FC-based shock-dynamics solver with neural-network localized artificial-viscosity assignment

Oscar P. Bruno [*] Jan S. Hesthaven [†] Daniel V. Leibovici [*]

Abstract

This paper presents a spectral scheme for the numerical solution of nonlinear conservation laws in *non-periodic domains under arbitrary boundary conditions*. The approach relies on the use of the Fourier Continuation (FC) method for spectral representation of non-periodic functions in conjunction with smooth localized artificial viscosity assignments produced by means of a Shock-Detecting Neural Network (SDNN). Like previous shock capturing schemes and artificial viscosity techniques, the combined FC-SDNN strategy effectively controls spurious oscillations in the proximity of discontinuities. Thanks to its use of a *localized but smooth artificial viscosity term*, whose support is restricted to a vicinity of flow-discontinuity points, the algorithm enjoys spectral accuracy and low dissipation away from flow discontinuities, and, in such regions, it produces smooth numerical solutions—as evidenced by an essential absence of spurious oscillations in level set lines. The FC-SDNN viscosity assignment, which does not require use of problem-dependent algorithmic parameters, induces a significantly lower overall dissipation than other methods, including the Fourier-spectral versions of the previous entropy viscosity method. The character of the proposed algorithm is illustrated with a variety of numerical results for the linear advection, Burgers and Euler equations in one and two-dimensional non-periodic spatial domains.

Keywords: Machine learning, Neural networks, Shocks, Artificial viscosity, Conservation laws, Fourier continuation, Non-periodic domain, Spectral method

[*]Computing and Mathematical Sciences, Caltech, Pasadena, CA 91125, USA

[†]Computational Mathematics and Simulation Science (MCSS), Ecole Polytechnique Federale de Lausanne, CH-1015 Lausanne, Switzerland

1 Introduction

This paper presents a new "FC-SDNN" spectral scheme for the numerical solution of nonlinear conservation laws under arbitrary boundary conditions. The proposed approaches relies on use of the FC-Gram Fourier Continuation method [1, 2, 6, 29] for spectral representation of non-periodic functions in conjunction with localized smooth artificial viscosity assignments prescribed by means of the neural network-based shock-detection method proposed in [38]. The neural network approach [38] itself utilizes Fourier series to discretize the gas dynamics and related equations, and it eliminates Gibbs ringing at shock positions (which are determined by means of an artificial neural network) by assigning artificial viscosity over a small number of discrete points in a close vicinity of shocks. The use of the classical Fourier spectral method in that contribution restricts the method's applicability to periodic problems (so that, in particular, the outer computational boundaries cannot be physical boundaries), and its highly localized viscosity assignments give rise to a degree of non-smoothness, resulting in certain types of unphysical oscillations manifested as serrated level-set lines in the flow fields. The FC-SDNN method presented in this paper addresses these challenges. In particular, in view of its use of certain newly-designed *smooth viscosity windows* introduced in Section 3.3, the method avoids the introduction of roughness in the viscosity assignments and thus it yields smooth flows away from shocks and other flow discontinuities. In addition, the underlying FC-Gram spectral representations enable applicability to general non-periodic problems, and, in view of the weak local viscosity assignments used, it gives rise to sharp resolution of shocks. As a result, and as demonstrated in this paper via application to a range of well known 1D and 2D shock-wave test configurations, the overall FC-SDNN approach yields accurate and essentially oscillation-free solutions for general non-periodic problems.

The computational solution of systems of conservation laws has been tackled by means of a variety of numerical methods, including low-order finite volume [26,27] and finite difference methods equipped with slope limiters [26, 27], as well as higher order shock-capturing methods such as the ENO [15] and WENO schemes [17,28]. An efficient FC/WENO hybrid solver was proposed in [39]. The use of artificial viscosity as a computational device for conservation laws, on the other hand, was first proposed in [37, 45] and the subsequent contributions [10, 22, 23]. The viscous terms proposed in these papers, which incorporate derivatives of the square of the velocity gradient, may induce oscillations in the vicinity of shocks [23] (since the velocity itself is not smooth in such regions), and, as they do not completely vanish away from the discontinuities, they may lead to significant approximation errors in regions were the fluid velocity varies rapidly. Reference [32] proposes the use of a shock-detecting sensor in order to localize the support of the artificial viscosity, which is used in the context of a Discontinuous Galerkin scheme.

The entropy viscosity method [14] (EV) incorporates a nonlinear viscous "entropy-residual" term which essentially vanishes away from discontinuities—in view of the fact that the flow is isentropic over smooth flow regions—and which is thus used to limit non-zero viscosity assignments to regions near flow discontinuities, including both shock waves and contact discontinuities. This method, however, relies on several problem-dependent algorithmic parameters that require tuning for every application. Additionally, this approach gives rise to a significant amount of dissipation even away from shocks, in particular in the vicinity of contact discontinuities and regions containing fast spatial variation in the flow-field variables. Considerable improvements concerning this issue were obtained in [21] (which additionally introduced a Hermite-based method to discretize the hyperbolic systems) by modifying the EV viscosity term. Like the viscous term introduced by [45], the EV viscosity assignments [14,21] are themselves discontinuous in the vicinity of shocks, and, thus, their use may

introduce spurious oscillations. The C-method [33, 34, 36], which augments the hyperbolic system with an additional equation used to determine a spatio-temporally smooth viscous term, relies, like the EV method, on use of several problem dependent parameters and algorithmic variations.

Recently, significant progress was made by incorporating machine learning-based techniques (ML) to enhance the performance of classical shock capturing schemes [9, 35, 38, 43]. The approach [9, 35, 38] utilizes ML-based methods to detect discontinuities which are then smeared by means of shock-localized artificial viscosity assignments in the context of various discretization methods, including Discontinuous Galerkin schemes [9, 35] and Fourier spectral schemes [38]. The ML-based approach utilized in [43], in turn, modifies the finite volume coefficients utilized in the WENO5-JS scheme by learning small perturbations of these coefficients leading to improved accurate representations of functions at cell boundaries.

Like the strategy underlying the contribution [38], the FC-SDNN method proposed in this paper relies on the occurrence of Gibbs oscillations for ML-based shock detection. Unlike the previous approach, however, the present method assigns a *smooth* (albeit also shock-localized) viscous term. In view of its smooth viscosity assignments this procedure effectively eliminates Gibbs oscillations while avoiding introduction of the flow-field roughness that is often evidenced by the serrated level sets produced by other methods. In view of its use of FC-based Fourier expansions, further, the proposed algorithm enjoys spectral accuracy away from shocks (thus, delivering, in particular, essentially vanishing dispersion in such regions; see Section 2.3 and Figure 6) while enabling solution under general (non-periodic) boundary conditions. Unlike other techniques, finally, the approach does not require use of problem-dependent algorithmic parameters.

The capabilities of the proposed algorithm are illustrated by means of a variety of numerical results, in one and two-dimensional contexts, for the Linear Advection, Burgers and Euler equations. In order to provide a useful reference point, this paper also presents an FC-based version, termed FC-EV, of the EV algorithm [14]. (The modified version [21] of the entropy viscosity approach, which was also tested as a possible reference solver, was not found to be completely reliable in our FC-based context, since it occasionally led to spurious oscillations in shock regions as grids were refined, and the corresponding results were therefore not included in this paper.) We find that the FC-SDNN algorithm generally provides significantly more accurate numerical approximations than the FC-EV, as the localized artificial viscosity in the former approach induces a much lower dissipation level than the latter.

This paper is organized as follows. After necessary preliminaries are presented in Section 2 (concerning the hyperbolic problems under consideration, as well as the Fourier Continuation method, and including basic background on the artificial-viscosity strategies we consider), Section 3 describes the proposed FC-SDNN approach. A link to a git repository containing basic Matlab implementations and test codes for the 1D FC algorithm is provided at the end of Section 2.3. Section 4 then demonstrates the algorithm's performance for a variety of non-periodic linear and nonlinear hyperbolic problems. In particular, cases are considered for the linear advection equation, Burgers equation and Euler's equations in one-dimensional and two-dimensional rectangular and non-rectangular spatial domains, including cases in which shock waves meet smooth and non-smooth physical boundaries.

2 Preliminaries

2.1 Conservation laws

This paper proposes novel spectral methodologies, applicable in general non-periodic contexts and with general boundary conditions, for the numerical solution of conservation-law equations of the form

$$\frac{\partial}{\partial t}\boldsymbol{e}(\boldsymbol{x},t) + \nabla \cdot \big(f(\boldsymbol{e}(\boldsymbol{x},t))\big) = 0 \tag{1}$$

on a bounded domain $\Omega \subset \mathbb{R}^q$, where $\boldsymbol{e} : \Omega \times [0,T] \to \mathbb{R}^r$ and $f : \mathbb{R}^r \to \mathbb{R}^r \times \mathbb{R}^q$ denote the unknown solution vector and a (smooth) convective flux, respectively.

The proposed spectral approaches are demonstrated for several equations of the form (1), including the Linear Advection equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{2}$$

with a constant propagation velocity a, where we have $\boldsymbol{e} = u$, and $f(u) = au$; the one- and two-dimensional scalar Burgers equations

$$\frac{\partial u}{\partial t} + \frac{1}{2}\Big(\frac{\partial u^2}{\partial x}\Big) = 0 \tag{3}$$

and

$$\frac{\partial u}{\partial t} + \frac{1}{2}\Big(\frac{\partial u^2}{\partial x}\Big) + \frac{1}{2}\Big(\frac{\partial u^2}{\partial y}\Big) = 0, \tag{4}$$

for each of which we have $\boldsymbol{e} = u$ and $f(u) = \frac{u^2}{2}$; as well as the one- and two-dimensional Euler equations

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho u\\ E\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}\rho u\\ \rho u^2 + p\\ u(E+p)\end{pmatrix} = 0 \tag{5}$$

and

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ E\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}\rho u\\ \rho u^2 + p\\ \rho uv\\ u(E+p)\end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix}\rho v\\ \rho uv\\ \rho v^2 + p\\ v(E+p)\end{pmatrix} = 0 \tag{6}$$

with

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho|\boldsymbol{u}|^2, \tag{7}$$

for each of which we have

$$\boldsymbol{e} = (\rho, \rho\boldsymbol{u}, E)^T, \quad f(\boldsymbol{e}) = (\rho\boldsymbol{u}, \rho\boldsymbol{u}\otimes\boldsymbol{u} + p\mathbb{I}, \boldsymbol{u}(E+p))^T. \tag{8}$$

Here $\mathbb{I}$ denotes the identity tensor, $\mathbf{a}\otimes\mathbf{b} = (a_i b_j)$ denotes the tensor product of the vectors $\mathbf{a} = (a_i)$ and $\mathbf{b} = (b_j)$, and ρ, $\boldsymbol{u}$, E and p denote the density, velocity vector, total energy and pressure, respectively. The speed of sound [26]

$$a = \sqrt{\frac{\gamma p}{\rho}} \tag{9}$$

for the Euler equations plays important roles in the various artificial viscosity assignments considered in this paper for Euler problems in both 1D and 2D.

