Structure Preserving Neural Networks: A Case Study in the Entropy Closure of the Boltzmann Equation

Steffen Schotthöfer [1] **Tianbai Xiao** [1] **Martin Frank** [1] **Cory D. Hauck** [2 3]

Abstract

In this paper, we explore applications of deep learning in statistical physics. We choose the Boltzmann equation as a typical example, where neural networks serve as a closure to its moment system. We present two types of neural networks to embed the convexity of entropy and to preserve the minimum entropy principle and intrinsic mathematical structures of the moment system of the Boltzmann equation. We derive an error bound for the generalization gap of convex neural networks which are trained in Sobolev norm and use the results to construct data sampling methods for neural network training. Numerical experiments demonstrate that the neural entropy closure is significantly faster than classical optimizers while maintaining sufficient accuracy.

1. Introduction

The Boltzmann equation laid the foundation for statistical physics, and describes the evolution of one-particle probability density function $f(t,x,v)$, where $\{x \in \mathbb{R}^3, v \in \mathbb{R}^3\}$ are the coordinates in phase space, in a many-particle system

$$\partial_t f + v \cdot \nabla_x f = Q(f). \tag{1}$$

The linear collision operator $Q(f)$ describes interactions between particles and with the background medium. The high dimensionality and nonlinearity brings tremendous challenge for large scale numerical solutions.
Moment methods eliminate the dependency of the phase space on the velocity variable by computing the moment hierarchy of the Boltzmann equation

$$\partial_t u + \nabla_x \cdot \langle v m f \rangle = \langle m Q(f) \rangle \,, \tag{2}$$

where $u(t,x)$ is the moment vector and $m(v)$ is a vector of velocity dependent basis functions. The bracket operator $\langle \cdot \rangle$ denotes the integral over the velocity space $\int_V \cdot dv$.
The moment system (2) usually requires a closure due to the existence of high-order unclosed terms in the advection operator. The so called M_N moment closure preserves entropy dissipation, hyperbolicity, positivity of the solution, conservation of mass and Boltzmann's H-Theorem, which are important for the stability and physical accuracy of numerical solvers. The M_N closure is constructed by solving a convex constrained optimization problem based on the entropy minimization principle (Levermore, 1997),

$$h(u) = \min_{g \in F_m} \langle \eta(g) \rangle \quad \text{s.t. } u = \langle m g \rangle \,. \tag{3}$$

The minimizer f_u of Eq. (3) closes the moment system (2). The minimal entropy closure gives a nonlinear reconstruction of f differs form Grad-type moment closures, that reconstruct f as a linear combination of the basis m. The set of all moments u for which the above problem is solvable is is called the realizable set $\mathcal{R}$ and h is the minimal entropy corresponding to the moment u. The entropy density is denoted by $\eta : \mathbb{R}_+ \to \mathbb{R}$. Although the M_N closure is methodically superior simple truncation closures, it is by far more expensive to compute. The authors of (Garrett & Hauck, 2013) have demonstrated, that in a high performance implementation, more than 80% of the computational time of the whole simulation is required for the solution of the entropy minimization problem, since the optimization problem needs to be solved in each grid cell at each time step of the simulation. This motivates the development of a neural network surrogate model to accelerate the M_N closure.

1.1. Previous Work

In (Alldredge et al., 2019), the authors have shown, that any convex approximation to the minimal entropy functional preserves the intrinsic structure of the Boltzmann moment system. Previously, convex splines and feed forward neural networks have been used in (Porteous et al., 2021) to build a data driven surrogate model for the minimal entropy closure.

[1]Department of Applied and Numerical Mathematics, Karlsruhe Institute of Technology, Karlsruhe, Germany [2]Computer Science and Mathematics Division,Oak Ridge National Laboratory, Oak Ridge, TN, USA [3]Department of Mathematics (Joint Faculty), University of Tennessee, Knoxville, TN, USA. Correspondence to: Steffen Schotthöfer <steffen.schotthoefer@kit.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Recently, much effort has been put into finding deep learning based moment closures for Grad type moment systems of the Boltzmann equation. In (Han et al., 2019; E et al., 2020a), 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 f. Second, they learn directly the reduced model for the set of generalized moments. The authors of (Li et al., 2021) directly model the kinetic density f to close the moment system using an U-Net that considers global information of the solution field. The authors of (Huang et al., 2021) close the moment system by learning the spatial gradient of the highest order moment.

1.2. Contributions

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 (Amos et al., 2017) 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 derived Lagrange multipliers and the reconstructed moments.

The second approach of in 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 an error bound of the generalization gap of input convex neural networks trained in Sobolev norm. Additionally, we provide insights in the topology of the realizable set $\mathcal{R}$, the data-to-solution map of the minimal entropy closure and its implications on the consruction of data driven closures.

We combine the above insights to propose a data sampling strategy for the neural network based minimal entropy closuers. Finally, we demonstrate our findings in numerical simulation test cases, where we compare accuracy and computational efficiency of the closures to a traditional numerical solver that provides a benchmark.

2. Moment methods for kinetic equations

The Boltzmann equation is an integro-differential equation model defined on a seven-dimensional phase space. With the nonlinear five-fold integral, it is challenging to solve accurately and efficiently. The well-known moment method encode the velocity dependence of the Boltzmann equation by multiplication with a vector of velocity dependent basis functions $m(v)$, that consists of polynomials up to order N and subsequent integration over v. The solution of the resulting moment equation is the moment vector

$$u(t,\mathbf{x}) = \langle m(\mathbf{v}) f(t,x,\mathbf{v}) \rangle \,. \tag{4}$$

The moment vector satisfies the system of transport equations (5) which is called moment system. By construction, the advection operator depends on f and thus, the moment system is unclosed. Moment methods aim to find a meaningful closure for this system. Since the kinetic equation dissipates entropy and fulfills a local entropy dissipation law, one can close the system by choosing the reconstructed kinetic density f_u out of the set of all possible functions $F_m = \{g(v) > 0 : \langle mg \rangle < \infty\}$, that fulfill $u(t,x) = \langle mg \rangle$ as the one with minimal entropy h and is formulated in Eq. (3). The minimal value of the objective function is denoted by $h(u) = \langle \eta(f_u) \rangle$ and f_u is the minimizer of Eq. (3), which we use to close the moment system

$$\partial_t u + \nabla_x \cdot \langle v m(v) f_u \rangle = \langle m(v) Q(f_u) \rangle \,. \tag{5}$$

The set of all moments corresponding to kinetic densities $f > 0$ is called the realizable set

$$\mathcal{R} = \{u : \langle mg \rangle = u,\, g \in F_m\} \,. \tag{6}$$

There does not always exists a solution for the minimal entropy problem (Hauck et al., 2008). However, a solution can only exist for $u \in \mathcal{R}$ and if it exists, it is unique and of the form

$$f_u = \eta_*'(\alpha_u \cdot m). \tag{7}$$

where the Lagrange multiplier $\alpha_u : \mathbb{R}^{\tilde{N}} \to \mathbb{R}^{\tilde{N}}$ maps u to the solution of the convex dual problem

$$\alpha_u = \underset{\alpha \in \mathbb{R}^{\tilde{N}}}{\operatorname{argmax}} \{\alpha \cdot u - \langle \eta_*(\alpha \cdot m) \rangle\} \tag{8}$$

and η_* is the Legendre dual of η. By the strong duality of the minimal entropy problem, the maximum of (8) equals the minimum of (3) and we can write at the optimal point (u, α_u)

$$h(u) = \alpha_u \cdot u - \langle \eta_*(\alpha_u \cdot m) \rangle \,. \tag{9}$$

The twice differentiable and convex function $h(u)$ serves as the entropy of the moment system (Alldredge et al., 2019). We can recover the moment u by using first order optimality conditions

$$\frac{\mathrm{d}}{\mathrm{d}\alpha_u} h = u - \langle m \eta_*'(\alpha_u \cdot m) \rangle = 0 \tag{10}$$

which yields also Eq. (7), since

$$u = \langle m f_u \rangle = \langle m \eta_*'(\alpha_u \cdot m) \rangle \,. \tag{11}$$

This yields the inverse of the solution map α_u of the dual optimization problem. Furthermore, the derivative of h recovers the optimal Lagrange multipliers of Eq. (8),

$$\frac{\mathrm{d}}{\mathrm{d}u} h = \alpha_u. \tag{12}$$

A traditional numerical method to solve the moment system therefore consists of an iterative discretization scheme for the moment system (5) and a Newton optimizer for the dual minimal entropy optimization problem in Eq. (8). The drawback of the method is the high computational cost associated with the Newton solver. The optimization problem in Eq. (8) needs to be solved in each grid cell at every time step of the kinetic solver. The computational effort to solve the minimal entropy optimization problem grows over proportionately with the order N of the moment basis $m(v)$. Using three basis functions, the optimizer requires 80% of the computation time and 87% when using seven basis functions (Kristopher Garrett et al., 2015).

3. Structure-preserving entropy closures using neural networks

The following section tackles the challenge of solving the minimal entropy closure computationally efficiently while preserving the key structural properties of the Boltzmann equation. We propose two neural network architectures, which map a given moment vector to the solution of the minimal entropy problem, replacing the costly Newton solver that is used in traditional numerical methods. Whereas a Newton solver requires the inversion of a near singular Hessian matrix multiple times, the usage of a neural network needs a comparatively small amount of fast tensor operations to compute the closure.

3.1. Data structure and dimension reduction

The structure of the underlying data is crucial for the construction of meaningful machine learning models. We state useful facts about the realizable set $\mathcal{R}$ and the moment to entropy map, which are used to construct the neural network architectures. For detailed explanations, we refer to Appendix B.

The realizable set $\mathcal{R}$ is in general a convex cone, which is bounded and convex for a fixed moment of order zero u_0. Thus we define the normalized realizable set $\mathcal{R}^n$ and the reduced normalized realizable set $\mathcal{R}^r$ as

$$\mathcal{R}^n = \{u \in \mathcal{R} : u_0 = 1\} \subset \mathbb{R}^{\tilde{N}}, \tag{13}$$

$$\mathcal{R}^r = \left\{ u^r \in \mathbb{R}^N : [1, {u^r}^T]^T \in \mathcal{R}^n \right\} \subset \mathbb{R}^{\tilde{N}-1}. \tag{14}$$

We denote normalized moments and reduced normalized moments as

$$u^n = \frac{u}{u_0} = [1, u_1^r, \dots, u_N^r]^T \in \mathbb{R}^{\tilde{N}}, \tag{15}$$

$$u^r = [u_1^r, \dots, u_N^r]^T \in \mathbb{R}^{\tilde{N}-1}. \tag{16}$$

We establish some relations between the Lagrange multiplier α_u, the Lagrange multiplier of the normalized moment α_u^n and of the reduced normalized moment α_u^r,

$$\alpha_u^n = \left[\alpha_{u,0}^r, \alpha_{u,1}^r, \dots, \alpha_{u,N}^r\right]^T \in \mathbb{R}^{\tilde{N}}, \tag{17}$$

$$\alpha_u^r = \left[\alpha_{u,1}^r, \dots, \alpha_{u,N}^r\right]^T \in \mathbb{R}^{\tilde{N}-1}. \tag{18}$$

The Lagrange multiplier of order zero $\alpha_{u,0}^r$ can be computed with

$$\alpha_{u,0}^r = -\ln(\langle \exp(\alpha_u^r \cdot m^r) \rangle), \tag{19}$$

and Lagrange multiplier of the non-normalized moment is given by

$$\alpha_u = [\alpha_{u,0}^r + \ln(u_0), \alpha_{u,1}^r, \dots, \alpha_{u,N}^r]^T, \tag{20}$$

where we use the Maxwell-Boltzmann entropy $\eta(f) = f\log(f) - f$.

3.2. Neural network approximations of the entropy closure

We present a twice differentiable, convex neural network approximation h_θ^n to the minimal entropy functional $h^n = h(u^n)$ of the normalized moment system. A neural network approximation, which we denote by $\mathcal{N}_\theta$, constructed with these properties in mind preserves the structural properties of the moment system. Using the intrinsic structure of the dual problem (8), we train the networks on the entropy h, the Lagrange multiplier α_u and the moment reconstruction u. Assuming the neural network is trained, i.e. it approximates the entropy h^n sufficiently well, we have the following relations,

$$h_\theta^n = \mathcal{N}_\theta(u^r) \approx h^n, \tag{21}$$

$$\alpha_\theta^r = \frac{\mathrm{d}}{\mathrm{d}u^r}\mathcal{N}_\theta(u^r) \approx \frac{\mathrm{d}}{\mathrm{d}u^r} h^n = \alpha_u^r, , \tag{22}$$

$$\alpha_{\theta,0}^r = -\ln(\langle \exp(\alpha_\theta^r \cdot m^r) \rangle) \approx \alpha_{u,0}^r \tag{23}$$

$$f_\theta = \eta_*'(\alpha_\theta^n \cdot m) \approx \eta_*'(\alpha_u^n \cdot m) = f_u, \tag{24}$$

$$u_\theta^n = \langle m \eta_*'(\alpha_\theta^n \cdot m) \rangle \approx \langle m \eta_*'(\alpha_u^n \cdot m) \rangle = u^n, \tag{25}$$

by using Eq. (7), Eq. (12), Eq. (19) and the definition of the moment vector. The neural network architecture and its integration in a moment solver is displayed in Fig. 1 and implementation details can be found in Appendix C.1.

The idea of the second neural network closure for the dual minimal entropy problem in Eq. (8), makes use of the following characterization of multivariate convex functions

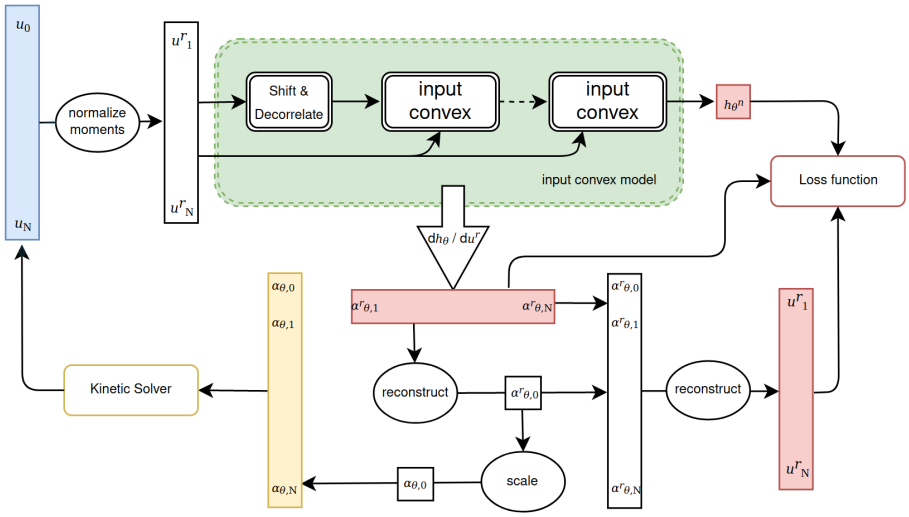


Figure 1. Input convex neural network closure. Model input vectors are depicted in blue. Red vectors are outputs, on which the model is trained on. When the trained model is employed, the yellow solution vector is used to construct the flux for the kinetic solver .

via montonicity of their gradients (Berkovitz, 2003). Let $U \subset \mathbb{R}^N$ be a convex set. A function $G : U \to \mathbb{R}^d$ is *monotonic*, if and only if $(G(x) - G(y)) \cdot (x - y) \geq 0$ for all $x, y \in U$. Let $g : U \to \mathbb{R}$ differentiable. Then g is convex, if and only if $\nabla g : U \to \mathbb{R}^d$ is monotonic. As a consequence, if the mapping $u^n \to \alpha^n_u$ is monotonic for all $u^n \in \mathcal{R}^n$, then the corresponding entropy functional is h^n is convex in u^n. A trained monotonic neural network, that approximates the moment to Lagrange multiplier map, fulfills the following relations,

$$\alpha^r_\theta = \mathcal{N}_\theta(u^r) \approx \alpha^r_u, \tag{26}$$
$$\alpha^r_{\theta,0} = -\ln(\langle \exp(\alpha^r_\theta \cdot m^r) \rangle) \approx \alpha^r_{u,0}, \tag{27}$$
$$f_\theta = \eta'_*(\alpha^n_\theta \cdot m) \approx \eta'_*(\alpha^n_u \cdot m) = f_u, \tag{28}$$
$$u^n_\theta = \langle m \eta'_*(\alpha^n_\theta \cdot m) \rangle \approx \langle m \eta'_*(\alpha^n_u \cdot m) \rangle = u^n, \tag{29}$$
$$h^n_\theta = \alpha^n_\theta \cdot u^n - \langle \eta_*(\alpha^n_\theta \cdot m) \rangle \approx h^n.s \tag{30}$$

The network architecture, end to end training loop and integration in the solver are displayed in Appendix C.2 in Fig. 8.

