A Priori Neural Networks Versus A Posteriori MOOD Loop: A High Accurate 1D FV Scheme Testing Bed

Alexandre Bourriaud, Raphaël Loubère, Rodolphe Turpault

J. Sci. Comput. manuscript No.
(will be inserted by the editor)

a priori Neural Networks vs *a posteriori* MOOD loop — A high accurate 1D FV scheme testing bed.

Alexandre Bourriaud · Raphaël Loubère · Rodolphe Turpault

the date of receipt and acceptance should be inserted later

Abstract In this work we present an attempt to replace an *a posteriori* MOOD loop used in a high accurate Finite Volume (FV) scheme by a trained artificial Neural Network (NN). The MOOD loop, by decrementing the reconstruction polynomial degrees, ensures accuracy, essentially non-oscillatory, robustness properties and preserves physical features. Indeed it replaces the classical *a priori* limiting strategy by an *a posteriori* troubled cell detection, supplemented with a local time-step re-computation using a lower order FV scheme (ie lower polynomial degree reconstructions). We have trained shallow NNs made of only two so-called hidden layers and few perceptrons which *a priori* produces an educated guess (classification) of the appropriate polynomial degree to be used in a given cell knowing the physical and numerical states in its vicinity. We present a proof of concept in 1D. The strategy to train and use such NNs is described on several 1D toy models: scalar advection and Burgers' equation, the isentropic Euler and radiative M1 systems. Each toy model brings new difficulties which are enlightened on the obtained numerical solutions. On these toy models, and for the proposed test cases, we observe that an artificial NN can be trained and substituted to the *a posteriori* MOOD loop in mimicking the numerical admissibility criteria and predicting the appropriate polynomial degree to be employed safely. The physical admissibility criteria is however still dealt with the *a posteriori* MOOD loop. Constructing a valid training data set is of paramount importance, but once available, the numerical scheme supplemented with NN produces promising results in this 1D setting.

Keywords Neural network · Machine learning · Finite Volume scheme · High accuracy · Hyperbolic system · *a posteriori* MOOD.

Mathematics Subject Classification (2010) 65M08 · 65A04 · 65Z05 · 85A25

A. Bourriaud
Institut de Mathématiques de Bordeaux (IMB), Université de Bordeaux, CNRS, Bordeaux INP, F33400, Talence, France
E-mail: alexandre.bourriaud@u-bordeaux.fr
Raphaël Loubère
Institut de Mathématiques de Bordeaux (IMB), Université de Bordeaux, CNRS, Bordeaux INP, F33400, Talence, France
E-mail: raphael.loubere@u-bordeaux.fr
Rodolphe Turpault
Institut de Mathématiques de Bordeaux (IMB), Université de Bordeaux, CNRS, Bordeaux INP, F33400, Talence, France
E-mail: rodolphe.turpault@u-bordeaux.fr

1 Introduction

Undoubtedly there is a frenetic activity revolving around the key words 'Machine Learning', 'Artificial Intelligence', 'Neural Networks', etc. in most, if not all, branches of science. While there has been some genuine success brought by the use of those seemingly new tools, some unreasonable expectations (overstated by the media, the society and, sometimes, the scientists themselves) creep in the scientific laboratories. Unfortunately the deep understanding of those 'revolutionary tools' is still far. This has generated some misunderstanding or exaggeration, leading unavoidably to fantasy and urban legends on their true usefulness or applicability. One of our goal in this paper is to test some of these ML tools in a genuine simple computational fluid dynamics (CFD) 1D framework.
We will focus more specifically on the sub-field of Neural Networks (NN) which begun in the early 1940's when the formal idea of using neural networks as computing machines was introduced in [31], and in [24] where the first rule for self-organized learning was proposed. In the following years a key person, Rosenblatt, yet another psychologist, introduced the McCulloch-Pitts model of a non-linear neuron [37, 38]; the Rosenblatt's perceptron. The perceptron, a single neuron with adjustable weights and bias, is the simplest form of a NN used for the classification of twofold separable patterns. This model was an inspiration for a whole generation of scientists (engineers, mathematicians, physicists, etc.) in the 1960s and 1970s, and this model is still in use nowadays. Later in [32] a precise mathematical analysis of a perceptron was proposed showing that it was not capable of representing many important problems. The important learning procedure called backpropagation of error was probably first introduced in [47] which, associated to the increase of the computer power, led to the application of multi-layer NN for pattern recognition, such as handwritten characters. Many applications later on have been added to the list of success while the access to more and more powerful computers opened the path for so-called deep learning using massive NNs. It is simply impossible to exhaustively cite or list all the applications using NNs nowadays, such as image recognition, language processing, etc. however many interesting historical or broad overviews have recently been produced such as [28, 23, 40] amongst many others on the web.
In this work we only consider and test feed-forward NN also called multi-layer perceptron network. Despite its obvious success, employing a NN in the field of numerical analysis and simulation, or, to design more efficient numerical methods is not yet a classical tool amongst all practitioners. Some researchers use NN to learn PDEs from data, to approximate numerical solutions of PDEs in complex geometries, to replace the underlying physical model by a NN, to predict the uncertain model parameters and many other examples. This broad range of applications appear in more and more complex CFD environments maintaining a mysterious aura for newcomers in the field.
Contrarily in this work we propose to present an ultra-simplified situation to test how a NN can be used within the design of a numerical scheme in one dimension. Very recently some works from the numerical analysis community (and numerical scheme designers) have partially brought some answers [35, 36, 45, 34, 1, 21]. We will follow their path and propose the simple testing bed constituted by a 1D high order finite volume (FV) scheme solving hyperbolic systems of partial differential equations (PDEs). All high order FV schemes are subject to the generation of spurious numerical oscillations in the vicinity of discontinuities. In this work we rely on an *a posteriori* MOOD loop (Multi-dimensional Optimal Order Detection) [8, 12, 13, 29] which allows the use of arbitrary accuracy. Indeed this *a posteriori* approach detects if the unlimited candidate solution (at the next time-step) fulfills some validity criteria: computer (`NaN`), physical (positivity preserving) and numerical admissibility (essentially non-oscillatory behavior based on relaxed Discrete Maximum Principle (DMP)). This stage is in fact a troubled cell detector [25, 35, 15, 18]. Then for any troubled cell, the numerical solution is discarded and locally recomputed starting again at the beginning of the time-step with a more robust and less accurate scheme. In the simplest approach one reduces the degree of the local polynomial reconstructions. This generates a new candidate solution which is again tested against the validity criteria, possibly detecting other troubled cells, etc. The resulting MOOD loop ensures that the converged solution is always valid according to the detection criteria provided that the lowest order scheme, the so-called parachute scheme, provides such a solution. Here we use the 1st

order accurate FV scheme. While the MOOD paradigm is in use in several codes see for instance [16, 5, 2, 9, 11, 3, 33, 6, 17, 41, 4, 43], some weaknesses can be pointed.
First the *a posteriori* MOOD loop breaks the parallel efficiency because some cells demand more attention than others: they are possibly recomputed several times while (most) others are accepted at the end of the very first MOOD iteration. Second, the MOOD paradigm has a fundamental explicit nature due to the *a posteriori* check and the local degree decrementing. Even if some recent works explore its extension to implicit schemes [10, 27], a fully *a priori* version would be more convenient and presumably more efficient even in parallel. Third, the numerical admissible detection criteria based on a relaxed DMP in the detection is not based on a rigorous theoretical and mathematical base and demands one parameter to be fixed. Moreover amongst the troubled cells (about 10% of the total amount of cells on average), the vast majority is flagged by this relaxed DMP. Therefore being able to *a priori* predict which ones are such cells would certainly accelerate their treatment.

The environment of this study is made as simple as possible on purpose

- only *a posteriori* MOOD FV schemes are considered [8, 12, 13, 29];
- the accuracy is restricted to 2nd, 3rd or 4th order using polynomial reconstruction with centered stencil and Strong Stability Preserving Runge-Kutta (SSPRK) time discretization [42, 19];
- only 1D experiments are carried out on four models of hyperbolic PDEs of increasing complexity: advection and Burgers' equation, isentropic hydrodynamics and M1 systems of conservation laws;
- only shallow NNs of exactly two hidden layers are addressed and trained via the MATLAB Deep Learning Toolbox$^{\text{TM}}$ R2019b;
- optimal performance or implementation are not considered here.

The goal is to built a 1D proof of concept that a shallow supervised NN can be trained to replace the *a posteriori* MOOD loop. We present the difficulties of this construction and the numerical evidences of their impact on the numerical results. This paper is organized as follows. In the first section, an arbitrarily high order FV scheme based on polynomial reconstructions and an *a posteriori* MOOD stabilization loop is presented thoroughly for a generic system of conservation laws. Then we briefly recall the concepts underlying Neural Network and comment its use in our framework. The third section is devoted to the replacement of the *a posteriori* MOOD loop by a supervised trained NN. The NN is constructed and trained with four different hyperbolic PDE models: advection, Burgers', isentropic Euler and M1. Numerical evidences are gathered in their dedicated sections. Several classical and demanding test cases are proposed and the NN-FV-MOOD scheme is compared with the classical *a posteriori* FV-MOOD scheme. A conclusion and perspective section terminates this paper.

2 High accurate Finite Volume scheme with *a posteriori* MOOD loop

In this section we present the governing equations and the high order FV scheme used to solve them. This family of FV schemes is devoted to solve hyperbolic system and, as such, is subject to the generation of spurious numerical oscillations in the vicinity of the discontinuities. Consequently, some form of limitation must be supplemented to stabilize the high order method. In this work we rely on an *a posteriori* MOOD loop (Multi-dimensional Optimal Order Detection), see [8, 12, 13, 29] and some applications in [16, 5, 2, 9, 11, 3, 33, 6, 17, 41, 4, 43]. This approach *a posteriori* checks if the unlimited candidate solution fulfills some validity criteria (computer, physical and numerical admissibility) and accordingly recomputes the current solution locally in space and time with a more robust but less accurate scheme by reducing the polynomial degree of the local reconstruction.

2.1 Finite volume scheme solving hyperbolic system of conservation laws

2.1.1 Governing equations

As stated earlier, 1D hyperbolic systems of conservation laws are considered in this work. They can therefore be written

$$\frac{\partial \boldsymbol{W}}{\partial t} + \frac{\partial \mathsf{F}(\boldsymbol{W})}{\partial x} = 0, \tag{1}$$

where $\boldsymbol{W}$ is the vector of conservative variables, $\mathsf{F}(\boldsymbol{W})$ is the associated flux vector, t and x are the time and space variables respectively. Four increasingly difficult 1D models are taken into account. First a scalar linear advection/transport equation

$$\boldsymbol{W} = w \in \mathbb{R}, \qquad \mathsf{F}(\boldsymbol{W}) = w. \tag{2}$$

Second, Burgers' equation [7]

$$\boldsymbol{W} = w \in \mathbb{R}, \qquad \mathsf{F}(\boldsymbol{W}) = \frac{1}{2}w^2, \tag{3}$$

Third, the isentropic Euler equations :

$$\boldsymbol{W} = (\rho, \rho u), \qquad \mathsf{F}(\boldsymbol{W}) = (\rho u, \rho u^2 + p), \tag{4}$$

where ρ is the gas density, u its velocity and p the pressure. We restrict the study to polytropic gases for which $p = p(\rho) = \kappa\rho^\gamma$, with $\gamma = 1.4$ and $\kappa = 1$. This system of PDEs is hyperbolic provided that the state remains admissible, see [44], that is within

$$\mathcal{A} = \left\{ \boldsymbol{W} = (\rho, \rho u)^\top \in \mathbb{R}^2 \;/\; \rho > 0 \right\}. \tag{5}$$

Fourth, the M1 model for radiative transfer in transparent media

$$\boldsymbol{W} = (E, F), \qquad \mathsf{F}(\boldsymbol{W}) = (F, c^2 P), \tag{6}$$

where E is the radiative energy, F the radiative flux, c the speed of light and $P = P(E, F)$ is the radiative pressure [14]. The set of admissible states is the following one

$$\mathcal{A} = \left\{ \boldsymbol{W} = (E, F) \in \mathbb{R}^2 \;/\; E > 0,\ |F| < cE \right\}. \tag{7}$$

2.1.2 Finite Volume discretization

The computational domain Ω is paved with M cells of constant size Δx. The cell centers are noted x_i and the cell interfaces $x_{i-1/2}$ and $x_{i+1/2}$ for $i = 1, \dots, M$. The time step is denoted Δt and changes during the simulation according to the CFL condition.
In a FV context, a constant per cell approximation is considered which approximates the mean value of the solution $\boldsymbol{W}(x, t)$ over the cell at time t^n

$$\boldsymbol{W}_i^n \simeq \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} \boldsymbol{W}(x, t^n)\, dx. \tag{8}$$

Using the method of lines with the explicit time discretization, a FV scheme for (1) writes

$$\boldsymbol{W}_i^{n+1} = \boldsymbol{W}_i^n - \frac{\Delta t}{\Delta x} \left[\mathcal{F}_{i+1/2}^n - \mathcal{F}_{i-1/2}^n \right], \tag{9}$$

where $\mathcal{F}_{i+1/2}^n = \mathcal{F}\left(\widetilde{\boldsymbol{W}}_i^n(x_{i+1/2}), \widetilde{\boldsymbol{W}}_{i+1}^n(x_{i+1/2}) \right)$ is the numerical flux at interface $i+\frac{1}{2}$ and $\widetilde{\boldsymbol{W}}_i^n(x_{i+1/2})$ is its extrapolated state. In this work the Rusanov flux is considered for the sake of simplicity

$$\mathcal{F}\left(\boldsymbol{W}_L, \boldsymbol{W}_R\right) = \frac{\mathsf{F}(\boldsymbol{W}_L) + \mathsf{F}(\boldsymbol{W}_R)}{2} - b_{LR} \frac{\boldsymbol{W}_R - \boldsymbol{W}_L}{2} \tag{10}$$

where b_{LR} is related to the fastest characteristic speed of the approximate Riemann problem involving $\boldsymbol{W}_L$ and $\boldsymbol{W}_R$. However, other classical first order schemes could have been considered.

Representation and reconstruction. In order to compute the extrapolated left and right states at each interface a polynomial per cell reconstruction is performed. It consists in finding the polynomial of degree $d > 0$ in each cell denoted $\widetilde{\boldsymbol{W}}_i^d(x)$ such that the mean value in the current cell is conserved

$$\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \widetilde{\boldsymbol{W}}_i^d(x)\,dx = \boldsymbol{W}_i\,. \tag{11}$$

The chosen polynomial is the one that best fits the mean values in the least-square sense on a stencil of neighbor cell $\mathcal{S}_i^d$, *i.e* the polynomial coefficients are set by minimizing the cost function

$$J(\widetilde{\boldsymbol{W}}_i^d) = \frac{1}{2}\sum_{k\in\mathcal{S}_i^d}\left|\left(\frac{1}{\Delta x}\int_{x_{k-1/2}}^{x_{k+1/2}} \widetilde{\boldsymbol{W}}_i^d(x)\,dx\right) - \boldsymbol{W}_k,\right|^2. \tag{12}$$

Obviously, the centered stencil $\mathcal{S}_i^d$ should contain at least $d+1$ cells to fulfill (11)-(12). In practice, a few more cells are considered to reduce oscillations, usually about $\lfloor\frac{3}{2}(d+1)\rfloor$ is employed [8].

Time discretization. The time discretization is performed using SSP (Strong Stability Preserving) Runge-Kutta methods (see [20] for instance). The same order is used for time and space approximations for the sake of consistency. The SSP property allows to preserve convex sets so that the effort in the limiting is entirely undertaken in the space discretization. As an example, the third order scheme used in this work is the classical SSP-RK(3,3) scheme

$$\boldsymbol{W}_h^{(1)} = \boldsymbol{W}_h^n + \Delta t\,\mathcal{L}(\boldsymbol{W}_h^n) \tag{13}$$

$$\boldsymbol{W}_h^{(2)} = \boldsymbol{W}_h^{(1)} + \Delta t\,\mathcal{L}(\boldsymbol{W}_h^{(1)}) \quad\to\quad \boldsymbol{W}_h^* = \frac{3}{4}\boldsymbol{W}_h^n + \frac{1}{4}\boldsymbol{W}_h^{(2)} \tag{14}$$

$$\boldsymbol{W}_h^{(3)} = \boldsymbol{W}_h^{(2)} + \Delta t\,\mathcal{L}(\boldsymbol{W}_h^*) \quad\to\quad \boldsymbol{W}_h^{n+1} = \frac{1}{3}\boldsymbol{W}_h^n + \frac{2}{3}\boldsymbol{W}_h^{(3)}. \tag{15}$$

where $\mathcal{L}$ is the spatial discrete operator associated to the current FV scheme. The SSP-RK(2,2) and SSP-RK(5,4) schemes are used whenever second or fourth orders are respectively required.

2.2 *A posteriori* MOOD loop as limiting strategy

A high order method is subject to the generation of spurious oscillations in the vicinity of shock waves, contacts, steep gradients, etc. An *a posteriori* MOOD loop is used here to stabilize the numerical scheme in these cases. The MOOD paradigm consists in *a posteriori* checking the admissibility of a candidate numerical solution at time t^{n+1} through relevant criteria. Whenever a cell does not fulfill these criteria, a local re-computation employing a more robust scheme is performed. The principle is illustrated on figure 1.

