Synergies between Numerical Methods for Kinetic Equations and Neural Networks

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN

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

DISSERTATION

von

Steffen Schotthöfer

Tag der mündlichen Prüfung: 3. Mai 2023

1. Referent: Prof. Dr. Martin Frank
2. Referent: Prof. Dr. André Platzer
3. Referent: Prof. Dr. Cory Hauck

per aspera ad astra

Acknowledgements

I would like to express my sincere gratitude to all those who have contributed to the completion of this dissertation. First and foremost, I would like to thank my supervisor, Martin Frank, for his encouragement to pursue my own research directions and for constructive feedback that has been instrumental in shaping the quality of this work. He gave me the freedom and support to present and collaborate internationally, resulting in several fruitful projects and introduced me to many talented researchers.
One such researcher is Cory Hauck, whose expertise in modern applications of entropy closures and kinetic modeling has been invaluable, and his in-depth feedback has improved my work significantly. He also hosted my research stay at the Oak Ridge National Laboratory, an experience that was both inspiring and motivating and played a key role in my decision to pursue a career in science.
I am also grateful to André Platzer, who took the time to discuss and improve my dissertation. As a logician, André provided me with a rigorous point of view, which helped me to structure my method validations more effectively.
My gratitude goes to Paul Laiu, who always had time for a question and discussion about entropy closures and moment methods. His mathematical precision greatly improved the quality of our papers.
I would like to express my appreciation to Jonas Kusch, who got me up to speed on kinetic schemes and numerical methods for transport. Additionally, he introduced me to the fascinating field of dynamical low-rank methods, and I am fortunate to have had the opportunity to learn from him. I would also like to acknowledge the outstanding team working on the low-rank training project, including Gianluca Ceruti, Francesco Tudisco, Emanuele Zangrando and Jonas. Their collaborative efforts and invaluable contributions have gone a long way, and we are not done yet.
I also owe thanks to everyone who took their time to proof-read my dissertation, namely Tim Ortkamp, Pia Stammer, Jonas Kusch, Alexandra Walter, Stephanie Hofmann, and Alexander Jesser.
I would like to extend my gratitude to my colleagues at KIT. I have always appreciated the lively atmosphere in the office, the shared struggles and successes, and of course the joint activities after work. Special thanks to Jonas Kusch, Pia Stammer, Jannick Wolters, and Tianbai Xiao within the KiT-RT project for the entertaining coffee breaks, and to Gaya Čaklović for the countless brainstorming sessions.
Last, but certainly not least, I am deeply grateful for the love and support of my family. My parents, Karin and Detlef, encouraged me to pursue a doctorate in the first place and supported me in countless ways during my studies. My grandfather Theodor sparked my interest in problem solving. My fiancée Antonia Chu provided constant moral support and was a reminder that there is more to life than work or this thesis.

Preface

The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble.

PAUL DIRAC, *circa 1929*

Motivation

The overarching theme of this work is the efficient computation of large-scale systems. Here we deal with two types of mathematical challenges, which are quite different at first glance but offer similar opportunities and challenges upon closer examination.

Physical descriptions of phenomena and their mathematical modeling are performed on diverse scales, ranging from nano-scale interactions of single atoms to the macroscopic dynamics of the earth's atmosphere. We consider such systems of interacting particles and explore methods to simulate them efficiently and accurately, with a focus on the kinetic and macroscopic description of interacting particle systems. Macroscopic governing equations describe the time evolution of a system in time and space, whereas the more fine-grained kinetic description additionally takes the particle velocity into account.
The study of discretizing kinetic equations that depend on space, time, and velocity variables is a challenge due to the need to preserve physical solution bounds, e.g. positivity, avoiding spurious artifacts and computational efficiency. In the pursuit of overcoming the challenge of computability in both kinetic and multi-scale modeling, a wide variety of approximative methods have been established in the realm of reduced order and surrogate modeling, and model compression. For kinetic models, this may manifest in hybrid numerical solvers, that switch between macroscopic and mesoscopic simulation, asymptotic preserving schemes, that bridge the gap between both physical resolution levels, or surrogate models that operate on a kinetic level but replace computationally heavy operations of the simulation by fast approximations.
Thus, for the simulation of kinetic and multi-scale systems with a high spatial resolution and long temporal horizon, the quote by Paul Dirac is as relevant as it was almost a century ago. The first goal of the dissertation is therefore the development of acceleration strategies for kinetic discretization methods, that preserve the structure of their governing equations. Particularly, we investigate the use of convex neural

networks, to accelerate the minimal entropy closure method. Further, we develop a neural network-based hybrid solver for multi-scale systems, where kinetic and macroscopic methods are chosen based on local flow conditions.

Furthermore, we deal with the compression and efficient computation of neural networks. In the meantime, neural networks are successfully used in different forms in countless scientific works and technical systems, with well-known applications in image recognition, and computer-aided language translation, but also as surrogate models for numerical mathematics. Although the first neural networks were already presented in the 1950s, the scientific discipline has enjoyed increasing popularity mainly during the last 15 years, since only now sufficient computing capacity is available. Remarkably, the increasing availability of computing resources is accompanied by a hunger for larger models, fueled by the common conception of machine learning practitioners and researchers that more trainable parameters equal higher performance and better generalization capabilities. The increase in model size exceeds the growth of available computing resources by orders of magnitude. Since 2012, the computational resources used in the largest neural network models doubled every 3.4 months[1], opposed to Moore's Law that proposes a 2-year doubling period in available computing power.
To some extent, Dirac's statement also applies to the recent computational challenges in the machine-learning community. The desire to evaluate and train on resource-limited devices sparked interest in model compression, where neural networks are sparsified or factorized, typically after training. The second goal of this dissertation is thus a low-rank method, originating from numerical methods for kinetic equations, to compress neural networks already during training by low-rank factorization.

This dissertation thus considers synergies between kinetic models, neural networks, and numerical methods in both disciplines to develop time-, memory- and energy-efficient computational methods for both research areas.

Structure of the Dissertation

The goal of this dissertation is to provide methods for accelerating kinetic and multi-scale simulations using neural networks, as well as neural network compression using kinetic methods. We aim to provide a full method development for each project, with a focus on solid theoretical foundations, efficient numerical methods, and modern open-source software implementation.

After a review of the foundations of kinetic equations, macroscopic discretization methods in §1 and neural networks in §2, we turn our attention to the KiT-RT C++ framework in §3, which acts as the computational backbone for the research conducted in the subsequent chapters. The parallel code contains second-order spatial-temporal discretizations of nodal and modal models for the Boltzmann equation and is excellent for comparing state-of-the-art methods with our newly developed methods.
We focus on the development of neural network-based surrogate models for the minimal entropy closure of the Boltzmann moment system in §4 and §5, where we make use of fast inference of neural networks to accelerate the structurally rich but computationally expensive minimal entropy closure method. The minimal entropy closure is a method to reconstruct the kinetic solution of the Boltzmann equation from its macroscopic variables, the moments of the equation, thus bridging the macroscopic and kinetic resolution.
In the application of multi-scale and non-equilibrium gas flows, we construct a neural network-based flow-regime classifier in §6, to develop a state-of-the-art hybrid simulation that operates on the more accurate kinetic and more efficient macroscopic level, depending on local flow-conditions.

[1] https://openai.com/blog/ai-and-compute/

Finally, we turn our attention to neural networks in §7. We use dynamical, rank-adaptive low-rank compression for dynamical systems, a numerical method that emerged from the computational burden of kinetic and quantum systems and apply it to the gradient flow of neural networks. We develop an efficient dynamical low-rank training algorithm, that significantly reduces memory and wall-time cost of neural network training and inference.
Thus, we inspect synergies between the fields of machine learning, i.e. neural networks, and kinetic modeling, and present multiple opportunities to employ these synergies in the development of efficient computational systems. On the one hand, we develop neural network based surrogate models for minimal entropy closures and neural network based hybrid kinetic solvers. On the other hand we develop an efficient low-rank optimization algorithm for neural networks based on low-rank time integration methods for kinetic equations.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Numerical Methods for the Boltzmann Equation

In this chapter, we introduce the Boltzmann equation as a mathematical model and provide basic concepts of its solution theory and structural properties. We review velocity space discretizations such as discrete ordinates, spherical harmonics, and minimal entropy methods and compare their advantages and disadvantages.

1.1. The Boltzmann Equation

A large variety of physical models are in their core a description of a system of particles interacting with each other and a background medium. Depending on the scale of modeling, a model of the particle system has to provide different values of interest and has different computational requirements.
Most physical simulations contain billions of particles, which have to be simulated. Usually, this renders a fine-grained computation on an individual particle level infeasible. Examples include neutron transport [163], radiative transport [41], semiconductor physics [180] and rarefied gas dynamics [34].
An abstraction of the system of individual particles are mesoscopic modeling approaches. Here, we consider the time-evolution of a probability density $f(t, \mathbf{x}, \mathbf{v})$. The observer is now interested in the probability of a particle traveling with velocity $\mathbf{v} \in \mathbf{V} \subset \mathbb{R}^d$ at time $t \geq 0$ and at place $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^d$, where we have typically $d = 1, 2, 3$. The underlying assumption is the indistinguishability of a great number of particles in the observed system.

Partial differential equations for mesoscopic models are often called kinetic equations and their solution f is called the kinetic density. A well known prototype is the Boltzmann equation

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = Q(f). \tag{1.1}$$

The advection operator $\mathcal{A} = \partial_t + \mathbf{v} \cdot \nabla_{\mathbf{x}}$ describes particle transport, of the particle density with velocity $\mathbf{v}$ in the spatial directions. The integral operator $Q(f)$ models interactions of the particle with the background

medium and collisions with other particles. If the particles only collide with a background material one can model this behavior with the linear Boltzmann collision operator

$$Q(f)(\mathbf{v}) = \int_{\mathbf{V}} k(\mathbf{v}_*, \mathbf{v}) \left[f(\mathbf{v}_*) - f(\mathbf{v}) \right] \, \mathrm{d}\mathbf{v}_*, \tag{1.2}$$

where the collision kernel $k(\mathbf{v}_*, \mathbf{v})$ models the strength of collisions at different velocities. If interactions among particles are considered, the collision operator becomes nonlinear and it reads

$$Q(f, f)(\mathbf{v}) = \int_{\mathbf{V}} \int_{\mathbb{S}^2} k(\mathbf{v}, \mathbf{w}, \mathbf{\Omega}) \left[f(\mathbf{v}_*) f(\mathbf{w}_*) - f(\mathbf{v}) f(\mathbf{w}) \right] \, \mathrm{d}\mathbf{\Omega} \, \mathrm{d}\mathbf{w}. \tag{1.3}$$

The pre- and post-collision velocities are related by the deflection operator

$$[\mathbf{v}_*, \mathbf{w}_*]^\top = T_{\mathbf{\Omega}} \, [\mathbf{v}, \mathbf{w}]^\top , \tag{1.4}$$

where $T_{\mathbf{\Omega}}$ is a reflection matrix. Well-posedness of Eq. (1.1) requires appropriate initial and boundary conditions, i.e.,

$$f(t, \mathbf{x}, \mathbf{v}) = f_0(\mathbf{x}, \mathbf{v}), \qquad t = 0, \mathbf{x} \in \mathbf{X}, \mathbf{v} \in \mathbf{V} \tag{1.5}$$

$$f(t, \mathbf{x}, \mathbf{v}) = f_b(t, \mathbf{v}), \qquad t > 0, \ \mathbf{x} \in \partial\mathbf{X}, \ \mathbf{v} \in \mathbf{V}_{\text{inc}}, \tag{1.6}$$

where

$$\mathbf{V}_{\text{inc}} = \{\mathbf{v} \in \mathbf{V} : \mathbf{v} \cdot \mathbf{n} > 0\} \tag{1.7}$$

are incoming velocity directions and $\mathbf{n} \in \mathbb{R}^d$ is the inward facing normal vector on the boundary of the spatial domain, $\partial\mathbf{X}$. Since the Boltzmann equation is hyperbolic, only incoming characteristics need to be specified. The Boltzmann equation is a first-principles model and can be derived from the individual particle level, for which we refer the reader to [13].

1.1.1. Structural Properties of the Boltzmann Equation

The Boltzmann equation possesses some key structural properties, which are intricately related to the physical processes and its mathematical existence and uniqueness theory. In the following we assume an unbounded domain $\mathbf{X} = \mathbb{R}^d$ and the farfield assumption, i.e. f decreases rapidly,

$$\lim_{\|x\| \to \infty} f(t, \mathbf{x}, \mathbf{v}) = 0, \tag{1.8}$$

to simplify the discussion about the equation's structure. Furthermore, we sometimes use the notation

$$\langle \cdot \rangle = \int_{\mathbf{V}} \cdot \, \mathrm{d}\mathbf{v} \tag{1.9}$$

to define integrals over velocity space for the sake of readability. We state these properties, where we follow [7, 160].

In general, a solution f is expected to be non-negative since it is a probability density, and depending on the application, it has an upper bound, too. The Boltzmann equation obeys the invariance of range.

Theorem 1.1 (Invariance of Range [160])
There is an interval $B \subset [0, \infty)$ representing the physical bounds of f, such that $f(t, \cdot, \cdot) \in B$ for all $t > 0$, if $f(0, \cdot, \cdot) \in B$.

The invariant range property is connected to the l_1 contractivity of the advection operator $\mathcal{A}$ [159] and the diffusive character of the collision operator Q [160].
The Boltzmann equation is a kinetic model for particle systems and obeys classical conservation laws for the system. To inspect conserved quantities of the Boltzmann equation, we define the notion of a system of conservation laws.

Definition 1.2 (System of Conservation Laws [159])
Let $\mathbf{g} : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}^n$ a quantity depending on t and $\mathbf{x}$ and $F_i : \mathbb{R}^n \to \mathbb{R}^n$ for $i = 1, \dots, d$ and $\mathbf{F} = [F_1, \dots, F_d]^\top$. Then,

$$\partial_t \mathbf{g} + \nabla_{\mathbf{x}} \cdot \mathbf{F}(\mathbf{g}) = \partial_t \mathbf{g} + \sum_{i=1}^{d} \partial_{x_i} F_i(\mathbf{g}) = 0 \tag{1.10}$$

is a system of n conservation laws with flux functions F_i and with initial conditions given by $\mathbf{g}_{\mathrm{IC}} = \mathbf{g}(t = 0, \mathbf{x})$. The conserved quantities

$$\int_{\mathbf{X}} \mathbf{g} \, \mathrm{d}\mathbf{x} \tag{1.11}$$

are time independent, and $\mathbf{g}$ is the density of the conserved quantity.

In case of the Boltzmann equation, where the solution f is velocity dependent, a conserved quantity, e.g.,

$$\int_{\mathbf{X}} \int_{\mathbf{V}} f(t, \mathbf{x}, \mathbf{v}) \, \mathrm{d}\mathbf{v} \, \mathrm{d}\mathbf{x}, \tag{1.12}$$

is independent of velocity, too. We expect several conserved quantities in a physical system, e.g. mass, momentum and energy. The question arises, how to identify conserved quantities of the Boltzmann equation. We consider the collision operator Q, and observe, that a collision should not change the total amount of these quantities in the system. Thus they are invariant under the collision operator in a macroscopic sense.

Definition 1.3 (Collision Invariant)
A quantity $\phi(\mathbf{v}) : \mathbf{V} \to \mathbb{R}$ is a collision invariant of Q, if

$$\int_{\mathbf{V}} \phi Q(f, f) \, \mathrm{d}\mathbf{v} = 0 \qquad \forall f \in Dom(Q), \tag{1.13}$$

and we call the linear span of all collision invariants $\mathbb{E}$ and the vector of all collision invariants $\boldsymbol{\phi} \in \mathbb{R}^m$. Here, $Dom(Q)$ denotes the domain of the collision operator Q.

Indeed, each collision invariant has an associated conserved quantity

$$q(t) = \int_{\mathbf{X}} \int_{\mathbf{V}} \phi(\mathbf{v}) f(t, \mathbf{x}, \mathbf{v}) \, \mathrm{d}\mathbf{v} \, \mathrm{d}\mathbf{x} \tag{1.14}$$

with

$$\frac{\mathrm{d}}{\mathrm{d}t}q(t) = 0. \tag{1.15}$$

The statement is straightforward to show by inserting the Boltzmann equation into Eq. (1.14) and using assumption (1.8). Conserved quantities represent the physical laws of mass, momentum, and energy conservation during collisions. There are no other conserved quantities except $\mathbb{E} = \operatorname{span}\left\{1, \mathbf{v}^\top, \left|\mathbf{v}^2\right|/2\right\}$ for non-linear the Boltzmann equation with collision operator $Q(f, f)$ [25, 160].
The Boltzmann equation is an integro-differential equation, and the advection operator $\mathcal{A}$ is a partial differential operator in time and space for each velocity. Classification of the operator $\mathcal{A}$ is of interest to construct suitable numerical methods and to have access to theoretical results and corresponding numerical methods.

Definition 1.4 (Hyperbolic Conservation Law [159])
Consider a system of conservation laws, see Definition 1.2, with flux functions F_i. Consider the matrices

$$A_i(\mathbf{g})_{j,k} = \left(\partial_{u_k} F_i^j(\mathbf{g})\right)_{j,k}. \tag{1.16}$$

The system is called hyperbolic, if for $a_i \neq 0$, $i = 1, \ldots, d$, the matrix

$$A(\mathbf{g}, \mathbf{a}) = \sum_{i=1}^{d} a_i A_i(\mathbf{g}) \tag{1.17}$$

has only real eigenvalues and is diagonalizable. If the eigenvalues are distinct and real, the system of conservation laws is called strictly hyperbolic.

Consequently, for each fixed $\mathbf{v} \in \mathbf{V}$, the advection operator $\mathcal{A}$, yields a scalar hyperbolic conservation law for $t \in [0, \infty)$, $\mathbf{x} \in \mathbf{X}$, i.e.,

$$\partial_t f(t, \mathbf{x}, \cdot) + \mathbf{v} \cdot \partial_{\mathbf{x}} f(t, \mathbf{x}, \cdot) = 0, \tag{1.18}$$

since scalar conservation laws are always strictly hyperbolic [159]. Thus without particle interaction, the system dynamics follow a hyperbolic transport law. This implies the existence of shock solutions of the Boltzmann equation in regions with little to no collisions [159, §3.3], which are not representable by classical solution theory.
To this end, we introduce weak solutions for conservation laws.

Definition 1.5 (Weak Solution of a Hyperbolic Conservation Law)
Let $\varphi \in C_0^\infty(\mathbb{R}_+ \times \mathbb{R}^d)$ be a differentiable test function. Consider a system of conservation laws, see Definition 1.2. The weak form of the conservation law is given by

$$\int_{\mathbb{R}_+ \times \mathbb{R}^d} \mathbf{g}\partial\varphi + \sum_{i=1}^{d} F_i(\mathbf{g})\partial_{x_i}\varphi \,\mathrm{d}\mathbf{x}\,\mathrm{d}t + \int_{\mathbb{R}^d} \mathbf{g}_{\mathrm{IC}}\varphi(0, \mathbf{x}) \,\mathrm{d}\mathbf{x} = 0, \quad \forall \varphi \in C_0^\infty(\mathbb{R}_+ \times \mathbb{R}^d). \tag{1.19}$$

Consequently, a solution does not need to be differentiable in space and time, which is the case in shock solutions. Unfortunately, weak solutions are generally not unique. The idea of entropy solutions is introduced to obtain a unique weak solution.

Definition 1.6 (Entropy/Entropy-Flux Pair [159])
Let $D \in \mathbb{R}^n$ be convex. Then a convex function $h : D \to \mathbb{R}$ is called an entropy for the conservation law (1.10)*, if there exist d functions $j_i : D \to \mathbb{R}$, called entropy fluxes, which fulfill the integrability condition*

$$\nabla_{\mathbf{g}} h(\mathbf{g}) \nabla_{\mathbf{g}} F_i(\mathbf{g}) = \nabla_{\mathbf{g}} j_i(\mathbf{g}), \qquad i = 1, \dots, d, \tag{1.20}$$

For smooth solutions, the integrability condition ensures the conservation of entropy. By multiplication of $\nabla_{\mathbf{g}} h(\mathbf{g})$ from the left to the conservation law (1.10), we get with the integrability condition (1.20)

$$\partial_t h(\mathbf{g}) + \nabla_{\mathbf{x}} \mathbf{j}(\mathbf{g}) = 0. \tag{1.21}$$

An entropy solution, which is a weak solution obeys the entropy dissipation law

$$\int_{\mathbb{R}_+ \times \mathbb{R}^d} h(\mathbf{g}) \partial \varphi + \sum_{i=1}^{d} j_i(\mathbf{g}) \partial_{x_i} \varphi \, \mathrm{d}\mathbf{x} \, \mathrm{d}t + \int_{\mathbb{R}^d} h(\mathbf{g}_{\mathrm{IC}}) \varphi(0, \mathbf{x}) \, \mathrm{d}\mathbf{x} \leq 0, \quad \forall \varphi \in C_0^\infty(\mathbb{R}_+ \times \mathbb{R}^d). \tag{1.22}$$

Let us consider again a weak solution in a situation where no classical solution exists, e.g. a shock and give physical intuition to the entropy dissipation law. To this end, we pick a unique, physically meaningful solution from the set of weak solution candidates. Physical systems include friction, which is typically modeled by a diffusion term and yields the so called viscous solution of a conservation law.

Definition 1.7 (Viscous Solution)
Consider the system of conservation laws (1.10) *with a diffusive right-hand side, i.e.,*

$$\partial_t \mathbf{g}^{(\epsilon)} + \nabla_{\mathbf{x}} \cdot \mathbf{F}(\mathbf{g}^{(\epsilon)}) = \epsilon \Delta_{\mathbf{x}} \mathbf{g}^{(\epsilon)} \tag{1.23}$$

with diffusion coefficient $\epsilon > 0$. The solution $\mathbf{g}^{(\epsilon)}$ is called viscous solution.

We are interested in the behavior of $\mathbf{g}^{(\epsilon)}$ as $\epsilon \to 0$, i.e., when we return to the hyperbolic balance law. Indeed, the limit $\mathbf{g}^{(\epsilon)}$ is provably unique and a weak solution of the conservation law (1.10), see [88]. To this end, multiply $\nabla_{\mathbf{g}} \mathbf{j}(\mathbf{g})$ to the viscous formulation (1.23), which yields

$$\partial_t h(\mathbf{g}^{(\epsilon)}) + \nabla_{\mathbf{x}} \cdot \mathbf{j}(\mathbf{g}) = \epsilon \nabla_{\mathbf{g}} h(\mathbf{g}^{(\epsilon)}) \Delta_{\mathbf{x}} \mathbf{g}^{(\epsilon)}. \tag{1.24}$$

With the equality

$$\nabla_{\mathbf{x}} \left((\nabla_{\mathbf{x}} \mathbf{g})^\top \nabla_{\mathbf{g}} h(\mathbf{g}) \right) = (\nabla_{\mathbf{x}} \mathbf{g})^\top \Delta_{\mathbf{g}} h(\mathbf{g}) \, (\nabla_{\mathbf{x}} \mathbf{g}) + \nabla_{\mathbf{g}} h(\mathbf{g}) \Delta_{\mathbf{x}} \mathbf{g} \tag{1.25}$$

we get the entropy balance equation

$$\partial_t h(\mathbf{g}^{(\epsilon)}) + \nabla_{\mathbf{x}} \left(\mathbf{j}(\mathbf{g}) - \epsilon \nabla_{\mathbf{x}} \left((\nabla_{\mathbf{x}} \mathbf{g})^\top \nabla_{\mathbf{g}} h(\mathbf{g}) \right) \right) = -\epsilon \, (\nabla_{\mathbf{x}} \mathbf{g})^\top \Delta_{\mathbf{g}} h(\mathbf{g}) \, (\nabla_{\mathbf{x}} \mathbf{g}) \, . \tag{1.26}$$

The right-hand side is negative due to the convexity of h and positivity of ϵ, thus the viscous solution dissipates entropy for all $\epsilon > 0$. One can show, that for the limit $\epsilon \to 0$, Eq. (1.26) still holds. There, we arrive at the entropy dissipation law for weak solutions (1.22). Thus, entropy can be seen as a tool to identify a unique weak solution.
We now identify a suitable entropy/entropy-flux pair for the Boltzmann equation and investigate entropy dissipation laws. Since the Boltzmann equation is velocity dependent, we start by defining the kinetic entropy density,

$$\eta : D \subset \mathbb{R} \to \mathbb{R}, \tag{1.27}$$

which we assume to be convex.

Table 1.1.: Entropy densities and their Legendre duals for the Boltzmann equation [7]

Name	D	$\eta(y)$	$\eta'(y)$	$\eta_*(z)$	$\eta'_*(z)$
Maxwell-Boltzmann	$[0,\infty)$	$y\log(y)-y$	$\log(y)$	$\exp(z)$	$\exp(z)$
Bose-Einstein	$[0,\infty)$	$(1+y)\log(1+y)-y\log(y)$	$\log\left(\frac{y}{1+y}\right)$	$-\log(1-\exp(z))$	$\frac{1}{\exp(z)-1}$
Fermi-Dirac	$[0,1]$	$(1-y)\log(1-y)+y\log(y)$	$\log\left(\frac{y}{1-y}\right)$	$\log(1+\exp(z))$	$\frac{1}{\exp(z)+1}$
Quadratic	$\mathbb{R}$	$\frac{1}{2}y^2$	y	$\frac{1}{2}z^2$	z

Theorem 1.8 (H-Theorem for the Linear Collision Operator)
The linear collision operator of the Boltzmann equation Q with positive collision kernel $k > 0$ and the property $k(\mathbf{v},\mathbf{v}_*) = k(\mathbf{v}_*, v)$ fulfills for a convex entropy density η and a kinetic density f the dissipation property

$$\langle \eta'(f)Q(f)\rangle \leq 0, \qquad \forall f \in Dom(Q) \text{ s.t. } Img(f) \subset D, \tag{1.28}$$

where $Img(f)$ denotes the image of f and $Dom(Q)$ denotes the domain of Q.

The proof can be found in §1.5.
Applying the Boltzmann equation to the expression

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbf{X}}\int_{\mathbf{V}} \eta(f)\,\mathrm{d}\mathbf{v}\,\mathrm{d}\mathbf{x} \tag{1.29}$$

and leveraging Theorem 1.8 yields the local dissipation law

$$\partial_t \int_{\mathbf{V}} \eta(f)\,\mathrm{d}\mathbf{v} + \nabla_{\mathbf{x}}\cdot\int_{\mathbf{V}} \mathbf{v}\eta(f)\,\mathrm{d}\mathbf{v} \leq 0, \tag{1.30}$$

for the entropy $h(f) = \langle\eta(f)\rangle$ and entropy-flux $\mathbf{j}(f) = \langle\mathbf{v}\eta(f)\rangle$.
The linear collision operator $Q(f)$ dissipates any convex entropy density, and a variety of choices have been developed depending on the application case, see Table 1.1. Often, D is consistent with the physical bounds B of the scenario, i.e., a modeling decision.

We investigate solutions f of the Boltzmann equation, for which the entropy dissipation law yields equality. Local equilibria of Q are characterized by a vanishing entropy dissipation

$$\langle \eta'(f)Q(f)\rangle = 0. \tag{1.31}$$

A consequence of the H-Theorem is the characterization of the equilibrium state of the Boltzmann equation, where no entropy is dissipated, i.e. ,

$$\langle \eta'(f)Q(f)\rangle = 0 \qquad \Leftrightarrow \qquad Q(f) = 0 \qquad \Leftrightarrow \qquad \eta'(f) \in \mathbb{E}.$$

Lastly, let us consider the collision operator Q and advection operator $\mathcal{A}$ of the Boltzmann equation under translations and rotations in the velocity domain.

Theorem 1.9 (Gallilean Invariance [126])
For a kinetic density f, an orthogonal matrix $R \in \mathbb{R}^{d\times d}$ and a translation vector $\mathbf{w} \in \mathbf{V}$ in the velocity domain, a Galilean transformation is given by

$$G_{R,\mathbf{w}} f(t,\mathbf{x},\mathbf{v}) = f(t, R(\mathbf{x}-t\mathbf{w}), R(\mathbf{v}-t\mathbf{w})), \tag{1.32}$$

and we have for the advection and collision operator of the Boltzmann equation

$$\mathcal{A}(G_{R,\mathbf{w}}f) = G_{R,\mathbf{w}}\mathcal{A}f, \qquad \forall f \in Dom(\mathcal{A}), \tag{1.33}$$
$$Q(G_{R,\mathbf{w}}f) = G_{R,\mathbf{w}}Qf, \qquad \forall f \in Dom(Q). \tag{1.34}$$

We refer for the proof to literature [126]. Consequently, a transformed kinetic density $G_{R,\mathbf{w}}f$ of a solution f to the Boltzmann equation is again a solution. Furthermore, a solution f is invariant under translations of space $\mathbf{x}$ and time t. The commutation relation of Q and $\mathcal{A}$ reflects the Galilean invariance of collisions and transport at a particle level.

1.2. Velocity Space Discretizations for the Boltzmann Equation

The structural richness of the Boltzmann equation is a result of its first principles-based derivation. Although an abstraction and thus a simplification of the individual particle level, the high-dimensional phase-space $\mathbf{X} \times \mathbf{V}$ of the Boltzmann equation yields tremendous challenges for discretization methods. A discretization method or surrogate model should therefore have two (often contradictory) goals: computational efficiency and preservation of the underlying equations' intrinsic structures. In the following, we review the most relevant models to reduce the phase-space dimensionality of the Boltzmann equation for this work. We consider the linear Boltzmann equation with $Q(f)$ and the unit sphere as directional domain $\mathbf{V} = \mathbb{S}^2 \subset \mathbb{R}^3$ in slab geometry [162], where the velocity space $\mathbf{V}$ is projected onto $\mathbb{R}^2$.

1.2.1. Nodal Discretizations - the S_N Method

A popular strategy is to discretize the velocity space in a point-wise manner, following the idea of hyperbolicity of the Boltzmann equation at fixed velocities [31, 163, 193] . The discrete ordinates, i.e. S_N, method [164] employs a nodal discretization for the directional domain. To facilitate the computation of integral terms that arise due to scattering, the nodal point sets are commonly chosen according to a quadrature rule. In the application case of radiative transport, the directional domain is assumed to be the unit sphere $\mathbb{S}^2$, thus a suitable parametrization is given by spherical coordinates

$$\mathbb{S}^2 = \left\{ \begin{bmatrix} \sqrt{1-\mu^2}\sin(\theta), & \sqrt{1-\mu^2}\cos(\theta), & \mu \end{bmatrix}^\top : \mu \in [-1,1]\,,\ \theta \in [0,2\pi) \right\}. \tag{1.35}$$

Note, that we can allow different particle velocities by scaling the unit sphere with a given maximum velocity. In the slab geometry setting, lower dimensional velocity spaces are described by a projection of $\mathbb{S}^2$ onto $\mathbb{R}^2$ and $\mathbb{R}$, respectively, i.e.,

$$P_{\mathbb{R}^2}(\mathbb{S}^2) = \left\{ \begin{bmatrix} \sqrt{1-\mu^2}\sin(\theta), & \sqrt{1-\mu^2}\cos(\theta) \end{bmatrix}^\top : \mu \in [0,1]\,,\ \theta \in [0,2\pi) \right\} \tag{1.36}$$

and

$$P_{\mathbb{R}}(\mathbb{S}^2) = \{\mu : \mu \in [-1,1]\} \tag{1.37}$$

Hence the task is to derive a quadrature formula for the direction of travel. A common approach is the product quadrature rule. Here, a Gauss quadrature is used for μ and equally weighted and spaced points for θ, i.e., for N_q points, we have

$$\theta_i = i\Delta\theta \quad \text{for} \quad i = 1, \dots, N_q \quad \text{and} \quad \Delta\theta = \frac{2\pi}{N_q} \,. \tag{1.38}$$

If the Gauss quadrature for μ uses N_q points, then we obtain a total of $Q = N_q^2$ possible directions. Denoting the Gauss weights as w_k^G with $k = 1, \dots, N_q$, we obtain the product quadrature weights

$$w_{k \cdot N_q + l} = \frac{2\pi w_k^G}{N_q}$$

and points

$$\mathbf{v}_{k \cdot N_q + l} = \left[\sqrt{1 - \mu_k^2} \sin(\theta_l),\ \sqrt{1 - \mu_k^2} \cos(\theta_l) \right]^\top . \tag{1.39}$$

A variety of different quadrature rules have been derived for specialized S_N application scenarios. A comparison of different quadrature sets and their approximation behavior for S_N methods can be found in [32].
Application of the S_N method to the Boltzmann equation (1.1) yields a system of evolution equations for $f_q(t, \mathbf{x}) := f(t, \mathbf{x}, \mathbf{v}_q)$, i.e.,

$$\partial_t f_q(t, \mathbf{x}) + \mathbf{v}_q \cdot \nabla_{\mathbf{x}} f_q(t, \mathbf{x}) = \sum_{p=1}^{Q} w_p k(\mathbf{v}_p, \mathbf{v}_q) \left[f_p(t, \mathbf{x}) - f_q(t, \mathbf{x}) \right]. \tag{1.40}$$

The hyperbolicity of the Boltzmann equation yields directly that Eq. (1.40) is a system of hyperbolic conservation laws. The equation system is only coupled via the discretized scattering operator.
The S_N method turns out to be computationally highly efficient, although at the cost of a large memory footprint. For accurate solutions, a high order S_N method is necessary and the amount of discretization points N_q grows quadratically in the quadrature order. Furthermore, the solution can exhibit non-physical artifacts, known as ray effects [153, 182, 188], which reduce the approximation quality. While methods to mitigate ray effects exist, see e.g. [3, 31, 75, 154, 221], they commonly require picking problem-dependent tuning parameters.

1.2.2. Modal Discretizations - The Boltzmann Moment System

Modal, i.e., moment methods encode the velocity dependence of the kinetic density f in a basis $\mathbf{m}(\mathbf{v}) \in \mathbb{R}^n$ of $\mathbf{V}$ [160] and enjoy broad scientific and application interest [7, 8, 82, 141, 161]. The basis typically consists of polynomials up to order N and contains the collision invariants $\boldsymbol{\phi}$ of the Boltzmann Equation. Common choices for the basis functions are monomials or spherical harmonics, depending on the application. In one spatial dimension, usually we have $n = N + 1$, whereas in higher spatial dimensions n equals the number of basis functions up to order N. Subsequent integration over $\mathbf{V}$ yields the moment vector $\mathbf{u} \in \mathbb{R}^n$

$$\mathbf{u}(t, \mathbf{x}) = \int_{\mathbf{V}} \mathbf{m}(\mathbf{v}) f(t, \mathbf{x}, \mathbf{v}) \, \mathrm{d}\mathbf{v}. \tag{1.41}$$

Analogously, testing the Boltzmann equation against $\mathbf{m}$ yields the Boltzmann moment system

$$\partial_t \mathbf{u}(t, \mathbf{x}) + \nabla_{\mathbf{x}} \cdot \langle \mathbf{v} \otimes \mathbf{m}(\mathbf{v}) f \rangle = \langle \mathbf{m}(\mathbf{v}) Q(f) \rangle \,, \tag{1.42}$$

whose solution is the moment vector $\mathbf{u}$. By construction, the advection and collision operators still depend on f, and thus the moment system is unclosed. Moment methods aim to find a meaningful closure relation $\mathcal{U}$ for this system [160, 161], i.e.,

$$f_{\mathbf{u}}(t, \mathbf{x}, \mathbf{v}) \simeq \mathcal{U}(u_0(t, \mathbf{x}), \dots, u_n(t, \mathbf{x})) \, . \tag{1.43}$$

Remark, that the closure of the moment system is a modeling choice and has an impact on the structural properties of the resulting closed moment system. In the following, we present the well-known M_N and P_N closure frequently used in the field of radiation transport.

1.2.3. Bases of the Velocity Space

Application requirements drive the choice of the velocity basis $\mathbf{m}(\mathbf{v})$. For radiation transport, we have $\mathbf{V} = \mathbb{S}^2$, which we parametrize in spherical coordinates.

Spherical harmonics basis

This particular choice of $\mathbf{V}$ motivates the usage of real-valued spherical harmonics as basis functions. These are defined as

$$Y_l^k(\mathbf{v}) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-k)!}{(l+k)!}}\, e^{ik\theta} P_l^k(\mu), \quad \forall l \geq 0 \tag{1.44}$$

where P_l^k is the associated Legendre polynomial of degree l and order k, where l and k are integers with $0 \leq k \leq l$. Furthermore, μ and ϕ are the polar coordinates parametrization of $\mathbb{S}^2$. The associated Legendre polynomials can be computed using a recursion relation [167]. Then, the real spherical harmonics are given as

$$m_l^k(\mathbf{v}) = \begin{cases} \frac{(-1)^k}{\sqrt{2}}\left(Y_l^k(\mathbf{v}) + (-1)^k Y_l^{-k}(\mathbf{v})\right), & k > 0 \, , \\ Y_l^0(\mathbf{v}) & k = 0 \, , \\ -\frac{(-1)^k i}{\sqrt{2}}\left(Y_l^{-k}(\mathbf{v}) - (-1)^k Y_l^k(\mathbf{v})\right), & k < 0 \, , \end{cases}$$

where i is the imaginary unit, and $-l \leq k \leq l$. Collecting all basis functions up to degree N in a vector

$$\mathbf{m}(\mathbf{v}) = \left[m_0^0(\mathbf{v}), m_1^{-1}(\mathbf{v}), m_1^0(\mathbf{v}), m_1^1(\mathbf{v}), \dots, m_N^N(\mathbf{v})\right]^T \in \mathbb{R}^n$$

yields the spherical harmonics basis of order N, where $n = (N+1)^2$, see Fig. 1.1. The spherical harmonics yield an orthonormal basis of $\mathbb{S}^2$ and possess symmetry and recursion properties, which makes them an attractive basis choice for applications with bounded velocity domains.

Monomial basis

Monomial and polynomialvelocity bases play an important role in the derivation of macroscopic models for gas dynamic, where they appear in Grad's 13 moment system [128, 218] and in the derivation of

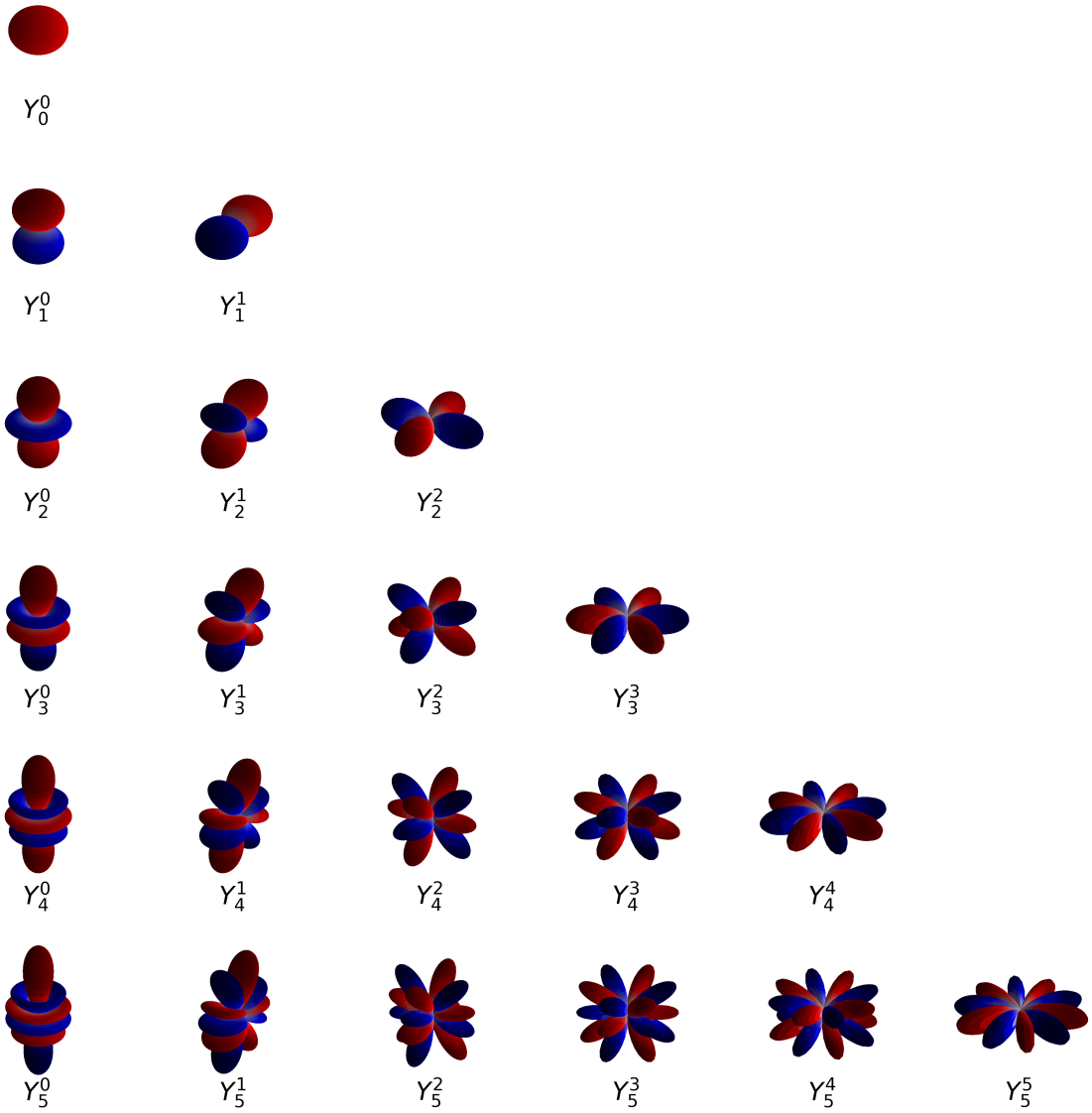


Figure 1.1.: Real valued spherical harmonics of non-negative order up to degree 4 in 3 velocity dimensions. Blue denotes negative and red positive values of the basis functions.

the Navier-Stokes and Euler equations [14, 15]. Monomial basis functions are defined in a tensorized manner

$$m_i(\mathbf{v}) = \otimes_{k=1}^{i} \mathbf{v}, \qquad i = 1, \dots l, \tag{1.45}$$

$$m_0(\mathbf{v}) = 1. \tag{1.46}$$

Then m_i is a d-dimensional l-tensor, containing mixed monomials with multi-index $\mathbf{k} = (k_1, \dots, k_l)$ of order $|\mathbf{k}| \leq l$, i.e., for $d = 3$,

$$m_l(\mathbf{v})_{k_1,\dots,k_l} = v_1^{\sum_i k_i \delta_{k,1}} v_2^{\sum_i k_i \delta_{k,2}} v_3^{\sum_i k_i \delta_{k,3}}, \tag{1.47}$$

where $\delta_{k,d}$ is the Kronecker delta. By definition, elements of m_i with permuted multi-index are equivalent. Elimination of equivalent entries and flattening of the tensors m_i yields

$$\mathbf{m}_i(\mathbf{v}) = \left[v_1^i, v_2^i, v_3^i, v_1^{(i-1)}v_2, \ldots, v_2^{(i-1)}v_3\right]^\top \tag{1.48}$$

and concatenation of $\mathbf{m}_i$ yields the monomial basis in vector form

$$\mathbf{m}(\mathbf{v}) = \left[\mathbf{m}_0(\mathbf{v})^\top, \ldots, \mathbf{m}_N(\mathbf{v})^\top\right]^\top . \tag{1.49}$$

1.3. Minimal Entropy Closures - the M_N Method

Entropy-based moment closures [60, 162] use the concept of entropy to close the hierarchy of moment equations. In general, entropy-based moment closures provide a way to accurately capture the macroscopic behavior of a gas or other many-particle system, while still taking into account the underlying microscopic dynamics.
The Boltzmann equation dissipates entropy and fulfills a local entropy dissipation law. Therefore, a suitable closure can be created by choosing the reconstructed kinetic density $f_\mathbf{u} = f_\mathbf{u}(\cdot, \cdot, \mathbf{v})$ out of the set

$$F_\mathbf{m} = \{f \in \mathrm{Dom}(Q) : \mathrm{Img}(f) \subset D \text{ and } \langle \mathbf{m} f \rangle < \infty\} \tag{1.50}$$

of all possible functions that fulfill

$$\mathbf{u}(t, \mathbf{x}) = \langle \mathbf{m} g \rangle , \tag{1.51}$$

as the one with minimal entropy, with entropy density η (1.27). The minimal entropy closure, dubbed M_N method, can be formulated as a constrained optimization problem for a given vector of moments $\mathbf{u}$, i.e.,

$$\min_{g \in F_\mathbf{m}} \langle \eta(g) \rangle \qquad \text{s.t. } \mathbf{u} = \langle \mathbf{m} g \rangle \ . \tag{1.52}$$

We define

$$h(\mathbf{u}) := \langle \eta(f_\mathbf{u}) \rangle , \tag{1.53}$$

where $f_\mathbf{u}$ is the minimizer of Eq. (1.52), which we use to close the moment system

$$\partial_t \mathbf{u}(t, \mathbf{x}) + \nabla_\mathbf{x} \cdot \langle \mathbf{v} \otimes \mathbf{m}(\mathbf{v}) f_\mathbf{u} \rangle = \langle \mathbf{m}(\mathbf{v}) Q(f_\mathbf{u}) \rangle . \tag{1.54}$$

The set of all moments corresponding to a kinetic density f with $\mathrm{Img}(f) \subset D$ is called the realizable set

$$\mathcal{R} = \{\mathbf{u} : \langle \mathbf{m} g \rangle = \mathbf{u},\ g \in F_\mathbf{m}\} . \tag{1.55}$$

The realizable set $\mathcal{R}$ is only dependent on the domain D of η, $\mathbf{V}$ and $\mathbf{m}$, and can be described as the set of all moments corresponding to kinetic densities f that fulfill the invariant range condition of the Boltzmann equation. The realizable set $\mathcal{R}$ of the minimal entropy closure has been thoroughly analyzed for the monomial basis [52, 187].
We derive an analytic expression for the minimizer of Eq. (1.52) by considering its dual optimization problem, which we construct through the Legendre transform.

Definition 1.10 (Legendre transform)
Let $g : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Its Legendre transform $g_* : \mathbb{R}^n \to \mathbb{R}$ is defined as,

$$g_*(\mathbf{y}) = \sup_{\mathbf{z}\in\mathbb{R}^n} \{\mathbf{y}\cdot\mathbf{z} - g(\mathbf{z})\}. \tag{1.56}$$

The Legendre transform has the properties

$$(g_*)_* = g \quad \textit{and} \quad g'_* = (g')^{-1}. \tag{1.57}$$

Theorem 1.11 (Dual minimal entropy closure [160])
The dual of the minimal entropy closure optimization problem 1.52 is itself a convex optimization problem and is given by

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \phi(\boldsymbol{\alpha};\mathbf{u}), \tag{1.58}$$

$$\phi(\boldsymbol{\alpha};\mathbf{u}) = \boldsymbol{\alpha}\cdot\mathbf{u} - \langle\eta_*(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle, \tag{1.59}$$

for a moment vector $\mathbf{u}$. The Legendre dual of the entropy density η is denoted by η_*. The maximizer

$$\boldsymbol{\alpha}_\mathbf{u} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{argmax}\, \phi(\boldsymbol{\alpha};\mathbf{u}) \tag{1.60}$$

is the Lagrange multiplier for moment $\mathbf{u}$. At the optimal point, we have

$$h(\mathbf{u}) = \phi(\boldsymbol{\alpha}_\mathbf{u};\mathbf{u}). \tag{1.61}$$

The proof is provided in §1.5. There does not always exists a solution for the minimal entropy problem [100]. However, if a solution exists for $\mathbf{u} \in \mathcal{R}$, it is unique and of the form

$$f_\mathbf{u}(\mathbf{v}) = \eta'_*(\boldsymbol{\alpha}_\mathbf{u}\cdot\mathbf{m}(\mathbf{v})). \tag{1.62}$$

The optimality and uniqueness of the primal solution $f_\mathbf{u}$ in (1.62) is then confirmed by using the strong duality of (1.52) and (1.58). In particular, the first-order optimality condition of (1.58) leads to

$$\mathbf{u} = \langle\mathbf{m}\eta'_*(\boldsymbol{\alpha}_\mathbf{u}\cdot\mathbf{m})\rangle, \tag{1.63}$$

which yields the inverse of the solution map $\mathbf{u} \mapsto \boldsymbol{\alpha}_\mathbf{u}$ of the dual optimization problem. Furthermore, the derivative of h recovers the optimal Lagrange multipliers of Eq. (1.58), i.e.,

$$\nabla_\mathbf{u} h(\mathbf{u}) = \boldsymbol{\alpha}_\mathbf{u}, \tag{1.64}$$

and the Hessian of the dual objective function ϕ with respect to $\boldsymbol{\alpha}$ is given by

$$H(\boldsymbol{\alpha}) = -\langle\mathbf{m}\otimes\mathbf{m}\eta''_*(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle. \tag{1.65}$$

Additionally we introduce $H_n(\boldsymbol{\alpha}) = -H(\boldsymbol{\alpha})$.

Structural Properties of the Minimal Entropy Closure

The main advantage of the M_N method as a closure for the Boltzmann moment system is its preservation property of the intrinsic structure of the Boltzmann equation itself. The structural properties have been proven in [160, 161], and we review them in the following.
The range of the reconstructed entropy density $f_{\mathbf{u}}$ is a consequence of the ansatz, see Eq. (1.62) and thus directly connected to the domain D of the entropy density η, see Table 1.1. In terms of the feasibility of the moment constraint, the invariance of the range of $f_{\mathbf{u}}$ is equivalent to the realizability of $\mathbf{u}$. Formally, we expect, that $\mathbf{u} \in \mathcal{R}$ for all $t > 0$, if $\mathbf{u}(t_0) \in \mathcal{R}$. Furthermore, if $B = D$, the physical bounds of the problem are consistent with the bounds of the reconstructed kinetic density.

We look at conservation properties of the moment system.

Corollary 1.12
Let a basis function $m_i \in \mathbb{E}$ be a collision invariant, then by Eq. (1.13)*, the i-th component of the moment system* (1.42)*, i.e.,*

$$\partial_t u_i + \nabla_{\mathbf{x}} \langle m_i \mathbf{v} f_{\mathbf{u}} \rangle = 0 \tag{1.66}$$

is a conservation law.

The proof follows directly using Definition 1.3. Consequently, the choice of the basis $\mathbf{m}$ determines, if Eq. (1.42) is a system of conservation laws.

Hyperbolicity of the moment system was originally shown in [160].

Theorem 1.13 (Hyperbolicity of the Moment System [160])
The advection operator of the moment system with minimal entropy closure (1.54) *is hyperbolic.*

The proof is provided in §1.5. Remark, that the crucial ingredient of the proof is the convexity of the minimal entropy functional.

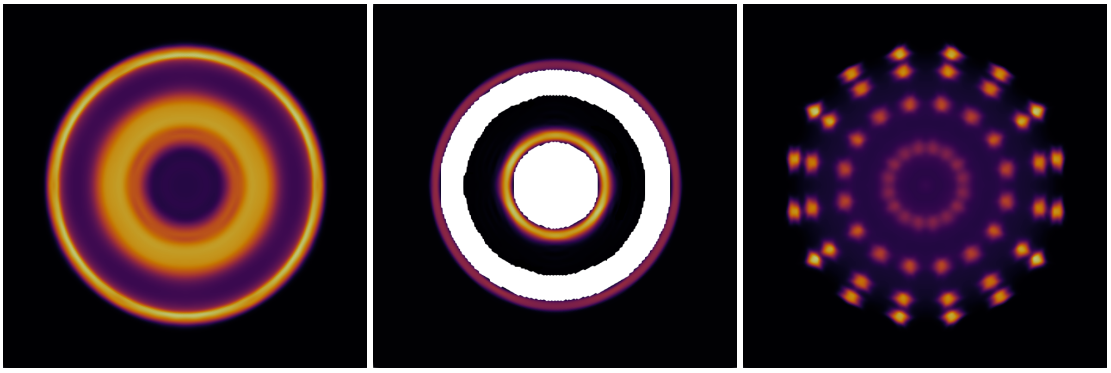
Consider again a Galilean transformation $G_{R,\mathbf{w}}$ defined in Eq. (1.32) and a moment basis $\mathbf{m}$. We define the inherited Galilean transform as

$$T_{R,\mathbf{w}}\mathbf{u} = \langle \mathbf{m} G_{R,\mathbf{w}} f_{\mathbf{u}} \rangle \tag{1.67}$$

Theorem 1.14 (Galilean Invariance of the Moment System [126])
The Boltzmann moment system (1.42) *is invariant under the inherited transform if the moment basis* $\mathbf{m}$ *is invariant under* $G_{R,\mathbf{w}}$.

The proof is provided by [126]. Specifically, this implies the need for an orthonormal basis $\mathbf{m}$ of the velocity space $\mathbf{V}$. The spherical harmonics basis is an example for an orthonormal basis.

Lastly, we discuss the inheritance of entropy-dissipation for the Boltzmann moment system



(a) M_3 simulation (b) P_3 simulation (c) S_8 simulation

Figure 1.2.: Simulation of the Linesource test case using different velocity space, but the same spatial-temporal discretizations. The plot shows the scalar flux, i.e., u_0. Black cells are zero-valued and white cells denote negative values. Notice the ray effects of the S_8 method, and the negative solution of the P_3 method.

Theorem 1.15 (Entropy Dissipation of the Moment System)
The objective functional of the minimal entropy closure $h(\mathbf{u})$, see (1.53) *and the function*

$$\mathbf{j}(\mathbf{u}) = [j(\mathbf{u})_1, \dots, j(\mathbf{u})_d]^\top, \text{ with } j(\mathbf{u})_i = \langle v_i \eta(f_{\mathbf{u}}) \rangle \tag{1.68}$$

is a suitable entropy/entropy-flux pair for the Boltzmann moment system with minimal entropy closure. For a solution $\mathbf{u}$ of the moment system, the entropy dissipation law

$$\partial_t h(\mathbf{u}) + \nabla_{\mathbf{u}} \cdot \mathbf{j}(\mathbf{u}) \leq 0 \tag{1.69}$$

holds.

The proof is provided in §1.5. Note, that the convexity of h is a core ingredient for the proof.
We compare Theorem 1.15 with the H-Theorem 1.8 and the entropy dissipation law for the Boltzmann equation (1.30) and identify the minimum of the entropy closure (1.53) with the entropy of the Boltzmann equation.

The H-theorem for the Boltzmann equation can be used to show the equivalence of the statements

$$\alpha_{\mathbf{u}}(\mathbf{u}) \cdot \langle \mathbf{m} Q(f_{\mathbf{u}}) \rangle = 0 \quad \Leftrightarrow \quad Q(f_{\mathbf{u}}) = 0 \quad \Leftrightarrow \quad \alpha_{\mathbf{u}} \cdot \mathbf{m} \in \mathbb{E}. \tag{1.70}$$

Here, $\mathbb{E}$ denotes the span of collision invariants. Overall, the M_N method preserves the positivity of particle distributions while yielding accurate results with little spurious oscillations and is structurally rich. However, the need to solve a possibly ill-posed optimization problem in every spatial cell and time step results in increased computational costs. While analytic solutions to the optimization problems are available at small truncation orders, they cannot capture all physical effects accurately.

1.4. Spherical Harmonics Closure - the P_N Method

A commonly used closure $\mathcal{U}(u_0(t, \mathbf{x}), \cdots, u_n(t, \mathbf{x}))$ is the so-called P_N closure [33]. In terms of minimal entropy closures, the P_N closure denotes the special case of a quadratic entropy density $\eta(y) = y^2$ and the

spherical harmonics basis on the unit sphere $\mathbb{S}^2$. The reconstruction formula of Eq. (1.62)

$$f_{\mathbf{u}}(\mathbf{v}) = \eta'_*(\boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{m}) = \boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{m} \tag{1.71}$$

and the definition of the moment vector $\mathbf{u}$ yield the simplification

$$\mathbf{u} = \langle \mathbf{m} \boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{m} \rangle \quad \Rightarrow \quad \boldsymbol{\alpha}_{\mathbf{u}} = \langle \mathbf{m} \otimes \mathbf{m} \rangle^{-1} \mathbf{u}. \tag{1.72}$$

Orthonormality of the spherical harmonics basis results in $\langle \mathbf{m} \otimes \mathbf{m} \rangle = I$ and we get $\boldsymbol{\alpha}_{\mathbf{u}} = \mathbf{u}$. Thus the P_N method expands the solution by spherical harmonics, i.e.,

$$f_{\mathbf{u}}(\mathbf{v}) = \mathbf{u} \cdot \mathbf{m}(\mathbf{v}) \tag{1.73}$$

where $\mathbf{u} \in \mathbb{R}^{(N+1)^2}$ collects all moments according to $\mathbf{u} = \left(u_0^0, u_1^{-1}, u_1^0, u_1^1, \cdots, u_N^N\right)^T \in \mathbb{R}^{(N+1)^2}$.
Then, the moment equations read

$$\partial_t \mathbf{u}(t, \mathbf{x}) + \mathbf{A} \cdot \nabla_{\mathbf{x}} \mathbf{u}(t, \mathbf{x}) = G\mathbf{u}(t, \mathbf{x}) \tag{1.74}$$

where G is the discretization of the linear collision operator Q, where we use the fact that the spherical harmonics are the eigenfunctions of Q. Further, the advection operator is given by

$$\mathbf{A} \cdot \nabla_{\mathbf{x}} := A_1 \partial_{x_1} + A_2 \partial_{x_2} + A_3 \partial_{x_3} \tag{1.75}$$

with the flux Jacobians A_i given by

$$A_i := \langle \mathbf{m} \otimes \mathbf{m} v_i \rangle \,. \tag{1.76}$$

This expression can be implemented efficiently using the recursion relation of the spherical harmonics

$$v_i \mathbf{m}_l = a_l^i \mathbf{m}_{l-1} + a_{l+1}^i \mathbf{m}_{l+1} \tag{1.77}$$

with the submatrices $a_l^i \in \mathbb{R}^{(2l-1)\times(2l+1)}$ and $\mathbf{m}_{l-1} \in \mathbb{R}^{(2l-1)}$. The submatrices a_l^i are given by

$$\mathbf{v} m_l^k = \frac{1}{2} \begin{pmatrix} (1-\delta_{k,-1}) \left(\tilde{c}_{l-1}^{|k|-1} m_{l-1}^{k^-} - \tilde{d}_{l+1}^{|k|-1} m_{l+1}^{k^-} \right) - \tilde{e}_{l+1}^{|k|+1} m_{l-1}^{k^+} + \tilde{f}_{l+1}^{|k|+1} m_{l+1}^{k^+} \\ \operatorname{sign}(k) \left((1-\delta_{k,1}) \left(-\tilde{c}_{l-1}^{|k|-1} m_{l-1}^{-k^-} + \tilde{d}_{l+1}^{|k|-1} m_{l+1}^{-k^-} \right) - \tilde{e}_{l-1}^{|k|+1} m_{l-1}^{-k^+} + \tilde{f}_{l+1}^{|k|+1} m_{l+1}^{-k^+} \right) \\ 2 \left(a_{l-1}^k m_{l-1}^k + b_{l+1}^k m_{l+1}^k \right) \end{pmatrix}, \tag{1.78}$$

where we modify the sign notation such that $\operatorname{sign}(0) = 1$ holds. With $k^\pm = k \pm \operatorname{sign}(k)$, and the coefficients are

$$\tilde{c}_l^k = \begin{cases} 0, & k < 0, \\ \sqrt{2} c_l^k, & k = 0, \\ c_l^k, & k > 0, \end{cases} \quad , \quad \tilde{d}_l^k = \begin{cases} 0, & k < 0, \\ \sqrt{2} d_l^k, & k = 0, \\ d_l^k, & k > 0, \end{cases} \tag{1.79}$$

$$\tilde{e}_l^k = \begin{cases} \sqrt{2} e_l^k, & k = 1, \\ e_l^k, & k > 1, \end{cases} \quad , \quad \tilde{f}_l^k = \begin{cases} \sqrt{2} f_l^k, & k = 1, \\ f_l^k, & k > 1, \end{cases} \tag{1.80}$$

and

$$a_l^k = \sqrt{\frac{(l-k+1)(l+k+1)}{(2l+3)(2l+1)}}, \qquad b_l^k = \sqrt{\frac{(l-k)(l+k)}{(2l+1)(2l-1)}}, \tag{1.81}$$

$$c_l^k = \sqrt{\frac{(l+k+1)(l+k+2)}{(2l+3)(2l+1)}}, \qquad d_l^k = \sqrt{\frac{(l-k)(l-k-1)}{(2l+1)(2l-1)}}, \tag{1.82}$$

$$e_l^k = \sqrt{\frac{(l-k+1)(l-k+2)}{(2l+3)(2l+1)}}, \qquad f_l^k = \sqrt{\frac{(l+k)(l+k-1)}{(2l+1)(2l-1)}}. \tag{1.83}$$

Furthermore, one can use the fact that the spherical harmonics are the eigenfunctions of the scattering part of the linear collision operator. Thus the right-hand side of the moment system can be expressed by a matrix multiplication with $G\mathbf{u}$ consisting of the eigenvalues of the spherical harmonics basis.
Then, the moment equations at degree l become

$$\partial_t \mathbf{u}_l(t,\mathbf{x}) + \sum_{i=1}^{3} \partial_{x_i} \left(\mathbf{a}_l^i \mathbf{u}_{l-1}(t,\mathbf{x}) + \mathbf{a}_{l+1}^i \mathbf{u}_{l+1}(t,\mathbf{x}) \right) = G\mathbf{u}_l(t,\mathbf{x}) \tag{1.84}$$

$$u_l = 0, \qquad \forall l > (N+1)^2 . \tag{1.85}$$

Note that the equations for degree l depend on the moments of degree $l+1$, and according to the minimal entropy closure with quadratic entropy, higher-order moments are simply truncated.
While P_N is a computationally efficient method (especially for scattering terms), it does not preserve positivity. One can see this by considering the domain D of the quadratic entropy density η, which is $\mathbb{R}$, see Table 1.1. Therefore, negative densities are feasible for the minimal entropy closure with this choice of entropy. This leads to simulation artifacts like spurious oscillations and even results in negative particle concentrations, especially in regimes with few collisions [30].

1.5. Additional Material

In the following, we prove the stated theorems and lemmas of this chapter.

Statements of Section 1.1

Proof: (Theorem 1.8) If η is convex, then η' is non-decreasing. Consequently, we have

$$\begin{aligned}\langle \eta'(F)Q(f)\rangle &= \int_V \int_V \eta'(f(\mathbf{v}))k(\mathbf{v}_*,\mathbf{v})(f(\mathbf{v}_*)-f(\mathbf{v}))\,\mathrm{d}\mathbf{v}_*\,\mathrm{d}\mathbf{v}\\ &= \int_V \int_V \eta'(f(\mathbf{v}_*))k(\mathbf{v},\mathbf{v}_*)(f(\mathbf{v})-f(\mathbf{v}_*))\,\mathrm{d}\mathbf{v}\,\mathrm{d}\mathbf{v}_*\\ &= \frac{1}{2}\int_V \int_V (\eta'(f(\mathbf{v}))-\eta'(f(\mathbf{v}_*))k(\mathbf{v},\mathbf{v}_*)(f(\mathbf{v}_*)-f(\mathbf{v}))\,\mathrm{d}\mathbf{v}\,\mathrm{d}\mathbf{v}_* \le 0,\end{aligned} \tag{1.86}$$

where we use the positiviy of the collision kernel k, as well as $k(\mathbf{v},\mathbf{v}_*) = k(\mathbf{v}_*,\mathbf{v})$. □

Statements of Section 1.3

Proof: (Theorem 1.11) We find the dual of (1.52) by consideration of the Lagrangian

$$L(g,\boldsymbol{\alpha}) = \langle \eta(g)\rangle + \boldsymbol{\alpha}\cdot(\mathbf{u}-\langle \mathbf{m}g\rangle)\,. \tag{1.87}$$

Formally minimizing L over g yields

$$\begin{aligned}\min_g L(g,\boldsymbol{\alpha}) &= \min_g \{\langle \eta(g)\rangle + \boldsymbol{\alpha}\cdot(\mathbf{u}-\langle \mathbf{m}g\rangle)\}\\ &= \min_g \{\langle \eta(g)-\boldsymbol{\alpha}\cdot\mathbf{m}g\rangle\} + \boldsymbol{\alpha}\cdot\mathbf{u}\\ &= -\langle \eta_*(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle + \boldsymbol{\alpha}\cdot\mathbf{u} \qquad \text{(Legendre duality)}\\ &= \phi(\boldsymbol{\alpha};\mathbf{u}).\end{aligned} \tag{1.88}$$

By strong duality, the maximum of (1.58) equals the minimum of (1.52), so we have

$$h(\mathbf{u}) = \phi(\boldsymbol{\alpha}_\mathbf{u};\mathbf{u}), \tag{1.89}$$

where $\boldsymbol{\alpha}_\mathbf{u}$ is the maximizer of the dual problem. □

Proof: (Theorem 1.13) Consider the flux functions of the Boltzmann moment system (1.42) in the sense of Definition 1.2, i.e.,

$$F_i(\mathbf{u}) = \left\langle v_i \mathbf{m}\eta_*'(\boldsymbol{\alpha}_\mathbf{u}\cdot\mathbf{m})\right\rangle, \qquad i=1,\dots,d. \tag{1.90}$$

We define the function $j_{i,*}(\boldsymbol{\alpha}) = \langle v_i\eta_*(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle$ and notice that $\nabla_{\boldsymbol{\alpha}} j_{i,*}(\boldsymbol{\alpha}_\mathbf{u}) = F_i(\mathbf{u})$. Furthermore, we have

$$\nabla_{\boldsymbol{\alpha}_\mathbf{u}}\mathbf{u} = H_n(\boldsymbol{\alpha}_\mathbf{u}), \tag{1.91}$$

by definition of the Hessian of the dual entropy functional and optimality conditions. Thus, by the chain rule, we can reformulate Eq. (1.54) to

$$\partial_t \mathbf{u} + \sum_{i=1}^{d} J_i H_n^{-1} \partial_{x_i} \mathbf{u} = 0, \tag{1.92}$$

with

$$J_i = j_{i,*}''(\boldsymbol{\alpha}_\mathbf{u}) = \left\langle v_i \mathbf{m} \otimes \mathbf{m} \eta_*''(\boldsymbol{\alpha}_\mathbf{u} \cdot \mathbf{m}) \right\rangle. \tag{1.93}$$

We show, that $J_i H_n^{-1}$ is diagonalizable with real eigenvalues. First, we note, that H_n is symmetric and positive definite since it is the hessian of $-\phi$, which is convex. Thus we can write $H_n^{1/2} H_n^{1/2} = H_n$ and we have

$$J H_n^{-1} = H_n^{1/2} H_n^{-1/2} J H_n^{-1/2} H_n^{-1/2}. \tag{1.94}$$

Consequently, $H_n^{-1/2} J H_n^{-1/2}$ is similar to $J H_n^{-1}$, i.e., it has the same eigenvalues. Since $H_n^{-1/2} J H_n^{-1/2}$ is symmetric, $J H_n^{-1}$, is diagonalizable with real eigenvalues.
Then,

$$\sum_{i=1}^{d} a_i J_i H_n^{-1}, \qquad i = 1, \dots, d, \tag{1.95}$$

with $a_i \neq 0$ is similar to a sum of symmetric matrices and thus is diagonalizable with real eigenvalues. By definition 1.4 the system is then hyperbolic. □

Proof: (Theorem 1.15) The entropy $h(\mathbf{u})$ is convex by the construction of the minimal entropy closure. We show the integrability condition for each element of the flux function of the moment system (1.54). Consider the entropy-flux

$$j(\mathbf{u})_i = \langle v_i \eta(f_\mathbf{u}) \rangle = \left\langle v_i \eta(\eta_*'(\boldsymbol{\alpha}_\mathbf{u} \cdot \mathbf{m})) \right\rangle, \tag{1.96}$$

where we use the ansatz (1.62) for the reconstruction of $f_\mathbf{u}$. We have

$$\begin{aligned}
\nabla_\mathbf{u} j(\mathbf{u})_i &= \nabla_\mathbf{u} \langle v_i \eta(f_\mathbf{u}) \rangle \\
&= \langle \eta'(\eta_*'(\boldsymbol{\alpha}_\mathbf{u} \cdot \mathbf{m})) v_i \mathbf{m} \eta_*''(\boldsymbol{\alpha}_\mathbf{u} \cdot \mathbf{m}) \nabla_\mathbf{u} \boldsymbol{\alpha}_\mathbf{u} \rangle \\
&= \langle \boldsymbol{\alpha}_\mathbf{u} v_i \mathbf{m} \otimes \mathbf{m} \eta_*''(\boldsymbol{\alpha}_\mathbf{u} \cdot \mathbf{m}) \nabla_\mathbf{u} \boldsymbol{\alpha}_\mathbf{u} \rangle && \text{(Legendre duality)} \\
&= \nabla_\mathbf{u} h(\mathbf{u}) \langle v_i \mathbf{m} \otimes \mathbf{m} \eta_*''(\boldsymbol{\alpha}_\mathbf{u} \cdot \mathbf{m}) \nabla_\mathbf{u} \boldsymbol{\alpha}_\mathbf{u} \rangle && \text{(Equation (1.64))} \\
&= \nabla_\mathbf{u} h(\mathbf{u}) \nabla_\mathbf{u} F_i(\mathbf{u})
\end{aligned} \tag{1.97}$$

We multiply Eq. (1.54) with $\nabla_\mathbf{u} h(\mathbf{u})$, i.e.,

$$\begin{aligned}
\nabla_\mathbf{u} h(\mathbf{u}) \partial_t \mathbf{u}(t, \mathbf{x}) + \nabla_\mathbf{u} h(\mathbf{u}) \nabla_\mathbf{x} \cdot \mathbf{F}(\mathbf{u}) &= \nabla_\mathbf{u} h(\mathbf{u}) \langle \mathbf{m}(\mathbf{v}) Q(f_\mathbf{u}) \rangle \\
&= \boldsymbol{\alpha}_\mathbf{u} \langle \mathbf{m}(\mathbf{v}) Q(f_\mathbf{u}) \rangle && \text{(Equation (1.64))} \\
&= \langle \boldsymbol{\alpha}_\mathbf{u} \mathbf{m}(\mathbf{v}) Q(f_\mathbf{u}) \rangle \\
&= \langle \eta'(\eta_*'(\boldsymbol{\alpha}_\mathbf{u} \cdot \mathbf{m})) Q(\eta_*'(\boldsymbol{\alpha}_\mathbf{u} \cdot \mathbf{m})) \rangle && \text{(Legendre duality)} \\
&\leq 0. && \text{(Theorem 1.8)}
\end{aligned} \tag{1.98}$$

The integrability condition above concludes the proof. □

CHAPTER 2

Neural Network-Based Machine Learning

In this chapter, we provide a comprehensive overview of the fundamental concepts of supervised machine learning using neural networks. We explain the principles and mechanics of training parametrized models, including a discussion of gradient-based optimization techniques as applied to neural network training. Furthermore, we examine recent advancements in over-parametrized neural networks, and their implications for memory footprint and computational expense. These insights serve to contextualize the subsequent chapters of this dissertation.

2.1. Supervised Machine Learning

Machine learning is generally a function approximation task where a parametrized map f_θ, called a model, aims to approximate a target function f^*. The target function f^* is often not explicitly stated, but rather given implicitly by a (training) data set $X_T = \{(\mathbf{x}, \mathbf{y})_i\}_{i \in T}$ with size T of independent $\mathbf{x}$ and dependent $\mathbf{y}$ variables. This scenario is called supervised machine learning. If no dependent variable $\mathbf{y}$ is given, but we have access to a residual, the task is called unsupervised learning. An example is the approximation of the solution of a differential equation by minimizing its residual formulation.
A machine learning task consists of four parts, all equally important to successfully develop a working approximation, i.e.,

- the representation of the target function, called hypothesis space,
- the evaluation metric, i.e. the objective or loss function,
- the optimization method to minimize the loss function,
- the structure and preparation of the training data.

The hypothesis space denotes the set of candidate functions for the model f_θ. In classical numerical analysis, this is often the space of polynomials or piecewise polynomials. Neural networks are another choice of hypothesis space.

Definition 2.1 (Neural Network)
We define a neural network as a function $N_\theta : \mathbb{R}^n \to \mathbb{R}^m$ of the form

$$N_\theta(\mathbf{x}) = \mathbf{z}_M. \tag{2.1}$$
$$\mathbf{z}_k = \sigma_k(W_k\mathbf{z}_{k-1} + \mathbf{b}_k), \qquad k = 1, \dots, M \tag{2.2}$$
$$\mathbf{z}_0 = \mathbf{x}. \tag{2.3}$$

The non-linear and differentiable activation function $\boldsymbol{\sigma}_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ is defined in a point-wise manner, i.e.,

$$\boldsymbol{\sigma}_k(\mathbf{z}_k) = \left[\sigma_k(\mathbf{z}_{k,1}), \dots, \sigma_k(\mathbf{z}_{k,n_k})\right]^\top. \tag{2.4}$$

The network is a concatenation of M layers. The layer output is denoted by $\mathbf{z}_k$. The set of weight matrices $W_k \in \mathbb{R}^{n_k \times n_{k-1}}$, and biases $\mathbf{b}_k \in \mathbb{R}^{n_k}$ of all layers yield the set of trainable parameters θ.

In practice, this definition is often modified, e.g. to convolutions, but at its core, most neural network architectures consist of non-linearities applied to affine linear maps. The hypothesis space is then spanned by all possibly obtainable trainable parameters θ and the given network architecture. The architectural parameters of the hypothesis space are called hyperparameters. Examples of hyperparameters are the number of layers M or the layer dimensions n_k, or the maximum degree of a polynomial model.
The next step in solving the supervised learning task is to choose a loss function to measure the distance of the current approximation f_θ to the target function f^*, which we denote as $\mathcal{L}(X_T; f_\theta)$ for the data set X_T. For the sake of simplicity, we use the Mean Squared Error (MSE) as the choice of $\mathcal{L}$ for the rest of this chapter, i.e.,

$$\mathcal{L}(X_T; f_\theta) = \frac{1}{T}\sum_{i \in T} \|f_\theta(\mathbf{x}_i) - (\mathbf{y}_i)\|_2^2. \tag{2.5}$$

Furthermore, the optimization algorithm has to be chosen to change the trainable parameters θ to approximate f^*. In neural network-based supervised machine learning, gradient descent methods are used frequently amongst which are stochastic gradient descent (SGD) [131], ADAM [135] or RMSProp [225]. Gradient-free methods, such as particle-swarm optimization [73] are also possible.
The last ingredient is the preprocessing of the data set X_T. Sometimes, X_T is finite and given; in machine learning-based surrogate modeling, one can often produce more data by evaluating the (often computationally expensive) governing equations of a physical system. Often, the raw data is not as useful for training the model with the chosen optimizer and has to be normalized, centered, and features have to be selected by dimension reduction techniques. Also, parts of the data have to be held back to test the developed model as an estimate for its performance on unseen data. We formalize the machine learning task in the following.

Definition 2.2 (Training, test and generalization error in supervised learning [62])
Consider a model f_θ, a target function f^*, and a training data set X_T. Then, the training error of the model is defined by

$$\mathcal{T}(f_\theta) = \mathcal{L}(X_T; f_\theta) = \frac{1}{T}\sum_{i \in T} \|f_\theta(\mathbf{x}_i) - (\mathbf{y}_i)\|_2^2. \tag{2.6}$$

Let $P_\mathbf{x}$ be the (real-world) distribution of the independent variable $\mathbf{x}$ of the data set. The generalization error of the model f_θ is defined by

$$\mathcal{G}(f_\theta, f^*) = \mathbb{E}_{\mathbf{x}\sim P_\mathbf{x}}\left(\left\|f_\theta(\mathbf{x}) - f^*(\mathbf{x})\right\|_2^2\right). \tag{2.7}$$

We call the estimate of $\mathcal{G}$ on a finite test data set X_{test}, which we assume to be sampled from $P_{\mathbf{x}}$, and $f^*(\mathbf{x})$ and to be disjoint from the training data set X_T , the test error

$$\widetilde{\mathcal{T}}(f_\theta) = \mathcal{L}(X_{test}; f_\theta,) . \tag{2.8}$$

Thus, the machine learning task, given X_T, is to find the minimizer $f_{\theta*}$ of $\mathcal{G}(f_{\theta^*}, f^*)$. Due to the unavailability of a direct evaluation of $\mathcal{G}(f_{\theta^*}, f^*)$, the task is reformulated as follows. Give X_T, choose a hypothesis space, hyperparameters, an evaluation method, an optimization algorithm, and a data-processing pipeline. Then compute

$$f_{\theta^*} = \underset{f_\theta}{\operatorname{argmin}} \mathcal{L}(X_T; f_\theta) , \tag{2.9}$$

such that $\widetilde{\mathcal{T}}(f_{\theta^*}, f^*)$ is minimal. Note, that the test-error must not be involved in the training (2.9) directly or indirectly for hyperparameter tuning. If it was, we would not be able to use X_{test} as a measure for the performance of the model on unseen data and thus it $\widetilde{\mathcal{T}}$ would not be a good approximation or $\mathcal{G}$.

2.2. Stochastic Gradient Descent

In this work, we mainly use neural networks N_θ as parametrized models f_θ. Thus, we give an overview of the neural network training techniques used in this dissertation, i.e., how to solve the optimization problem (2.9). Gradient descent based optimizers to solve (2.9) read as

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta \mathcal{L}(X_T; \mathrm{N}_\theta), \qquad t = 1, 2, \dots \tag{2.10}$$

where $\lambda_t > 0$ is the learning rate. For the a layer-based definition of a neural network, see Definition 2.1, the gradient descent step of iteration t is given by

$$W_k^{t+1} = W_k^t - \eta \nabla_{W_k} \mathcal{L}(X_T; \mathrm{N}_\theta(\cdot; W_1^t, \dots, W_M^t, \mathbf{b}_1^t, \dots, \mathbf{b}_M^t)), \qquad k = 1, \dots, M \tag{2.11}$$

$$\mathbf{b}_k^{t+1} = \mathbf{b}_k^t - \eta \nabla_{\mathbf{b}_k} \mathcal{L}(X_T; \mathrm{N}_\theta(\cdot; W_1^t, \dots, W_M^t, \mathbf{b}_1^t, \dots, \mathbf{b}_M^t)), \qquad k = 1, \dots, M \tag{2.12}$$

Remark, that from the non-linearity of neural networks, it is clear that the optimization problem (2.9) is not convex, and that we cannot hope to find its global minimum using gradient descent. Thus, many extensions like momentum-based gradient descent methods such as ADAM [135] or RMSProp [225] have been developed to mitigate the issue of the optimizer getting stuck in a local minimum. Furthermore, computing the global minimizer of (2.9) does not necessarily translate to finding the global minimizer of $\widetilde{\mathcal{T}}(\mathrm{N}_{\theta^*}, f^*)$ let alone $\mathcal{G}(\mathrm{N}_{\theta^*}, f^*)$, since test data is not available to the optimizer. Nevertheless, gradient-based methods are most commonly used for neural network training.
Note that the gradient evaluation of modern neural network implementations, like PyTorch and Tensorflow, is mostly carried out using reverse mode automatic differentiation. For an introduction to the topic, we refer to [91].

Consider the gradient computation

$$\nabla_\theta \mathcal{L}(X_T; \mathrm{N}_\theta) = \frac{1}{T} \sum_{i \in T} \nabla_\theta \left\| \mathrm{N}_\theta(\mathbf{x}_i) - f^*(\mathbf{x}_i) \right\|_2^2 , \tag{2.13}$$

whose evaluation requires averaging of the gradient evaluations for all samples of the training data set X_T. This is prohibitively expensive in practice since the X_T is often by orders of magnitude bigger than

Algorithm 2.1: Stochastic gradient descent training of neural networks

Input: θ^0: Weight initialization
N_θ: Neural network architecture
$X_T = \bigcup_t X_{B,t}$: Training data
λ: Learning rate
`epoch`: number of training iterations over X_T
`iter`: number of mini-batch evaluations per epoch

Result: N_{θ^*}: Trained network

for *each* `epoch` **do**
  for $t = 0$ *to* $t =$`iter` **do**
    Load mini-batch $X_{B,t}$
    Register θ for gradient tape /* Start of automatic differentiation */
    $z_{M,i} \leftarrow \mathrm{N}_\theta(\boldsymbol{x}_i)\,,\ \forall i \in B$ /* Forward pass, vectorized */
    $\mathcal{L} \leftarrow \frac{1}{B}\sum_{i\in B}\left\|z_{M,i} - f^*(\boldsymbol{x}_i)\right\|_2^2$ /* Batch loss evaluation */
    Evaluate $\nabla_\theta \mathcal{L}$ /* Evaluation of the gradient tape */
    $\theta^{t+1} \leftarrow \theta^t - \lambda \nabla_\theta \mathcal{L}$ /* Gradient descent step */

the available memory of the computation device. Motivated by this practical challenge, one relies on mini-batch training, i.e., approximating the gradient

$$\nabla_\theta \mathcal{L}(X_{B_i}; \mathrm{N}_\theta) \approx \nabla_\theta \mathcal{L}(X_T; \mathrm{N}_\theta), \tag{2.14}$$

where the training data set is partitioned into mini-batches X_{B_i}, s.t. $X_T = \bigcup_i X_{B_i}$ [217]. Then, stochastic gradient-based neural network training can be summarized in Algorithm 1.

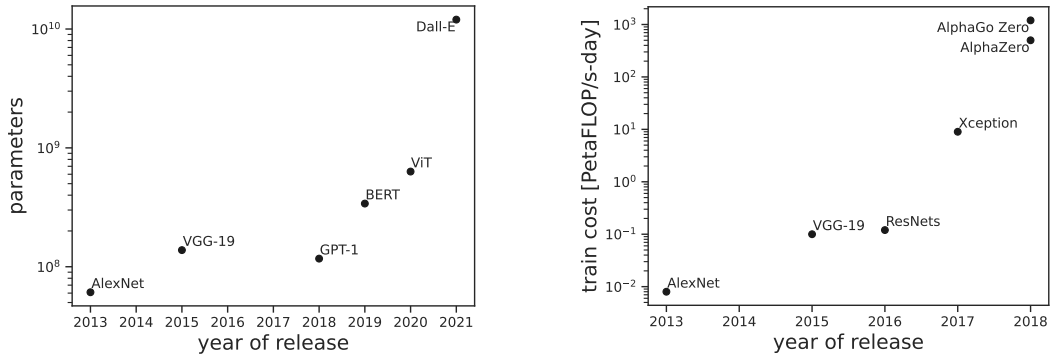
Besides the practical aspects, stochastic gradient descent has beneficial regularization properties, which are explored for linear inverse problems [123]. Furthermore, it has been investigated that the stochastic noise introduced by the SGD optimizer acts as an additional loss term that flattens the high dimensional solution landscape when training over-parametrized neural networks [244]. This is beneficial to avoid local minima in the training process.

2.3. Over-Parametrized Neural Networks

Let us consider a parametrized model f_θ with finite variance. The generalization error (2.7) can be rewritten to

$$\mathcal{G}(f_\theta, f^*) = \mathbb{E}_{\mathbf{x}\sim P_\mathbf{x}}\left(\left\|f_\theta(\mathbf{x}) - f^*(\mathbf{x})\right\|_2^2\right) = \mathbb{E}_{\mathbf{x}\sim P_\mathbf{x}}\left(\left\|f_\theta(\mathbf{x}) - f^*(\mathbf{x})\right\|_2\right)^2 + \mathrm{Var}_{\mathbf{x}\sim \mathrm{P}_\mathbf{x}}\left(\mathrm{f}_\theta(\mathbf{x})\right), \tag{2.15}$$

where the first term of the right-hand side is the squared bias, and the second is the variance of the model. If there is white noise in the training data set, a third term, specifying the standard deviation of the noise, is added to the generalization error.
Classical statistical learning theory considers the bias-variance trade-off [99]. Here, the squared bias of the model can be thought of as the error induced by simplifying assumptions built into the model, e.g., a linear approximation of a high-order polynomial. The variance of the model specifies how much f_θ moves around its mean. The idea of the bias-variance trade-off is that models with higher complexity,



(a) Network sizes

(b) Training cost

Figure 2.1.: Parameter count of popular network architectures (left) and computational cost for a single training run in Peta FLOP/S-days (right). The computational cost of large models increase by an order of magnitude each year.

have lower bias but a higher variance in their prediction. Generally, the number of trainable parameters of the model does not have a direct influence on its complexity. As an example, consider the model

$$f_{\theta=\{a,b\}}(x) = a \sin(bx), \tag{2.16}$$

which can approximate any number of points with a high enough frequency, resulting in a high-bias, high-variance model with only two parameters.

In neural network-based machine learning, the consensus is traditionally, that the variance of a model increases with its parameter count $|\theta|$, whereas its bias decreases [84]. Consider a neural network with $p = |\theta|$ degrees of freedom, and a training process on a dataset with $d = |X_T|$ data points. Three regimes of neural network training are denoted, i.e.,

- the under-parametrized regime for $p \ll d$,
- the critical regime for $p \approx d$ with the interpolation threshold $p = d$,
- the over-parametrization regime for $p \gg d$.

The test error of a neural network tends to increase in the critical regime. Different regularization techniques [216, 47] are used to mitigate this variance increase. However, newer works give empirical evidence [191] and theoretical reasoning [46, 61] that (unregularized) neural networks generalize well in the over-parametrization regime. This challenges the classical assumption as the variance does not increase with a high parameter count.

Further, over-parametrized neural networks became popular in applied aspects of machine learning, due to their effectiveness in the training phase. Thus, in recent years massively over-parametrized neural network models have been proposed and won several competitions for machine learning. Examples are illustrated in Fig. 2.1, which shows a trend of the exponential increase of trainable parameters. Since the majority of trainable parameters are given by weight matrices W of large dimension, the corresponding computational expense to train and evaluate the networks increases by approximately an order of magnitude per year. Consequently, training and inference of modern neural network architectures become increasingly capital, energy, and time-consuming.

