Received: 8 October 2022 | Accepted: 21 December 2022

DOI: 10.1002/pamm.202200207

A Reinforcement Learning Based Slope Limiter for Second-Order Finite Volume Schemes

Anna Schwarz[1,*], **Jens Keim**[1,**], **Simone Chiocchetti**[2], and **Andrea Beck**[1]

[1] Institute of Aerodynamics and Gas Dynamics, University of Stuttgart, 70569 Stuttgart, Germany
[2] Laboratory of Applied Mathematics, University of Trento, Via Mesiano 77, 38123 Trento, Italy

Hyperbolic equations admit discontinuities in the solution and thus adequate and physically sound numerical schemes are necessary for their discretization. Second-order finite volume schemes are a popular choice for the discretization of hyperbolic problems due to their simplicity. Despite the numerous advantages of higher-order schemes in smooth regions, they fail at strong discontinuities. Crucial for the accurate and stable simulation of flow problems with discontinuities is the adequate and reliable limiting of the reconstructed slopes. Numerous limiters have been developed to handle this task. However, they are too dissipative in smooth regions or require empirical parameters which are globally defined and test case specific. Therefore, this paper aims to develop a new slope limiter based on deep learning and reinforcement learning techniques. For this, the proposed limiter is based on several admissibility constraints: positivity of the solution and a relaxed discrete maximum principle. This approach enables a slope limiter which is independent of a manually specified global parameter while providing an optimal slope with respect to the defined admissibility constraints. The new limiter is applied to several well-known shock tube problems, which illustrates its broad applicability and the potential of reinforcement learning in numerics.

1 Introduction

Non-linear hyperbolic conservation laws are widely used in science and engineering for the modeling of physical processes [1]. An intrinsic feature of such partial differential equations is the breakdown of classical solutions and the occurrence of discontinuities, even if the initial data were smooth [1]. In the case of the Euler equations, these discontinuities may be identified as contact discontinuities or shocks, which are of linear or non-linear nature, respectively.

If one is interested in the solution of such systems, first-order finite volume (FV) schemes combined with a monotone numerical flux function turned out to be well suited for this task [2]. These schemes are per construction consistent with the conservation property of the underlying governing equations and provide a stable as well as entropy compliant approximation. However, in practical applications, the major part of the solution may have a smooth multiscale character, and only at some points discontinuities are part of the solution. Thus, first-order schemes require a massive grid resolution to be accurate. Especially for two- and three-dimensional problems, high-order methods are favorable in general.

Much effort has been spent in the development of higher-order schemes over the last decades [3–8]. However, high-order methods, e.g. finite-difference, finite-element, and discontinuous Galerkin schemes, are prone to oscillations at strong discontinuities. These oscillations are often called Gibbs phenomenon. Therefore, a widely used ansatz for such schemes is the (local) addition of artificial viscosity. This may be achieved via local filtering of the solution [9], a (local) diffusion operator [10], flux correction [11] or a locally h-refined low-order solution [12], only to mention some possibilities.

In the context of FV schemes, Godunov [13] proved that a linear monotone scheme can be at most of first-order. Hence, a non-linear reconstruction or limiting procedure is required if schemes of higher-order shall be constructed. Therefore, non-linear methods based on TVD limiting strategies [14, 15] or essentially non-oscillatory reconstruction schemes of (W)ENO-type [6, 16] have been proposed. The aforementioned, second-order TVD schemes of MUSCL-type [5, 17] are popular in academia and industry due to their algorithmic simplicity, robustness and superiority over first-order FV methods in terms of their convergence properties. However, a major drawback of these schemes is the optimal choice of the limiter function with respect to stability and accuracy. While the *minmod* limiter provides a robust scheme even for strong shocks, it is too dissipative in smooth regions of the solution. Other simple limiters, such as the *superbee* limiter, may perform well in smooth regions of the solution but fail at strong discontinuities and even induce unphysical numerical artifacts at local extrema. To resolve this issue, more sophisticated limiters have been proposed that allow the limiter to be tuned to the particular test case and, in general, to strike a balance between robustness and accuracy of the results [14, 15]. However, these tuning parameters are strongly test case dependent and a priori unknown. Furthermore, the developed limiters are generally based on schemes for linear hyperbolic equations or on even simpler constraints which do not account for the physical behavior of the considered partial differential equation system. Hence, they cannot guarantee that the TVD property is actually fulfilled when applied to non-linear hyperbolic equations. An alternative approach to the discussed a priori limiters is the MOOD limiter [8, 18], where the validity of the solution is examined in an a posteriori fashion based on some predefined admissibility criteria. If any

* Corresponding author: e-mail schwarz@iag.uni-stuttgart.de
** e-mail keim@iag.uni-stuttgart.de

of the defined criteria is violated, the solution of the corresponding elements has to be recomputed with a lower order, more dissipative, or in general more robust scheme. This procedure guarantees high-order solutions where possible and a nearly oscillation-free treatment of discontinuities. However, depending on the actual implementation, the prediction of a high-order solution at the next time level and the subsequent a posteriori corrector step may require several iterations until an admissible solution at the next time level is achieved. Thus, this procedure can get computationally expensive. Therefore, the present work aims to design an a priori limiter for second-order MUSCL-Hancock schemes [17] which has the MOOD properties of an a posteriori limiter. For this task, data-driven methods from machine learning may be employed.