We briefly examine the structural properties of a convex neural network based entropy closure. The primary output of the input convex neural network (21) directly takes on the role of an surrogate entropy of the moment system, since it is convex by design. In the proof of the hyperbolicity property, which is conducted in (Levermore, 1996) for α_u and u as the system variable, h'', respectively h''_*, must be symmetric positive definite, which is given for the input convex neural network. Strict convexity of the entropy functional h is the crucial requirement for the related properties entropy dissipation and the H-theorem (Levermore, 1996) as well. The monotonic neural network implicitly defines a convex entropy only after converged training, as seen in Eq. (30). The invariant range property of f_θ depends solely on the range of η'_*. By definition of the Maxwell-Boltzmann entropy, the neural network based entropy closure is of invariant range, since $f_\theta(v) = \exp(\alpha^n_\theta \cdot m(v)) > 0$. Interchanging the entropy functional by a neural network does not affect the conservation property of the moment system.

4. Generalization gap and data generation

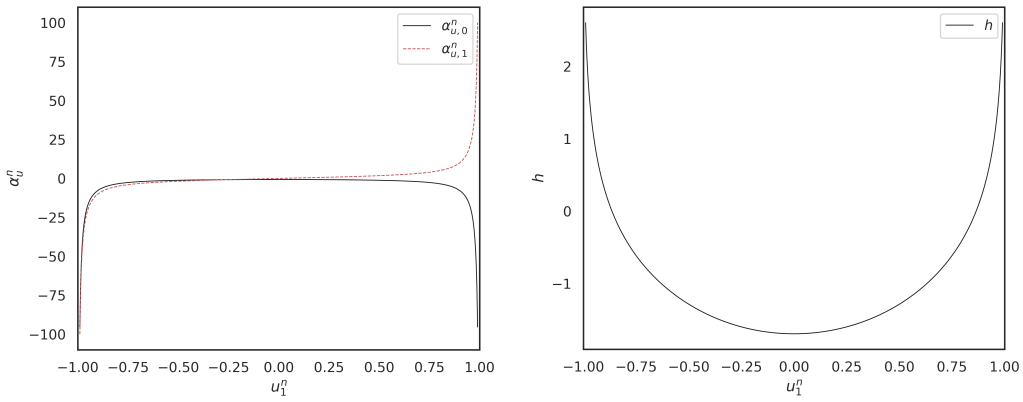
In this section, we construct a local bound to the generalization gap for the approximated gradient of a input convex neural network, which yields the Lagrange multiplier α_u in case of the neural entropy closure. Based on this error bound, we present a training data generation strategy.

In contrast to many applications of neural networks, the minimal entropy closure is a self contained problem with a clear data to solution map. Furthermore, the set of potential inputs $\mathcal{R}^r$ to the neural network is bounded and convex. This provides more options to sample training data than common machine learning applications. The training data distribution heavily influences the trained model and therefor the generalization gap (E et al., 2020b). The generalization gap is defined as

$$\left| L\left(X_T, \theta^*\right) - L\left(X, \theta^*\right) \right|, \tag{31}$$

where $\theta^* = \min_\theta L\left(X_T, \theta, \right)$ is the set of parameters, that minimizes the training loss. The generalization gap describes the performance difference of the neural network with parameters θ^* between the training data X_T and any unseen world data X. Thus we are left with a modelling decision about the data generation.

A popular method for generating data for neural network models that interact with numerical differential equation



a) α_u^n over u_1^n b) h over u_1^n

Figure 2. Data to solution maps for the $1D$ M_1 closure

solvers is to generate the training data by direct simulation, see e.g. (Huang et al., 2021; Xiao & Frank, 2021; Lei et al., 2020). However, the generated data is strongly biased towards the chosen simulation setups and the models might generalize poorly.

4.1. Generalization gap for input convex neural networks data sampling strategy

In this sections we present our findings to the generalization gap for the derivative approximation of a convex neural network $\mathcal{N}_\theta$ that approximates a convex function f^* and derive a data sampling strategy. The network is trained (at least) in Sobolev norm, i.e. the training loss reads

$$L\left(X_T, \theta^*\right) = \frac{1}{|T|} \sum_{i \in T} \Big(\|f^*(x_i) - \mathcal{N}_{\theta^*}(x_i)\|_2^2 + \|\nabla f^*(x_i) - \nabla \mathcal{N}_{\theta^*}(x_i)\|_2^2 \Big), \tag{32}$$

when evaluating the loss over the whole data set. In the following, we assume that the network is trained, i.e. $L\left(X_T, \theta^*\right) = 0$. Thus we have for all $x_i \in X_T$

$$f^*(x_i) = \mathcal{N}_\theta(x_i), \quad \nabla f^*(x_i) = \nabla \mathcal{N}_\theta(x_i). \tag{33}$$

Furthermore, let the sampling domain $X \subset \mathbb{R}^d$ be convex and bounded and the neural network be convex by design. We are interested in the generalization gap of the derivative neural network with respect to its input variable. To this end, we consider the local generalization gap of the neural network when using $d+1$ training data points $X_d = \{x_0, \dots, x_d\}$, if the sampling space $X \subset \mathbb{R}^d$ has dimension d. Let $\mathcal{C}(X_d)$ be the convex hull of X_d and $x^* \in \mathcal{C}(X_d)$, which we call the point of interest. We assume w.l.o.g $x^* = 0$; if this does not hold, one can consider the shifted setting $\mathcal{C}^\dagger(X_d) = \mathcal{C}(X_d) - x^*$, $f^\dagger = f^*(\cdot + x^*)$, $x^\dagger = x - x^*$ instead. Using the characterization of a monotonic function, we define the set A

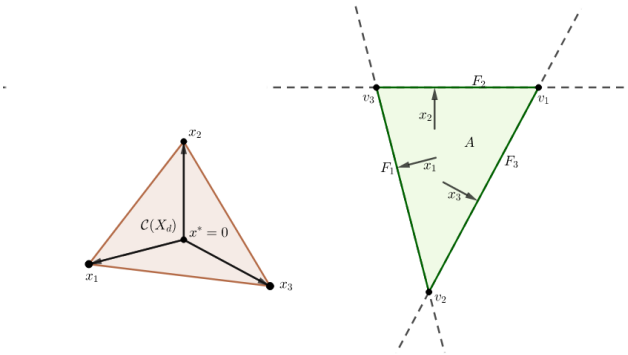


Figure 3. Illustration of the convex hull of the training points $\mathcal{C}(X_d)$ (left) and the set of feasible gradients A (right) for $d = 2$. The normal vectors to the faces F_i are the vectors of the training points x_i.

$$A = \left\{ v \in \mathbb{R}^d | v \cdot x_i \leq \nabla f^*(x_i) \cdot x_i, i = 0, \dots, d \right\} \tag{34}$$

which is the dual polygon defined by the gradients at the sampling points and the point of interest and can be seen in Fig. 3. A contains all values which the gradient of a convex function that has fixed gradients at the sampling points $x \in X_d$ can attain at the point of interest x^*.

Theorem 4.1. *Let f^* be convex, $x^* = 0$ the point of interest in the interior of $\mathcal{C}(X_d)$. Then A is a bounded polyhedron, whith $d+1$ faces, defined by $F_i = \left\{ v \in \mathbb{R}^d | v \cdot x_i = \nabla f^*(x_i) \cdot x_i \right\}$ and vertices $v_i = \bigcap_{j \neq i} F_j$.*

The proof can be found in Appendix D.1. A direct consequence of Theorem 4.1 is an local upper bound for the

generalization gap of the gradient of an input convex network trained on a given training data set X_T

$$\|\nabla f^*(x) - \nabla \mathcal{N}_\theta(x)\| \leq \text{diam}(A_{x^*}), \tag{35}$$

where A_{x^*} is the polyhedron of feasible gradients w.r.t the point of interest x^* and the local training points X_d. A first conclusion is that the $\text{diam}(A)$ does not depend on the distance between the point of interest and any of the local training data points X_d, since by definition of A in Eq. (34), one can divide by the norm of $x_i - x^*$ on both sides of the inequality for the boundary of A. Thus in the following we assume normalized x_i.
The following theorem gives a more precise representation of $\text{diam}(A_{x^*})$.

Theorem 4.2. *Let A be defined by Eq. (34). Let the relative vectors x_i have unit length and v_i is the vertex opposing the face F_i. The matrix $X_i = [x_0^n, \dots, x_{i-1}^n, x_{i+1}^n, \dots, x_d^n]^T$ contain the vectors of normalized sampling points relative to the point of interest x_*, i.e. $x_i^n = x_i / \|x_i\|_2$.*
Furthermore, let $b_i = [\nabla f^*(x_0) \cdot x_0^n, \dots, \nabla f^*(x_{i-1}) \cdot x_{i-1}^n, f^*(x_{i+1}) \cdot x_{i+1}^n, \dots, \nabla f^*(x_d) \cdot x_d^n]^T$ be a vector. Under the assumptions of Theorem 4.1, the vertex v_i is given by

$$X_i v_i = b_i \tag{36}$$

Additionally, we can estimate the distance between two vertices v_i and v_j by

$$\|v_i - v_j\|_2 \leq \left(\left\|X_i^{-1}\right\| + \left\|X_j^{-1}\right\|\right) C_{x^*}, \tag{37}$$

where $C_{x^*} = \max_{k,l} \|\nabla f^*(x_k) - \nabla f^*(x_l)\|_2$ and $\left\|X_i^{-1}\right\|$ denotes the corresponding operator norm of X_i^{-1}.

The proof can be found in Appendix D.2. A direct consequence is

$$\begin{aligned} \|\nabla f^*(x) - \nabla \mathcal{N}_\theta(x)\| &\leq \text{diam}(A_{x^*}) \\ &\leq \max_{i,j \in A} \left(\left\|X_i^{-1}\right\| + \left\|X_j^{-1}\right\|\right) C_{x^*}. \end{aligned} \tag{38}$$

First, $\text{diam}(A) \to \infty$, if $\text{dist}(x^*, \partial \mathcal{C}(X_d)) \to 0$, since the normals of at least two neighboring boundaries of A 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 $\text{diam}(A)$ depends only on the norm distance of $\nabla f^*(x_i)$, $i = 0, \dots, d$, which is encoded in the definition of C_{x^*}. The smaller the norm distance of the gradients of the sample points, the smaller gets C_{x^*}.
By chosing the maximum over all local error bounds for a given set of training points, we can establish a worst case generalization error for a input convex neural network trained on this set. In the application case of the neural entropy closure, the input data set is $\mathcal{R}^r$, which is bounded and convex. Thus, the argument is viable everywhere except at the boundary of the convex hull of all training data, assuming a suitable distribution of training data points. Remark, that the polyhedron can be shrunken by including more training points to the set X_d.

4.2. Sampling of the normalized realizable set

The entropy minimization problem of Eq. (8) becomes increasingly difficult to solve near the boundary of the realizable set $\partial \mathcal{R}$ (Alldredge et al., 2012). Close to $\partial \mathcal{R}^n$, the condition number of the Hessian matrix of the entropy functional h^n with respect to α_u^n in Eq. (9)

$$H(\alpha_u^n) = \langle m \times m \eta_*(\alpha_u^n \cdot m) \rangle, \tag{39}$$

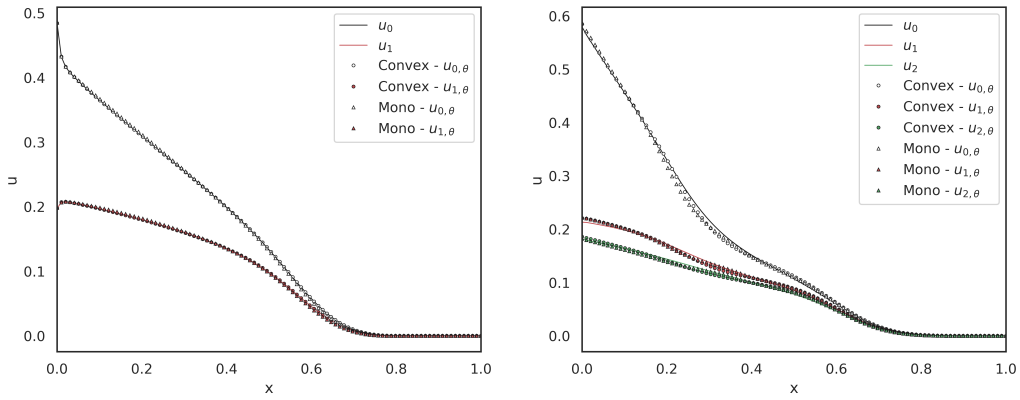
can become arbitrarily large, which causes numerical solvers to fail rather unforgivingly. At the boundary $\partial \mathcal{R}^n$, $H(\alpha_u^n)$ singular, and the minimal entropy problem has no solution. In the space of Lagrange multipliers, this translates to α_u^n growing beyond all bounds, which leads to numerical instabilities when computing the reconstruction of u. The simplest case of the minimal entropy closure, the $1D$ M_1 closure with $\mathcal{R}^r = [-1, 1]$, already incorporates these difficulties, see Fig. 2.
No matter if we sample u and then compute α_u^n or vice versa, a sampling strategy must incorporate a meaningful distance measure to $\partial \mathcal{R}^n$. Since we do not have a characterization of $\partial \mathcal{R}^n$ in general (Lasserre, 2009), we use the condition number of $H(\alpha_u^n)$ to establish an implicit notion of distance to $\partial \mathcal{R}^r$. The Hessian is symmetric and positive definite, thus condition number is the ratio of the biggest and smallest eigenvalue. It is singular at $\partial \mathcal{R}^r$, so the smallest possible eigenvalue $\lambda_{\min}$ is 0, and we use $\lambda_{\min}$ to measure the distance to the boundary of the realizable set. As a consequence of the above considerations, we generate the training data X_T by sampling reduced Lagrange multipliers in a set

$$B_{M,\tau} = \{\alpha_u^r : \|\alpha_u^r\| < M \cap \lambda_{\min}(H(\alpha_u^n)) > \tau\} \tag{40}$$

using rejection sampling. The Lagrange multipliers are sampled uniformly with a low-discrepancy sampling method from $B_{M,\tau}$ to reduce the generalization gap. Low-discrepancy sampling methods have a positive impact for neural network training, especially for high data dimensions (Mishra & Rusch, 2020). A detailed evaluation of the data sampling strategy is given in Appendix E.