Remark 1. As an example concerning notational conventions, note that in the case of the 2D Euler equations, for which f is given by (8), $\nabla \cdot \big(f(\boldsymbol{e})\big)$ can be viewed as a three coordinate vector whose first, second and third coordinates are a scalar, a vector and a scalar, respectively. Using a super-index notation for the velocity vector $\boldsymbol{u}$ (i.e., $u^1 = u$ and $(u^1, u^2) = (u, v)$ for the 1D and 2D equations, respectively), together with the Einstein summation convention, these three components are respectively given by $\nabla \cdot (\rho \mathbf{u}) = \partial_i(\rho u^i)$, $\big(\nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u} + p\mathbb{I})\big)_j = \partial_i(\rho u^j u^i + p)$ and $\nabla \cdot ((E+p)\mathbf{u}) = \partial_i((E+p)u^i)$.

2.2 Artificial viscosity

As is well known, the shocks and other flow discontinuities that arise in the context of nonlinear conservation laws of the form (1) give rise to a number of challenges from the point of view of computational simulation. In particular, in the framework of classical finite difference methods as well as Fourier spectral methods, such discontinuities are associated with the appearance of spurious "Gibbs oscillations". Artificial viscosity methods aim at tackling this difficulty by considering, instead of the inviscid equations (1), certain closely related equations which include viscous terms containing second order spatial derivatives. Provided the viscous terms are adequately chosen and sufficiently small, the resulting solutions, which are smooth functions on account of viscosity, approximate well the desired (discontinuous) inviscid solutions. In general terms, the viscous equations are obtained by adding a viscous term of the form $\nabla \cdot \big(f_{visc}[\boldsymbol{e}]\big)$ to the right hand side of (1), where the "viscous flux" operator

$$f_{visc}[\boldsymbol{e}] = \mu[\boldsymbol{e}]\mathbf{D}[\boldsymbol{e}], \tag{10}$$

(which, for a given vector-valued function $\mathbf{e}(x,t)$, produces a vector-valued function $f_{visc}[\boldsymbol{e}](x,t)$ defined in the complete computational domain), is given in terms of a certain "viscosity" operator $\mu[\boldsymbol{e}](x,t)$ (which may or may not include derivatives of the flow variables $\mathbf{e}$), and a certain matrix-valued first order differential operator $\mathbf{D}$. Once such a viscous term is included, the viscous equation

$$\frac{\partial \boldsymbol{e}(\boldsymbol{x},t)}{\partial t} + \nabla \cdot \big(f(\boldsymbol{e}(\boldsymbol{x},t))\big) = \nabla \cdot \big(f_{visc}[\boldsymbol{e}](\boldsymbol{x},t)\big) \tag{11}$$

results.

Per the discussion in Section 1, this paper exploits and extends, in the context of the Fourier-Continuation discretizations, two different approaches to viscosity-regularization—each one resulting from a corresponding selection of the operators μ and $\mathbf{D}$. One of these approaches, the EV method, produces a viscosity assignment $\mu[\boldsymbol{e}](x,t)$ on the basis of certain differential and algebraic operations together with a number of tunable problem-dependent parameters that are specifically designed for each particular equation considered, as described in Section 2.2.2. The resulting viscosity values $\mu[\boldsymbol{e}](x,t)$ are highest in a vicinity of discontinuity regions and decrease rapidly away from such regions. The neural-network approach introduced in Section 2.2.1, in turn, uses machine learning methods to pinpoint the location of discontinuities, and then produces a viscosity function whose support is restricted to a vicinity of such discontinuity locations. As a significant advantage, the neural-network method, which does not require use of tunable parameters, is essentially problem independent, and it can use a single pre-trained neural network for all the equations considered. Details concerning these two viscosity-assignment methods considered are provided in what follows.

2.2.1 Artificial viscosity via shock-detecting neural network (SDNN)

The SDNN approach proposed in this paper is based on the neural-network strategy introduced in [38] for detection of discontinuities on the basis of Gibbs oscillations in Fourier series, together with a novel selection of the operator μ in (10) that yields spatially localized but smooth viscosity assignments: per the description in Section 3.3, the FC-SDNN viscosity $\mu[\boldsymbol{e}](\boldsymbol{x},t)$ is a smooth function that vanishes except in narrow regions around flow discontinuities. The differential operator $\mathbf{D}$, in turn, is simply given by

$$\mathbf{D}[\boldsymbol{e}](\boldsymbol{x},t) = \nabla(\boldsymbol{e}(\boldsymbol{x},t)), \tag{12}$$

where the gradient is computed component-wise. As indicated in Section 1, the smoothness of the proposed viscosity assignments is inherited by the resulting flows away from flow discontinuities, thus helping eliminate the serrated level-set lines that are ubiquitous in the flow patterns produced by other methods.

2.2.2 Entropy viscosity methodology (EV)

The operators μ and $\mathbf{D}$ employed by the EV approach [14] are defined in terms of a number of problem dependent functions, vectors and operators. Indeed, starting with an equation dependent *entropy pair* (η,ν) where η is a scalar function and ν is a vector of the same dimensionality as the velocity vector, the EV approach utilizes an associated scalar *entropy residual* operator

$$R_{EV}[\boldsymbol{e}](\boldsymbol{x},t) = \frac{\partial\eta(\boldsymbol{e}(\boldsymbol{x},t))}{\partial t} + \nabla\cdot\nu(\boldsymbol{e}(\boldsymbol{x},t)) \tag{13}$$

together with a function $C = C(\boldsymbol{e})$ related to the local wave speed, and a normalization operator $N = N[e](x,t)$ obtained from the function η.

In practice, reference [14] proposes $\eta(\boldsymbol{e}) = \frac{u^2}{2}$, $\nu(\boldsymbol{e}) = a\frac{u^2}{2}$ and $C(\boldsymbol{e}) = a$ for the Linear Advection equation (2), $\eta(\boldsymbol{e}) = \frac{u^2}{2}$, $\nu(\boldsymbol{e}) = \frac{u^3}{3}$ and $C(\boldsymbol{e}) = u$ for the 1D and 2D Burgers equations (3) and (4), and $\eta(\boldsymbol{e}) = \frac{\rho}{\gamma-1}\log(p/\rho^\gamma)$, $\nu(\boldsymbol{e}) = \boldsymbol{u}\frac{\rho}{\gamma-1}\log(p/\rho^\gamma)$ and $C(\boldsymbol{e}) = |\boldsymbol{u}| + a$ (where a denotes the speed of sound (9)) for the 1D and 2D Euler equations (5) and (6). As for the normalization operator, reference [14] proposes $N = 1$ for the Euler equations and $N[e](x,t) = |\eta(e)(x,t) - \overline{\eta}(e)(t)|$ for the Linear advection and Burgers equations, where $\overline{\eta}(e)(t)$ denotes the spatial average of $\eta(e)$ at time t.

For a numerical discretization with maximum spatial mesh size h, the EV viscosity function is defined by

$$\mu[\boldsymbol{e}](\boldsymbol{x},t) = \min(\mu_{max}[\boldsymbol{e}](t), \mu_E[\boldsymbol{e}](\boldsymbol{x},t)) \tag{14}$$

where the maximum viscosity μ_{max} is given by

$$\mu_{max}[\boldsymbol{e}](t) = c_{max}h\max_{x\in\Omega}|C(\boldsymbol{e}(\boldsymbol{x},t))| \tag{15}$$

and where

$$\mu_E[\boldsymbol{e}](\boldsymbol{x},t) = c_E h^2\frac{|R_{EV}[\boldsymbol{e}](\boldsymbol{x},t)|}{N[\boldsymbol{e}](\boldsymbol{x},t)}. \tag{16}$$

In particular, the EV viscosity function depends on two parameters, c_{max} and c_E, both of size $\mathcal{O}(1)$, that, following [14], are to be tuned to each particular problem.

Finally, the EV differential operator $\mathbf{D}$ for the Linear Advection and Burgers equations is defined by

$$\mathbf{D}[\boldsymbol{e}](\boldsymbol{x},t) = \nabla(\boldsymbol{e}(\boldsymbol{x},t)), \tag{17}$$

while for the Euler equations it is given by

$$\mathbf{D}[\boldsymbol{e}](\boldsymbol{x},t) = \begin{pmatrix} 0 \\ \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) \\ \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)\boldsymbol{u} + \kappa \nabla(p/\rho) \end{pmatrix} \tag{18}$$

where, using once again the Einstein convention, $\left\{(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)\boldsymbol{u}\right\}_i = (\partial_i u^j + \partial_j u^i)u^j$, and where $\kappa = \frac{\mathcal{P}}{\gamma - 1}\mu$, with the Prandtl number $\mathcal{P}$ taken to equal 1.

2.3 Spatial approximation via Fourier Continuation

The straightforward Fourier-based discretization of nonlinear conservation laws generally suffers from crippling Gibbs oscillations resulting from two different sources: the physical flow discontinuities, on one hand, and the overall generic non-periodicity of the flow variables, on the other. Unlike the Gibbs ringing in flow-discontinuity regions, the ringing induced by lack of periodicity is not susceptible to treatment via artificial viscosity assignments of the type discussed in 2.2. In order to tackle this difficulty we resort to use of the Fourier Continuation (FC) method for equispaced-grid spectral approximation of non-periodic functions.

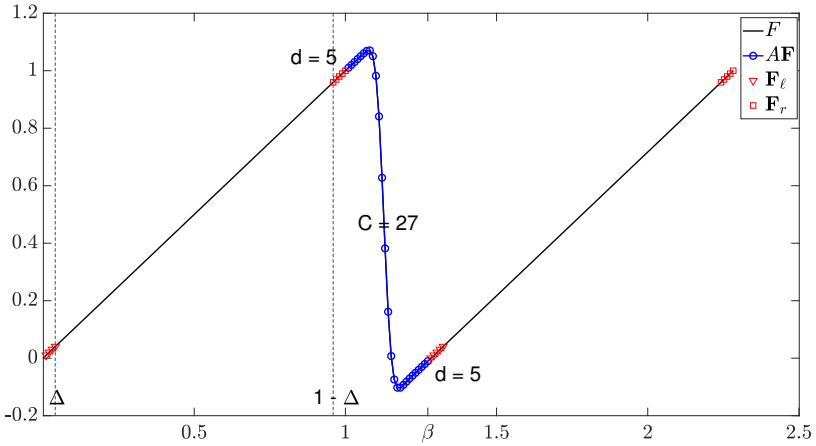


Figure 1: Fourier Continuation of the non-periodic function $F(x) = x$ on the interval $[0, 1]$. With reference to the text, the red triangles (resp. squares), represent the $d = 5$ left (resp. right) matching points, while the blue circles represent the $C = 27$ continuation points.

The basic FC algorithm [1], called FC-Gram in view of its reliance on Gram polynomials for near-boundary approximation, constructs an accurate Fourier approximation of a given generally non-periodic function F defined on a given one-dimensional interval—which, for definiteness, is assumed in this section to equal the unit interval $[0, 1]$. The Fourier continuation algorithm relies on use of the function values $F_j = F(x_j)$ of the function $F : [0, 1] \to \mathbb{R}$ at N points $x_j = jh \in [0, 1]$ $(h = 1/(N - 1))$ to produce a function

$$F^c(x) = \sum_{k=-M}^{M} \hat{F}_k^c \exp(2\pi i k x/\beta) \tag{19}$$

which is defined (and periodic) in an interval $[0, \beta]$ that strictly contains $[0, 1]$, where $\hat{F}_k^c$ denote the FC coefficients of F and where, as detailed below, M is an integer that, for N large, is close to (but different from) the integer $\lfloor N/2 \rfloor$.

