La Matematica (2022) 1:53–84
https://doi.org/10.1007/s44007-021-00004-9

ORIGINAL RESEARCH ARTICLE

Multi-layer Perceptron Estimator for the Total Variation Bounded Constant in Limiters for Discontinuous Galerkin Methods

Xinyue Yu[1] · **Chi-Wang Shu**[1]

Received: 12 March 2021 / Revised: 14 August 2021 / Accepted: 19 August 2021 /
Published online: 8 November 2021

Abstract

The discontinuous Galerkin (DG) method is widely used in numerical solution of partial differential equations, especially for hyperbolic equations. However, for problems containing strong shocks, the DG method often needs to be supplemented by a limiter to control spurious oscillations and to ensure nonlinear stability. The total variation bounded (TVB) limiter is a popular choice and can maintain the original high order accuracy of the DG scheme in smooth regions and keep a sharp and non-oscillatory discontinuity transition, when a certain TVB constant M is chosen adequately. For scalar conservation laws, suitable choice of this constant M can be based on solid mathematical analysis. However, for nonlinear hyperbolic systems, there is no rigorous mathematical guiding principle for the determination of this constant, and numerical experiments often use *ad hoc* choices based on experience and through trial and error. In this paper, we develop a TVB constant artificial neural network (ANN) based estimator by constructing a multi-layer perceptron (MLP) model. We generate the training data set by constructing piecewise smooth functions containing local maxima, local minima, and discontinuities. By using the supervised learning strategy, the MLP model is trained offline. The proposed method gives the TVB constant M with robust performance to capture sharp and non-oscillatory shock transitions while maintaining the original high order accuracy in smooth regions. Numerical results using this new estimator in the TVB limiter for DG methods in one and two dimensions are given, and its performance is compared with the classical *ad hoc* choices of this TVB constant.

Keywords Discontinuous Galerkin method · Total variation diminishing · Total variation bounded · Limiters · multi-layer perceptron · Artificial neural network · Supervised learning strategy

✉ Chi-Wang Shu
chi-wang_shu@brown.edu

Extended author information available on the last page of the article

1 Introduction

The discontinuous Galerkin (DG) method was firstly proposed by Reed and Hill [31] to solve the neutron transport problem, which is a linear hyperbolic equation. It was later coupled with the total variation bounded (TVB) limiter [34] and the nonlinearly stable Runge-Kutta time discretization [35] to solve nonlinear hyperbolic conservation laws by Cockburn et al. [4–7], and has been widely used in computational fluid dynamics (CFD) applications, due to its high order accuracy and easy and efficient parallel implementation in complicated geometry. As is well known, the solution of nonlinear conservation laws often generates discontinuities, even with smooth initial and boundary conditions. Although the DG method can be proved to be L^2 and entropy stable for nonlinear hyperbolic scalar equations and systems [2, 3, 15, 16], this does not prevent the numerical solution from generating spurious oscillations near discontinuities. These oscillations are unpleasant in visualization, and, more seriously, they may lead to nonlinear instability for hyperbolic systems since hyperbolicity may be lost when such oscillations bring the numerical solution outside of the physical constraints (e.g. the appearance of negative density or pressure for compressible gas dynamics). To control these oscillations, nonlinear limiters are often used. They might be applied in specific cells using shock detectors (also called troubled cell indicators), such as the KXRCF shock detector developed by Krivodonova et al. [20], the troubled cell indicator of Fu and Shu [10], and the artificial neural network (ANN) based troubled cell indicator [29]. They may also be applied everywhere, with a careful design attempting to retain the original high order accuracy in smooth regions. Examples include the minmod-based total variation diminishing (TVD) limiters [14, 25], the minmod-based total variation bounded (TVB) limiter [34], the moment limiter [1], the monotonicity-preserving limiter [38], and the weighted essentially non-oscillatory (WENO) limiter [28]. A summary and comparison of limiters can found in [44].

One drawback of many of the limiters, including the popular minmod-based TVD limiters [14, 25], is that they may degenerate to first order accuracy near smooth extrema, even though they could retain the original high order accuracy in smooth and monotone regions [26]. To overcome this difficulty, Shu [34] designed a minmod-based TVB limiter, which can retain the original high order accuracy in smooth regions, including regions near smooth extrema. The adaptation and application of this TVB limiter to DG methods for solving scalar one-dimensional hyperbolic conservation laws were carried out in [6], and this limiter was further extended to DG methods solving one-dimensional systems and multi-dimensional cases in [4, 5, 7]. Comparing with the minmod-based TVD limiters, this TVB limiter significantly improves accuracy in smooth regions near solution extrema. However, it involves a TVB parameter M, which must be determined in a problem-dependent fashion. In the two extremes, $M = 0$ returns to the TVD limiter, and $M = +\infty$ returns to the original scheme without any limiter. If M is chosen too small, accuracy near smooth extrema might be affected; while if M is chosen too large, noticeable spurious oscillations may reappear

near discontinuities. For scalar nonlinear conservation laws, there exists rigorous mathematical guidance on the choice of M to guarantee that accuracy is maintained in smooth regions [6, 34]. However, for nonlinear systems, no such mathematical guidance exists, and hence in practice, M is usually chosen in an *ad hoc* fashion based on experience and through trial and error. With proper choices of the TVB constant M, DG schemes with the TVB limiter can give excellent resolution in CFD simulations. Besides the examples for compressible gas dynamics in [5, 7], we could also mention the application in [22], combined with a wet-dry moving boundary treatment, for solving shallow water equations. Also for solving shallow water equations, it works well on unstructured triangular meshes [42]. The TVB limiter is used to indicate the troubled cells in the application of special relativistic hydrodynamics [43]. Effort has also been made to provide guidance for an automated choice of the TVB constant M. A unified approach for the determination of this constant in mixed type meshes was studied and applied by Kontzialis et al. [19] and by Panourgias et al. [27], where M was chosen according to the variation of the derivatives of the numerical solution. In [39], Vuik and Ryan proposed an automatic parameter selection strategy for this TVB constant M based on Tukey's boxplot method of outlier-detection, and its application with compact-WENO finite element method is shown in [11].

In this paper, we aim to introduce an artificial neural network (ANN) based estimator for this TVB constant M by constructing a multi-layer perceptron (MLP) model. ANNs have the ability to approximate mappings with high-level complexity and nonlinearity, and thus they have undergone rapid developments and applications in numerical computation in recent years. For example, the ANNs are studied to solve ordinary and partial differential equations [12, 21, 33]. The multi-layer perceptron (MLP) is one of the most widely-used ANN models. It consists of an input layer, an output layer, and functional hidden layers. In [29, 30], Ray and Hesthaven constructed a troubled-cell indicator based on the MLP model, and Wen et al. applied it in finite difference WENO methods [40]. A well trained MLP model is free of problem-dependent parameter and hence suitable to be used as a unified approach for determining the TVB constant M in the TVB limiter applied to DG methods solving general conservation laws. We will construct function values containing information of discontinuities and local smooth extrema, and give the corresponding values of M in the training data set. The training process is performed offline, and the trained model should be able to return suitable *TVB* constant M to keep high order accuracy in smooth regions and eliminate spurious oscillations near discontinuities. The model will be added online into the DG framework with minimal modification on the standard TVB DG code to solve general conservation laws.

The outline of this paper is as follows. In Sect. 2, the background knowledge of the discontinuous Galerkin method and the minmod-based TVB limiter will be given. We will present the details for the construction of the training data set and the MLP model, as well as its implementation in the DG method, in Sect. 3. Numerical examples in 1D and 2D will be provided in Sect. 4, to demonstrate the good performance of the MLP-based TVB limiter in comparison with the *ad hoc* choice of the TVB constant M. Concluding remarks are given in Sect. 4.

2 Problem Setup and Preliminaries

2.1 Introduction of the DG Method

We consider the following conservation law:

$$\begin{cases} u_t + \nabla \cdot F(u) = 0, & \text{on } \Omega \subset \mathbb{R}^d, \quad d = 1, 2, \\ u(\cdot, 0) = u_0(\cdot), \end{cases} \tag{2.1}$$

where F is a linear or nonlinear flux function and Ω is a bounded domain in $\mathbb{R}^d$. In the one dimensional case, the conservation law is

$$\begin{cases} u_t + f(u)_x = 0, & \text{on } \Omega \subset \mathbb{R}, \\ u(x, 0) = u_0(x), \end{cases} \tag{2.2}$$

where $\Omega = [a, b]$. We discretize the domain by the partition $a = x_{1/2} < x_{3/2} < \cdots < x_{N+1/2} = b$. The cell I_i is denoted as $I_i = \{x : x_{i-1/2} < x < x_{i+1/2}\}$, for $1 \le i \le N$, and the mesh sizes are $h_i = x_{i+1/2} - x_{i-1/2}$. In this paper we will use uniform meshes $h_i = h$ for simplicity, unless specifically explained. We define a piecewise continuous polynomial space $V_h^k = \{p \in L_2(\Omega) : p|_{I_i} \in P^k(I_i)\}$, where $P^k(I_i)$ is the space of polynomials of degree $\le k$ in I_i. Then the one-dimensional DG method is stated as follows: Find $u_h(\cdot, t) \in V_h^k$, such that for all $v_h \in V_h^k$, u_h satisfies:

$$\frac{d}{dt} \int_{I_i} u_h(x,t) v_h(x,t)\, dx - \int_{I_i} f(u_h(x,t))(v_h(x,t))_x\, dx + \hat{f}_{i+\frac{1}{2}}\, v_h(x^-_{i+\frac{1}{2}}, t) - \hat{f}_{i-\frac{1}{2}}\, v_h(x^+_{i-\frac{1}{2}}, t) = 0, \tag{2.3}$$

where $\hat{f}_{i+\frac{1}{2}} = \hat{f}(u_h(x^-_{i+\frac{1}{2}}, t), u_h(x^+_{i+\frac{1}{2}}, t))$ is a monotone numerical flux in the scalar case and an exact or approximate Riemann-solver based numerical flux in the system case, see [5, 6].

To implement the DG method, one can use a local basis over I_i: $\boldsymbol{v}_i = (v_i^0, \ldots, v_i^k)^T$, and the numerical solution is expressed as

$$u_h(x, t) = \sum_{\ell=0}^{k} u_i^\ell(t) v_i^\ell(x), \quad \text{for} \quad x \in I_i. \tag{2.4}$$

The time dependent coefficients $\boldsymbol{u}_i(t) = (u_i^0(t), \ldots, u_i^k(t))^T$ are the computational variables to be evolved in time. If we take the test functions as $v_h = v_i^l$, $l = 0, \ldots, k$, the scheme can be written as

$$\sum_{\ell=0}^{k} \frac{du_i^\ell}{dt} \int_{I_i} v_i^l v_i^\ell\, dx = \int_{I_i} f\left(\sum_{\ell=0}^{k} u_i^\ell v_i^\ell \right) (v_i^l)_x\, dx - \hat{f}_{i+\frac{1}{2}}\, v_i^l(x_{i+\frac{1}{2}}) + \hat{f}_{i-\frac{1}{2}}\, v_i^l(x_{i-\frac{1}{2}}), \quad l = 0, \ldots, k. \tag{2.5}$$

The integrals in (2.5) can be computed either exactly or via suitable quadratures. The coefficients $\boldsymbol{u}_i$ can be obtained by using a proper time discretization to solve the ordinary differential equation (ODE) (2.5). In this paper, we will use the

third order TVD Runge–Kutta scheme (RK3) [35] in the computation. Denote $\boldsymbol{U}(t) = (\boldsymbol{u}_1(t), \dots, \boldsymbol{u}_N(t))^T$, the equation (2.5) can be written as

$$\frac{d}{dt}\boldsymbol{U}(t) = L(\boldsymbol{U}(t)),$$

where L is the spatial discretization operator. With $\boldsymbol{U}^n = \boldsymbol{U}(t_n)$, where t_n is n-th time step, the third order Runge–Kutta scheme is stated as follows:

$$\begin{aligned} \boldsymbol{U}^{(1)} &= \boldsymbol{U}^n + \Delta t L(\boldsymbol{U}^n), \\ \boldsymbol{U}^{(2)} &= \tfrac{3}{4}\boldsymbol{U}^n + \tfrac{1}{4}(\boldsymbol{U}^{(1)} + \Delta t L(\boldsymbol{U}^{(1)})), \\ \boldsymbol{U}^{n+1} &= \tfrac{2}{3}\boldsymbol{U}^n + \tfrac{1}{3}(\boldsymbol{U}^{(2)} + \Delta t L(\boldsymbol{U}^{(2)})). \end{aligned} \tag{2.6}$$