The prohibitive computational cost motivates neural network compression methods with the goal to identify sparse substructures within a neural network that approximately resemble the accuracy of the original model. A multitude of compression methods have been developed in the last decade, which can be partitioned into (a) data-free methods that only consider the original (trained) network but no training data, (b) data-efficient methods that subsequently re-train and compress a trained network, and (c) methods that compress a network using the full training data-sets [150]. The memory footprint can be reduced by the identification of sparse sub-matrices, factorization of the weight matrices, or reduction of their floating point accuracy.
We investigate this matter and propose a low-rank factorization to compress a neural network during training in §7.

CHAPTER 3

KiT-RT: A Modular Simulation Toolkit for Kinetic Transport

In this chapter, we present KiT-RT (Kinetic Transport Solver for Radiation Therapy), an open-source C++ framework for solving kinetic equations in radiation therapy applications. This framework aims to provide a collection of classical deterministic solvers for unstructured meshes that allow for easy extendability. KiT-RT is a convenient base to test new numerical methods in various applications and compare them against convenotional solvers. The implementation includes spherical harmonics, minimal entropy, neural minimal entropy, and discrete ordinates methods. Solution characteristics and efficiency are presented through several test cases ranging from radiation transport to electron radiation therapy. Due to the variety of included numerical methods and extendability, the presented open-source code is attractive for both developers, who want a basis to build their numerical solvers, and users or application engineers, who want to gain experimental insights without directly interfering with the codebase.

3.1. Introduction

High-fidelity numerical simulation of radiation transport enabled on workstation computers is crucial for both scientific prototyping and hands-on applications like treatment planning in radiation oncology. Besides the aim to ensure sufficient accuracy, such simulations are required to run on limited computational resources such as workstation PCs. An open-source framework increases trustworthiness for crucial applications such as medicine or industrial radiation transport as well as enables extension by methodological and application engineers, mathematicians, and scientists.

3.1.1. Related Work on Radiation Oncology Planning

Although the solver can be used and modified for other applications, it is primarily tailored to simulate dose distributions for radiation treatment planning. Personalized medicine in radiation oncology has been an important research topic in the last decades to generate automated optimal treatment plans. To

allow for accurate, reliable, and efficient treatment planning tailored toward individual patient needs, there is a growing desire to undertake direct numerical simulation for radiation therapy.
Traditional methods to predict dose distributions in radiation oncology largely rely on simplified models, such as pencil beam models based on the Fermi-Eyges theory [67]. While such models are computationally efficient, they often lack the required accuracy, especially in cases including air cavities or other inhomogeneities [109, 140]. On the other hand, Monte Carlo (MC) algorithms, which simulate individual interacting particles, achieve a satisfactory accuracy [10]. However, despite ongoing research to accelerate MC methods, their high computational costs currently render them impractical for clinical usage [74, 122]. In [143], the modal P_N method has been employed to derive a macroscopic model for radiation treatment planning. While it does not preserve the positivity of the solution and can potentially yield oscillatory approximations, it allows for an efficient numerical treatment of scattering terms.

3.1.2. Novelty and Scientific Contribution

The variety of different methods allows for individual method choices tailored to different settings. The comparability of different methods in a uniform framework is not only interesting for clinical usage, but also for future research in computational radiation therapy. Our goal for the open-source C++ Kinetic Transport for Radiation Therapy (KiT-RT) framework is therefore to provide a collection of available deterministic methods. Special focus is put on extendability by the use of polymorphism to simplify the implementation of novel solution methods. The methods provided by our framework are optimized for an application on workstation PCs. This meets the typical requirements in radiation therapy applications: for clinical usage, the computational resources are often limited and the time between recording the CT image and the actual treatment must not exceed a certain time. Hence, radiation therapy codes that are applicable for clinical use should run efficiently on workstation PCs. Moreover, conventional codes often require structured grids [82, 141, 199, 211, 236], leading to inaccurate representations of structures on CT images. While accuracy in practice is also limited by the CT density values which are given on a structured grid, these are often down-sampled to a lower resolution for dose computations. Further, a re-computation of the CT values for unstructured grids is feasible if it improves the quality of dose computations. Therefore, our framework provides functionalities for both unstructured meshes which preserve organ outlines on CT images, as well as standard rectangular grids.
This is a collaborative work with Jonas Kusch, Pia Stammer, Jannick Wolters, and Tianbai Xiao. This chapter focuses on the author's contribution to the project and the full paper can be found at [147]. The code is available on Github[1] and extended details of the methods and documentation of the code can be found on ReadTheDocs[2].

3.1.3. The Chapter in Context of the Dissertation

This Chapter serves two purposes for the dissertation. First, we demonstrate the open-source software KiT-RT, which serves as the computational backbone for the computations of §4,§5 and §6. Secondly, we review the spatial-temporal discretizations given by second-order finite volume methods for unstructured grids, which lay the numerical base for all velocity space discretizations of this dissertation.

[1] https://github.com/CSMMLab/KiT-RT
[2] https://kit-rt.readthedocs.io

3.1.4. Organization of the Chapter

This chapter aims at presenting the developed framework and its functionality while providing the necessary mathematical and physical background on the principles the software is based on. In §3.2.1 we provide the underlying physical model as well as its reformulation as a time-dependent partial differential equation. §3.3 and §3.5 focus on the used discretizations and software architecture, respectively. Lastly, we validate our implementations and analyze their performance for different test cases in §3.7.

3.2. The Boltzmann Equation for Dose Computation

The KiT-RT solver's main purpose is to solve linear, time-dependent kinetic equations and steady state, linear, energy-dependent kinetic equations with continuous slowing down approximation. Specimen for the former is the linear Boltzmann equation, i.e.,

$$\partial_t f(t, \mathbf{x}, \mathbf{v}) + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = Q(f) \tag{3.1}$$

where the collision operator is given as

$$Q(f) = \int_{\mathbf{V}} k(t, \mathbf{x}, \mathbf{v}, \mathbf{v}^*) \left[f(\mathbf{v}^*) - f(\mathbf{v}) \right] \, \mathrm{d}\mathbf{v}^* \tag{3.2}$$

with a specifiable collision kernel $k(t, \mathbf{x}, \mathbf{v}, \mathbf{v}^*)$ with space, velocity and time dependence. Examples are given in §3.7.

3.2.1. Continuous Slowing Down Approximation

The second type of equations that are solvable by the KiT-RT solver suite is energy-dependent kinetic equations with continuous slowing down approximation. They can be seen as an extension of the linear Boltzmann model, and have their application in computational radiotherapy treatment planning. Here we model radiation transport with a single particle speed, thus we choose the velocity basis as the unit sphere, i.e., $\mathbf{V} = \mathbb{S}^2$. For radiotherapy treatment planning, one is generally interested in the computation of the radiation dose distribution

$$D(\mathbf{x}) = \frac{1}{\rho(\mathbf{x})} \int_0^\infty \int_{\mathbb{S}^2} S(E, \mathbf{x}) f(E, \mathbf{x}, \mathbf{v}) \, \mathrm{d}\mathbf{v} \, \mathrm{d}E \, , \tag{3.3}$$

that results from a given treatment plan. Here, $E \in \mathbb{R}_+$ is the energy, $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^3$ denotes the spatial domain, and $\mathbf{v} \in \mathbb{S}^2$ is the flight direction of particles. Moreover, $\rho : \mathbb{R}^3 \to \mathbb{R}$ models the space-dependent patient tissue density. The stopping power $S : \mathbb{R}_+ \times \mathbb{R}^3 \to \mathbb{R}$ models the continuous energy loss of particles due to scattering with tissue and is defined as

$$S(E, \mathbf{x}) = \int_0^\infty E' \int_{-1}^1 \sigma(E, E', \mathbf{x}, \mu) \, \mathrm{d}\mu \, \mathrm{d}E' \tag{3.4}$$

with the scattering cross-section $\sigma_s : \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}^3 \times [-1, 1] \to \mathbb{R}$. The radiation flux density, which describes the probability of finding a particle at a certain region in phase space, can be computed from

the continuous slowing down (CSD) approximation [151] of the energy-dependent linear Boltzmann equation, i.e.,

$$\begin{aligned} &-\partial_E \left(S(E,\mathbf{x})f(E,\mathbf{x},\mathbf{v})\right) + \mathbf{v}\cdot\nabla_{\mathbf{x}} f(E,\mathbf{x},\mathbf{v}) \\ &= \int_{\mathbb{S}^2} \sigma_s(E,\mathbf{x},\mathbf{v}\cdot\mathbf{v}')f(E,\mathbf{x},\mathbf{v}')\,\mathrm{d}\mathbf{v}' - \sigma_t(E,\mathbf{x})f(E,\mathbf{x},\mathbf{v})\,. \end{aligned} \tag{3.5}$$

This model assumes a continuous energy loss of particles traveling through a background material, which is modeled using the stopping power S. For clarity of exposition, we decompose the collision kernel into the scattering σ_s and total cross-section σ_t. The scattering cross-section $\sigma_s(E,\mathbf{x},\mathbf{v}\cdot\mathbf{v}')$ denotes the probability of particles at position $\mathbf{x}$ with energy E changing their flight direction from $\mathbf{v}'$ to $\mathbf{v}$ due to a collision with the patient tissue. The total cross-section σ_t is given by

$$\sigma_t(E,\mathbf{x}) = \sigma_{s,0}(E,\mathbf{x}) = 2\pi\int_{-1}^{1}\sigma_s(E,\mathbf{x},\mu)\,\mathrm{d}\mu\,. \tag{3.6}$$

KiT-RT is capable of constructing custom stopping powers, scattering, and total cross-sections specified by physics-databases [90] and specific material properties [143, 193, 237], for which we refer to the full paper [147].

Having defined the prerequisites of our physical model, we can focus on bringing it into a form that allows for computing numerical approximations efficiently. It turns out that the energy variable in (3.5) can be treated as a pseudo-time, which facilitates solving the CSD equation. For a given maximal energy $E_{\max}$ let us define the transformed energy as

$$\widetilde{E}(E) := \int_0^{E_{\max}} \frac{1}{S(E')}\,\mathrm{d}E' - \int_0^{E}\frac{1}{S(E')}\,\mathrm{d}E' \tag{3.7}$$

and the transformed particle density as

$$\widetilde{f}(\widetilde{E},\mathbf{x},\mathbf{v}) := S(E)\rho(\mathbf{x})f(E(\widetilde{E}),\mathbf{x},\mathbf{v})\,. \tag{3.8}$$

Then, multiplying Eq. (3.5) with $S(E)$ and plugging in the defined transformation gives

$$\partial_{\widetilde{E}}\widetilde{f}(\widetilde{E},\mathbf{x},\mathbf{v}) + \mathbf{v}\cdot\nabla_x\frac{\widetilde{f}(\widetilde{E},\mathbf{x},\mathbf{v})}{\rho(\mathbf{x})} + \widetilde{\sigma}_t(\widetilde{E})\widetilde{f}(\widetilde{E},\mathbf{x},\mathbf{v}) = \int_{\mathbb{S}^2}\widetilde{\sigma}_s(\widetilde{E},\mathbf{v}\cdot\mathbf{v}')\widetilde{f}(\widetilde{E},\mathbf{x},\mathbf{v}')\,\mathrm{d}\mathbf{v}'\,, \tag{3.9}$$

where we define $\widetilde{\sigma}_t(\widetilde{E}) := \sigma_t(E(\widetilde{E}))$ and $\widetilde{\sigma}_s(\widetilde{E},\mathbf{v}\cdot\mathbf{v}') := \sigma_s(E(\widetilde{E}),\mathbf{v}\cdot\mathbf{v}')$. Dropping the tilde notation and treating $\widetilde{E}$ as a pseudo-time t gives a slightly modified version of the classical linear Boltzmann equation

$$\begin{aligned} \partial_t f(t,\mathbf{x},\mathbf{v}) + \mathbf{v}\cdot\nabla_x\frac{f(t,\mathbf{x},\mathbf{v})}{\rho(\mathbf{x})} + \sigma_t(t)f(t,\mathbf{x},\mathbf{v}) &= \int_{\mathbb{S}^2}\sigma_s(t,\mathbf{v}\cdot\mathbf{v}')\psi(t,\mathbf{x},\mathbf{v}')\,\mathrm{d}\mathbf{v}' \\ f(t=0,\mathbf{x},\mathbf{v}) &= S(E_{\max})\rho(\mathbf{x})f(E_{\max},\mathbf{x},\mathbf{v})\,. \end{aligned} \tag{3.10}$$

Hence, the CSD equation can be treated numerically with classical closure methods and space-time discretizations.

3.3. Macroscopic Methods

To obtain a computationally feasible model with comparable accuracy, radiation particles are described on a mesoscopic level through the deterministic linear Boltzmann equation [24, 222, 223, 224]. An

efficient and accurate numerical approximation to the linear Boltzmann equation can be achieved through the construction of grid-based macroscopic approximations [60, 229, 230, 87]. Variants of grid-based methods for radiation therapy can e.g. be found in [16, 108, 193, 143, 148, 115].
The velocity space discretization of the Boltzmann equation is performed with the macroscopic models discussed in §1.2, i.e., the M_N method, namely

$$\begin{aligned} \partial_t \mathbf{u}(t,\mathbf{x}) + \nabla_{\mathbf{x}} \cdot \langle \mathbf{v}\mathbf{m}(\mathbf{v}) f_{\mathbf{u}} \rangle &= \langle \mathbf{m}(\mathbf{v}) Q(f_{\mathbf{u}}) \rangle \,, \\ f_{\mathbf{u}} &= \underset{f \in F_{\mathbf{m}}}{\operatorname{argmin}} \{ \langle \eta(f) \rangle : \langle \mathbf{m} f \rangle = \mathbf{u} \} \,, \end{aligned} \tag{3.11}$$

where we use $\langle \cdot \rangle$ for the velocity integral. Further, we introduce the regularized M_N method, i.e.,

$$\begin{aligned} \partial_t \mathbf{u}(t,\mathbf{x}) + \nabla_{\mathbf{x}} \cdot \langle \mathbf{v} \otimes \mathbf{m}(\mathbf{v}) f_{\mathbf{u}} \rangle &= \langle \mathbf{m}(\mathbf{v}) Q(f_{\mathbf{u}}) \rangle \,, \\ f_{\mathbf{u}} &= \underset{f \in F_{\mathbf{m}}}{\operatorname{argmin}} \left\{ \langle \eta(f) \rangle + \frac{1}{2\gamma} \| \langle \mathbf{m} f \rangle - \mathbf{u} \|_2^2 \right\} , \end{aligned} \tag{3.12}$$

which we discuss in detail in §5. The P_N method is given by

$$\partial_t \mathbf{u}(t,\mathbf{x}) + \mathbf{A} \cdot \nabla_{\mathbf{x}} \mathbf{u}(t,\mathbf{x}) = G\mathbf{u}(t,\mathbf{x}), \tag{3.13}$$

and the S_N method by

$$\partial_t \mathbf{f}_q(t,\mathbf{x}) + \mathbf{v}_q \cdot \nabla_{\mathbf{x}} f_q(t,\mathbf{x}) = \sum_{p=1}^{Q} w_p k(\mathbf{v}_p, \mathbf{v}_q) \left[f_p(t,\mathbf{x}) - f_q(t,\mathbf{x}) \right]. \tag{3.14}$$

Application to the continuous slowing down approximation is performed analogously.
The spatial and temporal discretization for the macroscopic systems follows the finite volume method for transport equations. We state the necessary notation and the numerical expressions to understand the implementation. For a detailed derivation of finite volume methods and numerical schemes for transport equations, we refer to [23, 159].

3.4. Spatial-Temporal Discretization

The KiT-RT framework is based on unstructured, cell-centered grids as spatial discretization. In the following, we restrict ourselves to a two-dimensional spatial grid, however, the notations are straightforwardly applicable to three or one spatial dimension. An unstructured grid $\widetilde{\mathbf{X}} = \{\mathbf{X}_i\}_{i \in I}$ is a partition of a bounded spatial domain $\mathbf{X} \subset \mathbb{R}^d$. A grid cell $\mathbf{X}_i$ holds information about the coordinates of its centroid $\mathbf{x}_i$, its measure A_i, the indices of its boundary vertices, indices of its neighbor cells $N(i)$ and cell faces. The information of the cell faces is encoded in the unit-normal vector of the face dividing cell i and its neighbor $j \in N(i)$, multiplied with the measure of the face and is denoted by $\mathbf{n}_{i,j}$. The grids used in this work are either triangular or quadrilateral unstructured grids in two spatial dimensions.

3.4.1. Finite Volume Methods

The nodal and modal methods are different approaches to discretizing the velocity space of the Boltzmann equation, which result all in a system of transport equations that can be solved using a finite volume scheme. Thus, we first describe a method agnostic finite volume scheme and afterward point out the differences of the S_N, P_N, and M_N based implementations. We denote the temporal variable by t,

however, the results hold for energy interpreted as a pseudo-time as well. Let $\mathbf{g}(t, \mathbf{x}) \in \mathbb{R}^m$ be the vector of conserved variables of a system of transport equations

$$\partial_t \mathbf{g}(t, \mathbf{x}) + \nabla_{\mathbf{x}} \cdot \mathbf{F}(\mathbf{g}(t, \mathbf{x})) = \mathbf{R}(t, \mathbf{x}, \mathbf{g}(t, \mathbf{x})), \qquad \mathbf{x} \in \mathbf{X},\ t \in [0, t_f) \tag{3.15}$$

where $\mathbf{F}$ is the general flux function describing the solution transport, and $\mathbf{R}$ is a general right-hand side, containing velocity discretizations of collision terms, sources, and absorption terms. The main discretization strategy is to divide the spatial domain into an unstructured grid with N_x cells and the time domain into N_t discrete values $0 = t_0 < \ldots < t_{N_t-1}$. We consider the solution as an average

$$\mathbf{g}_i^n = \frac{1}{A_i} \int_{\mathbf{X}_i} \mathbf{g} \, \mathrm{d}\mathbf{x} \tag{3.16}$$

over one space-time cell. Then, we average Eq. (3.15) over one space-time cell

$$\begin{aligned} \frac{1}{\Delta t A_i} \int_{\mathbf{X}_i} \int_{t_n}^{t_{n+1}} \partial_t \mathbf{g}(t, \mathbf{x}) \, \mathrm{d}t \, \mathrm{d}\mathbf{x} + \frac{1}{\Delta t A_i} \int_{\mathbf{X}_i} \int_{t_n}^{t_{n+1}} \nabla_{\mathbf{x}} \cdot \mathbf{F}(\mathbf{g}(t, \mathbf{x})) \, \mathrm{d}t \, \mathrm{d}\mathbf{x} \\ = \frac{1}{\Delta t A_i} \int_{\mathbf{X}_i} \int_{t_n}^{t_{n+1}} \mathbf{R}(t, \mathbf{x}, \mathbf{g}(t, \mathbf{x})) \, \mathrm{d}t \, \mathrm{d}\mathbf{x}, \end{aligned} \tag{3.17}$$

where $\Delta t = t_{n+1} - t_n$. Solving the integrals using the Gauss theorem for the advection term and an explicit Euler scheme for the time derivative yields

$$\frac{1}{\Delta t} \left(\mathbf{g}_i^{n+1} - \mathbf{g}_i^n \right) + \frac{1}{\Delta t A_i} \int_{t_n}^{t_{n+1}} \sum_{j \in N(i)} \mathbf{F}(\mathbf{g}(t, x_{i,j})) \cdot \mathbf{n}_{i,j} \, \mathrm{d}t = \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \mathbf{R}(t, \mathbf{x}, \mathbf{g}(t, x_i)) \, \mathrm{d}t, \tag{3.18}$$

where $g(t, x_i)$ is the conserved variable evaluated at cell i and $g(t, x_{i,j})$ is the conserved variable evaluated at the interface between cell i and its neighbor j. To compute the actual value of $\mathbf{g}_j^{n+1}$, one needs to find approximations for the flux integral. A common ansatz is of the form

$$\mathbf{F}(\mathbf{g}_j^n, \mathbf{g}_i^n) \approx \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \mathbf{F}(\mathbf{g}(t, \mathbf{x}_{i,j})) \cdot \mathbf{n}_{i,j} \, \mathrm{d}t, \tag{3.19}$$

where the numerical flux $F(\mathbf{g}_j^n, \mathbf{g}_i^n)$ at face (i, j) is approximated using the cell averaged conserved variable at cell i and j. For transport equations, a well-known numerical flux is given by the upwind scheme [159]

$$\mathbf{F}(\mathbf{g}_j^n, \mathbf{g}_i^n)_{up} = \mathbf{F}(\mathbf{g}_i^n) \cdot \mathbf{n}_{i,j} \mathcal{H}(\mathbf{n}_{i,j} \cdot \mathbf{v}) + \mathbf{F}(\mathbf{g}_j^n) \cdot \mathbf{n}_{i,j} \left(1 - \mathcal{H}(\mathbf{n}_{i,j} \cdot \mathbf{v}) \right), \tag{3.20}$$

where $\mathcal{H}$ is the heaviside step function and $\mathbf{v}$ is the transport velocity vector. Finally, we approximate the source, absorption, and collision terms using the current cell average. Thus the explicit solution iteration of a first-order scheme reads

$$\mathbf{g}_i^{n+1} = \mathbf{g}_i^n - \frac{\Delta t}{A_i} \sum_{j \in N(i)} \mathbf{F}(\mathbf{g}_j^n, \mathbf{g}_i^n)_{up} + \Delta t \mathbf{R}(t, \mathbf{x}_i, \mathbf{g}_i^n). \tag{3.21}$$

Since the scattering term $\mathbf{R}$ is commonly stiff, implicit-explicit (IMEX) schemes can be used to remove influences of scattering from time step restrictions. If we assume a linear scattering term, that is with a given matrix $\mathbf{R}_i^{n+1}$ we have $R(t^{n+1}, \mathbf{x}_i, \mathbf{g}_i^n) = \mathbf{R}_i^{n+1} \mathbf{g}_i^n$, the IMEX scheme reads

$$(\mathbf{I} - \Delta t \mathbf{R}_i^{n+1}) \mathbf{g}_i^{n+1} = \mathbf{g}_i^n - \frac{\Delta t}{A_i} \sum_{j \in N(i)} \mathbf{F}(\mathbf{g}_j^n, \mathbf{g}_i^n)_{up}. \tag{3.22}$$

3.4.2. Second Order Finite Volume Methods

The KiT-RT solver provides the option to evaluate the space and time discretizations using second-order accurate schemes. To this end, we use a Heun scheme for the temporal discretization and a second-order upwind flux for the numerical flux [17]. Whereas, first-order spatial fluxes assume a constant solution value $\mathbf{g}_i^n$ in a cell i, a second-order upwind scheme is based on a linear reconstruction of the conserved variable. Therefore the inputs $\mathbf{g}_i^n$ and $\mathbf{g}_j^n$ to the numerical flux of Eq. (3.20) are replaced by

$$\tilde{\mathbf{g}}_i^n = \mathbf{g}_i^n + \Psi_i \left(\nabla_{\mathbf{x}} \mathbf{g}_i^n \cdot \mathbf{r}_{i,j} \right), \tag{3.23}$$

$$\tilde{\mathbf{g}}_j^n = \mathbf{g}_j^n + \Psi_j \left(\nabla_{\mathbf{x}} \mathbf{g}_j^n \cdot \mathbf{r}_{j,i} \right), \tag{3.24}$$

where $\mathbf{r}_{i,j}$ is the vector pointing from cell centroid $\mathbf{x}_i$ of cell i to the interface midpoint between cells i and j, and Ψ_i is the flux limiter for cell i. This reconstruction is formally second-order accurate on regular grids [4] assuming exact evaluation of the gradient $\nabla_{\mathbf{x}} \mathbf{g}_i^n$. The gradient of the conserved variable $\mathbf{g}_i^n$ is evaluated using the Green-Gauss theorem with interpolated solution values at the cell interfaces,

$$\nabla_{\mathbf{x}} \mathbf{g}_i^n \approx \frac{1}{A_i} \sum_{j \in N(i)} \frac{1}{2} \left(\mathbf{g}_i^n + \mathbf{g}_j^n \right) \cdot \mathbf{n}_{\mathbf{i},\mathbf{j}}. \tag{3.25}$$

Second or higher-order upwind spatial discretizations require the use of flux limiters to prevent the generation of oscillations in shock regions and to achieve a monotonicity-preserving scheme. In the KiT-RT package, the Barth and Jespersen limiter [17], as well as the Venkatakrishnan limiter [232], are implemented. Exemplary, we show the computation of the Barth and Jespersen limiter at cell i, i.e.,

$$\Psi_i = \min_j \begin{cases} \min(1, \frac{\mathbf{g}_{\max} - \mathbf{g}_i}{\Delta_2}), & \text{if } \Delta_2 > 0 \\ \min(1, \frac{\mathbf{g}_{\min} - \mathbf{g}_i}{\Delta_2}), & \text{if } \Delta_2 < 0 \\ 1, & \text{else} \end{cases}, \tag{3.26}$$

where we have

$$\Delta_2 = \frac{1}{2} \nabla_{\mathbf{x}} \mathbf{g}_i^n \cdot \mathbf{r}_{i,j}, \tag{3.27}$$

$$\mathbf{g}_{\max} = \max(\mathbf{g}_i, \mathbf{g}_j), \tag{3.28}$$

$$\mathbf{g}_{\min} = \min(\mathbf{g}_i, \mathbf{g}_j). \tag{3.29}$$

The second-order Heun scheme for temporal discretization is a two-step Runge-Kutta scheme with the iteration formula

$$\mathbf{g}_i^* = \mathbf{g}_i^n - \frac{\Delta t}{A_i} \sum_{j \in N(i)} \mathbf{F}(\mathbf{g}_j^n, \mathbf{g}_i^n)_{up} + \frac{\Delta t}{A_i} \mathbf{R}(\mathbf{g}_i^n), \tag{3.30}$$

$$\mathbf{g}_i^{**} = \mathbf{g}_i^* - \frac{\Delta t}{A_i} \sum_{j \in N(i)} \mathbf{F}(\mathbf{g}_j^*, \mathbf{g}_i^*)_{up} + \frac{\Delta t}{A_i} \mathbf{R}(\mathbf{g}_i^*), \tag{3.31}$$

$$\mathbf{g}_i^{n+1} = \frac{1}{2} \left(\mathbf{g}_i^n + \mathbf{g}_i^{**} \right), \tag{3.32}$$

which is based on the implicit trapezoidal integration method.

3.4.3. Numerical Fluxes

In the following, we adapt the introduced numerical methods to the method-specific notation and present the detailed implementation. The space and time-averaged conservative variable $\mathbf{g}_i^n$ for the nodal discretization at cell i and time step n is given by the vector of the radiation flux

$$\mathbf{f}_i^n = [f(\mathbf{v}_1), \dots, f(\mathbf{v}_{N_q})]^T \in \mathbb{R}^{N_q} \tag{3.33}$$

evaluated at the quadrature points. The different methods are distinguishable by their numerical flux function. The corresponding numerical flux for the S_N method is given by

$$F(\mathbf{f}_i^n) = \mathbf{v}\mathbf{f}_i^n, \qquad i \in I \tag{3.34}$$

and the corresponding upwind flux reads

$$\mathbf{F}(\mathbf{f}_j^n, \mathbf{f}_i^n)_{up} = \mathbf{v} \cdot \mathbf{n}_{i,j}\mathbf{f}_i^n \mathcal{H}(\mathbf{v} \cdot \mathbf{n}_{i,j}) + \mathbf{v} \cdot \mathbf{n}_{i,j}\mathbf{f}_j^n \left(1 - \mathcal{H}(\mathbf{v} \cdot \mathbf{n}_{i,j})\right). \tag{3.35}$$

The conservative variables $\mathbf{g}_i^n$ of modal methods are given by the moment vector $\mathbf{u}_i^n$. The numerical flux for the P_N method is then

$$\mathbf{F}(\mathbf{u}_\mathbf{i}^\mathbf{n}) = \left[A_1\mathbf{u}_i^n, A_2\mathbf{u}_i^n, A_3\mathbf{u}_i^n\right]^T, \tag{3.36}$$

where A_i are the flux Jacobians emerging from the spherical harmonics recursion scheme. To evaluate the numerical flux with an upwind scheme, we decompose the flux Jacobians in their positive and negative definite parts, i.e.,

$$A_l = A_l^+ + A_l^-, \qquad l = 1, \dots, d. \tag{3.37}$$

Then the numerical flux is given by

$$\mathbf{F}(\mathbf{u}_i^n, \mathbf{u}_j^n)_{up} = \sum_{l=1}^{d} \left(A_l^+\mathbf{u}_i^n + A_l^-\mathbf{u}_j^n\right) n_l \mathcal{H}(n_l) + \left(A_l^-\mathbf{u}_i^n + A_l^+\mathbf{u}_j^n\right) n_l \left(1 - \mathcal{H}(n_l)\right). \tag{3.38}$$

In contrast to the P_N method, the flux function of the M_N method cannot be expressed as a matrix multiplication but reads

$$\mathbf{F}(\mathbf{u}_\mathbf{i}^\mathbf{n}) = \left\langle \mathbf{v} \otimes \mathbf{m}(\mathbf{v}) f_{\mathbf{u}_n^i}(\mathbf{v}) \right\rangle, \tag{3.39}$$

where $\psi_{\mathbf{u}_n^i}$ is the reconstructed radiation flux density of the minimal entropy closure at the cell averaged moment $\mathbf{u}_n^i$. Using a quadrature rule for the velocity integral discretization and a numerical flux for every quadrature point, we arrive at the kinetic numerical flux

$$\mathbf{F}(\mathbf{u}_j^n, \mathbf{u}_i^n)_{up} = \sum_{q=1}^{Q} w_q \mathbf{m}_q \mathbf{v}_q \cdot \mathbf{n}_{i,j} \left[f_{\mathbf{u}_i^n,q} \mathcal{H}(\mathbf{v}_q \cdot \mathbf{n}_{i,j}) + f_{\mathbf{u}_j^n,q} \left(1 - \mathcal{H}(\mathbf{v}_q \cdot \mathbf{n}_{i,j})\right)\right]. \tag{3.40}$$

Note, that the updated solution $\mathbf{u}_i^n$ of the M_N method must still be a feasible, i.e., a realizable moment for the minimal entropy closure of Eq. (1.52). To ensure this, one must either employ a flux limiter [141], construct a realizability reconstruction [144] or employ the regularized entropy closure formulation [7]. The implentation of all solvers is summarized in Algorithm 2.

Algorithm 3.1: Explicit time iteration scheme for the linear Boltzmann equation

Input: $\mathbf{u}_i^0$ or $\mathbf{f}_i^0$: Initial condition of the test case
$\widetilde{\mathbf{X}}$: Mesh
$\mathbf{R}_i$: Boundary and source terms

Result: $\mathbf{u}_i^{n_{tf}}$ or $\mathbf{f}_i^{n_{tf}}$: Solution values at final time n_{tf} (and intermediate results, if specified)

if *modal* **then**
  $\mathbf{g}_i^0 \leftarrow \mathbf{u}_i^0 \quad \forall i \in I$ /* Modal initial Condition */
if *nodal* **then**
  $\mathbf{g}_i^0 \leftarrow \mathbf{f}_i^0 \quad \forall i \in I$ /* Nodal initial Condition */
for $n = 0, \dots, n_{tf}$ **do**
  $\mathbf{g}_i^*, \mathbf{g}_i^{k=0} \leftarrow \mathbf{g}_i^n \quad \forall i \in I$ /* Runge Kutta initialization */
  for $k = 0, \dots, k_{rk} - 1$ **do**
    if M_N *method* **then**
      $f_{\mathbf{u},i}^k \leftarrow \texttt{Closure}(\mathbf{u}_i^n) \quad \forall i \in I$ /* Entropy closure */
    if 2^{nd} *order* **then**
      $\Psi_i^k \leftarrow \texttt{SlopeLimiter}(i, j) \quad \forall i \in I$ /* Compute and limit spatial slopes */
    $\mathbf{F}(\mathbf{g}_i^k) \leftarrow \sum_{j \in N(i)} \mathbf{F}_{\text{up,i,j}}^k \quad \forall i \in I$ /* Flux computation */
    $\mathbf{g}_i^{k+1} \leftarrow \mathbf{g}_i^k - \frac{\Delta t}{A_i} \left(\mathbf{F}(\mathbf{g}_i^k) + \mathbf{R}(\mathbf{g}_i^k) \right) \quad \forall i \in I$ /* Pseudo time integration */
  $\mathbf{g}_i^{n+1} \leftarrow \frac{1}{2} \left(\mathbf{g}_i^* + \mathbf{g}_i^{k_{rk}} \right) \quad \forall i \in I$ /* Time integration */

3.4.4. Discretization of the Collision Operator for the CSD Approximation

Radiation therapy applications exhibit forward-peaked scattering, which cannot be well-captured by classical quadrature rules. To allow for moderate computational costs when computing scattering terms and to efficiently treat forward-peaked scattering, we transform the nodal solution to a modal description and apply the more efficient P_N methodology for scattering terms. For this, we define a truncation order N and construct the matrices $O \in \mathbb{R}^{Q \times (N+1)^2}$ which maps the modal onto its nodal representation. Conversely, $M \in \mathbb{R}^{(N+1)^2 \times Q}$ maps the nodal onto its modal representation. Such matrices can be constructed by

$$O = (\mathbf{m}(v_k))_{k=1}^Q \text{ , and } \quad M = (w_k \mathbf{m}(v_k))_{k=1}^Q .$$

In this case, we can replace the scattering term on the right-hand side of Eq. (3.14) by its P_N counterpart

$$\sum_{p=1}^{Q} w_p k(\mathbf{v}_p, \mathbf{v}_q) f_p(t, \mathbf{x}) = \sum_{p=1}^{Q} (O \sigma_s M)_{q,p} \, f_p. \tag{3.41}$$

Thus, the continuous slowing down approximation with S_N discretization reads

$$\partial_t f_q(t, \mathbf{x}) + \mathbf{v}_q \cdot \nabla_{\mathbf{x}} \frac{f_q(t, \mathbf{x})}{\rho(\mathbf{x})} = \sum_{p=1}^{Q} (\mathbf{O \Sigma M})_{q,p} \, f_p - \sigma_t f_q(t, \mathbf{x}) \tag{3.42}$$

3.5. Software Architecture

The design principle of the KiT-RT software package is focused on efficient implementation, high reusability of its components, and ease of extension. It contains a set of efficient numerical solvers for radiation transport, which are constructed of basic, reusable building blocks. These building blocks can be freely arranged to implement new solvers or tools for completely different applications. On the other hand, KiT-RT is equipped with an easy-to-use command line interface based on readable configuration files, which allows easy manipulation of the solvers. Thus the software is attractive for developers, who want to experiment with the framework and build their numerical solvers as well as users and application engineers, who want to gain experimental insights without directly interfering with the code base.
KiT-RT is implemented in modern C++ and uses mainly polymorphism for its construction. In the following, we present the class structures used to build the numerical solvers and explain the used building blocks, which are displayed in Fig. 3.1. Most building blocks consist of a virtual base class, which contains a static factory method to build an instance of the concrete-derived class, defined by the given configuration details. Furthermore, the virtual base class defines the interface of this building block with other parts of the KiT-Framework. For implementation details of the most important classes, we refer to §3.9.

3.6. Parallel Scaling

In the following, we investigate the parallel performance of the three base solver implementations S_N, P_N, and M_N, where we follow [189] for the brief review of parallel scalings. The speedup of a parallel algorithm is defined as

$$S(n,p) = \frac{T^*(n)}{T(n,p)}, \tag{3.43}$$

where $T^*(n)$ is the execution time of the best inherently serial algorithm with input size n and $T(n,p)$ the time for the parallel implementation with p processing workers and input size n. In general, the best serial algorithm may be different from the parallel algorithm, however, in our application case, the finite-volume discretization scheme does not change for serial implementation.
In theory the best possible speedup is linear [68], i.e., $S(n,p) = p$, thus the measure of parallel efficiency is

$$E(n,p) = \frac{S(n,p)}{p}. \tag{3.44}$$

In practice, the speedup and parallel efficiency of an algorithm is limited by spawning and communication overhead of the parallel workers as well as the fraction of inherently serial code f, that exists in any algorithm. Thus, the upper bound for the speedup is given by Amdahl [94] as

$$S(n,p) \leq \frac{1}{f + (1-f)/p} \tag{3.45}$$

For larger input sizes, the fraction of inherently serial code f typically decreases, which enables the use of highly parallel implementations. Two common approaches to measuring the parallel performance are given by the strong and weak scaling approach. The former describes an experiment, where for fixed input size n the number of parallel workers p is increased and their timing is measured, which directly results in the speedup of Eq. (3.43). The latter increases the input size n proportionally to the worker count p. A perfectly parallel algorithm would have a constant parallel time $T(n,p)$.

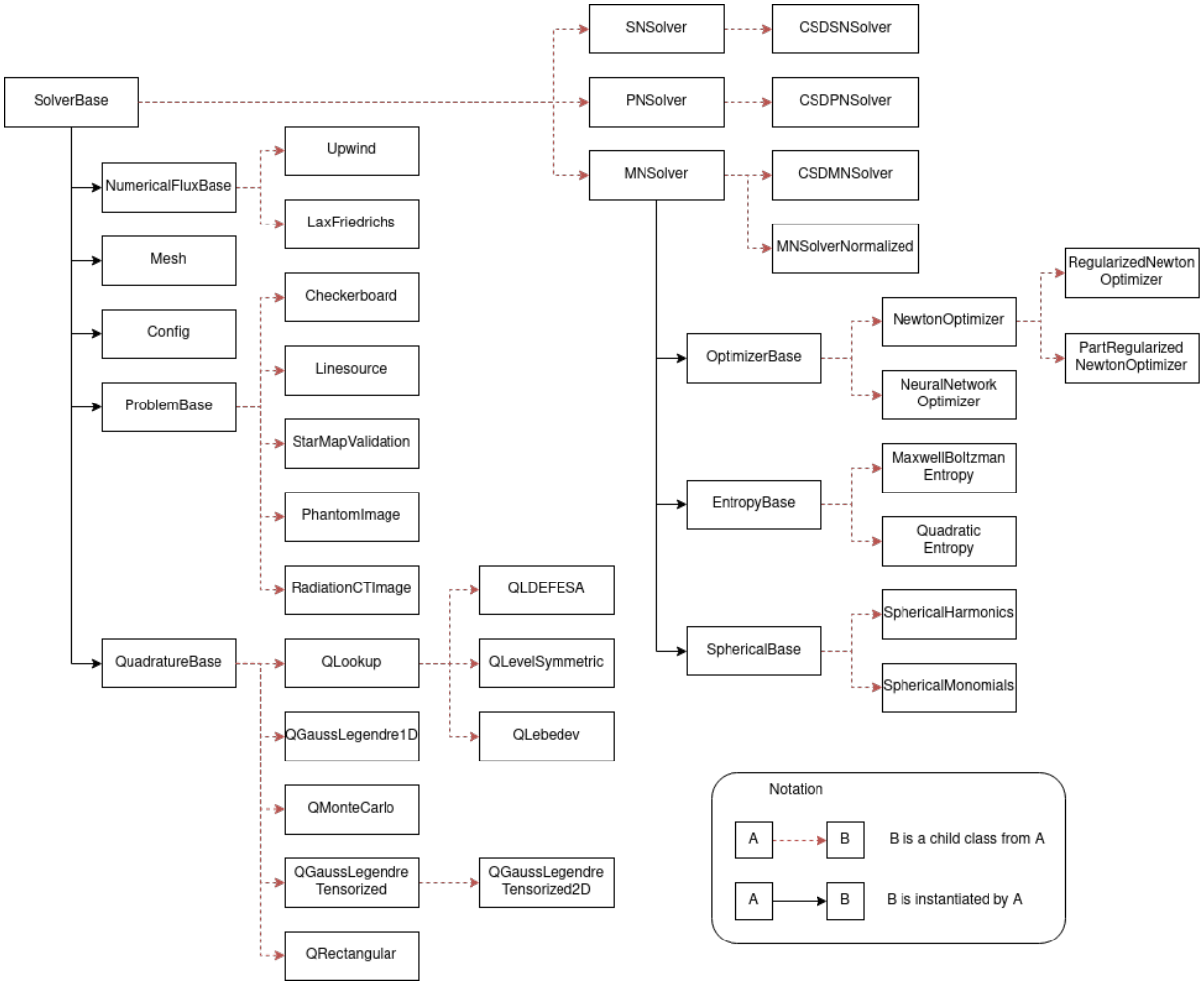
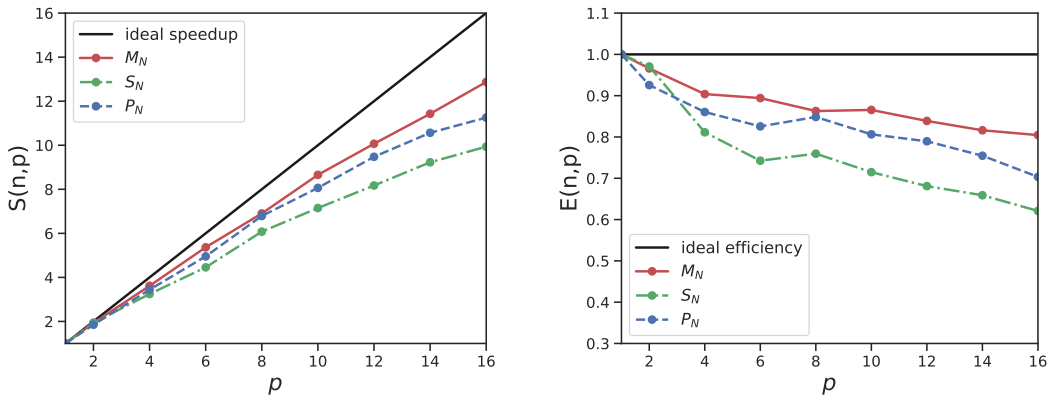


Figure 3.1.: Class and inheritance structure of the virtual `SolverBase` Class. Each instantiated solver has class members and routines specific to its numerical structure.



(a) Strong scaling

(b) Parallel efficiency

Figure 3.2.: Strong parallel scaling (left) and parallel efficiency (right) for the M_N, S_N, and P_N solver. M_N methods have the best scaling, due to their immense CPU load during the entropy closure.

The philosophy of the parallel implementation of the solvers of this work is based on the independence of the performed computations in the grid cells. As Algorithm 2 displays, during one (pseudo-) time iteration of a solver, a set of instructions are calculated. Each instruction can be carried out independently for each grid cell and only in between two instruction sets, the communication between parallel workers needs to be established. Therefore, the parallel implementation spawns a set of parallel workers with shared memory access and distributes the spatial grid among them to carry out the current instruction. The input size n is thus given by the number of grid cells of the spatial discretization.

We perform a strong parallel scaling study for the implementation of the M_N, S_N, and P_N solver on the Linesource test case, as described in §3.7.1, with a fixed unstructured triangular mesh of size $n = 578290$ and a varying number of shared memory parallel workers p. We choose $p = 1, \dots, 16$, furthermore, the solver's allocated memory does not exceed the system's memory. Figure. 3.2 shows a comparison of the solver's parallel scalings and efficiency. It is apparent that the M_N solver enjoys the highest speedup even for a high parallel worker count, while the P_N and S_N solver experience diminishing returns for more than $p = 12$ workers. Reason for this is the higher computational load per core for the M_N solver which requires the computation of the minimal entropy closure to construct its flux function. Furthermore, S_N solvers have a higher memory load compared to modal methods for comparable numerical accuracy. Here, the CPU cores may run idle while the CPU cache has to be exchanged more often in comparison with modal methods, which impares parallel scaling.
The performance of the continuous slowing down solver implementations follows the corresponding base solver performance since the same spatial, velocity, and (pseudo-) temporal discretizations are used.

3.7. Validation of KiT-RT

For validation and a comparison of the implemented solvers, we consider a selection of the test cases provided within the class `ProblemBase` of KiT-RT. The S_N solver is validated against the Julia package Kinetic.jl [239]. The continuous slowing down solvers are further compared to a reference Monte Carlo solution computed using TOPAS [199] as well as the validated spherical harmonics solver StaRMAP [211].

Table 3.1.: Computational setup for the numerical test cases

Test Case	CFL	t_f	Δ_t	Δ_x	N_x
Inhomogeneous Linesource	0.7	0.1	2.33e−3	3.33e−3	9e5
Checkerboard	0.45	10	3.15e−2	7e−2	2.5e5

An overview of the computational setup of all test cases is given in Table 3.1

3.7.1. Inhomogeneous Linesource Test Case

In the following, we compare the numerical results for the continuous slowing down approximation, see §3.2.1 of our framework to a Monte Carlo solution, computed using TOPAS [199] as well as the staggered-grid spherical harmonics solver StarMap [211]. The problem considered is an inhomogeneous Linesource test case, which extends the classical Linesource benchmark [79, 78] to a steady-state but energy-dependent setting. We consider a spatial domain $\mathbf{X} = [0, 1]^2$ and a velocity domain $\mathbf{V} = P_{\mathbb{R}^2}(\mathbb{S}^2)$. As background density, we choose a piece-wise constant function

$$\rho(\mathbf{x}) = 1 + 4 \cdot \mathbb{1}_{X_{\text{right}}}(\mathbf{x}), \qquad \mathbf{x} \in \mathbf{X}. \tag{3.46}$$

In the left part of the spatial domain $\mathbf{X}_{\text{left}} = [0, 1] \times [0, 0.56]$ a reduced density is used, compared to the right part of the spatial domain $\mathbf{X}_{\text{right}} = [0, 1] \times [0.56, 1]$. At a maximal energy of $E_{\max} = 1$, i.e., initial pseudo time, a particle beam is positioned in the center of the spatial domain $\mathbf{x}_0 = [0.5, 0.5]^T$, which is modeled as

$$f(E, \mathbf{x}, \mathbf{v}) = \frac{1}{2\pi\epsilon^2} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{2\sigma^2}\right), \quad E = E_{\max} \tag{3.47}$$

$$f(E, \mathbf{x}, \mathbf{v}) = 0, \qquad E = E_{\max}, \mathbf{x} \in \partial\mathbf{X}. \tag{3.48}$$

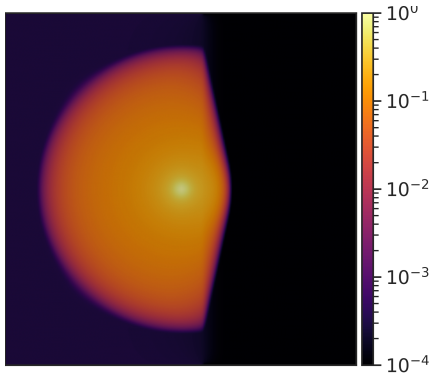
Here, a standard deviation of $\epsilon = 0.01$ is chosen to obtain a sharp particle beam in the center. No source term is used in the simulation and boundary conditions are zero-valued Dirichlet conditions, i.e.,

$$f(E, \mathbf{x}, \mathbf{v}) = 0, \qquad \mathbf{x} \in \partial\mathbf{X}. \tag{3.49}$$

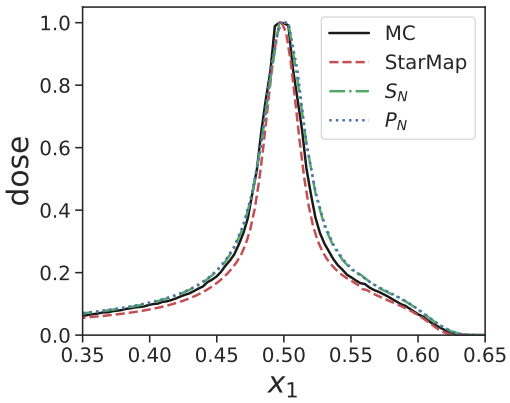
The scattering cross-section σ_s of the collision operator of the continuous slowing down approximation (3.10) is set constant to 1. The spatial grid for all deterministic methods is a structured rectangular grid with 300^2 cells. Due to the functionality of the Monte Carlo software, we use a three-dimensional grid and project the x_3-domain onto the $x_1 - x_2$ plane. To allow for feasible costs, the Monte Carlo method uses a coarser grid resolution of 100 spatial cells per dimension and 100000 Monte Carlo runs are computed to reduce statistical noise. The S_N solver uses a product quadrature rule of order 20 for the streaming step and spherical moments up to order 8 to compute scattering terms. Similarly, the P_N solver employs spherical moments up to order 8 and the M_N solver as well as its regularized counterpart uses shperical harmonics of order 3. The time step restriction of all deterministic methods picks a CFL number of 0.7. All methods are second order in space and time using the Barth and Jespersen slope limiter.

The resulting dose profile, i.e., the zeroth order moment of the kinetic density

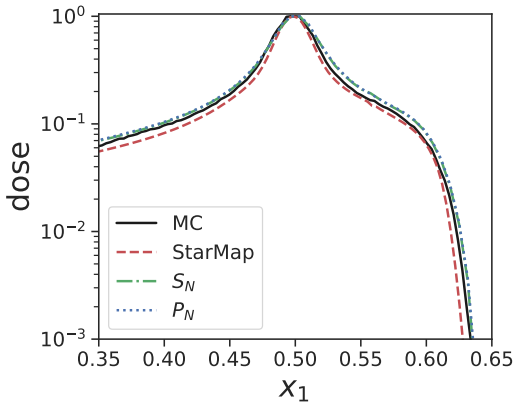
$$u_0(\mathbf{x}, t) = \int_{\mathbf{V}} f(t, \mathbf{x}, \mathbf{v}) \, \mathrm{d}\mathbf{v}, \tag{3.50}$$



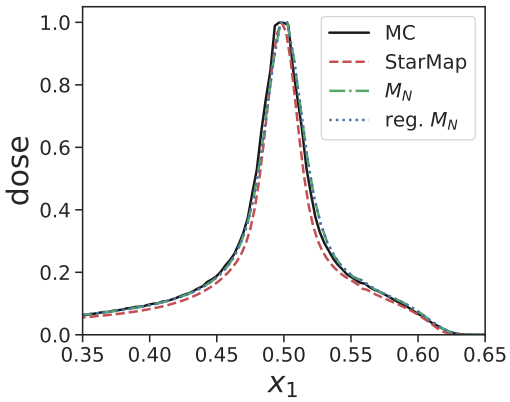
(a) Regularized M_3, inhomogenious linesource simulation with continuous slowing down approximation



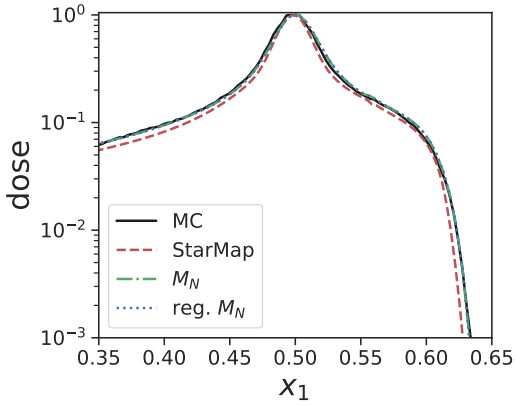
(b) S_{20}, P_8, linear scale



(c) S_{20}, P_8, log scale



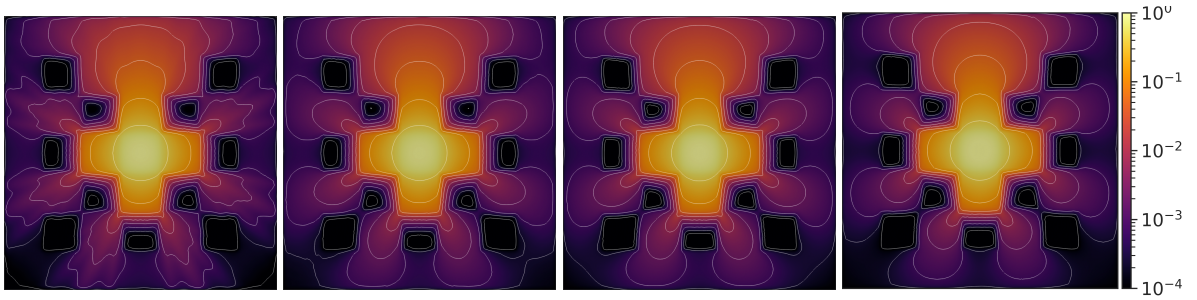
(d) M_3, reg M_3, linear scale



(e) M_3, reg M_3, log scale

Figure 3.3.: Comparison of simulation results of deterministic and stochastic methods. The M_3 method and regularized M_3 method with $\gamma = 1e-3$ coincide best with the Monte Carlo reference, surpassing StarMap in accuracy, as seen in the logarithmic scale plots.

is plotted in Fig. 3.3 along the x_1-axis in the interval $x_1 \in [0.3, 0.65]$ at $x_2 = 0.5$. All methods show similar behavior and agree well with the Monte Carlo results. Moreover, it is observed that the regularized M_N method seems to coincide with its non-regularized counterpart and is closer to the Monte Carlo reference than the S_N and P_N methods. The StarMap solution mostly undershoots the Monte Carlo reference, whereas all KiT-RT solutions slightly overshoot it.



(a) S_{10} (b) P_5 (c) M_3 (d) reg. M_3, $\gamma = 0.001$

Figure 3.4.: Simulation results of various KiT-RT solvers for the Checkerboard test case in log scale; S_{10}, P_5, M_3 and regularized M_3 solver (left to right). Ray effects are visible at the oscillating contours of the S_{10} solution. The modal methods correspond well with each other.

3.7.2. Checkerboard Test Case

The heckerboard test case mimics a nuclear reactor block with a strong radiative source in the domain center, which is denoted by $\mathbf{X}_q$, and several highly absorptive regions $\mathbf{X}_a$ placed in a checkerboard pattern around it, see Fig. 3.4. The corresponding time-dependent linear Boltzmann equation reads

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = \sigma_s(\mathbf{x}) \int_{\mathbf{V}} f \, \mathrm{d}\mathbf{v} - \sigma_a(\mathbf{x}) f + q(\mathbf{x}) \tag{3.51}$$

for $\mathbf{x} \in \mathbf{X} = [0, 7]^2$, $t \in [0, 10)$ and $\mathbf{v} \in \mathbf{V} = P_{\mathbb{R}^2}(\mathbb{S}^2)$, i.e., the projection of the unit sphere onto $\mathbb{R}^2$. We equip the equation with Dirichlet boundary conditions and initial condition

$$f(t, \mathbf{x}, \mathbf{v}) = 0, \qquad \mathbf{x} \in \partial\mathbf{X} \tag{3.52}$$

$$f(t, \mathbf{x}, \mathbf{v}) = 0, \qquad t = 0 \tag{3.53}$$

to obtain a well-posed problem. Furthermore, the collision operator simplifies to

$$Q(f) = \sigma_s(\mathbf{x}) \int_{\mathbf{V}} f \, \mathrm{d}\mathbf{v} - \sigma_a(\mathbf{x}) f. \tag{3.54}$$

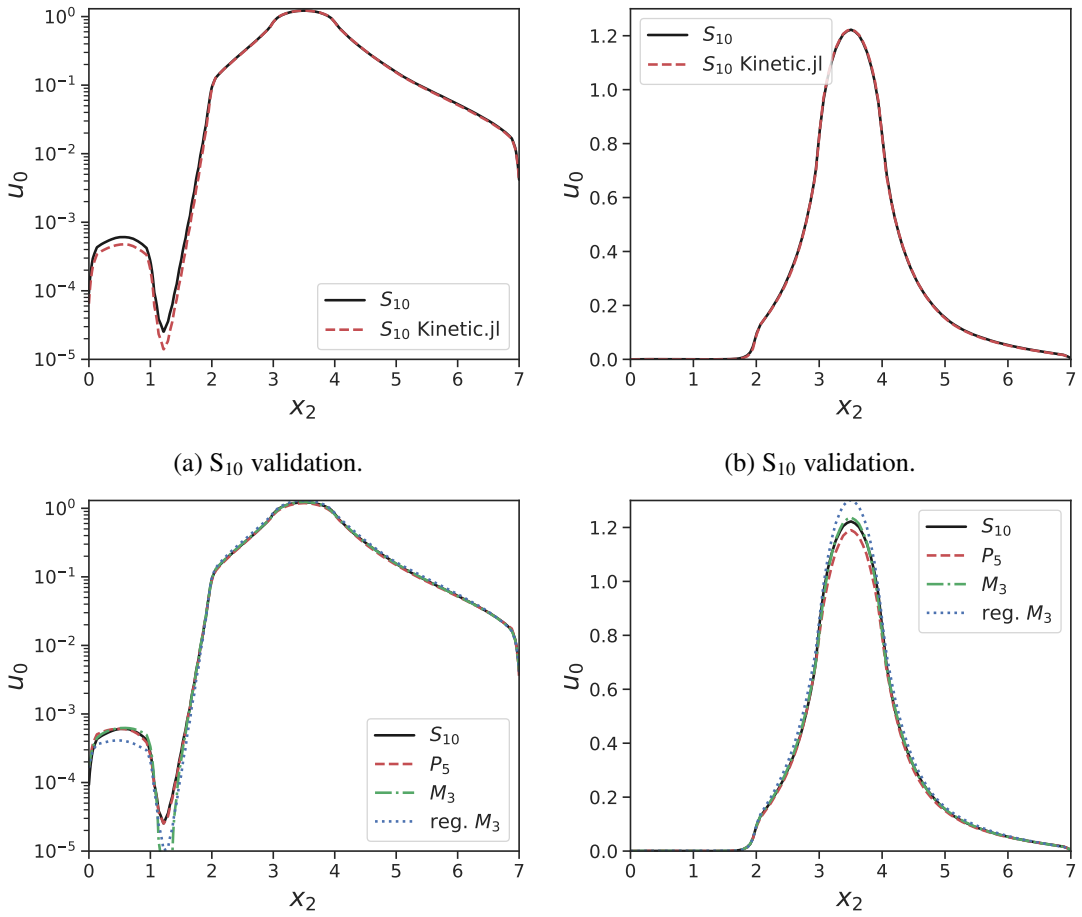
The scattering cross and absorption cross-sections are given by

$$\sigma_s(\mathbf{x}) = \begin{cases} 0 & \mathbf{x} \in X_a \\ 1 & \text{else} \end{cases}, \qquad \sigma_t(\mathbf{x}) = \begin{cases} 10 & \mathbf{x} \in \mathbf{X}_a \\ 1 & \text{else} \end{cases}, \tag{3.55}$$

The source term $q(\mathbf{x})$ is assumed to be isotropic and constant in time and is given by

$$q(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \mathbf{X}_q \\ 0 & \text{else} \end{cases}. \tag{3.56}$$

We create an unstructured triangular mesh with 25000 cells to discretize the spatial domain with regard to the absorption and source regions, such that the region boundaries coincide with the mesh faces. The simulation is computed until the final time $t_f = 10$ using various solver configurations. All employed solvers use a second-order upwind flux as the spatial discretization and the Heun scheme for temporal



(a) S_{10} validation.

(b) S_{10} validation.

(c) Macr. methods.

(d) Macr. methods.

Figure 3.5.: Cross-verification of the S_{10} solver against Kinetic.jl [239] (a)-(b), cross-section comparison of all KiT-RT solvers (c)-(d). M_3 solvers report lower values at absorption regions. The regularized M_3 method yields higher values at the radiation source, the M_3 and S_{10} methods correspond well.

discretization with CFL number equals 0.45 since M_N solvers with non-regularized entropy closure require a CFL number smaller than 0.5 for stability [8, 141, 144]. The solution computed at final time $t_f = 10$ is displayed in Fig. 3.4, where we can see the scalar flux

$$u_0(\mathbf{x}, t) = \int_{\mathbf{V}} f(t, \mathbf{x}, \mathbf{v}) \, \mathrm{d}\mathbf{v}, \tag{3.57}$$

in the contour plot in log scale, i.e., the moment of order zero. The radiation flux is highest at the source region $\mathbf{X}_q$ and almost zero in the absorption region $\mathbf{X}_a$ for all solvers. Towards the top of the domain, the radiation travels freely, whereas towards the left, right, and bottom, the radiation expansion is damped by absorption regions.

Figure 3.4 shows the S_{10} solver with a tensorized Gauss Legendre Quadrature, the P_5 solution with a spherical harmonics basis, the M_3 solution with a spherical harmonics basis, and a Newton optimizer with line-search configured to accuracy $1e-7$ and the regularized M_3 solution using the same optimizer and basis. As seen in Fig. 3.5, the deviation between implementations of the S_{10} solver and the reference

solution given by [239] is below the 1e−3, which is the characteristic length of a grid cell. The macroscopic methods yield quantitative differences in their solution. Regularization gives higher values of u_0 at the radiation source, and entropy-based methods have smaller values at absorption regions, compared to the P_N and S_N solutions.

3.8. Chapter Conclusion

In this chapter, we introduced the KiT-RT solver package for kinetic and radiation transport.

3.8.1. Summary

We have presented a collection of deterministic transport solvers for radiation therapy applications. The methods agree well with results obtained with conventional radiation therapy codes. Due to the use of polymorphism, we can guarantee a straightforward extension to further numerical methods, which facilitates the investigation of novel radiation therapy solvers and their comparison to conventional methods.

3.8.2. Limitations and Future Work

The framework can be extended in multiple directions. First, a low-rank acceleration of the macroscopic methods [148] can be implemented. Second, the MPI capabilities of the framework can be extended to support distributed memory systems. Optimization of treatment plans in a gradient-based or gradient-free manner can be conducted using the existing solvers to sample the objective functions. Uncertainty quantification in radiotherapy planning is a further research direction, in which KiT-RT can be extended.

3.9. Additional Material

Additional Material of Section 3.5

In the following, additional resources to understand the class structure of the KiT-RT framework, and explain the governing principles of its implementation.

Solver Class

The virtual `SolverBase` class is the basic blueprint for all time or energy-dependent finite volume solvers of the KiT-RT framework. It contains an abstract factory to create the instances of its child classes, which takes a pointer to the `Config` class as an argument. All child classes of `SolverBase` hold an instance of the `NumericalFlux`, `ProblemBase`, `QuadratureBase`, and `Mesh` class. Depending on the solver type and the physical setting instances of other classes may be members.
The `SolverBase` class controls the screen, log, and volume output of the solver. The screen output provides instantaneous feedback of the solver state via the command line and gives information on the current iteration, the total mass of the system, the residual of the radiation flux as well as the flow field, and whether logs and volume outputs have been written to file. The file log carries the same information as the screen output in a tabular format. Lastly, the volume output consists of `.vtk` files with solver and problem-specific solution data. The output data can be specified in the solver configuration.

The method `Solve()` of the `SolverBase` class drives the execution of all derived solvers by iterating over the time discretization of the numerical methods described in §3.3. This main time iteration is displayed in Algorithm 2. Each command is specified in the derived solver classes such as the P_N, S_N, and M_N solver and does not induce any additional communication overhead for the parallelization architecture.
The class `PNSolver` inherits from `SolverBase` and does not own additional instances of other custom building blocks and overwrites the subroutines of Algorithm 2 for the P_N equation specific numerical method, which allows for runtime solver assembly. Its child class is the `CSDPNSolver`, which is the implementation of the P_N based continuous slowing down solver, that overwrites the solver-preprocessing routines for the continuous slowing down specific energy transformation. The P_N-based solvers produce radiation flux and moments as output.
The class `SNSolver` adapts the subroutines of Algorithm 2 for the ordinate-based numerical methods and is the parent class of the `CSDSNSolver`. S_N-based solvers produce the radiation flux as output.
Lastly, the `MNSolver` class contains the implementation of the M_N numerical method and holds the module `SphericalBase`, which controls the choice of basis functions $\mathbf{m}(\mathbf{v})$ of the velocity space, the module `EntropyBase`, that controls the choice of the entropy functional for the entropy closure and lastly the module `OptimizerBase`, which controls the choice of numerical optimizer or neural network used to compute the entropy closure. The class `CSDMNSolver` inherits from the `MNSolver` class and analogously overwrites the subroutines of Algorithm 2 for the continuous-slowing down equations. M_N based solvers produce the radiation flux, moments, and dual variable of the entropy closure as output.

Mesh Class

The mesh class handles the computational meshes of the spatial discretization of the underlying differential equation. It can handle 1D meshes and 2D unstructured triangular and quadrilateral meshes in the `SU2` [195] mesh format. The mesh class keeps a record of all geometry and adjacency information required for the finite volume methods with first and second-order fluxes.

Computational Problem Class

The child classes of the `ProblemBase` class are responsible for the setup of the corresponding computational problems and test cases. These classes establish the initial conditions for the numerical solver solution and manage space, time, or energy-dependent material properties. The abstract class `ProblemBase` holds references to the `Mesh` and `Config` classes and creates instances of specific problems based on the selected configurations. Each implemented problem has two child classes, one for ordinate-based and one for moment-based solvers. The moment-based problem classes calculate the moments of the initial conditions and sources for the specified kinetic densities, which are specified in the nodal-based problem class.

The solver framework is equipped with several pre-implemented test cases and functionalities. This includes standard 1D and 2D test cases, such as Linesource and Checkerboard for the radiative transfer solvers, as well as isotropic and directed sources with different background media that can be loaded from a user-supplied image file for the continuous slowing down solvers. Custom test cases can be easily added by the user, following the provided examples and our modular approach.

Quadrature Class

The `QuadratureBase` class provides a virtual framework for the creation of specific numerical quadratures using its static factory method. These quadratures are designed to perform integration in velocity space, but they can also be utilized in other applications. Each quadrature has a defined order and is capable of handling integration points and weights in both Cartesian and spherical coordinates. The various quadratures are differentiated by their dimensionality and the region of integration. By default, the quadratures are designed to perform integration over the unit sphere, however, full velocity space integration can also be configured. Implemented quadrature methods include Spherical Monte Carlo, Levelsymmetric [171], Lebedev [178], Tensorized Gauss Legendre, and LDFESA [121].

Velocity Base Class

The class `VelocityBase` manages the basic choice of the velocity space $\mathbf{V}$ for modal solvers. Currently, monomial and spherical harmonics bases in $d = 1, 2, 3$ spatial dimensions are available and implemented as child-classes, each available for the (truncated) full velocity space and the unit sphere $\mathbb{S}^2 = \mathbb{V}$. For a given pointer to a child of the `QuadratureBase` class, the `VelocityBase` class creates an evaluation of the velocity base at the corresponding quadrature points. The spherical harmonics are implemented following the recursive Algorithm of [167].

Entropy Class

The M_N solver classes rely on the choice of an entropy functional to close the moment system via the minimal entropy closure ansatz. The abstract `EntropyBase` class yields an interface to implement any possible choice of the entropy functional alongside its Legendre dual and the corresponding derivatives and Hessians. Currently implemented are the Maxwell-Boltzmann, Fermi-Dirac, Bose-Einstein, and Quadratic entropy. The `EntropyBase` class can also be used to reconstruct kinetic density functions, moments, and the entropy functional value for a given Lagrange multiplier $\boldsymbol{\alpha}$. This feature is used to create the training data sampler for the neural network training in §4, §5, and §6.

Optimizer Class

The minimal entropy closure ansatz traditionally relies on some sort of numerical approximation to close the Boltzmann moment system. The `OptimizerBase` class gives a general interface to compute this task. The child class `NewtonOptimizer` provides an implementation of a Newton optimizer with line-search back-tracing as a locally quadratically converging optimizer for the convex entropy closure. The `NewtonOptimizer` holds an instance of a child of `EntropyBase`, and `QuadratureBase` to compute the entropy functional, its derivative, and Hessian. The regularized minimal entropy closure is implemented in the `RegularizedNewtonOptimizer`, which inherits `NewtonOptimizer` and adds the regularization terms to the entropy functional, derivative, and Hessian.

IO/Use of Config Files

The KiT-RT solver is a program that operates through a command line interface and requires a single argument, which is the configuration file. This file is analyzed to assemble the desired modules of the KiT-RT framework for a solver instance or any other custom tool.
The configuration file serves as a document that contains specifications for options in the form of `CONFIG_OPTION=VALUE`. It includes information regarding the input and output of files, such as the location of the mesh file, volume output files, and log files. Additionally, the computational problem and problem-specific parameters, such as the scattering coefficient, final time, spatial dimension, and boundary conditions, are defined in the configuration file. Furthermore, the solver-specific options are established, as exemplified by the M_N solver, where the velocity basis, the maximum degree of the basis functions, the CFL number, spatial integration order, entropy functional, optimizer, quadrature, and quadrature order are specified.
Finally, the configuration file includes specifications for the screen, volume, and log output, along with their respective output frequencies. Example configuration files for all currently implemented solvers and the numerical results can be found in the relevant GitHub repository[3].

Practices of Modern Software Development

The solver and its associated documentation are managed using version control with Git [40], which significantly improves collaboration among team members. The code is hosted on the web-based platform GitHub, which provides global access to the open-source MIT licensed code. This service also acts as a central hub for progress tracking, issue tracking, deployment, and maintaining code quality.

[3] https://github.com/CSMMLab/KiT-RT

To ensure code quality, automated testing is performed through unit tests and regression tests. Unit tests validate the functionality of individual code instances such as functions or classes by comparing expected results to actual results. Regression tests validate the solver as a whole by comparing results to reference solutions. The tests are automatically run every time changes are made to the code and submitted to the development branch or when a merge request is opened. If any of the tests fail, the submission is rejected to maintain the integrity of the code. Metrics such as test coverage, which describes the percentage of code lines validated through testing, help to build trust in the code. The test coverage for the KiT-RT framework is reported to the coveralls.io service[4].

Building and running the KiT-RT framework can be challenging due to its modest software dependencies. To simplify this process, a pre-configured build environment is provided through Docker [184]. Docker containers are isolated instances that have a minimal software stack and provide consistent software development and deployment. The specialized docker image is publicly available[5] and includes support for Tensorflow-based surrogate models[6].

The documentation is generated as part of the software build and is written in reStructuredText Markup language. The Sphinx [28] documentation framework is used to compile the markup files into linked HTML files, which are hosted by ReadTheDocs [7]. GitHub is also used for the deployment of precompiled software packages and the associated documentation.

The development workflow begins on GitHub, where each developer can create a new branch or fork the KiT-RT framework to obtain a personal workspace. After changes have been made, a merge request is filed, which is automatically tested by the continuous integration process. A core developer will then review the code, and if the tests are successful and the quality of the code is deemed sufficient, the changes will be merged into the development branch. When enough new features have been added to the development branch, it will be merged into the master branch and a new version number will be assigned (major or minor).

[4] https://coveralls.io/github/CSMMLab/KiT-RT
[5] https://hub.docker.com/r/kitrt/test
[6] https://hub.docker.com/r/kitrt/test_ml
[7] https://kit-rt.readthedocs.io

CHAPTER 4

Neural Network-Based, Structure-Preserving Minimal Entropy Closures

This chapter focuses on the development of neural network-based surrogate models for the minimal entropy closures of the Boltzmann moment system. These models aim to preserve the mathematical structure of the system, such as entropy dissipation and hyperbolicity, while improving computational efficiency of its corresponding numerical solver. To ensure the preservation of convexity of the moment-to-entropy map, the neural network approximation is designed to embed this feature on an archtectural level. An error bound for the interpolation error of convex neural networks trained in Sobolev norm is derived and used to develop an effective data sampling strategy. Numerical experiments are conducted to evaluate the performance of these neural network-based entropy closures and demonstrate their ability to significantly speed up kinetic solvers while maintaining an acceptable level of accuracy.

4.1. Introduction

The Boltzmann equation is a high dimensional integro-differential equation, with phase space dependency on space and particle velocity. The high dimensionality of the phase space presents a severe computational challenge for large-scale numerical simulations.
Moment methods eliminate the dependency of the phase space on the velocity variable by computing the moment hierarchy of the Boltzmann equation. Due to the structure of the advection term, the resulting moment system is typically unclosed. One distinguishes moment methods according to their closure model, e.g. the M_N and P_N closure described in §4.7. The M_N method, although methodically superior to the P_N counterpart, has the disadvantage of excessively high computation times in comparsion with other discretization methods of the Boltzmann equation. This motivates the development of surrogate models to accelerate the M_N closure. Additionally, such a surrogate model should preserve the advantageous mathematical structure of the M_N method.

4.1.1. Related Work on Neural Network-Based Moment Closures

In recent years, various machine learning-based techniques have been proposed in the field of mathematical modeling of moment closures. In [61, 96], the authors pursue two strategies. First, they use a encoder-decoder network to learn the generalized moments the moment system and then learn its closure by reconstruction of the kinetic density. Second, they learn directly the reduced model for the set of generalized moments. In [114], Galilean invariant machine learning methods for partial differential equations are developed using the conservation dissipation formalism. The authors of [166] directly model the kinetic density to close the moment system using an U-Net that considers global information of the solution field. The authors of [112] close the moment system by learning the spatial gradient of the highest order moment. The authors of [61] considers the challenges of imposing physical constrains on machine learning models in the example of Galilean invariance of Boltzmann equation. Iterative exploration strategies of the data space are proposed using of sequential learning of the physical model in pursuit of obtaining optimal datasets.

4.1.2. Novelty and Scientific Contribution

This work explores the capacity of deep neural networks in the application of the Boltzmann equation and its implications on numerical differential equation solvers. We provide two new deep neural network-based structure-preserving minimal entropy closures for the moment system of the Boltzmann equation. The first approach uses an input convex neural network inspired by the architecture of [9] and learns the convex moment to entropy map. By this ansatz, the learned closure automatically inherits all structural properties of the entropy closure for the moment system. The derivative of the network with respect to the moments maps to the corresponding optimal Lagrange multipliers of the entropy minimization problem. The network is trained on the predicted entropy, the Lagrange multipliers, and the reconstructed moments.
The second ansatz of this work is a monotonic neural network that maps the moments directly to the Lagrange multipliers of the entropy minimization problem. We use a penalty function to train the neural network to be monotonic and otherwise use the same loss as in the input convex approach.
Furthermore, we construct a maximum error bound for the interpolation error of input convex neural networks trained in the Sobolev norm. Additionally, we provide insights into the topology of the realizable set, the data-to-solution map of the minimal entropy closure, and its implications on the construction of data-driven closures.
We combine the above insights to propose a data sampling strategy for the neural network-based minimal entropy closures. Finally, we demonstrate our findings in numerical simulation test cases, where we compare the accuracy and computational efficiency of the closures to a traditional numerical solver that provides a benchmark.

This chapter was published in the proceeding of the ICML Conference on Machine Learning 2022 and authored in collaboration with Tianbai Xiao, Cory Hauck, and Martin Frank. The publication can be found at [208] and the code of the described implementations are published in the open source GitHub repositories KiT-RT[1] and NeuralEntropyClosures[2]. The chapter focuses on the authors contribution to the publication.

[1] https://github.com/CSMMLab/KiT-RT
[2] https://github.com/ScSteffen/neuralEntropyClosures

4.1.3. The Chapter in Context of the Dissertation

This chapter presents the author's first neural network-based surrogate model for the Boltzmann moment system, i.e., the first used synergy of neural network function approximation and numerical methods for kinetic equations. Furthermore, the chapter provides a deeper understanding of the minimal entropy closure, leveraged by subsequent chapters.

4.1.4. Chapter Outline

We provide a brief overview of existing numerical methods for the minimal entropy closure and describe the corresponding computational challenges in §4.2. In §4.3, we present our findings for structure-preserving, neural network based surrogate models for the minimal entropy closure. In §4.4, we discuss the structure of the realizable set, state interpolation error bounds, and propose a data-sampling strategy. Lastly, §4.5 presents a range of numerical studies to validate the proposed methods.

4.2. Numerical Methods for Minimal Entropy Closures

We state the relevant equations of §1 to construct a neural network-based surrogate model for the minimal entropy closure. The Boltzmann moment system with linear collision operator is given by

$$\partial_t \mathbf{u}(t,\mathbf{x}) + \nabla_{\mathbf{x}} \cdot \langle \mathbf{v} \otimes \mathbf{m}(\mathbf{v}) f_{\mathbf{u}} \rangle = \langle \mathbf{m}(\mathbf{v}) Q(f_{\mathbf{u}}) \rangle \,, \tag{4.1}$$

with the minimal entropy closure that yields $f_{\mathbf{u}}$ as the minimizer of the convex optimization problem

$$\min_{f \in F_{\mathbf{m}}} \langle \eta(f) \rangle \qquad \text{s.t. } \mathbf{u} = \langle \mathbf{m} f \rangle \,. \tag{4.2}$$

The bracket operator $\langle \cdot \rangle$ denotes as usual the integral over the velocity domain $\mathbf{V} = \mathbb{S}^2$. Its convex dual is given by

$$\boldsymbol{\alpha}_{\mathbf{u}} = \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathbb{R}^n} \phi(\boldsymbol{\alpha}; \mathbf{u}) \tag{4.3}$$

$$\phi(\boldsymbol{\alpha}; \mathbf{u}) = \boldsymbol{\alpha} \cdot \mathbf{u} - \langle \eta_*(\boldsymbol{\alpha} \cdot \mathbf{m}) \rangle \,, \tag{4.4}$$

and we call the maximizer $\boldsymbol{\alpha}_{\mathbf{u}}$ for a given $\mathbf{u}$. By the strong duality of the minimal entropy problem, the maximum of (4.3) equals the minimum of (4.2). We define

$$h(\mathbf{u}) := \langle \eta(f_{\mathbf{u}}) \rangle = \phi(\boldsymbol{\alpha}_{\mathbf{u}}; \mathbf{u}) = \boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{u} - \langle \eta_*(\boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{m}) \rangle \,. \tag{4.5}$$

The minimizer of the primal problem $f_{\mathbf{u}}$ can then be reconstructed using the Lagrange multiplier,

$$f_{\mathbf{u}} = \eta_*'(\boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{m}(\mathbf{v})). \tag{4.6}$$

Lastly, the Hessian of ϕ is given by

$$H(\boldsymbol{\alpha}) = -\left\langle \mathbf{m} \otimes \mathbf{m} \eta_*''(\boldsymbol{\alpha} \cdot \mathbf{m}) \right\rangle . \tag{4.7}$$

Algorithm 4.1: Newton optimizer for the dual entropy optimization with line-search

Input: $\boldsymbol{\alpha}^0$: Initial guess
$\mathbf{u}$: Moment vector
ϵ: Tolerance to minimum
λ: Newton stepsize
$t_{\max}$: Maximal number of iterations

Result: $\boldsymbol{\alpha}_{\mathbf{u}}$: Lagrange multiplier

$t = 0$
$\boldsymbol{\alpha}_0^t \leftarrow \boldsymbol{\alpha}^0$
do
  $\boldsymbol{p}^t \leftarrow \lambda \nabla_{\boldsymbol{\alpha}} \phi_n(\boldsymbol{\alpha}^t; \mathbf{u})$ `/* Compute scaled gradient */`
  $k \leftarrow 0, s \leftarrow 1$
  $\boldsymbol{p}_k^t \leftarrow \boldsymbol{p}^t$
  do
    $\boldsymbol{\alpha}_{k+1}^t \leftarrow \boldsymbol{\alpha}_k^t - s H_n^{-1,t} \boldsymbol{p}^t$ `/* Line-search update */`
    $\boldsymbol{p}_{k+1}^t \leftarrow \lambda \nabla_{\boldsymbol{\alpha}} \phi_n(\boldsymbol{\alpha}_{k+1}^t; \mathbf{u})$ `/* Gradient update */`
    $s \leftarrow \frac{1}{2} s$ `/* Half step size */`
    if $\left\| \boldsymbol{p}_{k+1}^t \right\| < \epsilon$ **then**
      $\boldsymbol{\alpha}_{\mathbf{u}} \leftarrow \boldsymbol{\alpha}_{k+1}^t$ `/* Terminate if minimizer is found */`
      `break`
    $k \leftarrow k + t$
  while $\left\| p_0^t \right\| < \left\| p_k^t \right\|$ *or* $\left\| p_k^t \right\| = \infty$
  $\boldsymbol{\alpha}^{t+1} \leftarrow \boldsymbol{\alpha}_{k+1}^t$
  $t \leftarrow t + 1$
while $\boldsymbol{p}^t < \epsilon$ *and* $t < t_{\max}$
$\boldsymbol{\alpha}_{\mathbf{u}} \leftarrow \boldsymbol{\alpha}^t$

4.2.1. The Newton Optimizer

The analytical structure of the minimal entropy closure problem provides a straightforward numerical algorithm. We solve the finite-dimensional dual optimization problem (4.3) and use the optimizer $\boldsymbol{\alpha}_{\mathbf{u}}$ to reconstruct the primal optimizer $f_{\mathbf{u}}$ for a given $\mathbf{u} \in \mathcal{R}$. Since $\phi(\boldsymbol{\alpha}; \mathbf{u})$ is twice differentiable and convex, a reasonable and commonly used [5, 7, 8, 82, 141, 146, 149] choice for the numerical optimizer is the Newton method with line-search, which converges locally quadratically to $\boldsymbol{\alpha}_{\mathbf{u}}$.
Following the convention of optimizing a convex functional, we redefine the dual problem as a convex minimization problem with objective functional and Hessian

$$\phi_n(\boldsymbol{\alpha}; \mathbf{u}) = -\phi(\boldsymbol{\alpha}; \mathbf{u}), \quad \text{and} \quad H_n(\boldsymbol{\alpha}) = -H(\boldsymbol{\alpha}). \tag{4.8}$$

Then, the Newton optimizer translates to the minimal entropy context as

$$\boldsymbol{\alpha}^{t+1} = \boldsymbol{\alpha}^t - \lambda H_n^{-1}(\boldsymbol{\alpha}^t) \nabla_{\boldsymbol{\alpha}} \phi_n(\boldsymbol{\alpha}^t; \mathbf{u}), \qquad t = 1, 2, \ldots, t_{\max} \tag{4.9}$$

with step-size $\lambda > 0$. The iteration is stopped if the criterion $\|\nabla_{\boldsymbol{\alpha}} \phi_n\| < \epsilon$ is met or a maximal iteration count $t_{\max}$ is reached.
Evaluating and inverting the Hessian H_n is the most expensive part of the Newton iteration. Thus iterative

Algorithm 4.2: One time step for a first order M_N solver

Input: $\mathbf{u}_i^n$: Moments at time n of the grid $\widetilde{\mathbf{X}}$

Result: $\mathbf{u}_i^{n+1}$: Moments at time $n+1$ of the grid $\widetilde{\mathbf{X}}$

for *each* $\mathbf{u}_i^n \in \widetilde{\mathbf{X}}$ **do**
  $\boldsymbol{\alpha}_{\mathbf{u}_i^n} \leftarrow \texttt{Closure}(\mathbf{u}_i^n)$ `/* Entropy closure */`
for *each* $\mathbf{u}_i^n \in \widetilde{\mathbf{X}}$ **do**
  $\mathbf{F}_i \leftarrow \texttt{Upwind}(\boldsymbol{\alpha}_{\mathbf{u}_i^n}, \boldsymbol{\alpha}_{\mathbf{u}_j^n}), \quad \forall j \in N(i)$ `/* Flux construction */`
  $\mathbf{u}_i^{n+1} \leftarrow \texttt{Euler}(\mathbf{u}_i^n, \mathbf{F}_i)$ `/* Time integration */`

approaches to determine the best step-size at the current iteration t are frequently used. Backtracking Line-search [11] aims to find loosely the local minimum

$$\min_{a>0} \phi_n \left(\boldsymbol{\alpha}^t - a\lambda {H_n}^{-1}(\boldsymbol{\alpha}^t) \nabla_{\boldsymbol{\alpha}} \phi_n(\boldsymbol{\alpha}^t; \mathbf{u}) \right) \tag{4.10}$$

to reduce the number of necessary gradient and Hessian evaluations. The minimizer is then set as $\boldsymbol{\alpha}^{t+1}$ for the original Newton iterator. The Newton optimizer with line-search for the dual entropy problem is summarized in Algorithm 4.1. The optimization tolerance ϵ measures the difference between the reconstructed moment $\langle \mathbf{m} \exp(\boldsymbol{\alpha} \cdot \mathbf{m}) \rangle$ and the original moment $\mathbf{u}$.

4.2.2. Computational Challenges

Spatial-temporal discretization strategies of the Boltzmann moment system using finite volume, see §3.4 and [141, 147], or discontinuous Galerkin [7] schemes generally lead to an iterative time-stepping scheme on a spatial grid. By construction of the M_N method, the minimal entropy closure has to be computed in each grid cell for each iteration step, see Algorithm 4.2. Although the Newton iterator converges locally quadratically, calculation of the optimization problem consumes the majority of the overall simulation wall-time, see Table 4.4 and [82, 141], where up to 90% of the simulation time is spent on the entropy closure computation.

4.3. Neural Network-Based Entropy Approximations

The goal of this work is to create a surrogate model for the minimal entropy closure using a neural network. The purpose of this model is to accurately and efficiently approximate the Boltzmann moment system, bypassing the computationally expensive Newton solver. Although training a neural network is resource-intensive, its inference is typically fast and can be easily parallelized. Additionally, the computational cost of the network inference is only dependent on its architecture, unlike the Newton iteration, which is dependent on the condition number of the closure problem. We propose two neural network architectures as data-driven surrogate models for the minimal entropy closure. The task of these surrogate models is to approximate the solution that maps a set of moments $\mathbf{u}$ to the kinetic density $f_{\mathbf{u}}(\mathbf{v})$ to with minimal entropy, i.e.,

$$\mathcal{U} : \mathbb{R}^n \to F_m, \quad \mathbf{u} \mapsto f_u(\mathbf{v}), \tag{4.11}$$

corresponding to the minimal entropy closure given the Maxwell-Boltzmann entropy density $\eta : D \to \mathbb{R}$ with $D = [0, \infty)$, where it holds that

$$\eta(f) = f\log(f) - f, \quad \eta'(f) = \log(f), \quad \eta_*(z) = \exp(z), \quad \eta'_*(z) = \exp(z)\,. \tag{4.12}$$

Direct surrogate model approximation of $f_{\mathbf{u}}$ with preservation of its inherent structure is a daunting and computationally expensive task since $F_m \subset L_1$ is a function space. Instead, we use the minimal entropy ansatz to build surrogate models that map onto a finite-dimensional vector space.

