numerical schemes: Unlike the SG method, the moments $\hat{\boldsymbol{u}}$ of the numerical solution must stay within the realizable set, to ensure that the IPM ansatz can be reconstructed. Another challenge with IPM methods is that while they successfully dampen oscillations near the bounds u_- and u_+, the solutions can still oscillate heavily between these bounds.

We tackle these two challenges in this chapter. First, we give a naïve, out-of-the-box numerical method for the IPM equations in Section 2.1 to demonstrate the problem of maintaining realizability. Next, in Section 2.2, we begin to address the problem of numerically maintaining realizability with a first-order scheme through either a time-step restriction or modification of the numerical method. In Section 2.3 we extend these results to a second-order scheme. In Section 2.4, we discuss properties of the minimum-entropy approximation, and study an entropy which leads to smaller oscillations in the solution while enabling a maximum-principle. Section 2.5 presents numerical results for the uncertain Burgers' and advection equations.

2.1 Discretization of the IPM system

The IPM system (1.108) for a scalar conservation law can be rewritten as

$$\partial_t \hat{\boldsymbol{u}} + \partial_x \boldsymbol{F}(\hat{\boldsymbol{u}}) = \boldsymbol{0}$$

with the flux $\boldsymbol{F} : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$, $\boldsymbol{F}(\hat{\boldsymbol{u}}) = \langle f(u(\Lambda(\hat{\boldsymbol{u}})))\boldsymbol{\varphi}\rangle$ depending on the dual state

$$\Lambda(\hat{\boldsymbol{u}}) = \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}})^T \boldsymbol{\varphi},$$

where $\hat{\boldsymbol{v}} \in \mathbb{R}^{N+1}$ solves the dual problem (1.122b) for the scalar, one-dimensional case, i.e.

$$\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}) := \underset{\boldsymbol{v}\in\mathbb{R}^{N+1}}{\arg\min} \left\{ \langle s_*(\boldsymbol{v}^T\boldsymbol{\varphi})\rangle - \boldsymbol{v}^T\hat{\boldsymbol{u}} \right\}. \tag{2.3}$$

In the scalar case, the IPM ansatz becomes $u(\Lambda) = (s')^{-1}(\Lambda)$. Here, we use the notation of the dual state $\Lambda \equiv v_N$ (compare to (1.104)) to suppress the subscript N. Note that all dual states in this chapter use an order N expansion, i.e. the number of moments is always $N+1$. Furthermore, for efficiency of exposition, we sometimes omit the dependence on $\hat{\boldsymbol{u}}$.

The IPM system is hyperbolic, so it is naturally solved by a finite-volume method. First we discretize the spatial domain into cells. The discrete unknowns are chosen to be the spatial averages over each cell at time t_n, given by

$$\hat{u}_{ij}^n \simeq \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \hat{u}_i(t_n, x)dx.$$

If a moment vector in cell j at time t_n is denoted as $\hat{\boldsymbol{u}}_j^n = (u_{0j}^n, \cdots, u_{Nj}^n)^T \in \mathbb{R}^{N+1}$, the finite-volume scheme can be written in conservative form with the numerical flux

$\boldsymbol{F}^* : \mathbb{R}^{N+1} \times \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ as

$$\hat{\boldsymbol{u}}_j^{n+1} = \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^*(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) - \boldsymbol{F}^*(\hat{\boldsymbol{u}}_{j-1}^n, \hat{\boldsymbol{u}}_j^n)\right) \tag{2.4}$$

for $j = 1, \cdots, N_x$ and $n = 0, \cdots, N_t$, where N_x is the number of spatial cells and N_t is the number of time steps. The numerical flux is assumed to be consistent, i.e., that $\boldsymbol{F}^*(\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}}) = \boldsymbol{F}(\hat{\boldsymbol{u}})$. To ensure stability, a CFL condition has to be derived by investigating the eigenvalues of $\nabla \boldsymbol{F}$.

When a consistent numerical flux $f^* : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $f^* = f^*(u_\ell, u_r)$ is available for the deterministic problem (1.1), then for the IPM system we can simply take

$$\boldsymbol{F}^*(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) = \langle f^*(u(\Lambda(\hat{\boldsymbol{u}}_j^n)), u(\Lambda(\hat{\boldsymbol{u}}_{j+1}^n)))\boldsymbol{\varphi}\rangle. \tag{2.5}$$

This choice of the numerical flux is a common choice in kinetic theory and is called *kinetic flux* or *kinetic scheme* [26, 50, 107, 108]. The time update of the moment vector now becomes

$$\hat{\boldsymbol{u}}_j^{n+1} = \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\langle f^*(u(\Lambda_j^n), u(\Lambda_{j+1}^n))\boldsymbol{\varphi}\rangle - \langle f^*(u(\Lambda_{j-1}^n), u(\Lambda_j^n))\boldsymbol{\varphi}\rangle\right), \tag{2.6}$$

where $\Lambda_j^n := \Lambda(\hat{\boldsymbol{u}}_j^n)$ for all j. Note that the computation of Λ_j^n requires solving the dual problem (2.3) for the moment vector $\hat{\boldsymbol{u}}_j^n$.

Unfortunately (2.6) cannot be implemented because the dual problem cannot be solved exactly.[1] Instead, it must be solved numerically, for example with Newton's method. The stopping criterion for the numerical optimizer ensures that the approximate multiplier vector it returns, which we denote $\bar{\boldsymbol{v}}_j^n \in \mathbb{R}^{N+1}$ for the moment vector $\hat{\boldsymbol{u}}_j^n$, satisfies the stopping criterion (1.116), i.e. for the scalar case

$$\left\| \hat{\boldsymbol{u}}_j^n - \left\langle u\left(\left(\bar{\boldsymbol{v}}_j^n\right)^T \boldsymbol{\varphi}\right)\boldsymbol{\varphi}\right\rangle \right\| < \tau. \tag{2.7}$$

Once the numerical optimizer finds such a $\bar{\boldsymbol{v}}_j^n$, the corresponding dual state $\bar{\Lambda}_j^n := \left(\bar{\boldsymbol{v}}_j^n\right)^T \boldsymbol{\varphi} \in \mathbb{P}(\Theta)$ (where $\mathbb{P}$ denotes the space of polynomials with domain Θ) can be used in (2.6) for the unknown Λ_j^n. This gives Algorithm 1.

Algorithm 1 IPM for Uncertainty Quantification

1: **for** $j = 0$ to $N_x + 1$ **do**
2: $\quad \hat{\boldsymbol{u}}_j^0 = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x, \cdot)\boldsymbol{\varphi}\rangle dx$
3: **for** $n = 0$ to N_t **do**
4: $\quad$ **for** $j = 0$ to $N_x + 1$ **do**
5: $\quad\quad \bar{\boldsymbol{v}}_j^n \approx \arg\min_{\boldsymbol{v}} \left(\langle s_*(\boldsymbol{v}^T\boldsymbol{\varphi})\rangle - \boldsymbol{v}^T\hat{\boldsymbol{u}}_j^n\right)$ such that (2.7) holds
6: $\quad\quad \bar{\Lambda}_j^n = \left(\bar{\boldsymbol{v}}_j^n\right)^T \boldsymbol{\varphi}$
7: $\quad$ **for** $j = 1$ to N_x **do**
8: $\quad\quad \hat{\boldsymbol{u}}_j^{n+1} = \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\langle f^*(u(\bar{\Lambda}_j^n), u(\bar{\Lambda}_{j+1}^n))\boldsymbol{\varphi}\rangle - \langle f^*(u(\bar{\Lambda}_{j-1}^n), u(\bar{\Lambda}_j^n))\boldsymbol{\varphi}\rangle\right)$

[1] Equation (2.6) also includes integral evaluations which cannot be computed in closed form. Their approximation by numerical quadrature, however, does not play a role in the realizability problems we discuss below.

For most test cases in this chapter, Dirichlet boundary conditions are used, i.e. ghost cells with moment vectors $\hat{\boldsymbol{u}}_0^n = \langle u_L \boldsymbol{\varphi} \rangle$ and $\hat{\boldsymbol{u}}_{N_x+1}^n = \langle u_R \boldsymbol{\varphi} \rangle$ are implemented. Algorithm 1 crashes when the numerical optimizer cannot find a $\overline{\boldsymbol{v}}_j^n$ satisfying the stopping criterion. This can only[2] happen when the moment vector $\hat{\boldsymbol{u}}_j^n$ is not realizable. We tested an implementation of Algorithm 1 on the uncertain Burgers' equation as described in Section 1.4.1. We chose the parameters for the numerical scheme as in Section 2.5.1, and ran simulations with different values of the optimization tolerance τ and the solution-bound parameter Δu. We chose the time step Δt according to the classical time-step restriction

$$\frac{\Delta t}{\Delta x} \max_{u \in [u_-, u_+]} |f'(u)| \leq 1. \tag{2.8}$$

The solution bounds u_- and u_+, which parametrize the entropy (2.1), are important parameters in the implementation. Thus one would like to choose Δu as small as possible. Furthermore, since the maximum velocity $\max\{f'(u)\}$ is determined over the interval $u \in [u_-, u_+]$, the larger we take Δu, the larger the maximum velocity may be. A larger maximum velocity would lead the CFL condition to impose a tighter time-step restriction and add numerical viscosity. In [112] the authors chose $u_+ = u_{\max} + \Delta u$ and $u_- = u_{\min} - \Delta u$ with $\Delta u = 0.5$. Consequently, over- and undershoots as large as 0.5 are allowed, and we test a few values here. We chose all other parameters in the experiments as given in Section 2.5.1.

In Table 2.1, for different values of the optimization tolerance τ and the entropy parameter Δu we report how long Algorithm 1 ran until it crashed due to loss of realizability. The results indicate that this is more likely for smaller values of Δu, while decreasing the optimization tolerance seems to help slightly. It is clear, then, that this direct insertion of the numerical optimizer in Algorithm 1 gives a method which does not preserve realizability, i.e. the moments leave the realizable set $\mathcal{R}$.

TABLE 2.1: Number of time steps until the dual problem cannot be solved. Check marks indicate successful calculations for all time steps. Space and time resolution as well as the number of moments are kept fixed.

Δu \ τ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
10^{-1}	✓	✓	✓	✓	✓
10^{-3}	3	3	12	✓	✓
10^{-5}	✓	9	6	8	19

In addition to an increased chance of crashes, smaller values of Δu can also lead to more oscillatory solutions. We consider this aspect later in Section 2.4. First, we treat the problem of realizability.

2.2 Modified scheme to preserve realizability

To understand the reason for the loss of realizability in our tests, we analyze the effects of not being able to solve the optimization problem exactly. It turns out that

[2] Except for some realizable cases where the problem is so poorly conditioned that the numerical optimizer fails to find the minimizer even though it exists. See, e.g., [3].

the optimization error can destroy the monotonicity properties that would otherwise be inherited from the underlying scheme for the original PDE (1.1) and would guarantee bounds on the discrete solution.

2.2.1 Monotonicity and the optimization error

The main step in Algorithm 1 is

$$\hat{\boldsymbol{u}}_j^{n+1} = \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\langle f^*(u(\overline{\Lambda}_j^n), u(\overline{\Lambda}_{j+1}^n))\boldsymbol{\varphi}\rangle - \langle f^*(u(\overline{\Lambda}_{j-1}^n), u(\overline{\Lambda}_j^n))\boldsymbol{\varphi}\rangle\right). \tag{2.9}$$

We can analyze the right-hand side as a function of the point values of the dual states $\overline{\Lambda}_j^n$ by defining $H : \mathbb{R}\times\mathbb{R}\times\mathbb{R}\times\mathbb{R}\to\mathbb{R}$ as

$$\begin{aligned} H(\Lambda_\ell, \Lambda_c, \Lambda_r; \Delta\Lambda) := u(\Lambda_c + \Delta\Lambda) \\ - \frac{\Delta t}{\Delta x}\left(f^*(u(\Lambda_c), u(\Lambda_r)) - f^*(u(\Lambda_\ell), u(\Lambda_c))\right), \end{aligned} \tag{2.10}$$

where $\Delta\Lambda$ is used for a point value of the the optimization error in Λ. (We typically view $\Delta\Lambda$ as a fixed parameter and are more interested in the behavior of H as a function of its first three arguments.) With

$$\Delta\Lambda_j^n = \Delta\Lambda_j^n(\xi) := \Lambda_j^n(\xi) - \overline{\Lambda}_j^n(\xi),$$

(2.9) can now be written as

$$\hat{\boldsymbol{u}}_j^{n+1} = \left\langle H(\overline{\Lambda}_{j-1}^n, \overline{\Lambda}_j^n, \overline{\Lambda}_{j+1}^n; \Delta\Lambda_j^n)\boldsymbol{\varphi}\right\rangle. \tag{2.11}$$

Since $u(\hat{\Lambda}_j^n) = u(\overline{\Lambda}_j^n + \Delta\Lambda_j^n)$ fulfills the moment constraint in (1.120) exactly, the equality $\hat{\boldsymbol{u}}_j^n = \langle u(\overline{\Lambda}_j^n + \Delta\Lambda_j^n)\boldsymbol{\varphi}\rangle$ holds. Therefore, multiplying the first term of (2.10) when $\Lambda_c \equiv \overline{\Lambda}_{j-1}^n$ and $\Delta\Lambda \equiv \Delta\Lambda_j^n$ with $\boldsymbol{\varphi}$ and integrating with respect to ξ yields the moment vector $\hat{\boldsymbol{u}}_j^n$ in (2.9).

In (2.11), we have written $\hat{\boldsymbol{u}}_j^{n+1}$ simply as the moments of the update function H, so the realizability of $\hat{\boldsymbol{u}}_j^{n+1}$ can be established by considering whether H lies in (u_-, u_+). This leads directly to the concept of monotone schemes (see Section 1.2.1) for scalar conservation laws, because monotone schemes give numerical solutions which satisfy a maximum principle. Thus strategies in the proof of Theorem 2 can be used to show that monotonicity ensures realizability.

Proposition 1. *Assume H is monotonically increasing in each of its first three arguments. Then if the entropy ansatz $u(\Lambda)$ only takes values in (u_-, u_+), the moment vector $\hat{\boldsymbol{u}}_j^{n+1}$ computed according to (2.11) (or equivalently (2.9)) is realizable for any dual states $\overline{\Lambda}_{j-1}^n$, $\overline{\Lambda}_j^n$, and $\overline{\Lambda}_{j+1}^n$.*

Proof. We must show that $\hat{\boldsymbol{u}}_j^{n+1}$ lies in the realizable set $\mathcal{R}$, which we defined in (2.2). Due to (2.11) it suffices to show that $H(\overline{\Lambda}_{j-1}^n(\xi), \overline{\Lambda}_j^n(\xi), \overline{\Lambda}_{j+1}^n(\xi); \Delta\Lambda_j^n(\xi)) \in (u_-, u_+)$ for all $\xi \in \Theta$. For an arbitrary but fixed ξ, let us define

$$\overline{\Lambda}_{j,\max}^n(\xi) := \max\left\{\overline{\Lambda}_{j-1}^n(\xi), \overline{\Lambda}_j^n(\xi), \overline{\Lambda}_{j+1}^n(\xi)\right\}.$$

By monotonicity we have for this ξ

$$\begin{aligned} H(\overline{\Lambda}^n_{j-1}(\xi), \overline{\Lambda}^n_j(\xi), \overline{\Lambda}^n_{j+1}(\xi); \Delta\Lambda^n_j(\xi)) &\le H(\overline{\Lambda}^n_{j,\max}(\xi), \overline{\Lambda}^n_{j,\max}(\xi), \overline{\Lambda}^n_{j,\max}(\xi); \Delta\Lambda^n_j(\xi)) \\ &= u(\overline{\Lambda}^n_{j,\max}(\xi) + \Delta\Lambda^n_j(\xi)) < u_+. \end{aligned}$$

Since ξ was arbitrary, we have $H < u_+$ for every ξ. The other direction, $H > u_-$ can be shown analogously. Finally, since $H \in (u_-, u_+)$ for every ξ, then $\hat{\boldsymbol{u}}^{n+1}_j = \langle H(\overline{\Lambda}^n_{j-1}, \overline{\Lambda}^n_j, \overline{\Lambda}^n_{j+1}; \Delta\Lambda^n_j)\boldsymbol{\varphi}\rangle$ is realizable. □

Now, monotonicity of H depends on the monotonicity of the scheme defined by the numerical flux $f^* = f^*(u_\ell, u_r)$ for the original PDE (1.1). We assume that under the standard CFL condition (2.8) which was

$$\frac{\Delta t}{\Delta x} \max_{u\in[u_-,u_+]} |f'(u)| \le 1,$$

$f^*(u_\ell, u_r)$ gives a monotone scheme for the underlying equation. Hence the deterministic update function defined in (1.34) which was

$$h(u, v, w) = v - \frac{\Delta t}{\Delta x}\left(f^*(v, w) - f^*(u, v)\right)$$

is monotonically increasing in each argument.[3] Recalling (1.37), this implies

$$\frac{\partial f^*}{\partial u_\ell} \ge 0, \tag{2.12a}$$

$$1 - \frac{\Delta t}{\Delta x}\left(\frac{\partial f^*}{\partial u_\ell} - \frac{\partial f^*}{\partial u_r}\right) \ge 0, \tag{2.12b}$$

$$\frac{\partial f^*}{\partial u_r} \le 0. \tag{2.12c}$$

Using this along with properties of the entropy ansatz, we can immediately show that H is monotone in the first and third arguments, since

$$\frac{\partial H}{\partial \Lambda_\ell} = \frac{\Delta t}{\Delta x}\frac{\partial f^*}{\partial u_\ell}u'(\Lambda_\ell) = \frac{\Delta t}{\Delta x}\underbrace{\frac{\partial f^*}{\partial u_\ell}}_{\substack{\ge 0 \\ \text{by (2.12a)}}}\underbrace{\frac{1}{s''(u(\Lambda_\ell))}}_{\substack{\ge 0 \\ \text{by convexity}}} \ge 0, \tag{2.13a}$$

$$\frac{\partial H}{\partial \Lambda_r} = -\frac{\Delta t}{\Delta x}\underbrace{\frac{\partial f^*}{\partial u_r}}_{\substack{\le 0 \\ \text{by (2.12c)}}}\underbrace{\frac{1}{s''(u(\Lambda_r))}}_{\substack{\ge 0 \\ \text{by convexity}}} \ge 0, \tag{2.13b}$$

where we used that $u'(\Lambda) = \left((s')^{-1}(\Lambda)\right)' = 1/s''(u(\Lambda))$. The properties in (2.13) hold for *any* value of $\Delta\Lambda^n_j$. But in the second argument, the optimization error $\Delta\Lambda^n_j$

[3] The update functions H and h are simply related by

$$h(u(\Lambda_\ell), u(\Lambda_c), u(\Lambda_r)) = H(\Lambda_\ell, \Lambda_c, \Lambda_r; 0).$$

can destroy monotonicity:

$$\frac{\partial H}{\partial \Lambda_c} = u'(\Lambda_c + \Delta\Lambda_j^n) - \frac{\Delta t}{\Delta x}\left(\frac{\partial f^*}{\partial u_\ell}u'(\Lambda_c) - \frac{\partial f^*}{\partial u_r}u'(\Lambda_c)\right) \tag{2.14a}$$

$$= u'(\Lambda_c + \Delta\Lambda_j^n)\left(1 - \frac{u'(\Lambda_c)}{u'(\Lambda_c + \Delta\Lambda_j^n)}\frac{\Delta t}{\Delta x}\left(\frac{\partial f^*}{\partial u_\ell} - \frac{\partial f^*}{\partial u_r}\right)\right). \tag{2.14b}$$

The u' factor in front is again nonnegative by convexity of s, but since the ratio $u'(\Lambda_c)/u'(\Lambda_c + \Delta\Lambda_j^n)$ can certainly be larger than one, the standard CFL condition (2.8) cannot be applied to show nonnegativity of the second factor in (2.14b).

There are now two ways to achieve monotonicity despite the optimization error.

2.2.2 Modifying the CFL condition

The more precisely the numerical optimizer solves the optimization problem, the smaller the ratio $u'(\Lambda_c)/u'(\Lambda_c + \Delta\Lambda_j^n)$ becomes. This suggests using it as a stopping criterion and then incorporating it into a modified CFL condition. Summing up the findings from Section 2.2.1, we obtain the following theorem:

Theorem 8. *Assume that the entropy ansatz only takes values in (u_-, u_+) and that f^* gives a monotone scheme. Then when the numerical optimizer enforces the stopping criterion*

$$\max_{\xi\in\Theta}\left\{\max_{\Lambda\in\left[\bar{\Lambda}_{j,min}^n, \bar{\Lambda}_{j,max}^n\right]}\frac{u'(\Lambda(\xi))}{u'(\Lambda(\xi) + \Delta\Lambda_j^n(\xi))}\right\} \leq \gamma, \tag{2.15}$$

where

$$\bar{\Lambda}_{j,min}^n(\xi) := \min\left\{\bar{\Lambda}_{j-1}^n(\xi), \bar{\Lambda}_j^n(\xi), \bar{\Lambda}_{j+1}^n(\xi)\right\},$$

$$\text{and}\quad \bar{\Lambda}_{j,max}^n(\xi) := \max\left\{\bar{\Lambda}_{j-1}^n(\xi), \bar{\Lambda}_j^n(\xi), \bar{\Lambda}_{j+1}^n(\xi)\right\},$$

the new moment vector $\hat{\boldsymbol{u}}_j^{n+1}$ computed by (2.11) (i.e., (2.9)) is realizable under the modified CFL condition

$$\gamma\frac{\Delta t}{\Delta x}\max_{u\in[u_-,u_+]}|f'(u)| \leq 1. \tag{2.16}$$

The condition (2.15) can be used instead of or in addition to (2.7). The user chooses the parameter γ. Larger values of γ make the condition easier to fulfill, i.e., require fewer optimization iterations, but come at the cost of requiring smaller time steps and leading to more diffusive solutions.

But a stopping criterion based on (2.15) cannot be implemented directly because $\Delta\Lambda_j^n$ is of course unknown. An approximation of $\Delta\Lambda_j^n = (\hat{\boldsymbol{v}}_j^n - \bar{\boldsymbol{v}}_j^n)^T\boldsymbol{\varphi}$ can be constructed using the Newton step. If we let $\bar{\boldsymbol{v}} \in \mathbb{R}^{N+1}$ denote an iterate in the optimization algorithm, $\hat{\boldsymbol{H}} \in \mathbb{R}^{N+1\times N+1}$ the Hessian and $\boldsymbol{g} \in \mathbb{R}^{N+1}$ the gradient of the dual problem (2.3), we approximate $\hat{\boldsymbol{v}}$ by

$$\hat{\boldsymbol{v}} \approx \bar{\boldsymbol{v}} - \zeta\hat{\boldsymbol{H}}^{-1}(\bar{\boldsymbol{v}})\boldsymbol{g}(\bar{\boldsymbol{v}}). \tag{2.17}$$

A safety parameter ζ is used to prevent underestimating the distance between $\boldsymbol{v}$ and $\hat{\boldsymbol{v}}$.

Even with this approximation, a stopping criterion based on (2.15) faces the problem that, since it includes $\overline{\Lambda}^n_{j\pm1}$, the numerical solutions at neighboring cells are coupled, thus destroying parallelizability. We avoid this by simply taking $\overline{\Lambda}^n_{j,\min} = \overline{\Lambda}^n_{j,\max} = \overline{\Lambda}^n_j$ and assuming that the safety parameter ζ can account for the error this introduces.

There are potential drawbacks of using Algorithm 1 with the modified CFL condition (2.16). First, it further restricts the time step, which introduces numerical diffusion. Second, the stopping criterion is difficult to implement, and it's not immediately clear if we can practically satisfy it for a reasonably small value of γ. Third, the choice $\Delta u = 0$ is prohibited if the initial condition takes on values of $\min u_{\text{IC}}$ or $\max u_{\text{IC}}$ on a nonzero measure. This is because in this case, the correct dual state $\hat{\Lambda}$ goes to infinity, leading to an infinite value of γ no matter how precisely the numerical optimizer solves the dual problem. We explore these potential problems in our numerical results in Section 2.5.

The issue of realizability is also an issue for minimum-entropy methods in kinetic theory. A realizability-preserving modified CFL condition similar to the one presented in Theorem 8 was derived in [3]. But in kinetic theory, one only needs to ensure the nonnegativity of the underlying update, whereas we need to enforce both upper and lower bounds. Because of this difference the modified CFL condition from [3] is not enough to ensure realizability for the IPM method. We illustrate this in the following example:

Example 3. *Assume that we wish to solve the uncertain linear advection equation*

$$\begin{aligned} \partial_t u + \xi \partial_x u &= 0, \\ u(t=0,x,\xi) &= u_{IC}(x) \end{aligned}$$

where ξ is uniformly distributed between 0 and 1. For the underlying scheme, we use an upwind flux $f^*(u,v) = \xi u$. The underlying scheme is now given by

$$u^{n+1}_j = u(\Lambda^n_j) - \frac{\xi \Delta t}{\Delta x}\left(u(\overline{\Lambda}^n_j) - u(\overline{\Lambda}^n_{j-1})\right),$$

where we choose u such that the solution is bounded by $u_+ > u_- \geq 0$. Defining $u^n_j := u(\Lambda^n_j)$ and $\bar{u}^n_j := u(\overline{\Lambda}^n_j)$, as well as $\eta^n_j := \bar{u}^n_j / u^n_j$, we can rewrite this scheme in difference form as

$$u^{n+1}_j = \left(1 - \frac{\xi \Delta t}{\Delta x}\eta^n_j\right) u^n_j + \frac{\xi \Delta t}{\Delta x}\eta^n_{j-1} u^n_{j-1}. \tag{2.18}$$

Now, since we solve the optimization problem only approximately, in general $\eta_{j-1} \neq \eta_j$. Thus u^{n+1}_j is not—for any time step Δt—a convex combination of u^n_{j-1} and u^n_j. Thus we cannot guarantee $u^{n+1}_j \in [u_-, u_+]$. Even if we both enforce the stopping criterion[4]

$$\eta_{\min} \leq \eta^n_j \leq \eta_{\max}, \qquad \text{for all } j \text{ and } n,$$

in the numerical optimizer, where $0 < \eta_{\min} < 1 < \eta_{\max}$ are parameters, and restrict the time-step by the CFL condition

$$\frac{\xi \Delta t}{\Delta x}\eta_{\max} < 1,$$

[4]This stopping criterion is a stronger version of the criterion in [3].

the best bounds we can get on u_j^{n+1} *in* (2.18) *are*

$$\left(1 - \frac{\xi \Delta t}{\Delta x}\left(\eta_{\max} - \eta_{\min}\right)\right) u_- \leq u_j^{n+1} \leq \left(1 + \frac{\xi \Delta t}{\Delta x}\left(\eta_{\max} - \eta_{\min}\right)\right) u_+,$$

assuming $u_{j-1}^n \in [u_-, u_+]$ *and* $u_j^n \in [u_-, u_+]$. *These bounds are sharp (i.e., they can be arbitrarily closely reached, depending on the particular values of* u_{j-1}^n, u_j^n, η_{j-1}, *and* η_j*), and both lie outside of the interval* $[u_-, u_+]$ *for any non-exact solution of the optimization* ($\eta_{\min} < \eta_{\max}$).

2.2.3 Modifying the scheme

Another way to prevent the optimization error from destroying the monotonicity properties of the underlying scheme is to remove the optimization error completely from our application of the underlying scheme, so that the ratio $u'(\Lambda_c)/u'(\Lambda_c + \Delta\Lambda_j^n)$ doesn't even appear in (2.14). This is the case if $\Delta\Lambda_j^n = 0$, i.e. if the dual state belonging to the first term of H equals the inexact dual state used in the numerical fluxes. Since the exact dual state cannot be computed, this means using the dual states $\overline{\Lambda}_j^n$ also in the first term of H.

More specifically, let us define the modified update function $\widetilde{H}(\Lambda_\ell, \Lambda_c, \Lambda_r) = H(\Lambda_\ell, \Lambda_c, \Lambda_r; 0)$, i.e.,

$$\widetilde{H}(\Lambda_\ell, \Lambda_c, \Lambda_r) := u(\Lambda_c) - \frac{\Delta t}{\Delta x}\left(f^*(u(\Lambda_c), u(\Lambda_r)) - f^*(u(\Lambda_\ell), u(\Lambda_c))\right).$$

Now $\widetilde{H}$ immediately inherits the monotonicity properties of h in (1.34) under the original CFL condition (2.8), no matter how big or small the optimization error is. The algorithm when using $\widetilde{H}$ instead of H as underlying function can be written in the original form as

$$\hat{\boldsymbol{u}}_j^{n+1} = \left\langle \widetilde{H}(\overline{\Lambda}_{j-1}^n, \overline{\Lambda}_j^n, \overline{\Lambda}_{j+1}^n)\boldsymbol{\varphi} \right\rangle \tag{2.19a}$$

$$= \overline{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\langle f^*(u(\overline{\Lambda}_j^n), u(\overline{\Lambda}_{j+1}^n))\boldsymbol{\varphi}\rangle - \langle f^*(u(\overline{\Lambda}_{j-1}^n), u(\overline{\Lambda}_j^n))\boldsymbol{\varphi}\rangle\right), \tag{2.19b}$$

where $\overline{\boldsymbol{u}}_j^n := \langle u(\overline{\Lambda}_j^n)\boldsymbol{\varphi}\rangle \in \mathbb{R}^{N+1}$, and we present it in Algorithm 2.

Algorithm 2 Modified IPM algorithm

for $j = 0$ to $N_x + 1$ **do**
  $\hat{\boldsymbol{u}}_j^0 = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x, \cdot)\boldsymbol{\varphi}\rangle dx$
for $n = 0$ to N_t **do**
  for $j = 0$ to $N_x + 1$ **do**
    $\overline{\boldsymbol{v}}_j^n \approx \arg\min_{\boldsymbol{v}} \left(\langle s_*(\boldsymbol{v}^T\boldsymbol{\varphi})\rangle - \boldsymbol{v}^T\hat{\boldsymbol{u}}_j^n\right)$ such that (2.7) holds
    $\overline{\Lambda}_j^n = \left(\overline{\boldsymbol{v}}_j^n\right)^T\boldsymbol{\varphi}$
    $\overline{\boldsymbol{u}}_j^n = \langle u(\overline{\Lambda}_j^n)\boldsymbol{\varphi}\rangle$
  for $j = 1$ to N_x **do**
    $\hat{\boldsymbol{u}}_j^{n+1} = \overline{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\langle f^*(u(\overline{\Lambda}_j^n), u(\overline{\Lambda}_{j+1}^n))\boldsymbol{\varphi}\rangle - \langle f^*(u(\overline{\Lambda}_{j-1}^n), u(\overline{\Lambda}_j^n))\boldsymbol{\varphi}\rangle\right)$

But of course, one cannot simply use any value of the optimization tolerance τ and expect to end up with accurate results. However, if the numerical flux $\boldsymbol{F}^*$ is Lipschitz continuous in each argument with constant K, the error between the update of Algorithm 2 and the exact update of (2.6) is simply $\mathcal{O}(\tau)$. Indeed, let $c := \Delta t/\Delta x$; then we have

$$\left\| \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^*(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n)) - \boldsymbol{F}^*(\hat{\boldsymbol{u}}_{j-1}^n, \hat{\boldsymbol{u}}_j^n)\right) - \left(\bar{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^*(\bar{\boldsymbol{u}}_j^n, \bar{\boldsymbol{u}}_{j+1}^n) - \boldsymbol{F}^*(\bar{\boldsymbol{u}}_{j-1}^n, \bar{\boldsymbol{u}}_j^n)\right)\right) \right\| \leq (1 + 4cK)\,\tau.$$

Therefore we simply need to choose τ with the order of accuracy of the one-step error, which in this case is $\mathcal{O}(\Delta t \Delta x) = \mathcal{O}(\Delta x^2)$. Then the results computed by Algorithm 2 have the same order of accuracy as those computed by the exact method.

The main drawback to Algorithm 2 is that it is no longer in conservative form. However, when we take $\tau = \mathcal{O}(\Delta x^2)$, the nonconservative part vanishes as the grid is refined. Furthermore, in our numerical results below we did not observe any large increases in error compared to the solutions computed using the method presented in Section 2.2.2 with small values of γ.

Remark 9. *Proposition 1 shows realizability if integrals are evaluated exactly, by showing that $H \in (u_-, u_+)$ for all ξ. When using quadrature rules to approximate integrals, it suffices to show $H \in (u_-, u_+)$ for all quadrature points ξ_k, hence the requirements of Proposition 1 ensure realizability when using quadrature rules. For more details on the realizable set for quadrature rules, see [5].*

2.3 Extending the scheme to higher order

The main computational expense of minimum-entropy methods comes from the repeated numerical solution of the dual problem, which needs to be solved for every spatial cell. With a high-order method, fewer spatial cells can achieve a desired level of accuracy. In this section we show how to construct a realizability-preserving second-order method.

2.3.1 Second-order spatial reconstruction

First we give a stable second-order method for the original PDE (1.1), following Section 1.2.4 and then plug the entropy ansatz $u(\Lambda)$ into this method and integrate the equations against the basis functions $\boldsymbol{\varphi}$ to get a second-order method for the IPM system.

We start by defining a linear spatial reconstruction of the solution in each cell j by $p_j^n(x) = u_j^n + (x - x_j)\sigma_j^n$. Here $\sigma_j^n := \sigma(u_{j-1}^n, u_j^n, u_{j+1}^n)$ is the slope of the reconstruction in cell j at time step t_n. We use the second-order stable *minmod* slope (1.60). The reconstructions give cell edge values

$$u_{j\pm1/2}^{n,\mp} := u_j^n \pm \sigma_j^n \frac{\Delta x}{2}, \tag{2.20}$$

which are inserted into the numerical flux to give the time update:

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}(f^*(u_{j+1/2}^{n,-}, u_{j+1/2}^{n,+}) - f^*(u_{j-1/2}^{n,-}, u_{j-1/2}^{n,+})). \tag{2.21}$$

When we use the slopes given by the minmod limiter, the reconstructions further have the property that the edge values are bounded by the values of the cell means. This property is crucial for realizability.[5]

For the IPM method, we apply this numerical scheme point-wise in ξ using the entropy ansätze computed by the numerical optimizer. That is, for every ξ we compute the slope

$$\tilde{\sigma}_j^n = \sigma(u(\bar{\Lambda}_{j-1}^n), u(\bar{\Lambda}_j^n), u(\bar{\Lambda}_{j+1}^n)). \tag{2.22}$$

This gives the edge values

$$u_{j\pm1/2}^{n,\mp}(\bar{\Lambda}_{j-1}^n, \bar{\Lambda}_j^n, \bar{\Lambda}_{j+1}^n) := u(\bar{\Lambda}_j^n) \pm \frac{\Delta x}{2}\tilde{\sigma}_j^n.$$

Now we want to consider the monotonicity of the time update (2.21) with respect to the dual states of the cell average and both sides of the neighboring edges. For this we need to define the dual states of the edges,

$$\bar{\Lambda}_{j\pm1/2}^{n,\mp} := s'(u_{j\pm1/2}^{n,\mp}(\bar{\Lambda}_{j-1}^n, \bar{\Lambda}_j^n, \bar{\Lambda}_{j+1}^n)), \tag{2.23}$$

so that we can write (2.21) applied to IPM with

$$\begin{aligned} H_2(\Lambda_c, \Lambda_r^-, \Lambda_r^+, \Lambda_\ell^-, \Lambda_\ell^+; \Delta\Lambda) &:= u(\Lambda_c + \Delta\Lambda) \\ &- \frac{\Delta t}{\Delta x}\Big(f^*(u(\Lambda_r^-), u(\Lambda_r^+)) - f^*(u(\Lambda_\ell^-), u(\Lambda_\ell^+))\Big), \end{aligned} \tag{2.24a}$$

as

$$\hat{\boldsymbol{u}}_j^{n+1} = \langle H_2(\bar{\Lambda}_j^n, \bar{\Lambda}_{j+1/2}^{n,-}, \bar{\Lambda}_{j+1/2}^{n,+}, \bar{\Lambda}_{j-1/2}^{n,-}, \bar{\Lambda}_{j-1/2}^{n,+}; \Delta\Lambda_j^n)\boldsymbol{\varphi}\rangle. \tag{2.24b}$$

After having derived this underlying scheme we can find a time-step restriction which ensures realizability, following the derivation in Section 1.2.4.

Theorem 10. *Assume that the entropy ansatz only takes values in (u_-, u_+) and that f^* gives a monotone scheme. Then the time-updated moment vector $\hat{\boldsymbol{u}}_j^{n+1}$ from the second-order in space scheme (2.24) is realizable under the time-step restriction*

$$\widetilde{\gamma} \max_{u\in[u_-,u_+]} |f'(u)| \frac{\Delta t}{\Delta x} \leq \frac{1}{2} \tag{2.25}$$

where $\widetilde{\gamma}$ satisfies

$$\max_{\xi\in\Theta}\left\{\max_{\Lambda\in\left[\bar{\Lambda}_{j,min}^n, \bar{\Lambda}_{j,max}^n\right]} \frac{u'(\Lambda)}{u'\left(\Lambda + \Delta\Lambda_{j\pm1/2}^{n,\mp}\right)}\right\} \leq \tilde{\gamma}, \tag{2.26}$$

[5] For other slopes which do not have this property, one would have to implement a bound-preserving limiter, see Section 1.2.4.

with

$$\overline{\Lambda}^n_{j,min}(\xi) := \min_{j-2\le i\le j+2} \overline{\Lambda}^n_i(\xi) \quad \textit{and} \quad \overline{\Lambda}^n_{j,max}(\xi) := \max_{j-2\le i\le j+2} \overline{\Lambda}^n_i(\xi).$$

Proof. As in Proposition 1, we show that $H_2(\overline{\Lambda}^n_j, \overline{\Lambda}^{n,-}_{j+1/2}, \overline{\Lambda}^{n,+}_{j+1/2}, \overline{\Lambda}^{n,-}_{j-1/2}, \overline{\Lambda}^{n,+}_{j-1/2}; \Delta\Lambda^n_j)$ increases monotonically in its first five arguments.

We show monotonicity by adopting the technique of writing H_2 as a convex combination of evaluations of the first-order scheme of (2.11) [83]. We write

$$u(\overline{\Lambda}^n_j + \Delta\Lambda^n_j) = u(\hat{\Lambda}^n_j) = \frac{1}{2}\left(u(\Lambda^{n,+}_{j-1/2}) + u(\Lambda^{n,-}_{j+1/2})\right), \tag{2.27}$$

where $\Lambda^{n,\mp}_{j\pm1/2}$ are defined as in (2.23) but with $(\overline{\Lambda}^n_{j-1}, \overline{\Lambda}^n_j, \overline{\Lambda}^n_{j+1})$ replaced by $(\hat{\Lambda}^n_{j-1}, \hat{\Lambda}^n_j, \hat{\Lambda}^n_{j+1})$,[6] and insert this into (2.24a), so that after adding and subtracting

$$\frac{\Delta t}{\Delta x} f^*\left(u\left(\overline{\Lambda}^{n,+}_{j-1/2}\right), u\left(\overline{\Lambda}^{n,-}_{j+1/2}\right)\right)$$

we can write H_2 as

$$\begin{aligned} &H_2(\overline{\Lambda}^n_j, \overline{\Lambda}^{n,-}_{j+1/2}, \overline{\Lambda}^{n,+}_{j+1/2}, \overline{\Lambda}^{n,-}_{j-1/2}, \overline{\Lambda}^{n,+}_{j-1/2}; \Delta\Lambda^n_j) \\ &\quad = \frac{1}{2}\left(H_1(\overline{\Lambda}^{n,+}_{j-1/2}, \overline{\Lambda}^{n,-}_{j+1/2}, \overline{\Lambda}^{n,+}_{j+1/2}; \Delta\Lambda^-_{j+1/2}) + H_1(\overline{\Lambda}^{n,-}_{j-1/2}, \overline{\Lambda}^{n,+}_{j-1/2}, \overline{\Lambda}^{n,-}_{j+1/2}; \Delta\Lambda^+_{j-1/2})\right), \end{aligned} \tag{2.28}$$

where

$$H_1(\Lambda_\ell, \Lambda_c, \Lambda_r; \Delta\Lambda) := u(\Lambda_c + \Delta\Lambda) - 2\frac{\Delta t}{\Delta x}\left(f^*(u(\Lambda_c), u(\Lambda_r)) - f^*(u(\Lambda_\ell), u(\Lambda_c))\right)$$

and

$$\Delta\Lambda^{n,\mp}_{j\pm1/2} := \Lambda^{n,\mp}_{j\pm1/2} - \overline{\Lambda}^{n,\mp}_{j\pm1/2}. \tag{2.29}$$

The function H_1 is similar to the first-order update function (2.10), so that one readily recognizes that each H_1 term in (2.28) is monotone in the relevant arguments under the conditions

$$2\frac{u'(\Lambda)}{u'\left(\Lambda + \Delta\Lambda^{n,\mp}_{j\pm1/2}\right)} \max_{u\in[u_-,u_+]} |f'(u)| \frac{\Delta t}{\Delta x} \le 1, \tag{2.30}$$

for $\Lambda \in \left[\overline{\Lambda}^n_{j,\min}, \overline{\Lambda}^n_{j,\max}\right]$ respectively.

Thus under (2.25) H_2 is monotone in its first five arguments, and the realizability of $\hat{\boldsymbol{u}}^{n+1}_j$ follows. □

Unfortunately a stopping criterion based on (2.26) leads to an even stronger coupling of the numerical optimization. We avoid this by adopting the same strategy as in Section 2.2.3: that is, we replace H_2 with

$$\widetilde{H}_2(\Lambda_c, \Lambda^-_r, \Lambda^+_r, \Lambda^-_\ell, \Lambda^+_\ell) := H_2(\Lambda_c, \Lambda^-_r, \Lambda^+_r, \Lambda^-_\ell, \Lambda^+_\ell; 0).$$

[6] In words, $\Lambda^{n,\mp}_{j\pm1/2}$ are derived from the pointwise linear reconstruction using the values of the exact entropy ansatz instead of the approximate entropy ansatz returned by the numerical optimizer.

Thus we do not have to consider the optimization error when checking monotonicity, and we get monotonicity under the condition

$$\max_{u\in[u_-,u_+]} |f'(u)|\frac{\Delta t}{\Delta x} \leq \frac{1}{2}. \tag{2.31}$$

In order to maintain accuracy, we use the stopping criterion (2.7) with $\tau = \mathcal{O}(\Delta x^3)$.

Remark 11. *When replacing moments as proposed in Section 2.2.3, the optimization error no longer affects realizability. In this case, the IPM solution inherits the bounds guaranteed by the underlying scheme $\widetilde{H}$. This can be used to further increase the order of the spatial discretization: A scheme of arbitrarily high order guaranteeing bounds on the solution can be constructed with DG or WENO methods using bound-preserving limiters. Choosing such a method as underlying scheme yields a realizable moment update of arbitrarily high order. Furthermore, since bound-preserving methods exist for systems, this strategy can also be used to construct realizability preserving methods if the original problem is a system of equations. It is however important to point out the need to control the non-conservative error, which arises when replacing moments.*

2.3.2 Second-order time integration

For time integration we use strong stability-preserving (SSP) methods. These are the standard choice for hyperbolic equations and allow us to build on our analysis of forward Euler steps, since SSP methods can be written as convex combinations of forward Euler steps. We rewrite a forward Euler step in the form

$$\hat{\boldsymbol{u}}_j^{n+1} = \hat{\boldsymbol{u}}_j^n + \Delta t L_j(\overline{\Lambda}_{j-2}^n, \overline{\Lambda}_{j-1}^n, \overline{\Lambda}_j^n, \overline{\Lambda}_{j+1}^n, \overline{\Lambda}_{j+2}^n), \tag{2.32}$$

where

$$L_j(\overline{\Lambda}_{j-2}^n, \overline{\Lambda}_{j-1}^n, \overline{\Lambda}_j^n, \overline{\Lambda}_{j+1}^n, \overline{\Lambda}_{j+2}^n) := -\frac{1}{\Delta x}\Bigg(\left\langle f^*(u(\overline{\Lambda}_{j+1/2}^-), u(\overline{\Lambda}_{j+1/2}^+))\boldsymbol{\varphi} \right\rangle - \left\langle f^*(u(\overline{\Lambda}_{j-1/2}^-), u(\overline{\Lambda}_{j-1/2}^+))\boldsymbol{\varphi} \right\rangle \Bigg).$$

In particular, we use multistep SSP methods [121]. With multistep methods, we are able to re-use the evaluations of L_j from previous time steps, in contrast to single-step (i.e., multistage Runge–Kutta) methods, which require multiple evaluations of L_j for each time step. The time update for a general multistep SSP method has the form

$$\hat{\boldsymbol{u}}_j^{n+1} = \sum_{i=1}^{m} \alpha_i \hat{\boldsymbol{u}}_j^{n+1-i} + \Delta t \beta_i L_j(\overline{\Lambda}_{j-2}^{n+1-i}, \overline{\Lambda}_{j-1}^{n+1-i}, \overline{\Lambda}_j^{n+1-i}, \overline{\Lambda}_{j+1}^{n+1-i}, \overline{\Lambda}_{j+2}^{n+1-i}),$$

where m is the number of past steps used to compute the $(n+1)$-th time step. When a forward Euler step remains realizable under time step Δt_{FE}, then the multistep SSP method remains realizable under time step $c\Delta t_{\mathrm{FE}}$, where

$$c := \min_{\{i:\beta_i>0\}} \frac{\alpha_i}{|\beta_i|}.$$

We use the four-step second-order method found in [47]:

$$\boldsymbol{\alpha} = \left(\frac{8}{9}, 0, 0, \frac{1}{9}\right)^T, \quad \boldsymbol{\beta} = \left(\frac{4}{3}, 0, 0, 0\right)^T, \quad c = \frac{2}{3}. \tag{2.33}$$

With this multistep SSP method, the new CFL condition is given by

$$\max_{u\in[u_-,u_+]} |f'(u)| \frac{\Delta t}{\Delta x} \leq \frac{1}{3}. \tag{2.34}$$

2.3.3 Kinetic flux

Note that the previous discussion of realizability relies on the choice of the kinetic flux (2.5). The kinetic flux choice leads to a scheme, which can be interpreted as a nodal discretization of the moment system (which eventually yields an intrusive method). Besides yielding a realizable scheme, such a nodal intrusive scheme has several desirable properties: By simply taking moments of a given numerical flux for the deterministic problem, the method can easily be applied to various physical problems whenever an implementation of the original numerical flux $f^* = f^*(u_\ell, u_r)$ is available. Intrusive numerical methods which compute arising integrals analytically and therefore directly depend on the moments (i.e. they do not necessitate the evaluation of the gPC expansion on quadrature points) can be constructed by performing a gPC expansion on the system flux directly [24]. Examples can be found in [60, 59, 137, 29] for the computation of numerical fluxes and sources. While the analytic computation of arising integrals is not always more efficient [42, Section 6], it can also complicate recycling a deterministic solver. In the following, we briefly discuss the number of operations needed when precomputing integrals versus the use of a kinetic flux for Burgers' equation when using SG (i.e. we use the quadratic entropy $s(u) = \frac{u^2}{2}$). Again, the scalar random variable ξ is uniformly distributed in the interval $[-1, 1]$, hence the gPC basis functions $\boldsymbol{\varphi} = (\varphi_0, \cdots, \varphi_N)^T$ are the Legendre polynomials. Choosing the SG ansatz (1.84) and testing with the gPC basis functions yields the SG moment system

$$\partial_t \hat{u}_i + \partial_x \frac{1}{2} \sum_{n,m=0}^{N} \hat{u}_n \hat{u}_m \langle \varphi_n \varphi_m \varphi_i \rangle = 0.$$

Defining the matrices $\boldsymbol{C}_i := \langle \boldsymbol{\varphi}\boldsymbol{\varphi}^T \varphi_i \rangle \in \mathbb{R}^{M\times M}$ gives

$$\partial_t \hat{\boldsymbol{u}} + \partial_x \boldsymbol{F}(\hat{\boldsymbol{u}}) = \boldsymbol{0}$$

with $F_i(\hat{\boldsymbol{u}}) = \frac{1}{2}\hat{\boldsymbol{u}}^T \boldsymbol{C}_i \hat{\boldsymbol{u}}$. Note that $\boldsymbol{C}_i$ can be computed analytically, hence choosing a Lax-Friedrichs flux

$$F_i^{(LF)}(\hat{\boldsymbol{u}}_\ell, \hat{\boldsymbol{u}}_r) = \frac{1}{4}\left(\hat{\boldsymbol{u}}_\ell^T \boldsymbol{C}_i \hat{\boldsymbol{u}}_\ell + \hat{\boldsymbol{u}}_r^T \boldsymbol{C}_i \hat{\boldsymbol{u}}_r\right) - \frac{\Delta x}{2\Delta t}(\hat{\boldsymbol{u}}_r - \hat{\boldsymbol{u}}_\ell)_i \tag{2.35}$$

requires no integral evaluations. Recall, that the numerical flux choice made in this work gives

$$\boldsymbol{F}^*(\hat{\boldsymbol{u}}_\ell, \hat{\boldsymbol{u}}_r) = \sum_{k=1}^{Q} w_k f^*(\mathcal{U}(\hat{\boldsymbol{u}}_\ell; \xi_k), \mathcal{U}(\hat{\boldsymbol{u}}_r; \xi_k)) \boldsymbol{\varphi}(\xi_k) f_\Xi(\xi_k), \tag{2.36}$$

where $\mathcal{U}$ is the SG ansatz (1.84). When the chosen deterministic flux f^* is Lax-Friedrichs, the order of the polynomials inside the sum is $3N = 3(M-1)$. Choosing a Gauss-Lobatto quadrature rule, $Q = \frac{3}{2}M - 1$ quadrature points suffice for an exact computation of the numerical flux. Indeed, with this choice of quadrature points, the numerical fluxes (2.35) and (2.36) are equivalent. Counting the number of operations, one observes that our choice of the numerical flux (2.36) uses $O(M^2)$ operations whereas (2.35) requires $O(M^3)$ operations: When computing and storing the values in a matrix $\boldsymbol{A} \in \mathbb{R}^{Q\times M}$ with entries $a_{ki} = \varphi_i(\xi_k)$ before running the program, the numerical flux (2.36) can be split into two parts. First, we determine the SG solution at all quadrature points, i.e. we compute $\boldsymbol{u}^{(\ell)} := \boldsymbol{A}\hat{\boldsymbol{u}}_\ell$ and $\boldsymbol{u}^{(r)} := \boldsymbol{A}\hat{\boldsymbol{u}}_r$ which requires $O(M \cdot Q)$ operations. These solution values are then used to compute the numerical flux

$$F_i^*(\hat{\boldsymbol{u}}_\ell, \hat{\boldsymbol{u}}_r) = \sum_{k=1}^{Q} w_k f^*(u_k^{(\ell)}, u_k^{(r)}) a_{ki} f_\Xi(\xi_k),$$

which again requires $O(M \cdot Q)$ operations, i.e. the costs are $O(M^2)$. The evaluation of (2.35) however requires $O(M^3)$ operations.

When not using a quadratic entropy in the IPM method or when the physical flux of the deterministic problem is not a polynomial, it is not clear how many quadrature points the numerical quadrature rule requires to guarantee a sufficiently small quadrature error. We will study the approximation properties of IPM with different quadrature orders at a later point in Section 4.5.1.

2.4 Choosing the entropy

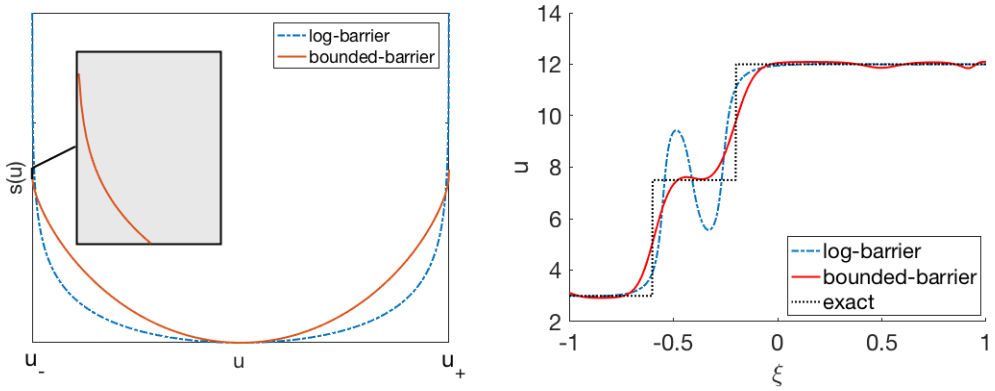
While the log-barrier does the job of enforcing bounds on the oscillations around $\min u_{\mathrm{IC}}$ and $\max u_{\mathrm{IC}}$, it is not the only choice which achieves such bounds. If we look at the form of the entropy ansatz $u(\Lambda) = (s')^{-1}(\Lambda)$, we see that it is sufficient that the derivative s' maps the open interval (u_-, u_+) to the entire real line. I.e., it suffices that

$$\lim_{u \nearrow u_+} s'(u) \to \infty \quad \text{and} \quad \lim_{u \searrow u_-} s'(u) \to -\infty \tag{2.37}$$

to achieve the desired bounds on the entropy ansatz. We can use this to find a new entropy with better properties.

In choosing an entropy, our goals are to satisfy the original maximum principle as closely as possible and to obtain a solution which oscillates as little as possible. The first step is ensured by condition (2.37), as long as we take $\Delta u = u_+ - \max u_{\mathrm{IC}} = \min u_{\mathrm{IC}} - u_-$ as small as possible. In fact, ideally we would like to just choose $\Delta u = 0$.

The log-barrier entropy (2.1) achieves condition (2.37) indirectly: by ruling out values outside of (u_-, u_+) using barriers in s itself. There exist, however, moment vectors for which the ansatz must take on the value $\max u_{\mathrm{IC}}$ or $\min u_{\mathrm{IC}}$ on sets of nonzero measure. The moment vectors of the initial condition $\hat{\boldsymbol{u}}(0,x) = \langle u_{\mathrm{IC}}(x,\cdot)\boldsymbol{\varphi}\rangle$ take on such values when, for example, u_{IC} attains its maximum or minimum (over all $x \in D$ and $\xi \in \Theta$) at some point in space with certainty (i.e., constant in ξ). These moments lie on the boundary of the set of realizability when $\Delta u = 0$, but since the realizable set is open, here the optimization problem has no solution. In the limit



(A) (B)

FIGURE 2.1: (A) Comparison of entropies and (B) resulting approximation with $\Delta u = 0.1, N = 10$ for a given function (exact).

as a sequence of moment vectors approaches such a nonrealizable moment, the corresponding limit of entropy ansätze does converge, but the entropy value $\langle s(\mathcal{U}(\hat{\boldsymbol{u}}))\rangle$ goes to infinity. In this sense, the log-barrier entropy does not always recover the certain case gracefully.

But we can fulfill condition (2.37) without forcing s itself to take infinite values. An entropy which achieves this is

$$s(u) = (u - u_-)\ln(u - u_-) + (u_+ - u)\ln(u_+ - u). \tag{2.38}$$

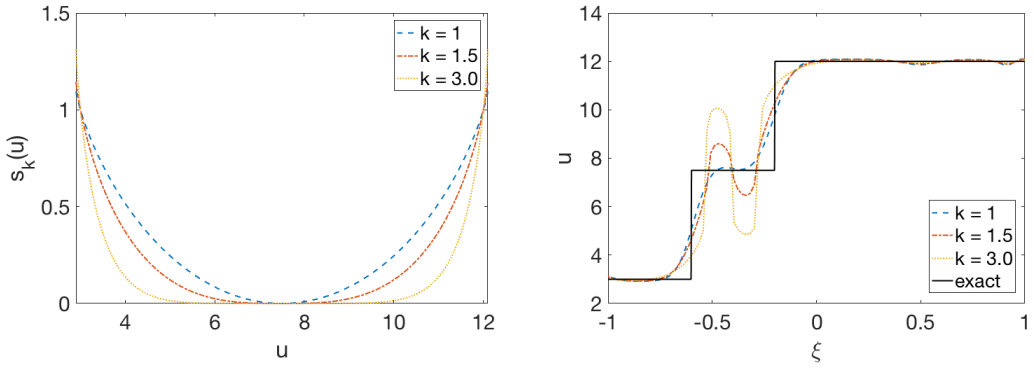
Note that a similar version of this entropy has also been used in [111]. With $u_- = 0$ and $u_+ = 1$, this is the entropy for particles with Fermi–Dirac statistics. In the following, this entropy is called bounded-barrier (BB) entropy. It satisfies condition (2.37) but is finite on the interval $[u_-, u_+]$. We compare the two entropy functions in Figure 2.1A.

In Figure 2.1B, we study the approximation properties of the log-barrier and BB entropies when reconstructing a given function, which solely depends on the random variable ξ. Here, an interesting difference sticks out: The BB entropy not only gives a much better solution, but in contrast to the solution using the log-barrier entropy it is not oscillatory around the value $u = (u_+ + u_-)/2 =: u_M$. Further consideration of the shapes of the entropy functions offers a possible explanation. In Figure 2.1A we notice that the log-barrier entropy is much flatter than the BB entropy around their minimum at $u = u_M$. Thus the log-barrier entropy does not distinguish among these values very well, and as a result the oscillations in its entropy ansatz seen in Figure 2.1B are allowed because they have only a small effect on the value of the entropy. Correspondingly, values near the boundaries of the domain u_- and u_+ are strongly punished by the log-barrier entropy; this is in contrast to the bounded-barrier entropy, which simply takes finite values even at the end points.

We tested this hypothesis by modifying the values of the slope around u_M using the family of entropies

$$s_k(u) = \left(\frac{s(u) - s\left(\frac{1}{2}(u_- + u_+)\right)}{s(u_{\max}) - s\left(\frac{1}{2}(u_- + u_+)\right)}\right)^k,$$

where s is the bounded-barrier entropy. As we show in Figure 2.2A, the higher k is, the flatter the entropy is around u_M, so for higher values of k, we expect the entropy



(A) (B)

FIGURE 2.2: (A) Family of entropies and (B) corresponding reconstruction for $\Delta u = 0.1, N = 10$.

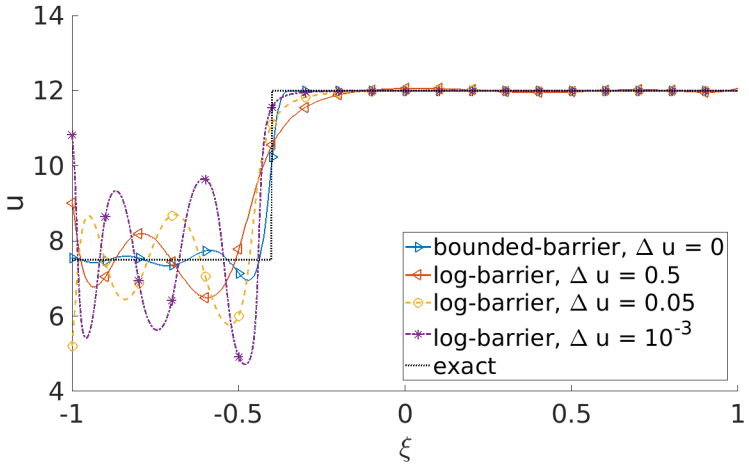


FIGURE 2.3: Approximation behavior for different values of Δu with $N = 10$.

ansatz to be more oscillatory. This is then exactly what we observe in Figure 2.2B.

Another difference between the log- and bounded-barrier entropies is the dependence of the oscillations on the choice of Δu. In numerical experiments, we noticed that with the log-barrier entropy, smaller values of Δu are disadvantageous because the solutions are more oscillatory for smaller values of Δu. We show an example of this behavior in Figure 2.3. Here, we reconstruct a shock from u_M to u_{max}. The bounded-barrier entropy with $\Delta u = 0$ again gives the best result. As we will see in the numerical results in the next section, the bounded-barrier entropy's more gentle behavior near the bounding values u_- and u_+ allows us to choose $\Delta u = 0$ in all our numerical tests, thus exactly enforcing the original maximum principle.

2.4.1 Connection to Kružkov's entropy

In the following, we will use theoretical findings [12] for the Fermi-Dirac entropy to link the bounded-barrier entropy to the Kružkov entropy (1.24), which is an important concept of hyperbolic equations. As already stated, the Kružkov entropy is used to derive several important properties of entropy solutions, for example L^1 stability or total variation diminishing properties. In the following, we show that the relative bounded barrier entropy (times a constant independent of t) bounds the

squared Kružkov entropy from above. If the relative entropy is dissipated, we obtain a bound on the Kružkov entropy. The result can be used to show control over oscillations at intermediate solution values $u \in (u_-, u_+)$.

We will start by looking at the relative entropy

$$S(u|w) = \int_D \left\langle (u-u_-)\ln\left(\frac{u-u_-}{w-u_-}\right) + (u_+-u)\ln\left(\frac{u_+-u}{u_+-w}\right)\right\rangle dx. \tag{2.39}$$

Here, u and w are two functions which depend on t, x and ξ. The dependence on the phase space is omitted for better readability. We will later choose w such that the relative entropy is dissipated and u will be the IPM solution. An estimate which is similar to the work presented in [12] will show that the relative entropy will bound the squared L^1 norm from above:

Theorem 12. *For $u \in [u_-, u_+]$ we have that*

$$S(u|w) \geq c\|u-w\|^2_{L^1(D;\Omega)}.$$

The constant c is given by

$$c := \left(\frac{1}{2u_+L - \frac{4}{3}m_w - \frac{2}{3}m_u} + \frac{1}{\frac{4}{3}m_w + \frac{2}{3}m_u - 2u_-L}\right)$$

where m_u is the mass of u, meaning that $m_u := \int_D \langle u\rangle dx$ and L is the area of the spatial domain.

Proof. As in [12], we use

$$\left(\frac{4}{3}+\frac{2}{3}\theta\right)(\theta\ln\theta - \theta + 1) - (\theta-1)^2 \geq 0$$

for $\theta \in [0,\infty)$. This inequality is easy to verify by showing that the function on the left hand side is convex and its minimum at $\theta = 1$ is zero. Using

$$\theta = \frac{u-u_-}{w-u_-}$$

gives

$$\left(\frac{4}{3}w + \frac{2}{3}u - 2u_-\right)\left((u-u_-)\ln\frac{u-u_-}{w-u_-} - u + w\right) - (u-w)^2 \geq 0.$$

Taking the square root and integrating over x and ξ gives

$$\begin{aligned}
\int_D \langle |u-w|\rangle dx &\leq \int_D \left\langle \sqrt{\frac{4}{3}w+\frac{2}{3}u-2u_-}\sqrt{(u-u_-)\ln\frac{u-u_-}{w-u_-} - u + w}\right\rangle dx \\
&\overset{CS}{\leq} \sqrt{\int_D \left\langle \frac{4}{3}w+\frac{2}{3}u-2u_-\right\rangle dx}\sqrt{\int_D \left\langle (u-u_-)\ln\frac{u-u_-}{w-u_-} - u + w\right\rangle dx} \\
&= \sqrt{\frac{4}{3}m_w + \frac{2}{3}m_u - 2u_-L}\sqrt{\int_D \left\langle (u-u_-)\ln\frac{u-u_-}{w-u_-}\right\rangle dx - m_u + m_w}
\end{aligned}$$

where we used the Cauchy-Schwartz inequality. Squaring both sides gives

$$\frac{1}{\frac{4}{3}m_w + \frac{2}{3}m_u - 2u_- L}\|u-w\|_{L^1}^2 \leq \int_D \left\langle (u-u_-)\ln\frac{u-u_-}{w-u_-}\right\rangle dx - m_u + m_w. \quad (2.40)$$

Analogously with

$$\theta = \frac{u_+ - u}{u_+ - w}$$

we can show that

$$\frac{1}{2u_+ L - \frac{4}{3}m_w - \frac{2}{3}m_u}\|u-w\|_{L^1}^2 \leq \int_D \left\langle (u_+-u)\ln\frac{u_+-u}{u_+-w}\right\rangle dx + m_u - m_w. \quad (2.41)$$

Adding the two terms (2.40) and (2.41) gives

$$c\|u-w\|_{L^1}^2 \leq \int_D \left\langle (u_+-u)\ln\frac{u_+-u}{u_+-w} + (u-u_-)\ln\frac{u-u_-}{w-u_-}\right\rangle dx$$

where $c := \left(\frac{1}{2u_+L-\frac{4}{3}m_w-\frac{2}{3}m_u} + \frac{1}{\frac{4}{3}m_w+\frac{2}{3}m_u-2u_-L}\right)$, which completes the proof. □

In the following we choose u to be the IPM solution. Choosing a constant function w will ensure dissipation of the relative entropy as

$$\begin{aligned}\frac{d}{dt}S(u|w) &= \frac{d}{dt}\left(S(t) - \int_D \langle (u-u_-)\ln(w-u_-)\rangle dx - \int_D \langle (u_+-u)\ln(u_+-w)\rangle dx\right)\\ &= \frac{d}{dt}S(t) - \underbrace{\frac{d}{dt}\int_D \langle u\rangle\, dx}_{=0}\cdot(\ln(w-u_-) - \ln(u_+-w)) = \frac{d}{dt}S(t) \leq 0.\end{aligned}$$

Hence, we can ensure an upper bound of the squared Kružkov entropy.

Furthermore, we can use this result to control oscillations: The constant value of w can be chosen to be the intermediate state, where we expect to see oscillations. Contrary to $\|u\|_{L^1}^2$, the quantity $\|u-w\|_{L^1}^2$ will capture oscillations around this state. Consequently, the oscillations are bounded from above by the relative entropy.

Remark 13. *For scalar hyperbolic equations, every convex function is an entropy. This is not the case for systems of equations, meaning that the bounded-barrier entropy cannot be used in such a setting. However, the study shows that given a set of admissible entropies for such a system, one should choose an entropy which sufficiently distinguishes between different solution values (provided that such an entropy exists).*

2.5 Results

In the following, we first compare the log-barrier and the bounded-barrier entropy in different test cases before turning to investigating the effectiveness of the two strategies to impose realizability.

2.5.1 Comparing different entropies

We start by comparing results when making use of the log- and bounded-barrier entropies. Following [112], we solve the uncertain Burgers' equation, which has

been defined in (1.80) by

$$\partial_t u(t,x,\xi) + \partial_x \frac{u(t,x,\xi)^2}{2} = 0,$$
$$u(t=0,x,\xi) = u_{\mathrm{IC}}(x,\xi),$$

with the first-order method in Algorithm 2. As in [112], we choose the random initial condition

$$u_{\mathrm{IC}}(x,\xi) := \begin{cases} u_L, & \text{if } x < x_0 + \sigma\xi \\ u_L + \frac{u_R - u_L}{x_0 - x_1}(x_0 + \sigma\xi - x), & \text{if } x \in [x_0 + \sigma\xi, x_1 + \sigma\xi] \\ u_R, & \text{else} \end{cases} \tag{2.42}$$

which is a forming shock with a linear connection from x_0 to x_1. In our case, ξ is uniformly distributed on the interval $[-1,1]$. Note that this test case has been described and investigated for the exact solution in Section 1.4.1. Due to the fact that we recalculate moments to ensure realizability, we can use the original CFL condition (2.8). We use the following parameter values:

$[a,b] = [0,3]$	range of spatial domain
$N_x = 160$	number of spatial cells
$t_{end} = 0.15$	end time
$x_0 = 0.5, x_1 = 1.5, u_L = 12, u_R = 3, \sigma = 0.2$	parameters of initial condition (2.42)
$N+1 = 5$	number of moments
$\tau = 10^{-7}$	gradient tolerance (2.7)
$\Delta u \in \{0, 0.001, 0.5\}$	distance u_{IC} to IPM bounds

Additionally, we computed all integrals in ξ using a forty-point Gauss-Legendre quadrature.

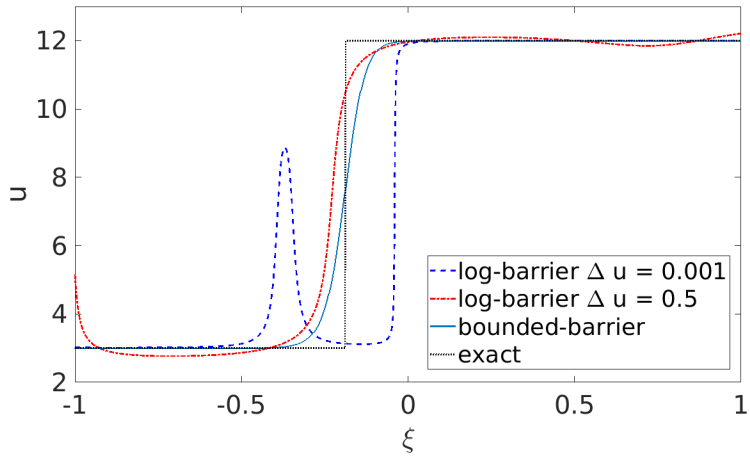
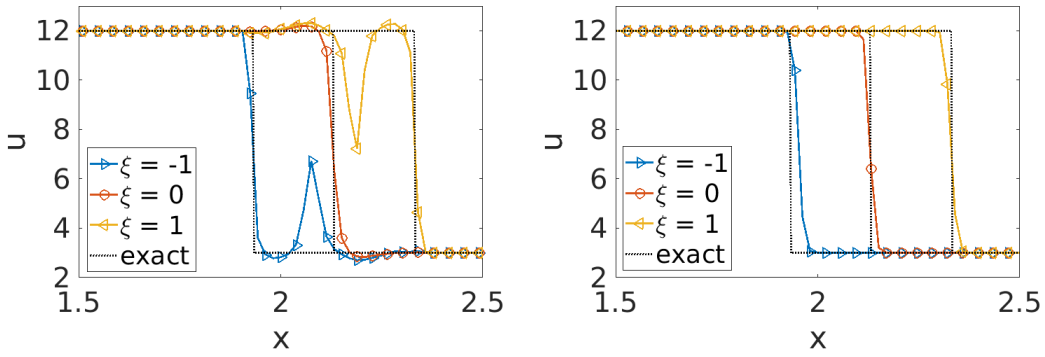


FIGURE 2.4: Solutions for log-barrier and bounded-barrier entropies at fixed spatial position x and time $t_{end} = 0.15$.

Since the log-barrier entropy is infinite at u_+ and u_-, we need to choose $\Delta u > 0$. We choose $\Delta u = 0.5$ as in [112] as well as $\Delta u = 0.001$ to demonstrate the effects when the solutions lie close to the minimal and maximal value of the exact solution. Note that the maximal velocity of the equation is $u_+ = u_L + \Delta u$, so consequently the



(A) log-barrier entropy, $\Delta u = 0.5$. (B) bounded-barrier entropy, $\Delta u = 0$.

FIGURE 2.5: Solution for different entropies evaluated at $\xi \in \{-1, 0, 1\}$ at time $t_{end} = 0.15$.

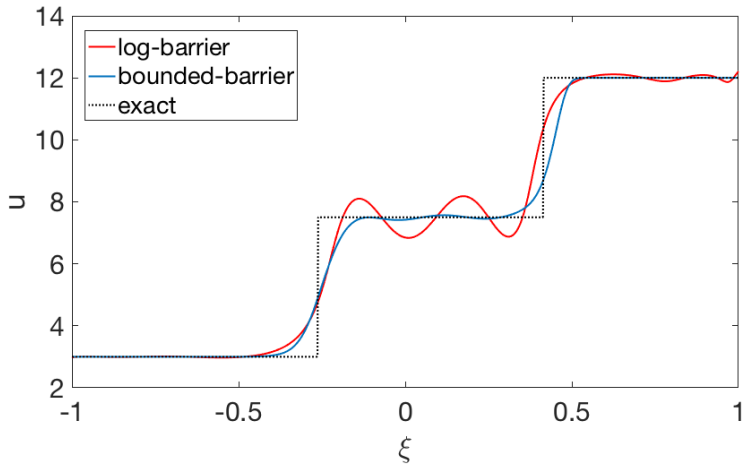


FIGURE 2.6: Solutions for log-barrier and bounded-barrier entropies at $x^* = 2.1$ time $t_{end} = 0.15$.

CFL condition of the deterministic problem (where velocities are bounded by u_L) cannot be used. The bounded-barrier entropy shows good approximation results for small values of Δu, so we set this parameter to zero, allowing the use of the deterministic CFL condition. Plotting the solutions at fixed values for ξ in Figure 2.5 shows the expected poor approximation behavior of the log-barrier entropy for small values of Δu. The choice $\Delta u = 0.5$ leads to over- and undershoots when using the log-barrier entropy, whereas the bounded-barrier entropy nicely approximates the solution. Furthermore, the solution obtained with the bounded-barrier entropy fulfills the original maximum principle. Looking at the dependency on ξ for a fixed spatial cell in Figure 2.4, one observes that the log-barrier entropy has oscillations whereas the bounded-barrier entropy gives a nonoscillatory solution.

Let us now turn to a new initial condition for the uncertain Burgers' equation in order to investigate the oscillations arising at a noncritical state $u_M = \frac{1}{2}(u_L + u_R)$:

$$u_{\mathrm{IC}}(x,\xi) := \begin{cases} u_L, & \text{if } x \leq x_0 + \sigma\xi \\ u_L + (u_M - u_L)\cdot \frac{x_0+\sigma\xi - x}{x_0 - x_1}, & \text{if } x \in (x_0+\sigma\xi, x_1+\sigma\xi] \\ u_M, & \text{if } x \in (x_1+\sigma\xi, x_2+\sigma\xi] \\ u_M + (u_R - u_M)\cdot \frac{x_3+\sigma\xi - x}{x_3 - x_2}, & \text{if } x \in (x_2+\sigma\xi, x_3+\sigma\xi] \\ u_R, & \text{if } x > x_3 + \sigma\xi, \end{cases} \tag{2.43}$$

This initial condition describes two forming shocks that connect the three states u_L, u_M, and u_R. All parameters which have been modified can be found in the following table:

$t_{end} = 0.04$	end time
$x_0 = 0.8, x_1 = 0.98, x_2 = 1.32, x_3 = 1.5, \sigma = 0.5$	parameters of initial condition (2.43)
$N+1 = 16$	number of moments

The results for this problem can be seen in Figure 2.6. One observes that the solution using the log-barrier entropy is oscillatory, whereas with the bounded-barrier entropy the solution shows only small oscillations. While the IPM scheme with the bounded-barrier entropy fulfills the maximum principle, the solution of the log-barrier entropy has over- and undershoots as large as Δu.

2.5.2 Comparison of entropies in two-dimensional random space

To compare both entropies in a two-dimensional random domain (i.e., $p = 2$), the initial condition of the Burgers' test case is changed to

$$u_{\mathrm{IC}}(x) := \begin{cases} u_L + \sigma_0\xi_0, & \text{if } x < x_0, \\ u_M + \sigma_1\xi_1, & \text{if } x \in [x_0, x_1], \\ u_R, & \text{else}, \end{cases} \tag{2.44}$$

where ξ_0 and ξ_1 are both uniformly distributed in $[-1,1]$. This test case represents an uncertain multiple-shock flow, which is studied in compressible fluid mechanics, see [112]. Realizability is again preserved by recalculating moments, meaning that the original CFL condition (2.8) can be used. As in [112], a tensorized Clenshaw-Curtis quadrature rule of level 3 and an increased number of $N_x = 6000$ spatial grid points is used. In contrast to the other test cases, we need to choose a fine resolution of the spatial grid to minimize the effects of numerical diffusion, which significantly affects the solution in this test case.