In the two dimensional case, the conservation law becomes

$$u_t + f(u)_x + g(u)_y = 0, \quad \text{on } \Omega \subset \mathbb{R}^2. \tag{2.7}$$

We consider the simple box geometry, and let $\Omega = [a_x, b_x] \times [a_y, b_y]$. Likewise, for simplicity of presentation, we use a rectangular mesh to cover the domain, consisting of the cells $I_{ij} = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}]$ for $1 \le i \le N_x$ and $1 \le j \le N_y$. Similar to the 1D case, we define $V_h^k = \{p \in L_2(\Omega) : p|_{I_{ij}} \in P^k(I_{ij})\}$ where $P^k(I_{ij})$ is the set of polynomials of degree $\le k$ over the cell I_{ij}. Recall the notation in (2.1) that $F(u) = (f(u), g(u))$. The 2D DG method is stated as follows: Find $u_h(\cdot, t) \in V_h^k$, such that for all $v_h \in V_h^k$, u_h satisfies:

$$\begin{aligned} &\frac{d}{dt}\int_{I_{ij}} u_h(x,y,t)v_h(x,y)dxdy - \int_{I_{ij}} F(u_h(x,y,t)) \cdot \nabla v_h(x,y)dxdy \\ &+ \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \hat{g}_{i,j+\frac{1}{2}}\, v_h(x, y^-_{j+\frac{1}{2}})dx - \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \hat{g}_{i,j-\frac{1}{2}}\, v_h(x, y^+_{j-\frac{1}{2}})dx \\ &+ \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \hat{f}_{i+\frac{1}{2},j}\, v_h(x^-_{i+\frac{1}{2}}, y)dy - \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \hat{f}_{i-\frac{1}{2},j}\, v_h(x^+_{i-\frac{1}{2}}, y)dy = 0, \end{aligned} \tag{2.8}$$

where $\hat{f}_{i+\frac{1}{2},j} = \hat{f}(u_h(x^-_{i+\frac{1}{2}}, y, t), u_h(x^+_{i+\frac{1}{2}}, y, t))$ is a one-dimensional numerical flux as defined before, likewise for $\hat{g}_{i,j+\frac{1}{2}}$. Consider a proper local basis over I_{ij}: $\boldsymbol{v}_{ij} = (v^0_{ij}, \dots, v^K_{ij})$ where $K = (k+1)(k+2)/2$, then the numerical solution is expressed as

$$u_h(x,y,t) = \sum_{\ell=0}^{K} u^\ell_{ij}(t) v^\ell_{ij}(x,y), \quad \text{for} \quad (x,y) \in I_{ij}. \tag{2.9}$$

Define the coefficients as $\boldsymbol{u}_{ij} = (u^0_{ij}, \dots, u^K_{ij})$, and take the test functions as $v_h = v^l_i$, $l = 0, \dots, K$, then the scheme can be written as

$$\sum_{\ell=0}^{k} \frac{du_{ij}^{\ell}}{dt} \int_{I_{ij}} v_{ij}^{l} v_{ij}^{\ell} dx = \int_{I_{ij}} F(\sum_{\ell=0}^{K} u_{ij}^{\ell}(t) v_{ij}^{\ell}(x,y)) \cdot \nabla v_{ij}^{l} dxdy$$
$$- \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \hat{g}_{i,j+\frac{1}{2}} v_{ij}^{l}(x, y_{j+\frac{1}{2}}^{-}) dx + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \hat{g}_{i,j-\frac{1}{2}} v_{ij}^{l}(x, y_{j-\frac{1}{2}}^{+}) dx$$
$$- \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \hat{f}_{i+\frac{1}{2},j} v_{ij}^{l}(x_{i+\frac{1}{2}}^{-}, y) dy + \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \hat{f}_{i-\frac{1}{2},j} v_{ij}^{l}(x_{i-\frac{1}{2}}^{+}, y) dy. \tag{2.10}$$

Again, the coefficients $\boldsymbol{u}_{ij}(t)$ can be obtained by solving the ODE (2.10) by the third order Runge–Kutta time discretization (2.6).

2.2 The Minmod-Based TVB Limiter

As mentioned in the introduction, the DG scheme provides high order accurate simulation of smooth solutions, and maintains L^2 and entropy stability for discontinuous solutions. However, this does not prevent the DG solution from showing spurious Gibbs oscillations near discontinuities, which may lead to nonlinear instability for solving nonlinear hyperbolic systems. Various nonlinear limiters are designed in the literature to control those spurious oscillations, while attempting to retain the original high order accuracy in smooth regions. In this section we describe the minmod-based TVB limiter [6, 34], which is the focus of our study in this paper.

In the one dimensional case, we denote the cell average of u_h in each cell I_i as:

$$\bar{u}_i = \frac{1}{h_i} \int_{I_i} u_h(x) dx.$$

We further denote by $\tilde{u}_i$ and $\tilde{\tilde{u}}_i$ the differences between the point values of the numerical solution at the cell boundaries and the cell average, and by $\Delta^+ \bar{u}_i$ and $\Delta^- \bar{u}_i$ the differences between the cell average of I_i and that of its neighboring cells:

$$\tilde{u}_i = u_h(x_{i+\frac{1}{2}}^{-}) - \bar{u}_i, \quad \tilde{\tilde{u}}_i = \bar{u}_i - u_h(x_{i-\frac{1}{2}}^{+}), \quad \Delta^+ \bar{u}_i = \bar{u}_{i+1} - \bar{u}_i, \quad \Delta^- \bar{u}_i = \bar{u}_i - \bar{u}_{i-1}. \tag{2.11}$$

A nonlinear limiter changes the polynomial solution u_h in the cell I_i, while keeping the cell average $\bar{u}_i$ unchanged to maintain conservation. The purpose of the nonlinear limiter is to control spurious oscillations near discontinuities, while attempting to retain the original high order accuracy in smooth regions. The minmod-based TVD limiter [14, 25] modifies $\tilde{u}_i$ and $\tilde{\tilde{u}}_i$ by a limiter function:

$$\tilde{u}_i^{mod} = m(\tilde{u}_i, \Delta^+ \bar{u}_i, \Delta^- \bar{u}_i), \qquad \tilde{\tilde{u}}_i^{mod} = m(\tilde{\tilde{u}}_i, \Delta^+ \bar{u}_i, \Delta^- \bar{u}_i). \tag{2.12}$$

Once the modified values $\tilde{u}_i^{(mod)}$ and $\tilde{\tilde{u}}_i^{(mod)}$ are obtained, we can obtain the modified point values of the numerical solution at the cell boundaries:

$$u_h^{(mod)}(x^-_{i+\frac{1}{2}}) = \bar{u}_i + \tilde{u}_i^{(mod)}, \qquad u_h^{(mod)}(x^+_{i-\frac{1}{2}}) = \bar{u}_i - \tilde{\tilde{u}}_i^{(mod)}.$$

With the two modified point values $u_h^{(mod)}(x^-_{i+\frac{1}{2}})$, $u_h^{(mod)}(x^+_{i-\frac{1}{2}})$ and the original cell average $\bar{u}_i$, we can recover a unique p^k polynomial with $k \leq 2$ as the limited solution $u_h^{(mod)}$. For $k > 2$, we still recover a quadratic polynomial if the limiter is enacted (that is, if the limiter function m in (2.12) returns other than the first argument), since accuracy is not expected to be maintained in this case.

We now turn to the specific choices of the limiter function m in (2.12).

For the minmod-based TVD limiter [14, 25], m is defined as the minmod function

$$m(a_1, a_2, a_3) = \begin{cases} s \ \min(|a_1|, |a_2|, |a_3|), & \text{if } s = \text{sign}(a_1) = \text{sign}(a_2) = \text{sign}(a_3), \\ 0, & \text{otherwise}. \end{cases} \tag{2.13}$$

In words, the minmod function m returns the smallest argument (in magnitude), if all arguments have the same sign; otherwise it returns zero.

It can be proved [6] that, when the minmod limiter (2.13) is used and if the time discretization is via a TVD Runge–Kutta method such as (2.6), then the limited DG solution is total variation diminishing in the means (TVDM). This is a rather strong nonlinear stability property and prevents completely any spurious oscillations in the means near discontinuities. However, the drawback is that, as any TVD schemes, the method will suffer from accuracy degeneracy to first order near smooth extrema [26], hence the global accuracy in L^1 is at most second order for generic smooth solutions with finitely many smooth extrema.

For the minmod-based TVB limiter [34], m is defined as

$$m^{tvb}(a_1, a_2, a_3, h, M) = \begin{cases} a_1, & \text{if } |a_1| \leq Mh^2, \\ m(a_1, a_2, a_3), & \text{otherwise}, \end{cases} \tag{2.14}$$

where h is a local mesh size, $M \geq 0$ is a TVB constant, and m is the minmod function defined in (2.13). It can be shown [6] that, when the TVB limiter (2.14) is used and if the time discretization is via a TVD Runge–Kutta method such as (2.6), then the limited DG solution is total variation bounded in the means (TVBM). This is again a rather strong nonlinear stability property.

It is expected that the performance of the TVB limiter depends strongly on the choice of the TVB constant M. If M is chosen too large, noticeable spurious oscillations may reappear near discontinuities. After all, for $M = +\infty$, the limiter m^{tvb} in (2.14) will always return the first argument, namely we will obtain the unlimited solution. On the other hand, if M is chosen too small, the scheme may lose the original high-order accuracy near smooth extrema, just like the TVD minmod limiter. After all, for $M = 0$, we recover the TVD minmod limiter defined in (2.13). On the approximation level, given a smooth function u, the following result is proved in [6].

Lemma 2.1 *If u is a smooth function, and $M_2 = \max_x |u_{xx}|$. Then, if M is taken as*

$$M \geq \frac{2}{3} M_2, \tag{2.15}$$

the limiter (2.14) will not affect accuracy. That is, it will always return the first argument.

In fact, M_2 can be taken as a upper bound for the magnitude of the second derivative near the smooth extrema, rather than over the whole range of x.

The approximation result in the lemma above is also valid for linear or nonlinear scalar conservation laws. For one dimensional scalar conservation laws (2.2), we have the following lemma.

Lemma 2.2 *If u is the solution of the one dimensional scalar conservation law (2.2), the initial condition $u_0(x)$ is smooth near $x = x_0$, and $u_0'(x_0) = 0$, then along the forward characteristic line*

$$x(t) = x_0 + f'(u_0(x_0))t,$$

we have

$$u(x,t) = u(x_0), \qquad u_x(x,t) = 0, \qquad u_{xx}(x,t) = u_0''(x_0).$$

That is, along a smooth local extremum, the second derivative u_{xx} is invariant (constant in time).

Lemma 2.2 can be easily proved by solving the ODEs involving the evolution of u, u_x and u_{xx} along the forward characteristic line. Based on Lemmas 2.1 and 2.2 , we conclude that the choice of M by (2.15), where $M_2 = \max_x |u_0''(x)|$, ensures that the limiter (2.14) will not affect accuracy. That is, it will always return the first argument. Thus, the choice of M to ensure high order accuracy in smooth regions for scalar conservation laws can be given with solid mathematical justification. In practice, because of the numerical errors near smooth extrema, we often take a slightly larger value of M than that given by (2.15), e.g. by $M = cM_2$ with $c > \frac{2}{3}$.

However, for nonlinear hyperbolic systems, there is no such mathematical guidance for the choice of the TVB constant M. This is because the value of u_{xx} at a smooth extremum is no longer invariant in time, hence cannot be determined based solely on the initial condition. The choice of M in such cases is then often given in an *ad hoc* fashion, based on experience and through trial and error. In this paper, we would like to develop a constant estimator for M, based on an artificial neural network (ANN) based model, so that the TVB constant M can be chosen automatically .