In order to produce the FC function F^c, the FC-Gram algorithm first uses two subsets of the function values in the vector $\mathbf{F} = (F_0, \ldots, F_{N-1})^T$ (namely the function values at the"matching points" $\{x_0, .., x_{d-1}\}$ and $\{x_{N-d}, ..., x_{N-1}\}$ located in small matching subintervals $[0, \Delta]$ and $[1-\Delta, 1]$ of length $\Delta = (d-1)h$ near the left and right ends of the interval $[0, 1]$, where d is a small integer independent of N), to produce, at first, a discrete (but "smooth") periodic extension vector $\mathbf{F}^c$ of the vector $\mathbf{F}$. Indeed, using the matching point data, the FC-Gram algorithm produces and appends a number C of continuation function values in the interval $[1, \beta]$ to the data vector $\mathbf{F}$, so that the extension $\mathbf{F}^c$ transitions smoothly from F_{N-1} back to F_0, as depicted in Figure 1. (The FC method can also be applied on the basis of certain combinations of function values and derivatives by constructing the continuation vector $\mathbf{F}^c$, as described below in this section, for a given vector $\mathbf{F} = (F_0, \ldots, F_{N-2}, F'_{N-1})^T$, where $F_j \approx F(x_j)$ for $1 \leq j \leq N-2$ and where $F'_{N-1} \approx F'(x_{N-1})$. Such a procedure enables imposition of Neumann boundary conditions in the context of the FC method.) The resulting vector $\mathbf{F}^c$ can be viewed as a discrete set of values of a smooth and periodic function which can be used to produce the Fourier continuation function F^c via an application of the FFT algorithm. The function F^c provides a spectral approximation of F throughout the interval $[0, 1]$ which does not suffer from either Gibbs-ringing or the associated interval-wide accuracy degradation. Throughout this paper we assume, for simplicity, that $N + C$ is an odd integer, and, thus, the resulting series has bandwidth $M = \frac{N+C-1}{2}$; consideration of even values of $N + C$ would require a slight modification of the index range in (19).

To obtain the necessary discrete periodic extension $\mathbf{F}^c$, the FC-Gram algorithm first produces two polynomial interpolants, one per matching subinterval, using, as indicated above, a small number d of function values or a combination of function values and a derivative near each one of the endpoints of the interval $[0, 1]$. This approach gives rise to high-order interpolation of the function F over the matching intervals $[0, \Delta]$ and $[1-\Delta, 1]$. The method for evaluation of the discrete periodic extension is based on a representation of these two polynomials in a particular orthogonal polynomials basis (the Gram polynomials), for each element of which the algorithm utilizes a precomputed smooth function which blends the basis polynomial to the zero function over the distance $\beta - 1$ [1, 2]. Certain simple operations involving these "blending to zero" functions are then used, as indicated in these references and as illustrated in Figure 1, to obtain smooth transitions-to-zero from the left-most and right-most function values to the extension interval $[1, \beta]$. The values of this transition function at the points $N/(N-1), (N+1)/(N-1), \ldots, (N+C-1)/(N-1)$ provide the necessary C additional point values from which, as mentioned above, the discrete extension $\mathbf{F}^c$ is obtained. The continuation function F^c then easily results via an application of the FFT algorithm to the function values F^c in the interval $[0, \beta]$.

The discrete continuation procedure can be expressed in the matrix form

$$\mathbf{F}^c = \begin{pmatrix} \mathbf{F} \\ A_\ell Q^T \mathbf{F}_\ell + A_r Q^T \mathbf{F}_r \end{pmatrix} \tag{20}$$

where the d-dimensional vectors $\mathbf{F}_\ell$ and $\mathbf{F}_r$ contain the point values of F at the first and last d discretization points in the interval $[0, 1]$, respectively; where Q is a $d \times d$ matrix, whose columns contain the point values of the elements of the Gram polynomial bases on the left matching intervals; and where A_ℓ and A_r are $C \times d$, matrices containing the C values of the blended-to-zero Gram

polynomials in the left and right Gram bases, respectively. These small matrices can be computed once and stored on disc, and then read for use to produce FC expansions for functions $G : [a, b] \to \mathbb{R}$ defined on a given 1D interval $[a, b]$, via re-scaling to the interval $[0, 1]$.

A minor modification of the procedure presented above suffices to produce a Fourier continuation function on the basis of data points at the domain interior and a derivative at interval endpoints. For example, given the vector $\mathbf{F} = (F_0, \ldots, F_{N-2}, F'_{N-1})^T$, using an adequately modified version $\widetilde{Q}$ of the matrix Q, an FC series $F^c(x)$ can be produced which matches the function values $F_0, \ldots, F_{N-2}$ at $x = x_0, \ldots, x_{N-2}$, and whose derivative equals F'_{N-1} at x_{N-1}. The matrix $\widetilde{Q}$ is obtained by using the matrix Q to obtain a value F_{N-1} such that the derivative $F'(x_{N-1})$ equals the given value F'_{N-1}. Full details in this regard can be found in [2, Sec. 3.2].

Clearly, the approximation order of the Fourier Continuation method (whether derivative values or function values are prescribed at endpoints) is restricted by the corresponding order d of the Gram polynomial expansion, which, as detailed in various cases in Section 4, is selected as a small integer: $d = 2$ or $d = 5$. The relatively low order of accuracy afforded by the $d = 2$ selection, which must be used in some cases to ensure stability, is not a matter of consequence in the context of the problems considered in the present paper, where high orders of accuracy are not expected from any numerical method on account of shocks and other flow discontinuities. Importantly, even in this context the FC method preserves one of the most significant numerical properties of Fourier series, namely its extremely small numerical dispersion. In fact, with exception of the cyclic advection example presented in Section 4.1.2, for which errors can accumulate on account of the spatio-temporal periodicity, for all cases in Section 4 for which both the $d = 2$ and $d = 5$ simulations were performed (which include those presented in Sections 4.1.1 (1D linear advection), 4.2.2 (2D Burgers equation) and 4.3.1 (1D Euler equations), the lower and higher order results obtained were visually indistinguishable.

The low dispersion character resulting from use of the FC method is demonstrated in Figure 6, which displays solutions produced by means of two different methods, namely, the FC-based order-5 FC-SDNN algorithm (Section 3) and the order-6 centered finite-difference scheme (both of which use the SSPRK-4 time discretization scheme), for a linear advection problem. The FC-SDNN solution presented in the figure does not deteriorate even for long propagation times, thus illustrating the essentially dispersion-free character of the FC-based approach. The finite-difference solution included in the Figure, in turn, does exhibit clear degradation with time, owing to the dispersion and diffusion effects associated with the underlying finite difference discretization.

An online repository containing containing basic Matlab implementations and test codes for the 1D FC algorithm is available at `https://github.com/oscarbruno/FC.git`.

3 FC-based time marching under neural network-controlled artificial viscosity

3.1 Spatial grid functions and spatio-temporal FC-based differentiation

We consider in this work 1D problems on intervals $I = [\xi_\ell, \xi_r]$ as well as 2D problems on open domains Ω contained in rectangular regions $I \times J$, where $I = [\xi_\ell, \xi_r]$ and $J = [\xi_d, \xi_u]$ ($\xi_\ell < \xi_r$, $\xi_d < \xi_u$). Using a spatial meshsize h, the spatially discrete vectors of unknowns and certain related flow quantities will be represented by means of scalar and vector grid functions defined on 1D or

2D discretization grids of the form

$$G = \{x_i \ : \ x_i = x_0 + ih, \quad i = 0, \ldots, N-1\} \quad (x_0 = \xi_\ell, \quad x_{N-1} = \xi_r),$$

and

$$G = \overline{\Omega} \cap \big\{(x_i, y_j) \ : \ x_i = x_0 + ih, y_j = y_0 + jh, 0 \le i \le N_1 - 1, \ 0 \le j \le N_2 - 1\big\},$$

respectively. Here $\overline{\Omega}$ denotes the closure of Ω, $x_0 = \xi_\ell$, $x_{N_1-1} = \xi_r$, $y_0 = \xi_d$ and $y_{N_2-1} = \xi_u$. In either case a function

$$b : G \to \mathbb{R}^q$$

will be called a "q-dimensional vector grid function". Letting

$$\mathcal{I} = \big\{(i,j) \in \{0, \ldots, N_1 - 1\} \times \{0, \ldots, N_2 - 1\} \ : \ (x_i, y_j) \in G\big\},$$

we will also write $b(x_i) = b_i$ $(0 \le i \le N-1)$ and $b(x_i, y_j) = b_{ij}$ $((i,j) \in \mathcal{I})$. The set of q-dimensional vector grid functions defined on G will be denoted by $\mathcal{G}^q$.

It is important to mention that, although the two-dimensional setting described above does not impose any restrictions on the character of the domain Ω, for simplicity, the FC-SDNN solver presented in this paper assumes that the boundary of $\overline{\Omega}$ is given by a union of horizontal and vertical straight segments, each one of which runs along a Cartesian discretization line; see e.g. the Mach 3 forward-facing step case considered in Figure 19a. Extensions to general domains Ω, which could rely on either an embedded-boundary [6, 7, 29] approach, or an overlapping patch boundary-conforming curvilinear discretization strategy [1, 2, 5], is left for future work.

A spatially-discrete but time-continuous version of the solution vector $\boldsymbol{e}(\mathbf{x}, t)$ considered in Section 2 for 1D problems (resp. 2D problems) can be viewed as a time-dependent q-dimensional vector grid function $\boldsymbol{e}_i = \boldsymbol{e}_i(t)$ (resp. $\boldsymbol{e}_{ij} = \boldsymbol{e}_{ij}(t)$). Using $\boldsymbol{e}_h = \boldsymbol{e}_h(t)$ to refer generically to the $1D$ and $2D$ time-dependent grid functions $\boldsymbol{e}_i$ and $\boldsymbol{e}_{ij}$, the semidiscrete scheme for equation (11) becomes

$$\frac{d\boldsymbol{e}_h(t)}{dt} = L[\boldsymbol{e}_h(t)], \tag{21}$$

where L denotes a consistent discrete approximation of the spatial operator in (11).

The discrete time evolution of the problem, on the other hand, is produced, throughout this paper, by means of the 4-th order strong stability preserving Runge-Kutta scheme (SSPRK-4) [13]—which, while not providing high convergence orders for the non-smooth solutions considered in this paper, does lead to low temporal dispersion and diffusion over smooth space-time regions of the computational domain. The corresponding time step is selected adaptively at each time-step $t = t_n$ according to the expression

$$\Delta t = \frac{\mathrm{CFL}}{\pi\big(\frac{\max_{x \in \Omega} |S[\boldsymbol{e}](\mathbf{x},t)))|}{h} + \frac{\max_{x \in \Omega} \mu[\boldsymbol{e}](\boldsymbol{x},t)}{h^2}\big)}. \tag{22}$$

Here CFL is a constant parameter that must be selected for each problem considered (as illustrated by the various selections utilized in Section 4), and $\mu[\boldsymbol{e}](\mathbf{x}, t)$ and $S = S[\boldsymbol{e}](\mathbf{x}, t)$ denote the artificial viscosity (equations (37) and (42)) and a *maximum wave speed bound* (MWSB) operator (which must be appropriately selected for each equation; see Section 3.3) at the spatio-temporal point $(\mathbf{x}, t)$. (To avoid confusion we emphasize that equation (22) utilizes the *maximum* value for all $\mathbf{x} \in \Omega$ of the selected bound $S[\boldsymbol{e}](\mathbf{x}, t)$ on the *maximum* wave speed.)

