Controlling oscillations in spectral methods by local artificial viscosity governed by neural networks

Lukas Schwander[a], Deep Ray[b], Jan S. Hesthaven[c]

[a]*ETH Zürich*
CH-8092 Zürich, Switzerland
lukas.schwander@alumni.ethz.ch
[b]*Department of Computational and Applied Mathematics*
Rice University, Houston, TX, USA
deep.ray@rice.edu
[c]*Computational Mathematics and Simulation Science (MCSS)*
Institute of Mathematics
École Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
jan.hesthaven@epfl.ch

Abstract

While a nonlinear viscosity is used widely to control oscillations when solving conservation laws using high-order elements based methods, such techniques are less straightforward to apply in global spectral methods as a local estimate of solution regularity generally is required. In this work we demonstrate how to train and use a local artificial neural network to estimate the local solution regularity and demonstrate the efficiency of nonlinear artificial viscosity methods based on this in the context of Fourier spectral methods. We compare with entropy viscosity techniques and illustrate the promise of the neural network based estimators when solving one- and two-dimensional conservation laws, including the Euler equations.

1. Introduction

While the benefits of using high-order methods for solving time dependent partial differential equations are well known, see e.g. [7] and references therein, it is likewise well known that such methods generate artificial numerical oscillations when non-smooth solutions are approximated. Since high-order methods are particularly beneficial for solving conservation laws, this presents a challenge.

For element based high-order methods, e.g., spectral element and discontinuous Galerkin methods, several methods based on the addition of a nonlinear viscosity have been proposed. Most of these rely on a local estimate of the solution regularity based on the decay of the polynomial expansion coefficients on each element [16, 11, 23]. An alternative utilizes the concept of entropy production as an indicator of the regularity of the solution to estimate a local viscosity and control the numerical oscillations [6]. A recent work combines past ideas and techniques to train an artificial neural network to locally estimate the viscosity and identify the most appropriate model [2].

However, for global methods such as spectral methods or reduced basis methods, it is less obvious how to apply these techniques as there is no immediate way to extract an estimate of the local solution regularity. Standard filter techniques [8] and spectrally vanishing viscosity [15] have a global effect and tend to over-dissipate high frequency smooth features. The only known option is the entropy viscosity [6] which has the disadvantage of having several free parameters that may require tuning.

In this work we propose to use an artificial neural network to estimate the local regularity of the solution in physical space. We focus on the spectral Fourier methods but there is nothing specific about this choice and other families of spectral methods can likewise be considered. A feedforward multiperceptron network is trained on canonical families of functions and the resulting network is universal and used for all one- and two-dimensional examples, including the Euler equations, to demonstrate the flexibility and efficiency of the proposed method.

What remains of the paper is organized as follows. In Section 2 we introduce conservation laws, Fourier spectral methods, and the concept of non-linear viscosity, all of which are central components of the proposed work. In Section 3 we offer a brief overview of feedforward artificial neural networks, setting the stage for Section 4 in which we discuss how to train a network as a local smoothness classifier in the current context. In Section 5 we offer a range of one- and two-dimensional examples of the performance of the nonlinear viscosity for problems with discontinuous solutions. Section 6 concludes with a few summarizing remarks and suggestions for future work.

2. Governing model and its discretization

Let $\Omega \in \mathbb{R}^d$ be a bounded domain. Consider the convection-diffusion equation

$$\frac{\partial \boldsymbol{u}(\boldsymbol{x},t)}{\partial t} + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}(\boldsymbol{x},t)) = \nabla \cdot \boldsymbol{g}(\boldsymbol{u}(\boldsymbol{x},t);\mu), \quad \forall\ (\boldsymbol{x},t) \in \Omega \times (0,T], \tag{2.1}$$

where $\boldsymbol{u} : \Omega \times [0,T] \mapsto \mathbb{R}^m$ is the vector of unknowns, $\boldsymbol{f} : \mathbb{R}^m \mapsto \mathbb{R}^m \times \mathbb{R}^d$ is the smooth convective flux, while $\boldsymbol{g} : \mathbb{R}^m \mapsto \mathbb{R}^m \times \mathbb{R}^d$ is the viscous flux controlled by the viscosity coefficient $\mu := \mu(\boldsymbol{u}(\boldsymbol{x},t))$. In the absence of viscosity, i.e., $\mu = 0$, solutions to (2.1) can develop discontinuities. This is particularly true for hyperbolic conservation laws which are characterised by the flux Jacobian $\nabla_{\boldsymbol{u}}(\boldsymbol{f}(\boldsymbol{u}) \cdot \boldsymbol{n})$ having real eigenvalues and a complete basis of eigenvectors for any $\boldsymbol{n} \in S^3$. The presence of discontinuities makes it challenging to approximate the solution using high-order numerical methods due to the appearance of Gibbs oscillations. Augmenting the hyperbolic conservation law with a viscous term provides a powerful mechanism to artificially inject diffusion to locally control the spurious oscillations. A commonly used form of the viscous flux is

$$\boldsymbol{g} = \mu(\boldsymbol{u})\nabla \boldsymbol{u}, \tag{2.2}$$

which is also the choice we adhere to for the rest of this work. The naturally arising key question is how to choose $\mu(\boldsymbol{u})$ in a robust way to eliminate oscillations in the neighborhood of a discontinuity while leaving smooth features of the solution unchanged.

2.1. Fourier collocation method

We solve the system (2.1) by using the Fourier collocation method for the spatial discretization and prescribe periodic boundary conditions with a suitable initial condition. For convenience, we describe the scheme for the scalar one-dimensional problem

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right), \tag{2.3}$$

defined on the domain $\Omega = [0, 2\pi]$. We introduce N uniformly placed collocation points $x_j = (2\pi j)/N$ for $j = 0, 1, ..., N-1$, where N is taken to be a positive even integer and let $u(x)$ be a 2π periodic function with values $u(x_j) = u_j$. By the discrete Fourier transform (DFT), we define the band-limited interpolant $\mathcal{I}_N$ (see [22] for details) to approximate $u(x)$

$$\mathcal{I}_N u(x) = \frac{1}{2\pi} \sum_{k=-N/2}^{N/2}{}' \hat{u}_k e^{ikx}, \tag{2.4}$$

where the prime indicates that the terms $k = \pm N/2$ are multiplied by $1/2$. The Fourier coefficients in (2.4) are given by the Fourier integral approximated by the trapezoidal integration as

$$\hat{u}_k = \frac{2\pi}{N} \sum_{j=0}^{N-1} u_j e^{-ikx_j} \quad \text{for } -\frac{N}{2} + 1 \leqslant k \leqslant \frac{N}{2} \quad \text{and } \hat{u}_{-N/2} = \hat{u}_{N/2}. \tag{2.5}$$

Combining (2.4) and (2.5), we obtain an equivalent expression through the periodic Cardinal function of the form

$$\mathcal{I}_N u(x) = \sum_{j=0}^{N-1} u_j \phi_j(x - x_j), \quad \phi_j(x) = \frac{1}{N} \frac{\sin\left(\frac{Nx}{2}\right)}{\tan\left(\frac{x}{2}\right)}. \tag{2.6}$$

We can approximate the derivative of $u(x)$ at the collocation points by differentiating (2.6) and evaluating it at the collocation points

$$\frac{\mathrm{d}}{\mathrm{d}x} u(x)\bigg|_{x_l} \approx \frac{\mathrm{d}}{\mathrm{d}x} \mathcal{I}_N u(x)\bigg|_{x_l} = \sum_{j=0}^{N-1} u_j \frac{\mathrm{d}}{\mathrm{d}x} \phi_j(x - x_j)\bigg|_{x_l} = \sum_{j=0}^{N-1} D_{lj} u_j,$$

where the $N \times N$ differentiation matrix $\boldsymbol{D}$ is given as

$$D_{lj} = \begin{cases} 0, & \text{if } l = j, \\ \frac{(-1)^{l+j}}{2} \cot\left(\frac{(l-j)\pi}{N}\right) & \text{if } l \neq j. \end{cases} \tag{2.7}$$

Let $u_h(x_j, t)$ denote the approximate solution of (2.3) at the collocation point x_j and time t. We assume that the interpolant (2.6) approximates the solution on the whole domain

$$u(x, t) \approx u_h(x, t) = \sum_{j=0}^{N-1} u_h(x_j, t) \phi_j(x), \quad x \in \Omega.$$

The semi-discrete scheme for (2.3) is obtained by requiring that the residual, obtained by inserting the assumed solution into the equation, vanishes at the collocation points to recover the scheme

$$\frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{u}_h(t) + \boldsymbol{D} \boldsymbol{f}_h(t) = \boldsymbol{D}(\boldsymbol{\mu}_h(t) \odot \boldsymbol{D} \boldsymbol{u}_h(t)), \tag{2.8}$$

where

$$\boldsymbol{u}_h(t) = (u_h(x_0, t), ..., u_h(x_{N-1}, t))^\top,$$

$$\begin{aligned}\boldsymbol{f}_h(t) &= (f(u_h(x_0,t)),...,f(u_h(x_{N-1},t)))^\top,\\ \boldsymbol{\mu}_h(t) &= (\mu_h(x_0,t),...\mu(x_{N-1},t))^\top,\end{aligned}$$

and we define the point-wise multiplication operator $\boldsymbol{x} \odot \boldsymbol{y} = (x_0y_0, ..., x_{N-1}y_{N-1})^\top$.

The system (2.8) can be solved using any reasonable time-marching strategy, such as Runge-Kutta schemes. In this work, we use a 4-th order strong stability preserving Runge-Kutta (SSPRK-4) scheme [5, 4, 17]. The time-step Δt is chosen adaptively based on the following CFL condition

$$\Delta t \leqslant CFL \, \frac{1}{\pi(\lambda_{\max}/h + \mu_{\max}/h^2)}, \tag{2.9}$$

where the CFL>0, $\lambda_{\max} = \max(|f'(u_i)|)$ is the maximum wave velocity and $\mu_{\max} = \max(\mu_i)$.