3 The Multi-layer Perceptron (MLP) Limiter

Our work on the construction of an ANN-based constant estimator is enlightened by the MLP troubled cell indicator developed by Hesthaven and Ray [29], which detects the location of discontinuities according to the function values at the cell

boundaries and the local cell averages. Inspired by [29], we aim at constructing a constant estimator using the ANN model, which is able to (1) distinguish the cells near local extrema, discontinuities and in smooth monotone regions by the point values at the cell interface and the cell averages; (2) directly return the TVB limiter constant M accordingly, that maintains high order accuracy in smooth regions and non-oscillatory transaction at discontinuities. The multi-layer perceptron (MLP) model is one of the most commonly used artificial neural network model. The idea of developing a hypothetical nervous system (called a perceptron) and imitating learning curves from neurological variables is introduced by Rosenblatt in 1958 [32]. In [24], Novikoff proved the perceptron convergence theorem, i.e., if the training data set is linearly separable, the convergence of the perceptron is guaranteed. The capability of approximating continuous functions of MLP is studied in [9, 13]. The MLP model is well-known for its ability to estimate the relationship with high degree of complexity and nonlinearity. The other advantage of the model is that, the main computational cost of the model comes from the offline training procedure, and the online computational procedure involves simple matrix multiplications with negligible extra cost over the original DG scheme. As shown in Fig. 1, the MLP model consists of an input layer, an output layer, and several hidden layers, including a normalization layer and fully connected layers.

This model can be viewed as an approximation map from the input layer to the output layer,

$$\boldsymbol{F}: \quad \mathbb{R}^{N_1} \mapsto \mathbb{R}^{N_o}, \qquad y = f(x|(\boldsymbol{w}, \boldsymbol{b})), \tag{3.1}$$

where the weights $\boldsymbol{w}$, the bias $\boldsymbol{b}$ and the activation function contained in the hidden layer determine the value of the predicted outputs. The cost function is then applied to measure the error between the network predicted output and the true output value given in the training data set. During the training process, proper training strategy, like the supervised learning [18] we use in this paper, is utilized to minimize the error by adjusting the weight and the bias. A well-trained model is capable

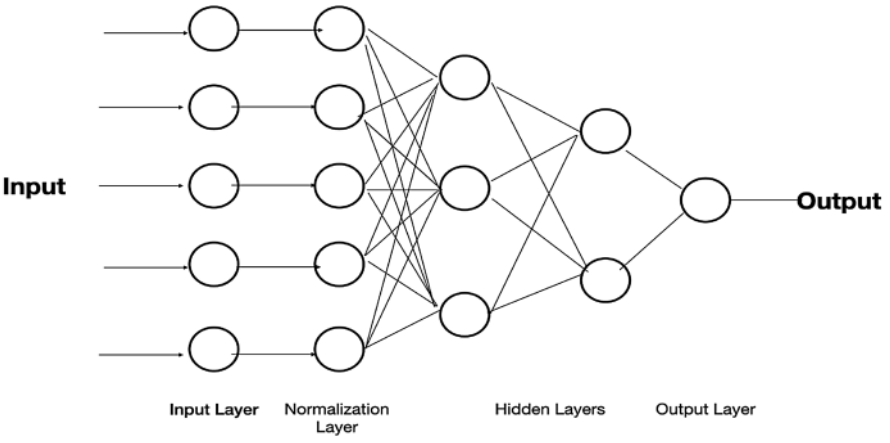


Fig. 1 An MLP model with an input layer, a normalization layer, hidden layers, and an output layer

of precisely predicting the outputs according to the input data, even when the input is not included in the training set.

3.1 Construction of the Training Data

Now we will introduce the design of the MLP-based estimator. In our case, the input data are function values in and near the cell I_i, i.e $\boldsymbol{v} = (\bar{u}_{i-1}, \bar{u}_i, \bar{u}_{i+1}, u_h(x^-_{i+1/2}), u_h(x^+_{i-1/2}))^T \in \mathbb{R}^5$. The output would be the corresponding TVB limiter constant M_i for the cell I_i. The input and output training data sets are denoted as $\mathbb{V}_x$ and $\mathbb{V}_y$ respectively, which are generated via the following two ways. Firstly, due to the fact that the DG solutions are piecewise polynomial functions approximating the real PDE solution, the type I data are function values from the L^2 projection of designed functions into suitable piecewise polynomial spaces. Secondly, inspired by the work of Sun et al. [37], we consider the effect of the numerical method on the solution's structure, such as the Gibbs oscillations near discontinuities or the smearing caused by the numerical dissipation. To enable the model to learn the feature of the numerical solutions, the data from numerical solutions of the DG method solving the advection equation $u_t + au_x = 0$ with discontinuous initial conditions are added. The detailed procedure is listed below.

Type I. Data from piecewise polynomial functions.

1. In the interval $[a, b]$, choose piecewise smooth functions $u(x)$ containing one or more features listed below:
   - Containing smooth monotone regions;
   - Containing discontinuity points;
   - Containing local smooth maxima and/or local smooth minima.
2. Pick a point x and a mesh size h randomly, such that $a < x - \frac{3}{2}h < x + \frac{3}{2}h < b$, and construct a three-cell stencil containing $I_{i-1} = (x - \frac{3}{2}h, x - \frac{1}{2}h)$, $I_i = (x - \frac{1}{2}h, x + \frac{1}{2}h)$, and $I_{i+1} = (x + \frac{1}{2}h, x + \frac{3}{2}h)$.
3. Use the standard L^2 projection to project $u(x)$ onto the piecewise polynomial space with different degrees of freedom within each cell, and denote the obtained polynomials in each cell of the three-cell stencil as $u_{i-1}(x)$, $u_i(x)$, and $u_{i+1}(x)$.
4. Collect the input data, i.e, $\boldsymbol{v} = (\bar{u}_{i-1}, \bar{u}_i, \bar{u}_{i+1}, u_i(x + \frac{1}{2}h), u_i(x - \frac{1}{2}h))^T$.
5. Determine corresponding output value $y = M \in \mathbb{V}_y$ by the following strategy:
   - If the interval $I = (x - \frac{3}{2}h, x + \frac{3}{2}h)$ contains a discontinuity point, the standard minmod limiter should be applied to control spurious oscillations, i.e. $y = M = 0$;
   - If the interval I contains a local maximum or a local minimum, we define $M = \frac{2}{3}c \max_{x \in I} |u''(x)|$. Here c is a constant greater than 1, to make M a safer upper bound according to Lemmas 2.1 and 2.2 for maintaining the original high order accuracy. In our numerical computation, we have taken the value $c = 5$.

- If $u(x)$ in the interval I is smooth and monotone, we choose M big enough so that the minmod limiter is not enacted (i.e. it returns the first argument). In our numerical computation, we have taken the value $M = 1000$ in this case.

Type II. Data from the numerical solution.

1. We generate the piecewise smooth initial condition u_0 by the following procedure:
   - Select the number of discontinuities contained in the initial condition: $1 \le N_d \le 6$;
   - Randomly select N_d locations for the discontinuities in the domain $[-1, 1]$, and divide the domain into $N_d + 1$ subdomains;
   - Within each subdomain, create random Fourier series $a_0 + \sum_{n=1}^{N_f}(a_n \cos(nx) + b_n \sin(nx))$ with different $1 \le N_f \le 6$, and i.i.d random variables a_0, a_n, and b_n.
2. Use different mesh sizes $h = \frac{1}{30}, \frac{1}{60}, \frac{1}{90}, \frac{1}{180}$ to generate uniform meshes with N_x cells.
3. With a random advection coefficient $a \in [-1, 1]$, apply the Runge–Kutta DG (RKDG) scheme with the degree of freedom k to compute the solution for N_t time steps, where $N_t = 1, 2, 3$ and $k = 1, 2, 3, 4$. The time step size is chosen as $\Delta t = \mathrm{C}\ \frac{h}{a}$, where the CFL constant is chosen as $\mathrm{C} < \frac{1}{2k+1}$. The obtained numerical solution in cell I_i is denoted as u_i.
4. Collect the data from the numerical solution, i.e, $\boldsymbol{v} = (\bar{u}_{i-1}, \bar{u}_{hi}, \bar{u}_{i+1}, u_i(x^-_{i+1/2}), u_i(x^+_{i-1/2}))^T$.
5. The cell is considered to contain a discontinuity or a local smooth extremum if the exact solution $u(x, t) = u_0(x - aN_t\Delta t)$ has discontinuity or a local extremum within the cell or its left or right neighbor cell, and $y = M \in \mathbb{V}_y$ is determined using the same strategy of step 5 in Type I. In general there could exist differences in the locations of discontinuities between the exact and the numerical solutions. In our case, only a few time steps are computed, therefore the difference can be neglected. It enables us to use the location of discontinuities in the exact solution to determine M.

Based on the above guideline, the training data set is constructed, and the details of this data set can be viewed in Table 1. For the Type I data, the mesh size h and the degrees of freedom of the projected polynomial space $k \in \{1, 2, 3, 4\}$ are varied.

3.2 The MLP Model

We now briefly introduce the MLP training model. The input is a 5-dimensional vector $\boldsymbol{v}$. Before feeding the data into the hidden layers, we firstly add a normalization layer to normalize the data as follows. Denote the l-th element of the input vector $\boldsymbol{v}$ as v^l, $l = 1 \ldots 5$. The normalized function value would be $\tilde{\boldsymbol{v}}$, with the l-th element $\tilde{v}^l$ given by

Table 1 Rows 2–6 are the functions used to generate the Type I data

u(x)	Domain	Varied parameters	Discontinuities	Local extrema	Total
$a\|x\|$	$[-0.5,0.5]$	$a \in [1,10]$	1000	0	1000
$u_l I_{x<a} + u_r I_{x>a}$	$[-1,1]$	$(u_l, u_r) \in [-4,4]^2$ $a \in [-0.56, 0.56]$	3200	0	3200
$\sin(k\pi x)$	$[0, \frac{k}{4}]$	$k = 1,\ldots,25$	0	720	6480
$\sin(2\pi x)\cos(3\pi x)\sin(4\pi x)$	[0,1]		0	504	1400
$\sin^4(\pi x)$	[0,1]		0	144	1400
Type II data			950	1695	8451
Total			5150	3063	21931

The last three columns are the numbers of cells containing discontinuities, local extrema, and total cell numbers. The second last row is the number of different types of cells in the data generated by the numerical solution of the DG scheme. The last row is the total data number in the data set, which is obtained by adding the data above within each column

$$\tilde{v}^l = \frac{v^l - \mu}{\sigma}, \tag{3.2}$$

where μ and σ are the mean and the standard deviation of the elements of all $\boldsymbol{v}$ in $\mathbb{V}_x$.

We apply five hidden layers containing 128, 64, 32, and 16 neurons respectively. Within each hidden layer, the weights and bias are randomly initialized using a normal distribution, and Leaky rectified linear unit (Leaky ReLU) is chosen as the activation [23]. The output layer has one neuron, as the output is the value of the limiter TVB constant M. The cost function is given by the mean squared error (MSE) function. The data set is split into two subsets, with 80% data used for training and the remaining 20% data for validation. The model is trained using the Adam optimization [17] with the batch size $S_b = 500$, and with 2000 iterations. Keras API is used for the model training (https://keras.io/).

3.3 Implementation of the Estimator

After obtaining the well-trained model, it is simple to implement the estimator. The algorithm in the one-dimensional scalar case is described as follows:

1. Apply the DG method for the spatial discretization, and proceed with one Euler forward step in the third order Runge–Kutta time discretization.
2. Generate $\boldsymbol{v}_i = (\bar{u}_{i-1}, \bar{u}_i, \bar{u}_{i+1}, u_h(x^-_{i+1/2}), u_h(x^+_{i-1/2}))^T$ within each cell I_i.
3. Feed the data into the estimator, and obtain the corresponding M_i for each cell.
4. Apply M_i in the minmod-based TVB limiter, and obtain the limited solution.
5. Repeat Steps 1-4 twice for the next two Runge–Kutta inner stages, and finish the computation of the current time step.

There is no need to change the structure of the original DG code to implement the estimator. Since $\boldsymbol{v}_i$ in Step 2 is also needed in the minmod-based TVB limiter, the only extra work is adding Step 3 to predict the value of M, and in practice it is an one-line addition in the code.