5. Numerical results

In this section, we present numerical results and investigate the performance of the neural entropy closure. The training performance on for the $1D$ M_1, $1D$ M_2 and $2D$ M_1 closures for both neural network architectures using training



a) $1D$ M_1 closure b) $1D$ M_2 closure

Figure 4. Moments of the solution of the an-isotropic inflow test case at $t_f = 0.7$. Comparison of the benchmark solution, the input convex closure and the monotonic closure.

data generated using the sampling strategy of Sec. 4 are discussed in Appendix F. The converged mean squared error on the validation set is around $\mathcal{O}(1\text{e}{-4})$ to $\mathcal{O}(1\text{e}{-6})$.
The neural network based closures are orders of magnitude more computationally compared to a Newton based solver. A Newton solver needs to compute and invert the Hessian of the dual minimal entropy closure of Eq. (39), which requires the evaluation of the velocity integral, in every descent step. Depending on the condition and the distance to $\partial\mathcal{R}$, the optimizer requires up to $\mathcal{O}(1\text{e}3)$ iterations for the closure of a single moment u. Thus the neural networks yield speedups of up to $\mathcal{O}(1\text{e}3)$ compared to the Newton based closure. This yields a speedup of the solver of over 80%, depending on the grid size and order of the closure. For bigger spatial grids and higher moment orders, the speedup increases. A quantitative discussion the computational efficiency in Appendix G.
We conduct synthetic tests to evaluate the generalization gap on the whole realizable set $\mathcal{R}^n$ for both network architectures trained on identical data sets. The relative error of the neural network prediction of h_θ, α_θ^n and u_θ^n is measured. The relative error of the monotonic architecture is larger by half an order of magnitude compared to the input convex network. Note, that the local error bound for the generalization gap is only strictly applicable for the input convex architecture. Moreover, the generalization error for u_θ^n is smallest and of order $\mathcal{O}(1\text{e}{-3})$, whereas the error in α_θ^n biggest and of order $\mathcal{O}(1\text{e}{-2})$. Reason for this is, that the reconstruction map $\alpha_\theta^n \to u_\theta^n$ is well conditioned. A detailed discussion can be found in Appendix H.

5.1. An-isotropic inflow into an isotropically scattering, homogeneous medium in 1D

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

$$\partial_t f + v\partial_x f = \sigma \int_{-1}^{1} \frac{1}{2}\left(f(v_*) - f(v)\right)\, \mathrm{d}v_* - \tau f, \quad (41)$$

where σ is a scattering coefficient and τ is an absorbtion coefficient. The corresponding moment model becomes

$$\partial_t u + \partial_{\mathbf{x}} \langle \mathbf{v} m f_u \rangle = \sigma \langle m Q(f_u) \rangle - \tau u \quad (42)$$

$$f_u = \eta_*(\alpha_\theta \cdot m) \quad (43)$$

The initial condition of the computational domain is set as vacuum with $f(0, x, \mathbf{v}) = \epsilon$, where $1 \gg \epsilon > 0$ is a safety treshold, since the normalized vector u^n is undefined for $u_0 = 0$ and $0 \in \partial\mathcal{R}$. An an-isotropic inflow condition is imposed at the left boundary of domain with

$$f(t > 0, x = 0, v) = \begin{cases} 0.5 & \text{if } v > 0 \\ 0 & \text{if } v \leq 0, \end{cases} \quad (44)$$

and the right hand side boundary is equipped with a farfield condition. The domain is resolved using a structured grid in space using a kinetic upwind scheme (Kristopher Garrett et al., 2015) and an explicit Euler scheme in time. The benchmarking solver uses a Newton based optimizer with linesearch to compute the minimal entropy closure. The CFL number is set to 0.4 to avoid that the finite volume update steps outside the realizable domain $\mathcal{R}$, (Olbrant et al., 2012). The detailed computational setup can be found in Table 3. The solution profiles at final time $t_f = 0.7$ of the neural network based entropy closed moment system and the reference solver are presented in Fig. 4 for the M_1 and M_2 system. We can see that the systems dynamics are well captured by both neural network architectures.
The errors of the neural network based simulations in comparison with the benchmark solution are displayed in Fig. 12 and the relative errors in Fig. 13 in Appendix I.1. One can

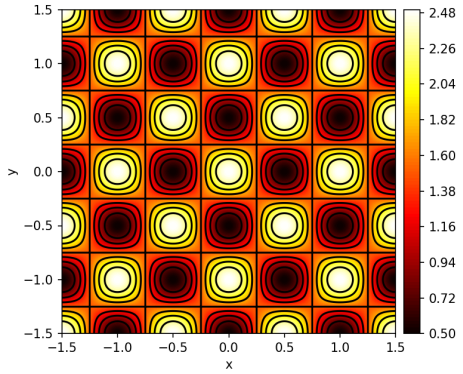


Figure 5. Snapshot of the benchmark solution of the $2D$ M_1 test case. The colorbar indicates the value of u_0 at a grid cell.

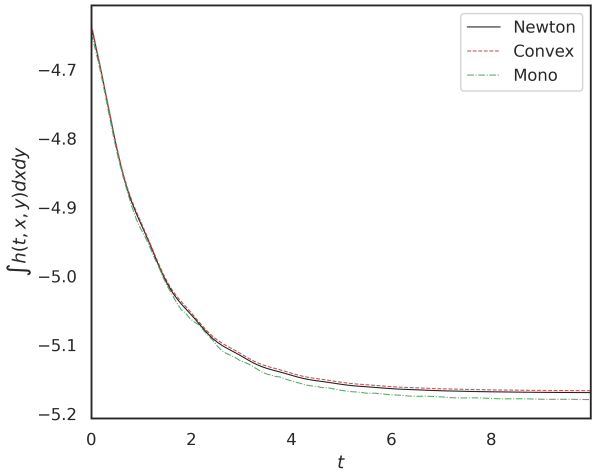


Figure 6. Comparison of the total entropy of the systems using neural network closures and Newton based closures. For better readability only a fraction of the time steps are displayed.

observe in both test cases, that the errors exhibited by the input convex network are one order of magnitude lower than the errors of the monotonic network. Overall, the $M1$ network architectures stay mostly within 1% relative error to the benchmark solution and the $M2$ network closures within 5%. In both test cases, the region which exceeds this error level is the tip of the wave-front, where the appearing moments are very close to $\partial\mathcal{R}$, where the minimal entropy closure is very ill conditioned and the networks are hard to train. A detailed error explanation and a convergence study is given in Appendix I.1.

5.2. Particles in a 2D non scattering medium with periodic initial conditions

We consider a rectangular domain in two spatial dimensions. This test case considers a non scattering and non absorbing medium, i.e. $\sigma = \tau = 0$, for which the Boltzmann moment system minimal entropy closure reads

$$\begin{aligned} \partial_t u + \partial_{\mathbf{x}} \langle \mathbf{v}_{\mathbf{x}} m f_u \rangle + \partial_{\mathbf{y}} \langle \mathbf{v}_{\mathbf{y}} m f_u \rangle = 0 \\ f_u = \eta_*(\alpha_u \cdot m) \end{aligned} \tag{45}$$

The Boltzmann equation is equipped with periodic initial conditions that translate to the M_1 moment equations

$$u_0 = 1.5 + \cos(2\pi\mathbf{x})\cos(2\pi\mathbf{y}), \qquad (\mathbf{x},\mathbf{y}) \in X, \tag{46}$$

$$u_1 = 0.3u_0, \qquad u_2 = 0.3u_0. \tag{47}$$

Periodic boundary conditions are imposed on the equations to get a well posed system of equations. Figure 5 in Appendix I.2 shows a snapshot of the flow field computed with the benchmark solver and Fig. 14 displays snapshots of the relative error at each grid cell of the flow field at the same iteration as the benchmark solver in Fig. 5. The relative errors of both neural networks exhibit periodic behavior and are in the range of 1% or lower. Similarly to the $1D$ test cases, the input convex architecture is again slightly more accurate than the monotonic counterpart. Lastly, we analyze the total entropy of the system at each time step. Due to the periodic boundary conditions and $\sigma = \tau = 0$, we have no particle sinks or sources in the system and the system is closed. We have chosen the upwind scheme for the numerical flux of the moment system, which is an entropy dissipating scheme. Figure 6 shows the dissipation of the mathematical entropy of the system over time and compares the entropy of the reference solution with the two neural network architectures. All methods are entropy dissipating, however, the input convex neural network exhibits a smaller difference to the reference entropy. We can conclude, that the neural network based hybrid solver preserves the structural properties of the reference system and computes the numerical solution within reasonable accuracy. Convergence analysis and additional discussion can be found in Appendix I.2.

6. Summary and conclusion

In this work we model the minimal entropy closure of the Boltzmann moment system with two different deep neural network architectures and thereby successfully construct a hybrid differential equation solver consisting of neural network and numerical components. With enforced convexity of the neural network based closures, crucial structural properties of the underlying system of partial differential equations are conserved, which allows for physically accurate simulations while increasing training performance of the network. Convexity of the neural networks and the fact, that both primary network output and the first derivative is used in the training loss enables a local analysis and bound of the generalization gap. The error analysis is used to construct an optimal data sampling strategy for the minimal

entropy closure for arbitrary spatial dimension and arbitrary length of the velocity basis. Synthetic and numerical test cases confirm the accuracy of the of the hybrid solver compared to a traditional benchmark solution. The neural networks accelerate the solver by 80% in our test cases. Future work will focus on solving the challenges of generating data and predicting accurate closures near the boundary of the realizable set.

Acknowledgements

The work of Steffen Schotthöfer, Tianbai Xiao and Martin Frank funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) in the frame of the priority programme SPP 2298 "Theoretical Foundations of Deep Learning" – Projectnumber 441826958. The work of Cory Hauck is sponsored by the Office of Advanced Scientific Computing Research, U.S. Department of Energy, and performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

References

Alldredge, G. W., Hauck, C. D., and Tits, A. L. 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. doi: 10.1137/11084772X. URL https://doi.org/10.1137/11084772X.

Alldredge, G. W., Frank, M., and Hauck, C. D. A regularized entropy-based moment method for kinetic equations. *SIAM Journal on Applied Mathematics*, 79(5): 1627–1653, 2019. doi: 10.1137/18M1181201. URL https://doi.org/10.1137/18M1181201.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 146–155. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/amos17b.html.

Berkovitz, L. D. *Convexity and optimization in Rn*, volume 63. John Wiley & Sons, 2003.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441.

Brunner, T. Forms of approximate radiation transport. 2002.

Camminady, T., Frank, M., Küpper, K., and Kusch, J. Ray effect mitigation for the discrete ordinates method through quadrature rotation. *J. Comput. Phys.*, 382:105–123, 2019.

Cercignani, C. *The Boltzmann Equation and Its Applications*. Springer, New York, NY, 1988. ISBN 978-1-4612-1039-9. doi: https://doi.org/10.1007/978-1-4612-1039-9.

Chahine, M. T. Foundations of radiation hydrodynamics (dimitri mihalas and barbara weibel mihalas). *Siam Review*, 29:648–650, 1987.

Chen, Y., Shi, Y., and Zhang, B. Optimal control via neural networks: A convex approach, 2019.

Curto, R. E. and Fialkow, L. A. Recursiveness, positivity, and truncated moment problems. *Houston J. Math*, pp. 603–635.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. Sobolev training for neural networks. *CoRR*, abs/1706.04859, 2017. URL http://arxiv.org/abs/1706.04859.

E, W., Han, J., and Zhang, L. Integrating machine learning with physics-based modeling, 2020a. URL https://arxiv.org/abs/2006.02619.

E, W., Ma, C., Wojtowytsch, S., and Wu, L. Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. *CoRR*, abs/2009.10713, 2020b. URL https://arxiv.org/abs/2009.10713.

Garrett, C. K. and Hauck, C. D. 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. doi: 10.1080/00411450.2014.910226. URL https://doi.org/10.1080/00411450.2014.910226.

Han, J., Ma, C., Ma, Z., and E, W. Uniformly accurate machine learning-based hydrodynamic models for kinetic equations. *Proceedings of the National Academy of Sciences*, 116(44):21983–21991, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1909854116. URL https://www.pnas.org/content/116/44/21983.

Hauck, C., Levermore, C. D., and Tits, A. Convex duality and entropy-based moment closures: Characterizing degenerate densities. *2008 47th IEEE Conference on Decision and Control*, pp. 5092–5097, 2008.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Huang, J., Cheng, Y., Christlieb, A. J., and Roberts, L. F. Machine learning moment closure models for the radiative transfer equation iii: enforcing hyperbolicity and physical characteristic speeds, 2021.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Junk, M. Domain of Definition of Levermore's Five-Moment System. *Journal of Statistical Physics*, 93 (5-6):1143–1167, December 1998. doi: 10.1023/B:JOSS.0000033155.07331.d9.

Junk, M. Maximum entropy for reduced moment problems. 1999. URL http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-6135.

Junk, M. and Unterreiter, A. Maximum entropy moment systems and galilean invariance. 2001. URL http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-11866.

Kershaw, D. Flux limiting nature's own way – a new method for numerical solution of the transport equation. 1976.

Kreĭn, M., Louvish, D., and Nudel'man, A. A. The markov moment problem and extremal problems. 1977.

Kristopher Garrett, C., Hauck, C., and Hill, J. Optimization and large scale computation of an entropy-based moment closure. *Journal of Computational Physics*, 302:573 – 590, 2015. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2015.09.008. URL http://www.sciencedirect.com/science/article/pii/S0021999115005896.

Lasserre, J. B. *Moments, Positive Polynomials and Their Applications*. IMPERIAL COLLEGE PRESS, 2009. doi: 10.1142/p665. URL https://www.worldscientific.com/doi/abs/10.1142/p665.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. *Efficient BackProp*, pp. 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_3. URL https://doi.org/10.1007/978-3-642-35289-8_3.

Lei, H., Wu, L., and E, W. Machine-learning-based non-newtonian fluid model with molecular fidelity. *Physical Review E*, 102(4), Oct 2020. ISSN 2470-0053. doi: 10.1103/physreve.102.043309. URL http://dx.doi.org/10.1103/PhysRevE.102.043309.

Levermore, C. Moment closure hierarchies for kinetic theories. *Journal of Statistical Physics*, 83:1021–1065, 1996.

Levermore, C. D. Entropy-based moment closures for kinetic equations. *Transport Theory and Statistical Physics*, 26(4-5):591–606, 1997. doi: 10.1080/00411459708017931. URL https://doi.org/10.1080/00411459708017931.

Lewis, E. and Miller, W. *Computational methods of neutron transport*. John Wiley and Sons, Inc, 1984. URL http://inis.iaea.org/search/search.aspx?orig_q=RN:17089238.

Li, Z., Dong, B., and Wang, Y. Learning invariance preserving moment closure model for boltzmann-bgk equation, 2021. URL https://arxiv.org/abs/2110.03682.

Markowich, P., Ringhofer, C., and Schmeiser, C. Semiconductor equations. 1990.

Mishra, S. and Rusch, T. K. Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. *CoRR*, abs/2005.12564, 2020. URL https://arxiv.org/abs/2005.12564.

Monreal, P. *Moment realizability and Kershaw closures in radiative transfer*. PhD thesis, Aachen, 2012. URL https://publications.rwth-aachen.de/record/210538. Prüfungsjahr: 2012. - Publikationsjahr: 2013; Aachen, Techn. Hochsch., Diss., 2012.