2.2. Exponential filters

The idea of filtering has its origin in signal processing and involves the modification of specific Fourier components. Low pass filters are used to damp the high-frequency modes, thereby making the signal less noisy. Thus, this strategy can also be used to partially control spurious Gibbs oscillations in Fourier methods. A widely used filter is the exponential filter [7] of order p given by

$$\sigma_p(\eta) = \begin{cases} 1, & \text{if } 0 \leqslant \eta \leqslant \eta_c = \frac{N_c}{N}, \\ \exp\left(-\beta\left(\frac{\eta-\eta_c}{1-\eta_c}\right)^p\right), & \text{if } \eta_c \leqslant \eta \leqslant 1, \\ 0, & \text{if } \eta > 1, \end{cases} \tag{2.10}$$

where $\beta \geqslant 0$ controls the degree of the damping, while $0 \leqslant N_c \leqslant N$ denotes the filter cutoff below which the Fourier coefficients are left unchanged. Using such a filter, we modify the Fourier coefficients as

$$\hat{u}_k \longrightarrow \sigma\left(\frac{|k|}{N/2}\right)\hat{u}_k \quad \text{for } -\frac{N}{2} \leqslant k \leqslant \frac{N}{2}. \tag{2.11}$$

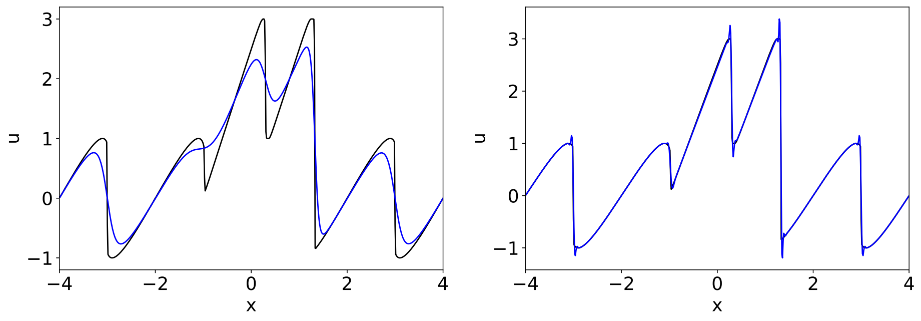
Filtering is a global approach, i.e., it influences the signal everywhere. Using a low-order filter, e.g., $p = 2$ in (2.10), can dampen spurious oscillations but lead to excessive smearing near smooth extrema. On the other hand, a higher-order filter is able to retain good resolution in smooth regions but unable to eliminate the oscillations, see e.g. [7] for a detailed discussion of this.

As an example, consider the solution to the scalar Burgers' equation with a non-linear flux $f(u) = u^2/2$ and the initial condition (5.1). Figure 1 shows the effect of using either a low- and high-order filter.

Note that while high-order filters are unable to remove the oscillations, they localize these near the discontinuities in the solution. Thus, we seek to use a combination of high-order filters and localized artificial viscosity to control the spurious features.

Remark 2.1. *In order to apply the exponential filter with the Fourier spectral scheme* (2.8)*, we first obtain the Fourier coefficients $\hat{u}_k$ from the nodal values u_j using* (2.5)*, modify the coefficients using* (2.11) *and then transform back to the physical domain using the inverse DFT*

$$u_j(t) = \frac{1}{2\pi}\sum_{k=-N/2}^{N/2}{}' \hat{u}(t)_k e^{ikx_j}.$$



(a) $p = 2$.

(b) $p = 4$.

Figure 1: Solution for the Burgers' equation using exponential filters. The initial condition is taken to be (5.1) and $N = 600$.

2.3. Artificial viscosity

The specification of the artificial viscosity $\boldsymbol{\mu}_h$ in the scheme (2.8) is a key component of the scheme and needs to be defined in such a way that in the neighborhood of discontinuities, adequate viscosity is applied to control the Gibbs oscillations, whereas the artificial viscosity should vanish in the neighborhood of smooth features of the solution. Traditionally, the definition of the local nonlinear viscosity is based on an estimate of the local regularity of the solution. However, it is not immediately clear how such local information can be recovered in global methods such as spectral methods or projection based reduced order models [9].

One approach to accomplish this is proposed in [6] and is based on the entropy production in the neighborhood of shocks. For the scalar conservation law (2.3), we assume that there exists an entropy pair $(\eta(u), \psi(u))$ which satisfies

$$\frac{\partial \eta}{\partial t} + \frac{\partial \psi}{\partial x} \leqslant 0,$$

in a weak sense and define the entropy residual

$$R(u_h) = \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}\left(\psi(u_h) - \mu \frac{\partial \eta}{\partial x}\right) + \mu \eta'' \left(\frac{\partial u_h}{\partial x}\right)^2.$$

Following [6] the artificial viscosity is defined as

$$\mu = \min\left(\mu_{max}, \alpha h \frac{|R(u_h)|}{N(u_h)}\right), \tag{2.12}$$

where h is the grid size over the domain of size L, and the normalization factor $N(u)$ is defined as

$$N(u_h) = \frac{1}{2}\frac{\max|u_h|^2 - \min|u_h|^2}{L},$$

for the standard entropy of $\eta(u) = \frac{1}{2}u^2$.

The maximum viscosity, $\mu_{max} = \alpha_{max} h \max|f'(u_h)|$ and all parameters (α, α_{max}) are tunable of $\mathcal{O}(1)$. This model is the only previously considered model to enable a locally defined nonlinear viscosity and we shall use that as a benchmark to highlight the performance of the proposed techniques.

In this work, we assume that the viscosity at the node x_j is evaluated using the solution on the compact stencil $\mathcal{S}_j^r = \{x_{j-r}, ..., x_{j+r}\}$ with $2r-1$ points. In other words,

$$\mu_j = \mu_h(x_j, t) = \mu(u_h(x_{j-r}, t), ..., u_h(x_{j+r,t})).$$

Inspired by the viscosity formulation for discontinuous Galerkin schemes in [11] (see also [24]), we prescribe the point-wise viscosity as

$$\mu_j = \mathcal{Q}(\tau_j) h \max_{x \in \mathcal{S}_j^r} |f'(u)|, \quad \mathcal{Q}(\tau_j) = \begin{cases} 0.5, & \text{if } \tau_j = 1, \\ 0.25, & \text{if } \tau_j = 2, \\ 0.0, & \text{if } \tau_j \geqslant 3, \end{cases} \tag{2.13}$$

where τ_j is an estimate of the smoothness of the solution in the stencil S_j^r. We assign

$$\tau = \begin{cases} 1 & \text{if } u \text{ is discontinuous,} \\ 2 & \text{if } u \in C^0 \setminus C^1, \\ 3 & \text{if } u \in C^1 \setminus C^2, \\ 4 & \text{if } u \in C^2 \setminus C^3, \\ ... \end{cases}$$

The central element in this approach is the need to estimate τ_j directly from the solution on the local stencil $\mathcal{S}_j^r$. In Section 4, we discuss how this can be achieved using feedforward networks.

As the estimate of the vsconsity is local, it may have sudden jumps in its value, which lead to an increased level of numerical oscillations in the solution. To address this, we perform a global smoothening as

$$\mu_j^* = 0.25\mu_{j-1} + 0.5\mu_j + 0.25\mu_{j+1}, \tag{2.14}$$

which is finally used in the scheme (2.8).

While the above discussion focuses on the a one-dimensional scalar problems, the extension to multi-dimensional problems through a tensor product formulation is immediate with the viscosity determined in a dimension-by-dimension approach. For systems of equations, the analysis is performed on one conserved variable, typically the density or, in the case of the Euler equations, the Mach number is also found to be a suitable alternative.

3. Feedforward networks

As mentioned in Section 2.3, we seek to construct a function that analyzes the solution on a local stencil and estimates the local regularity τ, i.e., this is an example of classification problem. Consider a general function $\mathcal{C}$ which classifies the members of a set $\mathcal{B} \subset \mathbb{R}^m$ into n classes

$$\mathcal{C}(\boldsymbol{X}) = \boldsymbol{Y} \in \mathbb{R}^n, \quad \boldsymbol{X} \in \mathcal{B} \quad \text{such that} \quad Y_k = \begin{cases} 1 & \text{if } \boldsymbol{X} \text{ belongs to class } k, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Functions of the form (3.1) are routinely approximated using deep neural networks.

A feedforward network or multilayer perceptron (MLP) is a particular type of deep network in which computing units (neurons) are organized in a sequence of layers. The first layer is called the *source layer* and is responsible for providing an input signal to the network. The last layer is called the output layer while all remaining layers in between are referred to as *hidden layers.* An MLP with depth K consists of $K-1$ hidden layers and one output layer. The action of the network on an input vector $\boldsymbol{X} \in \mathbb{R}^m$ can be expressed as

$$\hat{\boldsymbol{Y}} = \mathcal{F} \circ \mathcal{G}_k \circ \mathcal{A} \circ \mathcal{G}_{k-1} \circ \mathcal{A} \circ \mathcal{G}_{k-2} \circ ... \circ \mathcal{A} \circ \mathcal{G}_1(\boldsymbol{X}) \tag{3.2}$$

Here $\mathcal{G}_k$ represents the affine transformation performed by the k-th layer on a vector $\boldsymbol{Z} \in \mathbb{R}^{N_{k-1}}$,

$$\mathcal{G}_k(\boldsymbol{Z}) = \boldsymbol{W}_k \boldsymbol{Z} + \boldsymbol{b}_k, \quad \boldsymbol{W}_k \in \mathbb{R}^{N_k \times N_{k-1}},\ \boldsymbol{b}_k \in \mathbb{R}^{N_k}, \tag{3.3}$$

where the weights matrix $\boldsymbol{W}_k$ and the bias vector $\boldsymbol{b}_k$ are trainable parameters. The signal vector from each layer passes through a non-linear activation function, which acts on the vector component-wise. A wide variety of activation functions have been designed for various machine learning problems [3]. In the present work, we use the exponential linear unit (ELU)

$$\mathcal{A}(x;\alpha) = \begin{cases} x, & \text{if } x > 0, \\ \alpha(e^x - 1), & \text{if } x \leqslant 0, \end{cases} \tag{3.4}$$

which has been shown to perform well for classification tasks [1]. Finally, $\mathcal{F}$ is a non-linear output function that transforms the output into a usable form. For classification problems, it is customary to use the softmax function

$$\hat{Y}_l = \mathcal{F}(\boldsymbol{Z})_l = \frac{e^{Z_l}}{\sum_j e^{Z_j}}. \tag{3.5}$$

Note that $\hat{Y}_l \in [0,1]$ and the components of $\hat{\boldsymbol{Y}}$ sum to one. Thus, the l-th component of the output vector $\hat{\boldsymbol{Y}}$ can be viewed as the probability that the input $\boldsymbol{X}$ belongs to the class l. To be consistent with target function (3.1), we set $N_0 = m$ and $N_K = n$.