Several researchers have proposed data-driven shock detection [19–21] or capturing methods [22, 23]. However, all of them have in common that they were trained in a supervised learning (SL) manner, i.e. during the training, none of the models actually took the response of the numerical scheme on their prediction into account. Thus, in the approaches available, either known shock capturing schemes were replaced by a data-driven model [22, 23], or models were trained to classify between a numerically sound and a troubled solution [19–21]. The latter ansatz is quite intricate as the user has to classify the training samples a priori. However, without considering the behavior of the numerical scheme at hand, it is impossible to infer from the state at a time level t^n whether or not the resulting solution will be admissible at the next time level t^{n+1}, or will guarantee an admissible transition to the time level t^{n+2}, even if at t^{n+1} a monotone first-order scheme is used. Hence, the task is to provide a training set which guarantees that for any forthcoming solution an admissible solution at the next time level can be reached, while keeping the high-order properties of the numerical scheme in the smooth parts of the solution. For this purpose, in the data-driven shock detection methods presented so far in the literature, an extensive amount of fitted analytical functions was used to mimic possibly occurring numerical solutions [21]. This approach, however, is quite complicated as a lot of testing is required to provide the necessary analytical functions and their optimal balance in the training data and is hard to transfer if the numerical scheme or the governing equations are changed. Thus, in the present work, methods from reinforcement learning (RL) are applied to train a data-driven a priori shock-capturing scheme.

Early successes of reinforcement learning have been achieved in the construction of artificial intelligences capable of playing complex games [24] such as chess or Go [25], and in contrast to SL, the training process is carried out by allowing the trained model itself, called the agent, to interact with an environment. Besides games, the approach is well suited for many types of control processes: for example, in this paper, the environment is a second-order MUSCL-Hancock scheme and the agent is a feedforward neural network. In computational fluid dynamics, RL has been applied successfully to control tasks [26] and in turbulence modeling [27]. Based on these encouraging results, it is the aim of the present work to develop a deep RL framework which can be used to train any kind of a priori shock capturing method, e.g. slope limiters, flux limiters or artificial viscosity, and has the properties of an a posteriori MOOD limiter. The inputs of the used neural network are the integral mean values of the primitive variables, density and pressure, of the three point stencil generally employed for second-order reconstruction schemes. Moreover, the training should be easily applicable if the discretization method or the underlying governing equations are changed. As a particular example, the framework is applied in combination with a MUSCL-Hancock scheme to train a slope limiter for the Euler equations with an ideal gas equation of state.

The remainder of the paper is structured as follows. First, in section 2, we give a brief introduction to the applied RL algorithm. The major contribution of this work, the framework developed for the training of a data-driven a priori shock capturing method with the properties of an a posteriori MOOD limiter is presented in section 3. Turning to numerical experiments, the applicability and predictive performance of the developed limiter on different well-known standard test cases is discussed in section 4. We conclude the paper with a short summary and an outlook in section 5.

2 Reinforcement Learning

Since the slope limiter presented in this work is trained via reinforcement learning, we give here a brief introduction of the applied RL algorithm. In contrast to SL, the general idea of RL is to learn a time-dependent problem through an agent which interacts with a given environment and receives feedback for its actions in this environment. For this, the problem is modeled as a Markov decision process (MDP), i.e. the agent is equipped with all necessary information to transit to the next state. In detail, at each discrete time level $t^n \in \mathbb{R}_+$ with $n \in \mathbb{N}$, the agent draws an action $a \in \mathcal{A}$ based on the policy π and its state $s^n \in \mathcal{S}$, obtains a reward $r : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ for its decision and transits to the next state s^{n+1}. The transition from s^n to s^{n+1} is stored in the so-called transition tuple $\mathcal{T} = (s^n, a, r, s^{n+1})$. Here, $\mathcal{A}$ and $\mathcal{S}$ denote the set of all possible actions (action space) and states (state space) in the environment, respectively. In this work, a deterministic policy is chosen, i.e. $\pi : \mathcal{S} \to \mathcal{A}$. With this in mind, it is the task of RL algorithms to find the optimal policy $\pi_{\boldsymbol{\phi}}(s^n) = \pi(s^n; \boldsymbol{\phi})$, with parameters $\boldsymbol{\phi} \in \mathbb{R}^k,\ k \in \mathbb{N}_{>0}$, which maximizes the accumulated reward over time, denoted as the return $R = \sum_{i=t^n}^{\infty} \gamma^{i-t^n} r(s_i, a_i)$. The discount factor $0 < \gamma < 1$ ensures that future returns are weighted less than the more immediate ones [24]. To maximize the return, in the present work, we employ policy gradient methods which update the parameters of the policy with respect to the gradient of the performance objective $J(\theta) = \mathbb{E}[R|\pi]$, e.g. via gradient ascent, where $\mathbb{E}[\cdot]$ denotes the expectation operator. This objective can be computed by the deterministic policy gradient theorem [28], given as