Olbrant, E., Hauck, C. D., and Frank, M. A realizability-preserving discontinuous galerkin method for the m1 model of radiative transfer. *Journal of Computational Physics*, 231(17):5612–5639, 2012. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2012.03.002. URL https://www.sciencedirect.com/science/article/pii/S0021999112001362.

Pavan, V. General entropic approximations for canonical systems described by kinetic equations. *Journal of Statistical Physics*, 142:792–827, 2011.

Porteous, W. A., Laiu, M. P., and Hauck, C. D. Data-driven, structure-preserving approximations to entropy-based moment closures for kinetic equations, 2021.

Sadr, M., Torrilhon, M., and Gorji, M. H. Gaussian process regression for maximum entropy distribution. *Journal of*

Computational Physics, 418:109644, 2020. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109644. URL https://www.sciencedirect.com/science/article/pii/S0021999120304186.

Schotthöfer, S. neuralentropyclosures. https://github.com/CSMMLab/neuralEntropyClosures, 2021.

Schotthöfer, S., Wolters, J., Kusch, J., Xiao, T., and Stammer, P. Kit-rt. https://github.com/CSMMLab/KiT-RT, 2021.

Xiao, T. and Frank, M. Using neural networks to accelerate the solution of the boltzmann equation. *Journal of Computational Physics*, 443:110521, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110521. URL https://www.sciencedirect.com/science/article/pii/S0021999121004162.

Xiao, T., Liu, C., Xu, K., and Cai, Q. A velocity-space adaptive unified gas kinetic scheme for continuum and rarefied flows. *Journal of Computational Physics*, 415: 109535, 2020.

A. Introduction to kinetic theory and moment methods

A.1. Kinetic theory

In many applications, a macroscopic description of the physical systems is no longer applicable and one has to rely on a more general description, which is given by kinetic equations such as the Boltzmann equation. Example include neutron transport (Lewis & Miller, 1984), radiative transport (Chahine, 1987) and semiconductors (Markowich et al., 1990) and rarefied gas dynamics (Cercignani, 1988). The Boltzmann equation is a high dimensional integro-differential equation, with phase space dependency on space and particle velocity. This high dimensionality of the phase space presents a severe computational challenge for large scale numerical simulations.
The Boltzmann equation describes the space-time evolution of the one-particle kinetic density function $f(t, \mathbf{x}, \mathbf{v})$ in a many-particle system

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = Q(f). \tag{48}$$

The phase space consists of time $t > 0$, space $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^3$, and particle velocity $\mathbf{v} \in \mathbf{V} \subset \mathbb{R}^3$. The left-hand side of the equation describes particle transport, where the advection operator $\mathbf{v} \cdot \nabla_{\mathbf{x}}$ describes the movement of the particle density with velocity $\mathbf{v}$ in the spatial directions. The integral operator $Q(f)$ on the right hand side of the equation models interaction 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

$$Q(f)(\mathbf{v}) = \int_{\mathbf{V}} \mathcal{B}(\mathbf{v}_*, \mathbf{v}) \left[f(\mathbf{v}_*) - f(\mathbf{v}) \right] d\mathbf{v}_*, \tag{49}$$

where the collision kernel $\mathcal{B}(\mathbf{v}_*, \mathbf{v})$ models the strength of collisions at different velocities. If the interactions among particles are considered, the collision operator becomes nonlinear, which is used in the application of gas dynamics. Well-posedness of Eq. (48) requires appropriate initial and boundary conditions. The Boltzmann equation is a first-principles model based on direct modeling. It possesses some key structural properties, which are intricately related to the physical processes and its mathematical existence and uniqueness theory. We refer the reader to (Alldredge et al., 2019; Levermore, 1996), where these properties are studied in detail.

A.2. Moment methods

Several numerical methods for solving the Boltzmann equation have been proposed, and are typically distinguished by the way of discretization of the velocity space. The first major set methods are called nodal methods, (Lewis & Miller, 1984; Camminady et al., 2019; Xiao et al., 2020), which evaluate the velocity space at specific quadrature points, which yields a system of equations only coupled by the integral scattering operator. While computationally efficient, these methods suffer from numerical artifacts, which are called ray effects (Camminady et al., 2019).
Moment methods encode the velocity dependence of the Boltzmann equation by multiplication with a vector of velocity dependent basis functions $m(\mathbf{v}) \in \mathbb{R}^{\tilde{N}}$, that consists of polynomials up to order N and subsequent integration over $\mathbf{V}$. In one spatial dimension, usually we have $\tilde{N} = N + 1$, whereas in higher spatial dimensions $\tilde{N}$ equals the number of basis functions up to order N. The solution of the resulting moment equation is the moment vector $u \in \mathbb{R}^{\tilde{N}}$ and is calculated by

$$u(t, \mathbf{x}) = \langle m(\mathbf{v}) f(t, \mathbf{x}, \mathbf{v}) \rangle \,. \tag{50}$$

Common choices for the basis functions are monomials, spherical harmonics or hermite polynomials, depending on the application. The moment vector satisfies the system of transport equations

$$\partial_t u(t, \mathbf{x}) + \nabla_{\mathbf{x}} \cdot \langle \mathbf{v} m(\mathbf{v}) f \rangle = \langle m(\mathbf{v}) Q(f) \rangle \,, \tag{51}$$

which is called moment system. By construction, the advection operator depends on f and thus, the moment system has $\tilde{N} + 1$ free variables, i.e. $u \in \mathbb{R}^{\tilde{N}}$ and f, and only $\tilde{N}$ equations and is therefore underdetermined, i.e. unclosed. Moment methods aim to find a meaningful closure for this system. The choice of the closure has severe implications on physical and numerical accuracy as well as the mathematical structure of the closure. The simplest closure is a truncation closure, with a linear reconstruction of f given by the first $\tilde{N}$ moments. However, this closure exhibits unphysical behaviour like oscillations in some flow regimes (Brunner, 2002). In this work, we focus on the minimal entropy closure, which on the reserves all above mentioned properties of the Boltzmann equation and accurately represents the modeled physics in a wide range of flow regimes (Garrett & Hauck, 2013). However, this comes at the expense of high computational cost.

B. Data Structure and Dimension Reduction

In the following we consider the primal and dual minimal entropy closure optimization problem, review the characterization of the realizable set as well as a dimension reduction and finally describe helpful relations between the moment u, Lagrange multiplier α_u, the entropy functional h and the corresponding variables of reduced dimension.
The minimal entropy optimization problem in Eq. (8) and the set of realizable moments $\mathcal{R}$ is studied in detail by by (Alldredge et al., 2019; Levermore, 1997; Curto & Fialkow; Hauck et al., 2008; Junk, 1998; 1999; Junk & Unterreiter, 2001; Pavan, 2011). The characterization of $\mathcal{R}$ uses the fact that the realizable set is uniquely defined by its boundaries (Kreĭn et al., 1977). First we remark that the realizable set $\mathcal{R} \subset \mathbb{R}^{\tilde{N}}$ of the entropy closure problem of order N is generally an unbounded convex cone. To see this consider the moment of order zero, $u_0 = \langle f \rangle$ for any kinetic density function $f \in F_m$, which can obtain values in $(0, \infty)$. For a fixed moment of order zero u_0, the subset of the corresponding realizable moments of higher order is bounded and convex (Kershaw, 1976; Monreal, 2012). Consequently, we consider the normalized realizable set $\mathcal{R}$ and the reduced normalized realizable set $\mathcal{R}^r$

$$\mathcal{R}^n = \{u \in \mathcal{R} : u_0 = 1\} \subset \mathbb{R}^{\tilde{N}}, \tag{52}$$

$$\mathcal{R}^r = \left\{ u^r \in \mathbb{R}^N : [1, {u^r}^T]^T \in \mathcal{R}^n \right\} \subset \mathbb{R}^{\tilde{N}-1}, \tag{53}$$

which are both bounded and convex (Kershaw, 1976; Monreal, 2012). This approach is also used in computational studies of numerical solvers of the minimal entropy closure problem (Alldredge et al., 2012). We denote normalized moments and reduced normalized moments as

$$u^n = \frac{u}{u_0} = [1, u_1^r, \dots, u_N^r]^T \in \mathbb{R}^{\tilde{N}}, \tag{54}$$

$$u^r = [u_1^r, \dots, u_N^r]^T \in \mathbb{R}^{\tilde{N}-1}. \tag{55}$$

We establish some relations between the Lagrange multiplier α_u, the Lagrange multiplier of the normalized moment α_u^n and of the reduced normalized moment α_u^r,

$$\alpha_u^n = \left[\alpha_{u,0}^r, \alpha_{u,1}^r, \dots, \alpha_{u,N}^r\right]^T \in \mathbb{R}^{\tilde{N}}, \tag{56}$$

$$\alpha_u^r = \left[\alpha_{u,1}^r, \dots, \alpha_{u,N}^r\right]^T \in \mathbb{R}^{\tilde{N}-1}. \tag{57}$$

We define the reduced moment basis, which contains all moments of order $n = 1, \dots, N$, as

$$m^r(\mathbf{v}) = [m_1(\mathbf{v}), \dots, m_{\tilde{N}}(\mathbf{v})]^T, \tag{58}$$

since $m_0(v) = 1$ is the basis function of order 0. For the computations we choose the Maxwell-Boltzmann entropy and a monomial basis, however the relations can be analogously computed for other choices of entropy function and moment basis. The Maxwell-Boltzmann entropy has the following definition, Legendre dual and derivative.

$$\eta(z) = z\ln(z) - z, \qquad z \in D = \mathbb{R}_+ \tag{59}$$

$$\eta'(z) = \ln(z), \qquad z \in D = \mathbb{R}_+ \tag{60}$$

$$\eta_*(y) = \exp(y), \qquad y \in \mathbb{R} \tag{61}$$

$$\eta_*'(y) = \exp(y), \qquad y \in \mathbb{R} \tag{62}$$

In one spatial dimension, we have $v = v \in \mathbb{R}$ and a monomial basis is given by $m(v) = [1, v, v^2, \dots]$. Assuming knowledge about the Lagrange multiplier α_u^r of the reduced normalized moment we can derive an expression for α_0^r using the definition of the moment of order zero,

$$1 = u_0^n = \langle m_0 \eta_*'(\alpha_u^n \cdot m) \rangle = \langle \exp(\alpha_u^n \cdot m) \rangle = \left\langle \exp(\alpha_u^r \cdot m^r) \exp(\alpha_{u,0}^r \cdot m_0) \right\rangle, \tag{63}$$

which we can transform to

$$\alpha_{u,0}^r = -\ln(\langle \exp(\alpha_u^r \cdot m^r) \rangle) \tag{64}$$

using $m_0(\mathbf{v}) = 1$. This yields the complete Lagrange multiplier α_u^n of the complete normalized moment vector u^n. Finally, we use a scaling relation (Alldredge et al., 2012) to recover the Lagrange multiplier of the original moment vector u, by considering again the definition of the normalized moment

$$u^n = \langle m \exp(\alpha_u^n \cdot m) \rangle = \left\langle m \exp(\alpha_u^r \cdot m^r) \exp(\alpha_{u,0}^r) \right\rangle \tag{65}$$

and multiply both sides with $u_0 > 0$

$$u = \left\langle \exp(\alpha_u^r \cdot m^r) \exp(\alpha_{u,0}^r) u_0 \right\rangle = \left\langle \exp(\alpha_u^r \cdot m^r) \exp(\alpha_{u,0}^r + \ln(u_0)) \right\rangle, \tag{66}$$

which yields the original Lagrange multiplier α_u

$$\alpha_u = [\alpha_{u,0}^r + \ln(u_0), \alpha_{u,1}^r, \dots, \alpha_{u,N}^r]^T. \tag{67}$$

This also implies that $\alpha_{u,i} = \alpha_{u,i}^r$ for all $i = 1, \dots, \tilde{N}$. For completeness, the entropy of the normalized moments $h^n = h(u^n)$ and the entropy $h(u)$ of the original moments have the relation

$$\begin{aligned} h(u) &= \alpha \cdot u - \langle \exp(\alpha \cdot m) \rangle && (68)\\ &= u_0 \left(\alpha^n \cdot u^n + \ln(u_0)\right) - \langle \exp(\alpha^n \cdot m + \ln(u_0)) \rangle && (69)\\ &= u_0 \left(\alpha^n \cdot u^n + \ln(u_0)\right) - \langle \exp(\alpha^n \cdot m \rangle u_0) && (70)\\ &= u_0 h(u^n) + u_0 \ln(u_0), && (71) \end{aligned}$$

where we use Eq. (9) and (67). These scaling relations enable a dimension reduction for faster neural network training. Furthermore, we use these relations to integrate the neural network models, which are trained on $\mathcal{R}^r$, into the kinetic solver that operates on $\mathcal{R}$.

C. Architecture of the neural network based closures

In the following, we present the architectural details of the input convex and monotonic neural networks used as the closure of the Boltzmann moment system. The implementation of the networks can be found in (Schotthöfer, 2021).

C.1. Input convex neural network approximation of the entropy functional h^n

Convex neural networks have been inspected in (Amos et al., 2017), where the authors propose several deep neural networks that are strictly convex with respect to the input variables by design. The design is led by the following principles (Boyd & Vandenberghe, 2004) that yield sufficient conditions to build a convex function. First, a positive sum of convex functions is convex. Second, let $f : \mathbb{R}^n \to \mathbb{R}$ be the concatenation of the functions $h : \mathbb{R}^k \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^k$. 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. Applying these conditions to the definition of a layer of a neural network, yields that all entries of the weight matrix W_k must be positive in all layers except the first. Furthermore, the activation function of each layer must be convex. The authors of (Chen et al., 2019) have shown, that such a network architecture with ReLU activations is able dense in the space of convex functions. They first show that an input convex network can approximate any maximum of afine functions, which itself can approximate any convex function in the limit of infinite layers. However, in practice it turns out that very deep networks with positive weights have difficulties to train. The authors of (Amos et al., 2017) therefore modify the definition of a hidden layer to

$$\begin{aligned} z_k &= \sigma(W_k^z z_{k-1} + W_k^x x + b_k^z), \qquad k = 2, \dots, M, && (72)\\ z_k &= \sigma(W_k^x u + b_k^z), \qquad k = 1, && (73) \end{aligned}$$

where W_k^z must be non-negative, and W_k^x may attain arbitrary values. We choose the strictly convex softplus function

$$\sigma : \mathbb{R} \to \mathbb{R}_+, \quad \sigma(y) = \ln(\exp(y) + 1) \tag{74}$$

as the layer activation function for $k = 1, \dots, M-1$ and a linear activation for the last layer, since we are dealing with a regression task. This leads to an at least twice continuously differentiable neural network. Non-negativity can be achieved by applying a projection onto $\mathbb{R}_+$ to the elements of W_k^z after a weight update. Next, we modify the first layer in Eq. (73) to