To approximate the function (3.1) with the network (3.2), we need to suitable define the parameters $\theta = \{\boldsymbol{W}_k, \boldsymbol{b}_k\}_k$. This is achieved by first creating a training dataset with $N_{\mathbb{T}}$ labelled samples

$$\mathbb{T} = \{(\boldsymbol{X}^i, \boldsymbol{Y}^i) : \boldsymbol{Y}^i = \mathcal{C}(\boldsymbol{X}^i)\ \forall\ i = 1, ..., N_{\mathbb{T}}\},$$

and then define a suitable cost function that measures the discrepancy between the actual class vector $\boldsymbol{Y}$ and the predicted class vector $\hat{\boldsymbol{Y}}$. The cross-entropy function [3]

$$\mathcal{L}(\theta) = -\frac{1}{N_{\mathbb{T}}} \sum_{i=1}^{N_{\mathbb{T}}} \sum_{j=1}^{n} Y_j^i \log(\hat{Y}_j^i), \tag{3.6}$$

is a commonly used loss function for classification problems. To train the neural network, we need to find the parameters θ that minimizes the loss function over the training set. This minimization can be performed using an iterative optimization algorithm, such as stochastic gradient descent, AdaGrad or ADAM [10].

If one is not careful, the trained network can overfit the training set $\mathbb{T}$. This can lead to a high *generalization error*, i.e., the network performs poorly on data outside the training set. Thus, suitable regularization techniques need to be used to circumvent this problem. A popular regularization strategy involves using a *dropout layer* [21]. During each optimization update step in the training-phase of the network, a dropout layer in front of the k-th layer randomly sets a predefined fraction of the components of the intermediate vector computed by the k-th layer to zero. There are several advantages of using dropout, e.g., the training is not biased towards a specific network architecture, additional stochasticity is injected into the optimization process to avoid getting trapped in local optima, and a sparsity structure is introduced into the network structure.

4. An MLP-based smoothness classifier

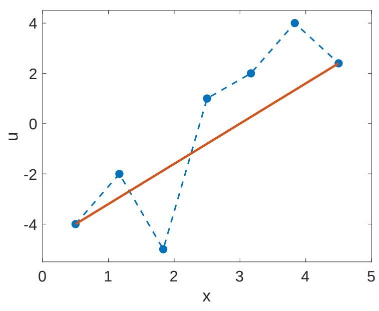
We now construct a network to classify the local smoothness of solutions to conservation laws approximated by a Fourier collocation scheme.

4.1. Scaled input vector

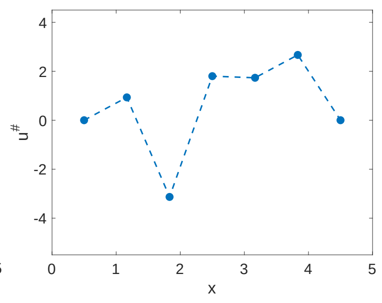
To predict the smoothness at the node x_j, we consider the stencil $\mathcal{S}_j^3 = \{x_{j-3}, ..., x_{j+3}\}$ of 7 nodes. The collocated values of the solution u on this stencil is used to construct the input to the network. In order to ensure good generalization of the network, we normalize the input before passing it to the network. This is done in three steps:

1. Compute the line passing through the end points values of the local solution
$$l(x) = u_{j-3} + (x - x_{j-3})\frac{(u_{j+3} - u_{j-3})}{x_{j+3} - x_{j-3}},$$
as shown in Figure 2(a).
2. Transform the local solution so that the end point values become null. This is achieved by setting
$$u^{\#}_{j-r} = u_{j-r} - l(x_{j-r}) \quad \text{for} \quad -3 \leqslant r \leqslant 3,$$
as shown in Figure 2(b).
3. Scale the local values so that they are normalized to lie in the interval $[-1, 1]$
$$u^{*}_{j-r} = \frac{2u^{\#}_{j-r} - M_j - m_j}{M_j - m_j} \quad \text{for} \quad -3 \leqslant r \leqslant 3,$$
where $M_j = \max\limits_{\mathcal{S}_j^3}(u^{\#}_i)$ and $m_j = \min\limits_{\mathcal{S}_j^3}(u\#_i)$ as shown in Figure 2(c).

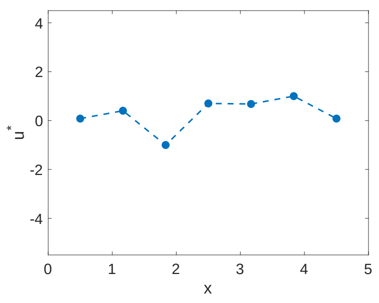
The input vector to the network is now taken as $\boldsymbol{X} = [u^*_{j-3}, ..., u^*_{j+3}]^\top \in \mathbb{R}^7$.



(a) Find line $l(x)$.



(b) Compute $u^{\#}$.



(c) Compute u^{*}.

Figure 2: Scaling input vector for neural network.

4.2. Datasets

The training data set is generated using pre-defined parametrized functions with varying regularity τ:

$$\begin{aligned}
f_1(x) &= \sin(2xa), \\
f_2(x) &= a|x-\pi|, \\
f_3(x) &= \begin{cases} a_1 & \text{if } |x-\pi| \leqslant a_3, \\ a_2 & \text{if } |x-\pi| > a_3, \end{cases}, \\
f_4(x) &= \begin{cases} a_1|x-\pi| - a_1a_3 & \text{if } |x-\pi| \leqslant a_3, \\ a_2|x-\pi| - a_2a_3 & \text{if } |x-\pi| > a_3, \end{cases}, \\
f_5(x) &= \begin{cases} 0.5a_1|x-\pi|^2 - a_1a_3 & \text{if } |x-\pi| \leqslant a_3, \\ a_2|x-\pi|^2 - a_2a_3 - 0.5a_3^2(a_1-a_2) & \text{if } |x-\pi| > a_3. \end{cases}
\end{aligned} \tag{4.1}$$

Note that the functions in (4.1) are symmetric about $x = \pi$, and thus their restrictions on $[0, 2\pi]$ can be periodically extended to $\mathbb{R}$. For each of these functions, we compute the Fourier approximation on $[0, 2\pi]$, collocated on a grid with 401 nodes. We then define a domain $\mathcal{D} \subset [0, 2\pi]$ from which we sample the 7-point stencil values of the target function. In order to capture spurious oscillations near discontinuities, we ensure that the nodes of the stencil is shifted in relation to the original collocation grid, as illustrated in Fig. 3. The parameter sampling for each function in (4.1), the domain $\mathcal{D}$, and the corresponding local-regularity are listed in Table 1. Depending on the regularity of the underlying function in $\mathcal{D}$, we set the target label following Table 2.

The final data set has around 380,000 samples (see the last column of Table 2), with 80% samples points used as the training set $\mathbb{T}$ and the remaining 20% used as the validation set $\mathbb{V}$. At the end of each epoch of training the network, the accuracy is evaluated on $\mathbb{V}$ to monitor and avoid overfitting of the training set.

4.3. Network architecture and training

The network is chosen to have depth $K = 4$, with each of the three hidden layers having a width $N_1 = N_2 = N_3 = 16$. The ELU activation function is used in all hidden layers with a parameter $\alpha = 1$. We use a dropout layer during the training phase after the first hidden layer, with the

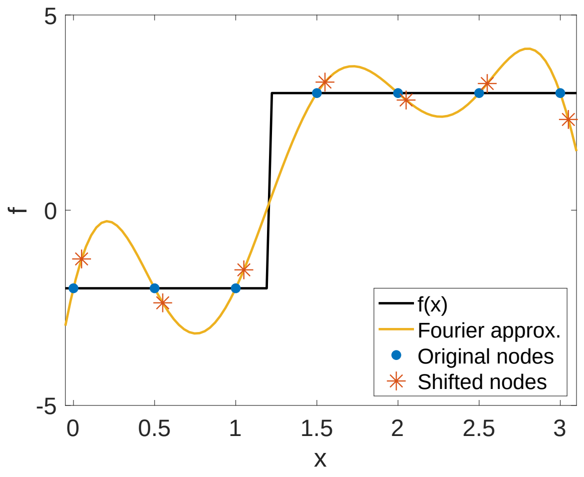


Figure 3: Sampling on shifted stencil.

f(x)	Parameters	$\mathcal{D}$	τ
f_1	$a = \{-20, -19.5, -19., \ldots, 19.5\}$	$[0, 2\pi]$	4
f_2	$a = \{-10, -9, \ldots, 10\}$	$[3.53, 5.89]$	4
f_3	$a_1 = \{-10, -9, \ldots, 9\}$ $a_2 = \{-10, -9, \ldots, 9\}$ $a_3 = \{0.25, 0.5, \ldots, 2.5\}$	$[a_3 - 0.05, a_3 + 0.05]$	1
f_4	$a_1 = \{-10, -9, \ldots, 9\}$ $a_2 = \{-10, -9, \ldots, 9\}$ $a_3 = \{0.25, 0.5, \ldots, 2.5\}$	$[a_3 - 0.05, a_3 + 0.05]$	2
f_5	$a_1 = \{-10, -9, \ldots, 9\}$ $a_2 = \{-10, -9, \ldots, 9\}$ $a_3 = \{0.25, 0.5, \ldots, 2.5\}$	$[a_3 - 0.05, a_3 + 0.05]$	3

Table 1: Functions in the training set.

Regularity	τ	Label	Samples
Discontinuous	1	0	104,400
$C^0 \setminus C^1$	2	1	79,200
$C^1 \setminus C^2$	3	2	63,474
C^2 and higher	$\geqslant 4$	3	133,404
			380,478

Table 2: Assigning target labels based on local regularity. The last column indicates the number of samples generated for each class.

dropout parameter $\lambda = 0.1$. i.e. 10% of the weights are randomly set to zero. The softmax function (3.5) is used as the output function. The output of the network is $\hat{\boldsymbol{Y}} \in \mathbb{R}^4$, with $\hat{Y}_l$ denoting the probability that the target label is $l-1$ (see Table 2).

The network was re-trained several times using the ADAM optimizer [10]. The bias vectors were initialized to null and the weight matrix in layer k was initialized using a uniform distribution on $[-\sqrt{3/q_k}, \sqrt{3/q_k}]$, where $q_k = 0.5 * (N_k + N_{k-1})$. For each case, the network was trained by setting the maximum number of epochs to 1000 and using an early-stopping criteria if the validation accuracy did not improve for 50 consecutive epochs. The training of the best performing network (based on the validation accuracy) terminated after 330 epochs, had a training accuracy of 99.57% and a validation accuracy of 99.72%.

5. Numerical results

In the following we demonstrate the performance of the MLP-based smoothness indicator when used in the framework of a Fourier collocation scheme (2.8). Spurious oscillations are controlled using a combination of artificial-viscosity and a high-order filter. The algorithm to compute $\boldsymbol{u}_h(t + \Delta t)$ from $\boldsymbol{u}_h(t)$ is outlined in Algorithm 1. The order p of the exponential filter at each time-step is set based on the global minimum regularity τ_{min} of the solution predicted by the network, as listed in Table 3. Note that we use a low-order filter, i.e., $p = 2$, only for the first time-step, if the initial condition has a $\tau_{\text{min}} \leqslant 2$. The regularity-based orders where determined empirically, with the same selection strategy used for all results presented in this work.