In the two dimensional scalar case, we need to generate in the x direction and in the y direction:

$$\begin{aligned} \boldsymbol{v}_{ij}^{x} &= (\bar{u}_{i-1,j}, \bar{u}_{i,j}, \bar{u}_{i+1,j}, u_h(x^{-}_{i+\frac{1}{2}}, y_j), u_h(x^{+}_{i-\frac{1}{2}}, y_j))^T, \\ \boldsymbol{v}_{ij}^{y} &= (\bar{u}_{i,j-1}, \bar{u}_{i,j}, \bar{u}_{i,j+1}, u_h(x_i, y^{-}_{j+\frac{1}{2}}), u_h(x_i, y^{+}_{j-\frac{1}{2}}))^T, \end{aligned} \tag{3.3}$$

we feed them into the estimator to obtain the predicted limiter TVB constants M_{ij}^{x} and M_{ij}^{y} respectively, and apply them in the limiter. It is clear that there is a low coding cost for the implementation of the estimator in the 2D case as well.

For hyperbolic systems, the estimator and the limiter could be applied component by component, but they are more effective if they are applied in local characteristic fields, which is the procedure that we adopt in our numerical tests. We refer to [5, 7] for more details.

4 Numerical Tests

In this section, we will perform several standard numerical tests in one- and two-dimensions. For the scalar case, we will solve the linear advection equation and the nonlinear Burgers equation, and in the case of systems, the Euler equation of compressible gas dynamics will be approximated. Within each subsection, accuracy tests will be given for the DG scheme with the MLP limiter for the degrees of freedom $k = 1, 2, 3$, when the exact solution is smooth. The results will be compared against DG schemes without the limiter. In the case that exact solutions are discontinuous, the performance of the MLP limiter will be presented and compared to that of the TVB limiter with the TVB constant M chosen in an *ad hoc* fashion through trial and error as given in the literature. In general, the MLP limiter has outstanding performance when applied to the DG method of different degrees of freedom. In all accuracy tests, periodic boundary condition is applied, and the simulations run until $t = 0.3$. The CFL conditions are set to be CFL $= 0.3$ for $k = 1$, CFL $= 0.18$ for $k = 2$, and CFL $= 0.1$ for $k = 3$, according to the linear stability analysis [8].

4.1 Linear Advection Equation

We firstly consider the one-dimensional linear advection equation with sine wave initial condition:

$$\begin{cases} u_t + u_x = 0, \\ u(x, 0) = \sin(x), \quad x \in [0, 2\pi]. \end{cases} \tag{4.1}$$

Table 2 demonstrates the error and order of accuracy for the DG scheme with and without the MLP limiter. The MLP limiter method obtains the desired second, third and fourth order accuracy respectively, when applied to the DG scheme with degrees of freedom $k = 1, 2, 3$. The error and order are very close to that of the DG method without the limiter, indicating that the MLP limiter has the correct estimate for the TVB constant M and can maintain the original high order of accuracy.

To check the behavior of the limiter under discontinuous situation, we consider the multi-wave problem, with the initial condition given by

$$u_0(x) = \begin{cases} 10(x - 0.2), & 0.2 < x < 0.3, \\ 10(0.4 - x), & 0.3 < x < 0.4, \\ 1, & 0.6 < x < 0.8, \\ 100(x - 1)(1.2 - x), & 1.0 < x < 1.2, \\ 0, & \text{otherwise}. \end{cases} \tag{4.2}$$

Table 2 Accuracy test for 1D linear advection equation

	$k = 1$ DG MLP-limiter				$k = 1$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16	4.83 E−03		3.10 E−03		4.39 E−03		2.27 E−03	
32	1.29 E−03	1.90	6.51 E−03	2.25	1.22 E−03	1.84	6.25 E−03	1.86
64	3.15 E−04	2.03	1.60 E−03	2.02	3.41 E−04	1.95	1.60 E−03	1.96
128	7.86 E−05	2.00	4.01 E−04	2.00	7.86 E−05	2.00	4.01 E−04	2.00
256	1.96 E−05	2.00	1.00 E−04	2.00	1.96 E−05	2.00	1.09 E−04	2.00
	$k = 2$ DG MLP-limiter				$k = 2$ DG no limiter			
# cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16	1.67 E−04		7.68 E−04		1.78 E−04		7.69 E−04	
32	2.18 E−05	2.94	1.35 E−04	2.50	2.28 E−05	2.96	1.46 E−04	2.39
64	2.47 E−06	3.13	1.55 E−05	3.11	2.46 E−06	3.21	1.51 E−05	3.27
128	3.12 E−07	2.98	1.97 E−06	2.94	3.12 E−07	2.98	1.97 E−06	2.94
256	3.90 E−08	2.97	2.46 E−07	3.00	3.89 E−08	2.97	2.46 E−07	3.00
	$k = 3$ DG MLP-limiter				$k = 3$ DG no limiter			
# cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16	4.02 E−06		2.07 E−05		4.02 E−06		2.07 E−05	
32	2.67 E−07	3.91	1.69 E−06	3.62	2.67 E−07	3.91	1.69 E−06	3.62
64	1.34 E−08	4.31	1.09 E−07	3.95	1.34 E−08	4.31	1.09 E−07	3.95
128	8.75 E−10	3.94	6.51 E−09	4.07	8.75 E−10	3.94	6.51 E−09	4.07
256	5.67 E−11	3.95	4.09 E−10	3.99	5.67 E−11	3.95	4.09 E−10	3.99

The domain is [0, 1.4], and periodic boundary condition is applied. The solution is evaluated at $t = 1.4$ using $N = 100$ cells. In this case, we use the randomly perturbed meshes, which is constructed based on a uniform mesh:

$$x_{i+\frac{1}{2}} \to x_{i+\frac{1}{2}} + \theta h_{i+\frac{1}{2}} \omega_{i+\frac{1}{2}}, \quad \omega_{i+\frac{1}{2}} \in \mathbb{U}([-0.5, 0.5]) \quad i = 1, \ldots, N-1,$$

where we choose $\theta = 0.15$. For all simulations as shown in Fig. 2, the performance of the TVB limiter with different TVB constants $M = 0, 10, 100, 1000$ and the MLP limiter are compared. The choices of $M = 0, 10$ smear significantly at the two local maxima, and $M = 100, 1000$ fail to control oscillations near the discontinuities. However, the MLP limiter can precisely catch the local extrema without causing oscillation near the discontinuities. Figure 3 depicts the temporal history of the TVB constant M chosen by the MLP model. The MLP model precisely captures the discontinuous points and local extrema, and returns the corresponding M.

In the two-dimensional linear case

$$\begin{cases} u_t + u_x + u_y = 0, \\ u(x, y, 0) = \sin(x + y), \quad (x, y) \in [0, 2\pi] \times [0, 2\pi], \end{cases} \tag{4.3}$$

the error and orders of the DG method with the MLP limiter and without the limiter are listed in Table 3. The MLP limiter again preserves high order accuracy in this 2D example.

4.2 Burgers Equation

We consider the nonlinear Burgers equation in $1D$:

$$\begin{cases} u_t + \left(\frac{u^2}{2}\right)_x = 0, \\ u(x, 0) = \frac{1}{4} + \sin(x), \quad x \in [0, 2\pi]. \end{cases} \tag{4.4}$$

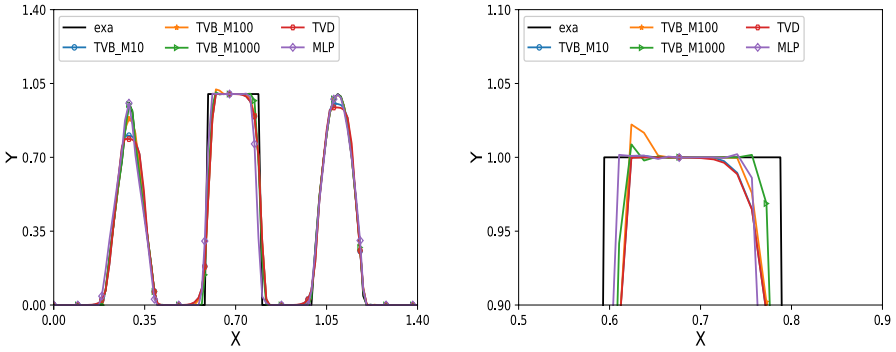


Fig. 2 Solution for the multi-wave problem using the fourth order DG method, at the final time $t = 1.4$. The right figure is zoomed near $x = 0.7$

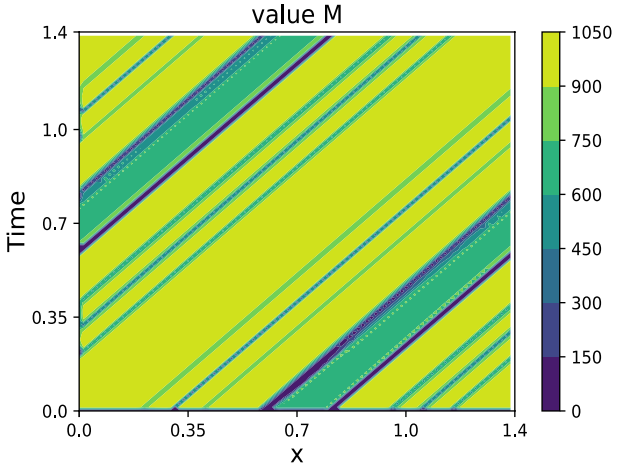


Fig. 3 Temporal history of the TVB constant M chosen by the MLP model of the multiwave problem, $k = 2$

Table 3 Accuracy test for 2D linear advection equation

	$k = 1$ DG MLP-limiter				$k = 1$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	1.03 E−02		9.54 E−02		1.03 E−02		9.54 E−02	
32×32	2.60 E−03	1.98	2.52 E−03	1.91	2.60 E−03	1.98	2.52 E−03	1.91
64×64	6.52 E−04	2.00	6.40 E−03	1.98	6.52 E−04	2.00	6.40 E−03	1.98
128×128	1.62 E−04	2.00	1.60 E−03	2.00	1.62 E−04	2.00	1.60 E−03	2.00
256×256	4.06 E−05	2.00	4.01 E−04	2.00	4.06 E−05	2.00	4.01 E−04	2.00
	$k = 2$ DG MLP-limiter				$k = 2$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	9.48 E−04		5.84 E−03		9.48 E−04		5.84 E−03	
32×32	9.89 E−05	3.26	1.21 E−03	2.26	9.89 E−05	3.26	1.21 E−03	2.26
64×64	1.14 E−05	3.11	1.46 E−04	3.05	1.14 E−05	3.11	1.46 E−04	3.05
128×128	1.42 E−06	3.00	1.87 E−05	2.97	1.42 E−06	3.00	1.87 E−05	2.97
256×256	1.78 E−07	3.00	2.34 E−06	3.00	1.78 E−07	3.00	2.34 E−06	3.00
	$k = 3$ DG MLP-limiter				$k = 3$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	5.11 E−05		9.79 E−04		5.11 E−05		9.79 E−04	
32×32	3.20 E−06	3.99	6.09 E−05	4.00	3.20 E−06	3.99	6.09 E−05	4.00
64×64	2.01 E−07	3.99	3.74 E−06	4.02	2.01 E−07	3.99	3.74 E−06	4.02
128×128	1.27 E−08	3.98	2.05 E−07	4.05	1.27 E−08	3.98	2.05 E−07	4.05
256×256	8.24 E−10	3.95	1.27 E−08	4.13	8.24 E−10	3.95	1.27 E−08	4.13

Before $t = 1$, the solution is smooth, and we can compare the accuracy of the DG scheme with and without the MLP limiter. From Table 4, we observe that applying the limiter does not affect accuracy also in this nonlinear case.

