Detecting troubled-cells on two-dimensional unstructured grids using a neural network

Deep Ray, Jan S. Hesthaven

École Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
deep.ray@epfl.ch, jan.hesthaven@epfl.ch

Abstract

In a recent paper [Ray and Hesthaven, J. Comput. Phys. 367 (2018), pp 166-191], we proposed a new type of troubled-cell indicator to detect discontinuities in the numerical solutions of one-dimensional conservation laws. This was achieved by suitably training an artificial neural network on canonical local solution structures for conservation laws. The proposed indicator was independent of problem-dependent parameters, giving it an advantage over existing limiter-based indicators. In the present paper, we extend this approach to train a similar network capable of detecting troubled-cells on two-dimensional unstructured grids. The proposed network has a smaller architecture compared to its one-dimensional predecessor, making it computationally efficient. Several numerical results are presented to demonstrate the performance of the new indicator.

1. Introduction

Solutions of (non-linear) hyperbolic conservation laws are known to be inherently discontinuous [12]. As a result, the approximate solution obtained with high-order numerical methods typically suffers from Gibbs oscillations, which can lead to numerical instabilities. Several strategies have been proposed during the past few decades to control these spurious oscillations. In the context of Runge-Kutta discontinuous Galerkin (RKDG) schemes [6, 10, 20], a popular method involves limiting the numerical solution in a post-processing step after each Runge-Kutta substage, using slope-limiters [8, 7, 9, 5], moment limiters [4, 24] or WENO-based limiters [29, 31, 32, 1]. Limiting the solution in every element of the mesh deteriorates the accuracy in smooth regions and increases the computational cost.

In order to localize the limiting to the relevant regions of the mesh, a two step strategy was proposed in [32]: i) detect all the troubled-cells in the mesh, i.e., the cells in which the numerical solution loses regularity; ii) limit the local polynomial only in the flagged cells. In [30], a numerical comparison of several limiter-based troubled-cell detectors was performed. The minmod-type TVB limiter [8], the KXRCF detector [23] and the indicator based on Harten's subcell resolution idea [18] were shown to perform better than the remaining detectors tested, although none of the three were clearly superior. In [14], an a posteriori subcell-based limiter was proposed, flagging troubled-cells based on physical constraints such as positivity or discrete maximum principles. A new and simple troubled-cell indicator was proposed in [15], and shown to perform well for the Euler equations.

A major drawback of most existing troubled-cell indicators is that they require the specification of problem-dependent parameters, which are usually determined empirically. A poor choice of these

parameters can either lead to the reappearance of Gibbs oscillations, or loss of accuracy in regions where the solution is smooth. Recently, an automatic parameter selection strategy was proposed in [36], which detects outliers using Tukey's boxplot method. While this is a promising attempt at removing the dependence on parameters, it has been demonstrated to perform well only on uniform grids. In [34], a new approach for constructing troubled-cell indicators was proposed using machine learning ideas. In particular, a deep fully-connected feedforward network (or multilayer perceptron) was trained offline to flag discontinuities in the mesh. The trained network is independent of problem-dependent parameters, relatively inexpensive to use, and demonstrated to perform well on a variety of test cases for one-dimensional scalar and systems of conservation laws on quasi-uniform grids.

In this work, we extend the machine learning techniques of [34] to train a suitable network to work as a troubled-cell indicator for two-dimensional conservation laws on unstructured triangular grids. To make the method cost effective on two-dimensional grids, the size of the network is drastically reduced as compared to that used in [34]. The results obtained with the network are compared to those obtained with the TVB indicator. We use the Barth-Jesperson limiter [2] to restrict the polynomials in the flagged cells.

The rest of the paper is structured as follows. In Section 2 we introduce the two-dimensional RKDG formulation. The algorithms for the TVB indicator and Barth-Jesperson limiter are outlined in Section 3. After a brief discussion of the architecture and training strategy of multilayer perceptrons in Section 4, we detail the construction and training of a network trained as a two-dimensional troubled-cell indicator in Section 5. Several numerical results are presented in Section 6 to demonstrate the performance of the proposed network. Concluding remarks are made in Section 7.

2. Discontinuous-Galerkin formulation

Let $\Omega \in \mathbb{R}^2$ be a bounded domain. Consider the following two-dimensional hyperbolic conservation law

$$\begin{aligned} \frac{\partial \boldsymbol{u}(\boldsymbol{x},t)}{\partial t} + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}(\boldsymbol{x},t)) = 0 \qquad & \forall\ \boldsymbol{x} \in \Omega,\ t \in [0,T], \\ \boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u}_0(\boldsymbol{x}) \qquad & \forall\ \boldsymbol{x} \in \Omega, \end{aligned} \tag{2.1}$$

where $\boldsymbol{u} \in \mathbb{R}^d$ is the vector of conserved variables with a smooth flux $\boldsymbol{f} = (\boldsymbol{f}_x, \boldsymbol{f}_y) \in \mathbb{R}^d \times \mathbb{R}^d$. Although boundary conditions must also be prescribed to complete the description of (2.1), we ignore them for the time-being. To approximate the solution of (2.1), the computational domain is discretized using non-overlapping triangles $\Omega = \bigcup_{k=1}^{K} T^k$. For simplicity, we restrict the discussion to scalar conservation laws, i.e., $d = 1$, with the extension to systems being straight-forward. We define the space of broken polynomials $\mathcal{V}_p^h = \{v \in L^2(\Omega) : v\big|_{T^k} \in \mathcal{P}^p(T^k)\}$, where $\mathcal{P}^p(T)$ is the space of two-dimensional polynomials of order p, on the element T. The dimension of the local polynomial space $\mathcal{P}^p(T)$ is $N_p = (p+1)(p+2)/2$ [20]. Thus, the solution $u(\boldsymbol{x},t)$ is approximated as

$$u(\boldsymbol{x},t) \approx u_h(\boldsymbol{x},t) = \bigoplus_{k=1}^{K} u_h^k(\boldsymbol{x},t).$$

On each triangle T^k, the *nodal* representation of the local solution $u_h^k(\boldsymbol{x},t)$ is given by

$$u_h^k(\boldsymbol{x},t)=\sum_{i=1}^{N_p} u_h^k(\boldsymbol{x}_i^k,t)l_i^k(\boldsymbol{x}) \quad \forall\ \boldsymbol{x}\in T^k, \tag{2.2}$$

where $\{l_i^k(\boldsymbol{x}) : i=1,...,N_p\}$ are the multidimensional Lagrange polynomials defined by some suitable grid-points $\boldsymbol{x}_i^k$ in T^k. Similarly, the local nodal approximation of the flux function $\boldsymbol{f}$ is expressed as

$$\boldsymbol{f}_h^k(\boldsymbol{x},t)=\sum_{i=1}^{N_p} \boldsymbol{f}\big(u_h^k(\boldsymbol{x}_i^k,t)\big)l_i^k(\boldsymbol{x}) \quad \forall\ \boldsymbol{x}\in T^k. \tag{2.3}$$

In practice, the general triangle T is mapped to a standard reference triangle $\widehat{T}=\{\boldsymbol{r}=(r,s) : r,s\geqslant -1;\ r+s\leqslant 0\}$, through the affine map

$$\boldsymbol{x}(\boldsymbol{r})=-\frac{r+s}{2}\boldsymbol{v}_1+\frac{r+1}{2}\boldsymbol{v}_2+\frac{s+1}{2}\boldsymbol{v}_3 \tag{2.4}$$

where $\boldsymbol{v}_1,\boldsymbol{v}_2,\boldsymbol{v}_3$ are the vertices of triangle T (see Figure 1). Using (2.4), we have

$$\frac{\partial \boldsymbol{x}}{\partial r}=\frac{\boldsymbol{v}_2-\boldsymbol{v}_1}{2}, \quad \frac{\partial \boldsymbol{x}}{\partial s}=\frac{\boldsymbol{v}_3-\boldsymbol{v}_1}{2}, \quad J=\frac{\partial x}{\partial r}\frac{\partial y}{\partial s}-\frac{\partial x}{\partial s}\frac{\partial y}{\partial r}, \tag{2.5}$$

and the transformation Jacobian J is a constant. Using (2.4), we consider the nodal representation on the reference triangle $\widehat{T}$

$$u_h(\boldsymbol{r},t)=\sum_{i=1}^{N_p} u_h(\boldsymbol{r}_i,t)l_i(\boldsymbol{r})=\sum_{i=1}^{N_p} u_i(t)l_i(\boldsymbol{r}) \quad \forall\ \boldsymbol{r}\in \widehat{T}, \tag{2.6}$$

where $\boldsymbol{x}(\boldsymbol{r}_i)=\boldsymbol{x}_i^k$, while $\{l_i(\boldsymbol{r}) : i=1,...,N_p\}$ are the two-dimensional Lagrange polynomials on $\widehat{T}$. In order to ensure well-behaved interpolations, the interpolating points $\boldsymbol{r}_i$ are chosen as the α-*optimal warp and blend* nodes [37, 20]. These nodes reduce to $p+1$ one-dimensional Legendre-Gauss-Lobatto nodes along each edge of the triangle.

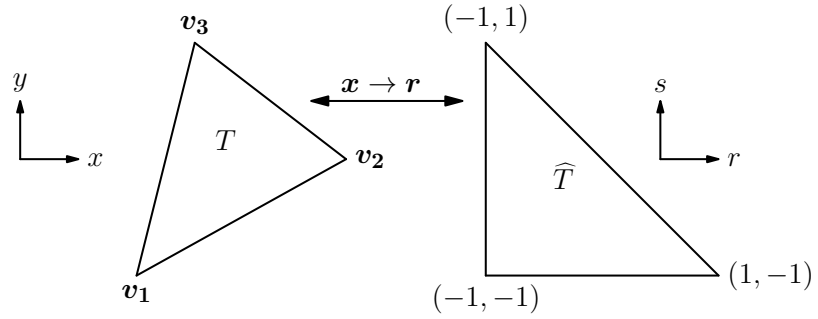


Figure 1: Affine mapping between a general triangle T and the reference triangle $\widehat{T}$.

Unlike the one-dimensional scenario, explicit expressions are unavailable for Lagrange polynomials in higher-dimensions on general nodes. Thus, it becomes useful to recast (2.6) into a *modal* representation

$$u_h(\boldsymbol{r},t) = \sum_{i=1}^{N_p} u_i(t) l_i(\boldsymbol{r}) = \sum_{j=1}^{N_p} \hat{u}_j(t) \psi_j(\boldsymbol{r}) \quad \forall\ \boldsymbol{r} \in \widehat{T},$$

where $\{\psi_j(\boldsymbol{r}) : j = 1, ..., N_p\}$ is a suitable two-dimensional polynomial basis of order p on $\widehat{T}$. Note that $\hat{u}_j$ are not the nodal values and are referred to as the modal coefficients. A commonly used choice for ψ_j is given by the orthonormal basis [20]

$$\psi_m(\boldsymbol{r}) = \sqrt{2} Q_i^{(0,0)}(a) Q_j^{(2i+1,0)}(b)(1-b)^i, \quad a = 2\frac{1+r}{1-s} - 1, \quad b = s,$$

where $Q_n^{(\alpha,\beta)}$ is the n-th order Jacobi polynomial and

$$m = j + (N+1)i + 1 - \frac{i}{2}(i-1), \quad i, j \geqslant 0, \quad i + j \leqslant N. \tag{2.7}$$

Using the generalized Vandermonde matrix $\boldsymbol{V}_{ij} = \psi_j(\boldsymbol{r}_i)$, the nodal values can be obtained from the modal coefficients using the relation

$$\boldsymbol{U} = \boldsymbol{V}\hat{\boldsymbol{U}}, \quad \boldsymbol{U} = \left[u_1, ..., u_{N_p}\right]^\top, \quad \hat{\boldsymbol{U}} = \left[\hat{u}_1, ..., \hat{u}_{N_p}\right]^\top.$$

Furthermore, we have the following relation connecting the Lagrange polynomials to the basis $\psi_i(\boldsymbol{r})$

$$\boldsymbol{V}^\top \boldsymbol{l}(\boldsymbol{r}) = \boldsymbol{\psi}(\boldsymbol{r}), \quad \boldsymbol{l} = \left[l_1, ..., l_{N_p}\right]^\top, \quad \boldsymbol{\psi} = \left[\psi_1, ..., \psi_{N_p}\right]^\top.$$

Substituting (2.2) and (2.3) in (2.1) yields an expression for the local residual

$$R_h^k(\boldsymbol{x},t) = \frac{\partial u_h^k(\boldsymbol{x},t)}{\partial t} + \nabla\cdot \boldsymbol{f}_h^k(\boldsymbol{x},t).$$

The semi-discrete nodal discontinuous Galerkin formulation is obtained by requiring that the local residual be orthogonal to the Lagrange polynomials in each element T^k

$$\int_{T^k} R_h^k(\boldsymbol{x},t) l_j^k(\boldsymbol{x})\ \mathrm{d}\boldsymbol{x} = \int_{T^k} \left(\frac{\partial u_h^k(\boldsymbol{x},t)}{\partial t} + \nabla\cdot \boldsymbol{f}_h^k(\boldsymbol{x},t) \right) l_j^k(\boldsymbol{x})\ \mathrm{d}\boldsymbol{x} = 0 \quad \forall\ j = 1, ..., N_p.$$

Integration-by-parts in space leads to the weak formulation

$$\int_{T^k} \left(\frac{\partial u_h^k(\boldsymbol{x},t)}{\partial t} l_j^k(\boldsymbol{x}) - \boldsymbol{f}_h^k(\boldsymbol{x},t)\cdot \nabla l_j^k(\boldsymbol{x}) \right) \mathrm{d}\boldsymbol{x} + \sum_{e \in \partial T_k} \int_e \mathbf{n}_e \cdot \widetilde{\boldsymbol{f}}^*(\boldsymbol{x}) l_j^k(\boldsymbol{x})\ \mathrm{d}\sigma = 0, \tag{2.8}$$

where $\mathbf{n}_e$ is the outward unit normal to the edge $e \in \partial T^k$ and $\widetilde{\boldsymbol{f}}^*$ is the interpolant of a consistent numerical flux $\boldsymbol{f}^*(a,b)$ defined at the $p+1$ nodes on e,

$$\widetilde{\boldsymbol{f}}^*(\boldsymbol{x}) = \sum_{m=1}^{p+1} \boldsymbol{f}^*(u_{e,i_m}^-, u_{e,i_m}^+) l_{i_m}^k(\boldsymbol{x}) = \sum_{m=1}^{p+1} \boldsymbol{f}_{i_m}^{e,*} l_{i_m}^k(\boldsymbol{x}) \quad \forall\ \boldsymbol{x} \in e.$$

We define the mass matrix $\boldsymbol{M}$ and the stiffness matrices $\boldsymbol{S}^r$ and $\boldsymbol{S}^s$ on the reference element $\widehat{T}$ as

$$\boldsymbol{M}_{ij} = \int_{\widehat{T}} l_i(\boldsymbol{r}) l_j(\boldsymbol{r}) \, \mathrm{d}\boldsymbol{r}, \quad \boldsymbol{S}^r_{ij} = \int_{\widehat{T}} l_i(\boldsymbol{r}) \frac{\partial l_j(\boldsymbol{r})}{\partial r} \, \mathrm{d}\boldsymbol{r}, \quad \boldsymbol{S}^s_{ij} = \int_{\widehat{T}} l_i(\boldsymbol{r}) \frac{\partial l_j(\boldsymbol{r})}{\partial s} \, \mathrm{d}\boldsymbol{r}.$$

Using the Vandermonde matrix leads to the expressions

$$\boldsymbol{M} = \left(\boldsymbol{V}\boldsymbol{V}^\top\right)^{-1}, \quad \boldsymbol{S}^r = \boldsymbol{M}\boldsymbol{D}^r, \quad \boldsymbol{S}^s = \boldsymbol{M}\boldsymbol{D}^s, \tag{2.9}$$

where

$$\boldsymbol{D}^r = \boldsymbol{V}^r\boldsymbol{V}^{-1}, \quad \boldsymbol{D}^s = \boldsymbol{V}^s\boldsymbol{V}^{-1}, \quad \boldsymbol{V}^r_{ij} = \left.\frac{\partial \psi_j(\boldsymbol{r})}{\partial r}\right|_{\boldsymbol{r}=\boldsymbol{r}_i}, \quad \boldsymbol{V}^s_{ij} = \left.\frac{\partial \psi_j(\boldsymbol{r})}{\partial s}\right|_{\boldsymbol{r}=\boldsymbol{r}_i}. \tag{2.10}$$