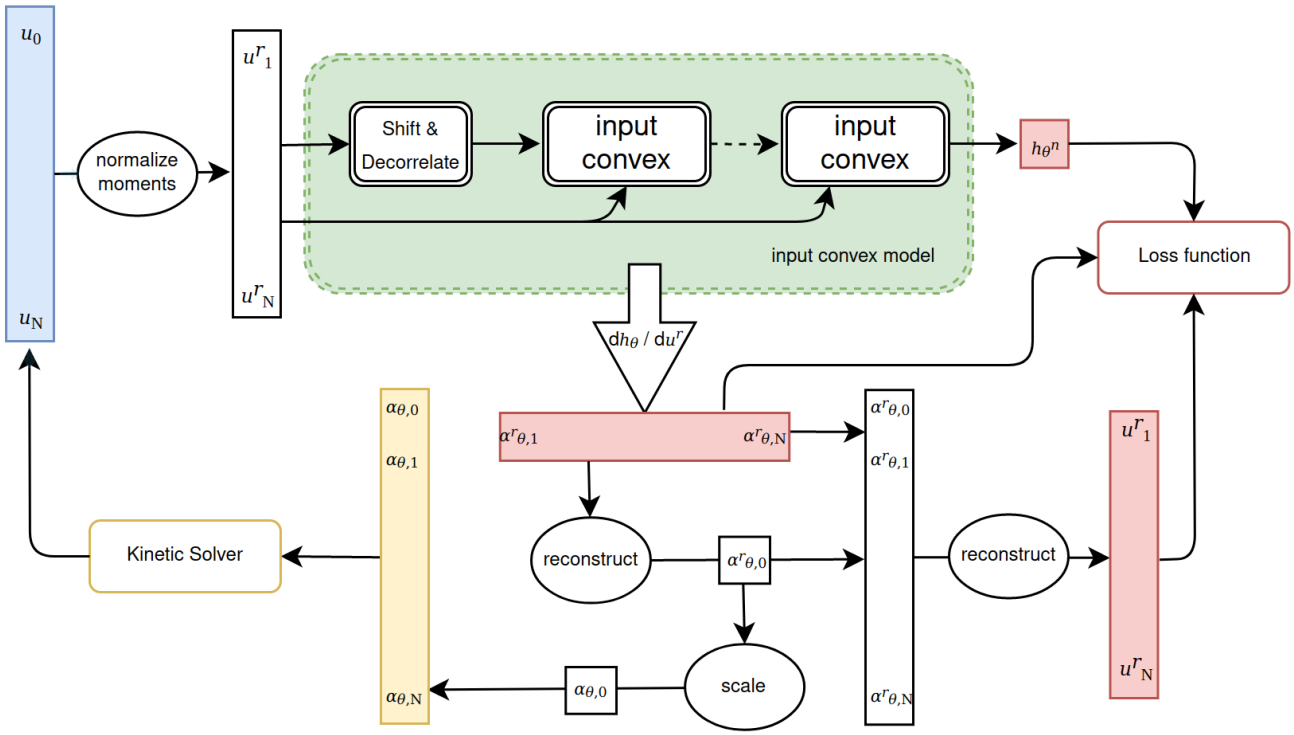


Figure 7. Input convex neural network closure. Model input vectors are depicted in blue. Red vectors are outputs, on which the model is trained on. When the trained model is employed, the yellow solution vector is used to construct the flux for the kinetic solver .

include two prepossessing layers. We first zero center the input data w.r.t the mean vector of the training data set μ_u, then we decorrelate the channels of the input vector w.r.t to the covariance matrix of the training data.

$$z_1^* = u - \mu_u, \tag{75}$$
$$z_1^{**} = \Lambda_u^T z_1 *, \tag{76}$$
$$z_1 = \sigma(W_k^x z_1^{**} + b_k^z), \tag{77}$$

where Λ_u is the eigenvector matrix of the covariance matrix of the training data set. The first two operations and the weight multiplication of the dense layer are a concatenation of linear operations and thus do not destroy convexity as well. Centering and decorrelation 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 may cause zig-zagging of the gradient vector (LeCun et al., 2012). Lastly, we rescale and center the entropy function values of the training data. Note, that in the following we switch to notation corresponding to the entropy closure problem. We scale the entropy function values h^n to the interval $[0, 1]$ via

$$h^{n,*} = \frac{h^n - \min_{l \in T} h_l^n}{\max_{l \in T} h_l^n - \min_{l \in T} h_l^n}, \tag{78}$$

which is equivalent to a shift and scale layer after the output layer of the neural network. Thus the gradient of the scaled neural network output α_θ^* needs to be re-scaled to recover the original gradient,

$$\alpha_\theta = \alpha_\theta^* \left(\max_{l \in T} h_l^n - \min_{l \in T} h_l^n \right) \tag{79}$$

Both operations are linear with a positive multiplicator, thus do not break convexity.

We briefly describe the workflow of the neural network in training and execution time, which is illustrated in Fig. 7. For training a given input convex neural network architecture, we use a training data-set $X_T = \left\{ u_i^r, \alpha_{u,i}^r, h_i^n \right\}_{i \in T}$, where we first scale h^n according to Eq. (78) and compute mean and covariance of $\{u_i^r\}_{i \in T}$ for the shift and decorrelation layer. After

a forward pass through the modified input convex neural network, we obtain $h_\theta^{n,*}$ and by automatic differentiation through the network w.r.t. u^r, we obtain $\alpha_\theta^{r,*}$, which we scale using Eq. (79) to get α_θ^r. Using Eq. (64) we reconstruct $\alpha_{\theta,0}^r$ and therefore α_θ^n with Eq. (56). The normalized moments u_θ^n and the reduced normalized moments u_θ^r are computed using Eq. (65). The training loss function is evaluated on the mean squared error of u_θ^r, $h_\theta^{n,*}$ and $\alpha_\theta^{r,*}$,

$$L(u^r, \alpha_u^{r,*}, h^{n,*}; \theta) = \frac{1}{|T|} \sum_{i \in T} \left\| h_i^{n,*} - h_{\theta,i}^{n,*} \right\|_2^2 + \lambda \left\| \alpha_{u,i}^{r,*} - \alpha_{\theta,i}^{r,*} \right\|_2^2 + \left\| u_i^r - u_{\theta,i}^r \right\|_2^2 . \tag{80}$$

The parameter λ is used to scale the loss in α_u^r to the same range as the loss in h^n and u^r. Training the neural network on the Lagrange multiplier $\alpha_u^{r,*}$ corresponds to fitting the neural network approximation to the entropy functional $h^{n,*}$ in Sobolev norm. The authors of (Czarnecki et al., 2017) found that neural network models trained on the additional knowledge of derivative information archive lower approximation errors and generalize better.
When integrating the neural network in the kinetic solver, we gather the moments of all grid cells of the spatial domain from the current iteration of the used finite volume scheme. The moments are first normalized in the sense of Eq. (54),then the predicted α_θ^r are obtained in the same manner as in the training workflow. Afterwards, we use Eq. (64) and (67) to obtain α_θ corresponding to the non-normalized moments u. Finally, Eq. (7) yields the closure of the moment system, from which the numerical flux for the finite volume scheme can be computed.

C.2. Monotone neural network approximation of the Lagrange multiplier α_u^n

No particular design choices about the neural network are made to enforce monotonicity, since the characterization of monotonic functions is not constructive. To the best of our knowledge, there exists no constructive definition of multidimensional monotonic function. Instead we construct an additional loss function to make the network monotonic during training time. This is an important difference to the first approach, where the network is convex even in the untrained stage and on unseen data points.

Definition C.1 (Monotonicity Loss)**.** Consider a neural network $\mathcal{N}_\theta : x \mapsto y$. Let X_T the training data set. The monotonicity loss is defined as

$$L_{\text{mono}}(x, \theta) = \frac{1}{|T|^2} \sum_{i \in T} \sum_{j \in T} \text{ReLU}\left(-\left(\mathcal{N}_\theta(x_i) - \mathcal{N}_\theta(x_j) \right) \cdot (x_i - x_j) \right). \tag{81}$$

The ReLU function is defined as usual,

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0. \end{cases} \tag{82}$$

The monotonicity loss checks pairwise the monotonicity property for all datapoints of thetraining data set. If the dot product is negative, the property is violated and the value of the loss is increased by the current dot product. This is a linear penalty function and can be potentiated by a concatenation with a monomial function. Note, that we only validate the monotonicity of the networkpointwise in a subset of the training data. As a consequence, the mathematical structures of the resulting moment closure is only preserved in an empirical sense, i.e. if the realizable set and more importantly, the set of Lagrange multipliers is sampled densely. The resulting neural network architecture is illustrated in Fig. 8. Normalization and the meanshift and decorrelation layers in Eq. (75) and Eq. (76) is implemented analogously to the input convex neural network. The core network architecture consists of a number of M ResNet blocks. The ResNet architecture has been successfully employed in multiple neural network designs for multiple applications and was first presented in (He et al., 2015). The ResNet blocks used in this work read as

$$z_k^1 = \text{BN}(z_{k-1}), \tag{83a}$$
$$z_k^2 = \sigma(z_k^1), \tag{83b}$$
$$z_k^3 = W_k^* z_k^2 + b_k^*, \tag{83c}$$
$$z_k^4 = \text{BN}(z_k^3), \tag{83d}$$
$$z_k^5 = \sigma(z_k^1), \tag{83e}$$
$$z_k^6 = W_k^{**} z_k^2 + b_k^{**}, \tag{83f}$$
$$z_k = z_k^6 + z_{k-1}, \tag{83g}$$

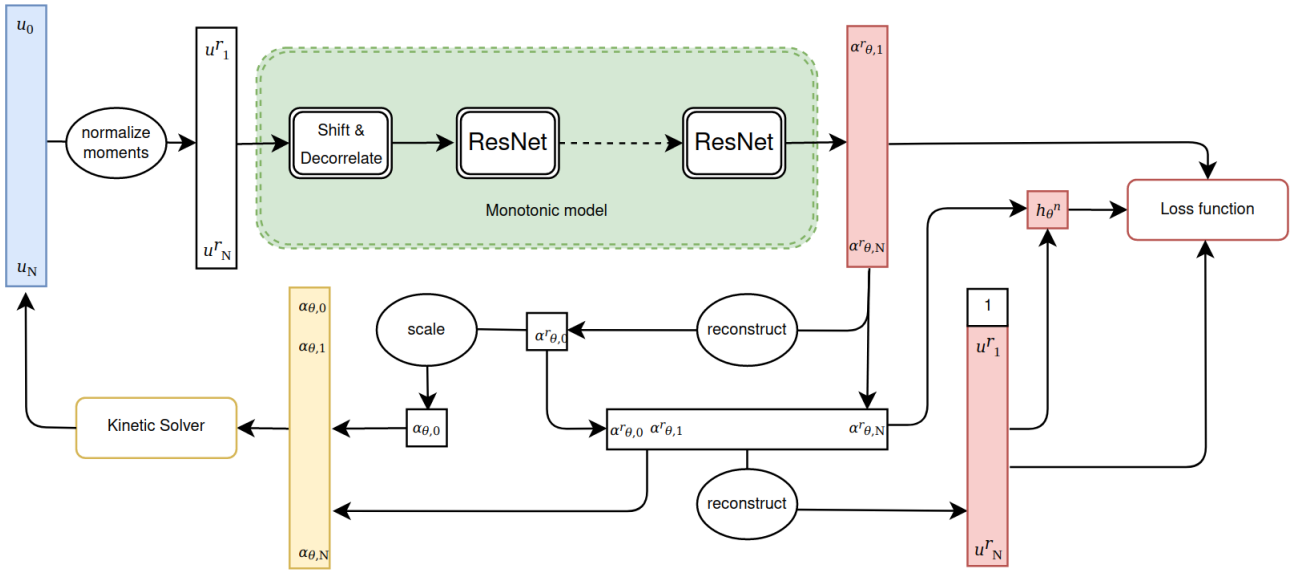


Figure 8. Input convex neural network closure. Model input vectors are depicted in blue. Red vectors are outputs, on which the model is trained on. When the trained model is employed, the yellow solution vector is used to construct the flux for the kinetic solver .

with the idea, that the skip connection in Eq. (83g) mitigates the gradient vanishing problem for deep neural networks. Furthermore, we include a batch normalization (BN) layer in front of each activation, which reduces the problem internal covariance shift (Ioffe & Szegedy, 2015), that many deep neural network structures suffer from, and which slows down the training. Batch normalization is performed by applying pointwise the following two transformation to the previous layers output z_k,

$$z_k^* = \frac{z_{k-1} - \mathbb{E}[z_{k-1}]}{\sqrt{\text{Var}[z_{k-1}] + \epsilon}}, \tag{84}$$

$$z_k = \theta_0 z_k^* + \theta_1, \tag{85}$$

where θ_0 and θ_1 are trainable weights and $\mathbb{E}[z_{k-1}]$ and $\text{Var}[z_{k-1}]$ denote the expectation value and the variance of the current batch of training data, respectively.
One transforms the network output α_θ^r to the values of interest α_θ and u_θ^r analogously to the input convex network design. The entropy functional h_θ^r directly computed from u_θ^n and α_θ^n using Eq. (9). Training data rescaling and integration in the kinetic solver follow the ideas of the input convex network design. The batchwise monotonicity loss is calculated using u^r and α_θ^r, the gradient of the convex entropy functional h^r. The loss function for the network training becomes

$$L(u^r, \alpha_u^{r,*}, h^{n,*}; \theta) = \frac{1}{|B|} \sum_{i \in B} \left(\left\| h_i^{n,*} - h_{\theta,i}^{n,*} \right\|_2^2 + \left\| \alpha_{u,i}^{r,*} - \alpha_{\theta,i}^{r,*} \right\|_2^2 + \left\| u_i^r - u_{\theta,i}^r \right\|_2^2 \right) + L_{\text{mono}}\left(u^r, \theta\right). \tag{86}$$

D. Proofs for the stated theorems

In the following we give proof to the two theorems stated in this work.