4.3.1. Normalized Moments

The structure of the underlying data is crucial for the construction of meaningful machine-learning models. In the following, we derive a normalization and scaling formula and describe helpful relations between the moment $\mathbf{u}$, Lagrange multiplier $\boldsymbol{\alpha}_{\mathbf{u}}$ and the entropy functional h. The normalized closure is leveraged in §4.3.2 and §4.4 for error control and data-sampling strategies.
To construct a proper neural network-based approximation, we introduce the normalized realizable set $\overline{\mathcal{R}}$ and its reduced counterpart $\widetilde{\mathcal{R}}$:

$$\overline{\mathcal{R}} = \{\mathbf{u} \in \mathcal{R} : u_0 = 1\} \subset \mathbb{R}^n\,, \tag{4.13}$$

$$\widetilde{\mathcal{R}} = \left\{\mathbf{w} \in \mathbb{R}^{n-1} : \left[1, \mathbf{w}^T\right]^T \in \overline{\mathcal{R}}\right\} \subset \mathbb{R}^{n-1}\,. \tag{4.14}$$

For $D = [0, \infty)$ and $\mathbf{V} = \mathbb{S}^2$, i.e., the unit sphere, both $\overline{\mathcal{R}}$ and $\widetilde{\mathcal{R}}$ are bounded [129, 187]. We further introduce the normalization operator

$$\bar{\cdot}\colon \mathbb{R}^n \to \mathbb{R}^n \text{ defined as } \overline{\mathbf{u}} := \mathbf{u}/u_0 \tag{4.15}$$

and the reduction operator

$$(\cdot)_{\#}\colon \mathbb{R}^n \to \mathbb{R}^{n-1} \text{ defined such that for } \mathbf{u}_{\#} \in \mathbb{R}^{n-1} \text{ we have } \mathbf{u} = \left[u_0, \mathbf{u}_{\#}^{\top}\right]^{\top}. \tag{4.16}$$

Then, for the entropy closure, we have $\mathbf{u} \in \mathcal{R}$, $\overline{\mathbf{u}} \in \overline{\mathcal{R}}$ and $\overline{\mathbf{u}}_{\#} \in \widetilde{\mathcal{R}}$ and $\boldsymbol{\alpha}_{\overline{u}}$ is the Lagrange multiplier of $\overline{\mathbf{u}}$. Let us define the function

$$\vartheta(\boldsymbol{\beta}) = -\log(\langle \exp(\boldsymbol{\beta} \cdot \mathbf{m}_{\#})\rangle) \tag{4.17}$$

for $\boldsymbol{\beta} \in \mathbb{R}^{n-1}$, which we use to reconstruct the order zero Lagrange multiplier for normalized moments.

Lemma 4.1 (Normalized Lagrange Multiplier Reconstruction)
Assume $m_0(\mathbf{v}) = 1$. For a normalized moment $\overline{\mathbf{u}}$ and its Lagrange multiplier $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ defined by Eq. (4.3)*, we have the relation*

$$\alpha_{\overline{\mathbf{u}},0} = \vartheta\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\right). \tag{4.18}$$

The proof is given in §4.7.

Lemma 4.2 (Lagrange Multiplier Scaling)
Assume $m_0(\mathbf{v}) = 1$. The Lagrange multipliers $\boldsymbol{\alpha}_{\mathbf{u}}$ of a moment $\mathbf{u}$ and its normalized counterpart $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ of $\overline{\mathbf{u}}$ have the relation

$$\boldsymbol{\alpha}_{\mathbf{u}} = \left[\vartheta\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\right) + \log(u_0), (\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}^{\top}\right]^{\top}, \tag{4.19}$$

where u_0 is the first element of $\mathbf{u}$. We have for $h(\overline{\mathbf{u}})$ and $h(\mathbf{u})$ the relation

$$h(\mathbf{u}) = u_0\left(h(\overline{\mathbf{u}}) + \log(u_0)\right). \tag{4.20}$$

The proof is given in [8] and stated in §4.7 for the sake of completeness. It also implies that $(\boldsymbol{\alpha}_{\mathbf{u}})_{\#} = (\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}$. Lastly, we define $\hat{h} : \mathbb{R}^{n-1} \to \mathbb{R}$ as

$$\hat{h}(\overline{\mathbf{u}}_{\#}) = h\left([1, \overline{\mathbf{u}}_{\#}]\right) = h(\overline{\mathbf{u}}). \tag{4.21}$$

These scaling relations allow to train the neural network only on $\overline{\mathcal{R}}$, which is bounded, in contrast to $\mathcal{R}$, and integrate the trained neural network seamlessly into the kinetic solver that operates on $\mathcal{R}$.

4.3.2. Neural Network-Based Surrogate Model Architectures

In the following, we present two ideas for entropy closures utilizing neural networks. Our approach is grounded on the premise that any convex, twice differentiable function is a suitable entropy for the linear Boltzmann equation. To denote a neural network with parameters θ, we use the symbol N_θ. The approximation of quantities by the network is indicated by a superscript θ.

ICNN - Input Convex Neural Network Approximation of the Entropy Functional

In the first concept, we introduce a twice-differentiable and convex approximation, denoted as $\hat{h}^\theta$, to the target function $\hat{h}$. The neural network N_θ utilized for this purpose has been constructed to be convex and serves as the approximator $\hat{h}^\theta$ directly, i.e.,

$$\hat{h}^\theta(\overline{\mathbf{u}}_{\#}) = \mathrm{N}_\theta(\overline{\mathbf{u}}_{\#}) \approx \hat{h}(\overline{\mathbf{u}}_{\#}). \tag{4.22}$$

Using Lemma 4.1, the neural network approximation $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\theta$ to the Lagrange multiplier $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ is given by

$$\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\theta = \left[\vartheta\left(\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\theta\right)_{\#}\right), \left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\theta\right)_{\#}^{\top}\right]^{\top}. \tag{4.23}$$

By Eq. (1.64), the network derivative $\boldsymbol{\alpha}_{\#}^\theta$ approximates the Lagrange multiplier $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}$, i.e.,

$$(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\theta)_{\#} = \nabla_{\overline{\mathbf{u}}_{\#}}\mathrm{N}_\theta(\overline{\mathbf{u}}_{\#}) \approx \nabla_{\overline{\mathbf{u}}_{\#}}\hat{\mathrm{h}}(\overline{\mathbf{u}}_{\#}) = (\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}. \tag{4.24}$$

In practice, the network derivative can be computed using automatic differentiation [91]. Using Eq. (1.62) the kinetic density

$$f_{\overline{\mathbf{u}}}^\theta = \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\theta \cdot \mathbf{m}) \approx \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m}) = f_{\overline{\mathbf{u}}}, \tag{4.25}$$

is computed to close the moment system. The approximated moment reconstruction is then given by

$$\overline{\mathbf{u}}^\theta = \left\langle \mathbf{m} \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\theta \cdot \mathbf{m}) \right\rangle \approx \langle \mathbf{m} \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m}) \rangle = \overline{\mathbf{u}}, \tag{4.26}$$

Algorithm 4.3: ICNN closure training

Input: $X_T = \bigcup_k X_{B_k}$: Training data-set $\left\{\left(h_i, \overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}\right)\right\}_{i\in T}$, partitioned in k_B batches
N_θ: Network architecture $\mathrm{N}_\theta : \overline{\mathbf{u}}_\# \mapsto \hat{h}(\overline{\mathbf{u}}_\#)$
θ^0: Weight initialization of the network
t_{epoch}: Maximum number of training iterations

Result: N_{θ^*}: Trained network for the minimal entropy closure

for $t = 0$ *to* $t = t_{\text{epoch}}$ **do**
  $\theta^{k=0} \leftarrow \theta^t$
  for $k = 0$ *to* $k = k_B$ **do**
    Load mini-batch X_{B_k}
    $\hat{h}_i^\theta \leftarrow \mathrm{N}_\theta\left((\overline{\mathbf{u}}_i)_\#\right),\ \forall i \in B_k$ /* Entropy approximation */
    $(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\# \leftarrow \nabla_{\overline{\mathbf{u}}_\#}\mathrm{N}_\theta\left((\overline{\mathbf{u}}_i)_\#\right),\ \forall i \in B_k$ /* Lagrange multiplier approximation */
    $\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i} \leftarrow \left[\vartheta\left((\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\#\right), (\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\#^\top\right]^\top,\ \forall i \in B_k$ /* Reconstruct full multiplier */
    $\overline{\mathbf{u}}_i^\theta \leftarrow \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i} \cdot \mathbf{m}\right)\right\rangle,\ \forall i \in B_k$ /* Reconstruct moment vector */
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{ICNN}}(X_{B_k}; N_\theta)$ /* Compute loss */
    $\theta^{k+1} \leftarrow \theta^k - \nabla_\theta \mathcal{L}$ /* Update network weights */
  $\theta^{t+1} \leftarrow \theta^{k_B}$
$\theta^* \leftarrow \theta^{t_{\text{epoch}}}$ /* Save final network weights */

where we use the definition of the moment $\mathbf{u}$. The network is trained on a loss function given by the sum of mean squared errors of the network prediction, the network derivative, and the moment reconstruction,

$$\mathcal{L}_{\text{ICNN}}(X_B; N_\theta) = \frac{1}{|B|}\sum_{i\in B}\left(\left\|\hat{h}((\overline{\mathbf{u}}_i)_\#) - \hat{h}^\theta((\overline{\mathbf{u}}_i)_\#)\right\|_2^2 + \lambda\left\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}},i})_\# - (\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\#\right\|_2^2 + \left\|\overline{\mathbf{u}}_i - \overline{\mathbf{u}}_i^\theta\right\|_2^2\right). \tag{4.27}$$

where B is the size of one batch of the training data set, $X_T = \left\{\overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}, h(\overline{\mathbf{u}})_i\right\}$. The parameter λ is used to scale the loss in $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#$ to the same range as the loss in $\hat{h}$ and $\overline{\mathbf{u}}$. The size of the training set is denoted by T. The neural network training is summarized in Algorithm 4.3.
Training the neural network on the Lagrange multiplier $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ and $\hat{h}(\overline{\mathbf{u}}_\#)$ corresponds to fitting the neural network approximation to the entropy functional $\hat{h}$ in Sobolev norm, which increases training performance [53] compared to standard regression loss.
During inference of the neural network in a kinetic solver, we gather the moments $\mathbf{u}$ for all grid cells of the spatial discretization from the current iteration of the used finite volume scheme. After normalization $\overline{\mathbf{u}}$, N_θ approximates $(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\#$ just as in Eq. (4.24), followed by a scaled reconstruction, given by Lemma 4.2. The closure is then computed using Eq. (4.25) and inference is summarized in Algorithm 4.4

IMNN - Input Monotonic Neural Network Approximation of Lagrange multipliers

The idea of the second neural network closure for the dual minimal entropy problem in Eq. (4.3), makes use of the following characterization of multivariate convex functions via the monotonicity of their gradients [18].

Algorithm 4.4: ICNN inference within a kinetic solver

Input: $\mathbf{u}_i$: Moments of the grid $\widetilde{\mathbf{X}}$

Result: $f^{\theta}_{\mathbf{u},i}$: Reconstructed kinetic densities of the grid $\widetilde{\mathbf{X}}$

for *each* $\mathbf{u}_i \in \widetilde{\mathbf{X}}$ **do**

$\hat{h}^{\theta}_i \leftarrow \mathrm{N}_\theta\left((\overline{\mathbf{u}}_i)_\#\right)$ /* Entropy approximation */

$(\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}},i})_\# \leftarrow \nabla_{\overline{\mathbf{u}}_\#}\mathrm{N}_\theta\left((\overline{\mathbf{u}}_i)_\#\right)$ /* Lagrange multiplier approximation */

$\boldsymbol{\alpha}^{\theta}_{\mathbf{u},i} \leftarrow \left[\vartheta((\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}},i})_\#) + \log(u_{0,i}), (\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}},i})^\top_\#\right]^\top$ /* Reconstruct and scale α_0 */

$f^{\theta}_{\mathbf{u},i} \leftarrow \exp\left(\boldsymbol{\alpha}^{\theta}_{\mathbf{u},i}\cdot\mathbf{m}\right)$ /* Reconstruct kinetic density */

Lemma 4.3 (Convexity and Monotonic Gradients)
Let $U \subset \mathbb{R}^n$ be a convex set. A function $G : U \to \mathbb{R}^n$ is monotonic, if and only if $(G(\mathbf{x}_1)-G(\mathbf{x}_2))\cdot(\mathbf{x}_1-\mathbf{x}_2) \geq 0$ for all $\mathbf{x}_1, \mathbf{x}_2 \in U$. Let $g : U \to \mathbb{R}$ be differentiable. Then g is convex, if and only if $\nabla g : U \to \mathbb{R}^n$ is monotonic.

The proof is provided in [85]. Consequently, if the map $\overline{\mathbf{u}}_\# \to (\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#$ is monotonic for all $\overline{\mathbf{u}}_\# \in \widetilde{\mathcal{R}}$, then the corresponding entropy functional is $\hat{h}$ is convex in $\overline{\mathbf{u}}_\#$. Let N_θ be a trained monotonic network with parameters θ to approximate $(\boldsymbol{\alpha}_{\mathbf{u}})_\#$, i.e.,

$$(\boldsymbol{\alpha}_{\mathbf{u}})^{\theta}_\# = \mathrm{N}_\theta(\overline{\mathbf{u}}_\#) \approx (\boldsymbol{\alpha}_{\mathbf{u}})_\#. \tag{4.28}$$

Using again Lemma 4.1, we get

$$\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}}} = \left[\vartheta\left(\left(\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}}}\right)_\#\right), \left(\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}}}\right)^\top_\#\right]^\top. \tag{4.29}$$

From here the reconstruction of $f_{\overline{\mathbf{u}}}$ and $\overline{\mathbf{u}}$ is given analogously to the convex case, i.e.,

$$f^{\theta}_{\overline{\mathbf{u}}} = \exp(\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}}}\cdot\mathbf{m}) \approx \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}\cdot\mathbf{m}) = f_{\overline{\mathbf{u}}}, \tag{4.30}$$

$$\overline{\mathbf{u}}^{\theta} = \left\langle \mathbf{m}\exp(\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}}}\cdot\mathbf{m})\right\rangle \approx \langle \mathbf{m}\exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}\cdot\mathbf{m})\rangle = \overline{\mathbf{u}}. \tag{4.31}$$

Lastly, we use the definition of the dual objective functional ϕ and the approximated $\overline{\mathbf{u}}^{\theta}$ and $\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}}}$ to approximate the entropy functional $h(\overline{\mathbf{u}})$ as

$$\hat{h}^{\theta}(\overline{\mathbf{u}}_\#) = h^{\theta}(\overline{\mathbf{u}}) = \phi(\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}}}; \overline{\mathbf{u}}) \approx h(\overline{\mathbf{u}}). \tag{4.32}$$

The monotonicity criterion of Lemma 4.3 needs to be checked pointwise and does not provide insights in how to construct a everywhere monotonic function. To the best of our knowledge, there exists no constructive definition of a multidimensional monotonic function. Thus we need to enforce the monotonicity of the neural network by a penalty function.

Definition 4.4 (Monotonicity Loss)
Consider a neural network $N_\theta : \mathbf{x} \mapsto \mathbf{y}$. Let X_B be a batch of the training data set. The monotonicity loss is defined as

$$\mathcal{L}_{mono}(X_B; N_\theta) = \frac{1}{|B|^2}\sum_{i\in B}\sum_{j\in B} ReLU\left(-\left(N_\theta(\mathbf{x}_i) - N_\theta(\mathbf{x}_j)\right)\cdot\left(\mathbf{x}_i - \mathbf{x}_j\right)\right). \tag{4.33}$$

The function ReLU: $\mathbb{R} \to \mathbb{R}$ is defined as usual in a pointwise manner,

$$ReLU(a) = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{if } a \leq 0. \end{cases} \tag{4.34}$$

Algorithm 4.5: IMNN closure training

Input: $X_T = \bigcup_k X_{B_k}$: Training data-set $\left\{\left(h_i, \overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}\right)\right\}_{i\in T}$, partitioned in k_B batches
N_θ: Network architecture $\mathrm{N}_\theta : \overline{\mathbf{u}}_\# \mapsto (\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#$
θ^0: Weight initialization of the network
t_{epoch}: Maximum number of training iterations

Result: N_{θ^*}: Trained network for the minimal entropy closure

for $t = 0$ *to* $t = t_{\text{epoch}}$ **do**
  $\theta^{k=0} \leftarrow \theta^t$
  for $k = 0$ *to* $k = k_b$ **do**
    Load mini-batch $X_{B,k}$
    $(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\# \leftarrow \mathrm{N}_\theta\left((\overline{\mathbf{u}}_i)_\#\right),\ \forall i \in B_k$ /* Lagrange multiplier approximation */
    $\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i} \leftarrow \left[\vartheta\left((\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\#\right), (\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})^\top_\#\right]^\top,\ \forall i \in B_k$ /* Reconstruct full multiplier */
    $\overline{\mathbf{u}}^\theta_i \leftarrow \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i} \cdot \mathbf{m}\right)\right\rangle,\ \forall i \in B_k$ /* Reconstruct moment vector */
    $h^\theta_i \leftarrow \phi(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i}; \overline{\mathbf{u}}_i),\ \forall i \in B_k$ /* Reconstruct entropy functional */
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{IMNN}}(X_{B_k}; N_\theta)$ /* Compute loss */
    $\theta^{k+1} \leftarrow \theta^{k_B} - \nabla_\theta \mathcal{L}$ /* Update network weights */
  $\theta^{t+1} \leftarrow \theta^k$
$\theta^* \leftarrow \theta^{t_{\text{epoch}}}$ /* Save final network weights */

Algorithm 4.6: IMNN inference within a kinetic solver

Input: $\mathbf{u}_i$: Moments of the grid $\widetilde{\mathbf{X}}$

Result: $f^\theta_{\mathbf{u},i}$: Reconstructed kinetic densities of the grid $\widetilde{\mathbf{X}}$

for *each* $\mathbf{u}_i \in \widetilde{\mathbf{X}}$ **do**
  $(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\# \leftarrow \mathrm{N}_\theta\left((\overline{\mathbf{u}}_i)_\#\right)$ /* Lagrange multiplier approximation */
  $\boldsymbol{\alpha}^\theta_{\mathbf{u},i} \leftarrow \left[\vartheta((\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\#) + \log(u_{0,i}), (\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})^\top_\#\right]^\top$ /* Reconstruct and scale α_0 */
  $f^\theta_{\mathbf{u},i} \leftarrow \exp\left(\boldsymbol{\alpha}^\theta_{\mathbf{u},i} \cdot \mathbf{m}\right)$ /* Reconstruct kinetic density */

The monotonicity loss evaluates the monotonicity property for all data points of the training data set in a pairwise fashion. If the dot product is negative, the property is violated and the value of the loss is increased by the current dot product. The overall loss functional of the neural network training is then given by

$$\begin{aligned}\mathcal{L}_{\text{IMNN}}(X_B; N_\theta) =& \frac{1}{|B|}\sum_{i\in B}\left(\left\|\hat{h}((\overline{\mathbf{u}}_i)_\#) - \hat{h}^\theta((\overline{\mathbf{u}}_i)_\#)\right\|_2^2 + \lambda\left\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}},i})_\# - (\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i})_\#\right\|_2^2 + \left\|\overline{\mathbf{u}}_i - \overline{\mathbf{u}}^\theta_i\right\|_2^2\right) \\ &+ \mathcal{L}_{\text{mono}}(X_B; N_\theta).\end{aligned} \tag{4.35}$$

Note, that we only validate the monotonicity of N_θ pointwise on the training data. As a consequence, the mathematical structures of the resulting moment closure are only preserved in an empirical sense, i.e., if the realizable set and more importantly, the set of Lagrange multipliers is sampled densely. The neural network training is summarized by Algorithm 4.5 and the network inference is displayed by Algorithm 4.6.

4.3.3. Structural Properties of the Neural Network-Based Closure

We briefly examine the structural properties of a convex neural network-based entropy closure, which are stated in §1.3. The invariant range property of f_θ depends solely on the range of η'_*. Since both, the ICNN and IMNN models use the ansatz (4.6) of the original entropy closure (with the Maxwell-Boltzmann entropy), the surrogate models are of invariant range.
Consider the elements of the moment system (4.1) that are collision invariants $m_i \in \mathbb{E}$. For any kinetic density, including the neural network-based reconstruction $f^\theta_{\mathbf{u}}$, the right-hand side vanishes, i.e.,

$$\left\langle m_i Q(f^\theta_{\mathbf{u}}) \right\rangle = 0, \tag{4.36}$$

since this is purely a property of the collision operator and not of the construction of $f^\theta_{\mathbf{u}}$. This yields the conservation law

$$\partial_t u_i + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} m_i f^\theta_{\mathbf{u}} \right\rangle = 0. \tag{4.37}$$

Next, the ICNN-based approximation of h for a normalized moment $\overline{\mathbf{u}}$. The structure preservation of the input convex surrogate model follows the idea of [7], where a convex approximation of h acts as the surrogate entropy of the Boltzmann moment system and preserves hyperbolicity and dissipates the system's entropy.

Theorem 4.5 (Hyperbolicity and Entropy Dissipation of the ICNN Closure)
The moment system (4.1) *with ICNN-based moment closure is symmetric hyperbolic in the variable* $\mathbf{u}$*. Furthermore, the neural network approximation*

$$h^\theta(\mathbf{u}) = u_0 \hat{h}^\theta(\overline{\mathbf{u}}) + u_0 \log(u_0) \tag{4.38}$$

acts as an entropy of (4.1)*, i.e., the solution* u^θ *follows the entropy dissipation law*

$$\partial_t h^\theta(\mathbf{u}) + \nabla_{\mathbf{u}} \cdot \mathbf{j}^\theta(\mathbf{u}) \leq 0, \tag{4.39}$$

where $\mathbf{j}^\theta$ *is given by*

$$\mathbf{j}(\mathbf{u})^\theta = [\mathbf{j}(\mathbf{u})^\theta_1, \dots, \mathbf{j}(\mathbf{u})^\theta_d]^\top, \text{ with } \mathbf{j}(\mathbf{u})^\theta_i = \left\langle v_i \eta(f^\theta_u) \right\rangle \tag{4.40}$$

The proof is provided in §4.7.
It is worth noting that only the approximation based on the ICNN meets the criteria for hyperbolicity and entropy dissipation. In contrast, in the IMNN model non-convexity is penalized, but not ensured everywhere by construction.

4.3.4. Architecture and Implementation Details

We briefly present the model details of the ICNN and IMNN closure.

ICNN Implementation

Convex neural networks have been inspected in [9], where the authors propose several deep neural networks architectures that are strictly convex with respect to the input variables by design.

Lemma 4.6 (Convex Function Concatenation [27])

1. *Let $g_i(\mathbf{x})$, $i = 1, \dots, m$ be convex and $\boldsymbol{c} \in \mathbb{R}^m$ a vector with $c_i \geq 0$, $i = 1, \dots, m$, then*

$$f(\mathbf{x}) = \boldsymbol{c} \cdot \boldsymbol{g}(\mathbf{x}) \tag{4.41}$$

   is convex in $\mathbf{x}$.

2. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be the concatenation of the functions $h : \mathbb{R}^k \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}^k$. Then $f(x) = h(g(x))$ is convex, if h is convex, h is non-decreasing in each argument and all $g_{i=1,\dots,k}$ are convex.*

For proof, we refer to [27, §3.2.]

Theorem 4.7 (Input Convex Neural Networks (ICNN))
Let $W_k^z \in \mathbb{R}^{n_k \times n_{k-1}}$, $W_k^x \in \mathbb{R}^{n_k \times n}$ and $\mathbf{b}_k \in \mathbb{R}^{n_k}$ be the weights and biases of layer k of network N_θ. Assume that

$$(W_k^z)_{i,j} \geq 0, \qquad \forall i = 1, \dots, n_k, \text{ and } \forall j = 1, \dots, n_{k-1}, \tag{4.42}$$

and that the layer activation $\sigma_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$, defined as a point-wise evaluation

$$\boldsymbol{\sigma}_k(\boldsymbol{a}) = \left[\sigma_k(a_1), \dots, \sigma_k(a_{n_k})\right]^\top, \tag{4.43}$$

and is assumed to be convex and non-decreasing and twice differentiable in each argument. Then a network $N_\theta : \mathbb{R}^n \to \mathbb{R}$ with M layers, defined as

$$N_\theta(\mathbf{x}) = \mathbf{z}_M \tag{4.44}$$

$$\mathbf{z}_k = \sigma_k(W_k^{\mathbf{z}} \mathbf{z}_{k-1} + W_k^{\mathbf{x}} \mathbf{x} + \mathbf{b}_k), \qquad k = 2, \dots, M, \tag{4.45}$$

$$\mathbf{z}_1 = \sigma_1(W_1^{\mathbf{x}} \mathbf{x} + \mathbf{b}_1) \tag{4.46}$$

is convex in $\mathbf{x}$ *and twice differentiable with respect to* $\mathbf{x}$.

We show the proof in §4.7. Exemplary choices for σ_k are the strictly convex softplus function

$$\sigma : \mathbb{R} \to \mathbb{R}_+ : \quad \sigma(y) = \ln(\exp(y) + 1), \tag{4.47}$$

the ReLU activation, see Eq. (4.34), or simply the identity map. The ICNN implementation of this chapter uses the softplus activation for layer $k = 1, \dots, M-1$ and ReLU for $k = M$. During the network training, the non-negativity of $W_k^{\mathbf{z}}$ can be achieved by applying a projection onto $\mathbb{R}_+$ after the gradient descent step. The authors of [43] have shown, that N_θ constructed by Theorem 4.7 with ReLU as the choice for σ_k and $W_k^{\mathbf{x}} = 0$ is dense in the space of convex functions. The main argument in their proof is based on a standard argument of density of piecewise affine functions in the space of Lipschitz continous convex functions on a compact set. Two practical downsides of this theoretical property appear. First, situations may arise, where an exponential number of layers are needed to achieve good approximation

accuracy in the proof [43]. Second, the Lipschitz constant may grow very large in the context of entropy approximation, as $\mathbf{u}$ approaches $\partial\mathcal{R}$, which is discussed in $4.4.2.

To achieve better training performance, we employ several commonly used modifications to the ICNN architecture and show that they preserve convexity.

Lemma 4.8 (Mean-shift and Decorrelation Layer)
Let μ_X be the mean of the training data set X and Λ_X the diagonal matrix of the eigenvalues of the covariance matrix of X, i.e.

$$\Lambda_X = \text{diag}(\lambda_1, \dots, \lambda_n), \tag{4.48}$$

with λ_i as the eigenvalues of X. The mean-shift and de-correlation layer is then defined as

$$\mathbf{z}_1 = \sigma(W_1^{\mathbf{x}} \Lambda_X^T (\mathbf{x} - \mu_X) + \mathbf{b}_1), \tag{4.49}$$

for a convex activation function σ. This layer is convex and twice differentiable with respect to $\mathbf{x}$.

We prove the statement in §4.7. In the implemented ICNN entropy approximation model, we replace layer $k = 1$ of the original ICNN architecture 4.7 by a mean-shift and de-correlation layer.
Centering and de-correlation of the input data accelerate training since the gradient of the first layer directly scales with the mean of the input data. Thus a nonzero mean can cause zig-zagging of the gradient vector [157].

IMNN Implementation

No particular design choices about the neural network are made to enforce monotonicity, since the characterization of monotonic functions are not constructive. Normalization and the mean-shift and de-correlation layers in Eq. (4.49) are used analogously to the input convex neural network. The core network architecture consists ResNet blocks which were first presented in [104]. The ResNet layer is given by

$$z_k = W_k^* \sigma((z_{k-1})) + b_k^* + z_{k-1}, \tag{4.50}$$

The advantage of the skip connection in Eq. (4.50) is to mitigate the gradient vanishing problem for deep neural networks. We choose σ_k as ReLU for $k = 1, \dots, M$.
Furthermore, we include a batch normalization (BN) layer in front of each activation, which reduces the problem internal covariance shift [119]. Batch normalization is given by

$$z_k = W_k \frac{z_{k-1} - \mathbb{E}[z_{k-1}]}{\sqrt{\text{Var}[z_{k-1}] + \epsilon}} + b_k, \tag{4.51}$$

where $\mathbb{E}[z_{k-1}]$ and $\text{Var}[z_{k-1}]$ denote the expectation value and the variance of the current batch of training data, respectively. Note that a batch normalization layer may destroy convexity if W_k contains negative elements and thus is not used in the ICNN model.

4.4. Data Sampling Strategies for the Minimal Entropy Closure

The function approximation problem given by neural network training can be regarded as a numerical analysis or a statistics problem. The trained model N_{θ^*} depends on the distribution of its training data X_T and the generalization error further depends on the distribution $P_{\mathbf{x}}$ of the inference data. The general objective is to structure the network training by choice of architecture, training-data, and loss-functional to minimize $\mathcal{T}$ and $\mathcal{G}$, given by Definition2.2.
In the context of the minimal entropy closure, this translates to the following questions:

- What is the analytical structure of the training data set X_T consisting of $\overline{\mathbf{u}}$, $\alpha_{\overline{\mathbf{u}}}$ and $h(\overline{\mathbf{u}})$?
- How can we estimate the distribution $P_{\mathbf{x}}$ at inference time?
- How should we sample X_T to minimize $\mathcal{T}$ and $\mathcal{G}$?

A popular method for generating data for neural network models that interact with numerical methods for partial differential equations is to generate the training data by direct simulation, see e.g. [112, 158, 240]. The underlying assumption is that the resulting training data set X_T is generated with a similar distribution to $P_{\mathbf{x}}$. However, failure to anticipate important simulation configurations may result in unreasonably high test errors. Another way to sample data using simulation is to use Fourier series with random coefficients [113] to generate initial and boundary conditions, to eliminate some human bias in the specification of simulation runs.
The minimal entropy closure is a self-contained problem with a clear data to solution map. This provides more options to sample training data than common machine learning applications.

4.4.1. The Realizable Set of the Minimal Entropy Closure

The minimal entropy optimization problem in Eq. (4.3) and the set of realizable moments $\mathcal{R}$ is studied in detail by [7, 52, 100, 124, 125, 126, 161, 196] and we briefly review the works. As a reminder, the set of all moments corresponding to a kinetic density f with $\text{Range}(f) \subset D$ is called the realizable set

$$\mathcal{R} = \{\mathbf{u} : \langle \mathbf{m} g \rangle = \mathbf{u},\ g \in F_{\mathbf{m}}\} \tag{4.52}$$

with

$$F_{\mathbf{m}} = \{f \in \text{Dom}(Q) : \text{Range}(f) \subset D \text{ and } \langle \mathbf{m} f \rangle < \infty\}. \tag{4.53}$$

The realizable set $\mathcal{R}$ is only dependent on the domain D of η, $\mathbf{V}$ and $\mathbf{m}$, and can be described as the set of all moments corresponding to kinetic densities f that fulfill the invariant range condition of the kinetic equation.
The characterization endeavors of $\mathcal{R}$ use the fact that the realizable set is uniquely defined by its boundaries [139]. First, we remark that the realizable set $\mathcal{R} \subset \mathbb{R}^n$ of the entropy closure problem of order N is generally an unbounded convex cone. To see this consider the moment of order zero, $u_0 = \langle f \rangle \in (0, \infty)$ for any kinetic density function $f \in F_m$. For a fixed u_0, and a bounded velocity space $\mathbf{V}$, the subset of the corresponding realizable moments of higher order is bounded and convex [129, 187].

Theorem 4.9 (Characterization of $\widetilde{\mathcal{R}}$ [129, 187])
Let η be the Maxwell-Boltzmann entropy and $\mathbf{m}$ a monomial basis up to order $N = 4$ and $\mathbf{V} = [-1, 1] \subset \mathbb{R}$. Then, $\widetilde{\mathcal{R}}$ can be described by

$$1 \geq \overline{u}_1 \geq -1, \tag{4.54}$$

$$1 \geq \overline{u}_2 \geq (\overline{u}_1)^2, \tag{4.55}$$

$$\overline{u}_2 - \frac{(\overline{u}_1 - \overline{u}_2)^2}{1 - \overline{u}_1} \geq \overline{u}_3 \geq -\overline{u}_2 + \frac{(\overline{u}_1 + \overline{u}_2)^2}{1 + \overline{u}_1}, \tag{4.56}$$

$$\overline{u}_2 - \frac{(\overline{u}_1 - \overline{u}_3)^2}{(1 - \overline{u}_2)} \geq \overline{u}_4 \geq \frac{(\overline{u}_2)^3 + (\overline{u}_3)^2 - 2\overline{u}_1\overline{u}_2\overline{u}_3}{\overline{u}_2 - (\overline{u}_1)^2}, \tag{4.57}$$

For proof, we refer to [187]. Higher moment order moments for $V = [-1, 1]$ can be characterized using the more general results in [52].

4.4.2. The Boundary of the Normalized Realizable Set

We use Theorem 4.9 to inspect the dual minimal entropy closure Eq. (4.3) near $\partial\widetilde{\mathcal{R}}$, which becomes increasingly difficult to solve [8]. Close to $\partial\widetilde{\mathcal{R}}$, the condition number of the Hessian matrix H of the dual entropy problem can become arbitrarily large, which causes numerical solvers to fail rather unforgivingly. This situation appears for moments of highly anisotropic distributions, vacuum states, where $f(x, \cdot, t) = 0$ or in the presence of strong sources [8]. At the boundary $\partial\widetilde{\mathcal{R}}$, the Hessian H is singular, and the minimal entropy problem has no solution. In the space of Lagrange multipliers, this translates to $\|\boldsymbol{\alpha}_{\overline{\mathbf{u}}}\| \to \infty$, which leads to numerical instabilities when computing the reconstruction of $u = \langle \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m}) \rangle$.
The simplest case of the minimal entropy closure, the 1D M_1 closure, already incorporates these difficulties. Here we have $\overline{u}_1 = \overline{\mathbf{u}}_\#$. We can see in Fig. 4.1(a) the map

$$\overline{\mathbf{u}}_\# \mapsto \left[\alpha_{\overline{\mathbf{u}},0}, \alpha_{\overline{\mathbf{u}},1}\right]^\top \tag{4.58}$$

and in Fig . 4.1(b) the minimal entropy functional

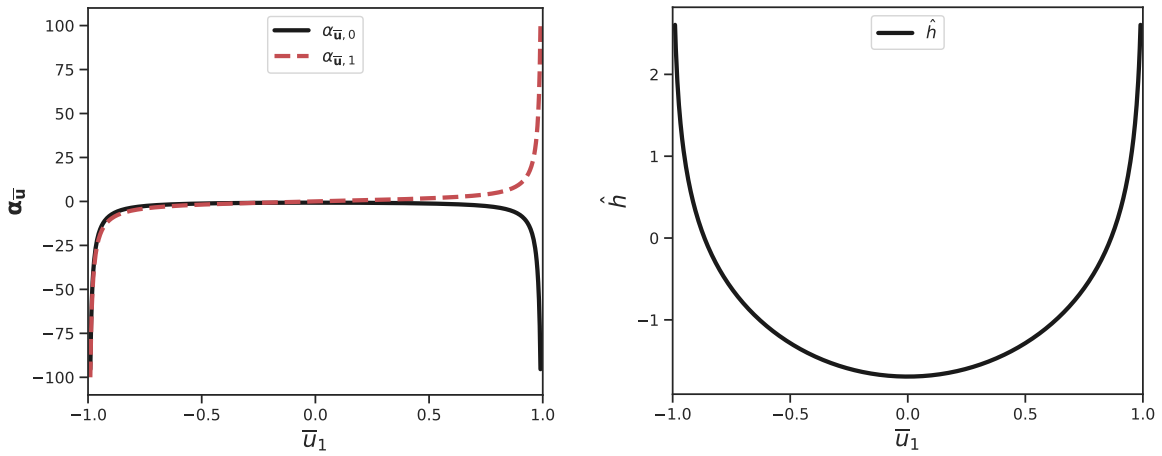
$$\overline{\mathbf{u}}_\# \mapsto \hat{h}(\overline{u}_1). \tag{4.59}$$

Since $\boldsymbol{\alpha}_{\overline{\mathbf{u}}} = \nabla_{\mathbf{u}} h(\overline{\mathbf{u}})$, the slope of h becomes steeper as $\overline{\mathbf{u}}_\# \to \partial\widetilde{\mathcal{R}}$.

Note that both, the input convex and monotonic neural network model, require the evaluation of $\vartheta((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#)$ for the reconstruction of $f^\theta_{\overline{\mathbf{u}}}$ and $\overline{\mathbf{u}}$. This induces high vulnerability for numerical overflows, for $\overline{\mathbf{u}}_\#$ near $\partial\widetilde{\mathcal{R}}$, especially when the networks are in the first iterations of the training process. No matter if we sample u and then compute α or vice versa, a sampling strategy must incorporate a meaningful distance measure $d(\overline{\mathbf{u}}_\#, \partial\widetilde{\mathcal{R}})$.

Direct boundary distance measures

Let us first consider proximity to $\partial\widetilde{\mathcal{R}}$ directly. Theorem 4.9 gives direct control over $\partial\widetilde{\mathcal{R}}$ since equality in one or more of the equations describes a boundary of the normalized realizable set. More general



(a) $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ over u_1 (b) $\hat{h}(u_1)$ over u_1

Figure 4.1.: Data to solution maps for the 1D M_1 closure.

results for arbitrarily high-order moments in one spatial dimension can be found in [129]. In this case, the distance measured to $\partial\widetilde{\mathcal{R}}$ is the norm distance, i.e.,

$$d(\overline{\mathbf{u}}_{\#}, \partial\widetilde{\mathcal{R}}) = \inf\left\{\left\|\overline{\mathbf{u}}_{\#} - \mathbf{w}\right\| : \mathbf{w} \in \partial\widetilde{\mathcal{R}}\right\}. \tag{4.60}$$

Note, that in Fig. 4.2(a-c), $\partial\widetilde{\mathcal{R}}$ is displayed by a dotted black line. The corresponding sampling strategy can be seen in Fig. 4.2(a) and (d).

In three spatial dimensions necessary and sufficient conditions have been constructed by [187] for up to order $N \leq 2$, but a full characterization of $\partial\widetilde{\mathcal{R}}$ remains an open problem [152]. Thus the combination of increasingly difficult direct characterization of $\partial\widetilde{\mathcal{R}}$ and the unavailability of such a characterization for $\mathbf{v} \in \mathbb{R}^d$, $d = 2, 3$, renders this approach impractical.

Indirect boundary distance estimates

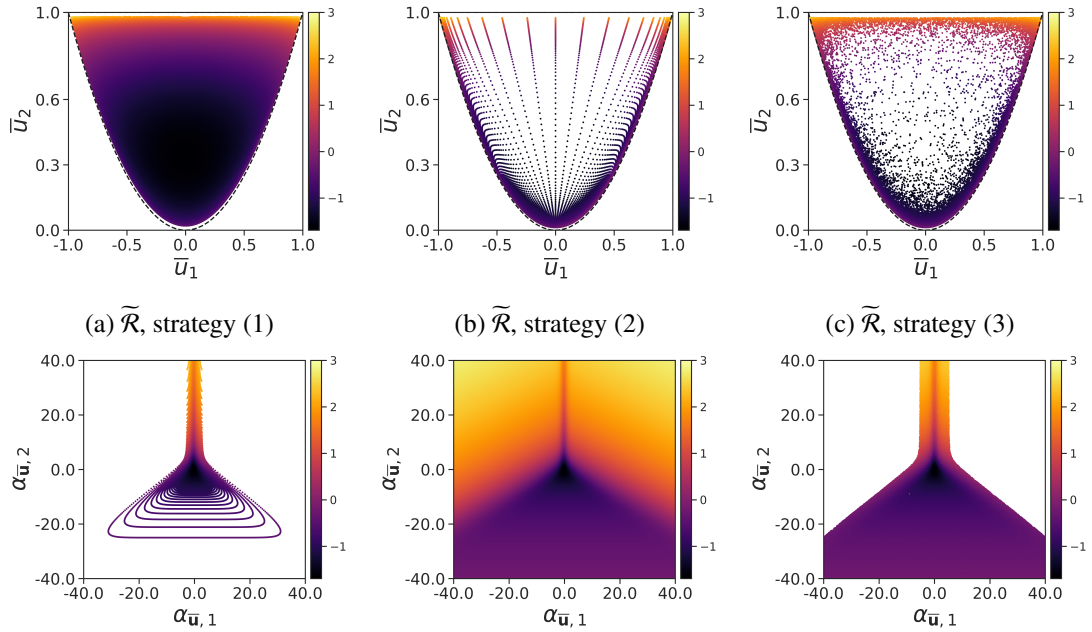
From a numerical point of view, it is interesting to construct a notion of distance to $\partial\widetilde{\mathcal{R}}$ in the space of Lagrange multipliers, since the magnitude of $\|\boldsymbol{\alpha}_{\overline{\mathbf{u}}}\|$ has implications on the numerical stability of the neural network training process.
A first idea consists of a norm bound of $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}$, i.e., $\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\| < M < \infty$, see [7, 200, 203].

Lemma 4.10 (Boundary Distance to $\widetilde{\mathcal{R}}$)
For $\boldsymbol{\alpha}_{\mathbf{u}} \neq 0$, we define the metric

$$d(\overline{\mathbf{u}}^1_{\#}, \overline{\mathbf{u}}^2_{\#}) = \left| \frac{1}{\left\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}^1})_{\#}\right\|} - \frac{1}{\left\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}^2})_{\#}\right\|} \right|. \tag{4.61}$$

It induces the boundary distance

$$d(\overline{\mathbf{u}}_{\#}, \widetilde{\mathcal{R}}) = \frac{1}{\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\|}, \tag{4.62}$$



(a) $\widetilde{\mathcal{R}}$, strategy (1) (b) $\widetilde{\mathcal{R}}$, strategy (2) (c) $\widetilde{\mathcal{R}}$, strategy (3)

(d) $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}$, strategy (1) (e) $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}$, strategy (2) (f) $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}$ strategy (3)

Figure 4.2.: Sampling strategies: (1) uniform grid sampling of $\widetilde{\mathcal{R}}$, (2) uniform grid sampling of $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}$ with norm bound, and (3) low-discrepancy sampling of $\boldsymbol{\beta}$ from $B_{M,\tau}$. The value of $h(\overline{\mathbf{u}}) = \phi(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}; \overline{\mathbf{u}})$ is color coded.

The proof is stated in §4.7. Thus we have established a boundary distance metric via the Lagrange multipliers, by enforcment of $\frac{1}{\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\|} > \frac{1}{M}$. Remark, that $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} = 0$ is the Lagrange multiplier of a non-boundary moment.

The approach is illustrated in Fig. 4.2(b) and (e), which shows a uniform grid in $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}$ with $\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\|_{\infty} < 40$, and the corresponding moments $\overline{\mathbf{u}}_{\#}$. When comparing Fig. 4.2(e) with Fig. 4.2(d) and Fig. 4.2(f), it is evident that a norm boundary distance leads to oversampling in the regions $\alpha_{\overline{\mathbf{u}},1} > 10$ and $\left\|\alpha_{\overline{\mathbf{u}},2}\right\| > 10$, since the corresponding moments $\overline{\mathbf{u}}_{\#}$ are too close to the boundary $\partial\widetilde{\mathcal{R}}$.

We introduce a second condition to the boundary distance, using the condition number of the Hessian of the (negative) dual objective functional H_n, i.e.,

$$H_n(\boldsymbol{\alpha}) = \left\langle \mathbf{m} \otimes \mathbf{m} \exp(\boldsymbol{\alpha} \cdot \mathbf{m}) \right\rangle, \tag{4.63}$$

as an estimate for the distance to $\partial\overline{R}$. Since it is symmetric and positive definite, the condition number k_2 is given by

$$k_2(H_n) = \frac{\lambda_{\max}}{\lambda_{\min}}, \tag{4.64}$$

where $\lambda_{\min}$, and $\lambda_{\max}$ are the biggest and smallest eigenvalues of H_n. We use $\lambda_{\min}$ to restric the sampling set of Lagrange multipliers $\boldsymbol{\alpha} = [\vartheta(\boldsymbol{\beta}), \boldsymbol{\beta}^{\top}]^{\top}$ as the bounded set

$$B_{M,\tau} = \left\{ \boldsymbol{\beta} \in \mathbb{R}^{n-1} : \|\boldsymbol{\beta}\| < M \cup \lambda_{\min}\left(H_n\left(\left[\vartheta(\boldsymbol{\beta}), \boldsymbol{\beta}^{\top}\right]^{\top}\right)\right) > \tau \right\}. \tag{4.65}$$

Figure 4.2(c) and (f) show $\overline{\mathbf{u}}_{\#}$ and $\beta \in B_{M=40,\tau=1\mathrm{e}-7}$, where the employed norm is the supremum norm. Note that there is no over-representation of the regions near $\overline{\mathbf{u}}_{\#} = [-1, 1]^{\top}$ and $\overline{\mathbf{u}}_{\#} = [1, 1]^{\top}$ and the set of sampled Lagrange multipliers, see Fig. 4.2(f), is similar to the Lagrange multipliers in Fig. 4.2(b).

4.4.3. Inference Error of Convex Neural Network Approximations

The assessment of the generalization error of arbitrary neural networks is a non-trivial task [62]. However, we can use the fact that the $\hat{h}$ is convex in $\overline{\mathbf{u}}$ and $\overline{\mathcal{R}}$ is a convex set. We consider the ICNN model, thus $\mathrm{N}_\theta(\overline{\mathbf{u}}_\#)$ is convex in $\overline{\mathbf{u}}_\#$ as well. For the reconstruction of the kinetic density

$$f^\theta_{\overline{\mathbf{u}}} = \exp\left(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}}} \cdot \mathbf{m}\right), \tag{4.66}$$

the Lagrange multiplier $\boldsymbol{\alpha}_{\mathbf{u}}$ is the quantity of interest, not necessarily the entropy functional itself. The idea is to consider the maximal interpolation error of the derivative of a trained, convex neural network on the convex hull of the training data set, i.e.,

$$\max_{\overline{\mathbf{u}}_\# \in C(X_T)} \left\| (\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\# - \nabla_{\overline{\mathbf{u}}_\#} \mathrm{N}_\theta(\overline{\mathbf{u}}_\#) \right\|_2, \tag{4.67}$$

for a given training data set X_T. The convex hull of the training data moments is denoted by $C(X_T)$.

Interpolation error bounds for input convex neural networks trained in Sobolev norm

We consider the interpolation error of the Lagrange multiplier of the minimal entropy closure using an ICNN, that approximates the entropy functional. Note that the results hold in general for any derivative approximation of a convex neural network that approximates a convex function defined on a convex set. For the sake of readability, we denote the truncated moment by

$$\mathbf{w} := \overline{\mathbf{u}}_\# \in \mathbb{R}^{n-1}, \tag{4.68}$$

which is the argument of the entropy functional $\hat{h}$. We assume, that the network is trained (at least) in Sobolev norm, i.e., the training loss reads

$$\mathcal{L}(X_T, N_{\theta^*}) = \frac{1}{|T|} \sum_{i \in T} \left(\left\| \hat{h}(\mathbf{w}_i) - \mathrm{N}_{\theta^*}(\mathbf{w}_i) \right\|_2^2 + \left\| \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) - \nabla_{\mathbf{w}} \mathrm{N}_{\theta^*}(\mathbf{w}_i) \right\|_2^2 \right) \tag{4.69}$$

when evaluated over the whole data set X_T. In the following, we assume that the network is trained, i.e., $\mathcal{L}(X_T, N_{\theta^*}) = 0$ with optimal training parameters θ^*. Thus we have

$$\hat{h}(\mathbf{w}_i) = \mathrm{N}_{\theta^*}(\mathbf{w}_i), \qquad \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) = \nabla_{\mathbf{w}} \mathrm{N}_{\theta^*}(\mathbf{w}_i) \qquad \forall i \in T. \tag{4.70}$$

Furthermore, let the domain $\widetilde{\mathcal{R}} \subset \mathbb{R}^{n-1}$ convex and bounded and the neural network be convex by design. We are interested in the maximal interpolation error of the derivative neural network with respect to its input variable $\mathbf{w}$. To this end, we consider the local maximal interpolation error of N_{θ^*} when using n training data points $X_n = \{\mathbf{w}_0, \dots, \mathbf{w}_{n-1}\}$, if the sampling space $X \subset \mathbb{R}^{n-1}$ has dimension $n-1$. Let $C(X_n)$ be the convex hull of X_n and $\mathbf{w}^* \in C(X_n)$, which we call the point of interest. We assume without loss of generality $\mathbf{w}^* = 0$. If this does not hold, one can consider the shifted setting $C^\dagger(X_n) = C(X_n) - \mathbf{w}^*$, $\hat{h}^\dagger = \hat{h}(\cdot + \mathbf{w}^*)$, $\mathbf{w}^\dagger = \mathbf{w} - \mathbf{w}^*$ instead. Using Lemma 4.3, we define the set $A_{\mathbf{w}^*}$ as

$$A_{\mathbf{w}^*} = \left\{ \mathbf{v} \in \mathbb{R}^{n-1} : \mathbf{v} \cdot \mathbf{w}_i \leq \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i, i = 0, \dots, n-1 \right\} \tag{4.71}$$

which is the dual polygon defined by the gradients at the sampling points and the point of interest and can be seen in Fig. 4.3. The set $A_{\mathbf{w}^*}$ contains all values which the gradient of a convex function with fixed gradients at the sampling points $\mathbf{w} \in X_n$ can attain at the point of interest $\mathbf{w}^*$, i.e., it contains all possible approximation values for the Lagrange multiplier $(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}}})_\#$.

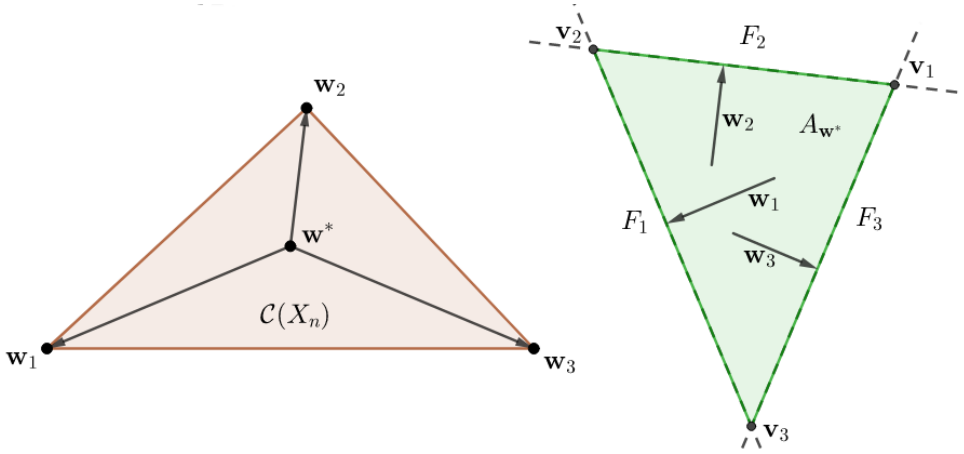


Figure 4.3.: The convex hull of the training points $C(X_n)$ (left) and the set of feasible gradients $A_{\mathbf{w}^*}$ (right) for $n = 3$. The normal vectors to the faces F_i are the vectors of the training points $\mathbf{w}_i$.

Theorem 4.11 (Feasible Gradients of a Convex Function)
Let $\hat{h}$ be convex, and $\mathbf{w}^* = 0$ the point of interest in the interior of $C(X_n)$. Then, $A_{\mathbf{w}^*}$ is a bounded polyhedron, whith n faces, defined by $F_i = \left\{\mathbf{v} \in \mathbb{R}^{n-1} : \mathbf{v} \cdot \mathbf{w}_i = \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i\right\}$ and vertices $\mathbf{v}_i = \bigcap_{j \neq i} F_j$.

We provide proof in §4.7. A direct outcome of Theorem 4.11 is the derivation of a local upper bound for the interpolation error of the gradient of an input convex network trained on a given training data set X_T, i.e.,

$$\left\|\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}^*) - \nabla_{\mathbf{w}} \mathrm{N}_{\theta^*}(\mathbf{w}^*)\right\| \leq \mathrm{diam}(A_{\mathbf{w}^*}), \tag{4.72}$$

where $A_{\mathbf{w}^*}$ is the polyhedron of feasible gradients w.r.t the point of interest $\mathbf{w}^*$ and the local training points X_n. A first consequence is that the $\mathrm{diam}(A_{\mathbf{w}^*})$ does not depend on the distance between the point of interest and any of the local training data points X_n since by definition of $A_{\mathbf{w}^*}$ in Eq. (4.71), one can divide by the norm of $\mathbf{w}_i - \mathbf{w}^*$ on both sides of the inequality for the boundary of $A_{\mathbf{w}^*}$. Thus in the following, we assume normalized $\mathbf{w}_i$.
The following theorem gives a more precise representation of $\mathrm{diam}(A_{\mathbf{w}^*})$.

Theorem 4.12 (Bound on the Feasible Gradients of a Convex Function)
Let $A_{\mathbf{w}^*}$ be defined by Eq. (4.71) *and $\mathbf{v}_i$ be defined as*

$$\mathbf{v}_i = \bigcap_{j \neq i} F_j. \tag{4.73}$$

Let $\mathbf{v}_i$ be the vertex opposing the face F_i. The matrix

$$X_i = [\mathbf{w}_0^n, \ldots, \mathbf{w}_{i-1}^n, \mathbf{w}_{i+1}^n, \ldots, \mathbf{w}_{n-1}^n]^T \tag{4.74}$$

contain the vectors of normalized sampling points relative to the point of interest $\mathbf{w}_*$,

$$\mathbf{w}_i^n = \mathbf{w}_i / \left\|\mathbf{w}_i\right\|_2 . \tag{4.75}$$

Furthermore, let

$$\mathbf{b}_i = \left[\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_0) \cdot \mathbf{w}_0^n, \ldots, \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_{i-1}) \cdot \mathbf{w}_{i-1}^n, \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_{i+1}) \cdot \mathbf{w}_{i+1}^n, \ldots, \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_{n-1}) \cdot \mathbf{w}_{n-1}^n\right]^T \tag{4.76}$$

be a vector. Under the assumptions of Theorem 4.11, the vertex $\mathbf{v}_i$ *is uniquely defined by*

$$X_i \mathbf{v}_i = \mathbf{b}_i. \tag{4.77}$$

Additionally, we can estimate the distance between two vertices $\mathbf{v}_i$ *and* $\mathbf{v}_j$ *by*

$$\left\|\mathbf{v}_i - \mathbf{v}_j\right\|_2 \leq \left(\left\|X_i^{-1}\right\| + \left\|X_j^{-1}\right\|\right) C_{\mathbf{w}^*}, \tag{4.78}$$

where $C_{\mathbf{w}^*} = \max_{k,l} \left\|\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_k) - \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_l)\right\|_2$ *and* $\left\|X_i^{-1}\right\|$ *denotes the corresponding operator norm of* X_i^{-1}.

Let us draw some conclusions from Theorem 4.12. First, we have as a direct consequence

$$\left\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} - (\boldsymbol{\alpha}^{\theta}_{\overline{\mathbf{u}}})_{\#}\right\| = \left\|\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}^*) - \nabla_{\mathbf{w}} \mathrm{N}_{\theta^*}(\mathbf{w}^*)\right\| \leq \operatorname{diam}(A_{\mathbf{w}^*}) \leq \left(\left\|X_i^{-1}\right\| + \left\|X_j^{-1}\right\|\right) C_{\mathbf{w}^*}. \tag{4.79}$$

First, $\operatorname{diam}(A_{\mathbf{w}^*}) \to \infty$, if $\operatorname{dist}(\mathbf{w}^*, \partial C(X_n)) \to 0$, since the normals of at least two neighboring boundaries of $A_{\mathbf{w}^*}$ become (anti)parallel to each other.
Additionally, we find, that for a fixed point of interest and angles between the local training points, the size of $\operatorname{diam}(A_{\mathbf{w}^*})$ depends only on the norm distance of $\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i)$, $i = 0, \ldots, n-1$, which is encoded in the definition of $C_{\mathbf{w}^*}$. The smaller the norm distance of the gradients of the sample points, the smaller gets $C_{\mathbf{w}^*}$.
Lastly, we consider the case of $\mathbf{w}^*$ on the boundary of the convex hull of the local training points. Then one selects a new set of local training points, such that $\mathbf{w}^*$ is in the interior of their convex hull. Remark, that the polyhedron can be shrunk by including more training points to the set X_n.

4.4.4. Sampling Strategy for the Minimal Entropy Closure

As a conclusion of the analysis of the realizable set $\overline{\mathcal{R}}$ and the maximal interpolation error of a convex neural network, we can consider the task at hand. Let $X_{T,S}$ be a training data set of size T for the minimal entropy closure sampled with strategy S. The task is to find a sampling strategy, that minimizes the maximal interpolation error bound of the network gradient, i.e.,

$$\min_{X_{T,S}} \max_{\overline{\mathbf{u}}_{\#} \in C(X_{T,S})} \left\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} - \nabla_{\overline{\mathbf{u}}_{\#}} \mathrm{N}_{\theta^*}(\overline{\mathbf{u}}_{\#})\right\|_2. \tag{4.80}$$

Consequently, we generate the training data X_T by uniform rejection sampling of $\boldsymbol{\beta}$ from $B_{M,\tau}$, given by Eq. (4.65). Uniform distribution of $\boldsymbol{\beta}$ is important since it is equivalent to a uniform norm distance between the gradients of N_{θ^*}. This minimizes the right-hand side of Eq. (4.79) and thus is a reasonable strategy for the problem (4.80).

The random number generator has a non-negligible influence on the quality of training data, as Fig. 4.2(c) and (e) display. In the former figure, $\overline{\mathbf{u}}_{\#}$ are generated by a uniform grid sampling of $\boldsymbol{\beta} \in B_{M,\tau}$, and the latter by uniform sampling of $\boldsymbol{\beta} \in B_{M,\tau}$ using a low-discrepancy sampling method. The deformed grid in Fig. 4.2(c) contains very steep triangles of local training points X_n at the upper boundary of $\widetilde{\mathcal{R}}$. Consequently, a point of interest is always close to the boundary of $C(X_n)$, which implies a big diameter for the polyhedron of admissible gradients $A_{\mathbf{w}^*}$. We see in Fig. 4.2(c), that low-discrepancy sampling mitigates this issue. Generally, low-discrepancy sampling methods have a positive impact on neural network training, especially for high data dimensions [172, 185]. The data-sampling strategy is summarized in Algorithm 4.7.

Algorithm 4.7: Training data generator

Input: M: Lagrange multiplier norm boundary
τ: Eigenvalue tolerance
N: Order of the moment closure

Result: X_T: Training data-set $\left\{\left(\hat{h}_i, \overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}\right)\right\}_{i \in T}$

for $i = 0$ *to* $i = T$ **do**
  do
    $\boldsymbol{\beta} \sim \texttt{uniform}\left(\left\{\boldsymbol{\beta} \in \mathbb{R}^{n-1} : \|\boldsymbol{\beta}\| < M\right\}\right)$ /* Sampling uniformly from $B_{M,\tau}$ */
    $\boldsymbol{\alpha}_{\overline{\mathbf{u}},i} \leftarrow \left[\vartheta\left(\boldsymbol{\beta}\right), \boldsymbol{\beta}^\top\right]^\top$ /* Compute Lagrange multiplier */
  while $\lambda_{min} < \tau$
  $\overline{\mathbf{u}}_i \leftarrow \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}},i} \cdot \mathbf{m}\right)\right\rangle$ /* Reconstruct normalized moment vector */
  $\hat{h}_i \leftarrow \boldsymbol{\alpha}_{\overline{\mathbf{u}},i} \cdot \overline{\mathbf{u}}_i - \left\langle \exp\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}},i} \cdot \mathbf{m}\right)\right\rangle$ /* Compute entropy functional */
  Append $\left(\hat{h}_i, \overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}\right)$ to X_T.

4.5. Numerical Results

In this section, we present numerical results and investigate the performance of the neural entropy closure. First, we validate the training performance of the ICNN and IMNN models and conduct synthetic tests to measure the performance of the networks on $\widetilde{\mathcal{R}}$ and the computational efficiency in comparison with a baseline Newton optimizer. Then, we employ the network in a 1D and 2D kinetic solver and compare the results with the benchmark solution in several simulation test cases. To ensure the significance of the errors compared to the spatial discretization errors, we perform a convergence analysis of the neural network-based and benchmark solver.

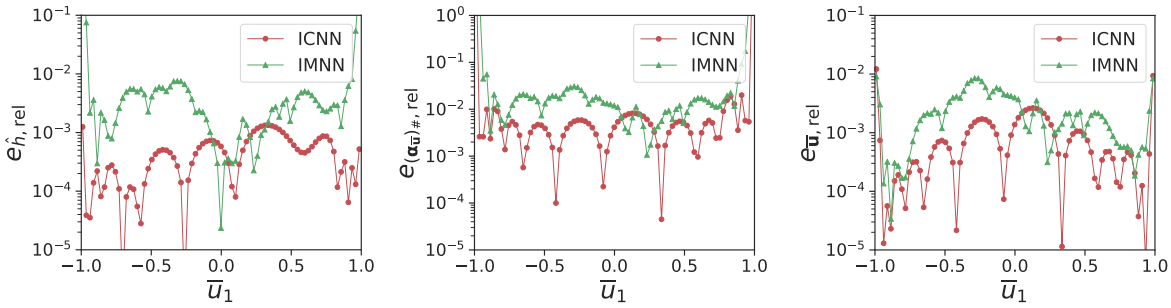
4.5.1. Neural Network Training

In the following, we evaluate the training performance of the neural network architectures, which are implemented in Tensorflow [1] and can be found in the GitHub repositories KiT-RT[3] and NeuralEntropyClosures[4].
The neural networks are trained on data sampled from $B_{M,\tau}$. The sampled data is partitioned into training and test data set, where the test data set consists of 10% of the total data. Table 4.1 compares the test error of the ICNN and IMNN models for different closures. For the sake of comparability, the output of the IMNN model, which approximates $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ is scaled to the same range as the derivative of the ICNN model. The networks are trained on an Nvidia RTX 3090 GPU in single-precision floating-point accuracy. For each network architecture, we present the mean squared and mean absolute error for all quantities of interest averaged over the test data set. For the IMNN model, the monotonicity loss is additionally displayed. We see that the MSE test errors of $\overline{\mathbf{u}}$ range from 1e−4 to 1e−6 for the ICNN and IMNN model. Remark, that the ICNN models require significantly fewer parameters to train to similar accuracy as the IMNN models due to the architectural bias towards convexity.
For the stability of the kinetic solver, the error in the reconstructed flux $\langle \mathbf{v}\mathbf{m} f_{\mathbf{u}} \rangle$, which is Lipschitz continuous in $\mathbf{u}$, is the most important quantity. The results are in line with the findings of similar approaches,

[3] https://github.com/CSMMLab/KiT-RT
[4] https://github.com/ScSteffen/neuralEntropyClosures

Table 4.1.: Approximation error of the neural network-based entropy closures against the validation data in different metrics. We display the mean results over 10 experiments for each model and closure each with less than 10% standard deviation.

		M_1 1D		M_2 1D		M_1 2D	
		ICNN	IMNN	ICNN	IMNN	ICNN	IMNN
Layout		10×7	30×2	15×7	50×2	18×8	100×3
Params		9.27e2	2.47e3	2.07e3	6.80e3	3.27e3	3.64e4
MSE	$\hat{h}$	7.87e−7	2.09e−5	1.33e−5	5.04e−4	1.10e−6	4.01e−4
	$(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#$	7.52e−4	5.56e−6	2.81e−4	2.56e−4	3.39e−5	4.54e−5
	$\overline{\mathbf{u}}$	1.47e−6	3.64e−6	2.81e−4	1.27e−4	3.39e−5	8.09e−5
$\mathcal{L}_{\text{mono}}$		n.a.	1.60e−14	n.a.	1.38e−14	n .a.	5.31e−16
MAE	$\hat{h}$	7.57e−4	3.05e−3	3.11e−3	1.66e−2	1.02e−3	1.49e−2
	$(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#$	1.26e−2	9.10e−3	1.23e−2	9.74e−3	3.36e−3	4.35e−3
	$\overline{\mathbf{u}}$	9.59e−4	1.51e−3	1.23e−2	7.96e−3	9.62e−4	7.02e−3



(a) $e_{\hat{h},\text{rel}}$ (b) $e_{(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#,\text{rel}}$ (c) $e_{\overline{\mathbf{u}}}$

Figure 4.4.: Relative l_2 test errors of the input convex and monotone network architectures for the 1D M_1 closure. Minimal distance to $\partial\widetilde{\mathcal{R}}$ is 0.01. ICNN outperforms IMNN in every metric. In the extrapolation regime, the approximation error increases significantly, since the bounds of Eq. (4.79) are not applicable here.

see [200]. Notice, that the mean absolute error in the prediction of $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ is significantly higher than the error in h or u^r for all input convex neural networks. The reason for this is the high range of values, that $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ can attain.

4.5.2. Synthetic Test Cases

In this section, we consider again the 1D M_1 entropy closure, see Fig. 4.1, and perform accuracy tests for the input convex and monotonic neural network architecture. The networks are trained on a data set generated from sampling $\alpha_{\overline{\mathbf{u}},1} \in [-50, 50]$ uniformly. Then, the networks are evaluated on twice as many samples in the displayed data range $\overline{u}_1 \in [-0.99, 0.99]$ which corresponds to $\alpha_{\overline{\mathbf{u}},1} \in [-95, 95]$. Consequently, we measure the interpolation error for $\alpha_{\overline{\mathbf{u}},1} \in [-50, 50]$ and the extrapolation error of the networks for $\alpha_{\overline{\mathbf{u}},1} \in [-95, 50) \cup (50, 95]$. The relative norm errors of the predictions of both network

Table 4.2.: Computational setup for the numerical test cases

Test Case	CFL	t_f	Δ_t	Δ_x	n_q
Anisotropic Inflow M_1 1D	0.4	0.7	8.00e−5	2.00e−4	28
Anisotropic Inflow M_2 1D	0.4	0.7	8.00e−5	2.00e−4	28
Adiabatic M_1 2D	0.4	10	3.00e−4	1.50e−3	400
Checkerboard	0.45	10	3.15e−2	7.0e−2	400

architectures, i.e.,

$$e_{\hat{h},\text{rel}} = \frac{\left\|\hat{h} - \hat{h}^{\theta}\right\|_2}{\left\|\hat{h}\right\|_2}, \tag{4.81}$$

can be seen in Fig. 4.4 and analogously for $e_{\overline{\mathbf{u}}}$ in and $e_{(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#,\text{rel}}$. Within the convex hull of the training data, the ICNN model obeys the error bound $e_{\hat{h},\text{rel}} < 5\text{e}{-3}$. The relative error increases by half an order of magnitude in the outside the convex hull of the training data, where the interpolation error bound does not hold. The relative error of the IMNN model displays more fluctuation with a mean of 1e−2. In the extrapolation area, the error of the IMNN model is by an order of magnitude larger than that of the ICNN model. Similar results can be seen for $e_{(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#,\text{rel}}$ and $e_{\overline{\mathbf{u}}}$. Remark, that $e_{\overline{\mathbf{u}}} < 1\text{e}{-2}$ even in the extrapolation areas. This shows, that the nature of the reconstruction map of Eq. (4.26) mitigates the approximation inaccuracies of $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#$ to some degree.

4.5.3. Anisotropic Inflow Test Case

Let us first study particle transport in an isotropically scattering and homogeneous medium. We consider the one-dimensional geometry, where the linear Boltzmann equation reduces to

$$\partial_t f + \mathbf{v}\partial_{\mathbf{x}} f = Q(f) = \sigma_s \langle f \rangle - \sigma_a f(\mathbf{v}), \tag{4.82}$$

where σ_s is a scattering coefficient and σ_a is an absorption coefficient. The corresponding moment model becomes

$$\partial_t \mathbf{u} + \partial_{\mathbf{x}} \langle v\mathbf{m} f_{\mathbf{u}} \rangle = \sigma_s u_0 - \sigma_a \mathbf{u} \tag{4.83}$$

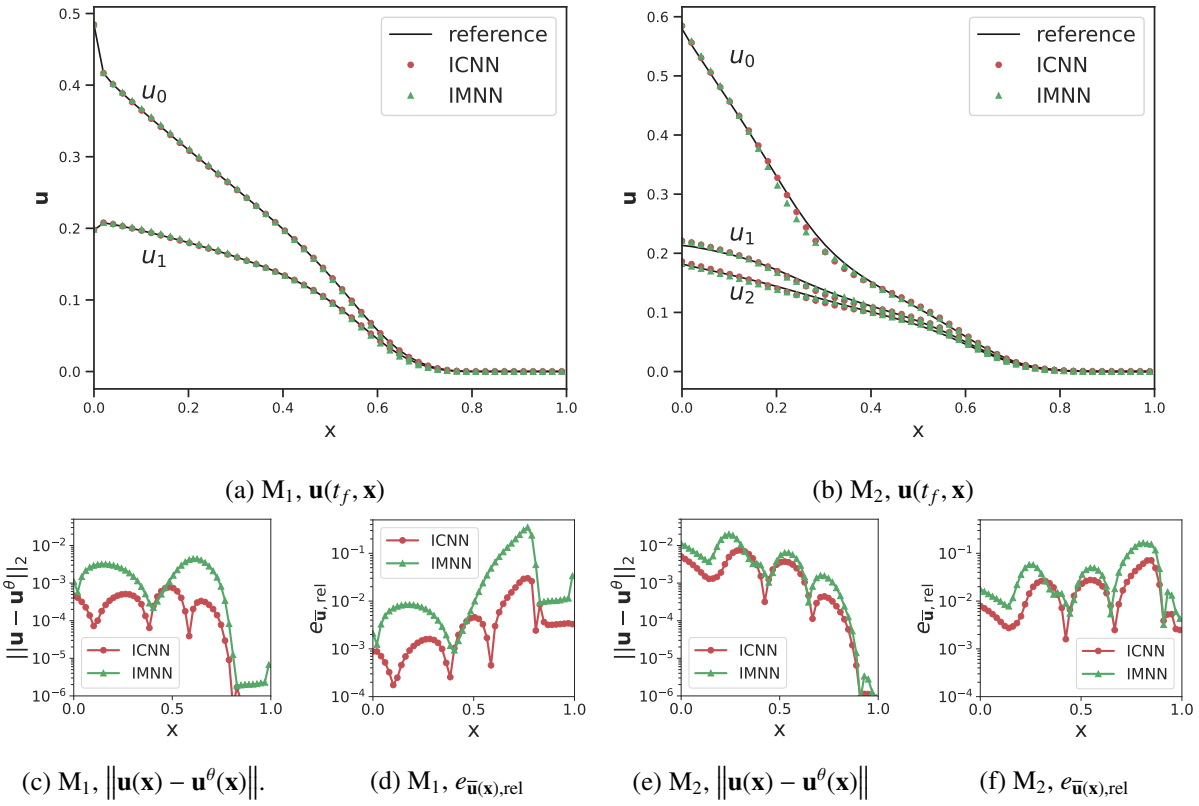
$$f_{\mathbf{u}} = \exp(\boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{m}), \tag{4.84}$$

with $\sigma_s = 1.0$ and $\sigma_a = 0.0$, domain $X = [0, 1]$ and $V = P_{\mathbb{R}}(\mathbb{S}^2)$. Initial and boundary conditions are given by

$$f(t, x, v) = \epsilon, \qquad t = 0 \tag{4.85}$$

$$f(t, x, v) = \begin{cases} 0.5 & \text{if } v > 0 \\ 0 & \text{if } v \leq 0, \end{cases}, \qquad x = 0 \tag{4.86}$$

with an anisotropic inflow condition imposed at the left boundary and a far-field condition on the right boundary. The initial condition of the computational domain is set as a vacuum state with a numerical safety $1 \gg \epsilon > 0$ since the normalized vector $\overline{\mathbf{u}}$ is defined for $u_0 > 0$. The test case is challenging for neural network surrogate models, since kinetic densities, that model vacuum states, and densities of anisotropic phenomena correspond to moments near $\partial\overline{\mathcal{R}}$. Thus the test case is well suited to test the



(a) M_1, $\mathbf{u}(t_f, \mathbf{x})$ (b) M_2, $\mathbf{u}(t_f, \mathbf{x})$

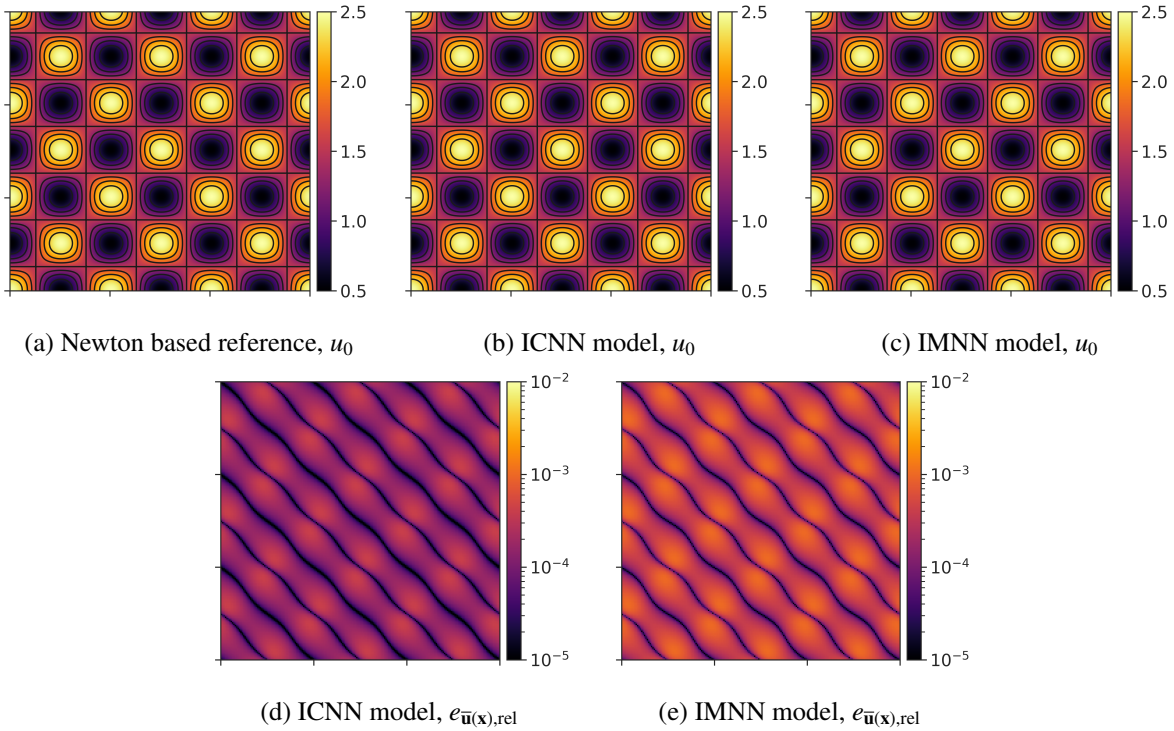
(c) M_1, $\left\|\mathbf{u}(\mathbf{x}) - \mathbf{u}^\theta(\mathbf{x})\right\|$. (d) M_1, $e_{\overline{\mathbf{u}}(\mathbf{x}),\mathrm{rel}}$ (e) M_2, $\left\|\mathbf{u}(\mathbf{x}) - \mathbf{u}^\theta(\mathbf{x})\right\|$ (f) M_2, $e_{\overline{\mathbf{u}}(\mathbf{x}),\mathrm{rel}}$

Figure 4.5.: Solution of the anisotropic inflow test case at t_f. The ICNN closure outperforms IMNN in all test cases. At the wavefront, the models exhibit the highest relative errors.

capabilities of the surrogate models in extreme conditions. The numerical PDE solver for the moment system is a first-order kinetic scheme specified in §3.

We compare the simulation performance of the ICNN and IMNN model with the Newton solver baseline set to single precision floating point accuracy. The CFL for all simulations is set smaller than 0.5 to maintain the realizability of the updated solution [194]. The detailed computational setup can be found in Table 4.2. The solution profiles at final time $t_f = 0.7$ are presented in Fig. 4.5 for the M_1 and M_2 system and we see that the system's dynamics are well captured by both neural network architectures.

Figure 4.5 depicts the deviations of the l_2 norm from the Newton baseline for the ICNN and IMNN models in the M_1 and M_2 moment systems, calculated at final time t_f. For the M_1 system, see Fig. 4.5(c), the point-wise norm error of the ICNN model is below of 1e−3.5 and below 1e−2.5 for the IMNN model. In the M_2 test case, see Fig. 4.5(e), the errors do not exceed 1e−2. An inspection of the relative errors in Fig. 4.5(d) and Fig. 4.5(f) confirms that in both the M_1 and M_2 system, the maximal relative error is exhibited at the wave-front of the anisotropic inflow at $x \in (0.7, 0.8)$ and $t = t_f$. In this area, the moments $\mathbf{u}$ are closest to $\partial\overline{\mathcal{R}}$. This confirms the performance degradation of the neural network models at near boundary moments.



(a) Newton based reference, u_0 (b) ICNN model, u_0 (c) IMNN model, u_0

(d) ICNN model, $e_{\overline{\mathbf{u}}(\mathbf{x}),\text{rel}}$ (e) IMNN model, $e_{\overline{\mathbf{u}}(\mathbf{x}),\text{rel}}$

Figure 4.6.: Solution of the periodic 2D M_1 closure at $t = t_f$ (top row) and $e_{\overline{\mathbf{u}}(\mathbf{x}),\text{rel}}$ (bottom row). The ICNN closure outperforms the IMNN closure, but both models capture the dynamics of the test case satisfactorily.

4.5.4. Adiabatic Test Case

We consider a rectangular domain in two spatial dimensions. The phase space of the Boltzmann equation is thus five dimensional, with

$$t > 0, \qquad \mathbf{X} = [-1.5, 1.5]^2, \quad \text{and} \quad \mathbf{V} = P_{\mathbb{R}^2}(\mathbb{S}^2) = \left\{\mathbf{v} \in \mathbb{R}^2 : \|\mathbf{v}\|_2 \leq 1\right\}. \tag{4.87}$$

We consider the M_1 closure with a monomial basis $\mathbf{m}(\mathbf{v}) = [1, v_{x_1}, v_{x_2}]^T$. The velocity domain $\mathbf{V}$ is parametrized in spherical coordinates and integrated with a tensorized Gauss-Legendre quadrature. This test case considers a non-scattering and non-absorbing medium, i.e., $\sigma_s = \sigma_a = 0$, and the Boltzmann equation reduces to a transport equation of the form

$$\partial_t f + \mathbf{v}\partial_{\mathbf{x}} f = 0. \tag{4.88}$$

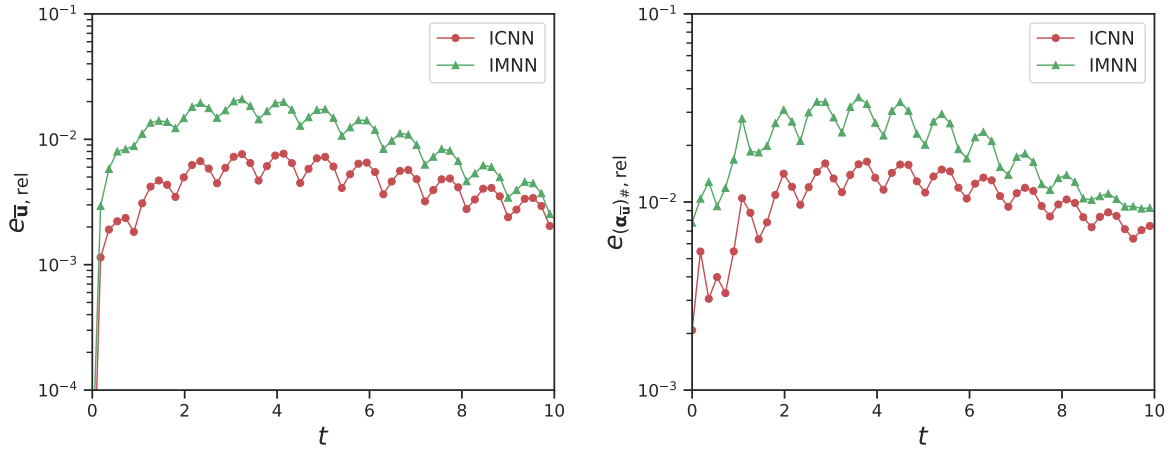
with the corresponding moment system

$$\begin{aligned} \partial_t \mathbf{u} + \partial_{\mathbf{x}} \left\langle \mathbf{v} \otimes \mathbf{m}^\top f_{\mathbf{u}} \right\rangle &= 0 \\ f_{\mathbf{u}} &= \exp(\boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{m}). \end{aligned} \tag{4.89}$$

Periodic initial conditions that translate to the M_1 moment system are given by

$$u_0 = 1.5 + \cos(2\pi x_1)\cos(2\pi x_2), \quad u_1 = 0.3u_0, \quad u_2 = 0.3u_0 \qquad \forall \mathbf{x} \in \mathbf{X}. \tag{4.90}$$

Periodic boundary conditions are imposed on the equations to get a well-posed system of equations. Note that due to the absence of gain and loss terms and the choice of boundary conditions, the system



(a) $e_{\overline{\mathbf{u}}(t),\text{rel}}$ (b) $e_{(\alpha_{\overline{\mathbf{u}}})_\#(t),\text{rel}}$

Figure 4.7.: The relative errors $e_{\overline{\mathbf{u}}(t),\text{rel}}$ and $e_{(\alpha_{\overline{\mathbf{u}}})_\#(t),\text{rel}}$ are stable and diminish over time. The ICNN model outperforms the IMNN model accuracy by half an order of magnitude.

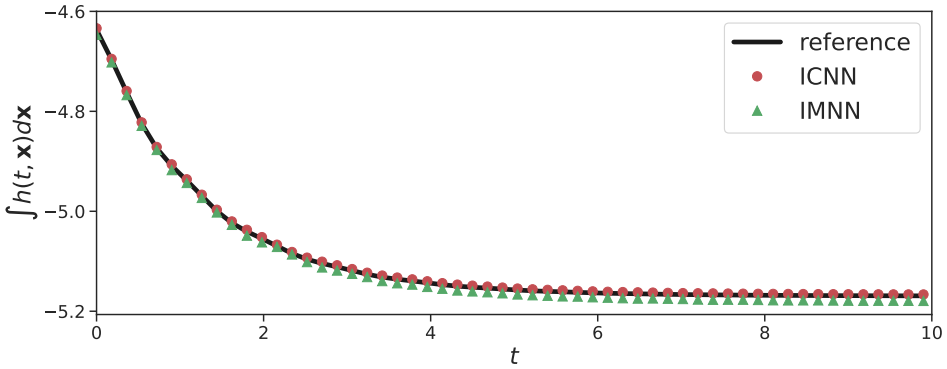
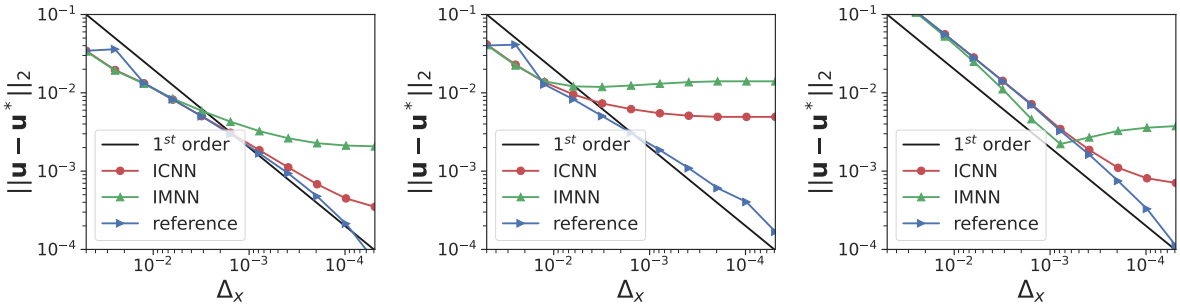


Figure 4.8.: Total entropy of the test case per time-step. Both surrogate models dissipate the system's entropy at the same rate as the reference solution.

is adiabatic. The numerical PDE solver is again a first-order kinetic scheme, see §3. The detailed solver configuration can be found in Table 4.2.

We present the Newton-based reference solution and the ICNN and IMNN surrogate models with their corresponding relative l_2 deviation from the Newton solution in Fig. 4.6 at $t = t_f$. The relative errors of both neural networks exhibit periodic behavior and are well below 1e−3 for the ICNN model and below 5e−3 for the IMNN model.
Figure 4.7 presents the relative l_2 error in $\mathbf{u}$ and $\boldsymbol{\alpha}_\mathbf{u}$ for both the ICNN and IMNN models, calculated at various time steps during the simulation. The errors are calculated as the average relative l_2 error across the entire spatial domain. Observe that the ICNN is again more accurate than the IMNN surrogate model, and both models are stable over a long time horizon $t_f = 10s$, discretized with 3000 time steps. The oscillations in the error curves stem from the periodic nature of the system's solution, in which the distance to $\partial\overline{\mathcal{R}}$ of the appearing moments changes periodically as well. Remark that at all times during the simulation, the total mass of the simulation, i.e., the integral of u_0 over the spatial domain, is constant, which validates the conservation property of the surrogate models.



(a) 1D M_1, inflow. (b) 1D M_2, inflow. (c) 2D M_1, periodic.

Figure 4.9.: Convergence analysis of the Newton reference, ICNN, and IMNN models with respect to grid refinement. The errors of the IMNN architectures plateau earlier than the ICNN models in all test cases.

Lastly, the total entropy of the system computed with the Newton closure and the ICNN and IMNN models is analyzed at each time step in Fig. 4.8. Due to the periodic boundary conditions and $\sigma = \tau = 0$, the system is adiabatic, allowing for the analysis of the entropy dissipation of the Boltzmann equation. The upwind scheme was selected for the numerical flux of the moment system, which is known to be an entropy-dissipating scheme. All methods demonstrate entropy dissipation, however, the ICNN model exhibits a smaller deviation from the reference entropy compared to the IMNN model. In conclusion, the neural network-based hybrid solver retains the structural properties of the reference system and computes the numerical solution with acceptable accuracy.