Next we test the compound wave problem, with a discontinuous initial condition:

$$u_0(x) = \begin{cases} l\sin(\pi x), & |x| \geq 1, \\ 3, & -1 < x \leq -0.5, \\ 1, & -0.5 < x \leq 0, \\ 3, & 0 < x \leq 0.5, \\ 2, & 0.5 < x \leq 1, \end{cases} \tag{4.5}$$

The domain is $[-4, 4]$ with a randomly perturbed mesh, and the periodic boundary condition is applied. We can see the numerical result at $t = 0.4$ in Fig. 4. The MLP

Table 4 Accuracy test for 1D Burgers equation

	$k = 1$ DG MLP-limiter				$k = 1$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16	4.53 E−03		2.67 E−02		4.53 E−03		2.67 E−02	
32	1.05 E−03	2.10	6.41 E−03	2.05	1.05 E−03	2.10	6.41 E−03	2.05
64	2.62 E−04	2.00	1.63 E−03	1.97	2.62 E−04	2.00	1.63 E−03	1.97
128	6.56 E−05	2.00	4.11 E−04	1.99	6.56 E−05	2.00	4.11 E−04	1.99
256	1.63 E−05	2.00	1.03 E−04	1.99	1.63 E−05	2.00	1.03 E−04	1.99
	$k = 2$ DG MLP-limiter				$k = 2$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16	1.17 E−04		4.96 E−04		1.17 E−04		4.96 E−04	
32	1.45 E−05	3.00	6.28 E−05	2.98	1.45 E−05	3.00	6.28 E−05	2.98
64	1.82 E−06	3.00	7.87 E−06	3.00	1.82 E−06	3.00	7.87 E−06	3.00
128	2.28 E−07	3.00	9.85 E−07	3.00	2.28 E−07	3.00	9.85 E−07	3.00
256	2.84 E−08	3.00	1.23 E−07	3.00	2.84 E−08	3.00	1.23 E−07	3.00
	$k = 3$ DG MLP-limiter				$k = 3$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	order	L^∞ error	order
16	9.88 E−06		1.51 E−04		9.88 E−06		1.51 E−04	
32	5.84 E−07	4.08	1.07 E−05	3.81	5.84 E−07	4.08	1.07 E−05	3.81
64	3.63 E−08	4.00	7.05 E−07	3.92	3.63 E−08	4.00	7.05 E−07	3.92
128	2.26 E−09	4.00	4.46 E−08	3.94	2.26 E−09	4.00	4.46 E−08	3.94
256	1.41 E−10	4.00	2.95 E−09	3.97	1.41 E−10	4.00	2.95 E−09	3.97

limiter gives good performance on capturing the discontinuities without spurious oscillations.

The two dimensional Burgers equation is stated as:

$$\begin{cases} u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = 0, \\ u(x, y, 0) = \frac{1}{4} + \sin(x + y), \quad (x, y) \in [0, 2\pi] \times [0, 2\pi]. \end{cases} \tag{4.6}$$

The error and order of accuracy of the solution at $t = 0.1$ are in Table 5. Similar to the one-dimensional case, the MLP-limiter does not affect the accuracy. When the time reaches $t = 1.2$, there is a shock in the exact solution, and as we can see in Fig. 5, compared to the DG scheme without limiter, the MLP-limiter effectively controls the oscillation near the shock.

4.3 Euler Equation

In this subsection we apply the MLP limiter to solve nonlinear systems. We firstly consider the compressible Euler equation in one dimension:

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho\mu \\ E \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho\mu \\ \rho\mu^2 + p \\ \mu(E + p) \end{pmatrix} = 0, \quad 0 < x < 2\pi, \tag{4.7}$$

where ρ, μ, and p denote the density, velocity and pressure of the fluids, respectively. The total energy $E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho\mu^2$, with $\gamma = 1.4$ for air. For the system case, we choose to use the limiter in the local characteristic fields. That is, we firstly project the conserved variable $\boldsymbol{U} = (\rho, \rho\mu, E)^T$ into the local characteristic fields, and then apply the TVB or the MLP limiter in each characteristic field. Finally we project the

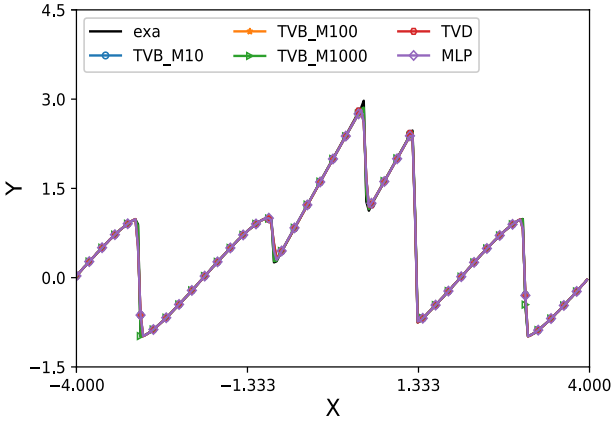


Fig. 4 Comparison of solutions on a randomly perturbed mesh for the compound wave problem using the fourth order DG method with the TVB limiter with $M = 0, 10, 100, 1000$ and the MLP limiter. Here $T = 0.4$ and cell of number $N = 200$

Table 5 Accuracy test for 2D Burgers equation

	$k=1$ DG MLP-limiter				$k=1$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	1.32 E−02		8.80 E−02		1.32 E−02		8.80 E−02	
32×32	3.40 E−03	1.95	2.26 E−02	1.96	3.40 E−03	1.95	2.26 E−02	1.96
64×64	8.67 E−04	1.97	5.73 E−03	1.98	8.67 E−04	1.97	5.73 E−03	1.98
128×128	2.18 E−04	1.99	1.43 E−03	1.99	2.18 E−04	1.99	1.43 E−03	1.99
256×256	5.47 E−05	2.00	3.60 E−04	1.99	5.47 E−05	2.00	3.60 E−04	1.99
	$k=2$ DG MLP-limiter				$k=2$ DG no limiter			
# cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	1.27 E−03		1.43 E−02		1.27 E−03		1.43 E−02	
32×32	1.61 E−04	2.98	1.72 E−03	3.06	1.61 E−04	2.98	1.72 E−03	3.06
64×64	4.48 E−05	1.84	6.24 E−04	1.85	2.04 E−05	3.00	2.17 E−04	3.01
128×128	2.48 E−06	4.17	2.92 E−05	7.73	2.48 E−06	3.00	2.92 E−05	3.01
256×256	3.11 E−07	3.00	3.69 E−06	2.98	3.11 E−07	3.00	3.96 E−06	3.00
	$k=3$ DG MLP-limiter				$k=3$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	9.53 E−05		1.06 E−04		9.53 E−05		1.06 E−04	
32×32	5.93 E−06	4.00	6.74 E−05	3.99	5.93 E−06	4.00	6.74 E−05	3.99
64×64	3.67 E−07	4.01	4.88 E−06	3.79	3.67 E−07	4.01	4.88 E−06	3.79
128×128	2.26 E−08	4.02	3.09 E−07	3.99	2.26 E−08	4.02	3.09 E−07	3.99
256×256	1.47 E−09	3.95	1.43 E−08	3.97	1.47 E−09	3.95	1.43 E−08	3.97

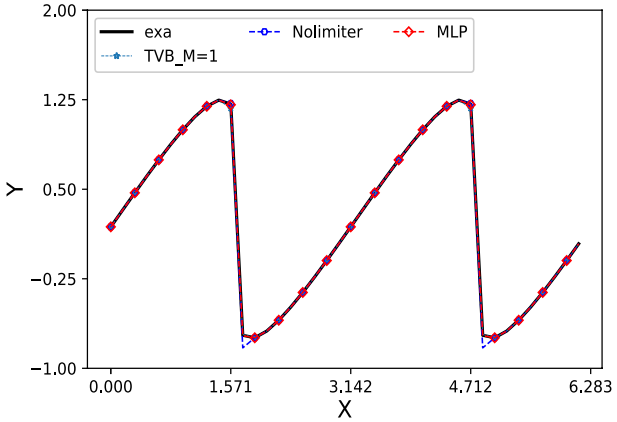


Fig. 5 Comparison of solutions of the 2D Burgers equation with the initial condition $u_0(x,y)=\frac{1}{4}+\sin(x+y)$ using the fourth order DG method without limiter, with the TVB limiter with $M=1$, and with the MLP limiter. Final time is $t=1.2$ and the number of cells corresponds to $N_x=N_y=40$

limited numerical solution back to the conserved variable space. More details can be found in [5]. We will compare the performance of the MLP-limiter with the TVB-limiter with *ad hoc* choices of the TVB constant M through trial and error as adopted in the literature. In all the test cases, we present the results for the density ρ as representations.

Example 4.3.1: Artificial accuracy test.

We firstly consider the accuracy test in [10]. We set the initial condition as:

$$\rho(x,0) = \frac{1+0.2\sin(x)}{2\sqrt{3}}, \quad \mu(x,0) = \sqrt{\gamma}\rho(x,0), \quad p(x,0) = \rho(x,0)^{\gamma}. \tag{4.8}$$

The computational domain is set to be $[0, 2\pi]$, and periodic boundary condition is imposed. We take $\gamma = 3$, which allows us to verify that $2\sqrt{3}\rho(x,t)$ is the exact solution of the Burgers equation:

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \qquad u(x,0) = 1 + 0.2\sin(x), \tag{4.9}$$

and

$$\mu(x,t) = \sqrt{\gamma}\rho(x,t), \qquad p(x,t) = \rho(x,t)^{\gamma}. \tag{4.10}$$

At $t = 0.3$, the solution is smooth, and the error and order of accuracy of density are listed in Table 6. It is clear that the MLP limiter does not affect the accuracy in this 1D nonlinear system example.

Example 4.3.2: The Sod problem.

This problem is a classic Riemann problem test, whose initial condition is

$$(\rho, \mu, p) = \begin{cases} (1,0,1), & x \le 0, \\ (0.125,0,0.1), & x > 0. \end{cases} \tag{4.11}$$

The domain is $x \in [-5, 5]$, and the simulation runs until $t = 2.0$ with the mesh size $N = 100$. We test the DG scheme with different orders of accuracy. If the TVB constant $M = 33$ or larger, the TVB limiter simulation fails with fourth or higher order DG schemes, due to the appearance of negative density. With $M = 33$, the TVB limiter gives good performance for the DG scheme with second and third order. On the other hand, while the solutions of TVB limiter with $M = 15$ smear a lot at discontinuities in lower order cases, it gives satisfying non-oscillatory result with fourth and fifth order DG schemes. Meanwhile, the MLP limiter gives good simulation in all cases, with results comparable to the $M = 33$ case in second and third order schemes, and to the $M = 15$ case in fourth and fifth order schemes. The details are shown in Fig. 6.

Example 4.3.3: The Lax problem.

Another famous Riemann problem test is the Lax problem, with the initial condition

Table 6 Accuracy test for 1D Euler equation

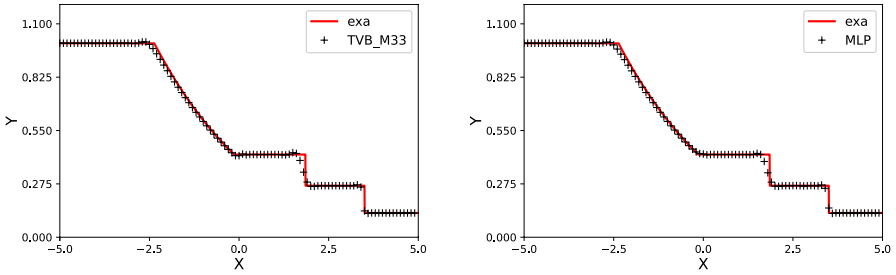
	$k = 1$ DG MLP-limiter				$k = 1$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16	4.96 E−03		8.86 E−03		4.96 E−03		8.86 E−03	
32	1.10 E−03	2.16	1.31 E−03	2.75	1.10 E−03	2.16	1.31 E−03	2.75
64	2.76 E−04	2.00	3.28 E−04	1.97	2.76 E−04	2.00	3.28 E−04	1.97
128	6.90 E−05	2.00	8.22 E−05	1.99	6.90 E−05	2.00	8.22 E−05	1.99
256	1.72 E−05	2.00	2.06 E−05	1.99	1.72 E−05	2.00	2.06 E−05	1.99
	$k = 2$ DG MLP-limiter				$k = 2$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16	1.93 E−04		3.76 E−04		1.93 E−04		3.76 E−04	
32	2.49 E−05	2.96	6.13 E−05	2.61	2.49 E−05	2.96	6.13 E−05	2.61
64	3.07 E−06	3.02	7.99 E−06	2.94	3.07 E−06	3.02	7.99 E−06	2.94
128	3.28 E−07	3.00	1.02 E−06	2.97	3.28 E−07	3.00	1.02 E−06	2.97
256	4.77 E−08	3.00	1.27 E−07	3.00	4.77 E−08	3.00	1.27 E−07	3.00
	$k = 3$ DG MLP-limiter				$k = 3$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16	7.27 E−06		1.21 E−05		7.27 E−06		1.21 E−05	
32	4.91 E−07	3.88	5.00 E−07	4.49	4.91 E−07	3.88	5.00 E−07	4.49
64	3.04 E−08	4.01	3.12 E−08	4.00	3.04 E−08	4.01	3.12 E−08	4.00
128	1.88 E−09	4.00	2.10 E−09	3.88	1.88 E−09	4.00	2.10 E−09	3.88
256	1.18 E−10	3.99	1.30 E−10	4.01	1.18 E−10	3.99	1.30 E−10	4.01

$$(\rho, \mu, p) = \begin{cases} (0.445, 0.698, 0, 3.528), & x \le 0, \\ (0.5, 0, 0, 0.571), & x > 0. \end{cases} \tag{4.12}$$

The domain is $x \in [-5, 5]$ and the number of cells is $N = 100$. We compute the solution until $t = 1.3$. In this case, we use $M = 33$ [5] which gives the best (sharpest) performance at discontinuities (especially at the contact discontinuity) for the third order DG scheme. Meanwhile, although the solution of the TVB limiter with $M = 70$ has huge oscillations at the discontinuity in lower order cases, it gives the best performance for the fifth order DG scheme. On the other hand, the MLP limiter works well for DG schemes with different orders of accuracy. The performance of the MLP limiter is as good as that of the $M = 33$ TVB limiter for the third order scheme, and of the $M = 70$ TVB limiter for the fifth order scheme. For the second order scheme, the MLP limiter describes the edge of the discontinuity better than that of the TVB limiters (Fig. 7).