2.2.1 Cascades and parachute schemes

As a consequence, in a MOOD loop one must design a cascade of increasingly robust schemes (and usually decreasingly accurate ones). At least two schemes must compose this cascade: the most accurate but possibly oscillatory scheme and a less accurate but robust one called the 'parachute scheme'. The unlimited $(d+1)$th order scheme is the starting and most accurate scheme and the first-order one as the parachute. In between these two some intermediate schemes are considered. For instance we will employ the following cascade: $d = 3 \to 2 \to 1 \to 0$ although shorter cascades such as $d = 3 \to 1 \to 0$ are also possible.

2.2.2 Detection criteria: Computer, Physical and Numerical admissibility

Detection criteria are used to determine if a candidate solution in a given cell is admissible or must be sent back for re-computation (see figure 1). These are the three sets of detection criteria used in this work for a given candidate solution $\boldsymbol{W}_i^{n+1,*}$ in cell Ω_i:

Computational - CAD. This criterion detects floating point exceptions *i.e.* the numerical solution passes the Computer Admissible Detection criteria if it does not contain any `NaN` (Not-a-Number), `Inf` (Infinite), etc.

Physical - PAD. The numerical solution in cell is validated by the PAD criterion if it remains in the set of admissible states $\mathcal{A}$. The PAD criteria must hence be adapted to the physics embedded into the model of PDEs. In the case of scalar equations (advection and Burgers), this is replaced by checking that the solution remains in-between the bounds of the initial data.

Numerical - NAD. The numerical solution in cell Ω_i is said to be Numerically admissible if it is essentially non-oscillatory. In this work we rely on a Relaxed Discrete Maximum Principle (RDMP) on the conservative variables, which writes for variable A as

$$-\delta + m_i^d \leq A_i^{n+1,*} \leq M_i^d + \delta, \tag{16}$$

where the bounds are given by $m_i^d = \min_{k\in\mathcal{S}_i^d}(A_k^n)$, $M_i^d = \max_{k\in\mathcal{S}_i^d}(A_k^n)$ and the relaxing parameter δ is fixed to $\delta = \min\left(10^{-4}, 10^{-3}|M_i^d - m_i^d|\right)$. Notice that if $\delta = 0$ then the strict DMP is enforced and the accuracy in L_∞ norm is bounded by 2.

Obviously, the criteria are crucial for the detection to be neither too permissive nor too preventive.

2.2.3 MOOD loop

The MOOD loop usually embraces the FV scheme solver. At the beginning of each time step t^n the maximal accuracy d is set for all cells and the FV scheme of order $d+1$ is employed to produce the so-called candidate solution $\boldsymbol{W}_h^{n+1,*}$. Then the detection procedure separates the valid cells which are accepted from the troubled ones.These cells demand to be re-computed by the next and more robust scheme in the cascade. The direct neighbors of a troubled cell are also sent back for re-computation because some common flux may change. Note that only the troubled cells and their neighbors are re-computed, which make the subsequent iterations of the MOOD loop relatively inexpensive. The next scheme in the cascade then produces a new candidate solution which is processed by the detection criteria. If some bad cells are still present then another iteration of the MOOD loop is performed using the next robust scheme from the cascade. If the parachute scheme has been reached, then the candidate solution is admissible by construction. In figure 1 we schematically present a high accurate *a posteriori* MOOD numerical scheme (bottom) in comparison with a classical *a priori* FV one (top).

2.3 Drawbacks of an *a posteriori* MOOD loop

The three main drawbacks of using an *a posteriori* MOOD loop are : 1- the parallelization efficiency is not optimal due to the different treatment of bad cells, 2- the explicit nature of the *a posteriori* treatment complexifies the design of an efficient implicit version, see [27, 9], and 3- the NAD criteria are not firmly based on mathematical concepts and the δ parameter in (16) still needs to be tuned. A Neural network could possibly mitigate these drawbacks. In particular the *a posteriori* detection is used to predict which scheme is the most appropriate in a given cell at the beginning of a time step without invoking a precautionary principle. A well trained neural network may be able to make an *a priori* educated guess. If so, then the *a posteriori* detection procedure could be reduced to its strict minimum, that is, only the CAD and PAD, to assure the fail-safe property of the scheme. Because

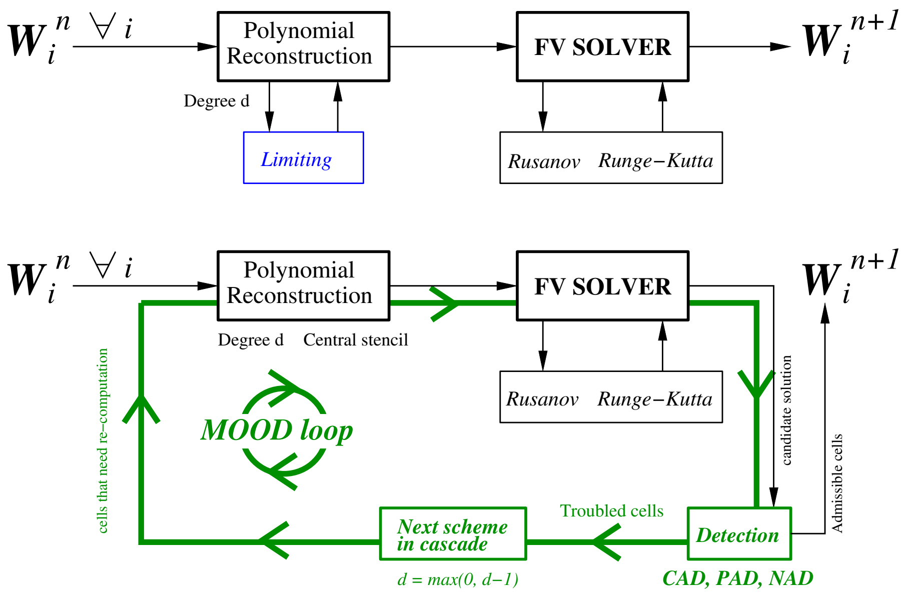

Fig. 1 Top: sketch of a FV limited scheme, for instance with slope limiter, WENO strategy, moment limiter, artificial viscosity, etc. — Bottom: a FV scheme stabilized by an *a posteriori* MOOD loop. The limiting box is replaced by an *a posteriori* MOOD loop

extremely few cells endure a CAD or PAD failure, then the parallelization would only be marginally impacted. Moreover, the neural network will guess the appropriate scheme to be used, only based on *a priori* data, therefore employing an implicit scheme becomes again practicable.

3 Neural Networks and high order FV schemes

In this section we present the basis of Neural Network (NN) theory and how it can fit in a high order FV scheme. The NN we are interested in is a feed-forward NN called a multi-layer perceptron network, see [28, 23, 40].

3.1 Perceptron

A NN is a network of elementary building blocks called perceptrons (later called neurons) which are organized in several layers. A perceptron δ_j is an entity illustrated in figure 2. It receives so-called input entries $\widehat{y}_j = (y_p)_{p=1,\dots,P}$ and delivers an output y_j. The perceptron operates a linear transformation on the input data via real weights $w_{p,j}$ as

$$u_j = \sum_{p=1}^{P} w_{p,j}\, y_p, \tag{17}$$

then, introduces non-linearity by means of an activation function f such that

$$y_j = f(u_j - b_j), \tag{18}$$

with a bias $b_j \in \mathbb{R}$. The activation function is usually common to all perceptrons of a layer and can be chosen among a lot of different functions. We use in this work a sigmoid, $f(x) = \tanh(x)$ for all layers but the last one for which a linear function $f(x) = x$ is employed. The degrees of freedom are the weights and the bias which we gather into one vector $\widetilde{w}_j = (w_1, \ldots, w_P, b_j)$ for the perceptron δ_j. They are determined by training so that the output matches some target value (see section 4.1).

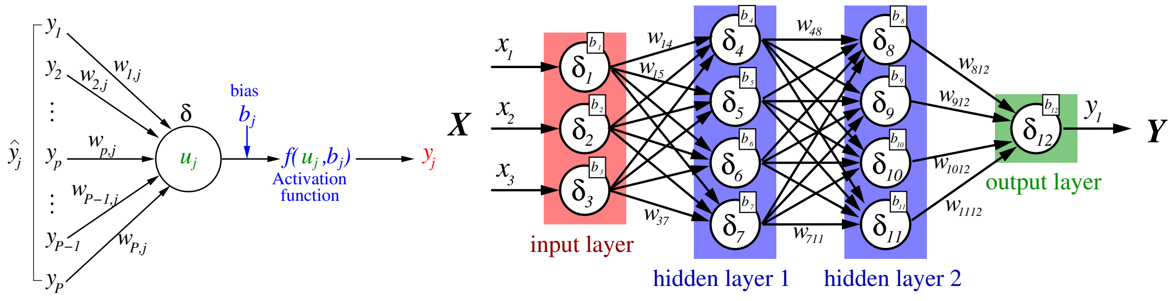

Fig. 2 Left: Illustration of the action of a perceptron δ. From a entry vector $(\widehat{y}_j)_{j=1\ldots P}$, the perceptron constructs the output $y_j = f(u_j - b_j)$ where $u_j = \sum_{p=1}^{P} w_{p,j}\, y_p$ — Right: Sketch of a neural network made of 2 inner/hidden layers of 4 neurons δ_j and the input and output layers, made respectively with 3 and 1 neurons. The weights are denoted by $w_{p,j}$ while the bias for each neuron is b_j. A weight is the link between two consecutive neurons from layer $\ell - 1$ and ℓ. The input vector is $\boldsymbol{X} = (x_1, x_2, x_3)$ and the output vector $\boldsymbol{Y} = (y_1)$ for illustration purposes. (A global numbering is adopted on this illustration to lighten the notation.)

3.2 Multi-layer perceptron (MLP) network

A multi-layer perceptron (MLP) network is a collection of perceptrons associated in L layers $\ell = 1, \ldots, L$ of $P_\ell \geq 1$ perceptrons. The first ($\ell = 1$) and last ($\ell = L$) layers are respectively called the input/source and output. The in-between layers are the hidden ones. Each perceptron $\delta_{\ell,p}$ can be identified by its layer index ℓ and its index within the layer $1 \leq p \leq P_\ell$. The signal resulting from a perceptron $\delta_{\ell,p}$ is denoted $y_{\ell,p}$ and the signal produced by a layer is the vector $y_\ell = (y_{\ell,p})_{p=1,\ldots,P_\ell}$.
A perceptron of layer ℓ, $\delta_{\ell,j}$, receives as input data the signals $y_{\ell-1}$ from the perceptrons located in the previous layer. From these data using the linear and non-linear operations (17) and (18), the perceptron produces a single output signal $y_{\ell,p}$, that is

$$y_{\ell,j} = f(u_{\ell,j} - b_{\ell,j}), \qquad \text{with} \qquad u_{\ell,j} = \sum_{p=1}^{P_{\ell-1}} w_{p,j}^{\ell}\, y_{\ell-1,p}. \tag{19}$$

$y_{\ell,j}$ is then sent to all subsequent perceptrons in the next layer $\ell + 1$.
A MLP network is therefore an association of massively inter-connected parameters *i.e* the set of all the weights and the biases for all the perceptrons. We denote the vector of input values by $\boldsymbol{X} = (x_1, x_2, \ldots, x_I)$ while the vector of output values by $\boldsymbol{Y} = (y_1, y_2, \ldots, y_O)$ where $I > 0$ and $O > 0$. The total number of perceptrons, weights and bias in a network is then given by

$$P = \sum_{\ell=1}^{L} P_\ell, \qquad W = \sum_{\ell=1}^{L-1} P_\ell P_{\ell+1}, \qquad \text{and} \qquad B = P. \tag{20}$$

A network is said to be deep when it has a large number of layers, and wide when it has a large number of perceptrons on at least one layer. In this study, only shallow networks of four layers (two hidden ones) are considered. The numbers of perceptrons in each layer is not trivial to determine and we will resort to numerical experiences to estimate them. For instance the network in figure 2 made of $L = 4$ layers made of $P_1 = 3$, $P_2 = 4$, $P_3 = 4$ and $P_4 = 1$ perceptrons each demands the determination of $B = 12$ bias and $W = 32$ weights, hence 44 unknowns.

3.3 Input and output of the NN in a FV context

A NN in an explicit MOOD FV context can be used as a decision support system in a way that is close to the approach of a multinomial logistic regression. It is designed to predict which polynomial degree d_i to use in a given cell Ω_i. In order to update the FV data $\boldsymbol{W}_i^{n+1}$, the FV scheme demands the knowledge of all cell values $\boldsymbol{W}_k^n$ in the stencil of the polynomial reconstruction, *i.e* for all $k \in \mathcal{S}_i^k$. Therefore, the input of the NN will be composed of (at least) $(\boldsymbol{W}_k^n)_{k\in\mathcal{S}_i^k}$, and, the output would be the degree of the polynomial reconstruction d_i to be employed. More precisely, the output of the NN is a real vector of 'probabilities' that degree d would have been used by the MOOD scheme. For instance for a fourth order scheme $\boldsymbol{Y}_i = (p_d)_{d=0,1,2,3}$ where p_d is the probability to use polynomial degree d. The NN should provide an ouput vector from which a clear and unique choice of degree can be extracted *i.e.* with only one value $p_d \simeq 1$. If it is not the case, then the closest p_d to 1 is selected. Ultimately the NN should replace the NAD criteria in the MOOD loop (see figure 4) .

3.4 Training the NN

The training methodology is one of the key points to build an efficient NN. Since the objective is to replace the detection criteria in the MOOD loop by the prediction of the NN, it is trained to mimic the behavior of the whole MOOD scheme including all detection criteria. To this end, a training data set is built by running the simulation of a training case during a few time steps with the MOOD scheme using all detection criteria. Elements of the training database are then created by associating an input (all cells in the stencil for a given case) to its corresponding output (the polynomial degree d of the reconstruction effectively used by the reference MOOD scheme).
In other words given data $\boldsymbol{X}_i^n$, MOOD creates the target vector

$$\boldsymbol{Y}_i^{\text{MOOD}} = \boldsymbol{e}_{d+1}, \tag{21}$$

the $(d+1)$th unit vector which is then stored in the data set. This means that for a N_t time-steps simulation made on N_c cells,the data set size is $N_t \times N_c$. The training methodology employed to build a MOOD-compatible NN is depicted on figure 3.

Remark 1 At first glance for instance for a 2nd order RK scheme, the MOOD scheme determines for any cell the degree d_i^1 for the first RK step, then d_i^2 for the second one. The pair could be taught to a NN. For a higher order RK scheme, even more degrees should be predicted which would demand larger NNs. In order to simplify the framework even further, in this study, one stores only the first step of the RK scheme. As such the NN has no notion of time discretization.

Once the training dataset is built, the genuine training ought to minimize the error made by the NN on it. That is, until convergence is reached, the following iterative procedure is applied: at each epoch, m patterns $(\boldsymbol{X}_i)_{i=1\dots m}$ are randomly picked in the database, then the weights and biases are adjusted taking into account the error $(\boldsymbol{Y}_i^{\text{MOOD}} - \boldsymbol{Y}_i)$ with a Levenberg-Marquart algorithm ([30], see also [22] for its application in the context of NN) used to compute the minimum of functional

$$J(\boldsymbol{X}_1, \dots, \boldsymbol{X}_m) = (\boldsymbol{Y}^{\text{MOOD}} - \boldsymbol{Y})^\top (\boldsymbol{Y}^{\text{MOOD}} - \boldsymbol{Y}),$$

where $\boldsymbol{Y} = (\boldsymbol{Y}_1^\top, \dots, \boldsymbol{Y}_m^\top)^\top$ and $\boldsymbol{Y}^\top$ denotes the transpose of $\boldsymbol{Y}$.

Obviously the NN should endure a relatively heavy supervised training period with a large number of patterns $\boldsymbol{X}_i$. These patterns should be representative of situations encountered during FV simulations, that is smooth, irregular and discontinuous profiles of all kinds. This training data set should be adapted depending on the system of PDEs solved to be a valid representation of the true encountered situations. The different choices made to build this data set are explained in section 4

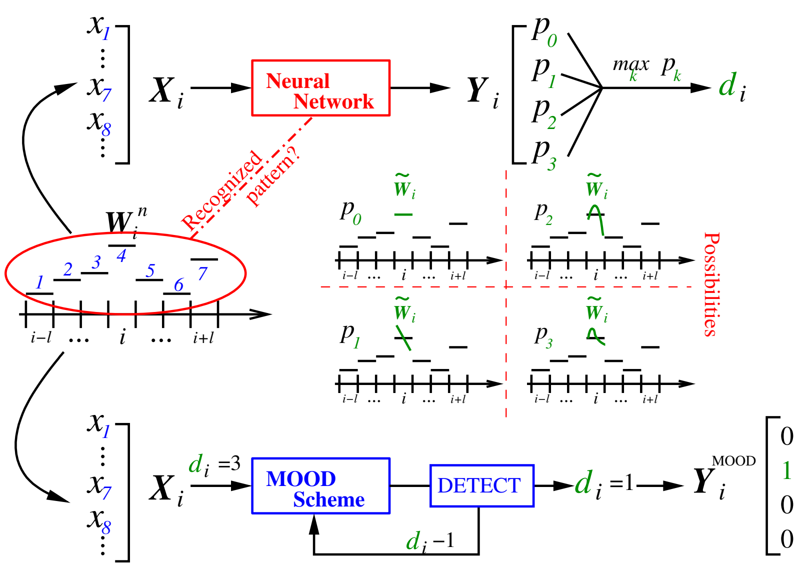

Fig. 3 Sketch of a neural network training with a MOOD FV scheme. Bottom part: Given a pattern $\boldsymbol{X}_i$ (a stencil of $\mathcal{S}_i^d$ FV data for cell Ω_i) the 4th order MOOD scheme computes the updated solution. During the MOOD detection/decrementing procedure the scheme has determined the appropriate polynomial degree $d \equiv d_i$ to use for the reconstruction procedure, leading to an output vector $\boldsymbol{Y}^{\text{MOOD}} = \boldsymbol{e}_{d+1}$ the $(d+1)$th unit vector — Top part: The NN should recognize or match at best the given pattern and predict the probability that MOOD would use the degree $d_i = 0$, 1, 2, or 3 in this situation. The degree for which the maximal probability is attained, is then selected.