4.5.5. Convergence Analysis of Neural Network-Based Entropy Closures

A convergence analysis for the M1 and M2 systems is presented in Figure 4.9. This analysis compares the numerical solutions of the moment system, as computed with the Newton baseline, the ICNN, and the IMNN closure, on increasingly finer computational grids. The l_2 deviation of the current numerical solution is displayed, relative to a reference solution given by a Newton closure with double precision, on the finest spatial grid. The results indicate that the Newton-based solver demonstrates first-order convergence, as expected.

In Figure 4.9(a), it is seen that the IMNN model converges with first-order accuracy up to a cell size of $\Delta x = 1\text{e}{-3}$. Beyond this point, the approximation error of the neural network dominates the spatial resolution error. On the other hand, the ICNN model converges up to $\Delta x = 1\text{e}{-4}$, which confirms its higher accuracy as measured in the synthetic test cases. Similar patterns are observed in Figure 4.9(b) and Figure 4.9(c).

4.5.6. Computational Efficiency - Synthetic Tests

We compare the computational efficiency of the ICNN model and the Newton baseline optimizer in a synthetic test case of the normalized M_2 closure in 1D. In contrast to the neural network, the computational performance of the Newton solver is affected by the proximity of the moments $\overline{\mathbf{u}}_\#$ to the boundary $\partial\widetilde{\mathcal{R}}$. Therefore, we consider three test cases. First, moments are sampled uniformly in $\widetilde{\mathcal{R}}$. Second, near the center of $\widetilde{\mathcal{R}}$, that is, in the ball $\overline{\mathbf{u}}_\# \in B[0, 0.5]^T, r$ with radius $r = 0.1$. Third, near $\partial\widetilde{\mathcal{R}}$. In the second

Table 4.3.: Wall time comparison for parallel computation of the normalized 1D M_2 closure

		Newton		ICNN on CPU		ICNN on GPU	
data set size	strategy	mean [s]	std. dev. [s]	mean [s]	std.dev. [s]	mean [s]	std. dev. [s]
1e+3	uniform	6.48e−3	1.17e−3	7.88e−3	5.10e−4	9.88e−3	4.76e−3
	near $\partial\overline{\mathcal{R}}$	3.83e+2	7.90e−2	8.02e−3	6.40e−4	9.74e−3	4.75e−3
	interior	5.14e−3	1.21e−3	8.75e−3	8.75e−4	9.57e−3	4.86e−3
1e+7	uniform	5.01e+0	1.23e−2	6.33e−1	8.91e−3	3.90e−2	3.82e−3
	near $\partial\overline{\mathcal{R}}$	2.71e+5	1.33e+3	6.32e−1	8.53e−3	3.88e−2	3.52e−3
	interior	4.24e+0	3.86e−2	0.63e−1	8.67e−3	3.84e−2	3.57e−3

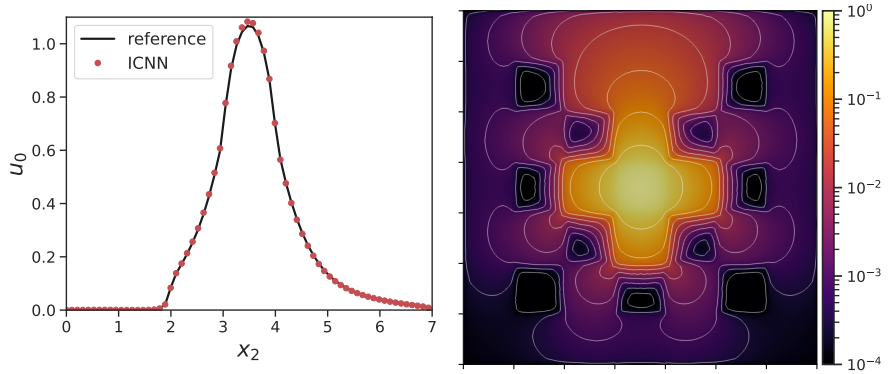
case, the closure is easy to solve, while it is difficult to solve in the last case for the Newton optimizer. The Newton solver is implemented in the KiT-RT framework.

As in a parallelized kinetic solver, we measure the parallel performance of Newton and ICNN. We test CPU and GPU implementations and different-sized data sets. To ensure comparability, the accuracy tolerance of the Newton solver is set to single-precision floating-point accuracy, to match the test error of the trained networks. The used CPU is an AMD Ryzen9 3900x with 32GB memory and 24 threads, and the GPU is an RTX3090 with 20GB memory. The mean and standard deviation of 100 measurements for each case are presented in Table 4.3.

We demonstrate that the network efficiency is not affected by the condition of the closure, while the Newton solver is three orders of magnitude slower near $\partial\widetilde{\mathcal{R}}$ compared to an interior moment $\overline{\mathbf{u}} \in B_{[0,0.5]^T,r}$. The largest computational cost for the Newton optimizer arises from computing and inverting Hessians using a 30-point Gauss-Legendre quadrature. A large number of quadrature points is necessary as the integrand $\mathbf{m} \otimes \mathbf{m} \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m})$ is highly nonlinear.
It should be noted that the neural network surrogate efficiency increases with larger sample sizes, especially for GPU evaluations, as the TensorFlow implementations are optimized to benefit greatly from data-level parallelism, as long as all data fit into the GPU memory. This suggests that the best application scenario for neural network-based surrogates is large-scale simulations.

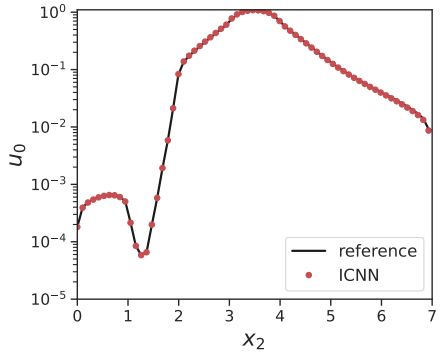
4.5.7. Computational Efficiency - Checkerboard Test Case

We revisit the checkerboard test case of §3.7.2 and compare the Newton and ICNN-based M_1 simulation with a monomial basis. In this test case, we compile the trained 2D M_1 ICNN model into the C++ KiT-RT solver for an efficient surrogate model implementation using the Tensorflow C++ back-end [1]. As depicted in Fig. 4.10, both the Newton-based and neural network-based solutions are in good agreement, with a small deviation only observed at the top of the domain ($x_2 \approx 6.5$) in the cross-section plot.
We investigate the timing profile of the Newton and ICNN-based M_1 solver on a computational grid with 1e6 cells computed with 4 CPU cores, see Table 4.4. It is evident that 90% of the iteration time is spent on computing the minimal entropy closure. The ICNN surrogate model is three orders of magnitude faster and thus able to accelerate the M_1 simulation by a factor of 10.48, i.e., to achieve a 90.45% reduction in simulation time for a simulation with 4 CPU cores.
Remark that the Newton-based entropy closure wall-times have an 8.06% standard deviation, which is caused by vastly different conditions of the minimal entropy closure optimization problem in different

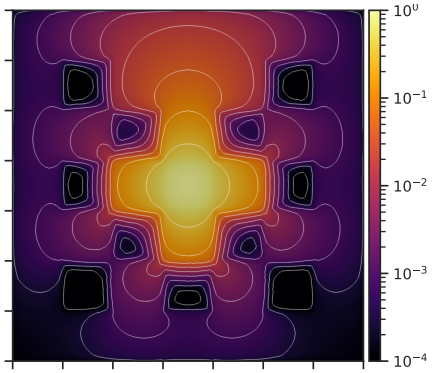


(a) Linear scale

(b) M_1, u_0



(c) Log scale



(d) ICNN-based M_1, u_0

Figure 4.10.: Newton and ICNN-based M_1 simulations at $t = t_f$ with the KiT-RT solver. Vertical cross-sections at $x_1 = 3.5$ (a),(c), and simulation results (b),(d). The ICNN approximation error is below grid accuracy, and flow fields and cross-sections are hardly distinguishable. The ICNN solution reports slightly larger values at the radiation source $\boldsymbol{x} = [3.5, 3.5]^\top$.

cells of the computational grid. Indeed, this is a challenging problem for distributed memory parallelization, where the computational domain $\mathbf{X}$ is partitioned on the processors. If in one partition more moments emerge near $\partial\mathcal{R}$, parallel scaling capabilities are impaired. Neural network-based closures timings have much less impact on the parallel load balancing.

4.6. Chapter Conclusion

In this chapter, we addressed the minimal entropy closure moment system of the Boltzmann equation and the challenges of classical numerical approaches.

4.6.1. Summary

We introduced two novel neural network-based approaches to close the moment hierarchy of the Boltzmann equation, once with an input convex neural network (ICNN) that approximates the entropy of the minimal entropy closure, and once with a monotonic neural network (IMNN) that approximates the Lagrange multipliers of the minimal entropy optimization problem.
We have analyzed the data structure of the entropy closure for normalized moments and investigated the

Table 4.4.: Performance profiling and comparison of the M_1 1^{st} order solver of the KiT-RT package on 10^6 grid cells, with order 20 Gauss-Legendre quadrature. Displayed are the mean timings of 100 iterations. Newton and ICNN-based entropy closures are compared, where ICNN shows a significant speedup.

	Newton		ICNN	
Function	time [ms]	% time	time [ms]	% time
Entropy closure	197.54 ± 17.56	90.65 ± 8.06	0.34 ± 0.03	1.81 ± 0.15
Flux Computation	20.02 ± 0.89	9.18 ± 0.40	20.09 ± 0.84	96.58 ± 3.3
Time Integration	0.33 ± 0.01	0.15 ± 0.01	0.33 ± 0.04	1.61 ± 0.21
Total time [ms]	217.90		20.76	
Time reduction by ICNN [%]		90.45		
Speedup by ICNN		10.48		

maximal interpolation error of convex neural network approximations to derive a comprehensive data-sampling strategy that minimizes the neural network interpolation error.
The presented neural network surrogate models are tested in a wide array of synthetic and simulation test cases examining numerical accuracy, long-time simulation behavior, and computational efficiency. In conclusion, the ICNN surrogate model is best suited as a neural network-based entropy approximation. Both presented surrogate models are far more computationally efficient than the baseline. The suggested data-sampling method allows for good training and test performance of the networks, where the ICNN model surpasses the IMNN model due to its inherent convex structure which obeys the derived maximum interpolation error bound.

4.6.2. Limitations of the Approach

Near the boundary of the realizable set, the entropy closure presents an ill-conditioned optimization problem, resulting in long simulation times of the Newton solver. This challenge unfortunately transfers to the neural network surrogates, in two ways. First, numerical instabilities of the network training prohibit sampling moments too close to the boundary of the realizable set. Second, as a result, the test performance of the networks declines as one infers on moments near the boundary. The problem exaggerates for higher order closures, where the condition of the Hessian of the dual problem increases even faster as one moves closer to the boundary. This yields a challenge for building surrogate models for high-order moment system closures.

We consider the treatment of the region near the boundary of the realizable set, where the neural networks exhibit the highest errors, in §5.

4.7. Additional Material

In the following, we provide proof for the stated theorems and lemmas of this chapter.

Statements of §4.3

We provide proof for Lemma 4.1.

Proof: (Lemma 4.1) Consider ϕ at the optimal point $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}; \overline{\mathbf{u}})$ and assume $m(\mathbf{v})_0 = 1$. By the first order necessary condition, we have

$$\nabla_{\alpha}\phi(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}; \overline{\mathbf{u}}) = \overline{\mathbf{u}} - \langle \mathbf{m} \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m}) \rangle = 0 \tag{4.91}$$

Inspecting the first element of $\overline{\mathbf{u}}$ gives

$$1 = \overline{u}_0 = \langle \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m}) \rangle . \tag{4.92}$$

On the other hand, using

$$\vartheta\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\right) = -\log\left(\langle \exp\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} \cdot \mathbf{m}_{\#}\right)\rangle\right), \tag{4.93}$$

we see that

$$\begin{aligned}\left\langle \exp\left(\left[\vartheta\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\right), (\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}^{\top}\right]^{\top} \cdot \mathbf{m}\right)\right\rangle &= \exp\left(\vartheta\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\right)\right) \left\langle \exp\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} \cdot \mathbf{m}_{\#}\right)\right\rangle \\ &= \frac{1}{\left\langle \exp\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} \cdot \mathbf{m}_{\#}\right)\right\rangle} \left\langle \exp\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} \cdot \mathbf{m}_{\#}\right)\right\rangle = 1 = u_0,\end{aligned} \tag{4.94}$$

and the assertion holds. □

We provide proof for Lemma 4.2.

Proof: (Lemma 4.2) The proof is structured in two steps.
1. Consider again the definition of the normalized moment, i.e,

$$\overline{\mathbf{u}} = \langle \mathbf{m} \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m}) \rangle = \left\langle \mathbf{m} \exp((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} \cdot \mathbf{m}_{\#}) \exp(\alpha_{\overline{\mathbf{u}},0}) \right\rangle \tag{4.95}$$

and multiply both sides with $u_0 > 0$, such that

$$\mathbf{u} = \left\langle \exp((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} \cdot \mathbf{m}_{\#}) \exp(\alpha_{\overline{\mathbf{u}},0}) u_0 \right\rangle = \left\langle \exp((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#} \cdot \mathbf{m}_{\#}) \exp(\alpha_{\overline{\mathbf{u}},0} + \log(u_0)) \right\rangle . \tag{4.96}$$

Using Lemma 4.1 yields the assertion.
2. Consider the functional ϕ at the optimal point $(\boldsymbol{\alpha}_{\mathbf{u}}; \mathbf{u})$. We use the statement of the first part of the proof and get

$$\begin{aligned} h(\mathbf{u}) = \phi(\boldsymbol{\alpha}_{\mathbf{u}}; \mathbf{u}) &= \boldsymbol{\alpha}_{\mathbf{u}} \cdot u - \langle \exp(\boldsymbol{\alpha}_{\mathbf{u}} \cdot \mathbf{m}) \rangle \\ &= u_0 \left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \overline{\mathbf{u}} + \log(u_0)\right) - \langle \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m} + \ln(u_0)) \rangle \\ &= u_0 \left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \overline{\mathbf{u}} + \ln(u_0)\right) - \langle \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m}) \rangle \\ &= u_0 \left(h(\overline{\mathbf{u}}) + \ln(u_0)\right), \end{aligned} \tag{4.97}$$

which yields the assertion, where we used Eq. (4.5) and (4.19). □

We provide proof for Theorem 4.5.

Proof: (Theorem 4.5) The proof is structured in two steps.
1. We note that h^θ is convex in $\mathbf{u}$, if $\hat{h}^\theta$ is convex in $\overline{\mathbf{u}}$ [200]. By Lemma 4.2 we see directly, that

$$\boldsymbol{\alpha}_{\mathbf{u}}^\theta = \nabla_{\mathbf{u}} h^\theta(\mathbf{u}). \tag{4.98}$$

We inspect the Hessian $H^\theta(\alpha_{\mathbf{u}}^\theta)$, which we define via its inverse

$$H^{\theta,-1}(\alpha_{\mathbf{u}}^\theta) = \nabla_{\mathbf{u}}^2 h^\theta(\mathbf{u}) = \nabla_{\mathbf{u}} \boldsymbol{\alpha}_{\mathbf{u}}. \tag{4.99}$$

Thus, $H^{\theta,-1}$ is symmetric positive definite due to the convexity of h^θ.
Since the neural network surrogate uses the same ansatz (4.6) for the reconstruction of $f_{\mathbf{u}}^\theta$ as the minimal entropy closure, we can choose $j_{i,*}(\alpha_{\mathbf{u}}^\theta)$ as in the proof of Theorem 1.13 and hyperbolicity y of the neural network based system follows immediately.
2. For entropy dissipation, consider the proof of Theorem 1.15. Due to the reconstruction ansatz (4.6), we can use the Legendre duality, and by Eq. (4.98), we can verify the entropy/entropy-flux pair and the entropy dissipation law in analogy to the proof of Theorem 1.15. □

We show Theorem 4.7.

Proof: (Theorem 4.7) We show that each layer $[\mathbf{z}_{k-1}^\top, \mathbf{x}^\top]^\top \mapsto \mathbf{z}_k$ is convex. For $k = 1$, we directly see, that

$$W_1^{\mathbf{x}} \mathbf{x} + \mathbf{b}_k \tag{4.100}$$

is linear and thus convex. We apply Lemma 4.6 iteratively and obtain the assertion. □

We show Lemma 4.8

Proof: (Lemma 4.8) The map $\mathbf{x} \mapsto W_1^{\mathbf{x}} \mathbf{z}_1^{**} + \mathbf{b}_1$ is linear, thus the assertion of the proof of Theorem 4.7 holds. □

Statements of §4.4

We provide proof for Theorem 4.11.

Proof: (Theorem 4.11) The proof is structured in two parts. First, we show that the vertices $\boldsymbol{\nu}_i \in \mathbb{R}^{n-1}$ are well defined, if $\mathbf{w}^*$ is element of the interior of $C(X_n)$. Second, we show that all $\boldsymbol{\nu}_i \in A_{\mathbf{w}^*}$. Thus any convex combination of $\boldsymbol{\nu}_i$ is in $A_{\mathbf{w}^*}$ and therefore, $A_{\mathbf{w}^*}$ is defined by a (bounded) polyhedron with vertices $\boldsymbol{\nu}_i$.
1. We show that $\boldsymbol{\nu}_i$ are well defined. First, if the point of interest is an element of the interior of $C(X_n)$, then all $\mathbf{w}_i \in X_n$ are linearly independent. The boundary of the set of feasible gradients with respect to the sampling point $\mathbf{w}_i$ and the point of $\mathbf{w}^*$ interest consists of the hyperplane given by

$$F_i = \left\{ \boldsymbol{\nu} \in \mathbb{R}^{n-1} : \boldsymbol{\nu} \cdot \mathbf{w}_i = \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i \right\}. \tag{4.101}$$

If all $\mathbf{w}_i \neq 0$ are linearly independent, no hyperplanes are parallel or lie in each other. The proper intersection of d hyperplanes in $\mathbb{R}^{n-1}$ yields a single point, which is

$$\boldsymbol{v}_i = \bigcap_{j \neq i} F_j. \tag{4.102}$$

which we define as vertex $\boldsymbol{v}_i \in \mathbb{R}^{n-1}$, that touches all hyperplanes except F_i.
2. We show that all $\boldsymbol{v}_i \in A_{\mathbf{w}^*}$. This means, that we have to show

$$\boldsymbol{v}_j \cdot \mathbf{w}_i \leq \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i, \qquad \forall i, j = 0, \ldots, n-1. \tag{4.103}$$

By the definition of $\boldsymbol{v}_j$, we have

$$\boldsymbol{v}_j \in F_i, \qquad j \neq i, \tag{4.104}$$

so we are only concerned with

$$\boldsymbol{v}_i \cdot \mathbf{w}_i \leq \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i. \tag{4.105}$$

We start by stating an auxiliary statement. Let $\boldsymbol{p}_i^j = \boldsymbol{v}_j - \boldsymbol{v}_i$ for $i \neq j$. If X_n is linearly independent and $\mathbf{w}^* = 0$ is in the interior of $C(X_n)$, then

$$\operatorname{sign}(\boldsymbol{p}_i^j \cdot \mathbf{w}_i) = \operatorname{sign}(\boldsymbol{p}_k^l \cdot \mathbf{w}_k), \qquad \forall i \neq j, k \neq l. \tag{4.106}$$

Linear independence of $\mathbf{w}_i \in X_n$ and $\mathbf{w}^* = 0$ being in the interior of $C(X_n)$ translates to

$$0 = \sum_{i=0}^{N} a_i \mathbf{w}_i, \qquad a_i > 0. \tag{4.107}$$

We have

$$\begin{aligned}
\boldsymbol{p}_i^j \cdot \mathbf{w}_i &= \frac{-1}{a_i} \sum_{m \neq i} a_m \left(\boldsymbol{v}_j - \boldsymbol{v}_i\right) \cdot \mathbf{w}_m \\
&= \frac{-1}{a_i} \left(\sum_{m \neq i,j} a_m \left(\boldsymbol{v}_j - \boldsymbol{v}_i\right) \cdot \mathbf{w}_m + a_j \left(\boldsymbol{v}_j - \boldsymbol{v}_i\right) \cdot \mathbf{w}_j \right) \\
&= \frac{-1}{a_i} \left(\sum_{m \neq i,j} a_m \left(\boldsymbol{v}_j \cdot \mathbf{w}_m - \boldsymbol{v}_i \cdot \mathbf{w}_m\right) + a_j \left(\boldsymbol{v}_j - \boldsymbol{v}_i\right) \cdot \mathbf{w}_j \right) \\
&= \frac{-1}{a_i} \left(\sum_{m \neq i,j} a_m \left(\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_m) \cdot \mathbf{w}_m - \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_m) \cdot \mathbf{w}_m\right) + a_j (\boldsymbol{v}_j - \boldsymbol{v}_i) \cdot \mathbf{w}_j \right) \\
&= \frac{-1}{a_i} a_j (\boldsymbol{v}_j - \boldsymbol{v}_i) \cdot \mathbf{w}_j = \frac{a_j}{a_i} (\boldsymbol{v}_i - \boldsymbol{v}_j) \cdot \mathbf{w}_j = \frac{a_j}{a_i} \boldsymbol{p}_j^i \cdot \mathbf{w}_j,
\end{aligned} \tag{4.108}$$

where we use the definition of the face F_m. Since $\frac{a_j}{a_i}$ is positive $\operatorname{sign}(\boldsymbol{p}_i^j \cdot \mathbf{w}_i) = \operatorname{sign}(\boldsymbol{p}_j^i \cdot \mathbf{w}_j)$ follows for all $i \neq j$. Assume $\boldsymbol{p}_i^j \cdot \mathbf{w}_i > 0$ and $\boldsymbol{p}_i^h \cdot \mathbf{w}_i < 0$. Then

$$\boldsymbol{v}_j \cdot \mathbf{w}_i > \boldsymbol{v}_i \cdot \mathbf{w}_i > \boldsymbol{v}_h \cdot \mathbf{w}_i. \tag{4.109}$$

Thus, we have

$$0 < (\boldsymbol{v}_j - \boldsymbol{v}_h) \cdot \mathbf{w}_i = \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i - \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i = 0, \tag{4.110}$$

which is a contradiction to the monotonicity of the gradient. Then we obtain

$$\operatorname{sign}(\boldsymbol{p}_i^j \cdot \mathbf{w}_i) = \operatorname{sign}(\boldsymbol{p}_i^k \cdot \mathbf{w}_i) = \operatorname{sign}(\boldsymbol{p}_k^i \cdot \mathbf{w}_k) = \operatorname{sign}(\boldsymbol{p}_k^l \cdot \mathbf{w}_l), \qquad \forall i \neq j, k \neq l. \tag{4.111}$$

This means that all face normals $\mathbf{w}_i$ are either facing outward of the polyhedron defined by the vertices $\{\boldsymbol{v}_i\}$ or all facing inward. Assuming inward-facing normals, then for each face of the polyhedron created by $A_{\mathbf{w}^*}$, the feasible set is the half-space outside the current face of the polyhedron. Due to convexity of the polyhedron defined by $\{\boldsymbol{v}_i\}$, this would imply that $A_{\mathbf{w}^*} = \emptyset$, which contradicts continuity of the gradient of $\hat{h}$. Therefor we have outward-facing normals. Finally, we have

$$0 < (\boldsymbol{v}_j - \boldsymbol{v}_i) \cdot \mathbf{w}_i = \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i - \boldsymbol{v}_i \cdot \mathbf{w}_i, \tag{4.112}$$

and thus $\boldsymbol{v}_i \cdot \mathbf{w}_i < \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) \cdot \mathbf{w}_i$, i.e., $\boldsymbol{v}_i \in A_{\mathbf{w}^*}$ for all i. we have that $A_{\mathbf{w}^*}$ is indeed a polygon defined by the vertices $\boldsymbol{v}_i$. By convexity, the polyhedron $A_{\mathbf{w}^*}$ contains all feasible gradients of the point of interest.

□

We provide proof for Theorem 4.12.

Proof: (Theorem 4.12) By definition of $\boldsymbol{v}_i = \bigcap_{j \neq i} F_j$ and the fact that we can divide Eq. (4.71) by $\|\mathbf{w}_i\|$ we get the linear systems. Let for $\boldsymbol{\xi} \in \mathbb{R}^d$ hold that

$$C_{\boldsymbol{\xi}} = \max_{k=0,\dots,d} \|\nabla_{\mathbf{w}} f(\mathbf{w}_k) - \boldsymbol{\xi}\|_2 \,. \tag{4.113}$$

Then, we have

$$\left|\mathbf{w}_j \cdot (\boldsymbol{v}_i - \boldsymbol{\xi})\right| = \left|\mathbf{w}_i \cdot \left(\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_i) - \boldsymbol{\xi}\right)\right| \leq \|\mathbf{w}_i\|_2 \, C_{\boldsymbol{\xi}} = C_{\boldsymbol{\xi}} \quad \forall i = 0, \dots, d \tag{4.114}$$

since $\mathbf{w}_i$ has unit norm. Thus, each entry of the vector $X_i \boldsymbol{v}_i$ has an absolute value smaller than $C_{\boldsymbol{\xi}}$. We interpret X_i as a linear operator mapping $(\mathbb{R}^d, \|\cdot\|_2) \to (\mathbb{R}^d, \|\cdot\|_\infty)$.
$X_i = [\mathbf{w}_0, \dots, \mathbf{w}_{i-1}, \mathbf{w}_{i+1}, \dots, \mathbf{w}_d]^T$ is invertible, if $\mathbf{w}^*$ is in the interior of $C(X_n)$ and defines a mapping $(\mathbb{R}^d, \|\cdot\|_\infty) \to (\mathbb{R}^d, \|\cdot\|_2)$. Consequently, we can estimate

$$\|X_i(\boldsymbol{v}_i - \boldsymbol{\xi})\|_\infty \leq C_{\boldsymbol{\xi}}, \tag{4.115}$$

and thus

$$\|\boldsymbol{v}_i - \boldsymbol{\xi}\|_2 \leq \left\|X_i^{-1}\right\| C_{\boldsymbol{\xi}}. \tag{4.116}$$

Finally, we get

$$\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2 \leq \|\boldsymbol{v}_i - \boldsymbol{\xi}\|_2 + \left\|\boldsymbol{\xi} - \boldsymbol{v}_j\right\|_2 \leq \left(\left\|X_1^{-1}\right\| + \left\|X_2^{-1}\right\|\right) C_{\boldsymbol{\xi}}. \tag{4.117}$$

We can choose $\boldsymbol{\xi} = \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_l)$ such that

$$\max_{k=0,\dots,d} \left\|\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_k) - \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_l)\right\|_2 = \max_{k,l=0,\dots,d} \left\|\nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_k) - \nabla_{\mathbf{w}} \hat{h}(\mathbf{w}_l)\right\|_2 =: C_{\mathbf{w}^*}. \tag{4.118}$$

□

Statements of §4.4

We state the proof of Lemma 4.10.

Proof: (Lemma 4.10) The map (4.62) is a metric, which is verifiable by the fact that $\mathbf{u} \mapsto \boldsymbol{\alpha}_{\mathbf{u}}$ is a bijection..
Consider a sequence of moments $\overline{\mathbf{u}}_{\#,n} \to \mathbf{w}$ for $\mathbf{w} \in \partial\widetilde{\mathcal{R}}$ and remark that $\left\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}_n})_{\#}\right\| \to \infty$. We see, that for $\overline{\mathbf{u}} \in \widetilde{\mathcal{R}}^{\mathrm{o}}$, i.e., its interior, the metric (4.62), i.e.,

$$\lim_{n\to\infty} d(\overline{\mathbf{u}}_{\#}, \overline{\mathbf{u}}_{\#,n}) = \lim_{n\to\infty} \left| \frac{1}{\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\|} - \frac{1}{\left\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}_n})_{\#}\right\|} \right| = \frac{1}{\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\|} \tag{4.119}$$

is well defined. Then we have

$$\begin{aligned} d(\overline{\mathbf{u}}_{\#}, \widetilde{\mathcal{R}}) = & \\ \lim_{n\to\infty} d(\overline{\mathbf{u}}_{\#}, \overline{\mathbf{u}}_{\#,n}) &= \lim_{n\to\infty} \inf \left\{ \left| \frac{1}{\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\|} - \frac{1}{\left\|(\boldsymbol{\alpha}_{[1,\mathbf{w}_n^\top]})_{\#}\right\|} \right| : \mathbf{w}_n \to \mathbf{w} \in \partial\widetilde{\mathcal{R}} \right\} \\ &= \inf \left\{ \lim_{n\to\infty} \left| \frac{1}{\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\|} - \frac{1}{\left\|(\boldsymbol{\alpha}_{[1,\mathbf{w}_n^\top]})_{\#}\right\|} \right| : \mathbf{w}_n \to \mathbf{w} \in \partial\widetilde{\mathcal{R}} \right\} \\ &= \frac{1}{\|(\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_{\#}\|}, \end{aligned} \tag{4.120}$$

which yields the assertion. □

CHAPTER 5

Regularized, Neural Network-Based, Structure-Preserving Minimal Entropy Closures

In this chapter, we extend structure-preserving, neural network-based approximations of the closure of the moment system to the context of regularized entropy closures. The main idea is to interpret structure-preserving neural network approximations of the regularized minimal entropy closure as an two stage approximation to the original entropy closure. We conduct numerical analysis of this approximation and investigate optimal parameter choices. Our numerical experiments demonstrate that the method has a much lower memory footprint than traditional methods with competitive computation times and simulation accuracy.

5.1. Introduction

We consider again the structurally rich, but computationally expensive minimal entropy closure for the linear Boltzmann moment system and address the open challenges and limitations of the work done in §4. There, we successfully proposed an input convex surrogate model to approximate the entropy of the Boltzmann moment system, in order to find a computationally cheap closure. Although successfully applied to low order entropy closures, these neural network-based closures face severe challenges when approximating high order closures in higher spatial dimensions, due to the ill condition of the underlying optimization problem in an-isotropic particle regimes, i.e., near the boundary of its feasible set [208].

5.1.1. Related Work on Moment Closures

The most relevant previous work to this chapter is done by [208, 200], who proposed data-driven approximations of the minimal entropy functional. The authors of [200] use convex splines for the M_1 closure in one dimension and show that this approach preserves the mathematical structure of the entropy closure. Furthermore, classical, fully-connected neural networks have been employed for the M_1 and M_2 closure in one spatial dimension, where penalty terms ensure convexity in an empirical sense. In [208],i.e.,

§4, the authors extend the approach to two spatial dimensions for the M_1 closure, where they employ input-convex neural networks [9], to ensure a structure-preserving, neural network-based approximation of the entropy closure. Furthermore, an error estimate and a sampling strategy for the approximation with input convex neural networks is proposed. However, both works are limited to low-order moment closures, since the approximation quality of both neural networks and splines decreases heavily near the boundary of the set of feasible moments, especially for high order closures. The regularized minimal entropy closure has been introduced by [7], which tackles the issues of ill condition of the optimization problem near the boundary of the realizable set.

5.1.2. Novelty and Scientific Contribution

The scope of this work is the fusion of the regularized formulation of the minimal entropy closure [7] and structure-preserving neural network-based closure approximations [208]. We investigate the challenges of the non-regularized entropy closure in terms of neural network approximation and data-sampling. Then we construct a normalized, regularized framework to create input convex, structure-preserving neural network approximations to the regularized entropy closure, which is in itself a convex approximation to the non-regularized entropy closure. We conduct an error analysis of this multi-step approximation including neural network approximation error, regularization error and scaling errors of the partially regularized closure to ensure a controlled deployment as a surrogate model in an high performance kinetic solver using the open source codebase KiT-RT [147]. We introduce dimension reduction techniques exploiting Galilean invariance of the Boltzmann moment system to further reduce the data-space of the neural network-based entropy for high order closures. Lastly we conduct extensive numerical experiments comparing different neural network approximations for different, higher order entropy closures of different regularization levels in synthetic tests as well as well known simulation benchmarks of the field of radiation transport.
The neural networks are trained using the Tensorflow-based open source Github repository NeuralEntropyClosures[1] and the trained network graphs are integrated in the open source high performance solver KiT-RT[2].

This chapter is a collaboration with Cory Hauck, Paul Laiu and Martin Frank and under review in the Journal of Computational Physics. The chapter focuses on the authors contributions.

5.1.3. The Chapter in Context of the Dissertation

This chapter directly continues the work of §4 and tackles the open challenges and limitations of the proposed surrogate models, i.e., limited accuracy near the boundary of the realizable set, and high dimensional closures. We make use of the data-sampler of §4.

5.1.4. Chapter Outline

We discuss challenges and limitations of the current neural network-based entropy closure in §5.2, and introduce the regularized entropy closure in §5.3 alongside normalization and reduction of the problem. The neural network-based approximation of the entropy closure is presented in §5.4 and preservation of the structure of the moment system including Galilean invariance is shown. In §5.5, we analyze the

[1] https://github.com/ScSteffen/neuralEntropyClosures
[2] https://github.com/CSMMLab/KiT-RT

errors of the partial regularization, reconstruction ansatz, and neural network approximation and their implications on data sampling. Numerical results are presented in §5.6.

5.2. Challenges of the Entropy-Based Moment Closure

We consider again the moment system of the linear Boltzmann equation with a spherical velocity domain $\mathbf{V} = \mathbb{S}^2$, i.e.,

$$\partial_t \mathbf{u}(t, \mathbf{x}) + \nabla_{\mathbf{x}} \cdot \langle \mathbf{v} \otimes \mathbf{m}(\mathbf{v}) f \rangle = \langle \mathbf{m}(\mathbf{v}) Q(f) \rangle \,, \tag{5.1}$$

with minimal entropy closure with dual formulation, i.e.,

$$\boldsymbol{\alpha}_{\mathbf{u}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmax}} \, \phi(\boldsymbol{\alpha}; \mathbf{u}) \tag{5.2}$$

$$\phi(\boldsymbol{\alpha}; \mathbf{u}) = \boldsymbol{\alpha} \cdot \mathbf{u} - \left\langle \exp(\boldsymbol{\alpha} \cdot \mathbf{m}) \right\rangle \tag{5.3}$$

for the Maxwell Boltzmann entropy, see Table 1.1 and the corresponding realizable set

$$\mathcal{R} = \{ \mathbf{u} : \langle \mathbf{m} g \rangle = \mathbf{u}, \; g \in F_{\mathbf{m}} \} \,. \tag{5.4}$$

For details we refer to §1.

Although the minimal entropy closure preserves many structural properties of the underlying equation and avoids shortcomings such as potentially negative solutions of the P_N method [82] and ray effects of the S_N method [31], it faces practical challenges. First, solving the convex optimization problem of Eq. (1.58) at every time step in each grid cell of a numerical simulation comes at a high computational cost, especially for higher spatial dimensions and high order closures [141]. This issue motivated the use of neural network-based entropy closures, whose computational effort are several orders of magnitude lower than that of an iterative optimizer if employed at scale [208]. The use of a neural network-based closure even for an M_1 2D simulation yields a speedup of up to 10.48 compared to the reference simulation that uses a numerical optimizer for the closure, see §4.5.7. Second, the entropy closure yields numerical challenges near the boundary of the realizable set. To give insights into the behavior of the entropy functional near the boundary of the realizable set, we introduce the normalized realizable set $\overline{\mathcal{R}}$ and the reduced, normalized realizable set $\widetilde{\mathcal{R}}$:

$$\overline{\mathcal{R}} = \{ \mathbf{u} \in \mathcal{R} : u_0 = 1 \} \subset \mathbb{R}^n \,, \tag{5.5}$$

$$\widetilde{\mathcal{R}} = \left\{ \mathbf{w} \in \mathbb{R}^{n-1} : [1, \mathbf{w}^\top]^\top \in \overline{\mathcal{R}} \right\} \subset \mathbb{R}^{n-1} \,. \tag{5.6}$$

For $D = [0, \infty)$ and $\mathbf{V} = \mathbb{S}^2$, i.e., the unit sphere, both $\overline{\mathcal{R}}$ and $\widetilde{\mathcal{R}}$ are bounded. We further introduce the normalization operator

$$\overline{(\cdot)} : \mathbb{R}^n \to \mathbb{R}^n \;\text{ defined such that }\; \mathbf{u} \mapsto \overline{\mathbf{u}} := \tfrac{\mathbf{u}}{u_0} \tag{5.7}$$

and the reduction operator

$$(\cdot)_{\#} : \mathbb{R}^n \to \mathbb{R}^{n-1} \;\text{ defined such that }\; \mathbf{u} \mapsto \mathbf{u}_{\#} := [u_1, \ldots, u_n]^\top \in \mathbb{R}^{n-1} \,. \tag{5.8}$$

Therefore, $\mathbf{u} \in \mathcal{R}$ if and only if $\overline{\mathbf{u}} \in \overline{\mathcal{R}}$ and $\overline{\mathbf{u}}_{\#} \in \widetilde{\mathcal{R}}$.

It is known [7, 208, 82, 8] that when $\overline{\mathbf{u}}$ approaches the boundary $\partial\overline{\mathcal{R}}$, the associated multiplier $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ becomes unbounded. In a reduced, one-dimensional slab geometry, the corresponding kinetic density $f_{\overline{\mathbf{u}}}$ of a realizable boundary moment for $D = [0, \infty)$ is a sum of delta functions [52, 187], i.e.

$$f_{\mathbf{u}}(\mathbf{v}) = \sum_i c_i \delta(p_i(\mathbf{v})), \tag{5.9}$$

where δ is a Dirac distribution and p_i are rational functions depending on the moments $\mathbf{u}_i$. We describe the limit behavior of the entropy functional for moments approaching the boundary of the reduced realizable set $\partial\widetilde{\mathcal{R}}$.

Theorem 5.1 (Entropy Divergence at $\partial\overline{\mathcal{R}}$)
When the Maxwell-Boltzmann entropy is considered, the entropy function h diverges to infinity as $\overline{\mathbf{u}} \to \partial\overline{\mathcal{R}}$.

We provide the proof in §5.8.

Unfortunately, the divergent behavior of h and the poor conditioning of its Hessian H given in (1.65) near the realizable boundary also affects the neural network approximation. In previous work [208], the entropy function h is approximated for normalized moments $\overline{\mathbf{u}}$ by an input convex neural network N_θ with parameters θ, i.e.,

$$\mathrm{N}_\theta(\overline{\mathbf{u}}) \approx \mathrm{h}(\overline{\mathbf{u}}) \qquad \text{and} \qquad \nabla_{\mathbf{u}} \mathrm{N}_\theta(\overline{\mathbf{u}}) \approx \boldsymbol{\alpha}_{\overline{\mathbf{u}}}. \tag{5.10}$$

Due to numerical overflow, it is infeasible to sample the entropy functional h or Lagrange multipliers $\boldsymbol{\alpha}_{\overline{\mathbf{u}}}$ nearby or at the boundary $\partial\overline{\mathcal{R}}$ as the training data, especially when training on GPUs in single precision. Thus, for moments near or at $\partial\overline{\mathcal{R}}$, there is no control over the neural network approximation error, although an error bound in the interior of $\overline{\mathcal{R}}$ has been established [208]. Moreover, it has been shown that $\overline{\mathcal{R}}$ may not be closed under the action of the numerical solver for the moment equation [8]. In such cases, realizable moments near $\partial\overline{\mathcal{R}}$ may not stay realizable after updating in time by the numerical solver. Although a neural network-based approximation to the entropy functional is technically defined for $\mathbf{u} \notin \mathcal{R}$, the extrapolation error is expected to be prohibitively large outside of the realizable set. Realizability-preserving limiters [82, 6] can mitigate this issue at the cost of a more complex solver.

5.3. Regularized Entropy-Based Moment Closures

5.3.1. Fully Regularized Entropy-Based Closure

To overcome the challenge of approximating the entropy closure near the boundary of the realizable set, a regularized version of the entropy closure is considered in [7]:

$$\min_{g \in F_{\mathbf{m}}} \langle \eta(g) \rangle + \frac{1}{2\gamma} \left\| \langle \mathbf{m}(\mathbf{v}) g \rangle - \mathbf{u} \right\|, \tag{5.11}$$

with regularization parameter $\gamma > 0$. Unlike the non-regularized problem, in the regularized entropy closure all $\mathbf{u} \in \mathbb{R}^n$ are realizable. The minimizer $f_{\mathbf{u}}^\gamma$ still has the form given in (1.62):

$$f_{\mathbf{u}}^\gamma = \eta_*'(\boldsymbol{\alpha}_{\mathbf{u},\mathrm{F}}^\gamma \cdot \mathbf{m}). \tag{5.12}$$

However, $\boldsymbol{\alpha}^{\gamma}_{\mathbf{u},\mathrm{F}}$ solves a regularized dual problem:

$$\boldsymbol{\alpha}^{\gamma}_{\mathbf{u},\mathrm{F}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmax}} \left\{ \boldsymbol{\alpha}\cdot\mathbf{u} - \langle \eta_*(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle - \frac{\gamma}{2}\|\boldsymbol{\alpha}\|^2 \right\} . \tag{5.13}$$

Finally, the Hessian of the negative of the dual objective function in Eq. (5.13) is given by

$$H^{\gamma}_n(\boldsymbol{\alpha}) = \langle \mathbf{m}\otimes\mathbf{m}\eta''_*(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle + \gamma I . \tag{5.14}$$

Since the regularization is a Tikhonov regularization of the dual problem, the condition number of $H(\boldsymbol{\alpha})$ is bounded from above by $1+\gamma^{-1}\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of $H^{\gamma}_n(\boldsymbol{\alpha})$ [7]. This mitigates the ill condition of the entropy closure near $\partial\mathcal{R}$. The regularized entropy closure inherits most of the structural properties of the standard closure; a notable exception is the translation [7] and scaling invariance.

5.3.2. Partially Regularized Entropy-Based Closure

The fully regularized entropy-based closure in (5.11) regularizes the full moment vector $\mathbf{u}$. In this work, we consider a partially regularized entropy-based closure that regularizes only the higher-order moments $\mathbf{u}_{\#}$ while maintaining the zeroth moment u_0. This partially regularized closure allows us to approximate the entropy function in the bounded, normalized realizable set $\overline{\mathcal{R}}$ and extend the approximation to the full realizable set $\mathcal{R}$ by a scaling that is based on the value of u_0. This scaling does not apply to the fully regularized closure in which the values of u_0 are affected by regularization.

To define the partially regularized entropy-based closure, we first introduce the decomposition of the velocity basis

$$\mathbf{m}(\mathbf{v}) = [m_0(\mathbf{v}), \mathbf{m}_{\#}(\mathbf{v})^{\top}]^{\top} \in \mathbb{R}^n \tag{5.15}$$

using the reduction operator # defined in (5.8). The partially regularized closure is then given by

$$\min_{g\in F_{\mathbf{m}}} \langle \eta(g)\rangle + \frac{1}{2u_0\gamma}\|\langle \mathbf{m}_{\#} g\rangle - \mathbf{u}_{\#}\|_2^2 \qquad \text{s.t. } u_0 = \langle m_0\, g\rangle \ . \tag{5.16}$$

Here, the convex entropy function h^{γ} is defined, such that for given moments $\mathbf{u}$ with $u_0 > 0$, $h^{\gamma}(\mathbf{u})$ takes the optimal objective function value of (5.16). The partially regularized closure is defined for all $\mathbf{u}\in\mathbb{R}^n$ with $u_0 > 0$ which is still much more relaxed than the realizability required by the standard closure.
If a solution to (5.16) exists, it is again of the same form as in Eq. (5.12) [7], except that the Lagrange multiplier $\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}}\in\mathbb{R}^n$ corresponding to $\mathbf{u}$ is now given by

$$\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmax}}\ \phi^{\gamma}(\boldsymbol{\alpha};\mathbf{u}), \tag{5.17}$$

$$\phi^{\gamma}(\boldsymbol{\alpha};\mathbf{u}) = \boldsymbol{\alpha}\cdot\mathbf{u} - \langle \exp(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle - \frac{u_0\gamma}{2}\|\boldsymbol{\alpha}_{\#}\|^2 . \tag{5.18}$$

Here it is straightforward to verify that ϕ^{γ} is twice differentiable and concave in $\boldsymbol{\alpha}$. Using the strong duality of (5.16)–(5.17), see [54] for proof, the entropy function for this partially regularized problem is given by $h^{\gamma}(\mathbf{u}) = \phi^{\gamma}(\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}};\mathbf{u})$.
We note that, for a fixed γ, the magnitude of this partial regularization is linearly proportional to u_0, which corresponds to a uniform regularization magnitude when the moments are normalized with respect to u_0, as considered below. In [208, 200], it was shown that it is important to construct the neural network approximations on the bounded, normalized realizable set $\overline{\mathcal{R}}$. In the following lemmas, we show that for a moment $\mathbf{u}$ with $u_0 > 0$, the multiplier $\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}}$ in (5.17) can be obtained from $\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}}$, the multiplier for the normalized moment $\overline{\mathbf{u}}$.

Lemma 5.2 (Normalized, Regularized Lagrange Multipliers)
Assume $m_0(\mathbf{v}) = 1$. Given a normalized moment $\overline{\mathbf{u}}$, let $\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}$ be the corresponding multiplier defined in (5.17)*. Then we have the relation*

$$\alpha^\gamma_{\overline{\mathbf{u}},0} = \vartheta\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_{\#}\right), \tag{5.19}$$

where the function $\vartheta\colon \mathbb{R}^{n-1} \to \mathbb{R}$ is defined as

$$\vartheta(\boldsymbol{\beta}) = -\log(\langle \exp(\boldsymbol{\beta}\cdot\mathbf{m}_{\#})\rangle)\,. \tag{5.20}$$

The proof is given in §5.8. Note that, in general, this relation does not hold for the multipliers given by the fully regularized problem (5.13).

To reduce the dimension of the dual problem, we define the function

$$\hat{\phi}^\gamma(\boldsymbol{\beta};\mathbf{w}) = \phi^\gamma\left(\left[\vartheta(\boldsymbol{\beta}),\boldsymbol{\beta}^\top\right]^\top;\left[1,\mathbf{w}^\top\right]^\top\right), \quad \text{where} \quad \mathbf{w}\in\mathbb{R}^{n-1} \text{ and } \boldsymbol{\beta}\in\mathbb{R}^{n-1}. \tag{5.21}$$

In the following analysis, we show that for normalized moments $\overline{\mathbf{u}}$, the partially regularized dual problem (5.17) can be formulated as a fully regularized problem in $\mathbb{R}^{n-1}$. The following theorem then gives properties of $\hat{\phi}^\gamma$ and relates the optimal multiplier defined in (5.17) to the solution of a reduced maximization problem given by $\hat{\phi}^\gamma$.

Theorem 5.3 (Reduced, Regularized Dual Entropy Closure)
Assume $m_0(\mathbf{v}) = 1$. Then, $\hat{\phi}^\gamma(\boldsymbol{\beta};\mathbf{w})$ can be written as

$$\hat{\phi}^\gamma(\boldsymbol{\beta};\mathbf{w}) = -1 - \log\left(\langle\exp(\boldsymbol{\beta}\cdot\mathbf{m}_{\#})\rangle\right) + \boldsymbol{\beta}\cdot\mathbf{w} - \frac{\gamma}{2}\|\boldsymbol{\beta}\|^2\,. \tag{5.22}$$

Furthermore, $\hat{\phi}^\gamma(\boldsymbol{\beta};\mathbf{w})$ is concave and twice differentiable. Consider the Hessian of $-\hat{\phi}^\gamma$ w.r.t. $\boldsymbol{\beta}$, denoted by $\hat{H}^\gamma_n(\boldsymbol{\beta})$. Let λ and λ^γ denote the eigenvalues of $\hat{H}^{\gamma=0}_n(\boldsymbol{\beta})$ and $\hat{H}^\gamma_n(\boldsymbol{\beta})$, respectively, then

$$\lambda^\gamma_{\min} = \lambda_{\min} + \gamma, \qquad \lambda^\gamma_{\max} = \lambda_{\max} + \gamma \tag{5.23}$$

and the condition number of $\hat{H}^\gamma_n(\boldsymbol{\beta})$ is bounded from above by $1 + \gamma^{-1}\lambda_{\max}$.
Next, let

$$\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_{\#}} := \operatorname*{argmax}_{\boldsymbol{\beta}\in\mathbb{R}^{n-1}} \hat{\phi}^\gamma(\boldsymbol{\beta};\overline{\mathbf{u}}_{\#})\,, \tag{5.24}$$

where $\overline{\mathbf{u}}_{\#}\in\mathbb{R}^{n-1}$ is the truncation of a normalized moment $\overline{\mathbf{u}}\in\mathbb{R}^n$. Then, $[\vartheta(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_{\#}}),(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_{\#}})^\top]^\top$ solves the maximization problem in Eq. (5.17) *at $\overline{\mathbf{u}}$, i.e., $\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}} = [\vartheta(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_{\#}}),(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_{\#}})^\top]^\top$. Further, the optimal objective function values are equal, i.e., $\phi^\gamma(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}};\overline{\mathbf{u}}) = \hat{\phi}^\gamma(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_{\#}};\overline{\mathbf{u}}_{\#})$.*

The proof is given in §5.8.

Theorem 5.3 implies that, for normalized moments $\overline{\mathbf{u}}$, the partially regularized multiplier $\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}$ in Eq. (5.17) can be obtained by solving the reduced problem in Eq. (5.24). We define $\hat{h}^\gamma(\overline{\mathbf{u}}_{\#}) = \hat{\phi}^\gamma((\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_{\#};\overline{\mathbf{u}}_{\#})$ and notice that $\hat{h}^\gamma(\overline{\mathbf{u}}_{\#}) = h^\gamma(\overline{\mathbf{u}})$ from the equivalence of the objective function values.

The results in Theorem 5.3 lead to the following corollary.

Corollary 5.4
Given the formulation of $\hat{\phi}^\gamma$, respectively $\hat{h}^\gamma$, in (5.22)*, we have*

$$\nabla_{\overline{\mathbf{u}}_\#} \hat{h}^\gamma \left(\overline{\mathbf{u}}_\#\right) = \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\# \tag{5.25}$$

and

$$\overline{\mathbf{u}}_\# = \left\langle \mathbf{m}_\# \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#\right), \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top \cdot \mathbf{m}\right)\right\rangle + \gamma \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\# . \tag{5.26}$$

Here (5.25) is a direct consequence of (5.22), and (5.26) follows from the relation between $\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}$ and $\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}$ given in Theorem 5.3 and the first order optimality condition of (5.24).

Lemma 5.5 (Scaled, Partially Regularized Lagrange Multipliers)
Consider the partially regularized dual problem (5.17) *with dynamic regularization $u_0\gamma$. Let $\overline{\mathbf{u}}$ be a normalized moment and u_0 the scaling factor to obtain $\mathbf{u}$, i.e., $u_0\overline{\mathbf{u}} = \mathbf{u}$. Then, the Lagrange multiplier of the non-normalized, partially regularized dual problem $\boldsymbol{\alpha}^\gamma_{\mathbf{u}}$ is given by*

$$\boldsymbol{\alpha}^\gamma_{\mathbf{u}} = \left[\vartheta\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#\right) + \log(u_0), \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top , \tag{5.27}$$

where $\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}$ is the regularized Lagrange multiplier of the normalized problem. Furthermore, one can reconstruct the entropy functional of the full problem given the entropy of the reduced one, i.e.,

$$h^\gamma(\mathbf{u}) = u_0 \hat{h}^\gamma(\overline{\mathbf{u}}_\#) + u_0 \log(u_0), \tag{5.28}$$

and $h^\gamma(\mathbf{u})$ is convex in $\mathbf{u}$.

The proof is given in §5.8.
Hence, we can focus on solving the fully regularized, reduced entropy closure for normalized moments, and then reconstruct the solution of the partially regularized, non-normalized closure.

5.4. Regularized, Neural Network-Based Entropy Approximations

We extend the structure-preserving neural network-based entropy closure presented in [208] to the setting of regularized entropy closures using Theorem 5.3 to solve the numerical challenge presented in §5.2. To this end, we consider an input convex neural network N_θ to approximate the entropy functional $\hat{h}^\gamma(\overline{\mathbf{u}}_\#)$ of the reduced, fully regularized closure, i.e.,

$$\hat{h}^p(\overline{\mathbf{u}}_\#) = \mathrm{N}_\theta(\overline{\mathbf{u}}_\#) \approx \hat{h}^\gamma(\overline{\mathbf{u}}_\#), \tag{5.29}$$

where we use $p = (\gamma, \theta)$. Convexity of the neural network enables preservation of the mathematical structure of the moment system, see §5.4.1. By Lemma 5.5 we get a convex approximation of the entropy $h^\gamma(\overline{\mathbf{u}}_\#)$ of the partially regularized entropy closure for non-normalized moments via $u_0\hat{h}^p(\overline{\mathbf{u}}_\#) + u_0 \log(u_0)$. We use Lemma 5.2 and the neural network approximation $(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}})_\#$ to the normalized Lagrange multiplier $(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_\#$, by Eq. (5.25), i.e.,

$$\left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}}\right)_\# = \nabla_{\overline{\mathbf{u}}_\#} \mathrm{N}_\theta(\overline{\mathbf{u}}_\#) \approx \nabla_{\overline{\mathbf{u}}_\#} \hat{\mathrm{h}}^\gamma(\overline{\mathbf{u}}_\#) = \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\# . \tag{5.30}$$

Algorithm 5.1: Network training

Input: $X_T = \bigcup_k X_{B,k}$: Training data-set $\left\{\left(h_i, \overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}\right)\right\}_{i\in T}$, partitioned in k_B batches
N_θ: Network architecture $\mathrm{N}_\theta : \overline{\mathbf{u}}_\# \mapsto \hat{h}(\overline{\mathbf{u}}_\#)$
θ^0: Weight initialization of the network
t_{epoch}: Maximum number of training iterations

Result: N_{θ^*}: Trained network for the minimal entropy closure

for $t = 0$ *to* $t = t_{\text{epoch}}$ **do**
  $\theta^{k=0} \leftarrow \theta^t$
  for $k = 0$ *to* $k = k_b$ **do**
    Load mini-batch X_{B_k}
    $\hat{h}_i^p \leftarrow \mathrm{N}_\theta\left((\overline{\mathbf{u}}_i)_\#\right)$ /* Entropy approximation */
    $(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}},i})_\# \leftarrow \nabla_{\overline{\mathbf{u}}_\#}\mathrm{N}_\theta\left((\overline{\mathbf{u}}_\mathrm{i})_\#\right), \forall \mathrm{i} \in \mathrm{B_k}$ /* Lagrange multiplier approximation */
    $\boldsymbol{\alpha}^p_{\overline{\mathbf{u}},i} \leftarrow \left[\vartheta\left(\left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}},i}\right)_\#\right), \left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}},i}\right)_\#^\top\right]^\top, \forall i \in B_k$ /* Reconstruct complete multiplier */
    $\overline{\mathbf{u}}_i^p \leftarrow \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}},i} \cdot \mathbf{m}\right)\right\rangle + \gamma\left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}},i}\right)_\#, \forall i \in B_k$ /* Reconstruct normalized moment */
    $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{ICNN}}(X_{B_k}; N_\theta)$ /* Compute loss */
    $\theta^{k+1} \leftarrow \theta^k - \nabla_\theta \mathcal{L}$ /* Update network weights */
  $\theta^{t+1} \leftarrow \theta^{k_B}$
$\theta^* \leftarrow \theta^{t_{\text{epoch}}}$ /* Save final network weights */

The reconstruction of the normalized moment is then given by Eq. (5.26)

$$\overline{\mathbf{u}}^\theta = \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}} \cdot \mathbf{m}\right)\right\rangle + \gamma\left[0, \left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top \approx \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}} \cdot \mathbf{m}\right)\right\rangle + \gamma\left[0, \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top = \overline{\mathbf{u}}. \tag{5.31}$$

The network is trained on a loss function given by the sum of mean squared errors of the network prediction, the network derivative, and the moment reconstruction,

$$\mathcal{L}_{\mathrm{ICNN}}(X_B; \theta) = \frac{1}{|T|}\sum_{i\in T}\left\|\hat{h}^\gamma((\overline{\mathbf{u}}_i)_\#) - \hat{h}^p((\overline{\mathbf{u}}_i)_\#)\right\|_2^2 + \lambda\left\|\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}},i}\right)_\# - \left(\boldsymbol{\alpha}^\theta_{\overline{\mathbf{u}},i}\right)_\#\right\|_2^2 + \left\|\overline{\mathbf{u}}_i - \overline{\mathbf{u}}_i^\theta\right\|_2^2, \tag{5.32}$$

where B is the size of one batch of the training data set, $X_T = \left\{\overline{\mathbf{u}}_\#, \boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}, \hat{h}^\gamma(\overline{\mathbf{u}}_\#)\right\}$, see Eq. (4.27). The parameter λ is used to scale the loss in $(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_\#$ to the same range as the loss in $\hat{h}^\gamma$ and $\overline{\mathbf{u}}_\#$. The size of the training set is denoted by T. For details of the network architecture, we refer to [208]. The training workflow is summarized in Algorithm 5.1

5.4.1. Entropy Dissipation

We show that the convexity of N_θ enables the preservation of key structural properties of the closed moment system, such as entropy dissipation, hyperbolicity, invariance of range, and conservation. We consider entropy dissipation first and identify a suitable entropy/entropy-flux pair for the moment system. Care must be taken when considering the entropy of the full moment system since the entropy gradient now differs from the non-regularized case (1.64) and does not recover the Lagrange multiplier of the dual closure anymore - an important property of the non-regularized moment closure.

Lemma 5.6 (Gradient of the Partially Regularized Entropy)
The gradient of the partially regularized entropy with respect to its moments, is given by

$$\nabla_{\mathbf{u}} h^{\gamma}(\mathbf{u}) = \boldsymbol{\alpha}^{\gamma}_{\mathbf{u},\mathbf{D}} = \boldsymbol{\alpha}^{\gamma}_{\mathbf{u}} - \left[\frac{\gamma}{2}\left\|\boldsymbol{\alpha}^{\gamma}_{\mathbf{u},\#}\right\|^2, 0\right]^{\top}, \tag{5.33}$$

and we denote the gradient by $\boldsymbol{\alpha}^{\gamma}_{\mathbf{u},\mathbf{D}}$.

We provide proof in §5.8. Thus the ansatz function for the reconstruction of the kinetic density needs to be adapted from (5.12) to

$$f^{\gamma}_{\mathbf{u},\mathbf{D}} = \exp\left(\boldsymbol{\alpha}^{\gamma}_{\mathbf{u},\mathbf{D}} \cdot \mathbf{m}\right) = \exp\left(\left(\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}} - \left[\frac{\gamma}{2}\left\|\left(\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}}\right)_{\#}\right\|^2, \left(\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}}\right)^{\top}_{\#}\right]^{\top}\right) \cdot \mathbf{m}\right). \tag{5.34}$$

Using Lemma 5.5, the neural network-based approximation of $\boldsymbol{\alpha}^{\gamma}_{\mathbf{u},\mathbf{D}}$ is given by

$$\boldsymbol{\alpha}^{p}_{\mathbf{u},\mathbf{D}} = \left[\vartheta\left(\left(\boldsymbol{\alpha}^{p}_{\overline{\mathbf{u}}}\right)_{\#}\right) + \log(u_0) - \frac{\gamma}{2}\left\|\left(\boldsymbol{\alpha}^{p}_{\overline{\mathbf{u}}}\right)_{\#}\right\|^2, \left(\boldsymbol{\alpha}^{p}_{\overline{\mathbf{u}}}\right)^{\top}_{\#}\right]^{\top}, \tag{5.35}$$

and $f^{p}_{\mathbf{u},\mathbf{D}}$ is defined analogously.

Theorem 5.7 (Hyperbolicity and Entropy Dissipation of the Partially Regularized Entropy Closure)
The reconstructed entropy h^{γ}*, respectively* h^{p}*, with the ansatz* (5.34) *is a suitable entropy of the moment system corresponding to the entropy-flux*

$$\mathbf{j}^{\gamma}_{\mathbf{D}}(\mathbf{u}) = \left\langle \mathbf{v}\left(f^{\gamma}_{\mathbf{u},\mathbf{D}} \log\left(f^{\gamma}_{\mathbf{u},\mathbf{D}}\right) - f^{\gamma}_{\mathbf{u},\mathbf{D}}\right)\right\rangle, \tag{5.36}$$

respectively $\mathbf{j}^{p}_{\mathbf{D}}(\mathbf{u})$*, and yields the entropy dissipation law*

$$\partial_t h^{\gamma}(\mathbf{u}) + \nabla_{\mathbf{x}} \cdot \mathbf{j}^{\gamma}_{\mathbf{D}}(\mathbf{u}) \leq 0. \tag{5.37}$$

Furthermore, the moment system

$$\partial_t \mathbf{u} + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m} f^{\gamma}_{\mathbf{u},\mathbf{D}}\right\rangle = \left\langle \mathbf{m} Q\left(f^{\gamma}_{\mathbf{u},\mathbf{D}}\right)\right\rangle \tag{5.38}$$

is hyperbolic. The same results hold for the neural network-based closure.

We provide proof in §5.8.

Next, we notice that the range of the dynamic ansatz (5.34) does not change compared to the original ansatz (5.12), i.e., $f^{\gamma}_{\mathbf{u},\mathbf{D}} > 0$, thus the physical bounds of solution vectors $\mathbf{u}$ of the moment system remain unchanged. Further, the moment system is still a conservation law for basis functions, that are collision invariants, since the dynamic ansatz does not affect the collision operator $Q(f)$.

5.4.2. Galilean Invariance

The minimal entropy closure is invariant under rotations of the velocity space [7, 161], thus any neural network-based approximation to the entropy closure is expected to be invariant under rotations as well. In the following, we construct a convex and rotation-invariant neural network-based entropy approximation.

Lemma 5.8 (Rotated Moments)
Let $\mathbf{m}(\mathbf{v})$ *be an orthonormal basis of the velocity space* $\mathbf{V} \subset \mathbb{R}^d$*. Let* $T_R : \mathbf{V} \to \mathbf{V}$ *define a rotation in* $\mathbf{V}$*. Then there exists a block diagonal orthogonal matrix* $R \in \mathbb{R}^{n\times n}$*, such that*

$$R\mathbf{u} = \langle R\mathbf{m}(\mathbf{v})f(\mathbf{v})\rangle = \langle \mathbf{m}(T_R(\mathbf{v}))f(T_R(\mathbf{v}))\rangle\,. \tag{5.39}$$

The proof is given in [89]. Note that the rotation operator is in general, not an orthogonal matrix, if the basis $\mathbf{m}$ is not orthonormal. An example of an orthonormal basis is the spherical harmonics basis. Next, we establish the relationship between a rotated moment $R\mathbf{u}$ and its Lagrange multiplier.

Lemma 5.9 (Rotated Lagrange Mutlipliers)
Let $\mathbf{m}(\mathbf{v})$ *be an orthonormal basis of the velocity space* $\mathbf{V} \subset \mathbb{R}^d$*. If* $\mathbf{u} \in \mathbb{R}^n$ *is a realizable moment for the (partially regularized or non-regularized) entropy closure, then* $R\mathbf{u}$ *is also feasible. Furthermore, we have*

$$\boldsymbol{\alpha}^{\gamma}_{R\mathbf{u}} = R\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}}. \tag{5.40}$$

We provide proof in §5.8. Lemma 5.9 also holds true for $\boldsymbol{\alpha}^{\gamma}_{\mathbf{u},\mathbf{D}}$, as the first component of the Lagrange multiplier is not affected by rotations. We assess the consequence of a rotation-invariant approximation to the entropy closure.

Definition 5.10 (Rotation-Invariance)
Consider a (neural network-based) approximation $\hat{h}^p$ *to the entropy functional* $\hat{h}^\gamma$ *and the corresponding approximation* $\boldsymbol{\alpha}^p_{\overline{u}}$ *to the Lagrange multiplier* $\boldsymbol{\alpha}^\gamma_{\overline{u}}$*. We call the approximation rotation-invariant if*

$$\boldsymbol{\alpha}^p_{R\overline{\mathbf{u}}} = R\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}} \tag{5.41}$$

for all R *corresponding to rotation in* $\mathbf{V}$*. We call the moment system of the Boltzmann equation rotation-invariant, if*

$$\partial_t R\mathbf{u} + \nabla_{\mathbf{x}} \cdot \langle \mathbf{v} \otimes \mathbf{m} f_{R\mathbf{u}}\rangle - \langle \mathbf{m}Q(f_{R\mathbf{u}})\rangle = R\left(\partial_t \mathbf{u} + \nabla_{\mathbf{x}} \cdot \langle \mathbf{v} \otimes \mathbf{m} f_{\mathbf{u}}\rangle - \langle \mathbf{m}Q(f_{\mathbf{u}})\rangle\right) \tag{5.42}$$

for all R *corresponding to rotation in* $\mathbf{V}$*.*

Theorem 5.11 (Rotation-Invariant Entropy Closure Approximation)
Consider a (neural network-based) approximation $\hat{h}^p$ *to the entropy functional* $\hat{h}^\gamma$ *of the reduced, normalized closure and the corresponding approximation* $\boldsymbol{\alpha}^p_{\overline{u}}$ *to the Lagrange multiplier* $\boldsymbol{\alpha}^\gamma_{\overline{u}}$*. Let the entropy approximation be rotation-invariant, i.e.,*

$$\boldsymbol{\alpha}^p_{R\overline{\mathbf{u}}} = R\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}} \tag{5.43}$$

for all orthogonal, block diagonal R *in the sense of Definition 5.10. Let the kinetic density of the Boltzmann moment system be given by* $f^p_{\mathbf{u},\mathbf{D}}$*, see Eq.* (5.34)*. Then the corresponding moment system of the Boltzmann equation is rotation-invariant, i.e.,*

$$\partial_t R\mathbf{u} + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m} f^p_{R\mathbf{u},\mathbf{D}}\right\rangle - \left\langle \mathbf{m}Q\left(f^p_{R\mathbf{u},\mathbf{D}}\right)\right\rangle = R\left(\partial_t \mathbf{u} + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m} f^p_{\mathbf{u},\mathbf{D}}\right\rangle - \left\langle \mathbf{m}Q\left(f^p_{\mathbf{u},\mathbf{D}}\right)\right\rangle\right) \tag{5.44}$$

The proof is provided in §5.8. Note, that Theorem 5.11 also holds true for $f^\gamma_{R\mathbf{u},\mathbf{D}}$. To construct a rotation-invariant, convex approximation to the (regularized) entropy functional, we define

$$\mathbf{V}_1 = \{\mathbf{v} \in \mathbf{V} : v_2 = 0, v_3 = 0\}\,, \qquad \mathbf{V}_{1,+} = \{\mathbf{v} \in \mathbf{V}_1 : v_1 \geq 0\}\,, \quad \text{and} \quad \mathbf{V}_{1,-} = \{\mathbf{v} \in \mathbf{V}_1 : v_1 \leq 0\}\,. \tag{5.45}$$

Additionally, we denote the velocity reference frame of the original, non-rotated moment $\overline{\mathbf{u}}$ as $\mathbf{V}_{\overline{\mathbf{u}}}$.

Algorithm 5.2: Network inference within a kinetic solver

Input: $\mathbf{u}_i$: Moments of the grid $\widetilde{\mathbf{X}}$

Result: $f^p_{\mathbf{u},i}$: Reconstructed, regularized kinetic densities of the grid $\widetilde{\mathbf{X}}$

for *each* $\mathbf{u}_i \in \widetilde{\mathbf{X}}$ **do**

$\overline{\mathbf{u}}_i^+ \leftarrow R_{\mathbf{V}_{1,+}} \overline{\mathbf{u}}_i$ /* Rotate moment onto $\mathbf{V}_{1,+}$ */

$\overline{\mathbf{u}}_i^- \leftarrow R_{\pm} R_{\mathbf{V}_{1,+}} \overline{\mathbf{u}}_i$ /* Rotate moment onto $\mathbf{V}_{1,-}$ */

$\hat{h}_i^{p,\pm} \leftarrow \mathrm{N}_\theta\left((\overline{\mathbf{u}}_i^{\pm})_\#\right)$ /* Entropy approximations */

$(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}^{\pm},i})_\# \leftarrow \nabla_{\overline{\mathbf{u}}_\#} \mathrm{N}_\theta\left((\overline{\mathbf{u}}_i^{\pm})_\#\right)$ /* Lagrange multiplier approximations */

$(\boldsymbol{\alpha}^{p,*}_{\overline{\mathbf{u}}^+,i})_\# \leftarrow \widetilde{R}_{\pm}^\top (\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}^-,i})_\#$ /* Rotate mirrored multiplier onto $\mathbf{V}_{1,+}$ */

$(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}^+,i})_\# \leftarrow \frac{1}{2}\left((\boldsymbol{\alpha}^{p,*}_{\overline{\mathbf{u}}^+,i})_\# + (\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}^+,i})_\#\right)$ /* Average */

$\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}^+,i} \leftarrow \left[\vartheta\left(\left(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}^+,i}\right)_\#\right), \left(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}^+,i}\right)_\#^\top\right]^\top$ /* Reconstruct α_0 */

$\boldsymbol{\alpha}^p_{\overline{\mathbf{u}},i} \leftarrow R_{\mathbf{V}_+}^\top \left(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}^+,i}\right)_\#$ /* Reverse original rotation of multiplier */

$f^p_{\overline{\mathbf{u}},i} \leftarrow \exp\left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}},i} \cdot \mathbf{m}\right)$ /* Reconstruct kinetic density */

$f^p_{\mathbf{u},i} \leftarrow u_{0,i} f^p_{\overline{\mathbf{u}},i}$ /* Re-scale kinetic density */

Theorem 5.12 (Rotation-Invariance of Algorithm 5.2)
Let N_θ *be an input convex neural network. Let* $\widetilde{R}$ *be defined as the rotation matrix acting on the truncated moment basis, i.e.,*

$$\widetilde{R}_{i,j} = R_{i+1,j+1}, \qquad \forall i, j = 1, \ldots, n-1. \tag{5.46}$$

Then, Algorithm 5.2 constructs a rotation-invariant, convex approximation $\hat{h}^p$ *to* $\hat{h}^\gamma$*, if* N_θ *is convex for*

$$\overline{\mathbf{u}}_\# = \left\langle \mathbf{m}_\#(T_{R_{V_{1,\pm}}}\mathbf{v}) f\left(T_{R_{V_{1,\pm}}}\mathbf{v}\right)\right\rangle, \tag{5.47}$$

i.e., moments, whose velocity reference frame is (rotated to) $V_{1,\pm}$.

The proof is provided in §5.8. Note, that $R_{1,j} = \delta_{1,j}$ and $R_{i,1} = \delta_{1,i}$.

Previous works [8, 200, 208] consider normalization of the realizable set as means to bound and reduce the size of the input-data-space of the entropy closure approximation problem. Dimension reduction by manually removing degrees of freedom associated with symmetries as the rotation has been successfully applied in, e.g., neural network-based simulation of molecular dynamics [247, 97, 61] and is applicable for the entropy closure.

5.5. Error Analysis

We have constructed an approximation to the minimal entropy closure by first introducing a reduced regularized version of the closure for normalized moments, then approximating its solution operator by a neural network, and finally re-scaling the solution to the full problem. This three-stage approximation introduces numerical errors through regularization and network approximation, which we quantify in the following.

5.5.1. Regularization Errors

The regularization errors affect the reconstruction of the kinetic density $f^{\gamma}_{\mathbf{u},\mathbf{D}}$. However, the quantity of interest is the error in the kinetic flux $\mathbf{F}(\mathbf{u}) = \left\langle \mathbf{v} \otimes \mathbf{m} f^{\gamma}_{\mathbf{u},\mathbf{D}} \right\rangle$, which is Lipschitz continuous in $\mathbf{u}$, so we consider the error in the corresponding moments. Here we need to distinguish the correct moment of the partially regularized problem

$$\mathbf{u} = \left\langle \mathbf{m} \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}} \right)_{\#} \right) + \log(u_0), \left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}} \right)^{\top}_{\#} \right]^{\top} \cdot \mathbf{m} \right) \right\rangle + u_0 \gamma \left[0, \left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}} \right)^{\top}_{\#} \right]^{\top} \tag{5.48}$$

the moment with regularization error,

$$\mathbf{u}^{\gamma} = \left\langle \mathbf{m} \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}} \right)_{\#} \right) + \log(u_0), \left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}} \right)^{\top}_{\#} \right]^{\top} \cdot \mathbf{m} \right) \right\rangle, \tag{5.49}$$

and the moment with an error through the dynamic ansatz,

$$\mathbf{u}^{\gamma}_{\mathbf{D}} = \left\langle \mathbf{m} \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}} \right)_{\#} \right) + \log(u_0) - \frac{\gamma}{2} \left\| \left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}} \right)_{\#} \right\|^2, \left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}} \right)^{\top}_{\#} \right]^{\top} \cdot \mathbf{m} \right) \right\rangle. \tag{5.50}$$

Theorem 5.13 (Approximation Error of the Partial Regularization)
Let $B^{\gamma}_M = \left\{ \boldsymbol{\beta} \in \mathbb{R}^{n-1} : \|\boldsymbol{\beta}\| < M \right\}$ be a set of truncated Lagrange multipliers with norm bound $M > 0$. We consider the regularized closure problem with parameter γ for moments $\mathbf{u}$ with $(\alpha^{\gamma}_{\overline{\mathbf{u}}})_{\#} \in B^{\gamma}_M$. Then, the regularization error in the reconstructed problem is given by

$$\left\| \mathbf{u} - \mathbf{u}^{\gamma} \right\| \leq \gamma u_0 M, \tag{5.51}$$

and the error in the dynamic ansatz is given by

$$\left\| \mathbf{u} - \mathbf{u}^{\gamma}_{\mathbf{D}} \right\| \leq u_0 \left(\gamma M + n \left(1 - \exp\left(-\frac{\gamma}{2} M^2 \right) \right) \right). \tag{5.52}$$

The proof for Theorem 5.13 is given in §5.8. Note that the error estimate of the dynamic ansatz is quite pessimistic, since typically $\left\| \overline{\mathbf{u}}^{\gamma} \right\| \ll n$, and $(\alpha^{\gamma}_{\overline{\mathbf{u}}})_{\#} < M$. Only for highly anisotropic moments, the dynamic ansatz error is expected to have an effect that exceeds the regularization error.
In [7], the authors consider the effect of a perturbation δ at the moment $\mathbf{u}$ on the error of the regularized entropy closure, due to numerical errors given by an inexact solution $\mathbf{u}_{\delta}$, of the closure in the previous iteration of the numerical solver or in its spatial-temporal discretization, i.e., $\|\mathbf{u}_{\delta} - \mathbf{u}\| \leq \delta$. Then the combined error of solving the partially regularized and normalized entropy problem, and numerical error δ is given by

$$\left\| \mathbf{u} - \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^{\gamma}_{\mathbf{u}_{\delta}} \cdot \mathbf{m} \right) \right\rangle \right\| \leq \delta + \gamma\, u_0\, M. \tag{5.53}$$

A consequence is to choose $\gamma M < \delta$, with $c > 0$, and to scale the simulation appropriately, such that $u_0 \approx 1$, then the numerical and regularization errors are in the same order of magnitude, i.e., $\delta \approx u_0 \gamma M$.

Algorithm 5.3: Regularized training data generator

Input: M: Lagrange multiplier norm boundary
τ: Eigenvalue tolerance
N: Order of the moment closure

Result: X_T: Training data-set $\left\{\left(\hat{h}_i^\gamma, \overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}^\gamma\right)\right\}_{i\in T}$

for $i = 0$ *to* $i = T$ **do**
  do
    $\boldsymbol{\beta} \sim \texttt{uniform}\left(\left\{\boldsymbol{\beta} \in \mathbb{R}^{n-1} : \|\boldsymbol{\beta}\| < M\right\}\right)$ /* Sample $B_{M,\tau}^\gamma$ */
  while $\lambda_{min}^\gamma < \tau$
  $\boldsymbol{\alpha}_{\overline{\mathbf{u}},i}^\gamma \leftarrow \left[\vartheta(\boldsymbol{\beta}), \boldsymbol{\beta}^\top\right]^\top$ /* Compute Lagrange multiplier */
  $\overline{\mathbf{u}}_i \leftarrow \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}},i}^\gamma \cdot \mathbf{m}\right)\right\rangle$ /* Reconstruct normalized moment vector */
  $\hat{h}_i^\gamma \leftarrow \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}^\gamma \cdot \overline{\mathbf{u}}_i - \left\langle \exp\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}},i}^\gamma \cdot \mathbf{m}\right)\right\rangle + \frac{\gamma}{2}\left\|\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}i}^\gamma\right)_\#\right\|^2$ /* Compute entropy functional */
  Append $\left(\hat{h}_i^\gamma, \overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}^\gamma\right)$ to X_T.

5.5.2. Neural Network Approximation Error

The last error source is the approximation error of the neural network, which can be categorized as part of the numerical error δ in the sense of Eq. (5.53). Due to the convexity of the neural network-based approximations of the entropy closure and the sampled moments, it is feasible to consider the maximum interpolation error of the neural network in the convex hull of the training data, i.e.

$$\max_{\overline{\mathbf{u}}_\# \in C(X_T)} \left\|\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\gamma\right)_\# - \nabla_{\overline{\mathbf{u}}_\#} \mathrm{N}_\theta(\overline{\mathbf{u}}_\#)\right\|, \tag{5.54}$$

where $C(X_T)$ is the convex hull of the sampled moments $\overline{\mathbf{u}}_\#$, which are part of the training data set

$$X_T = \left\{\left(\hat{h}_i^\gamma, \overline{\mathbf{u}}_i, \boldsymbol{\alpha}_{\overline{\mathbf{u}},i}^\gamma\right)\right\}_{i\in T}. \tag{5.55}$$

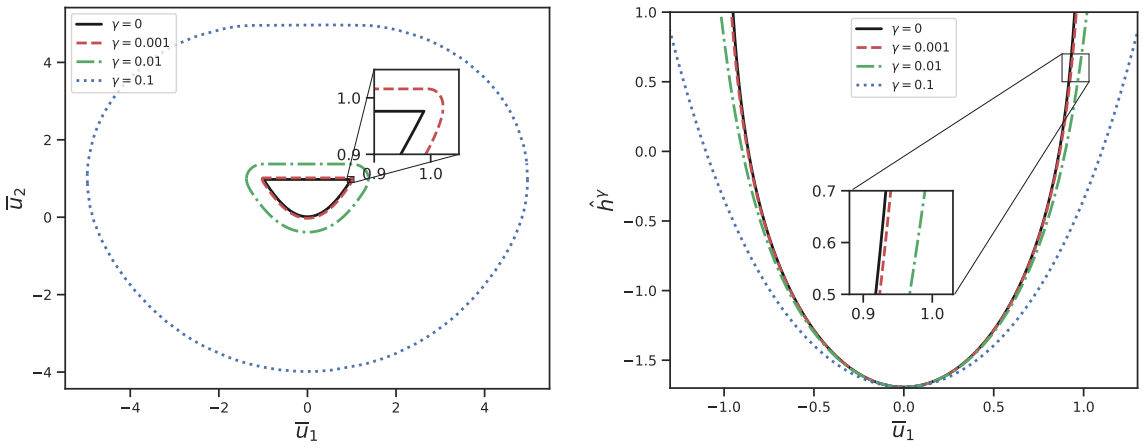
as the quantity of interest, which depends only on the distribution of the training data set X_T. For a given error bound in $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\gamma)_\#$, one can employ Eq. (5.26), to derive the perturbance in $\mathbf{u}$, which yields the influence of the neural network error in δ. For details, we refer to [208].

5.5.3. Data Sampling

In [208] a sampling strategy to minimize the maximal interpolation error (5.54) is provided. A reasonable strategy is to sample the bounded set

$$B_{M,\tau}^\gamma = \left\{\boldsymbol{\beta} \in \mathbb{R}^{n-1} : \|\boldsymbol{\beta}\| < M \cup \lambda_{\min}^\gamma > \tau\right\}. \tag{5.56}$$

uniformly [208], where $\lambda_{\min}^\gamma$ is the smallest eigenvalue of the Hessian $\hat{H}_n^\gamma(\boldsymbol{\beta})$ at the point $(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\gamma)_\#$ of the closure (5.24), and M is an additional norm boundary for the Lagrange multiplier. The training data moments and entropy values are then sampled from $B_{M,\tau}^\gamma$ using Eq. (5.26), see Algorithm 5.3. This sampling strategy applied to the reduced regularized entropy closure enables the generation of data $(\hat{h}^\gamma(\overline{\mathbf{u}}_\#), \boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\gamma, \overline{\mathbf{u}})$ with $\overline{\mathbf{u}}$ outside the realizable set $\widetilde{\mathcal{R}}$, since the feasible set of the regularized entropy closure is unbounded. Furthermore, analysis of the regularization error in Theorem 5.13 shows, that regularized moments and



(a) $\overline{\mathbf{u}} = \boldsymbol{\psi}^\gamma(\boldsymbol{\beta})$ for different γ of M_2 1D

(b) $\hat{h}^\gamma(\overline{\mathbf{u}}_\#)$ for different γ of M_1 1D

Figure 5.1.: Sampled $\hat{h}^\gamma$ and $\overline{u}_\# = \boldsymbol{\psi}(\boldsymbol{\beta})$ with $\boldsymbol{\beta} \in A = \left\{\boldsymbol{\beta} \in \mathbb{R}^{n-1} : \|\boldsymbol{\beta}\| < M\right\}$ for different γ. For small γ, the sampled data is close to the non-regularized data, but $\hat{h}^\gamma$ is finite-valued at $\partial\overline{\mathcal{R}}$, thus stabilizing training and inference of surrogate models.

non-regularized moments, generated by the same Lagrange multiplier with norm bound $\|\boldsymbol{\beta}\| < M$, have the distance

$$\left\|\boldsymbol{\psi}(\boldsymbol{\beta}) - \boldsymbol{\psi}^\gamma(\boldsymbol{\beta})\right\| \leq \gamma M, \tag{5.57}$$

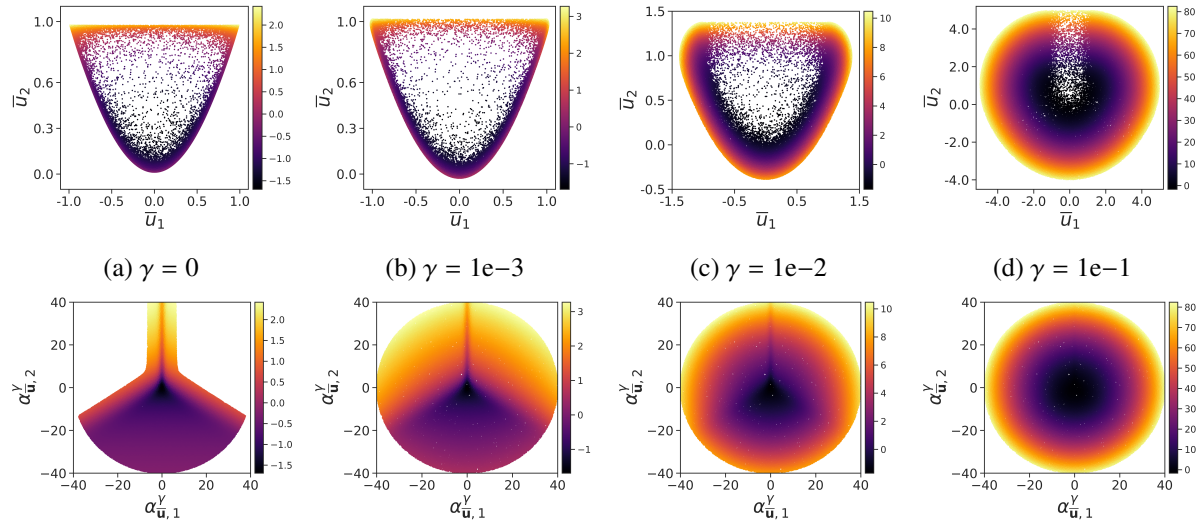
with the formal moment reconstruction maps for the regularized and non-regularized, normalized closures,

$$\boldsymbol{\psi}^\gamma(\boldsymbol{\beta}) = \left\langle \mathbf{m}_\# \exp\left(\left[\vartheta(\boldsymbol{\beta}), \boldsymbol{\beta}^\top\right]^\top \cdot \mathbf{m}\right)\right\rangle + \gamma\boldsymbol{\beta}, \quad \text{and} \quad \boldsymbol{\psi}(\boldsymbol{\beta}) = \boldsymbol{\psi}^{\gamma=0}(\boldsymbol{\beta}). \tag{5.58}$$

Remark, that $\mathbf{u} = \boldsymbol{\psi}^\gamma((\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_\#)$ and $\mathbf{u}^\gamma = \boldsymbol{\psi}((\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_\#)$ in the sense of Eq. (5.48) and Eq. (5.49).
Lastly, the regularized entropy closure functional $\hat{h}^\gamma$, $\gamma > 0$, is finite-valued at the boundary of the non-regularized realizable set $\partial\overline{\mathcal{R}}$, which mitigates the problem of diverging entropy values described by Theorem 5.1. Since $\hat{h}^\gamma$ is convex in $\overline{\mathbf{u}}_\#$, the interpolation error bound (5.54) can be extended beyond $\partial\overline{\mathcal{R}}$ of the non-regularized closure. Thus, we trade the uncontrollable extrapolation error near $\partial\mathcal{R}$ of the non-regularized entropy closure problem to a controllable interpolation error of the regularized entropy closure problem with an additional regularization error given by γM.