Combining (2.9), (2.10) and (2.5), we recover the expressions for the mass and stiffness matrices on the general element T^k

$$\boldsymbol{M}^k = J\boldsymbol{M}, \quad \boldsymbol{S}^x = J\left(\frac{\partial r}{\partial x}\boldsymbol{S}^r + \frac{\partial s}{\partial x}\boldsymbol{S}^s\right), \quad \boldsymbol{S}^y = J\left(\frac{\partial r}{\partial y}\boldsymbol{S}^r + \frac{\partial s}{\partial y}\boldsymbol{S}^s\right). \tag{2.11}$$

Additionally, we can rewrite the boundary integrals in (2.8) as,

$$\int_e \mathbf{n}_e \cdot \widetilde{\boldsymbol{f}}^*(\boldsymbol{x}) l_j^k(\boldsymbol{x}) \, \mathrm{d}\sigma = \sum_{m=1}^{p+1} \left(\int_e l_{i_m}^k(\boldsymbol{x}) l_j^k(\boldsymbol{x}) \, \mathrm{d}\sigma\right) \mathbf{n}_e \cdot \boldsymbol{f}_{i_m}^{e,*} = \sum_{m=1}^{p+1} \boldsymbol{M}^{k,e}_{j,m} (\mathbf{n}_e \cdot \boldsymbol{f}_{i_m}^{e,*}) \tag{2.12}$$

where $M^{k,e}$ is the $N_p \times (p+1)$ mass matrix associated with the edge e of T^k. Finally, substituting (2.11) and (2.12) into (2.8) leads to the semi-discrete DG scheme for each T^k

$$\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}t} - (\boldsymbol{S}^x)^\top \boldsymbol{F}^x - (\boldsymbol{S}^y)^\top \boldsymbol{F}^y + \frac{1}{J}\sum_{e \in \partial T^k} \boldsymbol{M}^{k,e}(\mathbf{n}_e \cdot \boldsymbol{F}^e) = 0, \tag{2.13}$$

where $\boldsymbol{F}^e = \left[\boldsymbol{f}_{i_1}^{e,*}, ..., \boldsymbol{f}_{i_{N+1}}^{e,*}\right]^\top$. The scheme (2.13) is solved using a suitable time-marching scheme, such as the third-order strong stability preserving Runge-Kutta (SSP-RK3) scheme [17]. For further details on the formulation of the scheme, we refer the readers to [20].

3. Limiting strategy

High-order methods suffer from Gibbs oscillations near solution discontinuities. These spurious oscillations can lead to numerical instabilities, especially for systems such as the Euler equations with physical constraints on the evolving quantities. Limiting the local polynomial approximation of the solution near discontinuities, is a popular strategy to overcome this issue. The limiting of solutions in RKDG schemes is carried out as a post-processing step after each RK substage, and comprises two key steps:

1. Flag the troubled-cells in the mesh,
2. Replace the local polynomial in the flagged cells by a suitably limited polynomial.

The focus of this paper is the first step of the procedure, as it has a substantial impact on the overall performance of the scheme.

3.1. Minmod-type TVB indicator

We now describe the minmod-type TVB indicator on triangular grids [9], which will be used as a reference in comparisons of the numerical results in Section 6. We begin by defining the *minmod* function

$$\mathcal{M}(a_1, ..., a_2) = \begin{cases} \text{sign}(a_1).\min(|a_1|, ..., |a_m|) & \text{if } \text{sign}(a_1) = ... = \text{sign}(a_m), \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

and the *TVB modified minmod* function,

$$\mathcal{M}^{tvb}(a_1, .., , a_m; h, M) = \begin{cases} a_1 & \text{if } |a_1| \leqslant Mh^2, \\ \mathcal{M}(a_1, ..., a_2) & \text{otherwise,} \end{cases} \tag{3.2}$$

where h represents some measure of the local mesh size, while $M \geqslant 0$ is a problem dependent parameter [19]. Note that (3.2) essentially reduces to (3.1) for small values of M.

Next, consider the triangle T_0 and its three neighbours T_1, T_2 and T_3, as shown in Figure 2. Let $\boldsymbol{b}_i$ denote the barycenters of each triangle T_i for $i = 0, 1, 2, 3$, and $\boldsymbol{m}_i$ denote the mid-point of the edges shared by the triangles T_0 and T_i for $i = 1, 2, 3$. Then, it is possible to find non-negative coefficients α_1^i, α_2^i and $k_i \in \{1, 2, 3\}/\{i\}$ such that

$$\boldsymbol{m}_i - \boldsymbol{b}_0 = \alpha_1^i(\boldsymbol{b}_i - \boldsymbol{b}_0) + \alpha_2^i(\boldsymbol{b}_{k_i} - \boldsymbol{b}_0), \quad i = 1, 2, 3. \tag{3.3}$$

For instance, for $\boldsymbol{m}_1$ shown in Figure 2, it is possible to find non-negative α_1^1, α_2^1 for $k_1 = 2$. The coefficients can be pre-computed and stored for all triangular patches in a given fixed mesh.

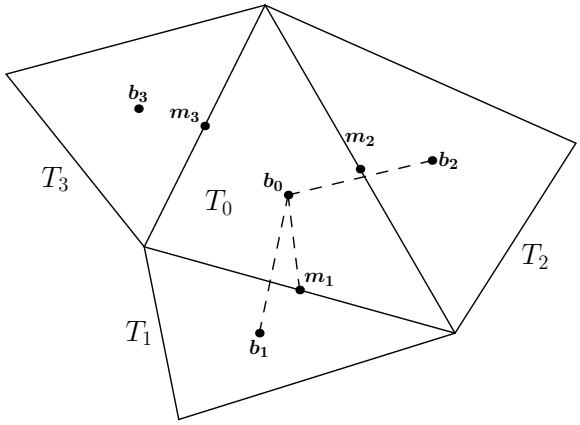


Figure 2: 4-cell stencil centered at T_0, with $\boldsymbol{b_i}$ denoting the barycenter of the triangle T_i and $\boldsymbol{m_i}$ denoting the mid-point of the edge shared by triangles T_0 and T_i.

We are now ready to outline the algorithm to detect troubled-cells, given the numerical approximation $u_h \in \mathcal{V}_r^h$. For each patch of 4 triangles (refer to Figure 2):

1. Project u_h to the space of piece-wise linear functions $\mathcal{V}_1^h$ with preserved cell-averages. In other words, find $\widetilde{u}_h(\boldsymbol{x}) = \Pi_h^1 u_h(\boldsymbol{x}) \in \mathcal{V}_1^h$, such that

$$\overline{u}_i = \frac{1}{|T_i|}\int_{T_i} u_h(\boldsymbol{x})\ \mathrm{d}\boldsymbol{x} = \frac{1}{|T_i|}\int_{T_i} \widetilde{u}_h(\boldsymbol{x})\ \mathrm{d}\boldsymbol{x}, \quad i = 0, 1, 2, 3.$$

2. For $i = 1, 2, 3$, define the quantities

$$\begin{aligned} \Delta\overline{u}_i &:= \alpha_1^i(\overline{u}_i - \overline{u}_0) + \alpha_2^i(\overline{u}_{k_i} - \overline{u}_0), \\ \Delta_i &:= \mathcal{M}^{tvb}\big(\ (\widetilde{u}_h(\boldsymbol{m}_i) - \overline{u}_0),\ \kappa\Delta\overline{u}_i\ ;\ h,\ M\ \big), \quad \kappa > 1 \end{aligned} \tag{3.4}$$

where α_j^i, k_i are pre-computed to satisfy (3.3), and h is the circumradius of T_0. As suggested in [9], we fix $\kappa = 1.5$.

3. If $\Delta_1 + \Delta_2 + \Delta_3 = 0$, proceed to step 4. Otherwise, compute

$$\beta^+ = \sum_{i=1}^{3} \max(0, \Delta_i), \quad \beta^- = \sum_{i=1}^{3} \max(0, -\Delta_i), \quad \theta^+ = \min\left(1, \frac{\beta^-}{\beta^+}\right), \quad \theta^- = \min\left(1, \frac{\beta^+}{\beta^-}\right),$$

to modify Δ_i as

$$\Delta_i \longleftarrow \theta^+ \max(0, \Delta_i) - \theta^- \max(0, -\Delta_i).$$

4. Compute modified values at the edge mid-points

$$\hat{u}_h(\boldsymbol{m}_i) = \overline{u}_0 + \Delta_i, \quad i = 1, 2, 3.$$

5. Flag the cell T_0 as a troubled-cell if $\widetilde{u}_h(\boldsymbol{m}_i) \neq \hat{u}_h(\boldsymbol{m}_i)$ at either of the three mid-points.

For further details, we refer to [9].

The problem-dependent parameter M in (3.4) plays a key role and needs to be appropriately set to flag the genuine troubled-cells. However, it is difficult to estimate M for a general system of conservation laws. The TVB indicator become very conservative if a very small value of M is chosen, leading to issues near smooth extrema. On the other hand, if M is chosen too large, the indicator may not flag all troubled-cells, leading to the re-appearance of Gibbs oscillations. Thus, it is highly desirable to use a troubled-cell indicator free from problem-dependent parameters. This objective was achieved in [34] for one-dimensional conservation laws, by training an feedforward neural network to detect troubled-cells. In Section 4, we extend this approach and propose a similar network for two-dimensional problems.

3.2. Barth-Jespersen limiter

Once the troubled-cells are flagged, the local polynomial is limited using the Barth-Jespersen limiter [2]. To limit the local solution $u_h^k(\boldsymbol{x})$ in the triangle T^k using its 3 neighbours, we proceed as:

1. Evaluate the maximum M and the minimum m of the cell-averages of the solution u_h in T^k and its 3 neighbours.
2. Compute $\widetilde{u}_h^k(\boldsymbol{x}) = \Pi_h^1 u_h^k(\boldsymbol{x}) \in P^1(T^k)$ such that the cell-average $\overline{u}_k$ is preserved.
3. Let $\boldsymbol{x}_i^*$, $i = 1, ..., K_{\partial T^k}$ be the computational nodes on ∂T^k. For each of these nodes, evaluate

$$\alpha_i = \begin{cases} \frac{M - \overline{u}_k}{\widetilde{u}_h^k(\boldsymbol{x}_i^*) - \overline{u}_k} & , \quad \text{if} \quad \widetilde{u}_h^k(\boldsymbol{x}_i^*) > M, \\ \frac{m - \overline{u}_k}{\widetilde{u}_h^k(\boldsymbol{x}_i^*) - \overline{u}_k} & , \quad \text{if} \quad \widetilde{u}_h^k(\boldsymbol{x}_i^*) < m, \\ 1 & , \quad \text{otherwise.} \end{cases}$$

and set

$$\alpha = \min_{1 \leqslant i \leqslant K_{\partial T^k}} \max(\alpha_i, 0).$$

4. Compute the limited polynomial $\hat{u}_h^k$ in T^k as

$$\hat{u}_h^k(\boldsymbol{x}) = \begin{cases} \overline{u}_k + \alpha \left[\nabla \widetilde{u}_h^k \cdot (\boldsymbol{x} - \boldsymbol{x}_c)\right] & , \quad \text{if} \quad \alpha \neq 1, \\ u_h^k(\boldsymbol{x}) & , \quad \text{if} \quad \alpha = 1, \end{cases}$$

where $\boldsymbol{x}_c$ is the barycenter of T^k.

The limiter ensures that the value of $\hat{u}_h^k(\boldsymbol{x})$ at the boundary nodes $\boldsymbol{x}_i^*$ lie between the cell-average bounds m and M.

4. Multilayer Perceptron (MLP)

Multilayer perceptrons represent a specific type of artificial neural network architecture, and have been demonstrated to have excellent classification and regression capabilities when applied to image recognition [11], speech recognition [13], and for solving differential equations [26, 16, 35, 21, 27, 33]. In such a network, the basic computing units (neurons) are stacked in multiple layers to form a feedforward network. The first layer is called the *source layer*, the last layer is termed as the *output layer*, while the remaining layers lying in between are called the *hidden layers*. An example of an MLP with two hidden layers, is shown in Figure 3. We note that no computations occur inside the source layer of an MLP, as it simply provides the source signal to the network.

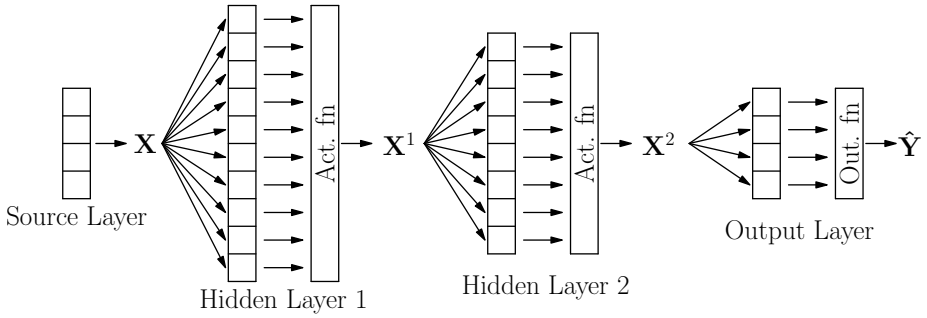


Figure 3: An MLP with 2 hidden layers. The source layer transmits the signal $\mathbf{X}$ to the first hidden layer. The final output of the network is $\hat{\boldsymbol{Y}}$.

A given neuron receiving an input $\mathbf{X} = (X_1, ...X_d) \in \mathbf{R}^d$, computes and stores a weighted linear sum of the signal

$$q = \sum_{i=1}^{d} w_i X_i + b,$$

where w_i, b are the weights and bias associated with the neuron. The accumulation q from each neuron then passes through a non-linear *activation function* $\mathcal{A}$, to give a final output signal transmitted by the neuron. The activation function plays an important role in enhancing the networks learning capabilities. Several suitable choices exist for the activation function, each having its own set of advantages and disadvantages. Motivated by the network designed in [34], we choose the *leaky Rectified Linear Unit* (ReLU)

$$\mathcal{A}(x) = \max(0, x) - \nu \max(0, -x), \tag{4.1}$$

as the activation function, with a leak coefficient $\nu > 0$. Furthermore, the signals from the output layer do not pass through an activation function, but may pass through an output function $\mathcal{R}$ to convert the signals into a meaningful form. A specific type of $\mathcal{R}$ for the application being considered in this paper, is discussed in Section 5.