3.5 NN and FV MOOD scheme

As already mentioned the NN is intended to replace the NAD detection procedure of a MOOD scheme. Instead of starting with $d_i = d_i^{\max}$ and dropping the degree according to the cascade, here, a trained NN makes an educated guess on d_i. If the candidate solution obtained with this degree is detected as invalid due to PAD or CAD criteria, then, the degree is dropped to $d_i = 0$, otherwise the solution is accepted. As a consequence the number of MOOD loop is reduced to a strict minimum because the cascade is restricted to $d_i \to 0$ where d_i is *a priori* predicted by the NN. As a rough estimate, for classical simulations, on average, few percents of troubled cells are detected. Among those cells the vast majority is flagged due to the NAD criteria and several MOOD loops (and polynomial reconstructions) are then performed. We expect to spare the associated CPU time if the NN evaluation cost is reasonable. Notice that the cost related to fixing PAD and CAD troubled cells was already negligible in a classical MOOD scheme, and, it remains so here. We can also anticipate a decrease in CPU time by parallelizing the NN more efficiently than the *a posteriori* MOOD loop. In figure 4 we present a sketch of the MOOD scheme supplemented with an *a priori* NN (red block).

4 Numerical experiments on the transport equation

In this section, the use of a NN as a predictor in the MOOD scheme is tested on the transport equation. This simple case will allow to obtain valuable insights on the practicability of the process.

4.1 Training and Validation sets for the Neural Network

Constructing training and validating data sets for the advection equation. In order to construct the training data set, we consider an initial condition H plotted on figure 5-left, which is a compound signal

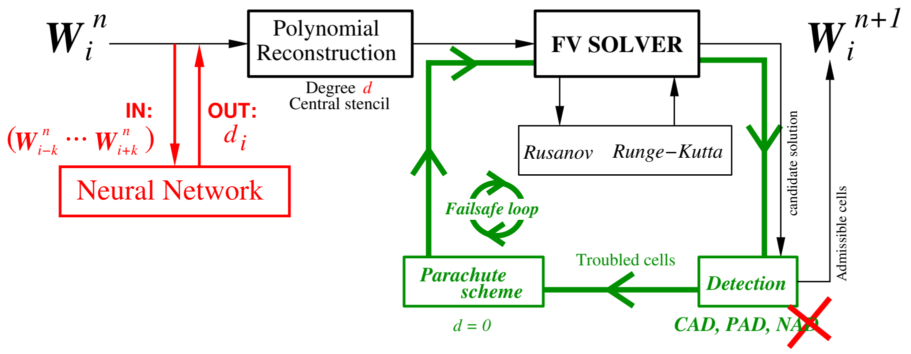

Fig. 4 sketch of a generic MOOD FV scheme with Neural Network *a priori* prediction of local polynomial degree d_i in replacement of the NAD detection criteria.

on domain $\Omega = [0, 10]$ made of Heaviside, linear, sine functions separated by plateaus. This initial condition contains all the important features for this equation. Periodic boundary conditions are considered. The numerical solution is computed by the high order MOOD scheme until a final time $T = 1$ on $M = 200$ cells. Consequently we can build a training data set by associating the mean values of the stencil, $\boldsymbol{W}_k^n$ where $k \in \mathcal{S}_i^{d^{\max}}$, gathered into $\boldsymbol{X}_i^n$ and the output vector $\boldsymbol{Y}_i^n = \boldsymbol{e}_{d+1}$ with $d = d_i^n$. Hence, a pattern $\boldsymbol{X}_i^n$ is coupled to $\boldsymbol{Y}_i^n$ for each cell i and time-step index n. The input data is normalized between -1 and $+1$ in order to reduce the size of the data set and to make it more convenient.

Remark 2 Neighborhood. For a 2nd order accurate MOOD scheme, the polynomial degrees can be 0 or $d^{\max} = 1$, and, due to the two-step RK method, the actual neighborhood $\mathcal{N}_i^{d^{\max}}$ spans cells $i-3, \ldots, i+3$. Likewise, for a nominally 4th order accurate MOOD scheme, the polynomial degrees can be $d = 0, 1, 2$ and $d^{\max} = 3$, and, due to the three-step RK method, the neighborhood $\mathcal{N}_i^{d^{\max}}$ spans cells $i-12, \ldots, i+12$. This effect is illustrated on Figure 6 for the 2nd order scheme. In this section we employ $\mathcal{N}_i^{d^{\max}} = 7/9/11$ for 2nd/3rd/4th order schemes respectively.

Remark 3 Normalisation. In all applications but this one, the input vector $\boldsymbol{X}_i^n$ is supplemented with a real scalar per variable corresponding to the normalisation factor and the Courant number. This signed normalisation factor $F = \text{sign}(\underline{x})(\overline{x} - \underline{x})$ is local to each stencil and computed with $\underline{x} = \min_{k \in \mathcal{N}_i^{d^{\max}}} x_k$ and $\overline{x} = \max_{k \in \mathcal{N}_i^{d^{\max}}} x_k$. An entry x is then re-scaled with formula: $\tilde{x} = \left(x - \dfrac{\overline{x} + \underline{x}}{2}\right)\left(\dfrac{2}{\overline{x} - \underline{x}}\right)$. As such F does not produce any shift in the data. This normalisation has obviously no impact on linear equations and therefore is omitted in this first application.

Remark 4 The design of our NNs is performed under the assumption that this normalisation factor spans a subset $[F^-; F^+] \in \mathbb{R}^+$ which corresponds to some physically relevant jump sizes depending on the non-linear system of PDEs and on the mesh resolution. Our investigations have shown that a crude sampling for F (three samples) and the Courant number allows the NNs to handle unknown situations.

Remark 5 Output functional. The output functional is not a softmax like function because we want to be able to exclude bad predictions, *i.e* output components negatives or greater than one, from acceptable ones. This operation is easier when the output vector is not re-scaled by a softmax output function. Moreover the numerical results generated when using the softmax output function gave unsatisfactory results for the test cases simulated in the numerical section.

Proceeding likewise for any of the N_t time-steps we can gather a set of $M \times N_t$ couples $(\boldsymbol{X}_i^n, \boldsymbol{Y}_i^n)$ for $1 \le i \le M$ and $1 \le n \le N_t$, leading in our case to 4800 training units. Among those, 20% are randomly chosen to construct the so-called Validation data set, $\mathcal{V}$, onto which the NN is further evaluated. The remaining 80% training units form the Training data set, $\mathcal{T}$.

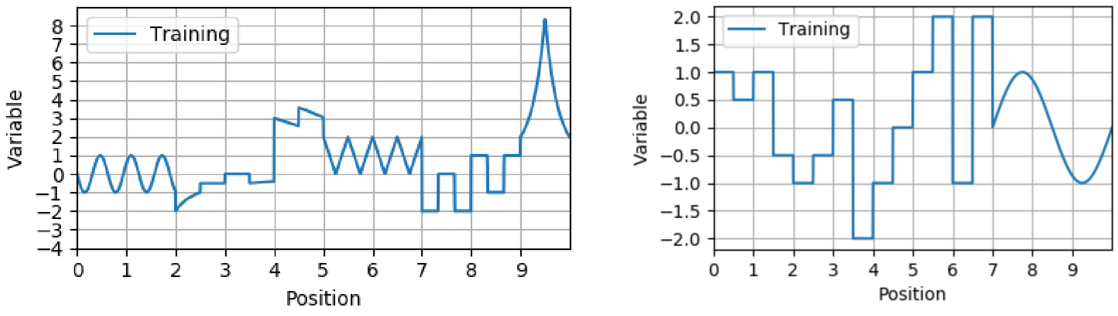

Fig. 5 Initial compound signal H used to build the training data set — Left: for the advection equation, H is constructed as a succession of discontinuities, linear and sine profiles — Right: for Burgers' equation, H is constructed as a succession of discontinuities and sine profiles.

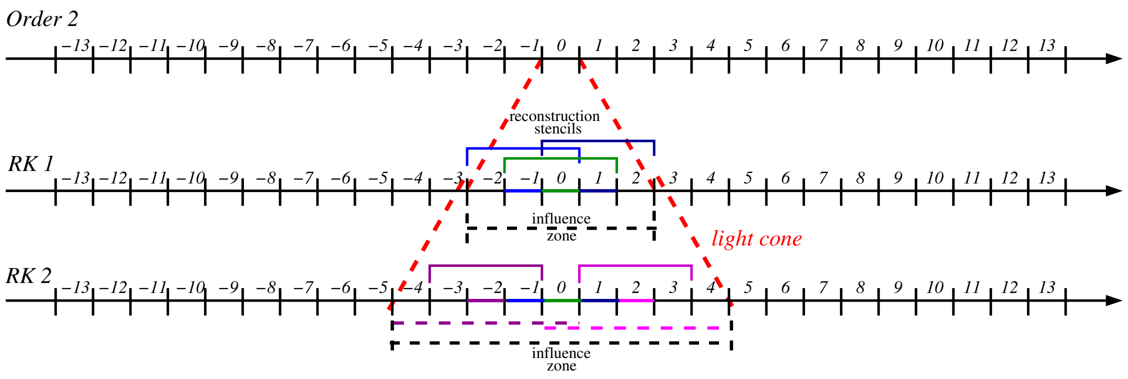

Fig. 6 Light cone for second-order FV methods: influence zone for a given cell 0 for a two-step Runge-Kutta method.

From the training data set $\mathcal{T}$ we randomly pick a couple $(\boldsymbol{X}_i^n, \boldsymbol{Y}_i^n)$ and, knowing $\boldsymbol{X}_i^n$ the NN attempts to predict $\boldsymbol{Y}_i^n$. The NN then learns from its mistakes and adapts its weights and biases accordingly. This random picking, predicting and learning operations continue up to convergence, *i.e* no mistake is no more made by the NN, or, if 3 epochs do not lead to any improvement. Then the training period is considered to be complete, the NN is stored and ready to use within the simulation code, see figure 4. The NN training performance is measured by the associated number of epochs and the misclassification percentages, that is the percentage of errors still made by the NN on the sets $\mathcal{T}$ and $\mathcal{V}$ (see table 1). More precisely these errors are computed as

$$\text{Err. on } \mathcal{S} := 100 \frac{\#\text{errors made by the NN on the dataset } \mathcal{S}}{\#\text{elements in } \mathcal{S}},$$

where $\mathcal{S}$ is either $\mathcal{T}$ or $\mathcal{V}$. It is important to recall that the validation set is composed of training units never seen by the NN. Moreover even if the NN performs perfectly on $\mathcal{T}$ and $\mathcal{V}$ it may still produce erroneous predictions for unknown situations. At last, two equivalent NNs trained with this

	Archi.	**Trainer**				**Epoch**	**Err.**	**Err.**	**Predic**	**CPU**
		Scheme	Ord.	NAD	Neigh.	#	on $\mathcal{T}$	on $\mathcal{V}$	>0.95	s
Advect.	3×3	MOOD	o3	DMP	5	1657	9.2%	9.9%	23.1%	17
	5×5	MOOD	o3	DMP	7	4230	5.8%	7.8%	20.0%	80
	10×10	MOOD	o3	DMP	9	2608	2.5%	4.8%	15.6%	188
Burgers	3×3	MOOD	o3	DMP	7	5399	8.5%	9.1%	22.6%	63
	5×5	MOOD	o3	DMP	9	1580	6.3%	7.3%	22.2%	74
	10×10	MOOD	o3	DMP	9	382	4.9%	6.3%	18.5%	78

Table 1 Neural Networks' architectures tested for the advection and Burgers' equation solved with a 3rd order accurate FV MOOD scheme. Recall that $\mathcal{T}$ is the training data set, while $\mathcal{V}$ is the validation one (made of 20% randomly picked units from $\mathcal{T}$).

procedure may generate slightly different biases and weights in an uncontrollable way. For this reason each of the NN in this paper has been retrained 5 times ensuring that its good behavior is observed at least 4 times and is not due to (un)fortunate random effects. The network used to produce the figures is then randomly picked among the 4 successful ones.

Some diagnostics on Neural Networks training. We present some diagnostics for the NN training in table 1 (top part) for the advection equation. In particular the number of epochs, the misclassification errors on $\mathcal{T}$ and $\mathcal{V}$, the percentage of prediction in the range $[0.95, 1.0]$, and the CPU time of training for three networks of 3rd order. We observe that the larger the network, the less error is produced during the training and validation. Also more confidence in its predictions is measured. Obviously the CPU time increases with larger architectures.

4.2 Test case description

Now, the trained NNs are used as *a priori* predictors of polynomial orders in the MOOD scheme on the following test-case. The computation domain is $\Omega = [0, 10]$ and the profile used for the simulation is made of four signals : a triangle, a square, a bump and an ellipsoidal profiles [26, 46]

$$W(x) = \begin{cases} |x-2| & \text{if} \quad 1.5 \le x \le 2.5, \\ 1 & \text{if} \quad 3.5 \le x \le 4.5, \\ -4(x-5.5)(x-7.5) & \text{if} \quad 5.5 \le x \le 7.5, \\ \left\{ \begin{array}{l} 1 - 2\sqrt{\frac{1}{4} - (x-7.5)^2} \text{ if } x < 8 \\ 1 - 2\sqrt{\frac{1}{4} - (x-8.5)^2} \text{ if } x \ge 8 \end{array} \right\} & \text{if} \quad 7.5 \le x \le 8.5, \\ 0 & \text{else.} \end{cases} \tag{22}$$

The same number of cells $M = 1000$ is employed and the simulations end at $t_{\text{final}} = 100$ after ten rotations of the initial profile. Obviously the exact solution at $t = t_{\text{final}}$ corresponds to the initial profile. However due to the presence of discontinuities and irregular profiles, the Gibbs phenomenon will be triggered for any high order scheme.

4.3 Numerical results: second- and third-order numerical schemes

We present the results for

- Unlim: the unlimited 1st, 2nd and 3rd order accurate FV schemes,
- MOOD: the classical 2nd or 3rd order MOOD scheme with PAD and DMP criteria,
- NN: the proposed 2nd or 3rd order FV schemes supplemented with a NN trained by MOOD.

In figure 7 we present the results obtained with a first and high order (unlimited) FV schemes along with the *a posteriori* MOOD scheme against the exact solution. The second/third order schemes results are depicted on the left/right panels. As expected the first order scheme is overly dissipative while the unlimited high order schemes generate spurious oscillations. On the contrary, the *a posteriori* MOOD scheme is able to overcome those drawbacks. Let us emphasize that while the third-order MOOD scheme produces genuinely acceptable solutions, the second-order one presents some artefacts which are particularly visible on the last two shapes on figure 8. Even though we could tweak a relaxed DMP principle to produce a second-order MOOD scheme that would overcome those issues, this is not the goal here. Contrarily we wish to show that the NNs will reproduce all scheme behaviors, should they be good or bad.