$$\nabla_{\boldsymbol{\phi}} J(\boldsymbol{\phi}) = \mathbb{E}_{s^n \sim \rho^{\pi}}[\nabla_a Q^{\pi}(s^n, a)|_{a=\pi_{\boldsymbol{\phi}}(s)} \nabla_{\boldsymbol{\phi}} \pi_{\boldsymbol{\phi}}(s^n)], \tag{1}$$

where $\mathbb{E}_{s^n\sim\rho^\pi}[\cdot]$ is the expected value as defined by [28] and $Q^\pi(s^n,a)$ the action-value function, i.e. the expected return obtained in the state s^n for performing an action $a=\pi_{\boldsymbol{\phi}}(s^n)$. In order to evaluate eq. (1), $Q^\pi(s^n,a)$ has to be known. One approach is to use a policy evaluation algorithm, e.g. temporal difference (TD) learning [28], where the true $Q^\pi(s^n,a)$ is estimated by $Q_{\boldsymbol{\theta}}(s^n,a)=Q(s^n,a;\boldsymbol{\theta})\approx Q^\pi(s^n,a)$ with the parameters $\boldsymbol{\theta}\in\mathbb{R}^k,\ k\in\mathbb{N}_{>0}$. TD learning is based on the Bellman equation, see e.g. [29], which is given as $Q_{\boldsymbol{\theta}}(s^n,a)=r+\gamma\, Q_{\boldsymbol{\theta}}(s^{n+1},\pi_{\boldsymbol{\phi}}(s^{n+1}))$.

A widely adopted family of policy gradient methods based on eq. (1) and TD learning is that of actor-critic algorithms [24, 28]. An actor-critic method is composed of two elements: An actor which updates the parameters of a deterministic policy and a critic to estimate the action-value function $Q_{\boldsymbol{\theta}}(s^n,a)$, e.g. via TD learning. Actor-critic algorithms can be further classified in on-policy and off-policy algorithms, depending on the way in which the target policy differs from the behavior policy. The (deterministic) target policy, $\pi_{\boldsymbol{\phi}}=\pi(s^n;\boldsymbol{\phi})$ is the policy used to compute the target of both the actor and the critic, while the action during training is drawn from a (stochastic) behavior policy $\tilde{\pi}_{\boldsymbol{\phi}}=\pi(s^n,a;\boldsymbol{\phi})$. The target policy and the behavior policy are different from each other for off-policy algorithms, while they are identical for on-policy algorithms. The advantage of off-policy compared to on-policy algorithms is that the update of both actor and critic can be decoupled from the sampling of transitions, which can be stored in an experience replay buffer, see e.g. [29]. In order to deal with continuous state and action spaces, neural networks are employed for the function approximation of both the actor and the critic.

In this work, a state of the art off-policy RL algorithm, twin delayed deep deterministic policy gradients (TD3) [29], is utilized. The idea of TD3 is to use a clipped variant of the original double Q-learning [30] to reduce the overestimation of the bias by the use of target networks with parameters $\boldsymbol{\theta}'$ and $\boldsymbol{\phi}'$ for both the actor and the critic. In addition, two critic networks are employed and the minimum of both is used to yield the target of the critic

$$y=r+\gamma\,\min_{i=1,2}\,Q_{\boldsymbol{\theta}_i'}(s^{n+1},\pi_{\boldsymbol{\phi}'}(s^{n+1})+\epsilon),\ \epsilon\sim\mathrm{clip}(\mathcal{N}(0,\tilde{\sigma}),-c,c), \tag{2}$$

where $\mathcal{N}(\mu,\sigma)$ denotes the normal Gaussian distribution with variance σ and mean μ, $\gamma=0.99$, $c=0.5$ and $\tilde{\sigma}=0.2$, as suggested in [29]. To reduce the error per update, the update of the actor is delayed to the critic, e.g. every $\tilde{d}$ iterations. For more details on the algorithm and an evaluation of its performance, see [29].

3 The *RL*Indi-Framework

In the following, we briefly present the methodology developed to train a reinforcement learning based slope limiter. The structure of the algorithm is depicted in algorithm 1.