In the following, we present results for both scalar and systems of conservation laws. Test cases which are inherently non-periodic are periodically extended in a suitable manner before using the Fourier collocation method. For such problem, we only mention the number of nodes used to discretize the original domain.

Remark 5.1. *As already discussed, the solution on a 7-point stencil needs to be scaled before being given to the network (see Section 4.1). To ignore small amplitude noise, we treat all input satisfying* $|M_j - m_j| < 0.075$ *as having regularity* $\tau_j = 4$. *Such small scale oscillations will be effectively removed by the high-order filter used.*

τ_{min}	Time	Filter order p
1	$t = 0$	2
1	$t > 0$	14
2	$t = 0$	8
2	$t > 0$	16
3	$t \geqslant 0$	18
$\geqslant 4$	$t \geqslant 0$	20

Table 3: Prescribing regularity based filter order. The order is lowered for the first time-step if the initial condition has low regularity.

```
Algorithm 1: Artificial viscosity + filtering
    Input: MLP predictor, solution $\boldsymbol{u}_h(t)$, time $t$.
    Output: Solution $\boldsymbol{u}_h(t+\Delta t)$.
    Use MLP to predict $\tau_j$ for $1 \leqslant j \leqslant N$;
    Compute viscosity $\mu_j$ using (2.13);
    Smooth viscosity using (2.14);
    Using $t$ and $\tau_j$, determine filter order $p$ based on Table 3;
    Update solution using SSPRK4 to get $\boldsymbol{u}_h(t+\Delta t)$;
```

5.1. Scalar problems

We shall first consider scalar problems in both one and two spatial dimensions.

5.1.1. Linear advection

The linear advection equation is given by the flux function $f(u) = u$ in (2.3)). We first consider a smooth initial condition

$$u_0(x) = \exp(\sin(2\pi(x - 0.25)),$$

on the domain $[0, 1]$, simulated until the final time $T = 1$ and a fixed time-step $\Delta t = 0.001$. Since the solution is smooth, no artificial viscosity is needed. We show the convergence of the scheme as a function of the grid points N in Figure 4. For poorly resolved cases with $N < 20$, the ANN erroneously classifies certain points with a lower regularity and viscosity is added. This problem vanishes for larger N and we observe the expected exponential convergence.

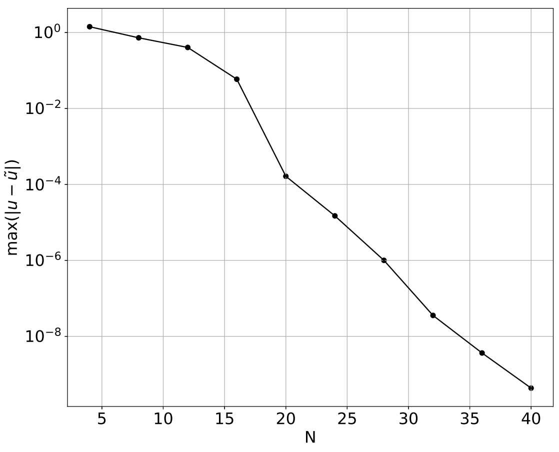
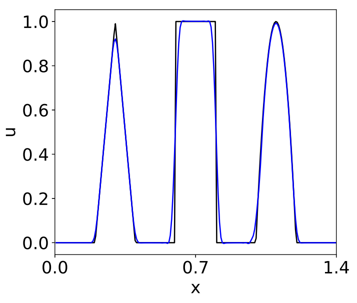


Figure 4: Error as a function of N for the smooth linear advection problem. Here u and u_h denote the exact and numerical solutions, respectively.

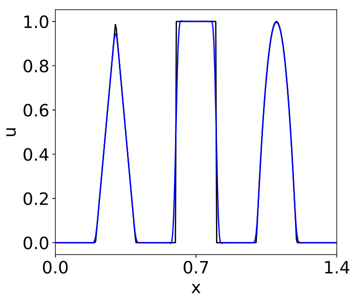
Next, we consider the problem with an initial condition comprising smooth and non-smooth features

$$u_0(x) = \begin{cases} 10(x-0.2) & \text{if } 0.2 < x \leqslant 0.3, \\ 10(0.4-x) & \text{if } 0.3 < x \leqslant 0.4, \\ 1 & \text{if } 0.6 < x \leqslant 0.8, \\ 100(x-1)(1.2-x) & \text{if } 1 < x \leqslant 1.2, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

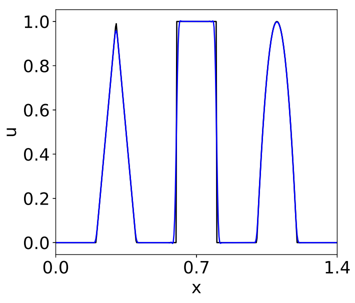
solved on the domain $[0, 1.4]$. The simulation is performed until $T = 1.4$ with $CFL = 2$ and at the conclusion the solution has completed one full rotation. The numerical solution on varying mesh sizes is shown in Figure 5 (a)-(c). Note that all existing discontinuities and corners are merely advected. Once such irregularities are smoothened by the viscous terms there is no mechanism in the underlying model to generate new discontinuities. Thus, it suffices to introduce local viscosity near such structures only for the first few time-step. This is indeed what is observed when we consider the time history of the viscosity introduced via the network, as shown in Figure 5 (d)-(f).



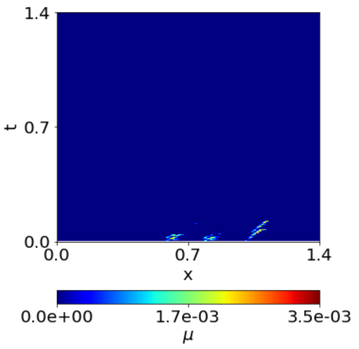
(a) $N = 200$.



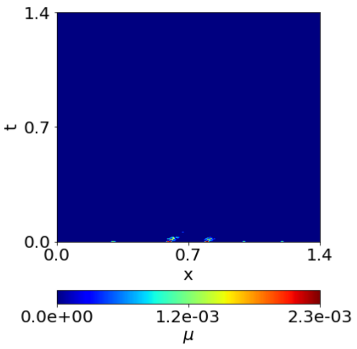
(b) $N = 300$.



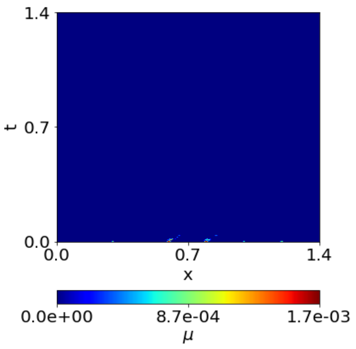
(c) $N = 400$.



(d) $N = 200$.



(e) $N = 300$.



(f) $N = 400$.

Figure 5: Solution for the linear advection equation. (a)-(c): solution at final time; (d)-(f): time history of artificial viscosity added.

5.1.2. *Burgers' equation*

We consider the Burgers' equation with the flux $f(u) = u^2/2$ in (2.3) with the initial data

$$u_0(x) = \begin{cases} -\sin(6\pi x) & \text{if } \frac{1}{6} \leqslant x \leqslant \frac{5}{6}, \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

on the domain $[0, 1]$, with the problem solved until time $T = 0.4$ with $CFL = 1.5$. We use this test case to assess the performance of the network when compared to the entropy viscosity method. The results shown in Figure 6 demonstrate that the network introduces local artificial viscosity with greater precision, leading to a much better resolution of the shock waves. We have chosen the tuneable parameters of the entropy viscosity method as $\alpha = 0.1$ and $\alpha_{\max} = 2.0$ (see (2.12)). One can argue that the numerical result can improve with a better choice these parameters. However, it is not possible to estimate their values a priori for a particular problem and they need to be determined empirically and for each new problem.

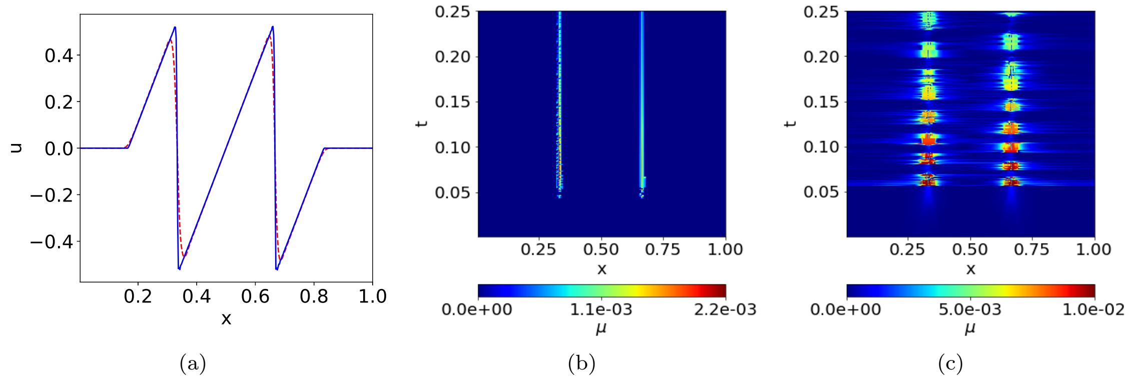
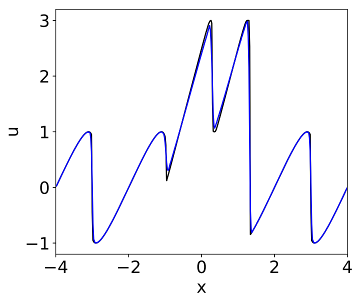


Figure 6: Solution for Burgers' equation with the initial condition (5.2) on a mesh with $N = 400$. (a) solution at final time with the network (solid) and the entropy viscosity method (dashed); (b) time history of artificial viscosity added by the network; (c) time history of artificial viscosity added by the entropy viscosity method.