Consider an MLP with L hidden layers, with N_I source neurons, N_O output neurons and N_k neurons in the k-th hidden layer. The weight and bias matrices associated with the network are given by

$$\begin{aligned} \text{First hidden layer} &: \quad \boldsymbol{W}^1 \in \mathbb{R}^{N_1 \times N_I}, \quad \boldsymbol{B}^k \in \mathbb{R}^{N_1}, \\ L-1 \text{ hidden layers} &: \quad \boldsymbol{W}^k \in \mathbb{R}^{N_k \times N_{k-1}}, \quad \boldsymbol{B}^k \in \mathbb{R}^{N_k}, \quad k = 2, ..., L, \\ \text{Output layer} &: \quad \boldsymbol{W}^O \in \mathbb{R}^{N_O \times N_L}, \quad \boldsymbol{B}^L \in \mathbb{R}^{N_O}, \end{aligned}$$

with the i-th row of the weight and bias matrix of a given layer specifying the weights and biases of the i-th neuron in that layer. For an input $\mathbf{X} \in \mathbb{R}^{N_I}$ to the network, the response $\hat{\boldsymbol{Y}}$ is given by,

$$\begin{aligned} \mathbf{X}^0 &= \mathbf{X}, \\ \mathbf{X}^k &= \mathcal{A}(\boldsymbol{W}^k \mathbf{X}^{k-1} + \boldsymbol{B}^k) \in \mathbb{R}^{N_k}, \quad k = 1, ..., L, \\ \hat{\boldsymbol{Y}} &= \mathcal{R}(\boldsymbol{W}^O \mathbf{X}^L + \boldsymbol{B}^O) \in \mathbb{R}^{N_O}. \end{aligned} \tag{4.2}$$

4.1. Training with supervised learning

Let us assume that we are given the values of an unknown function $\mathbf{G}$ on a set of input points $\mathbf{X}_p$

$$\mathbf{G} : \Omega \in \mathbb{R}^{N_I} \mapsto \mathbb{R}^{N_O}, \quad \text{with} \quad \mathbb{T} = \{(\mathbf{X}_p, \mathbf{Y}_p) \mid \mathbf{Y}_p = \mathbf{G}(\mathbf{X}_p) \ \forall \ p \in \Lambda\}.$$

We seek a suitable approximation $\hat{\mathbf{G}}$ for $\mathbf{G}$ by using the dataset $\mathbb{T}$, such that $\hat{\mathbf{G}}$ accurately predicts the responses corresponding to the $\mathbb{T}$, as well as for data points lying outside $\mathbb{T}$, i.e., the network can *generalize* well. The MLP (4.2) can be trained to represent $\hat{\mathbf{G}}$ by suitably choosing the weights and biases of the network. This is achieved by first defining a cost functional

$$\mathcal{C} := \mathcal{C}(\mathbf{Y}, \hat{\mathbf{Y}}), \quad \mathbf{Y} = \mathbf{G}(\mathbf{X}), \quad \hat{\mathbf{Y}} = \hat{\mathbf{G}}(\mathbf{X}), \qquad \forall \ \mathbf{X} \in \mathbb{R}^{N_I},$$

which measure the error between the predicted output and the truth. Then, an iterative optimization algorithm is employed to determine the optimal weights and biases that minimize $\mathcal{C}$ over the training set $\mathbb{T}$. We note that the training is performed *offline* and only once, after which the network can be used as a blackbox.

4.2. Stochasticity and overfitting

To speed up convergence of the training, a stochastic optimization algorithm is used [3]. More precisely, the full training set with S data-points is shuffled, after which mini-batches with $S_b < S$ samples are sequentially extracted to take S/S_b optimization steps. Once the entire training set is exhausted, i.e., the completion of an *epoch*, the training set is re-shuffled and the process is repeated. This is done over several epochs, until the cost function decreases below a desired threshold.

Special care must be taken to ensure that the trained network does not *overfit* the dataset $\mathbb{T}$. One method to overcoming this issue is to regularize the cost functional [28] by penalizing the weights $\boldsymbol{W}$ of the network

$$\widetilde{\mathcal{C}} = \mathcal{C} + \frac{\beta}{S_b} \|\boldsymbol{W}\|_2^2, \quad \beta \geqslant 0, \tag{4.3}$$

where $\|\boldsymbol{W}\|_2^2$ represents the squared sum of all the weights in the MLP. To monitor overfitting, we construct a secondary *validation* data set $\mathbb{V}$, independent of the training set $\mathbb{T}$. A high-accuracy on $\mathbb{T}$ but a low accuracy on $\mathbb{V}$ signifies overfitting. Adjusting the network hyper-parameters, such as β, S_b and ν (see (4.1)), helps to improve the performance on $\mathbb{V}$.

5. An MLP 2D trouble-cell indicator

Motivated by the numerical results in [34] for the one-dimensional set-up, we train an MLP to perform as a troubled-cell indicator in two-dimensions. The network input $\mathbf{X}$ is chosen to represent the local solution structure on the mesh. Consider the 4-cell patch shown in Figure 2, and let $u_h^i(\boldsymbol{x})$ be the local approximation in the triangles T_i for $i = 0, 1, 2, 3$, respectively. From each of these local polynomials, we extract the P^1 modal coefficients. Based on the local indexing (2.7), this corresponds to the coefficients $\hat{u}_1, \hat{u}_2$ and $\hat{u}_{p+2}$ when a basis of order p is used. Using these values, we assemble the input for the MLP corresponding to this triangular patch as

$$\mathbf{X} = \left[\hat{u}_1^0, \hat{u}_2^0, \hat{u}_{p+2}^0, \hat{u}_1^1, \hat{u}_2^1, \hat{u}_{p+2}^1, \hat{u}_1^2, \hat{u}_2^2, \hat{u}_{p+2}^2, \hat{u}_1^3, \hat{u}_2^3, \hat{u}_{p+2}^3\right]^\top \in \mathbb{R}^{12}. \tag{5.1}$$

We use 5 hidden layers, each having a width of $N_k = 20$ neurons. The leaky ReLU function (4.1) is chosen as the activation function for each hidden layer, with a leak factor of $\nu = 10^{-3}$. The output layer has a width of $N_0 = 2$ neurons, with the network giving the output $\hat{\mathbf{Y}} = [\hat{Y}^1, \hat{Y}^2]^\top$. The *softmax* function,

$$\text{softmax}(\mathbf{Z})_i = \frac{e^{Z_i}}{\sum_{j=1}^{M} e^{Z_j}}, \quad i = 1, ..., M,$$

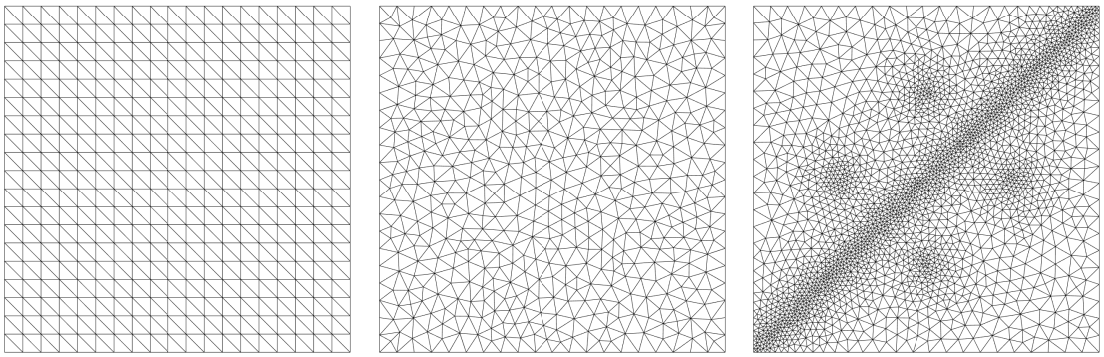
is used as the output function, ensuring that the components of the output vector form a convex combination. Thus, the final output can be seen as the probability of the cell T_0 falling into either of the two classes: a troubled-cell or a good-cell. Specifically, $\hat{Y}^1$ represents the probability that the cell being analyzed is a troubled-cell.

To train the network, we choose a regularized cost functional of the form (4.3), with $\mathcal{C}$ given by the *cross entropy function*

$$\mathcal{C} = -\frac{1}{S_b} \sum_{s=1}^{S_b} \left[Y_s^1 \log(\hat{Y}_s^1) + Y_s^2 \log(\hat{Y}_s^2) \right],$$

where the minibatch size is set to $S_b = 500$, while the coefficient of regularization is chosen as $\beta = 10^{-3}$. We use the momentum based Adam stochastic optimizer [22] to minimize the cost function $\widetilde{\mathcal{C}}$.

The training and validation sets are generated by evaluating a set of known functions on various discretizations of the domain $[-1, 1]^2$. The meshes used for this purpose are shown in Figure 4, where Mesh A is a structured mesh, Mesh B is an unstructured mesh with a fixed mesh size, while mesh C is an unstructured mesh with a locally varying mesh size. The functions used to generate $\mathbf{X}$ are listed in Tables 1 and 2, and locally represent canonical solutions observed as solutions of conservation laws. Note that the notation $\mathcal{U}[z_1, z_2]$ denotes the uniform distribution on the interval $[z_1, z_2]$. For a given function on a given mesh, the data (5.1) is extracted from each 4-cell patch to form sample points in the data sets. The corresponding truth value is set to $[1, 0]^\top$ if a discontinuity or a kink (observed for the last function in Tables 1 and 2) exists in the circumcircle around the central triangle of the patch. Otherwise, the value is set to $[0, 1]^\top$.



(a) Mesh A (b) Mesh B (c) Mesh C

Figure 4: Meshes used to generate $\mathbb{T}$ and $\mathbb{V}$. A refinement of each of these three meshes is used, where each triangle of the coarser mesh was divided into four similar triangles.

$\mathbf{u}(\mathbf{x})$	**Parameters**	**Good cells**	**Troubled cells**
$ax + by$	$a, b \in \mathcal{U}[-10, 10]$	34,776	0
$\sum_{k=1}^{N_f} a_k \sin(k\pi x) + b_k \cos(k\pi y)$	$a_k, b_k \in \mathcal{U}[-1, 1], \quad N_f = 1, 2, 3$	202,980	0
$\exp(a_1[(x-a_2)^2 + (y-a_3)^2])$ $+ \exp(a_4[(x-a_5)^2 + (y-a_6)^2])$	$a_i \in \mathcal{U}[-1, 1]$	135,320	0
4 values u_1, u_2, u_3, u_4 in 4 sections created by the lines $y - y_0 = m(x - x_0)$ and $y - y_0 = -1/m(x - x_0)$ (only troubled-cells selected)	$u_i \in \mathcal{U}[-1, 1], \quad m \in \mathcal{U}[0, 20],$ $x_0, y_0 \in \mathcal{U}[-0.5, 0.5]$	0	337,976
$a\lvert(y - y_0) - m(x - x_0)\rvert$ (only troubled-cells selected)	$a \in \mathcal{U}[-100, 100], \quad m \in \mathcal{U}[-1, 1],$ $x_0, y_0 \in \mathcal{U}[-0.5, 0.5]$	0	60,182
		373,076	**398,158**

Table 1: Functions used to generate $\mathbb{T}$ on the meshes shown in Figure 4.

$\mathbf{u}(\mathbf{x})$	**Parameters**	**Good cells**	**Troubled cells**
$\sum_{k=1}^{N_f} a_k \sin(k\pi x) + b_k \cos(k\pi y)$	$a_k, b_k \in \mathcal{U}[-1,1], \quad N_f = 3$	33,830	0
$\sum_{\lvert p\rvert <= N} a_p \boldsymbol{x}^p$	$a_p \in \mathcal{U}[-1,1], \quad p = 2,3,4$	101,490	0
4 values u_1, u_2, u_3, u_4 in 4 sections created by the lines $y - y_0 = m(x - x_0)$ and $y - y_0 = -1/m(x - x_0)$ (only troubled-cells selected)	$u_i \in \mathcal{U}[-1,1], \quad m \in \mathcal{U}[0,20],$ $x_0, y_0 \in \mathcal{U}[-0.5, 0.5]$	0	33,294
$a\lvert(y - y_0) - m(x - x_0)\rvert$ (only troubled-cells selected)	$a \in \mathcal{U}[-100,100], \quad m \in \mathcal{U}[-1,1],$ $x_0, y_0 \in \mathcal{U}[-0.5, 0.5]$	0	22,993
		135,320	**56,287**

Table 2: Functions used to generate $\mathbb{V}$ on the meshes shown in Figure 4.

Finally, the input data needs to be appropriately scaled before being fed into the network, to ensure faster convergence during training and improve the networks ability to generalize. This is achieved using the following function for the data $\mathbf{X} = [X_1, ..., X_{12}]^\top$

$$\text{Scale}(\mathbf{X}) = \begin{cases} \frac{\mathbf{X}}{\max_i(\lvert X_i\rvert)} & \text{if } \max_i(\lvert X_i\rvert) > 1, \\ \mathbf{X} & \text{otherwise.} \end{cases}$$

Remark 5.1. *The accuracy of the network on the training set $\mathbb{T}$ is measured as*

$$Acc_{\mathbb{T}} = \frac{\#\{(\mathbf{X}, \mathbf{Y}) \in \mathbb{T} \ : \ \hat{\mathbf{Y}} = \hat{\mathbf{G}}(\mathbf{X}), \ \ \lvert\hat{Y}^1 - Y^1\rvert \leqslant 0.5\}}{\#\mathbb{T}} \times 100,$$

with a similar expression for $\mathbb{V}$.

Remark 5.2. *The data sets could also have been constructed by using actual numerical solutions to conservation laws. However, tracking the genuine troubled cells as the solution evolves is difficult in general, making it hard to generate data for the supervised learning paradigm.*

6. Numerical results

We now demonstrate the capability of the MLP network when used as a troubled-cell indicator with a RKDG scheme. The DG scheme is computed with the local Lax-Friedrichs flux. We compare the performance of the proposed MLP indicator with that of the minmod-type TVB limiter. The notation TVB-1, TVB-2, and TVB-3 refer to the TVB limiter with the TVB constants $M = 10$, $M = 100$, and $M = 200$, respectively. Time-marching is performed using the SSP-RK3 scheme.

Triangles with constant solutions should not be flagged as troubled-cells, and do not require any limiting. However, it is not easy to train the network to correctly classify such cells for all possible constant values. Thus, we filter out elements with almost constant data using the following algorithm:

1. For the nodal values $\boldsymbol{U}^k = [u_1^k, ..., u_{N_p}^k]^\top$ in each trangle T^k, find

$$U_{\min}^k = \min_i(u_1^k), \quad U_{\max}^k = \max_i(u_1^k), \quad i = 1, ..., K.$$

2. Flag a cell as a constant cell if

$$|U_{\max}^k - U_{\min}^k| < \epsilon \max(|U_{\min}^k|,\ |U_{\max}^k|),$$

where $\epsilon = 0.01$.

To ensure a fair comparison, this filter is also applied for the TVB indicator to exclude constant cells from the limiting procedure.

For all the test cases below, the domains are discretized using two types of meshes. The nomenclature and description of these meshes are as follows:

- **Mesh S-M:** This mesh is obtained by considering a structured quadrilateral mesh with M elements on each edge, and diving each quadrilateral element into two triangles. Thus, the mesh will have $K = M \times M \times 2$ triangular elements.
- **Mesh U-q:** This is an unstructured triangular mesh with a characteristic mesh size q.

We use a fixed time-step Δt for each mesh to make a meaningful comparison between the indicators, as the number of cells flagged by an indicator is sensitive to the local time-step. Furthermore, the time-step is multiplied by a scaling function $w(p)$ to ensure that the time-steps satisfy stable CFL conditions as the basis order p is varied. We set the values of this function as

$$w(1) = 1, \quad w(2) = 0.4, \quad w(3) = 0.25. \tag{6.1}$$

6.1. Linear advection

We first consider the scalar linear advection problem

$$\frac{\partial u}{\partial t} + c^x \frac{\partial u}{\partial x} + c^y \frac{\partial u}{\partial y} = 0,$$

and set the advection speed $(c^x, c^y) = (1, 1)$. The initial condition is composed of smooth trigonometric functions

$$u_0(\boldsymbol{x}) = \sin(4\pi x)\cos(4\pi y), \quad \boldsymbol{x} \in [0, 1]^2$$

with periodic boundary conditions, and is advected diagonally till $T_f = 0.5$. The domain is discretized using four meshes, whose details along with the simulation time-step Δt are given in Table 3.

Mesh	$\boldsymbol{K}$	$\boldsymbol{\Delta t}$
S-100	20,000	$0.01 \times w(p)$
S-150	45,000	$0.005 \times w(p)$
U-0.01	26,414	$0.01 \times w(p)$
U-0.005	106,484	$0.005 \times w(p)$

Table 3: Meshes used for linear advection of a smooth wave, along with the time-step Δt used. Note that $w(p)$ is given by (6.1).

Indicator	Mesh	$p=1$		$p=2$		$p=3$	
		Max. % cells flagged	Avg. % cells flagged	Max. % cells flagged	Avg. % cells flagged	Max. % cells flagged	Avg. % cells flagged
TVB-1	S-100	31.71	29.51	32.10	28.57	33.78	29.91
	S-150	20.46	18.90	24.85	21.98	27.01	24.00
	U-0.01	29.67	28.33	33.15	31.23	32.29	30.35
	U-0.005	19.43	18.46	28.20	26.54	28.39	26.79
TVB-2	S-100	28.06	26.40	28.20	25.11	30.80	26.90
	S-150	17.35	16.41	22.10	19.73	25.51	22.30
	U-0.01	25.00	23.85	28.85	26.97	27.83	25.71
	U-0.005	16.30	15.40	25.44	23.87	25.32	23.72
TVB-3	S-100	23.86	22.66	24.48	22.13	26.34	23.63
	S-150	14.77	13.95	19.08	17.44	21.54	19.65
	U-0.01	20.88	19.99	25.34	23.46	24.12	22.27
	U-0.005	13.47	12.67	23.36	21.66	23.20	21.50
MLP	S-100	1.33	0.74	1.04	0.23	0.92	0.33
	S-150	0.24	0.12	0.23	0.02	0.18	0.01
	U-0.01	0.71	0.56	0.81	0.60	1.08	0.76
	U-0.005	0.13	0.11	0.19	0.13	0.18	0.13

Table 4: Maximum and time-averaged percentage of cells flagged as troubled-cells while advecting a smooth wave.