$[a,b] = [0,1]$	range of spatial domain
$N_x = 6000$	number of spatial cells
$t_{end} = 0.01115$	end time
$x_0 = 0.3, x_1 = 1.6, \sigma_0 = 0.2, \sigma_1 = 0.2,$	parameters of initial condition (2.44)
$u_L = 12, u_M = 6, u_R = 1$	
$N+1 = 5$	number of moments

The results are given in Figure 2.7. IPM again fulfills the maximum principle when the bounded-barrier entropy is used. The solution has only small oscillations

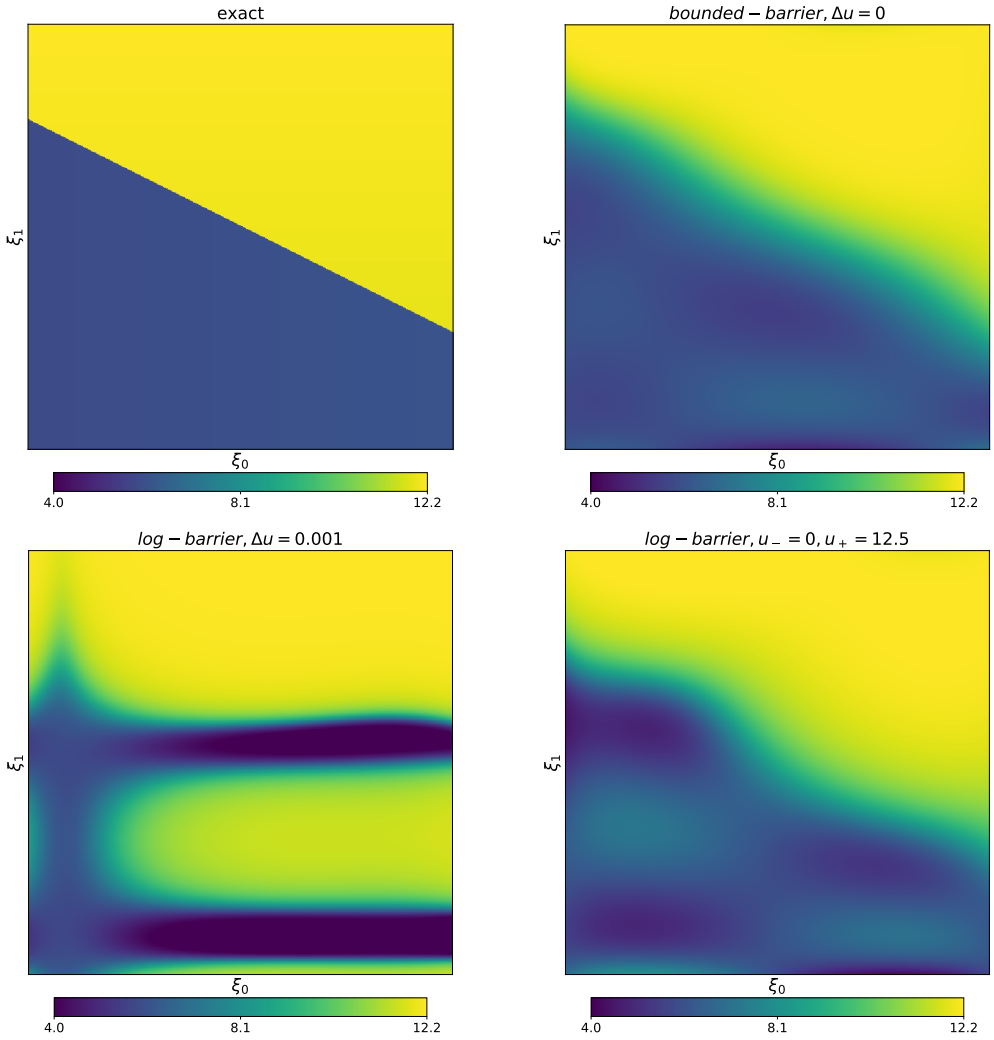


FIGURE 2.7: Solution at $x^* = 0.4$ and time $t_{end} = 0.01115$ with different entropies.

around the intermediate state u_M and shows good agreement with the exact solution. When trying to approach a maximum principle by choosing a small value of Δu with the log-barrier entropy, the solution starts to oscillate heavily at the intermediate state. The solution resembles the one-dimensional result for a small value of Δu depicted in Figure 2.4. Choosing the IPM bounds further away from the exact solution bounds (as in [112]), we obtain a more accurate solution. However the maximum principle is not fulfilled since the solution takes on values bigger than 12.34 (off the color scale) while showing oscillations at the intermediate state. This is also in agreement with the one-dimensional results shown before, where the maximum principle is violated by the log-barrier entropy.

2.5.3 Convergence of different schemes

Due to its advantages compared to the log-barrier entropy, the following results have been obtained using the bounded-barrier entropy with $\Delta u = 0$. To investigate the convergence properties of the proposed first- and second-order schemes, we look at

N_x	10	20	30	40	50	100	200	500	1000	2000	3000	4000	5000
runtime	2.2E-4	0.04	0.06	0.17	0.25	1.23	4.76	30.58	125.58	518.36	1146.34	2043.75	3221.47
error	5.2E-1	2.2E-1	1.4E-1	1.0E-1	8.3E-2	4.1E-2	2.0E-2	8.0E-3	4.0E-3	2.0E-3	1.3E-3	1.0E-3	8.0E-4

TABLE 2.2: Runtime (in seconds) and L^1 error (according to (2.45) using the first component of $\boldsymbol{e}$) for first order scheme as depicted in Figure 2.8.

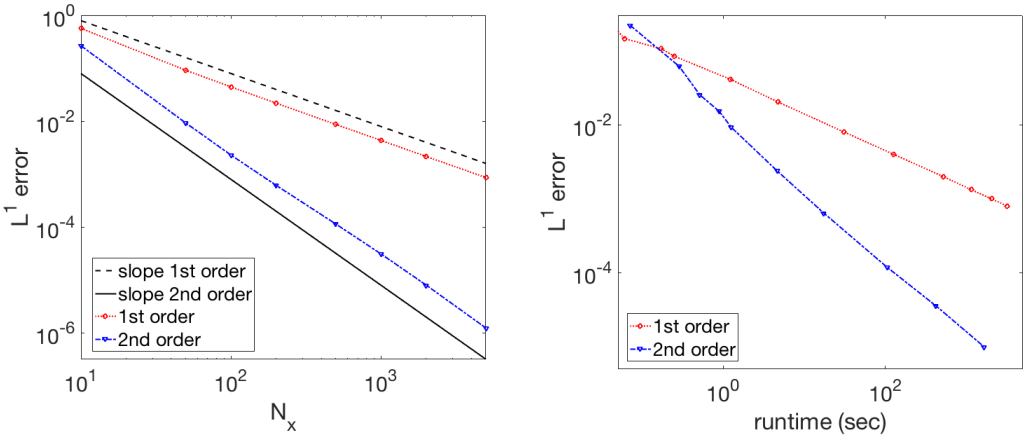
the advection equation with uncertain initial data

$$\begin{aligned}\partial_t u(t,x,\xi) + \partial_x u(t,x,\xi) &= 0,\\ u(t=0,x,\xi) &= \sin(x + 0.05\pi\xi),\end{aligned}$$

where $x \in [0,2]$ and $t_{end} = 0.1$. We use periodic boundary conditions at the boundaries of the spatial domain. The number of moments we calculate is 3. We study the L^1 error of the expected value for different numbers of spatial discretization points. Let $\hat{\boldsymbol{u}}_h$ denote a numerical solution. For first-order methods, it is constant across space in each spatial cell, and for second-order methods, it is defined according to the linear reconstructions given in Section 2.3. Then we compute the L^1 error for each moment component by

$$\boldsymbol{e} := \int_0^2 |\hat{\boldsymbol{u}}_h(t_{end},x) - \hat{\boldsymbol{u}}(t_{end},x)|\,dx, \tag{2.45}$$

where $\hat{\boldsymbol{u}}(t_{end},x)$ is the exact solution to the system of IPM moment equations (1.108) at the final time t_{end}, and the absolute value and integral are taken component-wise. In the following convergence results, we plot only the results for the zeroth component of $\boldsymbol{e}$. The resulting convergence plot is given in Figure 2.8A.



(A) (B)

FIGURE 2.8: (A) Convergence of different IPM discretizations and (B) efficiency when using first- and second-order methods. The L^1 error has been computed according to (2.45) using the first component of $\boldsymbol{e}$. For numerical values, see Tables 2.2 and 2.3.

Both methods recalculate moments with the inaccurate dual states, meaning that in order to preserve the expected convergence rate $p \in \{1,2\}$, the stopping criterion of the optimization method needs to be set to $\tau = \Delta x^{p+1}$. For the time discretization of the second-order scheme, the four-step SSP scheme (2.33) has been used. Heun's

N_x	10	20	30	40	50	100	200	500	1000	2000
runtime	0.07	0.28	0.51	0.88	1.26	4.62	17.32	105.19	419.92	1663.26
error	2.2E-1	6.2E-2	2.5E-2	1.5E-2	9.2E-3	2.3E-3	6.4E-4	1.2E-4	3.6E-5	1.0E-5

TABLE 2.3: Runtime (in seconds) and L^1 error (according to (2.45) using the first component of $\boldsymbol{e}$) for second order scheme as depicted in Figure 2.8.

method is used to calculate the first three time steps. That the different schemes show the expected convergence.

The efficiency of the two methods shown in Figure 2.8B demonstrates that the second-order scheme reaches most levels of accuracy with less computing time than the first-order scheme.

2.5.4 Comparison of strategies to preserve realizability

Two strategies to ensure realizability have been presented in Section 2.2.2 and Section 2.2.3, namely using a modified CFL condition or modifying moments. To compare these two strategies, we look at the uncertain advection equation given by

$$\begin{aligned} \partial_t u(t,x,\xi) + a(\xi)\partial_x u(t,x,\xi) &= 0, \\ u(t=0,x) &= u_{\mathrm{IC}}(x). \end{aligned}$$

We choose $a(\xi) := 11 + \xi$, where ξ is uniformly distributed on $[-1,1]$, so the velocity is uniformly distributed in the interval $[10,12]$. What is interesting about this equation is that the velocity of the system is not known, which means that our CFL condition adds artificial viscosity to smaller velocities, while high velocities are well resolved. We use the deterministic initial condition

$$u_{\mathrm{IC}}(x) := \begin{cases} u_L, & \text{if } x < x_0, \\ u_L + \frac{u_R - u_L}{x_0 - x_1}(x_0 - x), & \text{if } x \in [x_0, x_1], \\ u_R, & \text{else.} \end{cases} \tag{2.46}$$

Parameters of the calculation can be found in the following table:

$N_x = 80$	number of spatial cells
$t_{end} = 0.19$	end time
$x_0 = 0.5, x_1 = 0.55, u_L = 12, u_R = 3$	parameters of initial condition (2.46)
$N + 1 = 10$	number of moments
$\gamma \in \{1.5, 1.1, 1 + 10^{-7}\}, \zeta = 5$	CFL modification
$\zeta = 5$	safety factor in estimation of $\hat{v}$ (2.17)
$\Delta u \in \{0, 10^{-7}\}$	distance u_{IC} to IPM bounds

When modifying moments, we perform the computation using a first-order scheme as well as with second-order spatial reconstructions using the minmod limiter. Since the second-order time discretization adds artificial viscosity without improving the accuracy, we use the explicit Euler method. We use $\Delta u = 10^{-7}$ so that we can achieve the stopping criterion (2.16) on every quadrature point. Figure 2.9 shows the solution at a fixed position x^*. Using Algorithm 2 to ensure realizability allows the use of the deterministic CFL condition (2.8). We compare this solution to those obtained with a modified CFL conditions according to Section 2.2.2. As expected, modifying the CFL condition by $\gamma = 1.5$ leads to a heavily smeared-out solution. However, we found that for this problem, it was possible to set γ as small as $1 + 10^{-7}$. The

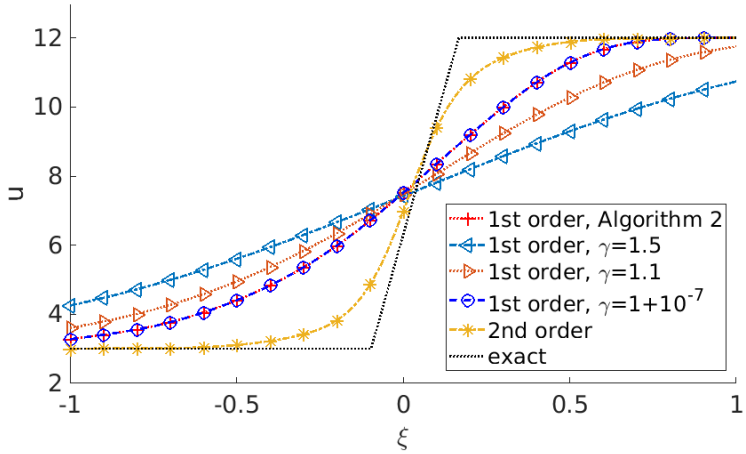


FIGURE 2.9: Solutions at $x^* = 2.6$ and $t_{end} = 0.19$. The 1st order results are computed without limiters, the 2nd order method uses a minmod limiter.

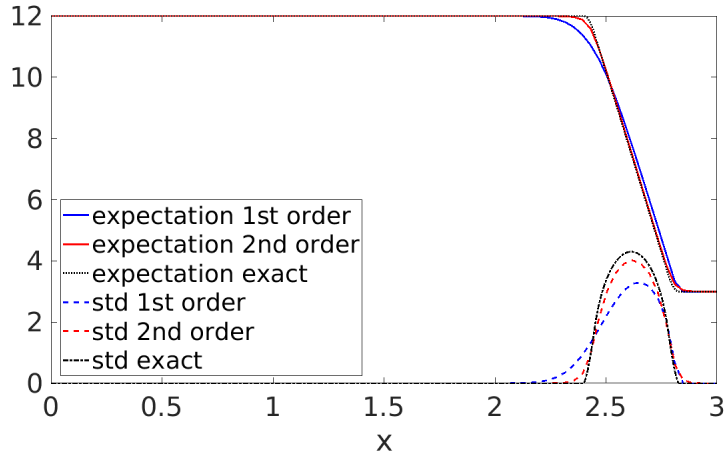
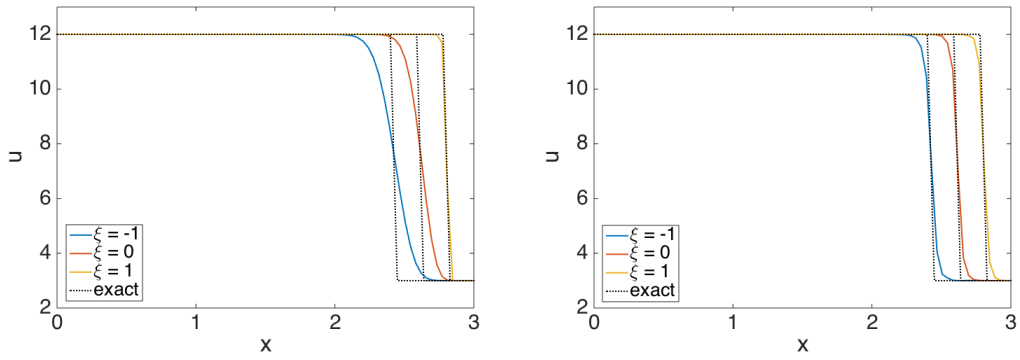


FIGURE 2.10: Expected value and standard deviation (std) at $t_{end} = 0.19$ with (2nd order) and without (1st order) limiters.

solution calculated with this value of γ is essentially identical to the solution using Algorithm 2; they differ only on the order of 10^{-3} in the L^∞ norm. One can conclude that both realizability-preserving strategies for first-order methods are satisfactory.

Figure 2.9 also shows that the second-order spatial reconstructions give much better results. This improvement is also seen in the expected value and standard deviation in Figure 2.10. The standard deviation is particularly improved by going to second-order.

To underline the effects of artificial viscosity, we plot the solution when recalculating moments for first- and second-order spatial reconstructions for $\xi \in \{-1, 0, 1\}$. Figure 2.11A shows that the solution is well resolved if $\xi = 1$. In the case of $\xi = -1$, the solution is smeared out, since the CFL condition does not allow the scheme to sharply capture shocks. In Figure 2.11B, we see that this effect is smaller when using second-order reconstructions.



(A) First order.

(B) Second order.

FIGURE 2.11: Solution evaluated at $\xi \in \{-1, 0, 1\}$ with (2nd order) and without (1st order) limiters for time $t_{end} = 0.19$.

Chapter 3

A nodal IPM method with adaptivity

In this chapter, we derive a nodal intrusive polynomial moment (nIPM) closure for uncertainty quantification when using scalar hyperbolic equations. By performing the IPM reconstruction on the nodal level, our method allows an efficient solution method for the IPM closure (1.120) while providing a convenient transition between nodal and modal IPM time updates. This transition can be used to combine benefits of both approaches, i.e. saving computational costs while achieving a satisfactory solution approximation. The main idea of the nodal closure is to study the IPM system (1.108) and the corresponding closure problem (1.120) after discretizing arising integral terms in the moment system by a quadrature rule. When choosing an appropriate set of basis vectors to represent the solution at the quadrature points, the number of unknowns in the resulting optimization problem can be reduced.

The chapter is structured as follows: We investigate the numerical approximation of integrals showing up in the moment system in Section 3.1. This investigation shows a well-known connection of intrusive and collocation methods. In Section 3.2, we introduce the nodal IPM approach. The implementation of the closure as well as the quadrature-refinement is discussed in Section 3.4. Section 3.3 discusses the connection between nIPM and IPM.

3.1 Transition from Stochastic Collocation to stochastic-Galerkin

We first investigate the moment system (1.118) discretized in the random variable ξ. This gives a connection between collocation and intrusive methods, which we use to discuss potential error sources of collocation methods. Furthermore, we use this link to later adaptively switch between a collocation and an intrusive method. Note, that this connection is well-known in various areas such as Discontinuous Galerkin (DG) (see for example [41]) and is reviewed in this section for the sake of completeness.

Let us start by looking at the moment system (1.118), when making use of the definition of the moments (1.85), which gives

$$\partial_t \langle u(t,x,\xi)\varphi_i(\xi)\rangle + \partial_x \langle f(u(t,x,\xi))\varphi_i(\xi)\rangle = 0$$

for $i = 0, \cdots, N$. This system describes the exact time evolution of the first $M = N+1$ moments. In order to calculate the time evolution numerically, one needs to discretize the moment system. For now, x and t remain continuous and we only

discretize the integral evaluations with the help of a quadrature rule

$$\langle g(\xi) \rangle \approx \sum_{k=1}^{N_q} w_k g(\xi_k) f_\Xi(\xi_k),$$

where ξ_k is the k^{th} quadrature point with the corresponding quadrature weight w_k. Approximating the terms of the moment system with this quadrature yields

$$\partial_t \sum_{k=1}^{N_q} w_k u(t,x,\xi_k)\varphi_i(\xi_k) f_\Xi(\xi_k) + \partial_x \sum_{k=1}^{N_q} w_k f(u(t,x,\xi_k))\varphi_i(\xi_k) f_\Xi(\xi_k) = 0. \tag{3.1}$$

Now we must choose a suitable number of quadrature points N_q to capture the correct time evolution of the moments. First, we see what happens if N_q equals the number of moments, meaning that $N_q = M$. This appears to be a striking choice, since it means that especially the time evolution equations for the higher moments (for which the corresponding physical fluxes are polynomials of high order) are approximated poorly. Nevertheless, we discuss this case because it yields the Stochastic Collocation solution:

Proposition 2. *Assume a quadrature rule with M quadrature points being exact for polynomials $p \in \mathbb{P}^{2N}$ under the integral density f_Ξ exists, i.e.*

$$\langle p(\xi) \rangle = \sum_{k=1}^{M} w_k p(\xi_k) f_\Xi(\xi_k). \tag{3.2}$$

When making use of this quadrature rule to discretize the exact moment system (1.118), the resulting time evolution of the moments is equivalent to the evolution of the Stochastic Collocation method (1.94).

Proof. Using the quadrature rule (3.2) to discretize the exact moment system (1.118) yields the discrete moment system

$$\partial_t \sum_{k=1}^{M} w_k u(t,x,\xi_k)\varphi_i(\xi_k) f_\Xi(\xi_k) + \partial_x \sum_{k=1}^{M} w_k f(u(t,x,\xi_k))\varphi_i(\xi_k) f_\Xi(\xi_k) = 0. \tag{3.3}$$

Making use of the matrix $\boldsymbol{A} \in \mathbb{R}^{M\times M}$ with $a_{ik} := w_k \varphi_i(\xi_k) f_\Xi(\xi_k)$, the discrete moment system can be rewritten as

$$\sum_{k=1}^{M} a_{ik} \left(\partial_t u(t,x,\xi_k) + \partial_x f(u(t,x,\xi_k))\right) = 0. \tag{3.4}$$

The inverse of $\boldsymbol{A}$ exists and is given by $\boldsymbol{A}^{-1} = \left(\varphi_i(\xi_j)\right)_{i,j=0,\cdots,N}$, since

$$\left(\boldsymbol{A}\boldsymbol{A}^{-1}\right)_{ij} = \sum_{k=1}^{N_q} \varphi_i(\xi_k)\varphi_j(\xi_k) f_\Xi(\xi_k) w_k \overset{(3.2)}{=} \langle \varphi_i(\xi)\varphi_j(\xi) \rangle = \delta_{ij}.$$

Multiplication of the discrete moment system (3.4) with $\boldsymbol{A}^{-1}$ decouples this system leading to

$$\partial_t u(t,x,\xi_k) + \partial_x f(u(t,x,\xi_k)) = 0 \quad \text{for } k = 1,\cdots,N_q, \tag{3.5}$$

which is the Stochastic Collocation method. □

Remark 14. *The extension to multidimensional problems is straight forward when using tensorized grids and maximum degree polynomials. However, this strategy cannot be applied for sparse grids.*

This result shows, that the Stochastic Collocation solution can be seen as the solution of the moment system when making use of an inaccurate quadrature rule. Consequently, especially the time evolution of high-order moments is described poorly due to integration errors when evaluating the physical flux.

Our goal is to improve the time evolution of the moment system in regions with high uncertainty. To gain further accuracy, one needs to add more quadrature points to the discrete moment system (3.1). When defining a rectangular matrix $\boldsymbol{A} \in \mathbb{R}^{M \times N_q}$ with $a_{ik} := w_k \varphi_i(\xi_k) f_\Xi(\xi_k)$, the discrete moment system (3.1) can be rewritten as

$$\sum_{k=1}^{N_q} a_{ik} \left(\partial_t u(t,x,\xi_k) + \partial_x f(u(t,x,\xi_k)) \right) = 0. \tag{3.6}$$

In this case, the M equations of the moment system no longer fully determine the time evolution of the solution u, due to the fact that one needs to know this solution at $N_q > M$ quadrature points. Hence, to obtain a better integral approximation of the flux, we need to feed additional information to the moment equations (i.e. define a closure). In order to actually achieve an improved result compared to collocation, this information should not violate the moment constraint and at the same time fulfill properties of the entropy solution (1.25). Before specifying the choice of the remaining degrees of freedom, we derive a method for choosing the degrees of freedom in $u(t,x,\xi_k)$ for $k = 1, \cdots, N_q$ without violating the moment constraint.

Remark 15. *The inexact integration of the flux function when using collocation methods also arises in DG methods when applying nodal instead of modal schemes: Nodal DG schemes can be seen as collocation methods. They suffer from quadrature errors of the physical fluxes, which can lead to poor results as well as stability issues. A strategy to avoid these problems has been presented in [41], where the flux evaluation uses the solution values belonging to a lower-order polynomial (i.e. high-order coefficients are set to zero), guaranteeing an exact integration. The approach taken in this chapter is different since we intend to keep all moments fixed but add quadrature points to obtain an improved accuracy of integral approximations.*

3.2 Nodal IPM closure approach

The moment constraint computed with the chosen quadrature rule (3.2) can be rewritten as

$$\hat{\boldsymbol{u}}(t,x) = \sum_{k=1}^{N_q} w_k \boldsymbol{\varphi}(\xi_k) f_\Xi(\xi_k) u(t,x,\xi_k) = \boldsymbol{A} \check{\boldsymbol{u}}(t,x), \tag{3.7}$$

where $\check{\boldsymbol{u}}(t,x) := (u(t,x,\xi_1), \cdots, u(t,x,\xi_{N_q}))^T$ collects the nodal solution values, i.e. the solution evaluated at the quadrature points. Following the idea of IPM, the solution vector $\check{\boldsymbol{u}}$ needs to be chosen such that the moment constraint (3.7) is fulfilled. Then, from all solution candidates which fulfill (3.7), one needs to pick the solution candidate which minimizes a chosen entropy.

The main idea of the nodal IPM closure is to find a suitable basis to span the solution evaluated at the quadrature points. For this, we calculate the kernel of $\boldsymbol{A}$,

which is given by

$$\operatorname{kern}(\boldsymbol{A}) = \operatorname{span}(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_{N_q - M}).$$

Choosing the vectors $\boldsymbol{v}_i$ to represent the solution, one needs M additional linearly independent vectors $\boldsymbol{b}_0, \cdots, \boldsymbol{b}_N$ to span the entire solution space $\mathbb{R}^{N_q}$.[1] The solution is now given by

$$\check{\boldsymbol{u}}(t,x) = \sum_{i=1}^{N_q - M} \alpha_i(t,x)\boldsymbol{v}_i + \sum_{i=0}^{N} \beta_i(t,x)\boldsymbol{b}_i, \tag{3.8}$$

hence we shifted the problem of choosing $\check{\boldsymbol{u}} \in \mathbb{R}^{N_q}$ to choosing the coefficient vectors $\boldsymbol{\alpha} \in \mathbb{R}^{N_q - M}$ and $\boldsymbol{\beta} \in \mathbb{R}^M$. Luckily, the coefficient vector $\boldsymbol{\beta}$ is directly given by the moment constraint: Plugging the solution reconstruction (3.8) into the moment constraint (3.7) leads to

$$\boldsymbol{A}\check{\boldsymbol{u}}(t,x) = \sum_{i=0}^{N} \boldsymbol{A}\boldsymbol{b}_i\beta_i(t,x) = \boldsymbol{C}\boldsymbol{\beta}(t,x) \overset{!}{=} \hat{\boldsymbol{u}}(t,x) \tag{3.9}$$

with $c_{ij} = \sum_l a_{il}b_{jl}$, where $b_{jl} := \left(\boldsymbol{b}_j\right)_l$. Hence, the coefficients $\boldsymbol{\beta}$ are directly given by the moment vector $\hat{\boldsymbol{u}}$. The coefficients $\boldsymbol{\alpha}$ do not influence the moments of the solution and can therefore be picked freely. These coefficients resemble the additional information we need to give to the moment system in order to apply an improved quadrature rule. Plugging $\check{\boldsymbol{u}}$ into the discretized moment system (3.1) gives

$$\begin{aligned}
\partial_t \hat{u}_i + \partial_x \sum_{k=1}^{N_q} a_{ik} f(\check{u}_k(t,x)) &= 0, \\
\check{\boldsymbol{u}}(t,x) = \sum_{i=1}^{N_q - M} \alpha_i(t,x)\boldsymbol{v}_i + \sum_{i=0}^{N} \left(\boldsymbol{C}^{-1}\hat{\boldsymbol{u}}\right)_i \boldsymbol{b}_i.&
\end{aligned}$$

It can be seen that we do not have an equation describing the time evolution of α_i. Clearly, the choice of $\boldsymbol{\alpha}$ is important since it influences the time evolution of $\hat{\boldsymbol{u}}$. This is due to the fact, that the vectors $\boldsymbol{v}_i$ span the kernel of the moment constraint, however do not lie in the null-space of the flux term. The degrees of freedom we obtain by adding more quadrature points to the moment system to improve the collocation solution can now easily be picked by choosing values for $\boldsymbol{\alpha}$. Our first choice yields the stochastic-Galerkin solution.

Proposition 3. *Representing the solution by* (3.8) *while choosing the degrees of freedom* $\boldsymbol{\alpha}$ *such that the solution minimizes*

$$\frac{1}{2}\sum_{k=1}^{N_q} w_k \check{u}_k^2 f_\Xi(\xi_k), \tag{3.10}$$

yields the stochastic-Galerkin method.

[1] Since $\boldsymbol{A}$ is a priori known, all basis vectors can be computed before running the program.

Proof. In the following, we assume $(\boldsymbol{b}_j)_k = b_{jk} = \varphi_j(\xi_k)$ for ease of presentation. For more general choices of $\boldsymbol{b}_j$, one can perform a transformation

$$b_{jk} = \sum_{n=0}^{N} \varphi_n(\xi_k) l_{nj},$$

where $(l_{ni})_{n,i} =: \boldsymbol{L} \in \mathbb{R}^{M\times M}$ is a transformation matrix. First, we need to pick $\boldsymbol{\alpha}$ such that we minimize the discrete L^2 norm (3.10), hence with $\tilde{w}_k := w_k f_\Xi(\xi_k)$ one needs to ensure that

$$\partial_{\alpha_j} \left(\frac{1}{2} \sum_k (\check{u}_k)^2 \tilde{w}_k \right) \stackrel{!}{=} 0.$$

Therefore, for $j = 1, \cdots, N_q - M$ we have

$$\sum_{k=1}^{N_q} \left(\sum_{i=1}^{N_q - M} \alpha_i(t,x) v_{ik} + \sum_{i=0}^{N} \beta_i(t,x) b_{ik} \right) \tilde{w}_k v_{jk} \stackrel{!}{=} 0.$$

Since $b_{ik}\tilde{w}_k = a_{ik}$ and $\boldsymbol{v}_j \in \operatorname{kern}(\boldsymbol{A})$, we must have

$$\sum_{k=1}^{N_q} \sum_{i=1}^{N_q - M} \alpha_i(t,x) v_{ik} \tilde{w}_k v_{jk} \stackrel{!}{=} 0$$

which yields $\alpha_i = 0$. It remains to show that when collecting b_{ik} in a matrix $\boldsymbol{B} \in \mathbb{R}^{M\times N_q}$, we have

$$\check{u}_k = (\boldsymbol{B}\boldsymbol{\beta})_k = \left(\boldsymbol{B}\boldsymbol{C}^{-1}\hat{\boldsymbol{u}} \right)_k \stackrel{!}{=} \boldsymbol{\varphi}(\xi_k)^T \hat{\boldsymbol{u}} = u_{SG}(\xi_k), \tag{3.11}$$

meaning that we must show $(\boldsymbol{B}\boldsymbol{C}^{-1})_{ik} \stackrel{!}{=} \varphi_i(\xi_k)$. Since

$$c_{ji} = \sum_{k=1}^{N_q} a_{jk} b_{ik} = \sum_{k=1}^{N_q} \tilde{w} \varphi_i(\xi_k) \varphi_j(\xi_k) = \delta_{ji},$$

condition (3.11) is fulfilled, concluding the proof. □

Remark 16. *Note that for linear problems (e.g. $f(u) = \xi u$), M quadrature points suffice to discretize the SG moment system in ξ. Therefore $\boldsymbol{\alpha}$ has dimension zero, meaning that the SG and collocation yield the same result.*

The choice of picking the degrees of freedom such that one minimizes the weighted L^2 norm of the solution is convenient in the context of spectral convergence for smooth solutions. However, it does not fit to the theory of hyperbolic equations, which is based on L^1 properties. Especially in non-smooth regimes, the minimization of the L^2 norm leads to oscillatory approximations violating the maximum–principle. Therefore, we choose the resulting degrees of freedom in accordance with the L^1 stability property (1.25a). When including the random variable ξ, this stability result becomes

$$\int_D \langle |u(t,x,\xi) - v(t,x,\xi)| \rangle \, dx \leq \int_D \langle |u_{\mathrm{IC}}(x,\xi) - v_{\mathrm{IC}}(x,\xi)| \rangle \, dx, \tag{3.12}$$

where u and v are entropy solutions of (1.78) with two different initial conditions $u_{\mathrm{IC}}, v_{\mathrm{IC}}$. One chooses $v_{\mathrm{IC}}(x,\xi) = u_{\mathrm{IC}}(x,\xi+h)$, multiplies (3.12) with $1/h$ and lets h go to zero to get

$$\int_D \left\langle \left| \partial_\xi u(t,x,\xi) \right| \right\rangle \, dx \leq \int_D \left\langle \left| \partial_\xi u_{\mathrm{IC}}(x,\xi) \right| \right\rangle \, dx,$$

which shows that oscillations w.r.t. ξ are bounded by oscillations of the initial condition. Therefore, $\boldsymbol{\alpha}$ is now picked such that the reconstruction has minimal oscillations: The total variation in ξ is denoted by

$$\mathrm{TV}(u) := \left\langle \left| \partial_\xi u(t,x,\xi) \right| \right\rangle.$$

Since we are only interested in the solution on a finite set of quadrature points, the discrete total variation

$$\mathrm{TV}_\Delta(\check{\boldsymbol{u}}) = \sum_{k=1}^{N_q} w_k f_\Xi(\xi_k) \left| \check{u}_{k+1} - \check{u}_k \right|$$

is used. The coefficient vector $\boldsymbol{\alpha}$ is now picked such that $\mathrm{TV}_\Delta(\check{\boldsymbol{u}})$ is minimized. The full nIPM moment system is then given by

$$\partial_t \hat{u}_i + \partial_x \sum_{k=1}^{N_q} a_{ik} f(\check{u}_k(t,x)) = 0, \tag{3.13a}$$

$$\check{\boldsymbol{u}}(t,x) = \sum_{i=1}^{N_q - M} \hat{\alpha}_i(t,x) \boldsymbol{v}_i + \sum_{i=0}^{N} \left(\boldsymbol{C}^{-1} \hat{\boldsymbol{u}} \right)_i \boldsymbol{b}_i, \tag{3.13b}$$

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \mathrm{TV}_\Delta(\check{\boldsymbol{u}}). \tag{3.13c}$$

To demonstrate the non-oscillatory behavior of the chosen nIPM reconstruction

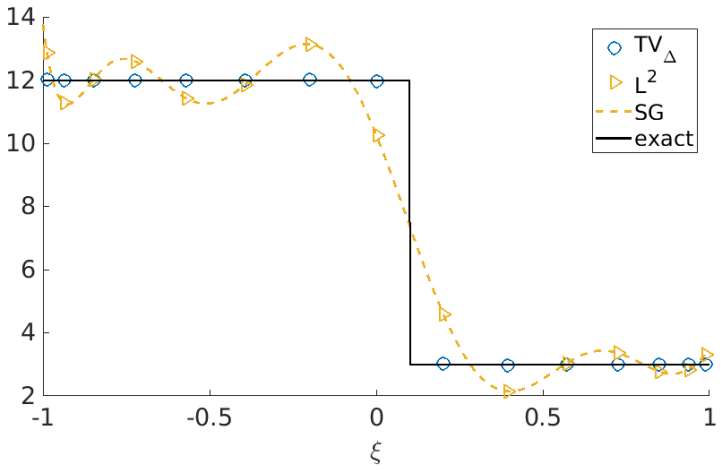


FIGURE 3.1: Reconstructions of nIPM.

(3.13b) and (3.13c), we approximate a shock which is shown in Figure 3.1: After

computing the moments of this shock, the solution is reconstructed on the quadrature points when using the discrete L^2 norm as well as the total variation as functions to minimize. The reconstructions match the first 10 moments at 15 quadrature points, i.e. $\boldsymbol{\alpha} \in \mathbb{R}^5$ needs to be determined. A non-oscillatory discrete solution, which nicely represents the shock on the quadrature points is obtained when choosing $\boldsymbol{\alpha}$ such that the solution minimizes the discrete total variation. When minimizing the L^2 norm, the solution at the quadrature points oscillates, destroying the maximum-principle and leading to a poor approximation. Comparing the solution to the continuous stochastic-Galerkin reconstruction $u_{SG} = \hat{\boldsymbol{u}}^T \boldsymbol{\varphi}$ shows that the nIPM reconstruction ansatz when minimizing the L^2 norm resembles the stochastic-Galerkin solution evaluated at the quadrature points as shown in Proposition 3.

3.3 Interpretation as IPM closure

In the following, we point out that the nodal closure approach is an efficient strategy to determine the minimizer of the IPM optimization problem (1.122) if the number of quadrature points is only slightly larger than the number of moments: First, let us revisit the constraint optimization problem (1.120). The strategy of the dual ansatz (1.122b) is to parametrize all possible solutions which are potential minimizers of the given entropy. To determine the parameters such that the solution fulfills the moment constraint, the dual problem is solved. We take the approach of first ensuring that the moment constraint is fulfilled and after that, we minimize the entropy, which is done on a discrete level. Again using $\check{u}_k := u(t, x, \xi_k)$, the constraint optimization problem (1.120) becomes

$$\mathcal{U}(\hat{\boldsymbol{u}}) = \underset{\check{\boldsymbol{u}} \in \mathbb{R}^{N_q}}{\arg\min} \sum_{k=1}^{N_q} w_k s(\check{u}_k) f_\Xi(\xi_k) \quad \text{subject to } \hat{\boldsymbol{u}} = \sum_{k=1}^{N_q} w_k \check{u}_k \boldsymbol{\varphi}(\xi_k) f_\Xi(\xi_k).$$

Now the task of finding a parameterization which fulfills the constraint is easy: We again span the solution vector $\check{\boldsymbol{u}}$ as in (3.8), i.e.

$$\check{\boldsymbol{u}} = \sum_{i=1}^{N_q - M} \alpha_i \boldsymbol{v}_i + \sum_{i=0}^{N} \beta_i \boldsymbol{b}_i.$$

The coefficient vector $\boldsymbol{\beta}$ is uniquely defined by the moment constraint, see (3.9). Hence we have a parametrization of all solutions fulfilling the moment constraint, where the number of parameters α_i is finite, namely $N_q - M$. We then choose these parameters such that the discrete entropy

$$\sum_{k=1}^{N_q} w_k s(\check{u}_k) f_\Xi(\xi_k)$$

is minimized. In our case, we choose this entropy to be the total variation. The main difference to the IPM method is that we construct the closure on the discretized level, enabling us to find a finite dimensional parameterization of all admissible reconstructions from which we pick the one with the smallest entropy. Consequently, the problem of finding M dual variables $\hat{\boldsymbol{v}}$ has been shifted to finding $N_q - M$ reconstruction parameters $\boldsymbol{\alpha}$. Hence, the computational costs decrease if $N_q - M < M$, i.e. in the transition between collocation and intrusive methods.

3.4 Implementation and refinement

As discussed in Section 3.1, collocation methods yield a poor approximation of the exact moment system. Therefore, our goal is to add quadrature points in regions with high uncertainty. The nIPM allows adding these quadrature points without violating the moment constraint or introducing oscillations. In the following section, we discuss the implementation of the refinement process. Furthermore, we point out how the nIPM moment system (3.13) can be implemented efficiently.

3.4.1 Implementation

The implementation of the nIPM moment system (3.13) does not require writing an entirely new program. In fact most of the code of a given finite volume (FV) implementation for solving the deterministic problem (1.1) can be reused.

As discussed in Section 1.2, the key idea of FV solvers is to divide the spatial domain into cells. The numerical solution is now a collection of intermediate cell values at discrete time steps t_n

$$u_j^n \simeq \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} u(t_n, x)\, dx.$$

Since only the discrete solution for $n = 0$ is known, a time update formula h of the form (1.34), updating the solution by a time step of Δt needs to be derived. The FV implementation now calls h repeatedly until the end time $t_{\text{end}} = N_t \Delta t$ is reached.

Our goal is to calculate the solution of the nIPM moment system (3.13). To solve this system we again use a finite volume approach, meaning that we represent the moments by

$$\hat{\boldsymbol{u}}_j^n := \boldsymbol{A} \check{\boldsymbol{u}}_j^n, \tag{3.14}$$

where $\check{\boldsymbol{u}}_j^n = (\check{u}_{j,1}^n, \cdots, \check{u}_{j,N_q}^n)^T$ with

$$\check{u}_{j,k}^n \simeq \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} u(t_n, x, \xi_k)\, dx.$$

One strategy to evolve the moment vector $\hat{\boldsymbol{u}}_j^n$ in time according to (3.13) is to reconstruct the solution at every quadrature point, meaning that we compute $\check{\boldsymbol{u}}$. On the discrete level, the reconstructed solution is given by

$$\check{\boldsymbol{u}}_j^n = \sum_{i=1}^{N_q - M} \hat{\alpha}_{j,i}^n \boldsymbol{v}_i + \sum_{i=0}^{N} \left(\boldsymbol{C}^{-1} \hat{\boldsymbol{u}}_j^n \right)_i \boldsymbol{b}_i \tag{3.15}$$

where $\hat{\boldsymbol{\alpha}}_j^n$ is given by

$$\hat{\boldsymbol{\alpha}}_j^n = \arg\min_{\boldsymbol{\alpha}} \mathrm{TV}_\Delta \left(\sum_{i=1}^{N_q - M} \alpha_i \boldsymbol{v}_i + \sum_{i=0}^{N} \left(\boldsymbol{C}^{-1} \hat{\boldsymbol{u}}_j^n \right)_i \boldsymbol{b}_i \right).$$

After that, the reconstructed solution in every cell at each quadrature point is evolved in time using the time update function of the FV implementation

$$\check{u}_{j,k}^{n+1} = h(\check{u}_{j-1,k}^n, \check{u}_{j,k}^n, \check{u}_{j+1,k}^n). \tag{3.16}$$

The time updated moments can then be calculated by (3.14). This process is repeated until t_{end} is reached. The extension of a given finite volume implementation to the nIPM scheme can be achieved according to Figure 3.2. We use $\check{\boldsymbol{u}}_\Delta^n$ to denote the field containing the solution at all spatial cells and quadrature points for time step t_n.

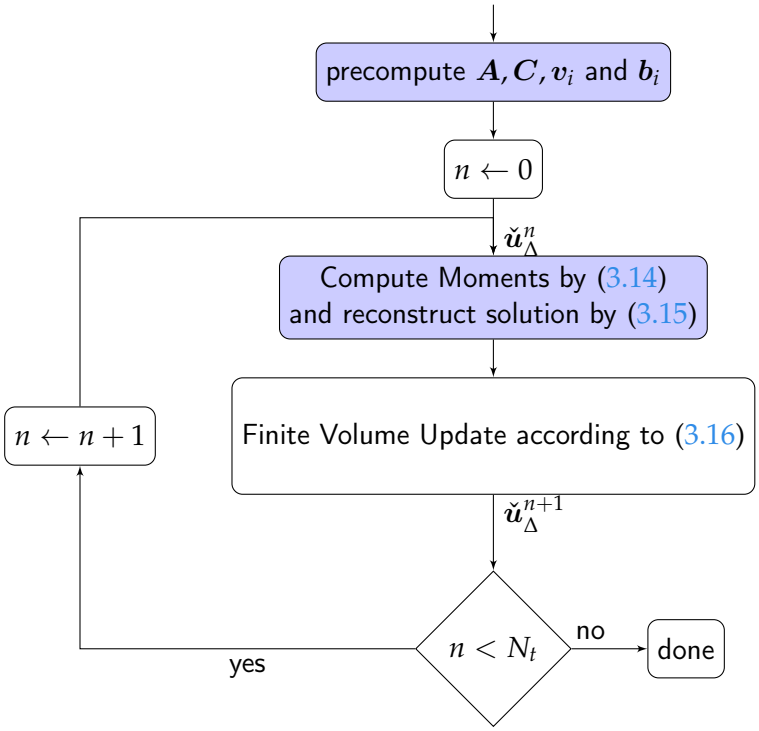


FIGURE 3.2: Integration of the nIPM scheme into the Finite Volume cycle.

3.4.2 Refinement

One can now think about which quadrature should be chosen. For this, we would like to take advantage of the fact that intrusive methods know the behavior of the solution in every time step. In contrast to the non-intrusive black-box approach, adapting the quadrature is possible: If a cell I^* with 'high uncertainty' is detected at spatial position x_i and time step t_m, we are now able to refine the quadrature locally by performing the reconstruction step (3.15) in I^* with a more accurate quadrature, which we denote by

$$\langle u \rangle \approx \sum_{l=1}^{\bar{N}_q} \bar{w}_l u(\bar{\xi}_l),$$

where $\bar{N}_q > N_q$. For ease of notation, the solution at the new quadrature points is denoted by

$$\bar{u}_{l,j}^n \simeq \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} u(t_n, x_j, \bar{\xi}_l)\, dx.$$

Defining the matrix $\bar{\boldsymbol{A}} \in \mathbb{R}^{M \times \bar{N}_q}$ with $\bar{a}_{il} = \bar{w}_l \varphi_i(\bar{\xi}_l) f_\Xi(\bar{\xi}_l)$, we compute $\bar{\boldsymbol{v}}, \bar{\boldsymbol{b}}$ and $\bar{\boldsymbol{C}}$ analogous to before making use of $\bar{\boldsymbol{A}}$ instead of $\boldsymbol{A}$. The reconstruction on the finer

quadrature set given a moment vector $\hat{\boldsymbol{u}}_j^n$ becomes

$$\bar{\boldsymbol{u}}_j^n = \sum_{i=1}^{\bar{N}_q - M} \bar{\alpha}_{j,i}^n \bar{\boldsymbol{v}}_i + \sum_{i=0}^{N} \left(\bar{\boldsymbol{C}}^{-1} \hat{\boldsymbol{u}}_j^n \right)_i \bar{\boldsymbol{b}}_i \tag{3.17}$$

with

$$\bar{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \mathrm{TV}_\Delta \left(\sum_{i=1}^{\bar{N}_q - M} \alpha_i \bar{\boldsymbol{v}}_i + \sum_{i=0}^{N} \left(\bar{\boldsymbol{C}}^{-1} \hat{\boldsymbol{u}}_j^n \right)_i \bar{\boldsymbol{b}}_i \right).$$

Assume that we wish to perform a time update after each cell has been reconstructed by either the standard or the refined quadrature rule. We now update the solution in the refined cell I^* in time by

$$\bar{u}_{i,l}^{m+1} = h(\bar{u}_{i-1,l}^m, \bar{u}_{i,l}^m, \bar{u}_{i+1,l}^m) \tag{3.18}$$

for $l = 1, \cdots, \bar{N}_q$. Here, one notices that solution values of the neighboring cells of I^* also need to be known on the refined quadrature points. Therefore, cells which have neighbors with a fine as well as a standard reconstruction need to be reconstructed on both the fine and the standard quadrature set. Using Clenshaw-Curtis quadrature sets allows reconstructing these interface cells only for the fine quadrature rule since the quadrature points of different refinement levels are nested. In the following, we present the nIPM algorithm with refinement for non-nested quadrature rules.

Assume that a spatial grid as depicted in Figure 3.3 is given and on each cell we know the moment vector. The cell types are then identified by choosing a function which indicates high uncertainty. Here, we use the highest order moments to identify non-smooth regimes in the uncertain domain similar to [65]. Note that this strategy has also been applied for DG methods [106]. To avoid unwanted artifacts from even or odd functions, we take the last two moments and divide by the solution's squared L^2 norm as smoothness indicator. If the indicator in a given cell lies above a specified value η, i.e.

$$\frac{|\hat{u}_N^n| + |\hat{u}_{N-1}^n|}{(\hat{\boldsymbol{u}}^n)^T \hat{\boldsymbol{u}}^n} > \eta \tag{3.19}$$

we indicate it as a *fine* cell. Cells which are not *fine*, but have a neighboring *fine* cell are *transition* cells. The cells which have a *transition* neighbor and are not fine are called *interface* cells. All remaining cells are *coarse*. In our example in Figure 3.3, the two red cells have a smoothness indicator lying above a specified value and are therefore *fine* (f). The neighboring cells are identified as *transition* (t), since they have *fine* neighbors, *interface* (i), since they are not fine and have a *transition* neighbor and *coarse* (c). In a next step, the reconstructions need to be computed: We do this on the fine quadrature grid if the cells are *fine* and on the coarse grid if they are *coarse*. For *interface* and *transition* cells we compute both a coarse and a fine reconstruction. The next task is to evolve the reconstructions in time. One needs to perform a coarse time update for *coarse* and *interface* cells and a fine time update for the remaining cells, which are exemplary illustrated once for each cell type in Figure 3.3. The coarse time update stencil of (3.16) is depicted by downward pointing and the fine stencil of (3.18) is depicted by upward pointing arrows. By looking at the dependencies of the stencils, one sees why it is necessary to reconstruct both the fine and the coarse solution for *interface* and *transition* cells. We then compute the time-updated

moments using $\boldsymbol{A}\check{\boldsymbol{u}}_j^{n+1}$ on *coarse* and *interface* cells and $\bar{\boldsymbol{A}}\bar{\boldsymbol{u}}_j^{n+1}$ on *transition* and *fine* cells. The entire process is then repeated until the end time is reached.

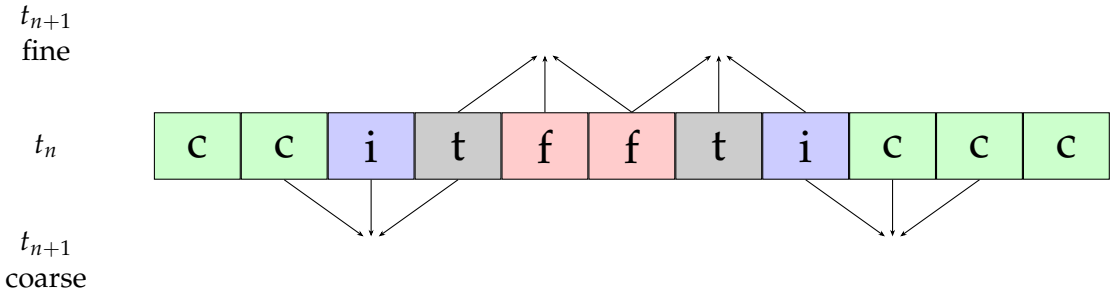


FIGURE 3.3: nIPM with refinement.

We use the proposed refinement strategy to locally switch between collocation and standard nIPM, i.e. the coarse quadrature set has M quadrature points. If a cell has high uncertainty, we add quadrature points by our refinement strategy, to improve the time evolution approximation of the moment system. We refer to this method as adaptive nIPM. Note that switching to collocation updates does not mean the proposed method is non-intrusive. However, we can circumvent the expensive computation of a closure (i.e. picking $\boldsymbol{\alpha} \in \mathbb{R}^{N_q - M}$ by solving (3.13c)), since $N_q - M = 0$.

3.5 Results

In the following, we present results of the nIPM method and compare them to results obtained with the SG method. To compare these different strategies, we again solve the uncertain Burgers' equation (1.80), given by

$$\begin{aligned} \partial_t u(t,x,\xi) + \partial_x \frac{u(t,x,\xi)^2}{2} &= 0, \\ u(t=0,x,\xi) &= u_{\mathrm{IC}}(x,\xi) \end{aligned}$$

on $x \in [0,3]$ with the uncertain initial condition

$$u_{\mathrm{IC}}(x,\xi) := \begin{cases} u_L, & \text{if } x < x_0 + \sigma\xi \\ u_L + \frac{u_R - u_L}{x_0 - x_1}(x_0 + \sigma\xi - x), & \text{if } x \in [x_0 + \sigma\xi, x_1 + \sigma\xi] \\ u_R, & \text{else} \end{cases}. \tag{3.20}$$

As described in Section 1.4.1, we choose ξ to be uniformly distributed in $\Theta = [-1,1]$. To ensure good integral approximations on Θ, we use a Gauss-Legendre quadrature rule. Parameters of the initial condition and the numerical scheme can be found in the following table:

$N_x = 500$	number of spatial cells
$t_{\text{end}} = 0.11$	end time
$x_0 = 0.5, x_1 = 1.5, u_L = 12, u_R = 1, \sigma = 0.3$	parameters of initial condition (3.20)
$M = 15$	number of moments
$\eta = 5 \cdot 10^{-4}$	barrier (3.19) for refinement

The basis vectors $\boldsymbol{v}_i$ of the nIPM are calculated by performing a singular value decomposition of $\boldsymbol{A}$ and the vectors $\boldsymbol{b}_i$ are chosen as

$$(\boldsymbol{b}_i)_k = \varphi_i(\xi_k)$$

for $i = 0, \cdots, N$.

3.5.1 Convergence of expected value

In the following, we study the effects of varying the number of quadrature points N_q to study the L^1 error of the expected value by

$$e := \int_0^3 |\hat{u}_0(t_{\text{end}}, x) - \hat{u}_{0,\text{ex}}(t_{\text{end}}, x)| \, dx, \tag{3.21}$$

where $\hat{u}_{0,\text{ex}}(t_{\text{end}}, x)$ is the expected value (which is equal to the zeroth moment) of the exact solution at the final time t_{end}. As discussed in Section 3.1, choosing $N_q > M$ necessitates feeding information to the moment system. The nIPM picks these degrees of freedom to minimize the total variation, whereas the SG method minimizes the L^2 norm. When applying the adaptive nIPM, we can switch between different quadrature rules to locally refine the set of integration points. In the following, the number of quadrature points on the coarse level N_q^c equals the number of moments, i.e. $N_q^c = M = 15$. Hence, we perform collocation updates on the coarse level. We switch to intrusive methods whenever the smoothness indicator lies above a value of $5 \cdot 10^{-4}$. The number of quadrature points on the fine level N_q is then varied to study the error of the expected value.

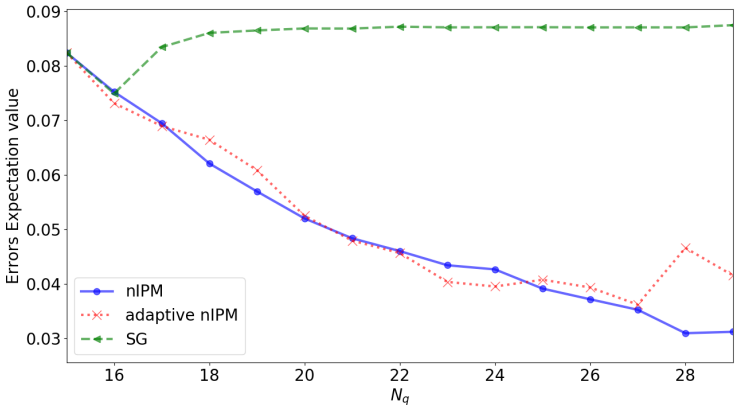


FIGURE 3.4: L^1 error of the expected value as defined in (3.21) at time $t_{\text{end}} = 0.11$ when using $N_x = 500$ spatial cells. The adaptive nIPM switches between collocation ($N_q^c = 15$ quadrature points) and the value of N_q depicted on the x-axis. The corresponding numerical values can be found in Table 3.1.

The resulting errors for different N_q are shown in Figure 3.4. If $N_q = 15$, all three methods resemble Stochastic Collocation , which is why they all have the same error. The methods differ in the choice of degrees of freedom when adding quadrature points. Taking a look at the SG method, improving the numerical integration has little effect. Moreover, the resulting error will start to grow after having added one quadrature point, leaving the SG method with a poorer approximation as the

N_q	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
SG	8.24	7.49	8.34	8.60	8.65	8.68	8.68	8.71	8.70	8.70	8.70	8.70	8.70	8.70	8.74	8.66
nIPM	8.24	7.52	6.94	6.21	5.70	5.20	4.84	4.60	4.34	4.27	3.91	3.72	3.53	3.10	3.12	2.85
anIPM	8.24	7.31	6.89	6.64	6.08	5.25	4.79	4.57	4.04	3.95	4.08	3.93	3.62	4.66	4.15	4.63

TABLE 3.1: L^1 error (according to (3.21)) for SG, nIPM and adaptive nIPM (anIPM) as depicted in Figure 3.4. All values are depicted in E-2, i.e. need to be multiplied by 10^{-2}.

collocation result. In contrast to this, the nIPM steadily decreases the error when adding quadrature points. Hence, the choice of degrees of freedom helps improving the approximation of the integrals appearing in the moment system. The adaptive nIPM shows a similar error decrease. Locally switching to collocation whenever the solution shows low uncertainty decreases the computational costs while leading to a slightly increased error.

3.5.2 Comparison of expectation value and variance

In the following we fix the number of quadrature points at 25. The adaptive nIPM switches between collocation as well as nIPM with 25 quadrature points. We start by looking at the expectation value and the variance in Figure 3.5. The SG method shows a step-like approximation of the expectation value, whereas nIPM and adaptive nIPM show only small variations from the exact solution. Furthermore, one observes that the expected values computed with nIPM and its adaptive version differ only slightly. Consequently, the poor approximation of integrals in the moment system in cells with low smoothness indicator does not destroy the accuracy achieved with the standard nIPM. A similar result can be found when investigating the variance: While the variance of SG oscillates heavily, both nIPM and adaptive nIPM show mitigated oscillations.

Lastly, we take a look at the reconstruction of the solution at a fixed spatial position $x^* = 2.1$ for the final time t_{end}. The exact solution is a reversed shock from u_R to u_L. Note that since we use a quadrature rule with $N_q = 25$, we only need to know the solution at the corresponding 25 quadrature points. A continuous reconstruction of the SG method can be computed by interpolating the given solution points with Legendre polynomials. In the nIPM case, a continuous reconstruction can be calculated with the adjoint approach of IPM (1.122a). Since our interest lies in computing the solution's moments, we skip this step. Figure 3.6 shows that the SG method has great difficulties when representing the reversed shock, because this leads to oscillations. The solution violates the maximum-principle and shows high total variation in ξ. Both the nIPM and adaptive nIPM yield a reconstruction which does not violate the bounds of the initial condition, namely u_R and u_L. The solution nicely captures the exact shock structure. Also note that the adaptive nIPM makes use of 25 instead of 15 quadrature points at the shock position, meaning that the spatial cell at x^* is identified as a *fine* cell. Consequently, the shock position is nicely localized.

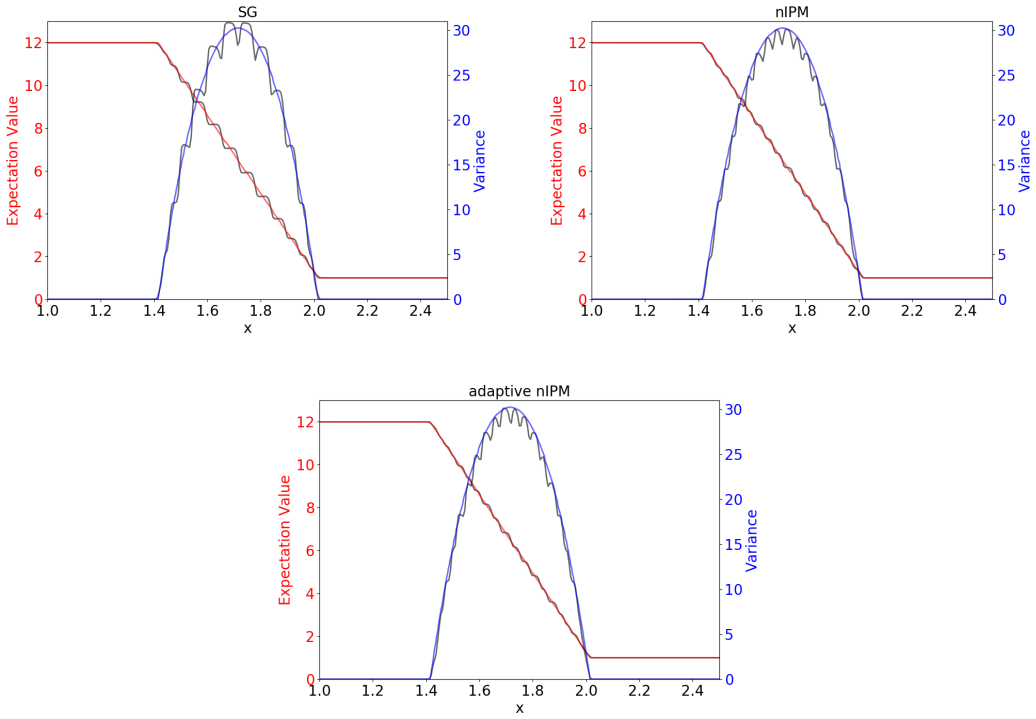


FIGURE 3.5: Expected value and variance for 15 moments with 25 quadrature points. The adaptive nIPM switches between collocation (15 quadrature points) and 25 quadrature points. The exact expectation value is plotted in red and the exact variance is plotted in blue.

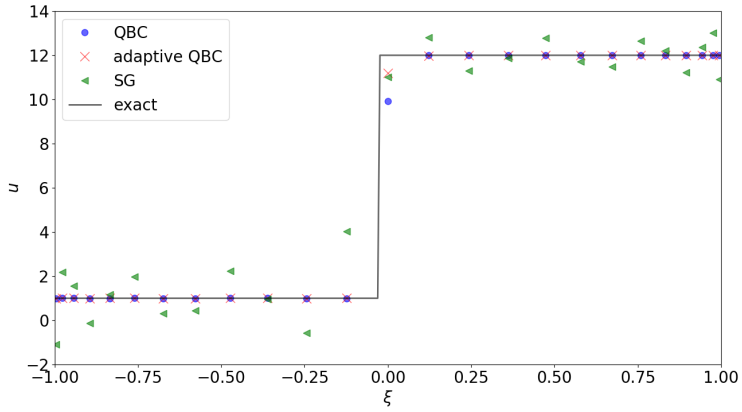


FIGURE 3.6: Solution at fixed spatial position $x^* = 2.1$

Chapter 4

Intrusive acceleration strategies

So far, we investigated the IPM method in scalar settings, in which a correctly chosen entropy can mitigate oscillation while providing a maximum–principle. Systems on the other hand do not necessarily fulfill a maximum-principle while having a limited choice of admissible entropies. I.e. unlike for scalar problems, not all convex functions fulfill the integrability condition (1.8) and imposing an entropy which promises non-oscillatory solutions becomes a challenging task. However, in contrast to stochastic-Galerkin , the IPM method promises hyperbolicity, which is why one should try to accelerate IPM for systems.

At this point, one might ask the question why one should even look at intrusive methods, since black-box collocation methods for example do not suffer from the loss of hyperbolicity, while providing a robust, convenient framework for uncertainty quantification. There are several arguments, which however point to the applicability of intrusive methods for challenging settings:

- Several known techniques to accelerate numerical methods are intrusive and therefore require an intrusive implementation. One example which has the potential of significantly decreasing numerical costs is adaptivity: Several applications with random hyperbolic equations tend to show shocks in ξ on only a small portion of the spatial domain. Hence, adaptively increasing the number of moments in this area while keeping a low number on the remaining part of the spatial discretization can heavily decrease numerical costs. Non-intrusive methods on the other hand need to run the deterministic code on the finest level (i.e. on a large number of samples) to obtain the same error level. Further examples for acceleration techniques requiring an intrusive implementation are the dynamical low-rank method [64] as well as filtering [91, 70].

- In our numerical experiments, a smaller number of unknowns suffices to obtain a given error level when using intrusive methods compared to SC. Especially for a high-dimensional random space Θ, this can heavily decrease the total number of unknowns. Furthermore, we observe that steady state problems require less iterations to reach a steady state solution when being treated with stochastic-Galerkin or the IPM method compared to Stochastic Collocation .

- As mentioned in Section 1.4.4, the number of quadrature points needed to integrate polynomials up to a certain total degree N exactly, is larger than the number of basis functions with total degree smaller or equal to N. Hence, intrusive methods (if they only work with the moments) require a reduced number of unknowns compared to SC since here, the number of unknowns is given by the number of quadrature points. This can yield significantly reduced numerical costs for high-dimensional problems when using intrusive methods.

- The moments are smoother with respect to the spatial variable than the solution at a fixed sample point ξ. As an example, one can look at the expectation value and variance for the Euler equations in Section 1.4.1. While expectation value and variance show a smooth profile, the corresponding solution at a fixed ξ is a shock. Hence, the chosen mesh to represent the moments does not need to be as fine as for the deterministic problem and solvers for the moment system do not require methods such as shock capturing.

Therefore, one aim should be to accelerate intrusive methods, since they can potentially outperform non-intrusive methods in complex and high-dimensional settings.

In this chapter, we propose acceleration techniques for intrusive methods and compare them against Stochastic Collocation . For steady and unsteady problems, we use adaptivity, for which intrusive methods provide a convenient framework:

- Since complex structures in the uncertain domain tend to arise in small portions of the spatial mesh, our aim is to locally increase the accuracy of the stochastic discretization in regions that show a complex structure in the random domain, while choosing a low order method in the remainder. Such an adaptive treatment cannot be realized with non–intrusive methods, since one needs to break up the black-box approach. To guarantee an efficient implementation, we propose an adaptive discretization strategy for IPM.

A steady problem provides different opportunities to take advantage of features of intrusive methods:

- When using adaptivity, one can perform a large number of iterations to the steady state solution on a low number of moments and increase the maximal truncation order when the distance to the steady state has reached a specified barrier. Consequently, a large number of iterations will be performed by a cheap, low order method, i.e. we can reduce numerical costs.

- Perform an inexact map from the moments to the dual variables for IPM: Since the moments during the iteration process are inaccurate, i.e. they are not the correct steady state solution, we propose to not fully converge the dual iteration, which solves the IPM optimization problem. Consequently, the entropic expansion coefficients and the moments are converged simultaneously to their steady state, which is similar to the idea of One-Shot optimization in shape optimization [53].

The effectiveness of these acceleration ideas is tested by comparing results with Stochastic Collocation for the NACA test case [61] with uncertainties as well as a bent shock tube problem. Our numerical studies show the following main results:

- In our test cases, the need to solve an optimization problem when using the IPM method leads to a significantly higher run time than SC and SG. However when using the discussed acceleration techniques, IPM requires the shortest time to reach a given accuracy.

- Comparing SG with IPM, one observes that for the same number of unknowns, SG yields more accurate expectation values, whereas IPM shows improved variance approximations.

- Using sparse grids for the IPM discretization when the space of uncertainty is multi-dimensional, the number of quadrature points needed to guarantee sufficient regularity of the Hessian matrix is significantly increased.

The IPM and SG calculations use a semi-intrusive C++ code framework [72][1], meaning that the discretization allows recycling a given deterministic code to generate the IPM solver. The main idea of this framework is based on the kinetic flux (2.5). While facilitating the task of implementing general intrusive methods, this framework reduces the number of operations required to compute numerical fluxes. Also, it provides the ability to base the intrusive method on the same deterministic solver as used in the implementation of a black-box fashion Stochastic Collocation code, which allows for, what we believe, a fair comparison between intrusive and non-intrusive methods. The code is publicly available to allow reproducibility [72].

This chapter is structured as follows: A numerical IPM discretization for hyperbolic systems (similar to Algorithm 2, which has been defined for scalar problems) is introduced in Section 4.1. In Section 4.2, we discuss the idea of not converging the dual iteration. Section 4.3 extends the presented numerical framework to an algorithm making use of adaptivity. Implementation and parallelization details are given in Section 4.4. A comparison of results computed with the presented methods is then given in Section 4.5.

4.1 A realizability-preserving IPM algorithm for systems

Note that several building blocks of the IPM algorithm have already been discussed in Sections 1.4.5 and 2.1. In the following, we extend these ideas to systems when using a finite volume scheme to discretize space and time as well as quadrature rules to compute integrals. Omitting initial conditions and assuming a one-dimensional spatial domain, we can write the IPM system (1.108) as

$$\partial_t \hat{\boldsymbol{u}} + \partial_x \boldsymbol{F}(\hat{\boldsymbol{u}}) = \boldsymbol{0} \tag{4.1}$$

with the flux $\boldsymbol{F} : \mathbb{R}^{M\times m} \to \mathbb{R}^{M\times m}$, $\boldsymbol{F}(\hat{\boldsymbol{u}}) = \langle \boldsymbol{f}(\mathcal{U}(\hat{\boldsymbol{u}}))\boldsymbol{\varphi}^T\rangle^T$. Note that the inner transpose represents a dyadic product and therefore the outer transpose is applied to a matrix. Furthermore, we make use of the IPM closure $\mathcal{U}(\hat{\boldsymbol{u}})$ defined in (1.120). Due to hyperbolicity of the IPM moment system, one can use a finite volume method to approximate the time evolution of the IPM moments. We choose the discrete unknowns which represent the solution to be the spatial averages over each cell at time t_n, given by

$$\hat{\boldsymbol{u}}_{ij}^n \simeq \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \hat{\boldsymbol{u}}_i(t_n, x)dx.$$

If a moment vector in cell j at time t_n is denoted as $\hat{\boldsymbol{u}}_j^n = (\hat{\boldsymbol{u}}_{ij}^n)_{|i|\leq N} \in \mathbb{R}^{M\times m}$, the finite-volume scheme can be written in conservative form with the numerical flux $\boldsymbol{F}^* : \mathbb{R}^{M\times m}\times\mathbb{R}^{M\times m} \to \mathbb{R}^{M\times m}$ as

$$\hat{\boldsymbol{u}}_j^{n+1} = \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^*(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) - \boldsymbol{F}^*(\hat{\boldsymbol{u}}_{j-1}^n, \hat{\boldsymbol{u}}_j^n)\right) \tag{4.2}$$

[1] A more detailed description of the implementation and especially the parallelization used in [72] can be found in Section 4.4.

for $j = 1, \cdots, N_x$ and $n = 0, \cdots, N_t$. Here, the number of spatial cells is denoted by N_x and the number of time steps by N_t. The numerical flux is assumed to be consistent, i.e. $\boldsymbol{F}^*(\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}}) = \boldsymbol{F}(\hat{\boldsymbol{u}})$.

When a consistent numerical flux $\boldsymbol{f}^* : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$, $\boldsymbol{f}^* = \boldsymbol{f}^*(\boldsymbol{u}_\ell, \boldsymbol{u}_r)$ is available for the original problem (1.78), then for the IPM system we can simply take the numerical flux

$$\widetilde{\boldsymbol{F}}^*(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) = \langle \boldsymbol{f}^*(\mathcal{U}(\hat{\boldsymbol{u}}_j^n), \mathcal{U}(\hat{\boldsymbol{u}}_{j+1}^n))\boldsymbol{\varphi}^T \rangle^T$$

in (4.2). Note that this is the kinetic flux as defined in (2.5) extended to systems. In general, the integral term inside the numerical flux cannot be evaluated analytically and therefore needs to be approximated by a quadrature rule

$$\langle h \rangle \approx \langle h \rangle_Q := \sum_{k=1}^{Q} w_k h(\boldsymbol{\xi}_k) f_\Xi(\boldsymbol{\xi}_k).$$

The approximated numerical flux then becomes

$$\boldsymbol{F}^*(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) = \langle \boldsymbol{f}^*(\mathcal{U}(\hat{\boldsymbol{u}}_j^n), \mathcal{U}(\hat{\boldsymbol{u}}_{j+1}^n))\boldsymbol{\varphi}^T \rangle_Q^T. \tag{4.3}$$

Note that the numerical flux requires evaluating the ansatz $\mathcal{U}(\hat{\boldsymbol{u}}_j^n)$, which involves solving the dual problem (1.122b). Again using $\boldsymbol{u} : \mathbb{R}^m \to \mathbb{R}^m$ as defined in Section 1.1.3, given by

$$\boldsymbol{u}(\boldsymbol{v}) := (\nabla_{\boldsymbol{u}} s)^{-1}(\boldsymbol{v}),$$

allows writing the IPM ansatz (1.120) at cell j in timestep n as

$$\mathcal{U}(\hat{\boldsymbol{u}}_j^n) = \boldsymbol{u}(\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_j^n)^T \boldsymbol{\varphi}).$$

The computation of the dual variables $\hat{\boldsymbol{v}}_j^n := \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_j^n)$ requires solving the dual problem (1.122b) for the moment vector $\hat{\boldsymbol{u}}_j^n$ when interpreting IPM as a closure method, see Section 1.4.6. Therefore, to determine the dual variables for a given moment vector $\hat{\boldsymbol{u}}$, the cost function

$$L(\boldsymbol{v}; \hat{\boldsymbol{u}}) := \langle s_*(\boldsymbol{v}^T \boldsymbol{\varphi}) \rangle_Q - \sum_{i \leq N} \boldsymbol{v}_i^T \hat{\boldsymbol{u}}_i \tag{4.4}$$

needs to be minimized. Hence, one needs to find the root of

$$\nabla_{\boldsymbol{v}} L(\boldsymbol{v}; \hat{\boldsymbol{u}}) = \langle \nabla s_*(\boldsymbol{v}^T \boldsymbol{\varphi}) \boldsymbol{\varphi}^T \rangle_Q^T - \hat{\boldsymbol{u}} = \langle \boldsymbol{u}(\boldsymbol{v}^T \boldsymbol{\varphi}) \boldsymbol{\varphi}^T \rangle_Q^T - \hat{\boldsymbol{u}},$$

where we used $\nabla s_* \equiv \boldsymbol{u}$. Again, we determine the root by using Newton's method. For simplicity, let us define the full gradient of the Lagrangian to be $\nabla_{\boldsymbol{v}} L(\boldsymbol{v}; \hat{\boldsymbol{u}}) \in \mathbb{R}^{M \cdot m}$, i.e. we store all entries in a vector. Newton's method uses the iteration function $\boldsymbol{d} : \mathbb{R}^{M \times m} \times \mathbb{R}^{M \times m} \to \mathbb{R}^{M \times m}$, which has been defined in (1.114). With our numerical discretization of integral terms, this iteration function reads

$$\boldsymbol{d}(\boldsymbol{v}, \hat{\boldsymbol{u}}) := \boldsymbol{v} - \hat{\boldsymbol{H}}(\boldsymbol{v})^{-1} \cdot \nabla_{\boldsymbol{v}} L(\boldsymbol{v}; \hat{\boldsymbol{u}}), \tag{4.5}$$

where $\hat{\boldsymbol{H}} \in \mathbb{R}^{M \cdot m \times M \cdot m}$ is the Hessian of (4.4), given by

$$\hat{\boldsymbol{H}}(\boldsymbol{v}) := \langle \nabla \boldsymbol{u}(\boldsymbol{v}^T \boldsymbol{\varphi}) \otimes \boldsymbol{\varphi} \boldsymbol{\varphi}^T \rangle_Q^T.$$

The function $\boldsymbol{d}$ will in the following be called dual iteration function. Now, the Newton iteration l for spatial cell j is given by

$$\boldsymbol{v}_j^{(l+1)} = \boldsymbol{d}(\boldsymbol{v}_j^{(l)}, \hat{\boldsymbol{u}}_j). \tag{4.6}$$

The exact dual state is then obtained by computing the fixed point of $\boldsymbol{d}$, meaning that one converges the iteration (4.6), i.e. $\hat{\boldsymbol{v}}_j^n := \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_j^n) = \lim_{l\to\infty} \boldsymbol{d}(\boldsymbol{v}_j^{(l)}, \hat{\boldsymbol{u}}_j^n)$. Again, to obtain a finite number of iterations for the iteration in cell j, a stopping criterion

$$\sum_{i=0}^{m} \left\| \nabla_{\boldsymbol{v}_i} L(\boldsymbol{v}_j^{(l)}; \hat{\boldsymbol{u}}_j^n) \right\| < \tau \tag{4.7}$$

is used.