Several NN architectures' results are displayed in figure 8, namely 3×3 (top panels in red), 5×5 (middle blue), and 10×10 (bottom green) for second and third-order schemes on left and right panels respectively. All the schemes are consistently reproducing MOOD solutions, even with the

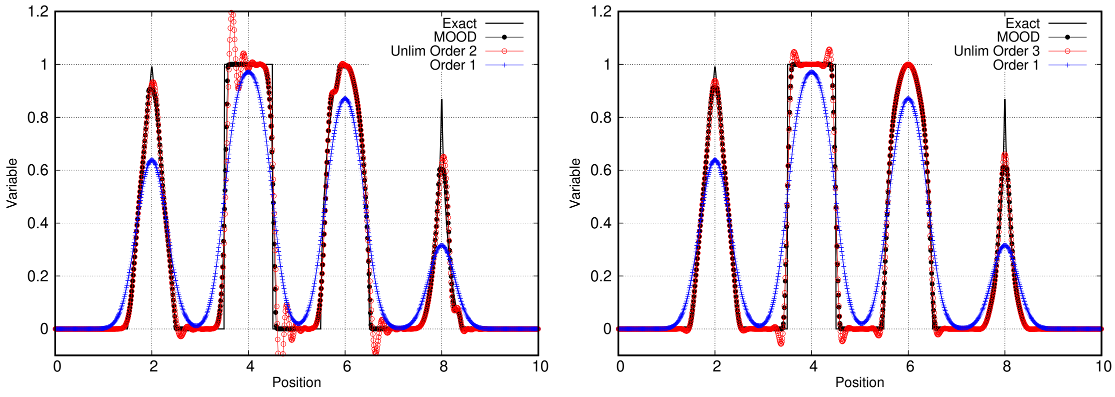

Fig. 7 Advection equation — 500 cells — 10 full turns — Exact solution in black straight line, MOOD solutions in black symbols against the first order FV scheme (blue symbol) and the and unlimited high order FV scheme (red symbols) — Left: nominally second-order schemes — Right: nominally third-order schemes.

smallest NNs. We will see on the next sections that the 3×3 NN is generally not wide enough for more complex models. However, they behave correctly here and the differences are extremely small, meaning that these NNs are able to mimic an *a posteriori* MOOD scheme for both second- and third-order schemes. Notice that the spurious effects for the second-order MOOD schemes are also retrieved, which is expected because the NNs have learned from this trainer, consequently there should be no surprise to observe the NNs also reproducing its bad behaviors.

Next, we present some quantitative results on the cell polynomial degrees chosen by the MOOD scheme and the NN versions of it. In figure 9 we present a space/time plot of the reconstruction degrees chosen by the 3×3 NN for all steps of RK, all time-steps and every 4 cells (to lighten the visualization). The results of second- and third-order schemes are respectively presented on the left and right panels. The classical *a posteriori* MOOD scheme results are displayed on top panels, while the current NN results are on bottom ones. From the plots we can observe that the vast majority of cells are updated with the maximal possible reconstruction degrees (1 for the second-order scheme and 2 for the third-one). The remaining "low" order cells are those crossed by one of the discontinuities present in the profiles. They gather on oblique and parallel lines, which is expected for these constant velocity advection simulations. Notice that the third-order scheme presents some $\mathbb{P}_1$ updated cells, justifying that this intermediate order between 3rd and 1st may be useful. The comparison with MOOD results indicates that the NNs seem less active, with nonetheless no negative impact on the final solution. Probably this extra-activity of MOOD schemes occurs on almost flat region which can accept any reconstruction degree anyway.

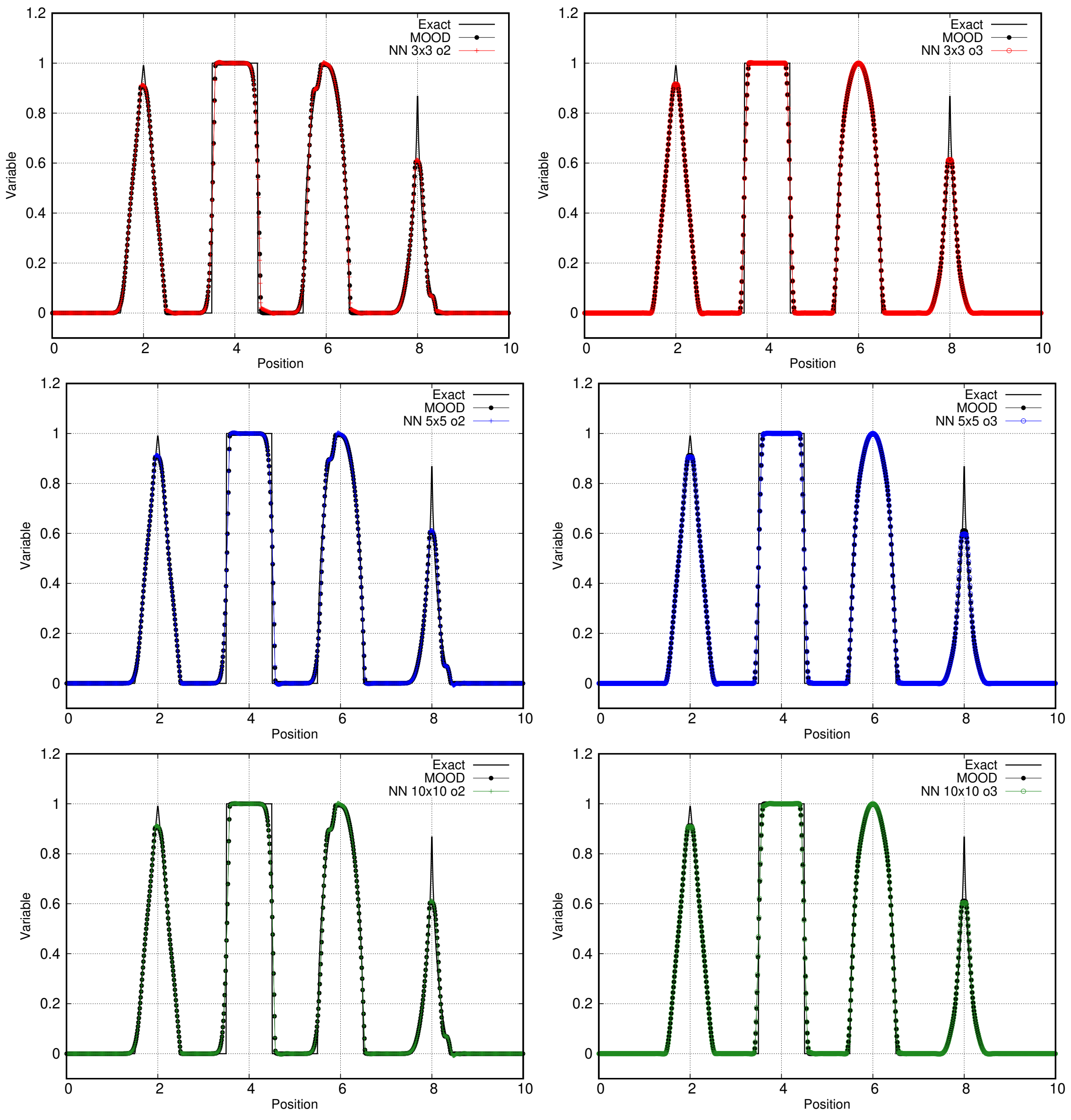

Fig. 8 Advection equations — 500 cells — 10 full turns — Exact solution in black straight line, MOOD solution in black symbols, NN solution in colored symbols. Left/right panels: second (cross)/third (circle)order schemes — Top panels: solutions for a 3×3 NN driven FV scheme (red)— Middle panels: solutions for a 5×5 NN driven FV scheme (blue)— Bottom panels: solutions for a 10×10 NN driven FV scheme (green).

A more quantitative statement is gathered in table 2 where we report the reconstruction degrees used during the simulations for the 3×3 NNs and the MOOD scheme. From the NN results we observe that more than 97% of updates are made with the maximal degree and less than $1-2\%$ updates are made with other degrees. Even if those percents are low, they are absolutely mandatory to avoid spurious effects or failure of the code. Meanwhile MOOD schemes employ about $90-96\%$ of maximal degrees and $3-6\%$ of lower orders. This shows that the NN is able to mimic the classical *a posteriori* MOOD scheme without too much extra dissipation, nor spurious oscillations. Also we report on the last column, the number of inadequate predictions made by the NN compared to the

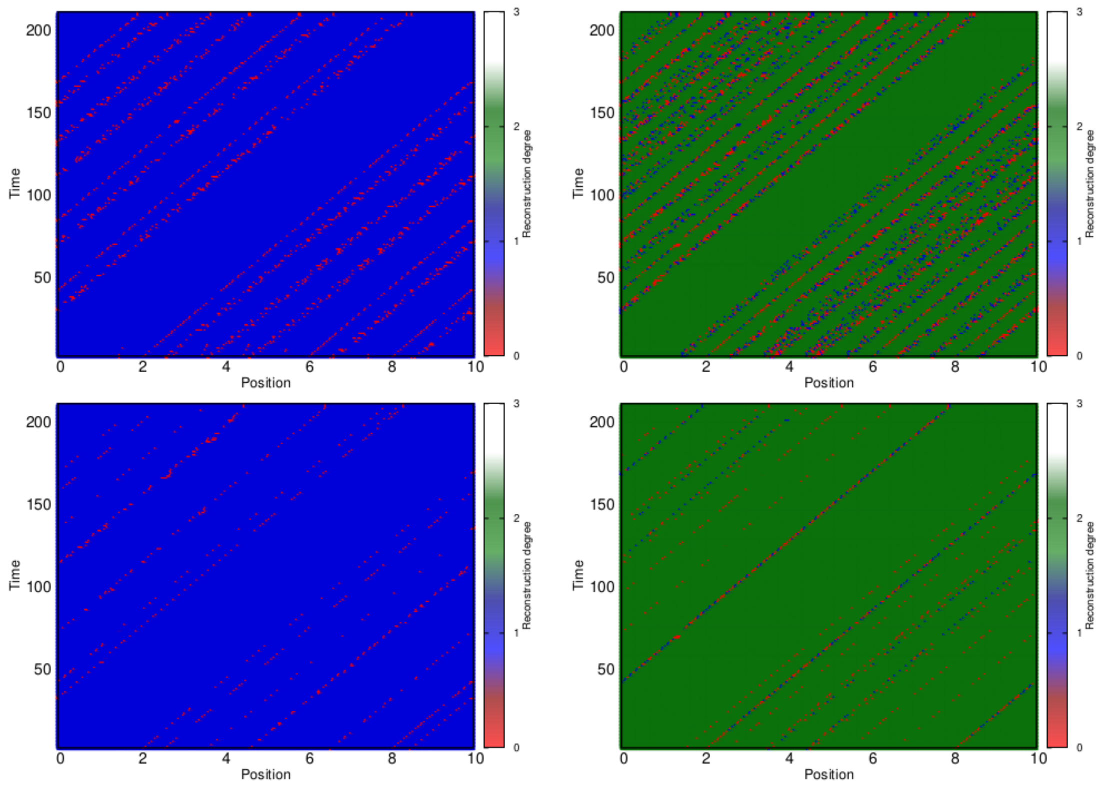

Fig. 9 Advection equation — Space/time plot — Reconstruction degree selected for each RK step and each time-step, every 4 cells — Top panels: MOOD results — Bottom panels: NN results — Left: NN order 2, $\mathbb{P}_1$ in blue and $\mathbb{P}_0$ in red — Right: NN order 3, $\mathbb{P}_2$ in green, $\mathbb{P}_1$ in blue and $\mathbb{P}_0$ in red.

classical MOOD scheme starting from the very same data at time t^n. One observes about 4 to 10% of 'wrong' prediction. Notice that most of those updates lay on smooth or flat zones, which has, indeed, no visible impact on the solution quality.

5 Numerical experiments on Burgers' equations

Let us increase the difficulty by considering Burger's equation. The problem is hence still scalar but non-linear. As a consequence, the initial profile is no more transported but the creations of shocks and rarefaction waves are expected.

5.1 Neural Network design and training

The NNs are trained starting from an initial function H shown in figure 5-right, constituted of several Riemann problems supplemented by a piece of sine function. The training database is constructed following the procedure described in section 4.1 by using the MOOD scheme with PAD and NAD criteria on $M = 200$ cells to produce a numerical solution up to time $t_{\text{final}} = 2.0$, which is about $N_t = 135$ time-steps. Shock and rarefaction waves are generated as expected, but for this short final time no wave interaction has yet occurred. The only difference with the advection training is that the signed normalization factor is now taken into account. Examples of such NNs are given in table 1 and we observe that the 3×3 NN has had some difficulty to converge. After 5400 epochs, the error

Equ.	**Scheme**	**Total** $M \times N_t \times RK$	**Degree** Degree 0	Degree 1	Degree 2	Degree 3	**% Error** vs MOOD
Advection	MOOD, o2	1000 × 210 × 2= 420 000	15669 3.7%	404331 96.3%	— —	— —	— —
Advection	MOOD, o3	1000 × 210 × 3= 630 000	36447 5.8%	227249 4.3%	566304 89.9%	— —	— —
Advection	NN 3×3, o2	1000 × 210 × 2= 420 000	6503 1.5%	413497 98.5%	— —	— —	17731 4.2%
Advection	NN 3×3, o3	1000 × 210 × 3= 630 000	11012 1.7%	3207 0.5%	615781 97.7%	— —	69395 11%
Burgers	MOOD, o2	1000 × 210 × 2= 420 000	12806 3.0%	407194 97.0%	— —	— —	— —
Burgers	MOOD, o3	1000 × 210 × 3= 630 000	19486 3.1%	7455 1.2%	603059 95.7%	— —	— —
Burgers	MOOD, o4	1000 × 210 × 5= 1 050 000	30346 2.9%	9601 0.9%	5512 0.5%	1004541 95.7%	— —
Burgers	NN 5×5, o2	1000 × 210 × 2= 420 000	11871 2.8%	408129 97.2%	— —	— —	9838 2.3%
Burgers	NN 5×5, o3	1000 × 210 × 3= 630 000	13848 2.2%	7015 1.1%	609137 96.7%	— —	23073 3.7%
Burgers	NN 5×5, o4	1000 × 210 × 5= 1 050 000	22304 2.1%	5667 0.5%	0 0.0%	1022029 97.3%	61427 5.9%

Table 2 MOOD and NN based FV schemes tested for the advection and Burgers' equation solved with a nominally 2nd, 3rd and 4th order accuracy. Diagnostics on the reconstruction degree used. N_t is the number of time-steps, M the number of cells and RK the number of Runge-Kutta steps. $M \times N_t \times RK$ is the total number of cell updates during the simulation. The last column reports the number of 'wrong' predictions of the NN compared to the answer of the classical MOOD scheme starting from the same data at each beginning of time-step.

is still of the order of 10% on non-plateau data. When more perceptrons are added in the NNs, the convergence is faster with lower errors ($\sim 5-7\%$) The NN makes bullet-proof prediction for about 80% of the data set[1].

5.2 Simulations

A test-case is performed using an initial condition given by the same function as for the advection equation (22). The final time is set to $t_{\text{final}} = 2$. At this time each discontinuity generates one rarefaction and one shock wave, possibly interacting, see figure 10 top panels. The presence of these wave interactions is interesting since no interaction is present in the training database. A reference solution is computed with the first-order MOOD scheme using $M = 20000$ cells, and is referred to as 'exact' in the figures. In this section we check the efficiency of NNs with a higher order method (fourth order instead of third order like in the previous section) to see how it can be handled. On the bottom panels of figure 10 we present the results for the first-order FV scheme and the second (left panel) and fourth-order (right panel) unlimited FV schemes. As expected, the high-order schemes without any limiting strategy do oscillate at shocks while the first-order scheme, while being slightly more dissipative, remains oscillation free. MOOD second- and fourth-order schemes seem to associate the best of these two behaviors. The results obtained by a 3×3 NN (top panels in red), a 5×5 NN (middle panels blue) and a 10×10 NN driven FV scheme (bottom panels in green) are shown on figure 11. The left/middle/right panels present the second/third/fourth-order results respectively. Different NN architectures' results are displayed from top to bottom. The smaller networks systematically produce some (small but) visible over- or under-shoots. This shows the inability of some NNs to reproduce the ultimate MOOD behaviors if they do not have weights (perceptrons and/or layers). On the other hand, the 10×10 NN solutions seem free from spurious oscillations and are able to reproduce the

[1] We must be careful that the NN may be certain of its prediction, even if it is a wrong one.

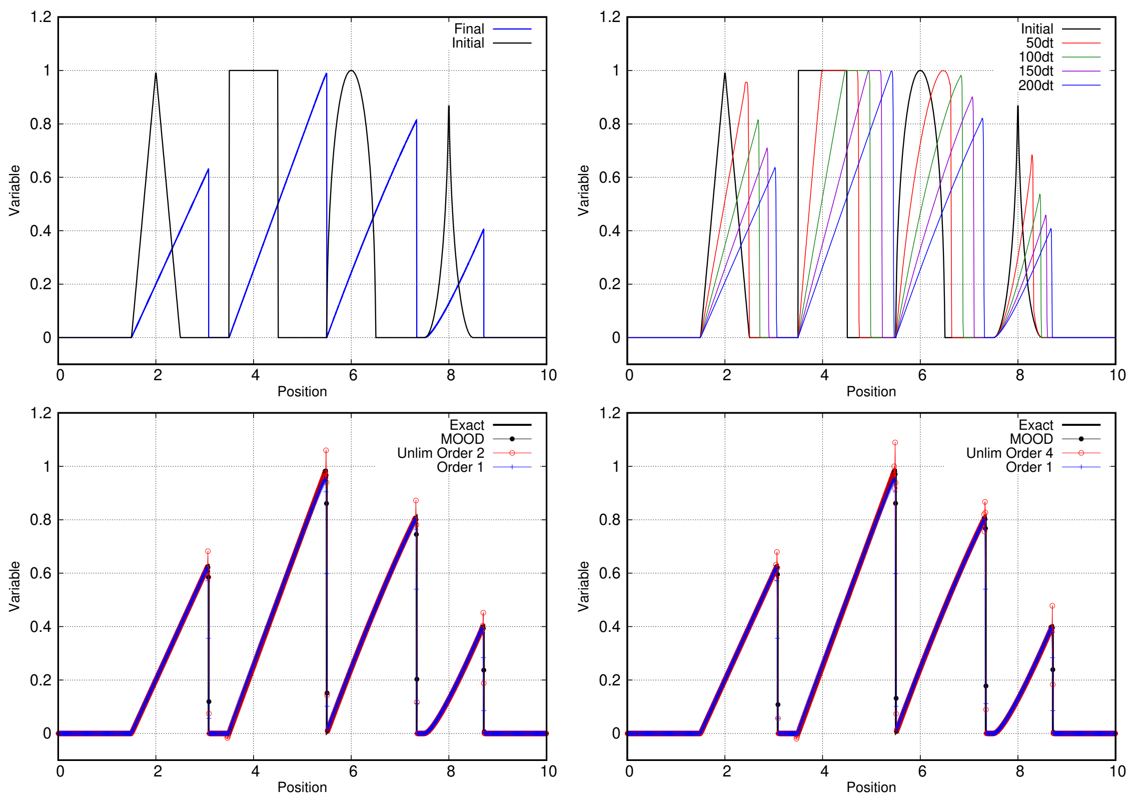

Fig. 10 Burgers' equation — $M = 1000$ cells — $t_{\text{final}} = 2$ — Top: initial and exact solutions (left) and intermediate time solutions (right) — Bottom: MOOD solutions in black symbols against the first order FV scheme (blue symbols) and the unlimited high order FV scheme (red symbols) — Left: nominally second-order scheme — Right: fourth-order scheme.

expected behavior even when fourth-order schemes are used.