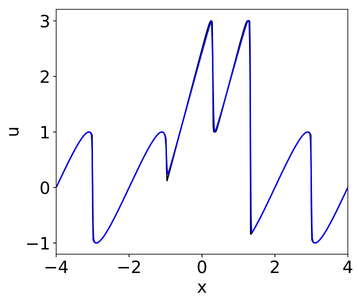
Next, we consider a more complex initial condition

$$u_0(x) = \begin{cases} \sin(\pi x) & \text{if } |x| \geqslant 1, \\ 3 & \text{if } -1 < x \leqslant -0.5, \\ 1 & \text{if } -0.5 < x \leqslant 0, \\ 3 & \text{if } 0 < x \leqslant 0.5, \\ 2 & \text{if } 0.5 < x \leqslant 1, \end{cases} \tag{5.3}$$

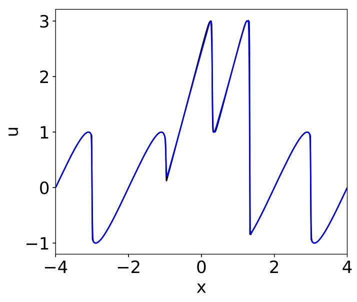
on the domain $[-4, 4]$. The problem is solved until time $T = 0.4$ with $CFL = 3$. The solution evolves into a number of shocks and rarefaction waves, as shown in Figure 7 for varying mesh sizes. The time history of the added artificial viscosity clearly confirms that the network tracks compressive discontinuous waves as they evolve and detects new discontinuities spontaneously emerging in the solution.



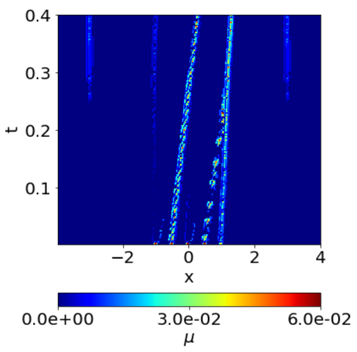
(a) $N = 400$.



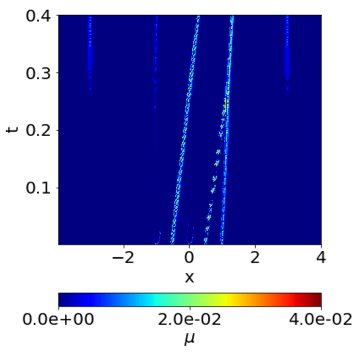
(b) $N = 600$.



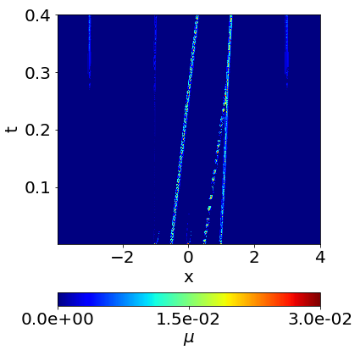
(c) $N = 800$.



(d) $N = 400$.



(e) $N = 600$.



(f) $N = 800$.

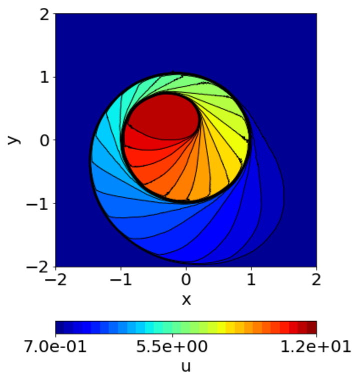
Figure 7: Solution for Burgers' equation with the initial condition (5.3). (a)-(c): solution at final time; (d)-(f): time history of artificial viscosity added.

5.1.3. 2D KPP rotating wave problem

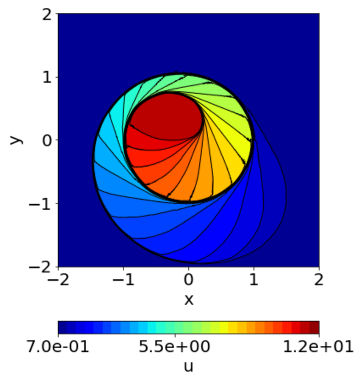
The KPP problem [12] describes the solution of a scalar two-dimensional equation with the non-linear flux $\boldsymbol{f}(u) = (\sin(u), \cos(u))$. The initial condition is given by

$$u_0(x, y) = \begin{cases} 3.5\pi, & \text{if } x^2 + y^2 < 1, \\ 0.25\pi, & \text{otherwise.} \end{cases}$$

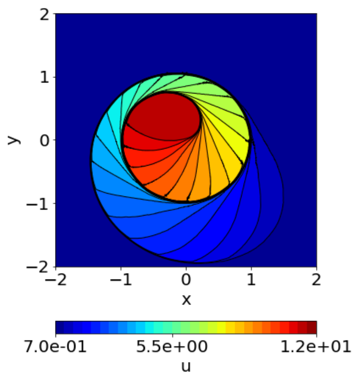
on the domain $[-2, 2]^2$. As the solution evolves, it twists into a spiral structure with sharp features. This is a challenging problem and several schemes have been shown to give incorrect solutions [12, 7]. We solve the problem until $T = 1$ with a CFL=1.5. We clearly observe in Figure 8 that the MLP introduces artificial viscosity only along the boundaries of the spiral structure where the solution develops a discontinuity.



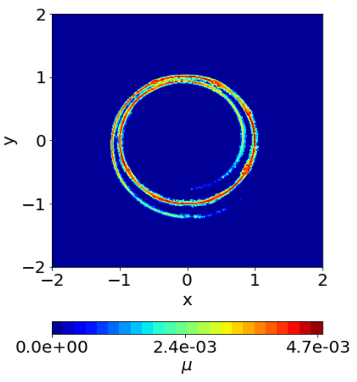
(a) 400×400.



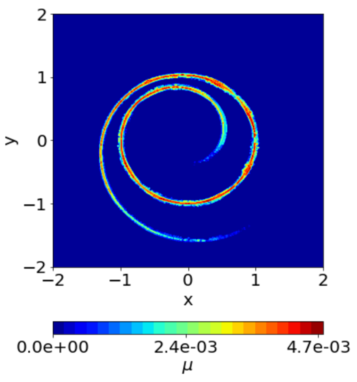
(b) 500×500.



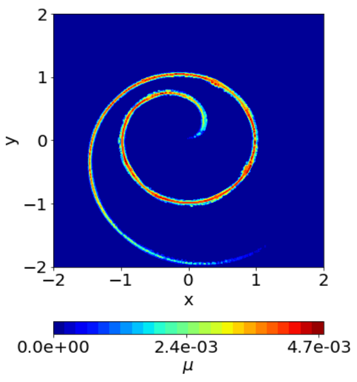
(c) 600×600.



(d) 600×600, $t = 0.2$



(e) 600×600, $t = 0.6$



(f) 600×600, $t = 1$

Figure 8: Solution to the rotating KPP problem. (a)-(c): contour plots of the solution at $T = 1$ on different grids, using 30 contour lines between 0.7 and 11.5; (d)-(f) artificial viscosity added at different time instance on the finest grid.

5.2. Euler equations

Consider the two-dimensional Euler equations given by

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{pmatrix}, \quad \boldsymbol{f}_x(\boldsymbol{u}) = \begin{pmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ v_x(E+p) \end{pmatrix}, \quad \boldsymbol{f}_y(\boldsymbol{u}) = \begin{pmatrix} \rho v_y \\ \rho v_x v_y \\ \rho v_y^2 + p \\ v_y(E+p) \end{pmatrix}, \tag{5.4}$$

where ρ, $\boldsymbol{v} = (v_x, v_y)$, p and E denote the fluid density, velocity, pressure, and total energy, respectively. The system is closed by an equation of state taken to be that of an ideal gas law,

$$E = \frac{1}{2}\rho|\boldsymbol{v}|^2 + \frac{p}{\gamma - 1}.$$

where γ denotes the ratio of specific heats. For all simulations in this work, we choose $\gamma = 1.4$ for a di-atomic gas such as air. The Mach number M of the flow is given by

$$M = \frac{|\boldsymbol{v}|}{a}, \quad a = \sqrt{\frac{\gamma p}{\rho}},$$

where a is the speed of sound in air.

5.2.1. 1D problems

We first consider the one-dimensional Euler equations. Note that we must choose the proxy variable to be used with the neural network to predict the local regularity. Based on empirical results, we take the Mach number M as the proxy variable for all one-dimensional simulations.

Sod shock tube [20]: We begin by considering the Sod problem with the initial condition on the domain $[0, 1]$ given as

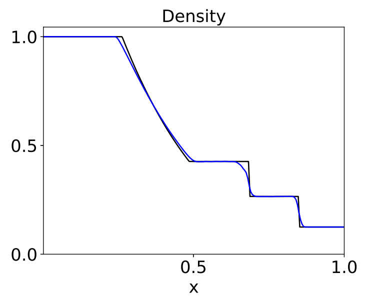
$$(\rho, u, p) = \begin{cases} (1, 0, 1) & \text{if } x < 0.5, \\ (0.125, 0, 0.1) & \text{if } x > 0.5, \end{cases}$$

and solve the problem until $T = 0.2$ with CFL=3. The solution obtained on the original domain is shown in Figure 9. Note that there is a contact and a shock discontinuity in the solution, both of which originate from the initial solution jump at $x = 0$. Since the contact wave is a linear wave, artificial viscosity is needed only for the first few time-steps, as was the case for the waves in the linear advection equation. Thus, the network primarily tracks the shock wave in the solution, as can be seen from the time history plot for the viscosity shown in Figure 9(d)-(f).

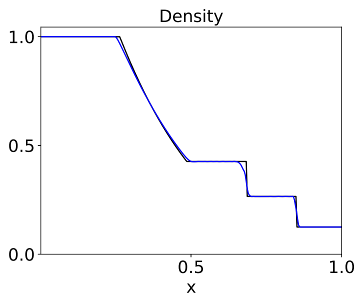
Lax shock tube [14]: The initial condition for this problem is given by

$$(\rho, u, p) = \begin{cases} (0.445, 0.698, 3.528) & \text{if } x < 0, \\ (0.5, 0, 0.571) & \text{if } x > 0 \end{cases}$$

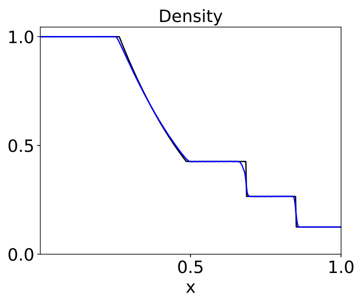
on the domain $[-5, 5]$. The final time is $T = 1.3$ and we take CFL=3. The numerical solution shown in Figure 10 once again demonstrates that the network is able to track the shock wave throughout



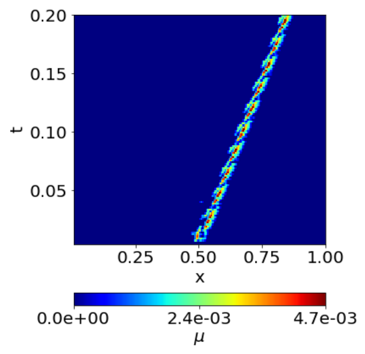
(a) $N = 200$.