The algorithm begins with the initialization of the networks and a replay buffer $\mathcal{B}\in\mathbb{R}^{n_{RB}}$ of size $n_{RB}=5\cdot10^4$. At this stage, the number of training epochs $n_{\text{epoch}}=1000$ is also set. The actor and both critics, as well as their targets, are multilayer perceptrons with two hidden layers, each composed of 300 nodes, and the SiLU function [31] is applied between both layers. After the output of the actor, a tanh function is applied. In each training epoch, arbitrary initialized shock tube problems are computed until a predefined end time $t_{\text{end}}=0.1$ is reached. The initial conditions are given as $\mathbf{v}_l=(\rho_l,u_l,p_l)^T$ for $x<0.5$ and $\mathbf{v}_r=(\rho_r,u_r,p_r)^T$ for $x\geq0.5$, and vice versa, with density $\rho_l\sim\mathcal{U}[0.01,1]$, $\rho_r\sim\mathcal{U}[0.9,4]$, velocity $u_r,u_l\sim\mathcal{U}[-2,2]$, pressure $p_l\sim\mathcal{U}[0.01,1]$, $p_r\sim\mathcal{U}[1,10]$ and $x\in[0,1]$. The operator $\mathcal{U}[a,b]$ denotes a uniform distribution in the interval $I=[a,b]$. The training is performed with a representative discretization of $N=200$, CFL $=0.99$, and supersonic outflow conditions are imposed at the boundaries.

In each epoch, the procedure starts with the initialization of the test case and the calculation of the time step dt. After that, the slope $\delta\mathbf{q}$ is calculated as, see e.g. [15],

$$\delta\mathbf{q}=0.5\left[(1-a_i)(\mathbf{q}_{i+1}-\mathbf{q}_i)+(1+a_i)(\mathbf{q}_i-\mathbf{q}_{i-1})\right],\ a_i\in[-1,1] \tag{3}$$

by the use of the action predicted by the actor network $a_i=\mathrm{clip}(\pi_{\boldsymbol{\phi}}(\tilde{\mathbf{s}}_i^n)+\epsilon,\ a_{\text{low}}=-1,\ a_{\text{up}}=1)$ to which an exploration noise ϵ drawn from $\mathcal{N}(0,\sigma),\sigma\in\mathbb{R}_{>0}$ is added, here $\sigma=0.1$, as suggested in [29]. The input $\tilde{\mathbf{s}}_i^n$ to the actor is a three-point stencil composed of the normalized solution at the current time level t^n in cell i and its direct neighbors, i.e. the left $(i-1)$ and right $(i+1)$ cells. For this, the physical state $\mathbf{s}_i^n=(V_{i-1}^n,V_i^n,V_{i+1}^n)^T,\ i=1,\dots,N$ with $V_i=\mathrm{sign}(\min(\rho_i,p_i))\rho_ip_i$ is normalized by a min-max normalization $\textsc{normalize}(\mathbf{s}_i):\mathbf{s}_i\in\mathbb{R}^3\mapsto\tilde{\mathbf{s}}_i\in[0,1]$, generally defined as

$$\textsc{normalize}(\mathbf{s}_i)=\begin{cases}\frac{\mathbf{s}_i-\min(\mathbf{s}_i)}{\max(\mathbf{s}_i)-\min(\mathbf{s}_i)} & :\max(\mathbf{s}_i)-\min(\mathbf{s}_i)>\epsilon_1\max(|\mathbf{s}_i|),\\ 1 & :\text{otherwise},\ i=1,\dots,N,\end{cases}$$

where the time level is omitted as the normalization has to be applied to the current state $\mathbf{s}^n$ and the next state $\mathbf{s}^{n+1}$. For states which are nearly constant $\mathbf{s}_{c,i}=(\max(\mathbf{s}_i)-\min(\mathbf{s}_i)\leq\epsilon_1\max(\mathbf{s}_i))$ the action is set to $a_i=1$ for cell i. This can be interpreted as a restrictive oscillation indicator [18], here $\epsilon_1=10^{-5}$. With the calculated slope, the solution $\mathbf{q}^n$ can be updated in time using the MUSCL-Hancock scheme, yielding $\mathbf{q}^{n+1}$ at $t^{n+1}=t^n+dt$; as a Riemann solver, the HLL method [32] is adopted. Since RL algorithms are based on MDP, for each discrete timestep dt, transition tuples $\mathcal{T}=(\tilde{\mathbf{s}}^n,\mathbf{a},\mathbf{r},\tilde{\mathbf{s}}^{n+1},\mathbb{S})$ are stored in the replay buffer $\mathcal{B}$. For $\mathbf{s}_c^n$, the slope is known and the corresponding transition tuple is not written to $\mathcal{B}$. A transition tuple is

composed of the normalized state at t^n, $\tilde{\mathbf{s}}^n$, the normalized state at the next, now current, time level t^{n+1}, $\tilde{\mathbf{s}}^{n+1}$, the action $\mathbf{a}$, the reward $\mathbf{r} = \text{REWARD}(\mathbf{s}^{n+1}, \mathbf{s}^n, \mathbf{a})$ and the positivity status of the new state $\mathbb{S} = \text{SANITY}(\mathbf{s}^{n+1})$, defined as

$$\text{SANITY}(\mathbf{s}_i^{n+1}) = \begin{cases} 1 & : \min(\mathbf{s}_i^{n+1}) < \epsilon_2,\ \epsilon_2 = 10^{-6}, \\ 0 & : \text{otherwise},\ i = 1, \dots, N. \end{cases}$$