Likewise for the advection equation, the space/time plot of the reconstruction degrees chosen by the 5×5 NN for all steps of RK, all time-steps (but every 3 cells to enhance the main features) is shown on figure 12. The vast majority of cells are updated with the maximal possible reconstruction degrees. The "low" order cells are located on the discontinuities generated by the initial solution and further transported. They arrange on oblique lines/curves up to their interactions, after which reflected and refracted waves may be generated. The comparison with MOOD results on top-panels indicates that the NNs seem slightly less active, but unlike the advection case, this has an impact on the solution. In particular the waves present on MOOD results around $x \in [3:5]$ do not seem to be well anticipated by the second and third-order NN based FV schemes. This explains the presence of spurious oscillations in the simulations. On the contrary the fourth-order NN results seem to mimic well MOOD behaviors.

On the bottom part of table 2 we have reported the reconstruction degrees used during the simulations of figure 12. One observes again that about 97% of updates are made with the maximal degree and less than 3% with lower degrees. Notably, for the fourth-order accurate scheme, the $\mathbb{P}_2$ reconstruction is never selected by the NN. This may be explained by the fact that the solution is either flat, linear or irregular, leading to seldom use of $\mathbb{P}_2$ reconstructions. The last column reports how many 'wrong' predictions were made by the NN compared to the use of MOOD scheme starting from the same data, leading to about from 2 to 6%. We also report the results of MOOD schemes for which a little more cells are selected with low degrees, but the main tendencies are comparable. As a

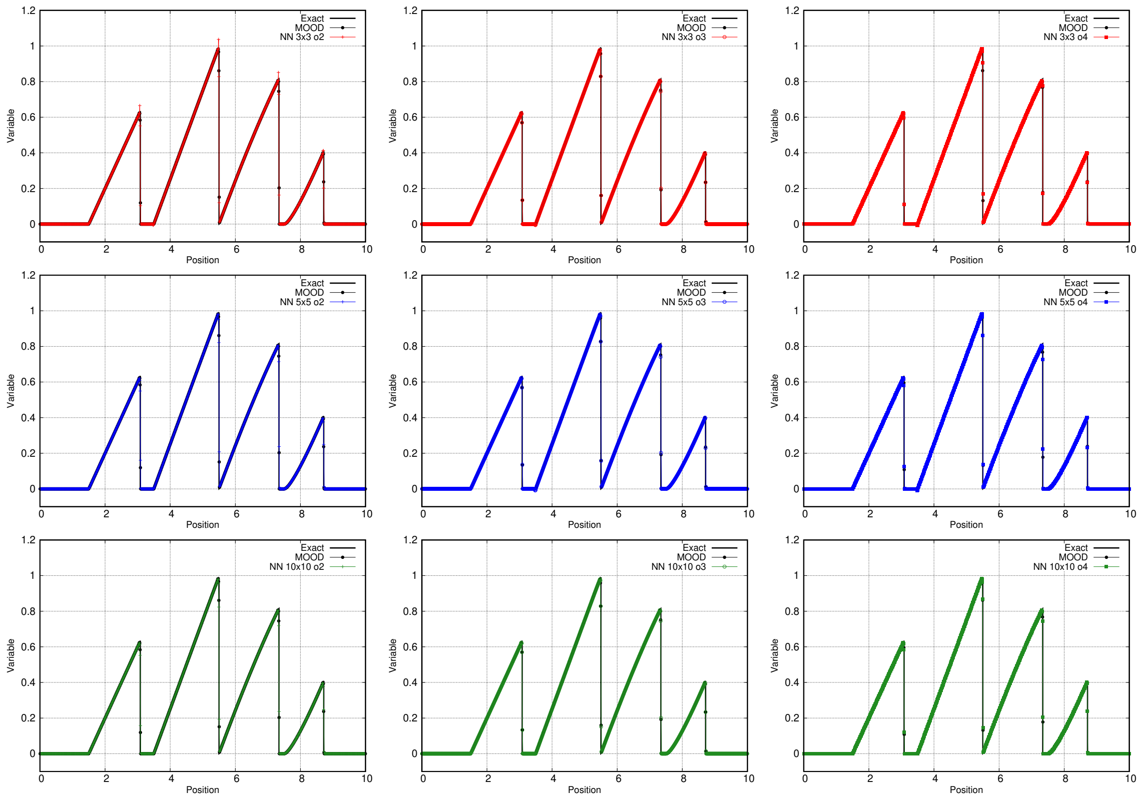

Fig. 11 Burgers' equation — $M = 1000$ cells — $t_{\text{final}} = 2$ — Exact solution in black straight line, MOOD solution in black symbols, NN solution in colored symbols. Left/right panels: second (cross)/third (circle)/fourth (square) order schemes — Top panels: solutions for a 3×3 NN driven FV scheme (red)— Middle panels: solutions for a 5×5 NN driven FV scheme (blue)— Bottom panels: solutions for a 10×10 NN driven FV scheme (green).

temporary conclusion, as long as large enough NNs are considered, they are again able to correctly guess the polynomial orders so that the approximation is very close to the one provided by the classical MOOD scheme. Fourth-order simulations are well reproduced, as well as wave interactions, despite the fact that they are not learned during training. Note that some simulations have also been carried out considering an initial condition shifted by 1, that is $1 \leq w(t = 0, x) \leq 2$. The problems are of course no longer equivalent because of the non-linearity of Burgers' equation. However, no significant discrepancy was encountered, and consequently the curves are not presented here.

6 Numerical experiments on isentropic Euler equations

We now turn to the case of systems and start with the isentropic Euler equations, whose stiffness remains generally moderate. In addition to the inherent difficulty of using a system, it is now also necessary to preserve the set of admissible states (5), namely the positivity of the density here.

6.1 Neural Network design and training

The input data are constituted of both the cell density ρ and the velocity u in each cell of the stencil in addition to the CFL number and the normalisation factor for each variable. The size of the input vector depends on the size of the stencil $\mathcal{N}$ which depends on the maximal polynomial degree $d^{\max}$.

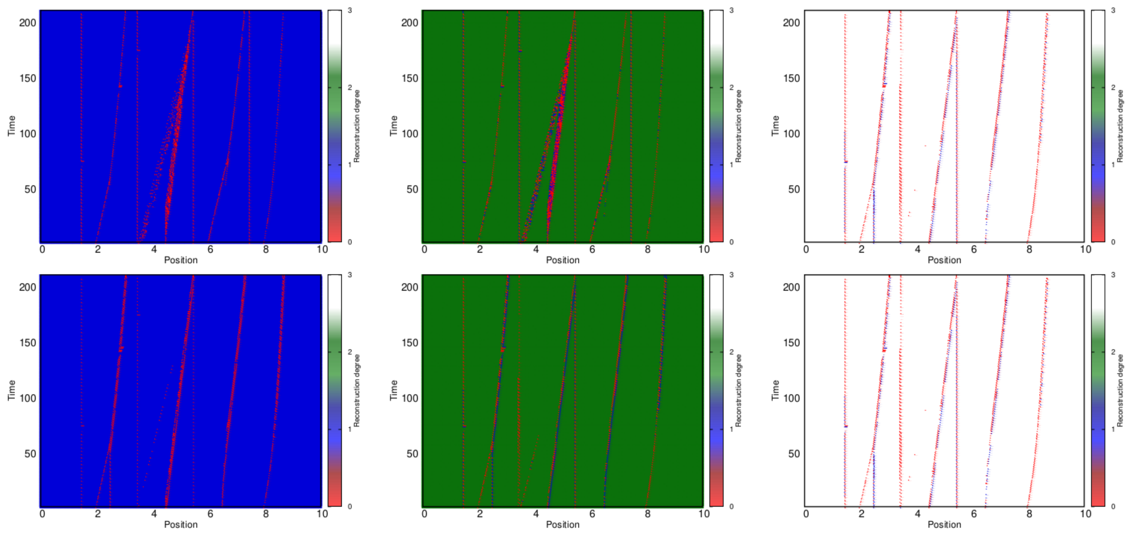

Fig. 12 Burgers' equation — Space/time plot — Reconstruction degree selected for each RK step and each time-step, every 3 cells — Top panels: MOOD results — Bottom panels: 5×5 NN results — Left: NN order 2, $\mathbb{P}_1$ cells in blue and $\mathbb{P}_0$ ones in red — Middle: NN order 3, add $\mathbb{P}_2$ cells in green — Right: NN order 4, add $\mathbb{P}_3$ cells in white.

	ρ_L	u_L	ρ_R	u_R
RP1	1	0	0.5	0
RP2	0.5	0	1	0
RP3	1	-1	0.5	1
RP4	1	1	0.5	-1
RP5	1	-1	1	1
RP6	1	1	1	-1

Table 3 Riemann problems used to train the NNs in the case of the hydrodynamics system of PDEs.

This means that for a stencil of size $|\mathcal{N}| = s$, the size of the input data is $2s + 3$. The output vector is constituted by the actual polynomial degree used by the target MOOD scheme using all detection criteria. The training data set is constructed by solving the following Riemann problems (and their symmetric version) depicted in table 3, with 5 different CFL numbers (0.2, 0.4, 0.6, 0.8 and 0.95), leading to 72 states. The target MOOD scheme of appropriate order is used with $M = 2000$ cells (~ 25 cells per state) on $\Omega_{\mathcal{T}} = [0 : 1]$ up to time $t = 0.002$, leading to $N_t = 21$ time-steps, yielding about 55000 training units in the database. Notice that with only 21 time-steps, interactions between waves do not occur, meaning that the training does not explicitly take into account such situations.

6.2 Simulations

Riemann Problem (RP). We first test the prediction of NNs on a problem of average difficulty, namely a Riemann problem which, if not available in the database, corresponds to the RP1 for which the data are only doubled. The computational domain is set to $\Omega = [-1 : 1]$ meshed with $M = 100$ cells with the following initial conditions

$$\boldsymbol{H}^0(x) = (\rho^0(x), u^0(x)) = \begin{cases} (2.0, 0.0) & \text{if } -1 \le x \le 0, \\ (1.0, 0.0) & \text{if } \ 0 \le x \le 1, \end{cases} \tag{23}$$

leading to a left moving rarefaction and a right moving shock waves. The exact solution is presented in figure 13 and zoomed on $[-0.75 : 0.75]$ at final time $t_{\text{final}} = 0.25$. On the same figure we also plot the numerical solutions obtained with a first, second (left panel) or fourth-order (right panel) unlimited and MOOD schemes. In figure 14 we present the density obtained by the 2nd and 4th order

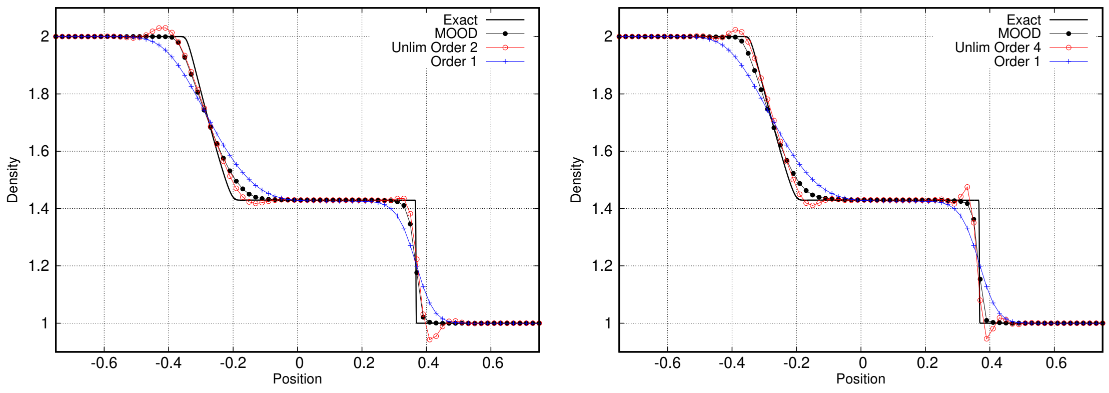

Fig. 13 Isentropic Euler equation — Riemann problem — Simulation at $t_{\text{final}} = 0.25$ — Comparison of the 1st and 2nd order unlimited FV schemes and 2nd order MOOD against the exact solution (black line). Density variable.

FV schemes supplemented with a 3×3, 5×5 and 10×10 NNs (from top to bottom panels) against the associated classical MOOD schemes. We observe that the smallest NN does not seem to be able to reproduce MOOD results for 2nd order, meaning that the first-order scheme (degree 0) is chosen too often leading to excessive diffusion. Clearly this NN was not able to train appropriately, it was not large enough for this task. For the 4th order scheme, the NN predicts more 'good' choices amongst high degrees: $3, 2, 1$. The larger NNs seem to reasonably well reproduce MOOD results, although some minor inaccuracies occur. In figure 15-16 we present the polynomial degrees employed by the second and fourth-order MOOD (left and right panels) and the three NNs FV schemes for the two RK steps respectively (top to bottom panels). The possibilities are restricted to degrees $d = 0$ (red) and 1 (blue) for the second-order schemes and degrees $d = 0, 1, 2$ and 3 for the fourth-order one. On these figures, we can clearly see the areas in which the NNs predict different orders than MOOD does. Apart from the plateaus, on which the prediction does not significantly impact the results, one can clearly see that the other differences have an impact on the quality of the solution. The main waves seem to be detected by all NNs. However the smallest and largest NNs are more conservative, leading the activation of more first-order cells. Without any clear explanation MOOD seems to detect some troubled situation around position $x \simeq 0$, which is a plateau as soon as the waves move away. The NNs have the same prediction but only for very few time steps.