We now write down the entire scheme including the recalculation idea to preserve realizability as discussed in Section 2.2.3: To obtain a more compact notation, we define

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{v}_\ell, \boldsymbol{v}_c, \boldsymbol{v}_r) :=& \langle \boldsymbol{u}(\boldsymbol{v}_c^T\boldsymbol{\varphi})\boldsymbol{\varphi}^T\rangle_Q^T \\ &- \frac{\Delta t}{\Delta x}\left(\langle \boldsymbol{f}^*(\boldsymbol{u}(\boldsymbol{v}_c^T\boldsymbol{\varphi}), \boldsymbol{u}(\boldsymbol{v}_r^T\boldsymbol{\varphi}))\boldsymbol{\varphi}^T\rangle_Q^T - \langle \boldsymbol{f}^*(\boldsymbol{u}(\boldsymbol{v}_\ell^T\boldsymbol{\varphi}), \boldsymbol{u}(\boldsymbol{v}_c^T\boldsymbol{\varphi}))\boldsymbol{\varphi}^T\rangle_Q^T\right). \end{aligned} \tag{4.8}$$

Note that this scheme directly includes the recalculation step, since the first term on the right hand side recalculates the moment vector with the inexact dual state. The moment iteration is then given by

$$\hat{\boldsymbol{u}}_j^{n+1} = \boldsymbol{c}\left(\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_{j-1}^n), \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_j^n), \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_{j+1}^n)\right), \tag{4.9}$$

where the map from the moment vector to the dual variables, i.e. $\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_j^n)$, is obtained by iterating

$$\boldsymbol{v}_j^{(l+1)} = \boldsymbol{d}(\boldsymbol{v}_j^{(l)}; \hat{\boldsymbol{u}}_j^n). \tag{4.10}$$

until condition (4.7) is fulfilled. This gives Algorithm 3.

Algorithm 3 IPM algorithm for systems

for $j = 0$ to $N_x + 1$ **do**
  $\boldsymbol{u}_j^0 \leftarrow \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x,\cdot)\boldsymbol{\varphi}\rangle_Q dx$
for $n = 0$ to N_t **do**
  for $j = 0$ to $N_x + 1$ **do**
    $\boldsymbol{v}_j^{(0)} \leftarrow \hat{\boldsymbol{v}}_j^n$
    while (4.7) is violated **do**
      $\boldsymbol{v}_j^{(l+1)} \leftarrow \boldsymbol{d}(\boldsymbol{v}_j^{(l)}; \hat{\boldsymbol{u}}_j^n)$
      $l \leftarrow l + 1$
    $\hat{\boldsymbol{v}}_j^{n+1} \leftarrow \boldsymbol{v}_j^{(l)}$
  for $j = 1$ to N_x **do**
    $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \boldsymbol{c}(\hat{\boldsymbol{v}}_{j-1}^{n+1}, \hat{\boldsymbol{v}}_j^{n+1}, \hat{\boldsymbol{v}}_{j+1}^{n+1})$

4.2 One-Shot IPM

In the following section we only consider steady state problems, i.e. the original hyperbolic equation (1.1) reduces to

$$\nabla \cdot \boldsymbol{f}(\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\xi})) = \boldsymbol{0} \quad \text{in } D \tag{4.11}$$

with adequate boundary conditions. A general strategy for computing the steady state solution to (4.11) is to introduce a pseudo-time and numerically treat (4.11) as an unsteady problem. Hence, when denoting the pseudo-time by $t \in \mathbb{R}_+$, the above equations become

$$\partial_t \boldsymbol{u}(t, \boldsymbol{x}, \boldsymbol{\xi}) + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}(t, \boldsymbol{x}, \boldsymbol{\xi})) = \boldsymbol{0} \quad \text{in } D \tag{4.12}$$

with a user-determined initial condition. A steady state solution is then obtained by discretizing the pseudo-time in (4.12) with a forward Euler method and iterating in pseudo-time until the solution remains constant. Note that this method can only converge if the the physical flux $\boldsymbol{f}$ admits a steady state solution. Furthermore, it is important to point out that the time it takes to converge to a steady state solution is crucially affected by the chosen initial condition and its distance to the steady state solution. Similar to the unsteady case (1.78), we can again derive a moment system for (4.11) given by

$$\nabla \cdot \langle \boldsymbol{f}(\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\xi}))\boldsymbol{\varphi}^T \rangle^T = \boldsymbol{0} \quad \text{in } D \tag{4.13}$$

which is again needed for the construction of intrusive methods. By introducing a pseudo-time t and using the IPM closure, we obtain the same system as in (4.1), i.e. Algorithm 3 can be used to iterate to a steady state solution. Note that the CFL condition from the unsteady moment system (4.1), which gives a stable time step size Δt. Now, the time iteration is not performed for a fixed number of time steps N_t, but until the condition

$$\sum_{j=1}^{N_x} \Delta x_j \| \hat{\boldsymbol{u}}_j^n - \hat{\boldsymbol{u}}_j^{n-1} \| \leq \varepsilon \tag{4.14}$$

is fulfilled. Condition (4.14), which is for example being used in the SU2 code framework [32], measures the change of the solution by a single time iteration. Note, that in order to obtain an estimate of the distance to the steady state solution, one has to include the Lipschitz constant of the corresponding fixed point problem. Since one is generally interested in low order moments such as the expectation value, the residual (4.14) can be modified by only accounting for the zero order moments.

In this section we aim at breaking up the inner loop in the IPM Algorithm 3, i.e. to just perform one iteration of the dual problem in each time step. Consequently, the IPM reconstruction given by (1.120) is not done exactly, meaning that the reconstructed solution does not minimize the entropy while not fulfilling the moment constraint. However, the fact that the moment vectors are not yet converged to the steady solution seems to permit such an inexact reconstruction. Hence, we aim at iterating the moments to steady state and the dual variables to the exact solution of the IPM optimization problem (1.120) simultaneously. By successively performing

one update of the moment iteration and one update of the dual iteration, we obtain

$$\boldsymbol{v}_j^{n+1} = \boldsymbol{d}(\boldsymbol{v}_j^n, \boldsymbol{u}_j^n) \quad \text{for all } j \tag{4.15a}$$

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{c}\left(\boldsymbol{v}_{j-1}^{n+1}, \boldsymbol{v}_j^{n+1}, \boldsymbol{v}_{j+1}^{n+1}\right) \quad \text{for all } j. \tag{4.15b}$$

This yields Algorithm 4.

Algorithm 4 One-Shot IPM implementation

for $j = 0$ to $N_x + 1$ **do**
  $\boldsymbol{u}_j^0 \leftarrow \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x, \cdot)\boldsymbol{\varphi} \rangle_Q dx$
while (4.14) is violated **do**
  for $j = 1$ to N_x **do**
    $\boldsymbol{v}_j^{n+1} \leftarrow \boldsymbol{d}(\boldsymbol{v}_j^n; \hat{\boldsymbol{u}}_j^n)$
  for $j = 1$ to N_x **do**
    $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \boldsymbol{c}(\boldsymbol{v}_{j-1}^{n+1}, \boldsymbol{v}_j^{n+1}, \boldsymbol{v}_{j+1}^{n+1})$
  $n \leftarrow n + 1$

We call this method One-Shot IPM, since it is inspired by One-Shot optimization, see for example [53], which uses only a single iteration of the primal and dual step in order to update the design variables. Note that the dual variables from the One-Shot IPM iteration are written without a hat to indicate that they are not the exact solution of the dual problem.

In the following, we will show that this iteration converges, if the chosen initial condition is sufficiently close to the steady state solution. For this we take an approach commonly chosen to prove local convergence properties of Newton's method: In Theorem 17, we show that the iteration function is contractive at its fixed point and conclude in Theorem 18 that this yields local convergence. Hence, we preserve the convergence property of the original IPM method, which uses Newton's method and therefore only converges locally as well.

Theorem 17. *Assume that the classical IPM iteration is contractive at its fixed point* $\hat{\boldsymbol{u}}^*$. *Then the Jacobian* $\boldsymbol{J}$ *of the One-Shot IPM iteration* (4.15) *has a spectral radius* $\rho(\boldsymbol{J}) < 1$ *at the fixed point* $(\boldsymbol{v}^*, \hat{\boldsymbol{u}}^*)$.

Proof. First, to understand what contraction of the classical IPM iteration implies, we rewrite the moment iteration (4.9) of the classical IPM scheme: When defining the update function

$$\tilde{\boldsymbol{c}}\left(\hat{\boldsymbol{u}}_\ell, \hat{\boldsymbol{u}}_c, \hat{\boldsymbol{u}}_r\right) := \boldsymbol{c}\left(\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_\ell), \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_c), \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}_r)\right)$$

we can rewrite the classical moment iteration as

$$\hat{\boldsymbol{u}}_j^{n+1} = \tilde{\boldsymbol{c}}\left(\hat{\boldsymbol{u}}_{j-1}^n, \hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n\right). \tag{4.16}$$

Since we assume that the classical IPM scheme is contractive at its fixed point, we have $\rho(\nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}}(\hat{\boldsymbol{u}}^*)) < 1$ with $\nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}} \in \mathbb{R}^{M\cdot N_x \times M\cdot N_x}$ defined by

$$\nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}} = \begin{pmatrix} \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_1 & \partial_{\hat{\boldsymbol{u}}_r}\tilde{\boldsymbol{c}}_1 & 0 & 0 & \dots \\ \partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_2 & \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_2 & \partial_{\hat{\boldsymbol{u}}_r}\tilde{\boldsymbol{c}}_2 & 0 & \dots \\ 0 & \partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_3 & \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_3 & \partial_{\hat{\boldsymbol{u}}_r}\tilde{\boldsymbol{c}}_3 & \\ \vdots & & & \ddots & \\ 0 & \dots & 0 & \partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_{N_x} & \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_{N_x} \end{pmatrix},$$

where we define $\tilde{\boldsymbol{c}}_j := \tilde{\boldsymbol{c}}\left(\hat{\boldsymbol{u}}^*_{j-1}, \hat{\boldsymbol{u}}^*_j, \hat{\boldsymbol{u}}^*_{j+1}\right)$ for all j. Now for each term inside the matrix $\nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}}$ we have

$$\partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \hat{\boldsymbol{v}}_\ell}\frac{\partial \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}^*_{j-1})}{\partial \hat{\boldsymbol{u}}}, \quad \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \hat{\boldsymbol{v}}_c}\frac{\partial \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}^*_j)}{\partial \hat{\boldsymbol{u}}}, \quad \partial_{\hat{\boldsymbol{u}}_r}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \hat{\boldsymbol{v}}_r}\frac{\partial \hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}^*_{j+1})}{\partial \hat{\boldsymbol{u}}}. \tag{4.17}$$

We first wish to understand the structure of the terms $\partial_{\hat{\boldsymbol{u}}}\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}})$. For this, we note that the exact dual variables fulfill

$$\hat{\boldsymbol{u}} = \langle \boldsymbol{u}(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi})\boldsymbol{\varphi}^T\rangle^T =: \boldsymbol{h}(\hat{\boldsymbol{v}}), \tag{4.18}$$

which is why we have the mapping $\hat{\boldsymbol{u}} : \mathbb{R}^{M\times m} \to \mathbb{R}^{M\times m}$, $\hat{\boldsymbol{u}}(\hat{\boldsymbol{v}}) = \boldsymbol{h}(\hat{\boldsymbol{v}})$. Since the solution of the dual problem for a given moment vector is unique, this mapping is bijective and therefore we have an inverse function

$$\hat{\boldsymbol{v}} = \boldsymbol{h}^{-1}(\hat{\boldsymbol{u}}(\hat{\boldsymbol{v}})). \tag{4.19}$$

Now we differentiate both sides w.r.t. $\hat{\boldsymbol{v}}$ to get

$$\boldsymbol{I}_d = \frac{\partial \boldsymbol{h}^{-1}(\hat{\boldsymbol{u}}(\hat{\boldsymbol{v}}))}{\partial \hat{\boldsymbol{u}}}\frac{\partial \hat{\boldsymbol{u}}(\hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}.$$

We multiply with the matrix inverse of $\frac{\partial \hat{\boldsymbol{u}}(\hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}$ to get

$$\left(\frac{\partial \hat{\boldsymbol{u}}(\hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}\right)^{-1} = \frac{\partial \boldsymbol{h}^{-1}(\hat{\boldsymbol{u}}(\hat{\boldsymbol{v}}))}{\partial \hat{\boldsymbol{u}}}.$$

Note that on the left-hand-side we have the inverse of a matrix and on the right-hand-side, we have the inverse of a multi-dimensional function. By rewriting $\boldsymbol{h}^{-1}(\hat{\boldsymbol{u}}(\hat{\boldsymbol{v}}))$ as $\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}})$ and simply computing the term $\frac{\partial \hat{\boldsymbol{u}}(\hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}$ by differentiating (4.18) w.r.t. $\hat{\boldsymbol{v}}$, one obtains

$$\partial_{\hat{\boldsymbol{u}}}\hat{\boldsymbol{v}}(\hat{\boldsymbol{u}}) = \langle \nabla \boldsymbol{u}(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi})\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle^{-T}. \tag{4.20}$$

Now we begin to derive the spectrum of the *One-Shot IPM* iteration (4.15). Note that in its current form this iteration is not really a fixed point iteration, since it uses the time updated dual variables in (4.15b). To obtain a fixed point iteration, we plug the dual iteration step (4.15a) into the moment iteration (4.15b) to obtain

$$\begin{aligned} \boldsymbol{v}_j^{n+1} &= \boldsymbol{d}(\boldsymbol{v}_j^n, \hat{\boldsymbol{u}}_j^n) \quad \text{for all j} \\ \hat{\boldsymbol{u}}_j^{n+1} &= \boldsymbol{c}\left(\boldsymbol{d}(\boldsymbol{v}_{j-1}^n, \hat{\boldsymbol{u}}_{j-1}^n), \boldsymbol{d}(\boldsymbol{v}_j^n, \hat{\boldsymbol{u}}_j^n), \boldsymbol{d}(\boldsymbol{v}_{j+1}^n, \hat{\boldsymbol{u}}_{j+1}^n)\right). \end{aligned}$$

The Jacobian $\boldsymbol{J} \in \mathbb{R}^{2N \cdot N_x \times 2N \cdot N_x}$ has the form

$$\boldsymbol{J} = \begin{pmatrix} \partial_{\boldsymbol{v}} \boldsymbol{d} & \partial_{\hat{\boldsymbol{u}}} \boldsymbol{d} \\ \partial_{\boldsymbol{v}} \boldsymbol{c} & \partial_{\hat{\boldsymbol{u}}} \boldsymbol{c} \end{pmatrix}, \tag{4.21}$$

where each block has entries for all spatial cells. We start by looking at $\partial_{\boldsymbol{v}} \boldsymbol{d}$. For the columns belonging to cell j, we have

$$\begin{aligned} \partial_{\boldsymbol{v}} \boldsymbol{d}(\boldsymbol{v}_j^n, \hat{\boldsymbol{u}}_j^n) &= \boldsymbol{I}_d - \hat{\boldsymbol{H}}(\boldsymbol{v}_j^n)^{-1} \cdot \langle \nabla \boldsymbol{u}(\boldsymbol{\varphi}^T \boldsymbol{v}_j^n) \boldsymbol{\varphi} \boldsymbol{\varphi}^T \rangle^T - \partial_{\boldsymbol{v}} \hat{\boldsymbol{H}}(\boldsymbol{v}_j^n)^{-1} \cdot \left(\langle \boldsymbol{u}(\boldsymbol{\varphi}^T \boldsymbol{v}_j^n) \boldsymbol{\varphi}^T \rangle^T - \hat{\boldsymbol{u}}_j^n \right) \\ &= -\partial_{\boldsymbol{v}} \hat{\boldsymbol{H}}(\boldsymbol{v}_j^n)^{-1} \cdot \left(\langle \boldsymbol{u}(\boldsymbol{\varphi}^T \boldsymbol{v}_j^n) \boldsymbol{\varphi}^T \rangle^T - \hat{\boldsymbol{u}}_j^n \right). \end{aligned}$$

Recall that at the fixed point $(\boldsymbol{v}^*, \hat{\boldsymbol{u}}^*)$, we have $\langle \boldsymbol{u}(\boldsymbol{\varphi}^T \boldsymbol{v}^*) \boldsymbol{\varphi}^T \rangle^T = \hat{\boldsymbol{u}}^*$, hence if $(\boldsymbol{v}_j^n, \hat{\boldsymbol{u}}_j^n)$ belongs to the fixed point one obtains $\partial_{\boldsymbol{v}} \boldsymbol{d} = \boldsymbol{0}$. For the block $\partial_{\hat{\boldsymbol{u}}} \boldsymbol{d}$, we get

$$\partial_{\hat{\boldsymbol{u}}} \boldsymbol{d}(\boldsymbol{v}_j^n, \hat{\boldsymbol{u}}_j^n) = \hat{\boldsymbol{H}}(\boldsymbol{v}_j^n)^{-1},$$

hence $\partial_{\hat{\boldsymbol{u}}} \boldsymbol{d}$ is a block diagonal matrix. Let us now look at $\partial_{\boldsymbol{v}} \boldsymbol{c}$ at a fixed spatial cell j. For a more compact notation, let us denote

$$\boldsymbol{c}_j := \boldsymbol{c}\left(\boldsymbol{d}(\boldsymbol{v}_{j-1}^n, \hat{\boldsymbol{u}}_{j-1}^n), \boldsymbol{d}(\boldsymbol{v}_j^n, \hat{\boldsymbol{u}}_j^n), \boldsymbol{d}(\boldsymbol{v}_{j+1}^n, \hat{\boldsymbol{u}}_{j+1}^n) \right).$$

Then when differentiating with respect to $\boldsymbol{v}_{j-1}^n$ we have

$$\frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{v}_\ell} \frac{\partial \boldsymbol{d}(\boldsymbol{v}_{j-1}^n, \hat{\boldsymbol{u}}_{j-1}^n)}{\partial \boldsymbol{v}} = \boldsymbol{0},$$

since we already showed that by the choice of $\hat{\boldsymbol{H}}(\boldsymbol{v})^{-1}$ the term $\partial_{\boldsymbol{v}} \boldsymbol{d}$ is zero. We can show the same result for all spatial cells and all inputs of $\boldsymbol{c}_j$ analogously, hence $\partial_{\boldsymbol{v}} \boldsymbol{c}_j = \boldsymbol{0}$. For the last block, we have that

$$\frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{v}_\ell} \frac{\partial \boldsymbol{d}(\boldsymbol{v}_{j-1}^n, \hat{\boldsymbol{u}}_{j-1}^n)}{\partial \hat{\boldsymbol{u}}} = \frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{v}_\ell} \hat{\boldsymbol{H}}(\boldsymbol{v}_{j-1}^n)^{-1} = \frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{v}_\ell} \langle \nabla \boldsymbol{u}(\boldsymbol{\varphi}^T \boldsymbol{v}_{j-1}^n) \boldsymbol{\varphi} \boldsymbol{\varphi}^T \rangle^{-T} = \partial_{\hat{\boldsymbol{u}}_\ell} \tilde{\boldsymbol{c}}_j$$

by the choice of $\hat{\boldsymbol{H}}(\boldsymbol{v})^{-1}$ as well as (4.17) and (4.20). We obtain an analogous result for the second and third input. Hence, we have that $\partial_{\hat{\boldsymbol{u}}} \boldsymbol{c} = \nabla_{\hat{\boldsymbol{u}}} \tilde{\boldsymbol{c}}$, which only has eigenvalues between -1 and 1 by the assumption that the classical IPM iteration is contractive. Since $\boldsymbol{J}$ is an upper triangular block matrix, the eigenvalues are given by $\lambda(\partial_{\boldsymbol{v}} \boldsymbol{d}) = 0$ and $\lambda(\partial_{\hat{\boldsymbol{u}}} \boldsymbol{c}) \in (-1, 1)$, hence the One-Shot IPM is contractive around its fixed point. □

Theorem 18. *With the assumptions from Theorem 17, the One-Shot IPM converges locally, i.e. there exists a $\delta > 0$ s.t. for all starting points $(\boldsymbol{v}^0, \hat{\boldsymbol{u}}^0) \in B_\delta(\boldsymbol{v}^*, \hat{\boldsymbol{u}}^*)$ we have*

$$\|(\boldsymbol{v}^n, \hat{\boldsymbol{u}}^n) - (\boldsymbol{v}^*, \hat{\boldsymbol{u}}^*)\| \to 0 \qquad \text{for } n \to \infty.$$

Proof. By Theorem 17, the One-Shot scheme is contractive at its fixed point. Since we assumed convergence of the classical IPM scheme, we can conclude that all entries in the Jacobian $\boldsymbol{J}$ are continuous functions. Furthermore, the determinant of $\tilde{\boldsymbol{J}} :=$

$\boldsymbol{J} - \lambda \boldsymbol{I}_d$ is a polynomial of continuous functions, since

$$\det(\tilde{\boldsymbol{J}}) = \sum_{\sigma} \operatorname{sgn}(\sigma) \prod_{i=1}^{2N_x N} \tilde{J}_{\sigma(i),i}.$$

Since the roots of a polynomial vary continuously with its coefficients, the eigenvalues of $\boldsymbol{J}$ are continuous w.r.t $(\boldsymbol{v}, \hat{\boldsymbol{u}})$. Hence there exists an open ball with radius δ around the fixed point in which the eigenvalues remain in the interval $(-1, 1)$. □

4.3 Adaptivity

The following section presents the adaptivity strategy used in this work. Since random hyperbolic problems generally experience shocks in a small portion of the space-time domain, the idea is to perform arising computations on a high accuracy level in this small area, while keeping a low level of accuracy in the remainder. The idea is to automatically select the lowest order moment capable of approximating the solution with given accuracy, i.e. the same error is obtained while using a significantly reduced number of unknowns in most parts of the computational domain and thus boost the performance of intrusive methods. Note that in this work, we only adaptively change the truncation order of our approximation as well as the quadrature order of our nodal discretization, while keeping a fixed number of spatial cells.

In the following, we discuss the building blocks of the IPM method for refinement levels $\ell = 1, \cdots, N_{\text{ad}}$, where level 1 uses the coarsest discretization and level N_{ad} uses the finest discretization of the uncertain domain. At a given refinement level ℓ, the total degree of the basis function is given by N_ℓ with a corresponding number of moments M_ℓ. The number of quadrature points at level ℓ is denoted by Q_ℓ. To determine the refinement level of a given moment vector $\hat{\boldsymbol{u}}$ we choose techniques used in discontinuous Galerkin (DG) methods. Adaptivity is a common strategy to accelerate this class of methods and several indicators to determine the smoothness of the solution exist. Translating the idea of the so called discontinuity sensor which has been defined in [106] to uncertainty quantification, we define the polynomial approximation at refinement level ℓ as

$$\tilde{\boldsymbol{u}}_\ell := \sum_{|i| \leq N_\ell} \hat{\boldsymbol{u}}_i \varphi_i.$$

Now the indicator for a moment vector at level ℓ is defined as

$$\boldsymbol{S}_\ell := \frac{\langle (\tilde{\boldsymbol{u}}_\ell - \tilde{\boldsymbol{u}}_{\ell-1})^2 \rangle}{\langle \tilde{\boldsymbol{u}}_\ell^2 \rangle}, \tag{4.22}$$

where divisions and multiplications are performed element-wise. Note that a similar indicator has been used in [65] for intrusive methods in uncertainty quantification. In this work, we use the first entry in $\boldsymbol{S}_\ell$ to determine the refinement level, i.e. in the case of gas dynamics, the regularity of the density is chosen to indicate an adequate refinement level. If the moment vector in a given cell and at a certain timestep is initially at refinement level ℓ, this level is kept if the error indicator (4.22) lies in the interval $I_\delta := [\delta_-, \delta_+]$. Here $\delta_\pm$ are user determined parameters. If the indicator is smaller than δ_-, the refinement level is decreased to the next lower level, if it lies above δ_+, it is increased to the next higher level.

Now we need to specify how the different building blocks of IPM can be modified to work with varying truncation orders in different cells. Let us first add dimensions to the notation of the dual iteration function $\boldsymbol{d}$, which has been defined in (4.5). Now, we have $\boldsymbol{d}_\ell : \mathbb{R}^{M_\ell \times m} \times \mathbb{R}^{M_\ell \times m} \to \mathbb{R}^{M_\ell \times m}$, given by

$$\boldsymbol{d}_\ell(\boldsymbol{v}, \hat{\boldsymbol{u}}) := \boldsymbol{v} - \hat{\boldsymbol{H}}_\ell^{-1}(\boldsymbol{v}) \cdot \left(\langle \boldsymbol{u}(\boldsymbol{v}^T \boldsymbol{\varphi}_\ell) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T - \hat{\boldsymbol{u}} \right), \tag{4.23}$$

where $\boldsymbol{\varphi}_\ell \in \mathbb{R}^{M_\ell}$ collects all basis functions with total degree smaller or equal to N_ℓ. The Hessian $\hat{\boldsymbol{H}}_\ell$ is given by

$$\hat{\boldsymbol{H}}_\ell(\boldsymbol{v}) := \langle \nabla \boldsymbol{u}(\boldsymbol{v}^T \boldsymbol{\varphi}_\ell) \otimes \boldsymbol{\varphi}_\ell \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T.$$

An adaptive version of the moment iteration (4.8) is denoted by $\boldsymbol{c}_\ell^{\boldsymbol{\ell}'} : \mathbb{R}^{M_{\ell_1'} \times m} \times \mathbb{R}^{M_{\ell_2'} \times m} \times \mathbb{R}^{M_{\ell_3'} \times m} \to \mathbb{R}^{M_\ell \times m}$ and given by

$$\begin{aligned} \boldsymbol{c}_\ell^{\boldsymbol{\ell}'}(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3) :=& \langle \boldsymbol{u}(\boldsymbol{v}_2^T \boldsymbol{\varphi}_{\ell_2'}) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T \\ &- \frac{\Delta t}{\Delta x} \left(\langle \boldsymbol{g}(\boldsymbol{u}(\boldsymbol{v}_2^T \boldsymbol{\varphi}_{\ell_2'}), \boldsymbol{u}(\boldsymbol{v}_3^T \boldsymbol{\varphi}_{\ell_3'})) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T - \langle \boldsymbol{g}(\boldsymbol{u}(\boldsymbol{v}_1^T \boldsymbol{\varphi}_{\ell_1'}), \boldsymbol{u}(\boldsymbol{v}_2^T \boldsymbol{\varphi}_{\ell_2'})) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T \right). \end{aligned} \tag{4.24}$$

Hence, the index vector $\boldsymbol{\ell}' \in \mathbb{N}^3$ denotes the refinement levels of the stencil cells, which are used to compute the time updated moment vector at level ℓ.

The strategy now is to perform the dual update for a set of moment vectors $\hat{\boldsymbol{u}}_j^n$ at refinement levels ℓ_j^n for $j = 1, \cdots, N_x$. Thus, the dual iteration makes use of the iteration function (4.23) at refinement level ℓ_j^n. After that, the refinement level at the next time step ℓ_j^{n+1} is determined by making use of the smoothness indicator (4.22). The moment update then computes the moments at the time updated refinement level ℓ_j^{n+1}, utilizing the the dual states at the old refinement levels $\boldsymbol{\ell}' = (\ell_{j-1}^n, \ell_j^n, \ell_{j+1}^n)^T$.

Note that we use nested quadrature rules, which facilitate the task of evaluating the quadrature in the moment update (4.24). Assume that we want to compute the moment update in cell j with refinement level ℓ_j where a neighboring cell $j-1$ has refinement level ℓ_{j-1}. Now if $\ell_{j-1} \geq \ell_j$, the solution of cell $j-1$ is known at all Q_ℓ quadrature points, hence the integral inside the moment update can be computed. Vice versa, if $\ell_{j-1} \leq \ell_j$, we need to evaluate the neighboring cell at the finer quadrature level ℓ_j. Except from this, increasing or decreasing the refinement level does not lead to additional costs.

The IPM algorithm with adaptivity results in Algorithm 5, which uses Algorithm 6 to determine the refinement level ℓ_j^n, i.e. the refinement level of spatial cell j at time step n. Algorithm 6 computes the refinement indicator $\boldsymbol{S}$ according to (4.22) and checks the size of its first component, i.e. the density (and not momentum or energy) is used as an indicator. The parameters $\delta_\pm$ are user-determined parameters which need to be chosen before running the program.

Algorithm 5 Adaptive IPM implementation

for $j = 0$ to $N_x + 1$ **do**
  $\ell_j^0 \leftarrow$ choose initial refinement level
  $\boldsymbol{u}_j^0 \leftarrow \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x, \cdot) \boldsymbol{\varphi}_{\ell_j^0} \rangle_{Q_{\ell_j^0}} dx$
for $n = 0$ to N_t **do**
  for $j = 0$ to $N_x + 1$ **do**
    $\boldsymbol{v}_j^{(0)} \leftarrow \hat{\boldsymbol{v}}_j^n$
    while (4.7) is violated **do**
      $\boldsymbol{v}_j^{(l+1)} \leftarrow \boldsymbol{d}_{\ell_j^n}(\boldsymbol{v}_j^{(l)}; \hat{\boldsymbol{u}}_j^n)$
      $l \leftarrow l + 1$
    $\hat{\boldsymbol{v}}_j^{n+1} \leftarrow \boldsymbol{v}_j^{(l)}$
    $\ell_j^{n+1} \leftarrow \mathrm{DetermineRefinementLevel}\left(\hat{\boldsymbol{v}}_j^{n+1}, \ell_j^n\right)$
  for $j = 1$ to N_x **do**
    $\boldsymbol{\ell}' \leftarrow (\ell_{j-1}^n, \ell_j^n, \ell_{j+1}^n)^T$
    $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \boldsymbol{c}_{\ell_j^{n+1}}^{\boldsymbol{\ell}'}(\hat{\boldsymbol{v}}_{j-1}^{n+1}, \hat{\boldsymbol{v}}_j^{n+1}, \hat{\boldsymbol{v}}_{j+1}^{n+1})$

Algorithm 6 Algorithm to determine refinement level.

procedure DETERMINEREFINEMENTLEVEL($\hat{\boldsymbol{v}}, \ell$)
  $\hat{\boldsymbol{u}} \leftarrow \langle \boldsymbol{u}(\hat{\boldsymbol{v}}^T \boldsymbol{\varphi}_\ell) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T$
  $\tilde{\boldsymbol{u}}_\ell \leftarrow \sum_{|i| \le N_\ell} \hat{\boldsymbol{u}}_i \boldsymbol{\varphi}_i$
  $\tilde{\boldsymbol{u}}_{\ell-1} \leftarrow \sum_{|i| \le N_{\ell-1}} \hat{\boldsymbol{u}}_i \boldsymbol{\varphi}_i$
  $\boldsymbol{S} \leftarrow \frac{\langle (\tilde{\boldsymbol{u}}_\ell - \tilde{\boldsymbol{u}}_{\ell-1})^2 \rangle_{Q_\ell}}{\langle \tilde{\boldsymbol{u}}_\ell^2 \rangle_{Q_\ell}}$
  if $S_0 < \delta_-$ **then**
    return $\ell - 1$
  else if $S_0 > \delta_+$ **then**
    return $\ell + 1$
  else
    return ℓ

Adaptivity can be used for intrusive methods in general as well as for steady and unsteady problems. In the case of steady problems, we can make use of a strategy, which we call *refinement retardation*. Recall that the convergence to an admissible steady state solution is expensive and a high accuracy and desirable solution properties are only required at the end of this iteration process. Hence, we propose to iteratively increase the maximal refinement level whenever the residual (4.14) lies below a certain tolerance ε. For a given set of maximal refinement levels ℓ_l^* and a set of tolerances ε_l^* at which the refinement level must be increased, we can now perform a large amount of the required iterations on a lower, but cheaper refinement level. The same strategy can be applied for One-Shot IPM. In this case, the algorithm is given by Algorithm 7.

Algorithm 7 Adaptive One-Shot IPM implementation with refinement retardation

1: **for** $j = 0$ to $N_x + 1$ **do**
2: $\quad \boldsymbol{u}_j^0 \leftarrow \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x, \cdot) \boldsymbol{\varphi} \rangle_Q dx$
3: **while** (4.14) is violated **do**
4: $\quad$ **for** $j = 1$ to N_x **do**
5: $\quad\quad \boldsymbol{v}_j^{n+1} \leftarrow \boldsymbol{d}_{\ell_j^n}(\boldsymbol{v}_j^n; \hat{\boldsymbol{u}}_j^n)$
6: $\quad\quad \ell_j^{n+1} \leftarrow \max\{\text{DetermineRefinementLevel}\left(\boldsymbol{v}_j^{n+1}, \ell_j^n\right), \ell_l^*\}$
7: $\quad$ **for** $j = 1$ to N_x **do**
8: $\quad\quad \boldsymbol{\ell}' \leftarrow (\ell_{j-1}^n, \ell_j^n, \ell_{j+1}^n)^T$
9: $\quad\quad \hat{\boldsymbol{u}}_j^{n+1} \leftarrow \boldsymbol{c}_{\ell_j^{n+1}}^{\boldsymbol{\ell}'}(\boldsymbol{v}_{j-1}^{n+1}, \boldsymbol{v}_j^{n+1}, \boldsymbol{v}_{j+1}^{n+1})$
10: $\quad n \leftarrow n + 1$
11: $\quad$ **if** the residual (4.14) lies below ε_l^* **then**
12: $\quad\quad l \leftarrow l + 1$

4.4 Parallelization and implementation

It remains to discuss the parallelization of the presented algorithms. In order to minimize the parallelization overhead, our goal is to minimize the communication between processors. Note that the dual problem (line 8 in Algorithm 5 and line 5 in Algorithm 7) does not require communication, i.e. it suffices to distribute the spatial cells between processors. In contrast to that, the finite volume update (line 14 in Algorithm 5 and line 8 in Algorithm 7) requires communication, since values at neighboring cells need to be evaluated. Hence, distributing the spatial mesh between processors will yield communication overhead since data needs to be sent whenever a stencil cell lies on a different processor. Therefore, we choose to parallelize the quadrature points, which minimizes the computational time spend on communication. As mentioned in Section 2.3.3, the kinetic flux first computes the solution at stencil cells for all quadrature points. I.e. we determine $\boldsymbol{u}_k^{(j)} \in \mathbb{R}^m$ and the corresponding stencil cells for $k = 1, \dots, Q$ by

$$\boldsymbol{u}_k^{(j-1)} := \boldsymbol{u}(\boldsymbol{v}_{j-1}^T \boldsymbol{\varphi}_{\ell_1'}(\boldsymbol{\xi}_k)), \quad \boldsymbol{u}_k^{(j)} := \boldsymbol{u}(\boldsymbol{v}_j^T \boldsymbol{\varphi}_{\ell_2'}(\boldsymbol{\xi}_k)), \quad \boldsymbol{u}_k^{(j+1)} := \boldsymbol{u}(\boldsymbol{v}_{j+1}^T \boldsymbol{\varphi}_{\ell_3'}(\boldsymbol{\xi}_k)).$$

Thus, the finite volume update function (4.24) can be written as

$$\boldsymbol{c}_\ell^{\boldsymbol{\ell}'}(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3) = \sum_{k=1}^{Q} w_k \left[\boldsymbol{u}_k^{(j)} - \frac{\Delta t}{\Delta x} \left(\boldsymbol{f}^*(\boldsymbol{u}_k^{(j)}, \boldsymbol{u}_k^{(j+1)}) - \boldsymbol{f}^*(\boldsymbol{u}_k^{(j-1)}, \boldsymbol{u}_k^{(j)}) \right) \right] \boldsymbol{\varphi}_\ell(\boldsymbol{\xi}_k)^T. \tag{4.25}$$

Instead of distributing the spatial mesh on the different processors, we now distribute the quadrature set, i.e. the sum in (4.25) can be computed in parallel. Now, after having performed the dual update, the dual variables are send to all processors. With these variables, each processor computes the solution on its portion of the quadrature set and then computes its part of the sum in (4.25) on all spacial cells. All parts from the different processors are then added together and the full time-updated moments are distributed to all processors. From here, the dual update can again be performed. The standard IPM Algorithm 3 and One-Shot IPM Algorithm 4

use this parallelization strategy accordingly. Again, we point out that stochastic-Galerkin is a variant of IPM, i.e. all presented techniques for IPM can also be used for SG. The SC algorithm that we use to compare intrusive with non-intrusive methods uses a given deterministic solver as a black box. Here, we distribute the quadrature set between all processors. Note that both, SC and IPM are based on the same deterministic solver, i.e. we use the same deterministic numerical flux $\boldsymbol{f}^*$. To our best knowledge, this allows a fair comparison of the different intrusive and non-intrusive techniques. In the following section, we will study the convergence of the expectation and variance error in pseudo-time. Recording this error is straight forward with intrusive methods, however non-intrusive methods only yield expectation value and variance at the final, steady state solution. Therefore, to record the error for SC, we have implemented a collocation code, which couples all quadrature points in each time step, allowing the computation of the error in pseudo-time. Since this adds additional communication costs, we do not use the run time of this method, but instead make use of the run times from the black-box SC code. Thereby, we are able to record the convergence of expectation values and variances in pseudo-time for the non-intrusive SC method without including additional communication costs.

4.5 Results

4.5.1 Euler 2D with a one-dimensional uncertainty

We start by quantifying the effects of an uncertain angle of attack $\phi \sim U(0.75, 1.75)$ for a NACA0012 airfoil computed with different methods. The Euler equations in two dimensions are given by

$$\partial_t \begin{pmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho E \end{pmatrix} + \partial_{x_1} \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ v_1(\rho E + p) \end{pmatrix} + \partial_{x_2} \begin{pmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ v_2(\rho E + p) \end{pmatrix} = \mathbf{0}. \tag{4.26}$$

These equations determine the time evolution of the conserved variables $(\rho, \rho \boldsymbol{v}, \rho E)$, i.e. density, momentum and energy. In Section 4.5.1 and Section 4.5.2, we are interested in steady state problems, i.e. the time t plays the role of a pseudo-time and initial conditions can be chosen freely. As described in Section 4.2, the iterations in pseudo-time are performed until the solution fulfills (4.14). Section 4.5.3 investigate the unsteady Euler equations and we specify initial conditions and end time t_{end} in this section. A closure for the pressure p is given by

$$p = (\gamma - 1)\rho \left(E - \frac{1}{2}(v_1^2 + v_2^2) \right). \tag{4.27}$$

Here, the heat capacity ratio γ is chosen to be 1.4. In our first test case, we are interested in the flow around an airfoil. The spatial flow domain D is a circular domain with a diameter of 40 meters. In the center of this domain, we place a NACA0012 airfoil with a length of 1 meter. The spatial domain D consists of two boundaries: 1) The boundary of the circular domain, which we denote as *farfield boundary* Γ_∞ and 2) the boundary of the airfoil, which we denote as *airfoil boundary* Γ_0. At the airfoil boundary, we use the Euler slip condition $\boldsymbol{v}^T \boldsymbol{n} = 0$, where $\boldsymbol{n}$ denotes the surface normal. At the far field boundary Γ_∞, we assume a constant flow (i.e. Dirichlet boundary conditions) with a given Mach number $\text{Ma}_\infty = 0.8$, pressure $\text{p}_\infty = 101\,325$

Pa and a temperature of $T_\infty = 273.15$ K. The far field velocity enters the spatial domain at the far field with an angle of attack ϕ. We assume that ϕ is uncertain and that it is uniformly distributed in the interval of $[0.75, 1.75]$ degrees, i.e. we choose $\phi(\xi) = 1.25 + 0.5\xi$ where $\xi \sim U(-1, 1)$. As commonly done, the initial condition is equal to the far field boundary values. Consequently, the wall condition at the airfoil boundary Γ_0 is violated initially and will correct the flow solution.

The spatial mesh is composed of a coarsely discretized far field and a finely resolved region around the airfoil, since we are interested in the flow solution at the airfoil. Altogether, the mesh consists of 22361 triangular elements.

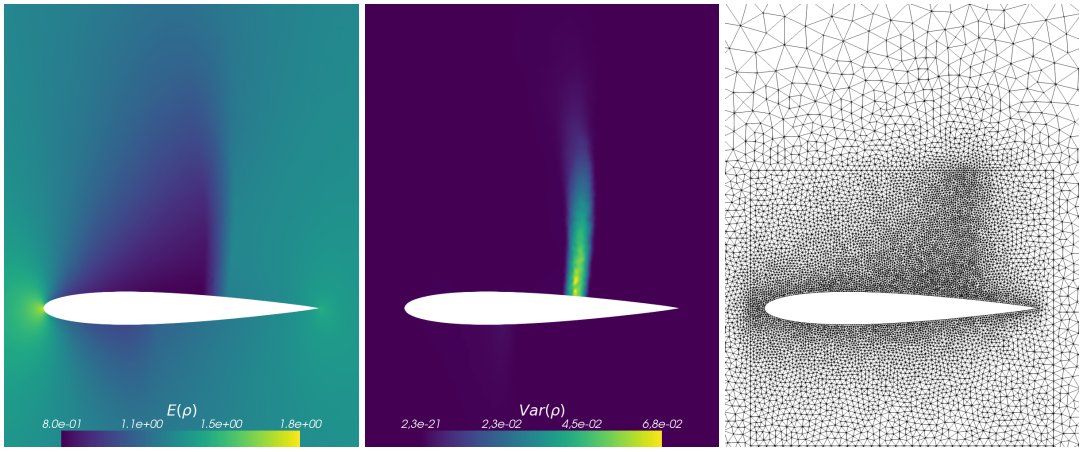


FIGURE 4.1: Reference solution $\mathrm{E}[\rho]$ and $\mathrm{Var}[\rho]$ and the mesh close to the airfoil which is used in the computation of all presented methods.

The aim is to quantify the effects arising from the one-dimensional uncertainty ξ and to investigate its effects on the solution with different methods. To be able to measure the quality of the obtained solutions, we compute a reference solution using Stochastic Collocation with 100 Gauss-Legendre quadrature points, which can be found in Figure 4.1. In the following, we investigate the L^2-error of the variance and the expectation value. The L^2-error of the discrete quantity $\boldsymbol{e}_\Delta = (\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{N_x})^T$, where $\boldsymbol{e}_j$ is the cell average of the quantity $\boldsymbol{e}$ in spatial cell j, is denoted by

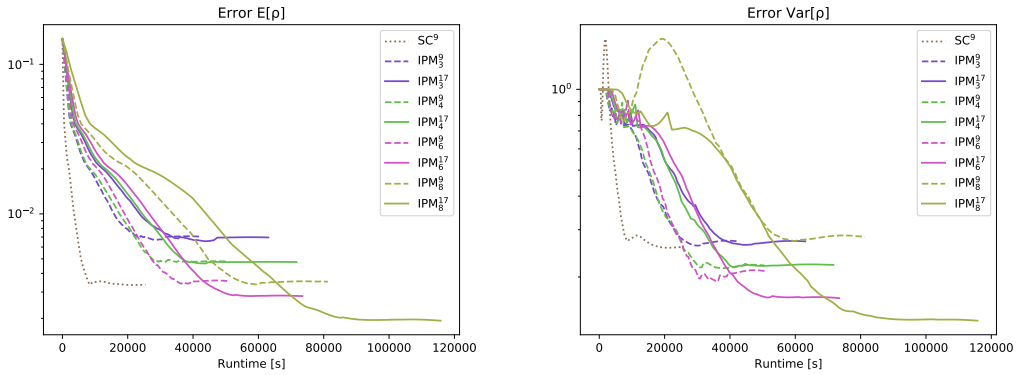
$$\|\boldsymbol{e}_\Delta\|_\Delta := \sqrt{\sum_{j=1}^{N_x} \Delta x_j \boldsymbol{e}_j^2}\,.$$

Hence, when denoting the reference solution by $\boldsymbol{u}_\Delta$ and the moments obtained with the numerical method by $\hat{\boldsymbol{u}}_\Delta$, we investigate the relative L^2 error

$$\frac{\|\mathrm{E}[\boldsymbol{u}_\Delta] - \mathrm{E}[\mathcal{U}(\hat{\boldsymbol{u}}_\Delta)]\|_\Delta}{\|\mathrm{E}[\boldsymbol{u}_\Delta]\|_\Delta} \quad \text{and} \quad \frac{\|\mathrm{Var}[\boldsymbol{u}_\Delta] - \mathrm{Var}[\mathcal{U}(\hat{\boldsymbol{u}}_\Delta)]\|_\Delta}{\|\mathrm{Var}[\boldsymbol{u}_\Delta]\|_\Delta}. \tag{4.28}$$

The error is computed inside a box of one meter height and 1.1 meters length around the airfoil to prevent small fluctuations in the coarsely discretized far field from affecting the error.

The quantities of interests are now computed with the different methods. All methods in this section have been computed using five MPI threads. For more information on the chosen entropy and the resulting solution ansatz for IPM, see [71, Appendix B].



(A) (B)

FIGURE 4.2: Relative L^2-error (4.28) with different quadrature levels for IPM in comparison with SC. The subscript denotes the moment order, the superscript denotes the number of Clenshaw-Curtis quadrature points.

Recall that the numerical flux (4.3) uses a quadrature rule to approximate integrals. We start by investigating the effects this quadrature has on the solution accuracy. For this, we run the IPM method with a moment order ranging from 3 to 7 using a Clenshaw-Curtis quadrature rule with level three (i.e. 9 quadrature points) and level four (i.e. 17 quadrature points). A comparison of the error obtained with these two quadrature levels is given in Figure 4.2. To denote the number of quadrature points, we use a superscript and the moment order is denoted by a subscript. We observe the following:

- When for example comparing the error obtained with IPM_8^9 and IPM_8^{17} (i.e. the polynomial order is 8, meaning that 9 moments are used and the computation is done using 9 and 17 quadrature points) it can be seen that the error stagnates when the chosen quadrature is not sufficiently accurate. Hence, the accuracy level is dominated by aliasing effects that result from an inaccurate quadrature rule in the numerical flux and not the truncation error of the moment system.

- If the truncation order is sufficiently small, both quadrature levels yield the same accuracy. We observe this behavior for the expectation value until a truncation order of $N = 5$ and for the variance for $N = 4$. Hence, the variance is more sensitive to aliasing errors.

- Figure 4.2A reveals that the IPM error of $\mathrm{E}[\rho]$ with 9 quadrature points stagnates at the error level of SC^9, i.e. the aliasing error heavily affects the accuracy. This behavior becomes more dominant when looking at the variance error in Figure 4.2B. Here, IPM_8^{17} yields a significantly improved result compared to IPM_8^9. Again, the IPM_8^9 result stagnates at the SC^9 accuracy level, which can be reached with IPM when only using four moments (combined with a sufficiently accurate quadrature level). Hence, the number of moments needed for IPM to obtain a certain variance error is significantly smaller than the number of quadrature points needed for SC. This result can be observed throughout the numerical experiments of this work. Especially for high-dimensional problems, this potentially decreases the number of unknowns to reach a certain accuracy level significantly.

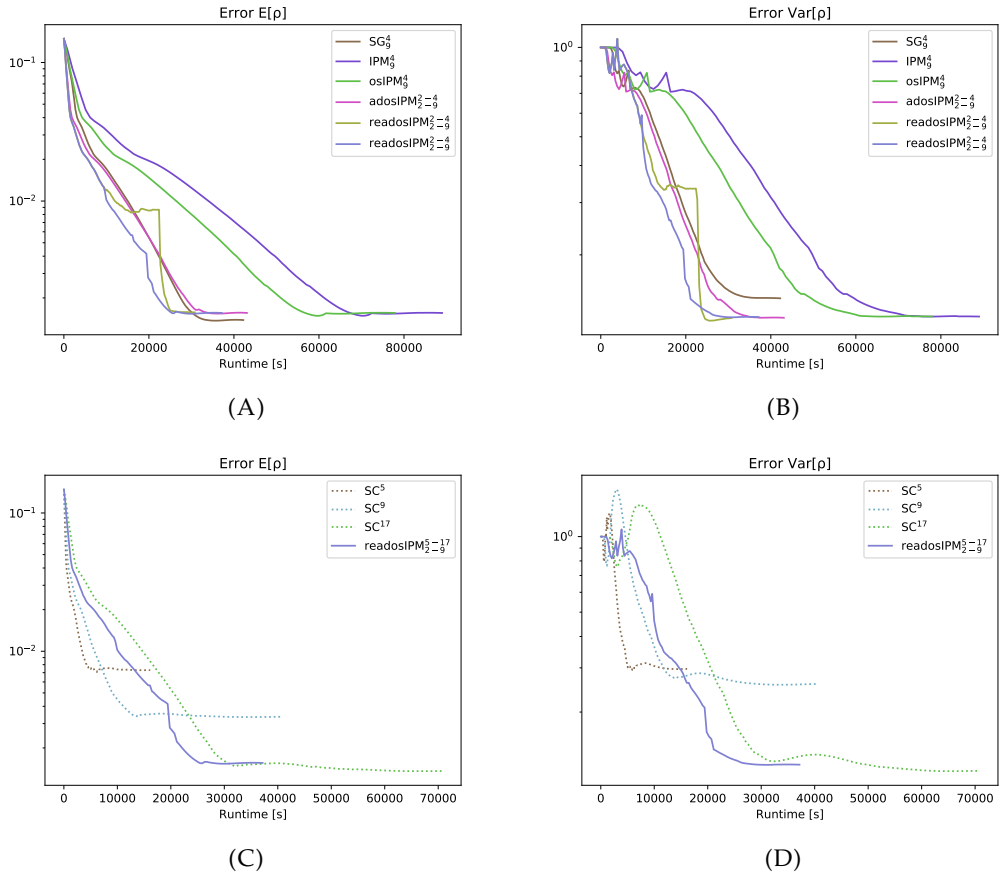


FIGURE 4.3: Comparison of the relative L^2-error (4.28) for the density for IPM related methods (first row) and of the best performing IPM method in comparison with SC (second row). All intrusive methods are iterated until they fulfill condition (4.14) with $\varepsilon = 6 \cdot 10^{-6}$, whereas all non-intrusive methods converge to a residual $\varepsilon = 1 \cdot 10^{-7}$. All computations are performed with 5 MPI threads. The subscript denotes the range of the moment order, the superscript denotes the range of the number of Clenshaw-Curtis quadrature points from the coarsest to finest refinement level.

Let us now compare results obtained with stochastic-Galerkin and IPM as well as its proposed acceleration techniques at a fixed moment order 9. Note, that since IPM generalizes SG, all proposed techniques can be used for stochastic-Galerkin as well. All adaptive methods use gPC polynomials of order 2 to 9 (i.e. 3 to 10 moments). Order 2 uses 5 quadrature points, orders 3 to 6 use 9 quadrature points and orders 8 and 9 use 17 quadrature points. When the smoothness indicator (4.22) lies below $\delta_- = 2 \cdot 10^{-4}$, the adaptive methods decrease the truncation order, if it lies above $\delta_+ = 2 \cdot 10^{-5}$ the truncation order is increased. The remaining methods have been computed with 17 quadrature points. The iteration in pseudo-time is performed until the expectation value of the density fulfills the stopping criterion (4.14) with $\varepsilon = 6 \cdot 10^{-6}$, however it can be seen that the error saturates already at a bigger residual.

First, let us mention that the adaptive SG method fails, since it yields negative densities during the iteration. The standard SG method however preserves positivity of mass, energy and pressure. The change of the relative L^2-error during the iteration to the steady state has been recorded in Figure 4.3A for the expectation value

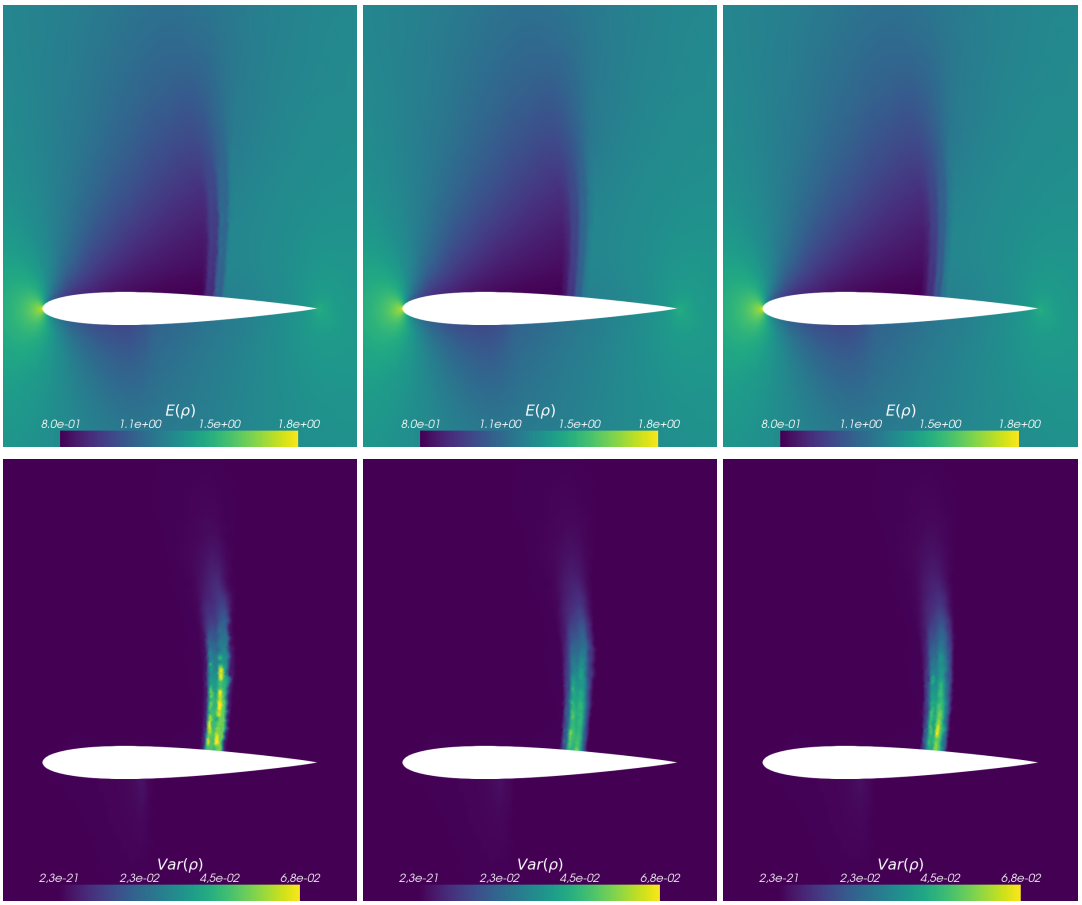


FIGURE 4.4: $\mathrm{E}[\rho]$ and $\mathrm{Var}[\rho]$ computed with SC^5, SG_4, IPM_4 (from left to right). Compare to the reference solution shown in Figure 4.1.

and in Figure 4.3B for the variance. When comparing intrusive methods without acceleration techniques as well as SC, the following properties emerge:

- Compared to IPM, stochastic-Galerkin comes at a significantly reduced runtime, meaning that the IPM optimization problem requires a significant computational effort.
- For the expectation value SG_9 shows a smaller error compared to IPM_9, while for the variance, we see the opposite, i.e. IPM yields a better solution approximation than SG.

The proposed acceleration techniques show the following behavior:

- The One-Shot IPM (osIPM) method proposed in Section 4.2 reduces the runtime while yielding the same error as the classical IPM method.
- When using adaptivity (see Algorithm 5) in combination with the One-Shot idea, the method is denoted by *adaptive One-Shot IPM* (adosIPM). This method reaches the steady state IPM solution with a faster runtime than SG.
- The idea of refinement retardation combined with adosIPM (see Algorithm 7) is denoted by *retardation adosIPM* (readosIPM), which further decreases runtime. Here, we use two different strategies to increase the accuracy: First, we

steadily increase the maximal truncation order when the residual approaches zero. To determine residual values for a given set of truncation orders $2, 4, 5$ and 8, we study at which residual level the IPM method reaches a saturated error level for each truncation order. The residual values are then determined to be $6 \cdot 10^{-5}, 3 \cdot 10^{-5}, 2.2 \cdot 10^{-5}$ and $2 \cdot 10^{-5}$. The second, straight forward strategy converges the solution on a low truncation order of 2 to a residual of 10^{-5} and then switches to a maximal truncation order of 9. Strategy 1 is depicted in purple, Strategy 2 is depicted in yellow. It can be seen that both approaches reach the IPM_9 error for the same run time. Hence, we deduce that a naive choice of the refinement retardation strategy suffices to yield a satisfactory behavior.

Let us now compare the results from intrusive methods with those of SC. For every quadrature point, SC iterates the solution until the density lies below a threshold of $\varepsilon = 1 \cdot 10^{-7}$. Recording the error of intrusive methods during the iteration is straight forward. To record the error of SC, we couple all quadrature points after each iteration to evaluate the expectation value and variance, which destroys the black-box nature and results in additional costs. The collocation runtimes, that are depicted on the x-axis in Figures 4.2, 4.3C, 4.3D and 4.6, are however rescaled to the runtimes that we achieve when running the collocation code in its original, black-box framework without recording the error. Our Stochastic Collocation computations use Clenshaw-Curtis quadrature levels $2, 3$ and 4, i.e. $5, 9$ and 17 quadrature points. The comparison of intrusive methods with non-intrusive methods shows the following:

- SC requires a smaller residual to converge to a steady state solution (we converge the results to $\varepsilon = 1 \cdot 10^{-7}$ compared to $\varepsilon = 6 \cdot 10^{-6}$ for intrusive methods).
- Again, intrusive methods yield improved solutions compared to SC with the same number of unknowns. Actually, the error obtained with 17 unknowns when using SC is comparable with the error obtained with 10 unknowns when using intrusive methods.

Let us finally take a look at the expectation value and variance computed with different methods. All results are depicted for a zoomed view around the airfoil. Figure 4.4 shows the expectation value (first row) and variance (second row) computed with 5 quadrature points for SC and 5 moments for SG and IPM. One can observe the following

- All methods yield non-physical step-like profiles of the expectation value and variance along the airfoil. This effect can be observed in various settings [77, 70, 110, 8, 31] and stems from Gibb's phenomena in the polynomial description of the uncertainty, either in the gPC polynomials of intrusive methods or the Lagrange polynomials used in collocation techniques.
- The jump position of the intrusive solution profiles capture the exact behavior more accurately.

The readosIPM results as well as the corresponding refinement levels are depicted in Figure 4.5. One observes that the solution no longer shows the previously observed discontinuous profile and yield a satisfactory agreement with the reference solution depicted in Figure 4.1. The refinement level shows that away from the airfoil, a refinement level of 0 (i.e. a truncation order of 2) suffices to yield the IPM_9 solution. The region with a high variance requires a refinement level of 7 (truncation order 9).

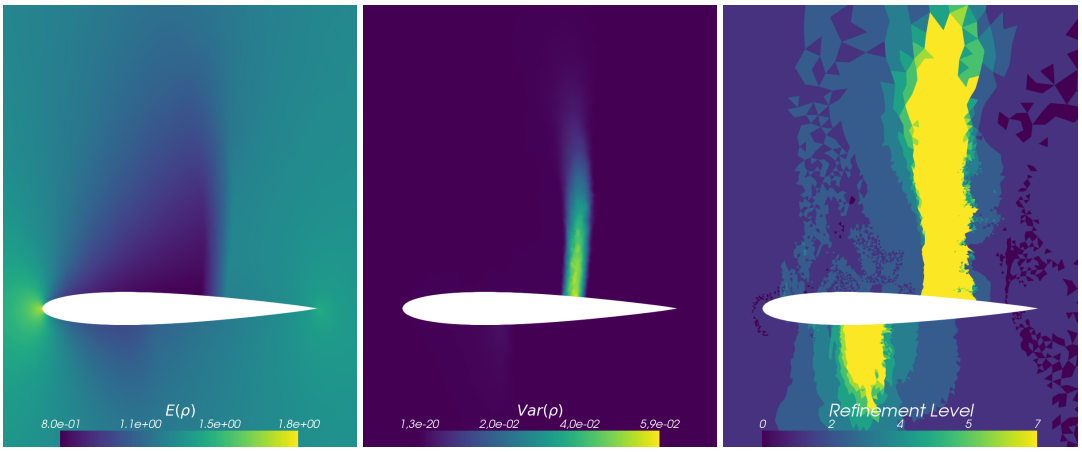


FIGURE 4.5: E[ρ], Var[ρ] and refinement level for readosIPM$_{2-9}$.

4.5.2 Euler 2D with a two-dimensional uncertainty

In the following, we assume a fixed angle of attack with $\phi = 1.25$ degrees and study the effect of two sources of uncertainties, namely the farfield pressure and Mach number. The farfield pressure is $p \sim U(100\,325, 102\,325)$ Pa and the Mach number is $Ma \sim U(0.775, 0.825)$. Since this problem only has a two-dimensional uncertainty, we use a tensorized quadrature set, which in our experiments proved to be more efficient than a sparse grid quadrature. For SC, we use quadrature sets with 5^2, 9^2 and 17^2 quadrature nodes and compare against SG with moments up to total degree 9 as well as adaptive IPM with refinement retardation (with and without the One-Shot strategy). The IPM method uses moment orders ranging from 1 to 9 adaptively with refinement barriers $\delta_- = 1 \cdot 10^{-4}$ and $\delta_+ = 1 \cdot 10^{-5}$. The refinement retardation allows the truncation order to have a maximal total degree of 1 until a residual of $\varepsilon = 1.5 \cdot 10^{-5}$ and then increases the truncation order by one when the residual is reduced by an amount of $5 \cdot 10^{-6}$. Hence, the maximal truncation order of 9 is reached when the residual is below $\varepsilon = 7 \cdot 10^{-6}$. Again, the refinement strategy let to negative densities for SG resulting in a failure of the method. The error during the iteration has been recorded in Figure 4.6. To determine the error, we used Stochastic Collocation with a 50 by 50 Gauss-Legendre quadrature rule. As in the one-dimensional NACA testcase, the acceleration techniques lead to a heavily reduced runtime of the IPM method. Furthermore, the error obtained with the intrusive methods requires a total degree of $M = 9$, i.e. $N = 55$ moments to reach the error level of SC with 17 quadrature points per dimension i.e. $17^2 = 289$ collocation points. Note that this effect has also been observed in the one-dimensional case, however now for multi-dimensional problems, the reduced number of unknowns required for intrusive methods to obtain a certain error becomes more apparent. This shows one promising characteristic of intrusive techniques and points to their applicability for higher-dimensional problems. In contrast to before, the SG error level is smaller than the IPM error for both, the expectation value and variance. Furthermore, when comparing IPM with and without One-Shot, one can observe that the effect of this acceleration strategy weighs in more heavily than it did for the one-dimensional case. This behavior is expected, since every iterate of the optimization problem becomes more expensive when the dimension is increased. This means, using a method with less iterations for every computation of the dual variables heavily reduces computational costs, which is why we consider the One-Shot strategy to be an effective

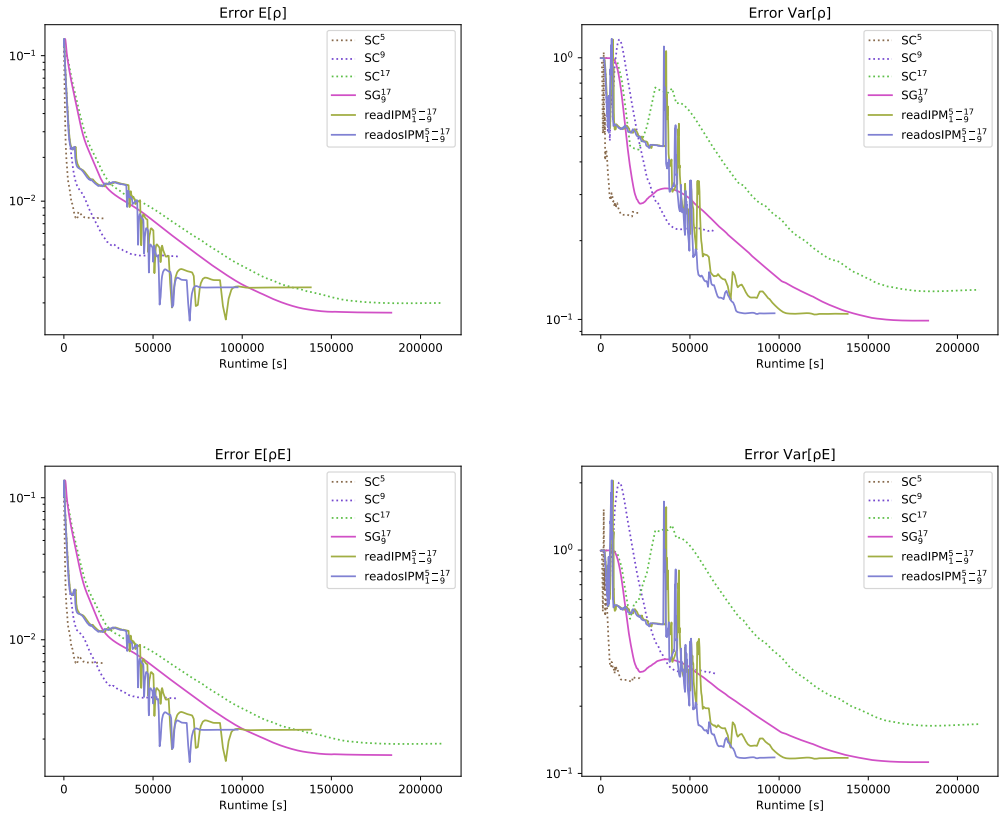


FIGURE 4.6: Relative L^2-error (4.28) for density and energy computed with 16 MPI threads. All intrusive methods are converged to a residual (4.14) with $\varepsilon = 6 \cdot 10^{-6}$ and all non-intrusive methods are converged to a residual of $\varepsilon = 1 \cdot 10^{-6}$. The subscript denotes the range of the moment order, the superscript denotes the range of the number of Clenshaw-Curtis quadrature points from the coarsest to finest refinement level.

approach, especially for problems with high uncertain dimension. When looking at the computed IPM expectation value and variance (see Figure 4.7D and 4.7E) and comparing the results against the reference solution in Figure 4.7A and 4.7B, one can observe that since the uncertainties have a bigger effect on the result than in the one-dimensional case, the IPM solution still shows a step-like profile. However, we are able to capture the main features of the reference solution. The increased effect of the uncertainty can again be seen in Figure 4.7F, where a refinement level of 8, i.e. a truncation order of 9 is chosen on a larger portion than for the one-dimensional case. Visualizing the variance on a logarithmic scale in Figure 4.7C further shows that the applied refinement indicators work well and enforce the usage of higher-order moments in areas of high variance.

4.5.3 Euler 2D with a three-dimensional uncertainty

For the last numerical study we again use the Euler equations, but this time not in the context of the NACA airfoil, but with a two-dimensional bent Sod shock tube experiment. This testcase is similar to the one-dimensional shock tube which has been discussed in Section 1.4.1. However, now we assume a two-dimensional, curved

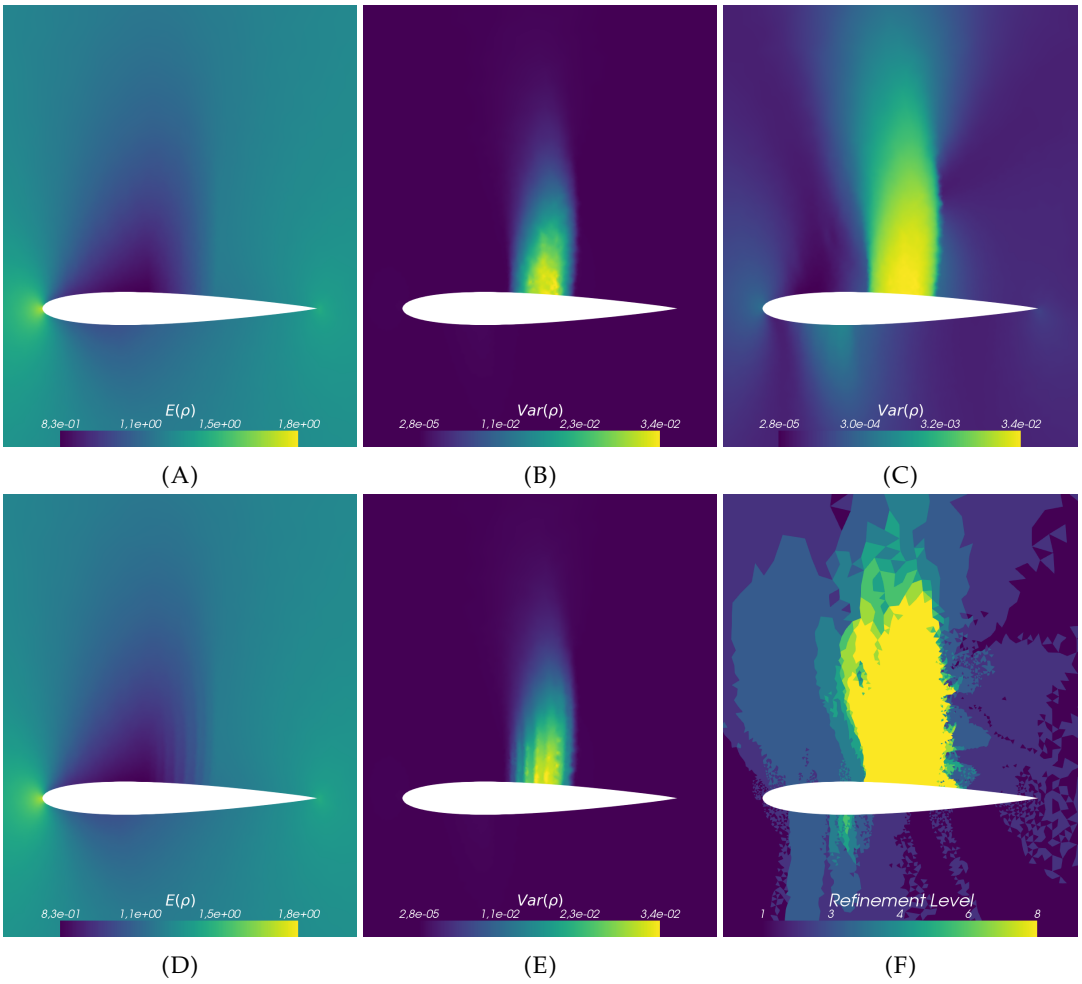


FIGURE 4.7: Reference solution (A) E[ρ] and (B) Var[ρ] and with logarithmic scaled (C) Var[ρ]. readosIPM$_{1-9}$ solution for (D) E[ρ] and (E) Var[ρ], with the resulting (F) refinement levels.

tube geometry. One can find the corresponding spatial discretization in Figure 4.8C. As in in the one-dimensional setting, the prescribed initial condition for this testcase describes a gas at rest, but with a discontinuity, in density as well as energy in the upper part of the tube. The density $\rho_u = p/(R \cdot T)$ in the upper part is set to $\rho_u = 1.289$ with $p = 101\,325$ Pa, a temperature of 273.15 K and the specific gas constant for dry air $R = 287.87$. The energy ρE_u is set to exactly 1.0. For the initial conditions in the *lower* part we set $\rho_l = 0.5 \cdot \rho_u$ and $\rho E_l = 0.3$. The heat capacity ratio γ is set to 1.4 as in the previous studies. We again augment the deterministic case by inflicting uncertainties and again increase the number uncertainties to three. The applied boundary conditions are Euler slip conditions $\boldsymbol{v}^T \boldsymbol{n} = 0$ for the walls and Dirichlet type boundary conditions at the ends of the tube which set to the deterministic initial conditions. The first two uncertainties are the density and energy of the lower part of the shock tube. Here we set $\rho_l \sim U(1.189, 1389)$ and $\rho E_l \sim U(0.2, 0.4)$. The third uncertainty enters through the position of the shock itself and thus $y_{\text{shock}} \sim U(1.0, 1.2)$. Note, that in contrast to the upper testcases, here we are not interested in the steady state of the system, but rather the of time evolution of the expectancy and variance. The computational mesh (see Figure 4.8C) is an unstructured mesh of 25 458 triangular cells, where the cells are all similar in size, i.e. there are not refined regions as in the NACA testcase. The simulations were run until a time of 2.0s. For the refinement

Sparse grids				
Moment order	1	2	-	-
Number of quadrature points	25	441	-	-
Tensorized grids				
Moment order	1	2	3	4
Number of quadrature points	27	125	125	729

TABLE 4.1: Order of moments and corresponding number of quadrature nodes used.

barries values of $\delta_- = 2 \cdot 10^{-2}$ and $\delta_+ = 4 \cdot 10^{-3}$ were set for all adaptive simulations. The corresponding results can be seen in Figure 4.8. The reference solution in Figure 4.8A and 4.8B was computed using SC with 50 quadrature points in each stochastic dimension, yielding a total of $50^3 = 125\,000$ quadrature points.
For this testcase we want to compare the results for sparse grids with respect to tensorized grids for higher stochastical dimensions, where both quadrature sets are based on Clenshaw-Curtis nodes. As we made used of adaptivity, the corresponding moment orders and used quadrature points can be seen in Table 4.1. The following properties emerge:

- It was not possible to obtain a result for sparse grids with moments of order 3 or higher. Any level up to level 11 (72 705 quadrature nodes) of the used Clenshaw-Curtis sparse grids (see Section 1.4.4), resulted in an ill conditioned Hessian matrix of the dual problem and thereby to a failure of the method.
- Sparse grids required a significantly higher number of quadrature points for the same order of moments in comparison to tensorized grids.

Even though the solutions in Figure 4.8 show the same characteristics and sparse grids are generally well suited and often used in combination with the SC method, our tests reveal that, at least for the Clenshaw-Curtis based sparse grids, these quadrature rules are not well suited to be used in combination with IPM as they require significantly more quadrature points and are only able to generate results for low orders of moments. Tensorized quadrature on the other hand performs similarly well as in the previous testcases and manages to yield a closer solution for the expectation as well as the variance due to the usage of higher moments. Tensorized quadrature rules should thus be preferred for lower stochastical dimensions.