To obtain FC-based approximate derivatives of a function $F : K \to \mathbb{R}$ defined on a one-dimensional interval $K = [x_0, x_{N-1}]$, whose values $(F_0, F_1, \ldots, F_{N-1})^T$ are given on the uniform mesh $\{x_0, x_1, \ldots, x_{N-1}\}$, the interval K is re-scaled to $[0, 1]$ and the corresponding continuation function F^c is obtained by means of the FC-Gram procedure described in Section 2.3. The approximate derivatives at all mesh points are then obtained by applying the IFFT algorithm to the Fourier coefficients

$$(\hat{F}^c)'_k = \frac{2\pi ik}{\beta}\hat{F}^c_k. \tag{23}$$

of the derivative of the series (19) and re-scaling back to the interval K.

All of the numerical derivatives needed to evaluate the spatial operator $L[\boldsymbol{e}_h(t)]$ are obtained via repeated application of the 1D FC differentiation procedure described above. For a function $F = F(x, y)$ defined on a two-dimensional domain Ω and whose values F_{ij} $((i, j) \in \mathcal{I})$ are given on a grid G of the type described above in this section, for example, partial derivatives with respect to x along the line $y = y_{j_0}$ for a relevant value of j_0 are obtained by differentiation of the FC expansion obtained for the function values $(F(x_i, y_{j_0}))_i$ for integers i such that $(i, j_0) \in \mathcal{I}$. The y differentiation process proceeds similarly. Mixed derivatives, finally, are produced by successive application of the x and y differentiation processes. Details concerning the filtered derivatives used in the proposed scheme are provided in Section 3.4.

The boundary conditions of Dirichlet and Neumann considered in this paper are imposed as part of the differentiation process described above. Dirichlet boundary conditions at time $t_{n,\nu}$ $(t_n < t_{n,\nu} \leq t_{n+1})$ corresponding to the ν-th SSPRK-4 stage $(\nu = 1, \ldots, 4)$ for the time-step starting at $t = t_n$, are simply imposed by overwriting the boundary values of the unknown solution vector $\boldsymbol{e}_h$ obtained at time $t = t_{n,\nu}$ with the given boundary values at that time, prior to the evaluation of the spatial derivatives needed for the subsequent SSPRK-4 stage. Neumann boundary conditions are similarly enforced by constructing appropriate continuation vectors (Section 2.3) after each stage of the SSPRK-4 scheme on the basis of the modified pre-computed matrix $\widetilde{Q}$ mentioned in Section 2.3.

It is known that enforcement of the given physical boundary conditions at intermediate Runge-Kutta stages, which is referred to as the "conventional method" in [8], may lead to a reduced temporal order of accuracy at spatial points in a neighborhood of the boundary of the domain boundary. This is not a significant concern in the context of this paper, where the global order of accuracy is limited in view of the discontinuous character of the solutions considered. Alternative approaches that preserve the order of accuracy for smooth solutions, such as those introduced in [8,31], could also be used in conjunction with the proposed approach. Another alternative, under which no boundary conditions are enforced at intermediate Runge-Kutta stages [20], can also be utilized in our context, but we have found the conventional method leads to smoother solutions near boundaries.

3.2 Neural network-induced smoothness-classification

3.2.1 Smoothness-classification operator and data pre-processing

The method described in the forthcoming Section 3.3 for determination of the artificial viscosity values $\mu[\boldsymbol{e}](\mathbf{x}, t)$ (cf. also Section 2.2) relies on the "degree of smoothness" of a certain function $\Phi(\boldsymbol{e})(\mathbf{x}, t)$ (called the "proxy variable") of the unknown solution vector $\boldsymbol{e}$. In detail, following [38], in this paper a proxy variable $\Phi(\boldsymbol{e})$ is used, which equals the velocity u, $\Phi(\boldsymbol{e}) = u$, (resp. the Mach number, $\Phi(\boldsymbol{e}) = \|\mathbf{u}\| \sqrt{\frac{\rho}{\gamma p}}$) for equations (2) through (4) (resp. equations (5) and (6)). The degree of

smoothness of the function $\Phi(\boldsymbol{e})$ at a certain time t is characterized by a smoothness-classification operator $\tau = \tau[\Phi(\boldsymbol{e})]$ that analyzes the oscillations in an FC expansion of $\Phi(\boldsymbol{e})$—which is itself obtained from the discrete numerical values $\boldsymbol{\phi} = \Phi(\boldsymbol{e}_h)$, so that, in particular, $\tau[\Phi(\boldsymbol{e})] = \tilde{\tau}[\boldsymbol{\phi}]$ for some discrete operator $\tilde{\tau}$. The determination (or, rather, estimation) of the degree of smoothness by the operator $\tilde{\tau}$ is effected on the basis of an Artificial Neural Network (ANN). (We introduce the operators τ and $\tilde{\tau}$ for the specific function $\Phi(\boldsymbol{e})$, but, clearly, the algorithm applies to arbitrary scalar or vector functions, as can be seen e.g. in the application of these operators, in the context of network training, in Section 3.2.2.)

We first describe the operator $\tilde{\tau} = \tilde{\tau}[\boldsymbol{\phi}]$ for for a conservation law over a one-dimensional interval $I = [\xi_\ell, \xi_r]$ discretized by an N-point equispaced mesh $(x_0, \ldots, x_{N-1})$ of mesh-size h, and for which FC expansions are obtained on the basis of the extended equispaced mesh $\{x_0, \ldots, x_{N+C-1}\}$. (Note that, in accordance with Section 2.3, this extended mesh includes the discrete points $\{x_0, \ldots, x_{N-1}\}$ in the interval I as well as the discrete points $\{x_N, \ldots, x_{N+C-1}\}$ in the FC extension region.) In this case, the evaluation of the operator $\tilde{\tau}$ proceeds as follows.

(i) Obtain the FC expansion coefficients $(\hat{\boldsymbol{\phi}}^c_{-M}, \ldots, \hat{\boldsymbol{\phi}}^c_M)^T$ of $\Phi(\boldsymbol{e})$ by applying the FC procedure described in Section 2.3 to the column vector $(\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{N-1})^T$. (Note that in the present 1D case we have $\boldsymbol{\phi}_j = \Phi(\boldsymbol{e}_j)$.)

(ii) For a suitable selected non-negative number $\delta < h$, evaluate the values $\boldsymbol{\phi}^{(\delta)}_j$ $(0 \le j \le N+C-1\,)$ of the FC expansion obtained in point (i) at the shifted grid points $x_0+\delta, x_1+\delta, \ldots, x_{N+C-1}+\delta$. This is achieved by applying the FFT algorithm to the "shifted" Fourier coefficients $\hat{\boldsymbol{\phi}}^\delta = (\hat{\boldsymbol{\phi}}^\delta_{-M}, \ldots, \hat{\boldsymbol{\phi}}^\delta_M)$ where $\hat{\boldsymbol{\phi}}^\delta_j = \hat{\boldsymbol{\phi}}^c_j \exp(\frac{2\pi i j \delta}{\beta})$. Here, as in Section 2.3 and equation (19), β denotes the length of the FC periodicity interval. Throughout this work, the value $\delta = \frac{h}{10}$ is used for classification of flow discontinuities. As indicated in Section 3.2.2, different values of δ are used in the training process.

(iii) For each $j \in \{0, \ldots, N-1\}$, form the seven-point stencil

$$\boldsymbol{\phi}^{(\delta,j)} = \big(\boldsymbol{\phi}^{(\delta)}_{m(j-3,N+C)}, \ldots, \boldsymbol{\phi}^{(\delta)}_{m(j,N+C)}, \ldots, \boldsymbol{\phi}^{(\delta)}_{m(j+3,N+C)}\big)^T$$

of values of the shifted grid function obtained per point (ii). (Here, for an integer $0 \le j \le P$, $m(j,P)$ denotes the remainder of j modulo P, that is to say, $m(j,P)$ is the only integer between 0 and $P-1$ such that $j - m(j,P)$ is an integer multiple of P. In view of the extended domain inherent in the continuation method, use of the remainder function m allows for the smoothness classification algorithm to continue to operate correctly even at points x_j near physical boundaries—for which the seven-point subgrid $(x_{j-3}, \ldots, x_j, \ldots, x_{j+3})$ may not be fully contained within the physical domain.)

(iv) Obtain the modified stencils $\widetilde{\boldsymbol{\phi}}^{(\delta,j)} = \big(\widetilde{\boldsymbol{\phi}}^{(\delta)}_{m(j-3,N+C)}, \ldots, \widetilde{\boldsymbol{\phi}}^{(\delta)}_{m(j,N+C)}, \ldots \widetilde{\boldsymbol{\phi}}^{(\delta)}_{m(j+3,N+C)}\big)^T$ given by

$$\widetilde{\boldsymbol{\phi}}^{(\delta)}_{m(j+r,N+C)} = \boldsymbol{\phi}^{(\delta)}_{m(j+r,N+C)} - \ell_{j+r} \quad (-3 \le r \le 3), \tag{24}$$

that result by subtracting the "straight line"

$$\ell_{j+r} = \boldsymbol{\phi}^{(\delta)}_{m(j-3,N+C)} + \frac{r+3}{6}(\boldsymbol{\phi}^{(\delta)}_{m(j+3,N+C)} - \boldsymbol{\phi}^{(\delta)}_{m(j-3,N+C)}) \quad (-3 \le r \le 3) \tag{25}$$

passing through the first and last stencil points.

(v) Rescale each stencil $\widetilde{\boldsymbol{\phi}}^{(\delta,j)}$ so as to obtain the ANN input stencils

$$\check{\boldsymbol{\phi}}^{(\delta,j)} = \big(\check{\boldsymbol{\phi}}^{(\delta)}_{m(j-3,N+C)}, \ldots, \check{\boldsymbol{\phi}}^{(\delta)}_{m(j,N+C)}, \cdots \check{\boldsymbol{\phi}}^{(\delta)}_{m(j+3,N+C)}\big)^T,$$

given by

$$\check{\boldsymbol{\phi}}^{(\delta)}_{m(j+r,N+C)} = \frac{2\widetilde{\boldsymbol{\phi}}^{(\delta)}_{m(j+r,N+C)} - M_j^{(+)} - M_j^{(-)}}{M_j^{(+)} - M_j^{(-)}} \quad (-3 \le r \le 3) \tag{26}$$

where

$$M_j^{(+)} = \max_{-3\le r\le 3} \widetilde{\boldsymbol{\phi}}^{(\delta)}_{m(j+r,N+C)} \quad \text{and} \quad M_j^{(-)} = \min_{-3\le r\le 3} \widetilde{\boldsymbol{\phi}}^{(\delta)}_{m(j+r,N+C)}. \tag{27}$$

Clearly, the new stencil entries satisfy satisfy $-1 \le \check{\boldsymbol{\phi}}^{(\delta)}_{m(j+r,N+C)} \le 1$.