Ideally, no cell should be flagged by an indicator since the solution is smooth. In Table 4, we list the maximum and time-averaged percentage of cells flagged as troubled-cells, by the four indicators on the various meshes. We clearly see that for the values of parameter M considered, the TVB indicator flags an average of 20% cells or more, especially for higher order p. This leads to a loss in accuracy of the solution, as seen in Figures 5 and 6. On the other hand, the MLP indicator flags an average of 1% (or less) cells, leading to substantially more accurate approximations.

To measure the overall computational cost, we first note that TVB-1 is expected to be the most expensive among TVB-1, TVB-2 and TVB-3, since it flags the maximum number of cells on given mesh. In Table 5, we list the computational times obtained with TVB-2, TVB-3 and MLP relative to TVB-1 on the various meshes. It can be seen that the MLP indicator leads to a scheme which is marginally more expensive compared to TVB-1. However, the MLP indicator yields far more accurate solutions without the prescription of problem-dependent parameters.

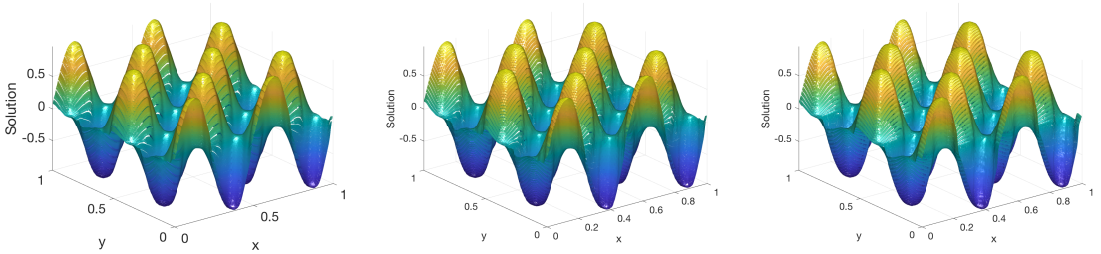
6.2. Burgers equation

We consider the two-dimensional extension of Burgers equation

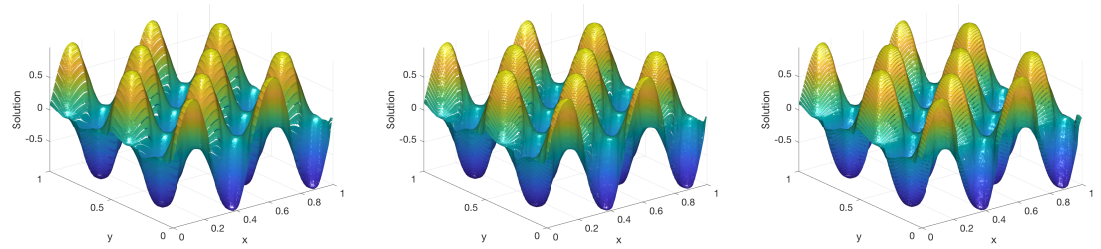
$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) + \frac{\partial}{\partial y}\left(\frac{u^2}{2}\right) = 0,$$

and an initial condition

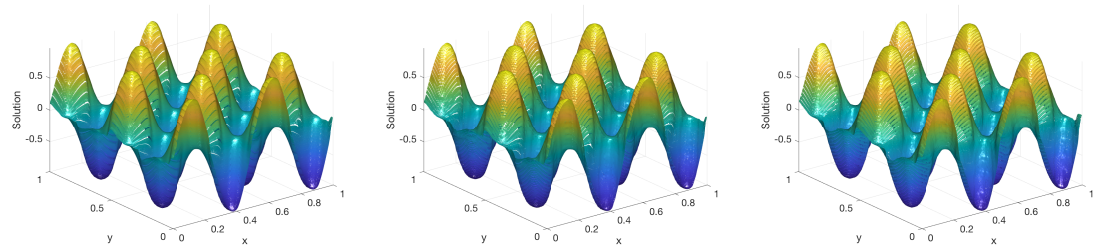
$$u_0(\boldsymbol{x}) = \begin{cases} \sin(2\pi(x+0.5))\sin(2\pi(y+0.5)) & \text{if } |x| \leqslant 0,\ |y| \leqslant 0 \\ 0 & \text{otherwise,} \end{cases}$$



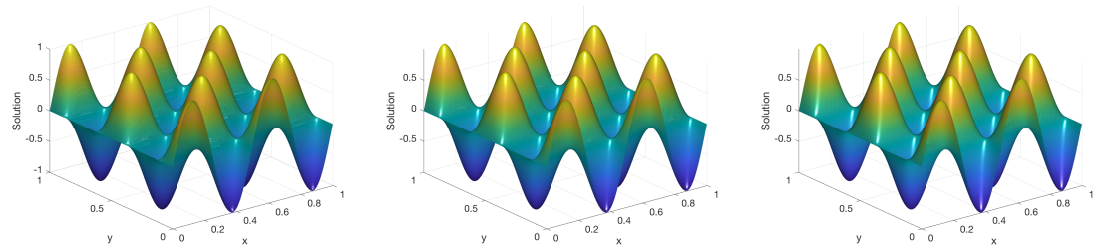
(a) TVB-1, p=1 (b) TVB-1, p=2 (c) TVB-1, p=3



(d) TVB-2, p=1 (e) TVB-2, p=2 (f) TVB-2, p=3

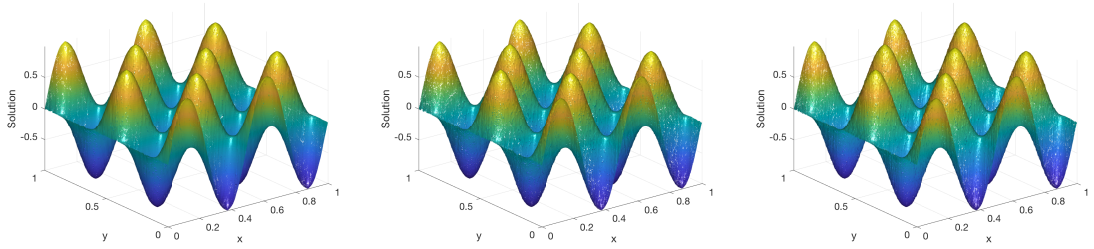


(g) TVB-3, p=1 (h) TVB-3, p=2 (i) TVB-3, p=3

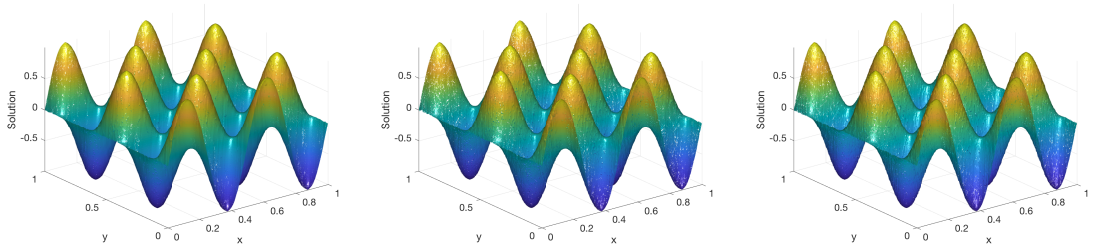


(j) MLP, p=1 (k) MLP, p=2 (l) MLP, p=3

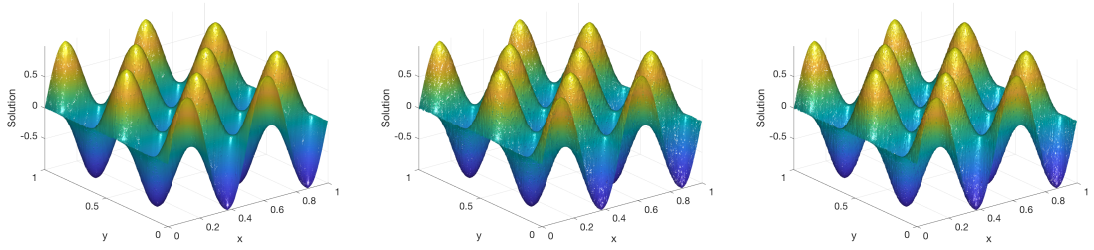
Figure 5: Solution of the linear advection of a smooth wave at final time, obtained on the mesh S-150.



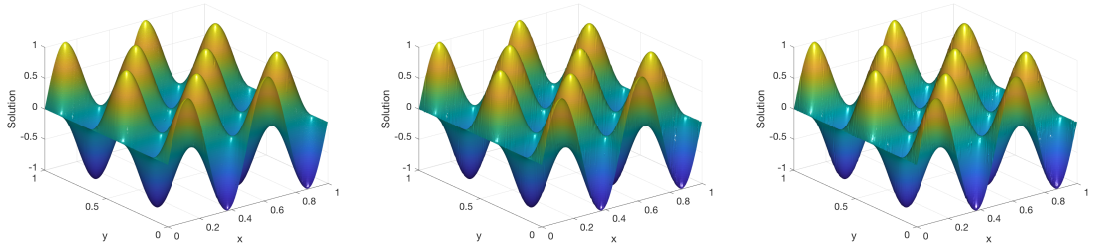
(a) TVB-1, p=1 (b) TVB-1, p=2 (c) TVB-1, p=3



(d) TVB-2, p=1 (e) TVB-2, p=2 (f) TVB-2, p=3



(g) TVB-3, p=1 (h) TVB-3, p=2 (i) TVB-3, p=3



(j) MLP, p=1 (k) MLP, p=2 (l) MLP, p=3

Figure 6: Solution of the linear advection of a smooth wave at final time, obtained on the mesh U-0.005.

Indicator	Mesh	$p=1$	$p=2$	$p=3$
TVB-2	S-100	0.984	0.976	0.983
	S-150	0.979	0.979	0.989
	U-0.01	0.983	0.981	0.977
	U-0.005	0.986	0.987	0.972
TVB-3	S-100	0.973	0.963	0.964
	S-150	0.969	0.972	0.975
	U-0.01	0.964	0.961	0.958
	U-0.005	0.977	0.970	0.975
MLP	S-100	1.015	0.974	0.913
	S-150	1.254	1.166	1.056
	U-0.01	1.171	1.012	0.964
	U-0.005	1.366	1.159	1.078

Table 5: Computational time with various indicators relative to TVB-1, for linear advection of a smooth wave.

on the domain $[-1,1]^2$ with periodic boundary conditions. The solution is computed till time $T_f = 0.2$, with the details of the meshes given in Table 6.

Mesh	K	Δt
S-100	20,000	$0.01 \times w(p)$
U-0.02	26,790	$0.01 \times w(p)$

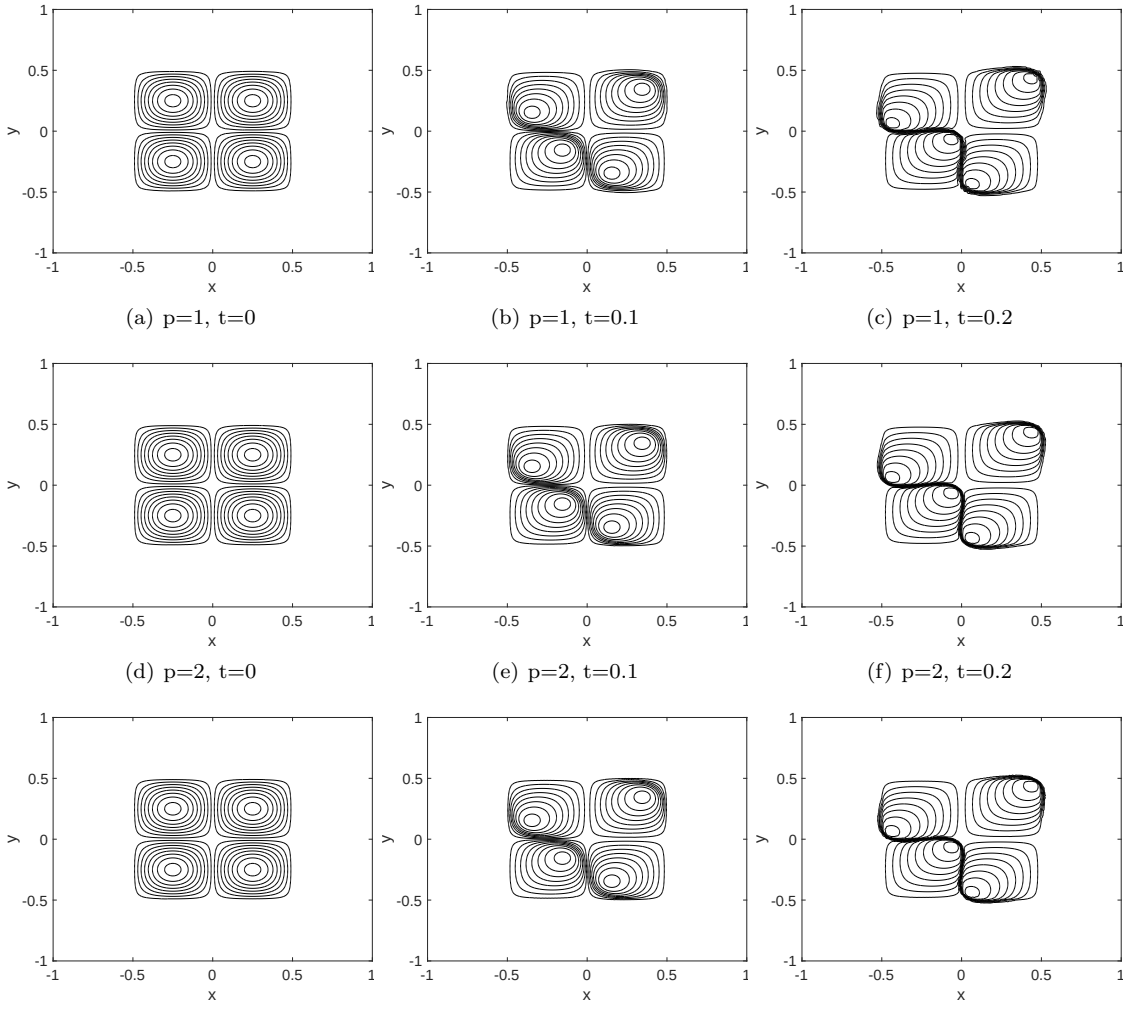
Table 6: Meshes used for Burgers equation, along with the time-step Δt used. Note that $w(p)$ is given by (6.1).

The contour plots of the solution at times $t = 0, 0.1$ and 0.2 computed using the MLP indicator on both meshes, are shown in Figures 7 and 9. The corresponding plots obtained with the TVB indicator are indistinguishable compared to MLP, and thus not shown here. Note that the initial condition is smooth in the interior of the square $[-0.5, 0.5]^2$, with a sharp kink all along its boundary. The solution remains smooth inside this square till $t = 0.1$. Thus a minimal number of cells should be flagged in the interior. After $t = 0.1$, two shock-fronts appear in the solution. Furthermore, a significant portion of the sharp kink is smoothened out by rarefaction waves. To assess the performance and efficiency of the indicators, we analyse the time-history of the troubled-cells flagged. We only show the results for $p = 3$ due to paucity of space. In Figure 8, we observe that TVB-1 and TVB-2 flag cells with smooth extrema on the mesh S-100, and overall mark more cells than necessary. TVB-3 flags a smaller number of cells compared to MLP. However, TVB-3 is unable to detect cells with a kink, unlike MLP. The MLP and TVB-3 indicators mark a comparable number of cells on the mesh U-0.02, with MLP performing marginally better, as shown in Figure 10.