Hence, all states with a nearly zero density and/or pressure are marked as an invalid solution. The choice of ϵ_2 has to be based on the considered physical model. The definition of the reward is crucial for the success of the training. A suitable reward is derived under the following considerations. First, samples for which the positivity of the solution is not ensured have to be penalized by a negative reward. Second, the action predicted by the actor has to be optimal in the sense that the maximum permissible slope guarantees that oscillations are bounded over time. With these considerations in mind, the reward $\mathbf{r}$, which of course depends strongly on the discretization scheme considered, is defined as

$$\text{REWARD}(\mathbf{s}_i^{n+1}, \mathbf{s}_i^n, a_i) = \begin{cases} c_1 & : \mathbb{S}_i = 1, \\ |a_i| & : \mathbb{M}_i = 1 \wedge ((a_i < -1 + \epsilon_3 : |\hat{r}| > 1) \vee (a_i > 1 - \epsilon_3 : |\hat{r}| < 1)), \\ c_2|a_i| & : \mathbb{M}_i = 1 \wedge \neg((a_i < -1 + \epsilon_3 : |\hat{r}| > 1) \vee (a_i > 1 - \epsilon_3 : |\hat{r}| < 1)), \\ a_i & : \mathbb{M}_i = 0 \wedge |\hat{r}| > 1, \\ -a_i & : \mathbb{M}_i = 0 \wedge |\hat{r}| < 1,\ i = 1, \dots, N. \end{cases} \tag{4}$$

In eq. (4), the ratio $\hat{r}$ is defined as $\hat{r} = f(\mathbf{s}_i^n) = \frac{\rho_i - \rho_{i-1}}{\rho_{i+1} - \rho_i}$ with the tuning parameters $c_1 = -50$, $c_2 = -5$, $\epsilon_3 = 10^{-2}$, which may depend on the considered $\mathcal{S}$ and an oscillation indicator (relaxed discrete maximum principle) [18] is used, given as

$$\mathbb{M} = f(\mathbf{s}_i^n, \mathbf{s}_i^{n+1}) = \begin{cases} 1 & : (V_i^{n+1} < \min(\mathbf{s}_i^n)(1 - \epsilon_4)) \vee (V_i^{n+1} > \max(\mathbf{s}_i^n)(1 + \epsilon_4)),\ \epsilon_4 = 10^{-4}, \\ 0 & : \text{otherwise},\ i = 1, \dots, N, \end{cases}$$

where the sensitivity of the indicator can be tuned by ϵ_4, a common value in the literature is $\epsilon_4 = 10^{-3}$ [18]. The first case in eq. (4) accounts for invalid states. The other conditions are the following: First, states with $\mathbb{M}_i = 1$ (oscillating solution) require their action to be changed in order to obtain a less oscillatory representation and the reward is thus negative, scaled by the taken action (third case). However, it is important to ensure that these states are not penalized if their actions result in the lowest permissible slope (second case), i.e. the slope predicted by the *minmod* limiter. Second, for states with a less oscillatory or non-oscillating solution, the optimal action in terms of the maximum allowed slope depends on the size of $\hat{r}$. For $\hat{r} > 1$, the optimal action is $a^* \to 1$, otherwise $a^* \to -1$, see [14]. Hence, for $\hat{r} > 1$ (fourth case) or $\hat{r} < 1$ (fifth case), an action with a negative or positive sign, respectively, results in a negative reward and forces the actor to converge to a^*. The condition $\hat{r} = 1$ is never true as no constant samples are saved in $\mathcal{B}$.

After that, mini-batches of size $m = 100$ are randomly drawn from $\mathcal{B}$. The parameters of both critic networks and the ones of the actor and the target networks are optimized via the Adam algorithm [33] and a learning rate of $\eta = 10^{-3}$. The actor and the target networks are updated every $\tilde{d} = 2$ epoch [29]. For the first 100 epochs, the action is randomly drawn from $\mathbf{a} \sim \mathcal{U}[-1, 1]$ to ensure that the actor-critic networks have enough exploration and in turn are able to find the optimal action according to the above definition.