(vi) Apply the ANN algorithm described in Section 3.2.2 to each one of the stencils $\check{\boldsymbol{\phi}}^{(\delta,j)} = \big\{\check{\boldsymbol{\phi}}^{(\delta)}_{m(j-3,N+C)}, \ldots, \check{\boldsymbol{\phi}}^{(\delta)}_{m(j,N+C)}, \cdots \check{\boldsymbol{\phi}}^{(\delta)}_{m(j+3,N+C)}\big\}$, to produce a four-dimensional vector w^j of estimated probabilities (EP) for each $j \in [0, \ldots, N-1]$, where w_1^j is the EP that $\Phi(\boldsymbol{e})$ is discontinuous on the subinterval $I_j = [x_j - 3h, x_j + 3h]$, where, for $i = 2, 3$, w_i^j equals the EP that $\Phi(\boldsymbol{e}) \in \mathcal{C}^{i-2} \setminus \mathcal{C}^{i-1}$ on I_j, and where, for $i = 4$, w_i^j equals the EP that $\Phi(\boldsymbol{e}) \in \mathcal{C}^2$ on I_j. Define $\tau[\boldsymbol{\phi}]_j$ as the index i corresponding to the maximum entry of w_i^j $(i = 1, \ldots 4)$:

$$\tilde{\tau}[\boldsymbol{\phi}]_j = \arg\max_{1\le i\le 4}(w_i^j) \quad (0 \le j \le N-1). \tag{28}$$

(Note that, for points x_j close to physical boundaries, the interval $I_j = [x_j - 3h, x_j + 3h]$, within which the smoothness of the function $\Phi(\boldsymbol{e})$ is estimated, can extend beyond the physical domain and into the extended Fourier Continuation region; cf. also point iii above.)

This completes the definition of the 1D smoothness classification operator $\tilde{\tau}$.

For 2D configurations, in turn, we define a two-dimensional smoothness classification operator $\tau_{xy}[\Phi(\boldsymbol{e})] = \tilde{\tau}_{xy}[\boldsymbol{\phi}]$, similar to the 1D operator, which classifies the smoothness of the proxy variable $\Phi(\boldsymbol{e})$ on the basis of its discrete values $\boldsymbol{\phi} = \boldsymbol{\phi}_{ij}$. Note the xy subindex which indicates 2D classification operators τ_{xy} and $\tilde{\tau}_{xy}$; certain associated 1D "partial" discrete classification operators in the x and y variables, which are used in the definition of $\tilde{\tau}_{xy}$, will be denoted by $\tilde{\tau}_x$ and $\tilde{\tau}_y$, respectively.

In order to introduce the operator $\tilde{\tau}_{xy}$ we utilize certain 1D sections of both the set $\mathcal{I}$ and the grid function $\boldsymbol{\phi} = \boldsymbol{\phi}_{ij}$ (see Section 3.1). Thus, the i-th horizontal section (resp. the j vertical section) of $\mathcal{I}$ is defined by $\mathcal{I}_{i:} = \{j \in \mathbb{Z} \ : \ (i,j) \in \mathcal{I}\}$ (resp. $\mathcal{I}_{:j} = \{i \in \mathbb{Z} \ : \ (i,j) \in \mathcal{I}\}$). Similarly, for a given 2D grid function $\boldsymbol{\phi} = \boldsymbol{\phi}_{ij}$, the i-th horizontal section $\boldsymbol{\phi}_{i:}$ (resp. the j vertical section $\boldsymbol{\phi}_{:j}$) of $\boldsymbol{\phi}$ is defined by $(\boldsymbol{\phi}_{i:})_j = \boldsymbol{\phi}_{ij}$, $j \in \mathcal{I}_{i:}$ (resp. $(\boldsymbol{\phi}_{:j})_i = \boldsymbol{\phi}_{ij}$, $i \in \mathcal{I}_{:j}$). Utilizing these notations we define

$$\tilde{\tau}_{xy}[\boldsymbol{\phi}]_{ij} = \min\left\{\tilde{\tau}_x[\boldsymbol{\phi}_{i:}]_j, \tilde{\tau}_y[\boldsymbol{\phi}_{:j}]_i\right\}, \tag{29}$$

where, as suggested above, $\tilde{\tau}_x$ (resp. $\tilde{\tau}_y$) denotes the discrete one-dimensional classification operator along the x direction (resp. the y direction), given by (28) but with $j \in \mathcal{I}_{i:}$ (resp. $i \in \mathcal{I}_{:j}$). In other words, the 2D smoothness operator $\tilde{\tau}_{xy}$ equals the lowest degree of smoothness between the classifications given by the two partial classification operators.

Remark 2. Small amplitude noise in the proxy variable can affect ANN analysis, leading to misclassification of stencils and under-prediction of the smoothness of the proxy variable. In order to eliminate the effect of noise, stencils $\check{\boldsymbol{\phi}}_j^{(\delta,j)}$ for which $M_j^{(+)} - M_j^{(-)} \le \varepsilon$, for a prescribed value of ε, are assigned regularity $\tilde{\tau}[\boldsymbol{\phi}]_j = 4$. Throughout this paper we have used the value $\varepsilon = 0.01$.

3.2.2 Neural network architecture and training

The proposed strategy relies on standard neural-network techniques and nomenclature [12, Sec. 6]: it utilizes an ANN with a depth of four layers, including three fully-connected hidden layers of sixteen neurons each. The ANN takes as input a seven-point "preprocessed stencil" $z = (z_1, z_2, z_3, z_4, z_5, z_6, z_7)^T$—namely, a stencil z that results from an application of points (i) to (v) in the previous section to the 401-coordinate vector $\mathbf{F}$ of grid values obtained for a given function F on a 401-point equispaced grid in the interval $[0, 2\pi]$—in place of the grid values of the proxy variable $\Phi(\boldsymbol{e})$—resulting in a total of 401 stencils, one centered around each one of the 401 grid points considered; cf. points (i) to (iii) and note that, on the basis of the FC-extended function, the stencils near endpoints draw values at grid points outside the interval $[0, 2\pi]$. (A variation of point (ii) is used in the training process: shift values $\delta = \frac{h}{10}, \frac{2h}{10} \dots \frac{10h}{10}$ are used to produce a variety of seven-point stencils for *training* purposes instead of the single value $\delta = \frac{h}{10}$ used while employing the ANN in the *classification* process.) The output of the final layer of the ANN is a four-dimensional vector $\widetilde{w} = (\widetilde{w}_1, \widetilde{w}_2, \widetilde{w}_3, \widetilde{w}_4)^T$, from which, via an application of the softmax activation function [12, Sec. 4.1], the EP mentioned in point (vi) of the previous section, are obtained:

$$w_i = \frac{e^{\widetilde{w}_i}}{\sum_{\ell=1}^{4} e^{\widetilde{w}_\ell}}, \quad 1 \le i \le 4. \tag{30}$$

(The values w_i^j $(1 \le i \le 4)$ mentioned in point (vi) result from the expression (30) when the overall scheme described above in the present Section 3.2.2 is applied to $z = \check{\boldsymbol{\phi}}^{(\delta,j)}$.) The ELU activation function

$$\mathrm{ELU}(x; \alpha_0) = \begin{cases} x & \text{if } x > 0 \\ \alpha_0(e^x - 1) & \text{if } x \le 0, \end{cases} \tag{31}$$

with $\alpha_0 = 1$, is used in all of the hidden layers.

In what follows we consider, for both the ANN training and validation processes, the data set $\mathcal{D}_{\mathcal{F}}$ of preprocessed stencils resulting from the set $\mathcal{F} = \big\{\big(F_k, D_k\big), k = 1, 2, \dots\big\}$ of all pairs $\big(F_k, D_k\big)$, where F_k is a function defined on the interval $[0, 2\pi]$ and where D_k is a certain "restriction domain", as described in what follows. The functions F_k are all the functions obtained on the basis of one of the five different parameter-dependent analytic expressions

$$\begin{aligned}
f_1(x) &= \sin(2ax) \\
f_2(x) &= a|x - \pi| \\
f_3(x) &= \begin{cases} a_1 & \text{if} \quad |x - \pi| \le a_3 \\ a_2 & \text{if} \quad |x - \pi| > a_3 \end{cases} \\
f_4(x) &= \begin{cases} a_1|x - \pi| - a_1 a_3 & \text{if} \quad |x - \pi| \le a_3 \\ a_2|x - \pi| - a_2 a_3 & \text{if} \quad |x - \pi| > a_3 \end{cases} \\
f_5(x) &= \begin{cases} 0.5a_1|x - \pi|^2 - a_1 a_3 & \text{if} |x - \pi| \le a_3 \\ a_2|x - \pi|^2 - a_2 - 0.5a_3^2(a_1 - a_2) & \text{if} |x - \pi| > a_3 \end{cases}
\end{aligned} \tag{32}$$

proposed in [38], for each one of the possible selections of the parameters a, a_1, a_2, a_3, as prescribed in Table 1. The corresponding parameter dependent restriction domains D_k are also prescribed in Table 1; in all cases D_k is a subinterval of $[0, 2\pi]$.

The restriction domains D_k are used to constrain the choice of stencils to be used among all of the 401 stencils available for each function F_k—so that, for a given function F_k, the preprocessed

stencils associated with gridpoints contained within D_k, but not others, are included within the set $\mathcal{D}_{\mathcal{F}}$. The set $\mathcal{D}_{\mathcal{F}}$ is randomly partitioned into a training set $\mathcal{D}_{\mathcal{F}}^{\mathcal{T}}$ containing 80% of the elements in $\mathcal{D}_{\mathcal{F}}$ (which is used for optimization of the ANN weights and biases), and a validation set $\mathcal{D}_{\mathcal{F}}^{\mathcal{V}}$ containing the remaining 20%—which is used to evaluate the accuracy of the ANN after each epoch [12, Sec. 7.7].

The network training and validation processes rely on use of a one-hot encoded label function C defined on $\mathcal{D}_{\mathcal{F}}$ which takes one of four possible values. Thus each stencil $z \in \mathcal{D}_{\mathcal{F}}$ is labeled by a class vector $C(z) = (C_1(z), C_2(z), C_3(z), C_4(z))^T$, where for each z, $C(z) = (1,0,0,0)^T$, $(0,1,0,0)^T$, $(0,0,1,0)^T$ or $(0,0,0,1)^T$ depending on whether z was obtained from a function F_k that is $\mathcal{C}^2$, $\mathcal{C}^1 \backslash \mathcal{C}^2$, $\mathcal{C}^0 \setminus \mathcal{C}^1$, or discontinuous over the subinterval $I_z \cap [0, 2\pi]$, where I_z denotes the interval spanned by the set of seven consecutive grid points associated with the prepocessed stencil z.

The ANN is characterized by a relatively large number of parameters contained in four weight matrices of various dimensions (a 16×7 matrix, a 4×16, and two 16×16 matrices), as well as four bias vectors (one 4-dimensional vector and three 16-dimensional vectors). In what follows a single parameter vector X is utilized which contains all of the elements in these matrices and vectors in some arbitrarily prescribed order. Utilizing the parameter vector X, for each stencil z the ANN produces the estimates w_i, given by (30), of the actual classification vector $C(z)$. In order to account for the dependence of w_i on the parameter vector X for each stencil z, in what follows we write $w_i = A_i(X, z)$ $(1 \leq i \leq 4)$.