A Riemann problem with wave interactions and close-to-vacuum states (RP2). This next test case is designed to be challenging for NNs. To this end, we propose to simulate situations which have never been taught to them, including wave interactions and close-to-vacuum states. This test-case is genuinely difficult because it involves very low densities with values under machine precision. The computational domain is set to $\Omega = [-2:2]$ meshed with $M = 500$ cells. The initial conditions

$$\boldsymbol{H}^0(x) = (\rho^0(x), u^0(x)) = \begin{cases} (0.5, 0.0) & \text{if} \quad x \leq -1, \\ (0.1, -4.0) & \text{if} \; -1 \leq x \leq 0, \\ (0.1, 4.0) & \text{if} \quad 0 \leq x \leq 1, \\ (0.5, 0.0) & \text{if} \quad x > 1, \end{cases} \tag{24}$$

lead to an almost vacuum area close to $x = 0$ after two rarefaction waves move outwards, following two faster diverging shock waves. Ultimately, at later time, the central rarefaction waves interact with the shocks, see figure 17 for a time evolution of the solution and zooms on interesting regions. The

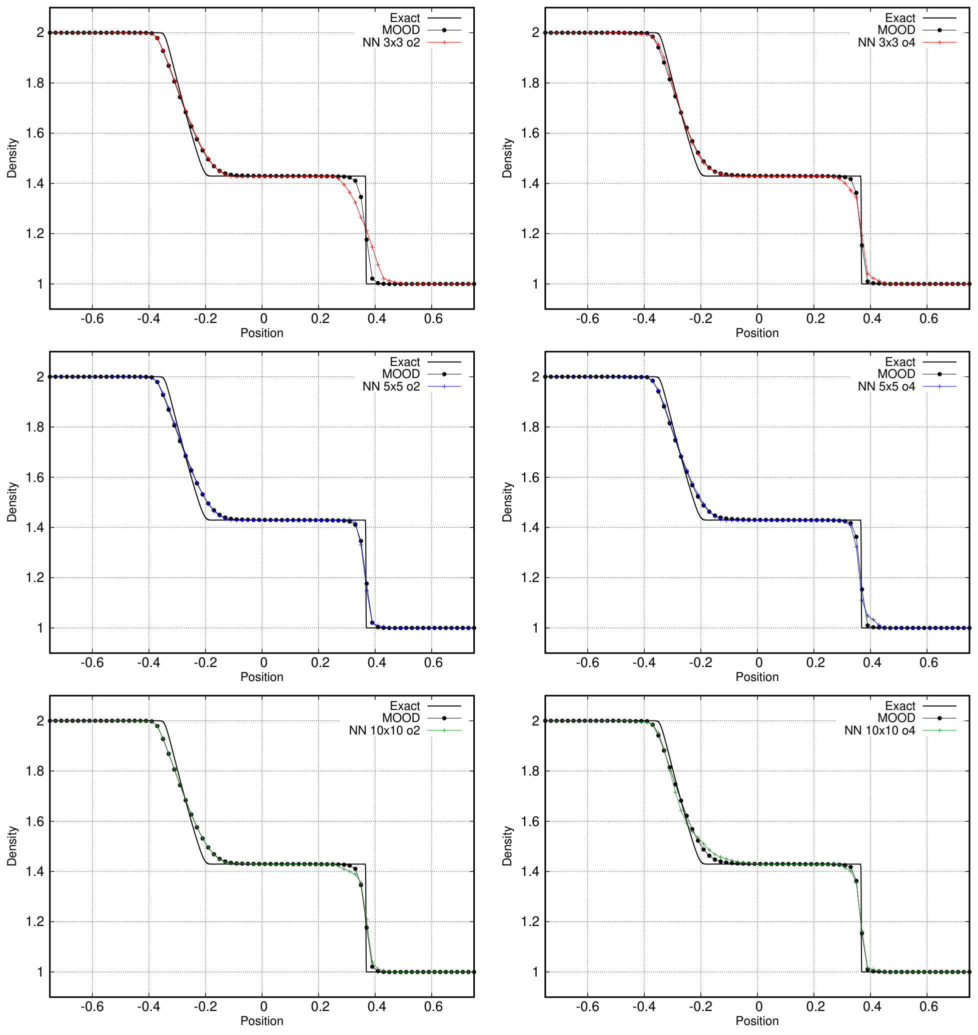

Fig. 14 Isentropic Euler equation — Riemann problem — Simulation at $t_{\text{final}} = 0.25$ — Density — Left-panels: 2nd order NN vs MOOD schemes results — Right-panels: 4th order NN vs MOOD schemes results — Top: 3×3 NN — Middle: 5×5 NN — Bottom: 10×10 NN — The exact solution is displayed in straight black line.

final time is set to $t_{\text{final}} = 0.25$ and the CFL number is set to 0.8. The minimal density of the exact solution is roughly 10^{-12} and the first order FV scheme produces densities of the order 2×10^{-17}. This usually generates robustness difficulty for any high order numerical method.

A first set of results is given by the first-order FV scheme which is naturally able to handle the close-to vacuum state as well as the shocks without oscillations to the price of an excessive numerical dissipation, see figure 18. From a $M = 10000$ cell mesh we have computed a reference solution with the first-order FV scheme. We also display the numerical solutions produced by a 2nd and 4th order FV scheme *a posteriori* limited by the PAD only; that is, only the occurrence of negative density does trigger the decrementing procedure, yielding to a fail-safe FV scheme not preventing any spurious oscillations[2]. At last we present in figure 19 the results obtained by the genuine MOOD schemes and

[2] Notice that without the *a posteriori* MOOD loop the 2nd and 4th order schemes crash due to the occurrence of negative densities.

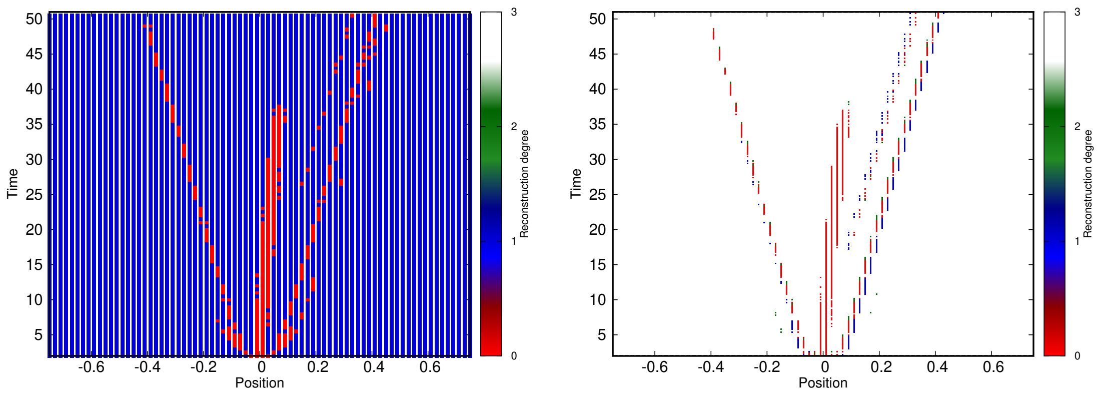

Fig. 15 Isentropic Euler equation — Riemann problem — Simulation at $t_{\mathrm{final}} = 0.25$ — Polynomial degrees for MOOD scheme — Left-panel: 2nd order NN results — Right-panel: 4th order NN results

a FV scheme using the following NN architectures: 3×3, 5×5 and 10×10, from top to bottom and 2nd and 4th order scheme (left and right panels respectively).

As expected the MOOD schemes, used to train the NNs behave appropriately, while the NNs seem to perform nicely with almost the same quality as MOOD. Some slight over or undershoots may however been observed. Nonetheless all schemes are robust in the presence of almost vacuum states and after the wave interactions.

Notice that due to the statistical nature of the training, the NNs of the same structure do not produce exactly the same numerical solutions. A few NNs have presented slight spurious over/undershoots close to the shock waves while others were producing good results. The polynomial degrees employed by MOOD and the NNs is also shown on figures 20-21. First of all the *a posteriori* MOOD loop for the positivity is truly active in a zone starting at position $x = 0$ and spanning up to $[-0.5 : 0.5]$ at final time (recall that red cells correspond to first-order updated cells). Then the main discontinuities, i.e the head and tail of rarefaction and shock waves, are captured and followed in time. They are captured and followed by the NNs for second and fourth-orders. Again the plateaus are approximated by the maximal polynomial degree but for the 5×5 NN the 4th order scheme uses $\mathbb{P}_0$ reconstructions without any visible problem even if it is not expected. This highlights the importance for the NN to be right mainly in the important or difficult situations, elsewhere the impact of a wrong prediction is less dramatic. Notice that the colors in figure 21 correspond only to the prediction of the NN before the *a posteriori* positivity loop. Therefore the central zone is not colored in red, but, nonetheless, those cells are necessarily updated with first-order. It is important to note that the *a posteriori* MOOD loop has to be maintained for the PAD criteria in order to guarantee a failsafe algorithm by ensuring that the numerical solution remains in the admissible set $\mathcal{A}$ in (5). Aside from this, it seems that the use of well trained NNs leads to approximate numerical solutions of the same quality as the original MOOD scheme.

7 Numerical experiments on the M1 model

Finally, the behavior of NNs to predict polynomial order is investigated on the M1 model for radiative transfer in transparent media (6). For such a model, if SI units are used then generally quantities very far from 1 are observed, such as characteristic times in nanoseconds and large amplitudes for E and F. In addition, there is a definite stiffness at the limit of the admissibility domain when $|F| \simeq cE$. Finally, Riemann's problems are genuinely different from those for Euler's equations [39]. Among the notable differences are the shape of the Hugoniot curves, the profile of the rarefaction waves

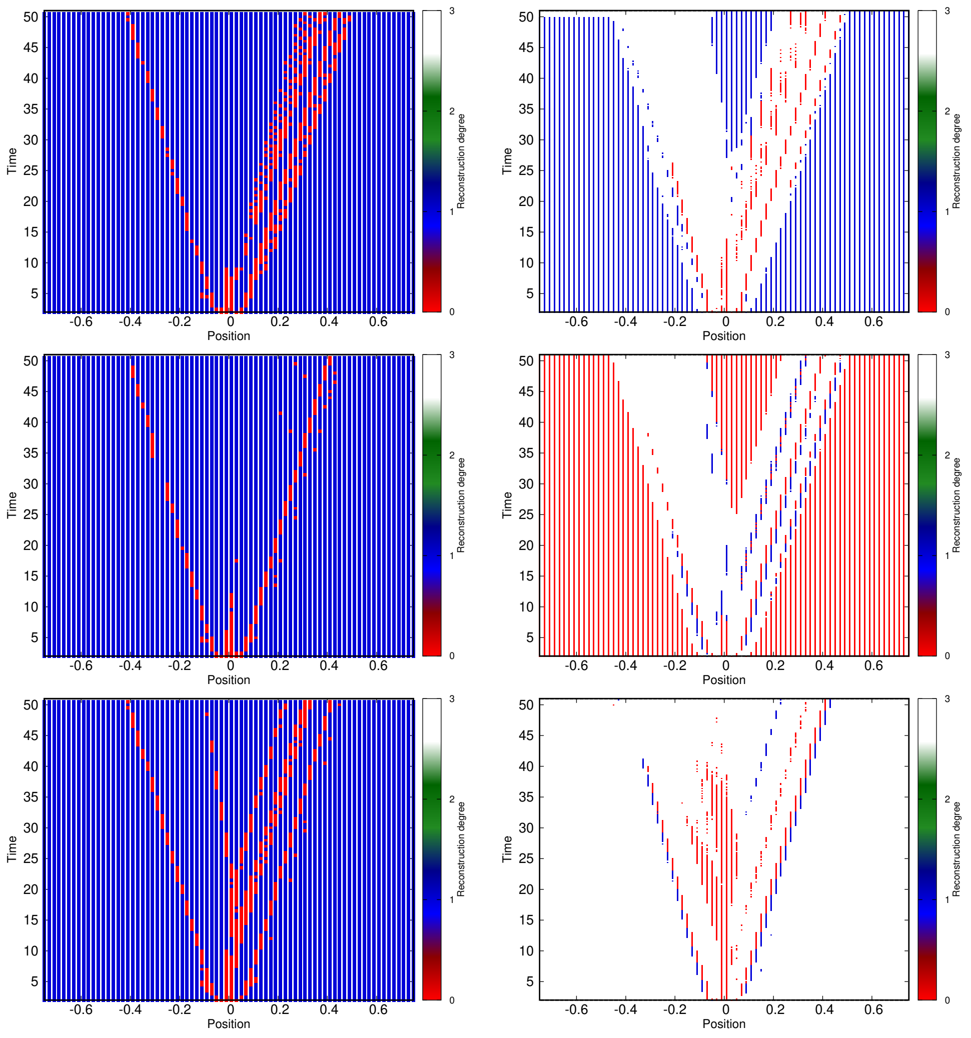

Fig. 16 Isentropic Euler equation — Riemann problem — Simulation at $t_{\text{final}} = 0.25$ — Polynomial degrees — Left-panels: 2nd order NN results — Right-panels: 4th order NN results — Top: 3×3 NN — Middle: 5×5 NN — Bottom: 10×10 NN.

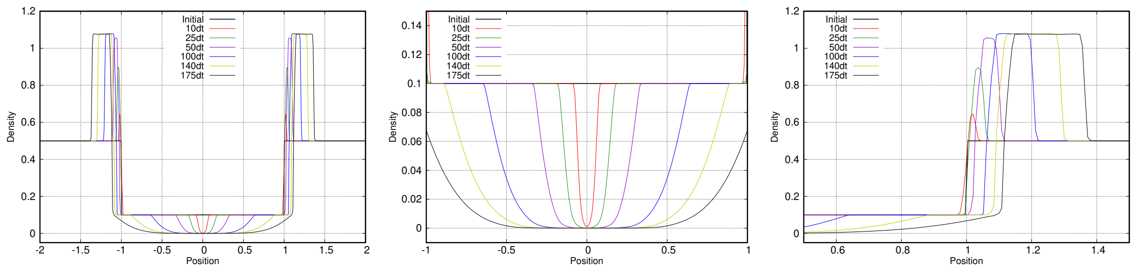

Fig. 17 Isentropic Euler equation — (RP2) — Density — Time evolution of the solution on the full domain (left) and zooms in the central rarefied region (middle) and wave interaction and shock region (right)

and the fact that the variables used to solve the problem are complex nonlinear combinations of the conservative variables.

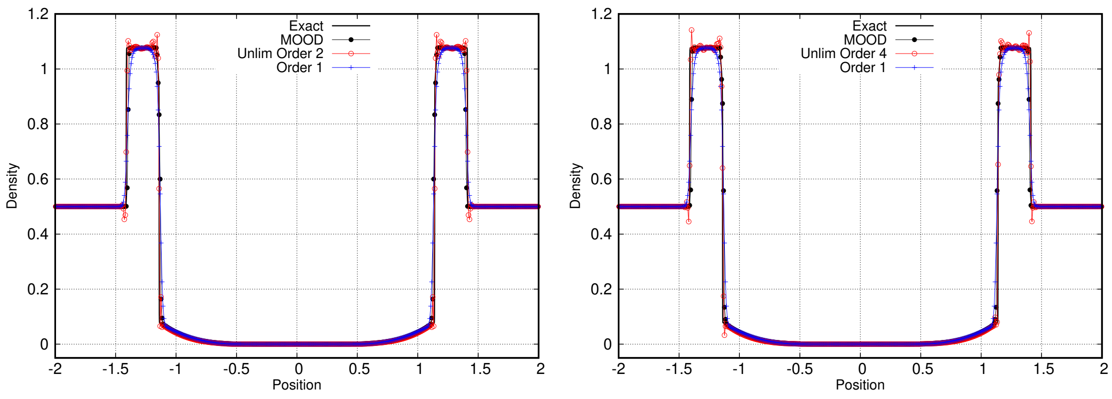

Fig. 18 Isentropic Euler equation — (RP2) — Density — Comparison of the first and 4th order schemes (unlimited, classical MOOD and NN) against the reference solution (black line) at final time $t_{\text{final}} = 0.25$.

	E_L	F_L	E_R	F_R
RP1	$E_0 \times 10^{-6}$	0	1.0×10^{-6}	0
RP2	$E_0 \times 10^{-6}$	0	$E_0 \times 10^{-6}$	-100
RP3	$E_0 \times 10^{-6}$	-100	1.0×10^{-6}	100
RP4	$E_0 \times 10^{-6}$	100	$E_0 \times 10^{-6}$	-100
RP5	$E_0 \times 10^{-6}$	100	1.0×10^{-6}	-100

Table 4 Riemann problems used to train the NNs in the case of the M1 model system of PDEs.

7.1 Neural Network design and training

The input data are constituted of both the radiative energy and the radiative flux in each cell of the stencil in addition to the normalisation factor for each variable. This means that for a stencil of size s, the size of the input data is $2s + 2$. The output vector is constituted by the actual polynomial degree used by the MOOD scheme using all detection criteria. The training data-set is constructed by solving the following Riemann problems (and their symmetric version) depicted in table 4, for 6 different values of E_0 ($E_0 \in \{2; 3; 5; 7; 10; 15\}$), leading to 66 states. The Riemann problems are solved until time $t = 1.0 \times 10^{-10}$ with a 2000 cell mesh, leading to $N_t = 150$. The first 15 time-steps are stored, then only one every five is further used to build the training data-set. This leads to a training data-set of 27000 units.

7.2 Simulations

The computational domain is set to $\Omega = [-1 : 1]$ and the Riemann problem located at $x = 0$ is such that the radiative energy is $(E_L, E_R) = (1.211, 6.128) \times 10^{-6}$ and radiative fluxes are set to 0. This problem is obviously not in the training data set. At final time, $t_{\text{final}} = 2 \times 10^{-9}$ the exact solution is composed of a left-moving shock wave located at $x_S \simeq -0.4204$ and a right-moving rarefaction wave between $x_F \simeq 0.1782$ and $x_H \simeq 0.3462$, see figure 22.