D.1. Proof of Theorem 4.1

Proof. The proof is structured in two parts. First, we show that the vertices $v_i \in \mathbb{R}^d$ are well defined, if x^* is element of the interior of $\mathcal{C}(X_d)$. Second, we show that all $v_i \in A$. Thus any convex combination of v_i is in A and therefore, A is defined by a (bounded) polyhedron with vertices v_i.
1. We show that v_i are well defined. First, if the point of interest is element of the interior of $\mathcal{C}(X_d)$, then all $x_i \in X_d$ are

linearly independent. The boundary of the set of feasible gradients with respect to the sampling point x_i and the point of x^* interest consists of the hyperplane given by

$$F_i = \left\{ v \in \mathbb{R}^d | v \cdot x_i = \nabla f^*(x_i) \cdot x_i \right\}. \tag{87}$$

Clearly, if all $x_i \neq 0$ are linearly independent, no hyperplanes are parallel or lie in each other. The proper intersection of d hyperplanes in $\mathbb{R}^d$ yields a single point,

$$v_i = \bigcap_{j \neq i} F_j. \tag{88}$$

which we define as vertex $v_i \in \mathbb{R}^d$, that touches all hyperplanes except F_i.
2. We show that all $v_i \in A$. This means, that we have to show

$$v_j \cdot x_i \leq \nabla f^*(x_i) \cdot x_i, \qquad \forall i, j = 0, \dots, d \tag{89}$$

By the definition of v_j, we have

$$v_j \in F_i, \qquad j \neq i, \tag{90}$$

so we are only concerned with

$$v_i \cdot x_i \leq \nabla f^*(x_i) \cdot x_i. \tag{91}$$

We start by stating an auxiliary statement. Let $p_i^j = v_j - v_i$ for $i \neq j$. If X_d is linearly independent and $x^* = 0$ is in the interior of $\mathcal{C}(X_d)$, then

$$\text{sign}(p_i^j \cdot x_i) = \text{sign}(p_k^l \cdot x_k), \qquad \forall i \neq j, k \neq l \tag{92}$$

Linear independence of $x_i \in X_d$ and $x^* = 0$ being in the interior of $\mathcal{C}(X_d)$ translates to

$$0 = \sum_{i=0}^{N} a_i x_i, \qquad a_i > 0. \tag{93}$$

We have

$$\begin{aligned}
p_i^j \cdot x_i &= \frac{-1}{a_i} \sum_{m \neq i} a_m \left(v_j - v_i\right) \cdot x_m && (94)\\
&= \frac{-1}{a_i} \left(\sum_{m \neq i,j} a_m \left(v_j - v_i\right) \cdot x_m + a_j \left(v_j - v_i\right) \cdot x_j \right) && (95)\\
&= \frac{-1}{a_i} \left(\sum_{m \neq i,j} a_m \left(v_j \cdot x_m - v_i \cdot x_m\right) + a_j \left(v_j - v_i\right) \cdot x_j \right) && (96)\\
&= \frac{-1}{a_i} \left(\sum_{m \neq i,j} a_m \left(\nabla f^*(x_m) \cdot x_m - \nabla f^*(x_m) \cdot x_m\right) + a_j (v_j - v_i) \cdot x_j \right) && (97)\\
&= \frac{-1}{a_i} a_j (v_j - v_i) x_j = \frac{a_j}{a_i} (v_i - v_j) x_j = \frac{a_j}{a_i} p_j^i \cdot x_j, && (98)
\end{aligned}$$

where we use the definition of the Face F_m. Since $\frac{a_j}{a_i}$ is positive $\text{sign}(p_i^j \cdot x_i) = \text{sign}(p_j^i \cdot x_j)$ follows for all $i \neq j$. Assume $p_i^j \cdot x_i > 0$ and $p_i^h \cdot x_i < 0$. Then

$$v_j \cdot x_i > v_i \cdot x_i > v_h \cdot x_i. \tag{99}$$

Thus, we have

$$0 < (v_j - v_h) \cdot x_i = \nabla(f^*(x_i) \cdot x_i - \nabla(f^*(x_i) \cdot x_i = 0, \tag{100}$$

which is an contradiction to monotonicity of the gradient. Thus,

$$\text{sign}(p_i^j \cdot x_i) = \text{sign}(p_i^k \cdot x_i) = \text{sign}(p_k^i \cdot x_k) = \text{sign}(p_k^l \cdot x_l), \qquad \forall i \neq j, k \neq l. \tag{101}$$

This means, that all face normals x_i are either facing outward of the polyhedron defined by the vertices $\{v_i\}$ or all face inward. Assume inward facing normals, then for each face of the polyhedron created by A, the feasible set is the half space outside the current face of the polyhedron. Due to convexity the polyhedron defined by $\{v_i\}$, this would imply, that $A = \emptyset$, which contradicts continuity of the gradient of f^* Thus we have outward facing normals. Finally, we have

$$0 < (v_j - v_i) \cdot x_i = \nabla f^*(x_i) \cdot x_i - v_i \cdot x_i, \tag{102}$$

and thus $v_i \cdot x_i < \nabla f(x_i) \cdot x_i$, i.e. $v_i \in A$ for all i. Thus A is indeed a polygon defined by the vertices v_i. By convexity, the polyhedron A contains all feasible gradients of the point of interest.

□

D.2. Proof of Theorem 4.2

Proof. By definition of $v_i = \bigcap_{j \neq i} F_j$ and the fact that we can divide Eq. (34) by $\|x_i\|$ we get the linear systems. Let for $\xi \in \mathbb{R}^d$.

$$C_\xi = \max_{k=0,\dots,d} \|\nabla f(x_k) - \xi\|_2 \tag{103}$$

Then we have

$$|x_j \cdot (v_i - \xi)| = |x_i \cdot (\nabla f^*(x_i) - \xi)| \leq \|x_i\|_2 C_\xi = C_\xi \quad \forall i = 0, \dots, d \tag{104}$$

since x_i has unit norm. Thus each entry of the vector $X_i v_i$ has an absolute value smaller than C_ξ. We interpret X_i as an linear operator mapping $(\mathbb{R}^d, \|\cdot\|_2) \to (\mathbb{R}^d, \|\cdot\|_\infty)$. $X_i = [x_0, \dots, x_{i-1}, x_{i+1}, \dots, x_d]^T$ is invertible, if x^* is in the interior of $\mathcal{C}(X_d)$ and defines a mapping $(\mathbb{R}^d, \|\cdot\|_\infty) \to (\mathbb{R}^d, \|\cdot\|_2)$. Consequently, we can estimate

$$\|X_i(v_i - \xi)\|_\infty \leq C_\xi, \tag{105}$$

$$\|v_i - \xi\|_2 \leq \left\|X_i^{-1}\right\| C_\xi. \tag{106}$$

Finally we get

$$\|v_i - v_j\|_2 \leq \|v_i - \xi\|_2 + \|\xi - v_j\|_2 \leq \left(\left\|X_i^{-1}\right\| + \left\|X_j^{-1}\right\|\right) C_\xi, \tag{107}$$

We can choose $\xi = \nabla f^*(x_l)$ s.t.

$$\max_{k=0,\dots,d} \|\nabla f^*(x_k) - \nabla f^*(x_l)\|_2 = \max_{k,l=0,\dots,d} \|\nabla f^*(x_k) - \nabla f^*(x_l)\|_2 =: C_{x^*} \tag{108}$$

□

E. Suplementary material for the data generation strategy

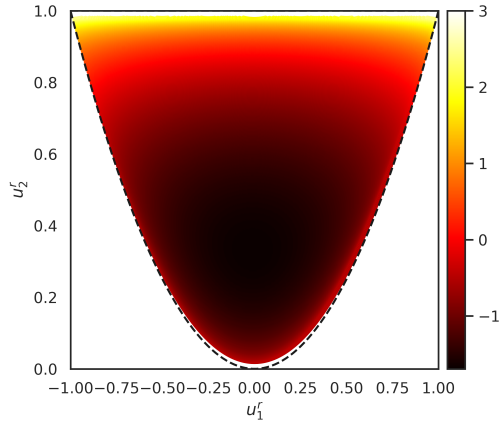
In the following we give extended context to the chosen data generation strategy, and its superiority to data generation by direct sampling of normalized moments u^n. Let us first consider proximity to the boundary in $\mathcal{R}^r$ directly. There exist extensive studies about the characterization of the boundary $\partial\mathcal{R}^r$ and we use results by Kershaw (Kershaw, 1976) and Monreal (Monreal, 2012). For the Maxwell-Boltzmann entropy and a monomial basis, $\mathcal{R}^r$ can be described in one spatial dimension, i.e. $V, X \subset \mathbb{R}^1$ up to order $N = 4$ using the inequalities

$$1 \geq u_1^r \geq -1, \tag{109a}$$

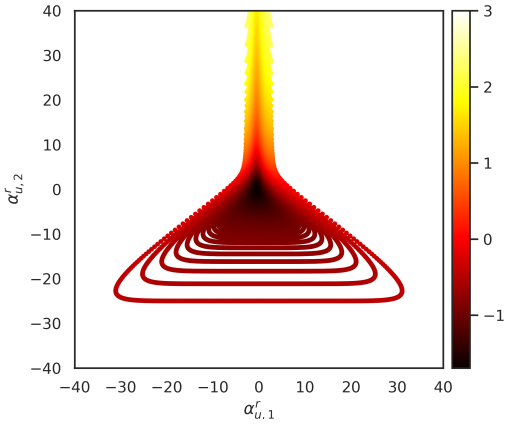
$$1 \geq u_2^r \geq (u_1^r)^2, \tag{109b}$$

$$u_2^r - \frac{(u_1^r - u_2^r)^2}{1 - u_1^r} \geq u_3^r \geq -u_2^r + \frac{(u_1^r + u_2^r)^2}{1 + u_1^r}, \tag{109c}$$

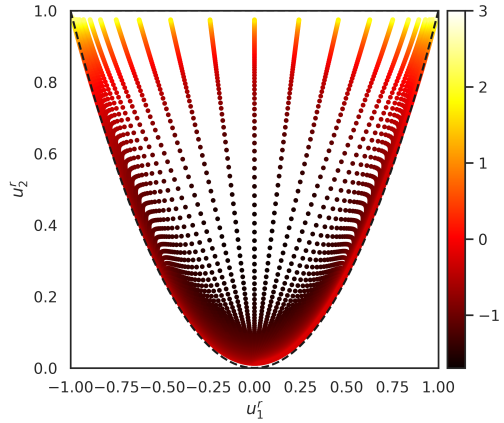
$$u_2^r - \frac{(u_1^r - u_3^r)^2}{(1 - u_2^r)} \geq u_4^r \geq \frac{(u_2^r)^3 + (u_3^r)^2 - 2u_1^r u_2^r u_3^r}{u_2^r - (u_1^r)^2}, \tag{109d}$$



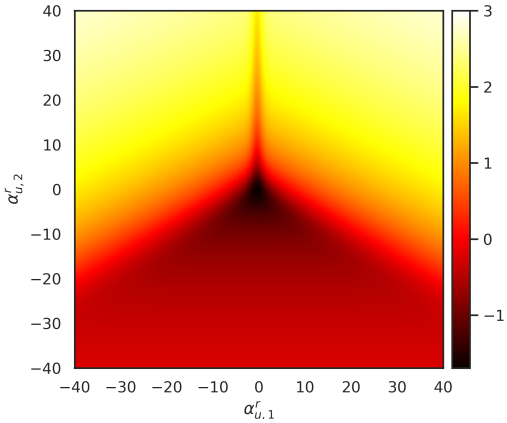
a)$\mathcal{R}^r$, using uniform grid sampling of u^r



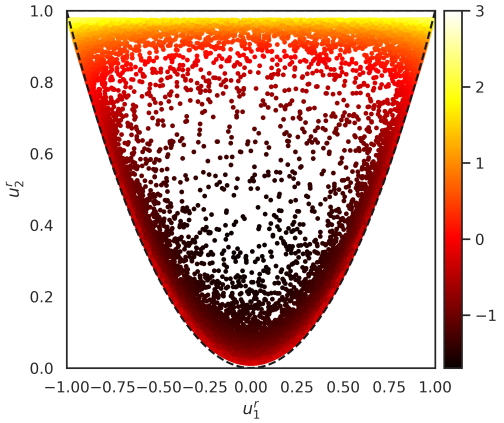
b) α_u^r, using uniform grid sampling of u^r



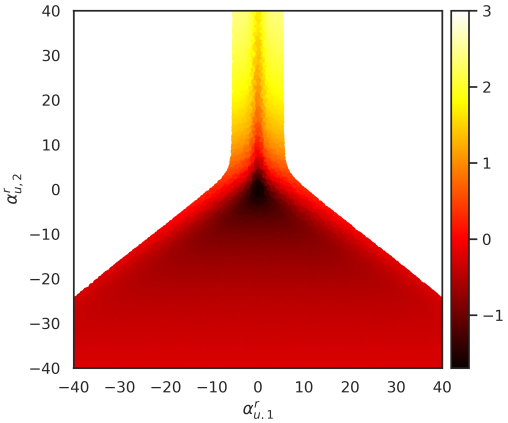
c) $\mathcal{R}^r$, using uniform grid sampling of α^r with ∞ norm bound



d) α_u^r, using uniform grid sampling of α^r with ∞ norm bound



e) $\mathcal{R}^r$, using uniform low-discrepancy sampling of α^r with eigenvalue bound



f) α_u^r, using uniform low-discrepancy sampling of α^r with eigenvalue bound

Figure 9. Scatter plots of 70000 data points for the $1D$ M_2 model with data generated from different sampling strategies. The color bar indicates the value of the minimal entropy functional h.

whereas higher moment order moments can be characterized using the more general results in (Curto & Fialkow). Equation (109) gives direct control over the boundary $\partial\mathcal{R}^r$, since equality in one or more of the equations describes a boundary of the normalized realizable set. In this case, the distance measure to $\partial\mathcal{R}^r$ is the norm distance. An example for normalized moments of the M_2 closure in $d = 1$ spatial dimensions with norm boundary distance 0.01 is shown in Fig. 9a) and the corresponding Lagrange multipliers are shown in Fig. 9b). Note, that in Fig. 9a),c) and e), $\partial\mathcal{R}^r$ is displayed by the dotted black line. More general results for arbitrarily high order moments in one spatial dimension can be found in (Kershaw, 1976). In three spatial dimensions necessary conditions have been constructed by (Monreal, 2012) for up to order $N \leq 2$, but a full characterization of $\partial\mathcal{R}$ remains an open problem (Lasserre, 2009).
From a numerical point of view, it is interesting to construct a notion of distance to $\partial\mathcal{R}^r$ directly in the space of Lagrange multipliers, since it turns out that the magnitude of the $\|\alpha_u^r\|$ has implications on the numerical stability of the neural network training process. A first idea consists of a norm bound of α_u^r, i.e. $\|\alpha_u^r\| < M < \infty$ (Alldredge et al., 2019; Porteous et al., 2021; Sadr et al., 2020), which yields a convex subset of Lagrange multipliers. Fig. 9d) shows a uniform distribution of $\alpha_{u,1}^r$ and $\alpha_{u,2}^r$, where $\alpha_{u,i}^r \in [-40, 40]$, and Fig. 9c) displays the corresponding reconstructed moments u^n. However, this approach gives no control over the boundary distance and produced very biased data distributions of u^r. A comparison of Fig. 9c) and d) shows, that two thirds of the sampled moments are concentrated in the regions near $u^r = (-1, 1)$ and $u^r = (1, 1)$, which correspond to high entropy values $h > 1.5$ and are colored yellow. In contrast, Fig. 9a) and b) show that there are no samples in the regions $\alpha_{u,1}^r > 10$ and $\left|\alpha_{u,2}^r\right| > 10$, since the corresponding moments u^r are too close to the boundary $\partial\mathcal{R}^r$. As a conclusion, the second sampling strategy does not produce data with uniform distance to $\partial\mathcal{R}^r$.
Lastly we employ the condition number as a boundary distance measure. Figure 9e)and f) show a uniform sampling, where α_u^r is sampled with eigenvalue $\lambda_{\text{min}} > 1\text{e}{-7}$. Note, that on the one hand, the near boundary region of $\mathcal{R}^r$ is more densly sampled than the interior, compare Fig. 9a) and e), whereas there is no over-representation of the regions near $u^r = (-1, 1)$ and $u^r = (1, 1)$ and the set of sampled Lagrange multipliers, see Fig. 9f), is similar in shape to the Lagrange multipliers in Fig. 9b).
The data generation method has an non negligible influence on the quality of training data, as Fig. 9c) and e) display. The former are moments u^r generated by a uniform grid sampling of α_u^r, and the latter by uniform sampling of α_u^r using a low-discrepancy sampling method. The deformed grid in u^r consists near $u_2^r = 1.0$ of very steep triangles of local training points X_d, that means that a point of interest is always close to the boundary of $\mathcal{C}(X_d)$ which implies a big diameter for the polyhedron of admissible gradients A_{x^*}.