6.3. Euler equations

Finally, we consider the two dimensional Euler equations given by

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{pmatrix}, \qquad \boldsymbol{f}_x(\boldsymbol{u}) = \begin{pmatrix} \rho v_x \\ \rho v_x^2 + \mathcal{P} \\ \rho v_x v_y \\ (E+\mathcal{P}) v_x \end{pmatrix}, \quad \boldsymbol{f}_y(\boldsymbol{u}) = \begin{pmatrix} \rho v_y \\ \rho v_y v_x \\ \rho v_y^2 + \mathcal{P} \\ (E+\mathcal{P}) v_y \end{pmatrix}. \tag{6.2}$$

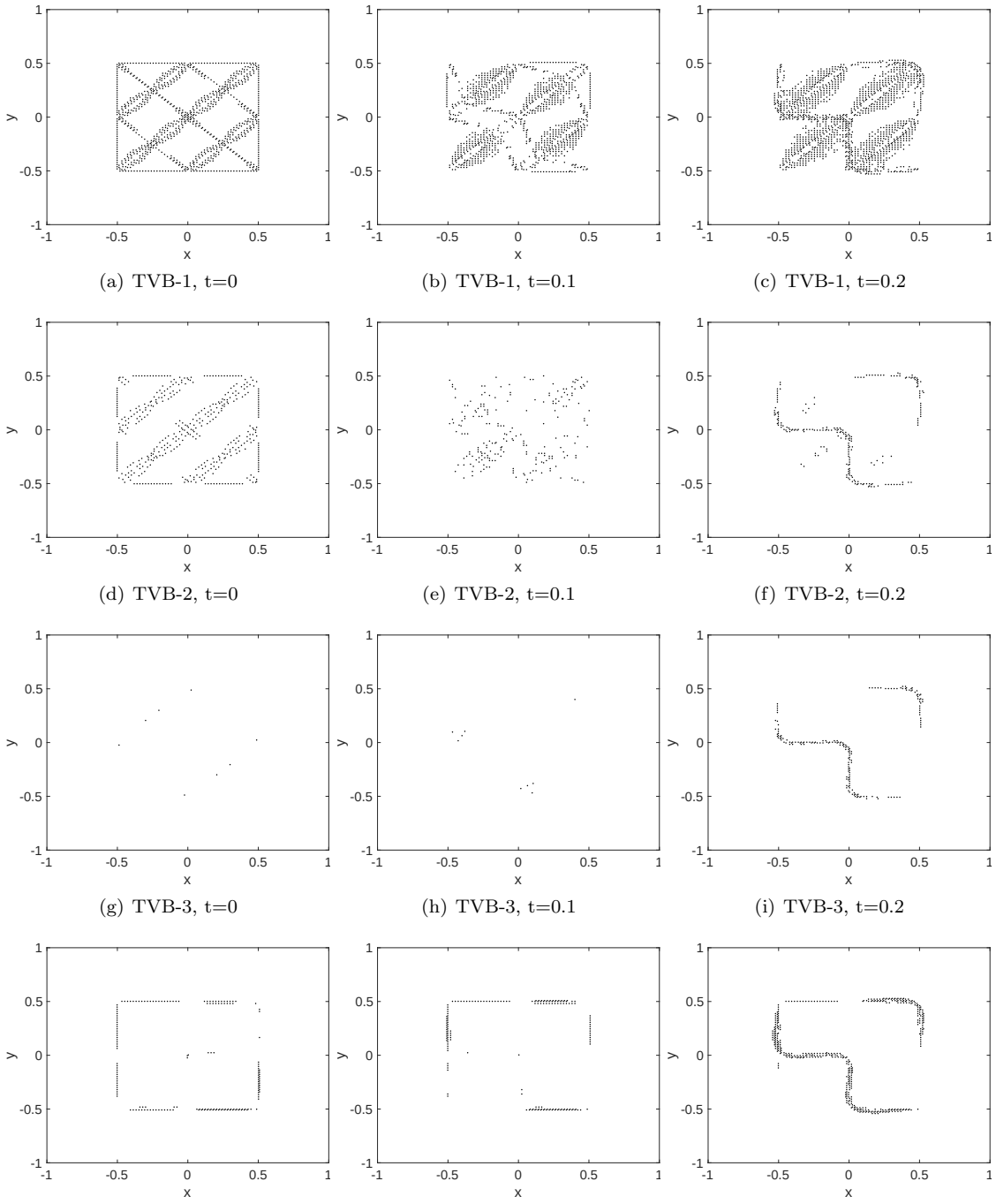


(a) p=1, t=0 (b) p=1, t=0.1 (c) p=1, t=0.2

(d) p=2, t=0 (e) p=2, t=0.1 (f) p=2, t=0.2

(g) p=3, t=0 (h) p=3, t=0.1 (i) p=3, t=0.2

Figure 7: Solution for Burgers equation at $t = 0, 0.1, 0.2$ on S-100 using the MLP indicator, with 20 equally spaced contour lines between -1.2 and 1.2.



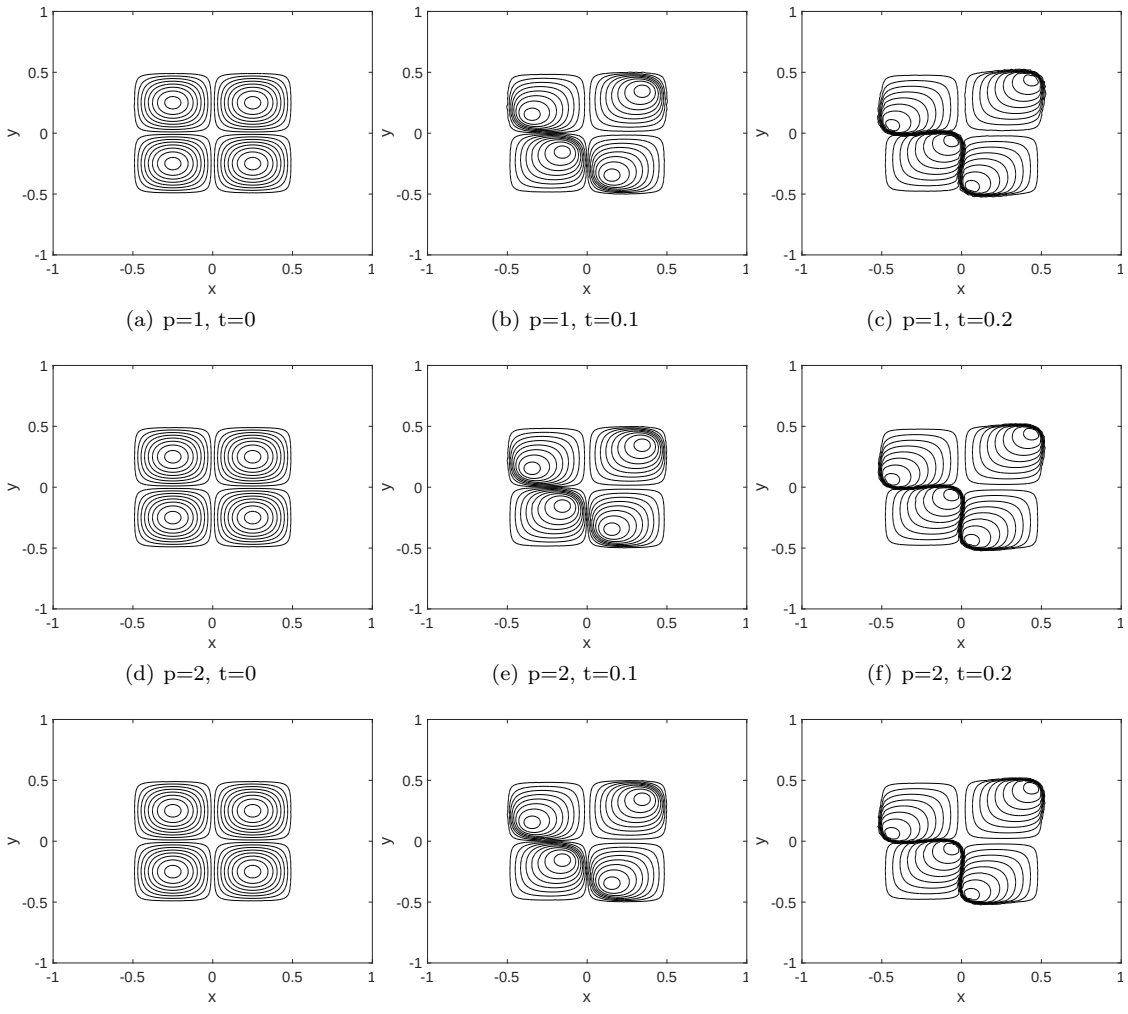
(a) TVB-1, t=0 (b) TVB-1, t=0.1 (c) TVB-1, t=0.2

(d) TVB-2, t=0 (e) TVB-2, t=0.1 (f) TVB-2, t=0.2

(g) TVB-3, t=0 (h) TVB-3, t=0.1 (i) TVB-3, t=0.2

(j) MLP, t=0 (k) MLP, t=0.1 (l) MLP, t=0.2

Figure 8: Troubled-cells flagged for Burgers equation at $t = 0, 0.1, 0.2$ on S-100 and p=3.

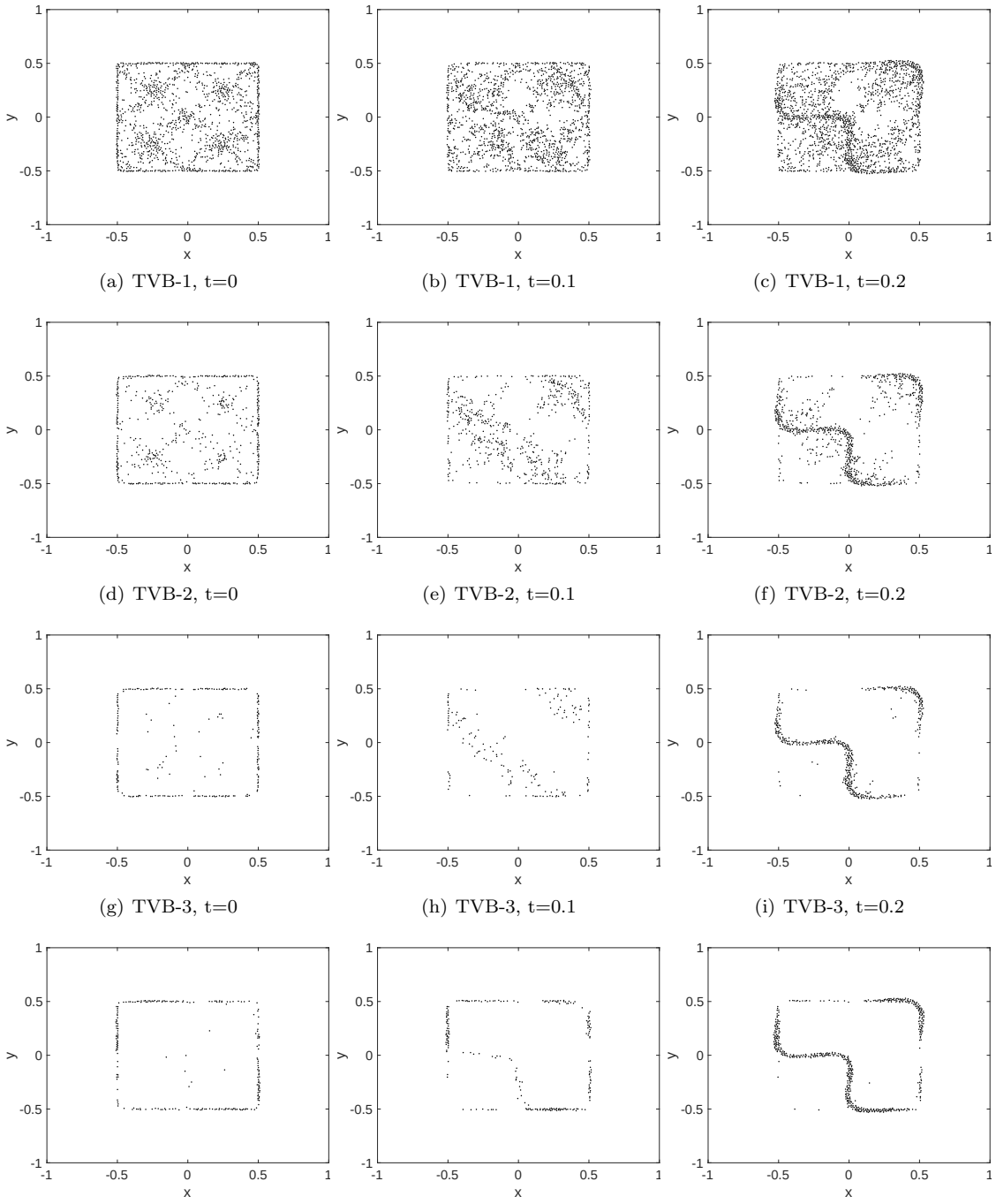


(a) p=1, t=0 (b) p=1, t=0.1 (c) p=1, t=0.2

(d) p=2, t=0 (e) p=2, t=0.1 (f) p=2, t=0.2

(g) p=3, t=0 (h) p=3, t=0.1 (i) p=3, t=0.2

Figure 9: Solution for Burgers equation at $t = 0, 0.1, 0.2$ on U-0.02 using the MLP indicator, with 20 equally spaced contour lines between -1.2 and 1.2.



(a) TVB-1, t=0 (b) TVB-1, t=0.1 (c) TVB-1, t=0.2

(d) TVB-2, t=0 (e) TVB-2, t=0.1 (f) TVB-2, t=0.2

(g) TVB-3, t=0 (h) TVB-3, t=0.1 (i) TVB-3, t=0.2

(j) MLP, t=0 (k) MLP, t=0.1 (l) MLP, t=0.2

Figure 10: Troubled-cells flagged for Burgers equation at $t = 0, 0.1, 0.2$ on U-0.02 and p=3.

In (6.2), ρ, $\mathbf{u} = (v_x, v_y)^\top$ and $\mathcal{P}$ denote the fluid density, velocity and pressure, respectively. The quantity E represents the total energy per unit volume

$$E = \rho e + \frac{1}{2}\rho|\mathbf{u}|^2,$$

where e is the specific internal energy given by a caloric equation of state, $e = e(\rho, \mathcal{P})$. In this work, we choose the equation of state for an ideal gas,

$$e = \frac{\mathcal{P}}{(\gamma - 1)\rho},$$

where $\gamma = c_p/c_v$ is the ratio of specific heats. We choose $\gamma = 1.4$ for our simulations.

6.3.1. 2D Riemann problem: configuration 4

We consider the two-dimensional Riemann problem for the Euler equations [25], given by the initial conditions

$$(\rho,\ v_x,\ v_y,\ \mathcal{P})_0(\boldsymbol{x}) = \begin{cases} (1.1,\ 0,\ 0,\ 1.1), & \text{if } x > 0,\ y > 0, \\ (0.5065,\ 0.8939,\ 0,\ 0.35), & \text{if } x < 0,\ y > 0, \\ (1.1,\ 0.8939,\ 0.8938,\ 1.1), & \text{if } x < 0,\ y < 0, \\ (0.5065,\ 0,\ 0.8939,\ 0.35), & \text{if } x > 0,\ y < 0, \end{cases}$$

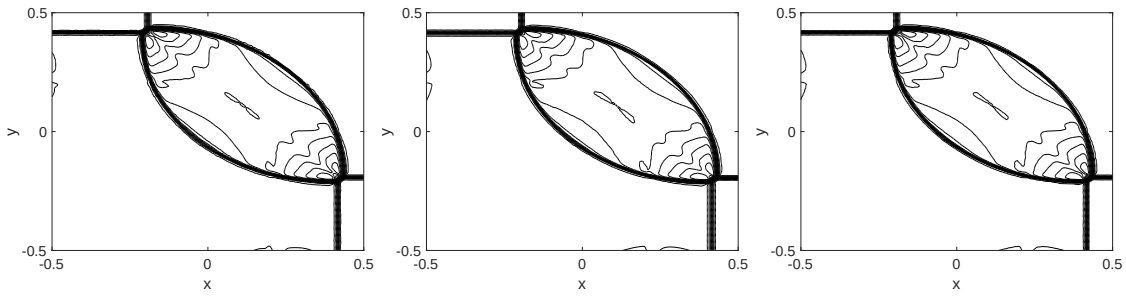
on the domain $[-0.5, 0.5]^2$. The solution of this problem consists of four interacting shock waves. We simulate the results on the mesh S-160 till $T_f = 0.25$, with a fixed time-step $\Delta t = 4 \times 10^{-4} \times w(p)$. Unlike scalar conservation laws, we have several choices for the variables used for detecting troubled-cells, as well as the variables finally limited. In this work, we used the vector of conserved variables $\boldsymbol{u}$ to accomplish both these tasks. As can be seen in Figures 11 and 12, the solution at the final time with the three TVB indicators are almost indistinguishable, with TVB-1 flagging the most number of cell. The MLP indicator, on the other hand, shows a drastic improvement in terms of the complex features captured by the solution, while marking a far fewer cells compared to TVB.