The numerical solutions obtained by the first-order (blue symbols), second-order unlimited (red symbols) and MOOD schemes (black symbols) are plotted on figure 22-left versus the exact solution at final time. A zoom on $[-0.5 : 0.5]$ is displayed. As expected the first-order scheme produces dissipative results while the 3rd order one creates spurious oscillations. On the other hand, the 3rd order MOOD scheme is able to create an accurate and non-oscillatory solution. At last on the same figure one plots the space-time distribution of the degree of the polynomial reconstructions for MOOD scheme (for all cells and all RK steps). We extract from these data that the degree is $d = 2$ for 93% of the updates, $d = 1$ for 1% and $d = 0$ for 5%, see table 5. Now, the results obtained with 3rd

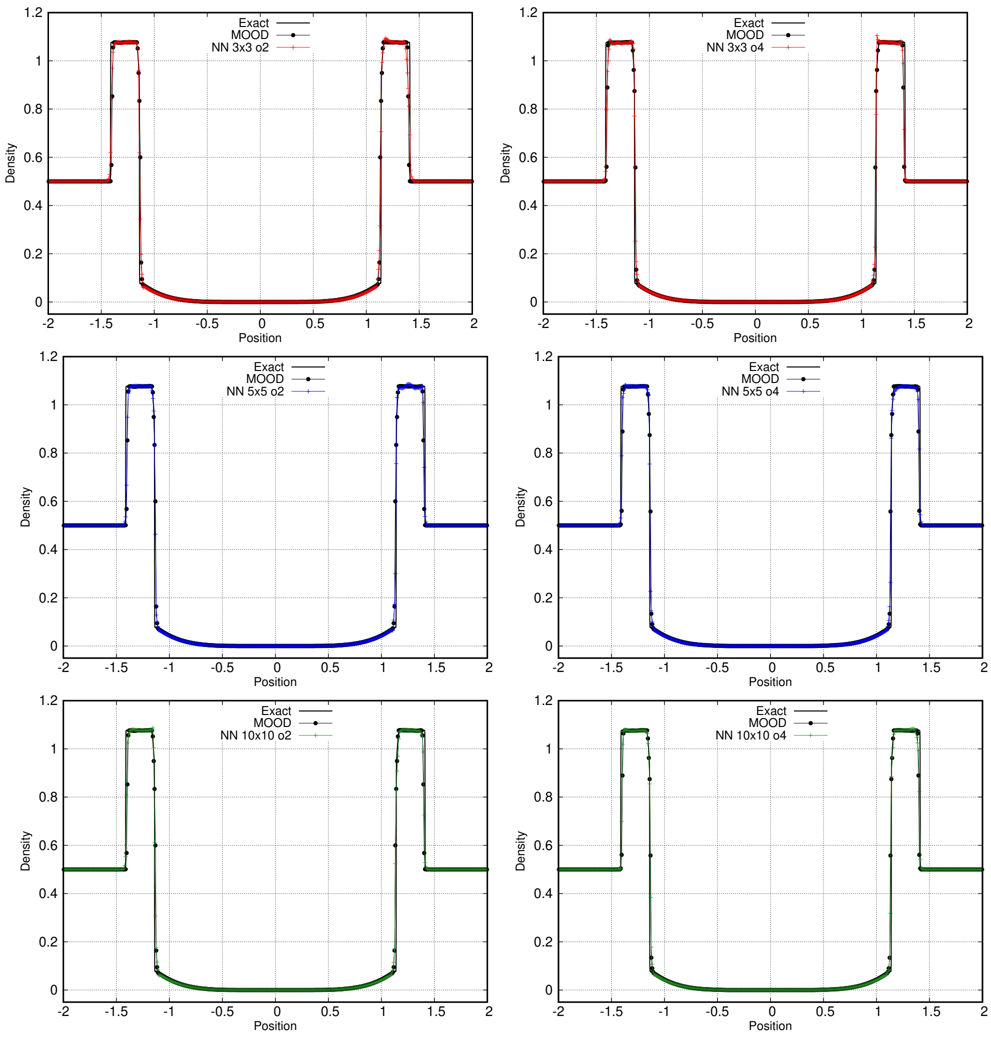

Fig. 19 Isentropic Euler equation — (RP2) — Density — Left-panels: 2nd order NN vs MOOD schemes results — Right-panels: 4th order NN vs MOOD schemes results — Top: 3×3 NN — Middle: 5×5 NN — Bottom: 10×10 — The exact solution is displayed in straight black line. at final time $t_{\text{final}} = 0.25$.

order accurate schemes supplemented with a 10×10 and 20×20 NN are presented in figure 23. The numerical results are extremely close to the original MOOD scheme. The bottom panels present the polynomial degrees and we observe that the distribution does not match MOOD's results from figure 22. The two networks do not even fully agree with each other, the smallest one overestimates the need for low order reconstructions, and both seem over-conservative on the rarefaction wave compared to the classical MOOD scheme. These facts are visible on table 5 where these numbers are reported altogether in terms of number of cells and percentages. The larger NN seems to be closer to the expected behavior by selecting more second-order reconstructions and a lot less first-order ones, following the lead of MOOD. These differences, which may seem enormous at first glance, should be balanced by the fact that they are mainly located on plateaus. Now we specifically focus on the three discontinuous features: shock, tail and head of the rarefaction waves, the position of which are exactly known as a function of time, and denoted $x_w(t)$. The wave is located in a unique cell at any t^n, $x_w(t^n) \in [x_{i-1/2}, x_{i+1/2}]$ and we consider 10 cells on each side of cell i. In figure 24 we re-plot

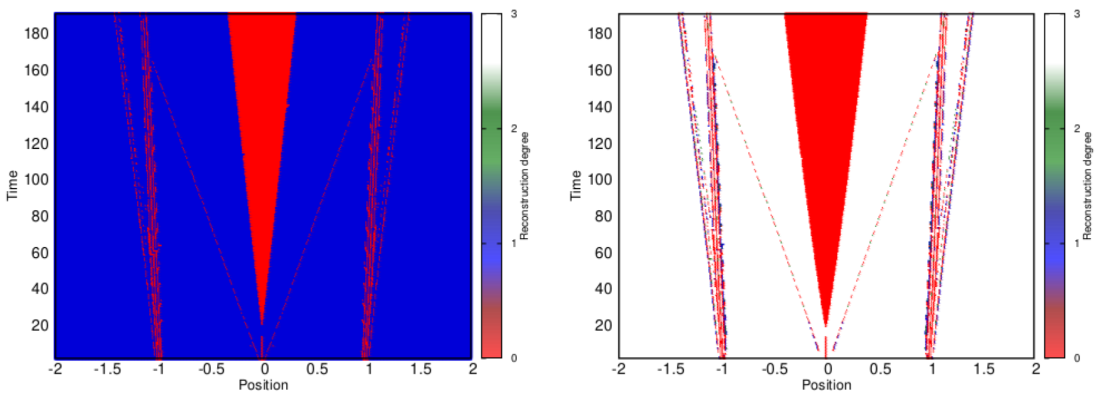

Fig. 20 Isentropic Euler equation — (RP2) — Simulation at $t_{\text{final}} = 0.25$ — Polynomial degrees for MOOD schemes — Left-panel: 2nd order results — Right-panel: 4th order results. The large central red zone corresponds to the positivity preserving action of the MOOD scheme which must decrement the accuracy to first-order on these cells.

Archi.	Number of cells			Percentages		
	$\mathbb{P}_0$	$\mathbb{P}_1$	$\mathbb{P}_2$	$\mathbb{P}_0$	$\mathbb{P}_1$	$\mathbb{P}_2$
MOOD	23492	6023	420485	5.2%	1.3%	93.4%
10×10	90886	3187	355927	20.2%	0.7%	79.1%
20×20	63149	7782	379069	14.0%	1.7%	84.2%

Table 5 M1 model test problem. Number of cells treated with first-, second- and third-order accurate reconstructions (for all cells, all time-steps and all RK steps) and their relative percentages.

the polynomial degrees from figure 23-22 but for these cells only. The cell containing the wave is centered in the middle of its band. We can precisely observe what are the degrees predicted by the schemes in front and after the waves. Qualitatively the NNs are able to predict the MOOD degrees for those very important regions. This explains why the NNs produce good solutions even if some (less important) regions seem wrongly predicted. It is important to note that in the case of the M1 model, the training dataset has revealed to be extremely sensitive during the NN construction. Some inappropriate construction of the data-set inexorably leads to oscillatory solutions.

Another feature encountered here is the fact that many cells were predicted by NNs with a relatively low confidence level, sometimes even artificially large or negative values are computed. In order to investigate the impact of this fact, two additional tests are carried out for a 20×20 NN and a third order FV scheme. The first one, shown in figure 25 was obtained by randomly choosing the polynomial degree between 0, 1, and 2 (top panels). The goal is to see if a pure random choice may not produce the same solution quality. We directly observe that, as expected, some excessive diffusion is generated by such an approach and the overall accuracy corresponds to no more than a first-order FV result, see figure 22.

The second test is performed by resorting to a randomly chosen polynomial degree whenever the NNs produce unsure predictions. In other words if we have not $0.9 \leq p_k \leq 1.1$ then the degree is randomly chosen. The results are presented on the bottom panels of figure 25. The overall quality of the solution is not changed. This figure proves that even if the NN produces seemingly uncertain predictions, then those are still the "best" ones according to its training. Moreover the NN is confident about its predictions around the shock wave, which is the most sensitive location.

The FV scheme with NNs produce acceptable numerical solutions with a reasonably small number of perceptrons. The M1 model being stiffer a 5×5 NN is not large enough and produces oscillatory results, while 10×10 and 20×20 NNs can match MOOD solution quality. Even if some NNs lack of certainty in large zones, it still produce better results than a pure random choice method. At last we present a rough estimate of the CPU times generate by the classical 1D MOOD scheme against its NN versions in table 6 solving the first Riemann problem of this section with 3rd order accurate

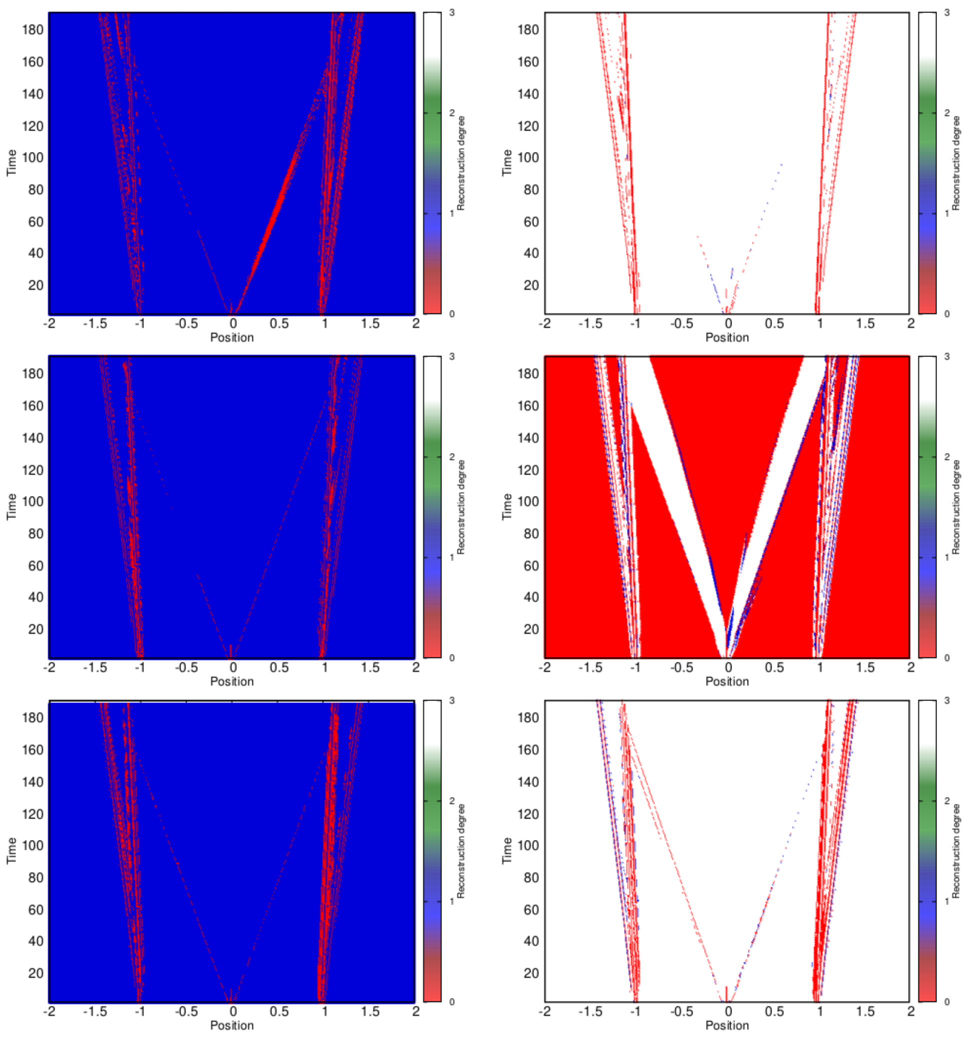

Fig. 21 Isentropic Euler equation — (RP2) — Simulation at $t_{\text{final}} = 0.25$ — Polynomial degrees — Left-panels: 2nd order NN results — Right-panels: 4th order NN results — Top: 3×3 NN — Middle: 5×5 NN — Bottom: 10×10 NN.

	MOOD	NN 5×5	NN 10×10	NN 20×20
CPU time (s)	4.87	3.95	4.57	6.22
Ratio	1.0	0.81	0.93	1.28

Table 6 M1 model — 1st Riemann problem — CPU times and ratio — Classical 1D MOOD scheme against NN versions with 3rd order accurate schemes.

schemes. Using the 10×10 NN instead of the MOOD loop is roughly equivalent in terms of CPU for this test. However these figures are highly dependent on the test case, more precisely, on the number of recomputed bad cells. On a smooth problem the MOOD scheme would certainly be less expensive, while in presence of many bad cells the NN may become extremely performing. The genuine study of efficiency is postponed to a future work in multi-dimensions where we expect true gains.

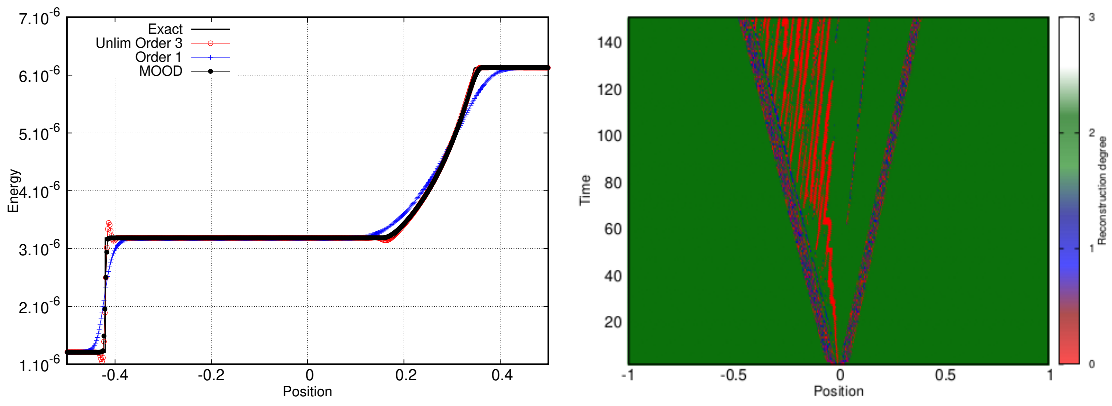

Fig. 22 M1 model — Left panel: Numerical solutions obtained by the first-order (blue symbols), second-order unlimited (red) and MOOD schemes (black) vs the exact solution at final time $t_{\text{final}} = 2 \times 10^{-9}$ — Right panel: space-time distribution of the degree of the polynomial reconstructions for 3rd order MOOD scheme for all cells and all RK steps.

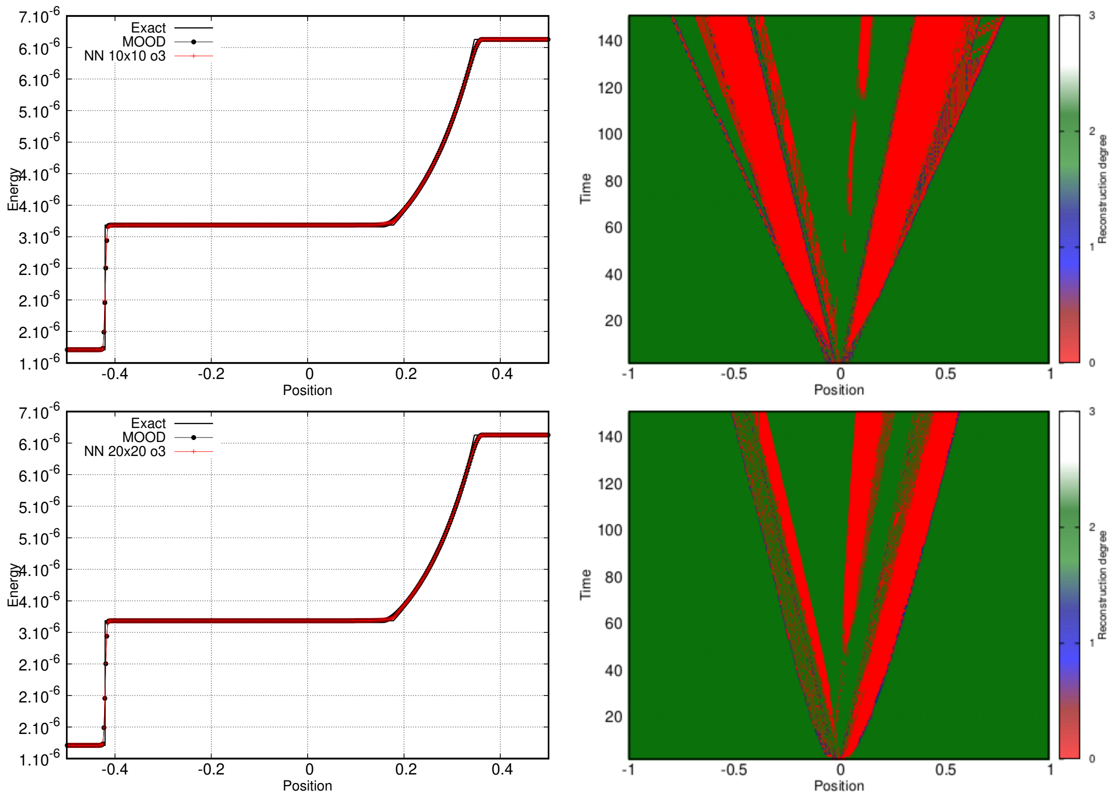

Fig. 23 M1 model — Simulation at $t_{\text{final}} = 2 \times 10^{-9}$ for 10×10 (top) and 20×20 NN — Left panels: numerical solution for the MOOD+NN and *a posteriori* MOOD schemes — Right panels: space-time distribution of the degree of the polynomial reconstructions for all cells and all RK steps.