Figure 5.1a displays the convex hull of reconstructed moments $\boldsymbol{\psi}^\gamma(\boldsymbol{\beta})$, where $\boldsymbol{\beta} \in A = \left\{\boldsymbol{\beta} \in \mathbb{R}^{n-1} : \|\boldsymbol{\beta}\| < M\right\}$ and $\gamma \in \{0, 1e{-}1, 1e{-}2, 1e{-}3\}$. Indeed, the convex hull of the sampled $\boldsymbol{\psi}^\gamma(\boldsymbol{\beta})$, $\gamma = 1e{-}3$ and the hull of the non-regularized reconstruction $\boldsymbol{\psi}(\boldsymbol{\beta})$ are almost identical, whereas, for $\gamma = 1e{-}1$, the corresponding convex hull covers a far greater region, since γM is large in this case. Figure 5.1b gives corresponding result for $\hat{h}^\gamma$. Figure 5.2 illustrates the sampling distributions of $\overline{\mathbf{u}}_\#$ and $(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_\#$ for different γ. For $\gamma > 0$, we see in Fig. 5.2f, 5.2g and 5.2h that within the norm boundary $\left\|(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_\#\right\| \leq M = 40$, all Lagrange multipliers fulfill the eigenvalue threshold $\tau = 0.01$. The data generator is part of the KiT-RT framework [147] and can be found in the corresponding GitHub repository[3].

[3] https://github.com/CSMMLab/KiT-RT



(a) $\gamma = 0$ (b) $\gamma = 1\text{e}{-3}$ (c) $\gamma = 1\text{e}{-2}$ (d) $\gamma = 1\text{e}{-1}$

(e) $\gamma = 0$ (f) $\gamma = 1\text{e}{-3}$ (g) $\gamma = 1\text{e}{-2}$ (h) $\gamma = 1\text{e}{-1}$

Figure 5.2.: Sampling distributions of $\overline{\mathbf{u}}_{\#} = \boldsymbol{\psi}^{\gamma}\left(\left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}}\right)_{\#}\right)$ (top row) and corresponding $\left(\boldsymbol{\alpha}^{\gamma}_{\overline{\mathbf{u}}}\right)_{\#} \in B^{\gamma}_{M=40,\tau=0.01}$ (bottom row). The value of $\hat{h}^{\gamma}$ is color coded. Less regularization leads to steeper slopes of $\hat{h}^{\gamma}$ and thus higher sampling densities in regions with large $\left\|\overline{\mathbf{u}}\right\|$.

Table 5.1.: Architecture and the number of trainable parameters of the models for each closure after architecture search. We report the layer output dimension and the number of layers for each model. ICNNs require an order of magnitude fewer parameters than ResNets to achieve similar test error levels.

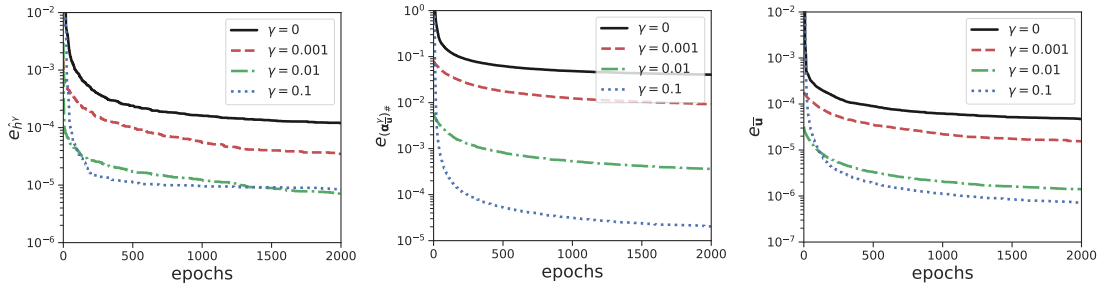
	ICNN			ResNet		
closure	units	layers	params	units	layers	params
M_2	100	3	3.2e5	300	6	5.4e6
M_3	300	3	2.8e6	400	6	9.6e6
M_4	400	3	5.4e6	600	6	2.1e7

5.6. Numerical Results

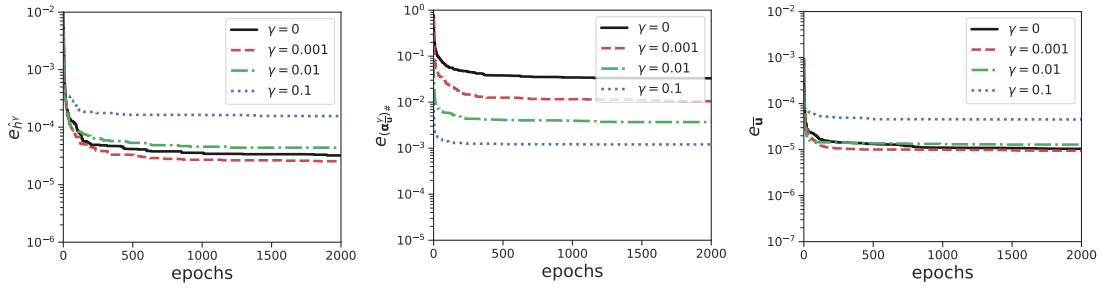
5.6.1. Neural Network Training

In this section, we evaluate the training performance of the neural networks approximating the reduced, regularized entropy closure for different regularization levels and moment orders. We consider test cases with spatial domain $\mathbf{X} \subset \mathbb{R}^2$ and $\mathbf{V}$ projected onto $\mathbb{R}^2$, so we train models with $d = 2$ spatial dimensions. For a given order of the closure N, we sample $B^{\gamma}_{M,\tau}$, where M and τ are chosen, s.t. training the closure for $\gamma = 0$ is numerically stable. Then $B^{\gamma}_{M,\tau}$ is sampled for $\gamma = 1\text{e}{-1}, 1\text{e}{-2}, 1\text{e}{-3}$ using the same M and τ. We partition the sampled data in 90% training and 10% test data.

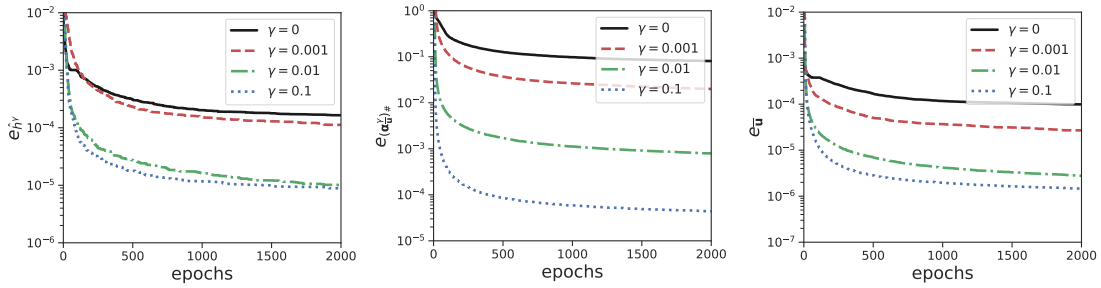
The test accuracy of the input convex neural network (ICNN), and therefore structure preserving architecture, is compared with a non-convex ResNet architecture that serves as a baseline. The ICNN models



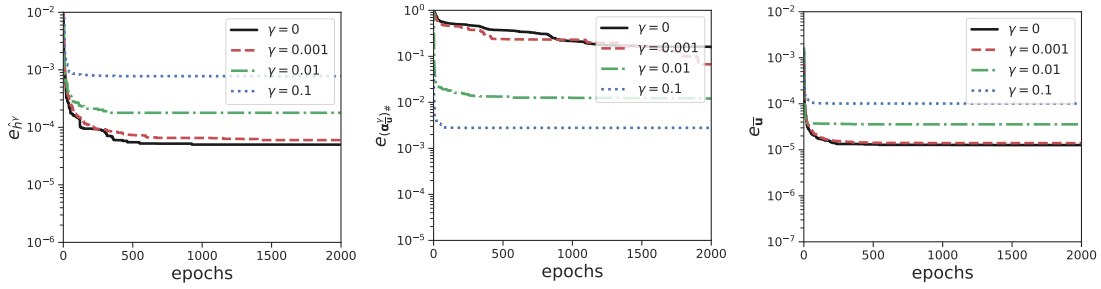
(a) M_3, ICNN, $e_{\hat{h}^\gamma}$ (b) M_3, ICNN, $e_{(\alpha^\gamma_{\mathbf{u}})_\#}$ (c) M_3, ICNN, $\hat{e}^\gamma_{\mathbf{u}}$



(d) M_3, ResNet, $e_{\hat{h}^\gamma}$ (e) M_3, ResNet, $e_{(\alpha^\gamma_{\mathbf{u}})_\#}$ (f) M_3, ResNet, $\hat{e}^\gamma_{\mathbf{u}}$



(g) M_4, ICNN, $e_{\hat{h}^\gamma}$ (h) M_4, ICNN, $e_{(\alpha^\gamma_{\mathbf{u}})_\#}$ (i) M_4, ICNN, $\hat{e}^\gamma_{\mathbf{u}}$



(j) M_4, ResNet, $e_{\hat{h}^\gamma}$ (k) M_4, ResNet, $e_{(\alpha^\gamma_{\mathbf{u}})_\#}$ (l) M_4, ResNet, $\hat{e}^\gamma_{\mathbf{u}}$

Figure 5.3.: Comparison of ICNN- and ResNet-based M_3 (a)-(f) and M_4 (g)-(l) closures of different regularization levels γ. The test errors of the ICNN model reduce heavily for increasing γ, whereas ResNet test errors reduce only slightly by regularization. Results for M_2 are comparable.

Table 5.2.: Mean test error (MSE), see Eq. (5.61) of neural network-based entropy closures of 10 experiment repetitions each with less than 10% standard deviation. The regularization level with the best result for each architecture and closure order is marked.

		ICNN			ResNet		
closure	γ	$e_{\hat{h}^\gamma}$	$e_{(\alpha^\gamma_{\overline{\mathbf{u}}})_\#}$	$e_{\overline{\mathbf{u}}}$	$e_{\hat{h}^\gamma}$	$e_{(\alpha^\gamma_{\overline{\mathbf{u}}})_\#}$	$e_{\overline{\mathbf{u}}}$
M_2	0	1.45e−5	5.23e−3	1.17e−5	1.77e−5	4.28e−3	4.37e−6
M_2	1e−3	1.02e−5	2.69e−3	8.87e−6	1.44e−5	2.57e−3	4.05e−6
M_2	1e−2	1.34e−6	9.32e−5	**7.81e−7**	**1.08e−6**	7.78e−4	**3.92e−6**
M_2	1e−1	**1.24e−6**	**5.12e−5**	1.71e−6	3.22e−5	**3.57e−4**	1.65e−5
M_3	0	1.19e−4	4.05e−2	4.73e−5	3.92e−5	3.23e−2	1.04e−5
M_3	1e−3	3.52e−5	9.21e−3	1.55e−5	**2.55e−5**	1.04e−2	**9.44e−6**
M_3	1e−2	**7.04e−6**	3.61e−4	1.40e−6	4.40e−5	3.70e−3	1.28e−5
M_3	1e−1	8.09e−6	**2.03e−5**	**7.14e−7**	1.50e−4	**1.25e−3**	4.51e−5
M_4	0	1.65e−4	8.07e−2	9.88e−5	**5.80e−5**	1.64e−1	1.26e−5
M_4	1e−3	1.12e−4	2.00e−2	2.69e−5	6.21e−5	6.71e−2	**1.39e−5**
M_4	1e−2	1.01e−5	7.94e−4	2.76e−6	1.79e−4	1.12e−2	3.56e−5
M_4	1e−1	**8.87e−6**	**4.39e−5**	**1.46e−6**	7.75e−4	**2.79e−3**	1.01e−4

consist of convex layers [208, 9] build from two weight matrices each, i.e.,

$$\begin{aligned} \mathrm{N}_\theta(\overline{\mathbf{u}}_\#) &= \mathbf{z}_M, \\ \mathbf{z}_k &= \sigma_k(W_k^{\mathbf{z}}\mathbf{z}_{k-1} + W_k^{\mathbf{u}}\overline{\mathbf{u}}_\# + \mathbf{b}_k), \qquad k = 2, \dots, M, \\ \mathbf{z}_1 &= \sigma_1(W_1^{\mathbf{u}}\overline{\mathbf{u}}_\# + \mathbf{b}_1), \end{aligned} \tag{5.59}$$

where $W_k^{\mathbf{z}}$ has positive entries and σ_k is convex and non-decreasing, see §4 for implementation details. The ResNet consists of dense layers with skip connection, i.e.,

$$\begin{aligned} \mathrm{N}_\theta(\overline{\mathbf{u}}_\#) &= \mathbf{z}_M, \\ \mathbf{z}_k &= \sigma_k(W_k\mathbf{z}_{k-1} + \mathbf{b}_k) + \mathbf{z}_{k-1}, \qquad k = 1, \dots, M, \\ \mathbf{z}_0 &= \overline{\mathbf{u}}_\#. \end{aligned} \tag{5.60}$$

We give an overview of the used layers and output dimension of the weight matrices of each layer as well as the total number of learnable parameters in Table 5.1. For a given closure, the number of layers and neurons are determined in an architecture search at $\gamma = 0$ and then fixed for all regularization levels for comparability, and we see in Table 5.1, that the ICNN models require fewer parameters for similar or better training performance.

Each model for each closure and regularization level is initialized with normally distributed weights and then trained in single precision on an RTX3090 GPU for a total of 2000 epochs with a batch size of 256 on the loss function of Eq. (5.32), to ensure convergence. The experiment is repeated 10 times for each closure, regularization, and model until the results have a standard deviation of less than 10%, i.e., a total of 240 networks are trained. All following statements refer to the mean of the results for each case. Table 5.2 displays the mean test errors of the training runs, i.e.,

$$e_{\hat{h}^\gamma} = \frac{1}{T_{\text{test}}} \sum_{i \in T_{\text{test}}} \left\| \hat{h}^\gamma((\overline{\mathbf{u}}_i)_\#) - \hat{h}^p((\overline{\mathbf{u}}_i)_\#) \right\|_2^2, \tag{5.61}$$

Table 5.3.: Mean test error (MSE) of neural network-based entropy closures of 10 experiment repetitions each with less than 10% standard deviation. Models of all regularization levels are tested against data of the non-regularized closure to measure the combined regularization and network approximation error, see Eq. (5.62). The regularized models with the least combined error are marked.

		ICNN			ResNet		
closure	γ	$\hat{e}^{\gamma}_{\hat{h}}$	$\hat{e}^{\gamma}_{(\alpha_{\overline{\mathbf{u}}})_\#}$	$\hat{e}^{\gamma}_{\overline{\mathbf{u}}}$	$\hat{e}^{\gamma}_{\hat{h}}$	$\hat{e}^{\gamma}_{(\alpha_{\overline{\mathbf{u}}})_\#}$	$\hat{e}^{\gamma}_{\overline{\mathbf{u}}}$
M_2	0	1.45e−5	5.23e−3	1.17e−5	1.77e−5	4.28e−3	4.37e−6
M_2	1e−3	**1.78e−4**	**1.68e−2**	**1.34e−5**	**9.33e−5**	**1.12e−2**	**7.94e−5**
M_2	1e−2	3.39e−3	2.76e−1	1.69e−4	3.44e−3	2.92e−1	1.09e−4
M_2	1e−1	7.97e−2	1.13e−0	5.5e−3	7.7e−2	1.12e−0	8.83e−3
M_3	0	1.19e−4	4.05e−2	4.73e−5	3.92e−5	3.23e−2	1.04e−5
M_3	1e−3	**2.01e−4**	**1.10e−1**	**2.01e−5**	**6.58e−5**	**1.32e−1**	**7.49e−5**
M_3	1e−2	3.84e−3	5.82e−1	9.45e−5	3.86e−3	5.7e−1	1.12e−4
M_3	1e−1	7.47e−2	1.32e−0	3.06e−3	8.39e−2	1.32e−0	3.13e−3
M_4	0	1.65e−4	8.07e−2	9.88e−5	5.80e−5	1.64e−1	1.26e−5
M_4	1e−3	**2.69e−4**	**4.07e−1**	**3.1e−5**	**2.12e−4**	**4.10e−1**	**2.48e−5**
M_4	1e−2	4.75e−3	1.29e0	6.11e−5	2.78e−3	1.15e0	1.04e−4
M_4	1e−1	6.71e−2	1.85e0	1.73e−3	7.35e−2	1.83e0	3.06e−3

and analogously for $e_{\overline{\mathbf{u}}}$ and $e_{(\alpha^{\gamma}_{\overline{\mathbf{u}}})_\#}$. Figure 5.3 shows the training performance for ICNN and ResNet approximations of entropy (left column), reduced Lagrange multiplier (middle column), and reconstructed moment (right column) of the M_3 and M_4 closure for different regularization parameters. The plots display the best test error until the current epoch.
We can see, that the test error reduces by several orders of magnitude for larger values of γ for h, $(\alpha^{\gamma}_{\overline{\mathbf{u}}})_\#$ as well as $\overline{\mathbf{u}}_\#$ for the input convex network for the M_2, M_3 and M_4 closure. The highly regularized models train until single precision floating point accuracy. We believe that the steeper slope of the entropy function of the non-regularized problem is harder to approximate, see Fig. 5.1b. This trend is not as pronounced for the ResNet approximations, where models exhibit inconsistent approximation performance for $\hat{h}^{\gamma}$,$(\alpha^{\gamma}_{\overline{\mathbf{u}}})_\#$ and $\overline{\mathbf{u}}$. Furthermore, highly regularized ICNN approximations outperform ResNet approximations by an order of magnitude, especially for M_3 and M_4 closures.

Finally, we measure the combined neural network approximation and regularization error in Table 5.3, i.e.,

$$\hat{e}^{\gamma}_{\hat{h}} = \frac{1}{T_{\text{test}}} \sum_{i \in T_{\text{test}}} \left\| \hat{h}^{\gamma=0}((\overline{\mathbf{u}}_i)_\#) - \hat{h}^{p}((\overline{\mathbf{u}}_i)_\#) \right\|_2^2, \tag{5.62}$$

and analogously for $\hat{e}^{\gamma}_{(\alpha_{\overline{\mathbf{u}}})_\#}$. The error in the reconstructed moment is defined as

$$\hat{e}^{\gamma}_{\overline{\mathbf{u}}} = \frac{1}{T_{\text{test}}} \sum_{i \in T_{\text{test}}} \left\| \overline{\mathbf{u}}_i - \overline{\mathbf{u}}^{p}_i \right\|_2^2 \tag{5.63}$$

Table 5.4.: Computational setup for the numerical test cases

Test-Case	CFL	t_f	Δ_t	Δ_x
Linesource	0.3	0.75	1.68e−3	1e−2
Hohlraum	0.2	2	9.5e−4	7.5e−3

with $\overline{\mathbf{u}}^p = \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}} \cdot \mathbf{m}\right)\right\rangle$ in the sense of Eq. (5.49). Since we have

$$\langle \mathbf{m} \exp(\boldsymbol{\alpha}_{\overline{\mathbf{u}}} \cdot \mathbf{m})\rangle = \overline{\mathbf{u}} = \overline{\mathbf{u}}^\gamma + \gamma \left[0, \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)^\top_{\#}\right] \top = \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}} \cdot \mathbf{m}\right)\right\rangle + \gamma \left[0, \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)^\top_{\#}\right] \top \tag{5.64}$$

using Eq. (5.49) and Eq. (5.48), the approximation error in $\overline{\mathbf{u}}$ can be reformulated as

$$\left\|\overline{\mathbf{u}} - \overline{\mathbf{u}}^p\right\| \leq \left\|\overline{\mathbf{u}} - \overline{\mathbf{u}}^\gamma\right\| + \left\|\overline{\mathbf{u}}^\gamma - \overline{\mathbf{u}}^p\right\| \leq \gamma M + \left\|\overline{\mathbf{u}}^\gamma - \overline{\mathbf{u}}^p\right\| \qquad \text{(Theorem 5.13)} \tag{5.65}$$

$$\leq \gamma M + \left\|\overline{\mathbf{u}} - \overline{\mathbf{u}}^\theta\right\| + \gamma \left\|\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_{\#} - \left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}}\right)_{\#}\right\| \qquad \text{(Eq (5.49))}, \tag{5.66}$$

where we can estimate $\left\|(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_{\#} - (\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}})_{\#}\right\| \approx (e_{(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_{\#}})^{\frac{1}{2}}$ and $\left\|\overline{\mathbf{u}} - \overline{\mathbf{u}}^\theta\right\| \approx (e_{\overline{\mathbf{u}}})^{\frac{1}{2}}$. Comparing with the results of Table 5.2 we see that $\hat{e}^\gamma_{\overline{\mathbf{u}}}$ dominated by the regularization error, which is confirmed by the results of Table 5.3.
Although regularization plays a dominant role in the moment error, remark that the error control of the neural network, see Eq. (5.54), is only possible for regularized closures in the case of anisotropic moments. Here, $e_{\overline{\mathbf{u}}}$ can be large in the non-regularized closure. A trade-off must be found between approximation and regularization errors, see Table 5.5.

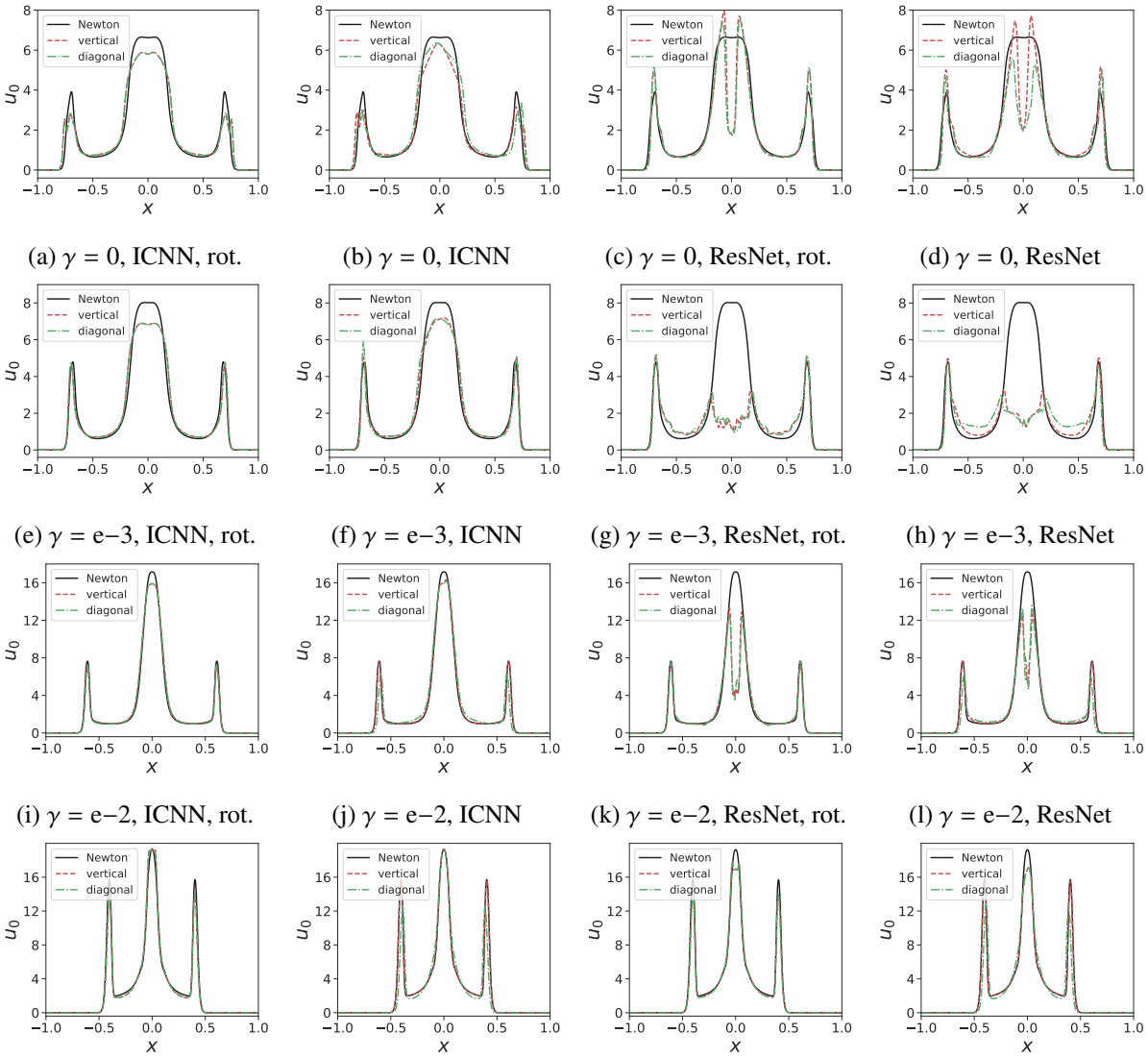
5.6.2. Linesource Test Case

The Linesource benchmark [79] is a torture test for numerical methods for kinetic equations and exposes the advantages and disadvantages of different velocity space discretizations [30, 82, 101, 147, 183, 201]. Remark, that the goal of this section is not to improve the quality of the existing (regularized) M_N method itself, but to evaluate the approximation stability corresponding to the M_N solution of the neural network-based closure surrogate model under extreme conditions. The spatial-temporal discretization is computed using a kinetic scheme of the open-source radiative transport package KiT-RT [147] and available on GitHub[4].

The physical setup is given by an initial pulse of particles distributed isotropically along an infinite line in three-dimensional space. The particles travel through a homogeneous material medium with a constant scattering cross-section σ_s. Since the Linesource problem is invariant to the third spatial dimension, one typically considers the projected problem onto two-dimensional space, i.e., $\mathbf{X} \in \mathbb{R}^2$ and $\mathbf{V} = \left\{\mathbf{v} \in \mathbb{R}^2 : \|\mathbf{v}\| \leq 1\right\}$.
In this work, we consider the problem with an isotropic collision kernel, i.e. $k(\mathbf{v}, \mathbf{v}') = 1.0$. The collision operator is given by

$$Q(f)(\mathbf{v}) = \int_{P_{\mathbb{R}^2}\mathbb{S}^2} f(\mathbf{v}_*) - f(\mathbf{v})\, \mathrm{d}\mathbf{v}_*. \tag{5.67}$$

[4] https://github.com/CSMMLab/KiT-RT



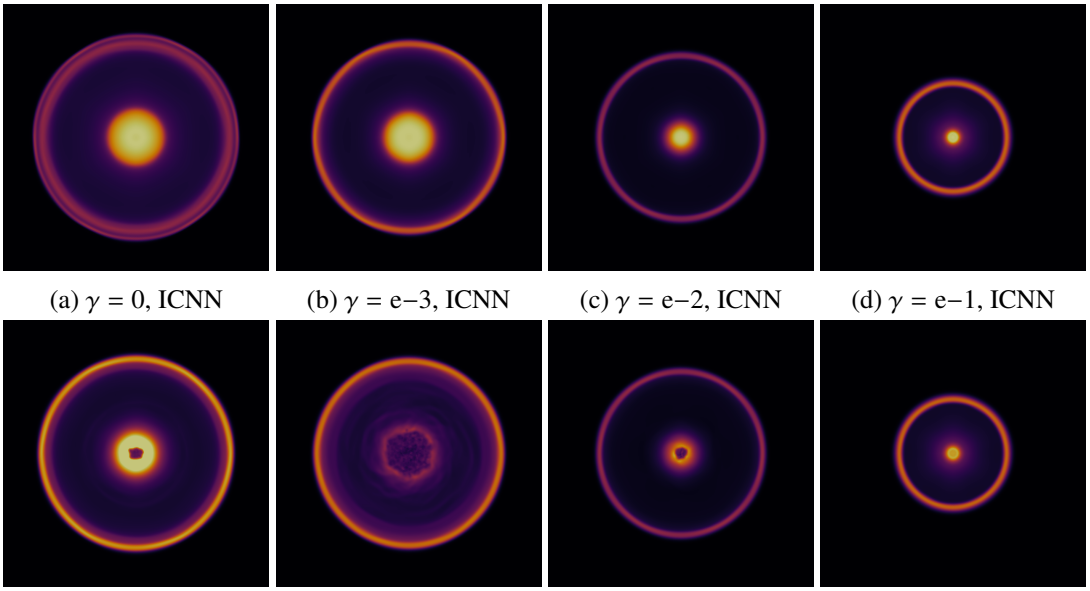
(a) $\gamma = 0$, ICNN, rot. (b) $\gamma = 0$, ICNN (c) $\gamma = 0$, ResNet, rot. (d) $\gamma = 0$, ResNet

(e) $\gamma = \mathrm{e}{-3}$, ICNN, rot. (f) $\gamma = \mathrm{e}{-3}$, ICNN (g) $\gamma = \mathrm{e}{-3}$, ResNet, rot. (h) $\gamma = \mathrm{e}{-3}$, ResNet

(i) $\gamma = \mathrm{e}{-2}$, ICNN, rot. (j) $\gamma = \mathrm{e}{-2}$, ICNN (k) $\gamma = \mathrm{e}{-2}$, ResNet, rot. (l) $\gamma = \mathrm{e}{-2}$, ResNet

(m) $\gamma = \mathrm{e}{-1}$, ICNN, rot. (n) $\gamma = \mathrm{e}{-1}$, ICNN (o) $\gamma = \mathrm{e}{-1}$, ResNet, rot. (p) $\gamma = \mathrm{e}{-1}$, ResNet

Figure 5.4.: Linesource test case, neural network-based M_2 simulations (vertical and horizontal cross-sections) in comparison to the (regularized) Newton-based closure. Each row denotes a regularization level. Columns (from left to right) denote rotated ICNN, ICNN, rotated ResNet, and ResNet. Rotated ICNN captures the test-case dynamics best, whereas ResNet-based simulations experience heavy artifacts. Remark the regularization error for $\gamma > 0$, that results in a lagging wave-front. The M_3 and M_4 closures yield similar results.



(a) $\gamma = 0$, ICNN (b) $\gamma = \mathrm{e}{-3}$, ICNN (c) $\gamma = \mathrm{e}{-2}$, ICNN (d) $\gamma = \mathrm{e}{-1}$, ICNN

(e) $\gamma = 0$, ResNet (f) $\gamma = \mathrm{e}{-3}$, ResNet (g) $\gamma = \mathrm{e}{-2}$, ResNet (h) $\gamma = \mathrm{e}{-1}$, ResNet

Figure 5.5.: Linesource test case, neural network-based M_2 simulations with rotated ICNN and ResNet-based closures and increasing regularization. Rotated ICNN with captures the test-case dynamics well, whereas ResNet-based simulations experience heavy artifacts, especially in the domain center. The M_3 and M_4 closures yield similar results.

The moment system then becomes

$$\partial_t \mathbf{u}(t,\mathbf{x}) + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m}(\mathbf{v}) f^p_{\mathbf{u},\mathbf{D}} \right\rangle = \left\langle \mathbf{m}(\mathbf{v}) Q\left(f^p_{\mathbf{u},\mathbf{D}}\right) \right\rangle. \tag{5.68}$$

Whereas the analytic initial condition is an isotropic Dirac pulse, for numerical simulations a steep Gaussian is used as an approximation,

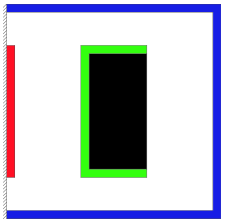
$$f(t=0,\mathbf{x},\mathbf{v}) = \frac{1}{4\pi\epsilon} \exp\left(\frac{-\|\mathbf{x}\|_2^2}{4\pi\epsilon}\right), \qquad \forall \mathbf{x} \in \mathbf{X}, \mathbf{v} \in \mathbf{V}, \tag{5.69}$$

with $\epsilon = 0.0032$. Thus, artifacts of the spatial discretization are avoided, whereas velocity artifacts remain. The infinite physical domain is represented by a sufficiently large computational domain $\mathbf{X} = [-1,1]^2$, such that the wavefront never reaches the domain boundary. We use zero-value Dirichlet boundary conditions. Lastly, the moments with respect to the velocity basis $\mathbf{m}(\mathbf{v})$ are computed to yield the initial conditions for the M_N solver. The solver settings are displayed in Table 5.4.

The M_N methods perform comparatively well in the Linesource test case [82] and accurately track the wave-front, whereas linear Galerkin type closures as P_N tend to oscillate and nodal methods as S_N exhibit ray effects. Note that the M_2 closure, which we consider in the following, exhibits a standing wave in the domain center, which is a model artifact of even order entropy closures. It is a region of interest to test the neural network approximations since the standing wave displays an extreme case for Theorem 5.3, where the re-scaled regularization error of see Theorem 5.13, i.e.,

$$\left\|\mathbf{u} - \mathbf{u}^\gamma\right\| \leq \gamma \left\|u_0\right\| M. \tag{5.70}$$

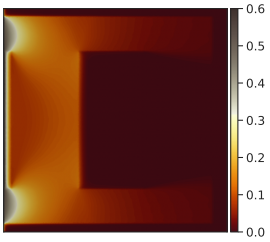
is big due to large u_0, which is in $O(100)$ during the first iterations of the simulation. This influences the numerical error δ of Eq. (5.53) for the next iterations. Additionally, near the wavefront of the moving



(a) Hohlraum Geometry

Region	σ_t	σ_s	σ_a
white	0.1	0.1	0.0
red	100.0	95.0	5.0
green	100.0	90.0	10.0
blue	100.0	0.0	100.0
black	100.0	50.0	50.0

(b) Physical properties



(c) S_{50} reference solution

Figure 5.6.: Hohlraum computational geometry (a), with inflow boundary conditions on the left face and Dirichlet boundary conditions on all other faces. Physical properties of the color coded areas of the Hohlraum geometry (b), and the reference solution computed with the ordinate S_{50} method (c).

particles, highly an-isotropic moments with large $\alpha_{\mathbf{u}}^{\gamma}$ appear, resulting in large regularization errors, and potentially high network approximation errors.

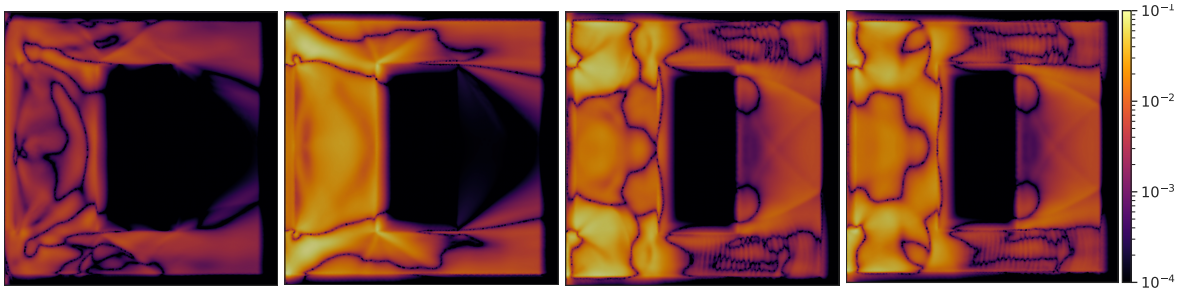
In Figure 5.5, we compare neural network-based closures for different regularization levels as well using Algorithm 5.2. Figure 5.4 displays the corresponding vertical and diagonal cross-sections alongside the analytic solution and the cross-section of the Newton-based closure, as well as non-rotated network-based closures.
The rotated ICNN closure indeed preserves the rotation-invariance of the solution, whereas the non-rotated ICNN as well as the ResNet solution fail to maintain rotational invariance for $\gamma = 0$ and $\gamma = 1\text{e}{-3}$. Furthermore, the approximation accuracy to the regularized Newton reference solution increases for increasing regularization, which follows the increased training performance for these closures. However, highly regularized solutions (Newton and neural network-based) deviate more from the analytic solution, as the regularization reduces the speed of information transport.
The two right columns of Figure 5.4 show the simulation with a non-convex ResNet and rotated ResNet-based closure, as well as their cross sections, respectively. Note, that the ResNet-based closure exhibits severe artifacts in the center of the Linesource pulse for both rotated and non-rotated solutions. This can be explained by the lack of convexity of the neural network approximation of the entropy functional and thus lack of hyperbolicity of the moment system.

5.6.3. Hohlraum Test Case

The Hohlraum test case [51] is a simplified version of Hohlraum configurations used in nuclear fusion devices. Originally proposed in [30] was a coupled system of radiative transfer and an energy equation for the background material. A subsequently simplified version is described in [103, 49, 50], where the nonlinear thermal absorption and re-emission of radiation are replaced by particle scattering. The corresponding test-case design is displayed in Fig. 5.6Again we consider the slab geometry setting with a collision operator given by

$$Q(f)(\mathbf{v}) = \int_{P_{\mathbb{R}^2}\mathbb{S}^2} \sigma_s(\mathbf{x}) \left[f(\mathbf{v}_*) - f(\mathbf{v}) \right] d\mathbf{v}_*, \tag{5.71}$$

where the isotropic collision kernel $k(\mathbf{x}, \mathbf{v}, \mathbf{v}') = \sigma_s(\mathbf{x})$ has a spatial dependence as specified in the color-coded areas described in Fig. 5.6a and Table 5.6b. Collision and space dependent absorption $\sigma_a \mathbf{x}$ mod-



(a) $\left|u_0^{\text{ICNN}} - u_0^{\text{Newton}}\right|$ (b) $\left|u_0^{\text{S}_{50}} - u_0^{\text{Newton}}\right|$ (c) $\left|u_0^{\text{ICNN}} - u_0^{\text{Newton}}\right|$ (d) $\left|u_0^{\text{S}_{50}} - u_0^{\text{Newton}}\right|$

Figure 5.7.: Norm differences of regularized M_2 (a-b) and M_3 (c-d) entropy closures with γ = 1e−3 to a Newton and S_{50} reference. The network approximation error (a) and (c)) is much smaller than the difference between M_N and S_N models (b) and (d).

eling yield the moment system

$$\partial_t \mathbf{u}(t,\mathbf{x}) + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m}(\mathbf{v}) f^p_{\mathbf{u},\mathbf{D}} \right\rangle = \left\langle \mathbf{m}(\mathbf{v}) Q\left(f^p_{\mathbf{u},\mathbf{D}}\right) \right\rangle - \sigma_a \mathbf{u}, \tag{5.72}$$

with collision operator Q is given by Eq. (5.71). We have $\sigma_t = \sigma_a + \sigma_s$. The reference solution for the Hohlraum test case is a very high order nodal simulation, given by the discrete ordinates method of order 50, i.e. S_{50}, [164] and displayed in Fig. 5.6c.

Simulation Quality of the regularized, ICNN-Based M_N Method

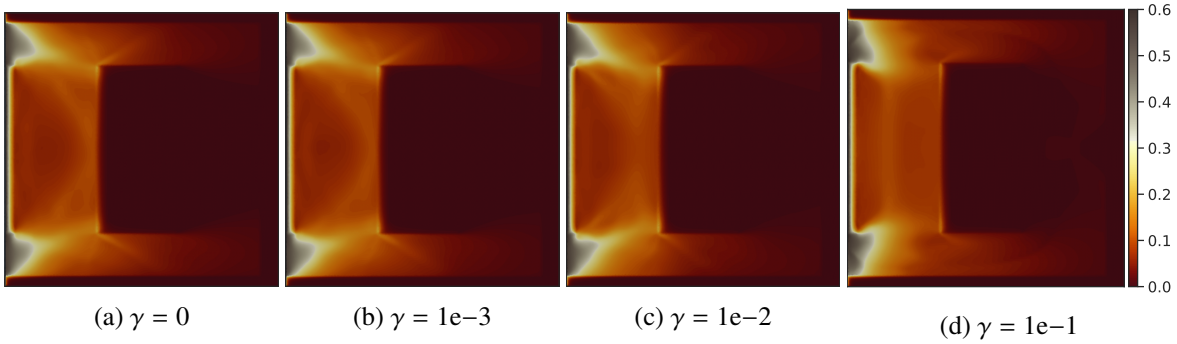
All figures of the Hohlraum solution display the order zero moment $u_0(t_f, \mathbf{x})$, which represents the scalar flux of the solution. Figures displaying the error to the reference S_{50} solution show the pointwise l_1 difference

$$e(t_f, \mathbf{x}) = \left|u_0^{\text{ref}}(t_f, \mathbf{x}) - u_0(t_f, \mathbf{x})\right| \tag{5.73}$$

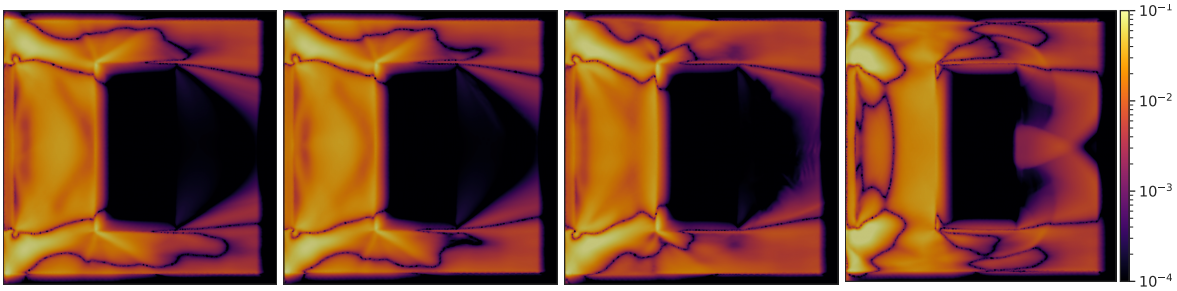
at final time t_f of the given numerical method.

In Fig. 5.7, we compare the Newton-based, regularized M_N simulation and the ICNN-based, regularized M_N simulation against the S_{50} reference solution for M_2 and M_3 with γ = 1e−3. One sees that the difference between ICNN and Newton-based regularized M_2 simulation is smaller than the l_1 difference between the Newton-based solution and the S_{50} reference solution. In the case of the regularized M_3 closure, both errors are in the same order of magnitude.

Thus, we consider the l_1 difference between the ICNN-based M_N and the reference S_{50} simulation for different closure and regularization levels. Figure 5.8 and Fig. 5.9 display the M_2 closure based on the non-rotated and rotated ICNN approximation, respectively. Figure 5.10 and 5.11 show the corresponding evaluations of the ICNN-based M_3 and M_4 closures. We see that with increased closure order, the errors in the streaming regions near the left side inflows are significantly reduced and the solution quality on the right half of the simulation increases by almost an order of magnitude. Considering Fig. 5.10 and 5.11, we further see that the difference to the S_{50} solution increases for $\gamma = 0.1$ in the M_3 and M_4 simulations compared to smaller γ, which is an effect of the regularization error of the entropy closure. Theorem 1 of [7] recommends choosing $\gamma \in \mathcal{O}(\Delta x)$, i.e., in the same order of magnitude as the spatial resolution of the simulation. Consequently, a trade-off between training performance and regularization error has to be made when configuring a simulation: Higher order entropy closures yield lower differences to the

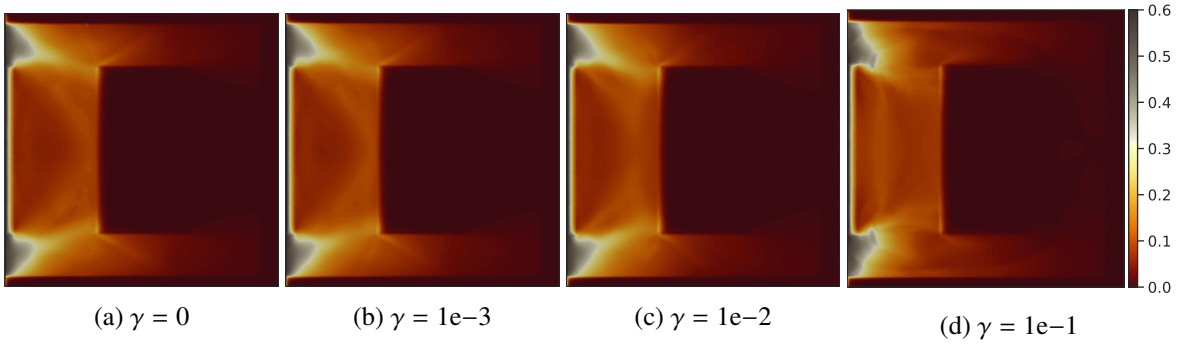


(a) $\gamma = 0$ (b) $\gamma = 1e{-}3$ (c) $\gamma = 1e{-}2$ (d) $\gamma = 1e{-}1$

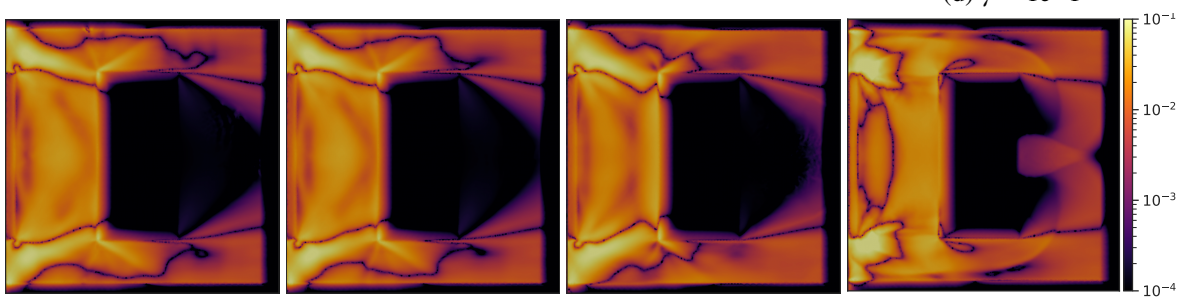


(e) $\gamma = 0$ (f) $\gamma = 1e{-}3$ (g) $\gamma = 1e{-}2$ (h) $\gamma = 1e{-}1$

Figure 5.8.: ICNN-based M_2 closures with different regularization (top row) and $e(t_f, \mathbf{x})$ (bottom row). The non-regularized surrogate model performs best. Regularization errors manifest as artifacts near the left-hand side inflows of radiation, which overshadow neural network approximation errors.

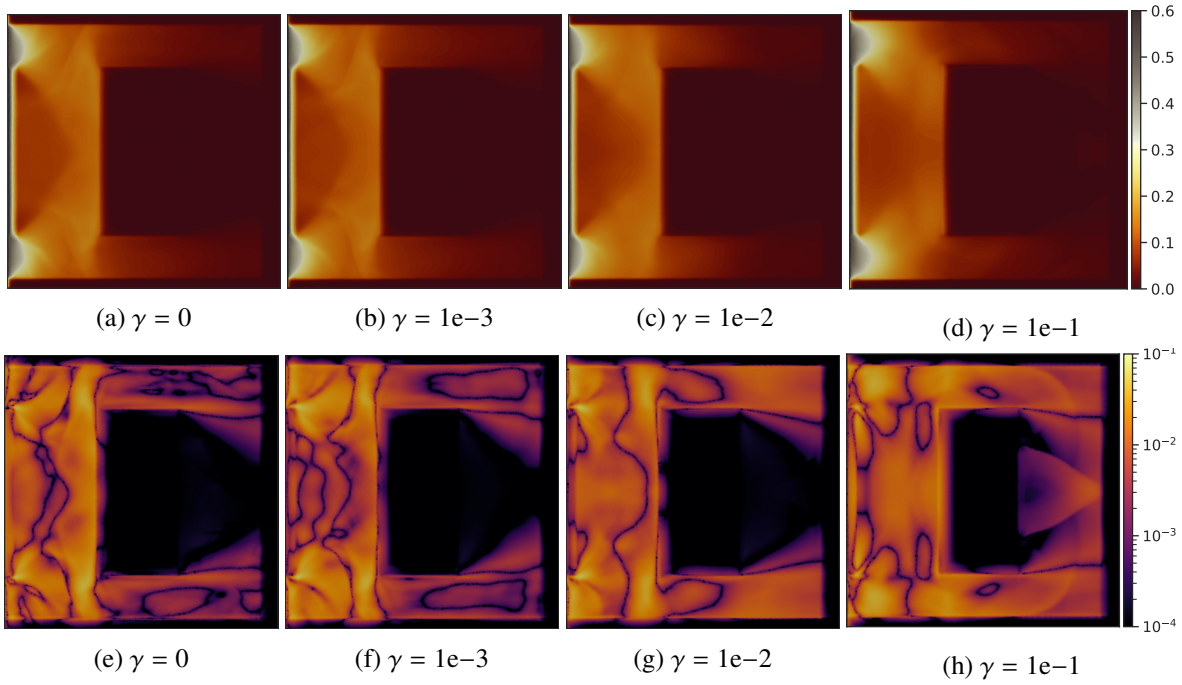


(a) $\gamma = 0$ (b) $\gamma = 1e{-}3$ (c) $\gamma = 1e{-}2$ (d) $\gamma = 1e{-}1$



(e) $\gamma = 0$ (f) $\gamma = 1e{-}3$ (g) $\gamma = 1e{-}2$ (h) $\gamma = 1e{-}1$

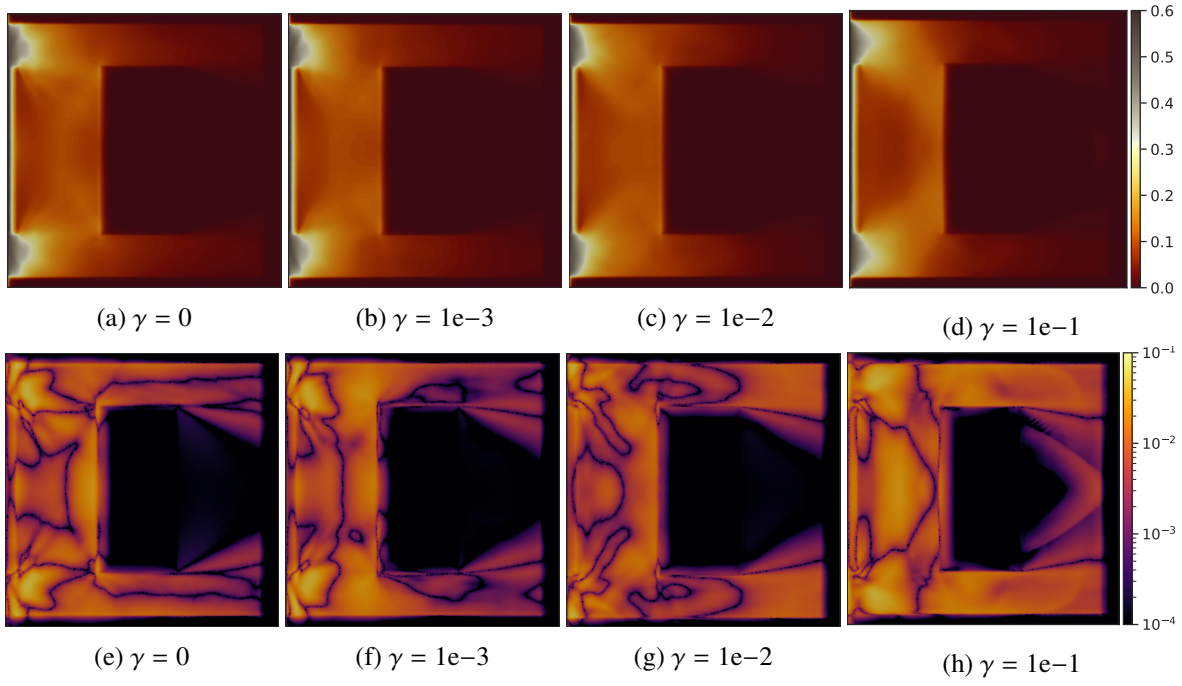
Figure 5.9.: Rotated, ICNN-based M_2 closures with different regularization (top row) and $e(t_f, \mathbf{x})$ (bottom row). The regularized model with $\gamma = 1e{-}3$ performs best.



(a) $\gamma = 0$ (b) $\gamma = 1e{-}3$ (c) $\gamma = 1e{-}2$ (d) $\gamma = 1e{-}1$

(e) $\gamma = 0$ (f) $\gamma = 1e{-}3$ (g) $\gamma = 1e{-}2$ (h) $\gamma = 1e{-}1$

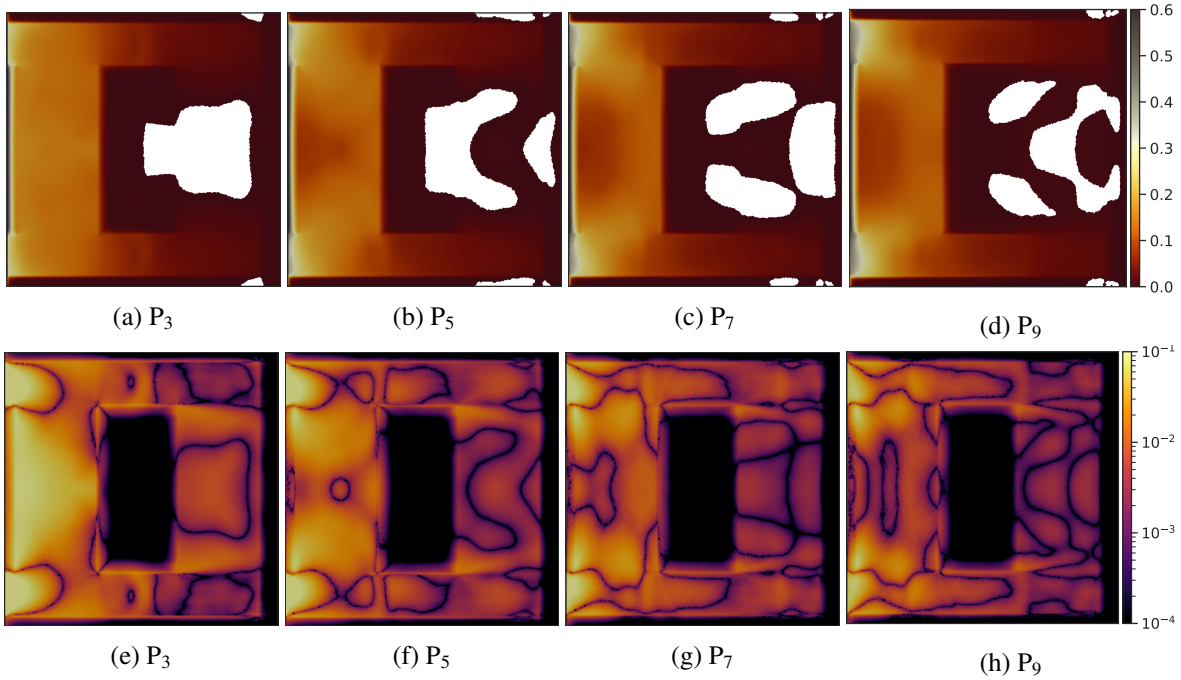
Figure 5.10.: ICNN-based M_3 closures with different regularization (top row) and $e(t_f, \mathbf{x})$ (bottom row). The $\gamma = 1e{-}3$ model has the smallest error. The heavily regularized model $\gamma = 1e{-}1$ displays artifacts in the absorption region on the right side of the domain.



(a) $\gamma = 0$ (b) $\gamma = 1e{-}3$ (c) $\gamma = 1e{-}2$ (d) $\gamma = 1e{-}1$

(e) $\gamma = 0$ (f) $\gamma = 1e{-}3$ (g) $\gamma = 1e{-}2$ (h) $\gamma = 1e{-}1$

Figure 5.11.: ICNN-based M_4 $e(t_f, \mathbf{x})$ (bottom row). The $\gamma = 1e{-}2$ model has the smallest error followed by the $\gamma = 1e{-}3$ model, where regularization and neural network approximation errors have the best trade-off.



(a) P_3 (b) P_5 (c) P_7 (d) P_9

(e) P_3 (f) P_5 (g) P_7 (h) P_9

Figure 5.12.: P_N closures (top row) and their absolute difference to the S_{50} solution (bottom row). The white areas in the solution plots denote unphysical negative solutions values. The P_N closure displays higher numerical errors compared to S_N and M_N simulations with similar degrees of freedom.

reference solutions, but are harder to train. Higher regularization increases training performance but introduces the regularization error if γ is significantly bigger than the grid resolution. Table 5.5 shows the best regularization level for each closure order and we see, that $\gamma \in (1e{-}3, 1e{-}2)$ yields the best results for $\Delta x = 7.5e{-}3$. The displayed errors are computed as

$$e_{\text{rel},u_0} = \frac{\int_{\mathbf{X}} \left\| u_0^{\text{ref}}(t_f, \mathbf{x}) - u_0(t_f, \mathbf{x}) \right\|_1 \, \mathrm{d}\mathbf{x}}{\int_{\mathbf{X}} \left\| u_0^{\text{ref}}(t_f, \mathbf{x})) \right\|_1 \, \mathrm{d}\mathbf{x}}, \tag{5.74}$$

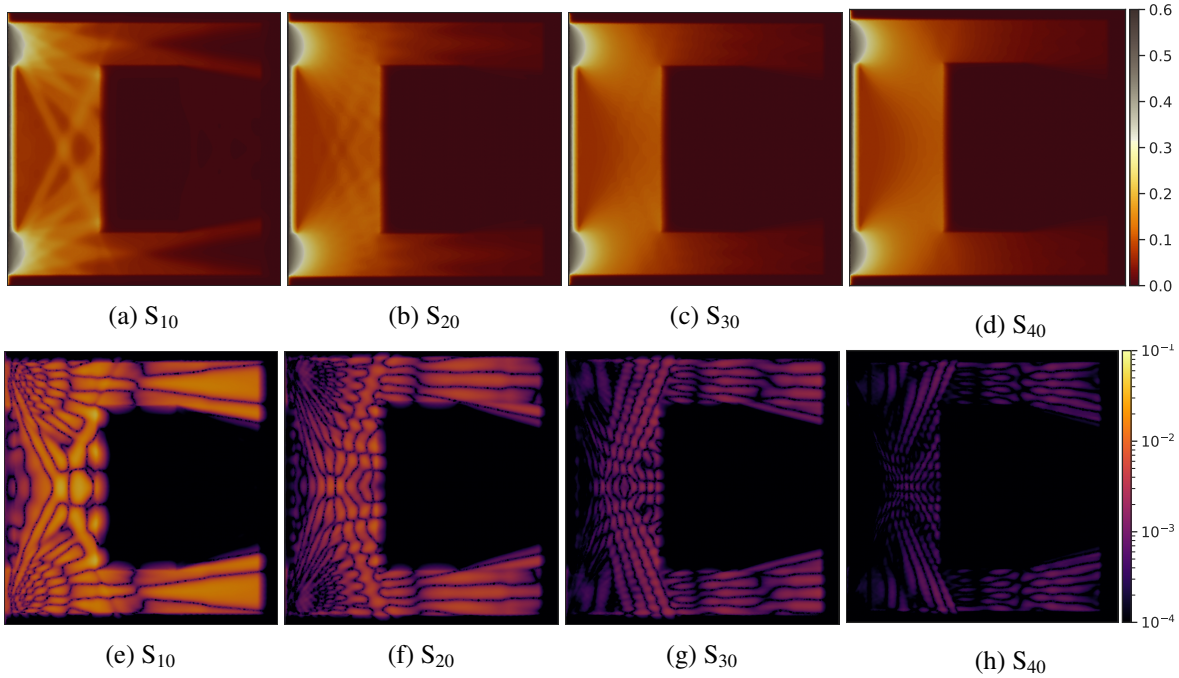
where $u_0^{\text{ref}}(t_f, \mathbf{x})$ is the order zero moment of the S_{50} solution. Note that the neural network validation results in Table 5.3 further support this regularization range.

Performance Comparison to the P_N and S_N Methods

Common other methods for radiative transport simulations are the P_N and S_N methods. The former is a moment method with a different closure. Instead of using the minimal entropy closure, a simple truncation closure is used, which yields a linear reconstruction of the kinetic density $f_{\mathbf{u}}$ from the moment basis,

$$f_{\mathbf{u}}(\mathbf{v}) = \mathbf{u} \cdot \mathbf{m}(\mathbf{v}). \tag{5.75}$$

The P_N closure can also be viewed as the special case of a quadratic entropy density $\eta(g) = g^2$. Although computationally highly efficient, the pitfalls of the P_N method are oscillations and negative solutions, which can be seen in Fig. 5.12. Furthermore, higher moment orders are required to ensure the same



(a) S_{10} (b) S_{20} (c) S_{30} (d) S_{40}

(e) S_{10} (f) S_{20} (g) S_{30} (h) S_{40}

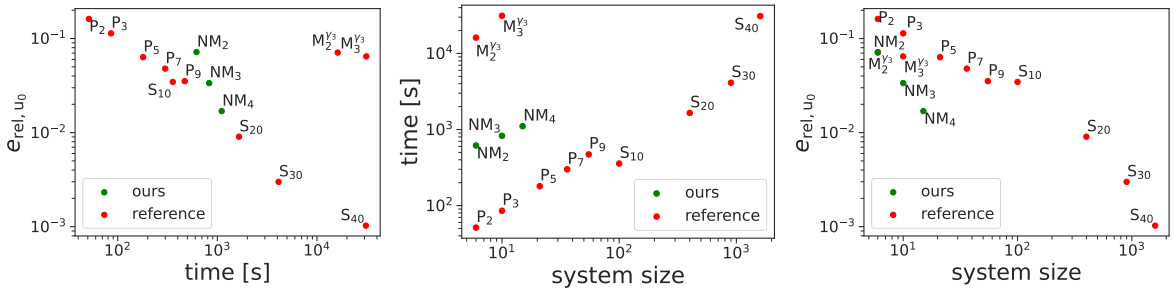
Figure 5.13.: S_N simulations (top row) and their absolute difference to the S_{50} solution (bottom row). The numerical artifacts are so-called ray effects, which can be mitigated by very high order S_N schemes or diffusive rotation methods [31].

Table 5.5.: Relative integrated errors e_{rel,u_0} of ICNN-based entropy closures corresponding to the S_{50} Hohlraum simulation. Particularly higher-order closures benefit from regularized surrogate models.

γ	M_2	M_2, rotated	M_3	M_4
0	**7.058e−2**	7.31e−2	3.60e−2	3.15e−2
1e−3	7.35e−2	**7.05e−2**	**3.34e−2**	2.06e−2
1e−2	7.18e−2	7.27e−2	3.86e−2	**1.69e−2**
1e−1	7.10e−2	7.73e−2	4.26e−2	3.12e−2

order of accuracy as M_N closures. Figure 5.12 shows, that especially in the shielded regions on the right-hand side of the computational domain the P_N method experience higher numerical errors than the M_N simulation.

The S_N method discretizes the velocity space directly using a quadrature rule. The resulting transport system is of size equal to the number of quadrature points and the system's equations are coupled only by the collision operator. Consequently, the S_N method has the lowest computational expense in comparison with P_N and M_N systems of similar size. However, the required quadrature order for a high-quality simulation is much higher than the moment order of P_N and M_N methods. Low-order S_N methods display numerical artifacts in the form of ray-effects [31], as seen in Fig. 5.13a, but very high-order S_N codes are typically used as reference models. The number of quadrature points scales quadratically with the quadrature order since the velocity domain is a sphere in $\mathbb{R}^d$, which results in a large system of equations for high-order S_N methods. On modern high-performance clusters, the performance bottleneck is typically the memory footprint of the simulation.



(a) e_{rel,u_0} vs sim. time (b) Sim. time vs system size (c) e_{rel,u_0} vs system size

Figure 5.14.: Comparison of memory footprint, simulation wall-time, and relative simulation error of P_N, S_N, M_N and ICNN-based M_N (denoted by NM_N) solutions. Less is better. The neural network-based M_N method has a particularly small memory footprint, paired with a competitive wall-time efficiency relative to the numerical error.

Thus a comparison between P_N, S_N, M_N, and neural network-based M_N methods needs to consider computational time and memory footprint besides the difference to the reference solution. To this end, we measure the wall-time of a Hohlraum simulation with the setup given by Table 5.4 for all previously discussed non-regularized, regularized, and neural network-based M_N, as well as the P_N and S_N methods. Each simulation is computed on the same, isolated hardware, using a 16-core CPU with 64 GB Memory, such that each test case fits entirely into the system's memory. Figure 5.14 compares the computational performance of all methods with different moment, respectively quadrature orders.
Note that the computation time and memory footprint of neural network-based M_N methods only depends on the size of the neural network, thus we only report the performance of one neural network-based simulation per moment order, i.e., the best neural network-based run of the reported regularization levels. The displayed errors are computed with Eq. (5.74). We denote the neural network-based methods by NM_N in the illustration. We see in Fig. 5.14a that the neural network-based entropy closure accelerates the M_N method to be computationally competitive, compared to the P_N and S_N methods. Note, that simulation time is a function of the code. The KiT-RT framework is constructed such that spatial and temporal discretizations use the same implementation across all macroscopic methods. However, remark that an M_N implementation leaves plenty of opportunities [141, 82] for advanced code optimization that may improve the methods timings.
Figure 5.14b compares the size of the transport system and computational time of the different methods showing that even for higher order closures, neural network-based M_N methods have the same simulation time as S_{20} simulations, whereas the memory footprint of the neural network-based M_N method is smaller by almost two orders of magnitude. Neural network-based entropy closure accelerates the Newton-based M_N method by more than an order of magnitude in terms of computational time, while keeping the memory footprint the same. Figure 5.14c shows the simulation error over the system size of the different methods. Here, the neural network-based M_3 and M_4 methods have the best trade-off between memory footprint and simulation accuracy.

5.7. Chapter Conclusion

In this chapter, we addressed the key challenges and limitations of the neural network-based entropy closure of the Boltzmann moment system.

5.7.1. Summary

We have presented a framework for regularized, structure-preserving neural network surrogate models for the minimal entropy closure of the Boltzmann moment system. Regularization addresses the challenge of training and inference of neural network surrogates near the boundary of the realizable set, and thus enables the creation of robust surrogates for high order closures in two and three spatial dimensions. We have provided an error analysis of regularized network approximation and put it into context of numerical errors of commonly used kinetic schemes for the Boltzmann equation. Rotational invariance and normalization of the partially regularized entropy closure is used to provide a dimension reduction technique and to improve simulation results. The presented methods are tested on a wide range of synthetic and simulation test cases, with applications to radiation transport. We show that rotationally invariant neural network surrogate models give good results even in the extreme Line-source test case. The neural network-based entropy closure leads to a computationally competitive simulation method with an advantageous trade-off between memory footprint and numerical error, compared to the spherical harmonics, the traditional minimal entropy and the nodal method.

5.7.2. Limitations of the Approach

Although we were able to transfer most structural properties of the Boltzmann equation to our surrogate model, invariance with respect to translations is lost due to the regularization of the entropy closure. A practical shortcoming is that the rotated ICNN approach needs to evaluate the entropy of each moment twice, thus introducing additional computational expense, compared to the non-rotated closure.

The proposed entropy closures are limited to the linear Boltzmann equation with bounded velocity space. The full Boltzmann Equation poses severe theoretical challenges for minimal entropy closures [125, 100] and the methods are note directly applicable to this situation.

5.7.3. Future Work

Neural network architectures, which are rotation invariant by design [69, 179, 44], can be considered as an alternative to the post-processing rotation proposed in this work. It needs to be investigated to which extent input-convex, rotationally invariant neural networks can be constructed. Entropy closures can be used for any truncated moment problem, i.e., it has a wide range of further application in statistics. Further, the developed tools, especially the error control can be applied to any other convex function approximation task.

5.8. Additional Material

Statements of §5.2

We provide proof for Theorem 5.1. As a reminder, a sequence of probability densities f_n converges in distribution to f^*,

$$\langle f_n(\mathbf{v})\phi(\mathbf{v})\rangle \xrightarrow{n\to\infty} \langle f^*(\mathbf{v})\phi(\mathbf{v})\rangle\,, \qquad \forall\phi \in C_b, \tag{5.76}$$

where C_b denotes the set of bounded and continuous functions. If f^* is a Dirac distribution δ, then the sequence f_n converging in distribution is called a Dirac sequence.

Proof: (Theorem 5.1) Consider the fact [187, 52], that $f_{\overline{\mathbf{u}}}$ for $\overline{\mathbf{u}} \in \partial\overline{\mathcal{R}}$ consists of a linear combination of Dirac distributions for $D = [0,\infty)$ and $\mathbf{V} = \mathbb{S}^2$. We show that the Maxwell Boltzmann entropy of a Dirac sequence f_n converging in distribution to $\delta_{\mathbf{v}^*}$ at any point $\mathbf{v}^* \in \mathbf{V}$ diverges to infinity. Consider

$$f_n(\mathbf{v}) = n\mathbb{1}_{B^{\mathbf{v}^*}_{\frac{1}{n}}}(\mathbf{v}), \qquad n \in \mathbb{N} \tag{5.77}$$

where $B^{\mathbf{v}^*}_{\frac{1}{n}} \subset \mathbf{V}$ is a subdomain of measure $\frac{1}{n}$ and $\mathbb{1}(\mathbf{v})$ is the indicator function. Then f_n is a Dirac sequence, since

$$\langle f_n(\mathbf{v})\rangle = 1, \quad \forall n \in \mathbb{N} \tag{5.78}$$

and

$$\langle f_n(\mathbf{v})\phi(\mathbf{v})\rangle \xrightarrow{n\to\infty} \phi(\mathbf{v}=\mathbf{v}^*) = \langle \delta_{\mathbf{v}^*}(\mathbf{v})\phi(\mathbf{v})\rangle \qquad \forall\phi \in C_b \tag{5.79}$$

For such a Dirac sequence, we investigate the limit

$$\lim_{n\to\infty} \langle f_n(\mathbf{v})\log f_n(\mathbf{v}) - f_n(\mathbf{v})\rangle\,. \tag{5.80}$$

Evaluating the Maxwell-Boltzmann entropy for f_n using the transformation theorem yields

$$\begin{aligned}\langle f_n(\mathbf{v})\log f_n(\mathbf{v}) - f_n(\mathbf{v})\rangle &= \int_{\mathbb{S}^2} n\mathbb{1}_{B^{\mathbf{v}^*}_{\frac{1}{n}}}(\mathbf{v})\log n\mathbb{1}_{B^{\mathbf{v}^*}_{\frac{1}{n}}}(\mathbf{v}) - n\mathbb{1}_{B^{\mathbf{v}^*}_{\frac{1}{n}}}(\mathbf{v})\,\mathrm{d}\mathbf{v} \\ &= \int_{B^{\mathbf{v}^*}_{\frac{1}{n}}} n\log n\,\mathrm{d}\mathbf{v} - n\int_{B^{\mathbf{v}^*}_{\frac{1}{n}}} 1\,\mathrm{d}\mathbf{v} \\ &= \log n - 1.\end{aligned} \tag{5.81}$$

This term diverges to infinity as $n \to \infty$, which concludes the proof. □

Note, that the result depends on the choice of the velocity space $\mathbf{V}$, the domain of the entropy density D, and the choice of the entropy density η itself.
In the case of the normalized 1D M_1 closure, we can specify the behavior of the entropy functional h. Let $\boldsymbol{\alpha} = [\vartheta(\beta),\beta]^\top$, where we use Eq. (5.19). Consider $-\phi(\boldsymbol{\alpha};\boldsymbol{\psi}(\beta))$ using Eq. (5.58), for $|\beta| \to \infty$, which is equivalent to $\overline{\mathbf{u}} \to \partial\overline{\mathcal{R}}$. Here, we have $V = [-1,1]$ and $\boldsymbol{m}(v) = [1, v]$. From Eq. (5.19) and Eq. (1.62) we have, that the kinetic density $f_{\mathbf{u}}$ with minimal entropy for the M_1 1D closure takes the form

$$f_\beta(v) = \frac{e^{\beta v}}{\langle e^{\beta v}\rangle}. \tag{5.82}$$

Thus, f_β converges in distribution to $\delta_{\pm 1}$ for $\beta \to \pm\infty$. Evaluating the entropy functional yields

$$\begin{aligned}\left\langle \eta(f_\beta) \right\rangle =& \left\langle \frac{e^{\beta v}}{\langle e^{\beta v} \rangle} \log\left(\frac{e^{\beta v}}{\langle e^{\beta v} \rangle} \right) - \frac{e^{\beta v}}{\langle e^{\beta v} \rangle} \right\rangle \\ =& \beta \frac{e^{\beta} + e^{-\beta}}{e^{\beta} - e^{-\beta}} + \log\left(\frac{\beta}{e^{\beta} - e^{-\beta}} \right) - 2\end{aligned} \tag{5.83}$$

Consider $\beta > 0$,

$$\left\langle \eta(f_\beta) \right\rangle = \beta \frac{e^{\beta} + e^{-\beta}}{e^{\beta} - e^{-\beta}} + \log(\beta) - \log\left(e^{\beta} - e^{-\beta} \right) - 2 \tag{5.84}$$

and for $\beta < 0$ we get by inserting -1 in enumerator and denominator

$$\left\langle \eta(f_\beta) \right\rangle = \beta \frac{e^{\beta} + e^{-\beta}}{e^{\beta} - e^{-\beta}} + \log(-\beta) - \log\left(-(e^{\beta} - e^{-\beta}) \right) - 2 \tag{5.85}$$

For large $|\beta|$, all terms except $\log(|\beta|) - 2$ vanish, thus h diverges as $\log(|\beta|) - 2$.

Statements of §5.3

We provide proof for Lemma 5.2, which is a tool for the elimination of the first degree of freedom of the regularized entropy closure.

Proof: (Lemma 5.2) Under the assumption that $m_0(\mathbf{v}) = 1$, the first order optimality condition of the partially regularized dual problem (5.17) leads to

$$\nabla_{\boldsymbol{\alpha}} \phi^\gamma \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}; \overline{\mathbf{u}} \right) = \overline{\mathbf{u}} - \left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}} \cdot \mathbf{m} \right) \right\rangle + \gamma \left[0, \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}} \right)_\#^\top \right]^\top = 0 \,. \tag{5.86}$$

Inspecting the first element of $\overline{\mathbf{u}}$ gives $\left\langle \exp\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}} \cdot \mathbf{m} \right) \right\rangle = \overline{u}_0 (= 1)$. Thus,

$$1 = \left\langle \exp\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}} \cdot \mathbf{m} \right) \right\rangle = \exp\left(\alpha^\gamma_{\overline{\mathbf{u}},0} \right) \left\langle \exp\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}} \right)_\# \cdot \mathbf{m}_\# \right) \right\rangle \,, \tag{5.87}$$

which, together with the definition of ϑ given in Eq. (5.20), proves the claim. □

We provide proof for Theorem 5.3, which gives the justification to work only on the reduced realizable set of the regularized minimal entropy closure with the Maxwell-Boltzmann entropy.

Proof: (Theorem 5.3) We structure the proof to show the asserted statements one by one.
1. The first claim that

$$\hat{\phi}^\gamma(\boldsymbol{\beta}; \mathbf{w}) = -1 - \log\left(\langle \exp(\boldsymbol{\beta} \cdot \mathbf{m}_\#) \rangle \right) + \boldsymbol{\beta} \cdot \mathbf{w} - \frac{\gamma}{2} \|\boldsymbol{\beta}\|^2 \tag{5.88}$$

follows directly from the definitions of $\hat{\phi}^\gamma$, ϕ^γ, and ϑ given in Eqs. (5.21), (5.18), and (5.20), respectively.
2. It is clear that $\hat{\phi}^\gamma$ is twice differentiable w.r.t. $\boldsymbol{\beta}$. To show that $\hat{\phi}^\gamma$ is concave, we show that $-\hat{\phi}^\gamma$ is convex. Since sums of convex functions are convex, we prove the claim by showing that $\log\left(\langle \exp(\boldsymbol{\beta} \cdot \mathbf{m}_\#) \rangle \right)$ satisfies Jensen's inequality, thus is convex. We prove Jensen's inequality by using the monotonicity of

the logarithm and Hölder's inequality with $1/p = t$ and $1/q = (1-t)$. Specifically, for $t \in (0,1)$ and $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^{n-1}$, we have

$$\begin{aligned}\log(\langle\exp((t\boldsymbol{\beta}_1 + (1-t)\boldsymbol{\beta}_2)\cdot\mathbf{m}_\#)\rangle) &= \log(\langle\exp(t\boldsymbol{\beta}_1\cdot\mathbf{m}_\#)\exp((1-t)\boldsymbol{\beta}_2\cdot\mathbf{m}_\#)\rangle)\\ &\leq \log(\langle\exp(\boldsymbol{\beta}_1\cdot\mathbf{m}_\#)\rangle^t\langle\exp(\boldsymbol{\beta}_2\cdot\mathbf{m}_\#)\rangle^{(1-t)}) \qquad \text{(Hölder's ineq.)}\\ &= \log(\langle\exp(\boldsymbol{\beta}_1\cdot\mathbf{m}_\#)\rangle^t) + \log(\langle\exp(\boldsymbol{\beta}_2\cdot\mathbf{m}_\#)\rangle^{(1-t)})\\ &= t\log(\langle\exp(\boldsymbol{\beta}_1\cdot\mathbf{m}_\#)\rangle) + (1-t)\log(\langle\exp(\boldsymbol{\beta}_2\cdot\mathbf{m}_\#)\rangle).\end{aligned}$$

Further, for $\gamma > 0$, $\hat{\phi}^\gamma$ is strictly concave since $-\frac{\gamma}{2}\|\boldsymbol{\beta}\|^2$ is strictly concave.

3. From the proof for 2., it is clear that $-\hat{\phi}^\gamma$ is still convex when $\gamma = 0$. Hence $\hat{H}_n^{\gamma=0}(\boldsymbol{\beta})$ is symmetric and positive semidefinite and has non-negative eigenvalues. By definition, $\hat{H}_n^\gamma(\boldsymbol{\beta})$ can be written as

$$\hat{H}_n^\gamma(\boldsymbol{\beta}) = \hat{H}_n^{\gamma=0}(\boldsymbol{\beta}) + \gamma I, \tag{5.89}$$

which gives the eigenvalue relations in Eq. (5.23). The bound on the condition number is then a direct consequence of the non-negativity of eigenvalues of $\hat{H}_n^{\gamma=0}(\boldsymbol{\beta})$.

4. Since $\hat{\phi}^\gamma$ is strictly concave, there exists a unique maximizer $\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}$ that satisfies the first order optimality condition, i.e.,

$$\begin{aligned}0 = \nabla_{\boldsymbol{\beta}}\hat{\phi}^\gamma(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#};\overline{\mathbf{u}}_\#) &= \overline{\mathbf{u}}_\# - \frac{1}{\left\langle\exp(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}\cdot\mathbf{m}_\#)\right\rangle}\left\langle\mathbf{m}_\#\exp(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}\cdot\mathbf{m}_\#)\right\rangle - \gamma\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}\\ &= \overline{\mathbf{u}}_\# - \left\langle\mathbf{m}_\#\exp(\left[\vartheta\left(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}\right),\left(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}\right)^\top\right]^\top\cdot\mathbf{m})\right\rangle - \gamma\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}.\end{aligned} \tag{5.90}$$

On the other hand, the first order optimality condition of (5.17) is given by

$$0 = \nabla_{\boldsymbol{\alpha}}\phi^\gamma(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}};\overline{\mathbf{u}}) = \overline{\mathbf{u}} - \left\langle\mathbf{m}\exp(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\cdot\mathbf{m})\right\rangle - \gamma\left[0,\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top. \tag{5.91}$$

From Eq. (5.90) and Lemma 5.2, it is straightforward to verify that $[\vartheta(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}),(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#})^\top]^\top$ satisfies the optimality condition in Eq. (5.91). The equivalence of $\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}$ and $[\vartheta(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#}),(\boldsymbol{\beta}^\gamma_{\overline{\mathbf{u}}_\#})^\top]^\top$ then follows directly from the strict concavity of ϕ^γ, and the equivalence of objective function values is a direct consequence of the definition of $\hat{\phi}^\gamma$ in Eq. (5.21).

Strict concavity of ϕ^γ: The Hessian of ϕ^γ is given by

$$H^\gamma := \langle\mathbf{m}\otimes\mathbf{m}\exp(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle + \gamma\begin{bmatrix}0 & 0\\ 0 & I\end{bmatrix}. \tag{5.92}$$

We show that when $\gamma > 0$, $\boldsymbol{\alpha}^\top H^\gamma\boldsymbol{\alpha} > 0$ for all $\boldsymbol{\alpha} \neq 0$. Since $\boldsymbol{\alpha}^\top H^\gamma\boldsymbol{\alpha} = \left\langle(\mathbf{m}\cdot\boldsymbol{\alpha})^2\exp(\boldsymbol{\alpha}\cdot\mathbf{m})\right\rangle + \gamma\|\alpha_\#\|^2$ and $\left\langle(\mathbf{m}\cdot\boldsymbol{\alpha})^2\exp(\boldsymbol{\alpha}\cdot\mathbf{m})\right\rangle \geq 0$, it is clear that $\boldsymbol{\alpha}^\top H^\gamma\boldsymbol{\alpha} > 0$ when $\boldsymbol{\alpha}_\# \neq 0$. Now let $\alpha_\# = 0$, then $\boldsymbol{\alpha}^\top H^\gamma\boldsymbol{\alpha} = \alpha_0^2\langle\exp(\alpha_0)\rangle > 0$ for any $\alpha_0 \neq 0$, which concludes the proof. □

We provide proof for Lemma 5.5, which yields a scaling formula to recover the non-normalized closure from the Lagrange multiplier of the normalized closure.

Proof: (Lemma 5.5) We structure the proof into three parts.

1. Let us define the reduced multiplier to moment map $\chi : \mathbb{R}^{n-1} \to \mathbb{R}^n$ as

$$\boldsymbol{\chi}^\gamma(\boldsymbol{\beta}) = \left\langle\mathbf{m}\exp\left(\left[\vartheta(\boldsymbol{\beta}),\boldsymbol{\beta}^\top\right]^\top\cdot\mathbf{m}\right)\right\rangle + \gamma\left[0,\boldsymbol{\beta}^\top\right]^\top. \tag{5.93}$$

We make use of the first order necessary condition of $\hat{\phi}^\gamma(\boldsymbol{\beta}; \mathbf{w})$ at its optimal point $(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}},\mathbf{D}})_\#$, i.e., Eq. (5.26) and get

$$\overline{\mathbf{u}} = \boldsymbol{\chi}^\gamma\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#\right) \tag{5.94}$$

Multiplication of both sides with u_0 yields

$$\begin{aligned}\mathbf{u} = u_0\overline{\mathbf{u}} &= u_0\boldsymbol{\chi}^\gamma\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#\right)\\ &= \left\langle \mathbf{m}u_0 \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#\right), \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top \cdot \mathbf{m}\right)\right\rangle + \gamma u_0\left[0, \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top\\ &= \left\langle \mathbf{m}\exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#\right) + \log(u_0), \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top \cdot \mathbf{m}\right)\right\rangle + \gamma u_0\left[0, \left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#^\top\right]^\top,\end{aligned} \tag{5.95}$$

which shows the assertion.
2. Expressing h^γ, see Eq. (5.18), in terms of $\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}$ then leads to

$$\begin{aligned}h^\gamma(\mathbf{u}) &= u_0\log u_0 + u_0\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\cdot\overline{\mathbf{u}} - \left\langle\exp\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\cdot\mathbf{m}\right)\right\rangle\exp(\log u_0) - \frac{u_0\gamma}{2}\left\|\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_\#\right\|_2^2\\ &= u_0\hat{h}^\gamma(\overline{\mathbf{u}}_\#) + u_0\log u_0\,.\end{aligned} \tag{5.96}$$

3. We show convexity of $h^\gamma(\mathbf{u})$ in $\mathbf{u}$. We write the entropy functional of the statically, partially regularized problem (5.11) as

$$h^\gamma_{orig}(\mathbf{u}) = \boldsymbol{\alpha}^\gamma_{\mathbf{u}}\cdot\mathbf{u} - \langle\exp(\boldsymbol{\alpha}_{\mathbf{u}}m)\rangle - \frac{\gamma}{2}\left\|\boldsymbol{\alpha}^\gamma_{\mathbf{u},\#}\right\|^2. \tag{5.97}$$

We show first the convexity of h^γ_{orig}. To show convexity of h^γ_{orig}, we notice

$$h^\gamma_{orig}(\mathbf{u}) = h_{orig}(\mathbf{u}) - \frac{\gamma}{2}\left\|\boldsymbol{\alpha}^\gamma_{\mathbf{u},\#}\right\|^2, \tag{5.98}$$

where $h_{orig}(\mathbf{u}) := h^{\gamma=0}_{orig}(\mathbf{u})$, i.e., the expression equals the non-regularized entropy functional. Thus by definition of the Legendre dual, we have:

$$h^\gamma_{orig,*}(\boldsymbol{\alpha}) = h_{orig,*}(\boldsymbol{\alpha}) + \frac{\gamma}{2}\|\boldsymbol{\alpha}_\#\|^2 = \langle\eta_*(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle + \frac{\gamma}{2}\|\boldsymbol{\alpha}_\#\|^2 = \langle\exp(\boldsymbol{\alpha}\cdot\mathbf{m})\rangle + \frac{\gamma}{2}\|\boldsymbol{\alpha}_\#\|^2 \tag{5.99}$$

where the last equality comes from using the Maxwell-Boltzmann entropy. The right hand side function is clearly convex in $\boldsymbol{\alpha}$, and by Legendre duality, we conclude convexity of $h^\gamma_{orig}(\mathbf{u})$.
Then, we conclude convexity of $\hat{h}^\gamma_{orig}(\overline{\mathbf{u}}_\#) = {h_{orig}}^\gamma(\overline{\mathbf{u}})$. Afterward, we notice that

$$\hat{h}^\gamma_{orig}(\overline{\mathbf{u}}_\#) = \hat{h}^\gamma(\overline{\mathbf{u}}_\#) \tag{5.100}$$

for normalized moments $\overline{\mathbf{u}}$, thus concluding convexity of the latter. Finally, convexity of $h^\gamma(\mathbf{u})$ is obtained by convexity of $\hat{h}^\gamma(\overline{\mathbf{u}}_\#)$ and [200, Theorem 3.1].