6.3.2. 2D Riemann problem: configuration 6

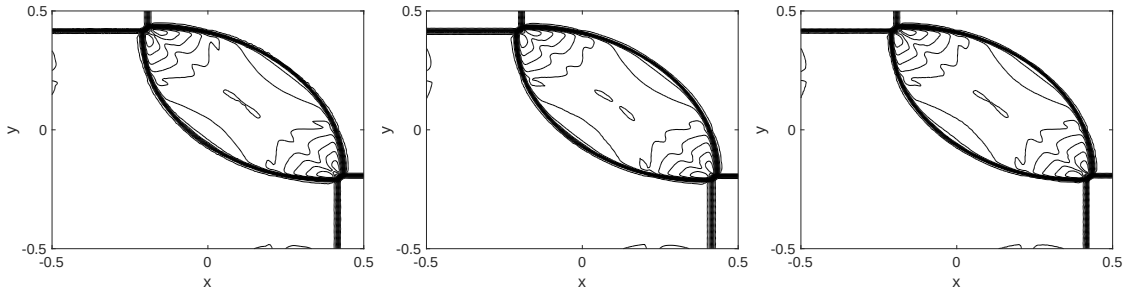
We consider another Riemann problem configuration [25] given by

$$(\rho,\ v_x,\ v_y,\ \mathcal{P})_0(\boldsymbol{x}) = \begin{cases} (1.0,\ 0.75,\ -0.5,\ 1.0), & \text{if } x > 0,\ y > 0, \\ (2.0,\ 0.75,\ 0.5,\ 1.0), & \text{if } x < 0,\ y > 0, \\ (1.0,\ -0.75,\ 0.5,\ 1.0), & \text{if } x < 0,\ y < 0, \\ (3.0,\ -0.75,\ -0.5,\ 1.0), & \text{if } x > 0,\ y < 0, \end{cases}$$

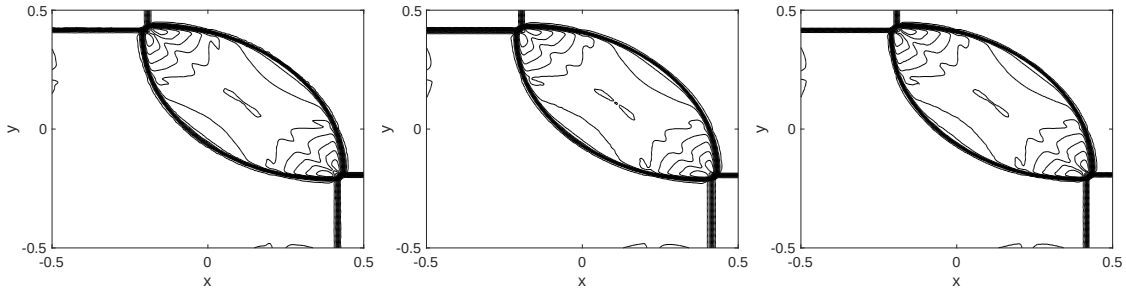
on the domain $[-0.5, 0.5]^2$. The solution of this problem consists of four contact waves. We simulate the results on the mesh S-160 till $T_f = 0.3$, with a fixed time-step $\Delta t = 4 \times 10^{-4} \times w(p)$. Since contact waves are linear waves, it is generally enough to limit the regions near the discontinuity for the initial few time-steps, after which the solution remains smooth and should require almost no limiting. However, the contact waves are smeared by the TVB indicator (see Figure 13) since a large number of cells are flagged though out the evolution of the numerical solution (see Figure 14). On the other hand a minimal number of cells are flagged by the MLP indicator, leading to considerably sharper contact lines.



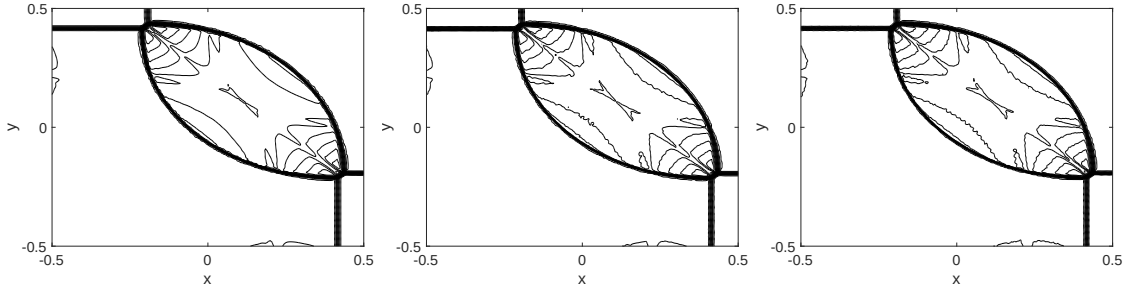
(a) TVB-1, p=1 (b) TVB-1, p=2 (c) TVB-1, p=3



(d) TVB-2, p=1 (e) TVB-2, p=2 (f) TVB-2, p=3

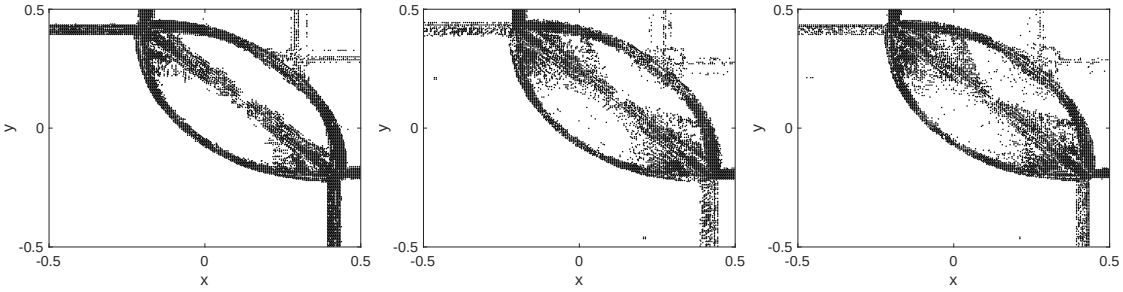


(g) TVB-3, p=1 (h) TVB-3, p=2 (i) TVB-3, p=3

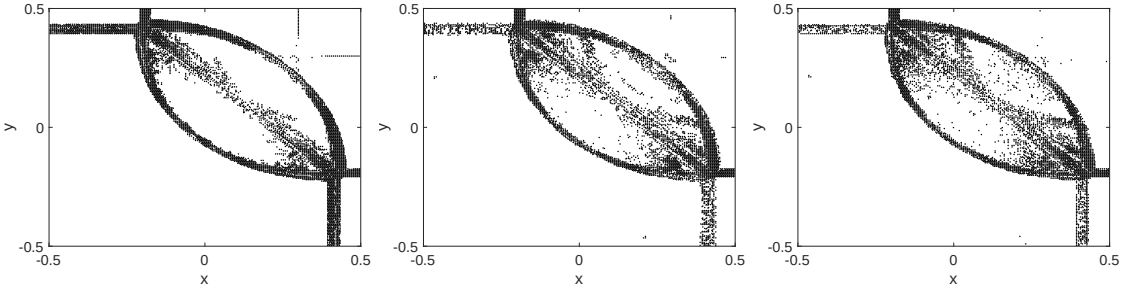


(j) MLP, p=1 (k) MLP, p=2 (l) MLP, p=3

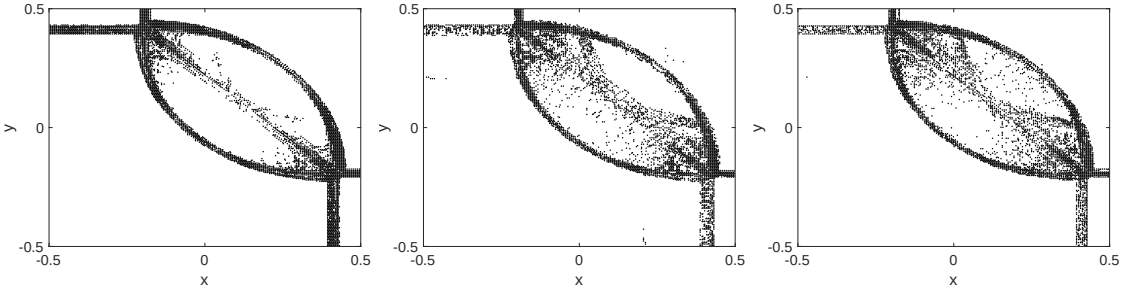
Figure 11: Solution (density) of 2D Riemann problem (config. 4) for the Euler equation at $t = 0.25$ on S-160, with 30 equally spaced contour lines between 0.255 and 1.9.



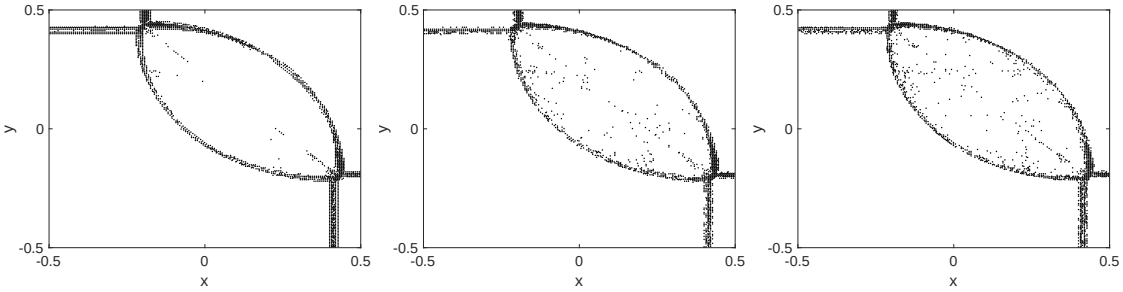
(a) TVB-1, p=1 (b) TVB-1, p=2 (c) TVB-1, p=3



(d) TVB-2, p=1 (e) TVB-2, p=2 (f) TVB-2, p=3

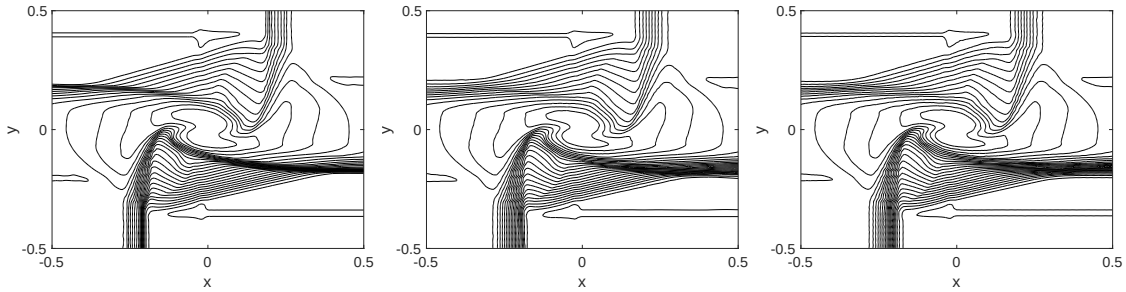


(g) TVB-3, p=1 (h) TVB-3, p=2 (i) TVB-3, p=3

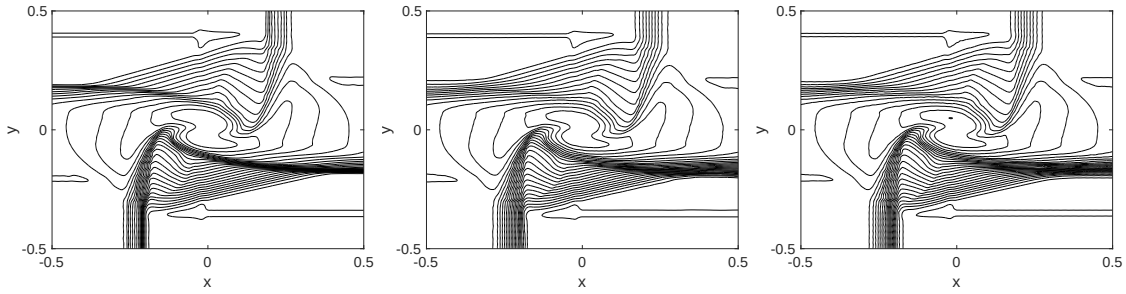


(j) MLP, p=1 (k) MLP, p=2 (l) MLP, p=3

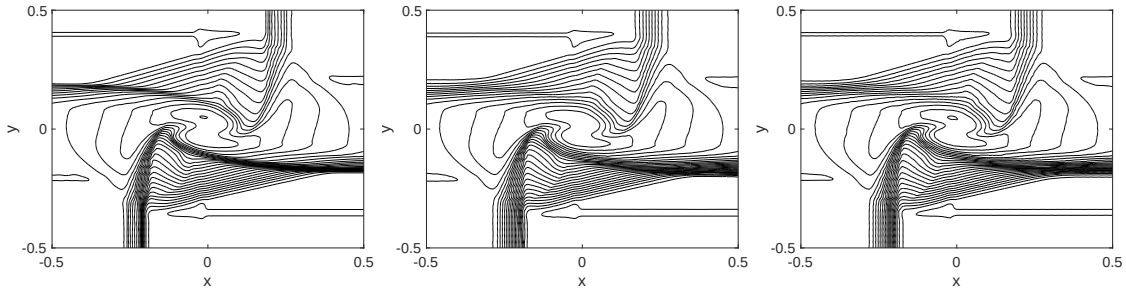
Figure 12: Troubled-cells flagged for 2D Riemann problem (config. 4) at $t = 0.25$ on S-160.