F. Neural network training

In the following we evaluate the training performance of the neural network architectures, which are implemented in Keras using Tensorflow 2.6 and can be found in the Github repository (Schotthöfer, 2021).
The neural networks are trained on a subset of $\mathcal{R}^r$ that corresponds with Lagrange multipliers sampled from the set $B_{M,\tau}$ of Eq. (40). The data sampler can be found in the Github repostiroy (Schotthöfer et al., 2021). M and τ are chosen such that the neural network training is numerically stable, since for high absolute values of $\alpha_{u,i}^n$, the term $u^n = \langle m \exp\left(\alpha_u^n m\right)\rangle$ leads to a numerical overflow in single precision floating point accuracy, if the neural network training is not yet converged. In this sense, the high condition number of the minimal entropy closure near $\partial\mathcal{R}$ translates to the neural network approximation.
The sampled data is split into training and validation set, where the validation consists of 10% randomly drawn samples of the total data. Table 1 compares the validation losses of different neural network architectures after the training process has converged. The layout of a neural network is defined in the format width $\times$ depth. Width describes the number of neurons in one layer of the network and depth the number of building blocks of the network. A building block of the input convex neural network is one (convex) dense layer. A building block of the monotonic neural network architecture is described by Eq. (83). In addition to these layers, each model is equipped with a mean shift (75) and decorrelation (76) layer followed by a dense layer as a preprocessing head. After the core architecture of the format width $\times$ depth, one more layer of the respective core architecture with half the specified width and finally the output layer follows. The linear output layer of the input convex neural network design is one dimensional, since we approximate the entropy h_θ and the linear output layer of the monotonic network design has dimension N, where N is the order of the moment closure and the length of the reduced Lagrange multiplier vector α_u^r. The input convex network with output data scaled to the interval $[0, 1]$ uses a ReLU activation, since we do not expect negative output values.
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 validation data set. For the monotonic network, the monotonicity loss is additionally displayed. The converged mean squared error on the validation set is around $\mathcal{O}(1\text{e}{-4})$ to $\mathcal{O}(1\text{e}{-6})$. For reference, the results of previous works of approximating the

minimal entropy closure using non-convex feed-forward neural networks in (Porteous et al., 2021) have been included in the table. We can see that the ICNN architecture manages to improves the validation loss by an order of magnitude in the M_1 and M_2 1D test case. The derivative of the ICNN architecture is far more accurate in the approximation of the Lagrange multiplier α_u^r than the compared non-convex models. Notice, that for all models the mean absolute error in α_u^r is

Table 1. Validation losses for different moment closure

Closure	M_1 1D			M_2 1D			M_1 2D	
Architecture	non-conv.	convex	monotone	non-conv.	convex	monotone	convex	monotone
Layout	30×5	10×7	30×2	45×4	15×7	50×2	18×8	100×3
MSE(h^n, h_θ^n)	1.82e−4	7.87e−7	2.09e−5	1.55e−4	1.33e−5	5.04e−4	1.10e−6	4.01e−4
MSE$(\alpha_u^r, \alpha_\theta^r)$	1.20e−2	7.52e−4	5.56e−6	2.17e−1	2.81e−4	2.56e−4	3.39e−5	4.54e−5
MSE(u^r, u_θ^r)	5.88e−4	1.47e−6	3.64e−6	5.63e−4	2.81e−4	1.27e−4	3.39e−5	8.09e−5
$L_{\text{mono}}(u^r)$	n.a	n.a.	1.60e−14	n.a.	n.a	1.38e−14	n.a.	5.31e−16
MAE(h^n, h_θ^n)	n.a	7.57e−4	3.05e−3	n.a	3.11e−3	1.66e−2	1.02e−3	1.49e−2
MAE$(\alpha_u^r, \alpha_\theta^r)$	n.a	1.26e−2	9.10e−3	n.a	1.23e−2	9.74e−3	3.36e−3	4.35e−3
MAE(u^r, u_θ^r)	n.a	9.59e−4	1.51e−3	n.a	1.23e−2	7.96e−3	9.62e−4	7.02e−3

significantly higher than the error in h or u^r. The reason for this is again the high range of values, that α_u^r can attain. In case of the input convex neural network, the values are obtained by differentiating through the network with primary output h_θ, and thus one always has a scaling difference between h^n and α_u^r of about one order of magnitude. Therefor, the scaling parameter λ of Eq. (80) is set to be $\lambda = 1/10$ to balance out the training.

G. Computational Efficiency

In the following, we compare the computational efficiency of the neural network surrogate model and the Newton optimizer in an isolated, synthetic test case. We consider the M_2 closure in $1D$ and use only normalized moments. In contrast to the neural network, the performance of the Newton solver is dependent on the proximity of the moments u^n to the boundary $\partial\mathcal{R}^n$, thus we consider three test cases. First, the moments are uniformly sampled in $\mathcal{R}^n$, second we only draw moments near the center of $\mathcal{R}^n$ and lastly, we use only moments in proximity to $\partial\mathcal{R}^n$, where the minimal entropy problem is hard to solve and has a high condition number. The Newton solver is implemented in the KiT-RT (Schotthöfer et al., 2021) framework. In the kinetic scheme, there is one optimization problem for each grid cell in a given time step. Furthermore, optimization problems of different grid cells are independent of each other. Options for the parallelization of the minimal entropy closure within a kinetic solver differs, whether we choose an CPU based implementation of a Newton solver or the GPU optimized tensorflow backend for the neural network closure. A meaningful and straight-forward way to parallelize the minimal entropy closure on CPU is to employ one instance of the Newton optimizer per available CPU-core that handles a batch of cells. On the other hand, we interpret the number of grid cells as the batch size of the neural network based entropy closure. Shared memory parallelization is carried out by the tensorflow backend. For comparability, we set the accuracy tolerance of the Newton solver to single-precision floating point accuracy, since the trained neural networks have a validation error between $\mathcal{O}(1\text{e}{-4})$ and $\mathcal{O}(1\text{e}{-6})$.

We execute the neural network once on the CPU and once on the GPU using direct model execution in tensorflow. The used CPU is a 24 thread AMD Ryzen9 3900x with 32GB memory and the GPU is a RTX3090 with 20GB memory. The experiments are reiterated 100 times to reduce time measurement fluctuations. Table 2 displays the mean timing for each configuration and corresponding standard deviation.

Considering Table 2, we see that the time consumption of a neural network is indeed independent of the condition of the optimization problem, whereas the Newton solver is 6300 times slower on a on a moment u^n with $\text{dist}(u^n, \partial\mathcal{R}^n) = 0.01$ compared to a moment in the interior. The average time to compute the Lagrange multiplier of a uniformly sampled moment u^n is 27% higher than a moment of the interior. Reason for this is, that the Newton optimizer needs more iterations, the more ill-conditioned the optimization problem is. In each iteration, the inverse of the Hessian must be evaluated and the integral $\langle\cdot\rangle$ must be computed using a 30 point Gauss-Legendre quadrature. One needs a comparatively high amount of quadrature points, since the integrand $m \times m \exp(\alpha_u^n \cdot m)$ is highly nonlinear. The neural network evaluation time is independent of the input data by construction and depends only on the neural network architecture and its size. Here we evaluate the input convex neural network for the $1D$ M_2 closure, whose size is determined by Table 1. The timings for the other networks are similar, since they do not differ enough in size. However, we need to take into account that the neural

Table 2. Computational cost for one iteration of the 1D solver in seconds

	Newton	neural closure CPU	neural closure GPU
uniform, 1e3 samples	0.00648 ± 0.00117 s	0.00788 ± 0.00051 s	0.00988 ± 0.00476 s
uniform, 1e7 samples	5.01239 ± 0.01491 s	0.63321 ± 0.00891 s	0.03909 ± 0.00382 s
boundary, 1e3 samples	38.35292 ± 0.07901 s	0.00802 ± 0.00064 s	0.00974 ± 0.00475 s
boundary, 1e7 samples	27179.51012 ± 133.393 s	0.63299 ± 0.00853 s	0.03881 ± 0.00352 s
interior, 1e3 samples	0.00514 ± 0.00121 s	0.00875 ± 0.00875 s	0.00956 ± 0.00486 s
interior, 1e7 samples	4.24611 ± 0.03862 s	0.63409 ± 0.00867 s	0.03846 ± 0.00357 s

entropy closure is less accurate near $\partial\overline{\mathcal{R}}$ as shown in Fig. 10. Furthermore, we see that the acceleration gained by usage of the neural network surrogate model is higher in cases with more sampling data. This is apparent in the uniform and interior sampling test cases, where the computational time increases by a factor of ≈ 73, when the data size increases by a factor of $1\mathrm{e}4$. The time consumption of the Newton solver increases by a factor of ≈ 840 in the interior sampling case, respectively ≈ 782 in the uniform sampling case. Note, that in this experiment, all data points fit into the memory of the GPU, so it can more efficiently perform SIMD parallelization. Reason for the smaller speedup of the neural network in case of the smaller dataset is the higher communication overhead of the parallelization relative to the workload. This indicates that the best application case for the neural network is a very large scale simulation.

H. Synthetic test cases

In this section, we consider again the $1D$ M_1 entropy closure, see Fig. 2, and perform accuracy tests for the input convex and monotnonic neural network architecture. The networks are trained on a the data set generated from $\alpha^r_{u,1}$ sampled from $[-50, 50]$ using the discussed sampling strategy. Then, the networks are evaluated on twice as many samples in the displayed data range $u^n_1 \in [-0.99, 0.99]$ and $\alpha^n_i \in [-95, 80]$, thus the extrapolation areas near the boundary consist of only unseen data and the interpolation area contains at least 50% unseen data.

The relative norm errors of the predictions of both network architectures can be seen in Fig. 10. Figure 10a) compares the input convex and monotonic network on the basis of their relative norm error in the Lagrange multiplier α^r_u. Within the intervall $[-0.75, 0.75]$ the relative error of the input convex neural network is in $\mathcal{O}(1\mathrm{e}{-2.5})$ and increases by half an order of magnitude in the extrapolation area. The relative error of the monotonic architecture displays more fluctuation with a mean of $\mathcal{O}(1\mathrm{e}{-2})$. In the extrapolation area, the error of the monotonic network increases by over an order of magnitude and is outperformed by the convex neural network. Remark, that the approximation quality declines as we approach $\partial\mathcal{R}^r$, which is expected, since the neural networks can not be trained close to the boundary and the output data α^r_u and h grow rapidly in this region.

Figure 10b) displays the relative error in the entropy prediction h_θ for of the respective neural networks. The monotonic architecture exhibits a larger relative error in h, compared to the input convex architecture. This can by explained by the fact, that the input convex neural network directly approximates h, whereas the monotonic neural network reconstructs h using α^r_θ and u_θ and thus the approximation error of both variables influence the error in h. In the extrapolation regime, one can see a similar error increase as in Fig. 10a).

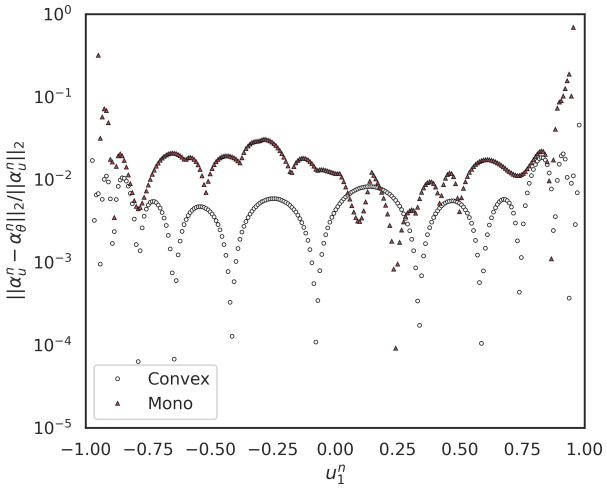
Overall, both networks do not perform well near $\partial\mathcal{R}^r$, however, when we consult Fig. 10c), we see that the error in the reconstructed moment u^n is below $\mathcal{O}(1\mathrm{e}{-2})$ for the input convex and the monotonic network, although the error in α^n_u is almost in the order of $\mathcal{O}(1\mathrm{e}0)$ in this region. This shows, that the nature of the reconstruction map $u = \langle m \exp(\alpha_u m)\rangle$ mitigates the inaccuracy in α^n_u to some degree. The reconstructed moments u^n_θ experience less relative error in the interior of $\mathcal{R}^r$ than near the boundary. For the stability of the solver however, the error in the reconstructed flux $\langle v m f_u\rangle$, which is Lipschitz continuous in u, is the most important quantity.

All in all, both network architecture are able to approximate the entropy closure within a reasonable error margin.

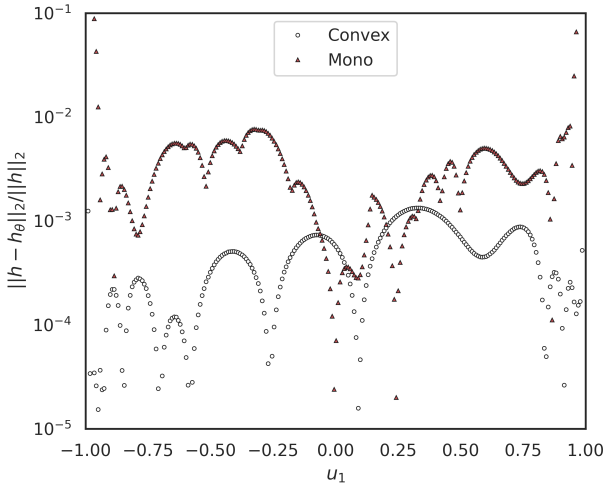
I. Simulation test cases

I.1. Supplementary material for the test case: An-isotropic inflow into an isotropically scattering homogeneous medium in $1D$

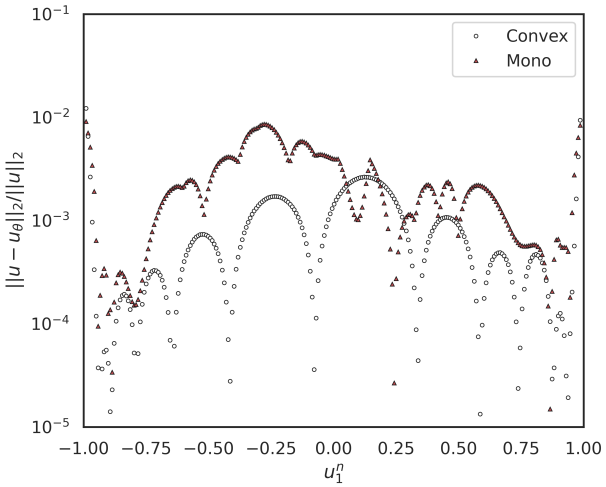
To verify the significance of the following error comparison of the neural network based closures with the Newton benchmark, we conduct a convergence analysis of the $M1$ and $M2$ cases with the used finite volume solver. Figure. 11a) compares the



a) Relative norm error of the prediction of α_u^n



b) Relative norm error of the prediction of h

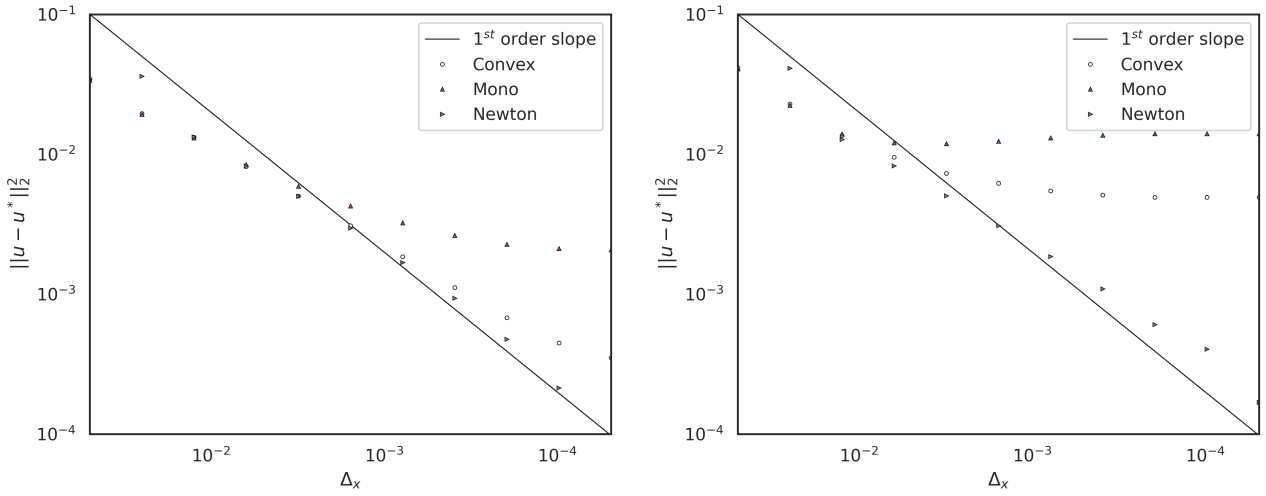


c) Relative norm error of the reconstruction of u^n

Figure 10. Relative test errors of both neural network architectures for the $1D$ M_1 closure. Distance to $\partial\mathcal{R}$ is 0.01