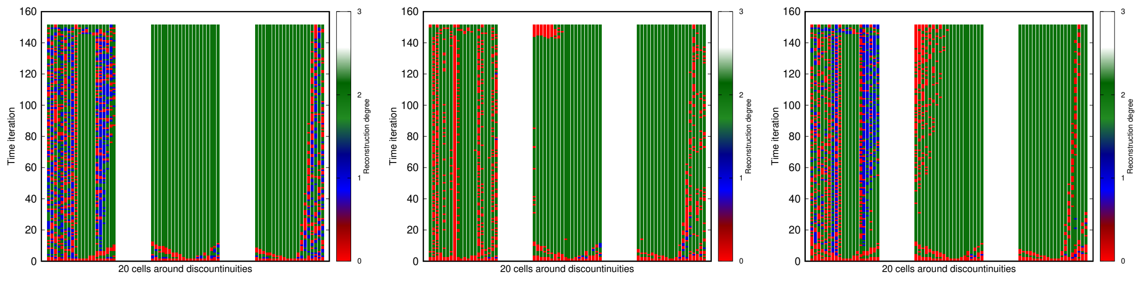

Fig. 24 M1 model — Simulation at $t_{\text{final}} = 2 \times 10^{-9}$ for 20×20 NN — Polynomial degrees for 20 cells around the shock (left band), the tail (middle) and the head (right) of the rarefaction waves. Left panel: MOOD results — Middle panel: 10×10 NN results. Right panel: 20×20 NN results.

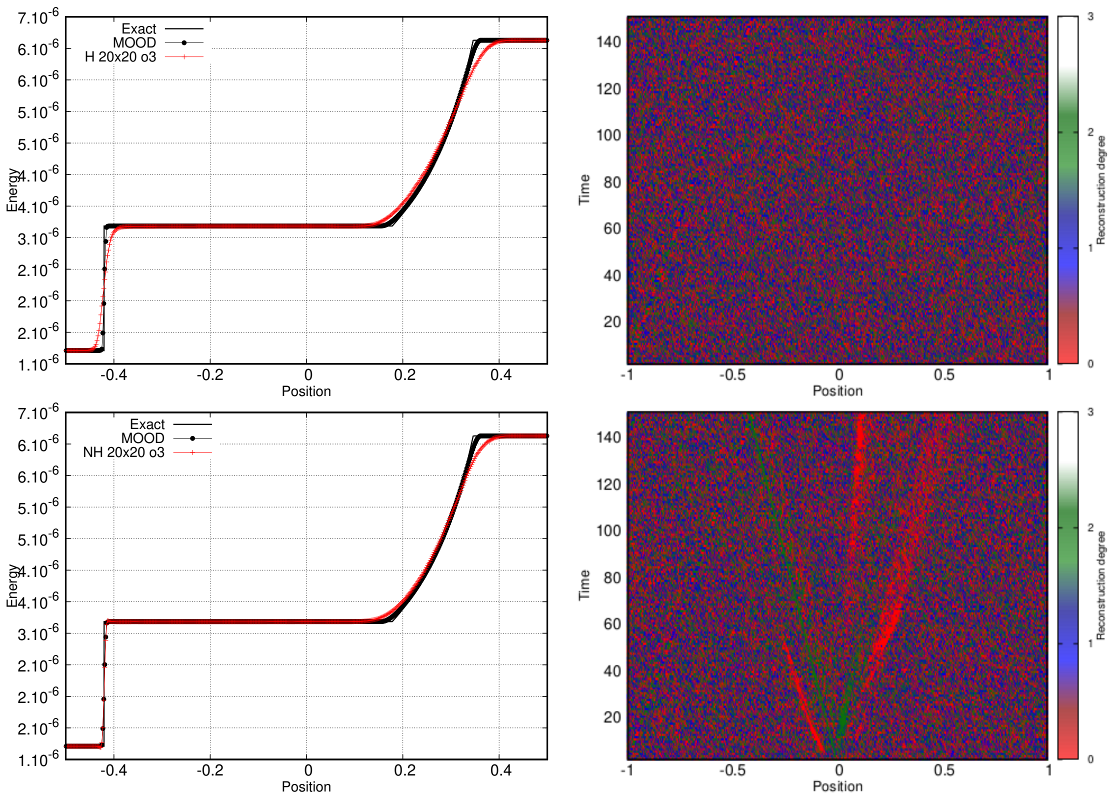

Fig. 25 M1 model — Simulation at $t_{\text{final}} = 2 \times 10^{-9}$ — Left panels: numerical solutions schemes — Right panels: space-time distribution of the degree of the polynomial reconstructions for all cells and all RK steps — Top panels: third-order FV scheme for which the polynomial degrees are randomly chosen — Bottom panels: third-order FV scheme with 20×20 NN for which only uncertain predictions are replaced by a random choice.

8 Conclusion and perspectives

In this paper we have investigated the use of a shallow Neural Network in replacement of the *a posteriori* Numerical Admissibility Detection MOOD loop within a high accurate FV scheme. Indeed a MOOD scheme of nominal order $d_{\max}$ picks the most appropriate reconstruction degree $d_i \leq d_{\max}$ in each cell which generates the best solution according to a set of detection criteria checked *a posteriori* on the candidate solution at time t^{n+1}. If the criteria are not fulfilled then the cell is recomputed decrementing the polynomial degree up to degree 0 in the worse case scenario. This is the philosophy behind an *a posteriori* MOOD limiting strategy. Our goal was to train a shallow NN by such a MOOD scheme expecting that it could *a priori* predict the correct cell polynomial degree. We have made the 1D framework as simple as possible avoiding most subtle choices like which variables to use in the training, varying the NN architectures, having large training data set, multiplying the input data (more neighbors, more variables), etc.
The goal was to observe how a NN behaves facing a relative simple task. To this end, multi-layer perceptron networks containing two hidden layers are trained in a supervised way, employing the MOOD scheme as the reference. We have observed that the NNs are able to recognize the vast majority of the training patterns and predict the correct polynomial degrees. We have presented how to build an appropriate training data-set for the scalar advection and Burgers' equation in 1D, train and use several of them for the 2nd up to 4th order accurate schemes. (Notice that there is no formal limitation in the approach to deal with higher order schemes, only practical limitations may apply.) Moreover we have extended the procedure to deal with the isentropic Euler and M1 systems of conservation laws. For all four models we have been able to reproduce MOOD results using network architectures which are small enough to consider their use in practical simulations in the future. The number of perceptrons never exceeds a few dozen. The input vector is restricted to 5 to 20 components and the training data-set is made of about 5000 to 50000 units. We have analyzed the space-time distribution of the polynomial degrees. Some evidences show that the NN is more permissive than a classical MOOD scheme. However it does not seem to generate large spurious parasitical phenomena.
The NN high accurate FV scheme in the case of system cannot be freed from the *a posteriori* loop checking for physical admissibility. In fact the intrinsic statistical nature of the NN training does not assure that the numerical solution will always remain physically valid. As a consequence we have coupled the *a priori* NN prediction of reconstruction polynomial degree with an *a posteriori* MOOD loop on positivity (and `NaN`) detection. Usually a lack of positivity leads to the failure of high accurate non-robust FV schemes, which is not the case for our schemes enjoying a fail-safe property rendering them extremely robust. Scalar equation and systems of equation seem to be similarly solved with the same quality by MOOD FV schemes or the FV schemes supplemented by a NN. Obviously a large enough number of weights should be considered to ensure that the NN can learn the training data set. On the downside remain some classical drawbacks of NNs:

- different training usually does not produce the same NNs, some may have a poor prediction capability;
- the constitution of the training data-set is of paramount importance to ensure good prediction capability. Our shallow NNs can possibly recognize unlearned patterns only if they remain close to already seen situations. As such one shallow NN which has been trained for a given non-linear model of PDEs can not in general be employed blindly with another model;
- if the NN prediction is wrong and some oscillation occurs, then the NN cannot recover from this situation because the training data-sets have not been specifically designed to teach such failures.

While those drawbacks could possibly be mitigated by using larger training data-sets and NN architectures, they may nonetheless remain. However it does not lower the interesting feature of having a NN replacing the *a posteriori* MOOD loop in a high order FV scheme.
In the future, relying on this first proof of concept in 1D, we will extend the approach to 2D and 3D MOOD FV schemes for which the number of input data will grow a lot. Consequently the number of

perceptrons and layers may also increase demanding the use of parallel machinery for the training and the use within the scheme. However we expect to reach a better efficiency compared to the *a posteriori* MOOD loop (with detection and decrementing). Moreover extending the approach to deal with more complex or larger systems of PDEs possibly with stiff source terms seems also our achievable.

Acknowledgments

RL would like to thank M. Han Veiga (University of Zürich) and S. Clain (Universidade do Minho) for sharing fruitful discussions on neural networks. The authors thank the anonymous reviewers who have led to an improved version of this paper.
The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

1. Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28 – 41, 2018.
2. F. Blachère and R. Turpault. An admissibility and asymptotic preserving scheme for systems of conservation laws with source term on 2d unstructured meshes with high-order mood reconstruction. *Computer Methods in Applied Mechanics and Engineering*, 317:836 – 867, 2017.
3. Ghislain Blanchard and Raphaël Loubère. High order accurate conservative remapping scheme on polygonal meshes using a posteriori mood limiting. *Computers and Fluids*, 136:83 – 103, 2016.
4. Walter Boscheri, Michael Dumbser, Raphaël Loubère, and Pierre-Henri Maire. A second-order cell-centered lagrangian ader-mood finite volume scheme on multidimensional unstructured meshes for hydrodynamics. *Journal of Computational Physics*, 358:103 – 129, 2018.
5. Walter Boscheri, Raphaël Loubère, and Michael Dumbser. Direct arbitrary-lagrangian–eulerian ader-mood finite volume schemes for multidimensional hyperbolic conservation laws. *Journal of Computational Physics*, 292:56–87, 2015.
6. Jean-Philippe Braeunig, Raphaël Loubère, Renaud Motte, Mathieu Peybernes, and Raphaël Poncet. A posteriori limiting for 2d lagrange plus remap schemes solving the hydrodynamics system of equations. *Computers and Fluids*, 169:249 – 262, 2018. Recent progress in nonlinear numerical methods for time-dependent flow and transport problems.
7. J.M. Burgers. *The Nonlinear Diffusion Equation: Asymptotic Solutions and Statistical Problems*. Springer, 1974.
8. S. Clain, S. Diot, and R. Loubère. A high-order finite volume method for systems of conservation laws—Multi-dimensional Optimal Order Detection (MOOD). *Journal of Computational Physics*, 230(10):4028 – 4050, 2011.
9. S. Clain and J. Figueiredo. The mood method for the non-conservative shallow-water system. *Computers and Fluids*, 145:99 – 128, 2017.
10. Stéphane Clain, Raphaël Loubère, and Gaspar J. Machado. a posteriori stabilized sixth-order finite volume scheme for one-dimensional steady-state hyperbolic equations. *Advances in Computational Mathematics*, 44(2):571–607, Apr 2018.
11. Yves Coudière and Rodolphe Turpault. A domain decomposition strategy for a very high-order finite volumes scheme applied to cardiac electrophysiology. *Journal of Computational Science*, 37:101025, 2019.
12. S. Diot, S. Clain, and R. Loubère. Improved detection criteria for the Multi-dimensional Optimal Order Detection (MOOD) on unstructured meshes with very high-order polynomials. *Computers and Fluids*, 64(Supplement C):43 – 63, 2012.

13. S. Diot, R. Loubère, and S. Clain. The Multidimensional Optimal Order Detection method in the three-dimensional case: very high-order finite volume method for hyperbolic systems. *International Journal for Numerical Methods in Fluids*, 73(4):362–392, 2013.
14. B. Dubroca and J. L. Feugeas. Etude théorique et numérique d'une hiérarchie de modèles aux moments pour le transfert radiatif. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 329(10):915 – 920, 1999.
15. M. Dumbser and R. Loubère. A simple robust and accurate a posteriori sub-cell finite volume limiter for the discontinuous Galerkin method on unstructured meshes. *Journal of Computational Physics*, 319(Supplement C):163 – 199, 2016.
16. Michael Dumbser, Olindo Zanotti, Raphaël Loubère, 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.
17. Javier Fernández-Fidalgo, Xesús Nogueira, Luis Ramírez, and Ignasi Colominas. An a posteriori, efficient, high-spectral resolution hybrid finite-difference method for compressible flows. *Computer Methods in Applied Mechanics and Engineering*, 335:91 – 127, 2018.
18. Pritam Giri and Jianxian Qiu. A high-order runge-kutta discontinuous galerkin method with a subcell limiter on adaptive unstructured grids for two-dimensional compressible inviscid flows. *International Journal for Numerical Methods in Fluids*, 91(8):367–394, 2019.
19. S. Gottlieb and C.W. Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of Computation*, 67:73–85, 1998.
20. Sigal Gottlieb and Chi-Wang Shu. Total Variation Diminishing Runge-Kutta Schemes. *Math. Comput.*, 67(221):73–85, January 1998.
21. V. Gyrya, M. Shashkov, A. Skurikhin, and S. Tokareva. Machine learning approaches for the solution of the riemann problem in fluid dynamics: a case study. *preprint*, 2020.
22. M.T. Hagan, H.B. Demuth, and M.H. Beale. *Neural Network Design*. PWS Publishing, 1996.
23. Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, third edition, 2009.
24. Donald Olding Hebb and DO Hebb. *The organization of behavior*, volume 65. Wiley New York, 1949.
25. C.-W. Shu J. Qiu. A comparison of troubled-cell indicators for Runge–Kutta discontinuous Galerkin methods using weighted essentially nonoscillatory limiters. *Journal of Scientific Computating*, 27:995–1013, 2005.
26. Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted eno schemes. *Journal of Computational Physics*, 126(1):202 – 228, 1996.
27. Zhen-Hua Jiang, Chao Yan, and Jian Yu. Efficient methods with higher order interpolation and mood strategy for compressible turbulence simulations. *Journal of Computational Physics*, 371:528 – 550, 2018.
28. David Kriesel. *A Brief Introduction to Neural Networks*. 2007.
29. R. Loubère, M. Dumbser, and S. Diot. A New Family of High Order Unstructured MOOD and ADER Finite Volume Schemes for Multidimensional Systems of Hyperbolic Conservation Laws. *Communications in Computational Physics*, 16(3):718–763, 2014.
30. D. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2):431—-441, 1963.
31. W S McCulloch and W Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133, 1943.
32. M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.
33. Xesús Nogueira, Luis Ramírez, Stéphane Clain, Raphaël Loubère, Luis Cueto-Felgueroso, and Ignasi Colominas. High-accurate sph method with multidimensional optimal order detection limiting. *Computer Methods in Applied Mechanics and Engineering*, 310:134 – 155, 2016.
34. Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125 – 141, 2018.

35. Deep Ray and Jan S. Hesthaven. An artificial neural network as a troubled-cell indicator. *Journal of Computational Physics*, 367:166 – 191, 2018.
36. Deep Ray and Jan S. Hesthaven. Detecting troubled-cells on two-dimensional unstructured grids using a neural network. *Journal of Computational Physics*, 397:108845, 2019.
37. F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.
38. F. Rosenblatt. *Principles of neurodynamics.* New York, Spartan, 1962.
39. Céline Sarazin-Desbois. *Méthodes numériques pour des systèmes hyperboliques avec terme source provenant de physiques complexes autour du rayonnement.* PhD thesis, Université de Nantes, 2013.
40. Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.
41. Matteo Semplice and Raphaël Loubère. Adaptive-mesh-refinement for hyperbolic systems of conservation laws based on a posteriori stabilized high order polynomial reconstructions. *Journal of Computational Physics*, 354:86 – 110, 2018.
42. A. Suresh and H.T. Huynh. Accurate monotonicity-preserving schemes with runge-kutta time stepping. *Journal of Computational Physics*, 136:83–99, 1997.
43. Siengdy Tann, Xi Deng, Yuya Shimizu, Raphaël Loubère, and Feng Xiao. Solution property preserving reconstruction for finite volume scheme: a bvd+mood framework. *International Journal for Numerical Methods in Fluids*, n/a(n/a), 2019.
44. E. F. Toro. *Riemann Sovlers and Numerical Methods for Fluid Dynamics.* Springer, 2009.
45. Maria Han Veiga and Rémi Abgrall. Towards a general stabilisation method for conservation laws using a multilayer perceptron neural network: 1d scalar and system of equations. In *European Conference on Computational Mechanics and VII European Conference on Computational Fluid Dynamics*, number 1, pages 2525–2550. ECCM, June 2018.
46. François Vilar. A posteriori correction of high-order discontinuous galerkin scheme through subcell finite volume formulation and flux reconstruction. *Journal of Computational Physics*, 387:245 – 279, 2019.
47. P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Harvard University, 1975.