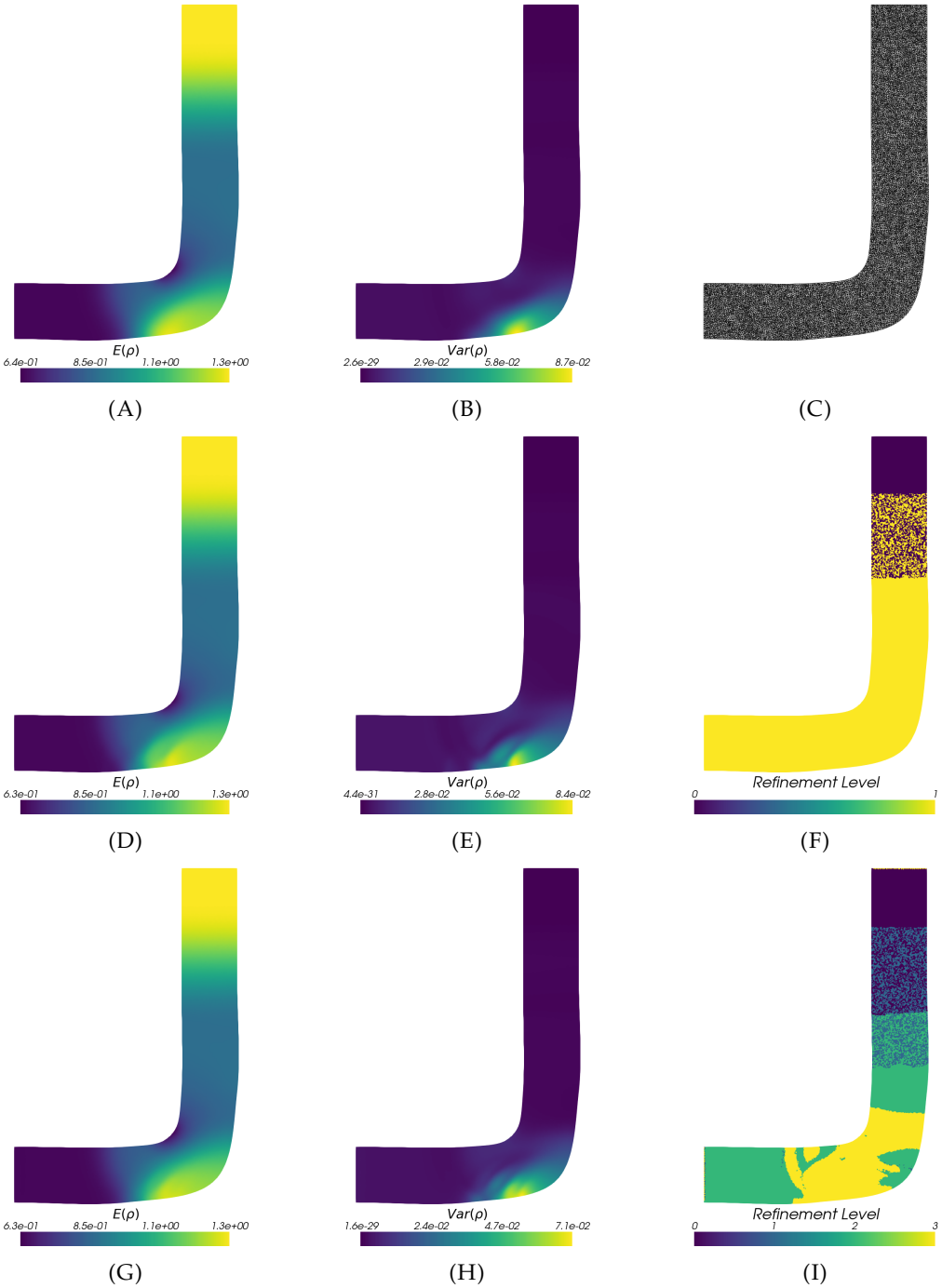


FIGURE 4.8: Reference solution $\mathrm{E}[\rho]$ and $\mathrm{Var}[\rho]$ and used computational mesh (top row) and corresponding adaptive IPM solutions for sparse (middle row) and tensorized grids (bottom row) with the used refinement levels at the last time step.

Chapter 5

Filtered stochastic-Galerkin method

In Chapter 2, we have discussed different entropy choices for the IPM method applied to scalar, hyperbolic problems to guarantee non-oscillatory solution approximations. This approach, however becomes challenging for systems, which restrict the choice of admissible entropies by the integrability condition (1.8). Results in Chapter 4 show, that the limited choice of entropies for the IPM method yields oscillatory solution approximations. Therefore, this chapter discusses another approach to dampen oscillations by filtering the coefficients of the gPC expansion (1.84). Filters are a common strategy to reduce oscillations in spectral methods, see for example [13, 56]. Applications of filters in the context of kinetic theory can be found in [91], where a filter is constructed by adding a penalizing term to the L^2 error of the solution. More choices of filter functions can be found in [35, 73, 114].

The strength of the penalizing term, often referred to as the filter strength, is a parameter which is typically user-determined and problem dependent. Therefore, choosing an adequate filter strength remains a cumbersome task. To automate the choice of the filter strength, we construct a new filter, which is based on Lasso regression [134]. The resulting filter depends on the gPC coefficients and sets small-magnitude and high-order coefficients to zero, yielding sparsity in the filtered coefficients. We use this property to adaptively pick the filter strength such that the moment with the highest order is set to zero. This automated choice of the filter strength avoids the tedious task of picking a suitable filter parameter and at the same time promises optimality of the optimization problem, i.e. we obtain a minimal value of the penalized L^2 error. We demonstrate the effectiveness of our method by investigating Burgers' and the Euler equations and comparing the results to SG and IPM. One observes that the filtered method yields an improved efficiency compared to stochastic-Galerkin in the Burgers' case. Due to its reduced runtime, the filtered SG is able to compete with IPM. When taking a look at the Euler equations, the filtered SG yields an improved approximation of quantities of interest at shocks.

This chapter is structured as follows. In Section 5.1, we review the concept of filtering and present the new Lasso filter. Section 5.2 discusses the numerical implementation of the filter. An automated way to adaptively pick the filter strength is derived in Section 5.3. The filtered solution is compared to common SG and IPM by investigating Burgers' as well as the Euler equations in Section 5.4.

5.1 Filters for Uncertainty Quantification

As already mentioned, filters have been applied in various fields. The idea of filters is to introduce a dampening function into the polynomial expansion of a given

function to mitigate oscillations. Let us assume that we want to represent a function $u : [-1,1] \to \mathbb{R}$ with $N+1$ given Fourier coefficients $a_i := \int_{-1}^{1} u(s)\phi_i(s)\,ds$, where $\phi_i : [-1,1] \to \mathbb{R}$ are orthonormal polynomials. The filtered expansion then takes the form

$$u(s) \approx u_\sigma(s) := \sum_{i=0}^{N} \sigma(i/N) a_i \phi_i(s). \tag{5.1}$$

The function $\sigma : \mathbb{R}_+ \to \mathbb{R}_+$ is called the filter function and its choice crucially affects the resulting approximation quality. Let σ fulfill the following properties:

1. $\sigma(0) = 1$,
2. $\sigma^{(\ell)}(0) = 0, \qquad \text{for } \ell = 1, \cdots, p-1$,
3. $\sigma^{(\ell)}(1) = 0, \qquad \text{for } \ell = 0, \cdots, p-1$.

Then, according to [14], the filtered approximation (5.1) fulfills

1. If u has p continuous derivatives, then
$$|u(s) - u_\sigma(s)| \leq C N^{1/2-p}.$$
2. If u has a jump discontinuity at $c \in [-1,1]$, then
$$|u(s) - u_\sigma(s)| \leq C \cdot d(s)^{1-p} N^{1-p},$$
where $d(s)$ is the distance to c.

Further discussions of filtering techniques can be found in [13, 49, 56]. A convergence proof of filters in combination with kinetic theory can be found in [35]. In the following, we apply the concept of filtering to UQ and derive the standard L^2 filter. The main idea is to use filters in order to dampen high-order expansion coefficients in the gPC expansion (1.84). The remainder will not focus on the mentioned filter order and we will leave this to future work. Let us start the derivation by choosing a fixed time t and a fixed spacial position x, i.e. the function which we want to approximate only depends on a scalar random variable $\xi \in \Theta \subset \mathbb{R}$. First, let us discuss how to approximate such a function $\boldsymbol{u} : \Theta \to \mathbb{R}^m$ with a polynomial

$$\boldsymbol{u}_N(\xi) = \sum_{i=0}^{N} \boldsymbol{\alpha}_i \varphi_i(\xi). \tag{5.2}$$

Here, the expansion coefficients $\boldsymbol{\alpha}_i \in \mathbb{R}^m$, are collected in a matrix $\boldsymbol{\alpha} \in \mathbb{R}^{(N+1)\times m}$. Hence the expansion coefficient for state l and polynomial order i is given by α_{il}. Now, these coefficients need to be chosen to guarantee a satisfactory solution approximation. Stochastic-Galerkin chooses these coefficients such that the L^2 distance between the approximation (5.2) and the exact solution $\boldsymbol{u}$ is minimized. Hence, the expansion coefficients $\boldsymbol{\alpha}$ are chosen to minimize the cost function

$$\mathcal{J}(\boldsymbol{\alpha}) := \frac{1}{2} \left\langle \left\| \boldsymbol{u} - \sum_{i=0}^{N} \boldsymbol{\alpha}_i \varphi_i \right\|_2^2 \right\rangle, \tag{5.3}$$

where $\|\cdot\|_2$ is the Euclidean norm. The minimizer which we denote by $\hat{\boldsymbol{\alpha}}$ is then given by $\hat{\boldsymbol{\alpha}}_i = \hat{\boldsymbol{u}}_i = \langle \boldsymbol{u}\varphi_i \rangle$. This choice suffers from oscillations when the function $\boldsymbol{u}$

lacks sufficient regularity. The filtered gPC expansion we suggest in this work tackles this problem. A filtered solution approximation can be constructed by including a term which punishes oscillation in the original cost function (5.3). Making use of an operator L to punish oscillations as well as a parameter $\lambda \in \mathbb{R}^+$ called the filter strength, we now minimize

$$\mathcal{J}_{\lambda,2}(\boldsymbol{\alpha}) := \frac{1}{2}\left\langle \left\| \boldsymbol{u} - \sum_{i=0}^{N} \boldsymbol{\alpha}_i \varphi_i \right\|_2^2 \right\rangle + \lambda \left\langle \left\| L \sum_{i=0}^{N} \boldsymbol{\alpha}_i \varphi_i \right\|_2^2 \right\rangle, \tag{5.4}$$

over the expansion coefficients $\boldsymbol{\alpha}$. A standard choice for L when dealing with uniform distributions is $Lu(\xi) = ((1-\xi^2)u'(\xi))'$, i.e. the differential operator defined in (1.87), since

$$L\varphi_i = -i(i+1)\varphi_i,$$

i.e. the Legendre polynomials[1] are eigenfunctions of L. In contrast to the cost function (5.3), we no longer solely focus on minimizing the distance to the exact solution, but we include a term to punish oscillations. This term is weighted by a user determined parameter λ, which is called the filter strength. It can be used to prescribe how much the cost function should focus on punishing oscillatory solution approximations. Differentiation of (5.4) w.r.t. $\boldsymbol{\alpha}_i$ gives the optimal coefficients

$$\hat{\boldsymbol{\alpha}}_i = \frac{1}{1+\lambda i^2(i+1)^2}\hat{\boldsymbol{u}}_i. \tag{5.5}$$

Hence, the optimal expansion coefficients of (5.4) are the moments $\hat{\boldsymbol{u}}_i$ multiplied by a filter function

$$\tilde{g}_\lambda(i) := \frac{1}{1+\lambda i^2(i+1)^2}. \tag{5.6}$$

This corresponds to the L^2 filter based on splines[2]. One can see that the filter damps high-order coefficients, while leaving the 0_{th} order coefficient untouched. Using the optimal coefficients (5.5) in the polynomial representation (5.2) yields the L^2 filtered gPC expansion

$$\boldsymbol{u}(t,\boldsymbol{x},\xi) \approx \sum_{i=0}^{N} \tilde{g}_\lambda(i)\hat{\boldsymbol{u}}_i(t,\boldsymbol{x})\varphi_i(\xi). \tag{5.7}$$

It remains to pick an adequate filter strength λ. The filter strength must be chosen such that oscillations are sufficiently dampened while the solution structure is preserved. Finding an adequate filter strength is challenging. In the following, we derive a filter which can be used to choose this filter strength.

[1] For arbitrary distributions, the operator should be chosen such that the corresponding gPC polynomials are eigenfunctions of L.

[2] This is not the only filter which damps oscillations. Indeed, several other filters can be used such as Lanczos and ErfcLog [114].

5.1.1 Construction of the Lasso filter

Our task is to find a smooth representation of $\boldsymbol{u}$ which promotes sparsity. Combining the ideas of Lasso regression and filtering, we introduce the cost functional

$$\mathcal{J}_\lambda(\boldsymbol{\alpha}) := \frac{1}{2}\left\langle \left\| \boldsymbol{u} - \sum_{i=0}^{N} \boldsymbol{\alpha}_i \varphi_i \right\|_2^2 \right\rangle + \lambda \left\langle \sum_{i=0}^{N} \|L\boldsymbol{\alpha}_i \varphi_i\|_1 \right\rangle, \tag{5.8}$$

where $\|\cdot\|_1$ is the 1-Norm for vectors. Compared to L^2 filtering, the punishing term has been changed to an L^1 term which acts on each expansion term of the solution individually. The corresponding filter takes the following form.

Theorem 19. *The minimizer of the cost functional (5.8) is given by $\hat{\alpha}_{il} = g_\lambda(i, \hat{u}_{il})\hat{u}_{il}$ for states $l = 1, \cdots, m$ and polynomial orders $i = 0, \cdots, N$. The Lasso filter function reads*

$$g_\lambda(i, \hat{u}_{il}) := \left(1 - \frac{\lambda i(i+1)\|\varphi_i\|_{L^1}}{|\hat{u}_{il}|}\right)_+, \tag{5.9}$$

with $(\cdot)_+ := \max(\cdot, 0)$.

Proof. The proof follows ideas from Lasso regression, see [134]. A key ingredient of Lasso regression is the minimization of a cost function which includes absolute values. In this case, optimality holds if the subdifferential of the cost function f is equal to zero. At differentiable points, the subdifferential of f is simply the gradient. At non-differentiable points x_0, the subdifferential is the set of slopes belonging to all straight lines that touch $f(x_0)$ and lie below f in the neighborhood of $f(x_0)$. To minimize the cost functional (5.8), we compute the subdifferential

$$\partial_{jl}\mathcal{J}_\lambda(\boldsymbol{\alpha}) = \begin{cases} \left\{ -\left\langle \left(u_l - \sum_i \alpha_{il}\varphi_i\right)\varphi_j \right\rangle + \lambda\eta \left\langle \left|L\varphi_j\right| \right\rangle : \eta \in [-1,1] \right\} & \text{if } \alpha_{jl} = 0 \\ -\left\langle \left(u_l - \sum_i \alpha_{il}\varphi_i\right)\varphi_j \right\rangle + \lambda \cdot \operatorname{sign}(\alpha_{jl}) \left\langle \left|L\varphi_j\right| \right\rangle & \text{else} \end{cases}. \tag{5.10}$$

Here, $\partial_{jl}\mathcal{J}_\lambda$ denotes the subdifferential with respect to the expansion coefficient α_{jl}, i.e. the expansion coefficient at state l and moment order j. The parameter $\eta \in [-1,1]$ is used to parametrize the set of straight lines at non differentiable points. The proof consists of three steps.

1. We show that if the optimal expansion coefficient $\hat{\alpha}_{jl}$ is equal to zero, then the moment $\hat{u}_{jl}$ fulfills

$$\hat{u}_{jl} \in [-\lambda j(j+1)\|\varphi_j\|_{L^1}, \lambda j(j+1)\|\varphi_j\|_{L^1}]. \tag{5.11}$$

2. If (5.11) does not hold, we show that the optimal expansion coefficient is given by

$$\hat{\alpha}_{jl} = \hat{u}_{jl}\left(1 - \lambda j(j+1)\|\varphi_j\|_{L^1}\frac{1}{|\hat{u}_{jl}|}\right),$$

   where the term inside the brackets must be positive.

3. We show that if (5.11) holds, the optimal expansion coefficient $\hat{\alpha}_{jl}$ is equal to zero.

To determine the optimal expansion coefficients, we need to pick $\hat{\alpha}_{jl}$, such that the cost function (5.8) is minimized. Assuming $\hat{\alpha}_{jl} = 0$, this translates into ensuring that the zero slope lies in the set of the first condition of $\partial_{jl}\mathcal{J}_\lambda$, i.e.

$$0 \in \left\{ -\left\langle \left(u_l - \sum_i \hat{\alpha}_{il}\varphi_i \right) \varphi_j \right\rangle + \lambda\eta \left\langle \left| L\varphi_j \right| \right\rangle : \eta \in [-1,1] \right\}.$$

Using orthonormality and the definition of gPC coefficients gives

$$0 \in \left\{ -\hat{u}_{jl} + \hat{\alpha}_{jl} + \lambda\eta \left\langle \left| L\varphi_j \right| \right\rangle : \eta \in [-1,1] \right\}.$$

Recalling that $\hat{\alpha}_{jl} = 0$ and $L\varphi_j = -j(j+1)\varphi_j$ yields

$$0 \in \left\{ -\hat{u}_{jl} + \lambda\eta j(j+1)\|\varphi_j\|_{L^1} : \eta \in [-1,1] \right\}.$$

Hence, if $\hat{\alpha}_{jl} = 0$ we have

$$\hat{u}_{jl} \in [-\lambda j(j+1)\|\varphi_j\|_{L^1}, \lambda j(j+1)\|\varphi_j\|_{L^1}]. \tag{5.12}$$

This concludes the first part of the proof. Now if

$$\hat{u}_{jl} \notin [-\lambda j(j+1)\|\varphi_j\|_{L^1}, \lambda j(j+1)\|\varphi_j\|_{L^1}], \tag{5.13}$$

we have that $\hat{\alpha}_{jl} \neq 0$ and the gradient becomes the second condition of (5.10). In this case, the optimality condition reads

$$\begin{aligned}
\partial_{jl}\mathcal{J}_\lambda(\hat{\boldsymbol{\alpha}}) =& -\left\langle \left(u - \sum_i \hat{\alpha}_{il}\varphi_i \right) \varphi_j \right\rangle + \lambda \mathrm{sign}(\hat{\alpha}_{jl}) \left\langle \left| L\varphi_j \right| \right\rangle \\
=& -\left\langle u\varphi_j \right\rangle - \sum_i \hat{\alpha}_{il} \left\langle \varphi_i\varphi_j \right\rangle + \lambda j(j+1)\mathrm{sign}(\hat{\alpha}_{jl}) \left\langle \left| \varphi_j \right| \right\rangle \\
=& -\hat{u}_{jl} + \hat{\alpha}_{jl} + \lambda j(j+1)\mathrm{sign}(\hat{\alpha}_{jl})\|\varphi_j\|_{L^1} = 0.
\end{aligned} \tag{5.14}$$

To determine the sign of $\hat{\alpha}_{jl}$, we rearrange

$$\begin{aligned}
\hat{u}_{jl} =& \hat{\alpha}_{jl} + \lambda j(j+1)\mathrm{sign}(\hat{\alpha}_{jl})\|\varphi_j\|_{L^1} \\
=& \hat{\alpha}_{jl} \underbrace{\left(1 + \lambda j(j+1)\frac{1}{|\hat{\alpha}_{jl}|}\|\varphi_j\|_{L^1} \right)}_{>0} \Rightarrow \mathrm{sign}\left(\hat{u}_{jl}\right) = \mathrm{sign}\left(\hat{\alpha}_{jl}\right).
\end{aligned}$$

Plugging this into (5.14) yields

$$\begin{aligned}
\hat{\alpha}_{jl} =& \hat{u}_{jl} - \lambda j(j+1)\mathrm{sign}(\hat{u}_{jl})\|\varphi_j\|_{L^1} \\
=& \hat{u}_{jl}\left(1 - \lambda j(j+1)\|\varphi_j\|_{L^1}\frac{1}{|\hat{u}_{jl}|} \right).
\end{aligned} \tag{5.15}$$

Note that since $\mathrm{sign}(\hat{u}_{jl}) = \mathrm{sign}(\hat{\alpha}_{jl})$ must hold, the case $1 - \lambda j(j+1)\|\varphi_j\|_{L^1}\frac{1}{|\hat{u}_{jl}|} \leq 0$ does not occur and the bracket term in (5.15) remains positive. This concludes the

second part of the proof. So if $\hat{\alpha}_{jl} \neq 0$, the bracket term is positive and we get

$$\begin{aligned} 1 - \frac{\lambda j(j+1)\|\varphi_j\|_{L^1}}{|\hat{u}_{jl}|} &> 0 \\ \Leftrightarrow |\hat{u}_{jl}| - \lambda j(j+1)\|\varphi_j\|_{L^1} &> 0 \\ \Leftrightarrow |\hat{u}_{jl}| &> \lambda j(j+1)\|\varphi_j\|_{L^1}. \end{aligned}$$

One can rewrite this expression as

$$\hat{u}_{jl} \notin [-\lambda j(j+1)\|\varphi_j\|_{L^1}, \lambda j(j+1)\|\varphi_j\|_{L^1}]. \tag{5.16}$$

This shows that if $\hat{\alpha}_{jl} \neq 0$, then (5.16) holds. I.e. if $\hat{u}_{jl} \in [-\lambda j(j+1)\|\varphi_j\|_{L^1}, \lambda j(j+1)\|\varphi_j\|_{L^1}]$, we need to look at the first condition of the subdifferential, meaning that $\hat{\alpha}_{jl}$ must be set to zero. Let us now derive a more compact notation for the derived minimizer, which uses the Lasso filter function (5.9). Using the notation $x_+ := \max(x, 0)$ the filtered coefficient can be written as

$$\hat{\alpha}_{jl} = \hat{u}_{jl}\left(1 - \frac{\lambda j(j+1)\|\varphi_j\|_{L^1}}{|\hat{u}_{jl}|}\right)_+,$$

which yields the filter function from the theorem. □

The constructed filter will in the following be called *Lasso filter*. In contrast to standard filters, the filter function (5.9) depends on the moments of the solution. The first moment is not modified while higher-order moments are dampened. Note that if

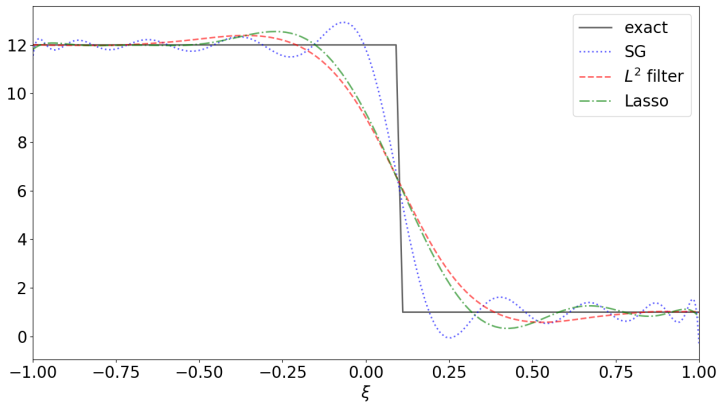
$$\frac{\lambda j(j+1)\|\varphi_j\|_{L^1}}{|\hat{u}_{jl}|} \geq 1,$$

i.e. when the order of the moment increases or the absolute value of the moment decreases, the filtered moment will be chosen to be zero.

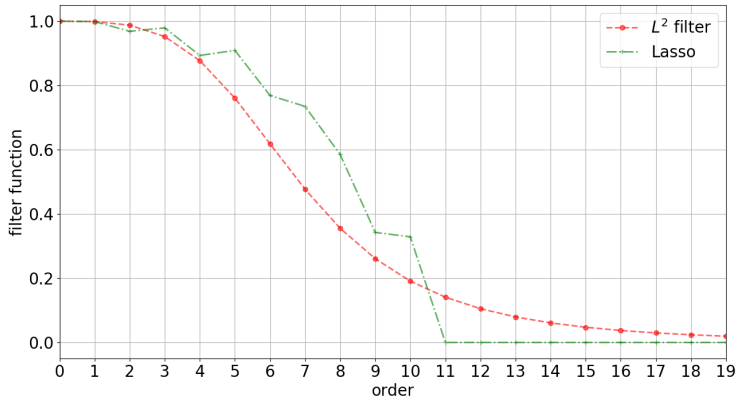
To demonstrate the effects of filtering, we consider a shock as depicted in Figure 5.1A, which we denote by u_{ex}. Note that again, we assume that the function which we wish to approximate is known and only depends on $\xi \in \Theta$, i.e. it does not depend on time or space. The random variable is now distributed uniformly in the interval $\Theta = [-1, 1]$ and the jump from $u_L = 12$ to $u_R = 1$ occurs at $\xi_0 = 0.1$. First, we determine the moments of this shock by simply computing the integral $\hat{\boldsymbol{u}} = \langle u_{ex}\boldsymbol{\varphi}\rangle$ with a 200 point Gauss-Lobatto quadrature rule. Using this moment vector, the SG approximation (1.84) can be used as a polynomial approximation of the shock, i.e. we have

$$u_{ex} \approx u_{SG}(\xi) = \sum_{i=0}^{N} \hat{u}_i \varphi_i(\xi). \tag{5.17}$$

Since the shock is scalar and not vector valued (i.e. $m = 1$), we dropped the dependency on the state $l = 1$, i.e. $\hat{u}_i := \hat{u}_{1,i}$. Figure 5.1A shows that this approximation leads to oscillations. The filtered approximation now replaces the moments in (5.17) by their filtered counterpart. For the L^2 filter, the moments $\hat{\boldsymbol{u}}$ are multiplied by the filter function $\tilde{g}_\lambda$, which is given in (5.6), i.e. we obtain the solution approximation



(A)



(B)

FIGURE 5.1: (A) Approximation of a given shock (which only depends on ξ) using SG and filtered SG with 20 expansion coefficients. (B) Filter functions for Lasso (5.9) with $\lambda = 0.0035$ and L^2 filter function (5.6) with $\lambda = 0.00035$.

(5.7). The Lasso filter uses the filter function (5.9). Hence, we get the filtered approximations

$$u_{L_2}(\xi) = \sum_{i=0}^{N} \frac{1}{1+\lambda i^2(i+1)^2} \hat{u}_i \varphi_i(\xi), \quad u_{\text{Lasso}}(\xi) = \sum_{i=0}^{N} \left(1 - \frac{\lambda i(i+1)\|\varphi_i\|_{L^1}}{|\hat{u}_i|}\right)_+ \hat{u}_i \varphi_i(\xi).$$

Filter strengths for both methods are chosen such that the solution approximations show similar behavior. For L^2 filtering, we use $\lambda = 0.00035$ and the Lasso filter uses $\lambda = 0.0035$. The corresponding filter functions are shown in Figure 5.1B. In contrast to the L^2 filter, the Lasso filter yields a sparse solution representation, due to the fact that all coefficients with degree bigger than 10 are set to zero. The L^2 filter keeps coefficients with high order, however their contribution to the solution approximation is negligible. Effects of using filter functions are shown in Figure 5.1A, where both, the Lasso and L^2 filtered solution mitigate oscillations at the cost of resulting in a smeared out shock approximation.

When the exact solution does not depend on ξ, i.e. only the zeroth order moment is non-zero, the filter does not affect the approximation since $g_\lambda(0) = 1$. Consequently, the filters allow a sharp approximation of a deterministic shock in the spatial domain.

Note, that we have only discussed scalar random variables. For multi-dimensional problems, i.e. if $\boldsymbol{\xi} \in \mathbb{R}^p$ with $p > 1$, the operator L, which punishes oscillations can be applied independently to each random dimension. If

$$L_k u(\boldsymbol{\xi}) := \partial_{\xi_k}((1 - \xi_k^2)\partial_{\xi_k} u(\boldsymbol{\xi})),$$

one can use the operator

$$Lu(\boldsymbol{\xi}) := (L_1 \circ L_2 \circ \cdots \circ L_p)(u(\boldsymbol{\xi}))$$

in the cost function of the optimization problem (5.4) or for the Lasso filter (5.8).

Before turning to the choice of the filter strength λ, we need to discuss how the filtering procedure can be integrated into the SG framework. The idea is to replace the standard gPC coefficients in the time update of the numerical discretization by the filtered coefficients. Details on how the filter can be integrated into a given SG solver are discussed in the following.

5.2 Numerical implementation

In this section, we discuss how to integrate the presented filters into a given stochastic-Galerkin code framework. Hence, we now include time and space.

Within the full stochastic-Galerkin algorithm, the filter is applied as follows. First, a finite-volume method with forward-Euler time discretization for the stochastic-Galerkin moment equations (1.102) is given by

$$\hat{\boldsymbol{u}}_j^{n+1} = \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}(\boldsymbol{F}^*(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) - \boldsymbol{F}^*(\hat{\boldsymbol{u}}_{j-1}^n, \hat{\boldsymbol{u}}_j^n)), \tag{5.18}$$

where $\hat{\boldsymbol{u}}_j^n$ is the matrix of moment components for spatial cell $j = 1, \cdots, N_x$ and time step $n = 0, \cdots, N_t$. The numerical flux $\boldsymbol{F}^*$ can again be chosen according to (4.3) when using a quadratic ansatz $\mathcal{U}(\hat{\boldsymbol{u}}) = \hat{\boldsymbol{u}}^T\boldsymbol{\varphi}$. For further information on the chosen discretization of the moment system, see Section 4.1. To simplify notation, let us collect the filter function in a matrix $\mathcal{L}(\lambda) := \operatorname{diag}\{\tilde{g}_\lambda(i)\}_{i=0}^N$ for the L^2 filter and $\mathcal{L}(\lambda, \hat{\boldsymbol{u}}) := \operatorname{diag}\{g_\lambda(i, \hat{\boldsymbol{u}})\}_{i=0}^N$ for the Lasso filter. Hence we can write the L^2 filtered gPC coefficients as $\overline{\boldsymbol{u}} = \mathcal{L}(\lambda)\hat{\boldsymbol{u}}$. The L^2 filtered method is then given by

$$\overline{\boldsymbol{u}}_j^n = \mathcal{L}(\lambda)\hat{\boldsymbol{u}}_j^n, \tag{5.19a}$$

$$\hat{\boldsymbol{u}}_j^{n+1} = \overline{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}(\boldsymbol{F}^*(\overline{\boldsymbol{u}}_j^n, \overline{\boldsymbol{u}}_{j+1}^n) - \boldsymbol{F}^*(\overline{\boldsymbol{u}}_{j-1}^n, \overline{\boldsymbol{u}}_j^n)). \tag{5.19b}$$

For the Lasso filter, we get

$$\overline{\boldsymbol{u}}_j^n = \mathcal{L}(\lambda, \hat{\boldsymbol{u}}_j^n)\hat{\boldsymbol{u}}_j^n, \tag{5.20a}$$

$$\hat{\boldsymbol{u}}_j^{n+1} = \overline{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}(\boldsymbol{F}^*(\overline{\boldsymbol{u}}_j^n, \overline{\boldsymbol{u}}_{j+1}^n) - \boldsymbol{F}^*(\overline{\boldsymbol{u}}_{j-1}^n, \overline{\boldsymbol{u}}_j^n)). \tag{5.20b}$$

The filtered SG scheme when using the Lasso filter is then given in Algorithm 8.

Algorithm 8 Filtered stochastic-Galerkin Method with Lasso filter

1: $\hat{\boldsymbol{u}}_j^0 \leftarrow setupInitialConditions$ for all cells j
2: choose λ
3: **for** $n = 0$ to N_t **do**
4: **for** $j = 1$ to N_x **do**
5: $\hat{\boldsymbol{u}}_j^n \leftarrow \mathcal{L}(\lambda, \hat{\boldsymbol{u}}_j^n)\hat{\boldsymbol{u}}_j^n$
6: **for** $j = 1$ to N_x **do**
7: $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}(\boldsymbol{F}^*(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) - \boldsymbol{F}^*(\hat{\boldsymbol{u}}_{j-1}^n, \hat{\boldsymbol{u}}_j^n))$

The algorithm for the L^2 filter is given when using the matrix $\mathcal{L}(\lambda)$ instead of $\mathcal{L}(\lambda, \hat{\boldsymbol{u}}_j^n)$. Hence, provided a stochastic-Galerkin implementation exists, the filtered stochastic-Galerkin method can be implemented in a straight forward manner by simply adding the filtering step in line 5.

5.3 Choosing the filter strength

A cumbersome task when using filters is to select an adequate filter strength λ, which sufficiently damps oscillations while preserving general characteristics of the exact solution. Since the optimal filter strength is problem dependent, a parameter study must be conducted for finding an optimal value for λ. Furthermore, the filter strength does not depend on the solution, i.e. smooth regions are as strongly dampened as discontinuities. In the following, an automated procedure to pick an adequate filter strength is proposed. The resulting filter is different for every spatial cell as well as every time step.

We start writing down the Lasso optimization problem (5.8) for a given truncation order P.

$$\mathcal{J}_\lambda(\boldsymbol{\alpha}) = \frac{1}{2}\left\langle \left\| \boldsymbol{u} - \sum_{i=0}^{P} \boldsymbol{\alpha}_i \varphi_i \right\|_2^2 \right\rangle + \lambda \left\langle \sum_{i=0}^{P} \| L\boldsymbol{\alpha}_i \varphi_i \|_1 \right\rangle.$$

Without the Lasso regression term, the solution of the optimization problem will yield the exact solution u for $P \to \infty$. When adding the Lasso term with some choice for λ, we observe that the solution to the optimization problem becomes sparse (according to Theorem 19) and for some $\tilde{N}$, all moments $\hat{u}_i$ with $i > \tilde{N}$ are set to zero. Thus solving the SG system with the truncation order $\tilde{N}$ or with a much higher order $P \gg \tilde{N}$, where P can even be infinite, will yield the same result.

Keeping this observation in mind, we have two options and are in zugzwang:

1. either make some choice for λ which then tells us a suitable truncation order $\tilde{N}$ or
2. pick a truncation order, which then determines the filtering coefficient λ.

In this work, we choose option 2, i.e. we derive a strategy to choose an adequate λ for a given truncation order. Denoting this truncation order by N, the filter strength of state l in cell j at time step n is given by

$$\lambda^* = \frac{|\hat{u}_{N,l,j}^n|}{N(N+1)\|\varphi_N\|_{L^1}}. \tag{5.21}$$

This choice ensures that the N_{th} filtered coefficient is zero. Note that now, the filter strength depends on time t_n, spacial index j as well as the truncation order N, i.e. the filter strength is chosen adaptively for every spacial cell at every time step. We will omit writing out this dependency for sake of readability. Since the filter function $g_\lambda(i, \hat{u}_{il})$ decreases quadratically in i, the event that all moments $\hat{u}_{il}$ with $i > N$ in the individual cell at the given time are zero is likely. Therefore, with this choice of the filter coefficient, we obtain the same solution as with an order $M \gg N$ moment system. The resulting filtering function is then given by

$$g_{\lambda^*}\left(i, \hat{u}^n_{ilj}\right) = \left(1 - \frac{i(i+1)}{N(N+1)} \frac{\|\varphi_i\|_{L^1}}{\|\varphi_N\|_{L^1}} \frac{|\hat{u}^n_{N,l,j}|}{|\hat{u}^n_{ilj}|}\right)_+ . \tag{5.22}$$

For better readability, we have suppressed the dependency of λ^* on $\hat{u}^n_{N,l,j}$. The filter strength (5.21) depends on the highest order moment, which can be seen as a smoothness indicator (w.r.t. the random variable ξ). A small value of the highest order moment, which indicates a smooth solution leads to a small filter strength, i.e. the effect of the filter is locally weakened at smooth solutions. Equivalently, the filter strength is increased in non-smooth regions. Areas in which the solution is only non-smooth with respect to the spatial variable x are not affected by the filter.

Substituting the adaptive filter strength (5.21) into Algorithm 8 yields the method we use in the following section to obtain non-oscillatory approximations of expected value and variance.

5.4 Results

5.4.1 Burgers' equation

In the following, we start by studying the random Burgers' equation (1.80), which reads

$$\begin{aligned} \partial_t u(t,x,\xi) + \partial_x \frac{u(t,x,\xi)^2}{2} &= 0, \\ u(t=0,x,\xi) &= u_{\mathrm{IC}}(x,\xi). \end{aligned}$$

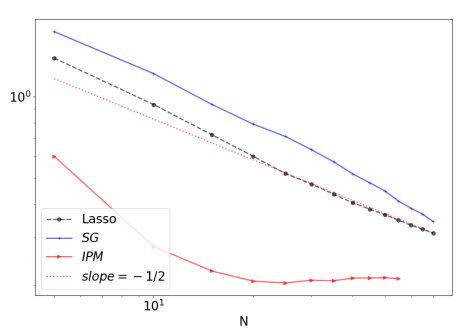
As done before, we choose the random initial condition as

$$u_{\mathrm{IC}}(x,\xi) := \begin{cases} u_L, & \text{if } x < x_0 + \sigma\xi \\ u_L + \frac{u_R - u_L}{x_0 - x_1}(x_0 + \sigma\xi - x), & \text{if } x \in [x_0 + \sigma\xi, x_1 + \sigma\xi] \\ u_R, & \text{else} \end{cases} . \tag{5.23}$$

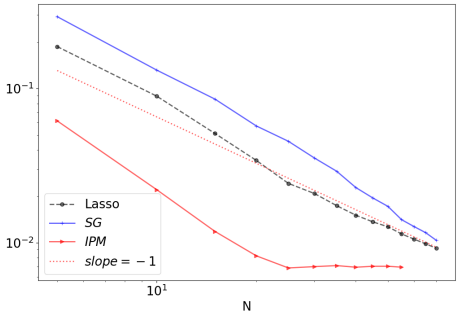
The random variable ξ is uniformly distributed on the interval $\Theta = [-1, 1]$. Furthermore, we have Dirichlet boundary conditions $u(t,a,\xi) = u_L$ and $u(t,b,\xi) = u_R$. The numerical flux is chosen according to (1.32) where the underlying numerical flux f^* is chosen to be Lax-Friedrichs. Additionally, we use the following parameter values.

$x \in D = [0,3]$	range of spatial domain
$N_x = 2000$	number of spatial cells
$t_{\text{end}} = 0.11$	end time
$x_0 = 0.5, x_1 = 1.5, u_L = 12, u_R = 1, \sigma = 0.2$	parameters of initial condition (5.23)

Note that at $t_{\text{end}} = 0.11$ the solution forms a discontinuity in the random as well as the spatial domain.



(A) L^2 error of the solution according to (5.24).



(B) L^2 error of the expectation value according to (5.25).

FIGURE 5.2: Convergence behavior of Burgers' equation for increasing truncation order N using $N_x = 2000$ spatial cells at time $t_{\text{end}} = 0.11$. The corresponding numerical values can be found in Tables 5.1, 5.2 and 5.3.

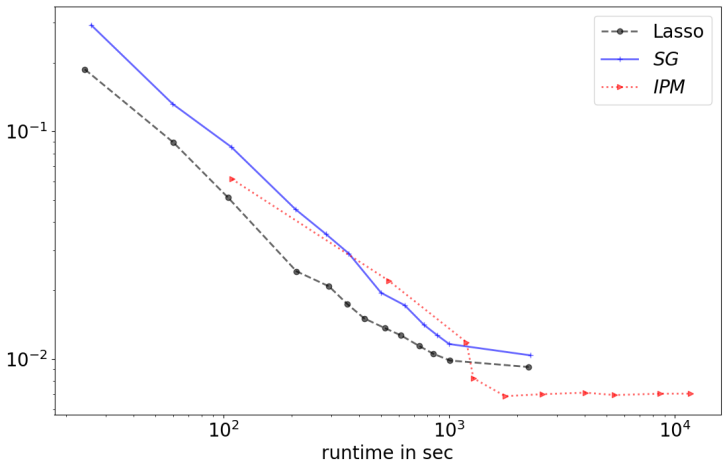


FIGURE 5.3: L^2 error of the expectation value according to (5.25) plotted over runtime. The number of spatial cells is $N_x = 2000$ and the error is computed at time $t_{\text{end}} = 0.11$. The plot uses runtime and error values from Tables 5.1, 5.2 and 5.3.

For the IPM solution, we use the bounded–barrier entropy, defined in (2.38), which is

$$s(u) = (u - u_-)\ln(u - u_-) + (u_+ - u)\ln(u_+ - u),$$

where $u_- = \min_{x,\xi} u_{\text{IC}}$ and $u_+ = \max_{x,\xi} u_{\text{IC}}$. Note that when using SG as well as the Lasso filter, all arising integrals in the numerical flux (1.32) can be computed exactly when choosing sufficiently many quadrature points, see 2.3.3. In the case of IPM, we use a Gauss-Legendre quadrature rule with $4N$ quadrature points to compute the numerical flux (2.36) as well as integrals arising in the dual problem (1.122b). With increasing time, the exact solution exhibits a jump in the random as well as physical

N	5	10	20	25	30	35	40	45	50	55	60	65	70
runtime	24.3	60.0	57.8	210.3	293.3	354.4	419.6	518.2	611.4	736.1	849.1	1008.7	2252.8
error	1.390	0.935	0.601	0.519	0.474	0.434	0.404	0.382	0.365	0.348	0.334	0.322	0.312
error E[u]	0.187	0.089	0.034	0.024	0.021	0.017	0.015	0.014	0.012	0.011	0.010	9.8E-3	9.2E-3

TABLE 5.1: Runtime (in seconds) and L^2 error (according to (5.24)) and L^2 error of E[u] (according to (5.25)) for the Lasso filter as depicted in Figure 5.2 and 5.3.

N	5	10	20	25	30	35	40	45	50	55	60	65	70
runtime	25.9	59.6	65.1	208.0	283.9	358.7	140.1	498.9	634.7	773.8	883.5	999.5	2290.17
error	1.743	1.217	0.791	0.712	0.636	0.573	0.517	0.479	0.446	0.409	0.385	0.366	0.344
error E[u]	0.293	0.131	0.057	0.045	0.035	0.029	0.023	0.020	0.017	0.014	0.013	0.012	0.010

TABLE 5.2: Runtime (in seconds) and L^2 error (according to (5.24)) and L^2 error of E[u] (according to (5.25)) for the SG method as depicted in Figures 5.2 and 5.3.

space. The properties of the SG and filtered SG solution, when approximating this discontinuity are studied in the following. We increase the number of moments and observe the resulting error of the solution as well as the expectation value. The error of the solution itself, i.e.

$$\|u - u_N\|_{L^2(D,\Theta)} := \sqrt{\int_D \left\langle \left(u(t_{\text{end}}, x, \xi) - u_N(t_{\text{end}}, x, \xi)\right)^2 \right\rangle dx} \tag{5.24}$$

is shown in Figure 5.2A. Note that the convergence does not only depend on the projection error which is $1/2$, but also on the method's closure error (i.e. the error arising from the error in the physical flux due to the approximation). Both methods show an overall convergence speed of $1/2$, however the filtered SG starts at a smaller error value. Consequently, the filtered Solution computed with 20 moments has a smaller error than the classical SG solution with 30 moments. When the moment number increases, the SG result approaches the filtered SG solution. This is due to the fact that the last moments is getting close to zero, i.e. the filter is turned off. The IPM method gives a good approximation, already for a small moment order. After a truncation order of 15, the error is not further decreased, which is most likely caused by a too dominant error of the spatial and time discretization. A similar behavior can be found in Figure 5.2B. Here, the errors of the expectation value

$$\|E[u] - E[u_N]\|_{L^2(D)} := \sqrt{\int_D \left(E[u(t_{\text{end}}, x, \cdot)] - E[u_N(t_{\text{end}}, x, \cdot)]\right)^2 dx} \tag{5.25}$$

are plotted for different numbers of moments. Since the expectation value is smoother than the solution, we expect a faster convergence to the exact solution. The order of convergence appears to be in the order of one. Again, the filtered SG yields a smaller error and is approached by the SG error values for increasing truncation order N.

N	5	10	20	25	30	35	40	45	50	55
runtime	77.1	426.5	1072.0	1592.4	2386.7	3443.0	5111.0	6449.3	8528.0	11762.7
error	0.601	0.278	0.207	0.204	0.209	0.208	0.213	0.213	0.213	0.212
error E[u]	6.17E-2	2.21E-2	8.22E-3	6.85E-3	7.0E-3	7.1E-3	6.93E-3	7.03E-3	7.03E-3	6.94E-3

TABLE 5.3: Runtime (in seconds) and L^2 error (according to (5.24)) and L^2 error of E[u] (according to (5.25)) for the IPM method as depicted in Figures 5.2 and 5.3.

The IPM yields a smaller error for the expectation value. Compared to the convergence of the solution, the error of the expectation value decreases until 25 moments, after which the discretization error dominates the overall error in the solution.

However, a main challenge of IPM is its increased numerical costs. Approaches to circumvent this are efficient high-order numerical schemes for the spatial and time discretization [67] as well as parallelization [39]. In the following, we compare the resulting error of the expectation value for a given runtime in Figure 5.3. All three methods are run on a desktop computer without parallelization. It can be seen that the efficiency curve of the Lasso filter lies below the other methods for most runtimes, i.e. the resulting error is the smallest for a given runtime. The IPM lies below the Lasso curve for very long computation times.

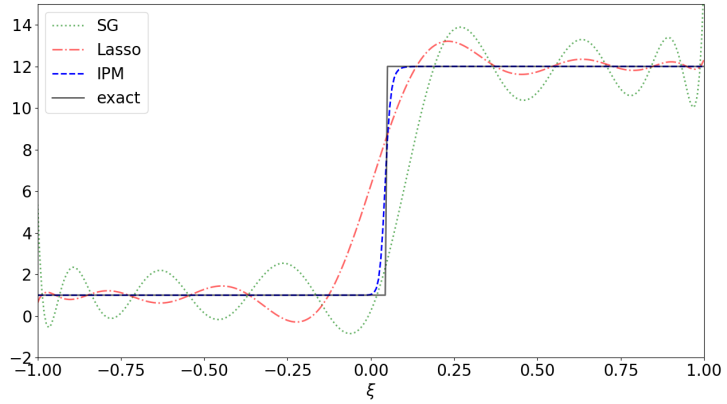


FIGURE 5.4: Solutions $u(t_{\text{end}}, x^*, \xi)$ at a fixed spatial position $x^* = 1.72$ and time $t_{\text{end}} = 0.11$ for Burgers' equation when using stochastic-Galerkin, filtered stochastic-Galerkin with the Lasso filter and IPM.

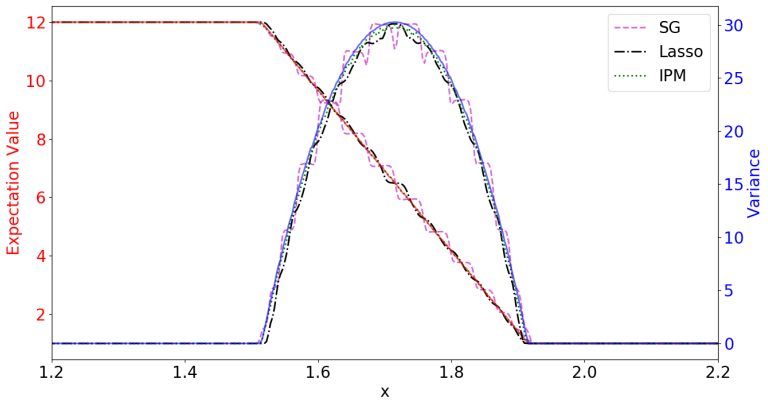


FIGURE 5.5: Expectation value and variance for Burgers' equation at time $t_{\text{end}} = 0.11$. The exact expectation value is depicted by the red and the exact variance by the blue solid line.

We now take a look at the solution of the random Burgers' equation for a fixed truncation order $N = 15$ and compare the results of SG, Lasso and IPM. Since the

moment system of IPM cannot be integrated analytically, we make use of a sixty-point Gauss-Legendre quadrature. The comparison of all three methods for a fixed spatial position $x^* = 1.72$ is shown in Figure 5.4. We can see that IPM yields a well-resolved solution approximation, which fulfills the maximum principle. Both SG and filtered SG violate the maximum principle, yet these solutions come at a cheaper numerical cost. Compared to SG, the filtered SG shows dampened oscillations and a better capturing of the shock position. Note that the polynomial order of the filtered SG is 14 instead of 15, since the last moment always has a value of zero by the construction of the filter strength. Taking a look at the comparison of the expectation value in Figure 5.5, we can see that the SG result shows a step-like profile, thus yielding a non-satisfactory solution approximation. The IPM and the filtered SG can approximate the exact solution nicely. Note that Lasso shows a small step in the middle. Taking a look at the variance, we see that SG yields an oscillatory result. The variance computed with IPM lies closer to the exact variance than the variance coming from Lasso, however both methods yield a satisfactory approximation.

5.4.2 Euler 1D

Though we have managed to avoid an opening loss, in solving a scalar, hyperbolic problem, the true utility of this method will require demonstration on hyperbolic systems and multi-dimensional problems. In the following, we investigate the random Euler equations in one spatial dimension before solving problems in higher spatial dimensions. The Euler equations (1.82), are given by

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix} + \partial_x \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(\rho E + p) \end{pmatrix} = \mathbf{0},$$

with the initial conditions

$$\begin{aligned} \rho_{\text{IC}} &= \begin{cases} \rho_L & \text{if } x < x_{\text{interface}}(\xi) \\ \rho_R & \text{else} \end{cases}, \\ (\rho u)_{\text{IC}} &= 0, \\ (\rho E)_{\text{IC}} &= \begin{cases} \rho_L E_L & \text{if } x < x_{\text{interface}}(\xi) \\ \rho_R E_R & \text{else} \end{cases}. \end{aligned}$$

For more information such as the chosen constitutive laws, see Section 1.4.1. Due to the random interface position $x_{\text{interface}}(\xi) = x_0 + \sigma\xi$, the solution is uncertain. Again, we use a uniformly distributed random variable with $\Theta = [-1, 1]$. Similar to the Burgers' test case, Dirichlet boundary conditions are chosen at the left and right boundary. The underlying numerical flux $\boldsymbol{f}^*$ is the HLL-flux [50]. We use the following parameter values.

$x \in D = [0, 1]$	range of spatial domain
$N_x = 2000$	number of spatial cells
$t_{\text{end}} = 0.14$	end time
$x_0 = 0.5, \sigma = 0.05$	interface position parameters
$\rho_L, p_L = 1.0, \rho_R, p_R = 0.3$	initial states
$N = 15$	polynomial degree
$\tau = 10^{-7}$	gradient tolerance for IPM

Note that compared to Sod's shock tube as described in Section 1.4.1, we increase density and presssure at the right side of the physical domain to preserve realizable solution values. Note however that methods to enforce positivity of the SG solution such as hyperbolicity limiters [117] exist and can be combined with filters. Solutions of test cases discussed in this chapter remain admissible without hyperbolicity limiters.

The entropy used by the IPM method is (1.117), for which the IPM system (1.108) is guaranteed to be hyperbolic.

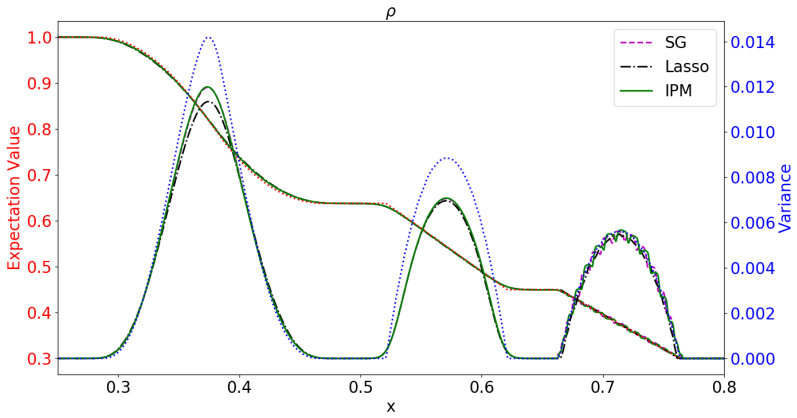
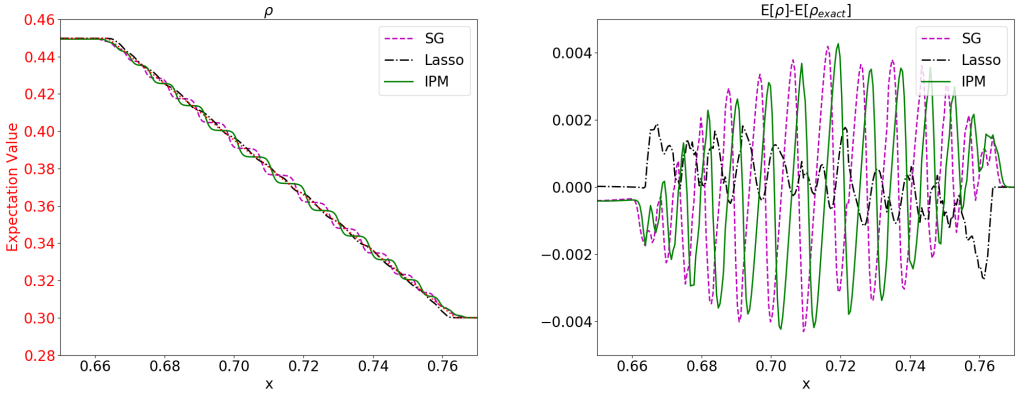


FIGURE 5.6: Expected value and variance of density at time $t_{\text{end}} = 0.14$ using $N_x = 2000$ spatial cells and $M = 16$ moments. The exact expectation value is given by the red dotted line and the exact variance is given by the blue dotted line. The exact solution of Sod's shock tube problem for a given ξ is determined from the literature (see Section 1.4.1) and we compute the expectation value and variance with a Gauss-Legendre quadrature rule using 100 quadrature points.



(A) Expectation value of the shock.

(B) Distance to exact solution at shock position.

FIGURE 5.7: Zoomed view of Figure 5.6 and corresponding difference to exact solution of the density shock.

We start by looking at the expected value of ρ in Figure 5.6. The exact solution (which is the dotted red line) shows the expected value for the rarefaction wave in the left, the contact discontinuity in the middle and the shock on the right side of the spatial domain. A non-zero variance is observed at these solution regions. The

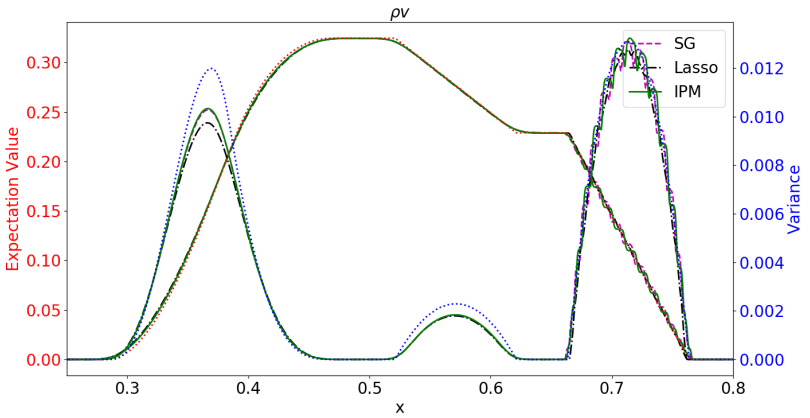


FIGURE 5.8: Expected value and variance of momentum.

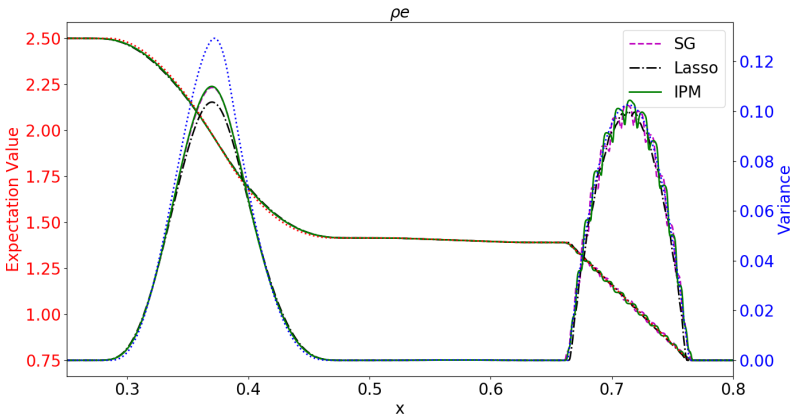


FIGURE 5.9: Expected value and variance of internal energy.

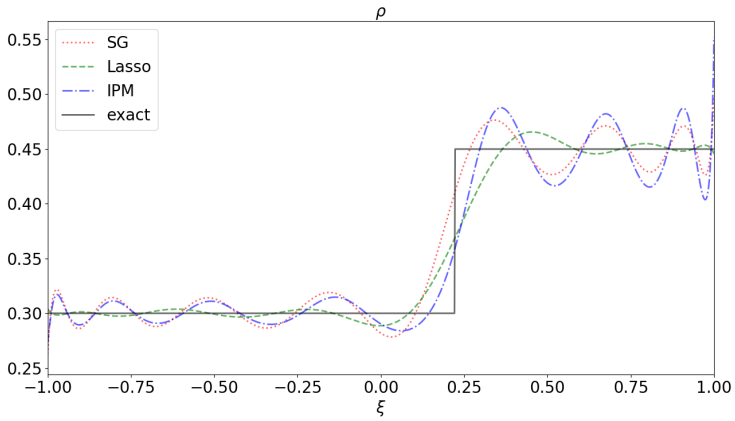


FIGURE 5.10: Density at fixed spatial position $x^* = 0.37$ and time $t_{\text{end}} = 0.14$.

exact variance is plotted by the blue dotted line. All three methods show a poor approximation of the variance, which is caused by the numerical diffusion of the finite volume scheme. The expected value is (except for the shock) nicely approximated by all three methods. A zoomed view of the shock can be found in Figure 5.7.

While SG and IPM show a step-like approximation, the Lasso yields a satisfactory approximation. The same holds for the variance of the shock, where SG and IPM show oscillations. Looking at the variance of the rarefaction wave, one notices that SG and IPM reach the value of the variance more closely. The same behavior can be found for the momentum in Figure 5.8 and the internal energy in Figure 5.9. Looking at the approximation of the density shock for a fixed spatial position $x^* = 0.37$ in Figure 5.10, one finds that the Lasso approximation, which is of polynomial order $N-1$ smears out the discontinuity, however damps oscillatory over- and undershoots while capturing the shock position nicely. The SG and IPM solutions show increased oscillations. Note that IPM especially oscillates at the right state, which leads to high density values. One needs to point out that in the chosen setting, the IPM yields no clear advantage compared to SG, since the choice of entropies is limited and we cannot prescribe upper and at the same time lower bounds as done for scalar problems. At the same time, the SG yields a similar result in a small portion of the computing time of IPM. However, for small densities both, the SG and Lasso will crash. By combining the Lasso method with a hyperbolicity preserving limiter [117], we are able to ensure admissible solutions.

5.4.3 Lightning strike with obstacles

The following problem investigates a high energy gas in the center of a two-dimensional spacial domain. This problem can be interpreted as constant along the z-axis in which case the high energy state occurs along a line along the z-direction, from which the test case gets the name "lightning strike". We study the random Euler equations in 2D as given in (4.26) with the initial conditions

$$\rho_{\text{IC}} = \begin{cases} \rho_L & \text{if } \|x\| < x_0 + \sigma\xi \\ \rho_R & \text{else} \end{cases}, \tag{5.26}$$

$$(\rho u)_{\text{IC}} = 0, \tag{5.27}$$

$$(\rho E)_{\text{IC}} = \begin{cases} \rho_L E_L & \|x\| < x_0 + \sigma\xi \\ \rho_R E_R & \text{else} \end{cases}. \tag{5.28}$$

Again, the pressure is given by (4.27), i.e. we have

$$p = (\gamma - 1)\rho\left(E - \frac{1}{2}(v_1^2 + v_2^2)\right).$$

The spatial domain is $D = [-0.3, 0.3] \times [-0.3, 0.3]$ and includes four square obstacles centered at positions $\boldsymbol{x}_{1,2,3,4}$ with length $l_{1,2,3,4}$. At the obstacles' boundaries, we use the Euler slip boundary condition. The calculations use the following parameter values:

$N_x = 700, N_y = 700$	number of spatial cells in each dimension
$t_{\text{end}} = 0.14$	end time
$x_0 = 0.05, \sigma = 0.05$	interface position parameters
$\rho_L, p_L = 1.0, \rho_R = 0.8, p_R = 0.3$	initial states
$N = 8$	polynomial degree
$\boldsymbol{x}_1 = (0, 0.15)^T, \boldsymbol{x}_2 = (0.1, 0)^T,$	obstacle positions
$\boldsymbol{x}_3 = (-0.1, 0.1)^T, \boldsymbol{x}_4 = (-0.1, 0)^T$	
$l_1 = 0.06, l_2 = 0.04, l_3 = 0.02, l_4 = 0.01$	obstacle length

Due to the fact that the spatial discretization consists of $N_x \cdot N_y$ cells and we have one additional equation (namely for the y-momentum), we do no longer study the results for IPM due to the much higher computational cost, and focus on comparing SG and the Lasso method. A reference solution has been computed using collocation with a 40 point Gauss-Lobatto quadrature set. The different columns of Figure 5.11 depict different methods to compute the solution, whereas the rows show the expectation value on the left and the variance on the right for the density. For both, expectation value and variance, stochastic-Galerkin yields oscillatory solutions. This is most obvious in the outer shock wave, but also reflected shocks suffer from oscillations as well. The Lasso filter yields results which agree nicely with the reference solution in both expected value and variance.

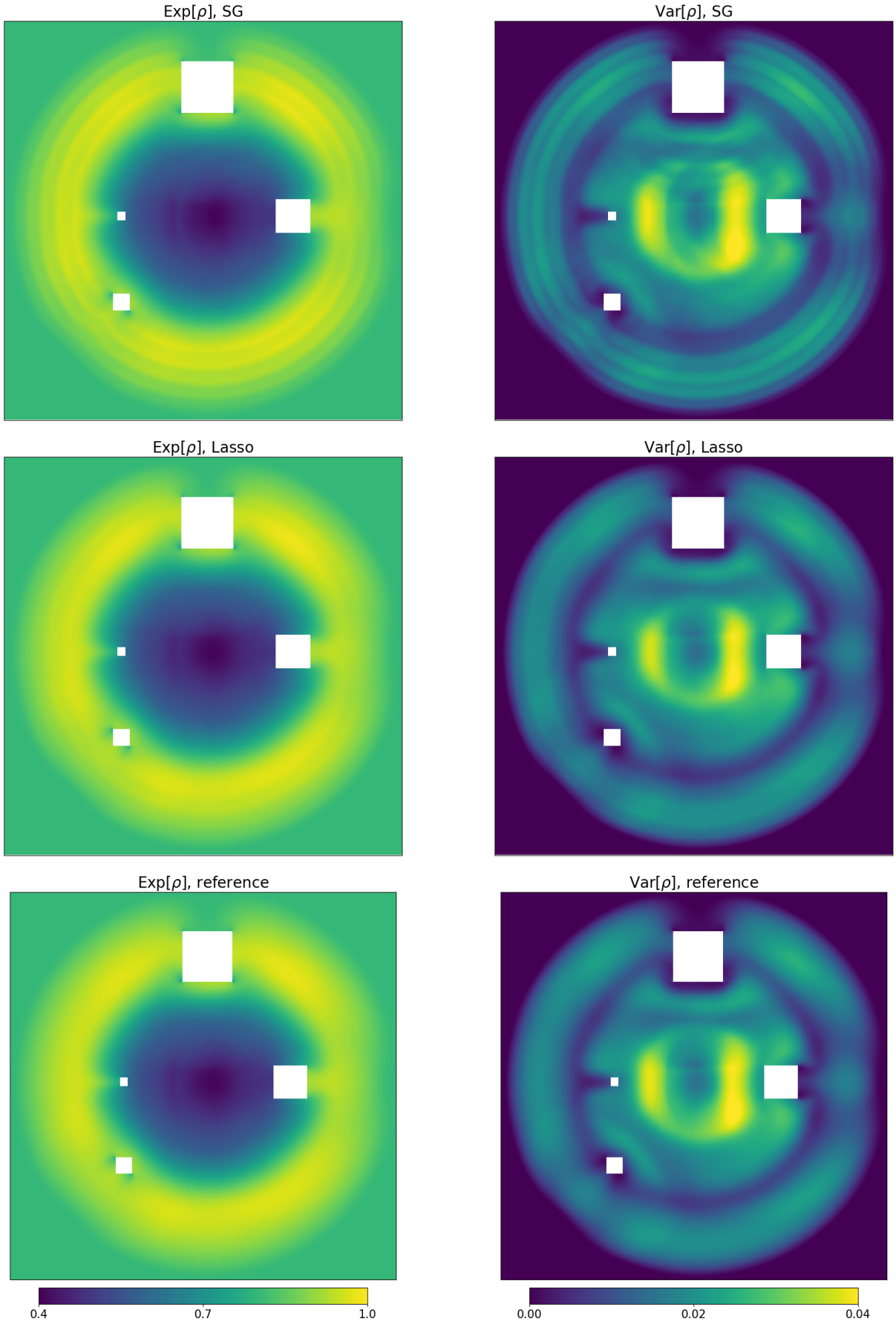


FIGURE 5.11: Expected value and variance with different methods plotted over the spatial domain.

5.4.4 Shock in a duct

The following test case is dedicated to comparing results obtained with different methods, including the L^2 filter. As before, we solve the two-dimensional Euler equations. As geometry, a duct is chosen. The initial condition is (5.26) with $x_0 = 0.5$, i.e. the gas is initially in a shock state with high density and pressure on the left hand side. Parameters, which differ from the settings in Section 5.4.3 are

$D = [0.0, 1.0] \times [0.0, 1.0]$	range of spatial domain
$N_x = 400, N_y = 400$	number of spatial cells in each dimension
$t_{\text{end}} = 0.35$	end time
$x_0 = 0.5, \sigma = 0.1$	interface position parameters

Again, a reference solution has been calculated with collocation using 40 Gauss-Lobatto quadrature points. To use the L^2 filter, we need to conduct a parameter study to obtain an adequate filter coefficient λ. Due to the test case's similarity to the one-dimensional shock tube, we can perform a parameter study for a one-dimensional shock using the same numerical parameters as in two dimensions. Because of the heavily reduced numerical costs, we were able to find a suitable filter parameter of $\lambda = 3.0 \cdot 10^{-6}$. The Lasso filter, as before, picks the filter parameter automatically. The results of the expectation value can be found in Figure 5.12 and the variance is depicted in Figure 5.13.

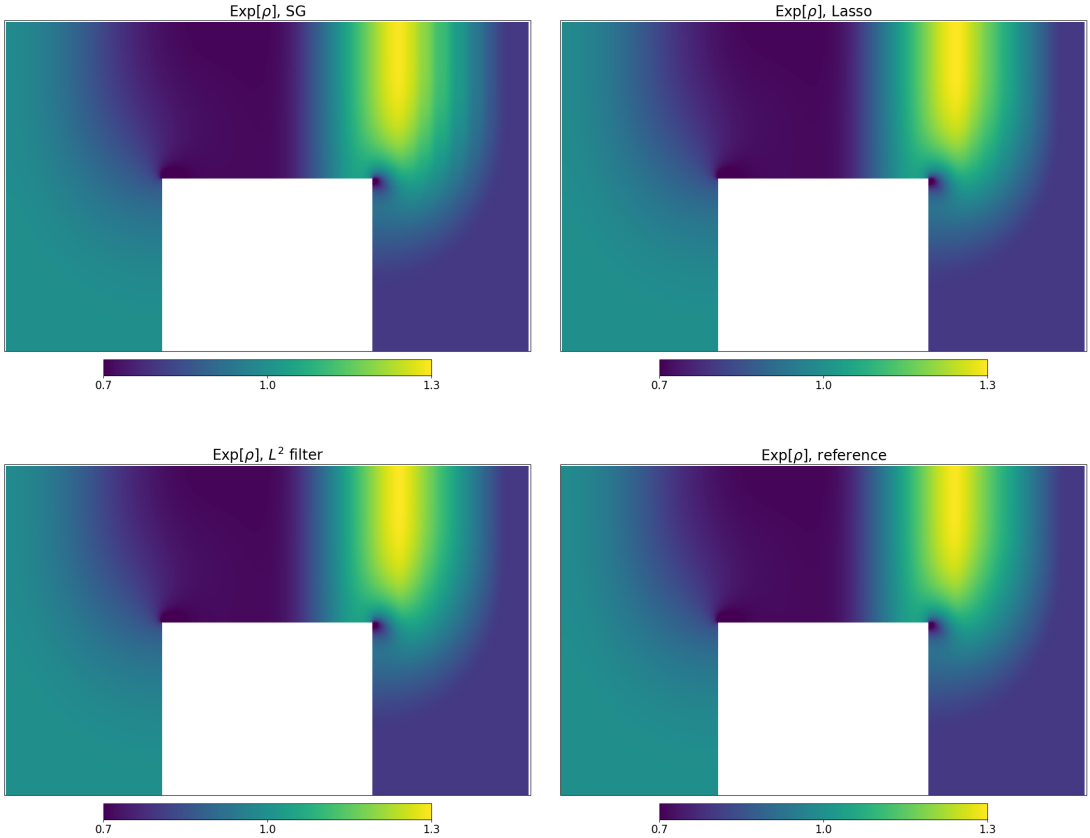


FIGURE 5.12: Expected value with different methods plotted over the spatial domain $[0, 1] \times [0.3725, 1]$.

Again, the SG solution of both expected value and variance shows non-physical oscillations at the front shock. Both filters are able to mitigate these oscillations, however, the L^2 filter yields more accurate results. Note, however that the chosen filter

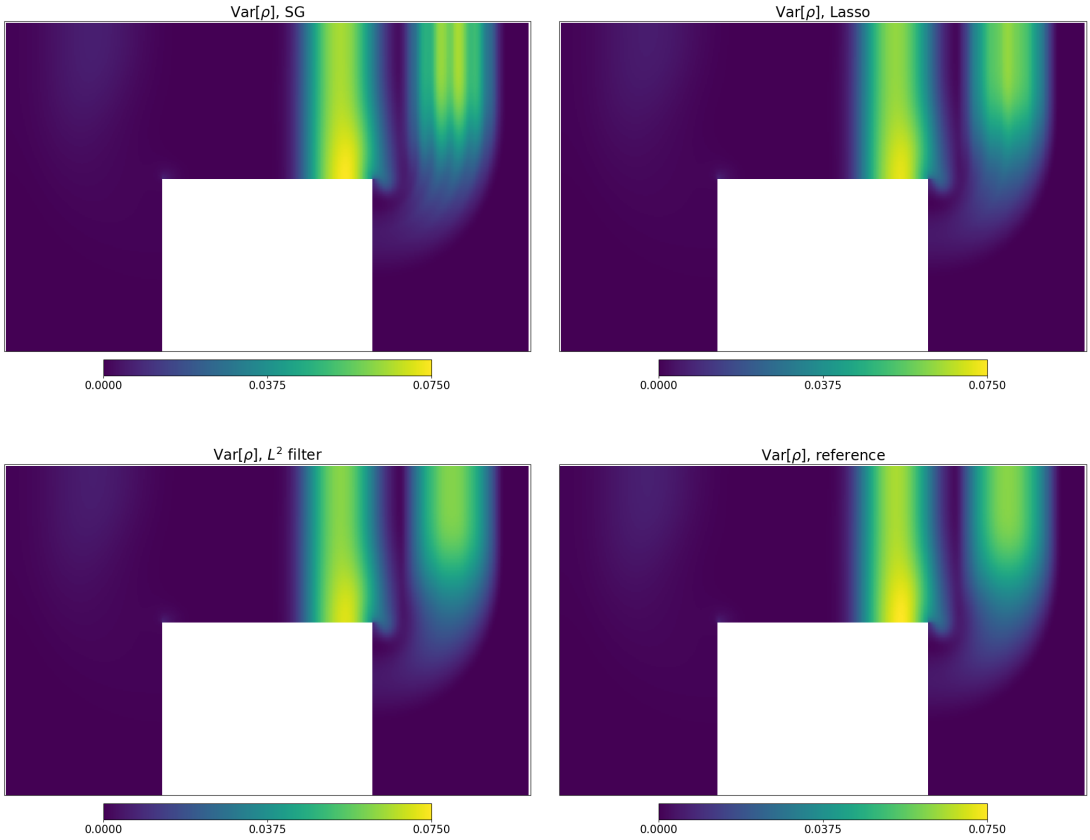


FIGURE 5.13: Variance with different methods plotted over the spatial domain $[0,1] \times [0.3725, 1]$.

strength was determined by a parameter study, which in general is prohibitively expensive if one cannot use simpler, one-dimensional problems to model filtering effect of two-dimensional problems.

5.4.5 Nozzle with shock

In the following, we test the effects of filtering for a diverging nozzle geometry ranging from $x_1 \in [0, 14]$ and $x_2 \in [0, 6]$. The geometry consists of a chamber (left, $x_1 \in [0, 4]$), a throat (middle, $x_1 \in [4, 5]$) and the nozzle (right, $x_1 \in [5, 14]$). Again, we are interested in quantifying the uncertainty of a shock with an uncertain shock position. The shock occurs inside the throat and the shock position is uniformly distributed within the entire throat section. The mesh is composed of 76696 triangular elements. The shock states are chosen to equal the settings from Section 5.4.3. Note that the quality of the Lasso filter strength will depend on the number of moments (which are the parameter that directly determines the filter strength). To test the quality at a reduced number of moments, we choose $N = 5$. At results are shown for $t_{\text{end}} = 4.0$.

To check the quality of our results, we compute a reference solution using 50 Gauss-Legendre quadrature points. For the L^2 filter, we again choose a filter parameter of $\lambda = 3.0 \cdot 10^{-6}$. The resulting expectation value can be found in Figure 5.12 and the variance is depicted in Figure 5.13.

The shock behaves similarly to the test cases from before: For the expectation value, shown in Figure 5.14, one observes a density peak, which travels to the right through

the nozzle. At the front, the reference solution shows a linear profile, which is generated by the shock. As before, the SG solution does not capture this solution structure satisfactorily as it yields a step-like approximation. The solution of both filtering methods show good agreement with the reference solution. Looking at the variance in Figure 5.15, the reference solution shows two waves with high variance, the left belonging to the contact discontinuity and the right which is generated by the shock. Here, SG shows clear oscillations at the front wave, which do not appear for both filtering methods. However, again the L^2 filter yields an improved approximation over Lasso. Note, that even for a reduced number of moments, the Lasso filter yields a satisfactory approximation. One should also point out that the variance structure of the contact discontinuity is slightly damped by the filter, which we also observed for Sod's shock tube in Section 5.4.2

Again, the SG solution of both expected value and variance shows non-physical oscillations at the front shock. Both filters are able to mitigate these oscillations, however, the L^2 filter yields more accurate results. Note, however that the chosen filter strength was determined by a parameter study, which in general is prohibitively expensive if one cannot use simpler, one-dimensional problems to model filtering effect of two-dimensional problems.