Algorithm 1 *RL*Indi-Framework

Initialize critic $Q_{\boldsymbol{\theta}_1}, Q_{\boldsymbol{\theta}_2}$ and actor $\pi_{\boldsymbol{\phi}}$ networks with random parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\phi} \sim \mathcal{N}(0, 1)$ and target networks, see [29]
Initialize replay buffer $\mathcal{B}$
for $i_{\text{epoch}} = 1$ **to** n_{epoch} **do**
  Initialize test case, $t^n = t_0$
  while $t < t_{\text{end}}$ **and** $\forall j \in \mathbb{S}\ :\ j = 0$ **do**
    Compute dt and state vector $\mathbf{s}_i^n = (V_{i-1}^n, V_i^n, V_{i+1}^n)^T,\ i = 1, \dots, N$ and $\tilde{\mathbf{s}}^n = \text{NORMALIZE}(\mathbf{s}^n)$
    Select action $\mathbf{a} = \text{clip}(\pi_{\boldsymbol{\phi}}(\tilde{\mathbf{s}}^n) + \epsilon, -1, 1),\ \epsilon \sim \mathcal{N}(0, \sigma)$ and compute slope $\delta\mathbf{q} = f(\mathbf{s}^n, \mathbf{a})$ according to eq. (3)
    Update solution to $t^{n+1} = t^n + dt$
    Compute state vector $\mathbf{s}_i^{n+1} = (V_{i-1}^{n+1}, V_i^{n+1}, V_{i+1}^{n+1})^T,\ i = 1, \dots, N$ and $\tilde{\mathbf{s}}^{n+1} = \text{NORMALIZE}(\mathbf{s}^{n+1})$
    Check positivity $\mathbb{S} = \text{SANITY}(\mathbf{s}^{n+1})$ and compute $\mathbf{r} = \text{REWARD}(\mathbf{s}^{n+1}, \mathbf{s}^n, \mathbf{a})$
    Store transition tuples $(\tilde{\mathbf{s}}^n, \mathbf{a}, \mathbf{r}, \tilde{\mathbf{s}}^{n+1}, \mathbb{S})$ in $\mathcal{B}$
  end while
  Sample mini-batches of m transitions from $\mathcal{B}$, update critics $\boldsymbol{\theta}_{1,2} \leftarrow \eta m^{-1} \sum_m (\mathbf{y} - Q_{\boldsymbol{\theta}_{1,2}}(\tilde{\mathbf{s}}^n, \mathbf{a}))^2$, $\mathbf{y}$ from eq. (2)
  if i_{epoch} mod $\tilde{d}$ **then**
    Update the actor according to eq. (1) and update the target networks, see [29]
  end if
end for

4 Numerical Results

For the validation of the methodology, a trained network is applied to several well-known shock tube problems. In this work, the Sod shock tube, the Toro 1 test case [15] (a modified version of the Sod problem, featuring a sonic point inside a rarefaction) and the Shu-Osher oscillatory shock tube [34], scaled to $x \in [0, 1]$, are discussed. The initial conditions are given in the same order by

$$\mathbf{v}_l = \begin{cases} (1,0,1)^T & : x < 0.5, \\ (1,0.75,1)^T & : x < 0.3, \\ (3.857143, 2.629369, 10.33333)^T & : x < 1/8, \end{cases} , \quad \mathbf{v}_r = \begin{cases} (0.125,0,0.1)^T & : x \geq 0.5, \\ (0.125,0,0.1)^T & : x \geq 0.3, \\ (1+0.2\sin(16\pi x),0,1)^T & : x \geq 1/8 \end{cases} \tag{5}$$

with simulation end times of $t_{\text{end}} = \{0.25, 0.2, 0.178\}$ and discretizations of $N = \{200, 200, 500\}$. The resolution increase to $N = 500$ for the Shu-Osher test is intended to adequately represent the oscillatory part of the solution.

The results are illustrated in fig. 1. The *RL*Indi is able to guarantee a stable simulation for all considered test cases, i.e. the solution is bounded over time. For the Sod shock tube, minor under-/overshoots at the edges of the contact discontinuity and the rarefaction wave are visible. The same holds for the Toro 1 test case, however, the oscillations are more pronounced. Although oscillating solutions such as the ones in the Shu-Osher shock tube are not considered during training, the *RL*Indi is capable of resolving these regions adequately. Moreover, the interaction of the oscillatory structure with the strong shock is well captured. The actions predicted by the *RL*Indi during the simulation are illustrated in fig. 2. The states between the waves are not always flagged as constant states due to numerical fluctuations. Hence, the ratio $\hat{r}$ is oscillating around $\hat{r} = 1$ and this in turn changes the optimal action due to the reward given in eq. (4) (case 4 and 5). Thus, the actions predicted by the *RL*Indi for these states are oscillating in the interval $\mathbf{a} \in [-1, 1]$. In addition, the actions $\mathbf{a}$ follow the definition of $\hat{r}$, i.e. $\mathbf{a} = -1$ for a convex ($\hat{r} < 1$) and $\mathbf{a} = 1$ for a concave density profile ($\hat{r} > 1$). Following the definition above, the action is set to 1 for $\hat{r} = 1$. Discontinuities are captured by a left-sided action of $a_L = 1$ and a right-sided action of $a_R = -1$.