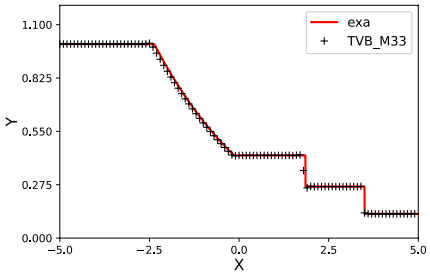
Example 4.3.4: The blast wave problem.

We now consider the interaction of two blast waves, with the initial condition:

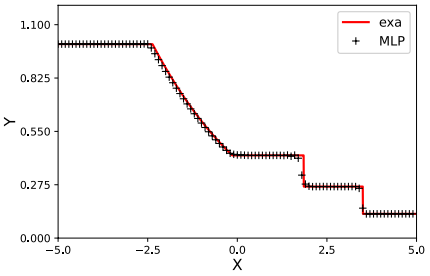


(a) second order TVB

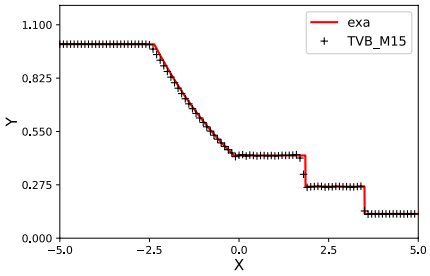
(b) second order MLP



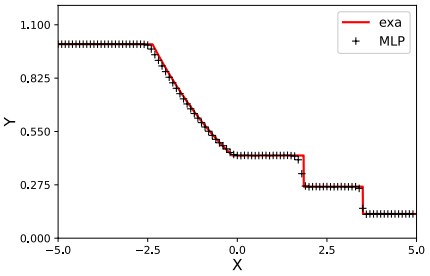
(c) third order TVB



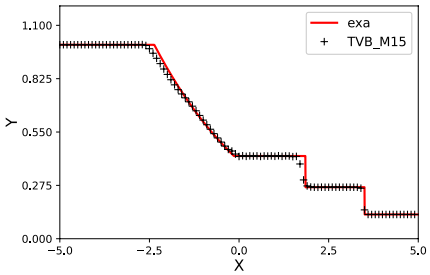
(d) third order MLP



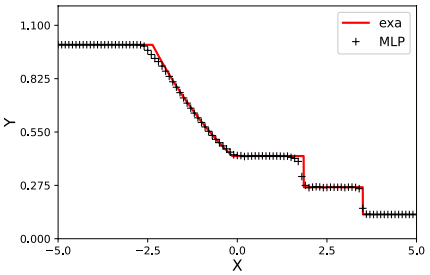
(e) fourth order TVB



(f) fourth order MLP



(g) fifth order TVB



(h) fifth order MLP

Fig. 6 Comparison of solutions for the Sod problem using the DG method of degree of freedom $k = 1, 2, 3, 4$ with the TVB limiter (left) and the MLP limiter (right). Final time $t = 2.0$ and the number of cells $N = 100$

$$(\rho, \mu, p) = \begin{cases} (1, 0, 1000), & 0 < x < 0.1, \\ (1, 0, 0.01), & 0.1 < x < 0.9, \\ (1, 0, 100), & 0.9 < x < 1. \end{cases} \tag{4.13}$$

The domain is $x \in [0, 1]$ and reflective boundary condition is applied. We present the numerical density of the TVB limiter DG method with the TVB constant $M = 33$ [5] and the MLP limiter DG method at the time $t = 0.038$ in Fig. 8. The solutions of the two methods are comparable.

Example 4.3.5: The Shu-Osher problem.

This example is introduced in [36], as a simple model for shock-turbulence interactions. Its initial condition is given by:

$$(\rho, \mu, p) = \begin{cases} (3.857143, 2.629369, 10.33333), & -5 \le x < -4, \\ (1 + 0.2\sin(5x), 0, 1), & -4 \le x \le 5, \end{cases} \tag{4.14}$$

The domain is $x \in [-5, 5]$. We present the numerical density of the TVB and the MLP limiter DG methods at the time $t = 0.038$ in Fig. 9. To achieve the best performance, the TVB constant is chosen as $M = 300$ [5] for $k = 1, 2, 3$ and $M = 550$ for $k = 4$. The overall performance are increased when higher order method are applied. The MLP model shows the performance similar to the TVB model at the (physically) high frequency wave area.

Now we consider the two-dimensional Euler equation:

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho\mu \\ \rho\nu \\ E \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho\mu \\ \rho\mu^2 + p \\ \rho\mu\nu \\ \mu(E+p) \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} \rho\nu \\ \rho\mu\nu \\ \rho\nu^2 + p \\ \nu(E+p) \end{pmatrix} = 0, \tag{4.15}$$

where ρ is the density, μ and ν are the velocities in the x and y directions, respectively, and p is the fluid pressure. The total energy $E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(\mu^2 + \nu^2)$, with $\gamma = 1.4$ for air.

Example 4.3.6: Artificial accuracy test for the 2D Euler equation.

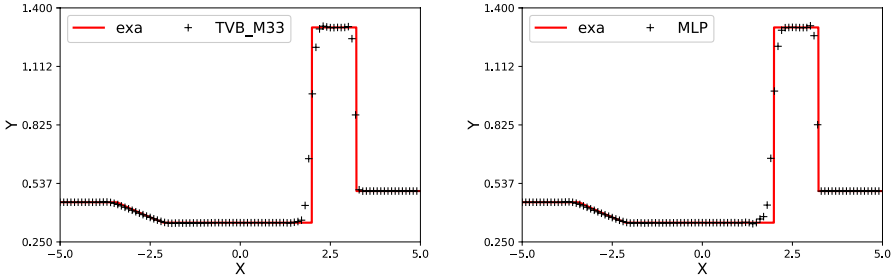
We conduct an accuracy test for the 2D Euler equation. The initial condition is:

$$\rho(x, y, 0) = \frac{1 + 0.2\sin(\frac{x+y}{2})}{\sqrt{6}}, \quad \mu(x, y, 0) = \nu(x, y, 0) = \sqrt{\frac{\gamma}{2}}\rho(x, y, 0), \quad p(x, y, 0) = \rho(x, y, 0)^{\gamma}. \tag{4.16}$$

The computational domain is $[0, 4\pi] \times [0, 4\pi]$. We set $\gamma = 3$, and it could be easily verified that $\sqrt{6}\rho(x, y, t)$ is the exact solution of the Burgers equation:

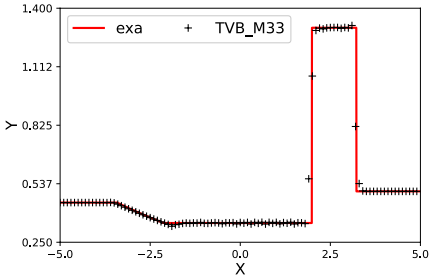
$$u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = 0, \qquad u(x, y, 0) = 1 + 0.2\sin(\frac{x+y}{2}), \tag{4.17}$$

and μ, ν and p satisfy:

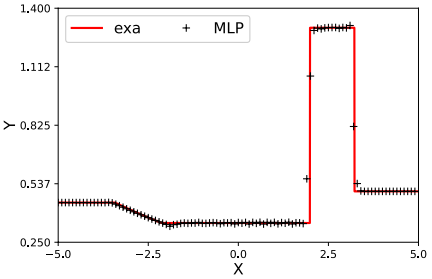


(a) second order TVB

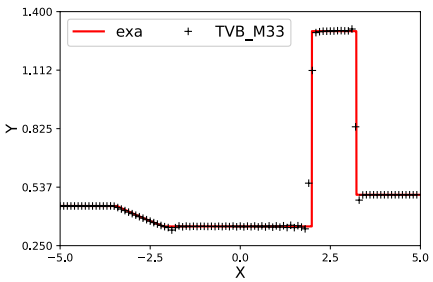
(b) second order MLP



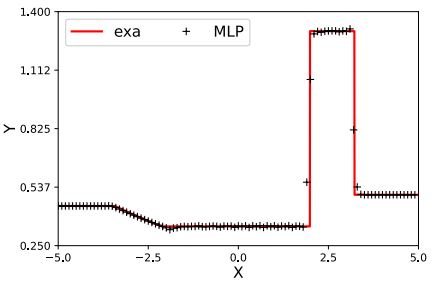
(c) third order TVB



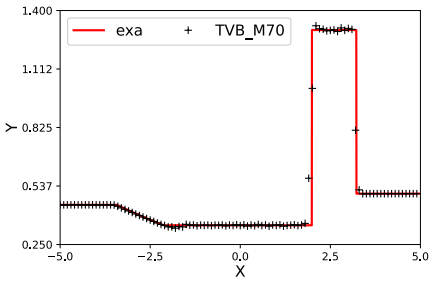
(d) third order MLP



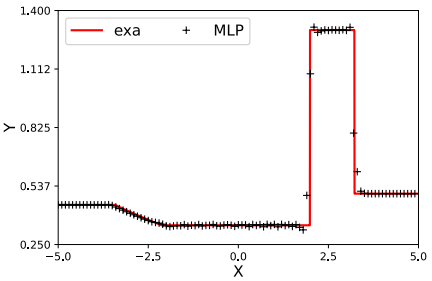
(e) fourth order TVB



(f) fourth order MLP

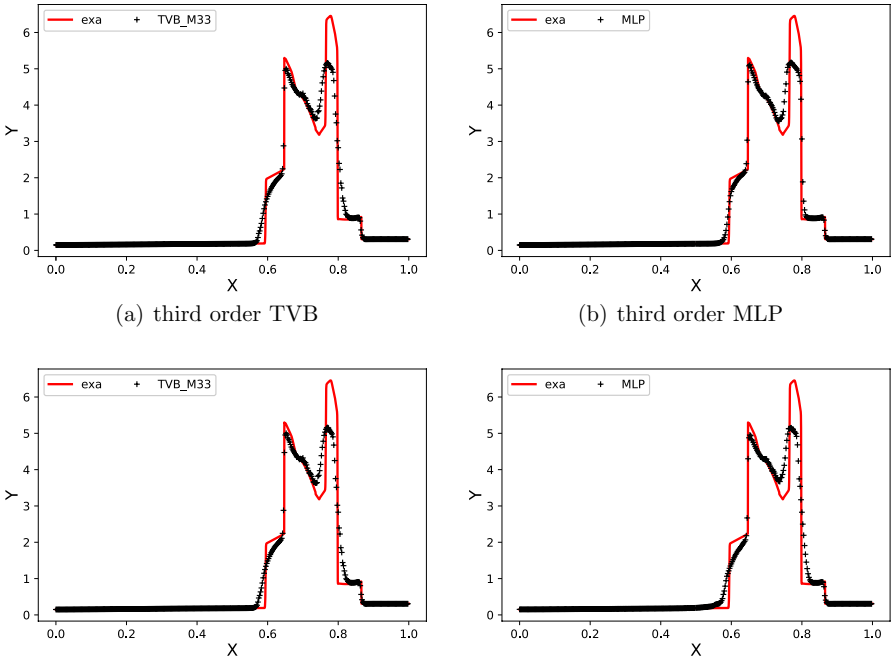


(g) fifth order TVB



(h) fifth order MLP

Fig. 7 Comparison of solutions for the Lax problem using the DG method of degree of freedom $k = 1, 2, 3, 4$ with the TVB limiter (left) and the MLP limiter (right). Final time $t = 1.3$ and the number of cells $N = 100$