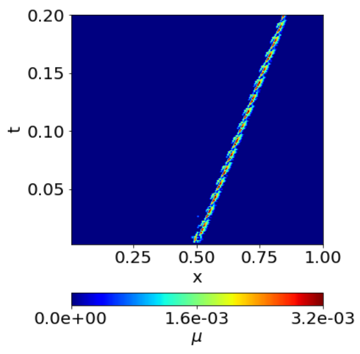
(b) $N = 300$.



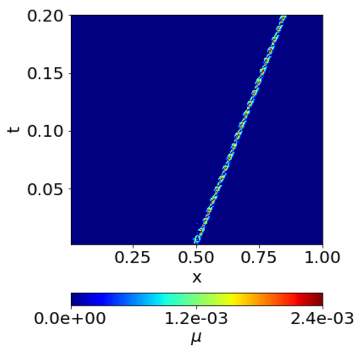
(c) $N = 400$.



(d) $N = 200$.



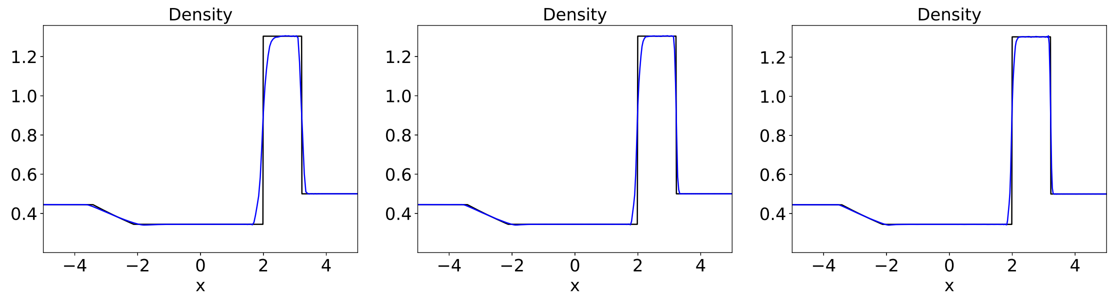
(e) $N = 300$.



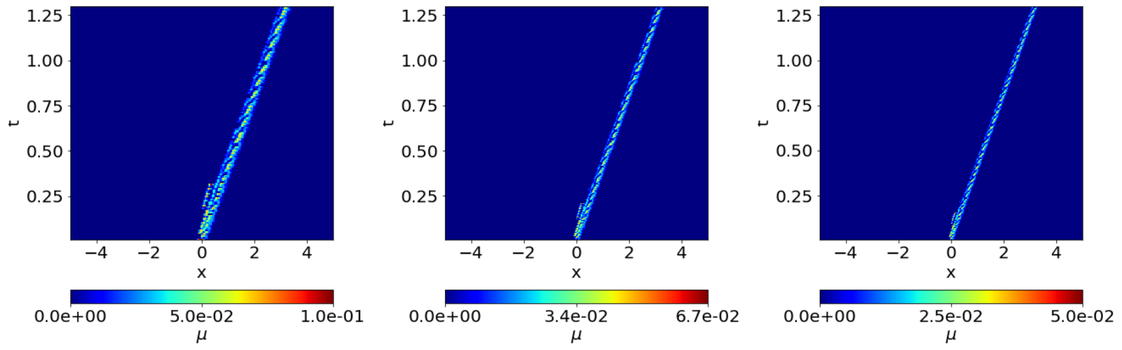
(f) $N = 400$.

Figure 9: Solution for the Sod shock-tube problem at $T = 0.2$. (a)-(c): solution at final time; (d)-(f): time history of artificial viscosity added.

the entire simulation.



(a) $N = 200$. (b) $N = 300$. (c) $N = 400$.



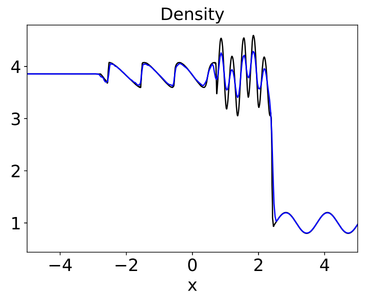
(d) $N = 200$. (e) $N = 300$. (f) $N = 400$.

Figure 10: Solution for the Lax problem at $T = 1.3$. (a)-(c): solution at final time; (d)-(f): time history of artificial viscosity added.

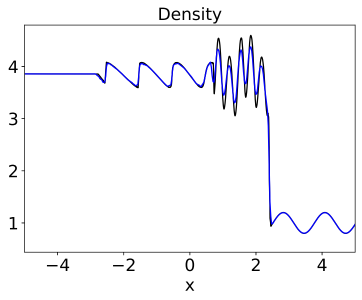
Shock-entropy problem [19]: Finally, we consider a problem describing the interaction of a strong shock with a smooth wave containing smooth physical oscillations. The initial condition on $[-5, 5]$ is prescribed as

$$(\rho, u, p) = \begin{cases} (3.857143, 2.629369, 10.33333) & \text{if } x < -4, \\ (1 + 0.2\sin(5x), 0, 1) & \text{if } x > 4. \end{cases}$$

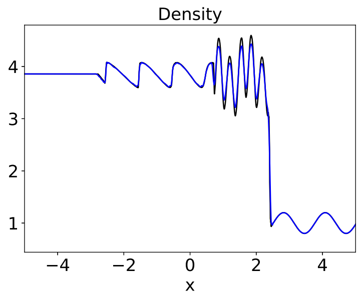
The final time is $T = 1.8$ and we take CFL=3. The numerical solution and the time-history of artificial viscosity are shown in Figure 11. Note that the high-frequency physical oscillations are captured with high resolution and no added viscosity, while no spurious oscillations appear near the primary shock wave or the secondary shock waves emerging upstream of the shock.



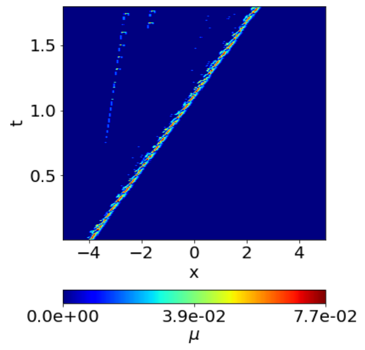
(a) $N = 300$.



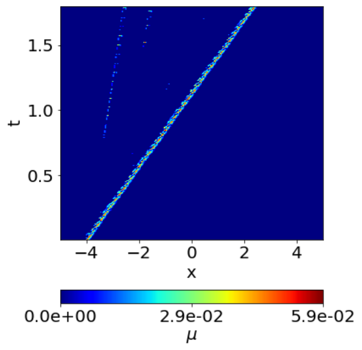
(b) $N = 400$.



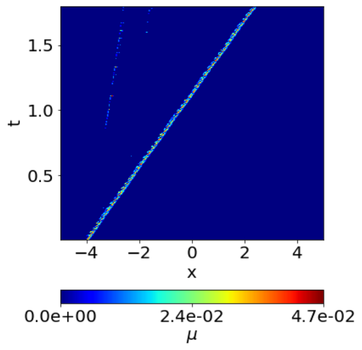
(c) $N = 500$.



(d) $N = 300$.



(e) $N = 400$.



(f) $N = 500$.

Figure 11: Solution for the shock-entropy problem at $T = 1.8$. (a)-(c): solution at final time; (d)-(f): time history of artificial viscosity added.

5.2.2. 2D problems

For the one-dimensional system, we used the Mach number as the proxy variable in the network. However, we found that using the density ρ leads to better results in two-dimensional simulations. We begin by solving 2D Riemann problems [13], where the initial condition comprises 4 constant states in each quadrant/zone of the domain $[0, 1.2]^2$ (see Figure 12).

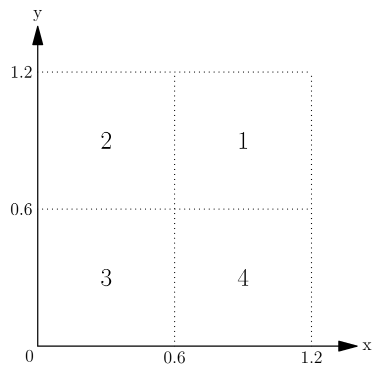


Figure 12: Domain for two-dimensional Riemann problems.

Riemann problem 4: This test case describes four interacting shock waves. The initial condition is given by

$$(\rho, v_x, v_y, p)_0(x, y) = \begin{cases} (1.1, 0, 0, 1.1), & \text{if } (x, y) \in \text{Zone } 1, \\ (0.5065, 0.8939, 0, 0.35), & \text{if } (x, y) \in \text{Zone } 2, \\ (1.1, 0.8939, 0.8939, 1.1), & \text{if } (x, y) \in \text{Zone } 3, \\ (0.5065, 0, 0.8939, 0.35), & \text{if } (x, y) \in \text{Zone } 4. \end{cases}$$

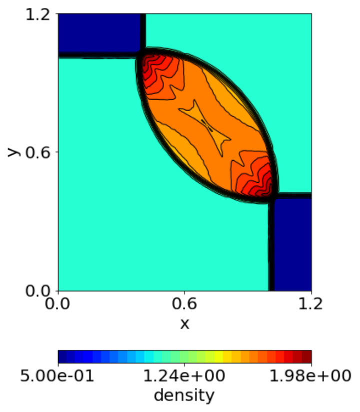
The solution at the final time $T = 0.25$ run with CFL=3 is shown in Figure 13. As can be seen in Figure 13(d)-(f), artificial viscosity is only introduced in the neighborhood of the shock waves.

Riemann problem 12: This problem develops two contact waves and two shock waves. The initial condition is

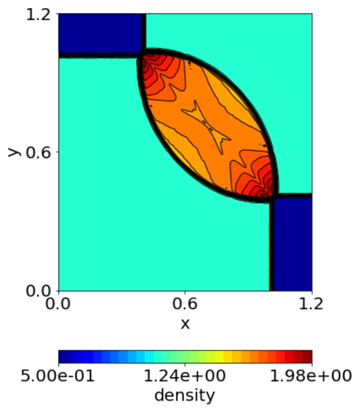
$$(\rho, v_x, v_y, p)_0(x, y) = \begin{cases} (0.5313, 0, 0, 0.4), & \text{if } (x, y) \in \text{Zone } 1, \\ (1, 0.7276, 0, 1), & \text{if } (x, y) \in \text{Zone } 2, \\ (0.8, 0, 0, 1), & \text{if } (x, y) \in \text{Zone } 3, \\ (1, 0, 0.7276, 1), & \text{if } (x, y) \in \text{Zone } 4. \end{cases}$$

The solution at the final time $T = 0.25$, obtained with CFL=3, is shown in Figure 14. Note that the artificial viscosity is primarily added near the shock-waves and not near the contact discontinuities. This is consistent with the fact that contact waves are linear waves and the viscosity introduced during the initial few time-steps is sufficient to smoothen such discontinuities.