The parameter vector X itself is obtained by training the network on the basis of existing data, which is accomplished in the present context by selecting X as an approximate minimizer of the "cross entropy" loss function [12, Sec. 6.2]

$$\mathcal{L}(X) = -\frac{1}{N_{\mathcal{T}}} \sum_{z \in \mathcal{D}_{\mathcal{F}}^{\mathcal{T}}} \sum_{i=1}^{4} C_i(z) \log(A_i(X, z)) \tag{33}$$

over all z in the training set $\mathcal{D}_{\mathcal{F}}^{\mathcal{T}}$, where $N_{\mathcal{T}}$ denotes the number of elements in the training set. The loss function $\mathcal{L}$ provides an indicator of the discrepancy between the EP $A(X, z) = (A_1(X, z), A_2(X, z), A_3(X, z), A_4(X, z))$ produced by the ANN and the corresponding classification vector $C(z) = (C_1(z), C_2(z), C_3(z), C_4(z))^T$ introduced above, over all the preprocessed stencils $z \in \mathcal{D}_{\mathcal{F}}$. The minimizing vector X of weights and biases define the network, which can subsequently be used to produce $A(X, z)$ for any given preprocessed stencil z.

The Neural Network is trained (that is, the loss function L is minimized with respect to X) by exploiting the stochastic gradient descent algorithm without momentum [12, Secs. 8.1, 8.4], with mini batches of size 128 and with a constant learning rate of 10^{-6}. The weight matrices and bias vectors are initialized using the Glorot initialization [11]. The training set is randomly re-shuffled after every epoch, and the validation data is re-shuffled before each network validation. The network with the highest validation accuracy that was obtained over a few neural network retrains, which is used for all the illustrations presented in this paper, has a training accuracy of 99.61% and validation accuracy of 99.58%.

3.3 SDNN-localized artificial viscosity algorithm

As indicated in Section 1, in order to avoid introduction of spurious irregularities in the flow field, the algorithm proposed in this paper relies on use of smoothly varying artificial viscosity assignments. For a given discrete solution vector $\boldsymbol{e}_h$, the necessary grid values of the artificial viscosity, which

$f(x)$	Parameters	restriction domains	τ
f_1	$a = \{-20, -19.5, \ldots, 19, 5\}$	$[0, 2\pi]$	4
f_2	$a = \{-10, -9, \ldots, 10\}$	$[3.53, 5.89]$	4
f_3	$a_1 = \{-10, -9, \ldots, 9\}$ $a_2 = \{-10, -9, \ldots, 9\}$ $a_3 = \{0.25, 0.5, \ldots, 2.5\}$ s.t. $a_1 \neq a_2$	$[\pi + a_3 - 0.05, \pi + a_3 + 0.05]$	1
f_4	$a_1 = \{-10, -9, \ldots, 9\}$ $a_2 = \{-10, -9, \ldots, 9\}$ $a_3 = \{0.25, 0.5, \ldots, 2.5\}$ s.t. $a_1 > 2a_2$ or $a_1 < 0.5a_2$	$[\pi + a_3 - 0.05, \pi + a_3 + 0.05]$	2
f_5	$a_1 = \{-10, -9, \ldots, 9\}$ $a_2 = \{-10, -9, \ldots, 9\}$ $a_3 = \{0.25, 0.5, \ldots, 2.5\}$ s.t. $a_1 > 5a_2$ or $a_1 < 0.2a_2$	$[\pi + a_3 - 0.05, \pi + a_3 + 0.05]$	3

Table 1: Data set.

correspond to discrete values of the continuous operator $\mu = \mu[\boldsymbol{e}]$ in (10), are provided by a certain discrete viscosity operator $\tilde{\mu} = \tilde{\mu}[\boldsymbol{e}_h]$. The discrete operator $\tilde{\mu}$ is defined in terms of a number of flow- and geometry-related concepts, namely the proxy variable $\boldsymbol{\phi}$ defined in Section 3.2.1 and the smoothness-classification operator given by equations (28) and (29) for the 1D and 2D cases, respectively, as well as certain additional functions and operators, namely a "weight function" R and "weight operator" $\widetilde{R}$, an MWSB operator S (see Section 3.1) and its discrete version $\widetilde{S}$, a sequence of "localization stencils" (denoted by L^i with $0 \leq i \leq N-1$ in the 1D case, and by $L^{i,j}$ with $(i,j) \in \mathcal{I}$ in the 2D case) , and a "windowed-localization" operator Λ. A detailed description of the 1D and 2D discrete artificial viscosity operators $\tilde{\mu} = \tilde{\mu}[\boldsymbol{e}_h]$ is provided in Sections 3.3.1 and 3.3.2, respectively.

3.3.1 One-dimensional case

The proxy variable $\boldsymbol{\phi}$ and 1D smoothness-classification operator τ that are used in the definition of the 1D artificial viscosity operator have been described earlier in this paper; in what follows we introduce the additional necessary functions and operators mentioned above.

The weight function R assigns a viscosity weight according to the smoothness classification; throughout this paper we use the weight function given by $R(1) = 2$, $R(2) = 1$, $R(3) = 0$, and $R(4) = 0$; the corresponding grid-function operator $\widetilde{R}$, which acts over the set of grid functions η with grid values 1, 2, 3 and 4, is defined by $\widetilde{R}[\eta]_i = R(\eta_i)$.

The MWSB operator $S : \mathcal{G}^q \to \mathcal{G}$ maps the q-dimensional vector grid function $\boldsymbol{e}_h$ onto a grid function corresponding to a bound on the maximum eigenvalue of the flux Jacobian $(J_{\boldsymbol{e}} f)_{k\ell} = (\partial_{\boldsymbol{e}^\ell} f_k)$ at $\boldsymbol{e} = \boldsymbol{e}_h$, where $\boldsymbol{e}^\ell$, (resp. f_k) denotes the ℓ-th (resp. k-th) component of the unknowns solution vector $\boldsymbol{e}$ (resp. of the convective flux $f(\boldsymbol{e})$). For the one-dimensional problems, the MWSB operator $S(\boldsymbol{e})$ (resp. the discretized operator $\widetilde{S}[\boldsymbol{e}_h]$ on the grid $\{x_i\}$) is taken to *equal* the maximum characteristic speed (since the maximum characteristic speed is easily computable from the velocity in the 1D case), so that $S(\boldsymbol{e}) = a$ (resp $\widetilde{S}[\boldsymbol{e}_h]_i = a_i$) for the 1D Linear Advection equation (2),

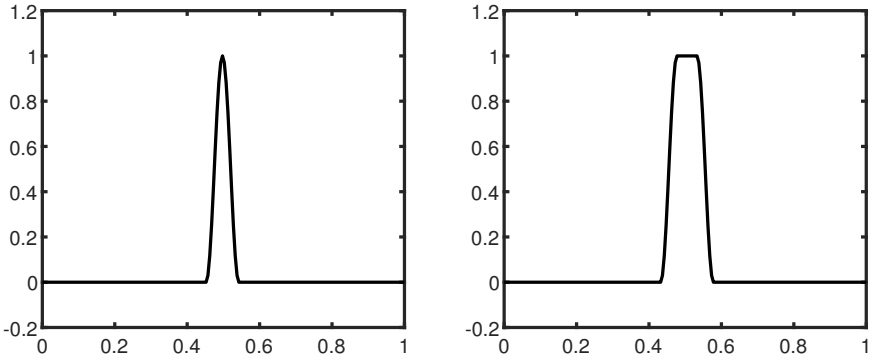


Figure 2: Left: Windowing function $W^j(x)$ (with $j = 100$, on the domain $[0, 1]$, and using $N = 200$ discretization points) utilized in the definition of the windowed-localization operator Λ. Right: Windowing function $q_{18,9}(x - z)$ utilized for the localized filtering of the initial condition on the domain $[0, 1]$ (using $N = 200$ discretization points) for a discontinuity located at $z = 0.5$.

$S(\boldsymbol{e}) = |u|$ (resp $\widetilde{S}[\boldsymbol{e}_h]_i = |u_i|$) for the 1D Burgers equation (3), and $S(\boldsymbol{e}) = |u| + a$ [26] (resp $\widetilde{S}[\boldsymbol{e}_h]_i = |u_i| + a_i$) in the case of the 1D Euler problem (5) (in terms of the sound speed (9)).

The localization stencil L^i $(0 \leq i \leq N - 1)$ is a set of seven points that surround x_i: $L^i = \{x_{i-3}, \ldots, x_i, \ldots x_{i+3}\}$ for $4 \leq i \leq N - 4$, $L^i = \{x_0, \ldots x_6\}$ for $i \leq 3$, and $L^i = \{x_{N-7}, \ldots x_{N-1}\}$ for $i \geq N - 3$.

The windowed-localization operator Λ is constructed on the basis of the window function

$$q_{c,r}(x) = \begin{cases} 1 & \text{if} \quad |x| < ch/2 \\ \cos^2\left(\frac{\pi(|x|-ch/2)}{rh}\right) & \text{if} \quad ch/2 \leq |x| \leq (c/2+r)h \\ 0 & \text{if} \quad |x| > (c/2+r)h, \end{cases} \tag{34}$$

depicted in Figure 2 left, where c and r denote small positive integer values, with c even. (Note that the $q_{c,r}$ notation does not explicitly display the h-dependence of this function.) Using the window function $q_{c,r}$, two sequences of windowing functions, denoted by W^j and $\check{W}^j$ $(0 \leq j \leq N - 1)$, are defined, where the second sequence is a normalized version of the former. In detail $W^j(x)$ is obtained by translation of the function $q_{c,r}$ with $c = 0$ and $r = 9$: $W^j(x) = q_{0,9}(x - x_j)$; the corresponding grid values of this function on the grid $\{x_i\}$ are denoted by $W_i^j = W^j(x_i)$. The normalized windowing functions $\check{W}^j$ and the windowed-localization operator Λ, finally, are given by

$$\check{W}_i^j = \frac{W_i^j}{\sum_{k=0}^{N-1} W_k^j}, \tag{35}$$

and

$$\Lambda[b]_i = \sum_{k=0}^{N-1} \check{W}_i^k b_k, \tag{36}$$

respectively. Using these operators and functions, we define the 1D artificial viscosity operator

$$\tilde{\mu}[\boldsymbol{e}_h]_i = \Lambda[\widetilde{R}(\tilde{\tau}[\boldsymbol{\phi}])]_i \cdot \max_{j \in L^i}(\widetilde{S}[\boldsymbol{e}_h]_j)h; \tag{37}$$

as mentioned in Section 1 and demonstrated in Section 4, use of the smooth artificial viscosity assignments produced by this expression yield smooth flows away from shocks and other discontinuities. Note that the localization stencils L^i $(0 \leq i \leq N)$ used in equation (37) were designed to