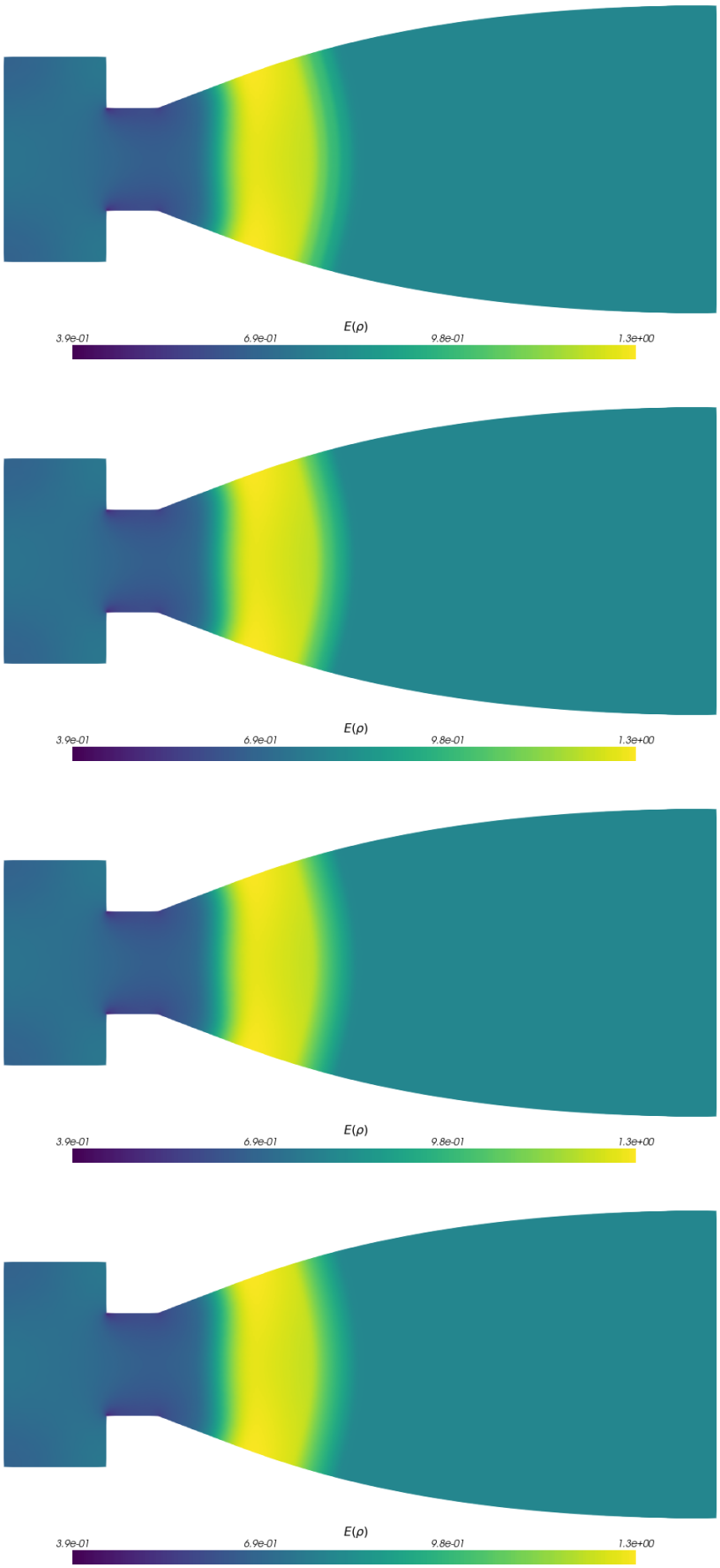


FIGURE 5.14: Expected value with different methods. From top to bottom: SG, Lasso, L^2, reference.

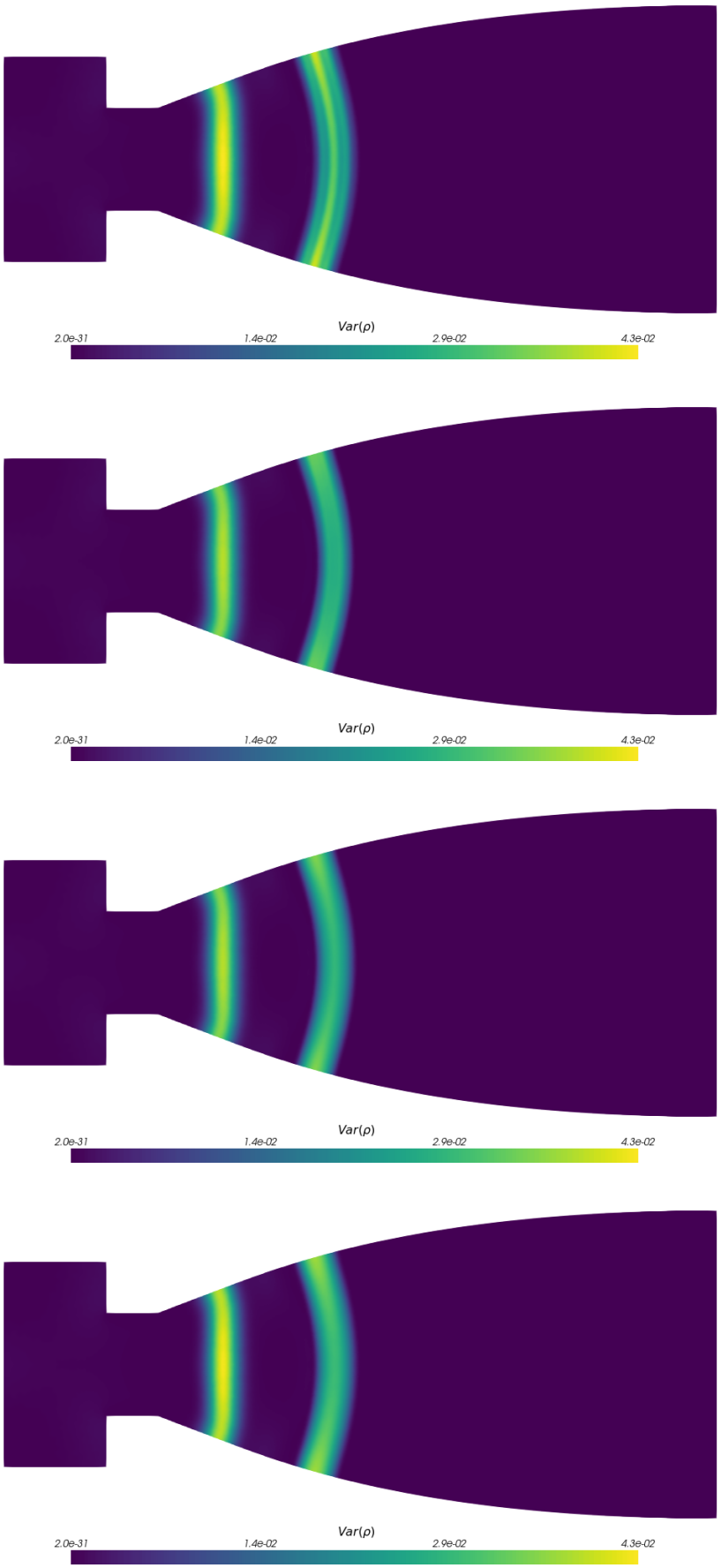


FIGURE 5.15: Variance with different methods. From top to bottom: SG, Lasso, L^2, reference.

Chapter 6

Filtered IPM method

So far, we have introduced a filter to the stochastic-Galerkin method, enabling the mitigation of step-like profiles as well as oscillations when using SG. However, despite the use of filters, the solution can still become non-realizable. In the context of fluid dynamics, this means that the solution can show negative density, pressure and energy values. Therefore, the objective of this chapter is to incorporate a filter into the IPM closure to suppress oscillations in the solution while maintaining a realizable solution.

There are many ways to incorporate filters into IPM closures, so one must make several choices when designing the closure in order to achieve the desired properties, not only non-oscillatory solutions and realizability but also low added computational cost. To suppress oscillations, first we simply apply the filter to the numerical solution of (1.108) between time steps. But since an important property of the optimization problem is that it is infeasible when the given moment vector is not realizable, we must consider two options: either the filter must be chosen to preserve realizability or the optimization problem must be modified to become feasible for nonrealizable moment vectors.

Before proceeding, we note that there are other ways to enforce the physical bounds of the solution and dampen oscillations in the closure. For example, one could enforce the physical bounds through constraints in the optimization problem instead of using the entropy s in the objective function. Also, instead of filtering the moment vector directly, oscillations could be dampened by penalizing them in the objective function. These alternatives also likely merit investigation, but we leave this for future work.

The rest of this chapter is structured as follows: First we discuss how applying a standard filter to a realizable moment vector does not necessarily preserve its realizability in Section 6.1. A realizability preserving filter which acts on the underlying kinetic description is derived in Section 6.2. Then in the following Section 6.3, we apply the regularization technique from [4] to the optimization problem, thus making it feasible even for nonrealizable moment vectors and allowing the use of standard filters. Section 6.4 discusses the implementation of both strategies, followed by results for the Euler equations in one and two spatial dimensions in Section 6.5.

6.1 Realizability of filtered moments

We consider two ways of incorporating a filter into the IPM method. First, we show that the filtered moments $\mathcal{L}(\lambda)\hat{\boldsymbol{u}}$ are not guaranteed to fulfill the realizability condition[1] (even when $\hat{\boldsymbol{u}}$ does) and therefore propose a modified filter which does preserve realizability. Second, we consider an alternative which modifies the IPM optimization problem (1.120) instead of the filter so that the problem is feasible even for nonrealizable moment vectors.

[1] Here, we only consider the L^2 filter defined in the previous chapter.

As already mentioned, combining filtering and minimal entropy methods requires a deeper study of realizability, which reveals that filtered moments $\mathcal{L}(\lambda)\hat{\boldsymbol{u}}$ are not necessarily realizable, even when the moments $\hat{\boldsymbol{u}}$ are. Let us consider the scalar case, $m = 1$ and the entropy

$$s(u) = u\log(u). \tag{6.1}$$

The corresponding realizable set is

$$\mathcal{R} := \left\{ \hat{\boldsymbol{u}} \in \mathbb{R}^{N+1} | \exists u : \Theta \to (0,\infty) \text{ such that } \hat{\boldsymbol{u}} = \langle \boldsymbol{\varphi} u \rangle \right\}, \tag{6.2}$$

i.e. we wish to preserve positivity of the solution, i.e. $u(\xi) > 0$ for all $\xi \in \Theta$, where we assume $\Theta = [-1,1]$. Then, for monomial basis functions with moments $\tilde{u}_i := \langle u\ \xi^i \rangle$, the Hankel matrix $A(N) := (\tilde{u}_{i+j})_{i,j=0}^{i+j\leq N}$ is positive semi definite if the solution is realizable (i.e. positive) [12, 120, 22]. Consequently, for $N = 2$ we must have

$$\det \begin{pmatrix} \tilde{u}_0 & \tilde{u}_1 \\ \tilde{u}_1 & \tilde{u}_2 \end{pmatrix} \geq 0. \tag{6.3}$$

However, this condition does not suffice to guarantee realizability. Defining $B(n) := (\hat{u}_{i+j+1})_{i,j=0}^{n}$ and $C(n) := (\hat{u}_{i+j})_{i,j=1}^{n}$, for $\xi \in [a,b]$ we additionally must have that

$$(a+b)B(0) \geq abA(0) + C(1).$$

Using $a = -1, b = 1$ we get

$$0 \geq -\tilde{u}_0 + \tilde{u}_2. \tag{6.4}$$

The moments $\tilde{\boldsymbol{u}}$ are realizable iff conditions (6.3) and (6.4) are fulfilled. Note that the first three orthonormal Legendre polynomials are $\varphi_0(\xi) = 1, \varphi_1(\xi) = \sqrt{3}\xi$ and $\varphi_2(\xi) = \sqrt{5}\left(3/2\xi^2 - 1/2\right)$. Switching back to Legendre moments and setting $\hat{u}_0 = 1$ we have with $\tilde{u}_0 = \hat{u}_0 = 1, \tilde{u}_1 = \frac{1}{\sqrt{3}}\hat{u}_1$ and $\tilde{u}_2 = \langle u\ \xi^2 \rangle = \langle u\ (\frac{2}{3\sqrt{5}}\varphi_2 + 1/3) \rangle$ that condition (6.3) gives

$$\det \begin{pmatrix} 1 & \frac{1}{\sqrt{3}}\hat{u}_1 \\ \frac{1}{\sqrt{3}}\hat{u}_1 & \frac{2}{3\sqrt{5}}\hat{u}_2 + \frac{1}{3} \end{pmatrix} \geq 0.$$

Hence, if we assume that the moment vector $(1, \hat{u}_1, \hat{u}_2)^T$ is realizable, the condition $\frac{2}{\sqrt{5}}\hat{u}_2 + 1 \geq \hat{u}_1^2$ holds. From condition (6.4) we get that

$$\tilde{u}_2 \leq 1 \Rightarrow \frac{2}{3\sqrt{5}}\hat{u}_2 + \frac{1}{3} \leq 1,$$

hence $\hat{u}_2 \leq \sqrt{5}$ and when plugging this into the first condition, i.e. $\hat{u}_1^2 \leq \frac{2}{\sqrt{5}}\hat{u}_2 + 1$ we obtain $\hat{u}_1 \in [-\sqrt{3}, \sqrt{3}]$. Let us now determine the boundary of realizability, i.e. the moments for which the determinant is zero. These moments fulfill $\frac{2}{\sqrt{5}}\hat{u}_2 + 1 = \hat{u}_1^2$. Hence, the boundary of realizability (for a given $\hat{u}_1$) reads

$$\phi(\hat{u}_1) := \left(\hat{u}_1, \frac{\sqrt{5}}{2}(\hat{u}_1^2 - 1) \right)^T, \qquad \text{with } \hat{u}_1 \in [-\sqrt{3}, \sqrt{3}]. \tag{6.5}$$

Now we check how filtering changes the realizable domain, hence we simply need to

replace $\hat{u}_i$ by $g_\lambda(i)\hat{u}_i$ in the realizability study above. In this case, condition (6.4) directly yields $\hat{u}_2 \leq \frac{\sqrt{5}}{g_\lambda(2)}$. For condition (6.3) we need to check that

$$\det \begin{pmatrix} 1 & \frac{1}{\sqrt{3}} g_\lambda(1)\hat{u}_1 \\ \frac{1}{\sqrt{3}} g_\lambda(1)\hat{u}_1 & \frac{2}{3\sqrt{5}} g_\lambda(2)\hat{u}_2 + \frac{1}{3} \end{pmatrix} \geq 0,$$

i.e. for the filtered Hankel matrix A_λ one must show

$$\det A_\lambda = \frac{2}{3\sqrt{5}} g_\lambda(2)\hat{u}_2 + \frac{1}{3} - \frac{1}{3}\left(g_\lambda(1)\hat{u}_1\right)^2 \overset{!}{\geq} 0. \tag{6.6}$$

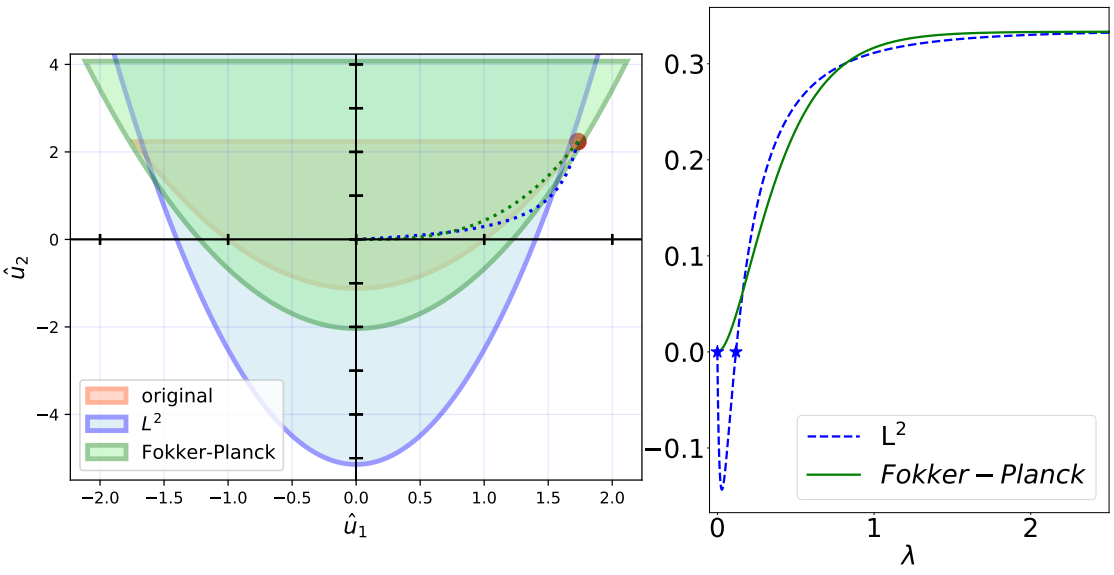
Rearranging leads to

$$\hat{u}_1^2 \leq \frac{1}{g_\lambda(1)^2}\left(\frac{2}{\sqrt{5}} g_\lambda(2)\hat{u}_2 + 1\right),$$

which yields $\hat{u}_1 \in [-\sqrt{3}/g_\lambda(1), \sqrt{3}/g_\lambda(1)]$. Hence, the filtered boundary of realizability is given by

$$\phi_\lambda(\hat{u}_1) := \left(\hat{u}_1, \frac{\sqrt{5}}{2g_\lambda(2)}\left(g_\lambda(1)^2\hat{u}_1^2 - 1\right)\right)^T, \qquad \text{with } \hat{u}_1 \in \left[-\frac{\sqrt{3}}{g_\lambda(1)}, \frac{\sqrt{3}}{g_\lambda(1)}\right]. \tag{6.7}$$

The original realizable domain, defined by (6.5), as well as its filtered versions (6.7)



(A) (B)

FIGURE 6.1: (A) Realizability domains for a fixed filter strength $\lambda = 0.1$ when using the L^2 and Fokker–Planck filtering. The lines indicated the path of the moments when filtering is applied with $\lambda = 0$ at the red dot and $\lambda \to \infty$ at $(0,0)$. The green line indicates Fokker–Planck filtering, the blue line is L^2 filtering. (B) Determinant of the filtered Hankel matrices plotted over the filter strength. For L^2 filtering, the determinant is zero for $\lambda \in \{0, 0.1187\}$ which we mark by a star.

when using the L^2 filter (5.6) and the Fokker–Planck filter which uses $g_\lambda(i) = e^{-\lambda i(i+1)}$ are plotted in Figure 6.1A for a fixed filtering strength. The derivation of the Fokker–Planck filter will be postponed to the next section. The colored areas indicate which of the first as well as second-order moments will be realizable. In the case of filtering, the colored areas show the moments which will be realizable after applying filtering with a filter strength of $\lambda = 0.1$. For the L^2 filter, there exist moment vectors which lie in the original realizable domain but not in the filtered version. Hence, after applying L^2 filtering, these moments will no longer be realizable, meaning that the standard IPM method will fail when trying to solve the dual problem (1.122b) with filtered moments. At least in this study, the Fokker–Planck filters appears to preserve realizability. Before discussing methods to cope with the loss of realizability due to L^2 filtering, we will further investigate realizability for varying filter strengths: Instead of fixing the filter strength as before, we will now fix the moment vector and investigate how the choice of λ affects realizability. We choose a moment vector which lies on the boundary of realizability, in which case the Hankel matrix becomes singular [12] and we therefore have $\frac{2}{\sqrt{5}}\hat{u}_2 + 1 = \hat{u}_1^2$. We plug this into condition (6.6) and check if the L^2 filter fulfills this condition. Making use of $g_\lambda(i) = 1/(1 + \lambda i^2(i+1)^2)$ (L^2 filtering), the filter strength $\lambda \geq 0$ must fulfill

$$\det A_\lambda = \frac{2}{3\sqrt{5}\cdot(1+36\lambda)}\hat{u}_2 + \frac{1}{3} - \frac{1}{3(1+4\lambda)^2}\left(\frac{2}{\sqrt{5}}\hat{u}_2 + 1\right) \overset{!}{\geq} 0.$$

The roots of the determinant are given by

$$\lambda \in \left\{0, -\frac{1}{360}\left((95 + 2\sqrt{5}\hat{u}_2) \pm \sqrt{7225 + 2900\sqrt{5}\hat{u}_2 + 20\hat{u}_2^2}\right)\right\}, \tag{6.8}$$

and it is easy to see that one of these roots is always negative, hence invalid. Furthermore, for $\lambda \to \infty$, we know that the filtered moments will be realizable. Therefore, we know that the determinant of the filtered Hankel matrix will be negative for

$$\lambda \in \left(0, -\frac{1}{360}\left((95 + 2\sqrt{5}\hat{u}_2) - \sqrt{7225 + 2900\sqrt{5}\hat{u}_2 + 20\hat{u}_2^2}\right)\right).$$

Hence, to generate filtered moments which are realizable, we must at least choose a filter strength outside this interval. Before further studying this behavior, we will apply the same study using the Fokker–Planck filter, i.e. $g_\lambda(i) = e^{-\lambda i(i+1)}$. In this case, the determinant of the filtered Hankel matrix changes to

$$\det A_\lambda = \frac{2}{3\sqrt{5}}e^{-6\lambda}\hat{u}_2 + \frac{1}{3} - \frac{1}{3}e^{-4\lambda}\left(\frac{2}{\sqrt{5}}\hat{u}_2 + 1\right) \overset{!}{\geq} 0.$$

The roots of $\det A_\lambda$ are then given by

$$\lambda \in \left\{0, -\frac{1}{2}\ln\left(\frac{\sqrt{5}}{4\hat{u}_2} \pm \frac{1}{4}\sqrt{(5 + 8\sqrt{5}\hat{u}_2)/\hat{u}_2^2}\right)\right\}. \tag{6.9}$$

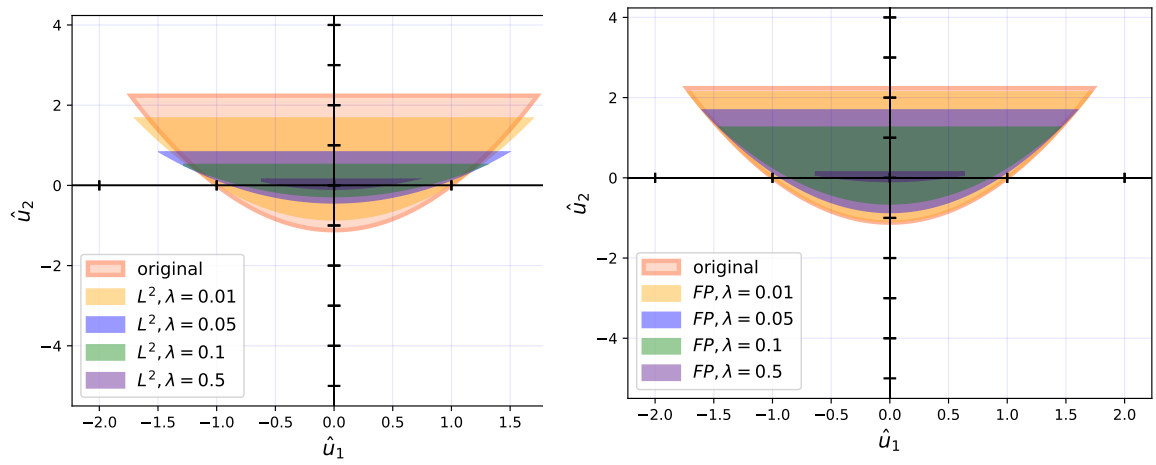
Let us fix $\hat{u}_1 = \sqrt{3}$ and choose $\hat{u}_2$ s.t. we lie on the boundary of realizability, i.e. $\hat{u}_2 = \sqrt{5}/2 \cdot (\hat{u}_1^2 - 1) = \sqrt{5}$. Studying the determinant of the filtered Hankel matrix in Figure 6.1B for different filter strengths, we observe that the L^2 filter yields non-realizable moments for certain filter strengths, namely in between the two roots that we determined, which are marked by a star. In the case of the Fokker–Planck filter, all roots of the determinant are equal to zero. Hence there is no positive filter strength for which

the determinant is negative. We now plot the path that the filtered moments take for increasing filter strengths in Figure 6.1A. For $\lambda = 0$, we start at the previously chosen moment vector with $\hat{u}_1 = \sqrt{3}$ and $\hat{u}_2 = \sqrt{5}$ (marked with a red dot). As λ increases, we move along the green and blue dotted lines. One can see that L^2 filtering moves the moment vector out of the realizable domain and, when sufficiently increasing the filter strength, the filtered moments are again pushed into the realizable domain. As expected, the Fokker–Planck filter generates moments which remain realizable for all filter strengths, i.e. the path remains in the original realizable domain. When λ goes to infinity, the filtered moments are pushed to zero.

Lastly, we plot the domain of realizability for fixed filter strengths in Figure 6.2A. Here, we see that $\mathcal{L}(\lambda)\hat{\boldsymbol{u}}$ may be not realizable even when $\hat{\boldsymbol{u}}$ is. In this figure, we plot the original, unfiltered realizable set $\mathcal{R}|_{\hat{u}_0=1}$ as well as

$$\mathcal{L}(\lambda)\mathcal{R}|_{\hat{u}_0=1} = \{\mathcal{L}(\lambda)\hat{\boldsymbol{u}} \,|\, \hat{\boldsymbol{u}} \in \mathcal{R} \text{ and } \hat{u}_0 = 1\}$$

and see that for $\lambda \in \{0.01, 0.05, 0.1\}$ we have $\mathcal{L}(\lambda)\mathcal{R}|_{\hat{u}_0=1} \not\subset \mathcal{R}|_{\hat{u}_0=1}$, i.e., there are realizable moments which become nonrealizable after the L^2 filter is applied.



(A) (B)

FIGURE 6.2: Images of the realizable set after application of filters. (A) L^2 filter, (B) Realizability-preserving filter

On the other hand, in Figure 6.2B we see the image of the realizable set $\mathcal{R}$ after the application of the Fokker–Planck filter, $F(\lambda) := \text{diag}\{e^{-\lambda i(i+1)}\}_{i=0}^{N}$, given below. Here one sees that $F(\lambda)\mathcal{R}|_{\hat{u}_0=1} \subset \mathcal{R}|_{\hat{u}_0=1}$ for each λ.

6.2 A realizability-preserving filter

So far, we have seen that the Fokker–Planck filter preserves realizability for the simplified setting from the previous section when using $N = 2$. In the following, we derive this filter and show that it generates realizable moments for arbitrarily truncation orders N as well as different physical problems. Let ξ be uniformly distributed on $\Theta = [-1, 1]$, and consider the Fokker–Planck equation for $u = u(\lambda, \xi)$, where λ plays the role of time:

$$\partial_\lambda u = Lu \qquad \text{with} \qquad u(\lambda = 0, \xi) = u_{\text{ex}}(\xi),$$

where L is is the differential operator (1.87), which has been used to construct the L^2 filter. Since the solution to this equation fulfills the maximum principle, as long as we assume $u_{ex}(\xi) \in (u_-, u_+)$ for all $\xi \in \Theta$, we know $u(\lambda, \xi) \in (u_-, u_+)$ for all $\xi \in \Theta$ and $\lambda \geq 0$. Therefore the moments

$$\langle \boldsymbol{\varphi} u(\lambda, \cdot) \rangle \tag{6.10}$$

remain in the more general realizable set

$$\mathcal{R} := \left\{ \hat{\boldsymbol{u}} \in \mathbb{R}^{N+1} | \exists u : \Theta \to (u_-, u_+) \text{ such that } \hat{\boldsymbol{u}} = \langle \boldsymbol{\varphi} u \rangle \right\},$$

for all $\lambda \in [0, \infty)$. Furthermore, these moments are straightforward to compute when each φ_i is the i-th Legendre polynomial, an eigenfunction of the self-adjoint operator L:

$$\partial_\lambda \langle \varphi_i u \rangle = \langle \varphi_i L u \rangle = \langle (L\varphi_i) u \rangle = -i(i+1) \langle \varphi_i u \rangle, \tag{6.11}$$

and thus

$$\langle \varphi_i u(\lambda, \cdot) \rangle = \exp(-i(i+1)\lambda) \langle \varphi_i u_{ex} \rangle. \tag{6.12}$$

Altogether with $\hat{\boldsymbol{u}} = \langle \boldsymbol{\varphi} u_{ex} \rangle$ and the definition of $F(\lambda) = \text{diag}\{\exp(-i(i+1)\lambda\}_{i=0}^N$, we can write the filtered moment vector as $F(\lambda)\hat{\boldsymbol{u}} = \langle \boldsymbol{\varphi} u(\lambda, \cdot) \rangle$. Note that $\langle \varphi_0 u(\lambda, \cdot) \rangle$ is constant in λ and furthermore, as $\lambda \to \infty$, the solution $u(\lambda, \xi)$ becomes constant in ξ. Thus λ indeed acts like a filter-strength parameter. The realizability of $F(\lambda)\hat{\boldsymbol{u}}$ is ensured by the maximum-principle satisfied by $u(\lambda, \xi)$.

Next we discuss how to extend this realizability-preserving filter to systems of conservation laws. First we define realizability more precisely for systems. Let $\mathcal{R}_{\boldsymbol{u}} \subseteq \mathbb{R}^m$ be the set of admissible states of the system. For example, above in the scalar case we took $\mathcal{R}_{\boldsymbol{u}} = (0, \infty)$ (although we could have just as easily taken $\mathcal{R}_{\boldsymbol{u}} = (u_{\min}, u_{\max})$, where $u_{\min}$ and $u_{\max}$ bound the initial condition, in order to enforce the maximum principle). In the case of the Euler equations, one would take

$$\mathcal{R}_{\boldsymbol{u}} = \left\{ (\rho, \rho v, \rho E) \in \mathbb{R}^{d+2} : \rho > 0, p > 0 \right\} = \left\{ (\rho, \rho v, \rho E) \in \mathbb{R}^{d+2} : \rho > 0, e > \frac{1}{2}|u|^2 \right\} \tag{6.13}$$

which is the set of $\boldsymbol{u}$ for which the Euler equations are hyperbolic. For a $\boldsymbol{u}$ which depends on the uncertainty ξ, let $\hat{\boldsymbol{u}} = (\hat{\boldsymbol{u}}_0, \hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_N)$ be the vector of moments with respect to the basis functions $\boldsymbol{\varphi}$. We call $\hat{\boldsymbol{u}}$ realizable when it belongs to the set

$$\mathcal{R} := \left\{ \hat{\boldsymbol{u}} \in \mathbb{R}^{(N+1)\times m} : \exists \boldsymbol{u} = \boldsymbol{u}(\xi) \in \mathcal{R}_{\boldsymbol{u}} \text{ for all } \xi \in \Theta \text{ such that } \hat{\boldsymbol{u}} = \langle \boldsymbol{\varphi} \boldsymbol{u} \rangle \right\}. \tag{6.14}$$

We assume that the underlying admissible set has the form

$$\mathcal{R}_{\boldsymbol{u}} := \left\{ \boldsymbol{u} \in \mathbb{R}^m : \exists f = f(v) \in I \subset \mathbb{R} \text{ for all } v \in V \subseteq \mathbb{R}^d \text{ such that } \boldsymbol{u} = \int_V \boldsymbol{\psi}(v) f(v)\, dv \right\} \tag{6.15}$$

where $\boldsymbol{\psi}$ is a given set of m basis functions in v, and I is usually $(0, \infty)$ or $(0, 1)$.[2] This is the case for the Euler equations (where the f is the Maxwellian, $\boldsymbol{\psi}(v) = (1, v, \frac{1}{2}|v|^2)$, and $I = (0, \infty)$) and more generally for entropy-based moment closures for kinetic

[2] For conservation laws derived from kinetic equations, v plays the role of the velocity variable from the underlying kinetic equation.

equations [80]. Under this assumption, $\mathcal{R}$ can be written as

$$\mathcal{R} = \Big\{ \hat{\boldsymbol{u}} \in \mathbb{R}^{(N+1)\times m} : \exists f = f(\xi, v) \in I \text{ for all } (\xi, v) \in \Theta \times V \\ \text{such that } \hat{\boldsymbol{u}} = \left\langle \int_V \psi(v) f(\cdot, v)\, dv \boldsymbol{\varphi}^T \right\rangle^T \Big\}. \tag{6.16}$$

Now, to filter a $\hat{\boldsymbol{u}} \in \mathcal{R}$ without destroying its realizability, let f be any distribution from (6.16) and then apply L to its ξ-dependence to define g:

$$\partial_\lambda g(\lambda, \xi, v) = Lg(\lambda, \xi, v), \qquad g(0, \xi, v) = f(\xi, v). \tag{6.17}$$

Again, the maximum principle for L ensures $g(\lambda, \xi, v) \in I$ for all $(\lambda, v, \xi) \in [0, \infty) \times V \times \Theta$. Thus

$$\hat{\boldsymbol{u}}(\lambda) := \left\langle \int_V \psi g(\lambda, \cdot, v)\, dv \boldsymbol{\varphi}^T \right\rangle^T \in \mathcal{R} \tag{6.18}$$

for all $\lambda \in [0, \infty)$. A practical expression for $\hat{\boldsymbol{u}}$ is also straightforward to derive, since

$$\begin{aligned} \partial_\lambda \hat{\boldsymbol{u}}_i &= \left\langle \varphi_i \int_V \psi \partial_\lambda g(\lambda, \cdot, v)\, dv \right\rangle \\ &= \left\langle \varphi_i \int_V \psi L g(\lambda, \cdot, v)\, dv \right\rangle \\ &= \left\langle L\varphi_i \int_V \psi g(\lambda, \cdot, v)\, dv \right\rangle \\ &= -i(i+1) \left\langle \varphi_i \int_V \psi g(\lambda, \cdot, v)\, dv \right\rangle \\ &= -i(i+1)\hat{\boldsymbol{u}}_i, \end{aligned}$$

i.e., $\hat{\boldsymbol{u}}_i(\lambda) = e^{-i(i+1)\lambda}\hat{\boldsymbol{u}}_i$. (This also confirms that $\hat{\boldsymbol{u}}_i(\lambda)$ is independent of the choice of f in (6.17).) Thus we see that we can simply apply the filter $F(\lambda)$ component-wise for systems of conservation laws without destroying realizability.

6.3 Regularization

When faced with the potential infeasibility of the filtered moments, instead of modifying the filter to preserve realizability, we can instead modify the optimization problem to guarantee feasibility even for nonrealizable moment vectors.

A modification which achieves this is the regularization proposed in [25, 4], in which the equality constraints are exchanged with a penalty term in the objective function. Specifically, the primal optimization problem (1.120) defining the ansatz is replaced by

$$\mathcal{U}_\eta(\hat{\boldsymbol{u}}) = \arg\min_{\boldsymbol{u}} \langle s(\boldsymbol{u}) \rangle + \frac{1}{2\eta} \left\| \hat{\boldsymbol{u}} - \langle \boldsymbol{\varphi} \boldsymbol{u} \rangle \right\|^2, \tag{6.20}$$

where $\eta \in (0, \infty)$ is the regularization parameter and

$$\|\hat{\boldsymbol{w}}\|^2 = \sum_{i=0}^{N} \sum_{k=1}^{m} \hat{w}_{ik}^2. \tag{6.21}$$

This regularized problem is feasible for a larger set of moment vectors $\hat{\boldsymbol{u}}$, in particular when Θ is compact, it is feasible for all moment vectors.

Just as with the equality-constrained problem, the dual problem to (6.20) is finite dimensional:

$$\hat{\boldsymbol{v}}_\eta(\hat{\boldsymbol{u}}) := \underset{\hat{\boldsymbol{v}}\in\mathbb{R}^{m\times(N+1)}}{\arg\min} \left\langle s_*(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi})\right\rangle - \hat{\boldsymbol{v}}\cdot\hat{\boldsymbol{u}} + \frac{\eta}{2}\|\hat{\boldsymbol{v}}\|^2, \tag{6.22}$$

and the form of the primal minimizer is the same:

$$\mathcal{U}_\eta(\hat{\boldsymbol{u}}) = u\left(\hat{\boldsymbol{v}}_\eta(\hat{\boldsymbol{u}})^T\boldsymbol{\varphi}\right), \tag{6.23}$$

and the flux for the resulting system is

$$\boldsymbol{F}(\hat{\boldsymbol{u}}) = \left\langle \boldsymbol{f}(\mathcal{U}_\eta(\hat{\boldsymbol{u}}))\boldsymbol{\varphi}^T\right\rangle^T. \tag{6.24}$$

This gives a hyperbolic system of conservation laws, and, when $\hat{\boldsymbol{v}}_\eta$ is defined for all $\hat{\boldsymbol{u}}$, we can implement an IPM version of (5.19) directly (with a numerical flux $\boldsymbol{F}^*$ based on (6.24)). One remaining question is how to choose the regularization parameter η. A discussion of this issue can be found in [4], where the authors choose η according to the Morozov discrepancy principle, when interpreting the numerical errors as noise. Hence, η is chosen such that the effect of regularization does not dominate the numerical error.

6.4 Implementation

In this section we discuss the implementation of our numerical scheme for solving the filtered system. As previously discussed for filtered SG, we apply the filter prior to every time step, see equations (5.19). However now the discretization is done for the IPM system, i.e. the numerical fluxes have to be consistent with $\boldsymbol{F}(\hat{\boldsymbol{u}}) = \langle \boldsymbol{f}(\mathcal{U}(\hat{\boldsymbol{u}}))\boldsymbol{\varphi}^T\rangle^T$ or, if we use the entropy variables as inputs $\boldsymbol{G}(\hat{\boldsymbol{v}}) = \langle \boldsymbol{f}(u(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi}))\boldsymbol{\varphi}^T\rangle^T$. In the following we will use a numerical flux $\boldsymbol{G}^* : \mathbb{R}^{M\times m} \to \mathbb{R}^{M\times m}$ which acts on the entropy variables, i.e. $\boldsymbol{G}^*(\hat{\boldsymbol{v}},\hat{\boldsymbol{v}}) = \boldsymbol{G}(\hat{\boldsymbol{v}})$. The flux can again be constructed as a kinetic scheme according to Section 2.3.3.

6.4.1 Solving the dual problem

The IPM method relies on solving the constrained optimization problem (1.120), or in the regularized case the unconstrained problem (6.20). Both of these problems are infinite-dimensional, so we solve their dual problems (1.122b) and (6.22), which are finite-dimensional and unconstrained. In both cases we used Newton's method stabilized with the standard backtracking linesearch. Our stopping criterion is the norm of the dual, i.e.,

$$\left\| \left\langle \boldsymbol{\varphi} s_*'(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi})\right\rangle - \hat{\boldsymbol{u}}\right\| < \tau \qquad \text{or} \qquad \left\| \left\langle \boldsymbol{\varphi} s_*'(\hat{\boldsymbol{v}}^T\boldsymbol{\varphi})\right\rangle + \eta\hat{\boldsymbol{v}} - \hat{\boldsymbol{u}}\right\| < \tau \tag{6.25}$$

for the original and regularized cases respectively, where $\tau \in (0,\infty)$ is the user-specified tolerance parameter. Recall that we have $s_*'(v) = (s')^{-1}(v)$. The integrals needed to compute the dual objective function and its derivatives are computed with quadrature.

6.4.2 Spatial and temporal dicsretization

Let us now write down the implementation of the regularized IPM method including filters. The implementation is primarily based on the IPM Algorithm 3 as well as the filtered SG Algorithm 8.

The basic strategy of Algorithm 8 is, in every time step, first to apply a filter to the moment vector in each spatial cell, and then perform an update in time starting from these filtered moment vectors. Some details of the implementation vary depending on which method we choose, so we present the methods separately in Algorithms 9 and 10.

In Algorithm 9, we note that after the dual variables are computed numerically for the filtered moment vector (using the Fokker–Planck filter), we compute in Line 7 the moment vector associated with these approximate dual variables. This is done to ensure the realizability of the moment vector at the next time step, $\hat{\boldsymbol{u}}_j^{n+1}$, see Chapter 2. In Algorithm 10, this step is unnecessary.

Algorithm 9 Realizable Filtered IPM Method

1: $\hat{\boldsymbol{u}}_j^0 \leftarrow$ setup Initial Conditions for all cells j
2: choose filter strength $\lambda \in (0, \infty)$
3: **for** $n = 0$ to N_t **do**
4: **for** $j = 1$ to N_x **do**
5: $\overline{\boldsymbol{u}}_j^n \leftarrow F(\lambda)\hat{\boldsymbol{u}}_j^n$
6: $\widetilde{\boldsymbol{v}}_j^n \leftarrow \hat{\boldsymbol{v}}(\overline{\boldsymbol{u}}_j^n)$ using Newton's method with gradient tolerance τ
7: $\widetilde{\boldsymbol{u}}_j^n \leftarrow \left\langle \boldsymbol{\varphi} s_*'((\widetilde{\boldsymbol{v}}_j^n)^T \boldsymbol{\varphi}) \right\rangle$
8: **for** $j = 1$ to N_x **do**
9: $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \widetilde{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}(\boldsymbol{G}^*(\widetilde{\boldsymbol{v}}_j^n, \widetilde{\boldsymbol{v}}_{j+1}^n) - \boldsymbol{G}^*(\widetilde{\boldsymbol{v}}_{j-1}^n, \widetilde{\boldsymbol{v}}_j^n))$

Algorithm 10 Regularized Filtered IPM Method

1: $\hat{\boldsymbol{u}}_j^0 \leftarrow$ setup Initial Conditions for all cells j
2: choose filter strength $\lambda \in (0, \infty)$ and regularization parameter $\eta \in (0, \infty)$
3: **for** $n = 0$ to N_t **do**
4: **for** $j = 1$ to N_x **do**
5: $\overline{\boldsymbol{u}}_j^n \leftarrow \mathcal{L}(\lambda)\hat{\boldsymbol{u}}_j^n$
6: $\widetilde{\boldsymbol{v}}_{\eta,j}^n \leftarrow \hat{\boldsymbol{v}}_\eta(\overline{\boldsymbol{u}}_j^n)$ using Newton's method with gradient tolerance τ
7: **for** $j = 1$ to N_x **do**
8: $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \overline{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x}(\boldsymbol{G}^*(\widetilde{\boldsymbol{v}}_{\eta,j}^n, \widetilde{\boldsymbol{v}}_{\eta,j+1}^n) - \boldsymbol{G}^*(\widetilde{\boldsymbol{v}}_{\eta,j-1}^n, \widetilde{\boldsymbol{v}}_{\eta,j}^n))$

6.5 Results

6.5.1 Effects of the regularization

Before moving to filtering, we study the effects of the regularization for different regularization strengths η. For this we investigate the following test case: We wish to solve the uncertain Sod shock tube problem as described in Section 1.4.1. For this, we again

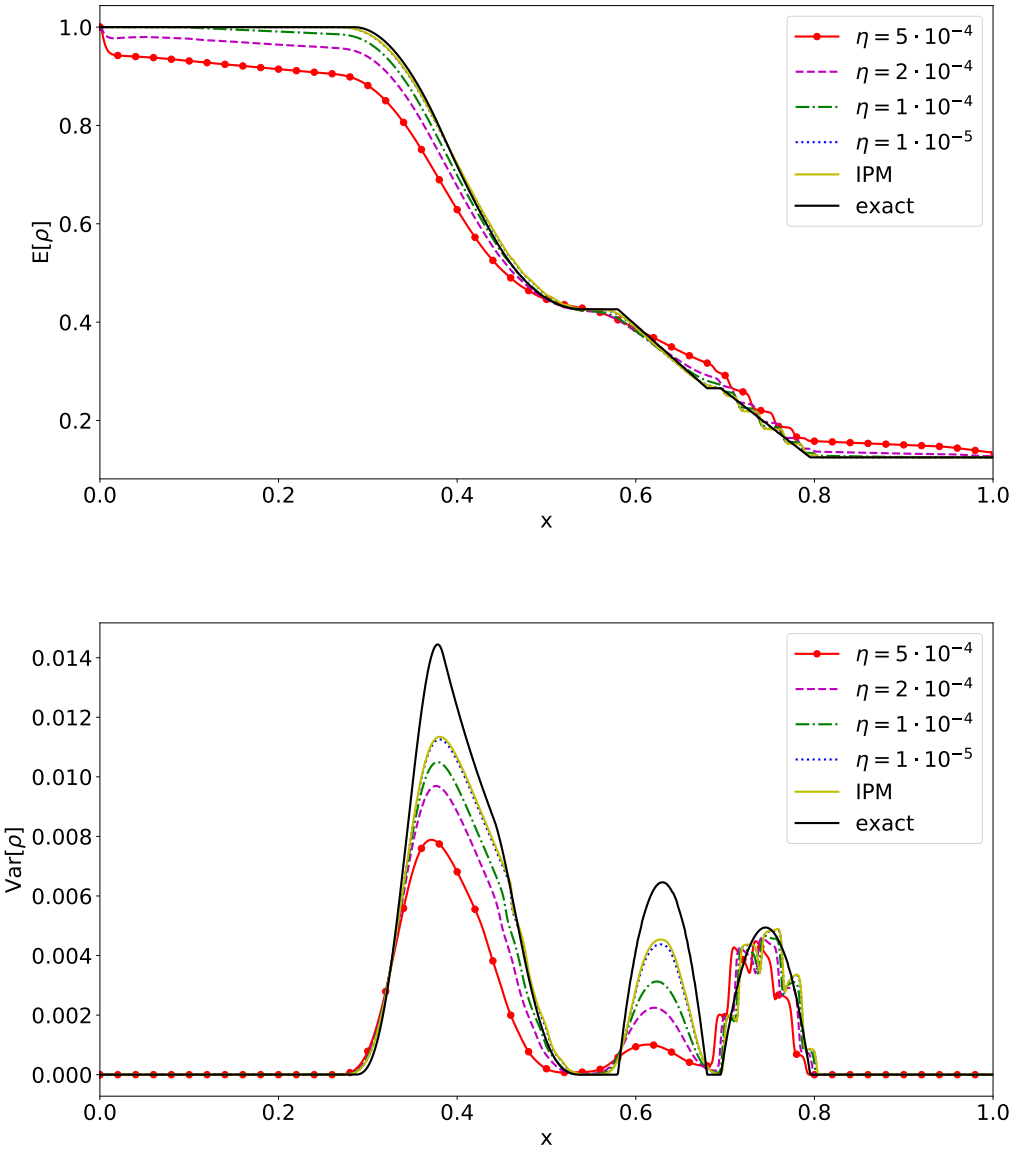


FIGURE 6.3: Expectation value and variance for different regularization strengths.

solve the one-dimensional Euler equations (1.82) with the initial conditions

$$\rho_{\text{IC}} = \begin{cases} \rho_L & \text{if } x < x_{\text{interface}}(\xi) \\ \rho_R & \text{else} \end{cases}$$
$$(\rho u)_{\text{IC}} = 0$$
$$(\rho E)_{\text{IC}} = \begin{cases} \rho_L E_L & \text{if } x < x_{\text{interface}}(\xi) \\ \rho_R E_R & \text{else} \end{cases}$$

The random interface position is given by $x_{\text{interface}}(\xi) = x_0 + \sigma\xi$ with a uniformly distributed random variable $\xi \sim U([-1,1])$. We use Dirichlet boundary conditions at the left and right boundary. The remaining parameter values are

$[a,b] = [0,1]$	range of spatial domain
$N_x = 1000$	number of spatial cells
$t_{\text{end}} = 0.14$	end time
$x_0 = 0.5, \sigma = 0.05$	interface position parameters
$\rho_L, p_L = 1.0, \rho_R = 0.125, p_R = 0.1$	initial states
$N = 5$	polynomial degree
$N_q = 20$	number of quadrature points
$\tau = 10^{-7}$	gradient tolerance for IPM

Again, the IPM method uses the entropy (1.117). One observes that this problem cannot be solved with SG or fSG, since negative densities ρ will show up already in the first time iteration. We now run this test case using IPM with different values for the regularization strength η. It is important to note that the choice of η does not only influence the solution, but also the runtime of the method: For $\eta = 5 \cdot 10^{-4}$, the computation takes 19.0 seconds, for $\eta = 2 \cdot 10^{-4}$ we obtain 21.1 seconds, for $\eta = 10^{-4}$ we have 21.3 seconds and for $\eta = 10^{-5}$ one obtains 22.4 seconds. This observation can be explained by the fact that the dual problem (6.22) is easier to solve than the original IPM optimization problem (1.122b). Taking a look at the resulting expectation value and variance of the density ρ in Figure 6.3, one sees that a big regularization parameter will heavily affect the solution. As the regularization strength decreases, the solution will approach the IPM solution and with $\eta = 10^{-5}$ the regularized solution shows good agreement with IPM. When the regularization parameter is not sufficiently small, we observe significant effects on higher order moments: When looking at the variance of the rarefaction wave and the contact discontinuity, one notices strong dampening. Note however that the variance in these regimes is very sensitive to the method used, see for example [70, 117].

The regularized solution with a small regularization parameter as well as the original IPM method show good approximation behavior in most parts of the solution. However, especially at the shock position, all methods will yield a non-physical, step-like approximation of the expectation value and oscillatory results for the corresponding variance. This work focuses on obtaining better approximations at the shock position through filtering techniques. Note that the filter will further dampen the variance approximations at the rarefaction wave and the shock discontinuity. One idea to mitigate these effects that we leave for future studies is choosing an adaptive filter strength as done in [70].

6.5.2 Filtering for Euler 1D

In the following we choose a constant regularization parameter $\eta = 10^{-5}$ and apply filtering to the above problem. We choose a filter strength of $\lambda = 5 \cdot 10^{-6}$ for the L^2

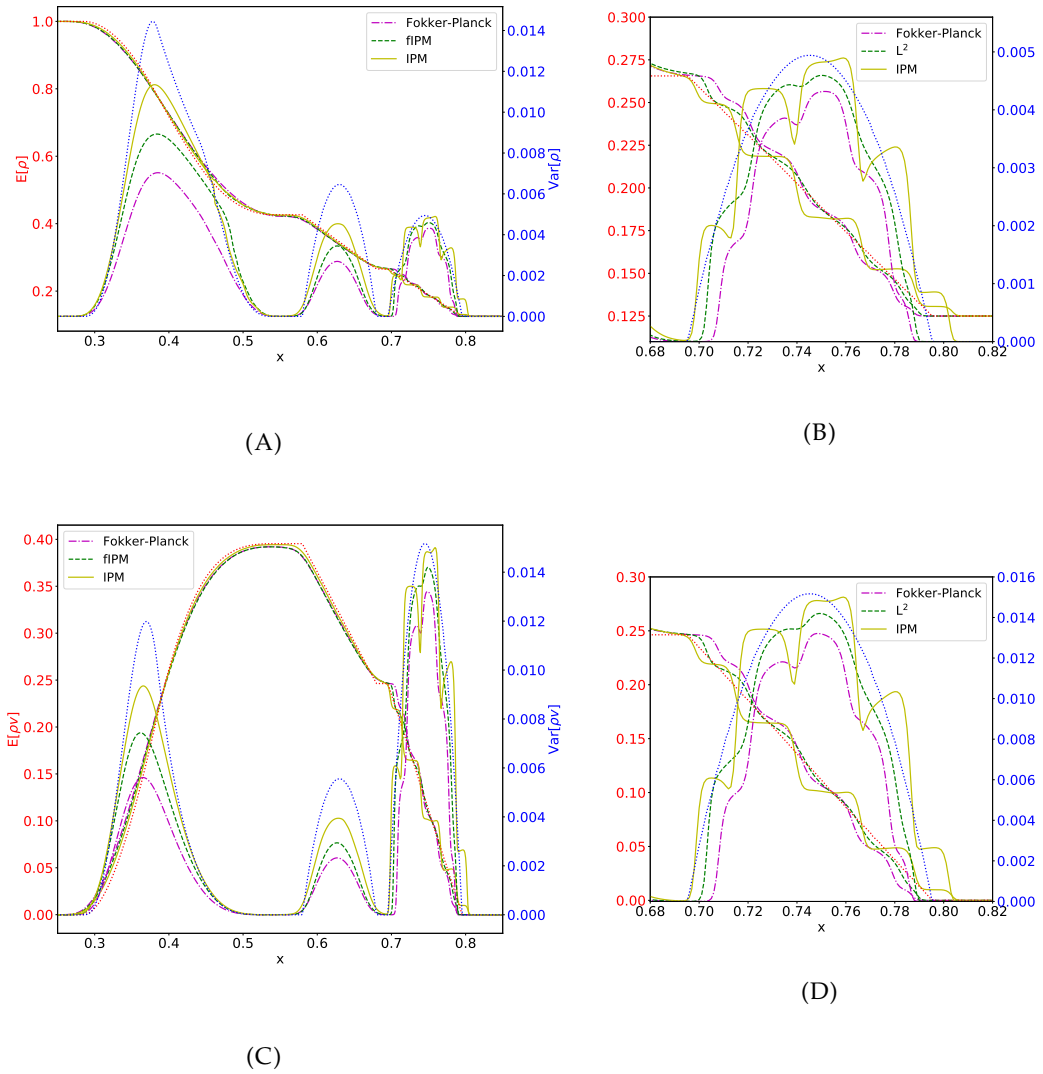


FIGURE 6.4: Expectation value and variance computed with IPM and filtered IPM ($\lambda = 5 \cdot 10^{-6}$ for L^2 filtering and $\lambda = 5 \cdot 10^{-5}$ for Fokker–Planck filtering). The exact expectation value is depicted in red, the exact variance in blue. (A) Density for $x \in [0.25, 0.85]$. (B) Zoomed view on shock for density and $x \in [0.68, 0.82]$. (C) Momentum for $x \in [0.25, 0.85]$. (D) Zoomed view on shock for momentum and $x \in [0.68, 0.82]$.

filter as well as $\lambda = 5 \cdot 10^{-5}$ for the Fokker–Planck filter (which have been determined by a parameter study) and depict the resulting expectation values and variances in Figure 6.4. Note, that the exact expectation value is shown in red and the exact variance is shown in blue. Both functions have been computed from the analytic solution of the Sod shock tube test case. We first examine the shock position: The expectation value and variance of the density, found in Figure 6.4A, again show heavy oscillations in the variance and a step-like profile at the shock if no filtering is applied. Using the L^2 filter leads to a smooth linear connection between the left and right shock state for the expectation value, which shows good agreement with the exact solution. The variance is slightly dampened, however its smooth profile nicely captures the main characteristics of the exact variance. This behavior can be examined in detail when zooming onto the shock position as done in Figure 6.4B.

As already observed for the regularization, the variance of both the rarefaction wave

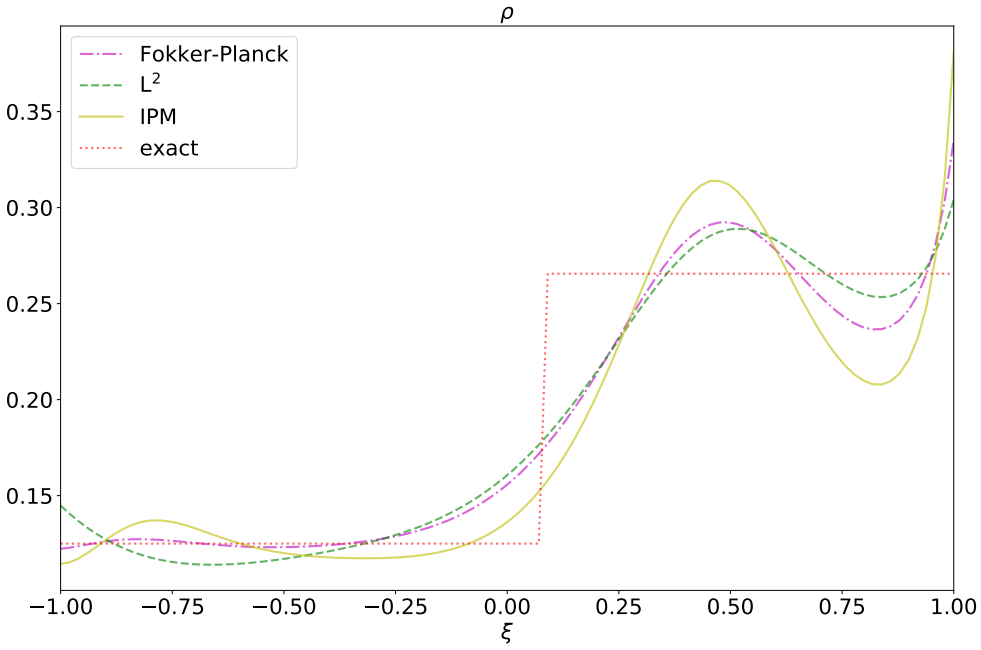


FIGURE 6.5: Density at $x = 0.75$ with and without filtering.

and the contact discontinuity are dampened heavily. Different possibilities to fix this behavior like using an adaptive filter strength could be chosen. In this work, however, we focus on mitigating oscillations at the shock and leave other open questions to future work. Note, that the L^2 filter shows less damping behavior for the variance, which is why we will use only this filter in the two-dimensional setting presented in Section 6.5.3.

The direct effects of filtering can be observed when fixing the spatial position at $x = 0.75$ and plotting the density of the gas as a function of ξ in Figure 6.5. Here, we observe that the resulting densities are all positive and show little oscillations around the lower shock state at the left. Especially the Fokker–Planck filter nicely captures the exact solution behavior in this region. At the right shock state, the different solutions oscillate, however both solutions computed with filtered IPM show less oscillations. In this region, the L^2 filters shows the best agreement with the exact solution.

6.5.3 Filtering for Euler 2D

In the following, we will again move to the two-dimensional Euler equations (4.26). As in 5.4.5, we will be interested in quantifying the uncertainty of a shock inside a two-dimensional nozzle. All settings remain the same, except for the pressure inside the chamber, which is now chosen to be $p_L = 0.125$. In this case the SG and fSG methods fail, since negative densities and energies occur. A reference solution, depicted in Figures 6.6 and 6.6 on the bottom, has been computed using Collocation with 50 Gauss-Legendre quadrature points. The IPM solution, which is depicted in Figures 6.6 and 6.6 on the top shows the expected step-like, oscillatory solution for the variance and expectation values. To mitigate these artifacts, we make use of filters. Unlike the Lasso filter, presented in the previous chapter, the discussed L^2 and Fokker–Planck filters require choosing an adequate filter strength. These parameters have been determined with a parameter study, which we will discuss in greater detail in the following. We wish to

obtain a non-oscillatory solution approximation of the expectation value and variance. To measure oscillations, we investigate the error of the second spatial derivatives of the expecation value and variance. Hence, if the distance of a discrete numerical density solution $\rho_\Delta := (\rho_1, \cdots, \rho_{N_x})^T$ to a reference density $\rho_{\text{ex},\Delta} := (\rho_{\text{ex},1}, \cdots, \rho_{\text{ex},N_x})^T$ is given by $e_\Delta := \rho_{\text{ex},\Delta} - \rho_\Delta$, we are interested in determining the error of second derivatives, which is

$$\delta_{\text{E}} := \sqrt{\sum_{j=1}^{N_x} \Delta x_j \left(\left(\partial_{xx} \text{E}[e_j]\right)^2 + \left(\partial_{yy} \text{E}[e_j]\right)^2 \right)}$$

and

$$\delta_{\text{Var}} := \sqrt{\sum_{j=1}^{N_x} \Delta x_j \left(\left(\partial_{xx} \text{Var}[e_j]\right)^2 + \left(\partial_{yy} \text{Var}[e_j]\right)^2 \right)}.$$

Now, we vary the filter strength for the Fokker-Planck filter and the L^2 filter with a fixed regularization parameter $\eta = 10^{-5}$ and investigate the effect on δ_{E} as well as δ_{Var}. The results for the L^2 filter can be found in Figure 6.8 and for the Fokker–Planck filter in Figure 6.9. One observes that both filters have an optimal filter strength, where we achieve optimality for both, the expectation value and the variance. The optimal value for the L^2 filter is $\lambda = 3 \cdot 10^{-6}$ and for the Fokker–Planck filter, we have $\lambda = 2 \cdot 10^{-5}$, where the L^2 filter achieves a smaller value for δ_{E} and δ_{Var}. For both filters, the error approaches the IPM error when the filter strength is sufficiently turned down. When the filter strength is too big, the value of δ increases heavily, leading to a weaker result than IPM. For the corresponding optimal filter strength, the resulting expectation value and variance of the density can be found in Figure 6.6 and Figure 6.7. On can observe that by using the L^2 filter, we can mitigate the oscillations in the variance while obtaining a smoother approximation of the expected values which show good agreement with the reference solution. The Fokker–Planck filter shows a less convincing result, as it still oscillates while heavily dampening the variance at the contact discontinuity.

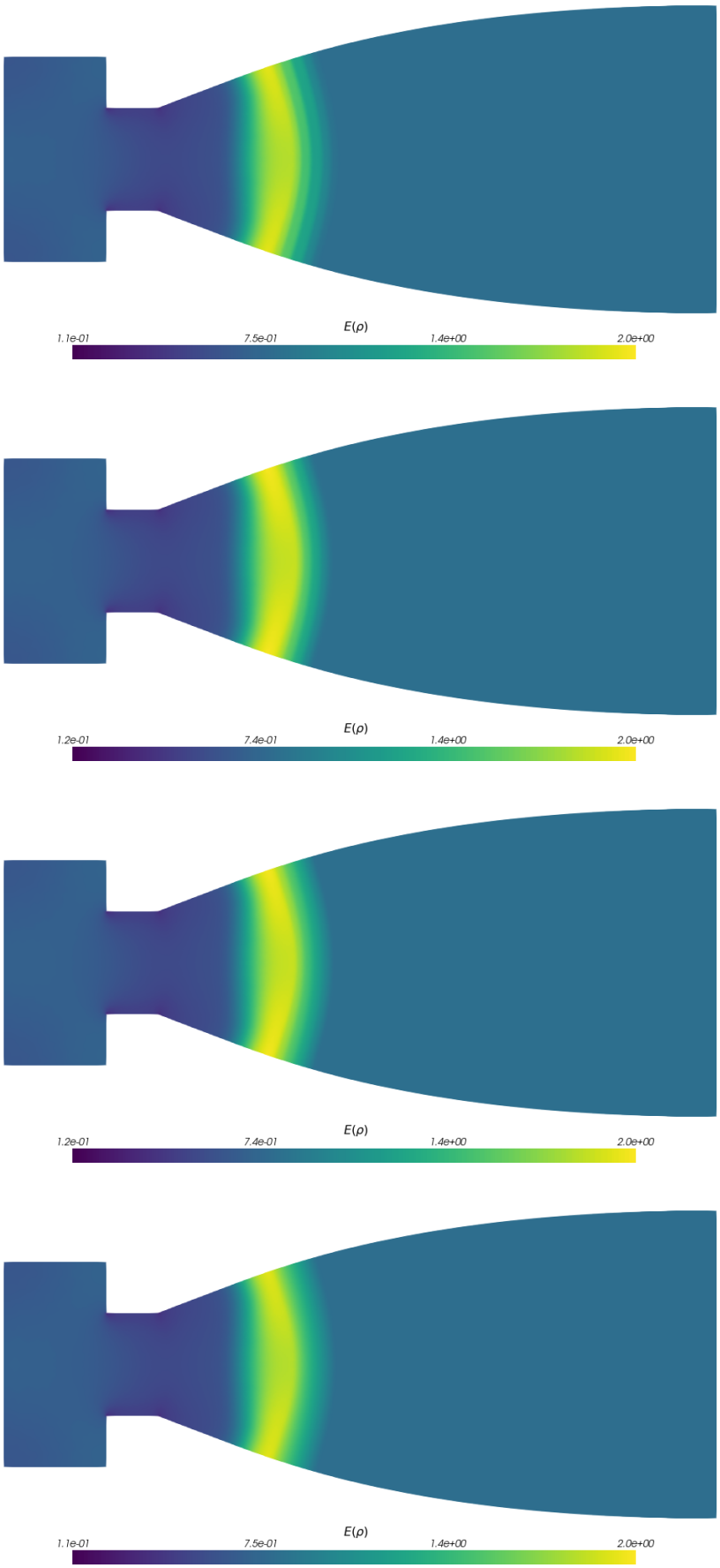


FIGURE 6.6: Expected value with different methods. From top to bottom: SG, Fokker–Planck, L^2, reference.

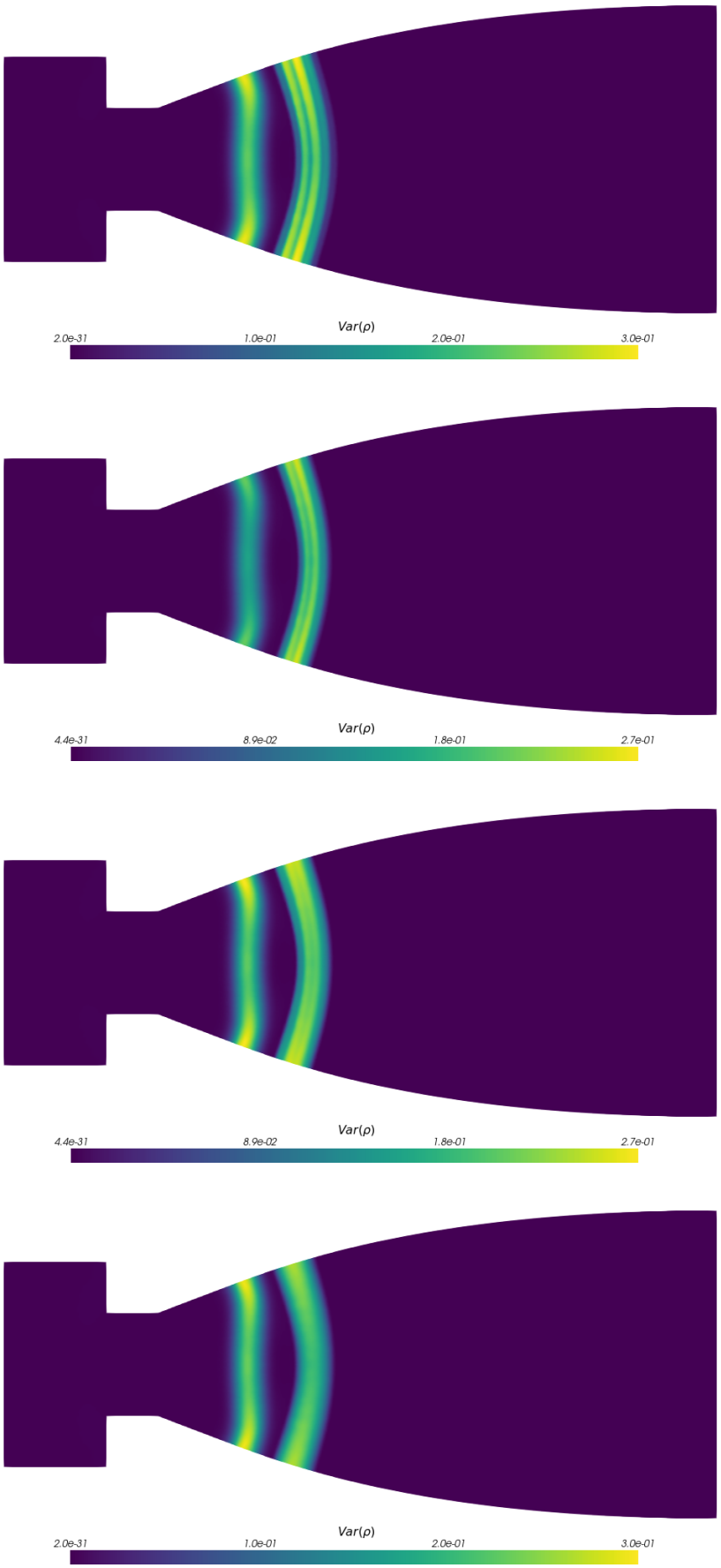


FIGURE 6.7: Variance with different methods. From top to bottom: SG, Fokker–Planck, L^2, reference.

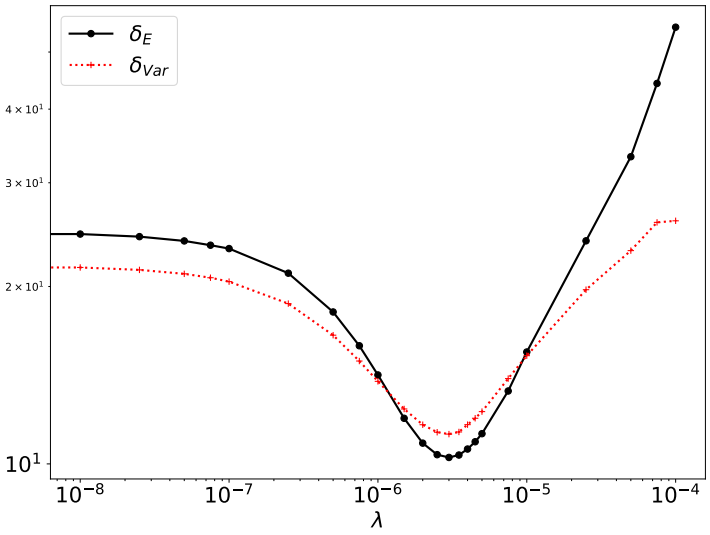


FIGURE 6.8: δ_{E} and δ_{Var} for different filter strengths when using the L^2 filter with regularization strength $\eta = 10^{-5}$.

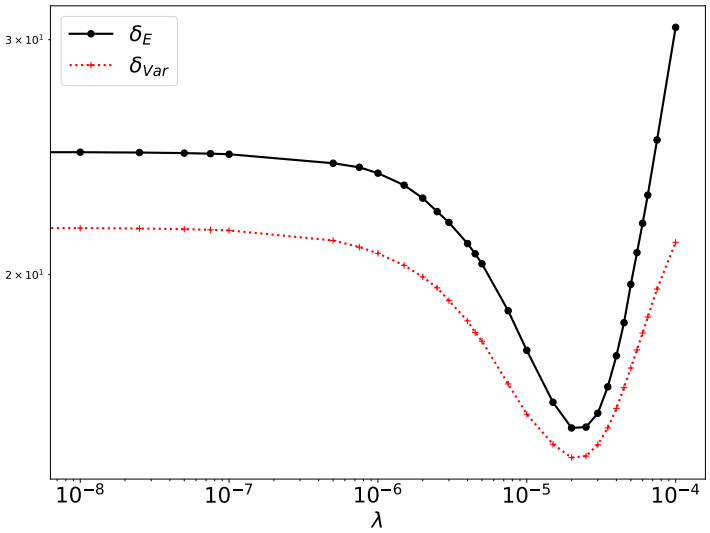


FIGURE 6.9: δ_{E} and δ_{Var} for different filter strengths when using the Fokker–Planck filter.

Chapter 7

Radiative transfer with uncertainties

After having discussed different techniques to discretize the uncertain domain, we will now move to radiative transfer equations, which in addition to the uncertain domain depend on the direction of particles. Radiative transfer is often prone to uncertain measurements or modeling assumptions. As discussed in Section 1.3, the interaction of particles with the background material consists of scattering and absorption and one assumes these processes to not change the properties of the background medium. Several problems however require such effects, since the material heats up when absorbing particles resulting in an emission of new particles from the material. Such an interplay between the background medium and particles is described by the *thermal radiative transfer equations*

$$\frac{1}{c}\frac{\partial I(t,z,\mu)}{\partial t} + \mu\frac{\partial I(t,z,\mu)}{\partial z} = \sigma_a(z)\left(B(T(t,z)) - I(t,z,\mu)\right) + \tilde{S}(t,z,\mu), \tag{7.1a}$$

$$\frac{1}{c}\frac{\partial e(t,z)}{\partial t} = \sigma_a(E(t,z) - aT(t,z)^4), \tag{7.1b}$$

where the internal energy e fulfills $e(T) = \int_0^T C_V(\bar{T})\,d\bar{T}$ and T is the material temperature. Note, that (7.1a) is the radiative transfer equation (1.64) without scattering when using a specific source term. Here, we change notation to coincide with the notation used by the nuclear engineering community. The specific intensity $I(t,z,\mu)$ depends on time $t \in \mathbb{R}_+$, spatial position $z \in \mathbb{R}$ and the projected angular components $\mu \in [-1,1]$. The emission of particles by the material is described by black body radiation $B(T) = \frac{ac}{4\pi}T^4$ with the black body constant $a = 4\sigma_{SB}/c$ and $\tilde{S}$ is an external source. The opacity σ_a, which plays the role of an absorption cross section, controls the interaction strength between particles and the background material. Note that these equations omit scattering effects and for now we assume deterministic equations, i.e. the phase space does not include random variables.

The radiative transfer equations have been studied extensively in the literature, see e.g. [104, 125, 94, 90, 92, 57]. Since several solution parameters are based on measurements or modeling assumptions, it is of great interest to determine the effects of uncertainties on these systems. The aim of this chapter is to apply the proposed intrusive uncertainty quantification concepts used in this work to thermal radiative transfer applications. Before discretizing the uncertain domain, we treat the angular domain by deriving a scaled P_1 system in Section 7.1. Then, in Section 7.2, we discretize the random domain with an intrusive polynomial moment method, which uses a stochastic-Galerkin ansatz for the angular moments as well as an exponential ansatz for the material energy. The method is then tested for the uncertain Su-Olson problem, radiative shocks and Marshak waves in Section 7.3.