Table 3. Computational setup of the test cases

	1D M1 Linesource	1D M2 Linesource	2D M1 Periodic
T	$(0, 0.7]$	$(0, 0.7]$	$(0, 10]$
Time steps	8750	8750	33333
$\mathbf{X}$	$[0, 1]$	$[0, 1]$	$[-1.5, 1.5] \times [-1.5, 1.5]$
Grid cells n_x	5000	5000	2000^2
Quadrature	Gauss-Legendre	Gauss-Legendre	Tensorized Gauss-Legendre
v	$[-1, 1]$	$[-1, 1]$	$[-1, 1] \times [0, 2\pi)$
Quadrature points	28	28	400
Basis	Monomial	Monomial	Monomial
CFL number	0.4	0.4	0.4
σ	1.0	1.0	0.0
τ	0.5	0.5	0.0



a) $1D$ M_1 closure

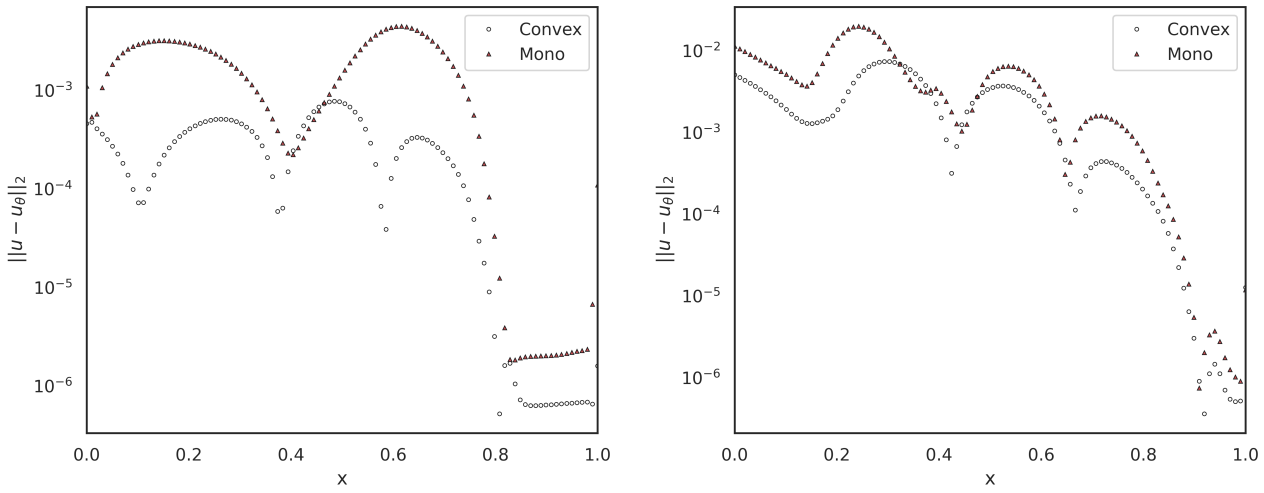
b) $1D$ M_2 closure

Figure 11. Comparison of the convergence rate of a first order finite volume solver with different closures. The ground truth u^* is given by the finest Newton based solution. The spatial cell size is denoted by Δ_x.

convergence of the solution of both neural network entropy closure and Newton closed solver of the $1D$ $M1$ test case and Fig. 11b) the corresponding solutions of the $2D$ $M1$ test case. We assume the solution of the Newton solver at final time t_f with the finest grid as the ground truth. Due to a fixed CFL number, the amount of time steps needed for each simulation is proportional to the number of used grid cells. The plots display first order convergence for the Newton based solver as expected. We can see in Fig. 11a), that the monotonic neural network in the $1D$ $M1$ inflow test case converges with first order accuracy up to an error level of $\mathcal{O}(1\text{e}{-2.5})$. For finer grid resolutions, the error in the neural network based closure dominates the spatial discretization error. The input convex neural network exhibits similar behavior, but the error plateau is reached at $\mathcal{O}(1\text{e}{-3.5})$.
In Fig. 12, we see the corresponding norm errors of the M_1 and M_2 solution for each grid cell at final time t_f. The point wise norm error is again in the range of $\mathcal{O}(1\text{e}{-3.5})$ in case of the input convex architecture and in the range of $\mathcal{O}(1\text{e}{-2.5}$ in case of the monotonic network architecture in the M_1 test case. In the M_2 test case, the errors do not exceed $\mathcal{O}(1\text{e}{-2})$. An inspection of the relative errors of these test cases is given in Fig. 13. One can spot the maximal relative error in both test cases at $x \in (0.7, 0.8)$ at final time t_f. The wave front is located in this area in the an-isotropic inflow simulation and the moments u are closest to the boundary of the realizable set $\partial\mathcal{R}$.

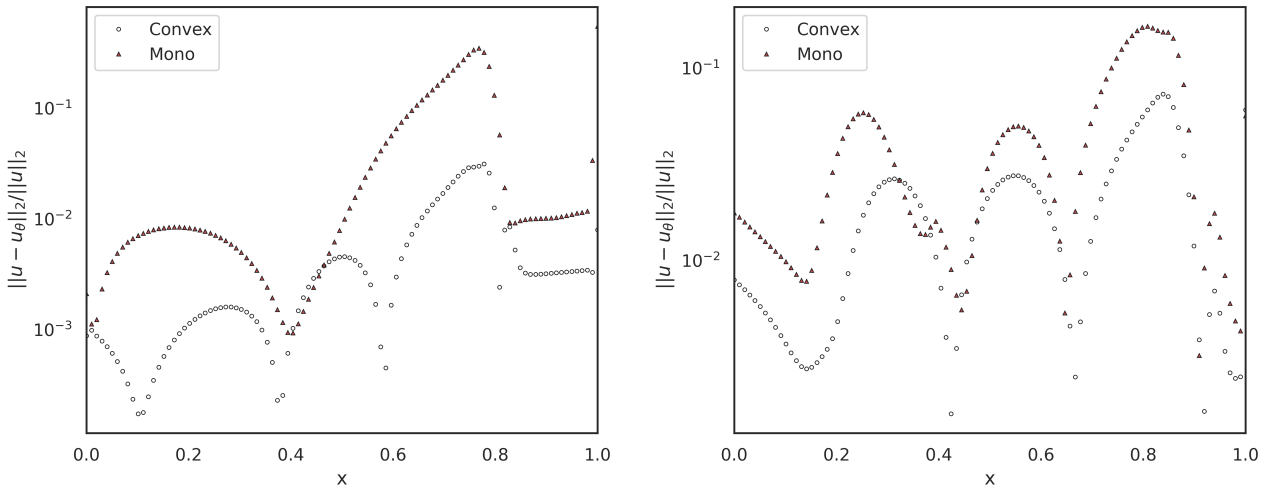
I.2. Supplementary material for the test case: Particles in a 2D non scattering medium with periodic initial conditions

Note that due to the absent of gain and loss terms and the choice of boundary conditions, the system is closed and cannot lose or gain particles. The M_1 system is solved using again a kinetic scheme with a $2D$ finite volume method in space, an explicit Euler scheme in time and a tensorized $2D$ Gauss-Legendre quadrature to compute the velocity integrals. The detailed solver configuration can be found in Table 3. Analogously to the $1D$ test cases, we compare the Newton based benchmark solution to the neural network based closures with the input convex and monotonic architectures. We run the simulation until a final time $t_f = 10.0$, which translates to 33333 time-steps.
A convergence analysis for the $2D$ $M1$ closures of both network architectures in Fig. 15 is conducted. The convergence of the input convex neural network levels of at $\mathcal{O}(1\text{e}{-3})$ and the convergence of the monotonic network at $\mathcal{O}(1\text{e}{-2.5})$, which is in line with the findings of the $1D$ closures. The size of the spatial grid is chosen correspondingly.
Figure 5 shows a snapshot of the flow field computed with the benchmark solver and Fig. 14 displays snapshots of the relative error at each grid cell of the flow field at the same iteration as the benchmark solver in Fig. 5. The relative errors of both neural networks exhibit periodic behavior and are in the range of $\mathcal{O}(10^{-2})$ or lower. Similarly to the $1D$ test cases, the input convex architecture is again slightly more accurate than the monotonic counterpart.
Figure 16a) and b) display the relative norm error of both α_u and the moment u of both neural network architectures at each time step of the simulation averaged over the whole computational domain. First, one can observe that in both figures again the relative error of the monotonic neural network is slightly bigger than the error of the input convex neural network.



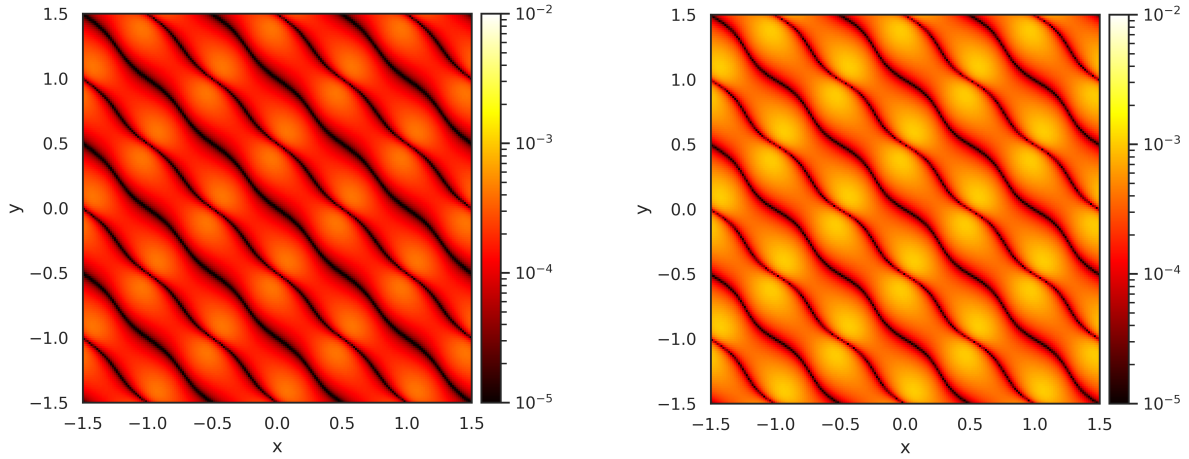
a) $1D$ M_1 closure b) $1D$ M_2 closure

Figure 12. Norm error at individual grid points of the neural network based closure with respect to the benchmark solution at time $t_f = 0.7$. For readability, not every grid cell is displayed.



a) $1D$ M_1 closure b) $1D$ M_2 closure

Figure 13. Relative norm error at individual grid points of the neural network based closure with respect to the benchmark solution at time $t_f = 0.7$. For readability, not every grid cell is displayed



a) Input convex neural network b) Monotonic neural network

Figure 14. Snapshot of the relative norm error of u_θ with respect to the benchmark solution in the $2D$ M_1 test case. The colorbar indicates the value of $\|u - u_\theta\|_2 \,/\, \|u\|_2$ at agrid cell.

Second, we can see that in the first time steps of the simulation, the error increases from $\mathcal{O}(1\mathrm{e}{-4})$ to $\mathcal{O}(1\mathrm{e}{-2})$ in case of the moments, respectively $\mathcal{O}(1\mathrm{e}{-3})$ to $\mathcal{O}(1\mathrm{e}{-1.5})$ in case of the Lagrange multipliers. After this initial increase, the error stays stable for the reminder of the simulation. The oscillations in the error curves stem from the periodic nature of the system's solution, in which the distance to $\partial\mathcal{R}$ of the appearing moments changes periodically as well.

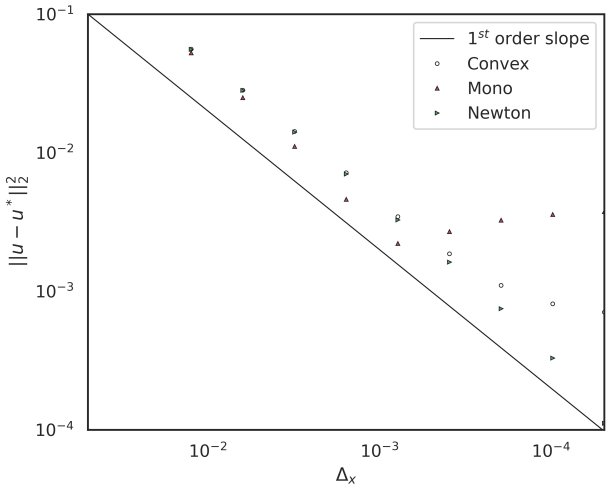
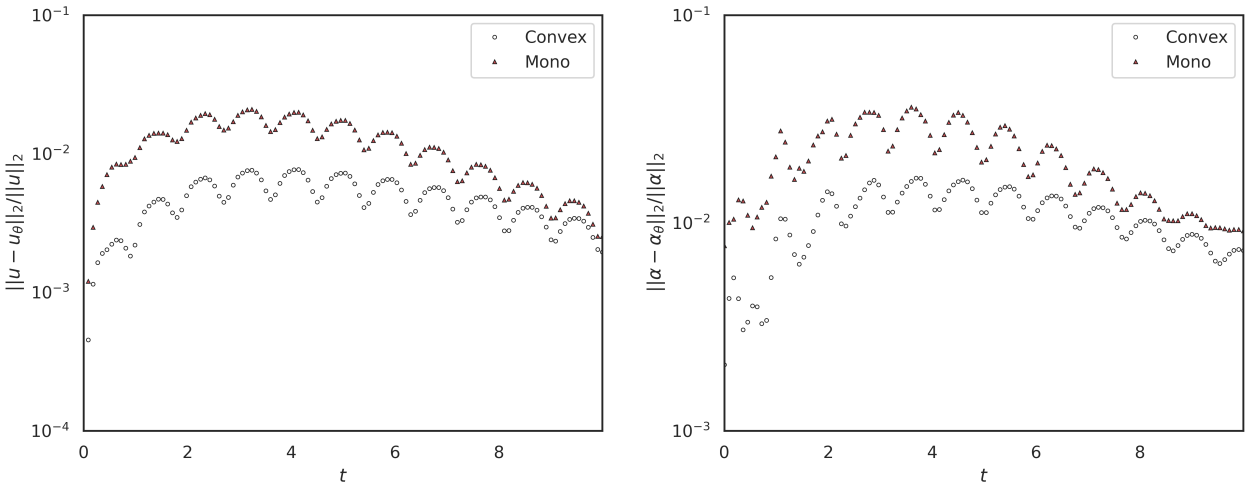


Figure 15. Comparison of the convergence rate of a first order finite volume solver with different closures. The ground truth u^* is given by the finest Newton based solution. The spatial cell size is denoted by Δ_x.



a) Relative error of u_θ at each time step

b) Relative error of α_θ at each time step

Figure 16. Mean over the spatial grid of the relative error of u_θ with respect to the benchmark solution over time. For better readability only a fraction of the time steps are displayed.