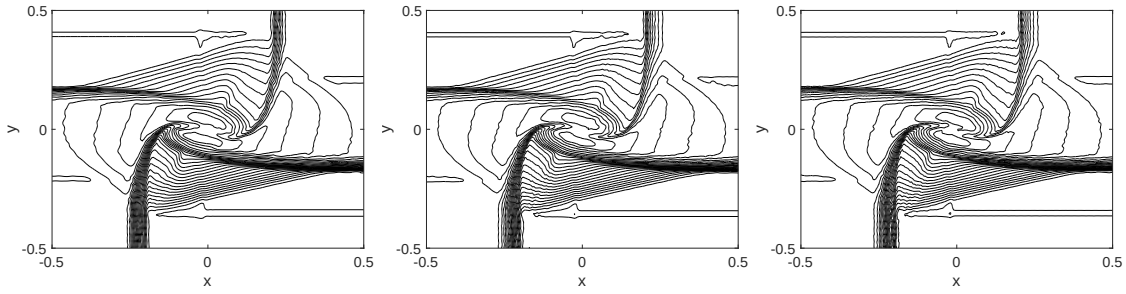
(a) TVB-1, p=1 (b) TVB-1, p=2 (c) TVB-1, p=3



(d) TVB-2, p=1 (e) TVB-2, p=2 (f) TVB-2, p=3

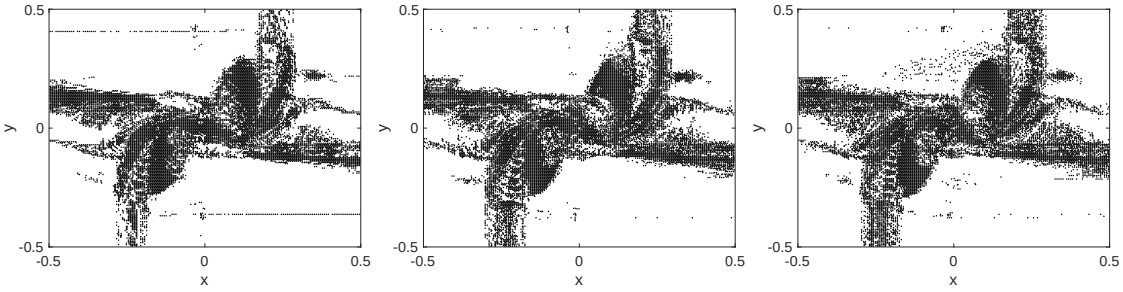


(g) TVB-3, p=1 (h) TVB-3, p=2 (i) TVB-3, p=3

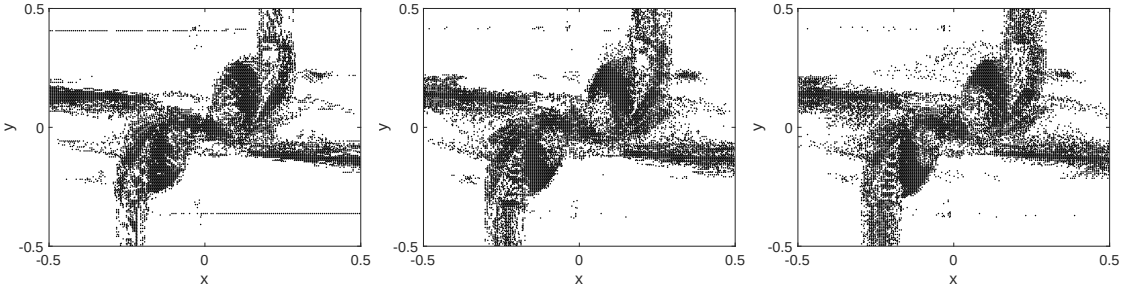


(j) MLP, p=1 (k) MLP, p=2 (l) MLP, p=3

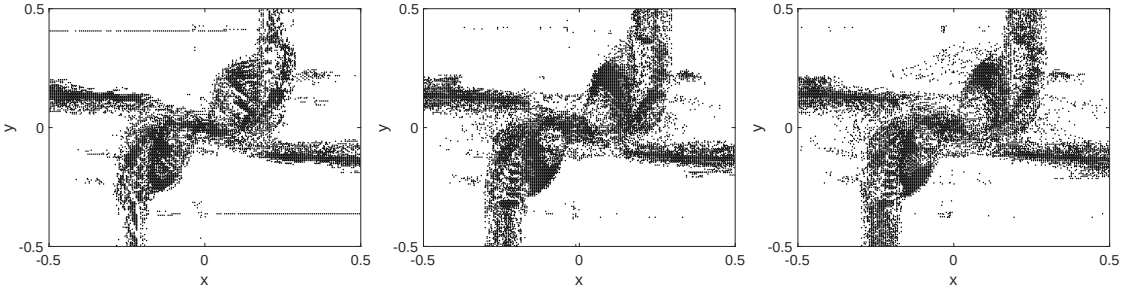
Figure 13: Solution (density) of 2D Riemann problem (config. 6) for the Euler equation at $t = 0.3$ on S-160, with 30 equally spaced contour lines between 0.42 and 3.38.



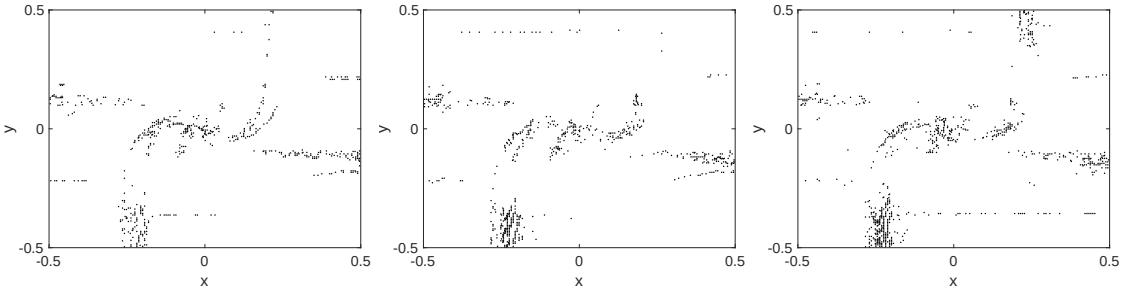
(a) TVB-1, p=1 (b) TVB-1, p=2 (c) TVB-1, p=3



(d) TVB-2, p=1 (e) TVB-2, p=2 (f) TVB-2, p=3



(g) TVB-3, p=1 (h) TVB-3, p=2 (i) TVB-3, p=3



(j) MLP, p=1 (k) MLP, p=2 (l) MLP, p=3

Figure 14: Troubled-cells flagged for 2D Riemann problem (config. 6) at $t = 0.3$ on S-160.

6.3.3. 2D Riemann problem: configuration 12

We finally consider the Riemann problem configuration 12 [25] given by

$$(\rho,\ v_x,\ v_y,\ \mathcal{P})_0(\boldsymbol{x}) = \begin{cases} (0.5313,\ 0,\ 0,\ 0.4), & \text{if } x > 0,\ y > 0, \\ (1.0,\ 0.7276,\ 0,\ 1.0), & \text{if } x < 0,\ y > 0, \\ (0.8,\ 0,\ 0,\ 1.0), & \text{if } x < 0,\ y < 0, \\ (1.0,\ 0,\ 0.7276,\ 1.0), & \text{if } x > 0,\ y < 0, \end{cases}$$

on the domain $[-0.5, 0.5]^2$. The solution of this problem consists of two shock waves and two contact waves. We simulate the results on the mesh S-160 till $T_f = 0.25$, with a fixed time-step $\Delta t = 4 \times 10^{-4} \times w(p)$. As can be seen in Figures 15 and 16, the solution at the final time with the three TVB indicators are almost indistinguishable, with TVB-1 flagging the most number of cell. Neither of the TVB indicators are able to resolve the fine solution structures observed near $\boldsymbol{x} = (0, 0)$. On the other hand, the MLP indicator performs substantially better at capturing the finer structures, while flagging the least number of cells.

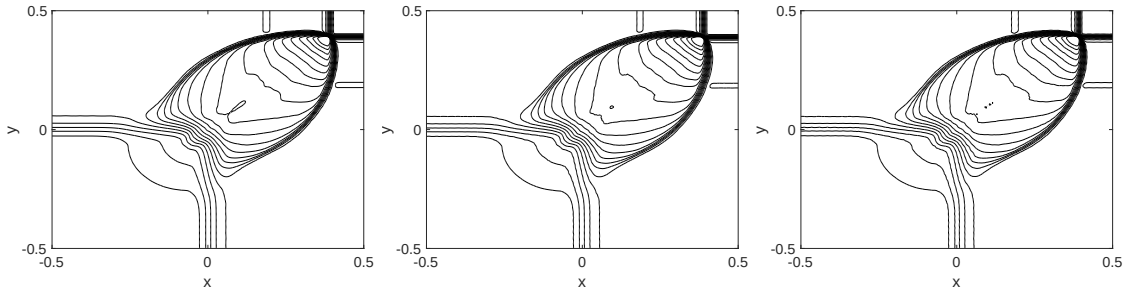
6.3.4. Double Mach reflection

This problem was first described in [38] and is a useful experiment to assess the robustness of numerical methods in the presence of strong shocks. The computational domain for this problem is $[0, 4] \times [0, 1]$, with a ramp beginning at $x = 1/6$ on the bottom boundary (see Figure 17). The initial Mach 10 shock propagating at an angle of 30°, is placed at the base of the ramp. Inflow boundary condition are set on the left and the section of the bottom boundary before the ramp, while outflow boundary conditions are prescribed to the right. A time-dependent boundary condition is prescribed on the top boundary to allow the shock to propagate through the domain, as if the top boundary extends to infinity. The solution is simulated till $T_f = 0.2$ on an unstructured mesh U-0.005, with the fixed local time-step $\Delta t = 2 \times 10^{-5} \times w(p)$. Since the current problem involves a strong shock, we implement an additional positivity fix after limiting the solution, to avoid unphysical values in density and pressure. More precisely, the local solution is replaced by cell averages in those cells where negative values of density or pressure are observed. We compare the results obtained with the MLP indicator and the TVB-3 indicator. Once again, we observe in Figures 18 and 19 that the MLP indicator marks fewer cells while yiedling sharper solution features, as compared to the TVB indicator.

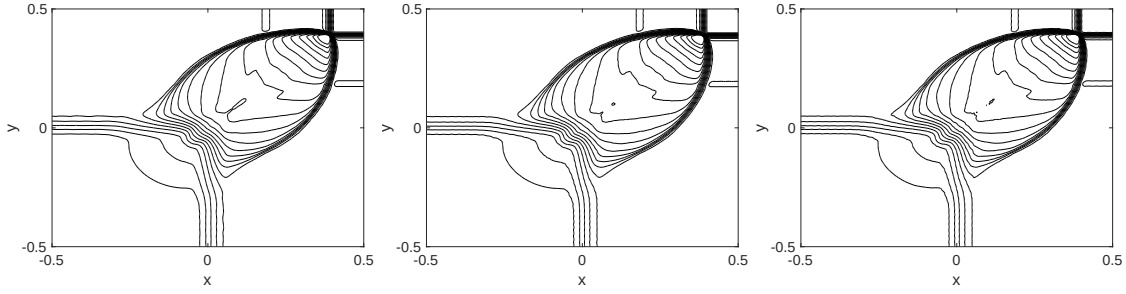
7. Conclusion

In this work, we have extended the approach of [34] and trained a feedforward network to behave as a troubled-cell indicator for two-dimensional problems. The training set is obtained by using a number of suitable functions that characterize the local solution structure for conservation laws. The main advantage of proposed indicator is that it does not require the specification of problem-dependent parameters. Furthermore, the network is trained offline, after which it can be used as a black box to flag troubled-cells for general conservational laws on triangular grids. Through several numerical tests, the MLP indicator has been shown to outperform the TVB indicator (with various values of M) both in terms of solution accuracy and the number of cells flagged, while maintaining a comparable computational cost.

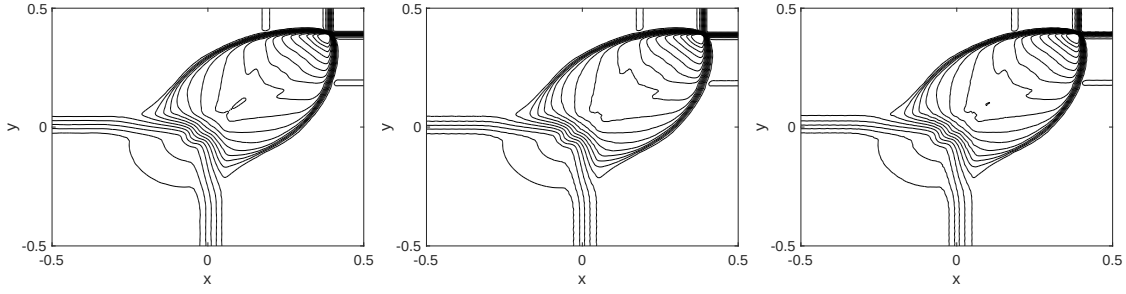
Future work will investigate the construction of similar networks on two-dimensional unstructured quadrilateral grids, as well as the extension to three-dimensional grids.



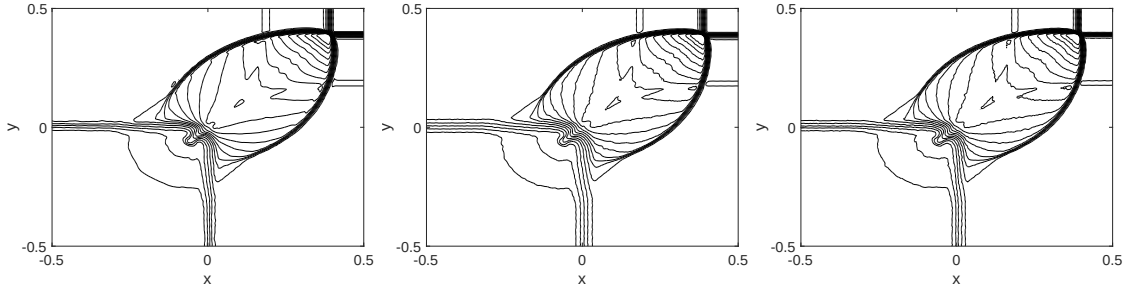
(a) TVB-1, p=1 (b) TVB-1, p=2 (c) TVB-1, p=3



(d) TVB-2, p=1 (e) TVB-2, p=2 (f) TVB-2, p=3

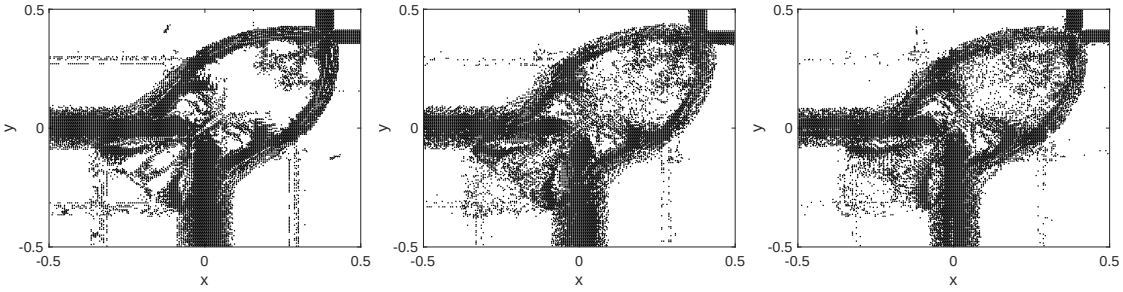


(g) TVB-3, p=1 (h) TVB-3, p=2 (i) TVB-3, p=3

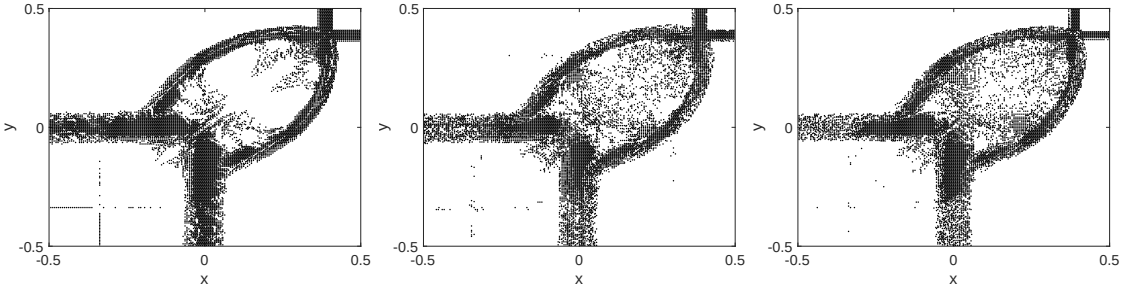


(j) MLP, p=1 (k) MLP, p=2 (l) MLP, p=3

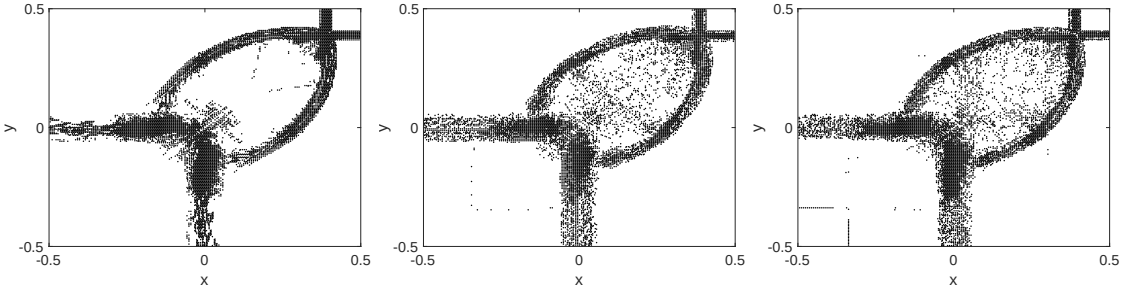
Figure 15: Solution (density) of 2D Riemann problem (config. 12) for the Euler equation at $t = 0.25$ on S-160, with 30 equally spaced contour lines between 0.255 and 1.9.