7.1 Scaled P_1 equations for thermal radiative transfer

In the following, we summarize the derivation of the scaled P_1 equations, see e.g. [92]. The dependence on the phase space will in the following be omitted for better readability. Two physical quantities related to the thermal radiative transfer equations (7.1) are the material temperature T as well as the energy density, which is given by

$$E = \frac{2\pi}{c} \int_{-1}^{1} I(\mu) d\mu.$$

The first-order moment, or radiation flux, is given by

$$F = 2\pi \int_{-1}^{1} \mu I(\mu)\, d\mu.$$

Averaging the radiative transfer equation (7.1a) over μ and multiplying by $2\pi/c$ gives an equation for the energy density, which reads

$$\frac{1}{c}\partial_t E + \frac{1}{c}\frac{\partial F}{\partial z} = \sigma_a(aT^4 - E) + S,$$

where $S = \frac{\pi}{c}\tilde{S}$. An equation for the radiation flux F is obtained by multiplying (7.1a) by $2\pi\mu$ and averaging over μ. To obtain a closed P_1 system (see Section 1.3.1), one assumes linear dependency of the specific intensity I on μ. Then, the equation for the radiation flux reads

$$\frac{1}{c}\frac{\partial F}{\partial t} + \frac{c}{3}\frac{\partial E}{\partial z} = -\sigma_a F.$$

The full P_1 approximation to the thermal radiative transfer equations is hence given by

$$\frac{1}{c}\frac{\partial E}{\partial t} + \frac{1}{c}\frac{\partial F}{\partial z} = \sigma_a(aT^4 - E) + S, \tag{7.2a}$$

$$\frac{1}{c}\frac{\partial F}{\partial t} + \frac{c}{3}\frac{\partial E}{\partial z} = -\sigma_a F, \tag{7.2b}$$

$$\frac{1}{c}\frac{\partial e}{\partial t} = \sigma_a(E - aT^4). \tag{7.2c}$$

Now, to impose adequate solution values, we scale the system by

$$x = \sigma_a z, \quad \tau = \varepsilon c \sigma_a t,$$
$$\mathcal{E} = \frac{E}{aT_r^4}, \quad \mathcal{T} = \frac{T}{T_r}, \quad \mathcal{F} = \frac{F}{aT_r^4}, \quad Q = \frac{S}{\sigma_a a T_r^4}, \quad \tilde{e} = \frac{e}{aT_r^4},$$

where ε is a constant parameter used to scale time and we assume a constant absorption cross section σ_a. With $\partial_t \tau = \varepsilon c \sigma_a$ and $\partial_z x = \sigma_a$, we then have

$$\varepsilon\sigma_a a T_r^4 \frac{\partial \mathcal{E}}{\partial \tau} + \frac{\sigma_a a T_r^4}{c}\frac{\partial \mathcal{F}}{\partial x} = \sigma_a(aT_r^4\mathcal{T}^4 - aT_r^4\mathcal{E}) + \sigma_a a T_r^4 Q,$$
$$\varepsilon\sigma_a a T_r^4 \frac{\partial \mathcal{F}}{\partial \tau} + \frac{c\sigma_a a T_r^4}{3}\frac{\partial \mathcal{E}}{\partial x} = -\sigma_a a T_r^4 \mathcal{F},$$
$$\varepsilon\sigma_a T_r \frac{\partial \tilde{e}}{\partial \tau} = \sigma_a(aT_r^4\mathcal{E} - aT_r^4\mathcal{T}^4).$$

Dividing all three equations by $\sigma_a a T_r^4$ yields the scaled P_1 approximation to the thermal radiative transfer equations

$$\varepsilon \frac{\partial \mathcal{E}}{\partial \tau} + \frac{1}{c} \frac{\partial \mathcal{F}}{\partial x} = \mathcal{T}^4 - \mathcal{E} + Q, \tag{7.4a}$$

$$\varepsilon \frac{\partial \mathcal{F}}{\partial \tau} + \frac{c}{3} \frac{\partial \mathcal{E}}{\partial x} = -\mathcal{F}, \tag{7.4b}$$

$$\varepsilon \frac{\partial \tilde{e}}{\partial \tau} = \mathcal{E} - \mathcal{T}^4. \tag{7.4c}$$

To obtain a closed system of equations we need to define a constitutive law, which relates $\tilde{e}$ to $\mathcal{T}$ and vice-versa. Hence, we need to find an expression for $C_V(T)$, which then gives

$$\tilde{e} = \frac{e}{a T_r^4} = \frac{\int_0^{T_r \mathcal{T}} C_V(\bar{T})\, d\bar{T}}{a T_r^4}.$$

7.1.1 Su-Olson closure

First, let us make a choice for C_V which eliminates the dependency on the scaled internal energy and yields a linear system of equations [125]: We choose $C_V = \alpha T^3$, where α is a user-determined constant. Then, when using $\varepsilon = \frac{\alpha}{4a}$ and switching to the modified temperature $U = \mathcal{T}^4$, we get

$$\tilde{e} = \frac{\alpha}{4a} \mathcal{T}^4 = \frac{1}{\varepsilon} U.$$

Then, (7.4a) becomes

$$\varepsilon \frac{\partial \mathcal{E}}{\partial \tau} + \frac{1}{c} \frac{\partial \mathcal{F}}{\partial x} = U - \mathcal{E} + Q, \tag{7.5a}$$

$$\varepsilon \frac{\partial \mathcal{F}}{\partial \tau} + \frac{c}{3} \frac{\partial \mathcal{E}}{\partial x} = -\mathcal{F}, \tag{7.5b}$$

$$\frac{\partial U}{\partial \tau} = \mathcal{E} - U. \tag{7.5c}$$

This system is linear and can even be solved analytically [92].

7.1.2 Linear closure

In multiple applications of practical interest, C_V can be approximated by a constant parameter. Therefore, let us define the closure $C_V = const$, i.e. the heat capacity is constant with respect to temperature. Then, we have

$$\tilde{e} = \frac{C_V \mathcal{T}}{a T_r^3} \qquad \text{and} \qquad \mathcal{T} = \frac{a T_r^3 \tilde{e}}{C_V}.$$

In this case, the P_1 approximation to the thermal radiative equations (7.4a) becomes

$$\varepsilon \frac{\partial \mathcal{E}}{\partial \tau} + \frac{1}{c}\frac{\partial \mathcal{F}}{\partial x} = \left(\frac{aT_r^3}{C_V}\right)^4 \tilde{e}^4 - \mathcal{E} + Q, \tag{7.6a}$$

$$\varepsilon \frac{\partial \mathcal{F}}{\partial \tau} + \frac{c}{3}\frac{\partial \mathcal{E}}{\partial x} = -\mathcal{F}, \tag{7.6b}$$

$$\varepsilon \frac{\partial \tilde{e}}{\partial \tau} = \mathcal{E} - \left(\frac{aT_r^3}{C_V}\right)^4 \tilde{e}^4. \tag{7.6c}$$

7.2 Intrusive formulation

7.2.1 Stochastic-Galerkin formulation

In the following, we assume that the initial condition as well as model parameters are uncertain. For ease of presentation, we assume an uncertain source $Q = Q(\xi)$, where ξ is a scalar random variable. We start by deriving a stochastic-Galerkin system for the linear system (7.5), i.e. when using the Su-Olson closure. Given $N+1$ gPC basis functions φ_i with $i = 0, \cdots, N$, one can represent the solution by

$$\mathcal{E}(t,x,\xi) \approx \mathcal{E}_N(t,x,\xi) := \sum_{i=0}^{N} \hat{\mathcal{E}}_i(t,x)\varphi_i(\xi)\,, \tag{7.7a}$$

$$\mathcal{F}(t,x,\xi) \approx \mathcal{F}_N(t,x,\xi) := \sum_{i=0}^{N} \hat{\mathcal{F}}_i(t,x)\varphi_i(\xi)\,, \tag{7.7b}$$

$$U(t,x,\xi) \approx U_N(t,x,\xi) := \sum_{i=0}^{N} \hat{U}_i(t,x)\varphi_i(\xi)\,. \tag{7.7c}$$

To derive time evolution equations for the gPC coefficient vectors $\hat{\boldsymbol{\mathcal{E}}}, \hat{\boldsymbol{\mathcal{F}}}, \hat{\boldsymbol{U}} \in \mathbb{R}^{N+1}$, which collect the gPC coefficients of each state respectively, we plug the SG ansätze (7.7) into the scaled equations (7.5) and project the resulting residual to zero. This gives the SG moment system

$$\begin{aligned}
\varepsilon \frac{\partial \hat{\boldsymbol{\mathcal{E}}}}{\partial \tau} + \frac{1}{c}\frac{\partial \hat{\boldsymbol{\mathcal{F}}}}{x} &= -(\hat{\boldsymbol{\mathcal{E}}} - \hat{\boldsymbol{U}}) + \langle Q\boldsymbol{\varphi}\rangle, \\
\varepsilon \frac{\partial \hat{\boldsymbol{\mathcal{F}}}}{\partial \tau} + \frac{c}{3}\frac{\partial \hat{\boldsymbol{\mathcal{E}}}}{x} &= -\hat{\boldsymbol{\mathcal{F}}}, \\
\frac{\partial \hat{\boldsymbol{U}}}{\partial \tau} &= (\hat{\boldsymbol{\mathcal{E}}} - \hat{\boldsymbol{U}}).
\end{aligned}$$

Note, that the structure of the moment equations are similar to the original system (7.5). Basically, every gPC coefficient fulfills the original equation, except for a moment-dependent value of the source term, which is $\langle Q\boldsymbol{\varphi}\rangle =: \hat{\boldsymbol{Q}}$.

In the case of the nonlinear equations (7.6), we use a polynomial representation for

$\tilde{e}$ instead of U and denote the expansion coefficients by $\hat{e}$. In this case, the stochastic-Galerkin system is given by

$$\varepsilon \frac{\partial \hat{\boldsymbol{\mathcal{E}}}}{\partial \tau} + \frac{1}{c}\frac{\partial \hat{\boldsymbol{\mathcal{F}}}}{x} = \left(\frac{aT_r^3}{C_V}\right)^4 \boldsymbol{C}(\hat{e}) - \hat{\boldsymbol{\mathcal{E}}} + \hat{\boldsymbol{Q}}, \tag{7.8a}$$

$$\varepsilon \frac{\partial \hat{\boldsymbol{\mathcal{F}}}}{\partial \tau} + \frac{c}{3}\frac{\partial \hat{\boldsymbol{\mathcal{E}}}}{x} = -\hat{\boldsymbol{\mathcal{F}}}, \tag{7.8b}$$

$$\varepsilon \frac{\partial \hat{e}}{\partial \tau} = \hat{\boldsymbol{\mathcal{E}}} - \left(\frac{aT_r^3}{C_V}\right)^4 \boldsymbol{C}(\hat{e}). \tag{7.8c}$$

Here, we use the function $\boldsymbol{C} : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ with

$$\boldsymbol{C}(\hat{e}) = \sum_{i,j,k,l=0}^{N} \hat{e}_i \hat{e}_j \hat{e}_q \hat{e}_l \langle \varphi_i \varphi_j \varphi_q \varphi_l \boldsymbol{\varphi} \rangle.$$

Note that even though the integral term can be computed before running the simulation, we perform an (exact) collocation step to approximate this integral, i.e. we compute

$$\tilde{e}_k := \sum_{i=0}^{N} \hat{e}_i \varphi_i(\xi_k)$$

and then, with quadrature points ξ_k and weights w_k, compute $\boldsymbol{C}$ by

$$\boldsymbol{C}(\hat{e}) = \sum_{k=1}^{N_q} w_k \tilde{e}_k^4 \boldsymbol{\varphi}(\xi_k)$$

which requires $O(N_q \cdot (N+1))$ instead of $O((N+1)^5)$ operations.

7.2.2 IPM formulation

Unfortunately, the polynomial solution ansatz of stochastic-Galerkin can lead to non-physical solution values of the scaled energy density $\mathcal{E}$ as well as the scaled internal energy $\tilde{e}$ (or modified temperature U). The following discussion assumes that we are using a linear closure, i.e. the underlying equations are (7.6). However, the same techniques proposed in the following can also be applied for the Su-Olson closure, in which case one solves (7.5). Since the P_1 approximation to the radiative transfer equations can already violate positivity of the energy density, we keep a polynomial solution ansatz and only choose a modified ansatz for the internal energy, which is now chosen to be

$$\tilde{e}(t,x,\xi) \approx e_N(t,x,\xi) := \exp\left(\hat{\boldsymbol{v}}(t,x)^T \boldsymbol{\varphi}(\xi)\right),$$

where $\hat{\boldsymbol{v}} \in \mathbb{R}^{N+1}$ are the dual variables. Here, in the spirit of the IPM method, we perform a gPC expansion of the entropy variables, when choosing the entropy

$$s(\mathcal{E}, \mathcal{F}, \tilde{e}) = \frac{1}{2}\mathcal{E}^2 + \frac{1}{2}\mathcal{F}^2 + \tilde{e}\ln(\tilde{e}) - \tilde{e}.$$

With the chosen solution ansatz, we obtain the IPM system

$$\varepsilon \frac{\partial \hat{\boldsymbol{\mathcal{E}}}}{\partial \tau} + \frac{1}{c}\frac{\partial \hat{\boldsymbol{\mathcal{F}}}}{x} = \left(\frac{aT_r^3}{C_V}\right)^4 \left\langle \exp\left(4\hat{v}^T\boldsymbol{\varphi}\right)\boldsymbol{\varphi}\right\rangle - \hat{\boldsymbol{\mathcal{E}}} + \hat{\boldsymbol{Q}}, \tag{7.9a}$$

$$\varepsilon \frac{\partial \hat{\boldsymbol{\mathcal{F}}}}{\partial \tau} + \frac{c}{3}\frac{\partial \hat{\boldsymbol{\mathcal{E}}}}{x} = -\hat{\boldsymbol{\mathcal{F}}}, \tag{7.9b}$$

$$\varepsilon \frac{\partial \hat{e}}{\partial \tau} = \hat{\boldsymbol{\mathcal{E}}} - \left(\frac{aT_r^3}{C_V}\right)^4 \left\langle \exp\left(4\hat{v}^T\boldsymbol{\varphi}\right)\boldsymbol{\varphi}\right\rangle. \tag{7.9c}$$

Note that the arising integral on the right hand side needs to be approximated by a quadrature rule. Since the IPM system requires the dual variables $\hat{v}$ to evaluate the source term, we need to use the mapping from the moments of the internal energy to their dual variables by the IPM optimization problem (1.122b).

Note that the IPM method appears to be a suited technique to solve thermal radiative transfer problems, since the IPM optimization problem only needs to be solved for the material energy. Let us remark that increasing the number of spherical moments or treating the source with an implicit method will result in additional numerical costs which dominate the costs of the IPM optimization problem.

7.3 Results

7.3.1 Su-Olson

In the following, we study the Su-Olson testcase with an uncertain source term. The Su-Olson problem considers a one-dimensional spatial domain $D = (-\infty, \infty)$ which includes a source in its center with a range $\left[-\frac{1}{2} - \sigma\xi, \frac{1}{2} + \sigma\xi\right]$, where $\xi \in [-1, 1]$ is a uniformly distributed random variable and $\sigma = 0.2$. The source emits particles, which then travel to the remainder of the spatial domain while heating up the material. The scaled source now reads

$$Q(x, \xi) = \begin{cases} 1 & \text{if } x \in [-\frac{1}{2} - \sigma\xi, \frac{1}{2} + \sigma\xi] \\ 0 & \text{else} \end{cases},$$

i.e. we have a source with an uncertain length in the center of the domain. The chosen closure is the Su-Olson closure, described in Section 7.1.1, i.e. the underlying equations are given by (7.5). Initially, energy and temperature are chosen to be zero. When studying the dynamics of the deterministic Su-Olson problem (i.e. $\sigma = 0$), one will observe particles entering the problem through the source as time increases, i.e. the particles start to heat up the material in the center of the spatial domain while traveling to the outer boundaries. A thorough investigation of the P_1 discretization of the Su-Olson problem can be found in [92]. In this test case, we now assume an uncertain length of the source, meaning that for different realizations, a varying number of particles will be emitted. As described in Section 7.2.2, we use an exponential ansatz to describe the modified temperature U. The numerical calculation is carried out with the general intrusive framework presented in Section 4.1, i.e. based on [71]. We treat the infinite spatial domain by choosing a sufficiently large computational domain $D = [-10, 10]$ to ensure that particles do not reach the boundaries. The chosen parameter values are

$N_x = 5000$	number spatial cells
$M = 5$	number of moments
$N_q = 10$	number of quadrature points
$\sigma_a = 1$	opacity
$T_r = 1$	reference temperature

Let us now investigate expectation value and variance of the scaled energy density and modified material temperature at scaled times $\tau \in \{1, 3.16, 10\}$, depicted in Figure 7.1. Comparing the results to the deterministic energy density and temperature [92], one can see that the respective expectation value shows a similar structure. The variance increases over time, while showing its maximum value at the source boundary for $\xi = 0$, i.e. at spatial position $x = 0.5$.

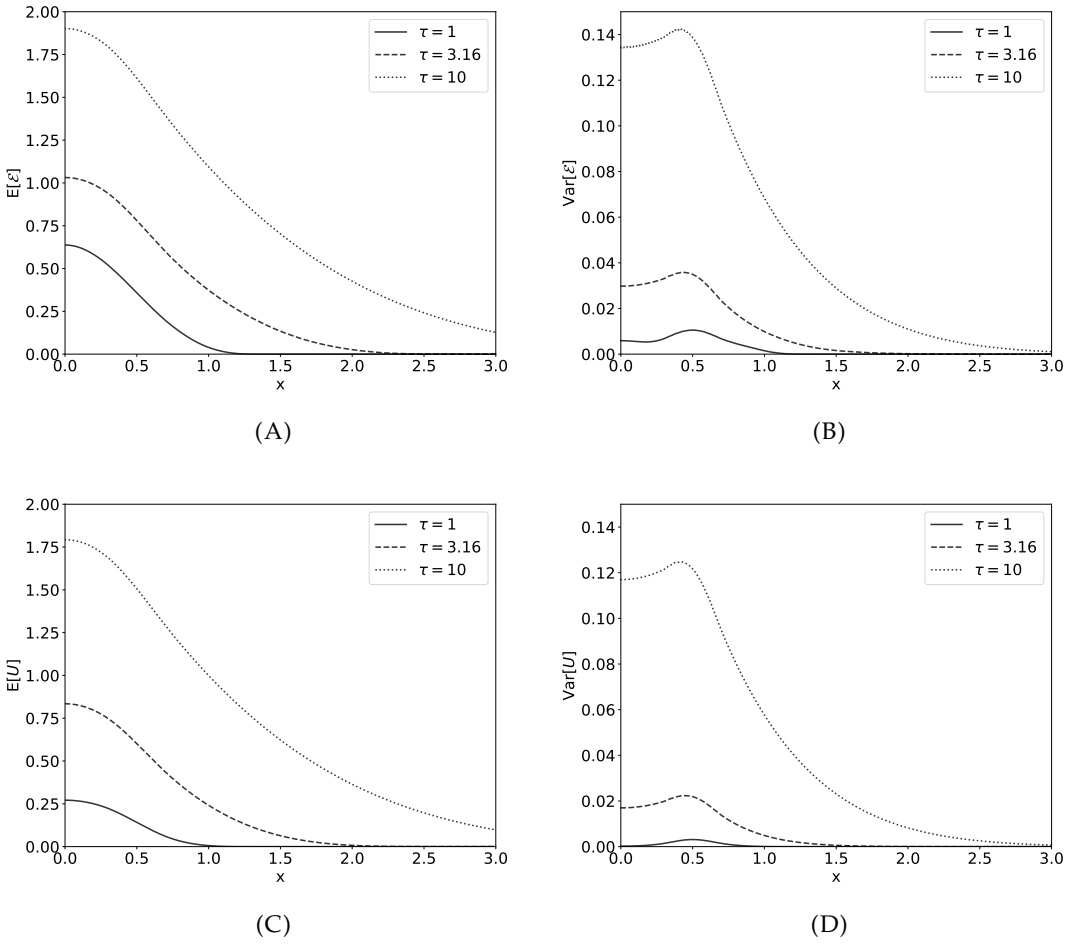


FIGURE 7.1: Zoomed view of expectation value (A) and variance (B) of the scaled energy density and expectation value (C) and variance (D) of the modified material temperature.

7.3.2 Steady radiative shock

The following testcase investigates a steady radiative shock for a constant material temperature as presented in [90]. We choose a linear closure according to Section 7.1.2, i.e. we need to pick the parameter C_V which effects how heavily the temperature is affected by the radiation energy. A big value for C_V will yield a small effect of radiation on the material temperature. In our case, we choose $C_V = 0.718 \cdot 10^7$ in which case the material

temperature will remain constant. The temperature is given by

$$T(x) = \begin{cases} \alpha & \text{if } x < 0 \\ 1 & \text{if } x \in [0, \tau_0 + \xi] \\ \beta & \text{else} \end{cases},$$

where ξ is again uniformly distributed in the interval $[-0.1, 0.1]$. Initially, we choose $\mathcal{E} = T^4/T_r^4$ as well as $F = 0$ and let the system (7.6) evolve in time until the solution remains constant. Since we are looking at a steady state problem, we use the osIPM method presented in Section 4.2. Furthermore, we make use of adaptivity as discussed in Section 4.3. The results are depicted in Figure 7.2, where the red line indicates the refinement level of the corresponding spatial cell. Here, we use truncation orders of $N = 2$ for a refinement level of 1 up to $N = 10$ for a refinement level of 8. Due to the high value of C_V, expectation value and variance of the temperature remain constant. The expectation value of the energy density (Figure 7.2A) shows a smooth profile, except for the position $x = 0$, where one can find a small jump. This discontinuity is an artifact from the spatial discretization and the jump height decreases when refining the spatial mesh. We observe a high refinement level for the discretization of the uncertainty in the center of the spatial domain. The corresponding variance is depicted in Figure 7.2B. Note that its maximum is not positioned at the center, but is slightly shifted to the right. The expectation value and variance of the radiation flux are depicted in Figures 7.2C and 7.2D. The expectation value shows a shock at $x = 0$, whereas the variance shows two maxima traveling to the left and right of the spatial domain.

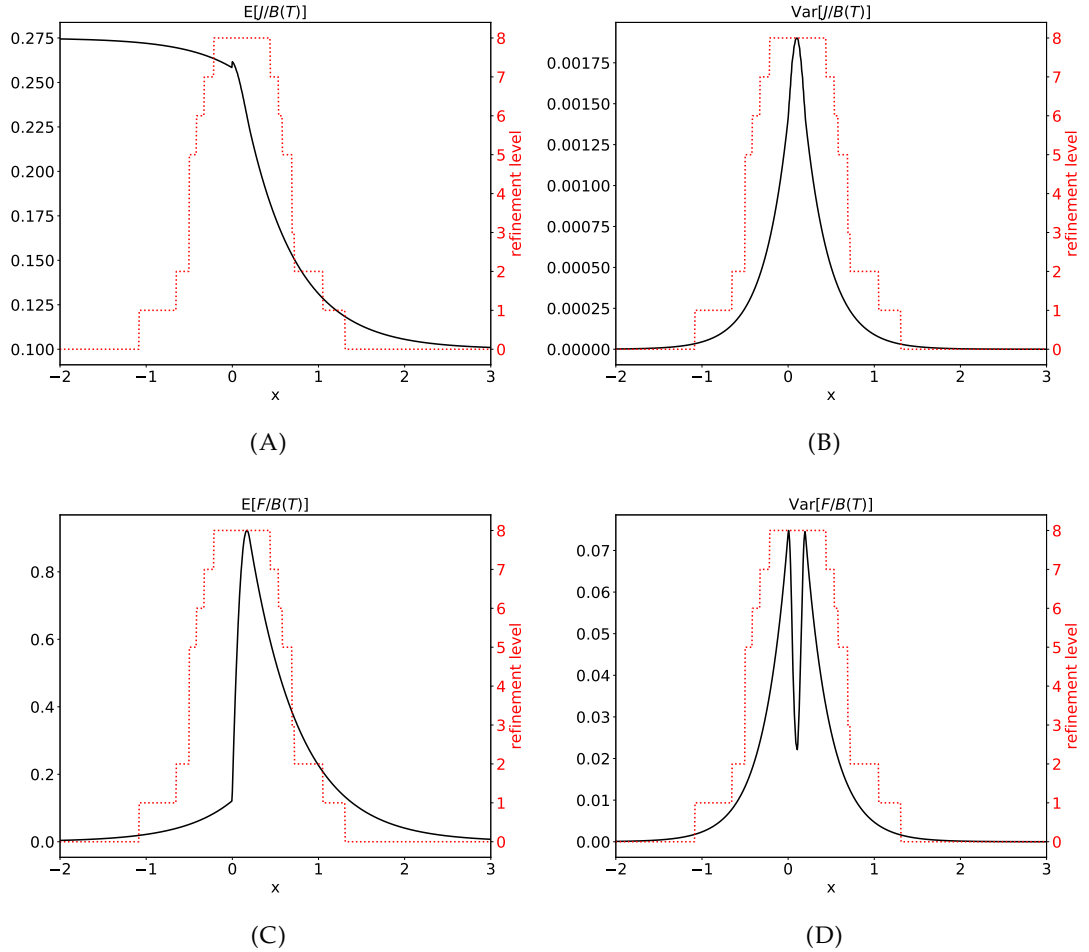


FIGURE 7.2: Expectation value and Variance for the steady radiative shock test case. The red line indicates the refinement level.

7.3.3 Unsteady radiative shock

In the following, we assume $C_V = 0.718 \cdot 10^{-13}$ in which case the temperature of the material will change in time. The initial condition of the temperature is now given by

$$T(\tau = 0, x) = \begin{cases} \alpha & \text{if } x < 0 \\ 1 + \xi_2 & \text{if } x \in [0, \tau_0 + \xi_1] \\ \beta & \text{else} \end{cases},$$

where again ξ_1 is randomly distribute in $[-0.1, 0.1]$ and ξ_2 is randomly distributed in $[-0.5, 0.5]$. Following the previous test case, we choose $\mathcal{E} = T^4/T_r^4$ as well as $F = 0$ at $\tau = 0$ and let the shock evolve in time using the scaled system (7.6) until an end time $\tau_{end} = 5$ is reached. The results are depicted in Figure 7.3. One can observe in Figures 7.3E and 7.3F that now the expectation value and variance of the temperature will change in time. Since the dynamics of the system changes, the refinement levels will be chosen differently by the method. However, one still observes a maximal refinement level in the center of the spatial domain. In contrast to the previously described steady problem, the expectation value of the radiation flux (Figure 7.3C) will not show a jump in the center of the domain, but yields a rather smooth profile. The corresponding variance in Figure 7.3D has bigger values on the right side of the domain.

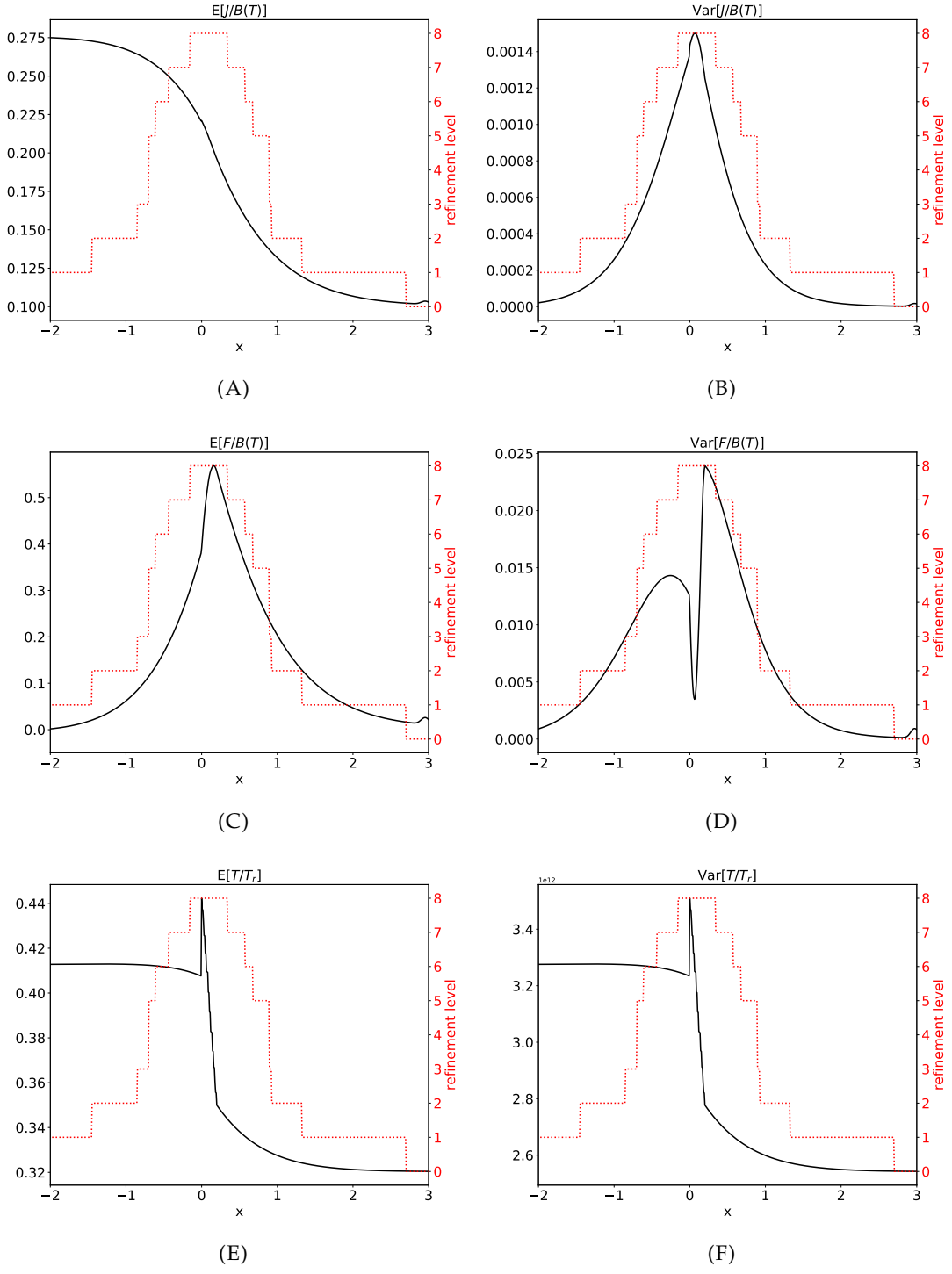


FIGURE 7.3: Expectation value and variance for the steady radiative shock test case. The red line indicates the refinement level.

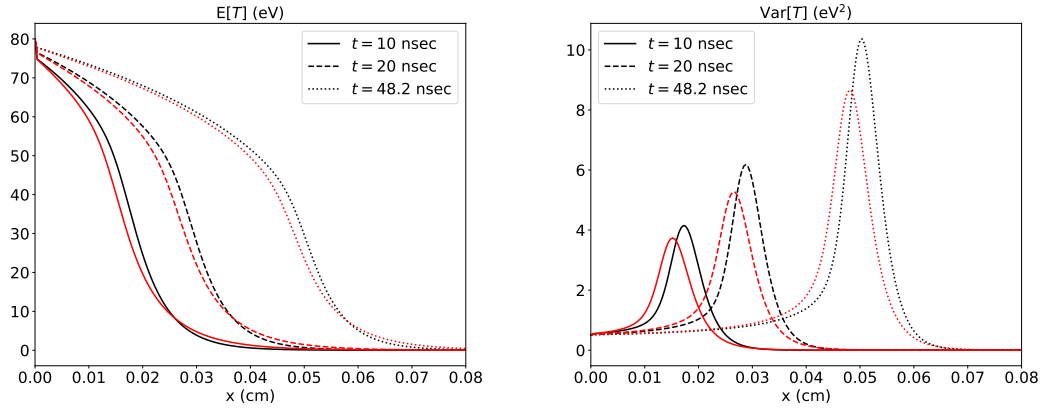
7.3.4 Marshak wave

In the following, we will investigate a heated wall of a cold domain, following [57]. Even though the equation for the temperature does not have a convective term, the temperature will move through the domain due to the interaction with particles (which do have a convective term) for this problem. This phenomenon is called a *Marshak wave*. Its velocity is smaller than the velocity of particles and the system now includes

two scales, namely the speed of particles and the convection of temperature. In the following, we assume that the wall temperature is $T_{\text{wall}} = (80 + \xi)$ eV, where ξ has a normal distribution, i.e. $\xi \sim \mathcal{N}(0,1)$, whereas the domain has a temperature of 0.02 eV. We wish to quantify the effects of this uncertainty with the presented IPM method using moments up to degree $N = 5$. Further problem parameters are given in the following table

$D = [0, 0.2]$	spatial domain in cm
$N_x = 5000$	number spatial cells
$\lambda_{tr} = 92.6$	transport path length in μm
$\rho = 2.7$	material density in g/cm^3
$c_V = 0.831$	specific heat in $J/(g \cdot K)$

The heat capacity is given by $C_V = \rho \cdot c_V$ and the opacity is $\sigma_a = 1/\lambda_{tr}$. As before, we choose $\mathcal{E} = T^4/T_r^4$ as well as $F = 0$ for the initial condition and let the solution evolve according to (7.6). The expectation value and variance of the Marshak wave are depicted in Figure 7.4 for different times and different methods (the black line uses a P_1 discretization, the red line uses a P_3 discretization of the spherical phase space). First, we can see that the expectation value shows a temperature wave, which travels into the spatial domain. The corresponding variance appears to grow in time at the front of the wave. So as time progresses, the wave front will become more and more uncertain. When comparing the solutions computed with different methods, one observes that the results differ significantly: The wave speed of the P_1 solution is increased, while yielding a bigger variance at the wave front.



(A) (B)

FIGURE 7.4: Expectation value (A) and variance (B) of the material temperature. The black line uses a P_1 approximation, the red line uses a P_3 discretization of the spherical phase space.

Chapter 8

Ray-effect mitigation techniques for the Discrete Ordinates Method

So far, we have focused on modal methods to find a finite dimensional description of uncertainties as well as angular variables. Our studies show that a modal solution representation, though possessing several desirable properties (as for example discussed in Section 4), requires one to tackle issues such as the loss of hyperbolicity and oscillatory, non-physical solution approximations. Nodal solution approximations commonly yield stable methods, which preserve important solution bounds. In the context of kinetic equations, properties of modal versus nodal methods can be seen in Figure 1.1. Here, one observes that the S_N method preserves positivity while the P_N method tends to show spurious oscillations and negative solution values. However, while the P_N method preserves rotational symmetry, spurious artifacts seen in the S_N solution destroy symmetry while yielding an unsatisfactory solution approximation. To use nodal discretization methods for problems such as the line-source test case, one therefore needs to come up with strategies to mitigate these so-called ray-effects.

Several methods to mitigate ray-effects have been studied in the literature: In [1] biased quadrature sets are used to reflect the importance of certain ordinates. To increase the number of possible ordinates, several differently oriented quadrature sets are used to compute an average over solutions obtained with different directional biases [132]. A combination of S_N and P_N methods has been introduced in [75] to reduce the appearance of rays. Further refinements can be found in [63, 115, 96] with a comparison given in [100]. The idea of filtering, which has been used for the P_N equations in [91], has been extended to the S_N context in [51]. By transforming the nodal solution on a modal level, filtering can be applied to smear out the solution and thereby mitigate ray-effects [91, 51].

In the following, we presents results from [17] and [36], which have been published with coauthors. This chapter presents portions of this work to which I contributed significantly: I worked out the analytic investigation of rotation and interpolation steps used in [17] and numerically investigated the influence of the spatial grid resolution on the proposed ray-effect mitigation technique. In [36] I implemented the implicit time discretization including the proposed sweeping and Krylov method. I derived stability properties of the second-order sweeping method and conducted the numerical experiments presented in this chapter.

The main idea of [17] is to rotate the set of ordinates around a random axis after each time step to enrich the set of directions in which particles can travel. To obtain solution values on the rotated ordinates, an interpolation step is performed, which is facilitated by the choice of a grid-based quadrature set, similar to [133]. The resulting numerical method is called rotated S_N (rS_N). This chapter presents the fundamental idea of rS_N and discusses the analytic investigation of effects resulting from the consecutive rotation and interpolation steps. Furthermore, we study the influence of the

spatial resolution on the solution. Unfortunately, rS_N does not allow for an efficient sweeping routine, since the rotation step necessitates identifying new sweeping directions in every time step. A method which yields a straight forward and efficient sweeping strategy is given in [36]. Here, we propose to include an artificial, forward-peaked scattering term in the radiative transfer equations. This resembles the idea of filtered P_N [91], which can be understood as adding a forward-peaked scattering operator to the P_N equations. This method is similar to [51]. However, no transformation to the moments is required. We call this method artificial scattering S_N (asS_N). In this chapter, we present the general idea of asS_N and discuss strategies to include the artificial scattering term in standard implicit methods for radiative transfer. To obtain a high accuracy level, we present a second-order upwind sweeping method and discuss stability properties.

8.1 The rS_N method

Before deriving the rS_N method, we present the finite volume discretization of the S_N equations (see Section 1.3.3) used in [17] (which is a re-implementation of [40]). Let us assume that the spatial domain is two-dimensional, which is the case for the test cases investigated in this chapter. We start by dividing the spatial domain into cells

$$V_{ij} := [x_i, x_{i+1}] \times [y_j, y_{j+1}],$$

with volume $|V_{ij}|$. The angular flux into direction $\mathbf{\Omega}_q$ averaged over cell V_{ij} at time step n is denoted by

$$\psi^n_{q,ij} \simeq \frac{1}{|V_{ij}|} \int_{V_{ij}} \psi_q(t_n, \boldsymbol{x})\, d\boldsymbol{x}.$$

Then, the explicit finite volume discretization of (1.64) reads

$$\begin{aligned}\psi^{n+1}_{q,ij} =& \psi^n_{q,ij} - \frac{\Delta t}{|V_{ij}|}\left(f^*_{q,i+1/2,j} - f^*_{q,i-1/2,j} + f^*_{q,i,j+1/2} - f^*_{q,i,j-1/2}\right) \\ &+ \Delta t\, \sigma_{s,ij} \sum_{p=1}^{Q} w_p \psi^n_{p,ij} - \sigma_{t,ij}\, \psi^n_{q,ij}\,. \end{aligned} \tag{8.1}$$

At the interface between cells $V_{i,j}$ and $V_{i,j+1}$ with unit normal $\boldsymbol{n}$, the flux into direction $\mathbf{\Omega}_q$ is approximated by

$$f^*_{q,i,j+1/2} = \begin{cases} \boldsymbol{n}^T\mathbf{\Omega}_q(\psi^n_{q,ij} + \frac{\Delta x}{2}\sigma_{mm}(\psi^n_{q,ij-1}, \psi^n_{q,ij}, \psi^n_{q,ij+1})) & \text{if } \boldsymbol{n}^T\mathbf{\Omega}_q > 0 \\ \boldsymbol{n}^T\mathbf{\Omega}_q(\psi^n_{q,i,j+1} - \frac{\Delta x}{2}\sigma_{mm}(\psi^n_{q,ij}, \psi^n_{q,ij+1}, \psi^n_{q,ij+2})) & \text{else} \end{cases}.$$

The chosen limiter is the *minmod* limiter (1.60) and the numerical fluxes in the remaining spatial directions are chosen accordingly.

8.1.1 Rotation around z-axis

Rotation and interpolation

In this section, we introduce our idea in a pseudo three-dimensional setting. The idea of this setting is to project the solution onto a two-dimensional spatial domain. Then

with $\mu \in [-1,1]$ and $\varphi \in [0,2\pi)$, the angular component introduced in (1.66) becomes

$$\mathbf{\Omega} = \left(\sqrt{1-\mu_k^2}\cos\varphi, \sqrt{1-\mu_k^2}\sin\varphi, 0\right)^T .$$

Now, in addition to evolving the angular flux in time by repeatedly calling the finite volume update (8.1), we rotate the set of ordinates after each timestep around the z-axis. This simplified setting is considered, since rotation around the z-axis as well as the corresponding interpolation step are straight forward when using a standard tensorized quadrature. The case when using an arbitrary rotation will be considered in Section 8.1.2.

First, we present the commonly used product (or tensorized) quadrature set on the sphere: For this, we choose some arbitrary quadrature for $\mu \in [-1,1]$ (e.g. Gauss quadrature for μ) and equally weighted, equally spaced points for φ. Thus, let

$$\varphi_i = i\Delta\varphi \quad \text{for} \quad i = 1,\dots,N_q \quad \text{and} \quad \Delta\varphi = \frac{2\pi}{N_q}, \tag{8.2}$$

where N_q is the number of quadrature points for φ. Then, when for example choosing N_q quadrature points for μ as well, we obtain $Q = N_q^2$ discrete directions

$$\mathbf{\Omega}_{i+(k-1)N_q} = \left(\sqrt{1-\mu_k^2}\cos\varphi_i, \sqrt{1-\mu_k^2}\sin\varphi_i, 0\right)^T .$$

Note that when for example discretizing μ with a Gauss quadrature rule using N_q quadrature points, we will obtain a total number of $Q = N_q^2$ discrete ordinates. Due to the alignment of μ with the z-axis, a rotation around the z-axis only affects the polar angle φ. If we rotate the quadrature set by an angle $\delta \in (0,\Delta\varphi)$, we can approximate the solution on the rotated points $\varphi_i^\delta = \varphi_i + \delta$, $i = 1,\dots,N_q$ using linear interpolation

$$\psi(t_n,\boldsymbol{x},\mu_k,\varphi_i^\delta) = (1-a)\psi(t_n,\boldsymbol{x},\mu_k,\varphi_i) + a\psi(t_n,\boldsymbol{x},\mu_k,\varphi_{i+1}) , \tag{8.3}$$

where $a = \frac{\delta}{\Delta\varphi} \in (0,1)$ and $\varphi_{N_q+1} = \varphi_1$. We call this a *rotation* and *interpolation* step. After having interpolated the solution at the new ordinates

$$\mathbf{\Omega}_{i+(k-1)N_q}^\delta = \left(\sqrt{1-\mu_k^2}\cos\varphi_i^\delta, \sqrt{1-\mu_k^2}\sin\varphi_i^\delta, 0\right)^T ,$$

a finite volume step is performed on the new ordinates, i.e.

$$\begin{aligned}\psi_{q,ij}^{\delta,n+1} =& \psi_{q,ij}^{\delta,n} - \frac{\Delta t}{|V_{ij}|}\left(f_{q,i+1/2,j}^\delta - f_{q,i-1/2,j}^\delta + f_{q,i,j+1/2}^\delta - f_{q,i,j-1/2}^\delta\right) \\ &+ \Delta t\sigma_{s,ij}\sum_{p=1}^{Q} w_p\psi_{p,ij}^{\delta,n} - \sigma_{t,ij}\psi_{q,ij}^{\delta,n} ,\end{aligned} \tag{8.4}$$

with

$$f_{q,i+1/2,j}^\delta = \begin{cases} \boldsymbol{n}^T\mathbf{\Omega}_q^\delta\psi_{q,ij}^{\delta,n} & \text{if } \boldsymbol{n}^T\mathbf{\Omega}_q^\delta > 0 \\ \boldsymbol{n}^T\mathbf{\Omega}_q^\delta\psi_{q,i,j+1}^{\delta,n} & \text{else} \end{cases} .$$

This process is repeated until a specified final time is reached. Conservation of the zeroth-order moment is guaranteed due to linear interpolation.

Modified equation analysis

In this section we analyze the effect of the interpolation. This analysis is based on modified equations, which are a common technique to determine the dispersion or diffusion of the numerical discretization of a hyperbolic balance law [78, Chapter 11.1]. In the following, we assume that δ is fixed and we rotate back and forth between the original and rotated quadrature set. For this setting, we analyze the effects resulting from the combination of rotation/interpolation and update steps.

For simplicity, we only consider the advection operator (which is responsible for the ray-effects)

$$\partial_t \psi + \boldsymbol{\Omega} \cdot \nabla_{\boldsymbol{x}} \psi = 0 \,, \tag{8.5}$$

i.e., collisions and absorption are omitted. The update in time is performed by the explicit Euler method with time step Δt, that is

$$\psi(t_{n+1}, \boldsymbol{x}, \boldsymbol{\Omega}) = \psi(t_n, \boldsymbol{x}, \boldsymbol{\Omega}) - \Delta t \boldsymbol{\Omega} \cdot \nabla_{\boldsymbol{x}} \psi(t_n, \boldsymbol{x}, \boldsymbol{\Omega}) \quad \text{with} \quad t_n = n\Delta t \,. \tag{8.6}$$

In our scheme, the update step (8.6) and the rotation step (8.3) are alternating. In the following, we want to analyze the concatenation of a rotation around the z-axis by an angle δ, an update from t_n to t_{n+1} for some n, and another rotation around the z-axis by an angle $-\delta$, so that we return to the original set of quadrature points. Note that since we are only interested in investigating how an interpolation and rotation step affects the standard S_N time update, we are not considering a full cycle of our scheme as this would include another time update. Furthermore, the spatial variable $\boldsymbol{x}$ and directional coordinate μ are continuous variables for now. First, we apply the interpolation

$$\psi(t_{n+1}, \boldsymbol{x}, \mu, \varphi_i) = (1-a)\psi(t_{n+1}, \boldsymbol{x}, \mu, \varphi_i^\delta) + a\psi(t_{n+1}, \boldsymbol{x}, \mu, \varphi_{i-1}^\delta) \,. \tag{8.7}$$

Second, we perform the update in time

$$\begin{aligned} \psi(t_{n+1}, \boldsymbol{x}, \mu, \varphi_i) = (1-a) &\left(\psi(t_n, \boldsymbol{x}, \mu, \varphi_i^\delta) - \Delta t \boldsymbol{\Omega}_i^\delta \cdot \nabla_{\boldsymbol{x}} \psi(t_n, \boldsymbol{x}, \mu, \varphi_i^\delta) \right) \\ &+ a \left(\psi(t_n, \boldsymbol{x}, \mu, \varphi_{i-1}^\delta) - \Delta t \boldsymbol{\Omega}_{i-1}^\delta \cdot \nabla_{\boldsymbol{x}} \psi(t_n, \boldsymbol{x}, \mu, \varphi_{i-1}^\delta) \right) \,, \end{aligned} \tag{8.8}$$

where $\boldsymbol{\Omega}_i^\delta$ is defined according to (1.66) using φ_i^δ. Finally, we apply the interpolation again

$$\begin{aligned} \psi(t_{n+1}, \varphi_i) = (1-a) \Big(&(1-a)\psi(t_n, \varphi_i) + a\psi(t_n, \varphi_{i+1}) \\ &- \Delta t \boldsymbol{\Omega}_i^\delta \cdot \nabla_{\boldsymbol{x}} \left((1-a)\psi(t_n, \varphi_i) + a\psi(t_n, \varphi_{i+1}) \right) \Big) \\ + a \Big(&(1-a)\psi(t_n, \boldsymbol{x}, \varphi_{i-1}) + a\psi(t_n, \boldsymbol{x}, \varphi_i) \\ &- \Delta t \boldsymbol{\Omega}_{i-1}^\delta \cdot \nabla_{\boldsymbol{x}} \left((1-a)\psi(t_n, \varphi_{i-1}) + a\psi(t_n, \varphi_i) \right) \Big) \,. \end{aligned} \tag{8.9}$$

Here, we omitted the arguments $\boldsymbol{x}$ and μ of the solution ψ to shorten the notation. The above equation can be rewritten as

$$\begin{aligned}\frac{\psi(t_{n+1},\varphi_i)-\psi(t_n,\varphi_i)}{\Delta t}&+\boldsymbol{\Omega}_i\cdot\nabla_{\boldsymbol{x}}\psi(t_n,\varphi_i)\\ &=\frac{a(1-a)}{\Delta t}\Big(\psi(t_n,\varphi_{i+1})-2\psi(t_n,\varphi_i)+\psi(t_n,\varphi_{i-1})\Big)\\ &\quad-a(1-a)\Big(\boldsymbol{\Omega}_i^{\delta}\cdot\nabla_{\boldsymbol{x}}\psi(t_n,\varphi_{i+1})+\boldsymbol{\Omega}_{i-1}^{\delta}\cdot\nabla_{\boldsymbol{x}}\psi(t_n,\varphi_{i-1})\Big)\\ &\quad-\Big((1-a)^2\boldsymbol{\Omega}_i^{\delta}+a^2\boldsymbol{\Omega}_{i-1}^{\delta}-\boldsymbol{\Omega}_i\Big)\cdot\nabla_{\boldsymbol{x}}\psi(t_n,\varphi_i)\,, \end{aligned}\tag{8.10}$$

so that the left-hand side is a discretization of the advection equation (8.5) and the right-hand side is the result of the rotation and interpolation.

Now we require that the scheme has a non-trivial limit for $\Delta t\to 0$. Because of the term $\frac{a(1-a)}{\Delta t}$, we have to choose $a=c\Delta t$ for some constant c. When $\Delta t\to 0$, we then get the following limiting equation

$$\partial_t\psi(t,\varphi_i)+\boldsymbol{\Omega}_i\cdot\nabla_{\boldsymbol{x}}\psi(t,\varphi_i)=c\Delta\varphi^2\frac{\psi(t,\varphi_{i+1})-2\psi(t,\varphi_i)+\psi(t,\varphi_{i-1})}{\Delta\varphi^2}\,.\tag{8.11}$$

This is a semi-discretized advection equation with a discrete second-order derivative in the azimuthal angle on the right-hand side, i.e. $\frac{\partial^2\psi}{\partial\varphi^2}$. However, the right-hand side scales with $\Delta\varphi^2$, so that the diffusive effect of the second-order derivative vanishes with increasing angular resolution $\Delta\varphi\to 0$. On the other hand, for fixed Δt and $\Delta\varphi\to 0$, the above equation (8.10) becomes

$$\frac{\psi(t_{n+1},\varphi)-\psi(t_n,\varphi)}{\Delta t}+\boldsymbol{\Omega}\cdot\nabla_{\boldsymbol{x}}\psi(t_n,\varphi)=0\,.\tag{8.12}$$

This means that the effect of the rotation vanishes when the angular discretization is refined.

The important point of this analysis is that the rotation introduces diffusion (in the angular variable) into the system and that we have to choose a proportional to Δt, i.e. $a=c\Delta t\in[0,1]$ for some constant c. In particular, the angle of the rotation $\delta=a\Delta\varphi=c\Delta t\Delta\varphi$ is proportional to the timestep Δt and the angular discretization $\Delta\varphi$.

Numerical results for S$_N$ with rotation

Let us discuss numerical results for the S$_N$ solution with and without rotation around the z-axis. The solution of S$_N$ with $N=8$ is computed, i.e. we make use of $2\cdot N$ equidistant discretization points for the angle φ and $N/2$ Gauss quadrature points for μ, i.e. the total number of quadrature points is $Q=N^2=64$. For the S$_N$ method with rotation, we rotate the quadrature set back and forth by an angle of $\delta=10\,\Delta t\Delta\varphi$.

Further parameters of the computation as well as details on the line-source test case can be found in Section 8.1.3. Results of the scalar flux

$$\Phi(t_{\text{end}},\boldsymbol{x})=\int_{\mathbb{S}^2}\psi(t_{\text{end}},\boldsymbol{x},\boldsymbol{\Omega})\,d\boldsymbol{\Omega}\tag{8.13}$$

plotted on the physical domain $\boldsymbol{x}\in\mathbb{R}^2$ are shown in Figure 8.2.

The exact scalar flux has the property of being rotationally symmetric, which is especially violated by the S$_N$ method, since the solution suffers from ray effects. By

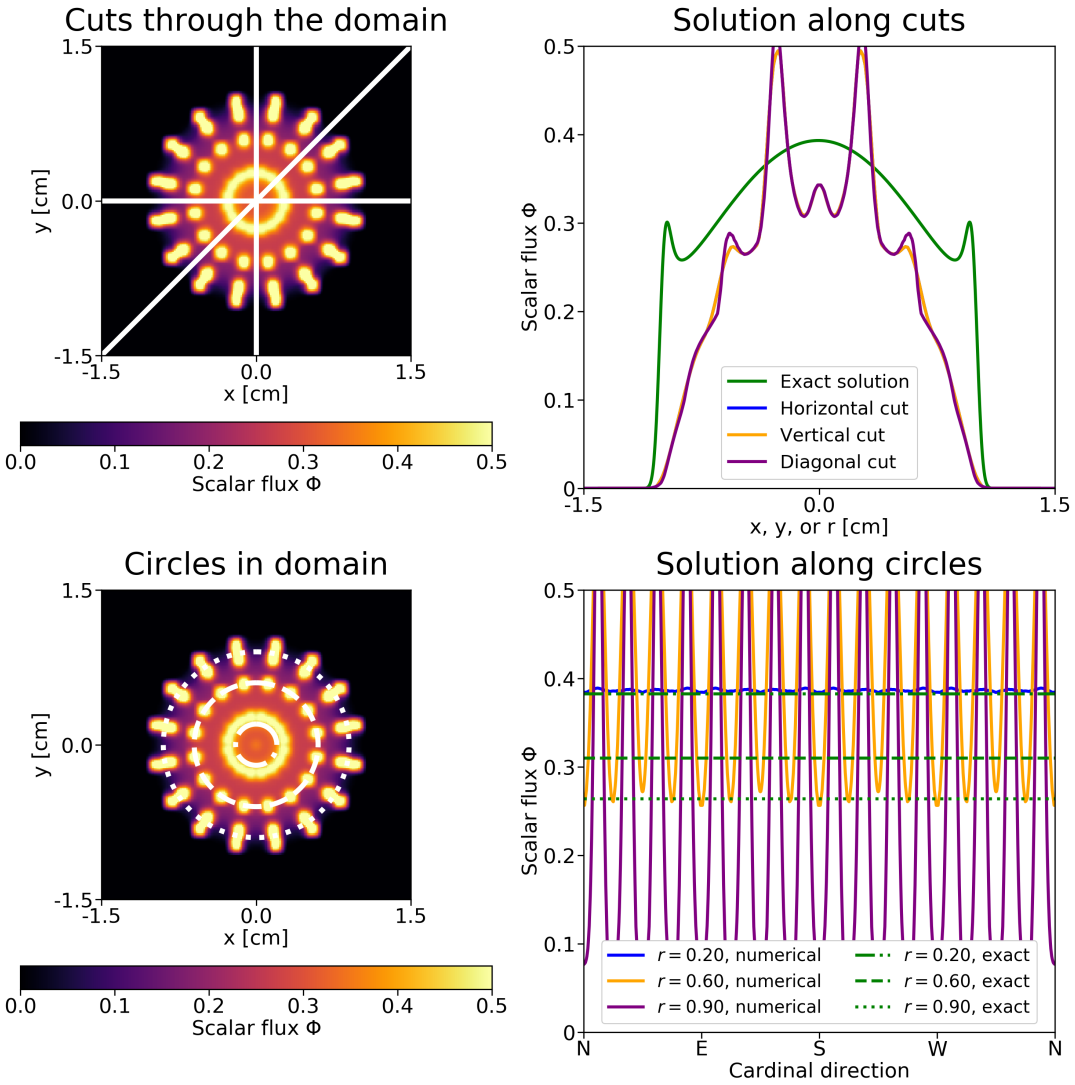


FIGURE 8.1: Scalar flux of S_8 solution with tensorized quadrature. The spatial domain is composed of $N_x \times N_y = 200 \times 200$ spatial cells and the CFL number is 0.95. Cuts through the domain and along circles with different radii are visualized in the right column.

rotating the ordinates of the S_N method around the z-axis, one mitigates ray-effects. However, oscillations are still present in the radial dimension, because we only rotate around the z-axis.

8.1.2 Rotation around arbitrary axis

In the following, we perform rotations along a random axis, meaning that the rotation and especially the interpolation step become more challenging. To enable an efficient interpolation procedure, we construct an ordinate set with an underlying connectivity. In this case, every point on the sphere can be related to a finite number of neighboring points. After having performed the rotation step, the values at the rotated ordinates can be interpolated from the solution values at the original points.

We make use of a quadrature set which results from an underlying triangulation of the unit sphere, i.e. every quadrature point is a node of such a triangulation. This quadrature will be called the *octahedron quadrature*. For more details on the chosen

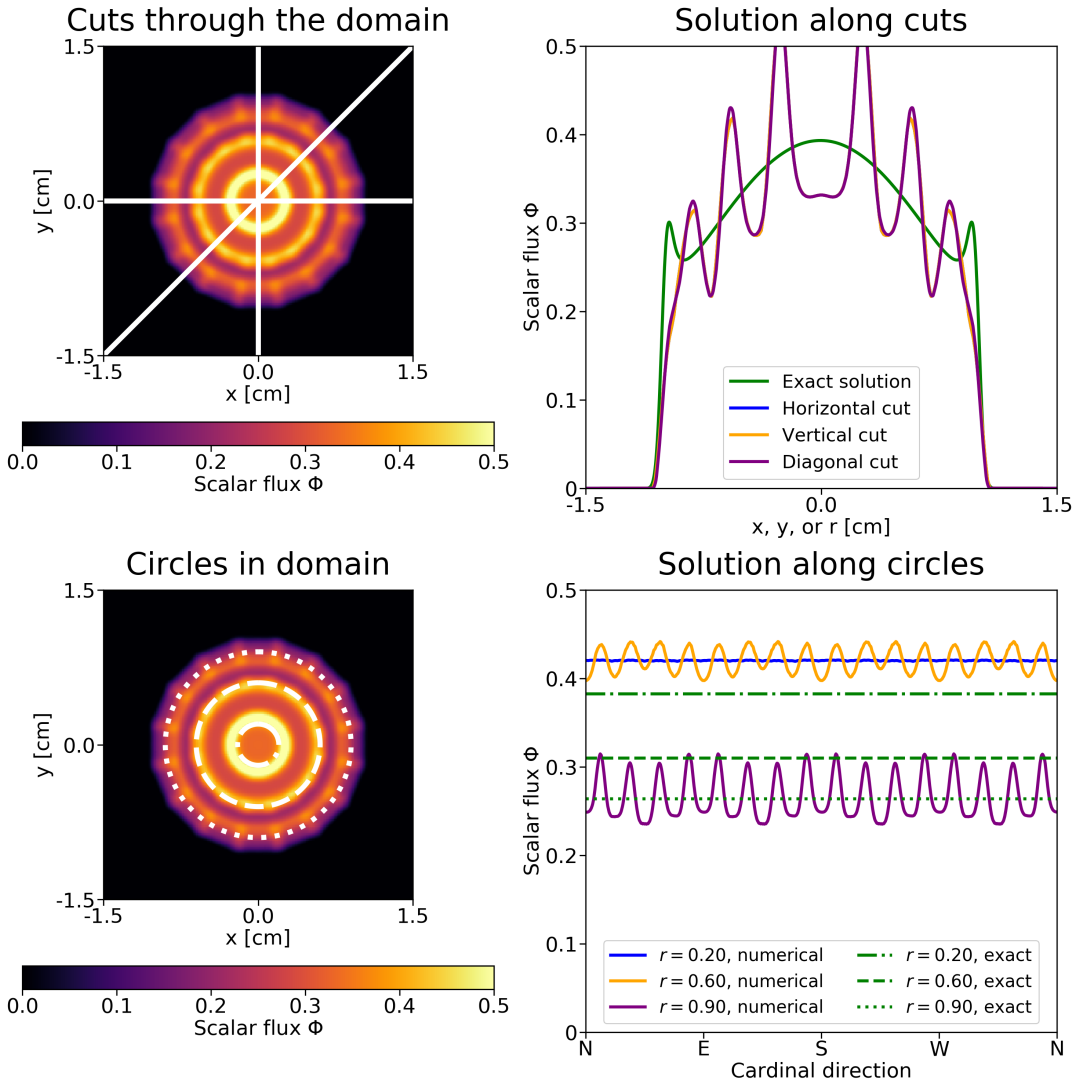


FIGURE 8.2: Scalar flux of S_8 solution with tensorized quadrature when adding the presented rotation.

quadrature set, see [17]. In the following, we discuss how the rotation and interpolation steps are performed.

Rotation and interpolation

In order to perform a rotation step around a given, normalized axis $\boldsymbol{n} = (n_x, n_y, n_z)^T \in \mathbb{R}^3$ with a rotation magnitude δ, we define the rotation matrix $\boldsymbol{R}_\delta^{\boldsymbol{n}}$ as

$$\boldsymbol{R}_\delta^{\boldsymbol{n}} = \begin{pmatrix} n_x^2 a_\delta + \cos(\delta) & u_x u_y a_\delta - n_z \sin(\delta) & n_x n_z a_\delta + n_y \sin(\delta) \\ n_y n_x a_\delta + n_z \sin(\delta) & n_y^2 a_\delta + \cos(\delta) & n_y n_z a_\delta - n_x \sin(\delta) \\ n_z n_x a_\delta - n_y \sin(\delta) & n_z n_y a_\delta + n_x \sin(\delta) & n_z^2 a_\delta + \cos(\delta) \end{pmatrix}, \tag{8.14}$$

where $a_\delta := 1 - \cos(\delta)$. First, a given quadrature point $\boldsymbol{\Omega}_q \in \mathbb{S}^2$ can be rotated around $\boldsymbol{n}$ with a magnitude δ by $\boldsymbol{R}_\delta^{\boldsymbol{n}} \boldsymbol{\Omega}_q =: \boldsymbol{\Omega}_q^\delta$. Now, the rotated quadrature point lies in a triangle of the unit sphere triangulation. The three ordinates of the original quadrature set are then used to compute the solution value at the rotated quadrature point via barycentric interpolation. Figure 8.3 depicts the interpolation step. After having

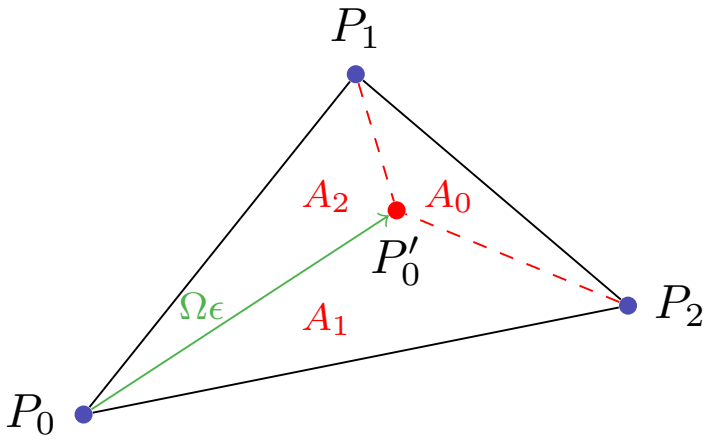


FIGURE 8.3: Original set of quadrature points (blue) and rotated point (red). The corresponding areas A_i are used to interpolate the function value at the rotated quadrature point.

determined the triangle in which the rotated point $\boldsymbol{P}_0'$ will lie, the rotated solution is interpolated using weights $w_i = A_i/A$, where A is the surface area of the entire triangle and $i \in \{0,1,2\}$.

Modified equation for the planar case

We will now consider a simplified setting to explore the effects of the rotation and interpolation step analytically. The fact that the quadrature is still anisotropic makes the analysis on the sphere difficult. We postpone a more detailed discussion to the end of this section.

Assume a triangulation in planar geometry with equilateral triangles that is being translated by a vector $\boldsymbol{\Omega}\,\epsilon$, where $\boldsymbol{\Omega} = (\cos(\alpha), \sin(\alpha))^T$. An excerpt of the original points, together with the surrounding triangles is shown in black and the shifted points with the corresponding triangles in dashed red in Figure 8.4. Each point $\boldsymbol{P}_i$ in the original set of points is being shifted to a new point $\boldsymbol{P}_i' = \boldsymbol{P}_i + \boldsymbol{\Omega}\,\epsilon$. The interpolation weights are again the barycentric weights, shown in Figure 8.3, that is $w_1 = A_1/A, w_2 = A_2/A$ and $w_0 = 1 - w_1 - w_2$, with $A = A_0 + A_1 + A_2$ being the area of the equilateral triangle. The interpolation weights can be computed analytically and expressed in terms of ϵ and α as

$$
\begin{aligned}
w_1(\alpha) &= \frac{2}{\sqrt{3}} \sin\left(\pi/3 - \alpha\right) \epsilon = c_1(\alpha)\epsilon, \\
w_2(\alpha) &= \frac{2}{\sqrt{3}} \sin(\alpha)\epsilon = c_2(\alpha)\epsilon, \\
w_0(\alpha) &= 1 - (c_1(\alpha) + c_2(\alpha))\epsilon.
\end{aligned}
$$

The interpolated function value at $\boldsymbol{P}_0'$ is then given by

$$
\tilde{f}(\boldsymbol{P}_0') = (1 - \epsilon\,(c_1(\alpha) + c_2(\alpha)))f(\boldsymbol{P}_0) + \epsilon\, c_1(\alpha) f(\boldsymbol{P}_1) + \epsilon\, c_2(\alpha) f(\boldsymbol{P}_2).
$$

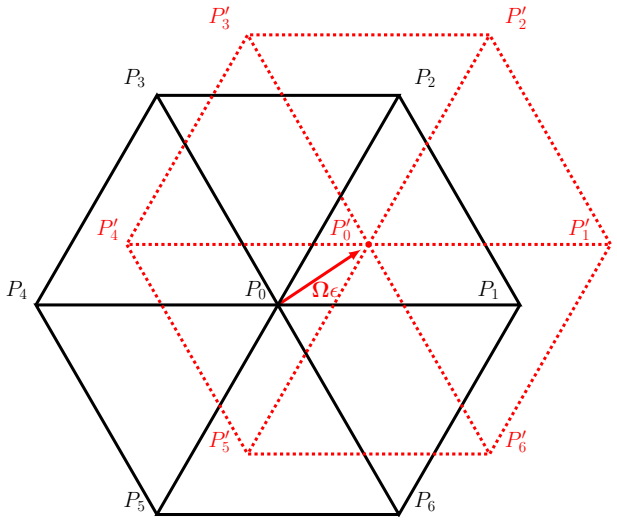


FIGURE 8.4: Original set of quadrature points (black) and the translated points (red), together with the respective triangulation.

Similarly we compute expressions for $\tilde{f}(\boldsymbol{P}_4')$ and $\tilde{f}(\boldsymbol{P}_5')$. If we now reverse the shift by moving all points back into the direction $-\boldsymbol{\Omega}\,\epsilon$, we can interpolate a new value for the point $\boldsymbol{P}_0$ that is

$$\begin{aligned}\tilde{f}(\boldsymbol{P}_0) =&(1-\epsilon\,(c_1(\alpha)+c_2(\alpha)))f(\boldsymbol{P}_0')+\epsilon\,c_1(\alpha)f(\boldsymbol{P}_4')+\epsilon\,c_2(\alpha)f(\boldsymbol{P}_5')\\ =&f(\boldsymbol{P}_0)+f(\boldsymbol{P}_0)+\epsilon\left[c_1(\alpha)\,(f(\boldsymbol{P}_1)-2f(\boldsymbol{P}_0)+f(\boldsymbol{P}_4))\right.\\ &\left.+\,c_2(\alpha)\,(f(\boldsymbol{P}_2)-2f(\boldsymbol{P}_0)+f(\boldsymbol{P}_5))\right].\end{aligned} \tag{8.15}$$

Note that the exact same result would hold in the case of an equilateral triangle with sides of length $\Delta\xi$ when the shift is performed by $\boldsymbol{\Omega}\,\Delta\xi\,\epsilon$. In order to identify the differential operator that the derived stencil approximates, we perform a Taylor expansion around $\boldsymbol{P}_0$, where we use the coordinate axis $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$. These are the axes that run along the hexagonal grid, centered in $\boldsymbol{P}_0$, show in Figure 8.5. From this, we obtain

$$\begin{aligned}f(\boldsymbol{P}_1) &= f(\boldsymbol{P}_0)+\frac{\partial}{\partial\boldsymbol{\xi}_1}f(\boldsymbol{P}_0)\Delta x+\frac{\partial^2}{\partial\boldsymbol{\xi}_1^2}f(\boldsymbol{P}_0)\frac{\Delta\xi^2}{2}+O(\Delta\xi^3),\\ f(\boldsymbol{P}_2) &= f(\boldsymbol{P}_0)+\frac{\partial}{\partial\boldsymbol{\xi}_2}f(\boldsymbol{P}_0)\Delta\xi+\frac{\partial^2}{\partial\boldsymbol{\xi}_2^2}f(\boldsymbol{P}_0)\frac{\Delta\xi^2}{2}+O(\Delta\xi^3).\end{aligned}$$

Plugging this into the derived stencil (8.15) gives

$$\tilde{f}(\boldsymbol{P}_0)=f(\boldsymbol{P}_0)+\epsilon\Delta\xi^2\left(c_1(\alpha)\left[\frac{\partial^2}{\partial\boldsymbol{\xi}_1^2}f(\boldsymbol{P}_0)\right]+c_2(\alpha)\left[\frac{\partial^2}{\partial\boldsymbol{\xi}_2^2}f(\boldsymbol{P}_0)\right]\right)+O(\Delta\xi^3).$$

Instead of writing the derivatives in dependency of $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ we transform the derivatives to only rely on the direction $\boldsymbol{\Omega}$ and the direction perpendicular to $\boldsymbol{\Omega}$, namely $\boldsymbol{\Omega}^\perp$. From the geometry of Figure 8.5 follows

$$\begin{aligned}&\langle\boldsymbol{\Omega},\boldsymbol{\xi}_1\rangle=\cos(\alpha),\quad\langle\boldsymbol{\Omega},\boldsymbol{\xi}_2\rangle=\cos(\pi/3-\alpha),\\ &\langle\boldsymbol{\Omega}^\perp,\boldsymbol{\xi}_1\rangle=-\sin(\alpha),\quad\langle\boldsymbol{\Omega}^\perp,\boldsymbol{\xi}_2\rangle=\sin(\pi/3-\alpha).\end{aligned}$$

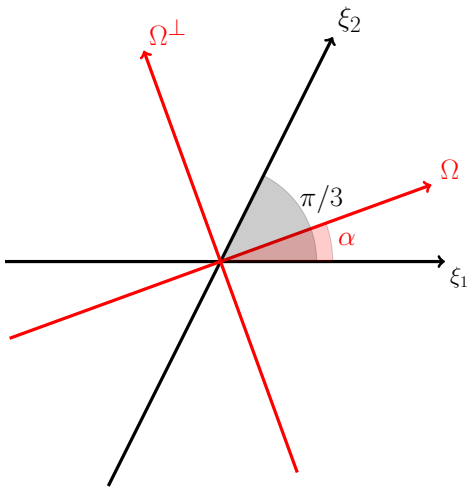


FIGURE 8.5: Transforming from the $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2)$ coordinate system to $(\boldsymbol{\Omega}, \boldsymbol{\Omega}^\perp)$.

Together with

$$\frac{\partial^2}{\partial \boldsymbol{\xi}_i^2} = \langle \boldsymbol{\Omega}, \boldsymbol{\xi}_i \rangle^2 \frac{\partial^2}{\partial \boldsymbol{\Omega}^2} + 2\langle \boldsymbol{\Omega}, \boldsymbol{\xi}_i \rangle \langle \boldsymbol{\Omega}^\perp, \boldsymbol{\xi}_i \rangle \frac{\partial^2}{\partial \boldsymbol{\Omega} \partial \boldsymbol{\Omega}^\perp} + \langle \boldsymbol{\Omega}^\perp, \boldsymbol{\xi}_i \rangle^2 \frac{\partial^2}{\partial \boldsymbol{\Omega}^{\perp 2}}$$

we obtain

$$\begin{aligned}
c_1(\alpha)\frac{\partial^2}{\partial \boldsymbol{\xi}_1^2} =& \frac{2}{\sqrt{3}} \sin\left(\pi/3 - \alpha\right) \left[\cos(\alpha)^2 \frac{\partial^2}{\partial \boldsymbol{\Omega}^2} - 2\cos(\alpha)\sin(\alpha)\frac{\partial^2}{\partial \boldsymbol{\Omega} \partial \boldsymbol{\Omega}^\perp} + \sin(\alpha)^2 \frac{\partial^2}{\partial \boldsymbol{\Omega}^{\perp 2}}\right], \\
c_2(\alpha)\frac{\partial^2}{\partial \boldsymbol{\xi}_2^2} =& \frac{2}{\sqrt{3}} \sin\left(\alpha\right) \big[\cos(\pi/3 - \alpha)^2 \frac{\partial^2}{\partial \boldsymbol{\Omega}^2} + 2\cos(\pi/3 - \alpha)\sin(\pi/3 - \alpha)\frac{\partial^2}{\partial \boldsymbol{\Omega} \partial \boldsymbol{\Omega}^\perp} \\
& + \sin(\pi/3 - \alpha)^2 \frac{\partial^2}{\partial \boldsymbol{\Omega}^{\perp 2}}\big].
\end{aligned}$$

Next, we substitute $\alpha = \beta + \pi/6$ with $\beta \in [-\pi/6, \pi/6]$ to obtain

$$\begin{aligned}
& c_1(\alpha)\frac{\partial^2}{\partial \boldsymbol{\xi}_1^2} + c_2(\alpha)\frac{\partial^2}{\partial \boldsymbol{\xi}_2^2} = c_1(\pi/6 + \beta)\frac{\partial^2}{\partial \boldsymbol{\xi}_1^2} + c_2(\pi/6 + \beta)\frac{\partial^2}{\partial \boldsymbol{\xi}_2^2} \\
& = c_{\boldsymbol{\Omega}^2}(\beta)\frac{\partial^2}{\partial \boldsymbol{\Omega}^2} + c_{\boldsymbol{\Omega}\boldsymbol{\Omega}^\perp}(\beta)\frac{\partial^2}{\partial \boldsymbol{\Omega} \partial \boldsymbol{\Omega}^\perp} + c_{\boldsymbol{\Omega}^{\perp 2}}(\beta)\frac{\partial^2}{\partial \boldsymbol{\Omega}^{\perp 2}}.
\end{aligned}$$

Here, we defined the following constants

$$\begin{aligned}
c_{\boldsymbol{\Omega}^2}(\beta) &= \frac{1}{2\sqrt{3}}(4\cos(\beta) - \cos(3\beta)), \\
c_{\boldsymbol{\Omega}\boldsymbol{\Omega}^\perp}(\beta) &= \frac{1}{2\sqrt{3}}\left(-4\sin(\beta) + 2\sin(3\beta)\right), \\
c_{\boldsymbol{\Omega}^{\perp 2}}(\beta) &= \frac{1}{2\sqrt{3}}\cos(3\beta),
\end{aligned}$$

which are visualized in Figure 8.6. Finalizing the change of coordinate systems, we

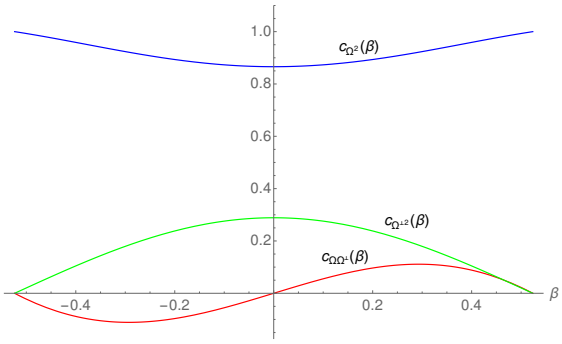


FIGURE 8.6: Magnitudes of the different second derivative operators in dependency of β.

derive

$$\begin{aligned}\tilde{f}(\boldsymbol{P}_0) =& f(\boldsymbol{P}_0) + \epsilon \left(c_1(\alpha) \left[\frac{\partial^2}{\partial \boldsymbol{\xi}_1^2} f(\boldsymbol{P}_0) \Delta\xi^2 \right] + c_2(\alpha) \left[\frac{\partial^2}{\partial \boldsymbol{\xi}_2^2} f(\boldsymbol{P}_0) \Delta\xi^2 \right] \right) + O(\Delta\xi^3) \\ =& f(\boldsymbol{P}_0) + \epsilon \Delta\xi^2 \left(c_{\boldsymbol{\Omega}^2}(\beta) \frac{\partial^2}{\partial \boldsymbol{\Omega}^2} + c_{\boldsymbol{\Omega}\boldsymbol{\Omega}^\perp}(\beta) \frac{\partial^2}{\partial \boldsymbol{\Omega} \partial \boldsymbol{\Omega}^\perp} + c_{\boldsymbol{\Omega}^{\perp 2}}(\beta) \frac{\partial^2}{\partial \boldsymbol{\Omega}^{\perp 2}} \right) f(\boldsymbol{P}_0) \\ &+ O(\Delta\xi^3). \end{aligned} \tag{8.16}$$

We are now going to draw some conclusions from this derivation. First, we observe a diffusive behavior, where diffusion is strongest along the direction of shifting. Furthermore, when the shift is performed in alignment with the lattice, diffusion only occurs along that direction.

Comparing equation (8.16) with (8.11) we observe a similar scaling behavior of the diffusion term. Choosing $\epsilon = \delta\,\Delta t/\Delta\xi$ again results in vanishing diffusion for a refinement of the angular discretization ($\Delta\xi \to 0$). Since $\Delta\xi$ scales like the number of angular quadrature points Q, we will perform rotations by $\delta\Delta t/Q$ instead of $\delta\Delta t/\Delta\xi$. This is due to the fact that $\Delta\xi$ is not constant for the different triangles on the sphere, it does however scale like Q.