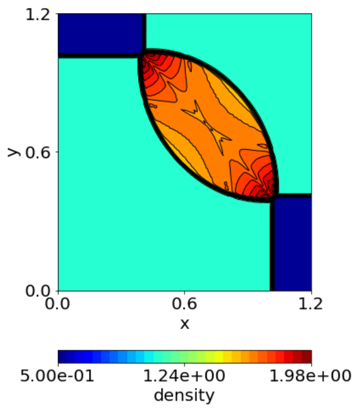
Riemann problem 17: This problem demonstrates the interaction of contact, shock, and



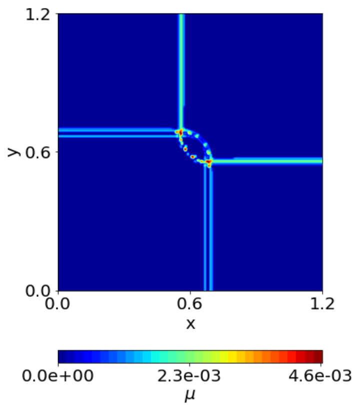
(a) 200 × 200.



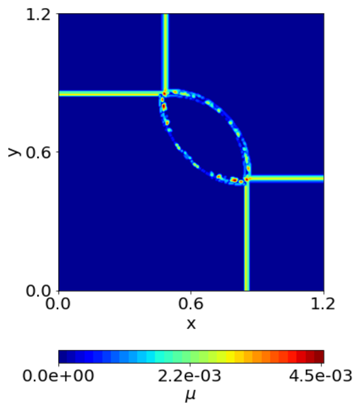
(b) 250 × 250.



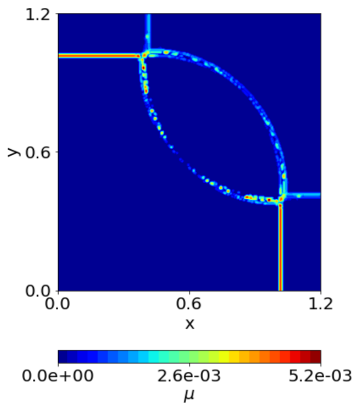
(c) 300 × 300.



(d) 300 × 300, $t = 0.05$

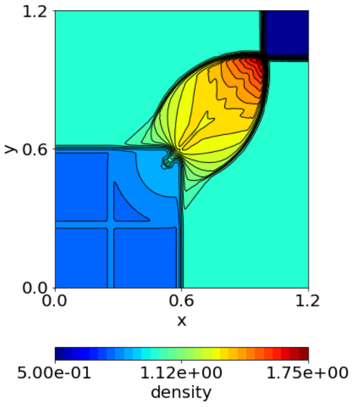


(e) 300 × 300, $t = 0.15$

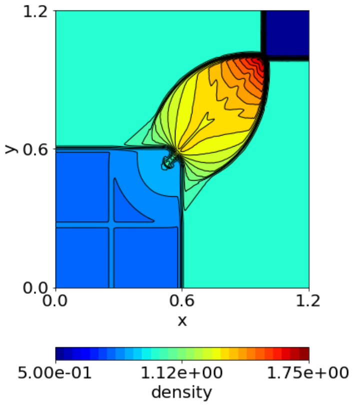


(f) 300 × 300, $t = 0.25$

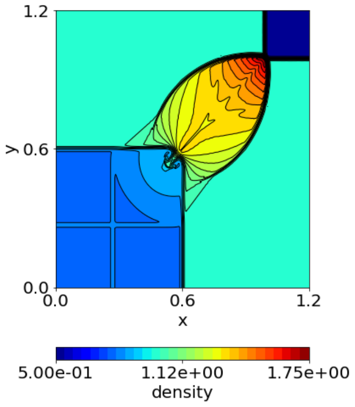
Figure 13: Solution for the 2D Riemann problem 4. (a)-(c) contour plot of density at $T = 0.25$, using 30 contour lines between 0.5 and 1.98; (d)-(f) artificial viscosity added at different time instance on the finest grid.



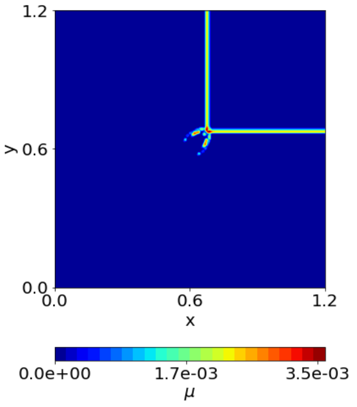
(a) 200×200.



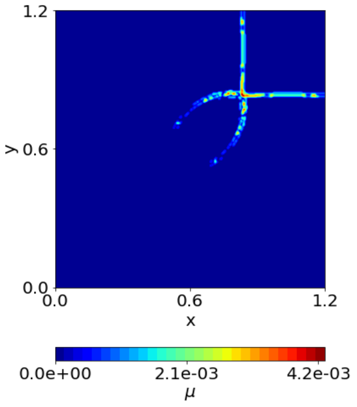
(b) 250×250.



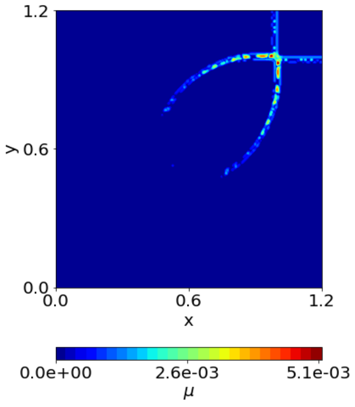
(c) 300×300.



(d) 300×300, $t = 0.05$



(e) 300×300, $t = 0.15$



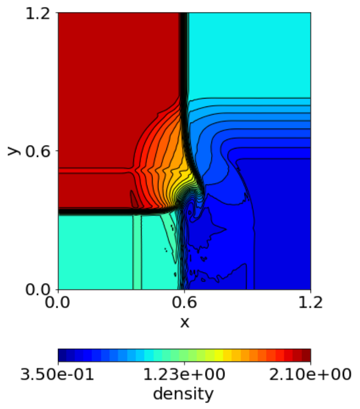
(f) 300×300, $t = 0.25$

Figure 14: Solution for the 2D Riemann problem 12. (a)-(c) contour plot of density at $T = 0.25$, using 30 contour lines between 0.5 and 1.75; (d)-(f) artificial viscosity added at different time instance on the finest grid.

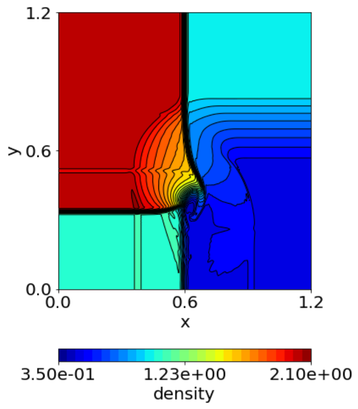
rarefaction waves. The initial condition is given as

$$(\rho, v_x, v_y, p)_0(x,y) = \begin{cases} (1, 0, -0.4, 1), & \text{if } (x,y) \in \text{Zone } 1, \\ (2, 0, -0.3, 1), & \text{if } (x,y) \in \text{Zone } 2, \\ (1.0625, 0, 0.2145, 0.4), & \text{if } (x,y) \in \text{Zone } 3, \\ (0.5197, 0, -1.1259, 0.4), & \text{if } (x,y) \in \text{Zone } 4, \end{cases}$$

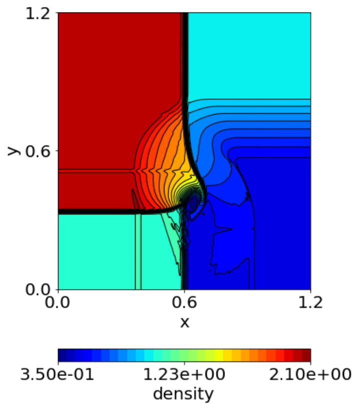
The solution at the final time $T = 0.3$, obtained with CFL=3, is shown in Figure 15. Once again, artificial viscosity is primarily injected near the shock wave. While viscosity is also introduced near the contact wave, it is done in a much more sparing manner as compared to the shock.



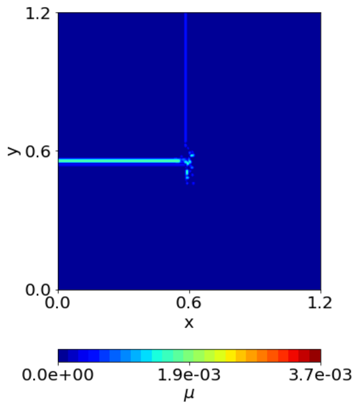
(a) 200×200.



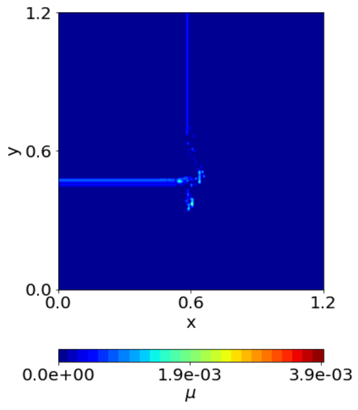
(b) 250×250.



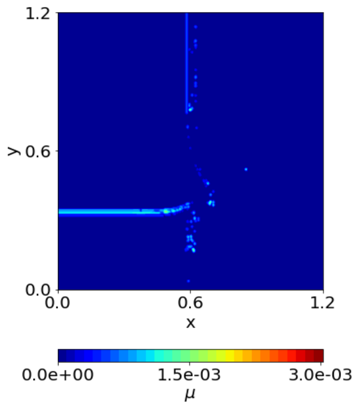
(c) 300×300.



(d) 300×300, $t = 0.05$



(e) 300×300, $t = 0.15$



(f) 300×300, $t = 0.3$

Figure 15: Solution for the 2D Riemann problem 17. (a)-(c) contour plot of density at $T = 0.3$, using 30 contour lines between 0.35 and 2.1; (d)-(f) artificial viscosity added at different time instance on the finest grid.