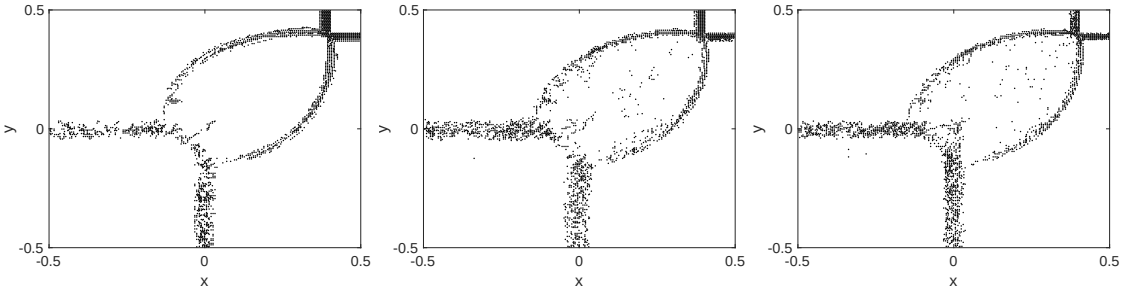
(a) TVB-1, p=1 (b) TVB-1, p=2 (c) TVB-1, p=3



(d) TVB-2, p=1 (e) TVB-2, p=2 (f) TVB-2, p=3



(g) TVB-3, p=1 (h) TVB-3, p=2 (i) TVB-3, p=3



(j) MLP, p=1 (k) MLP, p=2 (l) MLP, p=3

Figure 16: Troubled-cells flagged for 2D Riemann problem (config. 12) at $t = 0.25$ on S-160.

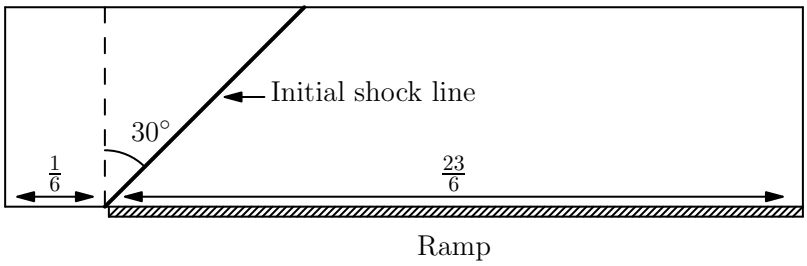


Figure 17: Computational domain for double Mach reflection.

References

[1] Dinshaw S. Balsara, Christoph Altmann, Claus-Dieter Munz, and Michael Dumbser. A sub-cell based indicator for troubled zones in rkdg schemes and a novel class of hybrid rkdg+hweno schemes. *Journal of Computational Physics*, 226(1):586 – 620, 2007.

[2] Timothy Barth and Dennis Jespersen. The design and application of upwind schemes on unstructured meshes. In *27th Aerospace Sciences Meeting*, Aerospace Sciences Meetings. American Institute of Aeronautics and Astronautics, Jan 1989.

[3] Yoshua Bengio. *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[4] Rupak Biswas, Karen D. Devine, and Joseph E. Flaherty. Parallel, adaptive finite element methods for conservation laws. *Applied Numerical Mathematics*, 14(1):255 – 283, 1994.

[5] A. Burbeau, P. Sagaut, and Ch.-H. Bruneau. A problem-independent limiter for high-order runge-kutta discontinuous galerkin methods. *Journal of Computational Physics*, 169(1):111 – 150, 2001.

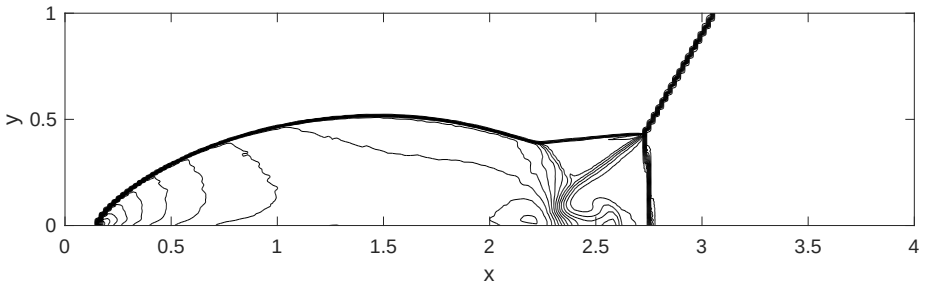
[6] Bernardo Cockburn, George E. Karniadakis, and Chi-Wang Shu. *Discontinuous Galerkin Methods: Theory, Computation and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[7] Bernardo Cockburn, San-Yih Lin, and Chi-Wang Shu. Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws iii: One-dimensional systems. *Journal of Computational Physics*, 84(1):90 – 113, 1989.

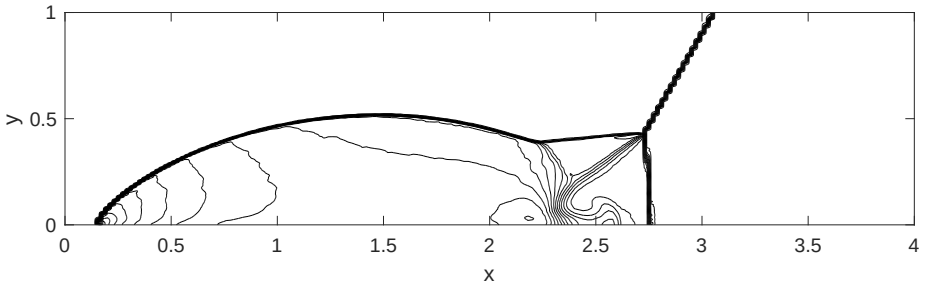
[8] Bernardo Cockburn and Chi-Wang Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. *Math. Comp.*, 52(186):411–435, 1989.

[9] Bernardo Cockburn and Chi-Wang Shu. The runge-kutta discontinuous galerkin method for conservation laws v: Multidimensional systems. *Journal of Computational Physics*, 141(2):199 – 224, 1998.

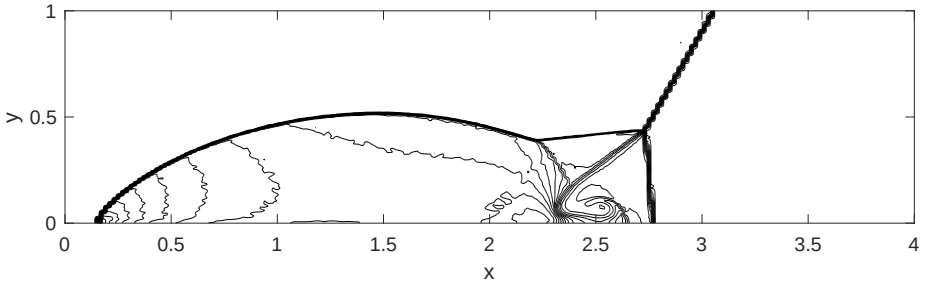
[10] Bernardo Cockburn and Chi-Wang Shu. Runge–kutta discontinuous galerkin methods for convection-dominated problems. *Journal of Scientific Computing*, 16(3):173–261, Sep 2001.



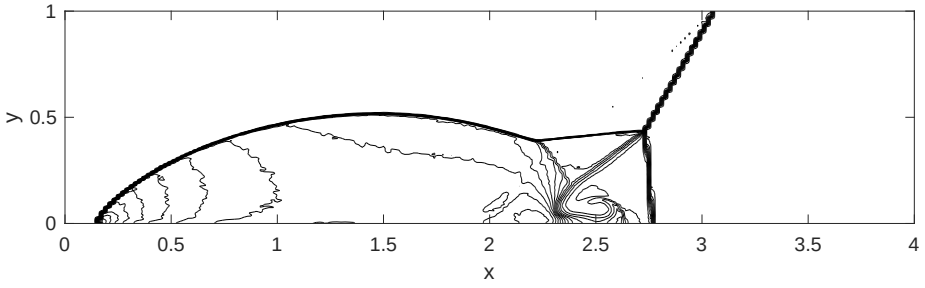
(a) TVB-3, p=1



(b) TVB-3, p=2

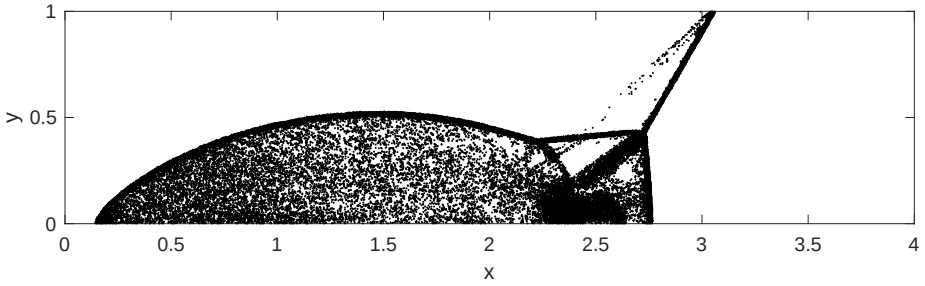


(c) MLP, p=1

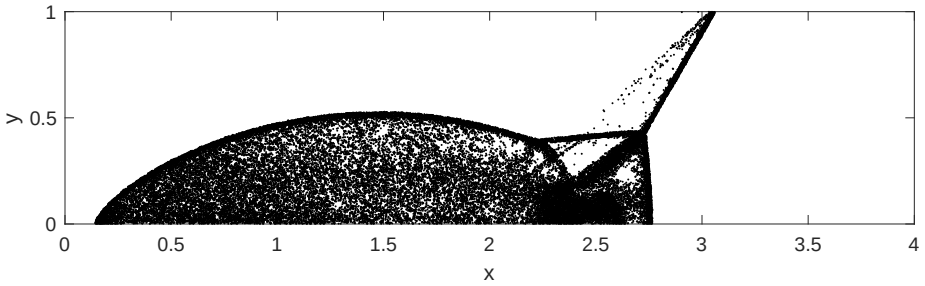


(d) MLP, p=2

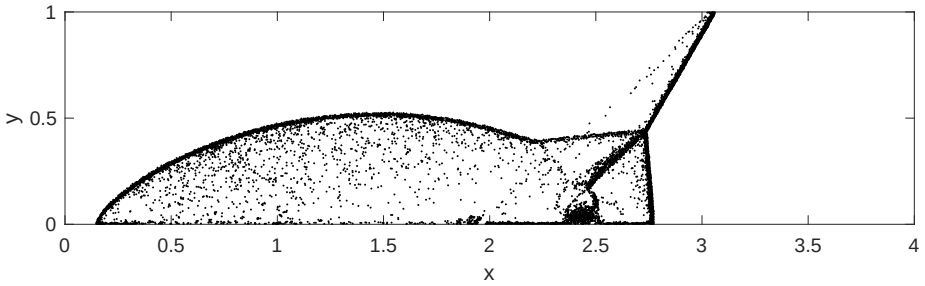
Figure 18: Solution (density) for the double Mach problem at $t = 0.2$ on S-0.005, with 30 equally spaced contour lines between 1.5 and 21.5.



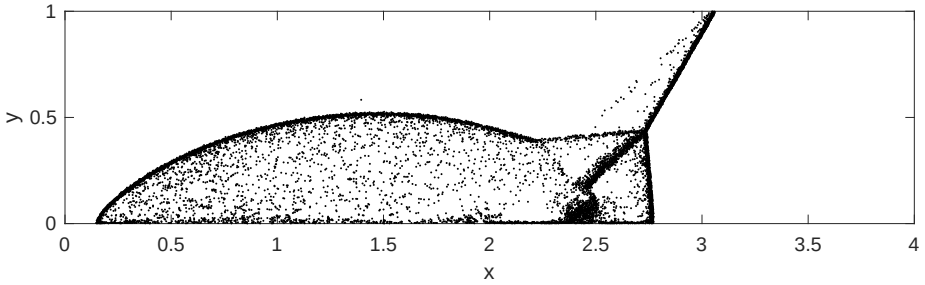
(a) TVB-3, p=1



(b) TVB-3, p=2



(c) MLP, p=1



(d) MLP, p=2

Figure 19: Troubled-cells flagged for the double Mach problem at $t = 0.2$ on S-0.005.

[11] Paul Dan Cristea. *Application of Neural Networks In Image Processing and Visualization*, pages 59–71. Springer Netherlands, Dordrecht, 2009.

[12] Constantine M. Dafermos. *Hyperbolic conservation laws in continuum physics*, volume 325 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, third edition, 2010.

[13] Gülin Dede and Murat Hüsnü Sazli. Speech recognition with artificial neural networks. *Digital Signal Processing*, 20(3):763 – 768, 2010.

[14] Michael Dumbser, Olindo Zanotti, Raphal Loubre, and Steven Diot. A posteriori subcell limiting of the discontinuous galerkin finite element method for hyperbolic conservation laws. *Journal of Computational Physics*, 278:47 – 75, 2014.

[15] Guosheng Fu and Chi-Wang Shu. A new troubled-cell indicator for discontinuous galerkin methods for hyperbolic conservation laws. *Journal of Computational Physics*, 347:305 – 327, 2017.

[16] Slawomir Golak. *A MLP Solver for First and Second Order Partial Differential Equations*, pages 789–797. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[17] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Rev.*, 43(1):89–112 (electronic), 2001.

[18] Ami Harten. Eno schemes with subcell resolution. *Journal of Computational Physics*, 83(1):148 – 184, 1989.

[19] Jan S. Hesthaven. *Numerical methods for conservation laws*, volume 18 of *Computational Science & Engineering*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2018. From analysis to algorithms.

[20] Jan S. Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods*, volume 54 of *Texts in Applied Mathematics*. Springer, New York, 2008. Algorithms, analysis, and applications.

[21] Tompson J., Schlachter K., Sprechmann P., and Perlin K., editors. *Accelerating Eulerian Fluid Simulation With Convolutional Networks*. PMLR, 2017.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[23] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, and J.E. Flaherty. Shock detection and limiting with discontinuous galerkin methods for hyperbolic conservation laws. *Applied Numerical Mathematics*, 48(3):323 – 338, 2004. Workshop on Innovative Time Integrators for PDEs.

[24] Lilia Krivodonova. Limiters for high-order discontinuous galerkin methods. *Journal of Computational Physics*, 226(1):879 – 896, 2007.

[25] Alexander Kurganov and Eitan Tadmor. Solution of two-dimensional riemann problems for gas dynamics without riemann problem solvers. *Numerical Methods for Partial Differential Equations*, 18(5):584–608, 2002.

[26] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *Trans. Neur. Netw.*, 9(5):987–1000, September 1998.

[27] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-net: Learning PDEs from data. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3208–3216, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[28] Steven J. Nowlan and Geoffrey E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Comput.*, 4(4):473–493, July 1992.

[29] Jianxian Qiu and Chi-Wang Shu. Hermite weno schemes and their application as limiters for runge-kutta discontinuous galerkin method: one-dimensional case. *Journal of Computational Physics*, 193(1):115 – 135, 2004.

[30] Jianxian Qiu and Chi-Wang Shu. A comparison of troubled-cell indicators for Runge-Kutta discontinuous Galerkin methods using weighted essentially nonoscillatory limiters. *SIAM J. Sci. Comput.*, 27(3):995–1013, 2005.

[31] Jianxian Qiu and Chi-Wang Shu. Hermite weno schemes and their application as limiters for runge-kutta discontinuous galerkin method ii: Two dimensional case. *Computers & Fluids*, 34(6):642 – 663, 2005.

[32] Jianxian Qiu. and Chi-Wang Shu. Runge–kutta discontinuous galerkin method using weno limiters. *SIAM Journal on Scientific Computing*, 26(3):907–929, 2005.

[33] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125 – 141, 2018.

[34] Deep Ray and Jan S. Hesthaven. An artificial neural network as a troubled-cell indicator. *Journal of Computational Physics*, 367:166 – 191, 2018.

[35] Keith Rudd and Silvia Ferrari. A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 155(Supplement C):277 – 285, 2015.

[36] M. Vuik and J. Ryan. Automated parameters for troubled-cell indicators using outlier detection. *SIAM Journal on Scientific Computing*, 38(1):A84–A104, 2016.

[37] T. Warburton. An explicit construction of interpolation nodes on the simplex. *Journal of Engineering Mathematics*, 56(3):247–262, Nov 2006.

[38] Paul Woodward and Phillip Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*, 54(1):115 – 173, 1984.