□

The validity of statement **1** of Lemma 5.5 is a feature of the dynamic regularization of the optimization problem 5.24. From [208, 200, 8], it is known that for the non-regularized problem, we have the Lagrange multiplier scaling relationships

$$\boldsymbol{\alpha}_{\mathbf{u}} = \left[\vartheta\left((\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#\right) + \log(u_0), (\boldsymbol{\alpha}_{\overline{\mathbf{u}}})_\#^\top\right]^\top, \tag{5.101}$$

holds.
However, in the partially regularized closure with *static* regularization, the linear and exponential influence of the Lagrange multiplier in the reconstruction would prevent an equivalent exact scaling formula for $\boldsymbol{\alpha}_{\mathbf{u}}^{\gamma}$ with $\gamma > 0$, i.e.

$$\begin{aligned}\left\langle \mathbf{m} \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma}\right)_{\#}\right) + \log(u_0), \left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma}\right)_{\#}^{\top}\right]^{\top} \cdot \mathbf{m}\right)\right\rangle + \gamma\left[0, \left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma}\right)_{\#}^{\top}\right]^{\top} \\ = u_0\left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma} \cdot \mathbf{m}\right)\right\rangle + \gamma\left[0, \left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma}\right)_{\#}^{\top}\right]^{\top} \\ \neq u_0\left(\left\langle \mathbf{m} \exp\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma} \cdot \mathbf{m}\right)\right\rangle + \gamma\left[0, \left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma}\right)_{\#}^{\top}\right]^{\top}\right) \\ = u_0\overline{\mathbf{u}} = \mathbf{u}.\end{aligned} \tag{5.102}$$

We provide proof for Lemma 5.6

Proof: (Lemma 5.6) Consider the first-order necessary condition of the dual, partially regularized problem (5.17), i.e.,

$$\nabla_{\alpha}\phi^{\gamma}(\boldsymbol{\alpha}, \mathbf{u}) = \mathbf{u} - \langle \mathbf{m} \exp(\boldsymbol{\alpha} \cdot \mathbf{m})\rangle - \gamma u_0\left[0, \boldsymbol{\alpha}_{\#}^{\top}\right]^{\top}, \tag{5.103}$$

which equals zero at the optimal point $\alpha_{\mathbf{u}}^{\gamma}$. Then, the total derivative of the entropy with respect to the moments $\mathbf{u}$ is given by

$$\begin{aligned}\nabla_{\mathbf{u}} h^{\gamma}(\mathbf{u}) &= \boldsymbol{\alpha}_{\mathbf{u}}^{\gamma} + \nabla_{\mathbf{u}}\alpha_{\mathbf{u}}^{\gamma}\mathbf{u} - \nabla_{\mathbf{u}}\alpha_{\mathbf{u}}^{\gamma}\left\langle \mathbf{m} \exp(\boldsymbol{\alpha}_{\mathbf{u}}^{\gamma} \cdot \mathbf{m})\right\rangle - \gamma u_0 \nabla_{\mathbf{u}}\alpha_{\mathbf{u}}^{\gamma}[0, \boldsymbol{\alpha}_{\mathbf{u},\#}^{\gamma,\top}]^{\top} - \frac{\gamma}{2}\left[\left\|\boldsymbol{\alpha}_{\mathbf{u}\#}^{\gamma}\right\|^2, 0\right]^{\top} \\ &= \boldsymbol{\alpha}_{\mathbf{u}}^{\gamma} - \frac{\gamma}{2}\left[\left\|\boldsymbol{\alpha}_{\mathbf{u}\#}^{\gamma}\right\|^2, 0\right]^{\top} + \nabla_{\mathbf{u}}\alpha_{\mathbf{u}}^{\gamma}\left(\mathbf{u} - \left\langle \mathbf{m} \exp(\boldsymbol{\alpha}_{\mathbf{u}}^{\gamma} \cdot \mathbf{m})\right\rangle - \gamma u_0[0, \boldsymbol{\alpha}_{\mathbf{u},\#}^{\gamma,\top}]^{\top}\right) \\ &= \boldsymbol{\alpha}_{\mathbf{u}}^{\gamma} - \left[\frac{\gamma}{2}\left\|\boldsymbol{\alpha}_{\mathbf{u},\#}^{\gamma}\right\|^2, 0\right]^{\top} = \alpha_{\mathbf{u},\mathbf{D}}^{\gamma}\end{aligned} \tag{5.104}$$

□

We provide proof for Theorem 5.7

Proof: (Theorem 5.7)We structure the proof into two parts.
1. We show entropy dissipation. Consider the flux function of the moment system,

$$\mathbf{F}(\mathbf{u}) = \left\langle \mathbf{v} \otimes \mathbf{m} f_{\mathbf{u},\mathbf{D}}^{\gamma}\right\rangle \tag{5.105}$$

with the ansatz function (5.34). By Lemma 5.5, $h^{\gamma}(\mathbf{u})$ is convex and thus a suitable entropy candidate. We show the integrability condition for the entropy/entropy-flux pair $h^{\gamma}(\mathbf{u})$ and $\mathbf{j}_{\mathbf{D}}^{\gamma}(\mathbf{u})$ for each element of $j_{\mathbf{D},i}^{\gamma}$, i.e.,

$$\begin{aligned}\nabla_{\mathbf{u}} j_{\mathbf{D},i}^{\gamma}(\mathbf{u}) &= \nabla_{\mathbf{u}}\left\langle v_i \eta\left(f_{\mathbf{u},\mathbf{D}}^{\gamma}\right)\right\rangle \\ &= \left\langle \eta'\left(\eta_*'\left(\boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{\gamma} \cdot \mathbf{m}\right)\right) v_i \mathbf{m} \eta_*''\left(\boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{\gamma} \cdot \mathbf{m}\right) \nabla_{\mathbf{u}}\boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{\gamma}\right\rangle \\ &= \left\langle \boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{\gamma} v_i \mathbf{m} \otimes \mathbf{m} \eta_*''\left(\boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{\gamma} \cdot \mathbf{m}\right) \nabla_{\mathbf{u},\mathbf{D}}\boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{\gamma}\right\rangle \qquad \text{(Legendre duality)} \\ &= \nabla_{\mathbf{u}} h^{\gamma}(\mathbf{u})\left\langle v_i \mathbf{m} \otimes \mathbf{m} \eta_*''\left(\boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{\gamma} \cdot \mathbf{m}\right) \nabla_{\mathbf{u}}\boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{\gamma}\right\rangle \\ &= \nabla_{\mathbf{u}} h^{\gamma}(\mathbf{u}) \nabla_{\mathbf{u}} F_i(\mathbf{u}),\end{aligned} \tag{5.106}$$

where we use Lemma 5.6, the ansatz (5.34), and the definition of the Maxwell-Boltzmann entropy η. We multiply the moment system with $\nabla_{\mathbf{u}} h^\gamma(\mathbf{u})$, i.e.,

$$\begin{aligned}
\nabla_{\mathbf{u}} h^\gamma(\mathbf{u})\partial_t \mathbf{u}(t,\mathbf{x}) + \nabla_{\mathbf{u}} h^\gamma(\mathbf{u})\nabla_{\mathbf{x}} \boldsymbol{F}(\mathbf{u}) &= \nabla_{\mathbf{u}} h^\gamma(\mathbf{u}) \left\langle \mathbf{m}(\mathbf{v}) Q\left(f^\gamma_{\mathbf{u},\mathbf{D}}\right)\right\rangle \\
&= \boldsymbol{\alpha}^\gamma_{\mathbf{u},\mathbf{D}} \left\langle \mathbf{m}(\mathbf{v}) Q\left(f^\gamma_{\mathbf{u},\mathbf{D}}\right)\right\rangle \\
&= \left\langle \boldsymbol{\alpha}^\gamma_{\mathbf{u},\mathbf{D}} \mathbf{m}(\mathbf{v}) Q\left(f^\gamma_{\mathbf{u},\mathbf{D}}\right)\right\rangle \\
&= \left\langle \eta'\left(\eta'_*\left(\boldsymbol{\alpha}^\gamma_{\mathbf{u},\mathbf{D}}\cdot \boldsymbol{m}\right)\right) Q\left(\eta'_*\left(\boldsymbol{\alpha}^\gamma_{\mathbf{u},\mathbf{D}}\cdot \boldsymbol{m}\right)\right)\right\rangle \quad \text{(Legendre duality)} \\
&\leq 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(H-Theorem + ansatz)}
\end{aligned}$$

and obtain entropy dissipation.
2. We show the hyperbolicity of the system. Note, that hyperbolicity of the moment system follows from the entropy dissipation property. Consider again the flux function of the moment system (5.105)

$$\mathbf{F}(\mathbf{u}) = \left\langle \mathbf{v}\otimes \mathbf{m} f^\gamma_{\mathbf{u},\mathbf{D}}\right\rangle = \left\langle \mathbf{v}\otimes \mathbf{m}\eta'_*\left(\boldsymbol{\alpha}^\gamma_{\mathbf{u},\mathbf{D}}\cdot \mathbf{m}\right)\right\rangle \tag{5.107}$$

We define the function $j_{i,*}(\boldsymbol{\alpha}) = \langle v_i \eta_*(\boldsymbol{\alpha}\cdot \boldsymbol{m})\rangle$ and notice, that $\nabla_{\boldsymbol{\alpha}} j_{i,*}\left(\boldsymbol{\alpha}^\gamma_{\mathbf{u},\mathbf{D}}\right) = F_i(\mathbf{u})$. Let us denote

$$J_i = \Delta_{\boldsymbol{\alpha}} j_{i,*}(\boldsymbol{\alpha}^\gamma_{\mathbf{u},\mathbf{D}}). \tag{5.108}$$

Furthermore, we consider the Hessian of $h^\gamma(\mathbf{u})$, i.e.

$$K = \Delta_{\mathbf{u}} h^\gamma(\mathbf{u}) = \nabla_{\mathbf{u}} \boldsymbol{\alpha}^\gamma_{\mathbf{u},\mathbf{D}}, \tag{5.109}$$

which is symmetric positive definite by the convexity of $h^\gamma(\mathbf{u})$. Now we rewrite Eq. (5.38)

$$\begin{aligned}
&\partial_t \mathbf{u} + \nabla_{\mathbf{x}} \left\langle \mathbf{v}\otimes \mathbf{m} f^\gamma_{\mathbf{u},\mathbf{D}}\right\rangle = \\
&\partial_t \mathbf{u} + \sum_{i=1}^d J_i K \partial_{x_i}\mathbf{u} = \left\langle \mathbf{m} Q\left(f^\gamma_{\mathbf{u},\mathbf{D}}\right)\right\rangle.
\end{aligned} \tag{5.110}$$

It is left to show that $J_i K$ is diagonalizable with real eigenvalues. Then

$$\sum_{i=1}^d a_i J_i K, \qquad i = 1,\dots,d \tag{5.111}$$

is diagonalizable with real eigenvalues and the system is hyperbolic.
Since K is symmetric and positive definite, we can write $K^{1/2}K^{1/2} = K$ and we have

$$JK = K^{-1/2}K^{1/2}JK^{1/2}K^{1/2}. \tag{5.112}$$

Consequently, $K^{1/2}JK^{1/2}$ is similar to JK, i.e., it has the same eigenvalues. Since $K^{1/2}JK^{-1/2}$ is symmetric, JK is diagonalizable with real eigenvalues, which yields hyperbolicity of Eq. (5.38). □

We provide proof for Lemma 5.9.

Proof: (Lemma 5.9)
It has been shown that the entropy functional of the regularized and non-regularized entropy closure is

invariant under rotations [126, 7], i.e., $h(R\mathbf{u}) = h(\mathbf{u})$ and $h^\gamma(R\mathbf{u}) = h^\gamma(\mathbf{u})$. We consider the regularized entropy functional of the rotated moment, which is given by

$$h^\gamma(R\mathbf{u}) = \boldsymbol{\alpha}^\gamma_{R\mathbf{u}} \cdot R\mathbf{u} - \left\langle \eta_*'\left(\boldsymbol{\alpha}^\gamma_{R\mathbf{u}} \cdot R\mathbf{m}\right)\right\rangle - \frac{\gamma}{2}\boldsymbol{\alpha}^\gamma_{R\mathbf{u}} \cdot \boldsymbol{\alpha}^\gamma_{R\mathbf{u}}, \tag{5.113}$$

where $\boldsymbol{\alpha}_{R\mathbf{u}}$ is the Lagrange multiplier of $R\mathbf{u}$. It's straightforward to verify that the choice $\boldsymbol{\alpha}_{R\mathbf{u}} = R\boldsymbol{\alpha}_{\mathbf{u}}$ leads to $h^\gamma(R\mathbf{u}) = h^\gamma(\mathbf{u})$. Strict convexity of the entropy functional and the bijection $\mathbf{u} \mapsto \boldsymbol{\alpha}_{\mathbf{u}}$ ensure the uniqueness of this choice. The non-regularized case follows immediately for $\gamma = 0$. □

Note that intuitively, for $D = [0, \infty)$ and $\mathbf{V} = \mathbb{S}^2$, in the partially regularized closure, all moments with $u_0 \geq 0$ are feasible. The order zero moment u_0 is not affected by rotations, i.e., the corresponding entry of the rotation matrix is 1. Thus rotation does not affect the feasibility of a moment.
We provide proof for Theorem 5.11

Proof: (Theorem 5.11) Assume a rotation-invariant approximation to the entropy closure with approximated Lagrange multiplier $\boldsymbol{\alpha}^p_{\mathbf{u}}$. Using Eq. (5.95) of the proof of Lemma 5.5 we rewrite the moment $\mathbf{u}$ as

$$\begin{aligned}\mathbf{u} =& \left\langle \mathbf{m} \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}}\right)_{\#}\right) + \log(u_0), \left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}}\right)_{\#}^\top\right]^\top \cdot \mathbf{m}\right)\right\rangle + \gamma u_0 \left[0, \left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}}\right)_{\#}^\top\right]^\top \\ =& \mathbf{u}^\gamma + \gamma u_0 \left[0, \left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}}\right)_{\#}^\top\right]^\top,\end{aligned} \tag{5.114}$$

where the moment with regularization error is defined as

$$\mathbf{u}^\gamma = \left\langle \mathbf{m} \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}}\right)_{\#}\right) + \log(u_0), (\boldsymbol{\alpha}^\gamma_{\overline{\mathbf{u}}})_{\#}^\top\right]^\top \cdot \mathbf{m}\right)\right\rangle \tag{5.115}$$

and have $f^p_{\mathbf{u}} = u_0 f^p_{\overline{\mathbf{u}}}$. Consider the rotated moment system

$$\begin{aligned}0 &= R\left(\partial_t \mathbf{u} + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m} f^p_{\mathbf{u},\mathbf{D}}\right\rangle - \left\langle \mathbf{m} Q\left(f^p_{\mathbf{u},\mathbf{D}}\right)\right\rangle\right) \\ &= R\left(\partial_t \mathbf{u}^\gamma + \partial_t \gamma u_0 \left[0, \left(\boldsymbol{\alpha}^p_{\overline{\mathbf{u}}}\right)_{\#}^\top\right]^\top + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m} u_0 f^p_{\overline{\mathbf{u}},\mathbf{D}}\right\rangle - \left\langle \mathbf{m} Q\left(u_0 f^p_{\overline{\mathbf{u}},\mathbf{D}}\right)\right\rangle\right) && \text{(Eq. (5.114))} \\ &= \partial_t R\mathbf{u}^\gamma + \partial_t \gamma u_0 \left[0, \left(\boldsymbol{\alpha}^p_{R\overline{\mathbf{u}}}\right)_{\#}^\top\right]^\top + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m} u_0 f^p_{R\overline{\mathbf{u}},\mathbf{D}}\right\rangle - \left\langle R\mathbf{m} Q\left(u_0 f^p_{\overline{\mathbf{u}},\mathbf{D}}\right)\right\rangle && \text{(Lem. 5.8, 5.9)} \\ &= \partial_t R\mathbf{u}^\gamma + \partial_t \gamma u_0 \left[0, \left(\boldsymbol{\alpha}^p_{R\overline{\mathbf{u}}}\right)_{\#}^\top\right]^\top + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m} u_0 f^p_{R\overline{\mathbf{u}},\mathbf{D}}\right\rangle - \left\langle \mathbf{m} Q\left(u_0 f^p_{R\overline{\mathbf{u}},\mathbf{D}}\right)\right\rangle && \text{([7, 160])} \\ &= \partial_t R\mathbf{u} + \nabla_{\mathbf{x}} \cdot \left\langle \mathbf{v} \otimes \mathbf{m} f^p_{R\mathbf{u},\mathbf{D}}\right\rangle - \left\langle \mathbf{m} Q\left(f^p_{R\mathbf{u},\mathbf{D}}\right)\right\rangle && \text{(Eq. (5.114)),}\end{aligned} \tag{5.116}$$

which yields the assertion. Note, that the order zero moment u_0 is invariant with respect to rotations, since we have $R_{1,j} = \delta_{1,j}$ and $R_{i,1} = \delta_{1,i}$. □

We provide proof for Theorem 5.12.

Proof: (Theorem 5.12) By assumption, N_θ is convex in $\overline{\mathbf{u}}_{\#} = \langle \mathbf{m}_{\#}(\mathbf{v}) f(\mathbf{v})\rangle$ for restriction of the velocity variable onto $\mathbf{V}_{1,+}$ and $\mathbf{V}_{1,-}$. Thus $\hat{h}^p|_{\mathbf{V}_-}$ and $\hat{h}^p|_{\mathbf{V}_+}$ are convex, respectively.
Consider Algorithm 5.2. First, we notice, that normalization and re-scaling are invariant under rotations

since we have $R_{1,j} = \delta_{1,j}$ and $R_{i,1} = \delta_{1,i}$. We have the following relation between the moments $\overline{\mathbf{u}}^+$ and $\overline{\mathbf{u}}^-$,

$$\overline{\mathbf{u}}^- = R_\pm \overline{\mathbf{u}}^+, \qquad \text{and} \qquad \overline{\mathbf{u}}^+ = R_\pm^\top \overline{\mathbf{u}}^-, \tag{5.117}$$

where $R_\pm$ rotates a moment $\mathbf{u}$ by 180 degrees in the v_1, v_2 plane of $\mathbf{V}$, so the moments are "mirrored". Note that $R_\pm^\top = R_\pm$. We have $R_{\mathbf{V}_{1,-}} = R_\pm R_{\mathbf{V}_{1,+}}$.
1. The function

$$\hat{h}^{p,\mathrm{sym}}(\overline{\mathbf{u}}_\#) = \frac{1}{2}\left(\hat{h}^p\left(\left(R_{\mathbf{V}_{1,-}}\overline{\mathbf{u}}\right)_\#\right) + \hat{h}^p\left(\left(R_{\mathbf{V}_{1,+}}\overline{\mathbf{u}}\right)_\#\right)\right) \tag{5.118}$$

is convex for all $\overline{u}$, since $\hat{h}^p|_{\mathbf{V}_-}$ and $\hat{h}^p|_{\mathbf{V}_+}$ are convex. Further, a concatenation of a linear and a convex function is convex and the average of convex functions is convex.
2. Consider $\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}}$,

$$\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}} = \left[\vartheta\left(\left(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}}\right)_\#\right), \left(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}}\right)^\top_\#\right]^\top \tag{5.119}$$

with

$$\begin{aligned}(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}})_\# = \nabla_{\overline{\mathbf{u}}_\#}\hat{h}^{p,\mathrm{sym}}(\overline{\mathbf{u}}_\#) &= \frac{1}{2}\left(\frac{\mathrm{d}\overline{\mathbf{u}}^+_\#}{\mathrm{d}\overline{\mathbf{u}}_\#}\frac{\mathrm{d}\overline{\mathbf{u}}^-_\#}{\mathrm{d}\overline{\mathbf{u}}^+_\#}\frac{\mathrm{d}\hat{h}^p}{\mathrm{d}\overline{\mathbf{u}}^-_\#}(\overline{\mathbf{u}}^-_\#) + \frac{\mathrm{d}\overline{\mathbf{u}}^+_\#}{\mathrm{d}\overline{\mathbf{u}}_\#}\frac{\mathrm{d}\hat{h}^p}{\mathrm{d}\overline{\mathbf{u}}^+_\#}(\overline{\mathbf{u}}^+_\#)\right)\\ &= \frac{1}{2}\left(\widetilde{R}^\top_{\mathbf{V}_{1,+}}\widetilde{R}^\top_\pm\frac{\mathrm{d}\hat{h}^p}{\mathrm{d}\overline{\mathbf{u}}^-_\#}(\overline{\mathbf{u}}^-_\#) + \widetilde{R}^\top_{\mathbf{V}_{1,+}}\frac{\mathrm{d}\hat{h}^p}{\mathrm{d}\overline{\mathbf{u}}^+_\#}(\overline{\mathbf{u}}^+_\#)\right)\\ &= \frac{1}{2}\left(\widetilde{R}^\top_{\mathbf{V}_{1,+}}\widetilde{R}^\top_\pm\left(\boldsymbol{\alpha}^p_{\overline{u}^-}\right)_\# + \widetilde{R}^\top_{\mathbf{V}_{1,+}}\left(\boldsymbol{\alpha}^p_{\overline{u}^+}\right)_\#\right)\end{aligned} \tag{5.120}$$

where we denote

$$\widetilde{R}_{i,j} = R_{i+1,j+1}, \qquad \forall i, j = 1, \ldots, n-1 \tag{5.121}$$

as the Rotation matrix acting on the truncated Lagrange multiplier and remark that $R_{1,j} = \delta_{1,j}$ and $R_{i,1} = \delta_{1,i}$. This justifies the first 6 steps of the algorithm.
We directly see, that

$$\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}} = \left[\vartheta\left(\left(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}}\right)_\#\right), \left(\boldsymbol{\alpha}^{p,\mathrm{sym}}_{\overline{\mathbf{u}}}\right)^\top_\#\right]^\top \tag{5.122}$$

is indeed an approximation to $\alpha^\gamma_{\overline{\mathbf{u}}}$, since

$$\left(\boldsymbol{\alpha}^p_{\overline{u}^-}\right)_\# \approx \left(\boldsymbol{\alpha}^\gamma_{\overline{u}^-}\right)_\#, \qquad \text{and} \qquad \left(\boldsymbol{\alpha}^p_{\overline{u}^+}\right)_\# \approx \left(\boldsymbol{\alpha}^\gamma_{\overline{u}^+}\right)_\# \tag{5.123}$$

by construction. Furthermore,

$$\begin{aligned}
\vartheta\left(\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{p,\mathrm{sym}}\right)_{\#}\right) =& \vartheta\left(\frac{1}{2}\left(\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\widetilde{R}_{\pm}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{-}}^{p})_{\#} + \widetilde{R}_{\mathbf{V}_{1,+}}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{+}}^{p})_{\#}\right)\right) \\
=& -\log\left(\left\langle \exp\left(\frac{1}{2}\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{-}}^{p})_{\#}\cdot\mathbf{m}(\mathbf{v})_{\#}\right)\exp\left(\frac{1}{2}\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{+}}^{p})_{\#}\cdot\mathbf{m}(\mathbf{v})_{\#}\right)\right\rangle\right) \\
=& -\log\left(\left\langle \exp\left(\frac{1}{2}\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\widetilde{R}_{\pm}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{-}}^{p})_{\#}\cdot\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\widetilde{R}_{\pm}^{\top}\mathbf{m}(T_{\widetilde{R}_{\pm}^{\top}}T_{\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}}\mathbf{v})_{\#}\right)\right.\right. \\
&\qquad \left.\left.\exp\left(\frac{1}{2}\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{+}}^{p})_{\#}\cdot\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\mathbf{m}(T_{\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}}\mathbf{v})_{\#}\right)\right\rangle\right) \\
\approx& -\log\left(\left\langle \exp\left(\frac{1}{2}\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\widetilde{R}_{\pm}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{-}}^{\gamma})_{\#}\cdot\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\widetilde{R}_{\pm}^{\top}\mathbf{m}(T_{\widetilde{R}_{\pm}^{\top}}T_{\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}}\mathbf{v})_{\#}\right)\right.\right. \\
&\qquad \left.\left.\exp\left(\frac{1}{2}\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{+}}^{\gamma})_{\#}\cdot\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\mathbf{m}(T_{\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}}\mathbf{v})_{\#}\right)\right\rangle\right) \\
=& -\log\left(\left\langle \exp\left(\frac{1}{2}\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{+}}^{\gamma})_{\#}\cdot\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\mathbf{m}(T_{\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}}\mathbf{v})_{\#}\right)\right.\right. \\
&\qquad \left.\left.\exp\left(\frac{1}{2}\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{+}}^{\gamma})_{\#}\cdot\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\mathbf{m}(T_{\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}}\mathbf{v})_{\#}\right)\right\rangle\right) \\
=& -\log\left(\left\langle \exp\left(\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}(\boldsymbol{\alpha}_{\overline{u}^{+}}^{\gamma})_{\#}\cdot\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\mathbf{m}(T_{\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}}\mathbf{v})_{\#}\right)\right\rangle\right) \\
=& -\log\left(\left\langle \exp\left((\boldsymbol{\alpha}_{\overline{u}}^{\gamma})_{\#}\cdot\mathbf{m}(\mathbf{v})_{\#}\right)\right\rangle\right) \\
=& \vartheta\left(\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma}\right)_{\#}\right),
\end{aligned} \tag{5.124}$$

which gives the reconstruction of the first element of the Lagrange multiplier.
We show rotational invariance. Consider $\boldsymbol{\alpha}_{R\overline{\mathbf{u}}}^{p,\mathrm{sym}}$ for some rotation matrix R. Defining $\hat{\overline{\mathbf{u}}} = R\overline{\mathbf{u}}$

$$\begin{aligned}
\left(\boldsymbol{\alpha}_{R\overline{\mathbf{u}}}^{p,\mathrm{sym}}\right)_{\#} &= \left(\boldsymbol{\alpha}_{\hat{\overline{\mathbf{u}}}}^{p,\mathrm{sym}}\right)_{\#} = \frac{\mathrm{d}\overline{\mathbf{u}}_{\#}}{\mathrm{d}\hat{\overline{\mathbf{u}}}_{\#}}\frac{\mathrm{d}\hat{h}^{p,\mathrm{sym}}}{\mathrm{d}\overline{\mathbf{u}}_{\#}}(R\overline{\mathbf{u}}_{\#}) \\
&= \widetilde{R}\frac{1}{2}\left(\widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\widetilde{R}_{\pm}^{\top}\left(\boldsymbol{\alpha}_{\overline{u}^{-}}^{p}\right)_{\#} + \widetilde{R}_{\mathbf{V}_{1,+}}^{\top}\left(\boldsymbol{\alpha}_{\overline{u}^{+}}^{p}\right)_{\#}\right) \\
&= \left(\widetilde{R}\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{p,\mathrm{sym}}\right)_{\#},
\end{aligned} \tag{5.125}$$

thus we have an approximation for the reconstruction of the first element of the Lagrange multiplier. The same argument holds for $\vartheta\left((\boldsymbol{\alpha}_{R\overline{\mathbf{u}}}^{p,\mathrm{sym}})_{\#}\right)$. Naturally $\log(u_0)$ as well as $\left\|(\boldsymbol{\alpha}_{\mathbf{u}}^{p})_{\#}\right\|$ in the ansatz for $\boldsymbol{\alpha}_{\mathbf{u},\mathbf{D}}^{p}$ is invariant under rotations.
This yields a rotation-invariant approximation in the sense of Definition 5.10. □

Statements of §5.5

We provide proof for Theorem 5.13.

Proof: (Theorem 5.13) Let $(\alpha_{\overline{\mathbf{u}}}^{\gamma}) \in B_M^{\gamma}$, i.e., $\left\|(\alpha_{\overline{\mathbf{u}}}^{\gamma})\right\| \leq M$. Consider $\mathbf{u}$, see Eq. (5.48), and $\mathbf{u}^{\gamma}$, see Eq. (5.49), which directly yields

$$\left\|\mathbf{u}-\mathbf{u}^{\gamma}\right\| = \left\|u_0\gamma\left[0,\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma}\right)_{\#}^{\top}\right]^{\top}\right\| = u_0\gamma\left\|\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^{\gamma}\right)_{\#}\right\| = u_0\gamma M. \tag{5.126}$$

We consider now the difference between $\mathbf{u}^\gamma$ and $\mathbf{u}_\mathbf{D}^\gamma$, see Eq. (5.50), which gives

$$\begin{aligned}
\left\|\mathbf{u}^\gamma - \mathbf{u}_\mathbf{D}^\gamma\right\| &= \left\|\left\langle \mathbf{m} \exp\left(\left[\vartheta\left(\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\gamma\right)_\#\right) + \log(u_0), \left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\gamma\right)_\#^\top\right]^\top \cdot \mathbf{m}\right)\right\rangle \left(\exp(0) - \exp\left(-\frac{\gamma}{2}\left\|\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\gamma\right)_\#\right\|^2\right)\right)\right\| \\
&= u_0 \left\|\overline{\mathbf{u}}^\gamma\right\| \left\|\left(1 - \exp\left(-\frac{\gamma}{2}\left\|\left(\boldsymbol{\alpha}_{\overline{\mathbf{u}}}^\gamma\right)_\#\right\|^2\right)\right)\right\| \\
&\leq u_0 \left\|\overline{\mathbf{u}}^\gamma\right\| \left\|\left(1 - \exp\left(-\frac{\gamma}{2}M^2\right)\right)\right\| \\
&\leq u_0 n \left\|\left(1 - \exp\left(-\frac{\gamma}{2}M^2\right)\right)\right\|
\end{aligned} \tag{5.127}$$

where the last inequality follows from $\left\|\overline{u}^\gamma\right\| \leq n$, since $\overline{u}^\gamma \in \overline{\mathcal{R}} \in \mathbb{R}^n$. The error estimate of $\left\|\mathbf{u} - \mathbf{u}_\mathbf{D}^\gamma\right\|$ follows by the triangle inequality. □

CHAPTER 6

Neural Network-Based Continuum Breakdown Prediction in Multi-Scale Flows

In this chapter, we present a neural network based binary classifier to detect near-equilibrium and non-equilibrium regimes of gaseous flows based on local flow conditions. We draw on the minimal entropy closure and the Chapman-Enskog ansatz for hydrodynamic equations to construct a robust data generator for the binary classifier, which is superior to simulation-based sampling. An adaptive fluid-kinetic solver is constructed with the neural network-based regime prediction and validated in multi-scale and non-equilibrium flow physics. The neural-network classifier is compared to a traditional prediction of the continuum breakdown and shows significant improvement in a series of test cases.

6.1. Introduction

The non-linear Boltzmann equation is a powerful tool to model multi-scale gas dynamics from continuum to rarefied gas regimes. Flow regimes are often categorized according to the Knudsen number, which is defined as the ratio of molecular mean free path to a characteristic length scale. With the variation of Knudsen number, the domain of flow physics can be qualitatively divided into the continuum (Kn $<$ 0.001), slip (0.001 $<$ Kn $<$ 0.1), transition (0.1 $<$ Kn $<$ 10), and free molecular, i.e., rarefied regimes, (Kn $>$ 10) [227]. The Knudsen number indicates the relative importance between individual particle transports and their collective dynamics. Multi-scale simulations consider physical scenarios, where a wide range of Knudsen numbers appear, for example in hypersonic flight [86], gas turbine flows, and turbulent combustion [63] and hypersonic re-entry conditions [98]. The multi-scale nature of rarefied gaseous flows poses tremendous difficulties for theoretical and numerical analysis and is challenging to compute efficiently. Limited computational resources typically prohibit a full kinetic simulation of the scenario.

6.1.1. Numerical Methods for Different Gas Regimes

Different governing equations are routinely established to describe the fluid motions at different scales. As an example, in rarefied gas where Kn is of $O(1)$, the particle transport and collision processes are distinguishable and can thus be modeled by two independent operators in the Boltzmann equation. The direct Boltzmann solvers employ a discretized velocity phase space to compute transport and collision terms respectively, in analogy to the S_N method for the linear Boltzmann equation, see §1.2.1. An alternative methodology is the direct simulation Monte Carlo (DSMC) method [21], which mimics the probability distribution function with a large number of test particles and the collision term is calculated statistically.
In another limit with asymptotically small Kn, the Euler and Navier-Stokes equations are used to describe collective behaviors of fluid elements at a macroscopic level. It is worth mentioning that there is no quantitative description of the scale of a fluid element. Usually, it refers to a macroscopically infinitesimal concept, where the flow variables inside the element can be considered almost constant. With a high amount of intermolecular collisions, the fluid inside an element is considered to be in local thermodynamic equilibrium. On the other hand, the compressible Navier-Stokes solvers are mostly based on the Riemann solvers for inviscid flux and the central difference method for viscous terms. Only the macroscopic flow variables are tracked in the simulation.
In the continuum regime, it is feasible to employ computationally far more efficient macroscopic gas descriptions as Navier-Stokes, due to a drastically reduced number of degrees of freedom. As the Knudsen number increases, traditional Navier-Stokes equations fail, and gas kinetic equations are required to model the flow.
The well-known Chapman-Enskog expansion bridges the mesoscopic,i.e., kinetic and macroscopic [42] equations, where the Euler and Navier-Stokes equations can be derived from the asymptotic limits of expansion solutions of the Boltzmann equation. Although the hydrodynamic equations are based on first-principle modeling, the Chapman-Enskog ansatz provides a rigorous criterion to define their validity. In other words, the usage of hydrodynamic equations incorporates the assumption that the Chapman-Enskog solution is a proper approximation of the particle distribution function. However, this judgment cannot be verified in a macroscopic fluid simulation since the information on particle distribution functions has already been filtered in the coarse-grained modeling. Often, this leads to a misuse of the Navier-Stokes equations in simulation practice, when a local flow regime does not satisfy the continuum assumption anymore.
In typical gas simulations, the full kinetic model is only necessary for small parts of the computational domain, and the continuum assumption enables the use of efficient macroscopic models everywhere else, which motivates the use of adaptive fluid-kinetic solvers. The success of an adaptive flow solver relies on accurate prediction of the flow regimes.

6.1.2. Related Work on Flow Regime Classification

Different criteria have been proposed to predict the failure of continuum mechanics and construct the corresponding multi-scale numerical algorithms. Bird [21] proposed a parameter

$$\mathcal{P} = \frac{\partial_t(\ln \rho)}{\nu} \tag{6.1}$$

for the DSMC simulation of expansion flows, where ρ is gas density and ν is collision frequency, and the breakdown threshold of translational equilibrium is set as $\mathcal{P} = 0.05$.

Boyd et al. [26, 220] extended the above concept to a gradient-length-local Knudsen number

$$\mathrm{Kn}_{\mathrm{GLL}} = \frac{\ell}{\mathrm{I}} |\nabla_{\mathbf{x}} \mathrm{I}|, \tag{6.2}$$

where $\nabla_{\mathbf{x}} I$ is a local gradient, ℓ is the local molecular mean free path and I is a scalar quantity of interest, e.g. the local macroscopic density ρ, with the critical value being $C = 0.05$. Garcia et al. [81] proposed a breakdown parameter based on dimensionless stress τ_* and heat flux q_*, i.e.,

$$\mathcal{B} = \max(|\tau_*|, |q_*|), \tag{6.3}$$

with the switching criterion of $\mathcal{B} = 0.1$.
Levermore et al. [162] developed a criterion based on moment realizability of the minimal entropy closure for the Boltzmann moment system to estimate the validity of the Navier Stokes description of the flow field. The so-called validity matrix V_{NS} based on the first order moments, i.e., the macroscopic velocity of the gas, is constructed, which needs to have non-negative eigenvalues to correspond to positive kinetic densities f. Deviation of V_{NS} from the identity matrix is used as a measure of the physical accuracy of the Navier-Stokes simulation.
Recently, data-driven flow-regime classification techniques have emerged [204], where the authors develop an adaptive gas-kinetic solver based on the Navier Stokes and DSMC method. For given macroscopic solution variables, the corresponding kinetic density is approximated by a Gaussian process.

6.1.3. Novelty and Scientific Contribution

In this chapter we employ neural networks based classifier of the most probable flow regime based on local flow conditions. The inputs of the neural network are given by the macroscopic variables of the current grid cell, their spatial slopes, and the local collision time.
We extend the minimal entropy-based data-generation strategy of §4 and §5 to the gas-kinetic context and construct a simulation independent data-sampler to sample particle distributions near and out of equilibrium of the Boltzmann equation. Based on kinetic solutions, the ground-truth labels are rigorously determined by the deviation between the particle distribution functions and the Chapman-Enskog solutions. Based on the neural classifier, we develop a multi-scale adaptive method, which realizes a dynamic adaptation of flow regimes and fuses the continuum and kinetic solutions seamlessly.
The closest related works are [204, 162]. In contrast to [204], this work does not directly approximate the Lagrange multipliers of the minimal entropy closure for an unbounded velocity space $\mathbf{V}$, which may lead to numerical instabilities due to the Junk-Line [125], but gives a direct regime classification. The work of Levermore [162] relates to ours since we use moment realizability to sample distributions near the continuum breakdown, but the neural network uses other parameters as the slopes of the macroscopic variables to compute the regime classification.
This chapter is based on a collaboration with Tianbai Xiao and Martin Frank, where the authors contribution focuses mainly on the data-generation strategy. The work is currently under review in the Journal of Computational Physics and the preprint is available on ArXiv [241]. The code for the data generator, adaptive solver and the neural network implementation is available in the open source repositories KiT-RT[1] [147] and kinetic.jl[2] [239].

[1] https://github.com/CSMMLab/KiT-RT
[2] https://xiaotianbai.com/Kinetic.jl/dev/

6.1.4. The Chapter in Context of the Dissertation

This Chapter is an extension of the data-generation strategies of §4 and §5. We leverage the gained insights of the minimal-entropy closure, its data-structure, and in connection with equilibrium solutions of the Boltzmann Equation, we establish the data-generation strategy of this chapter. The resulting adaptive solver is a combination of a discrete ordinates discretization method of the non-linear Boltzmann Equation and the Navier-Stokes equation, a moment system with a specific closure. Thus, the chapter can be viewed as an excursion to the domain of compressible fluid mechanics, where we apply our findings of minimal-entropy closures to build efficient neural network based, hybrid numerical methods.

6.1.5. Organization of the Chapter

The paper is organized as follows. In §6.2, we introduce some fundamental concepts in the kinetic theory of gases, the Chapman-Enskog expansion, and the Navier-Stokes equation. §6.3 presents the idea and design of the neural network architecture and §6.4 introduces the strategy for generating data in training and test set. §6.5 details the numerical algorithm of the adaptive solver incorporated with the neural network classifier. In §6.6 we present several numerical test cases to validate the new method against kinetic reference simulations and $\mathrm{KN}_{\mathrm{GLL}}$-based adaptive simulations.

6.2. Kinetic Gas Dynamics

6.2.1. The Non-linear Boltzmann Equation

We reconsider the Boltzmann equation (1.1) for an unbounded velocity domain $\mathbf{V} = \mathbb{R}^d$ in d spatial dimensions with a non-linear collision operator that models interactions between particles based on the hard spheres model. Then, the Boltzmann equation describes the time-space evolution of a particle distribution function $f(t, \mathbf{x}, \mathbf{v})$ in dilute monatomic gas, i.e.,

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = Q(f, f) \tag{6.4}$$

The Boltzmann equation depicts a physical process with increasing physical, i.e., decreasing mathematical entropy. The H-theorem indicates that the (mathematical) entropy is a Lyapunov function for the Boltzmann equation and the logarithm of its minimizer must be a linear combination of the collision invariants [25, 160], see Definition 1.3. The equilibrium solution of the Boltzmann equation related to minimal (mathematical) entropy is the Maxwellian distribution function,

$$\mathcal{M}_{(\rho,\mathbf{U},T)}(\mathbf{v}) := \rho \left(\frac{m}{2\pi k T} \right)^{3/2} \exp\left(-\frac{m}{2kT} (\mathbf{v} - \mathbf{U})^2 \right), \tag{6.5}$$

where m is molecular mass, $\mathbf{U}$ is macroscopic fluid velocity, T is temperature, and k is the Boltzmann constant.
The macroscopic conservative flow variables can be obtained by taking moments from the particle distribution function over velocity space, i.e.,

$$\mathbf{u} = \left[\rho, \rho \mathbf{U}^\top, \rho E \right]^\top = \int_{\mathbf{V}} f \boldsymbol{\phi} \, \mathrm{d}\mathbf{v}, \tag{6.6}$$

where $\rho E = \rho |\mathbf{U}|^2 /2 + \rho e$, e is the internal energy per unit mass, and $\boldsymbol{\phi}$ is the vector of collision invariants, i.e.

$$\boldsymbol{\phi} = \left[1, \mathbf{v}^\top, |\mathbf{v}|^2 /2\right]^\top. \tag{6.7}$$

For an ideal gas, the internal energy is related to temperature as $\rho e = \frac{3}{2}\rho/mkT$. Taking moments of the Boltzmann equation with respect to collision invariants yields the transport equations for conservative variables,

$$\partial_t \mathbf{u} + \int_{\mathbf{V}} \boldsymbol{\phi} \mathbf{v} \cdot \nabla_{\mathbf{x}} f \, \mathrm{d}\mathbf{v} = 0, \tag{6.8}$$

where the right-hand side vanishes due to the collision invariants. The moment system reads,

$$\begin{aligned} &\partial_t \rho + \nabla_{\mathbf{x}} \cdot (\rho \mathbf{U}) = 0, \\ &\partial_t (\rho \mathbf{U}) + \nabla_{\mathbf{x}} \cdot (\rho \mathbf{U} \otimes \mathbf{U}) = \nabla \cdot \mathbf{P}, \\ &\partial_t (\rho E) + \nabla_{\mathbf{x}} \cdot (\rho E \mathbf{U}) = \nabla \cdot (\mathbf{P} \cdot \mathbf{U}) - \nabla \cdot \mathbf{q}, \end{aligned} \tag{6.9}$$

where the stress tensor $\mathbf{P}$ and heat flux $\mathbf{q}$ are defined as,

$$\mathbf{P} = \int_{\mathbb{V}} (\mathbf{v} - \mathbf{U})(\mathbf{v} - \mathbf{U}) f \, \mathrm{d}\mathbf{v}, \quad \mathbf{q} = \int_{\mathbb{V}} \frac{1}{2} (\mathbf{v} - \mathbf{U})(\mathbf{v} - \mathbf{U})^2 f \, \mathrm{d}\mathbf{v}. \tag{6.10}$$

It is clear that the flux terms in the above equations are one order higher than the leading terms, which leads to the well-known closure problem [160], see §1.2.2. From §1.3, we know that different closure strategies result in vastly different macroscopic transport equations.

6.2.2. The Chapmann-Enskog Expansion

In the following, we briefly show the methodology of Chapman-Enskog ansatz, where the Navier-Stokes equations can be derived from the asymptotic solution of the Boltzmann equation. With the introduction of the following dimensionless variables

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{L_0}, \ \tilde{t} = \frac{t}{L_0/V_0}, \ \tilde{\mathbf{v}} = \frac{\mathbf{v}}{V_0}, \ \tilde{f} = \frac{f}{n_0 V_0^3}, \tag{6.11}$$

where $V_0 = \sqrt{2kT_0/m}$ is the most probable molecular speed, the Boltzmann equation can be reformulated as

$$\partial_t \tilde{f} + \tilde{\mathbf{v}} \cdot \nabla_{\tilde{\mathbf{x}}} \tilde{f} = \frac{1}{\mathrm{Kn}} Q(\tilde{f}, \tilde{f}). \tag{6.12}$$

The Knudsen number is defined as

$$\mathrm{Kn} = \frac{V_0}{L_0 \nu_0} = \frac{\ell_0}{L_0}, \tag{6.13}$$

where ℓ_0 and ν_0 are the molecular mean free path and mean collision frequency in the reference state. For brevity, we drop the tilde notation to denote dimensionless variables henceforth.
Based on a small Knudsen number $\mathrm{Kn} = \varepsilon$, the Chapman-Enskog expansion approximates the particle distribution function [42] as

$$f \simeq f_\varepsilon = \sum_{n=0}^{\infty} \varepsilon^n f^{(n)}, \quad f^{(0)} := \mathcal{M}_{(\rho, \mathbf{U}, T)}. \tag{6.14}$$

Truncating the above expansion to the first non-trivial order, substituting it into Eq. (6.12) and projecting the kinetic system onto the hydrodynamic level, i.e., computing its moments, one can derive the Navier-Stokes equations. We refer the reader for a complete derivation to the literature [138]. The expansion solution for the Navier-Stokes regime writes

$$f_{\text{NS, Boltzmann}} = \mathcal{M}\left(1 - \frac{2\kappa}{5Rp}\left(\frac{\mathbf{c}^2}{2RT} - \frac{5}{2}\right)\mathbf{c}\cdot\nabla_{\mathbf{x}}(\ln T) - \frac{\mu}{RTp}\left(\mathbf{c}\otimes\mathbf{c} - \frac{1}{3}\mathbf{c}^2\mathbf{I}\right) : \nabla_{\mathbf{x}}\mathbf{U}\right), \tag{6.15}$$

where $\mathbf{c} = \mathbf{v} - \mathbf{U}$ is the peculiar velocity, and $\mathbf{I}$ is the identity tensor, R is the gas-constant and p is pressure. The viscosity μ and heat conductivity are determined by specific molecule models. For example, the viscosity coefficient for hard-sphere molecules takes the form

$$\mu = \mu_0\left(\frac{T}{T_0}\right)^{\omega}, \tag{6.16}$$

where the power index ω needs to be calibrated for different substances, and the heat conductivity is linked by the Prandtl number $\text{Pr} = c_p\mu/\kappa$ where c_p is the specific heat of the gas at a constant pressure. Lastly, one can approximate the collision time, i.e., the inverse of the collision frequency, $\tau = 1/\nu$ using the hard-sphere model [42, 20].

6.3. Neural Network-based Flow Regime Classification

Neural network-based surrogate models can provide an alternative for hand-crafted criteria to classify the continuum breakdown regions of a flow field, see e.g. §6.1.2. The idea is to train a neural network N_θ with variables θ to predict the local flow regime at the cell interface of two neighboring grid cells, based solely on the macroscopic variables, their slopes, and mean collision time, i.e.,

$$P_{\hat{r}} = \text{N}_\theta(\mathbf{Q}), \quad \text{with} \quad \mathbf{Q} = \left[\mathbf{u}^\top, \nabla_{\mathbf{x}}\mathbf{u}^\top, \tau\right]. \tag{6.17}$$

´The output $P_{\hat{r}}$ denotes the likelihood for the current cell to be in a non-equilibrium regime. The neural network employs the sigmoid function as activation in the last layer, and thus the output satisfies $P_{\hat{r}} \in [0, 1]$ naturally. With the floor function,$P_{\hat{r}}$ is transformed into a regime prediction $\hat{r}$, where 1 denotes rarefied (non-equilibrium) and 0 denotes continuum (near-equilibrium) regime.
For a given kinetic density function f_{ref}, the true flow regime label $r \in \{0, 1\}$ is defined as

$$r = \begin{cases} 1, & d > \epsilon \\ 0, & d \le \epsilon \end{cases}, \quad d = \frac{D_{KL}(f_{\text{NS}}, f_{\text{ref}})}{\rho}, \tag{6.18}$$

where D_{KL} denotes the Kullback-Leibler divergence of the reference particle distribution function and the reconstructed Navier-Stokes distribution, which is normalized by macroscopic density ρ. Following the Chapman-Enskog ansatz, the Navier-Stokes distribution function f_{NS} can be constructed using Eq. (6.15) for a given reference kinetic solution f_{ref}. The macroscopic quantities in the above equations can be obtained by taking moments of f_{ref} in Eq. (6.6), and the collision time $\tau = 1/\nu$ can be derived from the hard-sphere model.
Thus one may understand the neural networks' internal mechanism as an implicit reconstruction of the most probable kinetic solution, which is then compared to the Chapman-Enskog solution to determine the flow regime. The surrogate model provided by the neural network bridges macroscopic variables and flow regimes directly. Compared with classical criteria for continuum breakdown, no hand-crafted and

semi-empirical expansions are needed from asymptotic theory.
For this binary classification task, we employ the binary cross-entropy as a loss function, i.e.,

$$\mathcal{L} = -\frac{1}{T}\sum_{i\in T} r_i \cdot \log \hat{r}_i + (1 - r_i) \cdot \log\left(1 - \hat{r}_i\right), \tag{6.19}$$

where $\hat{r}$ is the model output, r is the ground truth regime value, and T denotes the size of the training set. The cross-entropy is equivalent to fitting the model using maximum likelihood estimation. Thus, the Kullback-Leibler divergence between the empirical distribution of training data and the distribution induced by the model is minimized.

6.4. Data Generation Strategies

As presented in Eq. (6.18), the information of exact particle distribution functions f_{ref} is needed to compute macroscopic quantities $\mathbf{Q}$ and regime labels. We define the sampling space of f_{ref} as

$$F_{\boldsymbol{\phi}} = \{f(\mathbf{v}) > 0 : |\langle \boldsymbol{\phi} f \rangle| < \infty\}, \tag{6.20}$$

which corresponds to the solution space of the minimal entropy closure (1.50), for the Maxwell-Boltzmann Entropy and a moment basis $\mathbf{m}$ consisting only of the collision invariants $\boldsymbol{\phi}$.
Any strategy to sample data from $F_{\boldsymbol{\phi}}$ creates a data-distribution P_x implicitly, which influences the training and test performance of the neural network, see §4.4. The goal of neural network-based classifier is to find a decision boundary between the near-equilibrium and non-equilibrium regimes. To this end we need to systematically create a data-distribution P_x that generates enough samples near the boundary between regimes. Similar to in §4.4, the baseline strategy is to sample data by kinetic simulations, which comes at the disadvantage of test case specific sampling bias and computational expense of a full kinetic solver.

6.4.1. Systematic Sampling of the Kinetic Density

Building upon the results of §4, we leverage the minimal entropy closure (1.52) and its reconstruction ansatz

$$f_{\text{ref}} = \eta_*'(\boldsymbol{\alpha} \cdot \mathbf{m}(\mathbf{v})), \tag{6.21}$$

for some Lagrange multiplier $\boldsymbol{\alpha}$ to systematically compute kinetic densities near and far away from m equilibrium solutions. We consider the non-linear Boltzmann equation for gas dynamics, thus the suitable choice for the entropy density is the Maxwell Boltzmann entropy [125],

$$\eta(f) = f\log(f) - f, \quad \eta'(f) = \log(f), \quad \eta_*(z) = \exp(z), \quad \eta_*'(z) = \exp(z). \tag{6.22}$$

We choose the basis $\mathbf{m}$ of $\mathbf{V}$ in a way that the first three moments coincide with the conservative variables of the Navier-Stokes equations in Eq. (6.6),i.e.,

$$\mathbf{m}(\mathbf{v}) = [\boldsymbol{\phi}^\top, \dots]^\top. \tag{6.23}$$

Remark, that we adapt $F_{\boldsymbol{\phi}}$ of Eq. (6.20) accordingly to $F_{\mathbf{m}}$ if we extend $\mathbf{m}$.
We sample the corresponding Lagrange multipliers $\boldsymbol{\alpha}$ to generate f_{ref}, using Eq. (6.21). This ansatz is able to reconstruct the Maxwellian of Eq. (6.5), i.e., equilibrium distribution, with

$$\mathcal{M}_{(\rho,\mathbf{U},T)}(\mathbf{v}) = \exp\left(\boldsymbol{\alpha} \cdot \boldsymbol{\phi}(\mathbf{v})\right), \tag{6.24}$$

Algorithm 6.1: Minimum entropy sampling of kinetic densities

Input: $\boldsymbol{V}_{tr}$: Truncated velocity domain
$\sigma_{\boldsymbol{\alpha}}$: Sampling std. deviation for $\boldsymbol{\alpha}$
$T, \boldsymbol{U}$: Temperature and bulk velocity
τ_{cond}: Condition number threshold

Result: F_T: Set of sampled kinetic densities

for $i = 0$ *to* $i = T$ **do**
  $\boldsymbol{\beta}_{\text{mean}} \leftarrow \left[\frac{\mathbf{U}^\top}{kT}, -\frac{1}{2kT}, 0, \dots\right]^\top$ /* Compute sampling mean */
  do
    $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta}_{\text{mean}}, \sigma_{\boldsymbol{\alpha}})$ /* Sample reduced multipliers */
    $\boldsymbol{\alpha} \leftarrow [\vartheta(\boldsymbol{\beta}), \boldsymbol{\beta}^\top]^\top$ /* Reconstruct normalized Lagrange multipliers */
  while $\lambda_{min} < \tau_{cond}$
  $f_i \leftarrow \exp(\boldsymbol{\alpha} \cdot \mathbf{m})$ /* Compute kinetic density */
  Append f_i to F_T.

choosing $\boldsymbol{\alpha}$ as

$$\alpha_0 = \log\left(\frac{\rho}{(2\pi kT)^{3/2}}\right) - \frac{|\mathbf{U}|^2}{2kT}, \quad \boldsymbol{\alpha}_1 = \frac{\mathbf{U}}{kT}, \quad \alpha_2 = -\frac{1}{2kT}. \tag{6.25}$$

The deviation from the equilibrium state can be generated by adding white noise to $\boldsymbol{\alpha}$. Thus the generated Lagrange multiplier vector $\boldsymbol{\alpha}_i$ is sampled from a normal distribution with a mean given by $\boldsymbol{\alpha}$ of Eq. (6.25) and with mean 0 for α_i with $i > 2$, i.e., in the case of an extended basis. The macroscopic variables $\mathbf{U}, \rho\, T$ are chosen based on the test case. Lagrange multipliers of higher order model deviation of the Maxwellian.
We make use of the data-normalization strategy of §4.3.1, to reduce a degree of freedom from the sampling data. Thus, we sample $\boldsymbol{\beta} = \boldsymbol{\alpha}_{\#}$, where $(\cdot)_{\#}$ is the reduction operator of Eq. (5.8), then we use the reconstruction map ϑ of Eq. (5.20), to get $\boldsymbol{\alpha}$.
Consider again the Hessian H_n of the entropy closure, see Eq. (1.65), i.e.,

$$H_n(\boldsymbol{\alpha}) = \left\langle \mathbf{m} \otimes \mathbf{m} \eta_*''(\boldsymbol{\alpha} \cdot \mathbf{m}) \right\rangle = \langle \mathbf{m} \otimes \mathbf{m} \exp(\boldsymbol{\alpha} \cdot \mathbf{m}) \rangle\,, \tag{6.26}$$

which we use similarly as in §4 and §5 as a measure of deviation from the equilibrium state since reconstructed kinetic densities with a low condition number are close to the Maxwellian. Highly anisotropic densities, i.e., where H_n has a high condition number, deviate strongly from a Maxwellian., see Fig. 6.1.
Although theoretical gas dynamics describe an unbounded velocity domain, numerical implementations use a local, truncated velocity domain $\mathbf{V}_{tr} \subset \mathbb{R}^d$ centered around the current macroscopic velocity $\mathbf{U}$, which is described in detail in §6.5. Thus, we assume a local, truncated velocity domain. The resulting sampling strategy is summarized in Algorithm 6.1.

Remark 6.1 The minimal entropy closure in an unbounded velocity domain yields theoretical challenges, since the set $F_{\mathbf{m}} = \{f > 0 : \langle \mathbf{m} f \rangle = \mathbf{u}\}$ is not closed when $\mathbf{V}$ is unbounded, since the map $f \mapsto \langle \mathbf{m} f \rangle$ is not continuous. As a consequence, the entropy functional $h(\mathbf{u})$ is looses strict convexity on a line starting from the Maxwellian distribution dubbed the Junk line [124, 125]. Furthermore, the realizable set $\mathcal{R}$, i.e.,

$$\mathcal{R} = \{\mathbf{u} : \langle \mathbf{m} g \rangle = \mathbf{u},\ g \in F_{\mathbf{m}}\}\,, \tag{6.27}$$

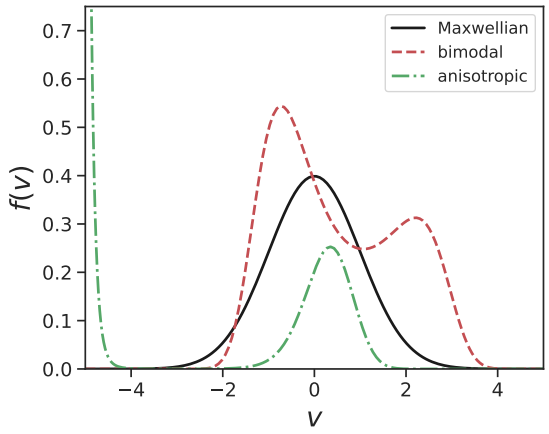


Figure 6.1.: Sampling of particle distribution functions. The more a function deviates from the Maxwellian, the higher is the condition number of the corresponding entropy problem.

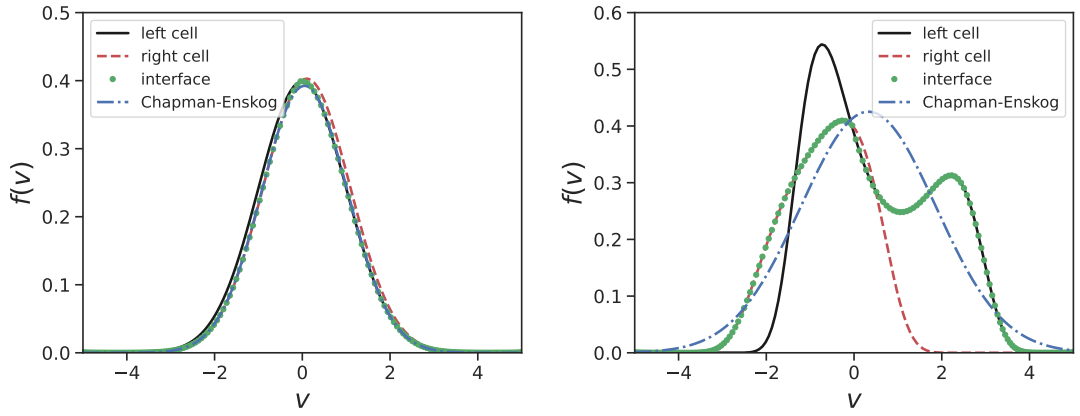
Algorithm 6.2: Sampling of labeled training data

Input: $[\mathrm{Kn}_{min}, \mathrm{Kn}_{max}]$: Range of Knudsen numbers
$[\rho_{min}, \rho_{max}]$: Range of particle densities
$[\boldsymbol{x}_{min}, \boldsymbol{x}_{max}]$: Range of cell-center distances

Result: $X_T = \{(\boldsymbol{Q}_i, r_i)\}_{i \in T}$: Training data-set

$F_T \leftarrow$ `Algorithm 6.1` `/* Compute reference kinetic density functions */`
for $i = 0$ *to* $i = T$ **do**
  $f_L,\ f_R \sim F_T$ `/* Sample` f `of left and right cell */`
  $\rho_L,\ \rho_R \sim$ `uniform`$([\rho_{min}, \rho_{max}])$ `/* Sample macroscopic density to` f_L `and` f_R `*/`
  $f_L,\ f_R \leftarrow \rho_L f_L,\ \rho_R f_R$ `/* Scale` f_L `and` f_R `*/`
  $\mathbf{u}_L, \mathbf{u}_R \leftarrow \langle \boldsymbol{\phi} f_L \rangle, \langle \boldsymbol{\phi} f_R \rangle$ `/* Compute macroscopic variables */`
  $\mathbf{x}_L,\ \mathbf{x}_R \sim$ `uniform`$[\boldsymbol{x}_{min}, \boldsymbol{x}_{max}]^d$ `/* Sample ghost cell centroids */`
  $\mathbf{n} \leftarrow (\mathbf{x}_L - \mathbf{x}_R) / \|\mathbf{x}_L - \mathbf{x}_R\|$ `/* Compute cell interface normal */`
  $f_{\mathrm{ref}}(\mathbf{v}) \leftarrow f_L(\mathbf{v}) \mathcal{H}(\mathbf{n} \cdot \mathbf{v}) + f_R(\mathbf{v}) (1 - \mathcal{H}(\mathbf{n} \cdot \mathbf{v}))$ `/* Upwind scheme */`
  $\mathbf{u}_i \leftarrow \langle \boldsymbol{\phi} f_{\mathrm{ref}} \rangle$ `/* Compute macroscopic variables */`
  $\nabla_{\mathbf{x}} \mathbf{u}_i \leftarrow (\mathbf{u}_L - \mathbf{u}_R) / \|\mathbf{x}_L - \mathbf{x}_R\|$ `/* Compute slope of macroscopic variables */`
  $\tau_i \leftarrow \mu / p$ `/* Hard-sphere model for` τ `*/`
  $f_{\mathrm{NS}} \leftarrow$Eq. (6.15) `/* BGK reconstruction of` f_{NS} `*/`
  $r_i \leftarrow$Eq. (6.18) `/* Determine local regime */`
  $\mathbf{Q}_i \leftarrow \left[\mathbf{u}_i^\top, \nabla_{\mathbf{x}} \mathbf{u}_i^\top, \tau_i\right]^\top$
  Append $(\mathbf{Q}_i, r_i)$ to X_T.

is not convex anymore, but star-shaped at the moment of the Maxwellian distribution. Near the Junk line, kinetic densities approach an exponential, which leads to serious robustness and stability problems of the entropy reconstruction and corresponding kinetic solvers [219, 196].



(a) Equilibrium solutions in ghost cells, $\tau = 0.0012$

(b) Non-equilibrium solutions in ghost cells, $\tau = 0.000511$

Figure 6.2.: Sampling of reference solutions at the left and right cell, cell interface, and the Chapman-Enskog reconstruction for $Kn = 1e{-}3$ and $\Delta x = 1e{-}2$ for $\mathbf{V} \subset \mathbb{R}$. At near equilibrium (left), the upwind and Chapman-Enskog reconstruction are similar, whereas for kinetic samples (right), they differ vastly.

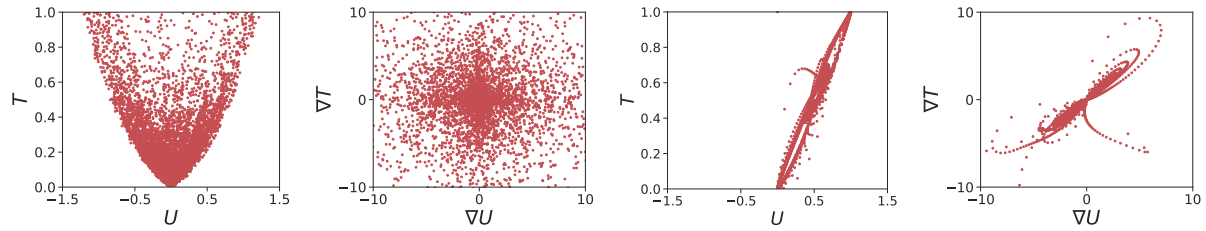
6.4.2. Sampling and Labeling of Macroscopic Data

The idea for data generation is to combine two sampled distribution functions $\{f_L, f_R\}$ with two adjacent ghost cells, where their spatial positions $\{\mathbf{x}_L, \mathbf{x}_R\}$ as well as the unit normal vector $\mathbf{n}$ of the cell interface are randomly sampled. Therefore, the reference particle distribution function at the interface can be approximated via an upwind reconstruction,

$$f_{\text{ref}}(\mathbf{v}) = f_L(\mathbf{v})\mathcal{H}(\mathbf{n}\cdot\mathbf{v}) + f_R(\mathbf{v})\left(1 - \mathcal{H}(\mathbf{n}\cdot\mathbf{v})\right), \tag{6.28}$$

where $\mathcal{H}$ is the heaviside step function. Figure 6.2 illustrates the process.
The conservative variables $\{\mathbf{u}, \mathbf{u}_L, \mathbf{u}_R\}$ are obtained by taking moments of f_{ref}, and the gradients $\nabla_x \mathbf{u}$ are computed with a finite difference formula.
Using a randomly sampled Knudsen-number Kn from a predefined range, we can compute the local collision time $\tau = 1/\nu$ and obtain a completely assembled training data point $\mathbf{Q} = (\mathbf{u}, \nabla_{\mathbf{x}}\mathbf{u}, \tau)$. Finally, we compute the label of the training data point by first computing f_{NS} using Eq. (6.15) and then calculating the distance to the sampled reference solution f_{ref} using Eq . (6.18). The resulting sampling strategy is displayed in Alogrithm 6.2.

We illustrate the sampling data distribution P_x obtained from Alogrithm 6.2 and compare it to simulation based sampling of Sod shock tube §6.6.1 with varying initial conditions in Fig. 6.3. The upper row displays the sampled macroscopic variables and the lower row the corresponding gradients for a 1D problem. The left column displays P_x obtained by Algorithm 6.2 and the right P_x obtained by simulation data. The results have been normalized via $\overline{\mathbf{u}} = \mathbf{u}/\rho$. Clearly, data sampled by Algorithm 6.2 produces better coverage of the realizable set $\mathcal{R}$, i.e., the set of feasible data, as the simulation sampler. The simulation data is naturally not i.i.d., accumulated along trajectories of the solution and heavily biased by the choice of the test case and initial conditions.



(a) Algorithmic $\mathbf{u}$. (b) Algorithmic $\nabla_{\mathbf{x}}\mathbf{u}$. (c) Solver $\mathbf{u}$. (d) Solver $\nabla_{\mathbf{x}}\mathbf{u}$.

Figure 6.3.: Data sampled by Algorithm 6.2 (a-b) and by kinetic simulation (c-d). The samples from the kinetic solver have a strong bias toward positive bulk velocity, and T and U are strongly correlated. Algorithmic samples cover large parts of the realizable set $\mathcal{R}$. Their spatial gradients are almost normally distributed with mean 0, whereas the gradients of simulation data follow the solution trajectories.

6.5. Solution Algorithm

In this section, we present the numerical implementation of the adaptive scheme based on the neural classifier. The solution algorithm is built on top of a finite volume method, which is in detail described in §3.

6.5.1. Kinetic solver

In principle, the finite volume scheme at the kinetic level is an S_N method for the non-linear Boltzmann equation with an unbounded velocity domain $\mathbf{V} = \mathbb{R}^d$, see §1.2.1. For the numerical implementation, a local velocity mesh is generated as a ball with radius $4\sqrt{RT_0}$, i.e.,

$$\mathbf{V}_{tr} = B_{r=4\sqrt{RT_0}}(\mathbf{V}_0) \subset \mathbf{V}, \tag{6.29}$$

where $\{\mathbf{V}_0, T_0\}$ are reference velocity and temperature, and R is the gas constant. The velocity grid is chosen such that more than 99% of values of the Maxwellian distribution fall into its range. Given the notation of a cell-averaged particle distribution function in the spatial element $\mathbf{X}_i$ with measure A_i at discrete velocity $\mathbf{v}_q \in \mathbf{V}_{tr}$ at time step n,

$$f_{i,q}^n = \frac{1}{A_i}\int_{\mathbf{X}_i} f(t^n, \mathbf{x}, \mathbf{v}_q)\,\mathrm{d}\mathbf{x}, \tag{6.30}$$

the update algorithm of the finite volume scheme writes

$$f_{i,q}^{n+1} = f_{i,q}^n - \frac{1}{A_i}\int_{t^n}^{t^{n+1}} \sum_{j\in N(i)} F(f_{j,q}, f_{i,q})_{up}\,\mathrm{d}t + \int_{t^n}^{t^{n+1}} Q_j\left(f_i(\mathbf{v}), f_i(\mathbf{v})\right)\,\mathrm{d}t, \qquad q = 1,\dots,N_q, \tag{6.31}$$

where $N(i)$ are the neighbor cells of cell i and F_{up}^{kin} is an upwind flux for a nodal discretization scheme described by Eq. (3.20), i.e.,

$$F(f_j, f_i)_{up,q}^{\text{kin}} = \mathbf{v}_q \cdot \mathbf{n}_{i,j} f_{i,q} \mathcal{H}(\mathbf{v}\cdot\mathbf{n}_{i,j}) + \mathbf{v}_q \cdot \mathbf{n}_{i,j} f_{j,q}^n \left(1 - \mathcal{H}(\mathbf{v}\cdot\mathbf{n}_{i,j})\right), \qquad q = 1,\dots,N_q \tag{6.32}$$

where $\mathcal{H}$ is the heaviside step function and $\mathbf{v}$ is the transport velocity vector. Naturally, one can also use its second-order counterpart. The time integral can be discretized with a suitable implicit or explicit integrator. Inside each element, the collision term $Q(f, f)$ is computed by the fast spectral method [239]. The discrete Fourier transform is employed to solve the convolution in the spectral domain efficiently. We refer to [190] for a detailed formulation of this method.

6.5.2. Navier-Stokes Solver

The Navier-Stokes solver is a finite volume scheme moment system of the non-linear Boltzmann equation with a special closure, see §3.4. In the absence of a velocity space, we define the average conservative flow variables in an element as

$$\mathbf{u}_i^n = \frac{1}{A_i} \int_{\mathbf{X}_i} \mathbf{u}(t^n, \mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{6.33}$$

and the finite volume algorithm writes

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{1}{A_i} \int_{t^n}^{t^{n+1}} \sum_{j \in N(i)} F(i,j)_{up}^{\mathrm{NS}} \, \mathrm{d}t. \tag{6.34}$$

A key step for solving conservation laws is to compute the fluxes $\mathbf{F}_{up}^{\mathrm{NS}}$ of conservative variables. We employ the Chapman-Enskog solution from the BGK-type relaxation model [19] to construct numerical fluxes and to close the moment system. The relaxation model writes

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = \nu(\mathcal{E} - f). \tag{6.35}$$

The equilibrium distribution $\mathcal{E}$ can be chosen as the Maxwellian in Eq. (6.5) or its variants [212, 110], and ν is the collision frequency. The above equation can be written in the following successive form

$$f = \mathcal{E} - \tau \partial_t \mathcal{E} + \tau \partial_t (\tau \partial_t \mathcal{E}) + \ldots, \tag{6.36}$$

where ∂_t denotes total derivative operator and $\tau = 1/\nu$. The above equation has the same structure as Eq. (6.14), and thus the first-order truncation of Chapman-Enskog expansion writes [192],

$$f \simeq \mathcal{E} - \tau(\partial_t \mathcal{E} + \mathbf{v} \cdot \nabla_{\mathbf{x}} \mathcal{E}). \tag{6.37}$$

In the solution algorithm, we follow the Chapman-Enskog expansion and construct the particle distribution function at the cell interface with an upwind approach, i.e.,

$$\begin{aligned} f_i &= \mathcal{E}_i \left(1 - \tau \left(\mathbf{a}_i \cdot \mathbf{v} + b_i\right)\right), \\ f_j &= \mathcal{E}_j (1 - \tau(\mathbf{a}_j \cdot \mathbf{v} + b_j)), \end{aligned} \tag{6.38}$$

where $\mathcal{E}_i$ and $\mathcal{E}_j$ are the equilibrium distributions computed from reconstructed macroscopic variables at the left and right side of the cell interface, i.e.,

$$\begin{aligned} \widetilde{\mathbf{u}}_i &= \mathbf{u}_i + \Psi_i \left(\nabla_{\mathbf{x}} \mathbf{u}_i \cdot \mathbf{r}_{i,j}\right), \\ \widetilde{\mathbf{u}}_j &= \mathbf{u}_j + \Psi_j \left(\nabla_{\mathbf{x}} \mathbf{u}_j \cdot \mathbf{r}_{j,i}\right), \end{aligned} \tag{6.39}$$

where we follow the second order upwind scheme of §3.4.2 with slope limiter Ψ. Again $\mathbf{r}_{i,j} = \mathbf{x}_{i,j} - \mathbf{x}_i$ is the vector pointing from the centroid of cell i to the interface midpoint between cells i and j. $\nabla_{\mathbf{x}} \mathbf{u}_i$ and $\nabla_{\mathbf{x}} \mathbf{u}_j$ are the gradients of the macroscopic values of neighboring cells, which can be computed with the Green-Gauss theorem or with finite differences. In a well-resolved region, the relation $\widetilde{\mathbf{u}}_i = \widetilde{\mathbf{u}}_j$ holds, and Eq. (6.38) deduces to the standard Chapman-Enskog expansion naturally. The spatial derivatives of the particle distribution function $\mathbf{a}_i$, $\mathbf{a}_j$ is related to macroscopic slopes via

$$\begin{aligned} \langle \mathbf{a}_i \mathcal{E}_i \boldsymbol{\phi} \rangle &= \nabla_{\mathbf{x}} \widetilde{\mathbf{u}}_i, \\ \left\langle \mathbf{a}_j \mathcal{E}_j \boldsymbol{\phi} \right\rangle &= \nabla_{\mathbf{x}} \widetilde{\mathbf{u}}_j, \end{aligned} \tag{6.40}$$

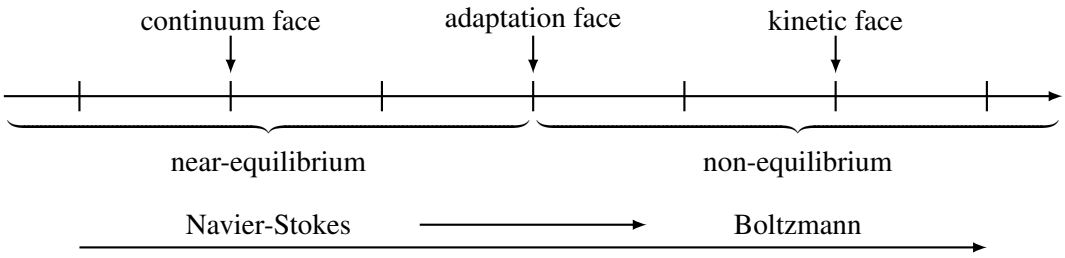


Figure 6.4.: Schematic of the adaptive scheme for multi-scale flow.

where $\boldsymbol{\phi}$ are the collision invariants. Then $\mathbf{a}_i$ and $\mathbf{a}_j$ can be obtained by solving a linear system [242]. Then the time derivative b_i and b_j can be obtained through the compatibility condition of the BGK model, i.e.,

$$\langle \nu(\mathcal{E} - f)\boldsymbol{\phi} \rangle = 0, \tag{6.41}$$

which yields

$$\begin{aligned} \langle b_i \mathcal{E}_i \boldsymbol{\phi} \rangle &= -\langle (\mathbf{a}_i \cdot \mathbf{v}) \mathcal{E}_i \boldsymbol{\phi} \rangle, \\ \left\langle b_j \mathcal{E}_j \boldsymbol{\phi} \right\rangle &= -\left\langle (\mathbf{a}_j \cdot \mathbf{v}) \mathcal{E}_j \boldsymbol{\phi} \right\rangle. \end{aligned} \tag{6.42}$$

After the coefficients for spatial and time variations are determined, the interface fluxes for macroscopic variables can be obtained by taking moments over particle velocity space, i.e.,

$$F(i,j)_{up}^{\mathrm{NS}} = \left\langle \mathbf{v} \otimes \boldsymbol{\phi} \left(f_i(\mathbf{v}) \mathcal{H}(\mathbf{v} \cdot \mathbf{n}_{i,j}) + f_j(\mathbf{v}) \left(1 - \mathcal{H}(\mathbf{v} \cdot \mathbf{n}_{i,j}) \right) \right) \right\rangle, \tag{6.43}$$

where $\mathcal{H}$ is the heaviside step function and $\mathbf{n}_{i,j}$ is the interface normal. Since the equilibrium state is based on Gaussian distribution, the above integral can be evaluated analytically. Remark, that the above numerical method can be understood as a simplification of gas-kinetic scheme [242].

6.5.3. Adaptation Strategy

The Boltzmann and Navier-Stokes solvers can be combined to solve multi-scale flow problems efficiently with an adaptive continuous-discrete velocity transformation. The work paradigm is shown in Fig. 6.4. For a near-equilibrium flow region, the kinetic density can be formulated analytically from the Chapman-Enskog expansion. Therefore, only the macroscopic flow variables are needed to store and iterate by the Navier-Stokes solver in Eq. (6.34). For non-equilibrium flows, the solution algorithm allocates the localized velocity quadrature to track the evolution of kinetic density in Eq. (6.31).
A core task of the adaptive solver lies in the dynamic adaptation of time-varying flow regimes at different locations. At every time step t^n, the spatial derivatives of the updated macroscopic variables are evaluated via

$$\nabla_{\mathbf{x}} \mathbf{u} = \frac{\nabla_{\mathbf{x}} \mathbf{u}_i + \nabla_{\mathbf{x}} \mathbf{u}_j}{2}. \tag{6.44}$$

The collision time τ is evaluated from the hard-sphere model and thus, the complete neural network input to predict the flow regime has been obtained. As shown in Fig. 6.4, we have two types of cells, i.e., the non-equilibrium one holding a discrete solution of the distribution function and the near-equilibrium one with Navier-Stokes variables, and three types of cell interfaces based on the flow regimes, i.e.,

- kinetic face: two neighboring cells are in the non-equilibrium flow regime;

Table 6.1.: Computational setup for the numerical test cases

Test Case	CFL	t_f	Δ_t	Δ_x	n_q
Sod shock tube 1D	0.5	0.15	2.50e−3	5.00e−3	$64 \times 32 \times 32$
Circular cylinder 2D	0.5	–	3.14e−2	6.28e−2	$50 \times 48 \times 48$

- continuum face: two neighboring cells of the face are in near-equilibrium flow regime;
- adaptation face: two neighboring cells of the face lie in different flow regimes.

The solution algorithm in kinetic and continuum type cells is straightforward following §6.5.1 and §6.5.2. At the adaptation face, both macroscopic and microscopic fluxes are evaluated to update the solutions in the left and right cells. This is uniformly done by computing the kinetic flux in Eq. (6.32), where its velocity moments results macroscopic fluxes, i.e.,

$$\mathbf{F}_{up}^{W} = \left\langle F_{up}^{\text{kin}} \boldsymbol{\phi} \right\rangle_{\mathbf{V}_{tr}} \approx \sum_{q}^{N_q} w_q F_{up,q}^{\text{kin}} \boldsymbol{\phi}(\mathbf{v}_q), \tag{6.45}$$

where N_q is the number of quadrature points and w_q the quadrature weights.
Note, that the numerical stability of the adaptive algorithm is given by the stability of the discretization schemes for the kinetic and macroscopic fluxes. Thus the adaptive solver can only increase the solution accuracy compared with a full macroscopic solver, and does not raise additional numerical stability concerns.

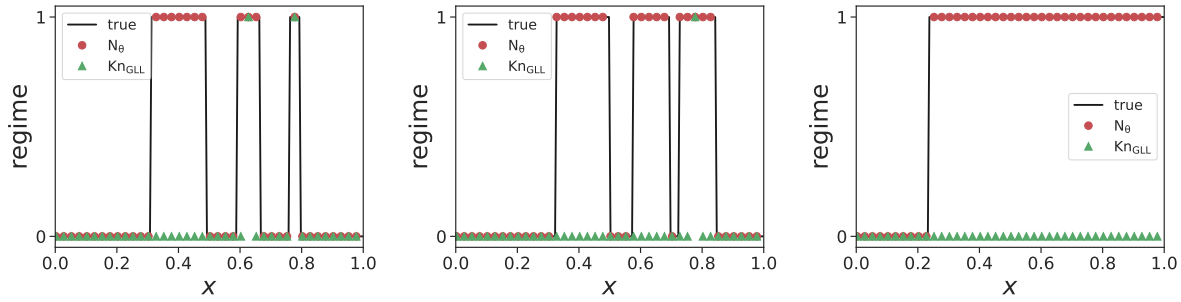
6.6. Numerical Results

In this section, we conduct numerical experiments of multi-scale flow problems to validate the neural network-based flow regime classifier and the corresponding adaptive solver. All the variables are nondimensionalized following the paradigm introduced in §6.2. The hard-sphere gas model is employed in all cases. We choose the gradient-length-local Knudsen number Kn_{GLL} [26] as a baseline for an adaptive solver and use a full kinetic solver as a high-fidelity reference solution. We provide quantitative comparisons of the flow regime predictions between Kn_{GLL} and the neural network, as well as simulation comparisons to the reference solution.
The employed neural networks are feed-forward architectures in ResNet design, where the input dimension depends on the spatial dimension of the test case at hand. The computational resources of the adaptive solver can be found in the software package kinetic.jl[3].
The ADAM optimizer is used for the neural network training with default learning rate. The training and testing data is produced by sampling and processing prescribed kinetic solutions of particle distribution functions, and the test set is generated with the help of kinetic simulation data from numerical cases.

6.6.1. Sod Shock Tube

The first numerical experiment is the Sod shock tube, where the longitudinal processes dominate the flow motion in the one-dimensional Riemann problem [159]. The domain is partitioned on the left and right

[3] https://github.com/vavrines/Kinetic.jl/



(a) Kn = 1e−4 (b) Kn = 1e−3 (c) Kn = 1e−2

Figure 6.5.: Prediction of flow regimes from fully kinetic solutions at $t = 0.15$ in the Sod shock tube with different criteria (0 is continuum, 1 is non-equilibrium). $\mathrm{Kn}_{\mathrm{GLL}}$ underestimates the influence of wave structures and makes inaccurate predictions, whereas network-based prediction is more accurate.

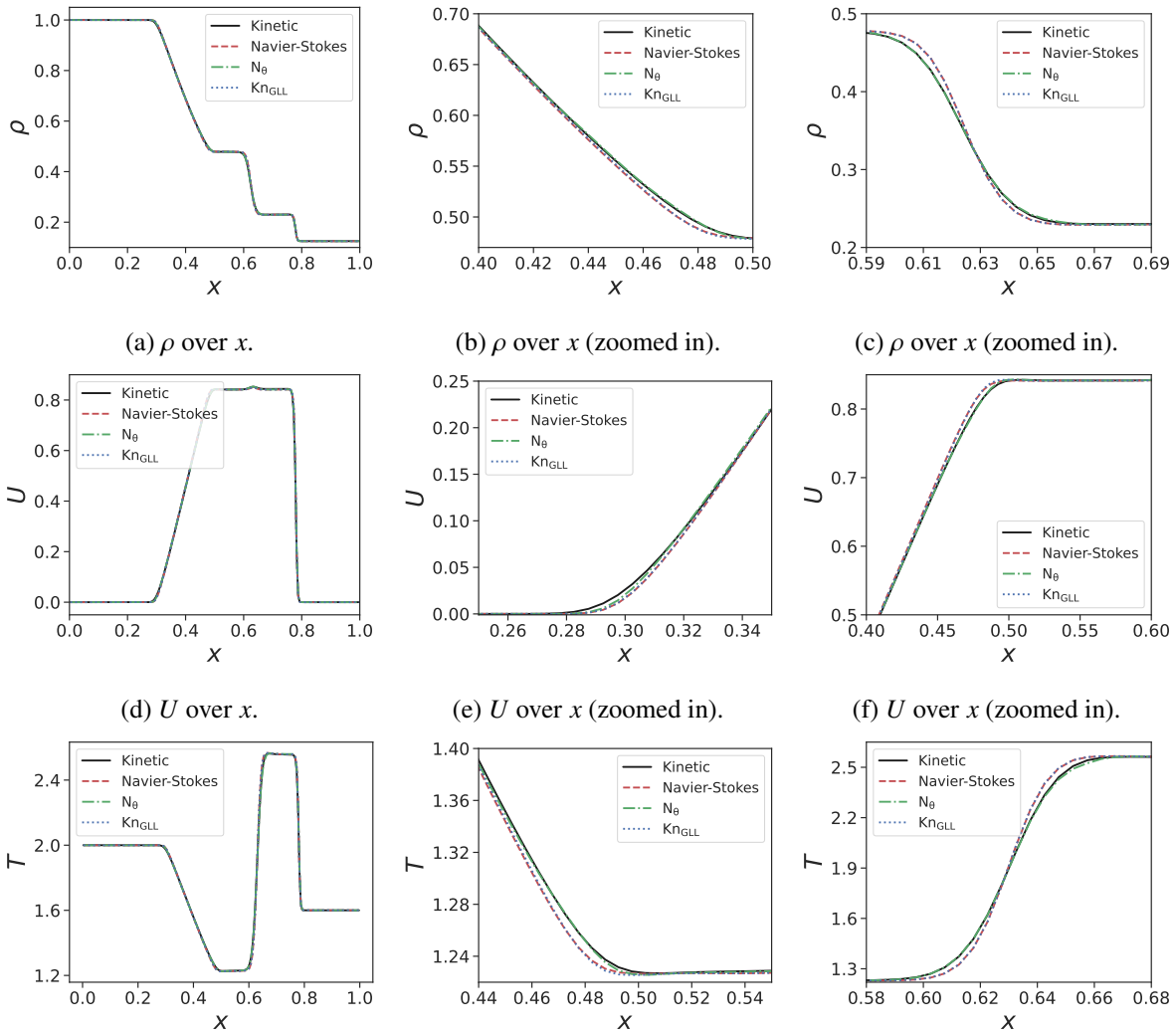
side of a shock with different macroscopic values, i.e.

$$\begin{bmatrix} \rho \\ U \\ T \end{bmatrix}_{t_0,L} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} \rho \\ U \\ T \end{bmatrix}_{t_0,R} = \begin{bmatrix} 0.125 \\ 0 \\ 1.6 \end{bmatrix}. \tag{6.46}$$

The kinetic density is initialized as a corresponding Maxwellian.
To test the capability of the current scheme to solve multi-scale flow problems, simulations are performed with different reference Knudsen numbers, i.e., $\mathrm{Kn} \in \{0.0001, 0.01\}$. We truncate $\mathbf{V} = [-8, 8]$ and choose spatial Dirichlet boundary conditions corresponding to the initial conditions. The detailed computation setup is listed in Table 6.1.

We first conduct a fully kinetic simulation with the Boltzmann equation, to obtain a ground truth for the flow regimes from the KL-divergence between the particle distribution and its Chapman-Enskog reconstruction of Eq. (6.18). Regime predictions based on neural networks and $\mathrm{Kn}_{\mathrm{GLL}}$ are shown in Fig. 6.5. With an increasing Knudsen number, the kinetic regime enlarges due to the increasing rarefied gas effect. Then, we employ the adaptive solver to conduct complete simulations based on the criteria from the neural network and $\mathrm{Kn}_{\mathrm{GLL}}$. It is known that localized flow structures, including rarefaction and shock waves, and contact discontinuities, contribute as sources of non-equilibrium effects. In the remaining near-equilibrium regions, the Chapman-Enskog expansion can approximate real particle distributions.
The profiles of density and temperature inside the shock tube at the time instant $t = 0.15$ under different Knudsen numbers are presented in Fig. 6.6, 6.7 and 6.8. The kinetic and Navier-Stokes solutions are plotted as a benchmark. As shown, although all the results are qualitatively similar, the zoom-in view demonstrates that the adaptive solution based on $\mathrm{Kn}_{\mathrm{GLL}}$ stands closer to the Navier-Stokes results, while the neural network corresponds to the Boltzmann solution. At $\mathrm{K}n = 0.01$, the Chapman-Enskog expansion yields negative values in kinetic density where the spatial slopes are large, resulting in the failure of Navier-Stokes solutions. In this case, the inaccurate prediction of flow regimes from $\mathrm{Kn}_{\mathrm{GLL}}$ results in unreasonable oscillations of macroscopic solutions, which is overcome by the neural network classifier.