Shock-vortex problem [18]: The final test for the 2D Euler equations describes the collision of a smooth isentropic vortex with a shock wave. This is a challenging problem as both the shock and the vortex must be well resolved. The initial condition is given on the domain $[0,2]^2$ as

$$(\rho, v_x, v_y, p)_0(x,y) = \begin{cases} (\rho_L, v_{xL}, v_{yL}, p_L), & \text{if } x < 0.25 \\ (\rho_R, v_{xR}, v_{yR}, p_R), & \text{otherwise.} \end{cases}$$

The left base state is given by $(\rho_L, v_{xL}, v_{yL}, p_L) = (1, \sqrt{\gamma}, 0, 1)$. This state is perturbed by adding a vortex, for which the exact values are specified by a perturbation in velocity, temperature, and entropy:

$$\tilde{v}_x = \frac{y - y_c}{r_c}\Phi(r),\ \tilde{v}_y = -\frac{x - x_c}{r_c}\Phi(r),\ \tilde{\theta} = -\frac{\gamma - 1}{4\alpha\gamma}\Phi(r)^2,\ \tilde{s} = 0,$$

where $\Phi(r) = \epsilon e^{\alpha(1-\tau^2)}$ with $\tau = r/r_c$, $r_c = 0.05$, $\alpha = 0.204$, $\epsilon = 0.3$, $(x_c, y_c) = (0.25, 1.0)$ and $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$. Here, (x_c, y_c) denotes the initial center of the vortex. The temperature θ is given by $\theta = p/\rho$ and the thermodynamic entropy s by $s = \log p - \gamma \log \rho$, i.e., $s = 0$ implies $p = \rho^\gamma$.
The right state is given by

$$\begin{aligned} \rho_R &= \rho_L\Big(\tfrac{(\gamma+1)p_R+\gamma-1}{(\gamma-1)p_R+\gamma+1}\Big), & v_{yR} &= 0, \\ v_{xR} &= \sqrt{\gamma} + \sqrt{2}\Big(\tfrac{1-p_R}{\sqrt{\gamma-1+p(\gamma+1)}}\Big), & p_R &= 1.3. \end{aligned}$$

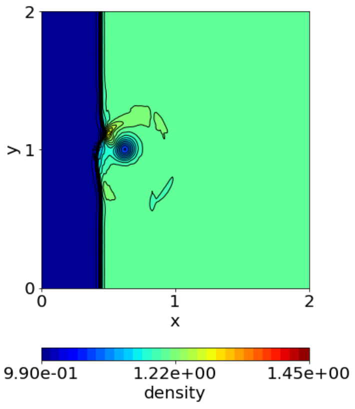
The problem is simulated with CFL= 3. As can be seen in Figure 16, the vortex cleanly passes through the shock at time $T = 0.35$ and is well resolved at $T = 0.8$. This is explained by the fact the the network only adds viscosity in the neighborhood of the shock, even when the vortex is interacting with the shock, illustrated in Figure 17).

6. Conclusion

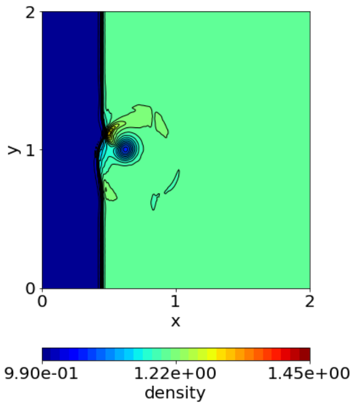
Nonlinear viscosity has been used with substantial success to regularize local solutions to conservation laws with the goal of reducing or eliminating artificial numerical oscillations when high order accurate methods are being used. However, the majority of such methods rely on a local estimate of the solution regularity, enabled by the local solution in element based methods, e.g., discontinuous Galerkin methods. For global methods such as spectral methods, such local regularity estimates is, however, not straightforward to recover, hence making the adaption of nonlinear viscosity to such methods difficult.

In this work we demonstrate how an appropriately trained artificial neural network can overcome this problems and be used to predict the local regularity of a solution with sufficient accuracy to allow the definition of an effective nonlinear viscosity. The emphasis of this work has been on Fourier spectral methods and we have illustrated the efficiency of the proposed techniques for a variety of one- and two-dimensional benchmark problems for scalar and systems of conservation laws.

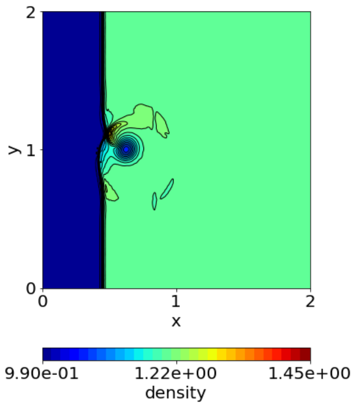
However, there is nothing unique about the use of Fourier methods for the proposed approach to be effective and its successful application to other methods with a globally defined basis, e.g., polynomial spectral methods or reduced basis methods [9], is likewise expected to be possible. We hope to report on such extensions in future work.



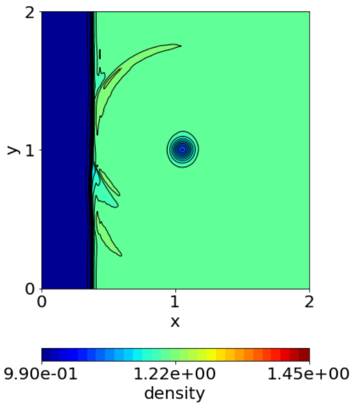
(a) 200×200, $t = 0.35$.



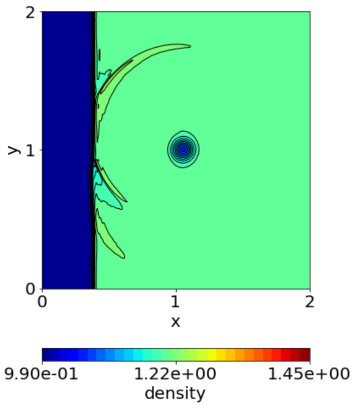
(b) 250×250, $t = 0.35$.



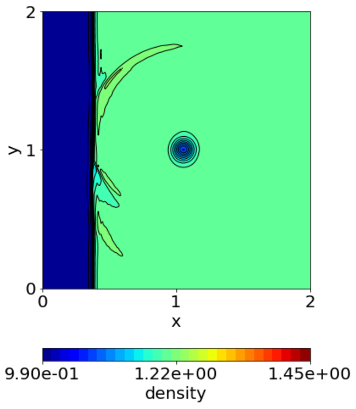
(c) 300×300, $t = 0.35$.



(d) 200×200, $t = 0.8$.

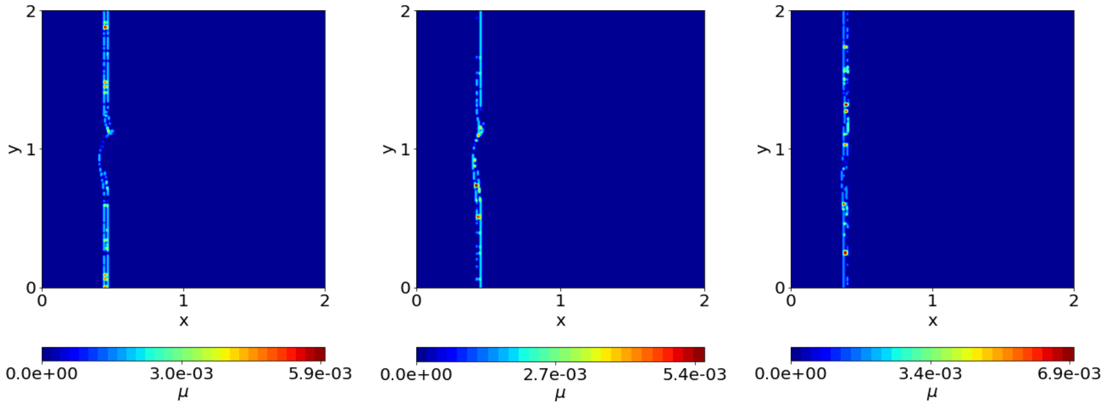


(e) 250×250, $t = 0.8$.



(f) 300×300, $t = 0.8$.

Figure 16: Density profile for the shock-vortex problem at $t = 0.35$ and $t = 0.8$, using 30 contour lines between 0.99 and 1.45.



(a) $t = 0.35$ (b) $t = 0.5$ (c) $t = 0.8$

Figure 17: Artificial viscosity added for the shock-vortex problem at different time instance on the finest grid (300×300).

References

[1] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2015.

[2] Niccolò Discacciati, Jan S Hesthaven, and Deep Ray. Controlling oscillations in high-order discontinuous galerkin schemes using artificial viscosity tuned by neural networks. *Journal of Computational Physics*, page 109304, 2020.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[4] Sigal Gottlieb. On high order strong stability preserving runge-kutta and multi step time discretizations. *Journal of Scientific Computing*, 25(1):105–128, 2005.

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

[6] Jean-Luc Guermond, Richard Pasquetti, and Bojan Popov. Entropy viscosity method for nonlinear conservation laws. *Journal of Computational Physics*, 230(11):4248–4267, 2011.

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

[8] Jan S Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral methods for time-dependent problems*, volume 21. Cambridge University Press, 2007.

[9] Jan S Hesthaven, Gianluigi Rozza, Benjamin Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016.

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

[11] Andreas Klöckner, Tim Warburton, and Jan S. Hesthaven. Viscous shock capturing in a time-explicit discontinuous galerkin method. *Math. Model. Nat. Phenom.*, 6(3):57–83, 2011.

[12] Alexander Kurganov, Guergana Petrova, and Bojan Popov. Adaptive semidiscrete central-upwind schemes for nonconvex hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 29(6):2381–2401, 2007.

[13] 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.

[14] Peter D. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Communications on Pure and Applied Mathematics*, 7(1):159–193, 1954.

[15] Yvon Maday and Eitan Tadmor. Analysis of the spectral vanishing viscosity method for periodic conservation laws. *SIAM Journal on Numerical Analysis*, 26(4):854–870, 1989.

[16] Per-Olof Persson and Jaime Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. *AIAA paper*, 112:2006, 2006.

[17] Steven J. Ruuth. Global optimization of explicit strong-stability-preserving runge-kutta methods. *Mathematics of Computation*, 75(253):183–207, 2006.

[18] Chi-Wang Shu. *High Order ENO and WENO Schemes for Computational Fluid Dynamics*, pages 439–582. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[19] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes, ii. *Journal of Computational Physics*, 83(1):32 – 78, 1989.

[20] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1 – 31, 1978.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[22] Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics, 2000.

[23] Jian Yu and Jan S Hesthaven. A study of several artificial viscosity models within the discontinuous galerkin framework. *Communication in Computational Physics*, page xx, 2020.

[24] Jian Yu, Jan S. Hesthaven, and Chao Yan. A data-driven shock capturing approach for discontinuous galekin methods. 2018.