The implementation of the rS$_N$ method differs from this simplified analysis. There, we are moving the quadrature points on the sphere and not in planar geometry. Furthermore, not all triangles have the same size. Additionally, determining the corresponding points from which we interpolate the function values is not trivial. It might happen, that two rotated points fall into the same triangle of the old quadrature set. In the implementation of the rS$_N$ method, we also rotate randomly around different axes and perform the usual S$_N$ update, i.e. stream and collide, in between interpolating. All these aspects make the theoretical analysis significantly more difficult than for the simplified planar geometry. However, the numerical results indicate a diffusive behavior which is equally strong in any direction. We believe that randomly choosing a rotation axis has an averaging effect. That is, diffusion will be equally strong along all directions since we rotate differently in each time step. As we are no longer restricted to rotations around the z-axis only, diffusion is also not restricted to occur in the azimuthal angle.

A different way to interpret the newly induced diffusion is artificial scattering. Rotating the quadrature set and interpolating the angular flux can be seen as particles scattering into new directions. As it is the case for scattering, the rotation and interpolation procedure is conservative by construction. Furthermore, we can write the interpolation procedure as $\tilde{\psi}_{i,j} = I\psi_{i,j}$. Here, $\psi_{i,j}$ is the angular flux in a given spatial cell $c_{i,j}$ before the rotation and interpolation step and $\tilde{\psi}_{i,j}$ is the angular flux at the new quadrature points after the rotation and interpolation step. The matrix I contains the interpolation

weights and has only three non zero elements per row. It does not depend on the spatial cell index, but is different for each time step since the rotation axis differs from time step to time step. Thus, the angular flux at Ω_q "scatters" into the directions for which it is used to interpolate function values at the new quadrature set. Since more scattering implies fewer ray-effects, the rS$_N$ method mitigates these undesired effects.

8.1.3 Results rS$_N$

Line-source test case

In the following, we will present the numerical results for the rS$_N$ method and compare it to S$_N$ for the line-source and the lattice test case. The line-source test case computes the evolution of particles for an initial isotropic Dirac-mass at the center of the spatial domain $D = [-1.5, 1.5] \times [-1.5, 1.5]$, which is composed of a material with $\sigma_t = \sigma_s = 1$. To be more specific, the initial particle distribution is $\psi(t = 0, \boldsymbol{x}, \boldsymbol{\Omega}) = 1/4\pi\,\delta(\boldsymbol{x})$, which on the discrete level is represented by a Gaussian, i.e. $\psi(t = 0, \boldsymbol{x}, \boldsymbol{\Omega}) = \max\{10^{-4}, 1/4\pi\delta \exp(-\boldsymbol{x}^2/4\delta)\}$ using a small variance $\delta = 0.03^2$. Since a semi-analytical solution to this problem exists [38], as well as the fact that this test case nicely reveals shortcomings of various numerical methods for radiation transport, it is frequently used in the literature. The exact solution shows a circular front, which moves away from the center, while leaving a tail of particles, which have not traveled in the x-y-plane during the entire time of the simulation.

Looking at Figure 8.7 one can see the S$_N$ solution for the chosen octahedron quadrature using $Q = 198$ quadrature points at time $t_{\text{end}} = 1$. Here, the ray-effects are seen as points on the unit sphere. To mitigate ray-effects, we will use $\delta = 8$, which is the result of a parameter study conducted in [17]. The resulting solution can be found in Figure 8.8. On can see that the rS$_N$ method reduces ray-effects significantly, yielding a solution which shows good agreement with the exact line-source solution.

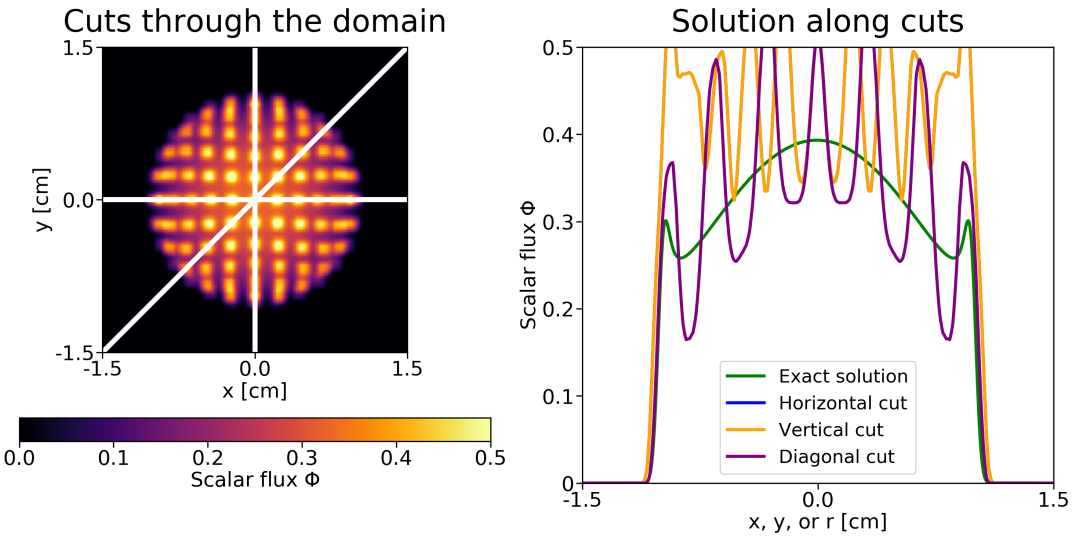


FIGURE 8.7: Scalar flux of S$_8$ solution with ray-effects. We choose $Q = 198$ quadrature points, the spatial domain is composed of $N_x \times N_y = 200 \times 200$ spatial cells and the CFL number is 0.95. Cuts through the domain and along circles with different radii are visualized in the right column. The chosen ordinates belong to octahedron quadrature.

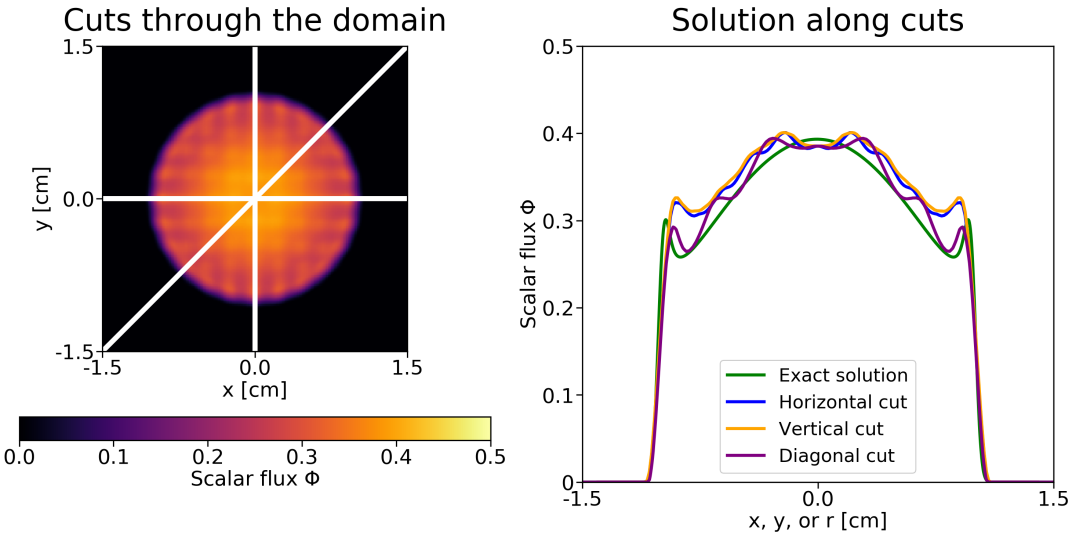


FIGURE 8.8: Scalar flux of rS$_8$ solution with mitigated ray-effects. We choose $Q = 198$ quadrature points, the spatial domain is composed of $N_x \times N_y = 200 \times 200$ spatial cells and the CFL number is 0.95. Cuts through the domain and along circles with different radii are visualized in the right column. We set $\delta = 8$ and use the octahedron quadrature.

Now let us fix all parameters except for the spatial and time resolution. Figure 8.9 shows the change of the numerical solution when refining the number of spatial discretization points, which directly affects the number of timesteps. It can be seen that the S$_N$ method shows more dominant ray-effects when the spatial resolution is refined. After a refinement of 500 cells per dimension, the solution does not change anymore. When looking at the rS$_N$ solution in the second row of Figure 8.9, on observes that the solution shows small ray-effects while still showing a satisfactory overall solution approximation when increasing the resolution. Compared to the S$_N$ case, the ray-effects are dampened heavily by the rotation step. Cuts into the spatial domain along horizontal, vertical and diagonal lines can be found in Figure 8.10. One again observes the previously seen picture: While both methods show increased ray-effects when the spatial resolution is refined, the rS$_N$ method heavily mitigates these artifacts and yields a solution which shows good agreement with the exact solution for all spatial resolutions.

Lattice test case

In the following, we investigate the lattice test case, which resembles a heavily simplified nuclear reactor. This test case has a quadratic source ($Q = 1$) in the center as well as purely absorbing blocks with $\sigma_a = 10$, which are surrounded by a purely scattering material with $\sigma_s = 1$. Particles enter through the source in the center of the spatial domain. The calculation is performed until an end time $t_{\text{end}} = 3.2$ is reached. The numerical S$_8$ solution shows ray-effects when particles travel through the scattering medium. To mitigate these artifacts, we run the rS$_N$ method with $\delta = 8$. This parameter has been determined by a study conducted in [17]. The effects of the rS$_N$ can be found in Figure 8.11, which shows that the rotation mitigates ray-effects, yielding a satisfactory solution approximation. Note that compared to the line-source, this test case does not yield a rotational symmetric solution. Hence, one can conclude that the rS$_N$ method is not tailored to problems that have certain symmetry properties but also works well for non-symmetric settings.

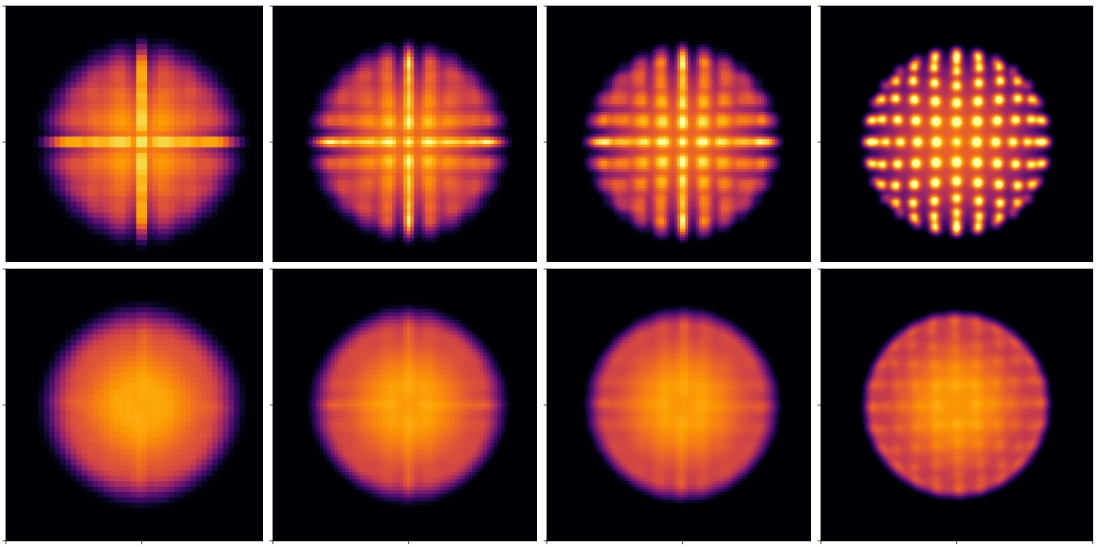


FIGURE 8.9: Scalar flux for the line-source problem. Parameters are chosen as in Figure 8.8, except for varying spatial resolution. First row shows S_8 solution, second row shows rS_8 solution. The number of spatial cells per dimension is $50, 75, 100, 500$ from left to right. Compare to the reference solution in Figure 1.1. The colorbar is the same as in Figure 1.1.

Now, let us again investigate how refining and coarsening the spatial grid affects the approximation quality of both, the S_N and rS_N methods. Therefore, we compute the solution to the lattice test case with different grid resolutions, yielding Figure 8.12. While the S_N method shows ray-effects (at all spatial resolutions), which become more dominant when the grid is refined, the rS_N method does not show ray-effects for all resolutions. Refining the spatial mesh will slightly improve solution properties, which can be seen when comparing the rS_N approximations to the reference solution in Figure 8.11.

8.2 The asS_N method

In the following, we propose a further technique to mitigate ray-effects. Despite convincing results and properties shown for the rS_N method, a certain effort is required to add rS_N to a given S_N framework. Also, in combination with implicit time discretizations, the quadrature rotation forces us to recompute so-called sweeping directions, which increases numerical costs. A simpler idea, which allows a straight forward integration into existing codes while enabling an efficient implicit implementation is to add an artificial, forward peaked kernel to the original Boltzmann equation (1.64). This kernel then adds neighboring directions on which particles can move, i.e. certain rays become less dominant. The Boltzmann equation with artificial scattering then reads

$$\begin{aligned}\partial_t \psi(t,\boldsymbol{x},\boldsymbol{\Omega}) + \boldsymbol{\Omega}\cdot\nabla_{\boldsymbol{x}}\psi(t,\boldsymbol{x},\boldsymbol{\Omega}) + \sigma_a(\boldsymbol{x})\psi(t,\boldsymbol{x},\boldsymbol{\Omega}) \\ = \sigma_s(S\psi)(t,\boldsymbol{x},\boldsymbol{\Omega}) + \sigma_{as}(S_{as}\psi)(t,\boldsymbol{x},\boldsymbol{\Omega}) + q(t,\boldsymbol{x},\boldsymbol{\Omega}),\end{aligned} \tag{8.17}$$

which we call artificial scattering S_N (asS_N) equation in the following. For ease of presentation, we assume isotropic scattering and split the scattering kernel into an in-scattering and out-scattering part. I.e. we have

$$S\psi = S^+\psi - \psi$$

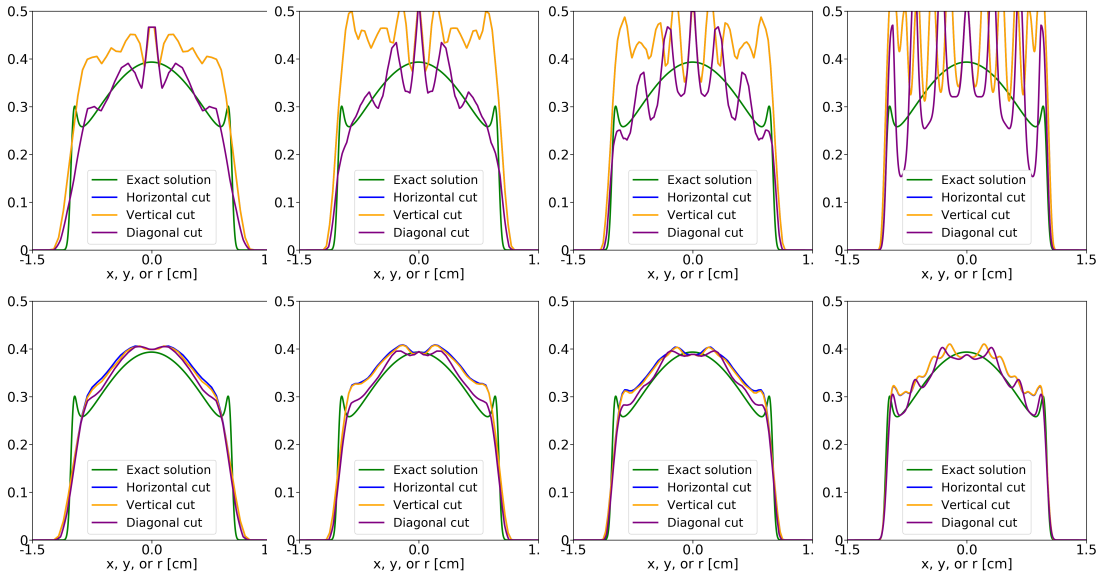


FIGURE 8.10: Scalar flux for the line-source problem. Parameters are chosen as in Figure 8.8, except for varying spatial resolution. First row shows S_8 solution, second row shows rS$_8$ solution. The number of spatial cells per dimension is $50, 75, 100, 500$ from left to right. The horizontal and vertical cuts are identical for S$_N$.

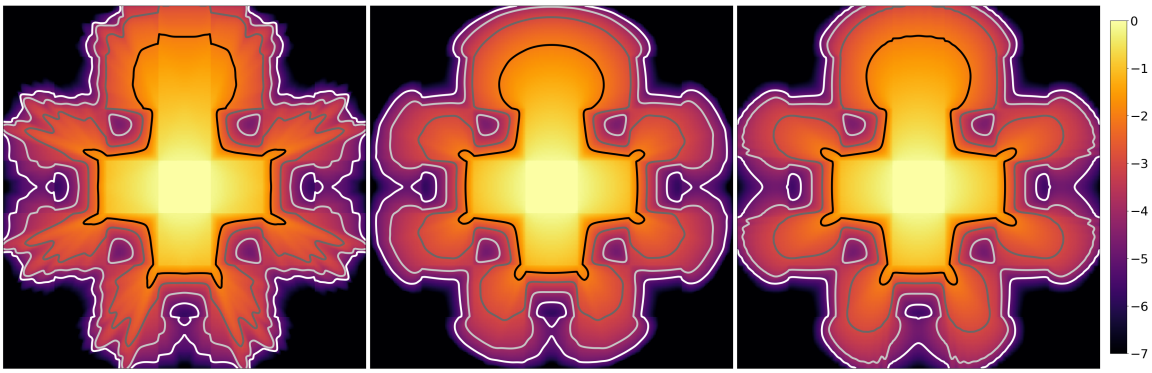


FIGURE 8.11: Scalar flux for the lattice problem. From left to right: S$_8$, rS$_8$ with $\delta = 8$, reference solution S$_{23}$ using $N_x = N_y = 280$.

where $S^+\psi := \int_{\mathbb{S}^2} \psi(t, \boldsymbol{x}, \boldsymbol{\Omega}')\, d\boldsymbol{\Omega}'$. The same is done for the artificial scattering kernel, which then reads

$$S_{as}\psi := S_{as}^+\psi - \psi$$

with $S_{as}^+\psi := \int_{\mathbb{S}^2} s_\varepsilon(\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}')\psi(t, \boldsymbol{x}, \boldsymbol{\Omega}')\, d\boldsymbol{\Omega}'$. The in-scattering cross section $s_\varepsilon : \mathbb{R} \to \mathbb{R}$ is chosen to be

$$s_\varepsilon(\mu) := \frac{c}{\varepsilon} e^{\frac{-(\mu-1)^2}{\varepsilon^2}},$$

where c is chosen to normalize the scattering kernel, i.e. $\int_{\mathbb{S}^2} s_\varepsilon(\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}')\, d\boldsymbol{\Omega}' = 1$ for all $\boldsymbol{\Omega} \in \mathbb{S}^2$. Note that the chosen kernel approaches a Dirac when the variance ε approaches zero, i.e. artificial scattering is turned off. The asS$_N$ method has the following effects:

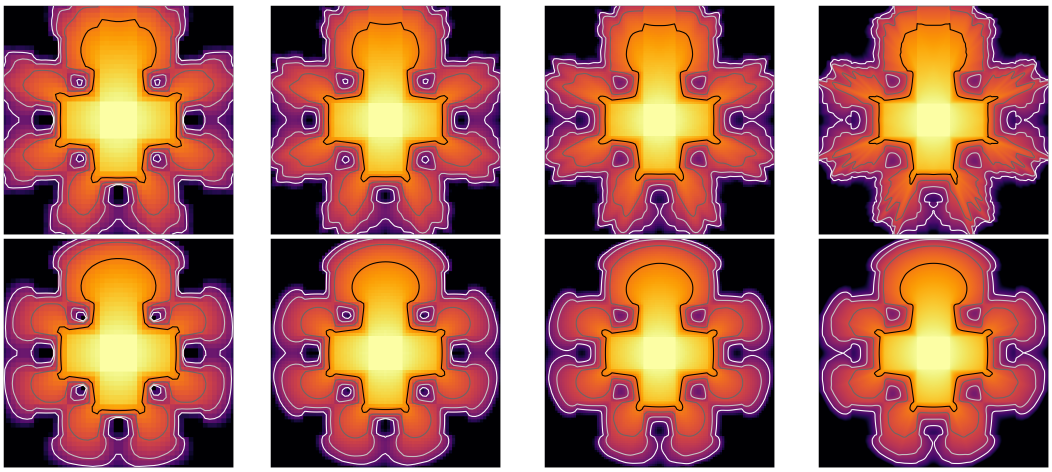


FIGURE 8.12: Scalar flux for the lattice problem. First row shows S_6 solution, second row shows rS_6 solution with $\delta = 8$. The number of spatial cells per dimension is $50, 75, 100, 500$ from left to right. The colorbar is the same as in Figure 8.11.

1. As shown in [36], the artificial scattering yields an angular diffusion term in the radiative transfer equations, similar to the artificial viscosity term in numerical schemes for hyperbolic equations [79, Chapter 16.1]. To ensure that this term vanishes when the number of quadrature points approaches infinity, we choose $\varepsilon = \beta/Q$, with a user determined parameter β. Consequently, the asS_N solution converges to the S_N solution when $Q \to \infty$.

2. While the asS_N method preserves the total number of particles, it dampens high-order moments. This behavior is similar to filtered P_N [91]. Similar to filtered P_N, the artificial scattering acts as a filter on the moment level [51].

3. Physically correct rays, such as a beam of particles inside a void will be smeared out by artificial scattering. The classical S_N method will however lead to unwanted artifacts for such a problem as well (if the direction of the beam is not aligned with a quadrature direction).

4. asS_N has similarities to the P_{N-1}-equivalent S_N method [96]: To mitigate ray-effect, this method adds a fictitious source to the radiative transfer equation. This source, though derived by a different strategy, requires similar modifications of the standard S_N implementation. The main difference is that the artificial scattering kernel of asS_N is forward peaked, which can be used to design an efficient numerical treatment.

5. The as-S_N equation (8.17) can—with appropriate boundary and initial conditions—be solved in a straight-forward manner using common S_N implementations. In the remainder of this chapter, we will focus on implicit discretization techniques and derive an efficient algorithm to treat the artificial scattering term.

8.2.1 Implicit time discretization

Implicit time discretization methods provide stability for large time steps, which are crucial in applications involving different time scales. However, when discretizing the radiative transfer equation, they require a matrix solve in every time step, which is commonly performed by a Krylov solver [34, 6]. We start with an implicit Euler discretization, where we, in an abuse of notation, denote the flux at the new time step by

$\psi(\boldsymbol{x}, \boldsymbol{\Omega})$ and at the old time step by $\psi^{\text{old}}(\boldsymbol{x}, \boldsymbol{\Omega})$. The equivalent asS$_N$ system is

$$\boldsymbol{\Omega} \cdot \nabla_{\boldsymbol{x}} \psi + \left(\sigma_a + \sigma_s + \sigma_{\text{as}} + \frac{1}{\Delta t} \right) \psi = \sigma_s S^+ \psi + \sigma_{\text{as}} S^+_{\text{as}} \psi + q + \frac{\psi^{\text{old}}}{\Delta t}. \tag{8.18}$$

Defining the streaming operator $L\psi := \boldsymbol{\Omega} \cdot \nabla_{\boldsymbol{x}} \psi + \left(\sigma_a + \sigma_s + \sigma_{\text{as}} + \frac{1}{\Delta t} \right) \psi$ as well as the modified source $\tilde{q} := q + \psi^{\text{old}} / \Delta t$, we can put this into more compact notation

$$L\psi = \sigma_s S^+ \psi + \sigma_{\text{as}} S^+_{\text{as}} \psi + \tilde{q}. \tag{8.19}$$

First, let us numerically treat the artificial scattering in the same way as commonly done for physical scattering. The physical in-scattering kernel can be written as

$$S^+ = O \Sigma M,$$

where Σ carries the respective expansion coefficients of the scattering kernel, M maps from the ordinates to the moments and O from the moments back to the angular space. Making use of this strategy to represent the artificial scattering, we get

$$S^+_{\text{as}} = O \Sigma_{\text{as}} M. \tag{8.20}$$

When denoting the moments as $\phi = M\psi$, equation (8.19) becomes

$$L\psi = \sigma_s O \Sigma \phi + \sigma_{\text{as}} O \Sigma_{\text{as}} \phi + \tilde{q} \, . \tag{8.21}$$

Inverting L and applying M to both sides yields the fixed point equations

$$\phi = \sigma_s M L^{-1} O \Sigma \phi + \sigma_{\text{as}} M L^{-1} O \Sigma_{\text{as}} \phi + M L^{-1} \tilde{q} \, . \tag{8.22}$$

Note that with $\sigma_{\text{as}} = 0$, this is the standard equation to which a Krylov solver is applied. Choosing a non-zero artificial scattering strength can result in significantly increased numerical costs when solving (8.22) with a Krylov method: To show this, let us move to the discrete level, i.e. discretizing the directional domain, which requires picking a finite number of moments. In this case Σ becomes a diagonal matrix with entries falling rapidly to zero (in the case of isotropic scattering, only the first entry is non-zero). Hence, few moments are required to capture the effects of physical scattering. However, since the artificial scattering kernel is strongly forward-peaked, the entries of the diagonal matrix Σ_{as} do not fall to zero quickly, meaning that the method requires a large number of moments to include artificial scattering, which results in a heavily increased run time [51]. The slow decay of the Legendre moments $k_{\varepsilon,n} = 2\pi \int_{-1}^{+1} s_\varepsilon(\mu) P_n(\mu) \, d\mu$ for $\varepsilon \to 0$ is visualized in Figure 8.13. In order to be able to choose the reduced number of moments required to resolve physical scattering, we move the artificial scattering into the sweeping step. Hence, going back to equation (8.19), we only perform the moment decomposition on the physical scattering to obtain

$$(L - \sigma_{\text{as}} S^+_{\text{as}}) \psi = \sigma_s O \Sigma \phi + \tilde{q} \, . \tag{8.23}$$

Moving the operator $(L - \sigma_{\text{as}} S^+_{\text{as}})$ to the right hand side and taking moments yields

$$\phi = \sigma_s M (L - \sigma_{\text{as}} S^+_{\text{as}})^{-1} O \Sigma \phi + M (L - \sigma_{\text{as}} S^+_{\text{as}})^{-1} \tilde{q} \, . \tag{8.24}$$

The Krylov solver is then applied to this fixed-point iteration. In contrast to (8.22), the physical scattering term dictates the number of moments. The computation of $(L - \sigma_{\text{as}} S^+_{\text{as}})^{-1}$ is performed by a source iteration, where the general equation $(L - \sigma_{\text{as}} S^+_{\text{as}}) \psi =$

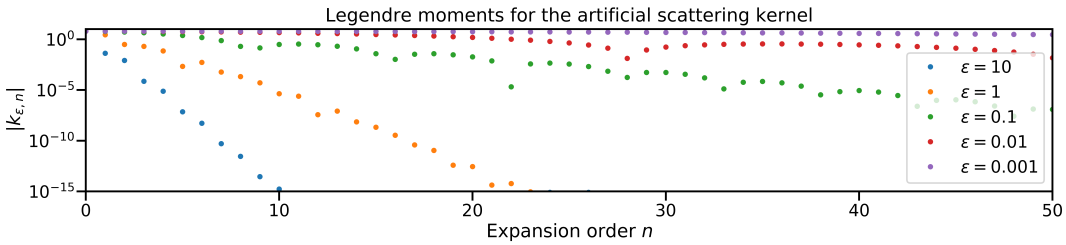


FIGURE 8.13: Decay of the Legendre moments $k_{\varepsilon,n} = 2\pi \int_{-1}^{+1} s_\varepsilon(\mu) P_n(\mu)\, d\mu$ for different values of ε and the expansion order n.

R is solved by iterating on

$$L\psi^{(l+1)} = \sigma_{\text{as}} S_{\text{as}}^{+} \psi^{(l)} + R. \tag{8.25}$$

This iteration is expected to converge fast since effects of artificial scattering will be small in comparison to physical scattering.

8.2.2 Implicit second-order upwind scheme

At this point, we choose a finite number of ordinates and moments, i.e. the flux ψ is now a vector with dimension Q and the moments ϕ have finite dimension N. Consequently, operators applied to the directional space become matrices. For better readability, we abuse notation and reuse the same symbols as before.

We observed that a second-order spatial scheme is required to capture the behavior of the test cases used in this work. To ensure an efficient sweeping step, we use a second-order upwind stencil without a limiter. Let us denote the operator L discretized in space and direction by L_Δ. For ease of presentation, we assume a slab geometry. I.e. we have the spatial variable $x \in \mathbb{R}$ and the directional variable $\mu \in [-1, 1]$. In the following, we split the directional variable into $\mu_- \in [-1, 0]$ and $\mu_+ \in (0, 1]$. An extension to arbitrary dimension is straight forward. Now with $\lambda_\pm := \mu_\pm \frac{\Delta t}{\Delta x}$ and $\sigma_t := \sigma_a + \sigma_s + \sigma_{\text{as}} + \frac{1}{\Delta t}$, we can write the discretized streaming operator as

$$L_\Delta \psi := \lambda_\pm (g_{j+1/2} - g_{j-1/2}) + \Delta t \sigma_t \psi. \tag{8.26}$$

The numerical flux for μ_+ is then given by

$$g_{j+1/2} := a\psi_j + b\psi_{j-1}, \tag{8.27}$$

and for μ_- by

$$g_{j+1/2} := a\psi_{j+1} + b\psi_{j+2}, \qquad \text{with } a := \frac{3}{2},\ b := -\frac{1}{2}. \tag{8.28}$$

In the following, we show that the chosen numerical flux is L^2 stable. For simplicity, we consider the one-dimensional advection equation

$$\partial_t \psi + \mu_+ \partial_x \psi = 0 \tag{8.29}$$

with $\mu_+ \in \mathbb{R}_+$. A finite volume discretization is given by

$$\psi_j^{n+1} = \psi_j^n - \lambda_+ \left(g_{j+1/2} - g_{j-1/2} \right). \tag{8.30}$$

Let us check if this scheme dissipates the L^2 entropy $\eta(\psi) := \psi^2/2$ following the procedure in Section 1.2.2. To derive an update formula of the form (1.50) we multiply our scheme (8.30) with ψ_j^{n+1}, i.e. we obtain

$$\psi_j^{n+1}\psi_j^{n+1} = \psi_j^n\psi_j^{n+1} - \lambda_+ \left(g_{j+1/2} - g_{j-1/2}\right) \psi_j^{n+1}. \tag{8.31}$$

Now one needs to remove the cross term $\psi_j^n\psi_j^{n+1}$ which can be done by reversing the binomial formula

$$\psi_j^n\psi_j^{n+1} = \frac{1}{2}(\psi_j^{n+1})^2 + \frac{1}{2}\left(\psi_j^n\right)^2 - \frac{1}{2}(\psi_j^{n+1} - \psi_j^n)^2. \tag{8.32}$$

Plugging this formulation for the cross term into (8.31) and making use of the definition of the square entropy η gives

$$\eta(\psi_j^{n+1}) = \eta(\psi_j^n) - \frac{1}{2}(\psi_j^{n+1} - \psi_j^n)^2 - \lambda_+ \left(g_{j+1/2} - g_{j-1/2}\right) \psi_j^{n+1}. \tag{8.33}$$

Note that the second term on the right hand side is essentially the entropy dissipation term (1.54) of implicit Euler when using the square entropy. Now, in order to achieve entropy dissipation, i.e.

$$\sum_{j=1}^{N_x} \eta(\psi_j^{n+1}) \leq \sum_{j=1}^{N_x} \eta(\psi_j^n), \tag{8.34}$$

we need

$$\mathcal{E} = \sum_{j=1}^{N_x} \frac{1}{2}(\psi_j^{n+1} - \psi_j^n)^2 + \lambda_+ \sum_{j=1}^{N_x} \left(g_{j+1/2} - g_{j-1/2}\right) \psi_j^{n+1} \overset{!}{\geq} 0.$$

Since the first term of $\mathcal{E}$ is always positive, it remains to show that

$$\sum_{j=1}^{N_x} \left(g_{j+1/2} - g_{j-1/2}\right) \psi_j^{n+1} \overset{!}{\geq} 0.$$

Let us rewrite this term for all spatial cells as a matrix vector product. I.e. when collecting the solution at time step $n+1$ for all N_x spatial cells in a vector $\psi \in \mathbb{R}^{N_x}$, this term becomes

$$\sum_{j=1}^{N_x} \left(\left(g_{j+1/2} - g_{j-1/2}\right) \psi_j^{n+1}\right) = \psi^T B \psi \tag{8.35}$$

where $B \in \mathbb{R}^{N_x \times N_x}$ is a lower triangular matrix. This product can be symmetrized with $S := \frac{1}{2}(B + B^T)$, meaning that we have $\psi^T B\psi = \psi^T S\psi$. For our stencil, the matrix S has entries $s_{jj} = \frac{3}{2}$ on the diagonal and $s_{j,j-1} = s_{j-1,j} = -1$ as well as $s_{j,j-2} = s_{j-2,j} = \frac{1}{4}$ on the lower and upper diagonals. Positivity of $\psi^T S\psi$ and thereby of the entropy dissipation term $\mathcal{E}$ in (8.35), is guaranteed if S is positive definite, i.e. has positive eigenvalues. The eigenvalues for S have been computed numerically to verify positivity in Figure 8.14.

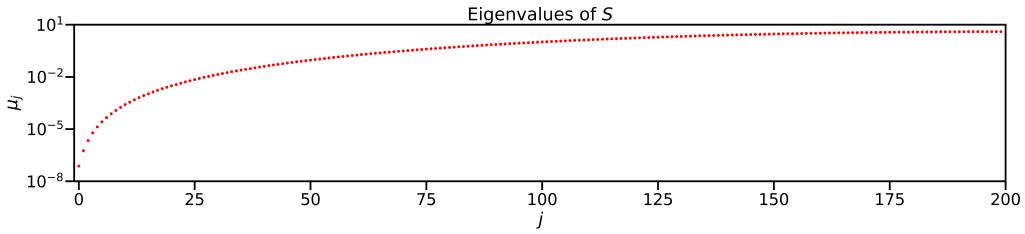


FIGURE 8.14: Eigenvalues of S for $N_x = 200$. All eigenvalues remain positive.

8.2.3 Implementation details

Let us now discuss the implementation of the implicit method in more detail. As mentioned earlier, a source iteration (8.25) is required to invert the operator $(L - \sigma_{\text{as}} S_{\text{as}}^{+})$. For an initial guess $\psi^{(0)}$ and an arbitrary right hand side R, this iteration is given by Algorithm 11. Note that the discrete artificial in-scattering S_{as}^{+} is a sparse matrix, which guarantees an efficient evaluation of the matrix vector product $S_{\text{as}}^{+}\psi^{(l)}$ in (8.25). Furthermore, the inverse of L_Δ can be computed by a sweeping procedure.

Algorithm 11 Source iteration algorithm

1: **procedure** SOURCEITERATION($\psi^{(0)}, R$)
2: $\quad \ell \leftarrow 0$
3: $\quad \psi^{(\ell+1)} \leftarrow L_\Delta^{-1}\left(\sigma_{\text{as}} S_{\text{as}}^{+}\psi^{(\ell)} + R\right)$
4: $\quad$ **while** $\|\psi^{(\ell+1)} - \psi^{(\ell)}\|_2 \geq \epsilon\tilde{c}$ **do**
5: $\quad\quad \psi^{(\ell+1)} \leftarrow L_\Delta^{-1}\left(\sigma_{\text{as}} S_{\text{as}}^{+}\psi^{(\ell)} + R\right)$
6: $\quad\quad \ell \leftarrow \ell + 1$
7: $\quad$ **return** $\psi^{(\ell)}$

In order to get a good error estimator, we set the constant $\tilde{c} := (1 - T)/T$ in Algorithm 11, where T is an estimate of the Lipschitz constant and ϵ is a user-determined parameter. Our implementation solves the linear system of equations

$$A\phi = b \tag{8.36a}$$

$$\text{with } A := I - \sigma_s M(L - \sigma_{\text{as}} S_{\text{as}}^{+})^{-1} O\Sigma \tag{8.36b}$$

$$b := M(L - \sigma_{\text{as}} S_{\text{as}}^{+})^{-1}\tilde{q} \tag{8.36c}$$

using a GMRES solver. The solver requires the evaluation of the left-hand side for a given ψ with an initial guess $\psi^{(0)}$, which is given by Algorithm 12.

Algorithm 12 Left-hand side of (8.36a)

1: **procedure** LHS($\psi^{(0)}, \phi$)
2: $\quad \tilde{\psi} \leftarrow$ SourceIteration$(\psi^{(0)}, \sigma_s O\Sigma\phi)$
3: $\quad$ **return** $\phi - M\tilde{\psi}$

The main time stepping scheme is then given by Algorithm 13. After initializing ψ and ϕ, the right hand side to (8.36a) is set up in line 4. Line 5 then solves the linear system (8.36a) and line 6 determines the time-updated flux ψ from the moments ϕ^{new}.

Algorithm 13 Sweeping-Krylov algorithm

$\psi^{\text{old}} \leftarrow \text{InitialCondition}()$
$\phi^{\text{old}} \leftarrow M\psi^{\text{old}}$
while $t < t_{\text{end}}$ **do**
  $b \leftarrow M \cdot \text{SourceIteration}(\psi^{old}, q + \frac{1}{\Delta t}\psi^{old})$
  $\phi^{\text{new}} \leftarrow \text{Krylov}(\text{LHS}(\psi^{\text{old}}, \phi^{\text{old}}), b)$
  $\psi^{\text{new}} \leftarrow \text{SourceIteration}(\psi^{\text{old}}, \sigma_s O\Sigma\phi^{\text{new}} + \frac{1}{\Delta t}\psi^{\text{old}})$
  $\psi^{\text{old}} \leftarrow \psi^{\text{new}}$
  $\phi^{\text{old}} \leftarrow \phi^{\text{new}}$

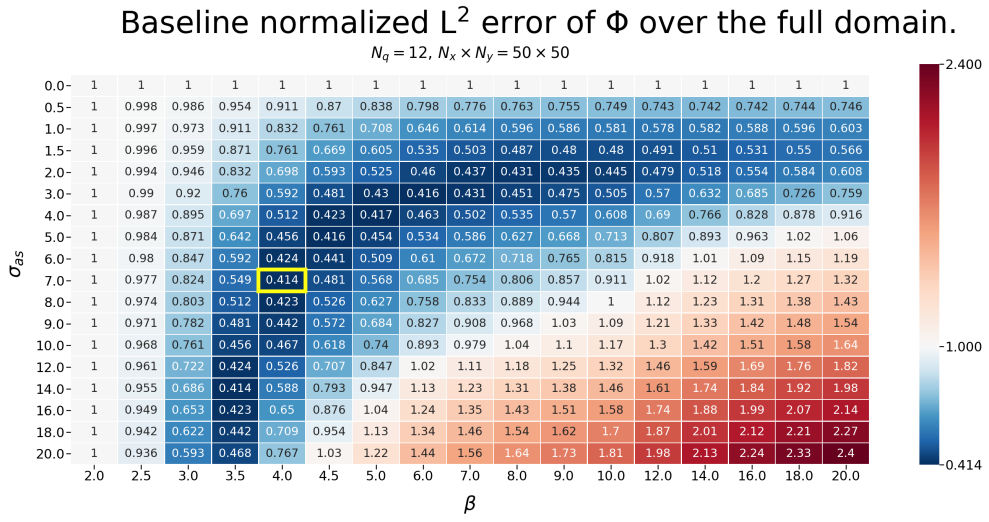
There exist several ways to modify the presented algorithm to achieve higher performance. For example, one can modify the presented method by not fully converging the source iteration in Algorithm 11. Instead, only a single iteration can be performed to drive the moments ϕ and the respective angular flux ψ to their corresponding fixed points simultaneously. In numerical tests, we observe that this will significantly speed up the calculation. However, since we do not focus on runtime optimization, we do not further discuss this idea and leave it to future work.

8.2.4 Results asS$_N$

Line-source test case

We start by picking suitable parameter values for the artificial scattering strength σ_{as} as well as the variance ε. For this, we conduct a parameter study for the line-source test case for a coarse spatial mesh and a small number of ordinates $Q \times N_x \times N_y = 12 \times 50 \times 50$. All computations are carried out with an implicit time discretization. For this and all proceeding computations, the tolerance for the GMRES solver was set to $1.5 \cdot 10^{-8}$, and we considered the inner source iteration to be converged at an estimated error of 10^{-4}. We make use of the icosahedron quadrature, see [36] for more details. The resulting relative L^2 error of the scalar flux $\Phi(t, \boldsymbol{x}) = \int_{S^2} \psi(t, \boldsymbol{x}, \boldsymbol{\Omega}')\, d\boldsymbol{\Omega}'$ at time $t_{\text{end}} = 1$ is depicted in Figure 8.15. Choosing the parameter values of $\beta = 4$ and $\sigma_{\text{as}} = 7$, we can decrease the error by 41.4% compared to the standard S$_N$ solution. The determined parameter values will be chosen from now on. To obtain a more detailed picture of how artifical scattering affects the numerical solution, we choose a finer grid resolution with $Q = 92$ and $N_x \times N_y = 200 \times 200$. For this setting, the scalar flux computed with S$_N$ is depicted in Figure 8.16, where we plot the scalar flux in the entire spatial domain on the left and the scalar flux along straight and circular cuts on the right. While the exact solution is non-osciallatory and even constant along circles, the S$_N$ solution again shows ray-effects. When adding artificial scattering to this S$_N$ computation, the ray-effects are significantly mitigated, see Figure 8.17.

Since the artificial scattering improves the solution accuracy, one can use a reduced number of ordinates while maintaining the same error level as S$_N$. Consequently the asS$_N$ method can be used to reduce runtime and in memory consumption. Consider therefore the results presented in Figure 8.18 and Figure 8.19. Both figures summarize the results for the line-source test case computed with the S$_N$ and asS$_N$ methods for different values of N. Figure 8.18 measures the error between the numerical solution and the analytical solution in the L^2 norm, called δ_1. Figure 8.19 considers the H^1 semi-norm. We observe an increase in runtime when activating artificial scattering, but a decrease in the errors δ_1 and δ_2. On the right, the errors are plotted against the number of ordinates which ultimately dictates the memory consumption. For example, an S$_8$ takes about as long as an as-S$_5$ computation and yields a similar δ_1 error. However, the



σ_{as} \ β	2.0	2.5	3.0	3.5	4.0	4.5	5.0	6.0	7.0	8.0	9.0	10.0	12.0	14.0	16.0	18.0	20.0
0.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0.5	1	0.998	0.986	0.954	0.911	0.87	0.838	0.798	0.776	0.763	0.755	0.749	0.743	0.742	0.742	0.744	0.746
1.0	1	0.997	0.973	0.911	0.832	0.761	0.708	0.646	0.614	0.596	0.586	0.581	0.578	0.582	0.588	0.596	0.603
1.5	1	0.996	0.959	0.871	0.761	0.669	0.605	0.535	0.503	0.487	0.48	0.48	0.491	0.51	0.531	0.55	0.566
2.0	1	0.994	0.946	0.832	0.698	0.593	0.525	0.46	0.437	0.431	0.435	0.445	0.479	0.518	0.554	0.584	0.608
3.0	1	0.99	0.92	0.76	0.592	0.481	0.43	0.416	0.431	0.451	0.475	0.505	0.57	0.632	0.685	0.726	0.759
4.0	1	0.987	0.895	0.697	0.512	0.423	0.417	0.463	0.502	0.535	0.57	0.608	0.69	0.766	0.828	0.878	0.916
5.0	1	0.984	0.871	0.642	0.456	0.416	0.454	0.534	0.586	0.627	0.668	0.713	0.807	0.893	0.963	1.02	1.06
6.0	1	0.98	0.847	0.592	0.424	0.441	0.509	0.61	0.672	0.718	0.765	0.815	0.918	1.01	1.09	1.15	1.19
7.0	1	0.977	0.824	0.549	0.414	0.481	0.568	0.685	0.754	0.806	0.857	0.911	1.02	1.12	1.2	1.27	1.32
8.0	1	0.974	0.803	0.512	0.423	0.526	0.627	0.758	0.833	0.889	0.944	1	1.12	1.23	1.31	1.38	1.43
9.0	1	0.971	0.782	0.481	0.442	0.572	0.684	0.827	0.908	0.968	1.03	1.09	1.21	1.33	1.42	1.48	1.54
10.0	1	0.968	0.761	0.456	0.467	0.618	0.74	0.893	0.979	1.04	1.1	1.17	1.3	1.42	1.51	1.58	1.64
12.0	1	0.961	0.722	0.424	0.526	0.707	0.847	1.02	1.11	1.18	1.25	1.32	1.46	1.59	1.69	1.76	1.82
14.0	1	0.955	0.686	0.414	0.588	0.793	0.947	1.13	1.23	1.31	1.38	1.46	1.61	1.74	1.84	1.92	1.98
16.0	1	0.949	0.653	0.423	0.65	0.876	1.04	1.24	1.35	1.43	1.51	1.58	1.74	1.88	1.99	2.07	2.14
18.0	1	0.942	0.622	0.442	0.709	0.954	1.13	1.34	1.46	1.54	1.62	1.7	1.87	2.01	2.12	2.21	2.27
20.0	1	0.936	0.593	0.468	0.767	1.03	1.22	1.44	1.56	1.64	1.73	1.81	1.98	2.13	2.24	2.33	2.4

FIGURE 8.15: Parameter study for σ_{as} and $\varepsilon = \beta/Q$ on a grid of $Q \times N_x \times N_y = 12 \times 50 \times 50$ in an implicit calculation. For every simulation we compute the L^2 error of the scalar flux Φ with respect to a semi-analytical reference solution on the same spatial grid. The number in each field of the heatmap is then the baseline normalized error, i.e. the L^2 error obtained for that specific parameter configuration divided by the error obtained without artificial scattering. Simulations are run until time $t_{end} = 1$ with a CFL number of 2.0. When choosing the optimal parameter values $\beta = 4$ and $\sigma_{as} = 7$ (highlighted in yellow) the error drops down to 41.4% of the original error without artificial scattering.

number of ordinates can be reduced from 492 to 162. For both, δ_1 and δ_2, the effect of artificial scattering vanishes in the limit of $Q \to \infty$.

Lattice test case

In the following, we will investigate the effects of artificial scattering for the lattice test case. Following the line-source problem, we set $\beta = 4.0$ and $\sigma_{as} = 7.0$. Our grid is chosen to be $Q = 92$, and $N_x \times N_y = 280 \times 280$ and again, we use a CFL number of 2. The numerical results for the scalar flux can be found in Figure 8.20. Since the chosen scheme is only L^2-stable, the solution becomes negative which is depicted by white regions in the solution. Again, we observe the ray-effect mitigation of the asS_N method, which also reduces negative solution regions. Figure 8.21 demonstrates the inherent advantage when performing implicit computations: We are able to use a very large CFL number, thus reducing the number of time steps and the overall computational costs drastically. Note that for the increased CFL numbers, the scheme preserves positivity.

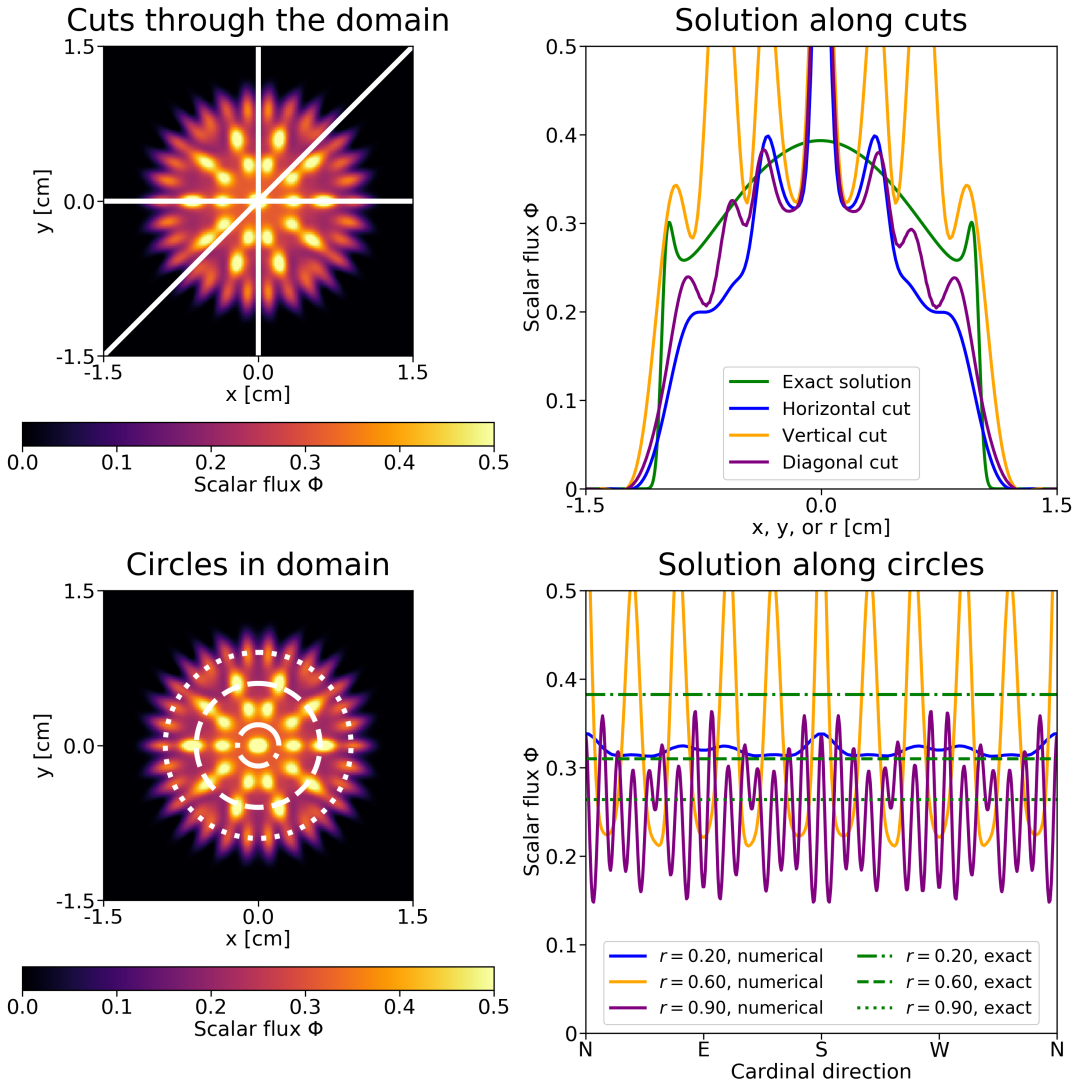


FIGURE 8.16: S_4 solution with ray-effects. We choose $Q = 92$ quadrature points, the spatial domain is composed of $N_x \times N_y = 200 \times 200$ spatial cells and the CFL number is 2. Cuts through the domain and along circles with different radii are visualized in the right column. Only the solution along the horizontal cut is symmetric for the icosahedron quadrature.

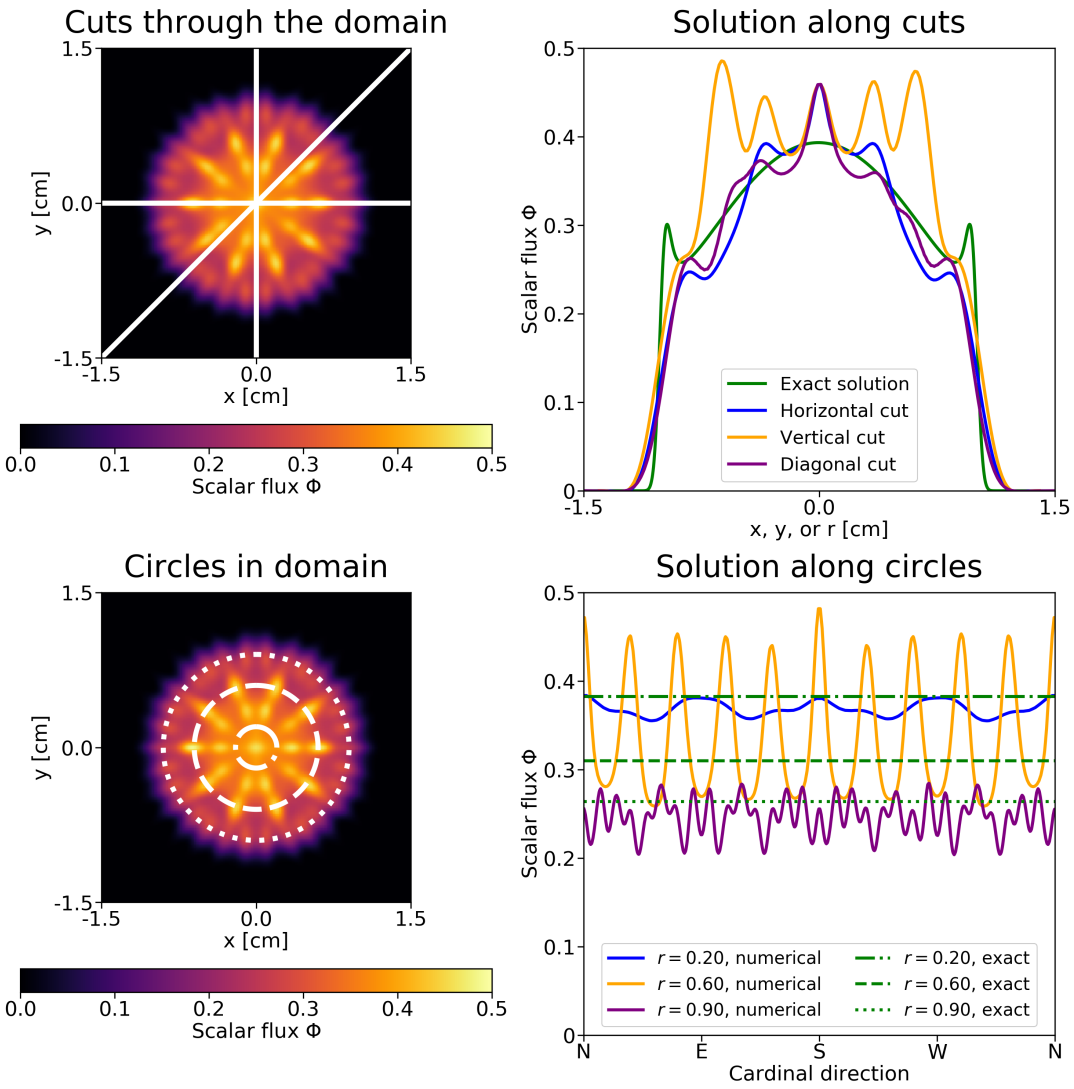


FIGURE 8.17: as-S_4 solution with mitigated ray-effects. We choose $Q = 92$ quadrature points, the spatial domain is composed of $N_x \times N_y = 200 \times 200$ spatial cells and the CFL number is 2. Cuts through the domain and along circles with different radii are visualized in the right column. We set $\sigma_{\text{as}} = 7$ and $\beta = 4$. Only the solution along the horizontal cut is symmetric for the icosahedron quadrature.

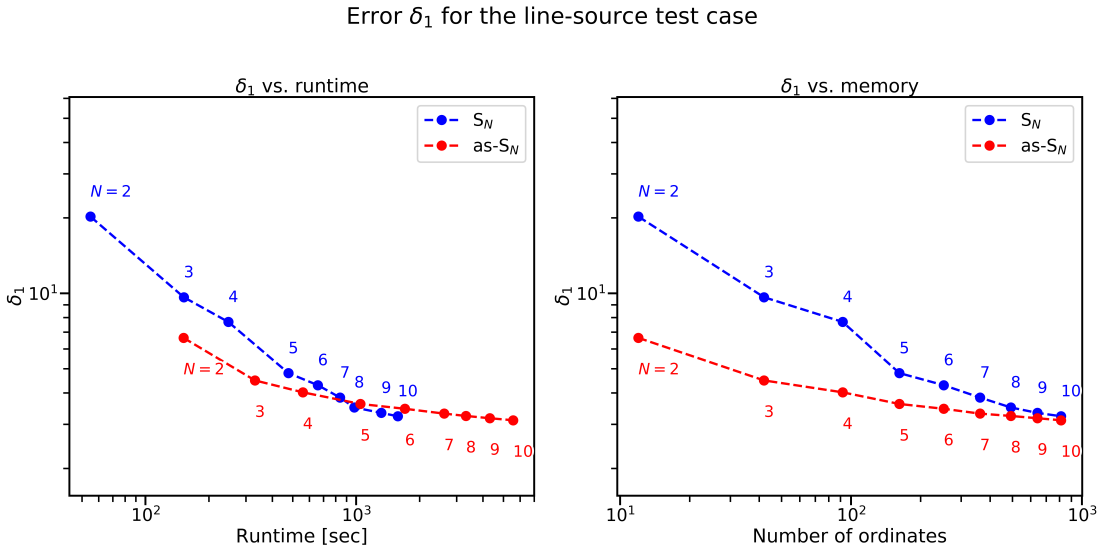


FIGURE 8.18: We computed $\delta_1 = \|\Phi_{\text{numerical}} - \Phi_{\text{analytical}}\|_2$ for the line-source test case using the implicit S$_N$ and as-S$_N$ method for $N_x \times N_y = 200 \times 200$. Computations were performed on a quad core Intel® i5-7300U CPU (2.60 GHz) with 12 GB memory.

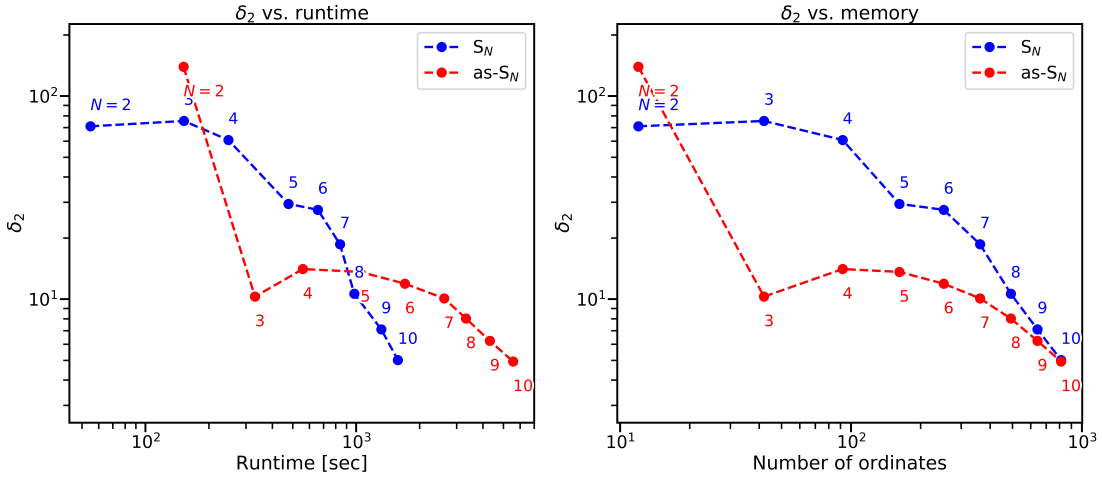
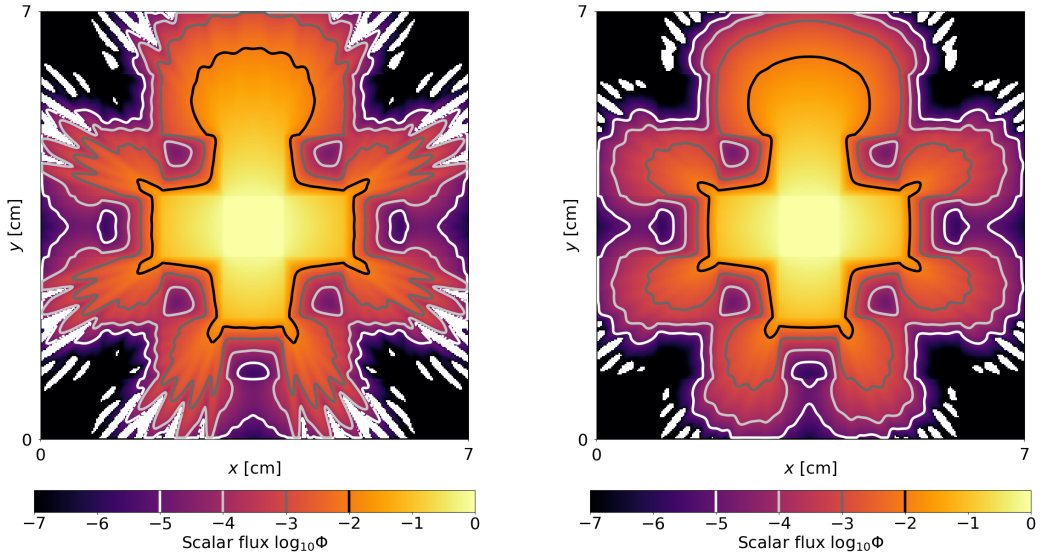


FIGURE 8.19: We computed $\delta_2 = \|\nabla\Phi_{\text{numerical}} - \nabla\Phi_{\text{analytical}}\|_2$ for the line-source test case using the implicit S$_N$ and as-S$_N$ method for $N_x \times N_y = 200 \times 200$. Computations were performed on a quad core Intel® i5-7300U CPU (2.60 GHz) with 12 GB memory.



(A) S_4. (B) as-S_4 with $\sigma_{as} = 7$ and $\beta = 4$.

FIGURE 8.20: Solutions to the lattice problem with an implicit computation for a CFL number of 2, $Q = 92$, and $N_x \times N_y = 280 \times 280$. The white regions indicate negativity of the solution.

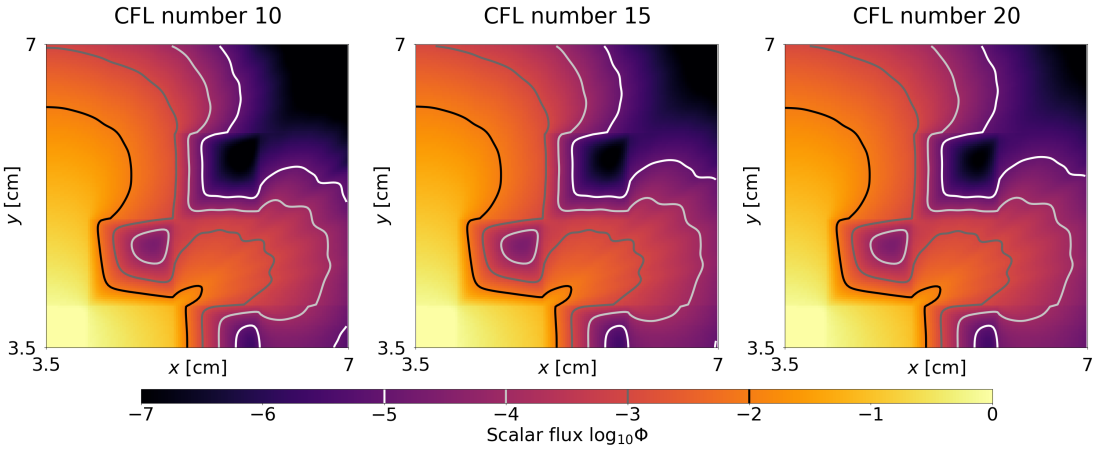


FIGURE 8.21: Solutions to the lattice problem with an implicit computation for different CFL numbers and $Q = 92$, $N_x \times N_y = 280 \times 280$, $\sigma_{as} = 7$, and $\beta = 4$. Zoom into the region $[3.5, 7] \times [3.5, 7]$.

Chapter 9

Minimal Entropy DG scheme

So far, we have discussed minimal entropy methods to discretize uncertainties as well as angular variables. In this section, we apply the concept of minimal entropy methods to discretize the spatial domain according to [9]. By proposing an entropy-consistent discretization of integral terms, we ensure the dissipation of a chosen entropy on a discrete level. This is a crucial property, since the IPM and M_N method only guarantee the dissipation of the chosen entropy on the continuous level and standard discretization techniques can potentially violate this dissipation property, especially when using higher-order methods.

We start by deriving a minimal entropy discontinuous Galerkin (DG) scheme in Section 9.1. Similar to the derivation of the IPM method in Section 1.4.5, the main idea is to derive a scheme, which works on the entropy- instead of the conserved variables. Section 9.2 discusses an implicit time discretization and in Section 9.3, we propose a discretization of the volume term, which preserves the entropy dissipation property. In Section 9.4, we study numerical results for different spatial resolutions.

9.1 Modal DG schemes for the entropy variables

We start by deriving the DG scheme for the entropy variables according to [9]. The spatial domain $D \subset \mathbb{R}$ is discretized into N_x non-overlapping elements $I_j = [x_j, x_{j+1}]$ and we use piece-wise polynomial basis functions from

$$\mathcal{V}^h := \left\{ w : w|_{I_j} \in \mathbb{P}_N(I_j), \text{supp}(w) = I_j \text{ with } j = 1, \cdots, N_x \right\} \tag{9.1}$$

to obtain a finite dimensional description of the solution in D. To simplify notation, we assume a scalar conservation law, which now only depends on space and time, i.e. we have

$$\partial_t u(t,x) + \partial_x f(u(t,x)) = 0. \tag{9.2}$$

As discussed in Section 1.1.3, one can rewrite this equation in terms of the entropy variable $v(u) = (s')^{-1}(u)$. Our aim is now to not expand the conserved variable u, but instead the entropy variable with the basis functions from (9.1), i.e. we get a piece-wise polynomial approximation of $v(u) = (s')^{-1}(u)$, which we denote by $v_h \in \mathcal{V}^h$. When replacing the exact entropy variable by this piece-wise polynomial approximation, we get

$$\partial_t u(v_h(t,x)) + \partial_x f(u(v_h(t,x))) = r(t,x), \tag{9.3}$$

where the residual r stems from the inexact piece-wise polynomial description of the solution. To remove the residual, one chooses spatial test functions $\tilde{\varphi}_{ij}(x) \in \mathcal{V}^h$, where i is the polynomial degree in spatial cell j. To shorten notation, we collect the test

functions with support in cell j in the vector $\tilde{\boldsymbol{\varphi}}_j \in \mathbb{R}^{N+1}$. Now multiplying (9.2) with $\tilde{\boldsymbol{\varphi}}_j$ and integrating over D gives

$$\frac{d}{dt}\int_{x_j}^{x_{j+1}} u(v_h(t,x))\tilde{\boldsymbol{\varphi}}_j\,dx + \int_{x_j}^{x_{j+1}} \partial_x g(v_h(t,x))\tilde{\boldsymbol{\varphi}}_j\,dx = 0,$$

where we again use $g(v) := f(u(v))$. Integration by parts gives

$$\frac{d}{dt}\int_{x_j}^{x_{j+1}} u(v_h(t,x))\tilde{\boldsymbol{\varphi}}_j\,dx + \left[g(v_h(t,x_{j+1}))\tilde{\boldsymbol{\varphi}}_j(x_{j+1}^-) - g(v_h(t,x_j))\tilde{\boldsymbol{\varphi}}_j(x_j^+)\right] - \int_{x_j}^{x_{j+1}} g(v_h(t,x))\tilde{\boldsymbol{\varphi}}_j'(x)\,dx = 0,$$

where x_j^+ denotes the left interface of cell j and x_j^- is the right interface of cell $j-1$. To approximate the physical flux at cell interfaces, we use numerical fluxes

$$g_{j+1/2} := f^*(u(v_h(t,x_{j+1}^-)), u(v_h(t,x_{j+1}^+))), \tag{9.4}$$

which gives

$$\frac{d}{dt}\int_{x_j}^{x_{j+1}} u(v_h)\tilde{\boldsymbol{\varphi}}_j\,dx = -\left[g_{j+1/2}\tilde{\boldsymbol{\varphi}}_j(x_{j+1}^-) - g_{j-1/2}\tilde{\boldsymbol{\varphi}}_j(x_j^+)\right] + \int_{x_j}^{x_{j+1}} g(v_h)\tilde{\boldsymbol{\varphi}}_j'(x)\,dx. \tag{9.5}$$

Note that we suppress the dependency on time t for sake of readability. Before moving to constructing a solver to determine the solution of (9.5), we would like to see if this discretization ensures entropy dissipation. For this, we apply the entropy dissipation strategy from Section 1.2.2 to our DG scheme. In cell j, the discretized entropy variable has the form $v_h = \sum_{i=0}^N v_i\tilde{\varphi}_{ij} = \boldsymbol{v}_j^T\tilde{\boldsymbol{\varphi}}$. Scalar multiplication of (9.5) with $\boldsymbol{v}_j$ yields

$$\frac{d}{dt}\int_{x_j}^{x_{j+1}} v_h\partial_t u(v_h)\,dx = \int_{x_j}^{x_{j+1}} s'(u_h)\partial_t u_h\,dx = \frac{d}{dt}\int_{x_j}^{x_{j+1}} s(u_h)\,dx$$

for the left-hand-side, where we use $u_h := (s')^{-1}(v_h)$. We then have

$$\frac{d}{dt}\int_{x_j}^{x_{j+1}} s(u_h)\,dx = -\left[v_h(x_{j+1}^-)g_{j+1/2} - v_h(x_j^+)g_{j-1/2}\right] + \int_{x_j}^{x_{j+1}} \underbrace{\partial_x v_h g(v_h)}_{=\partial_x v_h\psi'(v_h)=\partial_x\psi(v_h)}\,dx.$$

Hence

$$\begin{aligned}\frac{d}{dt}\int_{x_j}^{x_{j+1}} s(u_h)\,dx &= -\left[v_h(x_{j+1}^-)g_{j+1/2} - v_h(x_j^+)g_{j-1/2}\right] + \int_{x_j}^{x_{j+1}} \partial_x\psi(v_h)\,dx \\ &= -\left[v_h(x_{j+1}^-)g_{j+1/2} - v_h(x_j^+)g_{j-1/2}\right] + \psi(v_h(x_{j+1}^-)) - \psi(v_h(x_j^+)).\end{aligned}$$

Defining the numerical entropy flux

$$\mathcal{H}_{j+1/2}^{(\pm)} = v_h(x_{j+1}^\pm)g_{j+1/2} - \psi(v_h(x_{j+1}^\pm)), \tag{9.6}$$

we can rewrite this as

$$\frac{d}{dt}\int_{x_j}^{x_{j+1}} s(u_h)\,dx = -\mathcal{H}_{j+1/2}^{(-)} + \mathcal{H}_{j-1/2}^{(+)}.$$

Note that this scheme is not conservative since the flux in cell j at interface $j+1$ which is $\mathcal{H}^{(-)}_{j+1/2}$ does not equal the flux in cell $j+1$ at interface $j+1$ which is $\mathcal{H}^{(+)}_{j+1/2}$. Hence summing over all cells will not cancel fluxes. To split this flux into a conservative and non-conservative part, we add and substract

$$-\frac{1}{2}v_h(x^{\pm}_{j+1})g_{j+1/2}+\frac{1}{2}v_h(x^{\mp}_{j+1})g_{j+1/2}+\frac{1}{2}\left(\psi(v_h(x^{\pm}_{j+1}))-\psi(v_h(x^{\mp}_{j+1}))\right)$$

in the flux term (9.6). Hence we can split the numerical entropy flux $\mathcal{H}^{(\pm)}_{j+1/2}$ into a conserved and non-conserved part

$$\mathcal{H}^{(\pm)}_{j+1/2}=\bar{F}(v_h(x^-_{j+1}),v_h(x^+_{j+1}))\mp D(v_h(x^-_{j+1}),v_h(x^+_{j+1})),$$

where

$$\begin{aligned}\bar{F}(v_h(x^-_{j+1}),v_h(x^+_{j+1}))=&\frac{1}{2}\left[v_h(x^-_{j+1})+v_h(x^+_{j+1})\right]g_{j+1/2}-\frac{1}{2}\left(\psi(v_h(x^-_{j+1}))+\psi(v_h(x^+_{j+1}))\right),\\ D(v_h(x^-_{j+1}),v_h(x^+_{j+1}))=&\frac{1}{2}\left(v_h(x^-_{j+1})-v_h(x^+_{j+1})\right)g_{j+1/2}+\frac{1}{2}\left(\psi(v_h(x^+_{j+1}))-\psi(v_h(x^-_{j+1}))\right).\end{aligned}$$

In order to obtain a cell entropy inequality, we must enforce $D>0$, which gives the e-scheme condition

$$\left(v_h(x^-_{j+1})-v_h(x^+_{j+1})\right)g_{j+1/2}>\psi(v_h(x^-_{j+1}))-\psi(v_h(x^+_{j+1})),$$

which resembles the e-scheme condition from FV schemes (1.49). Note that in FV schemes, we can construct numerical fluxes such as Lax-Friedrichs s.t. $D>0$ for all convex entropies. In the DG context, we need to pick one entropy for our solution discretization $u(v_h)$. The e-scheme property now only ensures dissipation for this chosen entropy.