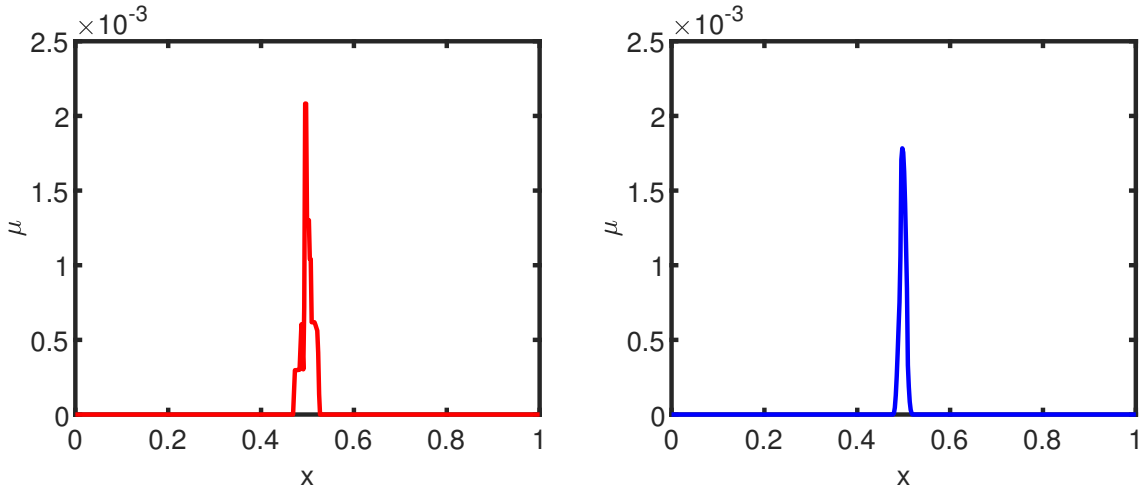


Figure 3: Comparison of the viscosity functions arising at the first viscous time-step for the Sod problem, using $N = 500$ discretization points. Left: viscosity used in [38]. Right: viscosity used in the present FC-SDNN method (equation (37)).

differ from those defined in point (iii) of Section 3.2.1: unlike the latter, the former ones do not use values of the FC-extension of the solution outside the physical domain. This difference relates to the nature of the functionality required in each case: viscosity assignment in the first case, which should be based on the local character of the solution in physical space, and detection of solution discontinuities in the second case, which can be gleaned from consideration of the FC extension.

It is important to note the essential role of the windowed-localization operator in the assignment of smooth viscosity profiles. The smooth character of the resulting viscosity functions is illustrated in Figure 3, which showcases the viscosity assignments corresponding to the second time-step in the solution process. (This run corresponds to the Sod problem described in Section 4.3.1.) The left image displays the viscosity profiles used in [38] (which do not utilize the smoothing windows (36)) and the right image presents the window-based viscosity profile (37). The right-hand profile, which is comparable in size but, in fact, more sharply focused around the shock than the non-smooth profile on the left-hand image, helps eliminate spurious oscillations that otherwise arise from viscosity non-smoothness, and allows the FC-SDNN method to produce smooth flow fields, as demonstrated in Section 4.3.1. The functions $q_{c,r}$ used to construct the windowed-localization operator are translated and scaled versions of the Hann window functions, also called "lag windows" [3, Sec. 5], which are commonly used as frequency filters in the field of signal processing. The normalized form used in this paper, equation (35), combines several such window functions to produce a "partition of unity" in the sense of differential geometry—that is, a set of non-negative functions whose sum equals one throughout the domain.

Remark 3. For the case of a 1D periodic problem, such as those considered in Section 4.1.2, the localization stencils and the windowing functions are defined by $L^i = \left\{x_{m(i-3,N)}, \ldots, x_{m(i,N)}, \ldots x_{m(i+3,N)}\right\}$, where the *modulo* function m is defined in Section 3.2.1, and where $W_i^j = q_{c,r}(|x_j - x_{\widetilde{m}(i,j,N)}|)$. Here, for $s = \frac{c}{2} + r$ we have set

$$\widetilde{m}(i,j,N) = \begin{cases} j+N-i & \text{if} \quad j < 2s \text{ and } N-2s+j \le i \le N-1 \\ j-1-i & \text{if} \quad N-2s < j \le N-1 \text{ and } 0 \le i \le 2s-(N-j) \\ i & \text{else.} \end{cases} \tag{38}$$

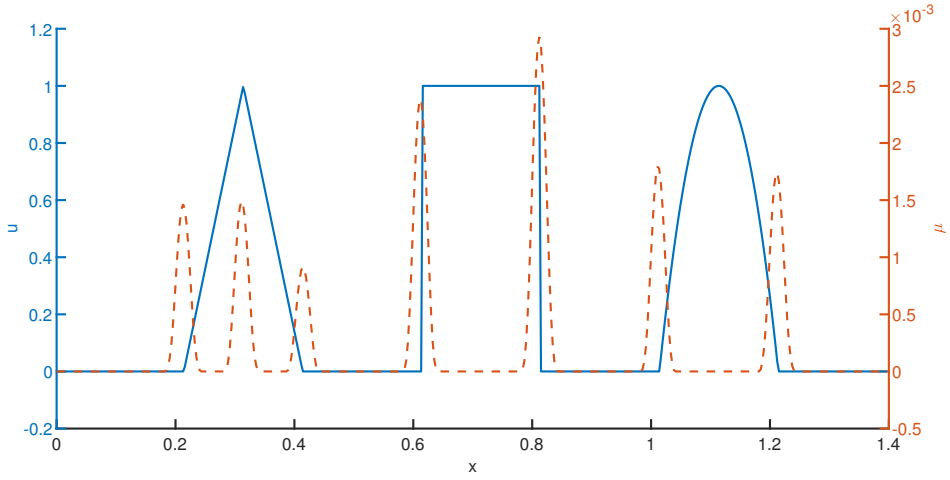


Figure 4: Viscosity assignment (orange dashed line) resulting from application of the SDNN-localized artificial viscosity algorithm to the initial condition (39) (blue solid line), using $N = 500$ discretization points.

The values $c = 0$ and $r = 9$ considered previously are once again used in the periodic context.

As an example, Figure 4 displays the viscosity assignments produced, by the method described in this section for the function

$$u(x) = \begin{cases} 10(x - 0.2) & \text{if } 0.2 < x \leq 0.3 \\ 10(0.4 - x) & \text{if } 0.3 < x \leq 0.4 \\ 1 & \text{if } 0.6 < x \leq 0.8 \\ 100(x - 1)(1.2 - x) & \text{if } 1 < x \leq 1.2 \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

in the interval $[0, 1.4]$. Clearly, the viscosity profiles are smooth and they are supported around points where the function u is not smooth.

3.3.2 Two-dimensional case

The definition of the 2D viscosity operator follows similarly as the one for the 1D case, with adequately modified versions of the underlying functions and operators. In detail, in the present 2D case we define $\widetilde{R}$ and R ($\widetilde{R}[\eta]_{ij} = R(\eta_{ij})$) as in the 1D case, but using the values $R(1) = 1.5$, $R(2) = 1$, $R(3) = 0.5$, and $R(4) = 0$. We note that, while the definition of the 1D weight function R led to stable 2D simulations, it was found through experimentation that use of the modified 2D definition yields smoother flow profiles away from shocks for all the 2D problems considered (and, conversely, while the 2D definition can stably be used for the 1D problems, sharper resolutions of shocks and contact discontinuities were obtained with the 1D definition). Additional comments in this regard can be found in the paragraph entitled "Higher spatial dimensionality" in Section 5). The 7×7 localization stencils $L^{i,j}$ ($(i, j) \in \mathcal{I}$) are defined in terms of the 1D localization stencils L^i and L^j via the relation $L^{i,j} = L^i \times L^j$. For the 2D scalar Burgers equation, the MWSB operator $S[\boldsymbol{e}]$ (resp. the discrete operator $\widetilde{S}[\boldsymbol{e}_h]$) is taken to equal the maximum characteristic speed, that is $S[\boldsymbol{e}] = |u|$ (resp. $\widetilde{S}[\boldsymbol{e}_h]_{ij} = |u_{ij}|$ for $(i, j) \in \mathcal{I}$). In the case of the 2D Euler problem, the MWSB operator S used in this paper assigns to $\boldsymbol{e}$ the upper bound $S(\boldsymbol{e}) = |u| + |v| + a$ on the speed of

propagation $\mathbf{u}\cdot\vec{\kappa}+a$ of the wave corresponding to the largest eigenvalue of the 2D Flux-Jacobian (which, in a direction supported by the unit vector $\vec{\kappa}$, equals $\mathbf{u}\cdot\vec{\kappa}+a$ [16, Sec. 16.3 and 16.5]), so that the discrete operator $\widetilde{S}$ we propose is given on the grid by

$$\widetilde{S}[\boldsymbol{e}_h]_{ij} = |u_{ij}| + |v_{ij}| + a_{ij}. \tag{40}$$

It is relevant to note that the MWSB operator (40), which equals a plus the sum of the absolute values of the components of the velocity vector $\mathbf{u}$, differs slightly from the upper-bound selected in [14, 21], where the (equivalent) Euclidean-norm of $\mathbf{u}$ was used instead. The two-dimensional local-window operator, finally, is given by

$$\Lambda[b]_{ij} = \sum_{(k,\ell)\in\mathcal{I}} \check{W}_i^k \check{W}_j^\ell b_{k\ell} \tag{41}$$

—which, clearly, can be obtained in practice by applying consecutively the 1D local window operator introduced in Section 3.3.1 in the horizontal and vertical directions consecutively. As in the 1D case, using these operators and functions, we define the 2D artificial viscosity operator

$$\tilde{\mu}[\boldsymbol{e}_h]_{ij} = \Lambda[\widetilde{R}(\tilde{\tau}_{xy}[\boldsymbol{\phi}])]_{ij} \cdot \max_{(k,\ell)\in L^{ij}} (S[\boldsymbol{e}_h]_{k\ell})h. \tag{42}$$

3.4 Spectral filtering

Spectral methods regularly use filtering strategies in order to control the error growth in the unresolved high frequency modes. One such "global" filtering strategy is employed in the context of this paper as well, in conjunction with FC [1, 2], as detailed in Section 3.4.1. Additionally, a new "localized" filtering strategy 3.4.2 is introduced in this paper, in order to regularize discontinuous initial conditions, while avoiding the over-smearing of smooth flow-features. Details regarding the global and localized filtering strategies are provided in what follows.

3.4.1 Global filtering strategy

As indicated above, the proposed algorithm employs spectral filters in conjunction with the FC method to control the error growth in unresolved high frequency modes [1, 2]. For a given Fourier Continuation expansion

$$\sum_{k=-M}^{M} \hat{F}_k^c \exp(2\pi i k x/\beta)$$

the corresponding globally-filtered Fourier Continuation expansion is given by

$$\widetilde{F}_g = \sum_{k=-M}^{M} \hat{F}_k^c \sigma\Big(\frac{2k}{N+C}\Big) \exp(2\pi i k x/\beta) \tag{43}$$

where

$$\sigma\Big(\frac{2k}{N+C}\Big) = \exp\Big(-\alpha_f\Big(\frac{2k}{N+C}\Big)^{p_f}\Big)$$

for adequately chosen values of the positive integer p_f and the real parameter $\alpha_f > 0$. For applications involving two-dimensional domains the spectral filter is applied sequentially, one dimension at a time.

In the algorithm proposed in this paper, all the components of the unknown solution vector $\boldsymbol{e}$ are filtered using this procedure at every time step following the initial time, using the parameter values $\alpha_f = 10$ and $p_f = 14$, as indicated in Algorithm 1.