(a) third order TVB (b) third order MLP

(c) fifth order TVB (d) fifth order MLP

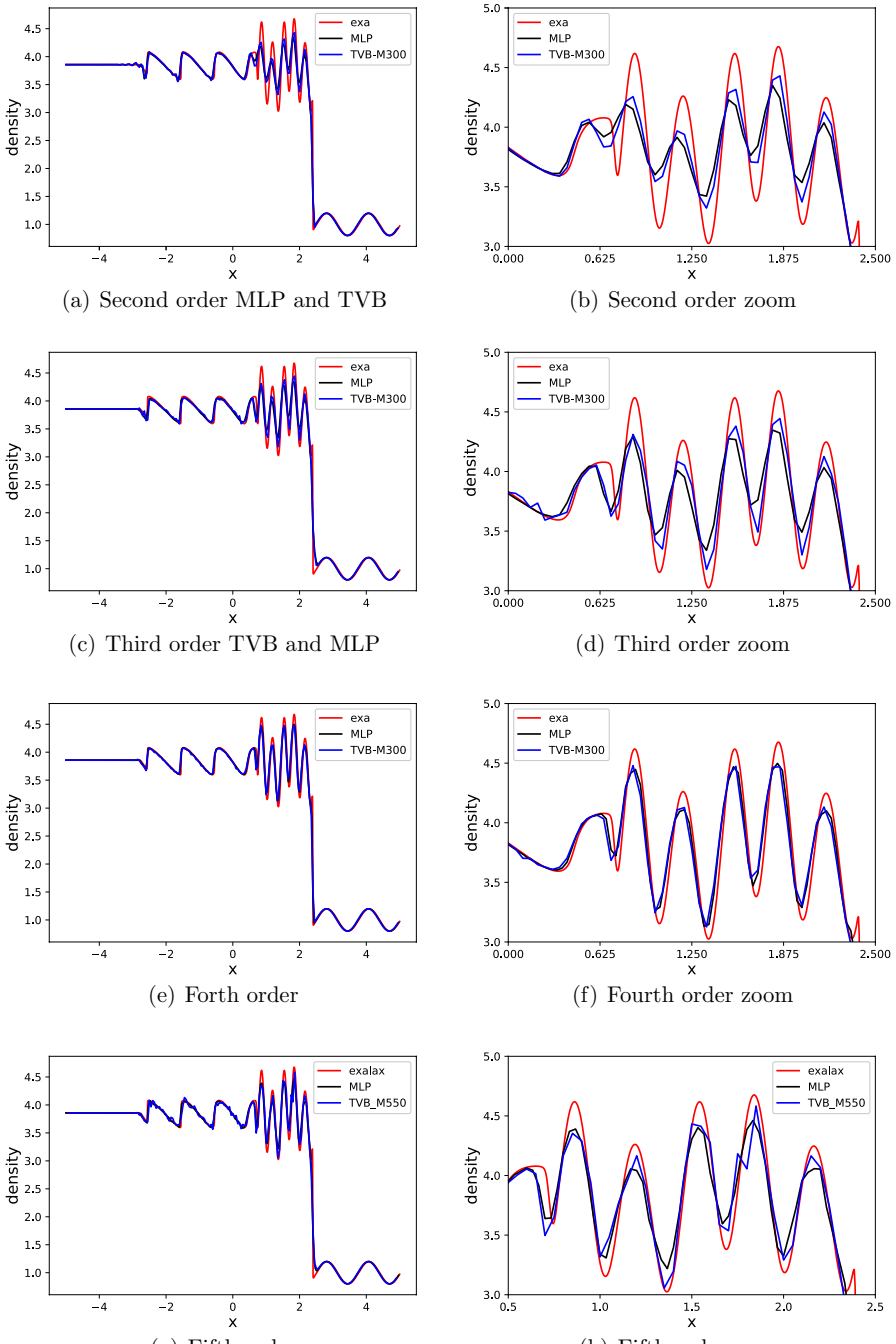
Fig. 8 Solution of the blast wave problem using the third order and fifth order DG schemes with the $M = 33$ TVB limiter (left), and the MLP limiter (right). Final time $T = 0.038$ and the number of cells $N = 400$

$$\mu(x, y, t) = \mu(x, y, t) = \sqrt{\frac{\gamma}{2}}\rho(x, y, t), \qquad p(x, y, t) = \rho(x, y, t)^{\gamma}. \tag{4.18}$$

At $t = 0.3$, the solution is smooth. The error and order of accuracy of density are shown in Table 7. It can be observed that the MLP limiter does not affect the high order accuracy of the scheme for this 2D nonlinear system test case. In Table 8, the cpu time of the simulations on an 100×100 mesh is analyzed and reported. The simulations have been run on Jupyter Notebook using a 2 GHz Quad-Core Intel Core i5 processor. The execution time of a single timestep (Tsp) increases when a higher order scheme is used. It can be observed that the gap between the cost of the TVB and the MLP limiter narrows when k increases. When $k = 3, 4$ the additional cost of applying the MLP model in the TVB DG scheme is negligible.

Example 4.3.7: The double Mach reflection problem.

This problem was introduced by Woodward and Colella [41]. We use the same setup as in [41], which describes a Mach 10 shock moving right into the undisturbed air, making a 60° angle with a reflecting wall. The density and pressure of the undisturbed air are 1.4 and 1 respectively. The computational domain is $[0, 4] \times [0, 1]$. We use the exact flow values of the Mach 10 shock at each time step as the top boundary condition. For the bottom boundary, we apply the post-shock condition



(a) Second order MLP and TVB

(b) Second order zoom

(c) Third order TVB and MLP

(d) Third order zoom

(e) Forth order

(f) Fourth order zoom

(g) Fifth order

(h) Fifth order zoom

Fig. 9 Numerical solution of the Shu–Osher problem (left). Zoomed region close to the high frequency fluctuation area (right). Final time $T = 1.8$ and the number of cells $N = 200$

Table 7 2D Euler equation accuracy test

	$k=1$ DG MLP-limiter				$k=1$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	1.00 E−3		7.24 E−02		1.00 E−3		7.24 E−02	
32×32	2.52 E−04	1.99	1.94 E−03	1.90	2.52 E−04	1.99	1.94 E−03	1.90
64×64	6.37 E−05	1.99	1.59 E−04	1.96	6.37 E−05	1.99	1.59 E−04	1.96
128×128	1.59 E−05	2.00	1.25 E−04	1.99	1.59 E−05	2.00	1.25 E−04	1.99
256×256	3.98 E−06	2.00	3.14 E−05	1.99	3.98 E−06	2.00	3.14 E−05	1.99
	$k=2$ DG MLP-limiter				$k=2$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	1.17 E−04		4.96 E−04		1.27 E−03		1.43 E−02	
32×32	1.45 E−05	3.00	1.61 E−04	2.98	1.72 E−03	2.98	6.28 E−05	3.06
64×64	1.82 E−06	3.00	2.04 E−05	3.00	2.17 E−04	3.01	7.87 E−06	2.98
128×128	2.28 E−07	3.00	2.48 E−06	3.00	2.92 E−05	3.01	9.85 E−07	2.90
256×256	2.84 E−08	3.00	3.11 E−07	3.00	3.96 E−06	3.00	1.23 E−07	3.00
	$k=3$ DG MLP-limiter				$k=3$ DG no limiter			
# Cells	L^1 error	Order	L^∞ error	Order	L^1 error	Order	L^∞ error	Order
16×16	9.53 E−05		1.06 E−04		9.53 E−05		1.06 E−04	
32×32	5.93 E−06	4.00	6.74 E−05	3.99	5.93 E−06	4.00	6.74 E−05	3.99
64×64	3.67 E−07	4.01	4.88 E−06	3.79	3.67 E−07	4.01	4.88 E−06	3.79
128×128	2.26 E−08	4.02	3.09 E−07	3.99	2.26 E−08	4.02	3.09 E−07	3.99
256×256	1.47 E−09	3.95	1.43 E−08	3.97	1.47 E−09	3.95	1.43 E−08	3.97

Table 8 Computational times, number of timesteps and execution time of a single timestep (TpS) for the 2D Euler problem. The total time and the time per timestep are expressed in seconds

	k=1			k=2			k=3			k=4		
Limiters	time	Steps	Tps	time	Steps	Tps	time	Steps	Tps	time	Steps	Tps
TVB	26.99	29	0.93	61.76	47	1.31	132.12	66	2.00	308.14	85	3.62
MLP	39.73	29	1.37	74.91	47	1.59	137.75	66	2.08	317.06	85	3.72

for $x\in[0,\frac{1}{6}]$, and reflecting wall condition for $x\in[\frac{1}{6},4]$. The numerical simulation is generated up to $t=0.2$. The simulations on uniformed meshes with 480×120 and 960×240 cells are shown in Figs. 10 and 12 , with the zoomed version near the Mach stem shown in Figs. 11 and 13 . For the TVB limiter, the TVB constant is chosen as $M=50$ for the second and third order DG schemes [7]. Compared to the traditional TVB limiter with empirically chosen M through trial and error, the MLP limiter provides equally satisfying results.

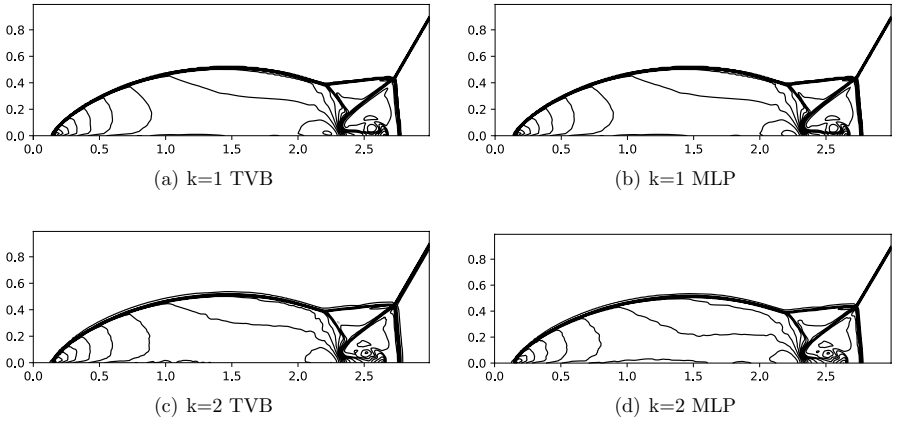
5 Concluding Remarks

In this paper, we design a MLP based TVB limiter for solving hyperbolic conservation laws in one and two dimensional scalar and system cases using DG schemes. Numerical results are shown on structured meshes.

In comparison with the classical minmod-based TVB limiter with an empirically chosen TVB constant M, the advantages of the new MLP based TVB limiter are as follows:

1. The MLP limiter is able to control spurious oscillations near discontinuities without excessive smearing, while maintaining the original high order accuracy in smooth regions including near smooth extrema.
2. The MLP procedure automates the choice of the TVB constant M, thus eliminates the need to choose M in an *ad hoc* fashion. This is especially important for hyperbolic systems, for which no rigorous mathematical guidance exists for the choice of M.
3. The model training can be performed offline, leaving the online computation efficient involving only a few low-cost matrix multiplications. Thus it is simple to modify the standard DG code to apply the new limiter, and the extra coding only involves a few lines.
4. The MLP based TVB limiter works well for the DG scheme of various orders of accuracy, and give the same or even better performance than the classical TVB limiter with manually chosen TVB constant M through trial and error, for an extensive list of numerical test problems in 1D and 2D.