5 Conclusion and Outlook

In this work, we proposed a reinforcement learning based slope limiter embedded in a second-order FV solver. Admissibility constraints based on the positivity of the solution and a relaxed discrete maximum principle were used to define a reward

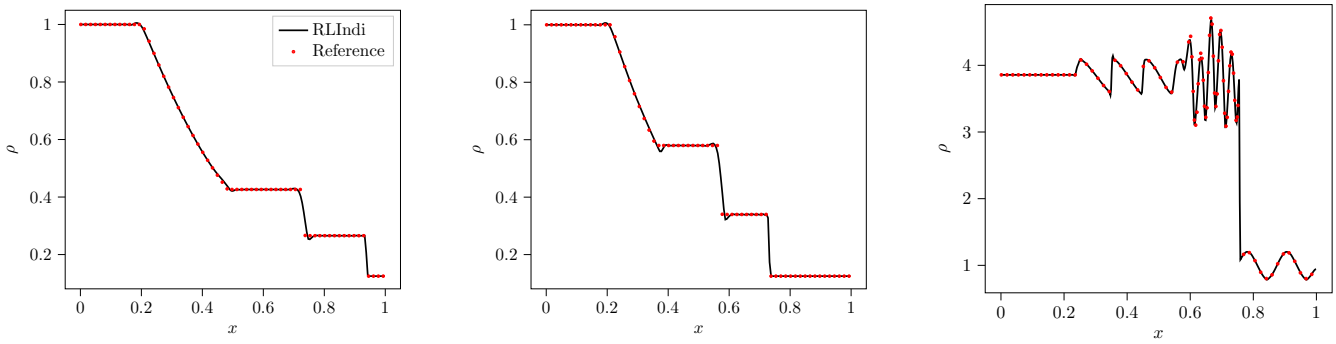

Fig. 1: Density of the Sod shock tube at $t = 0.25$ (left), the Toro 1 test case $t = 0.2$ (mid) and the Shu-Osher shock tube $t = 0.178$ (right). The reference solution (red) is computed on a grid of $N = 5000$ elements with the *minmod* limiter.

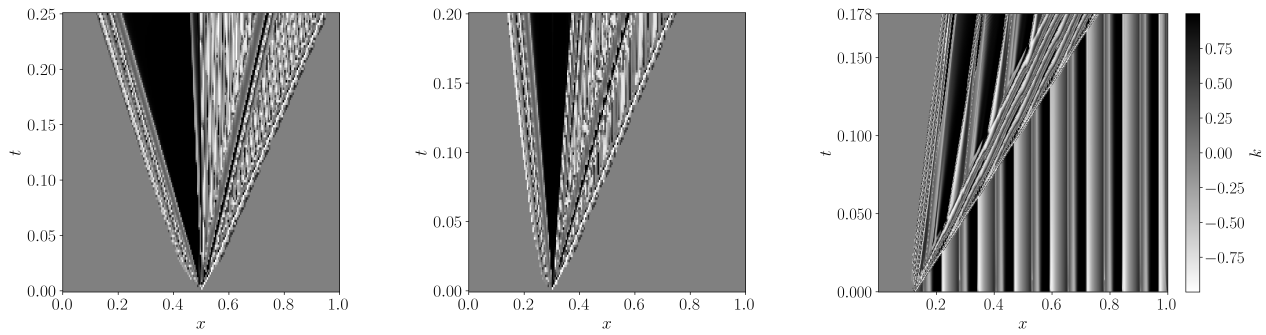

Fig. 2: Action, denoted as k, plotted in a $x - t$ diagram for the Sod shock tube up to $t = 0.25$ (left), the Toro 1 test case $t = 0.2$ (mid) and the Shu-Osher shock tube $t = 0.178$ (right). With actions of $k = 1$, $k = 0$ and $k = -1$, eq. (3) results in a left-sided, a central, and a right-sided slope, respectively.

function in an a posteriori fashion. Therefore, the long-term stability properties of the proposed a priori limiter converge towards those of the a posteriori limiter. The training was performed on arbitrarily initialized shock tube problems to enable a wide applicability. Furthermore, the efficiency of the framework is achieved by the use of an off-policy TD3 algorithm, which allows to store transitions in a replay buffer to guarantee a balanced training set with respect to the different reward classes. In addition, the replay buffer enables to store transitions which violate the positivity constraints. This ensures that the training process remains stable in its latter stages, where such cells are less often encountered. The limiter has been successfully applied to several well-known shock tube problems, which illustrates the potential of the proposed framework. Furthermore, it is very important to remark that training on simple two-state shock tubes was sufficient to generalize towards more complex flow phenomena not explicitly included in the training process, such as smooth oscillations occurring in the Shu-Osher problem.