We have seen that the adaptive solver has an accuracy advantage over the Navier-Stokes solver. In comparison with a full kinetic solver, it is advantageous in terms of computational effiency and memory footprint. In a continuum cell at t^n which holds the kinetic solution at t^{n-1}, the excess memory can be

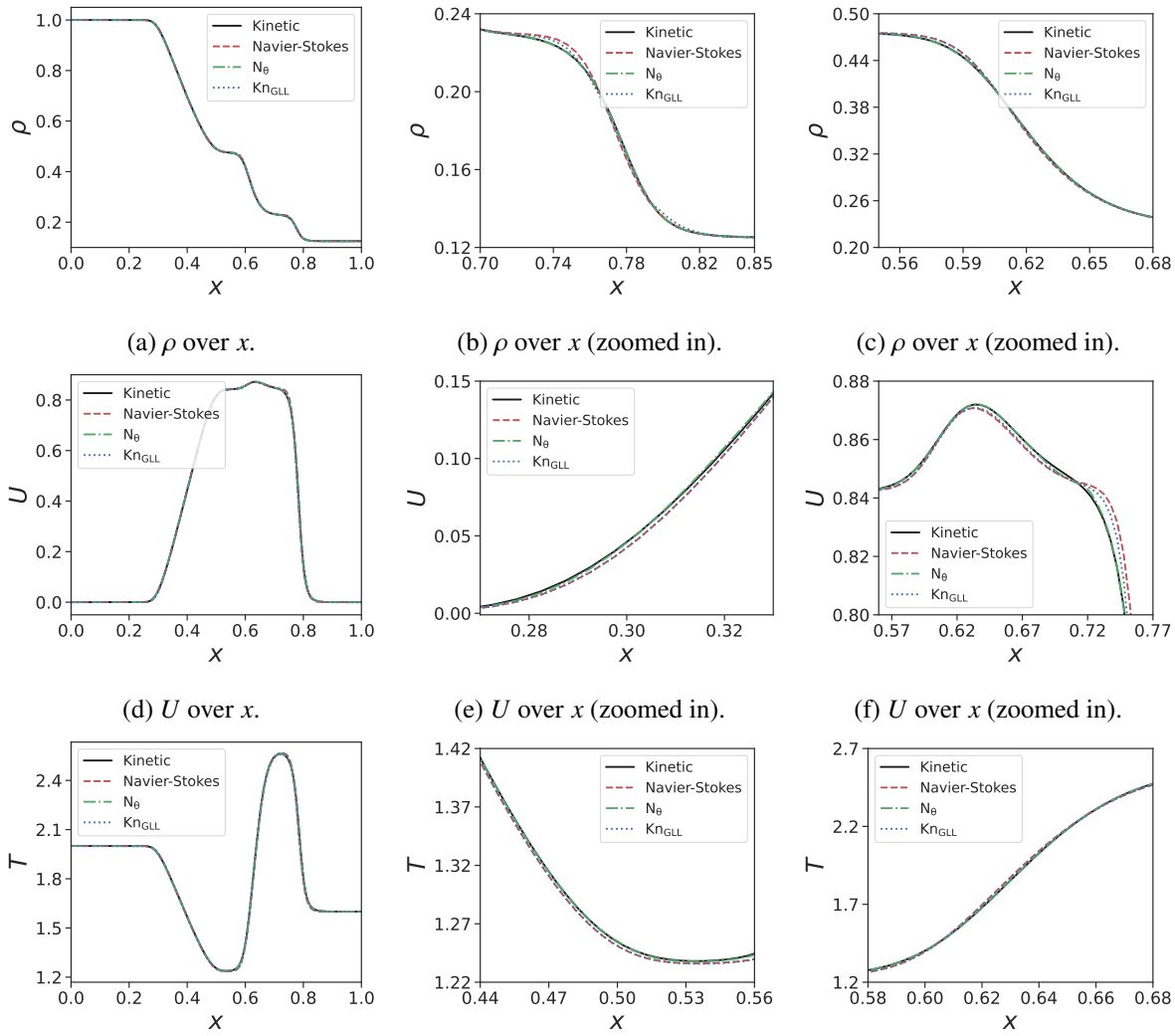


(a) ρ over x. (b) ρ over x (zoomed in). (c) ρ over x (zoomed in).

(d) U over x. (e) U over x (zoomed in). (f) U over x (zoomed in).

(g) T over x. (h) T over x (zoomed in). (i) T over x (zoomed in).

Figure 6.6.: Profiles of ρ, U and T in the shock tube at $t = 0.15$ under Kn = 1e−4. The KN_{GLL} solution aligns more with Navier-Stokes, whereas the neural network-based solution is close to the kinetic baseline.



(a) ρ over x.

(b) ρ over x (zoomed in).

(c) ρ over x (zoomed in).

(d) U over x.

(e) U over x (zoomed in).

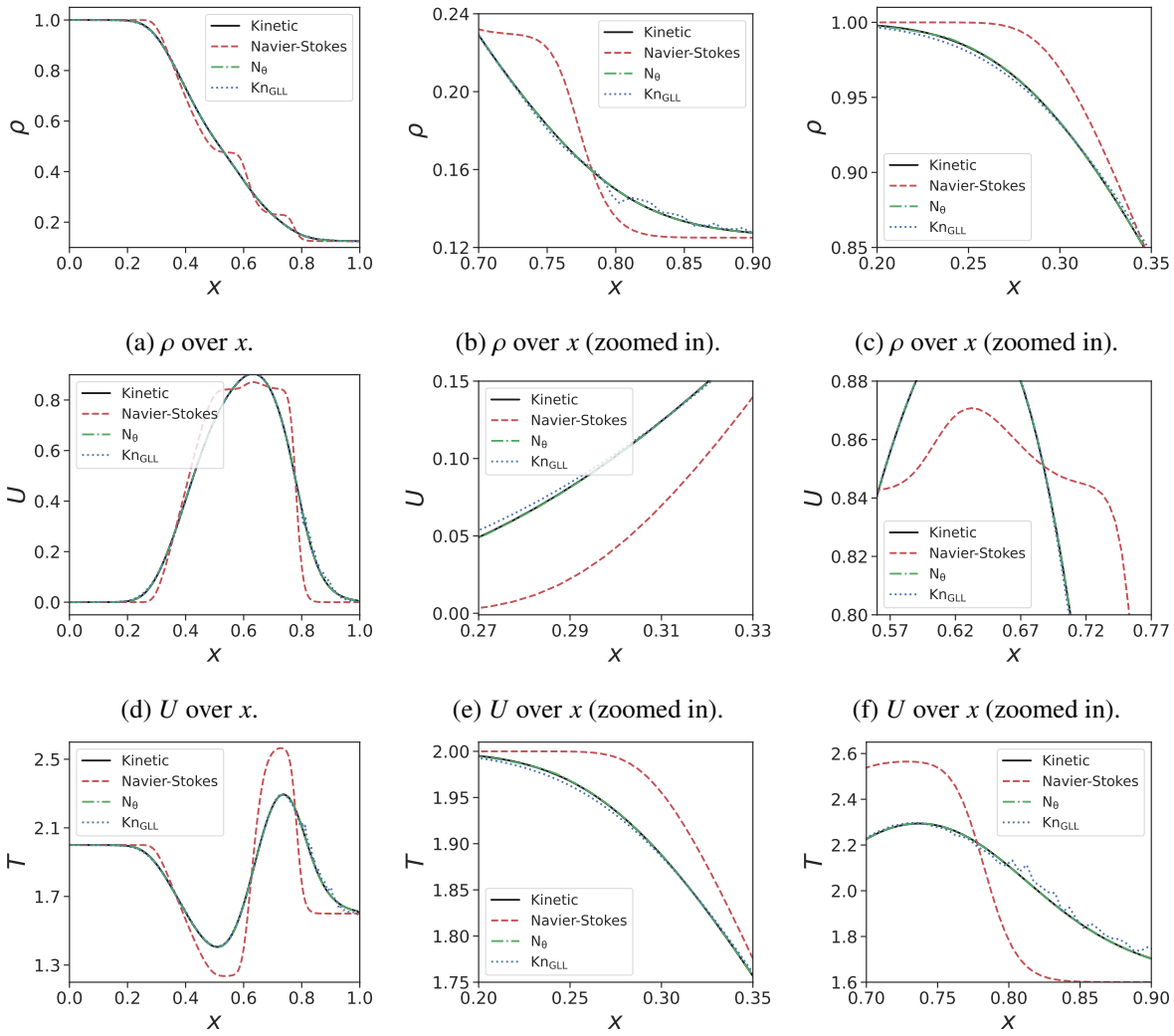
(f) U over x (zoomed in).

(g) T over x.

(h) T over x (zoomed in).

(i) T over x (zoomed in).

Figure 6.7.: Profiles of ρ, U and T in the shock tube at $t = 0.15$ under Kn = 1e−3. The KN_{GLL} solution aligns more with Navier-Stokes, whereas the neural network-based solution is close to the kinetic baseline.



(a) ρ over x. (b) ρ over x (zoomed in). (c) ρ over x (zoomed in).

(d) U over x. (e) U over x (zoomed in). (f) U over x (zoomed in).

(g) T over x. (h) T over x (zoomed in). (i) T over x (zoomed in).

Figure 6.8.: Profiles of density and temperature in the shock tube at $t = 0.15$ under Kn = 1e−2. The Navier-Stokes simulation is far from the kinetic baseline. The network-based hybrid simulation is more accurate than the KN_{GLL} based one, which oscillates in ρ and T.

Table 6.2.: Total number of memory allocations over the solver run-time and computational cost of the Sod shock tube problem.

	time [s]	total FLOP allocations	total allocated memory [GB]
Navier-Stokes	1.39e0	2.16e7	1.82e0
Kinetic	1.64e3	1.65e8	7.35e3
Neural network	5.14e2	5.57e7	9.48e2
Adaptive (Kn=0.0001)	9.75e1	2.72e7	1.21e2
Adaptive (Kn=0.001)	5.14e2	3.60e7	7.13e2
Adaptive (Kn=0.01)	1.21e3	9.88e7	3.88e3

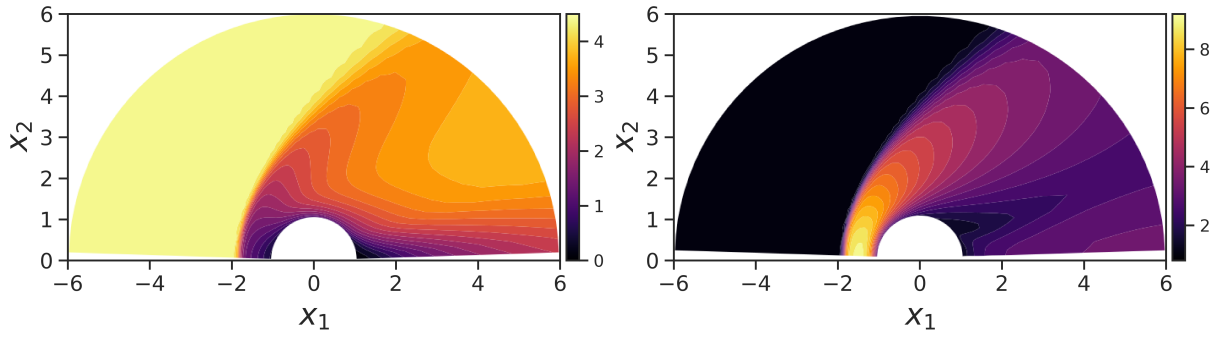
Algorithm 6.3: Workflow of steady flow problems

Converge the flow-field with the Navier-Stokes solver
Classify cell-wise the flow-regimes
Reconstruct the solution of kinetic cells
Converge the flow-field with the adaptive solver

deallocated. In a kinetic cell with no former record of the discretized distribution function, the solution is reconstructed from the Chapman-Enskog expansion in Eq. (6.15) in the continuum cell, and then used for flux evaluation. This way, a adaptive continuum-kinetic solver has been set up, where no buffer zone is required to transit solutions.
Table 6.2 provides the computational cost of the kinetic, Navier-Stokes, and adaptive solvers with different switching criteria. As can be seen, the adaptive scheme accelerates the simulation by 69%, and saves 66% of unnecessary allocations.

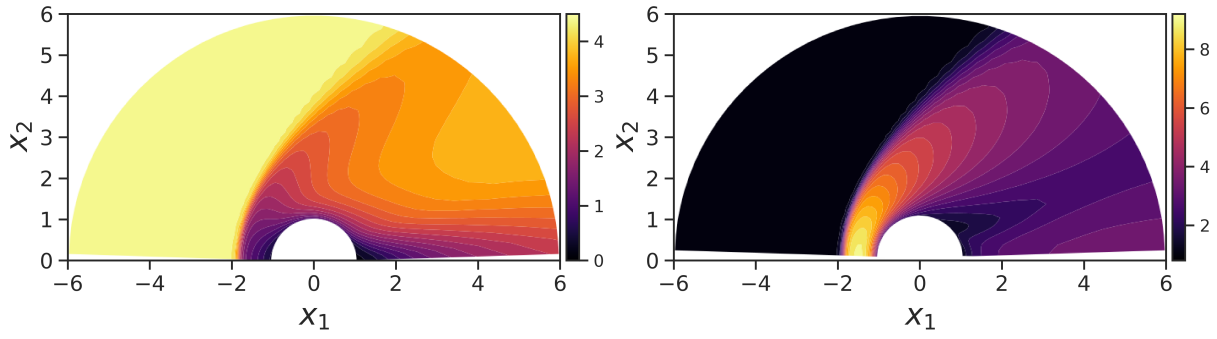
6.6.2. Flow Around a Circular Cylinder

In this numerical experiment, we simulate a two-dimensional hypersonic flow around a circular cylinder, where longitudinal and transverse processes coexist in the domain. The kinetic density is initialized as Maxwellian everywhere corresponding to the Mach number Ma = 5. Knudsen numbers are set as $\mathrm{Kn} \in \{0.001, 0.01\}$. The velocity domain is given by $\mathbf{V} = [-10, 10]^3$ and the computational domain is a polar grid with radius $r \in [1, 6]$ and angle ranging from $[0, \pi]$, i.e., the half plane surrounding a cylinder with radius 1. We impose Maxwell boundary conditions at the cylinder wall, symmetric conditions at the lower edge of the domain, and inflow conditions on the left side of the domain. The detailed computational setup is listed in Table 6.1.
In this steady-state problem, the computation can be accelerated with the help of the Navier-Stokes solver. A convergent coarse flow field can be first obtained by the Navier-Stokes solver, and then reconstructed as the initial state in the subsequent adaptive method, see Algorithm 6.3. We present the contours of horizontal velocity and temperature produced by the adaptive solver at Kn = 0.001 and Kn = 0.01 in Fig. 6.9. As shown, the bow shock and the expansion cooling region behind the cylinder are well captured. At Kn = 0.001, the cell size and time step in the computation are much larger than the particle mean free path and collision time, and all three methods result in a shock-capturing scheme. When the reference Knudsen number gets to Kn = 0.01, a larger particle mean free path leads to a wider shock structure. Due to the non-equilibrium gas dynamics in the shock wave and gas-surface interaction, a slight difference can be observed in the solutions provided by kinetic and Navier-Stokes solvers, where



(a) Kn = 0.001, U_1.

(b) Kn = 0.001, T.



(c) Kn = 0.01, U_1.

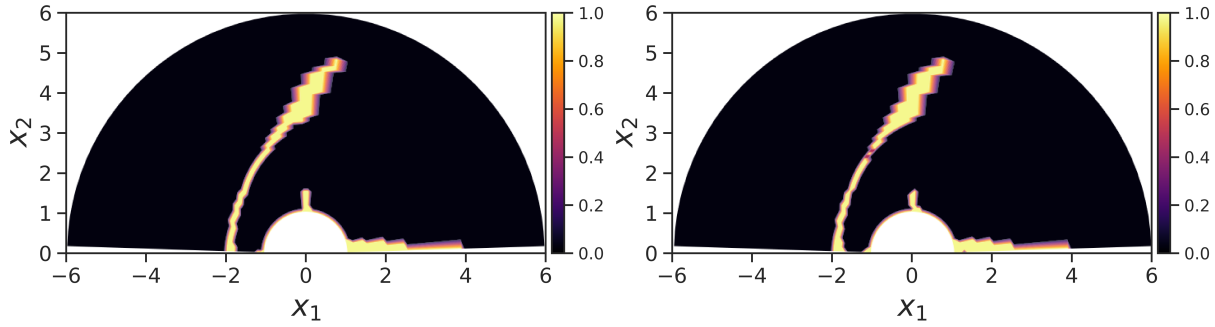
(d) Kn = 0.01, T.

Figure 6.9.: Profiles of U_1 and T of the cylinder flow with Kn = 0.001 (upper row) and Kn = 0.01 (lower row) computed with the neural network based adaptive solver. The bow shock and the expansion cooling region behind the cylinder are well captured.

the continuum scheme provides a narrower shock profile than the kinetic solution.
Based on the converged solution, the partition of flow regimes based on different criteria is shown in Fig. 6.10 for Kn = 0.001 and Kn = 0.01. Note that different critical values C are tested for the gradient-length-local Knudsen number criterion. For the commonly adopted value $C = 0.05$, $\mathrm{Kn}_{\mathrm{GLL}}$ underestimates the non-equilibrium effect and makes inaccurate predictions. After we reset it as $C = 0.01$, the predictions are still not precise enough. On the contrary, the neural network-based regime classification is close to the ground truth at different Knudsen numbers, especially at the shock bow. Differences can be observed at $x_1 = 0$ near the cylinder wall.
Figure 6.11 presents the quantitative comparison of solutions produced by the kinetic, Navier-Stokes, and the neural adaptive solver respectively at the cross-section $x_2 = 0$ from the left inflow-boundary to the cylinder wall at $x_1 = -1$. The neural network-based adaptive method switches to a kinetic regime at the shock in front of the cylinder and thus represents the full kinetic solution accurately, which confirms the validity of the neural network classifier in the two-dimensional case. On the other hand, a full Navier-Stokes solution yields differences from the kinetic solution, especially at the computation of the temperature T.

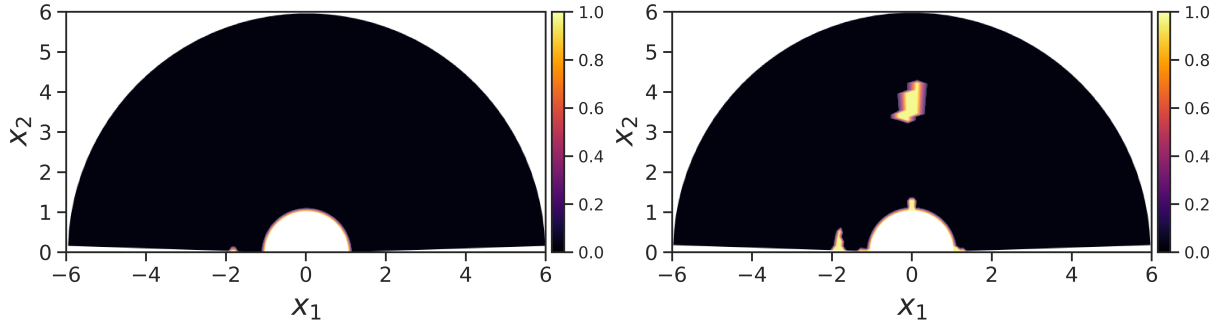
6.7. Chapter Conclusion

Gaseous flow is intrinsically a cross-scale problem due to the possible large variations of density and local Knudsen number. A quantitative criterion of continuum breakdown is crucial for developing sound flow theories and multi-scale solution algorithms.



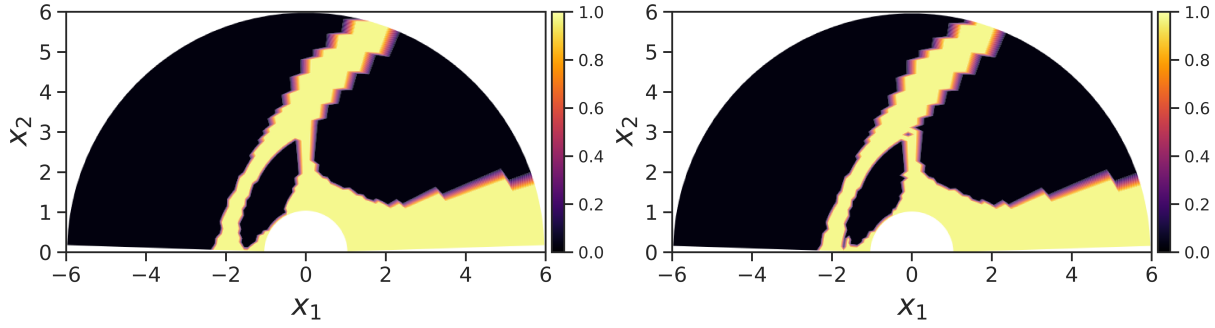
(a) Kn = 0.001, ground truth.

(b) Kn = 0.001, neural network.



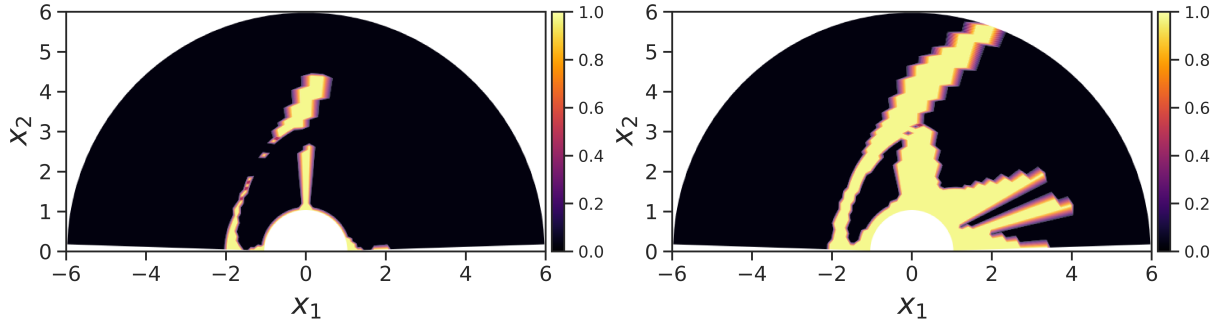
(c) Kn = 0.001, $\mathrm{Kn}_{\mathrm{GLL}}(C = 0.05)$.

(d) Kn = 0.001, $\mathrm{Kn}_{\mathrm{GLL}}(C = 0.01)$.



(e) Kn = 0.01, ground truth.

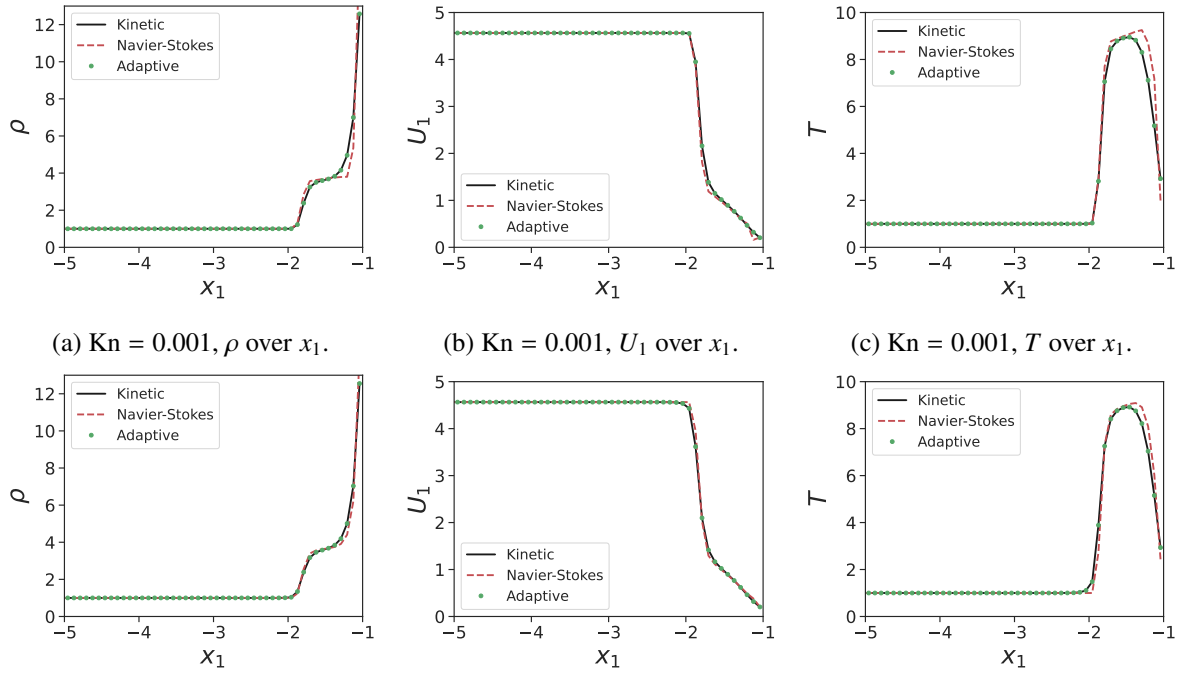
(f) Kn = 0.01, neural network.



(g) Kn = 0.01, $\mathrm{Kn}_{\mathrm{GLL}}(C = 0.05)$.

(h) Kn = 0.01, $\mathrm{Kn}_{\mathrm{GLL}}(C = 0.01)$.

Figure 6.10.: Flow regime prediction in the cylinder test case using different criteria at Kn = 0.001 (a)-(d), and Kn = 0.001 (c)-(h). Neural network-based predictions outperform $\mathrm{Kn}_{\mathrm{GLL}}$ based predictions in both cases and for different tuning parameters.



(a) Kn = 0.001, ρ over x_1. (b) Kn = 0.001, U_1 over x_1. (c) Kn = 0.001, T over x_1.

(d) Kn = 0.01, ρ over x_1. (e) Kn = 0.01, U_1 over x_1. (f) Kn = 0.01, T over x_1.

Figure 6.11.: Profile of macroscopic variables along the horizontal center line in front of a cylinder at Kn = 0.001 (upper row) and Kn = 0.01 (lower row). The neural network-based adaptive method yields solutions close to the kinetic benchmark, whereas the Navier-Stokes solver fails to capture the shock region accurately.

6.7.1. Summary

In this chapter, we developed the first neural network for binary classification of near-equilibrium and non-equilibrium flow regimes. This data-driven surrogate model provides an alternative to classical semi-empirical criteria and shows superiority in numerical experiments. Based on the minimal entropy closure of the Boltzmann moment system, an algorithmic strategy is designed to generate a dataset with a balanced distribution near and out of equilibrium state for model training and testing. An adaptive Boltzmann-Navier-Stokes flow solver is developed, which can dynamically adapt to local flow regimes using the neural network classifier. The current method provides an accurate and efficient tool for the study of cross-scale and non-equilibrium flow phenomena. It shows the potential to be extended to other complex systems, such as multi-component flows and plasma physics.

6.7.2. Limitations of the Approach

Neural network-based classifiers can only be as good as their data. Although we provide an alternative to biased simulation-based sampling, our data-generator has multiple user-defined inputs, such as the range of Knudsen numbers, macroscopic densities, choice of the velocity space, and most importantly, the threshold difference for the Kullback-Leibler Divergence from the equilibrium solution to determine the flow-regime.
The sampler removes simulation bias and abstracts human choices as setup of the flow-geometry and initial conditions, but does not remove them completely.

6.7.3. Future Work

Future research can be directed toward the creation of a more general sampling algorithm and the engineering of a more sophisticated neural network architecture for regime prediction. Currently, only a single grid cell is considered for the regime prediction. One could inspect the local neighborhoods of the current cell with a convolutional approach, graph-neural networks, or transformer-like architectures.

CHAPTER 7

DLRT: Dynamical Low-Rank Training for Efficient Neural Network Compression

Neural networks have achieved tremendous success in a large variety of applications. However, their memory footprint and computational demand can render them impractical in application settings with limited hardware or energy resources. In this chapter, we propose a novel algorithm to find efficient low-rank subnetworks. Remarkably, these subnetworks are determined and adapted already during the training phase and the overall time and memory resources required by both training and evaluating them are significantly reduced. The main idea is to restrict the weight matrices to a low-rank manifold and to update the low-rank factors rather than the full matrix during training. To derive training updates that are restricted to the prescribed manifold, we employ techniques from dynamic model order reduction for matrix differential equations. This allows us to provide approximation, stability, and descent guarantees. Moreover, our method automatically and dynamically adapts the ranks during training to achieve the desired approximation accuracy. The efficiency of the proposed method is demonstrated through a variety of numerical experiments on fully-connected and convolutional networks.

7.1. Introduction

While showing great performance in terms of classification records, most state-of-the-art neural networks require an enormous amount of computation and memory storage both for the training and the evaluation phases [116]. These requirements not only increase infrastructure costs and energy consumption but also prohibit deployment of artificial neural networks to infrastructures with limited resources such as mobile phones or smart devices. On the other hand, it is well-known that networks' weights contain structures and redundancies that can be exploited for reducing the parameter space dimension without significantly affecting the overall accuracy [22, 45, 76, 150].
Network pruning is a popular line of research that addresses this problem by removing redundant parameters from pre-trained models. Typically, the initial network is large and accurate, and the goal is

to produce a smaller network with similar accuracy. Methods within this area include weight sparsification [93, 107, 186] and quantization [238, 48], with different pruning techniques, including search-based heuristics [107], reinforcement learning [12, 106] and genetic algorithms [170]. More recent work has considered pruning during training, by formulating pruning as a data-driven optimization problem [93, 111, 116]. The resulting "dynamical pruning" boils down to a parameter-constrained training phase which, however, has been mostly focused on requiring sparse or binary weights so far.
Rather than enforcing sparsity or binary variables, in this work, we constrain the parameter space to the manifold of low-rank matrices. Neural networks' parameter matrices and large data matrices, in general, are seldom full rank [214, 228, 181, 70]. Constraining these parameters to lie on a manifold defined by low-rank matrices is thus a quite natural approach. By interpreting the training problem as a continuous-time gradient flow, we propose a training algorithm based on the extension of recent Dynamical Low-Rank Approximation (DLRA) algorithms [35, 36, 37]. This approach allows us to use low-rank numerical integrators for matrix-valued Ordinary Differential Equations (ODEs) to obtain modified forward and backward training phases that only use the small-rank factors in the low-rank representation of the parameter matrices and that are stable with respect to small singular values. This is a striking difference to recent alternative "vanilla" low-rank training schemes [233, 130] which simply factorize the weight matrices as the product of two low-rank factors $UV^\top$ and apply a descent algorithm alternatively on the two variables U and V.
We perform several experimental evaluations showing that the resulting dynamical low-rank training paradigm yields low-parametric neural network architectures. Compared to their full-rank counterparts they are both remarkably less demanding in terms of memory storage and require much less computational cost to be trained. Moreover, the trained low-rank neural networks achieve comparable accuracy to the original full architecture. This observation is reminiscent of the so-called lottery tickets hypothesis — dense neural networks contain sparse subnetworks that achieve high accuracy [76] — and suggests the presence of *low-rank winning tickets*: highly-performing low-rank subnetworks of dense networks. Remarkably, our dynamical low-rank training strategy seems to be able to find the low-rank winning tickets directly during the training phase independent of initialization.

7.1.1. Related Work on Low-Rank Methods

Low-rank factorization using the SVD and other matrix decomposition techniques have been extensively studied in the scientific computing and machine learning communities. The challenge of compressing and speeding up large-scale neural networks using low-rank methods has sparked widespread research interest in recent years and significant effort has been put towards developing low-rank factorization strategies for deep neural networks.
Previous works can roughly be categorized into approaches with fixed low-rank and variable low-rank during training time. Fixed rank approaches decompose weight matrices using SVD or tensor decompositions of pre-trained networks and fine-tune the factorized network [57, 155, 205, 226], constrain weight matrices to have a fixed low-rank during training [120, 233, 130], or create layers as a linear combination of layers of different rank [118]. Hence, these methods introduce the rank of the matrix decomposition as another hyperparameter to be fine-tuned. Rank-adaptive methods mitigate this issue by automatic determination and adaption of the low-rank structure after training. In particular, [133, 134] apply heuristics to determine the rank of the matrix decomposition ahead of time, whereas [235] encourages low-rank weights via a penalized loss that depends on approximated matrix ranks.
Few methods have been proposed recently that adapt the ranks of the weight matrix alongside the main network training phase. In [165], the authors set up the neural network training as a constrained optimization problem with an upper bound on the ranks of the weights, which is solved in an alternating

approach resulting in an NP-hard mixed integer program. The authors of [117] formulate a similar constrained optimization problem resulting in a mixed discrete-continuous optimization scheme that jointly addresses the ranks and the elements of the matrices. However, both these approaches require knowledge of the full weight matrix (and of its singular value decomposition) during training and overall are more computationally demanding than standard training.

7.1.2. Novelty and Scientific Contribution

In this work, we overcome the above issues and propose a training algorithm with reduced memory and computational requirements. To this end, we reinterpret the optimization problem of a neural network as a gradient flow of the network weight matrices and thus as a matrix-valued ODE. This continuous formulation allows us to use recent advances in DLRA methods for matrix ODEs which aim at evolving the solution of the differential equation on a low-rank manifold. The main idea of DLRA [136], which originates from the Dirac-Frenkel variational principle [58, 77], is to approximate the solution through a low-rank factorization and derive evolution equations for the individual factors. Thereby, the full solution does not need to be stored and the computational costs can be significantly reduced. To ensure the robustness of the method, stable integrators have been proposed in [173] and [37]. Instead of evolving individual low-rank factors in time, these methods evolve products of low-rank factors, which yields remarkable stability and exactness properties [132], both in the matrix and the tensor settings [137, 175, 174, 39]. In this work, we employ the "unconventional" basis update & Galerkin step integrator [37] as well as its rank-adaptive extension [35], see also [148, 38]. The rank-adaptive unconventional integrator chooses the approximation ranks according to the continuous-time training dynamics and allows us to find highly-performing low-rank subnetworks directly during the training phase while requiring reduced training cost and memory storage.
This chapter is published in the proceedings of the NeurIPS 2022 conference [209] and is a collaborative work with Emanuele Zangrando, Gianluca Ceruti, Jonas Kusch, and Francesco Tudisco. These numerical experiments are supplemented by the open-source DLRT implementations in Tensorflow[1] and PyTorch[2].

7.1.3. The Chapter in Context of the Dissertation

This chapter poses somewhat of a counterpart to the rest of the dissertation where we focus on using neural networks and machine learning methods to construct efficient surrogate models to compute solutions of kinetic equations. Now we turn our attention to the creation of efficient neural network training methods, using numerical methods for kinetic systems. This chapter is not only a counterpart but a complement to its predecessors: First, we give an example of how two mathematical fields can supplement each other nicely. Second, DLRT and pruning methods are highly relevant for the construction of efficient neural network-based surrogate models, for the whole point of their existence is to provide the best accuracy-performance tradeoff. Future work may push the boundaries of high-order neural-network-based closures even further with the help of large DLRT-trained models.

[1] https://github.com/CSMMLab/DLRTNet
[2] https://github.com/COMPiLELab/DLRT

7.1.4. Organization of the Chapter

We provide a brief overview of the Dynamical Low-Rank Approximation of matrix-valued dynamical systems in §7.2, explain how the method transfers to neural network training, and derive the system ODEs for dynamical low-rank-training with rank adaption. In §7.3 we develop the DLRT algorithm for efficient, dynamical, rank-adaptive low-rank training of neural networks, and conduct error and convergence analysis for the new method. We discuss implementation details and their effect on computational and memory costs. In §7.4, we discuss the applicability of DLRT to other layer architectures as transformers and convolutional layers. Lastly, §7.5 provides a wide array of numerical experiments to validate the method on different benchmarks with popular network architectures.

7.2. Dynamical Low-Rank Approximation

Consider the solution $A \in \mathcal{G}$ of a dynamical system

$$\dot{A}(t) = \mathcal{F}(A, t), \tag{7.1}$$

where $\mathcal{G}$ is a potentially high dimensional space and $\dot{A}$ denotes the time derivative. Dynamical low-rank seeks to approximate the true solution A by a low-rank representation $Y \in \mathcal{M} \subset \mathcal{G}$, where $\mathcal{M}$ is a low-rank manifold. A best approximation $Y(t) \in \mathcal{M}$ to $A(t)$ satisfies

$$Y(t) \in \mathcal{M} \qquad \text{such that} \qquad \|Y(t) - A(t)\| = \min \tag{7.2}$$

Consider the case of matrices $A(t) \in \mathbb{R}^{n_1 \times n_2}$. Then $Y(t) \in \mathbb{R}^{n_1 \times n_2}$ is a low-rank matrix with rank $r \ll n_1, n_2$. It has been shown [207], that in this case, a truncated SVD solves the optimization problem (7.2), however only in a point-wise fashion in time.
We are interested in a method, that yields a time evolution of $Y(t)$ by working only with the factors of the truncated SVD. If $T_{Y(t)}\mathcal{M}$ is the tangent space of $\mathcal{M}$ at $Y(t)$, we seek to determine

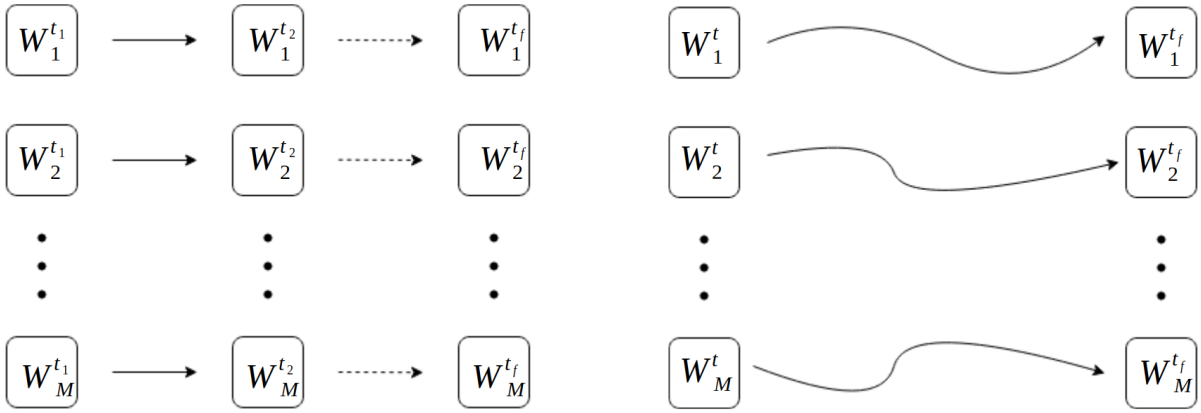
$$\dot{Y}(t) \in T_{Y(t)}\mathcal{M}, \qquad \text{such that} \qquad \left\|\dot{Y} - \dot{A}\right\| = \min . \tag{7.3}$$

A stable numerical scheme to solve this equation is the projector splitting integrator proposed in [176].
Beyond the original application area of quantum dynamics, dynamical low-rank approximation [136] has recently gained significant interest in several communities such as kinetic theory [136, 65, 198, 66, 197, 92, 64] and uncertainty quantification [71, 145, 206]. The increasing interest in dynamical low-rank approximation stems mainly from its low memory requirements, which tackle the core issue of modern HPC architectures. Furthermore, the significantly reduced computational costs of dynamical low-rank approximation enable computing accurate numerical results for complex problems even with limited computational resources. Two core ingredients of dynamical low-rank approximation are the representation of the solution as a low-rank matrix or tensor factorization (e.g. a truncated singular value decomposition for matrices) and the derivation of evolution equations for the individual factors of this low-rank representation.

7.2.1. Low-Rank Training via Gradient Flow

Consider a feed-forward fully-connected neural network

$$\begin{aligned} \mathrm{N}_\theta(\mathbf{x}) &= \mathbf{z}_M \\ \mathbf{z}_k &= \sigma_k(W_k \mathbf{z}_{k-1} + \boldsymbol{b}_k), \qquad \forall k = 1, \dots, M \\ \mathbf{z}_0 &= \mathbf{x}, \end{aligned} \tag{7.4}$$



(a) Discrete time weight update (b) Continuous gradient flow

Figure 7.1.: Graphical re-interpretation of the discrete weight update step (a) as a time-continuous process (b).

with weights-matrices $W_k \in \mathbb{R}^{n_k, n_{k-1}}$ and biases $\mathbf{b}_k \in \mathbb{R}^{n_k}$ as the set of weights θ and $\mathbf{z}_k$ as layer outputs. The convolutional and transformer setting is discussed in §7.4. We consider the training of N_θ based on the optimization of a loss function $\mathcal{L}(\theta; \mathrm{N}_\theta(\mathbf{x}), \mathbf{y})$ via a gradient-based descent algorithm. For example, when using gradient descent, the weight matrix W_k of N_θ at iteration $t \in \mathbb{N}$ is updated via

$$W_k^{t+1} = W_k^t - \lambda \nabla_{W_k} \mathcal{L}(\theta; \mathrm{N}_\theta(\mathbf{x}), \mathbf{y}) \qquad \forall \mathrm{k} = 1, \ldots, \mathrm{M} \tag{7.5}$$

with a learning rate λ. When the weight matrices W_k are dense, both the forward and gradient evaluations of the network require a large number of full matrix multiplications, with a high computational expense and large memory footprint. This renders the training and the use of large-scale neural networks a difficult challenge on limited-resource devices. At the same time, a wealth of evidence shows that dense networks are typically over-parameterized and that most of the weights learned this way are unnecessary [181, 70]. To reduce the memory and computation costs of training, we propose a method that performs the minimization over the manifold of low-rank matrices, based on DLRA theory for dynamical systems. Minimizing the loss function for W_k is equivalent to evaluating the long time behavior of the following matrix ODE that allows us to interpret the training phase as a continuous process, discretized with time-step λ in (7.5), i.e.,

$$\dot{W}_k(t) = -\nabla_{W_k} \mathcal{L}(\theta; \mathrm{N}_\theta(\mathbf{x}), \mathbf{y}), \tag{7.6}$$

The process is illustrated in Fig 7.1. We call the right-hand side

$$\mathcal{F}_k(W, t) = -\nabla_{W_k} \mathcal{L}(\theta; \mathrm{N}_\theta(\mathbf{x}), \mathbf{y}) \tag{7.7}$$

the gradient flow of the network with respect to layer W_k.
Let $\mathcal{M}_{r_k}$ denote the manifold of matrices with rank r_k and assume at a certain time t_0 the weights are in the manifold In the sense of Eq. (7.2) and (7.3), we identify $W_k(t)$ with $A(t)$, and propose a set of low-rank weight matrices $Y_k(t)$, with $Y_k(t_0) \in \mathcal{M}_{r_k}$. Using the continuous-time interpretation allows us to derive a low-rank strategy to evolve the weights according to the dynamics in (7.6) with

$$Y_k(t) \in \mathcal{M}_{r_k} \qquad \text{such that} \qquad \|Y_k(t) - W_k(t)\| = \min, \qquad \text{for } \mathrm{t} \geq \mathrm{t}_0. \tag{7.8}$$

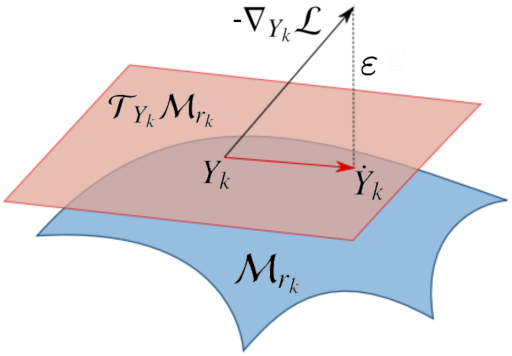


Figure 7.2.: Orthogonal projection onto the tangent space of the low-rank manifold $\mathcal{M}_r$. The dashed line depicts the projection resulting in $\dot{Y}_k(t)$, which is the tangent element minimizing the distance between $\nabla_{Y_k}\mathcal{L}(Y_k(t))$ and the tangent space $\mathcal{T}_{Y_k(t)}\mathcal{M}_r$ at the approximation $Y_k(t)$.

So we assume that the ideal W_k can be well-approximated by a matrix Y_k of rank $r_k \ll n_k, n_{k+1}$ of the form

$$W_k \approx Y_k = U_k S_k V_k^\top \in \mathbb{R}^{n_k \times n_{k-1}} \tag{7.9}$$

where $U_k \in \mathbb{R}^{n_k \times r_k}$, $V_k \in \mathbb{R}^{n_{k-1} \times r_k}$ are thin and tall matrices having orthonormal columns spanning optimal subspaces which capture essential properties of parameters, and $S_k \in \mathbb{R}^{r_k \times r_k}$ is a tiny full-rank matrix that allows us to extrapolate the useful information from the learned subspaces U_k and V_k.
Traditional descent algorithms applied individually to the factors U_k, S_k and V_k do not guarantee the preservation of the low-rank structure $U_k S_k V_k^\top$ when updating the weights during training or require knowledge of the whole Y_k rather than the factors U_k, S_k, V_k to couple the dynamics of the low-rank system.

7.2.2. Coupled Dynamics of the Low-Rank Factors via DLRA

We consider the dynamical system of a single weight matrix W_k, while the remaining weight matrices are fixed in time and are treated as parameters for the gradient. In the following, we omit writing these parameters down for the sake of efficiency. Assuming $Y_k(t) \in \mathcal{M}_{r_k}$, we can formulate (7.6) as

$$\min \left\{ \|\dot{Y}_k(t) + \nabla_{Y_k}\mathcal{L}(Y_k(t))\|_F \ : \ \dot{Y}_k(t) \in \mathcal{T}_{Y_k(t)}\mathcal{M}_{r_k} \right\} \tag{7.10}$$

where $\mathcal{T}_{Y_k(t)}\mathcal{M}_{r_k}$ is the tangent space of $\mathcal{M}_{r_k}$ at position $Y_k(t)$, $\|\cdot\|_F$ is the Frobenius norm and $\mathcal{F}_k$ denotes the gradient flow of the loss with respect to the k-th matrix variable. In order to solve (7.10), we further observe that (7.10) can be equivalently formulated as the following Galerkin condition [136], i.e.,

$$\langle \dot{Y}_k(t) + \nabla_{Y_k}\mathcal{L}(Y_k(t)), \delta Y_k \rangle = 0 \qquad \forall \delta Y_k \in \mathcal{T}_{Y_k(t)}\mathcal{M}_{r_k} \ , \tag{7.11}$$

where $\langle \cdot, \cdot \rangle$ is the standard scalar product. The Galerkin condition is illustrated in Fig. 7.2. From $Y_k = U_k S_k V_k^\top$, a generic element δY_k of the tangent space $\mathcal{T}_{Y_k(t)}\mathcal{M}_{r_k}$ can be written as

$$\delta Y_k = \delta U_k S_k V_k^\top + U_k \delta S_k V_k^\top + U_k S_k \delta V_k^\top \ , \tag{7.12}$$

where δU_k and δV_k are generic elements of the tangent space of the Stiefel manifold with r_k orthonormal columns at the points U_k and V_k, respectively, and δS_k is a generic $r_k \times r_k$ matrix, see e.g. [136, §2] for

details. Additionally, the Gauge conditions $U_k^\top \delta U_k = 0$ and $V_k^\top \delta V_k = 0$ must be imposed to ensure orthogonality of the basis matrices, and the uniqueness of the representation of the tangent space elements. Similarly, by the chain rule applied several times we have

$$\dot{Y}_k = \frac{\mathrm{d}}{\mathrm{d}t}\{U_k S_k V_k^\top\} = \dot{U}_k S_k V_k^\top + U_k \dot{S}_k V_k^\top + U_k S_k \dot{V}_k^\top . \tag{7.13}$$

Now, the Galerkin condition (7.11) becomes

$$\langle \dot{U}_k S_k V_k^\top + U_k \dot{S}_k V_k^\top + U_k S_k \dot{V}_k^\top + \nabla_{Y_k} \mathcal{L}(Y_k(t)), \delta Y_k \rangle = 0, \qquad \forall \delta Y_k \in \mathcal{T}_{Y_k(t)} \mathcal{M}_{r_k} \tag{7.14}$$

with $U_k^\top \dot{U}_k = 0$ and $V_k^\top \dot{V}_k = 0$. If we choose $\delta Y_k = U_k \delta S_k V_k^\top$ in (7.14), we obtain

$$\langle U_k^\top \dot{U}_k S_k V_k^\top V_k + U_k^\top U_k \dot{S}_k V_k^\top V_k + U_k^\top U_k S_k \dot{V}_k^\top V_k + U_k^\top \nabla_{Y_k} \mathcal{L}(Y_k(t)) V_k, \delta S_k \rangle = 0 .$$

Thus, using the Gauge conditions, we obtain $\langle \dot{S}_k + U_k^\top \nabla_{Y_k} \mathcal{L}(Y_k(t)) V_k, \delta S_k \rangle = 0$, which has to hold for a generic $r_k \times r_k$ matrix δS_k. We obtain this way an evolution equation for the $S_k(t)$ factor. Similarly, specifying (7.14) for the two choices $\delta Y_k = \delta U_k S_k V_k^\top$ and $\delta Y_k = U_k S_k \delta V_k^\top$, allows us to obtain the following system of differential equations for the individual factors of Y_k, i.e.,

$$\begin{cases} \dot{S}_k = -U_k^\top \nabla_{Y_k} \mathcal{L}(Y_k(t)) V_k , \\ \dot{U}_k = -(I - U_k U_k^\top) \nabla_{Y_k} \mathcal{L}(Y_k(t)) V_k S_k^{-1} , \\ \dot{V}_k = -(I - V_k V_k^\top) \nabla_{Y_k} \mathcal{L}(Y_k(t))^\top U_k S_k^{-\top} , \end{cases} \tag{7.15}$$

where I is the identiy matrix.

7.3. Dynamical Low-Rank Training

To perform an efficient and robust rank-constrained training step, we numerically integrate the system of ODEs (7.15). Our approach is based on the "unconventional KLS integrator" [37] and its rank-adaptive version [35]. The pseudocode of the proposed training strategy is presented in Algorithm 7.1.
The main idea of the DLRT algorithm is to alternately represent the product $Y_k = U_k S_k V_k^\top$ as $Y_k = K_k V_k^\top$ and $Y_k = U_k L_k^\top$, consider the corresponding coupled ODEs from (7.15), and then perform three main steps:

- **K&L-steps** (in parallel). Update the current K_k and L_k by integrating the differential equations

$$\begin{cases} \dot{K}_k(t) = -\nabla_{W_k} \mathcal{L}(K_k(t) V_k^\top) V_k, \quad K_k(0) = U_k S_k, \\ \dot{L}_k(t) = -\nabla_{W_k} \mathcal{L}(U_k L_k(t)^\top)^\top U_k, \quad L_k(0) = V_k S_k^\top, \end{cases} \tag{7.16}$$

  from $t = 0$ to $t = \eta$; then form new orthonormal basis matrices $\widetilde{U}_k$ and $\widetilde{V}_k$ spanning the range of the computed $K_k(\eta)$ and $L_k(\eta)$.

- **S-step**. Update the current S_k by integrating the differential equation

$$\dot{S}_k(t) = -\widetilde{U}_k^\top \nabla_{W_k} \mathcal{L}(\widetilde{U}_k S_k(t) \widetilde{V}_k^\top) \widetilde{V}_k \tag{7.17}$$

  from $t = 0$ to $t = \eta$, with initial value condition $S_k(0) = \widetilde{U}_k^\top U_k S_k V_k^\top \widetilde{V}_k$.

An important feature of this algorithm is that it can be extended to rank adaptivity in a relatively straightforward manner [35], letting us dynamically evolve the rank of S_k (and thus the rank of Y_k) during the computation. This is particularly useful, as we may expect the weight matrices to have low ranks but we may not know what the "best" ranks for each layer are. Typically, dynamically adapting the ranks of a low-rank optimization scheme is a challenging problem as moving from the manifold $\mathcal{M}_{r_k}$ to $\mathcal{M}_{r_k \pm 1}$ introduces singular points [80, 2]. Instead, treating the training problem as a system of matrix differential equations allows us to overcome this issue with a simple trick: at each step of the KLS integrator we double the dimension of the basis matrices $\widetilde{U}_k$ and $\widetilde{V}_k$ computed in the K- and L-steps by computing orthonormal bases spanning $[K_k(\eta) \mid U_k]$ and $[L_k(\eta) \mid V_k]$, respectively, i.e. by augmenting the current basis with the basis computed in the previous time step. Then, after the new S_k matrix is computed via the S-step, a truncation step is performed, removing from the newly computed S_k matrix all the singular values that are under a certain threshold ϑ.
Of course, adding rank adaptivity to the integrator comes at a cost. In that case, each step requires performing an SVD decomposition twice the size of the current rank of S_k to determin a threshold for the singular values. Moreover, the dimension of the bases U_k and V_k may grow, which also may require additional computational effort. However, if the ranks remain small throughout the dynamics, this computational overhead is negligible, as we will further discuss in §7.3.3 and §7.5.

7.3.1. Error Analysis and Convergence

In this section we present our main theoretical results, showing that (a) the low-rank matrices Y_k formed by the weights' factors $U_k S_k V_k^\top$ computed with Algorithm 7.1 are close to the true solution W_k of (7.6), and (b) that the loss function decreases during DLRT, provided the singular value threshold ϑ is not too large, i.e., is bounded by a constant times the square of the time-step size η (see Theorem 1). In the version we present here, part of the statements are presented informally for the sake of brevity. We refer to the supplementary material §7.7 for details and the proofs.

Theorem 7.1 (Low-Rank Approximation Error)
Assume the gradient flow $\mathcal{F}_k(Z) = -\nabla_{W_k}\mathcal{L}(W_1, \dots, Z, \dots, W_M, \mathrm{N}_\theta(\mathbf{x}), \mathbf{y})$ *in* (7.6) *is locally bounded and locally Lipschitz continuous, with constants* C_1 *and* C_2*, respectively. Then, for fixed* x *and* y*, let* $W_k(t)$ *be the (full-rank) continuous-time solution of* (7.6) *and let* $Y_k = U_k, S_k, V_k$ *be the factors computed with Algorithm 7.1 after* t *steps. Assume that the K, L, S steps* (7.16) *and* (7.17) *are integrated exactly from* 0 *to* η*. Assume moreover that, for any* $Z \in \mathcal{M}_{r_k}$ *sufficiently close to* $W_k(t\eta)$*, the whole gradient flow* $\mathcal{F}_k(Z)$ *is "*ε*-close" to* $\mathcal{M}_{r_k}$*. Then,*

$$\|U_k S_k V_k^\top - W_k(t\eta)\|_F \leq c_1\varepsilon + c_2\eta + c_3\vartheta/\eta \qquad k = 1, \dots, M \tag{7.18}$$

where the constants c_1*,* c_2 *and* c_3 *depend only on* C_1 *and* C_2*. In particular, the approximation bound does not depend on the singular values of the exact nor the approximate solution.*

Observe that, while the loss function $\mathcal{L}$ decreases monotonically along any continuous-time solution $W_k(t)$ of (7.6), it is not obvious that the loss decreases when the integration is done onto the low-rank manifold via Algorithm 7.1. The next result shows that this is indeed the case, up to terms of the order of the truncation tolerance ϑ. More precisely, we have

Theorem 7.2 (Monotonicity of DLRT)
Let $Y_k^t = U_k^t S_k^t (V_k^t)^\top$ *be the low-rank weight matrix computed at step* t *of Algorithm 7.1 and let* $\mathcal{L}(t) =$

Algorithm 7.1: Dynamic Low Rank Training Scheme (DLRT)

Input: $S_k^0 \sim r_k^0 \times r_k^0$; $U_k^0 \sim n_k \times r_k^0$;$V_k^0 \sim n_{k-1} \times r_k^0$ for $k = 1, \dots, M$: Initial low-rank factors
`iter`: maximal number of descent iterations per epoch
`adaptive`: Boolean flag that decides whether or not to dynamically update the ranks
ϑ: singular value threshold for adaptive procedure

Result: Trained and compressed low-rank network

for *each* `epoch` **do**
  for $t = 0$ *to* $t =$ `iter` **do**
    for *each layer* k **do**
      $K_k^t \leftarrow U_k^t S_k^t$ `/* K-step */`
      $K_k^{t+1} \leftarrow$ one-step-integrate$\{\dot{K}(t) = -\nabla_K \mathcal{L}(K(t)(V_k^t)^\top z_{k-1} + b_k^t),\ K(0) = K_k^t\}$
      $L_k^t \leftarrow V_k^t (S_k^t)^\top$ `/* L-step */`
      $L_k^{t+1} \leftarrow$ one-step-integrate$\{\dot{L}(t) = -\nabla_L \mathcal{L}(U_k^t L(t)^\top z_{k-1} + b_k^t),\ L(0) = L_k^t\}$
      if `adaptive` **then** `/* Basis augmentation step */`
        $K_k^{t+1} \leftarrow [K_k^{t+1} \mid U_k^t]$
        $L_k^{t+1} \leftarrow [L_k^{t+1} \mid V_k^t]$
      $U_k^{t+1} \leftarrow$ orthonormal basis for the range of K_k^{t+1} `/* S-step */`
      $M_k \leftarrow (U_k^{t+1})^\top U_k^t$
      $V_k^{t+1} \leftarrow$ orthonormal basis for the range of L_k^{t+1}
      $N_k \leftarrow (V_k^{t+1})^\top V_k^t$
      $\widetilde{S}_k^t \leftarrow M_k S_k^t N_k^\top$
    for *each layer* k **do**
      $S_k^{t+1} \leftarrow$ one-step-integrate$\{\dot{S}(t) = -\nabla_S \mathcal{L}(U_k^{t+1} S(t) (V_k^{t+1})^\top z_{k-1} + b_k^t),\ S(0) = \widetilde{S}_k^t\}$
      if `adaptive` **then** `/* Rank compression step */`
        $P, \Sigma, Q \leftarrow \mathrm{SVD}(S_k^{t+1})$
        $S_k^{t+1} \leftarrow$ truncate Σ using the singular value threshold ϑ
        $U_k^{t+1} \leftarrow U_k^{t+1} \widetilde{P}$ where $\widetilde{P} =$ [first r_k^{t+1} columns of P]
        $V_k^{t+1} \leftarrow V_k^{t+1} \widetilde{Q}$ where $\widetilde{Q} =$ [first r_k^{t+1} columns of Q]
      `/* Bias update step */`
      $b_k^{t+1} \leftarrow$ one-step-integrate$\{\dot{b}(t) = -\nabla_b \mathcal{L}(U_k^{t+1} S_k^{t+1} (V_k^{t+1})^\top z_{k-1} + b(t)),\ b(0) = b_k^t\}$

$\mathcal{L}(Y_1^t, \dots, Y_M^t, \mathrm{N}_\theta(\mathbf{x}), \mathbf{y})$. *Then, for a small enough time-step* η *we have*

$$\mathcal{L}(t+1) \leq \mathcal{L}(t) - \alpha\eta + \beta\vartheta \tag{7.19}$$

where α *and* β *are positive constants that do not depend on* t, η *and* ϑ.

7.3.2. Efficient Implementation of the Gradients

All three K, L, and S-steps require the evaluation of the gradient flow of the loss function with respect to the whole matrix W_k. Different approaches to efficiently compute this gradient may be used. The strategy we discuss below aims at reducing memory and computational costs by avoiding the computation of the full gradient, working instead with the gradient with respect to the low-rank factors.
To this end, we note that for the K-step

$$\nabla_{W_k} \mathcal{L}(K_k(t)V_k^\top)V_k = \nabla_{K_k} \mathcal{L}(K_k(t)V_k^\top) \tag{7.20}$$

holds Hence, the whole gradient can be computed through a forward run of the network with respect to K_k

$$\mathbf{z}_k = \boldsymbol{\sigma}_k\left(K_k(t)V_k^\top \mathbf{z}_{k-1} + \mathbf{b}_k\right), \qquad k = 1, \dots, M \tag{7.21}$$

and taping the gradient with respect to K_k using automatic differentiation. In this way, the full gradient does not need to be computed and the overall computational costs are comprised of running a forward evaluation while taping gradients with respect to K_k, analogously to the traditional back-propagation algorithm. The L- and S-steps can be evaluated efficiently in the same manner, by evaluating the network while taping the gradients with respect to L_k and S_k, respectively. Hence, instead of a single gradient tape (or chain rule evaluation) of the full-weight matrix network, we have three gradient tapes, one for each low-rank step, whose combined computational footprint is less than the full matrix tape. We provide detailed formulas for all three gradient tapes in the supplementary material §7.7.

7.3.3. Implementation Details of DLRT

Each step of Algorithm 7.1 requires the computation of two orthonormal bases for the ranges of K_k^{t+1} and L_k^{t+1}. There are of course different techniques to compute such orthonormal matrices. In our implementation, we use the QR algorithm, which is known to be one of the most efficient and stable approaches for this purpose. In the adaptive strategy, the singular values of S_k^{t+1} are truncated according to a parameter ϑ. To this end, in our implementation, we use the Frobenius norm of Σ. Precisely, we truncate $\Sigma = \mathrm{diag}(\sigma_i)$ at step 7.1 of Algorithm 7.1 by selecting the smallest principal $r \times r$ submatrix such that $(\sum_{i \geq r+1} \sigma_i^2)^{1/2} \leq \vartheta$. Finally, one-step-integrate denotes a numerical procedure that integrates the corresponding ODE from time $t = 0$ to $t = \eta$. In practice, one can employ different numerical integrators, without affecting the ability of the algorithm to reduce the loss function (see [35, Thm. 5]) while maintaining the low-rank structure. In our implementation we used two methods:

- Explicit Euler. This method applied to the gradient flow coincides with one step of Stochastic Gradient Descent (SGD), applied to the three factors K_k, L_k, S_k independently.
- Adam. Here we formally compute the new factors by modifying the explicit Euler step as in the Adam optimization method. Note that Nesterov accelerated SGD is known to coincide with a particular linear multistep ODE integrator [210]. While Adam does not directly correspond to a numerical integrator to our knowledge, in our tests it resulted in a faster decrease of the loss than both Euler (SGD) and Nesterov accelerated SGD.

For both choices, the target time step η corresponds to the value of the learning rate, which we set to 0.2 for Euler. For Adam, we use the default dynamical update, setting 0.001 as starting value.

7.3.4. Computational Cost of DLRT

To obtain minimal computational costs and memory requirements for the K-step, the ordering of evaluating $K_k V_k^\top \mathbf{z}_{k-1}$ in (7.21) is important. First, we compute

$$\widetilde{\mathbf{z}} := V_k^\top \mathbf{z}_{k-1} \in \mathbb{R}^{r_k}$$

which requires $O(r_k n_{k-1})$ operations. Second, we compute

$$K_k \widetilde{\mathbf{z}} \tag{7.22}$$

which requires $O(r_k n_k)$ operations. Adding the bias term and evaluating the activation function requires $O(n_k)$ operations. Hence, combined over all layers we have an asymptotic cost of

$$O\left(\sum_k r_k(n_k + n_{k+1})\right). \tag{7.23}$$

Taping the forward evaluation to compute the gradient with respect to K_k as discussed in §7.3.2 does not affect the asymptotic costs, i.e., the costs of computing the K-step at layer k assuming a single data point $\mathbf{x}$ requires

$$C_K \lesssim \sum_k r_k(n_k + n_{k+1}) \tag{7.24}$$

operations. Similarly, we obtain the computational costs of the L- and S-steps, which are again

$$C_{L,S} \lesssim \sum_k r_k(n_k + n_{k+1}). \tag{7.25}$$

Moreover, the QR decompositions used in the K- and L-step require

$$O\left(\sum_k r_k^2(n_k + n_{k-1})\right) \tag{7.26}$$

operations and computing the SVD in the truncation step has the worst-case cost of

$$O\left(\sum_k r_k^3\right). \tag{7.27}$$

Hence, assuming $r_k \ll n_k, n_{k+1}$, the cost per step of our low-rank method is

$$C_{\text{DLRA}} \lesssim \sum_k r_k^2(n_k + n_{k-1}), \tag{7.28}$$

opposed to dense network training, which requires

$$C_{\text{dense}} \lesssim \sum_k n_k n_{k+1} \tag{7.29}$$

operations. In terms of memory cost, note that we only need to store $r_k(r_k + n_k + n_{k+1})$ parameters per layer during the algorithm, corresponding to the matrices S_k^t, U_k^t, V_k^t. Moreover, at the end of the training, we can further compress memory by storing the product of the trained weight factors $U_k S_k$, rather than the individual matrices.

7.4. Low-Rank Matrix Representation and Implementation of Other Layer Architectures

Algorithm 7.1 shows DLRT for fully connected layers, however, modern neural network models employ a wide variety of layer designs. In the following, we adapt DLRT for convolutional and self-attention layers.

7.4.1. Convolutional Layers

A generalized convolutional filter is a four-mode tensor $W \in \mathbb{R}^{F\times C\times J\times K}$ consisting of F filters of shape $C \times J \times K$, which is applied to a batch of N input $C-$ channels image signals Z of spatial dimensions $U \times V$ as the linear mapping,

$$(Z * W)(n, f, u, v) = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{c=1}^{C} W(f, c, j, k)Z(n, c, u - j, v - k)\,. \tag{7.30}$$

To train the convolutional filter on the low-rank matrix manifold, we reshape the tensor W into a rectangular matrix $W^{\text{resh}} \in \mathbb{R}^{F\times CJK}$. This reshaping is also considered in e.g. [117]. An option is to see the convolution as the contraction between a three-mode tensor Z^{unfolded} of patches and the reshaped kernel matrix W^{resh} using Pytorch's fold-unfold function. We can construct the unfold by stacking the vectorized version of sliding patterns of the kernel on the original input, obtaining in this way a tensor $Z^{\text{unfolded}} \in \mathbb{R}^{N\times CJK\times L}$, where L denotes the dimension of flatten version of the output of the 2-D convolution. Thus, equation 7.30 can be rewritten as a tensor mode product:

$$\begin{aligned}(Z * W)(n, f, u, v) &= \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{c=1}^{C} W^{\text{resh}}(f, (c, j, k))Z^{\text{unfolded}}(n, (c, j, k), (u, v)) \\ &= \sum_{p=}^{r} U(f, p)\sum_{q=1}^{r} S(p, q)\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{c=1}^{C} V((c, j, k), q)Z^{\text{unfolded}}(n, (c, j, k), (u, v))\end{aligned} \tag{7.31}$$

As it is shown in (7.31), we can decompose the starting weight $W^{\text{resh}} = USV^\top$ and then do all the training procedures as a function of the factors (U, S, V), without ever reconstructing the kernel. Then we can apply the considerations of fully connected layers.

7.4.2. Self-Attention Layers

Self-Attention becomes more and more popular in deep learning research and engineering and is the core ingredient for transformer neural networks [59, 83, 127, 231]. Transformer neural network architectures are used for time series processing tasks in the most general sense with successful application in natural language processing [29], computer vision [59] and image synthesis [202].
We briefly describe the Transformer model architecture, where we follow [231]. A Transformer N_θ is an auto-regressive model, i.e., for a given sequence $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_M] \in \mathbb{R}^{d\times M}$, of maximum length M with elements $\mathbf{x}_i \in \mathbb{R}^d$, the next token $\mathbf{x}_{M+1}$ is predicted,

$$\mathrm{N}_\theta(\mathbf{x}) = \mathbf{x}_{\mathrm{M}+1}. \tag{7.32}$$

Then, the element is appended to the sequence, and the context window of size M is shifted by one element to the right, i.e.,

$$\mathbf{x} = [\mathbf{x}_2, \dots, \mathbf{x}_{M+1}], \tag{7.33}$$

to yield the input for the next prediction step. The transformer is built upon attention-heads followed by basic fully-connected layers with ResNet-based skip connections.

Definition 7.3
(Attention-Head) Let $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^{M\times d}$ be the query $\mathbf{q}$, key $\mathbf{k}$ and value $\mathbf{v}$ sequences. Let $\boldsymbol{\sigma} : \mathbb{R}^{M\times d} \to \mathbb{R}^{M\times d}$ be a point-wise defined non-linear activation function. For the weight matrices $W_k, W_q, W_v \in \mathbb{R}^{d\times d}$ and biases $b_k, b_q, b_b \in \mathbb{R}^d$, the attention-head is defined as

$$\mathrm{AH}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathrm{softmax}\left(\frac{\mathrm{Q}(\mathbf{q}) \cdot \mathrm{K}(\mathbf{k})}{\sqrt{d}}\right)\mathrm{V}(\mathbf{v}) \tag{7.34}$$

with

$$\begin{aligned} K(\mathbf{k}) &= \boldsymbol{\sigma}\left(W_k\mathbf{k} + \mathbf{b}_k\right) \\ Q(\mathbf{q}) &= \boldsymbol{\sigma}\left(W_q\mathbf{q} + \mathbf{b}_q\right) \\ V(\mathbf{v}) &= \boldsymbol{\sigma}\left(W_v\mathbf{v} + \mathbf{b}_v\right). \end{aligned} \tag{7.35}$$

The attention-head can be seen as a non-linear lookup table that can perform a query $\boldsymbol{Q}$ for keys $\boldsymbol{K}$ and relate the result to a value table $\boldsymbol{V}$ in parallel for the whole context window of the sequence.
Dynamical low-rank training for transformers is straightforward, where we approximate the weight matrices W_k, W_q, W_v, with low-rank matrices Y_k, Y_q, Y_v, thus Eqs. (7.35) become

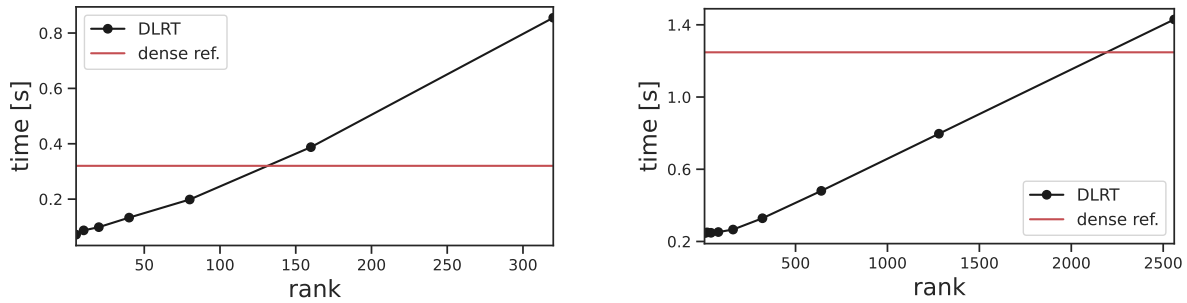
$$\begin{aligned} K(\mathbf{k}) &= \boldsymbol{\sigma}\left(U_k S_k V_k^\top \mathbf{k} + \mathbf{b}_k\right) \\ Q(\mathbf{q}) &= \boldsymbol{\sigma}\left(U_q S_q V_q^\top \mathbf{q} + \mathbf{b}_q\right) \\ V(\mathbf{v}) &= \boldsymbol{\sigma}\left(U_v S_v V_v^\top \mathbf{v} + \mathbf{b}_v\right) \end{aligned} \tag{7.36}$$

in the low-rank tensor decomposition.

7.5. Numerical Results

We illustrate the performance of DLRT Algorithm 7.1 on several test cases. The code is implemented in both Tensorflow[3] and PyTorch[4]. The networks are trained on an AMD Ryzen 9 3950X CPU and an Nvidia RTX 3090 GPU. Timings are measured on pure CPU execution. In the following, we denote an M layer low-rank network, factorized as $Y_k = U_k S_k V_k^\top$ with ranks r_k by $[r_1, \dots, r_M]$. If we refer to a full-rank reference network with weights W_k, the nation refers to the output dimension of the current layer, i.e., $[n_1, \dots, n_M]$, where $W_k \in \mathbb{R}^{n_k \times n_{k-1}}$. The following numerical tests are conducted using a well-known benchmark dataset. The MNIST dataset [56] contains $70K$ greyscale images of 10 classes and is partitioned in randomly sampled train-validation-test sets of size $50K$-$10K$-$10K$. The Cifar10 dataset [142] contains $60K$ rgb images in 10 classes, 1000 of which are test-data. Finally, the ImageNet1k dataset [55] contains 1.2 million RGB images of 1000 classes with additional $50K$ validation

[3] https://github.com/CSMMLab/DLRTNet
[4] https://github.com/COMPiLELab/DLRT



(a) Training timings

(b) Inference timings

Figure 7.3.: Comparison of averaged batch execution and training times of 5-layer, 5120-neurons low-rank networks with $Y_k \in \mathcal{M}_{r_k}$ of different ranks and a reference network with W_k on the MNIST dataset. Training times shown correspond to 200 iterations with batch-size 500. Prediction times refer instead to the whole dataset.

and $100K$ test images. In all inspected test cases, the corresponding images are pixel-wise normalized and no further data augmentation or regularization has been used. Numerical experiments on the transformer architecture are conducted using a natural language processing data set, the TED talk transcripts data set for the Portuguese-English language pair [246], consisting of $6K$ sentence pairs in the training, 600 in the validation, and 1000 in the test set. The words are tokenized using the TensorFlow tokenizer.

7.5.1. Computational Performance of Fully-Connected Networks

First, we compare the training time of the adaptive DLRT Algorithm 7.1 on a 5-layer fully-connected $[5120, 5120, 5120, 5120, 10]$ network fixing the ranks of layers 1-4, i.e. choosing a specific starting rank r_k^0 for the input weight factors and truncating Σ at line 7.1 of Algorithm 7.1 to the principal $r_k^0 \times r_k^0$ submatrix, rather than via a threshold. Table 7.1 displays the corresponding average batch training times on the MNIST dataset, with a batch size of 500 samples. We average the timings over 200 batches and additionally display the standard deviation of the timings corresponding to the layer ranks. The batch timing measures the full K, L, and S steps, including back-propagation and gradient updates, as well as the loss and metric evaluations.

Next, we measure the average inference time on the whole MNIST dataset over 1000 runs. Fig. 7.3(a) and 7.3(b) show that both timings scale linearly with the rank of the factorizations and that for sufficiently small ranks DLRT is faster than the full-rank baseline both in terms of training and prediction. Remark, that the amortization threshold is also a function of the DLRT implementation. Currently DLRT is implemented in python without CUDA Kernel acceleration which further reduces the amortization threshold. Table 7.2 shows the corresponding average inference times for different low-rank factorizations and the full-rank reference network. The timings are averaged over 1000 evaluations of the $60K$ sample MNIST training data set. We measure the K step forward evaluation of the low-rank networks as well as the loss and prediction accuracy evaluations.

Table 7.1.: Average batch training times for fixed low-rank training of a 5-layer fully-connected network with layer widths [5120, 5120, 5120, 5120, 10]. Different low-rank factorizations are compared

ranks	mean time [s]	std. deviation [s]
full-rank	0.320	±0.005227
[320, 320, 320, 320, 320]	0.855	±0.006547
[160, 160, 160, 160, 10]	0.387	±0.005657
[80, 80, 80, 80, 10]	0.198	±0.004816
[40, 40, 40, 40, 10]	0.133	±0.005984
[20, 20, 20, 20, 10]	0.098	±0.005650
[10, 10, 10, 10, 10]	0.087	±0.005734
[5, 5, 5, 5, 10]	0.071	±0.005369

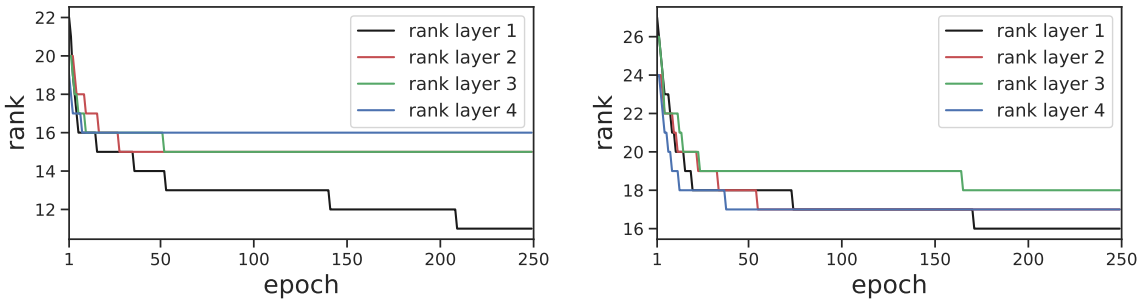
7.5.2. Rank Evolution of DLRT

Next, we demonstrate the capabilities of DLRT to determine the rank of the network's weight matrices automatically during the network training using Algorithm 7.1. The Adam optimizer with default learning rate is used for the gradient update. We train fully connected 5-layer networks, of which the first 4 are replaced by low-rank layers in the subsequent tests. The activation function is chosen to be ReLU for the hidden layers, and softmax for the output layer. The training loss is sparse categorical cross entropy and we additionally measure the model's accuracy. We use batch size 256 and train for 250 epochs. We choose $\vartheta = \tau \|\Sigma\|$, thus we truncate the singular values of the current S_k^t by a fraction τ of the total Frobenius norm. The smaller τ, the more singular values are kept.

Figure 7.4 shows the evolution of the rank adaptive layers of a 5-layer [500, 500, 500, 500, 10] network in a long time case study for $\tau = 0.03$ to $\tau = 0.17$. We can see that within the first epoch, the initial full matrix ranks are reduced significantly, to 27 for $\tau = 0.15$, and $\sim$ 85 for $\tau = 0.05$ respectively. Within the first 50 epochs, the layer ranks are already close to their final ranks. This indicates that the rank adaptive algorithm is only needed for the first few training epochs, and can then be replaced by the

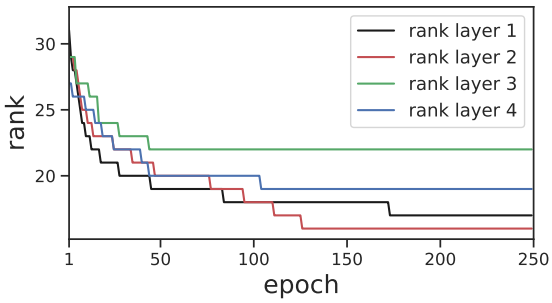
Table 7.2.: Average dataset inference times of a 5-layer fully-connected network with layer widths [5120, 5120, 5120, 5120, 10]. Different low-rank factorizations are compared.

ranks	mean time [s]	std. deviation [s]
full-rank	1.2476	±0.0471
[2560, 2560, 2560, 2560, 10]	1.4297	±0.0400
[1280, 1280, 1280, 1280, 10]	0.7966	±0.0438
[640, 640, 640, 640, 10]	0.4802	±0.0436
[320, 320, 320, 320, 10]	0.3286	±0.0442
[160, 160, 160, 160, 10]	0.2659	±0.0380
[80, 80, 80, 80, 10]	0.2522	±0.0346
[40, 40, 40, 40, 10]	0.2480	±0.0354
[20, 20, 20, 20, 10]	0.2501	±0.0274
[10, 10, 10, 10, 10]	0.2487	±0.0276
[5, 5, 5, 5, 10]	0.2472	±0.0322

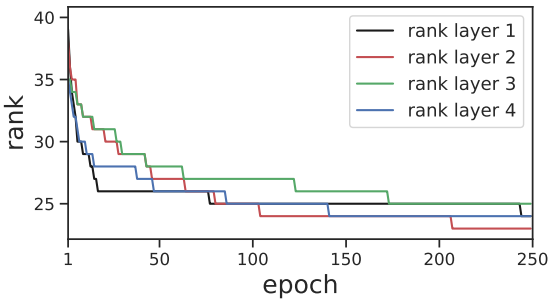


(a) $\tau = 0.17$

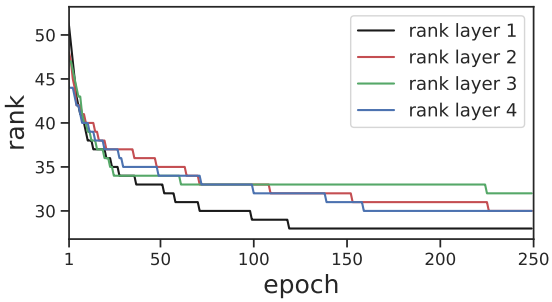
(b) $\tau = 0.15$



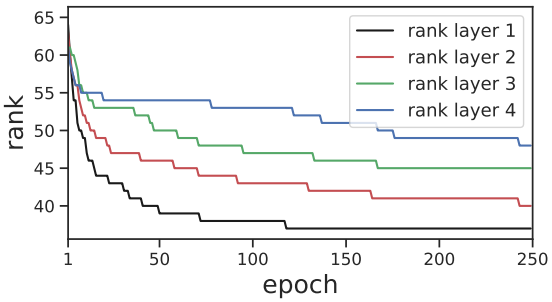
(c) $\tau = 0.13$



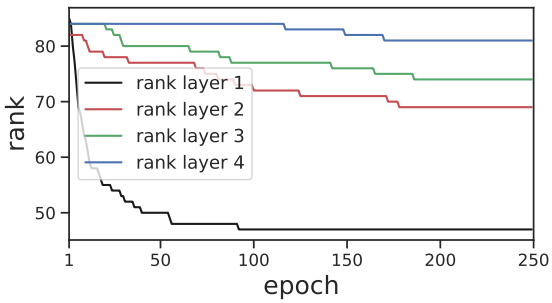
(d) $\tau = 0.11$



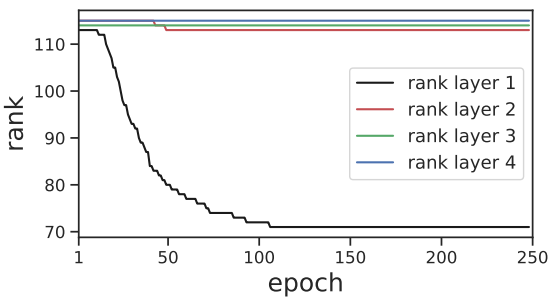
(e) $\tau = 0.09$



(f) $\tau = 0.07$



(g) $\tau = 0.05$



(h) $\tau = 0.03$

Figure 7.4.: Ranks of layers 1-4 of 5-layer $[500, 500, 500, 500, 10]$ fully-connected net on MNIST with adaptive DLRT at the end of each epoch for different compression rates τ. Ranks are significantly compressed after one epoch and stabilize after ≈ 80 epochs.

Table 7.3.: Dynamical low-rank training for a 5-layer 500-neurons network. c.r. denotes the compression rate relative to the full-rank dense network.

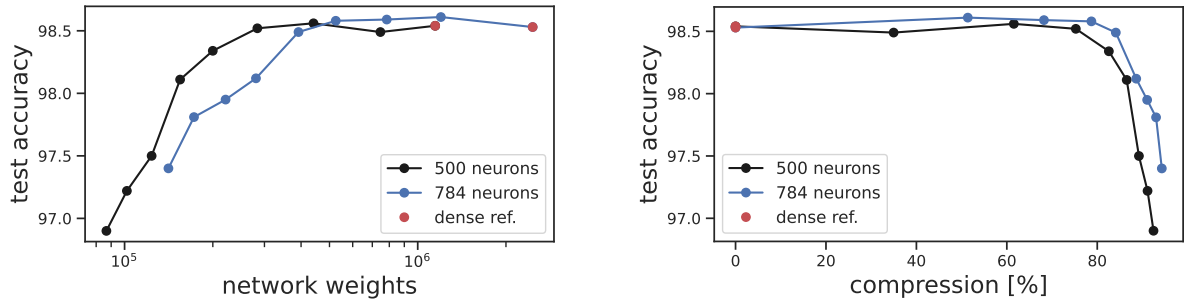
	NN metrics		Evaluation		Train	
τ	test acc.	ranks	params	c.r.	params	c.r.
full-rank	98.54 ± 0.03%	[500, 500, 500, 500, 10]	1147000	0%	1147000	0%
0.03	98.49 ± 0.02%	[176, 170, 171, 174, 10]	745984	34.97%	1964540	-71.27%
0.05	98.56 ± 0.02%	[81, 104, 111, 117, 10]	441004	61.56%	1050556	8.40%
0.07	98.52 ± 0.08%	[52, 67, 73, 72, 10]	283768	75.26%	633360	44.78%
0.09	98.34 ± 0.14%	[35, 53, 51, 46, 10]	199940	82.57%	429884	62.52%
0.11	98.11 ± 0.46%	[27, 40, 37, 38, 10]	154668	86.52%	324904	71.67%
0.13	97.50 ± 0.23%	[20, 31, 32, 30, 10]	123680	89.22%	255500	77.72%
0.15	97.22 ± 0.29%	[17, 25, 26, 24, 10]	101828	91.13%	207320	81.92%
0.17	96.90 ± 0.45%	[13, 21, 24, 20, 10]	86692	92.45%	174728	84.76%

Table 7.4.: Dynamical low-rank training for a 5-layer 784-neurons network. c.r. denotes the compression rate relative to the full-rank dense network.

	NN metrics		Evaluation		Train	
τ	test acc.	ranks	params	c.r.	params	c.r.
full-rank	98.53 ± 0.04%	[784, 784, 784, 784, 10]	2466464	0%	2466464	0%
0.03	98.61 ± 0.07%	[190, 190, 190, 190, 10]	1199520	51.37%	2968800	-20.36%
0.05	98.59 ± 0.06%	[124, 120, 125, 126, 10]	784000	68.22%	1805268	26.80%
0.07	98.58 ± 0.03%	[76, 86, 85, 83, 10]	525280	78.71%	1151864	53.29%
0.09	98.49 ± 0.05%	[56, 67, 63, 59, 10]	392000	84.41%	836460	66.08%
0.11	98.12 ± 0.21%	[35, 49, 47, 43, 10]	280672	88.63%	584240	76.31%
0.13	97.95 ± 0.23%	[29, 35, 38, 34, 10]	221088	91.04%	453000	81.63%
0.15	97.81 ± 0.17%	[22, 29, 27, 27, 10]	172480	93.01%	348252	85.88%
0.17	97.40 ± 0.25%	[17, 23, 22, 23, 10]	141120	94.28%	281724	88.57%

computationally cheaper fixed-low-rank training (by setting the Boolean variable `adaptive` to `False` in Algorithm 7.1).
Tables 7.3 and 7.4 display a detailed overview of the adaptive low-rank results for different τ. The displayed ranks are the ranks of the converged algorithm. The evaluation parameter count corresponds to the parameters of the K step of the dynamical low-rank algorithm since all other matrices are no longer needed in the evaluation phase. The training parameter count is evaluated as the number of parameters of the S step of the adaptive dynamical low-rank training, with maximal basis expansion by $2r$, where r is the current rank of the network. We use the converged ranks of the adaptive low-rank training to compute the training parameters. Note that during the very first training epochs, the parameter count is typically higher until the rank reduction has reached a sufficiently low level.