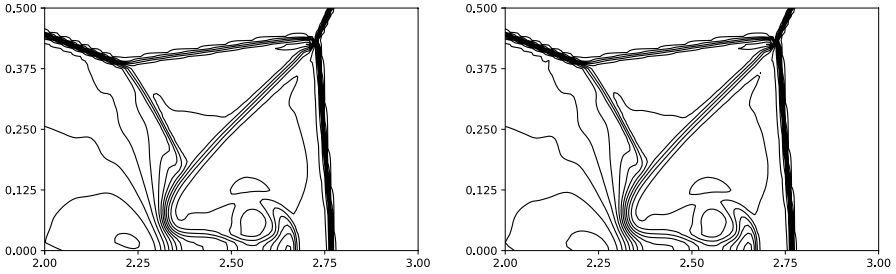
The methodology should work equally well for multi-dimensional unstructured meshes, which consists of our ongoing work.



(a) k=1 TVB (b) k=1 MLP

(c) k=2 TVB (d) k=2 MLP

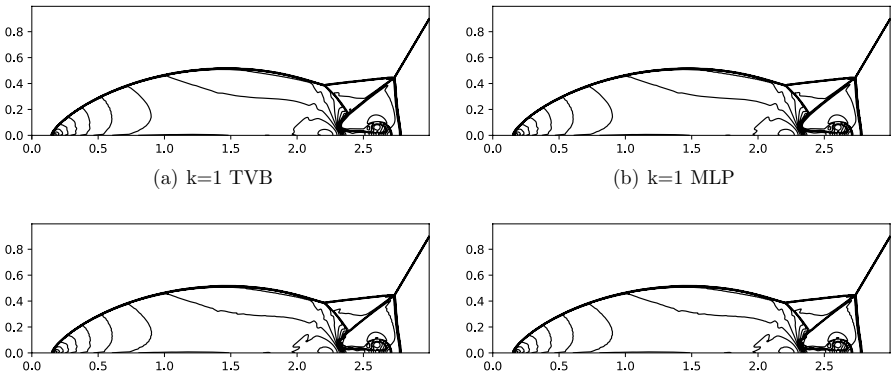
Fig. 10 Double Mach reflection problem. DG method with $k = 1, 2$. Left: results with the TVB limiter. Right: results with the MLP limiter. Density ρ. 30 equally spaced contour lines from $\rho = 1.5$ to $\rho = 22.7$. Mesh grid: 480×120



(a) k=1 TVB (b) k=1 MLP

(c) k=2 TVB (d) k=2 MLP

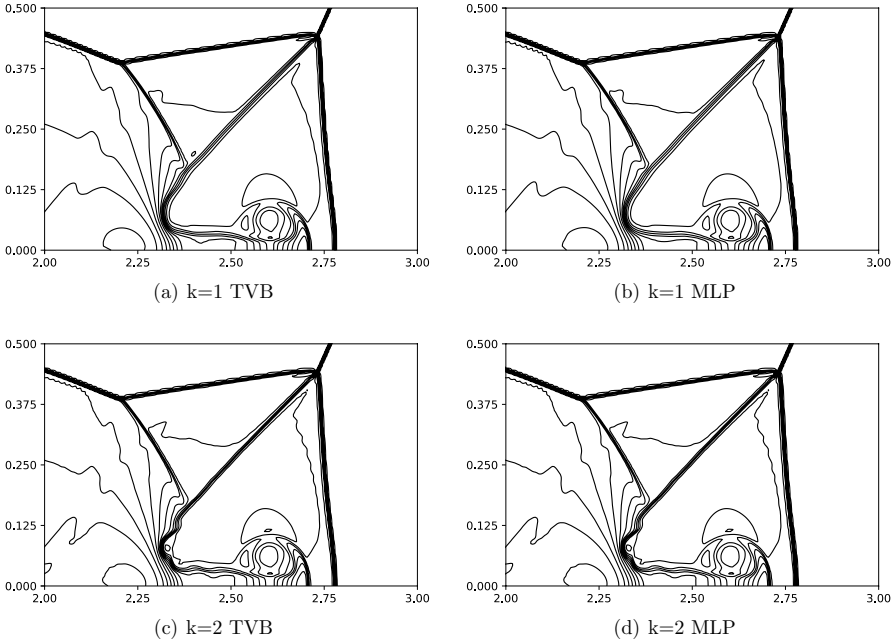
Fig. 11 Double Mach reflection problem. DG method with $k = 1, 2$. Blown-up region around the double Mach stem. Left: results with the TVB limiter. Right: results with the MLP limiter. Density ρ. 30 equally spaced contour lines from $\rho = 1.5$ to $\rho = 22.7$. Mesh grid: 480×120



(a) k=1 TVB (b) k=1 MLP

(c) k=2 TVB (d) k=2 MLP

Fig. 12 Double Mach reflection problem. DG method with $k = 1, 2, 3$. Left: results with the TVB limiter. Right: results with the MLP limiter. Mesh grid: 960×240



(a) k=1 TVB (b) k=1 MLP

(c) k=2 TVB (d) k=2 MLP

Fig. 13 Double Mach reflection problem. DG method with $k = 1, 2, 3$. Blown-up region around the double Mach stem. Left: results with the TVB limiter. Right: results with the MLP limiter. Density ρ. 30 equally spaced contour lines from $\rho = 1.5$ to $\rho = 22.7$. Mesh grid: 960×240

Acknowledgements Research supported by NSF Grant DMS-2010107 and AFOSR Grant FA9550-20-1-0055.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Biswas, R., Devine, K., Flaherty, J.: Parallel, adaptive finite element methods for conservation laws. Appl. Numer. Math. **14**, 255–283 (1994)
2. Chen, T., Shu, C.-W.: Entropy stable high order discontinuous Galerkin methods with suitable quadrature rules for hyperbolic conservation laws. J. Comput. Phys. **345**, 427–461 (2017)
3. Chen, T., Shu, C.-W.: Review of entropy stable discontinuous Galerkin methods for systems of conservation laws on unstructured simplex meshes. CSIAM Trans. Appl. Math. (CSAM) **1**, 1–52 (2020)
4. Cockburn, B., Hou, S., Shu, C.-W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case. Math. Comput. **54**, 545–581 (1990)
5. Cockburn, B., Lin, S.-Y., Shu, C.-W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems. J. Comput. Phys. **84**, 90–113 (1989)

6. Cockburn, B., Shu, C.-W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework. Math. Comput. **52**, 411–435 (1989)
7. Cockburn, B., Shu, C.-W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation law V: multidimensional systems. J. Comput. Phys. **141**, 199–224 (1998)
8. Cockburn, B., Shu, C.-W.: Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. J. Sci. Comput. **16**, 173–261 (2001)
9. Cybenko, G.: Continuous valued neural networks with two hidden layers are sufficient, Technical Report, Department of Computer Science, Tufts University, Medford, MA (1988)
10. Fu, G., Shu, C.-W.: A new troubled-cell indicator for discontinuous Galerkin methods for hyperbolic conservation laws. J. Comput. Phys. **347**, 305–327 (2017)
11. Gao, Z., Wen, X., Don, W.S.: Enhanced robustness of the hybrid compact-WENO finite difference scheme for hyperbolic conservation laws with multi-resolution analysis and Tukey's boxplot method. J. Comput. Phys. **73**, 736–752 (2017)
12. Golak, S.: A MLP solver for first and second order partial differential equations. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D. (eds.) Artificial Neural Networks-ICANN 2007, pp. 789–797. Springer, Berlin (2007)
13. Guliyev, N.J., Ismailov, V.E.: A single hidden layer feedforward network with only one neuron in the hidden layer can approximate any univariate function. Neural Comput. **28**, 1289–1304 (2016)
14. Harten, A.: High resolution schemes for hyperbolic conservation laws. J. Comput. Phys. **49**, 357–393 (1983)
15. Hou, S., Liu, X.-D.: Solutions of multi-dimensional hyperbolic systems of conservation laws by square entropy condition satisfying discontinuous Galerkin method. J. Sci. Comput. **31**, 127–151 (2007)
16. Jiang, G.-S., Shu, C.-W.: On cell entropy inequality for discontinuous Galerkin methods. Math. Comput. **62**, 531–538 (1994)
17. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
18. Kriesel, D.: A brief introduction to neural networks. http://www.dkriesel.com (2007)
19. Kontzialis, K., Panourgias, K., Ekaterinaris, J.: A limiting approach for DG discretizations on mixed type meshes. Comput. Methods Appl. Mech. Eng. **285**, 587–620 (2015)
20. Krivodonova, L., Xin, J., Remacle, J.-F., Chevaugeon, N., Flaherty, J.E.: Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws. Appl. Numer. Math. **48**, 323–338 (2004)
21. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. **9**, 987–1000 (1998)
22. Lee, H., Lee, N.: Wet-dry moving boundary treatment for Runge-Kutta discontinuous Galerkin shallow water equation model. KSCE J. Civ. Eng. **20**, 978–989 (2016)
23. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings International Conference on Machine Learning, p. 30 (2013)
24. Novikoff, A.B.: On convergence proofs on perceptrons. Sympos. Math. Theory Autom. **12**, 615–622 (1962)
25. Osher, S.: Convergence of generalized MUSCL schemes. SIAM J. Numer. Anal. **22**, 947–961 (1985)
26. Osher, S., Chakravarthy, S.: High resolution schemes and the entropy condition. SIAM J. Numer. Anal. **21**, 955–984 (1984)
27. Panourgias, K.T., Ekaterinaris, J.A.: A discontinuous Galerkin approach for high-resolution simulations of three-dimensional flows. Comput. Methods Appl. Mech. Eng. **299**, 245–282 (2016)
28. Qiu, J., Shu, C.-W.: Runge-Kutta discontinuous Galerkin method using WENO limiters. SIAM J. Sci. Comput. **26**, 907–929 (2005)
29. Ray, D., Hesthaven, J.S.: An artificial neural network as a troubled-cell indicator. J. Comput. Phys. **367**, 166–191 (2018)
30. Ray, D., Hesthaven, J.S.: Detecting troubled-cells on two-dimensional unstructured grids using a neural network. J. Comput. Phys. **397**, 108–845 (2019)
31. Reed, W., Hill, T.: Triangular mesh methods for neutron transport equation, Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM (1973)
32. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**, 386–408 (1958)

33. Rudd, K., Ferrari, S.: A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. Neurocomputing **155**, 277–285 (2015)
34. Shu, C.-W.: TVB uniformly high-order schemes for conservation laws. Math. Comput. **49**, 105–121 (1987)
35. Shu, C.-W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. J. Comput. Phys. **77**, 439–471 (1988)
36. Shu, C.-W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes II. J. Comput. Phys. **83**, 32–78 (1989)
37. Sun, Z., Wang, S., Chang, L.-B., Xing, Y., Xiu, D.: Convolution neural network shock detector for numerical solution of conservation laws. Commun. Comput. Phys. **28**, 2075–2108 (2020)
38. Suresh, A., Huynth, H.: Accurate monotonicity-preserving schemes with Runge-Kutta time stepping. Comput. Fluid Dyn. Conf. **13**, 83–99 (1997)
39. Vuik, M.J., Ryan, J.K.: Automated parameters for troubled-cell indicators using outlier detection. SIAM J. Sci. Comput. **38**, A84–A104 (2016)
40. Wen, X., Don, W.S., Gao, Z., Hesthaven, J.S.: An edge detector based on artificial neural network with application to hybrid compact-WENO finite difference scheme. J. Sci. Comput. **83**, 1–1 (2020)
41. Woodward, P., Colella, P.: The numerical simulation of two-dimensional fluid flow with strong shocks. J. Comput. Phys. **54**, 115–173 (1984)
42. Xing, Y., Zhang, X.: Positivity-preserving well-balanced discontinuous Galerkin methods for the shallow water equations on unstructured triangular meshes. J. Comput. Phys. **57**, 19–41 (2013)
43. Zhao, J., Tang, H.: Runge-Kutta central discontinuous Galerkin methods for the special relativistic hydrodynamics. Commun. Comput. Phys. **22**, 643–682 (2017)
44. Zhu, H., Cheng, Y., Qiu, J.: A comparison of the performance of limiters for Runge-Kutta discontinuous Galerkin methods. Adv. Appl. Math. Mech. **5**, 365–390 (2013)

Authors and Affiliations

Xinyue Yu[1] · Chi-Wang Shu[1]

Xinyue Yu
xinyue_yu@brown.edu

[1] Division of Applied Mathematics, Brown University, Providence, RI 02912, USA