In future works, arbitrary resolutions will be considered during training to improve the generalization capabilities of the limiter and the framework will be extended to two space dimensions to allow the simulation of complex flows. For this, convolutional neural networks are favorable with respect to efficiency. Further improvements with special emphasis on the composition of the training set and the speedup of convergence in general have to be tackled in the future. Moreover, ongoing research investigates the applicability of the framework to other discretization schemes such as discontinuous Galerkin methods and the limits of this data-driven approach.

Acknowledgements The research presented in this paper was funded in parts by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 - 390740016 and by the DFG Rebound - 420603919. J. K, and A. B. acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). S. C. acknowledges the support of the DFG via the project DROPIT, grant no. GRK 2160/2, and funding from Project HPC-EUROPA3 (INFRAIA-2016-1-730897). S. C. is a member of the INdAM GNCS group. The authors would like to thank Jonas Steiner and Lukas Zech for preliminary investigations. A. Schwarz and J. Keim agree to share the first authorship. Open access funding enabled and organized by Projekt DEAL.

References

[1] C. M. Dafermos, Hyperbolic conservation laws in continuum physics (Springer, 2010).
[2] A. Harten, Journal of Computational Physics **49**(3), 357–393 (1983).
[3] W. H. Reed and T. R. Hill, Los Alamos Scientific Laboratory Report LA-UR-73-479 (1973).
[4] S. K. Lele, Journal of Computational Physics **103**(1), 16–42 (1992).
[5] B. van Leer, Journal of Computational Physics **32**(1), 101–136 (1979).
[6] A. Harten and S. Osher, SIAM J. Num. Anal. **24**, 279–309 (1987).
[7] M. Dumbser, D. Balsara, E. Toro, and C. Munz, Journal of Computational Physics **227**(18), 8209–8253 (2008).
[8] S. Clain, S. Diot, and R. Loubère, Journal of Computational Physics **230**(10), 4028–4050 (2011).
[9] J. S. Hesthaven and R. M. Kirby, Mathematics of Computation **77**(263), 1425–1452 (2008).
[10] A. Jameson, W. Schmidt, and E. Turkel, AIAA Paper 81-1259 (1981).
[11] D. Kuzmin and M. Möller, in: Algebraic Flux Correction II. Compressible Euler Equations (Springer, 2005), pp. 207–250.
[12] M. Sonntag and C. D. Munz, Journal of Scientific Computing **70**(3), 1262–1289 (2017).
[13] S. K. Godunov and I. Bohachevsky, Matematičeskij sbornik **47(89)**(3), 271–306 (1959).
[14] R. J. LeVeque, Finite Volume Methods for Hyperbolic Problems (Cambridge University Press, 2002).
[15] E. F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics (Springer, 2009).
[16] C. Shu, NASA/CR-97-206253 ICASE Report No.97-65 (1997).
[17] B. Van Leer, On The Relation Between The Upwind-Differencing Schemes Of Godunov, Engquist-Osher and Roe (Springer, 1997).
[18] M. Dumbser, O. Zanotti, R. Loubère, and S. Diot, Journal of Computational Physics **278**, 47–75 (2014).
[19] D. Ray and J. S. Hesthaven, Journal of Computational Physics **367**, 166–191 (2018).
[20] D. Ray and J. S. Hesthaven, Journal of Computational Physics **397**, 108845 (2019).
[21] A. D. Beck, J. Zeifang, A. Schwarz, and D. G. Flad, Journal of Computational Physics **423**, 109824 (2020).
[22] M. Han Veiga and R. Abgrall, in: European Conference on Computational Mechanics and VII European Conference on Computational Fluid Dynamics, Glasgow, ECCM, 2525-2550., (2018).
[23] J. Yu and J. S. Hesthaven, Computers & Fluids **245**, 105592 (2022).
[24] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, Policy Gradient Methods for Reinforcement Learning with Function Approximation, Advances in Neural Information Processing Systems, Vol. 12 (MIT Press, 1999).
[25] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, Science **362**(6419), 1140–1144 (2018).
[26] D. Fan, L. Yang, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, Proceedings of the National Academy of Sciences **117**(42), 26091–26098 (2020).
[27] J. Kim, H. Kim, J. Kim, and C. Lee, Physics of Fluids (2022).
[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, Deterministic Policy Gradient Algorithms (PMLR, 2014).
[29] S. Fujimoto, H. van Hoof, and D. Meger, 35th International Conference on Machine Learning, ICML 2018 **4**, 2587–2601 (2018).
[30] H. V. Hasselt, Double Q-learning, in: Advances in Neural Information Processing Systems, (2010), pp. 2613–2621.
[31] D. Hendrycks and K. Gimpel, Gaussian Error Linear Units (GELUs), 2016.
[32] A. Harten, P. Lax, and B. van Leer, SIAM Review **25**, 35–61 (1983).
[33] D. P. Kingma and J. Bapp. 1–13 (2014).
[34] C. W. Shu and S. Osher, Journal of Computational Physics **83**(1), 32–78 (1989).