Figure 7.5 compares the mean test accuracy of 5-layer networks with 500 and 784 neurons with different levels of low-rank compression, over five independent runs each. The networks can be compressed via dynamical low-rank training by more than 95%, while only losing a little more than 1% test accuracy compared to the dense reference network marked in red. Remark that restricting the space of possible



(a) Test-accuracy over parameters

(b) Test-accuracy over compression-rate

Figure 7.5.: Mean test accuracy over parameter count and compression rate for 5 runs on 5-layer fully-connected networks. Red dots denote the full-rank baseline. Remark that compressed networks are exceeding the baseline accuracy for several rank choices.

networks to a given rank regularizes the problem since such a restriction can be understood as adding a PCR regularization term to the loss function. This can be seen from the tendency of not overfitting and reaching improved test accuracies compared to the corresponding dense network for moderate compression ratios. Also note that adaptive-low-rank training eliminates the need for hyperparameter grid search in terms of layer weights, due to automatic rank adaptation.

7.5.3. Low-Rank Pruning with DLRT

The proposed low-rank training algorithm does not need to be applied to train a network from random initial weight guesses. When an already trained network is available, the proposed method can be employed as a memory-efficient pruning strategy. A straightforward approach to reduce a trained fully-connected network to a rank r network is to compute an SVD for all weight matrices and to truncate those decompositions at rank r. However, while this choice is optimal to present weight matrices, it might significantly reduce the accuracy of the network. Hence, retraining the determined low-rank subnetwork is commonly necessary to obtain desirable accuracy properties. Three key aspects are important to obtain an efficient pruning method for low-rank methods:

- Retraining preserves the low-rank structure of the subnetwork.
- Retraining does not exhibit the memory footprint of the fully connected network.
- Retraining finds the optimal network among possible low-rank networks.

Let us note that the attractor of the proposed dynamical low-rank evolution equations fulfills these three requirements. Recall that for the evolution equations we have (7.10), i.e.,

$$\min \left\{ \|\dot{Y}_k(t) + \nabla_{Y_k} \mathcal{L}(Y_k(t))\|_F \ : \ \dot{Y}_k(t) \in \mathcal{T}_{Y_k(t)} \mathcal{M}_{r_k} \right\} \tag{7.37}$$

The condition $\dot{Y}_k(t) \in \mathcal{T}_{Y_k(t)} \mathcal{M}_{r_k}$ ensures that the weight matrices remain of low-rank. Moreover, as previously discussed, the training method only requires memory capacities to store low-rank factors. At the attractor, i.e., when $\dot{Y}_k = 0$, the last condition ensures that the attractor minimizes $\|\nabla_{Y_k} \mathcal{L}(Y_k(t))\|_F$. That is, the attractor is the optimal low-rank subnetwork in the sense that it picks the network with minimal gradient. To underline the effectiveness of our low-rank method as a pruning technique, we take the fully connected network from Table 7.4. To demonstrate the poor validation accuracy when simply doing an SVD on the full 784 by 784 weight matrices and truncating at a given smaller rank, we perform

Table 7.5.: Fixed-rank DLRT pruning for a [784, 784, 784, 784, 10] fully-connected network. Fixed-rank DLRT successfully prunes with high test accuracy, whereas SVD pruning accuracy declines.

test accuracy			evaluation	
SVD	low-rank training	ranks	params	c.r.
98.63%	98.63%	[784, 784, 784, 784, 10]	2466464	0%
9.91%	98.16%	[100, 100, 100, 100, 10]	635040	74.25%
9.67%	98.44%	[90, 90, 90, 90, 10]	572320	76.80%
9.15%	98.47%	[80, 80, 80, 80, 10]	509600	79.34%
9.83%	98.58%	[70, 70, 70, 70, 10]	446880	81.88%
9.67%	98.41%	[60, 60, 60, 60, 10]	384160	84.42%
9.83%	98.39%	[50, 50, 50, 50, 10]	321440	86.97%
10.64%	98.24%	[40, 40, 40, 40, 10]	258720	89.51%
10.3%	98.24%	[30, 30, 30, 30, 10]	196000	92.05%
9.15%	97.47%	[20, 20, 20, 20, 10]	133280	94.60%
10.9%	95.36%	[10, 10, 10, 10, 10]	70560	97.14%

this experiment for ranks $r \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. It turns out that though reducing memory requirements, this strategy leads to unsatisfactory accuracy of about 10%, see the first column of Table 7.5. Then, we use the proposed low-rank training methods with fixed rank r to retrain the network. As starting points, we use the low-rank networks which have been determined by the truncated SVD. Retraining then reaches desired accuracies that are comparable to the previously determined low-rank networks in Table 7.4.

7.5.4. Convolutional Layers: LeNet5

Here, we compare the proposed dynamical low-rank training scheme on LeNet5 [156] on MNIST, against the full-rank reference and several baselines. SVD prune [243] and LRNN [117] are the closest approaches to our DLRT: they dynamically train low-rank layers by adding a rank penalty to the loss function, and by complementing the standard training step via an SVD projection step in the latter and a pruning step in the former. While computing low-rank factors for each layer, thus reducing memory storage of the network, this training approach is more expensive than training the full network. GAL [169], SSL [245], and NISP [234] are pruning methods that aim at learning optimal sparse weights (rather than low-rank) by adding sparsity-promoting regularization terms to the training loss. As for LRNN, these methods do not reduce the computational cost of the training phase (as indicated with the $< 0\%$ in Table 7.6). Analogously to [117], our adaptive low-rank training technique is applied to the convolutional layers by flattening the tensor representing the convolutional kernel into a matrix, see §7.4.1.

All the models are trained for 120 epochs using SGD with a fixed learning rate of 0.2. Results in Table 7.6 show that the DLRT algorithm can find low-rank subnetworks with up to 96.4% fewer parameters than the full-rank reference while keeping the test accuracy above 95%. Compared to the baseline methods, we achieve better compression rates but observe lower accuracy. However, unlike the baseline references, DLRT automatically prunes the singular values during training, without the requirement to solve any additional optimization problem, thus significantly improving the time and memory efficiency of both forward and backward phases, compared to the full reference.

Table 7.6.: Results of the training of LeNet5 on MNIST dataset. Effective parameters represent the number of parameters we have to save for evaluating the network and those we need to train via the DLRT Algorithm 7.1. The abbreviation "c.r." is the relative compression rate compared to the full model (< 0% indicates that the rate is negative). The abbreviation "ft" indicates that the model has been fine-tuned. "LeNet5" denotes the standard LeNet5 architecture trained with SGD.

		NN metrics		Evaluation		Train	
	method	test acc.	ranks	params	c.r.	params	c.r.
	LeNet5	**99.2**%	[20, 50, 500, 10]	430500	0%	430500	0%
DLRT	$\tau = 0.11$	98.0%	[15, 46, 13, 10]	47975	88.86%	50585	88.25%
DLRT	$\tau = 0.15$	97.8%	[13, 31, 9, 10]	34435	92.0%	35746	91.7%
DLRT	$\tau = 0.2$	97.2%	[10, 20, 7, 10]	25650	94.04%	26299	93.89%
DLRT	$\tau = 0.3$	95.3%	[6, 9, 4, 10]	15520	**96.4**%	15753	**96.34**%
	SSL [245] (ft)	99.18%		110000	74.4%		< 0%
	NISP [234] (ft)	99.0%		100000	76.5%		< 0%
	GAL [169]	98.97%		30000	93.0%		< 0%
	LRNN [117]	98.67%	[3, 3, 9, 9]	18075	95.8%		< 0%
	SVD prune [243]	94.0%	[2, 5, 89, 10]	123646	71.2%		< 0%

7.5.5. Results on the ImageNet1K and Cifar10 Datasets

Table 7.7.: Results on ImageNet1k (left) and Cifar10 (right). The compression rate is the relative parameter reduction of the full model. DLRT is used with $\tau = 0.1$. The number of parameters of the full models is: 33.6M (VGG16); 23.6M (AlexNet); 29.6M (ResNet-50). We report the difference in test accuracy (top-5 test accuracy for ImageNet1k) to the full-rank baselines.

	ImageNet1k			
		test acc.[%]	compression rate	
	method	(to baseline)	eval[%]	train[%]
ResNet-50	DLRT	−0.56	54.1	14.2
ResNet-50	PP-2[213]	−0.8	52.2	< 0
ResNet-50	PP-1[213]	−0.2	44.2	< 0
ResNet-50	CP[107]	−1.4	50.0	< 0
ResNet-50	SFP[105]	−0.2	41.8	< 0
ResNet-50	ThiNet[177]	−1.5	36.9	< 0
VGG16	DLRT	−2.19	86	78.4
VGG16	PP-1[213]	−0.19	80.2	< 0
VGG16	CP[107]	−1.80	80.0	< 0
VGG16	ThiNet[177]	−0.47	69.04	< 0
VGG16	RNP(3X)[168]	−2.43	66.67	< 0

	Cifar10			
		test acc.[%]	compression rate	
	method	(to baseline)	eval[%]	train[%]
VGG16	DLRT	−1.89	56	77.5
VGG16	GAL[169]	−1.87	77	< 0
VGG16	LRNN[117]	−1.9	60	< 0
AlexNet	DLRT	−1.79	86.3	84.2
AlexNet	NISP[234]	−1.06	−	< 0

We assess the capability of compressing different architectures on large-scale training sets. We train a full-rank baseline model and compare it to DLRT using the same starting weights on an Nvidia A-100 GPU. The used optimizer is SGD with a momentum factor of 0.1 and no data-augmentation techniques are used. We compare the results of ResNet-50, VGG16, and AlexNet models on the Cifar10 and ImageNet1k data sets. DLRT compression results are first compared to a full-rank baseline training for

each model and then to several state-of-the-art baseline methods. The last layers of the networks have been adapted to match the corresponding classification tasks. Detailed results are reported in Table 7.7, where we show the test accuracy (reported as the difference to the full baseline) as well as compression ratios. With Cifar10, we achieve a train compression of 77.5% with an accuracy loss of just 1.89% for VGG16 and 84.2% train compression at 1.79% accuracy loss for AlexNet. In the ImageNet1k benchmark, we achieve a train compression rate of 14.2%, with a test accuracy loss of 0.5% in top-5 accuracy on ResNet-50 and 78.4% train compression with 2.19 top-5 accuracy loss on VGG16. Compression rates at inference time surpass the literature results in the ImageNet1k benchmark with 54.1% and 86%.

7.5.6. DLRT for Self-Attention: Low-Rank Transformers

We assess the capability of low-rank compression of attention heads for transformer neural networks. We investigate the classical Encoder-Decoder Transformer [231], on the TED talk transcripts data set for the Portuguese-English language pair [246]. We use the ADAM optimizer, where the learning rate is increased during a warm-up phase and then experiences an exponential decrease. Two models are investigated, one with 6 attention layers and latent dimension $d = 512$. The larger model has 55 million parameters. All weight matrices of each model, except for the embedding layers and the output layer are factorized using DLRT. We present compression results for fixed-rank DLRT for different layer ranks in Table 7.8. We report the test-accuracy difference between the baseline model and compression rate. Test accuracy refers to the prediction accuracy of the next token given the sequence up to the current one. We see, that DLRT works seamlessly in the context of self-attention.

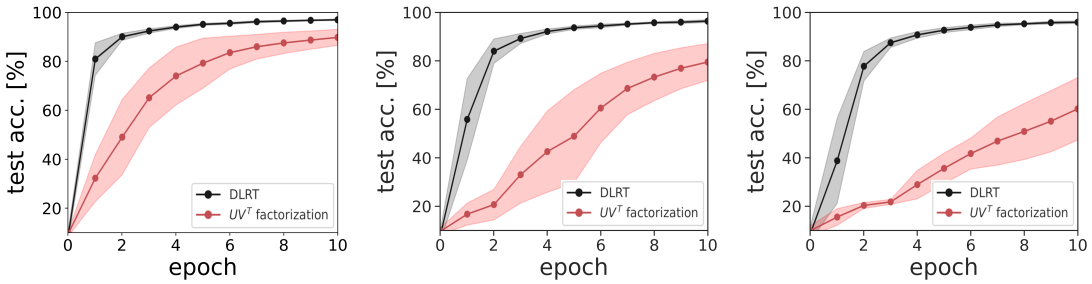
Table 7.8.: Results of the training of a language translation transformer on the TED talk transcripts Portuguese-English data set. The baseline model contains 55M parameters and has layer ranks $r = 512$. The model can be significantly compressed.

layer ranks	test acc. [%] (to baseline)	eval c.r. [%]	params
270	-0.17 ± 0.25	1.14	54.3M
250	-0.69 ± 0.33	8.91	50.1M
230	-0.74 ± 0.19	16.53	45.9M
210	-2.00 ± 0.15	23.97	41.8M
190	-2.58 ± 0.20	31.26	37.8M
170	-2.45 ± 0.12	38.39	33.8M
150	-3.51 ± 0.12	45.36	30.1M
130	-4.15 ± 0.26	52.16	26.3M
110	-4.57 ± 0.11	58.80	22.6M
90	-5.06 ± 0.16	65.29	19.0M

7.5.7. Robustness with Respect to Small Singular Values

A direct way to perform training enforcing a fixed rank for the weight matrices is to parameterize each weight as

$$W_k \approx Y_k = U_k V_k^\top \tag{7.38}$$



(a) Random initialization (b) Linear decay (c) Exponential decay

Figure 7.6.: Mean test accuracy standard deviation of LeNet5 on MNIST over 10 runs of DLRT compared to a vanilla layer factorization $W_k = U_k V_k^\top$. DLRT is robust with respect to weight initialization, whereas the vanilla factorization struggles to converge for initializations with linearly (b) and exponentially (c) decaying singular values.

and alternating training with respect to U_k and to V_k. This is the strategy employed for example in [233, 130]. This vanilla low-rank parametrization approach has several disadvantages compared to DLRT, on top of the obvious non-adaptive choice of the rank. First, DLRT guarantees approximation and descent via Theorems 7.1 and 7.2. Second, we observe that the vanilla factorization gives rise to an ill-conditioned optimization method when small singular values occur. This problem is immanent to the low-rank manifold itself [136, 72], whose local curvature is proportional to the inverse of the smallest singular value of the weight matrices. In contrast, the numerical integration strategy at the basis of DLRT is designed to take advantage of the structure of the manifold and is robust with respect to small singular values [132]. This can be seen from the bound of Theorem 7.1, where the constants are independent of the singular values of the weight matrices, and is illustrated by Figure 7.6, where DLRT shows a much faster convergence rate than vanilla SGD performed on each factor of the parametrization $U_k V_k^\top$ when applied to train LeNet5 on MNIST. Both methods are implemented with the same fixed learning rate of 0.01, and a batch size of 128. The weight matrices are either completely randomly initialized or are initialized with a random choice forced to have an exponential decay on the singular values.
Our results show that advantageous low-rank winning tickets exist, but are not easy to find. The vanilla low-rank subnetworks perform very poorly. From this point of view, our approach can be seen as an efficient dynamical pruning technique, able to determine high-performing low-rank subnetworks in a given dense network. Remarkably, our numerical experiments suggest that low-rank winning tickets can be trained from the start and do not to heavily depend on the initial weight guess.

7.6. Chapter Conclusion

In this chapter we developed a low-rank compression framework for neural networks based on numerical methods for kinetic equations.

7.6.1. Summary

We have introduced DLRT, a dynamic, rank-adaptive low-rank compression framework to compress neural networks during training and allow for wall-time and memory-efficient training and inference. Adaption of the low-rank basis and adaptive truncation based on the layers' singular values enables fast

rank reduction within only a few epochs. We have demonstrated the KLS-based update algorithm, a robust way to evolve the gradient flow of a neural network along low-rank manifolds. After a discussion of the computational complexity of DLRT was conducted, the approach was validated with several test cases on MNIST, Cifar10, and ImageNet1k and common network architectures as fully-connected or convolutional layers and transformers.

7.6.2. Limitations of the Approach

A requirement for DLRT's efficiency is that $r_k \ll n_k, n_{k+1}$. When the truncation threshold ϑ is too small, Algorithm 7.1 does not provide advantages over standard training. This is also shown by Fig. 7.3. Moreover, in the terminology of [215], DLRT is designed to reduce training costs corresponding to model parameters. To additionally decrease activation costs, DLRT can be combined with micro-batching or checkpointing approaches. Further, the activation costs can be reduced by decreasing the output dimensions of each layer as a post-processing step of the DLRT-algorithm. Possible options are channel pruning, which has been found to be complementary to layer factorization methods [150].
Finally, the choice of ϑ introduces one additional hyperparameter which at the moment requires external knowledge for tuning. However, our experiments in §7.5 show that relatively large values of ϑ yield competitive performance compared to several baselines, including standard training.

7.6.3. Future Work

DLRT opens the door to a rich field of future study. On a practical side, it is interesting to remove the hyper-parameter ϑ, e.g. by coupling it to the current validation error of the neural network or establishing a schedule similar to learning rate schedules. Furthermore, new advances in the rank-adaptive integrator, e.g. [102], can be incorporated to further accelerate the method. DLRT could also be used as a tool for neural architecture search, instead of compression, in the sense of enabling the training of larger architectures of a given rank. Here, the effects of low-rank compression of attention heads can be compared against the popular multi-head attention approach for transformers.
From a theoretical point of view, the implicit regularization of networks using DLRT by the singular value cutoff has to be inspected. In this regard, a comparison with the double-descent behavior [191] of massively over-parametrized neural networks can be inspected. Another field of future work is the inspection of the robustness of low-rank neural networks. One has to inspect, if the low-rank representation of the network possibly improves the robustness of the network with respect to adversarial attacks, i.e., noise in the input data, since cut-off of small singular values of the network weights might improve tolerance to high-frequency signals that emerge from additive noise on the data.
Lastly, DLRT can be inspected for neural ODEs, i.e., parametrized dynamical systems. Dynamical low-rank was originally developed to compress large dynamical systems and as their unparameterized counterparts, neural ODEs suffer from high computational costs during inference and training. To this end, low-rank compression of the adjoint ODE has to be investigated.

7.7. Additional Material

Statements of Section 7.3

We provide here proof of Theorems 7.1 and 7.2. We remark that the proof of 7.1 is the work of the co-authors Jonas Kusch and Gianluca Ceruti. The proof is based on some classical results as well as recent advances in DLRA theory, including [35, 37, 132, 136, 173].
Recall that, for a fixed layer k, we reinterpret the training phase as a continuous-time evolution of the weights on the manifold of low-rank matrices, as illustrated in Fig. 7.1. This boils down to solving the manifold-constrained matrix differential equation (7.10).
For the sake of simplicity and a cleaner notation, as all the results we will present hold for a generic k, we drop the subscript k from now on. In particular, we assume W is the weight matrix of a generic hidden layer with n input and m output neurons and Y is its low-rank counterpart with factorization $Y = USV^\top$.
For our derivation to hold, we require the following two properties:

P1 The gradient flow $\mathcal{F}$, see Eq. (7.7) is locally bounded and locally Lipschitz continuous, with constants C_1 and C_2, respectively. Namely, we assume there exist $C_1, C_2 > 0$ (independent of k) such that

$$\|\mathcal{F}(Z)\| \leq C_1 \qquad \|\mathcal{F}(Z) - \mathcal{F}(\tilde{Z})\| \leq C_2\|Z - \tilde{Z}\| \tag{7.39}$$

for all $Z, \tilde{Z} \in \mathbb{R}^{m\times n}$.

P2 The whole gradient flow is "not too far" from the rank-r manifold $\mathcal{M}_r$. Precisely, we assume that for any $Z \in \mathcal{M}_r$ arbitrary close to $W(t)$, the whole gradient flow $\mathcal{F}(Z)$ near t is such that

$$\|(I - \mathrm{P}(Z))\mathcal{F}(Z)\| \leq \varepsilon\,, \tag{7.40}$$

where $\mathrm{P}(Z)$ denotes the orthogonal projection onto $\mathcal{T}_Z\mathcal{M}_r$. The situation is illustrated in Fig. 7.2.

Note that both assumptions are valid for low-rank neural network training. In particular, Lipschitz continuity and boundedness of the gradient are standard assumptions in optimization and are satisfied by the gradient of commonly used neural networks' losses. Moreover, assuming the gradient flow to be close to the low-rank manifold is an often encountered empirical observation in neural networks [214, 181, 70].
To derive the proof of Theorems 7.1 and 7.2 we first present some relevant background lemmas. The first lemma shows that the subspace generated by the K-step in Algorithm 7.1 after the QR-decomposition is $O(\eta(\eta + \varepsilon))$ close to the range of the exact solution, where η is the time-step of the integrator and ε is the eigenvalue truncation tolerance.

Lemma 7.4 ([37, Lemma 2])
Let W^1 be the solution at time $t = \eta$ of the full problem (7.6) with initial condition W^0. Let U^1 be the matrix obtained with the K-step of the fixed-rank Algorithm 7.1, after one step. Under assumptions P1 and P2 above, we have

$$\|U^1U^{1,\top}W^1 - W^1\| \leq \theta \tag{7.41}$$

where

$$\theta = C_1C_2(4e^{C_2\eta} + 9)\eta^2 + (3e^{C_2\eta} + 4)\varepsilon\eta\,. \tag{7.42}$$

The proof is shown in [37, Lemma 2]. In the next lemma, we show that also the space generated by the L step is close to the exact solution. Namely, combined with the previous result, we have

Lemma 7.5 ([37, Lemma 3])
Let W^1, U^1 be defined as above. Let V^1 be the matrix obtained from the L-step of the fixed-rank Algorithm 7.1, after one step. The following estimate holds:

$$\|U^1 U^{1,\top} W^1 V^1 V^{1,\top} - W^1\| \leq 2\theta. \tag{7.43}$$

The proof is shown in [37, Lemma 3].

With the previous lemmas, we are in the position to derive the local error bound for the fixed-rank KLS integrator of Section 7.3.

Lemma 7.6 (Local Error, [37, Lemma 4])
Let W^1, U^1, V^1 be defined as above, and let S^1 be the matrix obtained with the S-step of Algorithm 7.1 after one step. The following local error bound holds:

$$\|U^1 S^1 V^{1,\top} - W^1\| \leq \eta(\hat{c}_1\varepsilon + \hat{c}_2\eta), \tag{7.44}$$

where the constants $\hat{c}_i$ are independent of the singular values of W^1 and S^1.

The proof is shown in [37, Lemma 4].

We are now in the position to conclude the proof of Theorem 7.1.

Proof: (Theorem 7.1) In Lemma 7.6, the local error for the fixed-rank integrator of §7.3 has been provided. The local temporal error of the rank-adaptive version is directly obtained via a triangle inequality

$$\|U^1 S^1 V^{1,\top} - W(\eta)\| \leq \hat{c}_1\varepsilon\eta + \hat{c}_2\eta^2 + \vartheta\,, \tag{7.45}$$

where ϑ is the tolerance parameter chosen for the singular value truncation procedure. Here, we abuse the notation and we maintain the same nomenclature U^1, S^1, and V^1 also for the novel low-rank approximation obtained via the truncation procedure.

Thus, we conclude the proof using the Lipschitz continuity of the function $\mathcal{F}$. We move from the local to the global temporal error by a standard argument of Lady Windermere's fan [95, Section II.3]. Therefore, the error after t steps of the rank-adaptive Algorithm 7.1 is given by

$$\|U^t S^t V^{t,\top} - W(t\eta)\| \leq c_1\varepsilon + c_2\eta + c_3\vartheta/\eta\,. \tag{7.46}$$

□

To conclude, we prove that after one step the proposed rank-adaptive DLRT algorithm decreases along the low-rank approximations. We recall that only property P1 needs to be assumed here.

Proof: (Theorem 7.2) Let $\widehat{Y}(t) = U^1 S(t) V^{1,\top}$. Here, $S(t)$ denotes the solution for $t \in [0, \eta]$ of the S-step of the rank-adaptive integrator. It follows that

$$\begin{aligned}
\frac{d}{dt}\mathcal{L}(\widehat{Y}(t)) &= \langle \nabla\mathcal{L}(\widehat{Y}(t)), \dot{\widehat{Y}}(t)\rangle \\
&= \langle \nabla\mathcal{L}(\widehat{Y}(t)), U^1 \dot{S}(t) V^{1,\top}\rangle \\
&= \langle U^{1,\top}\nabla\mathcal{L}(\widehat{Y}(t))V^1, \dot{S}(t)\rangle \\
&= \langle U^{1,\top}\nabla\mathcal{L}(\widehat{Y}(t))V^1, -U^{1,\top}\nabla\mathcal{L}(\widehat{Y}(t))V^1\rangle = -\|U^{1,\top}\nabla\mathcal{L}(\widehat{Y}(t))V^1\|^2\,.
\end{aligned} \tag{7.47}$$

The last identities follow by definition of the S-step. For $t \in [0, \eta]$ we have

$$\frac{d}{dt}\mathcal{L}(\widehat{Y}(t)) \leq -\alpha^2 \tag{7.48}$$

where $\alpha = \min_{0\leq\tau\leq 1} \|U^{1,\top}\nabla\mathcal{L}(\widehat{Y}(\tau\eta))V^1\|$. Integrating (7.48) from $t = 0$ until $t = \eta$, we obtain

$$\mathcal{L}(\widehat{Y}^1) \leq \mathcal{L}(\widehat{Y}^0) - \alpha^2\eta. \tag{7.49}$$

Because the subspaces U^1 and V^1 contain by construction the range and co-range of the initial value, we have that $\widehat{Y}^0 = U^0 S^0 V^{0,\top}$ [35, Lemma 1]. The truncation is such that $\|Y^1 - \widehat{Y}^1\| \leq \vartheta$. Therefore,

$$\mathcal{L}(Y^1) \leq \mathcal{L}(\widehat{Y}^1) + \beta\vartheta \tag{7.50}$$

where $\beta = \max_{0\leq\tau\leq 1} \|\nabla\mathcal{L}(\tau Y^1 + (1-\tau)\widehat{Y}^1)\|$. Hence, the stated result is obtained. □

Detailed Derivation of the Gradient

In this section, we derive the computation of the gradients in the K, L, and S steps in detail. For this, let us start with the full gradient, i.e., the gradient of the loss with respect to the weight matrix W_k. We have

$$\begin{aligned}\partial_{W^\ell_{jk}}\mathcal{L} &= \sum_{i_M=1}^{n_M} \partial_{z^M_{i_M}}\mathcal{L}\partial_{W^\ell_{jk}} z^M_{i_M} = \sum_{i_M=1}^{n_M} \partial_{z^M_{i_M}}\mathcal{L}\partial_{W^\ell_{jk}}\sigma_M\left(\sum_{i_{M-1}} W_{i_M i_{M-1}} z^{M-1}_{i_{M-1}} + b^M_{i_M}\right)\\ &= \sum_{i_M=1}^{n_M} \partial_{z^M_{i_M}}\mathcal{L}\sigma'_M\left(\sum_{i_{M-1}} W_{i_M i_{M-1}} z^{M-1}_{i_{M-1}} + b^M_{i_M}\right)\partial_{W^\ell_{jk}}\left(\sum_{i_{M-1}} W_{i_M i_{M-1}} z^{M-1}_{i_{M-1}}\right).\end{aligned} \tag{7.51}$$

For a general α, let us define

$$\sigma'_{\alpha,i_\alpha} := \sigma'_\alpha\left(\sum_{i_{\alpha-1}} W^\alpha_{i_\alpha i_{\alpha-1}} z^{\alpha-1}_{i_{\alpha-1}} + b^\alpha_{i_\alpha}\right) \tag{7.52}$$

and note that for $\alpha \neq \ell$

$$\partial_{W^\ell_{jk}}\left(\sum_{i_{\alpha-1}} W^\alpha_{i_\alpha i_{\alpha-1}} z^{\alpha-1}_{i_{\alpha-1}}\right) = \sum_{i_{\alpha-1}} W^\alpha_{i_\alpha i_{\alpha-1}}\partial_{W^\ell_{jk}} z^{\alpha-1}_{i_{\alpha-1}}, \tag{7.53}$$

whereas for $\alpha = \ell$ we have

$$\partial_{W^\ell_{jk}}\left(\sum_{i_{\alpha-1}=1}^{n_{\alpha-1}} W^\alpha_{i_\alpha i_{\alpha-1}} z^{\alpha-1}_{i_{\alpha-1}}\right) = \sum_{i_{\alpha-1}} \delta_{ji_\alpha}\delta_{ki_{\alpha-1}} z^{\alpha-1}_{i_{\alpha-1}}. \tag{7.54}$$

Therefore, recursively plugging (7.52), (7.53) and (7.54) into (7.51) yields

$$\begin{aligned}
\partial_{W^\ell_{jk}} \mathcal{L} &= \sum_{i_M=1}^{n_M} \partial_{z^M_{i_M}} \mathcal{L} \sigma'_{M,i_M} \sum_{i_{M-1}} W^\alpha_{i_M i_{M-1}} \partial_{W^\ell_{jk}} z^{M-1}_{i_{M-1}} \\
&= \sum_{i_M=1}^{n_M} \partial_{z^M_{i_M}} \mathcal{L} \sigma'_{M,i_M} \sum_{i_{M-1}} W^\alpha_{i_M i_{M-1}} \sigma'_{M-1,i_{M-1}} \sum_{i_{M-2}} W^\alpha_{i_{M-1} i_{M-2}} \partial_{W^\ell_{jk}} z^{M-2}_{i_{M-2}} = \cdots \\
&= \sum_{i_\ell, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \partial_{W^\ell_{jk}} \left(\sum_{i_{\ell-1}=1}^{n_{\ell-1}} W^\ell_{i_\ell i_{\ell-1}} z^{\ell-1}_{i_{\ell-1}} \right) \\
&= \sum_{i_\ell, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \sum_{i_{\ell-1}} \delta_{j i_\ell} \delta_{k i_{\ell-1}} z^{\ell-1}_{i_{\ell-1}} \\
&= \sum_{i_{\ell+1}, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,j} \delta_{j i_\ell} z^{\ell-1}_{k}
\end{aligned} \tag{7.55}$$

Written in matrix notation and making use of the Hadamard product defined as $y \circ A \circ x = (y_i A_{ij} x_j)_{ij}$, for $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, we have:

$$\partial_{W^\ell} \mathcal{L} = \partial_{z^M} \mathcal{L}^\top \left(\sigma'_\ell \circ \prod_{\alpha=\ell+1}^{M} W^\top_\alpha \circ \sigma'_\alpha \right)^\top z^\top_{\ell-1} \tag{7.56}$$

Now, let us derive the K, L, and S-steps for the dynamical low-rank training. For the K-step, we represent the weight matrix W_ℓ as $W^\ell_{i_\ell i_{\ell-1}} = \sum_m K^\ell_{i_\ell m} V^\ell_{i_{\ell-1} m}$. Hence, reusing the intermediate result (7.55) yields

$$\begin{aligned}
\partial_{K^\ell_{jk}} \mathcal{L} &= \sum_{i_\ell, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \partial_{K^\ell_{jk}} \left(\sum_{i_{\ell-1}=1}^{n_{\ell-1}} \sum_m K^\ell_{i_\ell m} V^\ell_{i_{\ell-1} m} z^{\ell-1}_{i_{\ell-1}} \right) \\
&= \sum_{i_\ell, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \sum_{i_{\ell-1}=1}^{n_{\ell-1}} \sum_m \delta_{j i_\ell} \delta_{km} V^\ell_{i_{\ell-1} m} z^{\ell-1}_{i_{\ell-1}} \\
&= \sum_{i_{\ell+1}, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \sum_{i_{\ell-1}=1}^{n_{\ell-1}} \delta_{j i_\ell} V^\ell_{i_{\ell-1} k} z^{\ell-1}_{i_{\ell-1}}
\end{aligned} \tag{7.57}$$

In matrix notation we obtain

$$\partial_{K_\ell} \mathcal{L} = \partial_{z^M} \mathcal{L}^\top \left(\sigma'_\ell \circ \prod_{\alpha=\ell+1}^{M} W^\top_\alpha \circ \sigma'_\alpha \right)^\top \left(V^\top_\ell z_{\ell-1} \right)^\top = \partial_{W_\ell} \mathcal{L} V_\ell, \tag{7.58}$$

which is exactly the right-hand side of the K-step. Hence, the K-step can be computed by a forward evaluation of $\mathcal{L}$ and recording the gradient tape with respect to K^ℓ. Similarly, for the L-step, we represent W_ℓ as $W^\ell_{i_\ell i_{\ell-1}} = \sum_m U^\ell_{i_\ell m} L^\ell_{i_{\ell-1} m}$. Hence,

$$\begin{aligned}
\partial_{L^\ell_{jk}} \mathcal{L} &= \sum_{i_\ell, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \partial_{L^\ell_{jk}} \left(\sum_{i_{\ell-1}=1}^{n_{\ell-1}} \sum_m U^\ell_{i_\ell m} L^\ell_{i_{\ell-1} m} \right) \\
&= \sum_{i_\ell, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \sum_{i_{\ell-1}=1}^{n_{\ell-1}} \sum_m U^\ell_{i_\ell m} \delta_{j i_{\ell-1}} \delta_{km} z^{\ell-1}_{i_{\ell-1}} \\
&= \sum_{i_\ell, \cdots, i_M} \partial_{z^M_{i_M}} \mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} U^\ell_{i_\ell m} z^{\ell-1}_{j}.
\end{aligned} \tag{7.59}$$

In matrix notation, we obtain

$$\partial_{L_\ell}\mathcal{L} = \left(U_\ell^\top \partial_{z^M}\mathcal{L}^\top \left(\sigma'_\ell \circ \prod_{\alpha=\ell+1}^{M} W_\alpha^\top \circ \sigma'_\alpha \right)^\top z_{\ell-1}^\top \right)^\top = (\partial_{W_\ell}\mathcal{L})^\top U_\ell. \tag{7.60}$$

Lastly, for the S-step we write $W^\ell_{i_\ell i_{\ell-1}} = \sum_{n,m} U^\ell_{i_\ell m} S_{mn} V^\ell_{i_{\ell-1} n}$. Then,

$$\begin{aligned}
\partial_{S^\ell_{jk}}\mathcal{L} &= \sum_{i_\ell,\cdots,i_M} \partial_{z^M_{i_M}}\mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \partial_{S^\ell_{jk}} \left(\sum_{n,m} U^\ell_{i_\ell m} S_{mn} V^\ell_{i_{\ell-1} n} \right) \\
&= \sum_{i_\ell,\cdots,i_M} \partial_{z^M_{i_M}}\mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} \sum_{i_{\ell-1}=1}^{n_{\ell-1}} \sum_m U^\ell_{i_\ell m} \delta_{jm} \delta_{kn} V^\ell_{i_{\ell-1} n} z^{\ell-1}_{i_{\ell-1}} \\
&= \sum_{i_\ell,\cdots,i_M} \partial_{z^M_{i_M}}\mathcal{L} \prod_{\alpha=\ell+1}^{M} \sigma'_{\alpha,i_\alpha} W^\alpha_{i_\alpha i_{\alpha-1}} \sigma'_{\ell,i_\ell} U^\ell_{i_\ell j} V^\ell_{i_{\ell-1} k} z^{\ell-1}_{i_{\ell-1}}.
\end{aligned} \tag{7.61}$$

In matrix notation, we have

$$\partial_{S_\ell}\mathcal{L} = U_\ell^\top \partial_{z^M}\mathcal{L}^\top \left(\sigma'_\ell \circ \prod_{\alpha=\ell+1}^{M} W_\alpha^\top \circ \sigma'_\alpha \right)^\top \left(V_\ell^\top z_{\ell-1} \right)^\top = U_\ell^\top \partial_{W_\ell}\mathcal{L} V_\ell. \tag{7.62}$$

CHAPTER 8

Summary and Outlook

Summary

In this thesis, we have investigated neural network-based surrogate models for kinetic and macroscopic partial differential equations, as well as an acceleration algorithm neural network training, which is inspired by numerical methods for kinetic equations. We have leveraged synergies between neural networks and kinetic modeling to develop fast and structure-preserving surrogate models for kinetic methods as well as efficient and stable low-rank training for neural networks.
We started with a thorough review of the Boltzmann equation and common discretization methods for the velocity variable of the phase space, with a focus on nodal and modal models in §1. Afterward, we built an efficient C++ based framework and implemented these macroscopic models on top of a modular, unstructured grid-based, second-order finite-volume-method for the spatial-temporal discretization of the Boltzmann equation in §3. The resulting open-source framework KiT-RT is extended to radiation transport applications and extensively validated against existing codebases for kinetic transport and radiation therapy. We have compared nodal and moment methods with different closures in terms of parallel scalability, wall-time and memory performance, and simulation accuracy. This layed out the computational foundation and motivattion for the neural network-based surrogate models in the following chapters.
Next, we turned to the minimal entropy closure of the Boltzmann equation in §4, which stands out on the one hand for its structural richness, i.e. preservation of the mathematical properties of the Boltzmann equation, and its modeling versatility, but on the other hand is prohibitively expensive due to the high computational effort of computing the minimal entropy problem, especially for high spatial dimensions or high order moment bases. We have developed a neural network-based surrogate model to bypass the iterative evaluation of the minimal entropy closure, which preserves the inherent structure of the Boltzmann equation and its moment system. Key to the structure preservation is the built-in convexity of the neural network, which was also used to analyze interpolation error bounds and in the development of a data-sampling strategy aimed to minimize inference errors of the network. The input convex and the related input-monotone neural network surrogates were validated in various numerical tests, where we highlighted their computational efficiency, the preservation of mass and dissipation of entropy, as well as numerical stability for lower-order moment systems.

In §5, we considered the pitfalls of the neural network surrogate models presented in §4, especially poor performance for moments near the boundary of the realizable set, i.e., in highly anisotropic simulations. We introduce a dynamically scalable regularization for the minimal entropy closure problem and transfer the surrogate model to this setting, with corresponding data generation based on the ansatz of the regularized problem. Furthermore, we guarantee Galilean invariance of the surrogate model by the introduction of a rotated and symmetrizing closure. We proved, that normalization, (partial) dynamical regularization and rotation of the closure still guarantee a convex approximation to the entropy functional of the moment system, thus preservation of the system's mathematical structure. We compared data-sampling of the new, regularized problem with the original strategies and remark, that the error-analysis transfers. Numerical tests showed, that regularized surrogate models train significantly faster and to higher test accuracy than their non-regularized counterparts, especially for higher dimensional closures. Rotational invariance of the surrogate models helps in an-isotropic simulations as in the Linesource test case. A broad numerical study in the Hohlraum test case shows that the ICNN-based entropy closure leads to a computationally competitive simulation method with an advantageous trade-off between memory footprint and numerical error, compared to the spherical harmonics, the traditional minimal entropy, and the nodal method.
We shift our attention to the non-linear Boltzmann equation for kinetic description of rarefied gases in §6, where we built a neural network-based flow-regime classifier to predict the breakdown of the continuum assumption in multi-scale flows. We again drew from the minimal entropy closure model, to construct an algorithm to generate training data for the neural networks. We bridged the Navier-Stokes and kinetic solver using the neural network switching criterion to construct a fast hybrid numerical method for multi-scale flows. Numerical evaluation at the Sod shock tube and cylinder test case validates the method and shows superiority over hand-crafted switching criteria as the gradient-length-local Knudsen number in terms of accuracy.
Opposed to the previous chapters, where we used neural networks to speed up kinetic simulations, we made use of the rich numerical analysis for kinetic systems to accelerate training and inference for neural networks in §7. We interpreted the gradient flow of neural networks as a high-dimensional dynamical system. The dynamical, adaptive low-rank evolution of this system is used to compress neural networks during training to a low-rank factorization, a novelty compared to state-of-the-art methods, which typically compress a network only after training. We showed the stability and monotonicity of the DLRT algorithm for low-rank training, which evolves the neural network on low-rank manifolds of its solution space. The approach is validated on state-of-the-art network architectures for computer vision and natural language translation, where we showed an order of magnitude speedup and reduction of the memory footprint during training and inference.

Outlook

A considerable amount of time has been spent over the last few years working on the projects that constitute this dissertation. However, no project is ever truly finished and thus we would like to point the reader into future directions of research that emerge from the conducted research.
The KiT-RT framework can be extended in multiple directions. First, a low-rank acceleration of the macroscopic methods [148] can be implemented. Second, the MPI capabilities of the framework can be extended to support distributed memory systems. Naturally, much software development time has to be invested to make KiT-RT a truly useful tool for radiation transport planning, where for example optimization of treatment plans in a gradient-based or gradient-free manner can be conducted using the existing solvers to sample the objective functions. Uncertainty quantification in radiotherapy planning is a further research direction, in which KiT-RT can be extended.

The projects focused on neural network-based surrogate modeling of entropy closures have covered a wide range of challenges. Future research may consider neural network architectures, which are rotation invariant by design [69, 179, 44]. They can be considered as an alternative to the post-processing rotation proposed in this work. It needs to be investigated to which extent input-convex, rotationally invariant neural networks can be constructed. Entropy closures can be used for any truncated moment problem, i.e., it has a wide range of further applications in statistics. In the domain of dynamical systems, the techniques can be applied for Lyapunov function approximations in high dimensions. Further, the developed tools, especially the error control can be applied to any other convex function approximation task.

In the flow-regime classification project, future research can be directed toward the creation of a more general sampling algorithm and the engineering of a more sophisticated neural network architecture for regime prediction. Currently, only a single grid cell is considered for the regime prediction. One could inspect local neighborhoods of the current cell with a convolutional approach, graph-neural networks, or transformer-like architectures.

The last project in focus is the dynamical low-rank training of neural networks. DLRT opens the door to a rich field of future study. On a practical side, it is interesting to remove the hyperparameter ϑ, e.g., by coupling it to the current validation error of the neural network or establishing a schedule similar to learning rate schedules. Furthermore, new advances in the rank-adaptive integrator, e.g. [102], can be incorporated to further accelerate the method. DLRT could also be used as a tool for neural architecture search, instead of compression, in the sense of enabling the training of larger architectures of a given rank. Here, the effects of low-rank compression of attention heads can be compared against the popular multi-head attention approach for transformers.

From a theoretical point of view, the implicit regularization of networks using DLRT by the singular value cutoff has to be inspected. In this regard, a comparison with the double-descent behavior [191] of massively over-parametrized neural networks has to be conducted. Another field of future work is the robustness of low-rank neural networks. Does a low-rank representation of the network possibly improve the robustness of the network with respect to adversarial attacks , i.e., noise in the input data? The cut-off of small singular values of the network weights might improve tolerance to high-frequency signals, that emerge from additive noise on the data.

Lastly, DLRT can be applied to neural ODEs, i.e. parametrized dynamical systems. Dynamical low-rank was originally developed to compress large dynamical systems and as their unparameterized counterparts, neural ODEs suffer from high computational costs during inference and training. To this end, a low-rank compression of the adjoint ODE has to be developed.

APPENDIX A

Nomenclature

We state a nomenclature for commonly used variables of this work.

Table A.1.: General nomenclature

Variable	Description
$\cdot : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$	Inner product of two vectors in $\mathbb{R}^n$
$\otimes : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^{m \times n}$	Outer product of two vectors
$\circ : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$	Hadamard product of two vectors in $\mathbb{R}^n$
$\nabla_{\cdot}$	Nabla operator with respect to variable $\cdot$
$\partial_{\cdot}$	Partial derivative with respect to variable $\cdot$
$\frac{\mathrm{d}}{\mathrm{d}\cdot}$	Total derivative of with respect to variable $\cdot$
$\dot{x}(t)$	Time derivative of a time dependent variable $x(t)$
$\delta(\cdot)$	Dirac distribution at point $\cdot$
$\mathbf{I}$	Identity tensor
$\mathbf{n}$	Unit normal vector
$\mathcal{H}$	Heaviside step function

Table A.2.: Nomenclature - Entropy closures and the Boltzmann equation

Variable	Description
$\mathbf{v} \in \mathbb{R}^d$	Velocity variable in d dimensions
$\mathbf{V} \subset \mathbb{R}^d$	Velocity domain in d dimensions
$f(t, \mathbf{x}, \mathbf{v}) : \mathbb{R}_+ \times \mathbf{X} \times \mathbf{V} \to \mathbb{R}$	Kinetic density
$\boldsymbol{\phi}(\mathbf{v}) : \mathbf{V} \to \mathbb{R}$	Vector of collision invariants of the Boltzmann collision operator
$\mathbb{E}$	Span of collision invariants of the Boltzmann collision operator
$\eta(f) : D \to \mathbb{R}$	Kinetic entropy density
$\mathbf{u} \in \mathbb{R}^n$	Moment vector of the kinetic density f

$\mathbf{m}(\mathbf{v}) : \mathbf{V} \to \mathbb{R}^n$	moment basis of the velocity space $\mathbf{V}$
$f_{\mathbf{u}}(\mathbf{v}) : \mathbf{V} \to D$	reconstructed kinetic density using the moment closure
$F_{\mathbf{m}}$	set of feasible functions for the minimal entropy closure problem
$h(\mathbf{u}) : \mathcal{R} \to \mathbb{R}$	Entropy functional of the Boltzmann moment system
$\mathbf{j}(\mathbf{u}) : \mathcal{R} \to \mathbb{R}^d$	Entropy-flux of the Boltzmann moment system
$\mathcal{R} \subset \mathbb{R}^n$	Realizable set of the minimal entropy closure problem
$\overline{\mathcal{R}} \subset \mathbb{R}^n$	Normalized realizable set of the minimal entropy closure problem
$\overline{\mathcal{R}} \subset \mathbb{R}^n$	Reduced, normalized realizable set of the minimal entropy closure
$\boldsymbol{\alpha} \in \mathbb{R}^n$	Lagrange multiplier of the dual minimal entropy closure problem
$\boldsymbol{\alpha}_{\mathbf{u}} \in \mathbb{R}^n$	Optimal Lagrange multiplier with respect to $\mathbf{u}$
$\phi(\boldsymbol{\alpha}; \mathbf{u}) : \mathbb{R}^n \to \mathbb{R}$	Objective function of the dual minimal entropy closure problem
$\phi_n(\boldsymbol{\alpha}; \mathbf{u}) : \mathbb{R}^n \to \mathbb{R}$	Negative objective function of the dual minimal entropy closure problem
$H(\boldsymbol{\alpha}) : \mathbb{R}^{n \times n}$	Hessian of the dual minimal entropy closure problem
$H_n(\boldsymbol{\alpha}) : \mathbb{R}^{n \times n}$	Negative of the Hessian of the dual minimal entropy closure
$\gamma \in \mathbb{R}_+$	Regularization parameter for optimization problems
$\langle \cdot \rangle$	Integral of $\cdot$ over the velocity space
$\bar{\cdot}$	Normalization operator for moments
$(\cdot)_{\#}$	Truncation operator for moments and Lagrange multipliers
$Q(f) : L_1(\mathbf{V}) \to L_1(\mathbf{V})$	Linear collision operator of the Boltzmann Equation
$Q(f, f) : L_1(\mathbf{V}) \to L_1(\mathbf{V})$	Non-linear collision operator of the Boltzmann Equation
$\mathcal{A}(f, f) : L_1(\mathbf{V}) \to L_1(\mathbf{V})$	Advection operator of the Boltzmann Equation
$\mathcal{B}(f, f) : \mathbf{V} \times \mathbf{V} \to \mathbb{R}_+$	Collision kernel of the Boltzmann Equation
$(\cdot)_*$	Legendre transform
$\mathcal{U} : \mathbb{R}^n \to F_{\mathbf{m}}$	Moment closure operator

Table A.3.: Nomenclature - Numerical methods for conservation laws

Variable	Description
$\mathbf{g}(t, \mathbf{x}) \in \mathbb{R}^n$	Solution of a general conservation law in space and time
$\mathbf{x} \in \mathbb{R}^d$	Spatial variable in d dimensions
$\mathbf{v} \in \mathbb{R}^d$	Velocity variable in d dimensions
$t \in \mathbb{R}_+$	Temporal variable
$\mathbf{X} \subset \mathbb{R}^d$	Spacial domain in d dimensions
$\mathbf{V} \subset \mathbb{R}^d$	Velocity domain in d dimensions
$h(\mathbf{g}) : \mathbb{R}^n \to \mathbb{R}$	Entropy of a conservation law
$\mathbf{j}(\mathbf{g}) : \mathbb{R}^n \to \mathbb{R}^d$	Entropy-flux of a conservation law
$\mathbb{S}^2 \subset \mathbb{R}^3$	Unit sphere in $\mathbb{R}^3$
$\mu \in [-1, 1]$, $\theta \in [0, 2\pi)$	Polar coordinates parametrization of $\mathbb{S}^2$
w_q	quadrature weight with index q
$\mathbf{u} \in \mathbb{R}^n$	moment vector of the kinetic density f
$\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^{n \times d}$	Flux function of a conservation law
$\mathbf{X}_i$	Spatial grid cell with index i
$N(i)$	Indices of neighboring spatial grid cells of cell i
$\mathbf{R} : \mathbb{R}^n \to \mathbb{R}^n$	Discretized right hand side of a conservation law
$\varphi \in \mathbb{C}_0^\infty(\mathbb{R}_+ \times \mathbb{R}^d)$	Test function for a weak solution of a conservation law

Table A.4.: Nomenclature - Gas dynamics

Variable	Description
Kn	Knudsen number
$\mathrm{Kn}_{\mathrm{GLL}}$	Gradient-length-local Knudsen number
$\mathcal{M}_{(\rho,\mathbf{U},T)}$	Maxwellian distribution function
k	Boltzmann constant
$\mathbf{q}$	Macroscopic conservative variables
ρ	Macroscopic particle Density
$\mathbf{U}$	Bulk velocity
T	Macroscopic temperature
R	Gas constant
μ	Viscosity coefficient
κ	Heat conductivity coefficient
$\mathbf{c}$	Peculiar velocity
$\mathcal{E}$	equilibrium distribution function
$\mathbf{a}$	spatial derivatives of kinetic density
b	time derivatives of kinetic density
$P_{\hat{r}}$	Neural network output - Flow regime probability

Table A.5.: Nomenclature - Neural networks

Variable	Description
N_θ	Neural network with trainable parameters θ
$\sigma_k(z) : \mathbb{R}^{n_k \times b} \to \mathbb{R}^{n_k \times b}$	Pointwise activation function of layer k of a neural network with batch size b
$W_k \in \mathbb{R}^{n_k \times n_{k-1}}$	Weight matrix of layer k of a neural network
$\boldsymbol{b}_k \in \mathbb{R}^{n_k}$	Bias term of layer k of a neural network
$\mathcal{T}$	Training error of a neural network
$\mathcal{G}$	Generalization error of a neural network
$\widetilde{\mathcal{T}}$	Test error of a neural network
$\lambda \in \mathbb{R}_+$	Learning rate of a neural network, step size of a numerical optimizer
$\mathcal{L}$	Loss functional for neural network training
$\boldsymbol{z}_k$	Output of layer k of a neural network
$\mathbf{x}$	Input data of a neural network
$\mathbf{y}$	Output data of a neural network

Table A.6.: Nomenclature - Low-rank compression for neural networks

Variable	Description
W	General full rank dynamical system
Y	General low-rank dynamical system
$\mathcal{F}$	Right hand side of a general dynamical system
$\mathcal{M}_r$	Low-Rank manifold of rank r
$T_{Y(t)}\mathcal{M}$	Tangent plane of the low-rank manifold at $Y(t)$

$U \in \mathbb{R}^{n\times r}, V^\top \in \mathbb{R}^{m\times r}$	Basis vectors of $\mathcal{M}_r$
$S \in \mathbb{R}^{r\times r}$	Coefficient matrix of the $\mathcal{M}_r$ corresponding to U and $V^\top$
δY	Element of the tangent plane
$\langle \cdot, \cdot \rangle$	Scalar product in $\mathbb{R}^n$

Bibliography

[1] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[3] IK Abu-Shumays. Angular quadratures for improved transport computations. *Transport Theory and Statistical Physics*, 30(2-3):169–204, 2001.

[4] Michael Aftosmis, Datta Gaitonde, and T. Sean Tavares. Behavior of linear reconstruction techniques on unstructured meshes. *AIAA Journal*, 33(11):2038–2049, 1995.

[5] Graham Alldredge, Martin Frank, Jonas Kusch, and Ryan McClarren. A realizable filtered intrusive polynomial moment method. *Journal of Computational and Applied Mathematics*, 407:114055, 2022.

[6] Graham Alldredge and Florian Schneider. A realizability-preserving discontinuous galerkin scheme for entropy-based moment closures for linear kinetic equations in one space dimension. *Journal of Computational Physics*, 295:665–684, aug 2015.

[7] Graham W. Alldredge, Martin Frank, and Cory D. Hauck. A regularized entropy-based moment method for kinetic equations. *SIAM Journal on Applied Mathematics*, 79(5):1627–1653, 2019.

[8] Graham W. Alldredge, Cory D. Hauck, and André L. Tits. High-order entropy-based closures for linear transport in slab geometry ii: A computational study of the optimization problem. *SIAM Journal on Scientific Computing*, 34(4):B361–B391, 2012.

[9] Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 08 2017.

[10] P Andreo. Monte Carlo techniques in medical radiation physics. 36(7):861–920, jul 1991.

[11] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[12] Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M. Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. In *International Conference on Learning Representations*, 2018.

[13] Hans Babovsky. *Die Boltzmann-Gleichung: Modellbildung-Numerik-Anwendungen*. Springer, 1998.

[14] Claude Bardos, François Golse, and David Levermore. Fluid dynamic limits of kinetic equations. I. Formal derivations. *Journal of Statistical Physics*, 63(1-2):323–344, April 1991.

[15] Claude W. Bardos, François Golse, and C. David Levermore. Fluid dynamic limits of kinetic equations ii convergence proofs for the boltzmann equation. *Communications on Pure and Applied Mathematics*, 46:667–753, 1993.

[16] Richard Barnard, Martin Frank, and Michael Herty. Optimal radiotherapy treatment planning using minimum entropy models. *Applied Mathematics and Computation*, 219(5):2668–2679, 2012.

[17] Timothy Barth and Dennis Jespersen. *The design and application of upwind schemes on unstructured meshes*.

[18] Leonard D Berkovitz. *Convexity and optimization in Rn*, volume 63. John Wiley & Sons, 2003.

[19] Prabhu Lal Bhatnagar, Eugene P Gross, and Max Krook. A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems. *Physical review*, 94(3):511, 1954.

[20] G. A. Bird. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*. Oxford University Press, 1972.

[21] Graeme Austin Bird. *Molecular gas dynamics and the direct simulation of gas flows*. Clarendon, 1994.

[22] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[23] Jiri Blazek. *Computational Fluid Dynamics (3. ed.)*. Butterworth-Heinemann, 2015.

[24] Christoph Börgers. Complexity of monte carlo and deterministic dose-calculation methods. *Physics in Medicine & Biology*, 43(3):517, 1998.

[25] François Bouchut, François Golse, and Mario Pulvirenti. *Kinetic equations and asymptotic theory*. Elsevier, 2000.

[26] Iain D Boyd, Gang Chen, and Graham V Candler. Predicting failure of the continuum fluid equations in transitional hypersonic flows. *Physics of fluids*, 7(1):210–219, 1995.

[27] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[28] Georg Brandl. Sphinx documentation. *URL http://sphinx-doc.org/*, 2021.

[29] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[30] Thomas A Brunner. Forms of approximate radiation transport. 6 2002.

[31] Thomas Camminady, Martin Frank, Kerstin Küpper, and Jonas Kusch. Ray effect mitigation for the discrete ordinates method through quadrature rotation. *J. Comput. Phys.*, 382:105–123, 2019.

[32] Thomas Camminady, Martin Frank, and Jonas Kusch. Highly uniform quadrature sets for the discrete ordinates method. *Nuclear Science and Engineering*, 2021.

[33] Kenneth M Case and Paul Frederick Zweifel. Linear transport theory. 1967.

[34] Carlo Cercignani. *The Boltzmann Equation and Its Applications*. Springer, New York, NY, 1988.

[35] Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 2022.

[36] Gianluca Ceruti and Christian Lubich. Time integration of symmetric and anti-symmetric low-rank matrices and Tucker tensors. *BIT Numerical Mathematics*, 60(3):591–614, 2020.

[37] Gianluca Ceruti and Christian Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT. Numerical Mathematics*, 62(1):23–44, 2022.

[38] Gianluca Ceruti, Christian Lubich, and Dominik Sulz. Rank-adaptive time integration of tree tensor networks. *arXiv:2201.10291*, 2022.

[39] Gianluca Ceruti, Christian Lubich, and Hanna Walach. Time integration of tree tensor networks. *SIAM Journal on Numerical Analysis*, 59(1):289–313, 2021.

[40] Scott Chacon and Ben Straub. *Pro git*. Springer Nature, 2014.

[41] M. T. Chahine. Foundations of radiation hydrodynamics (dimitri mihalas and barbara weibel mihalas). *Siam Review*, 29:648–650, 1987.

[42] Sydney Chapman, Thomas George Cowling, and David Burnett. *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge university press, 1990.

[43] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach, 2019.

[44] Gong Cheng, Peicheng Zhou, and Junwei Han. Rifd-cnn: Rotation-invariant and fisher discriminative convolutional neural networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[45] Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE international conference on computer vision*, pages 2857–2865, 2015.

[46] Y Cooper. The loss landscape of overparameterized neural networks, 2018.

[47] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels, 2012.

[48] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *Advances in neural information processing systems*, 2016.

[49] Michael M. Crockatt, Andrew J. Christlieb, C. Kristopher Garrett, and Cory D. Hauck. An arbitrary-order, fully implicit, hybrid kinetic solver for linear radiative transport using integral deferred correction. *Journal of Computational Physics*, 346(C), 10 2017.

[50] Michael M. Crockatt, Andrew J. Christlieb, C. Kristopher Garrett, and Cory D. Hauck. Hybrid methods for radiation transport using diagonally implicit runge–kutta and space–time discontinuous galerkin time integration. *Journal of Computational Physics*, 376(C), 1 2019.

[51] Michael M Crockatt, Andrew J Christlieb, and Cory D Hauck. Improvements to a class of hybrid methods for radiation transport: Nyström reconstruction and defect correction methods. *Journal of Computational Physics*, 422:109765, 2020.

[52] Raul E. Curto and Lawrence A. Fialkow. Recursiveness, positivity, and truncated moment problems. *Houston J. Math*, pages 603–635.

[53] Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *CoRR*, abs/1706.04859, 2017.

[54] Andrée Decarreau, Danielle Hilhorst, Claude Lemaréchal, and Jorge Navaza. Dual methods in entropy maximization. application to some problems in crystallography. *SIAM Journal on Optimization*, 2(2):173–197, 1992.

[55] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[56] Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[57] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

[58] Paul Adrien Maurice Dirac et al. *The principles of quantum mechanics*. Number 27. Oxford university press, 1981.

[59] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[60] Roland Duclous, Bruno Dubroca, and Martin Frank. A deterministic partial differential equation model for dose calculation in electron radiotherapy. *Physics in Medicine & Biology*, 55(13):3843, 2010.

[61] Weinan E, Jiequn Han, and Linfeng Zhang. Integrating machine learning with physics-based modeling, 2020.

[62] Weinan E, Chao Ma, Stephan Wojtowytsch, and Lei Wu. Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. *CoRR*, abs/2009.10713, 2020.

[63] Tarek Echekki. *The Emerging Role of Multiscale Methods in Turbulent Combustion*, pages 177–192. Springer Netherlands, Dordrecht, 2011.

[64] Lukas Einkemmer, Jingwei Hu, and Lexing Ying. An efficient dynamical low-rank algorithm for the boltzmann-bgk equation close to the compressible viscous flow regime. *SIAM Journal on Scientific Computing*, 43(5):B1057–B1080, 2021.

[65] Lukas Einkemmer and Christian Lubich. A quasi-conservative dynamical low-rank algorithm for the vlasov equation. *SIAM Journal on Scientific Computing*, 41(5):B1061–B1081, 2019.

[66] Lukas Einkemmer, Alexander Ostermann, and Chiara Piazzola. A low-rank projector-splitting integrator for the vlasov–maxwell equations with divergence correction. *Journal of Computational Physics*, 403:109063, 2020.

[67] Leonard Eyges. Multiple scattering with energy loss. *Physical Review*, 74(10):1534, 1948.

[68] V. Faber, Olaf M. Lubeck, and Andrew B. White. Superlinear speedup of an efficient sequential algorithm is not possible. *Parallel Computing*, 3(3):259–260, 1986.

[69] B. Fasel and D. Gatica-Perez. Rotation-invariant neoperceptron. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 336–339, 2006.

[70] Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. Rank diminishing in deep neural networks. *arXiv:2206.06072*, 2022.

[71] Florian Feppon and Pierre F. J. Lermusiaux. Dynamically orthogonal numerical schemes for efficient stochastic advection and lagrangian transport. *SIAM Review*, 60(3):595–625, 2018.

[72] Florian Feppon and Pierre FJ Lermusiaux. A geometric approach to dynamical model order reduction. *SIAM Journal on Matrix Analysis and Applications*, 39(1):510–538, 2018.

[73] Francisco Erivaldo Fernandes Junior and Gary G. Yen. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm and Evolutionary Computation*, 49:62–74, 2019.

[74] Matthias Fippel and Martin Soukup. A Monte Carlo dose calculation algorithm for proton therapy. *Medical physics*, 31(8):2263–2273, 2004.

[75] Martin Frank, Jonas Kusch, Thomas Camminady, and Cory D Hauck. Ray effect mitigation for the discrete ordinates method using artificial scattering. *Nuclear Science and Engineering*, 194(11):971–988, 2020.

[76] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[77] J Frenkel. *Wave mechanics, advanced general theory*, volume 1. Oxford, 1934.

[78] Barry Ganapol. Analytical benchmarks for nuclear engineering applications. *Case Studies in Neutron Transport Theory*, 2008.

[79] Barry Ganapol, R. Baker, J. Dahl, and Raymond Alcouffe. Homogeneous infinite media time-dependent analytic benchmarks for x-tm transport methods development. *Los Alamos National Laboratory*, 1999.

[80] Bin Gao and P-A Absil. A riemannian rank-adaptive method for low-rank matrix completion. *Computational Optimization and Applications*, 81(1):67–90, 2022.

[81] Alejandro L Garcia and Berni J Alder. Generation of the chapman–enskog distribution. *Journal of computational physics*, 140(1):66–70, 1998.

[82] C. Kristopher Garrett and Cory D. Hauck. A comparison of moment closures for linear kinetic transport equations: The line source benchmark. *Transport Theory and Statistical Physics*, 42(6-7):203–235, 2013.

[83] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. 2017.

[84] Stuart Geman, Elie Bienenstock, and René Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58, 01 1992.

[85] Jr. George James Minty. On the monotonicity of the gradient of a convex function. *Pacific Journal of Mathematics*, 14(1):243–247, 1963.

[86] Chris Gibson, James Neidhoefer, Stephen Cooper, Lindley Carlton, Chadwick Cox, and Charles Jorgensen. *Development and Flight Test of the X-43A-LS Hypersonic Configuration UAV*.

[87] Kent A Gifford, John L Horton, Todd A Wareing, Gregory Failla, and Firas Mourtada. Comparison of a finite-element multigroup discrete-ordinates code with monte carlo for radiotherapy calculations. *Physics in Medicine & Biology*, 51(9):2253, 2006.

[88] E. Godlewski and P.A. Raviart. *Hyperbolic Systems of Conservation Laws*. Mathématiques & applications. Ellipses, 1991.

[89] Robin Green. Spherical harmonic lighting: The gritty details. *Game Developers Conference*, 2003.

[90] V Grégoire and TR Mackie. State of the art on dose prescription, reporting and recording in intensity-modulated radiation therapy (icru report no. 83). *Cancer/Radiothérapie*, 15(6-7):555–559, 2011.

[91] Andreas Griewank and Andrea Walther. Introduction to automatic differentiation. *PAMM*, 2(1):45–49, 2003.

[92] Wei Guo and Jing-Mei Qiu. A low rank tensor representation of linear transport and nonlinear vlasov solutions and their associated flow maps. *J. Comput. Phys.*, 458(C), jun 2022.

[93] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

[94] John L. Gustafson. *Amdahl's Law*, pages 53–60. Springer US, Boston, MA, 2011.

[95] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I. Nonstiff problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993.

[96] Jiequn Han, Chao Ma, Zheng Ma, and Weinan E. Uniformly accurate machine learning-based hydrodynamic models for kinetic equations. *Proceedings of the National Academy of Sciences*, 116(44):21983–21991, 2019.

[97] Jiequn Han, Linfeng Zhang, Roberto Car, and Weinan E. Deep potential: A general representation of a many-body potential energy surface. *Communications in Computational Physics*, 23(3), 2018.

[98] Md. Zahid Hasan. The fidelity of dsmc method to analyze the aerothermodynamics of the pure continuum flow regime. *International Journal of Thermofluids*, 7-8:100047, 2020.

[99] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[100] C. Hauck, C. D. Levermore, and A. Tits. Convex duality and entropy-based moment closures: Characterizing degenerate densities. *2008 47th IEEE Conference on Decision and Control*, pages 5092–5097, 2008.

[101] Cory Hauck and Ryan McClarren. Positive pn closures. *SIAM Journal on Scientific Computing*, 32(5):2603–2626, 2010.

[102] Cory Hauck and Stefan Schnake. A predictor-corrector strategy for adaptivity in dynamical low-rank approximations, 2022.

[103] Cory D. Hauck and Ryan G. McClarren. A collision-based hybrid method for time-dependent, linear, kinetic transport equations. *Multiscale Modeling & Simulation*, 11(4):1197–1227, 2013.

[104] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[105] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks, 2018.

[106] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision*, pages 784–800, 2018.

[107] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks, 2017.

[108] Hartmut Hensel, Rodrigo Iza-Teran, and Norbert Siedow. Deterministic model for dose calculation in photon radiotherapy. *Physics in Medicine & Biology*, 51(3):675, 2006.

[109] Kenneth R Hogstrom, Michael D Mills, and Peter R Almond. Electron beam dose calculations. *Physics in Medicine & Biology*, 26(3):445, 1981.

[110] Lowell H Holway Jr. New statistical models for kinetic theory: methods of construction. *Physics of Fluids (1958-1988)*, 9(9):1658–1673, 1966.

[111] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[112] Juntao Huang, Yingda Cheng, Andrew J Christlieb, and Luke F Roberts. Machine learning moment closure models for the radiative transfer equation i: directly learning a gradient based closure. *Journal of Computational Physics*, page 110941, 2022.

[113] Juntao Huang, Yingda Cheng, Andrew J. Christlieb, Luke F. Roberts, and Wen-An Yong. Machine learning moment closure models for the radiative transfer equation ii: enforcing global hyperbolicity in gradient based closures, 2021.

[114] Juntao Huang, Zhiting Ma, Yizhou Zhou, and Wen-An Yong. Learning thermodynamically stable and galilean invariant partial differential equations for non-equilibrium flows. *Journal of Non-Equilibrium Thermodynamics*, 2021.

[115] Mi Huang. *Application of deterministic 3D SN transport driven dose kernel methods for out-of-field dose assessments in clinical megavoltage radiation therapy*. PhD thesis, Georgia Institute of Technology, 2015.

[116] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018.

[117] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán. Low-rank compression of neural nets: Learning the rank of each layer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8046–8056, 2020.

[118] Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training CNNs with low-rank filters for efficient image classification. In *International Conference on Learning Representations*, 2016.

[119] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[120] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[121] J.J. Jarrel and M.L. Adams. Discrete-ordinates quadrature sets based on linear discontinuous finite elements. *Proc. International Conference on Mathematics and Computational Methods applied to Nuclear Science and Engineering*, 2011.

[122] Xun Jia, Jan Schümann, Harald Paganetti, and Steve B Jiang. GPU-based fast Monte Carlo dose calculation for proton therapy. *Physics in Medicine & Biology*, 57(23):7783, 2012.

[123] Bangti Jin and Xiliang Lu. On the regularizing property of stochastic gradient descent. *Inverse Problems*, 35(1):015004, nov 2018.

[124] Michael Junk. Domain of Definition of Levermore's Five-Moment System. *Journal of Statistical Physics*, 93(5-6):1143–1167, December 1998.

[125] Michael Junk. Maximum entropy for reduced moment problems. *Mathematical Models and Methods in Applied Sciences*, 10(07):1001–1025, 2000.

[126] Michael Junk and Andreas Unterreiter. Maximum entropy moment systems and galilean invariance. *Continuum Mechanics and Thermodynamics*, 14:563–576, 2002.

[127] Ł ukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[128] Ilya Karlin. Derivation of regularized grad's moment system from kinetic equations: modes, ghosts and non-markov fluxes. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2118):20170230, 2018.

[129] D. Kershaw. Flux limiting nature's own way – a new method for numerical solution of the transport equation. 1976.

[130] Mikhail Khodak, Neil Tenenholtz, Lester Mackey, and Nicolo Fusi. Initialization and regularization of factorized neural layers. In *International Conference on Learning Representations*, 2021.

[131] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952.

[132] E. Kieri, C. Lubich, and H. Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM Journal on Numerical Analysis*, 54(2):1020–1038, 2016.

[133] Hyeji Kim, Muhammad Umar Karim Khan, and Chong-Min Kyung. Efficient neural network compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12569–12577, 2019.

[134] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications, 2015.

[135] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[136] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.

[137] O. Koch and C. Lubich. Dynamical tensor approximation. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2360–2375, 2010.

[138] Mikhail N Kogan. *Rarefied Gas Dynamics*. Plenum Press, 1969.

[139] M. Kreĭn, D. Louvish, and A. A. Nudel'man. The markov moment problem and extremal problems. 1977.

[140] Thomas Krieger and Otto A Sauer. Monte Carlo- versus pencil-beam-/collapsed-cone-dose calculation in a heterogeneous multi-layer phantom. 50(5):859–868, feb 2005.

[141] C. Kristopher Garrett, Cory Hauck, and Judith Hill. Optimization and large scale computation of an entropy-based moment closure. *Journal of Computational Physics*, 302:573 – 590, 2015.

[142] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[143] Kerstin Kuepper. *Models, numerical methods, and uncertainty quantification for radiation therapy*. PhD thesis, Universitätsbibliothek der RWTH Aachen, 2016.

[144] Jonas Kusch, Graham W Alldredge, and Martin Frank. Maximum-principle-satisfying second-order intrusive polynomial moment scheme. *The SMAI journal of computational mathematics*, 5:23–51, 2019.

[145] Jonas Kusch, Gianluca Ceruti, Lukas Einkemmer, and Martin Frank. Dynamical low-rank approximation for burgers' equation with uncertainty. *International Journal for Uncertainty Quantification*, 12(5):1–21, 2022.

[146] Jonas Kusch, Ryan G. McClarren, and Martin Frank. Filtered stochastic galerkin methods for hyperbolic equations, 2018.

[147] Jonas Kusch, Steffen Schotthöfer, Pia Stammer, Jannick Wolters, and Tianbai Xiao. Kit-rt: An extendable framework for radiative transfer and therapy, 2022.

[148] Jonas Kusch and Pia Stammer. A robust collision source method for rank adaptive dynamical low-rank approximation in radiation therapy. *arXiv preprint arXiv:2111.07160*, 2021.

[149] Jonas Kusch, Jannick Wolters, and Martin Frank. Intrusive acceleration strategies for uncertainty quantification for hyperbolic systems of conservation laws. *Journal of Computational Physics*, 419:109698, oct 2020.

[150] Andrey Kuzmin, Markus Nagel, Saurabh Pitre, Sandeep Pendyam, Tijmen Blankevoort, and Max Welling. Taxonomy and evaluation of structured compression of convolutional neural networks. 2019.

[151] Edward W Larsen, Moyed M Miften, Benedick A Fraass, and Iaın AD Bruinvis. Electron dose calculations using the method of moments. *Medical physics*, 24(1):111–125, 1997.

[152] Jean Bernard Lasserre. *Moments, Positive Polynomials and Their Applications*. IMPERIAL COLLEGE PRESS, 2009.

[153] Kaye D Lathrop. Ray effects in discrete ordinates equations. *Nuclear Science and Engineering*, 32(3):357–369, 1968.

[154] Kaye D Lathrop. Remedies for ray effects. *Nuclear Science and Engineering*, 45(3):255–268, 1971.

[155] V Lebedev, Y Ganin, M Rakhuba, I Oseledets, and V Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *International Conference on Learning Representations*, 2015.

[156] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[157] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[158] Huan Lei, Lei Wu, and Weinan E. Machine-learning-based non-newtonian fluid model with molecular fidelity. *Physical Review E*, 102(4), 10 2020.

[159] Randall J. LeVeque. *Numerical methods for conservation laws (2. ed.).* Lectures in mathematics. Birkhäuser, 1992.

[160] C. Levermore. Moment closure hierarchies for kinetic theories. *Journal of Statistical Physics*, 83:1021–1065, 1996.

[161] C. David Levermore. Entropy-based moment closures for kinetic equations. *Transport Theory and Statistical Physics*, 26(4-5):591–606, 1997.

[162] C. David Levermore, William J Morokoff, and BT Nadiga. Moment realizability and the validity of the navier–stokes equations for rarefied gas dynamics. *Physics of Fluids*, 10(12):3214–3226, 1998.

[163] E.E. Lewis and W.F. Miller. *Computational methods of neutron transport*. John Wiley and Sons, Inc, 1984.

[164] Elmer Eugene Lewis and Warren F Miller. Computational methods of neutron transport. 1984.

[165] Chong Li and C. J. Richard Shi. Constrained optimization based low-rank approximation of deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[166] Zhengyi Li, Bin Dong, and Yanli Wang. Learning invariance preserving moment closure model for boltzmann-bgk equation, 2021.

[167] Taweetham Limpanuparb and Josh Milthorpe. Associated legendre polynomials and spherical harmonics computation for chemistry applications. 2014.

[168] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[169] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured CNN pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019.

[170] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018.

[171] Gianluca Longoni. *Advanced quadrature sets and acceleration and preconditioning techniques for the discrete ordinates method in parallel computing environments*. PhD thesis, University of Florida, January 2004.

[172] Diego G. Loyola R, Mattia Pedergnana, and Sebastián Gimeno García. Smart sampling and incremental function learning for very large high dimensional data. *Neural Networks*, 78:75–87, 2016. Special Issue on "Neural Network Learning in Big Data".

[173] C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 54(1):171–188, 2014.

[174] C. Lubich, T. Rohwedder, R. Schneider, and B. Vandereycken. Dynamical approximation by hierarchical Tucker and tensor-train tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):470–494, 2013.

[175] C. Lubich, B. Vandereycken, and H. Walach. Time integration of rank-constrained Tucker tensors. *SIAM Journal on Numerical Analysis*, 56(3):1273–1290, 2018.

[176] Christian Lubich and Ivan Oseledets. A projector-splitting integrator for dynamical low-rank approximation, 2013.

[177] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression, 2017.

[178] G Marchuk and V I Lebedev. Numerical methods in the theory of neutron transport. 1 1986.

[179] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, dec 2016.

[180] Peter A Markowich, Christian A Ringhofer, and Christian Schmeiser. *Semiconductor equations*. Springer Science & Business Media, 2012.

[181] Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165):1–73, 2021.

[182] Kirk A Mathews. On the propagation of rays in discrete ordinates. *Nuclear science and engineering*, 132(2):155–180, 1999.

[183] Ryan G. McClarren and Cory D. Hauck. Robust and accurate filtered spherical harmonics expansions for radiative transfer. *Journal of Computational Physics*, 229(16):5597–5614, 2010.

[184] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[185] Siddhartha Mishra and T. Konstantin Rusch. Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. *CoRR*, abs/2005.12564, 2020.

[186] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017.

[187] Philipp Monreal. *Moment realizability and Kershaw closures in radiative transfer*. PhD thesis, Aachen, 2012. Prüfungsjahr: 2012. - Publikationsjahr: 2013; Aachen, Techn. Hochsch., Diss., 2012.

[188] JE Morel, TA Wareing, RB Lowrie, and DK Parsons. Analysis of ray-effect mitigation techniques. *Nuclear science and engineering*, 144(1):1–22, 2003.

[189] Kenneth Moreland and Ron Oldfield. Formal metrics for large-scale parallel performance. In Julian M. Kunkel and Thomas Ludwig, editors, *High Performance Computing*, pages 488–496, Cham, 2015. Springer International Publishing.

[190] Clément Mouhot and Lorenzo Pareschi. Fast algorithms for computing the boltzmann collision operator. *Mathematics of computation*, 75(256):1833–1852, 2006.

[191] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt, 2019.

[192] Taku Ohwada and Kun Xu. The kinetic scheme for the full-burnett equations. *Journal of computational physics*, 201(1):315–332, 2004.

[193] Edgar Olbrant and Martin Frank. Generalized fokker–planck theory for electron and photon transport in biological tissues: application to radiotherapy. *Computational and mathematical methods in medicine*, 11(4):313–339, 2010.

[194] Edgar Olbrant, Cory D. Hauck, and Martin Frank. A realizability-preserving discontinuous galerkin method for the m1 model of radiative transfer. *Journal of Computational Physics*, 231(17):5612–5639, 2012.

[195] Francisco Palacios, Juan Alonso, Karthikeyan Duraisamy, Michael Colonno, Jason Hicken, Aniket Aranake, Alejandro Campos, Sean Copeland, Thomas Economon, Amrita Lonkar, Trent Lukaczyk, and Thomas Taylor. *Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design*.

[196] V. Pavan. General entropic approximations for canonical systems described by kinetic equations. *Journal of Statistical Physics*, 142:792–827, 2011.

[197] Zhuogang Peng and Ryan G. McClarren. A high-order/low-order (holo) algorithm for preserving conservation in time-dependent low-rank transport calculations. *J. Comput. Phys.*, 447:110672, 2021.

[198] Zhuogang Peng, Ryan G. McClarren, and Martin Frank. A low-rank method for time-dependent transport calculations. *arXiv: Computational Physics*, 2019.

[199] J. Perl, J. Shin, J. Schumann, B. Faddegon, and H. Paganetti. TOPAS: An innovative proton Monte Carlo platform for research and clinical applications. *Medical Physics*, 39(11):6818–6837, November 2012.

[200] William A. Porteous, M. Paul Laiu, and Cory D. Hauck. Data-driven, structure-preserving approximations to entropy-based moment closures for kinetic equations, 2021.

[201] David Radice, Ernazar Abdikamalov, Luciano Rezzolla, and Christian D. Ott. A new spherical harmonics scheme for multi-dimensional radiation transport i. static matter configurations. *Journal of Computational Physics*, 242:648–669, 2013.

[202] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[203] Mohsen Sadr, Manuel Torrilhon, and M. Hossein Gorji. Gaussian process regression for maximum entropy distribution. *Journal of Computational Physics*, 418:109644, 2020.

[204] Mohsen Sadr, Qian Wang, and M. Hossein Gorji. Coupling kinetic and continuum using data-driven maximum entropy distribution. *Journal of Computational Physics*, 444:110542, 2021.

[205] Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6655–6659, 2013.

[206] Themistoklis P. Sapsis and Pierre F.J. Lermusiaux. Dynamically orthogonal field equations for continuous stochastic dynamical systems. *Physica D: Nonlinear Phenomena*, 238(23):2347–2360, 2009.

[207] Erhard Schmidt. Zur theorie der linearen und nichtlinearen integralgleichungen. i. teil: Entwicklung willkürlicher funktionen nach systemen vorgeschriebener. *Mathematische Annalen*, 63:433–476, 1907.

[208] Steffen Schotthöfer, Tianbai Xiao, Martin Frank, and Cory Hauck. Structure preserving neural networks: A case study in the entropy closure of the boltzmann equation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19406–19433. PMLR, 17–23 Jul 2022.

[209] Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20051–20063. Curran Associates, Inc., 2022.

[210] Damien Scieur, Vincent Roulet, Francis Bach, and Alexandre d'Aspremont. Integration methods and accelerated optimization algorithms. In *Advances In Neural Information Processing Systems*, 2017.

[211] Benjamin Seibold and Martin Frank. Starmap—a second order staggered grid method for spherical harmonics moment equations of radiative transfer. *ACM Trans. Math. Softw.*, 41(1), oct 2014.

[212] EM Shakhov. Generalization of the krook kinetic relaxation equation. *Fluid Dynamics*, 3(5):95–96, 1968.

[213] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P. Namboodiri. Play and prune: Adaptive filter pruning for deep model compression. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3460–3466. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[214] Sidak Pal Singh, Gregor Bachmann, and Thomas Hofmann. Analytic insights into structure and rank of neural network Hessian maps. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[215] Nimit Sharad Sohoni, Christopher Richard Aberger, Megan Leszczynski, Jian Zhang, and Christopher Ré. Low-memory neural network training: A technical report. *arXiv:1904.10631*, 2019.

[216] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[217] G. Strang. *Linear Algebra and Learning from Data*. Wellesley-Cambridge Press, 2019.

[218] Henning Struchtrup and Manuel Torrilhon. Regularization of grad's 13 moment equations: Derivation and linear analysis. *Physics of Fluids*, 15(9):2668–2680, 2003.

[219] D. P. Summy and D. P. Summy. On the five-moment hamburger maximum entropy reconstruction. *Jour. Stat. Phys.*, 172(01):854–879, 08 2018.

[220] Quanhua Sun, Iain D Boyd, and Graham V Candler. A hybrid continuum/particle approach for modeling subsonic, rarefied gas flows. *Journal of Computational Physics*, 194(1):256–277, 2004.

[221] John Tencer. Ray effect mitigation through reference frame rotation. *Journal of Heat Transfer*, 138(11), 2016.

[222] J Tervo, P Kolmonen, M Vauhkonen, LM Heikkinen, and JP Kaipio. A finite-element model of electron transport in radiation therapy and a related inverse problem. *Inverse Problems*, 15(5):1345, 1999.

[223] J Tervo, M Vauhkonen, and E Boman. Optimal control model for radiation therapy inverse planning applying the boltzmann transport equation. *Linear Algebra and its Applications*, 428(5-6):1230–1249, 2008.

[224] Jouko Tervo and Pekka Kolmonen. Inverse radiotherapy treatment planning model applying boltzmann-transport equation. *Mathematical Models and Methods in Applied Sciences*, 12(01):109–141, 2002.

[225] T. Tieleman and G. Hinton. Lecture. msprop: Divide the gradient by a running average of its recent magnitud, 2012.

[226] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Compressing recurrent neural network with tensor train. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4451–4458. IEEE, 2017.

[227] Hsue-Shen Tsien. Superaerodynamics, mechanics of rarefied gases. *Journal of the Aeronautical Sciences*, 13(12):653–664, 1946.

[228] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

[229] Oleg N Vassiliev, Todd A Wareing, Ian M Davis, John McGhee, Douglas Barnett, John L Horton, Kent Gifford, Gregory Failla, Uwe Titt, and Firas Mourtada. Feasibility of a multigroup deterministic solution method for three-dimensional radiotherapy dose calculations. *International Journal of Radiation Oncology* Biology* Physics*, 72(1):220–227, 2008.

[230] Oleg N Vassiliev, Todd A Wareing, John McGhee, Gregory Failla, Mohammad R Salehpour, and Firas Mourtada. Validation of a new grid-based boltzmann equation solver for dose calculation in radiotherapy with photon beams. *Physics in Medicine and Biology*, 55(3):581–598, jan 2010.

[231] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[232] V. Venkatakrishnan. *On the accuracy of limiters and convergence to steady state solutions*.

[233] Hongyi Wang, Saurabh Agarwal, and Dimitris Papailiopoulos. Pufferfish: communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 3:365–386, 2021.

[234] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[235] Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating filters for faster deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 658–666, 2017.

[236] Hans-Peter Wieser, Eduardo Cisternas, Niklas Wahl, Silke Ulrich, Alexander Stadler, Henning Mescher, Lucas-Raphael Müller, Thomas Klinge, Hubert Gabrys, Lucas Burigo, et al. Development of the open-source dose calculation and optimization toolkit matrad. *Medical physics*, 44(6):2556–2568, 2017.

[237] M. K. Woo and J. R. Cunningham. The validity of the density scaling method in primary electron transport for photon and electron beams. *Medical Physics*, 17(2):187–194, 1990.

[238] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4820–4828, 2016.

[239] Tianbai Xiao. Kinetic. jl: A portable finite volume toolbox for scientific and neural computing. *Journal of Open Source Software*, 6(62):3060, 2021.

[240] Tianbai Xiao and Martin Frank. Using neural networks to accelerate the solution of the boltzmann equation. *Journal of Computational Physics*, 443:110521, 2021.

[241] Tianbai Xiao, Steffen Schotthöfer, and Martin Frank. Predicting continuum breakdown with deep neural networks, 2022.

[242] Kun Xu. *Direct modeling for computational fluid dynamics: construction and application of unified gas-kinetic schemes*. World Scientific, 2015.

[243] Huanrui Yang, Minxue Tang, Wei Wen, Feng Yan, Daniel Hu, Ang Li, Hai Li, and Yiran Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 678–679, 2020.

[244] Ning Yang, Chao Tang, and Yuhai Tu. Stochastic gradient descent introduces an effective landscape-dependent regularization favoring flat solutions, 2022.

[245] Jianbo Ye, Xin Lu, Zhe Lin, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *International Conference on Learning Representations*, 2018.

[246] Qi Ye, Sachan Devendra, Felix Matthieu, Padmanabhan Sarguna, and Neubig Graham. When and why are pre-trained word embeddings useful for neural machine translation. In *HLT-NAACL*, 2018.

[247] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.*, 120:143001, Apr 2018.