9.2 Time discretization

Until now, we kept the time continuous. In this section we show that entropy dissipation is not destroyed by implicit schemes. The almost (since we keep the evaluation of integrals continuous for now) discrete implicit scheme reads

$$\begin{aligned}\int_{x_j}^{x_{j+1}}u(v_h(t^{n+1},x))\tilde{\varphi}_j\,dx=&\int_{x_j}^{x_{j+1}}u(v_h(t^n,x))\tilde{\varphi}_j\,dx\\ &-\Delta t\left[g^{n+1}_{j+1/2}\tilde{\varphi}_j(x^-_{j+1})-g^{n+1}_{j-1/2}\tilde{\varphi}_j(x^+_j)\right]+\Delta t\int_{x_j}^{x_{j+1}}g(v_h(t^{n+1},x))\tilde{\varphi}_j'\,dx.\end{aligned}\tag{9.7}$$

Here we use an implicit Euler step to update the solution in time. The notation $g^{n+1}_{j+1/2}$ denotes an evaluation of the numerical flux (9.4) at time step $n+1$. Scalar multiplication

with $\boldsymbol{v}_j^{n+1}$ gives

$$\begin{aligned}\int_{x_j}^{x_{j+1}} u(v_h(t^{n+1},x))v_h(t^{n+1},x)\,dx = &\int_{x_j}^{x_{j+1}} u(v_h(t^n,x))v_h(t^{n+1},x)\,dx\\ &-\Delta t\bar{F}(v_h(t^{n+1},x_{j+1}^-),v_h(t^{n+1},x_{j+1}^+))\\ &+\Delta t\bar{F}(v_h(t^{n+1},x_{j-1}^-),v_h(t^{n+1},x_{j-1}^+))\\ &+\Delta t D(v_h(t^{n+1},x_{j+1}^-),v_h(t^{n+1},x_{j+1}^+))\\ &+\Delta t D(v_h(t^{n+1},x_{j-1}^-),v_h(t^{n+1},x_{j-1}^+)),\end{aligned}\tag{9.8}$$

where we reused the results from the previous section. Now we wish to find an expression for the first two terms. To simplify notation we use $V(v) := s(u(v))$. As before we use the identity

$$V(v_h(t^{n+1},x)) - V(v_h(t^n,x)) = \int_{-1/2}^{1/2} \frac{d}{d\beta} V(v_h^{n+1/2}(\beta,x))\,d\beta,$$

where

$$\begin{aligned} v_h^{n+1/2}(\beta,x) &:= v_h(t^{n+1},x) + (\beta-1/2)\Delta v_h^{n+1/2}(x),\\ \text{with}\quad \Delta v_h^{n+1/2} &:= v_h(t^{n+1},x) - v_h(t^n,x).\end{aligned}$$

Note that v_h is only defined in the spatial cell j. From here, we can show that

$$\begin{aligned}\int_{-1/2}^{1/2} \frac{d}{d\beta} s(u(v_h^{n+1/2}(\beta)))\,d\beta &= \int s'(u(v_h^{n+1/2}))u'(v_h^{n+1/2})\frac{dv_h^{n+1/2}}{d\beta}\,d\beta\\ &= \int_{-1/2}^{1/2} v_h^{n+1/2}u'(v_h^{n+1/2})\Delta v_h^{n+1/2}\,d\beta.\end{aligned}$$

Using the definition of $v_h^{n+1/2}$ we get

$$\begin{aligned}\int_{-1/2}^{1/2} v_h^{n+1/2}u'(v_h^{n+1/2})\Delta v_h^{n+1/2}\,d\beta =& v_h(t^{n+1},x)\int u'(v_h^{n+1/2})\,d\beta\Delta v_h^{n+1/2}\\ &+\int(\beta-1/2)\Delta v_h^{n+1/2}u'(v_h^{n+1/2})\Delta v_h^{n+1/2}\,d\beta.\end{aligned}$$

Substituting $\eta = v_h^{n+1/2}(\beta)$ with $\frac{d\eta}{d\beta} = \Delta v_h^{n+1/2}$ into the first term yields

$$\begin{aligned}v_h(t^{n+1},x)\int u'(v_h^{n+1/2})\,d\beta\Delta v_h^{n+1/2} =& v_h(t^{n+1},x)\int u'(\eta)\frac{1}{\Delta v_h^{n+1/2}}d\eta\Delta v_h^{n+1/2}\\ =& v_h(t^{n+1},x)\left[u(v_h(t^{n+1},x)) - u(v_h(t^n,x))\right].\end{aligned}$$

Hence, we get

$$s(u(v_h(t^{n+1},x))) - s(u(v_h(t^n,x))) = v_h(t^{n+1},x)\left[u(v_h(t^{n+1},x)) - u(v_h(t^n,x))\right] + \mathcal{E}_j(x)$$

with the dissipation term

$$\mathcal{E}_j(x) := \int(\beta-1/2)\Delta v_h^{n+1/2}u'(v_h^{n+1/2})\Delta v_h^{n+1/2}\,d\beta > 0.$$

Integration w.r.t. x over the spatial cell j finally yields

$$\int_{x_j}^{x_{j+1}} u(v_h(t^{n+1},x))v_h(t^{n+1},x)\,dx - \int_{x_j}^{x_{j+1}} u(v_h(t^n,x))v_h(t^{n+1},x)\,dx$$
$$= \int_{x_j}^{x_{j+1}} s(u(v_h(t^{n+1},x)))\,dx - \int_{x_j}^{x_{j+1}} s(u(v_h(t^n,x)))\,dx - \int_{x_j}^{x_{j+1}} \mathcal{E}_j(x)\,dx.$$

Plugging this into (9.8) gives

$$\begin{aligned}\int_{x_j}^{x_{j+1}} s(u(v_h(t^{n+1},x)))\,dx = \int_{x_j}^{x_{j+1}} s(u(v_h(t^n,x)))\,dx + \int_{x_j}^{x_{j+1}} \mathcal{E}_j(x)\,dx \\ - \Delta t\bar{F}(v_h(t^{n+1},x_{j+1}^-),v_h(t^{n+1},x_{j+1}^+)) \\ + \Delta t\bar{F}(v_h(t^{n+1},x_{j-1}^-),v_h(t^{n+1},x_{j-1}^+)) \\ + \Delta tD(v_h(t^{n+1},x_{j+1}^-),v_h(t^{n+1},x_{j+1}^+)) \\ + \Delta tD(v_h(t^{n+1},x_{j-1}^-),v_h(t^{n+1},x_{j-1}^+)).\end{aligned}$$

Due to positivity of $\mathcal{E}_j$, the backward Euler step adds entropy dissipation to the dissipation of the space discretization. A forward Euler step yields an entropy production term and can therefore destroy the desired dissipation property. One could try to derive a modification or CFL condition for which the dissipation introduced by the spatial discretization surpasses the entropy production of the explicit time update.

9.3 Fully discrete scheme

Taking a closer look at the DG update (9.7), we see that three integral terms still need to be approximated. For the volume term

$$\int_{x_j}^{x_{j+1}} g(v_h(t^{n+1},x))\tilde{\varphi}_j'(x)dx,$$

we cannot simply use a Gauss quadrature rule, as this quadrature does not fulfill

$$\int \partial_x\psi(v_h(t,x))\,dx = \psi(v_h(x_{j+1}^-)) - \psi(v_h(x_j^+)),$$

i.e. entropy dissipation will be violated by the error of the numerical integration. To preserve this property on the discrete level, we use a summed mid-point rule

$$\int_{x_j}^{x_{j+1}} f(x)\,dx \approx \sum_{k=1}^{N_q} w_k f(\hat{x}_k^{(j)})$$

with $w_k = \Delta x/N_q$ and $\hat{x}_k^{(j)} = x_j + w_k(k-1)$. On this midpoint-grid, we can define a finite difference approximation

$$f'(x) \approx \frac{f(\hat{x}_k^{(j)}) - f(\hat{x}_{k+1}^{(j)})}{\Delta x},$$

We now replace the finite volume term by

$$\int_{x_j}^{x_{j+1}} g(v_h(t^{n+1},x))\tilde{\varphi}_j' dx \approx \sum_{k=1}^{N_q} w_k g_S\left(v_h(t^{n+1},\hat{x}_k^{(j)}), v_h(t^{n+1},\hat{x}_{k+1}^{(j)})\right)\frac{\tilde{\varphi}_j(\hat{x}_{k+1}^{(j)})-\tilde{\varphi}_j(\hat{x}_k^{(j)})}{\hat{x}_{k+1}^{(j)}-\hat{x}_k^{(j)}},$$

where we use the entropy conservative flux

$$g_S(v_1,v_2) := \begin{cases} \frac{\psi(v_1)-\psi(v_2)}{v_1-v_2} & \text{if } v_1 \neq v_2 \\ f(u(v_1)) & \text{else} \end{cases}.$$

Note that this is basically a finite difference approximation of $\psi'(v_1) \equiv f(u(v_1))$[1]. Now, scalar multiplication with $\boldsymbol{v}_j^{n+1}$ gives (dropping the time dependency for ease of exposition)

$$\begin{aligned}
&\boldsymbol{v}_j^T \sum_{k=1}^{N_q} w_k g_S\left(v_h(\hat{x}_k^{(j)}), v_h(\hat{x}_{k+1}^{(j)})\right)\frac{\tilde{\varphi}_j(\hat{x}_{k+1}^{(j)})-\tilde{\varphi}_j(\hat{x}_k^{(j)})}{\hat{x}_{k+1}^{(j)}-\hat{x}_k^{(j)}} \\
&= \sum_{k=1}^{N_q} w_k g_S\left(v_h(\hat{x}_k^{(j)}), v_h(\hat{x}_{k+1}^{(j)})\right)\frac{v_h(\hat{x}_{k+1}^{(j)})-v_h(\hat{x}_k^{(j)})}{\hat{x}_{k+1}^{(j)}-\hat{x}_k^{(j)}} \\
&= \sum_{k=1}^{N_q} w_k \frac{\psi(v_h(\hat{x}_{k+1}^{(j)}))-\psi(v_h(\hat{x}_k^{(j)}))}{\hat{x}_{k+1}^{(j)}-\hat{x}_k^{(j)}} \\
&= \sum_{k=1}^{N_q} \psi(v_h(\hat{x}_{k+1}^{(j)}))-\psi(v_h(\hat{x}_k^{(j)})) = \psi(v_h(x_{j+1}^-))-\psi(v_h(x_j^+)).
\end{aligned}$$

Here, we assumed that $v_h(\hat{x}_{k+1}^{(j)}) \neq v_h(\hat{x}_k^{(j)})$. If this is not the case, these terms cancel out of the sum and we recover the same result. Hence, the entropy dissipation is the same as the one derived when assuming exact computation of integrals.

The remaining two integrals of the scheme (9.7) can be approximated with some standard quadrature rule (e.g. Gauss) with weights $\tilde{w}_k$ and points $\tilde{x}_k^{(j)}$. The fully discrete scheme is now

$$\begin{aligned}
\sum_{k=1}^{N_q} \tilde{w}_k u(v_h(t^{n+1},\tilde{x}_k^{(j)}))\tilde{\varphi}_j(\tilde{x}_k^{(j)}) &= \sum_{k=1}^{N_q} \tilde{w}_k u(v_h(t^{n},\tilde{x}_k^{(j)}))\tilde{\varphi}_j(\tilde{x}_k^{(j)}) \\
&- \Delta t\left[g_{j+1/2}^{n+1}\tilde{\varphi}_j(x_{j+1}^-) - g_{j-1/2}^{n+1}\tilde{\varphi}_j(x_j^+)\right] \\
&+ \Delta t \sum_{k=1}^{N_q} w_k g_S\left(v_h(t^{n+1},\hat{x}_k^{(j)}), v_h(t^{n+1},\hat{x}_{k+1}^{(j)})\right)\frac{\tilde{\varphi}_j(\hat{x}_{k+1}^{(j)})-\tilde{\varphi}_j(\hat{x}_k^{(j)})}{\hat{x}_{k+1}^{(j)}-\hat{x}_k^{(j)}}.
\end{aligned} \tag{9.9}$$

[1] Or at some intermediate value, for example $0.5(v_1+v_2)$

In order to time update the entropy variables, we need to determine the root of

$$\mathcal{G}((\boldsymbol{v}_j)_{j\in\mathbb{Z}}) := \sum_{k=1}^{N_q} \tilde{w}_k u(v_h(t^{n+1}, \tilde{x}_k^{(j)}))\tilde{\boldsymbol{\varphi}}_j(\tilde{x}_k^{(j)}) - \sum_{k=1}^{N_q} \tilde{w}_k u(v_h(t^n, \tilde{x}_k^{(j)}))\tilde{\boldsymbol{\varphi}}_j(\tilde{x}_k^{(j)}) \tag{9.10}$$

$$- \Delta t \left[g_{j+1/2}^{n+1}\tilde{\boldsymbol{\varphi}}_j(x_{j+1}^-) - g_{j-1/2}^{n+1}\tilde{\boldsymbol{\varphi}}_j(x_j^+) \right]$$

$$+ \Delta t \sum_{k=1}^{N_q} w_k g_S \left(v_h(t^{n+1}, \hat{x}_k^{(j)}), v_h(t^{n+1}, \hat{x}_{k+1}^{(j)}) \right) \frac{\tilde{\boldsymbol{\varphi}}_j(\hat{x}_{k+1}^{(j)}) - \tilde{\boldsymbol{\varphi}}_j(\hat{x}_k^{(j)})}{\hat{x}_{k+1}^{(j)} - \hat{x}_k^{(j)}} \tag{9.11}$$

The numerical implementation uses Newton's method with the analytically derived Jacobi matrix of this function to perform time updates. In the following, we will refer to this method as *minimal entropy DG* (MEDG) method.

9.4 Results

In the following, we again investigate the Burgers' equation as done in Chapters 2 and 3, however we omit uncertainties. Then, we have to solve

$$\partial_t u(t,x) + \partial_x f(u(t,x)) = 0,$$
$$u(t=0,x) = u_{\mathrm{IC}}(x).$$

The initial condition is again a forming shock, which is now deterministic, i.e. we have

$$u_{\mathrm{IC}}(x) := \begin{cases} u_L, & \text{if } x < x_0 \\ u_L + \frac{u_R - u_L}{x_0 - x_1}(x_0 - x), & \text{if } x \in [x_0, x_1] \\ u_R, & \text{else} \end{cases}.$$

Furthermore, we use Dirichlet boundary conditions with function value u_L on the left and u_R on the right boundary. Again, our scheme should fulfill a maximum–principle (1.25b), which we wish to ensure by choosing the bounded–barrier entropy (2.38) for our MEDG scheme. The following parameters are used in the computation

$[a,b] = [0,3]$	range of spatial domain
$N_x = 9$	number of spatial cells
$t_{end} = 0.12$	end time
$x_0 = 0.5, x_1 = 1.5, u_L = 12, u_R = 1$	parameters of initial condition
$N+1 = 2$	number of moments
$\tau = 10^{-6}$	gradient tolerance for Newton's method
$\Delta u = 0$	distance u_{IC} to bounded–barrier bounds

Note, that we are using a small number of unknowns $N_x \cdot (N+1) = 18$ to represent the solution. Now it remains to pick a suitable CFL condition, which controls the time step, i.e. we have

$$\max_u \left| f'(u) \right| \frac{\Delta t}{\Delta x} = \mathrm{CFL}.$$

Since we make use of an implicit scheme, we could use a larger CFL number than one. However, we do not use the implicit discretization to allow for bigger time steps, but to ensure a discrete entropy dissipation. Indeed, for this test case, it can be seen that a big CFL condition will smear out the solution, leading to an unsatisfactory solution approximation. The solution obtained with the previously discussed MEDG scheme

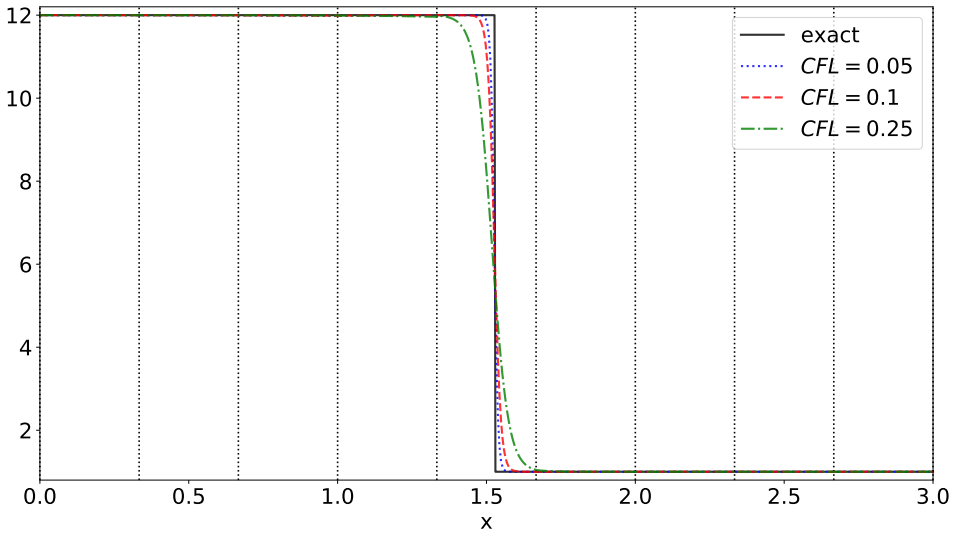


FIGURE 9.1: Minimal entropy DG result with $N_x = 9$, $N = 1$ using different CFL numbers. Spatial cells are indicated by dotted vertical lines.

for different CFL numbers can be found in Figure 9.1. One observes that the scheme captures the exact shock solution nicely with only a small number of unknowns, especially if the CFL number is chosen to be small. The solution accuracy can further be improved by increasing the number of spatial cells to $N_x = 19$, which yields the results in Figure 9.2. When further increasing the number of spatial cells to $N_x = 50$, bigger CFL numbers can be chosen to yield a satisfactory approximation result, see Figure 9.3. Note that here, we investigated CFL numbers up to 2.

Even though these results are in favor of the proposed MEDG scheme one should mention several shortcomings of this method:

- So far, we have looked at scalar problems, for which the bounded–barrier entropy seems to be an adequate choice. However, it is not clear how the method behaves for systems, and we already experienced oscillations for systems even with admissible entropies in the UQ context, see for example Section 4.5.1.

- The stability that the MEDG scheme possesses is the previously derived entropy dissipation. However, a stronger stability result, namely a TVD property in the mean is needed to guarantee convergence for DG methods. Such a result is usually obtained with limiters and one has to investigate how such limiters affect the entropy dissipation.

- It can happen that the Jacobian of (9.10) becomes ill conditioned in which case the numerical solver fails. In our numerical experiments, the method is robust when only two moments are used, however when increasing the number of moments, ill conditioned Jacobians appear frequently.

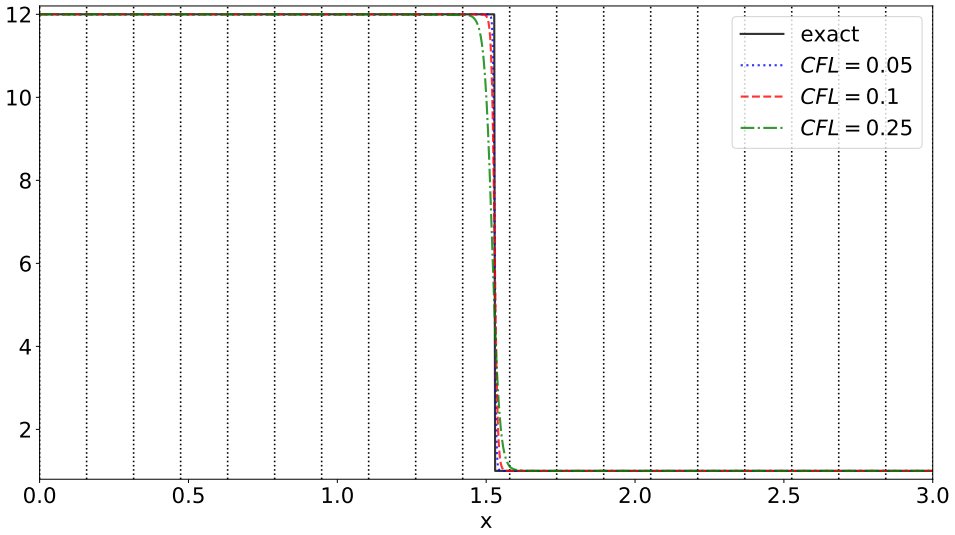


FIGURE 9.2: Minimal entropy DG result with $N_x = 19$, $N = 1$ using different CFL numbers. Spatial cells are indicated by dotted vertical lines.

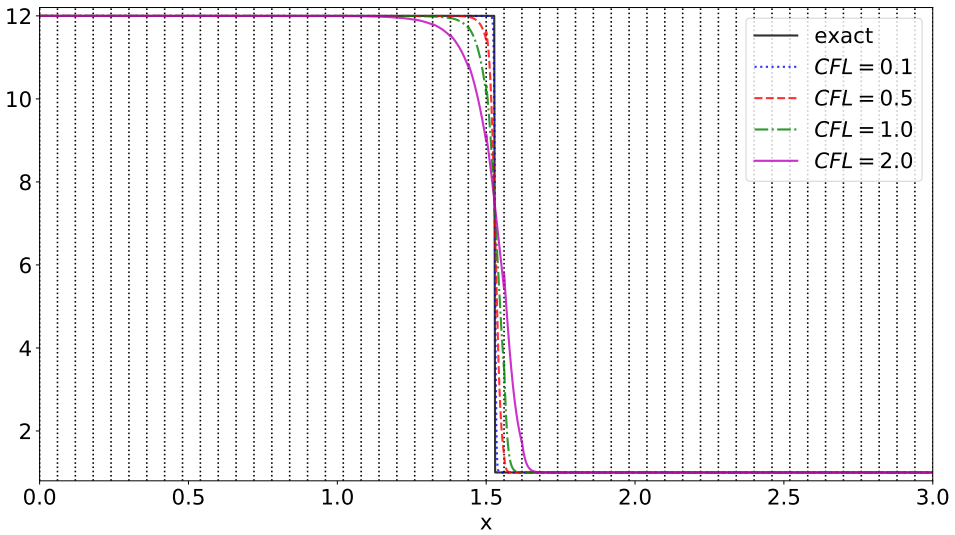


FIGURE 9.3: Minimal entropy DG result with $N_x = 50$, $N = 1$ using different CFL numbers. Spatial cells are indicated by dotted vertical lines.

Chapter 10

Summary and Outlook

10.1 Summary

In this thesis, we have investigated different discretization strategies, which maintain realizability (i.e. which fulfill certain solution bounds) while mitigating spurious oscillations and numerical artifacts. We started by considering phase spaces which, in addition to time and space, include a random dimension. When using a modal representation for the random dimension, the solution starts to oscillate while violating realizability. To guarantee a maximum–principle for scalar hyperbolic equations with uncertainites, we have studied the IPM method, which can show non-realizable moments, i.e. moments which do not belong to an admissible solution, due to errors in the optimization step. Consequently the numerical solver for the IPM system crashes, since the solution ansatz is in this case undefined. Therefore we constructed a robust, second-order accurate numerical solver, which maintains realizability in Chapter 2. Here, we investigated monotonicity of the underlying scheme for the original PDE, yielding two approaches to preserve realizability, namely a modification of the CFL condition and a recalculation step for the moment vector. We also investigated the approximation properties of the IPM scheme using different entropies. By considering the entropy ansatz directly, we showed that the solution is not bounded due to properties of the entropy density s itself but rather its derivative s'. This allowed us to use an entropy, which we called the bounded-barrier entropy, that takes finite values at the bounds u_- and u_+. The bounded-barrier entropy behaves more gracefully near the boundary values u_- and u_+, which we showed also leads to better solutions at intermediate values. This allowed us to take the IPM bounds to be the minimal and maximal value of the true solution, thus allowing the method to fulfill the exact maximum principle of the underlying PDE.

After having guaranteed realizability of the numerical IPM solver, it remains to cope with the numerical costs of the IPM method, which stem from the need to solve an optimization problem in every time step at every spatial cell. Therefore, we proposed to use certain basis vectors to represent the IPM solution on the chosen set of quadrature points in Chapter 3. By splitting the basis into vectors which span the kernel of the moment matrix, we could reduce the number of unknowns needed for the IPM optimization problem when the number of quadrature points lies close to the number of moments. This method facilitates the use of adaptivity, which we used to switch between numerically cheap collocation and expensive IPM updates. The IPM updates are used in regions which show a complex structure in the uncertainty, whereas collocation updates are used in the remainder.

In Chapter 4, we extended the realizability preserving scheme as well as adaptivity to systems. Furthermore, we proposed two additional acceleration techniques for steady problems, which speed up intrusive calculations. The first technique iterates the dual variables and moment vectors to their steady state simultaneously instead of

solving the IPM optimization problem in every time step. We could show that this scheme preserves the local convergence of the classical IPM algorithm. Furthermore, we proposed to run the first iterations to the steady state solution with a lower order method and successively increase the maximal number of moments when approaching the correct steady state solution. The effects of the proposed techniques have been demonstrated by comparing results obtained with IPM as well as SG against SC for a NACA test case. Here, the intrusive methods yield the same error level as SC for a reduced runtime, especially since intrusive methods require less unknowns to achieve a certain accuracy. In higher-dimensional problems, this effect is amplified since the number of unknowns to achieve a certain total degree is asymptotically smaller than the number of quadrature points in a tensorized or sparse grid. Furthermore, we could observe that the required residual at which the solution of intrusive methods reaches a steady state is smaller than for SC. Additionally, the ability to adaptively change the truncation order helps intrusive methods to compete with SC in terms of computational runtime.

So far, we mitigated non-physical solution artifacts by sufficiently increasing the accuracy of our method. In Chapter 5, we proposed to use filters, which allow for a non-oscillatory solution approximation even when using a smaller number of moments to represent the solution. A main challenge of filters for modal discretizations is to pick an adequate filter strength, which sufficiently mitigates unwanted artifact while maintaining a certain accuracy. We tackled this issue by proposing a filter based on Lasso regression. This filter picks its filter strength automatically for every individual spatial cell at every time step, i.e. it uses an adaptive filter strength, which is turned up in regions which are prone to spurious oscillations. In Chapter 6, we extended the idea of filters from SG to IPM methods. As seen before, numerical schemes for the IPM system require realizability and it turns out that standard filters violate this property. To obtain a robust IPM method with filters, we proposed two different techniques: First, we derived a filter, which is applied to the underlying kinetic picture of the given set of equations. Furthermore, we proposed to apply a regularization strategy to IPM, which allows using non-realizable moments, meaning that standard filters can be applied to the moment vector.

In Chapter 7, we further extended the phase space by investigating kinetic equations which include uncertainties, i.e. the phase space consists of an angular as well as an uncertain dimension, which we discretized with a modal representation. The kinetic equations considered are the thermal radiative transport equations, which describe the movement of particles through a background material, which can heat up by absorbing particles, leading to the emission of new particles. Here, we proposed to use the P_N method to discretize the angular dependency as well as an IPM discretization of the uncertain domain, which guarantees positivity of the material energy. We applied previously proposed acceleration techniques for IPM, namely adaptivity and the One-Shot approach to reduce the run time of the method.

Since the P_N method does not preserve positivity of the angular flux for the test-cases considered in the remainder, we move to nodal S_N discretizations, which preserve realizability at the cost of yielding ray-effects. In Chapter 8, we studied two techniques to mitigate ray-effects. First, we applied a rotation of the chosen quadrature set, which heavily increases the number of direction in which particles can travel, thereby mitigating ray-effects. By performing a modified equation analysis, we concluded that the effects of the rotation step yield a diffusive term for the angular components. Furthermore, we added an artificial scattering term to the S_N equations, which can be understood as filtering applied to the nodal discretization. Here, we proposed an implicit

time discretization, which treats the artificial scattering operator with a source iteration, thereby decreasing the number of moments needed for the Krylov solver, which is commonly applied for implicit discretizations. We showed that both methods mitigate ray-effects while yielding satisfactory solution approximations.

In Chapter 9, we derived a DG scheme, which dissipates a chosen entropy. This goal is achieved by using the main idea of the IPM method to discretize the spatial domain. To preserve the entropy dissipation property, we chose an implicit time discretization as well as a special treatment of the volume term inside the DG scheme. We can show that the resulting scheme preserves a maximum–principle, while gracefully representing the solution even with a small number of unknowns.

10.2 Outlook

There are still several open questions and tasks that should be performed in the future in order to fully understand the applicability of the IPM method and intrusive methods for uncertainty quantification in general. We have shown that the use of adaptivity is a key advantage of intrusive methods. Furthermore, the increased accuracy of intrusive methods compared to Stochastic Collocation enables one to pick a smaller number of unknowns to achieve a given accuracy. We believe that these two factors heavily weigh in when studying high dimensional problems for finely resolved, three dimensional meshes. However, a thorough comparison between intrusive methods and Stochastic Collocation has only been conducted for uncertainties up to dimension two for two-dimensional spatial meshes, which is why one should extend this investigation to more complex problems.

One characteristic, which we observed during our calculations for the NACA test-case is that Stochastic Collocation results tend to show numerical artifacts even for a large number of collocation points when the mesh resolution is too coarse, while intrusive methods with significantly less unknowns led to physically correct smooth solution approximations on the same spatial mesh. This observation could stem from the fact that Stochastic Collocation relies on a series of deterministic flow solutions, which tend to yield sharp discontinuities in the spatial domain and therefore require a finely resolved computational mesh as well as shock capturing techniques. Intrusive methods, which directly work on the moments and not the flow solution itself might not require these refinements, since the moments are smooth (or at least show smaller discontinuities). To understand this behavior, it remains to systematically investigate the effects of the spatial resolution on Stochastic Collocation and intrusive methods. Furthermore, such a comparison should include Monte-Carlo methods, which we have not studied in this thesis. Also note that several of the concepts proposed in this thesis (such as filters and adaptivity) can be applied to hyperbolicity limiters [143, 117], which should be included in future comparisons as well.

Furthermore, when breaking up the non-intrusive framework of Stochastic Collocation , one can use certain acceleration techniques such as adaptivity for nodal methods, by for example coupling all quadrature points after every time update. The number of quadrature points is then increased or decreased according to a refinement indicator. Here, increasing the number of quadrature points requires some refinement technique for the solution. This can for example be done by constructing a continuous approximation from the coarse set of quadrature points and evaluating this reconstruction at the fine quadrature set. When for example using the IPM reconstruction as a continuous approximation, one preserves the moments while maintaining an admissible solution. Here, we would also like to mention that the quadrature refinement for IPM and its effects on the realizability of the moments should further be investigated. Another idea

to make use of Stochastic Collocation is to use it as a preconditioner for steady state problems: For these kinds of problems, the initial guess of the steady state solution (i.e. the initial condition for the pseudo-time iteration) crucially affects the run time. To obtain a satisfactory initial guess, one could use a cheap, inaccurate method (for example Stochastic Collocation with a low number of quadrature points) to generate an initial condition and then use an expensive but accurate method (for example IPM) to perform the remaining iterations to the steady state.

Recent publications that study intrusive methods have found that using a local basis in the random variable yields improved solution approximations, even when using small polynomial degrees in each local element [29, 30]. Such a local basis seems to be an ideal choice for IPM, since it decouples the IPM optimization problems, enhancing the use of parallel implementations while heavily reducing computational costs: Let us assume divide Θ into elements I_j with $j = 1, \cdots N_I$ and define a local basis φ_i^ℓ, similar to the DG basis, with support in I_j and polynomial order i. Furthermore, we assume $\langle \varphi_i^\ell \varphi_n^k \rangle = \delta_{in}\delta_{\ell k}$. With moments $\hat{u}_i^\ell := \langle u \varphi_i^\ell \rangle$, the IPM system for a scalar problem reads

$$\partial_i \hat{u}_i^\ell + \partial_x \langle f(\mathcal{U}(\hat{\boldsymbol{u}}))\varphi_i^\ell \rangle = 0.$$

Let us derive the IPM closure according to Section 1.4.6, i.e. we minimize the functional

$$L(u, \boldsymbol{\lambda}) = \langle s(u) \rangle + \sum_{j=1}^{N_I} \sum_{|i| \leq N} \lambda_i^\ell \left(\hat{u}_i^\ell - \langle u \varphi_i^\ell \rangle \right).$$

In order to compute the exact minimizer of this functional, we make use of the Gateaux derivative

$$\delta_u L(u, \boldsymbol{\lambda}; v) := \frac{d}{dt} L(u + t \cdot v, \boldsymbol{\lambda}; v)|_{t=0}.$$

Then, if we differentiate with respect to u, we get

$$\delta_u L(u, \boldsymbol{\lambda}; v) = \langle s'(u) v \rangle - \sum_{j=1}^{N_I} \sum_{|i| \leq N} \lambda_i^\ell \langle v \varphi_i^\ell \rangle \stackrel{!}{=} 0 \qquad \forall v,$$

i.e. with $\Lambda := \boldsymbol{\lambda}^T \boldsymbol{\varphi} = \sum_{j=1}^{N_I} \sum_{|i| \leq N} \lambda_i^\ell \varphi_i^\ell$ we again have $u = (s')^{-1}(\Lambda)$. Now plugging this ansatz into L and differentiating with respect to λ_n^k gives

$$\begin{aligned}
\partial_{\lambda_n^k} L(u(\boldsymbol{\lambda}^T \boldsymbol{\varphi}), \boldsymbol{\lambda}) =& \langle s'(u(\Lambda)) u'(\Lambda) \varphi_n^k \rangle + \hat{u}_n^k - \langle u(\Lambda) \varphi_n^k \rangle - \sum_{j=1}^{N_I} \sum_{|i| \leq N} \lambda_i^\ell \langle u'(\Lambda) \varphi_n^k \varphi_i^\ell \rangle \\
=& \langle \Lambda u'(\Lambda) \varphi_n^k \rangle + \hat{u}_n^k - \langle u(\Lambda) \varphi_n^k \rangle - \langle u'(\Lambda) \Lambda \varphi_n^k \rangle \\
=& \hat{u}_n^k - \langle u(\Lambda) \varphi_n^k \rangle \stackrel{!}{=} 0.
\end{aligned} \tag{10.1}$$

Note that since φ_n^k has support I_k, the integral term becomes

$$\begin{aligned}
\langle u(\Lambda) \varphi_n^k \rangle &= \int_{I_k} u \left(\sum_{i,\ell} \lambda_i^\ell \varphi_i^\ell \right) \varphi_n^k(\boldsymbol{\xi}) f_\Xi(\boldsymbol{\xi}) \, d\boldsymbol{\xi} \\
&= \int_{I_k} u \left(\sum_{i} \lambda_i^k \varphi_i^k \right) \varphi_n^k(\boldsymbol{\xi}) f_\Xi(\boldsymbol{\xi}) \, d\boldsymbol{\xi}.
\end{aligned}$$

Hence, when collecting λ_i^k in a vector $\boldsymbol{\lambda}_k = (\lambda_i^k)_{|i|\leq N}$, we can solve (10.1) via

$$\hat{u}_n^k - \langle u(\boldsymbol{\lambda}_k^T \boldsymbol{\varphi}) \varphi_n^k \rangle \stackrel{!}{=} 0 \qquad \text{for } k = 1, \cdots, N_I,$$

i.e. the IPM optimization problems decouple and one can solve the N_I problems per spatial cell individually. Commonly, the multi-element approach allows using a smaller total degree of polynomials. Therefore, instead of solving one expensive optimization problem, we now solve N_I cheaper optimization problems in each cell, which can be distributed to different processors.

An issue with IPM which might lead to problems in high stochastic dimensions is the condition number of the Hessian when using sparse quadrature grids, as observed in Section 4.5.3. On possible strategy to cope with this issue could be the use of regularization strategies as discussed in Section 6.3. Here, one should further investigate the regularization, especially its effect on the run time. The observed run time speedup when increasing the regularization parameter holds the potential of speeding up standard IPM computations. However, one needs to keep in mind that the numerical experiments show strong solution manipulations when the regularization is chosen too big. Also, the construction of the Hessian (and its inverse) for the regularized IPM method could be made more efficient by the definition of rank one updates: Note that the Hessian of the regularized IPM method for scalar equations reads

$$\hat{\boldsymbol{H}}_\eta := \sum_{k=1}^{Q} w_k u(\hat{\boldsymbol{v}}^T \boldsymbol{\varphi}(\boldsymbol{\xi}_k)) \boldsymbol{\varphi}(\boldsymbol{\xi}_k) \boldsymbol{\varphi}(\boldsymbol{\xi}_k)^T + \eta \boldsymbol{I}.$$

The inverse of this Hessian can be computed by using the Sherman-Morison formula [119]. With $\breve{u}_k := u(\hat{\boldsymbol{v}}^T \boldsymbol{\varphi}(\boldsymbol{\xi}_k))$ and $\boldsymbol{\varphi}_k := \boldsymbol{\varphi}(\boldsymbol{\xi}_k)$, we then have

$$\hat{\boldsymbol{H}}_{k+1}^{-1} = \hat{\boldsymbol{H}}_k^{-1} - \frac{\hat{\boldsymbol{H}}_k^{-1} \breve{u}_k \boldsymbol{\varphi}_k \boldsymbol{\varphi}_k^T \hat{\boldsymbol{H}}_k^{-1}}{1 + \breve{u}_k \boldsymbol{\varphi}_k^T \hat{\boldsymbol{H}}_k^{-1} \boldsymbol{\varphi}_k}.$$

With $\hat{\boldsymbol{H}}_0^{-1} := \frac{1}{\eta} \boldsymbol{I}$ we then have $\hat{\boldsymbol{H}}_\eta^{-1} = \hat{\boldsymbol{H}}_Q^{-1}$, i.e. the inverse Hessian is constructed iteratively. This does not necessarily reduce the number of operations, however this approach gives control over potential quadrature refinements, similar to [5]. Further ideas to speed up the Hessian calculations for steady problems could follow the main idea of the One-Shot approach presented in Section 4.2: Similar to the idea of not fully converging the dual problem, it seems plausible to not spend too much time on computing the Hessian when the moments are not close to a steady state. Hessian approximations that can be interesting are BFGS and LBFGS [102, Chapter 6.1], which construct the Hessian from previously computed gradients. Note that this strategy will increase the used memory, since old Hessians or gradients from a certain number of old time steps need to be saved in every spacial cell.

One often mentioned shortcoming of intrusive methods is the need to implement new code. The intrusive code framework [72] simplifies this task, since it simply requires a numerical flux for the underlying original problem. Several open tasks remain to really make the code compatible with the standard simulation software, including high-order schemes, spacial refinement, multigrid approaches, implicit time discretizations and many more. Furthermore, the intrusive framework should be applied to more real-world test cases. Here, our aim is to focus on radiation transport, including applications of practical interest from radiation hydrodynamics and radiation therapy. Real world applications also include more interesting probability density functions (note that we used uniform distributions in almost all test cases) which need to be modeled

from real world data.

The proposed idea of filtering has proven to be an effective strategy to dampen oscillations in various modal but also nodal methods. Due to their easy integration into existing SG and IPM code as well as their nice approximation behavior, we consider filters to be a promising tool in uncertainty quantification. Various properties should be examined in future work. For uncertainty quantification one needs to further study the effect of filters, especially for more general distributions of the random variable. Furthermore, one should study the effects of different filters from the literature and try to come up with a strategy to adaptively change the filter strength, since especially the results for filtered IPM in Section 6.5.2 showed an unnecessary dampening of the variance at the rarefaction wave. Such an effect could be avoided by turning down the filter strength in these regions. An automated, adaptive change of the filtering strength could be derived by forcing the last moments to lie below a certain threshold (similar to the idea of the Lasso filter) or by investigating the regularity of the corresponding equation solved by the filter analytically.

As already mentioned, the intrusive setting can draw from different frequently used intrusive acceleration techniques. One very interesting method is the dynamical low-rank approach [64], which automatically chooses an optimal basis to represent uncertainties as well as the spatial basis and further dimensions of the phase space. Our goal is to implement and investigate such a method for uncertainty quantification.

The applicability of the presented ray-effect mitigation techniques should also be tested in real world applications, such as radiation therapy. To further improve the obtained results one could make the artificial scattering strength depend on the background medium, meaning that artificial scattering is especially active in void regions, whereas highly absorbing and scattering regions have a small artificial scattering cross section. Note that this strategy becomes more difficult for the rS_N method, since the rotated quadrature set must be the same for all spatial cells to ensure an efficient time update by the finite volume method. Furthermore, one can use the presented One-Shot idea to accelerate the implicit asS_N algorithm by performing only one step of the inner look (i.e. one sweeping step) and one step of the outer loop (i.e. one Krylov step). Here, it remains to thoroughly test the efficiency of this approach and (if possible) derive convergence properties.

Lastly, one should mention that when investigating different discretization techniques (or closures to be more precise) show certain similarities and we tried to identify these common ingredients by deriving building blocks. In the following, we lay out a framework which can help with the construction of new closures. The main objectives a successful closure should fulfill are

1. having low computational costs,
2. yielding a small distance to the exact solution,
3. providing a smooth, non-oscillatory reconstruction,
4. fulfilling important physical properties and bounds (realizability).

One can increase the list by for example adding sparsity or an automated parameter choice (e.g. the filter strength), however we will focus on the previously mentioned four properties in the following. The main strategy to construct a closure will be based on the underlying optimization problem the before mentioned closures have in common. In its most general form, this optimization problem consists of a cost function, which we call $\mathcal{J}$, as well as a set of constraints $\boldsymbol{c}$. The resulting constrained optimization problem can then be solved either analytically, which certainly helps fulfilling **Property 1**,

or numerically, which yields additional computational costs. Both, the cost function as well as the constraint can be constructed with certain building blocks which help fulfilling the previously named desirable properties. Some of these building blocks are obvious and are already used in the IPM, SG or fSG methods, others require additional explaining. To clarify notation, let us point out that we call the exact solution which we would like to reconstruct u_{ex}. The only known quantity of this exact solution is its moment vector $\hat{\boldsymbol{u}} = \langle u_{\text{ex}} \boldsymbol{\varphi} \rangle$, from which we want to reconstruct a function which we will call u.

Building blocks fulfilling **Property 2**, i.e. yielding a small distance to the exact solution, are:

i Using the L^2 distance $\langle (u - u_{\text{ex}})^2 \rangle$ (or any other metric) to the exact solution inside the cost function $\mathcal{J}$. This strategy is used to remain close to the exact solution in the construction of SG and fSG methods. Note that the choice of the metric crucially determines whether an analytic formula can be derived. Furthermore, the exact solution u_{ex} is only allowed to show up in terms of moments, since further properties of the exact solution are unknown, which further restricts the choice of the metric.

ii Using the moment constraint $\langle u \boldsymbol{\varphi} \rangle \overset{!}{=} \hat{\boldsymbol{u}}$ as part of $\boldsymbol{c}$ as done in the IPM method.

iii Using a regularization term of the form $\frac{1}{\eta} \| \langle u \boldsymbol{\varphi} \rangle - \hat{\boldsymbol{u}} \|^2$ in the cost function. In this case, the moments of the closure are only required to lie close to the exact moments, i.e. they do not need to match the moments of the exact solution. This term has been used in the regularized minimal entropy closure, introduced in [4].

To enforce non-oscillatory solutions, i.e. **Property 3**, one can make use of:

i a term to penalize oscillatory solutions added to the cost function as done in fSG

ii adding a filtered regularization term $\frac{1}{\eta} \| \langle u \varphi_i \rangle - g_\lambda(i) \hat{u}_i \|^2$ for $i = 0, \cdots, N$

iii using a filtered moment constraint as part of $\boldsymbol{c}$, i.e. using $\langle u \varphi_i \rangle \overset{!}{=} g_\lambda(i) \hat{u}_i$ for $i = 0, \cdots, N$. In this case, special care must be taken when choosing the filter function g_λ, since certain filter functions will generate moments which are not realizable, i.e. which do not belong to a function fulfilling certain physical bounds.

Important physical bounds, i.e. **Property 4**, can be enforced by:

i imposing these bounds through the constraint. To for example ensure positivity, the constraint $u > 0$ for all $\xi \in \Theta$ can be used. To obtain a finite number of constraints, one can impose this constraint on the quadrature set for ξ which is used to approximate integrals in the numerical method.

ii minimizing a correct mathematical entropy. If the slope of the entropy approaches infinity at the imposed bounds, e.g. for zero bounds we need $\lim_{u \searrow 0} s'(u) \to \pm\infty$, the resulting solution ansatz only takes on values bigger than zero [67].

iii making use of a maximum–principle of the equations which provide the reconstruction: If we for example assume that the reconstructed solution u is given by an equation $Mu = u_{\text{ex}}$ and the operator M fulfills a maximum–principle, then the reconstruction u will be bounded by the exact solution.

Remark 20. *Note that IPM is constructed by using 4ii in the cost function and 2ii in the constraint. Stochastic-Galerkin uses 2i in the cost function, filtered stochastic-Galerkin additionally includes 3i in $\mathcal{J}$.*

In the following, we present various promising combinations of the presented building blocks to derive closures, which fulfill the presented properties. We start by combining 2 i and 3 i without restricting the reconstruction to be a polynomial. In this case, we recover the ability to impose desired bounds on the solution by 4 iii. Choosing a differential operator D, which punishes oscillations yields the optimization problem

$$\min_u \left\{ \frac{1}{2}\langle (u - u_{\text{ex}})^2 \rangle + \frac{\lambda}{2}\langle (Du)^2 \rangle \right\}. \tag{10.2}$$

The solution u of the minimization problem (10.2) fulfills

$$\delta_u \mathcal{J}(u; v) = \langle (u - u_{\text{ex}})v \rangle + \lambda \langle Du \cdot Dv \rangle \stackrel{!}{=} 0$$

for all test functions v. If D^* is the adjoint operator of D, the optimality criterion reads

$$u + \lambda D^* D u \stackrel{!}{=} u_{\text{ex}}. \tag{10.3}$$

When the operator $M := 1 + \lambda D^* D$ fulfills a maximum principle, the reconstruction u will remain in the bounds of the exact solution u_{ex}. Note that for now, we are reconstructing u as a continuous function in ξ. The issue here is that the exact solution shows up, however we only know the first $N+1$ moments of this solution. Therefore, we need to find some reconstruction $\tilde{u}$ from the moments $\hat{\boldsymbol{u}}$, which fulfills bounds of the exact solution. Here, we can perform an entropy reconstruction

$$\tilde{u} = \arg\min_u \langle s(u) \rangle \quad \text{subject to } \hat{u}_i = \langle u \varphi_i \rangle \text{ for } i = 0, \cdots, N \tag{10.4}$$

and then replace u_{ex} in (10.3) by $\tilde{u}$. The resulting equation

$$u + \lambda D^* D u \stackrel{!}{=} \tilde{u} \tag{10.5}$$

needs to be solved numerically without violating the maximum-principle. The resulting reconstruction u will be non-oscillatory while fulfilling the bounds imposed by the entropy reconstruction $\tilde{u}$. Note that this method has high computational costs, since we need to compute the IPM reconstruction by solving the optimization problem (10.4) as well as computing the solution to the differential equation (10.5).

The next idea combines building blocks 2 ii, 3 i and 4 ii, i.e. we wish to include a punishing term for oscillations in the IPM optimization problem:

$$u = \arg\min_u \left\{ \langle s(u) \rangle + \lambda \langle (Lu)^2 \rangle \right\} \quad \text{subject to } \langle u \boldsymbol{\varphi} \rangle = \hat{\boldsymbol{u}}.$$

Setting up the Lagrangian of this problem with Lagrange multipliers $\hat{\boldsymbol{v}} \in \mathbb{R}^{N+1}$ and then determining the Karush-Kuhn-Tucker condition using the Gateaux derivative as done in the previous method yields

$$\begin{aligned} s'(u) + \lambda L^* L u &= \hat{\boldsymbol{v}}^T \boldsymbol{\varphi} \\ \langle u \boldsymbol{\varphi} \rangle &= \hat{\boldsymbol{u}}. \end{aligned}$$

Assuming that we discretize u on a quadrature set with N_q quadrature points, we need to solve this set of equations for $\check{\boldsymbol{u}} := (u(\xi_1), \cdots, u(\xi_{N_q}))^T \in \mathbb{R}^{N_q}$ and $\hat{\boldsymbol{v}} \in \mathbb{R}^{N+1}$.

The next idea uses the filtered stochastic-Galerkin method and enforces physical bounds (in our example a zero-bound) directly through the constraint, i.e. we combine

2 i and 3 i with 4 i to get

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \langle (\boldsymbol{\alpha}^T \boldsymbol{\varphi} - u_{\text{ex}})^2 \rangle + \lambda T(\boldsymbol{\alpha}) \text{ subject to } \boldsymbol{\alpha}^T \boldsymbol{\varphi}(\xi) > 0 \text{ for all } \xi \in \Theta.$$

Again, to obtain a finite number of constraints, we can restrict ξ to the chosen quadrature sets with N_q nodes. Note that this closure is similar to the positive P_N closure, used in kinetic theory [52] with an additional filtering term.

Furthermore, for a given filter function g_λ one can pick the filter strength, such that the solution fulfills specified bounds, hence we get the reconstruction $u = \sum_{i=0}^{N} g_{\lambda^*}(i)\hat{u}_i\varphi_i$ with

$$\lambda^* := \arg\min_{\lambda} \lambda^2 \quad \text{such that } \sum_{i=0}^{N} g_{\lambda^*}(i)\hat{u}_i\varphi_i(\xi) > 0 \quad \text{and } \lambda > \lambda_{\min},$$

where again it suffices to fulfill the first constraint on a finite set of quadrature points. The second constraint is used to ensure that filtering is not turned off in regions which fulfill positivity for $\lambda = 0$. Note that since $u \to \hat{u}_0$ for $\lambda^* \to \infty$, the optimization problem above will have a solution.

We can also simply solve the standard IPM optimization problem with filtered moments, i.e. we combine building blocks 2 ii and 3 iii as well as 4 ii to construct the constrained optimization problem

$$\mathcal{U}(\hat{\boldsymbol{u}}) = \arg\min_{u} \langle s(u) \rangle \quad \text{subject to } g_\lambda(i)\hat{u}_i = \langle u\varphi_i \rangle \text{ for } i = 0, \cdots, N.$$

As already mentioned using building block 2 ii is critical: One needs to ensure that the filter function g_λ generates moments that remain realizable. If this is not the case, we can instead use the following strategy:

Instead of using the original constrained optimization problem from IPM, we write down the regularized IPM system with filtered moments. Hence by combining building blocks 2 iii and 3 ii as well as 4 ii, we obtain the optimization problem

$$\min_{u} \left\{ \langle s(u) \rangle + \frac{1}{\eta} \sum_{i=0}^{N} \left(g_\lambda(i)\hat{u}_i - \langle u\varphi_i \rangle \right) \right\}.$$

Note that the moments $g_\lambda(i)\hat{u}_i$ might not be realizable, however by not imposing these moment as a constraint, the optimization problem will have a unique solution. The filter function g_λ is not restricted to the L^2 filter, but can be constructed freely.

These closures are not restricted to uncertainty quantification as they can for example be used in kinetic theory and other research areas which require the construction of closures. Furthermore, one should further investigate the regularization, especially its effect on the run time. The observed run time speedup when increasing the regularization parameter holds the potential of speeding up standard IPM computations. However, one needs to keep in mind that the numerical experiments show strong solution manipulations when the regularization is chosen too big.

Bibliography

[1] IK Abu-Shumays. "Angular quadratures for improved transport computations". In: *Transport Theory and Statistical Physics* 30.2-3 (2001), pp. 169–204.

[2] AK Alekseev, IM Navon, and ME Zelentsov. "The estimation of functional uncertainty using polynomial chaos and adjoint equations". In: *International Journal for numerical methods in fluids* 67.3 (2011), pp. 328–341.

[3] Graham Alldredge, Cory D Hauck, and Andre L Tits. "High-order entropy-based closures for linear transport in slab geometry II: A computational study of the optimization problem". In: *SIAM Journal on Scientific Computing* 34.4 (2012), B361–B391.

[4] Graham W. Alldredge, Martin Frank, and Cory D. Hauck. "A regularized entropy-based moment method for kinetic equations". In: *SIAM Journal on Applied Mathematics* 79.5 (2019), pp. 1627–1653.

[5] Graham W Alldredge et al. "Adaptive change of basis in entropy-based moment closures for linear kinetic equations". In: *Journal of Computational Physics* 258 (2014), pp. 489–508.

[6] SF Ashby et al. "Preconditioned iterative methods for discretized transport equations". In: *Proc. International Topical Meeting on Advances in Mathematics, Computations, Reactor Physics*. Vol. 2. 6.1. 1991, pp. 2–1.

[7] Ivo Babuška, Fabio Nobile, and Raul Tempone. "A stochastic collocation method for elliptic partial differential equations with random input data". In: *SIAM Journal on Numerical Analysis* 45.3 (2007), pp. 1005–1034.

[8] Timothy Barth. "Non-intrusive uncertainty propagation with error bounds for conservation laws containing discontinuities". In: *Uncertainty quantification in computational fluid dynamics*. Springer, 2013, pp. 1–57.

[9] Timothy Barth. "On discontinuous Galerkin approximations of Boltzmann moment systems with Levermore closure". In: *Computer methods in applied mechanics and engineering* 195.25-28 (2006), pp. 3311–3330.

[10] Richard M Beam and Robert F Warming. "An implicit finite-difference algorithm for hyperbolic systems in conservation-law form". In: *Journal of computational physics* 22.1 (1976), pp. 87–110.

[11] John B Bell, Clint N Dawson, and Gregory R Shubin. "An unsplit, higher order Godunov method for scalar conservation laws in multiple dimensions". In: *Journal of Computational Physics* 74.1 (1988), pp. 1–24.

[12] Jonathan M Borwein and Adrian Stephen Lewis. "Convergence of best entropy estimates". In: *SIAM Journal on Optimization* 1.2 (1991), pp. 191–205.

[13] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[14] JP Boyd. "The erfc-log filter and the asymptotics of the Euler and Vandeven sequence accelerations". In: *Proceedings of the Third International Conference on Spectral and High Order Methods*. Houston Math. J. 1996, pp. 267–276.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[16] Thomas A Brunner. "Forms of approximate radiation transport". In: *Sandia report* (2002).

[17] Thomas Camminady et al. "Ray effect mitigation for the discrete ordinates method through quadrature rotation". In: *Journal of Computational Physics* 382 (2019), pp. 105–123.

[18] Kenneth M Case and Paul Frederick Zweifel. *Linear transport theory*. Addison-Wesley Pub. Co., 1967.

[19] Charles W Clenshaw and Alan R Curtis. "A method for numerical integration on an automatic computer". In: *Numerische Mathematik* 2.1 (1960), pp. 197–205.

[20] Phillip Colella. "Multidimensional upwind methods for hyperbolic conservation laws". In: *Journal of Computational Physics* 87.1 (1990), pp. 171–200.

[21] Michael G Crandall and Andrew Majda. "Monotone difference approximations for scalar conservation laws". In: *Mathematics of Computation* 34.149 (1980), pp. 1–21.

[22] Raúl E Curto and Lawrence A Fialkow. "Recursiveness, positivity, and truncated moment problems". In: *Houston Journal of Mathematics* 17.4 (1991), pp. 603–635.

[23] Wolfgang Dahmen and Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Verlag, 2006.

[24] Bert J Debusschere et al. "Numerical challenges in the use of polynomial chaos representations for stochastic processes". In: *SIAM journal on scientific computing* 26.2 (2004), pp. 698–719.

[25] Andrée Decarreau et al. "Dual methods in entropy maximization. Application to some problems in crystallography". In: *SIAM Journal on Optimization* 2.2 (1992), pp. 173–197.

[26] Suresh Deshpande. "Kinetic theory based new upwind methods for inviscid compressible flows". In: *24th Aerospace Sciences Meeting*. 1986, p. 275.

[27] Bruno Després, Gaël Poëtte, and Didier Lucor. "Robust Uncertainty Propagation in Systems of Conservation Laws with the Entropy Closure Method". In: *Uncertainty Quantification in Computational Fluid Dynamics*. Ed. by Hester Bijl et al. Springer International Publishing, 2013, pp. 105–149.

[28] James J Duderstadt. *Nuclear reactor analysis*. Wiley, 1976.

[29] Jakob Dürrwächter et al. *A High-Order Stochastic Galerkin Code for the Compressible Euler and Navier-Stokes Equations*. Oct. 2019. DOI: 10.13140/RG.2.2.30609.99686.

[30] Jakob Dürrwächter et al. "A hyperbolicity-preserving discontinuous stochastic Galerkin scheme for uncertain hyperbolic systems of equations". In: *arXiv preprint arXiv:1805.10177* (2018).

[31] Richard P Dwight, Jeroen AS Witteveen, and Hester Bijl. "Adaptive uncertainty quantification for computational fluid dynamics". In: *Uncertainty Quantification in Computational Fluid Dynamics*. Springer, 2013, pp. 151–191.

[32] Thomas D Economon et al. "SU2: An open-source suite for multiphysics simulation and design". In: *AIAA Journal* 54.3 (2015), pp. 828–846.

[33] Michael Eldred and John Burkardt. "Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification". In: *47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*. 2009, p. 976.

[34] V Faber and Thomas A Manteuffel. *A look at transport theory from the point of view of linear algebra*. Tech. rep. Los Alamos National Lab., NM (USA), 1988.

[35] Martin Frank, Cory Hauck, and Kerstin Küpper. "Convergence of filtered spherical harmonic equations for radiation transport". In: *Commun. Math. Sci* 14.5 (2016), pp. 1443–1465.

[36] Martin Frank et al. "Ray Effect Mitigation for the Discrete Ordinates Method Using Artificial Scattering". In: *Nuclear Science and Engineering* (2020), pp. 1–18.

[37] Christopher L Fryer, Gabriel Rockefeller, and Michael S Warren. "SNSPH: a parallel three-dimensional smoothed particle radiation hydrodynamics code". In: *The Astrophysical Journal* 643.1 (2006), p. 292.

[38] BD Ganapol et al. *Homogeneous infinite media time-dependent analytical benchmarks*. Tech. rep. Los Alamos National Laboratory, 2001.

[39] C. Kristopher Garrett, Cory Hauck, and Judith Hill. "Optimization and large scale computation of an entropy-based moment closure". In: *Journal of Computational Physics* 302 (2015), pp. 573–590.

[40] C Kristopher Garrett and Cory D Hauck. "A comparison of moment closures for linear kinetic transport equations: The line source benchmark". In: *Transport Theory and Statistical Physics* 42.6-7 (2013), pp. 203–235.

[41] Gregor J Gassner and Andrea D Beck. "On the accuracy of high-order discretizations for underresolved turbulence simulations". In: *Theoretical and Computational Fluid Dynamics* 27.3-4 (2013), pp. 221–237.

[42] Roger Ghanem and S Dham. "Stochastic finite element analysis for multiphase flow in heterogeneous porous media". In: *Transport in porous media* 32.3 (1998), pp. 239–262.

[43] Roger G Ghanem and Pol D Spanos. *Stochastic Finite Elements: A Spectral Approach*. Dover, 2003.

[44] Jan Giesselmann, Fabian Meyer, and Christian Rohde. "A posteriori error analysis for random scalar conservation laws using the Stochastic Galerkin method". In: *arXiv preprint arXiv:1709.04351* (2017).

[45] Edwige Godlewski and Pierre-Arnaud Raviart. *Hyperbolic systems of conservation laws*. Ellipses, 1991.

[46] David Gottlieb and Dongbin Xiu. "Galerkin method for wave equations with uncertain coefficients". In: *Commun. Comput. Phys* 3.2 (2008), pp. 505–518.

[47] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. "Strong stability-preserving high-order time discretization methods". In: *SIAM review* 43.1 (2001), pp. 89–112.

[48] Jean-Luc Guermond et al. "A second-order maximum principle preserving Lagrange finite element technique for nonlinear scalar conservation equations". In: *SIAM Journal on Numerical Analysis* 52.4 (2014), pp. 2163–2182.

[49] Benyu Guo. *Spectral methods and their applications*. World Scientific, 1998.

[50] Amiram Harten, Peter D Lax, and Bram van Leer. "On upstream differencing and Godunov-type schemes for hyperbolic conservation laws". In: *SIAM review* 25.1 (1983), pp. 35–61.

[51] Cory Hauck and Vincent Heningburg. "Filtered Discrete Ordinates Equations for Radiative Transport". In: *Journal of Scientific Computing* 80.1 (2019), pp. 614–648.

[52] Cory Hauck and Ryan McClarren. "Positive P_N Closures". In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 2603–2626.

[53] SB Hazra et al. "Aerodynamic shape optimization using simultaneous pseudo-timestepping". In: *Journal of Computational Physics* 204.1 (2005), pp. 46–64.

[54] Stefan Heinrich. "Multilevel monte carlo methods". In: *International Conference on Large-Scale Scientific Computing*. Springer. 2001, pp. 58–67.

[55] Allan F Henry, CC Scott, and S Moorthy. "Nuclear reactor analysis". In: *IEEE Transactions on Nuclear Science* 24.6 (1977), pp. 2566–2567.

[56] Jan S Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral methods for time-dependent problems*. Vol. 21. Cambridge University Press, 2007.

[57] W Höbel et al. "Energy transport in matter with rapidly changing states". In: *Nuclear science and engineering* 137.3 (2001), pp. 334–351.

[58] Helge Holden and Nils Henrik Risebro. *Front tracking for hyperbolic conservation laws*. Vol. 152. Springer, 2015.

[59] Jingwei Hu and Shi Jin. "A stochastic Galerkin method for the Boltzmann equation with uncertainty". In: *Journal of Computational Physics* 315 (2016), pp. 150–168.

[60] Jingwei Hu, Shi Jin, and Dongbin Xiu. "A Stochastic Galerkin Method for Hamilton–Jacobi Equations with Uncertainty". In: *SIAM Journal on Scientific Computing* 37.5 (2015), A2246–A2269.

[61] Eastman N Jacobs, Kenneth E Ward, and Robert M Pinkerton. "The characteristics of 78 related airfoil sections from tests in the variable-density wind tunnel". In: (1933).

[62] Shi Jin, Jian-Guo Liu, and Zheng Ma. "Uniform spectral convergence of the stochastic Galerkin method for the linear transport equations with random inputs in diffusive regime and a micro–macro decomposition-based asymptotic-preserving method". In: *Research in the Mathematical Sciences* 4.1 (2017), p. 15.

[63] Jungchung Jung et al. "Discrete ordinate neutron transport equation equivalent to PL approximation". In: *Nuclear Science and Engineering* 49.1 (1972), pp. 1–9.

[64] Othmar Koch and Christian Lubich. "Dynamical low-rank approximation". In: *SIAM Journal on Matrix Analysis and Applications* 29.2 (2007), pp. 434–454.

[65] Ilja Kröker and Christian Rohde. "Finite volume schemes for hyperbolic balance laws with multiplicative noise". In: *Applied Numerical Mathematics* 62.4 (2012), pp. 441–456.

[66] Stanislav N Kružkov. "First order quasilinear equations in several independent variables". In: *Mathematics of the USSR-Sbornik* 10.2 (1970), p. 217.

[67] Jonas Kusch, Graham W. Alldredge, and Martin Frank. "Maximum-principle-satisfying second-order Intrusive Polynomial Moment scheme". en. In: *The SMAI journal of computational mathematics* 5 (2019), pp. 23–51. DOI: 10.5802/smai-jcm.42. URL: https://smai-jcm.centre-mersenne.org/item/SMAI-JCM_2019__5__23_0.

[68] Jonas Kusch and Martin Frank. "An adaptive quadrature-based moment closure". In: *International Journal of Advances in Engineering Sciences and Applied Mathematics* 11.3 (2019), pp. 174–186.

[69] Jonas Kusch and Martin Frank. "Intrusive methods in uncertainty quantification and their connection to kinetic theory". In: *International Journal of Advances in Engineering Sciences and Applied Mathematics* 10.1 (2018), pp. 54–69.

[70] Jonas Kusch, Ryan G McClarren, and Martin Frank. "Filtered stochastic galerkin methods for hyperbolic equations". In: *Journal of Computational Physics* 403 (2020), p. 109073.

[71] Jonas Kusch, Jannick Wolters, and Martin Frank. "Intrusive acceleration strategies for Uncertainty Quantification for hyperbolic systems of conservation laws". In: *Journal of Computational Physics* (2020), p. 109698.

[72] Jonas Kusch, Jannick Wolters, and Martin Frank. *UQCreator*. https://git.scc.kit.edu/uqcreator. 2019.

[73] Vincent M Laboure, Ryan G McClarren, and Cory D Hauck. "Implicit filtered P_N for high-energy density thermal radiation transport using discontinuous Galerkin finite elements". In: *Journal of Computational Physics* 321 (2016), pp. 624–643.

[74] Kaye D Lathrop. "Ray effects in discrete ordinates equations". In: *Nuclear Science and Engineering* 32.3 (1968), pp. 357–369.

[75] KD Lathrop. "Remedies for ray effects". In: *Nuclear Science and Engineering* 45.3 (1971), pp. 255–268.

[76] Peter Lax and Burton Wendroff. "Systems of conservation laws". In: *Communications on Pure and Applied mathematics* 13.2 (1960), pp. 217–237.

[77] OP Le Maıtre et al. "Uncertainty propagation using Wiener–Haar expansions". In: *Journal of computational Physics* 197.1 (2004), pp. 28–57.

[78] Randall J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser Verlag Basel, 1992.

[79] Randall J LeVeque and Randall J Leveque. *Numerical methods for conservation laws*. Vol. 132. Springer, 1992.

[80] C David Levermore. "Moment closure hierarchies for kinetic theories". In: *Journal of Statistical Physics* 83.5-6 (1996), pp. 1021–1065.

[81] Elmer Eugene Lewis and Warren F Miller. *Computational Methods of Neutron Transport*. John Wiley and Sons Inc., 1984.

[82] Xu-Dong Liu. "A maximum principle satisfying modification of triangle based adapative stencils for the solution of scalar hyperbolic conservation laws". In: *SIAM journal on numerical analysis* 30.3 (1993), pp. 701–716.

[83] Xu-Dong Liu and Stanley Osher. "Nonoscillatory high order accurate self-similar maximum principle satisfying shock capturing schemes I". In: *SIAM Journal on Numerical Analysis* 33.2 (1996), pp. 760–779.

[84] GJA Loeven and H Bijl. "Probabilistic collocation used in a two-step approach for efficient uncertainty quantification in computational fluid dynamics". In: *Computer Modeling in Engineering & Sciences* 36.3 (2008), pp. 193–212.

[85] RB Lowrie, JE Morel, and JA Hittinger. "The coupling of radiation and hydrodynamics". In: *The astrophysical journal* 521.1 (1999), p. 432.

[86] MM Marinak et al. "Three-dimensional HYDRA simulations of National Ignition Facility targets". In: *Physics of Plasmas* 8.5 (2001), pp. 2275–2280.

[87] Kirk A Mathews. "On the propagation of rays in discrete ordinates". In: *Nuclear science and engineering* 132.2 (1999), pp. 155–180.

[88] M Keith Matzen et al. "Pulsed-power-driven high energy density physics and inertial confinement fusion research". In: *Physics of Plasmas* 12.5 (2005), p. 055503.

[89] Ryan G. McClarren. *Uncertainty Quantification and Predictive Computational Science: A Foundation for Physical Scientists and Engineers*. Springer International Publishing, 2018.

[90] Ryan G McClarren and R Paul Drake. "Anti-diffusive radiation flow in the cooling layer of a radiating shock". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 111.14 (2010), pp. 2095–2105.

[91] Ryan G McClarren and Cory D Hauck. "Robust and accurate filtered spherical harmonics expansions for radiative transfer". In: *Journal of Computational Physics* 229.16 (2010), pp. 5597–5614.

[92] Ryan G McClarren, James Paul Holloway, and Thomas A Brunner. "Analytic P1 solutions for time-dependent, thermal radiative transfer in several geometries". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 109.3 (2008), pp. 389–403.

[93] Ryan G McClarren and Robert B Lowrie. "Manufactured solutions for the P1 radiation-hydrodynamics equations". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 109.15 (2008), pp. 2590–2602.

[94] Ryan G McClarren et al. "Semi-implicit time integration for PN thermal radiative transfer". In: *Journal of Computational Physics* 227.16 (2008), pp. 7561–7586.

[95] Lawrence R Mead and Nikos Papanicolaou. "Maximum entropy in the problem of moments". In: *Journal of Mathematical Physics* 25.8 (1984), pp. 2404–2417.

[96] WF Miller Jr and Wm H Reed. "Ray-effect mitigation methods for two-dimensional neutron transport theory". In: *Nuclear Science and Engineering* 62.3 (1977), pp. 391–411.

[97] Siddhartha Mishra and Ch Schwab. "Sparse tensor multi-level Monte Carlo finite volume methods for hyperbolic conservation laws with random initial data". In: *Mathematics of computation* 81.280 (2012), pp. 1979–2018.

[98] Siddhartha Mishra, Ch Schwab, and Jonas Šukys. "Multi-level Monte Carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions". In: *Journal of Computational Physics* 231.8 (2012), pp. 3365–3388.

[99] Siddhartha Mishra et al. "Numerical solution of scalar conservation laws with random flux functions". In: *SIAM/ASA Journal on Uncertainty Quantification* 4.1 (2016), pp. 552–591.

[100] JE Morel et al. "Analysis of ray-effect mitigation techniques". In: *Nuclear science and engineering* 144.1 (2003), pp. 1–22.

[101] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. Vol. 63. Siam, 1992.

[102] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[103] Philipp Öffner, Jan Glaubitz, and Hendrik Ranocha. "Stability of correction procedure via reconstruction with summation-by-parts operators for Burgers' equation using a polynomial chaos approach". In: *arXiv preprint arXiv:1703.03561* (2017).

[104] Gordon L Olson, Lawrence H Auer, and Michael L Hall. "Diffusion, P1, and other approximate forms of radiation transport". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 64.6 (2000), pp. 619–634.

[105] Stanley Osher and Sukumar Chakravarthy. "High resolution schemes and the entropy condition". In: *SIAM Journal on Numerical Analysis* 21.5 (1984), pp. 955–984.

[106] Per-Olof Persson and Jaime Peraire. "Sub-cell shock capturing for discontinuous Galerkin methods". In: *44th AIAA Aerospace Sciences Meeting and Exhibit*. 2006, p. 112.

[107] Benoît Perthame. "Boltzmann type schemes for gas dynamics and the entropy property". In: *SIAM Journal on Numerical Analysis* 27.6 (1990), pp. 1405–1421.

[108] Benoıt Perthame. "Second-order Boltzmann schemes for compressible Euler equations in one and two space dimensions". In: *SIAM Journal on Numerical Analysis* 29.1 (1992), pp. 1–19.

[109] Per Pettersson, Gianluca Iaccarino, and Jan Nordström. "Numerical analysis of the Burgers' equation in the presence of uncertainty". In: *Journal of Computational Physics* 228.22 (2009), pp. 8394–8412.

[110] Gaël Poëtte. "Contribute to the mathematical and numerical analysis of uncertaint systems of conservation laws and of the linear and nonlinear Boltzmann equation". PhD thesis. 2019.

[111] Gaël Poëtte, Bruno Després, and Didier Lucor. "Uncertainty propagation for systems of conservation laws, high order stochastic spectral methods". In: *Spectral and High Order Methods for Partial Differential Equations*. Springer, 2011, pp. 293–305.

[112] Gaël Poëtte, Bruno Després, and Didier Lucor. "Uncertainty quantification for systems of conservation laws". In: *Journal of Computational Physics* 228.7 (2009), pp. 2443–2467.

[113] Gerald C Pomraning. *The Equations of Radiation Hydrodynamics*. Oxford, 1973.

[114] David Radice et al. "A new spherical harmonics scheme for multi-dimensional radiation transport I. Static matter configurations". In: *Journal of Computational Physics* 242 (2013), pp. 648–669.

[115] Wm H Reed. "Spherical harmonic solutions of the neutron transport equation from discrete ordinate codes". In: *Nuclear Science and Engineering* 49.1 (1972), pp. 10–19.

[116] Nils Henrik Risebro, Christoph Schwab, and Franziska Weber. "Multilevel Monte Carlo front-tracking for random scalar conservation laws". In: *BIT Numerical Mathematics* 56.1 (2016), pp. 263–292.

[117] Louisa Schlachter and Florian Schneider. "A hyperbolicity-preserving stochastic Galerkin approximation for uncertain hyperbolic systems of equations". In: *Journal of Computational Physics* 375 (2018), pp. 80–98.

[118] Louisa Schlachter, Florian Schneider, and Oliver Kolb. "Weighted Essentially Non-Oscillatory stochastic Galerkin approximation for hyperbolic conservation laws". In: *arXiv preprint arXiv:1912.09171* (2019).

[119] Jack Sherman. "Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix". In: *Annals of mathematical statistics* 20.4 (1949), p. 621.

[120] James Alexander Shohat and Jacob David Tamarkin. *The problem of moments*. 1. American Mathematical Soc., 1943.

[121] Chi-Wang Shu. "Total-variation-diminishing time discretizations". In: *SIAM Journal on Scientific and Statistical Computing* 9.6 (1988), pp. 1073–1084.

[122] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*. Vol. 12. Siam, 2013.

[123] Joel Smoller. *Shock waves and reaction—diffusion equations*. Vol. 258. Springer Science & Business Media, 2012.

[124] Gary A Sod. "A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws". In: *Journal of computational physics* 27.1 (1978), pp. 1–31.

[125] Bingjing Su and Gordon L Olson. "An analytical benchmark for non-equilibrium radiative transfer in an isotropically scattering medium". In: *Annals of Nuclear Energy* 24.13 (1997), pp. 1035–1055.

[126] Timothy John Sullivan. *Introduction to uncertainty quantification*. Vol. 63. Springer, 2015.

[127] George W Sutton and Arthur Sherman. *Engineering magnetohydrodynamics*. Courier Dover Publications, 2006.

[128] Peter K Sweby. "High resolution schemes using flux limiters for hyperbolic conservation laws". In: *SIAM journal on numerical analysis* 21.5 (1984), pp. 995–1011.

[129] F Douglas Swesty and Eric S Myra. "A numerical algorithm for modeling multigroup neutrino-radiation hydrodynamics in two spatial dimensions". In: *The Astrophysical Journal Supplement Series* 181.1 (2009), p. 1.

[130] Eitan Tadmor. "Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems". In: *Acta Numerica* 12 (2003), pp. 451–512.

[131] Eitan Tadmor. "Numerical viscosity and the entropy condition for conservative difference schemes". In: *Mathematics of Computation* 43.168 (1984), pp. 369–381.

[132] John Tencer. "Ray Effect Mitigation Through Reference Frame Rotation". In: *Journal of Heat Transfer* 138.11 (2016), p. 112701.

[133] CP Thurgood, A Pollard, and HA Becker. "The TN quadrature set for the discrete ordinates method". In: *Journal of heat transfer* 117.4 (1995), pp. 1068–1070.

[134] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

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

[136] Lloyd N Trefethen. "Cubature, approximation, and isotropy in the hypercube". In: *SIAM Review* 59.3 (2017), pp. 469–491.

[137] J Tryoen et al. "Instrusive projection methods with upwinding for uncertain non-linear hyperbolic systems". In: *Preprint* (2010).

[138] Julie Tryoen, O Le Maıtre, and Alexandre Ern. "Adaptive anisotropic spectral stochastic methods for uncertain scalar conservation laws". In: *SIAM Journal on Scientific Computing* 34.5 (2012), A2459–A2481.

[139] JMAM Van Neerven. "Stochastic evolution equations". In: *ISEM lecture notes* (2008).

[140] Xiaoliang Wan and George Em Karniadakis. "Multi-element generalized polynomial chaos for arbitrary probability measures". In: *SIAM Journal on Scientific Computing* 28.3 (2006), pp. 901–928.

[141] Pieter Wesseling. *Principles of computational fluid dynamics*. Vol. 29. Springer Science & Business Media, 2009.

[142] Norbert Wiener. "The homogeneous chaos". In: *American Journal of Mathematics* 60.4 (1938), pp. 897–936.

[143] Kailiang Wu, Huazhong Tang, and Dongbin Xiu. "A stochastic Galerkin method for first-order quasilinear hyperbolic systems with uncertainty". In: *Journal of Computational Physics* 345 (2017), pp. 224–244.

[144] Shuhuang Xiang and Folkmar Bornemann. "On the convergence rates of Gauss and Clenshaw–Curtis quadrature for functions of limited regularity". In: *SIAM Journal on Numerical Analysis* 50.5 (2012), pp. 2581–2587.

[145] Dongbin Xiu. "Fast numerical methods for stochastic computations: a review". In: *Communications in computational physics* 5.2-4 (2009), pp. 242–272.

[146] Dongbin Xiu and Jan S Hesthaven. "High-order collocation methods for differential equations with random inputs". In: *SIAM Journal on Scientific Computing* 27.3 (2005), pp. 1118–1139.

[147] Dongbin Xiu and George Em Karniadakis. "The Wiener–Askey polynomial chaos for stochastic differential equations". In: *SIAM journal on scientific computing* 24.2 (2002), pp. 619–644.

[148] Xiangxiong Zhang and Chi-Wang Shu. "Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* (2011).

[149] Xiangxiong Zhang and Chi-Wang Shu. "On maximum-principle-satisfying high order schemes for scalar conservation laws". In: *Journal of Computational Physics* 229.9 (2010), pp. 3091–3120.