3.4.2 Localized discontinuity-smearing for initial data

In order to avoid the introduction of spurious oscillations arising from discontinuities in the initial condition, the spectral filter considered in the previous section is additionally applied, in a modified form, before the time-stepping process is initiated. A stronger filter is used to treat the initial conditions, however, since, unlike the flow field for positive times, the initial conditions are not affected by artificial viscosity. In order to avoid unduly degrading the representation of the smooth features of the initial data, on the other hand, a localized discontinuity-smearing method, based on use of filtering and windowing is used, that is described in what follows.

We first present the discontinuity-smearing approach for a 1D function $F : I \to \mathbb{R}$, defined on a one-dimensional interval I, which is discontinuous at a single point $z \in I$. In this case, the smeared-discontinuity function $\widetilde{F}_{\rm sm}$, which combines the globally filtered function $\widetilde{F}_g$ in a neighborhood of the discontinuity with the unfiltered function elsewhere, is defined by

$$\widetilde{F}_{\rm sm}(x) = q_{c,r}(x - z)\widetilde{F}_g(x) + (1 - q_{c,r}(x - z))F(x), \tag{44}$$

Remark 4. Throughout this paper, windows $q_{c,r}$ with $c = 18$ and $r = 9$ and globally filtered functions $\widetilde{F}_g$ with filter parameters $\alpha_f = 10$ and $p_f = 2$, which is depicted in Figure 2 right, were used for the initial-condition filtering problem, except as noted below in cases resulting in window overlap.

In case multiple discontinuities exist the procedure is repeated around each discontinuity point. Should the support of two or more of the associated windowing functions overlap, then each group of overlapping windows is replaced by a single window which equals zero outside the union of the supports of the windows in the group, and which equals one except in the rise regions for the leftmost and rightmost window functions in the group.

In the 2D case the localized discontinuity-smearing strategy for the initial condition is first performed along every horizontal line $y = y_{j:}$ for $0 \le j \le N_2 - 1$. The resulting "partially" filtered function is then filtered along every vertical line $x = x_i$ for $0 \le i \le N - 1$ using the same procedure. For PDE problems involving vectorial unknowns, further, the discontinuity-smearing strategy is applied to each component separately.

Remark 5. As indicated in line 10 of Algorithm 1, only initial data that are spatially discontinuous are treated by the discontinuity-smearing procedure. Note that, for such data, the spatial positions of the initial-data discontinuities are explicitly known, as required by the filtering procedure used.

3.5 Algorithm pseudo-code

A pseudo-code for the complete FC-SDNN numerical method for the various equations considered in this paper, and for both 1D and 2D cases, is presented in Algorithm 1.

It may be useful to emphasize that the FC procedure introduced in Section 2.3, without modifications, is used in Algorithm 1 even as shocks or other solution discontinuities exist at or around domain boundaries; cf. also the paragraph entitled "FC order" in Section 5. Additionally, we note

Algorithm 1 FC-SDNN algorithm

\\Initialization.
Input the trained ANN weights and biases (Section 3.2.2).
Initialize the unknown solution vector $\boldsymbol{e}_h$ (Section 2.1) to the given initial-condition values over the given spatial grid.
Initialize time: $t = 0$.
while $t < T$ **do**
    Evaluate the proxy variable $\boldsymbol{\phi}$ corresponding to $\boldsymbol{e}_h$ at all spatial grid points.
    Obtain the smoothness classification operator values ($\tilde{\tau}[\boldsymbol{\phi}]$, equation (28), in the 1D case, or $\tilde{\tau}_{xy}[\boldsymbol{\phi}]$, equation (29), in the 2D case) at all grid points by applying steps (i) through (vi) in Section 3.2.1 as required in each case, 1D or 2D.
    Evaluate the MWSB operator $\widetilde{S}[\boldsymbol{e}_h]$ at all spatial grid points (Section 3.3.1 in the 1D cases, and Section 3.3.2 in the 2D cases).
    Determine the artificial viscosity assignments $\tilde{\mu}[\boldsymbol{e}_h]$ (Equation (37) in the 1D case or equation (42) in the 2D case) at all spatial grid points.
    (Case $t = 0$, discontinuous initial data only) Apply localized discontinuity-smearing (Section 3.4.2) to the solution vector $\boldsymbol{e}_h$ and overwrite $\boldsymbol{e}_h$ with the resulting values.
    (Case $t > 0$) Apply global filtering (Section 3.4.1) to the solution vector $\boldsymbol{e}_h$ and overwrite $\boldsymbol{e}_h$ with the resulting values.
    Evaluate the temporal step-size Δt by substituting the discrete version $\widetilde{S}[\boldsymbol{e}_h]$ and $\tilde{\mu}[\boldsymbol{e}_h]$ of $S[\boldsymbol{e}]$ and $\mu[\boldsymbol{e}]$ in equation (22).
    Perform the FC-based spatial differentiations required for time-stepping for given (Dirichlet or Neumann) boundary conditions (Sections 2.3 and 3.1) and use them to time-step the discrete version of the viscous system of equations (10)-(11), with $\mu[\boldsymbol{e}]$ substituted by $\tilde{\mu}[\boldsymbol{e}_h]$, in accordance with the SSPRK-4 time stepping scheme.
    Update time: $t = t + \Delta t$
end while

a point of contrast with the related publication [38]: unlike the algorithm presented in Section 4 of that paper, the FC-SDNN use of the shifting procedure described in point (ii) in Section 3.2.1 is not limited to the training stage of the neural network, but is also performed at every time-step, as indicated in line 7 of Algorithm 1, as part of the evaluation of the smoothness classification operator. This strategy exploits the fact that Gibbs oscillations in the proxy variable can be detected on the shifted grid before spurious oscillations affect the solution vector grid function $\boldsymbol{e}_h$, and thus allows for up-front "corrective" viscosity assignments in that region—thus avoiding both, spurious oscillations, and necessary subsequent use of larger viscosity values and more frequent viscosity assignments. Experiments have shown that use of the shifting procedure during the time-stepping phase leads to reduced discontinuity smearing and sharper solution profiles.

4 Numerical results

This section presents results of application of the FC-SDNN method to a number of non-periodic test problems (with the exception of a periodic linear-advection problem, in Section 4.1.2, demonstrating the limited dispersion of the method), time-dependent boundary conditions, shock waves impinging on physical boundaries (including a corner point and a non rectangular domain), etc. All of the examples presented in this section resulted from runs on Matlab implementations of the various methods used. Computing times are not reported in this paper in view of the inefficiencies associated with the interpreter computer language used but, for reference, we note from [39] that, for the types of equations considered in this paper, the FC implementations can be quite competitive, in terms of computing time, for a given accuracy. Our experiments indicate, further, that the relative cost of application of smooth-viscosity operators of the type used in this paper do not vary substantially as the mesh is refined, and that for large enough discretizations the relative cost of the neural network algorithm becomes insignificant. We thus expect that, as in [39], efficient implementations of the proposed FC-SDNN algorithm will prove highly competitive for general configurations.

4.1 Linear advection

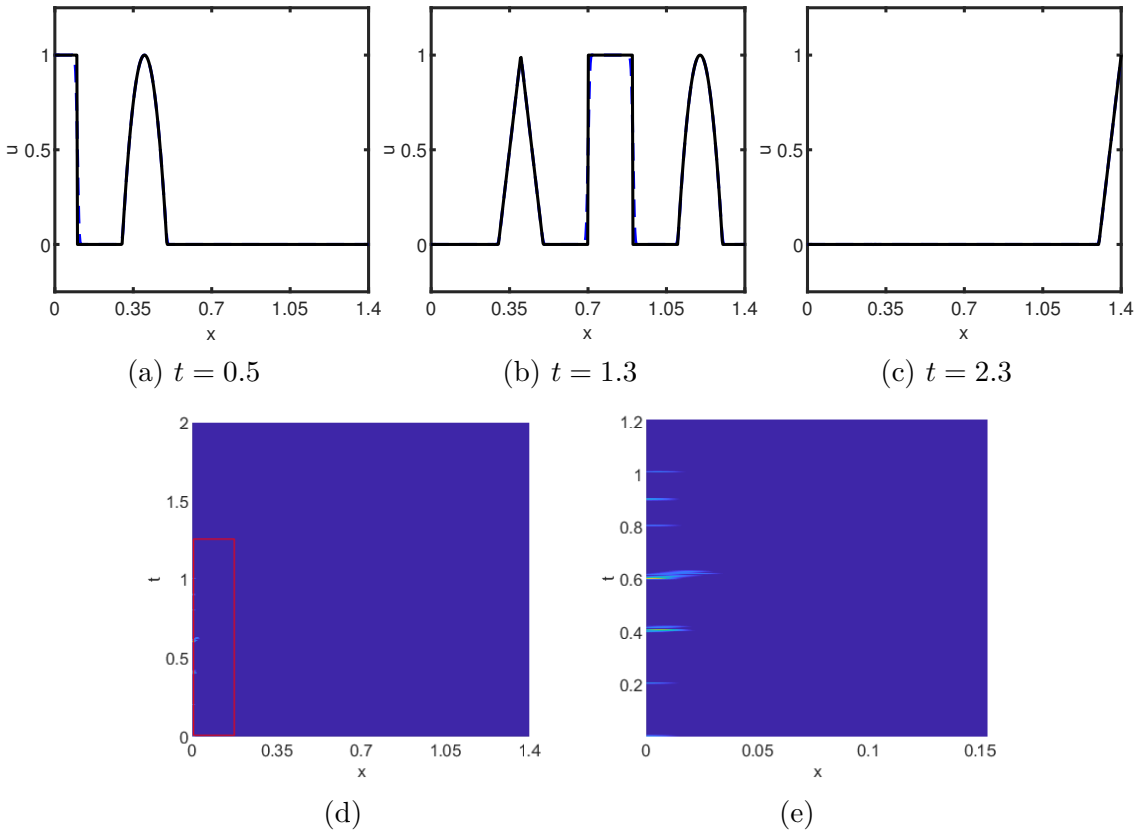
The simple 1D linear-advection results presented in this section demonstrate, in a simple context, two main benefits resulting from the proposed approach, namely 1) Effective handling of boundary conditions (Section 4.1.1); and 2) Essentially dispersionless character (Section 4.1.2). For the examples in this section FC expansions with $d = 5$ were used.

4.1.1 Boundary conditions

Figure 5 displays the FC-SDNN solution to the linear advection problem (2), in which three waves with various degrees of smoothness emanate from the left boundary and travel within the spatial interval $[0, 1.4]$, with an initial condition $u(x, 0) = 0$ and boundary condition at $x = 0$ given by

$$u(0,t) = \begin{cases} 100t(t-0.2) & \text{if } 0 < t < 0.2 \\ 1 & \text{if } 0.2 < t < 0.4 \\ 10(t-0.8) & \text{if } 0.8 < t < 0.9 \\ 1 - 10(t-0.9) & \text{if } 0.9 < t < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

At the outflow boundary $x = 1.4$, the solution was evolved in the same manner as the interior points, as befits an outflow boundary. The solution was computed up to time $T = 2.3$, using the adaptive time step given by (22), with CFL $= 2$. As shown in Figure 5, the waves travel within the domain matching the exact solution. The induction of waves and discontinuities at times $t = 0$, $t = 0.2$, $t = 0.4$, $t = 0.6$, $t = 0.8$, $t = 0.9$, $t = 1$ through the left boundary is automatically accompanied by the assignment of artificial viscosity on a small spatio-temporal area near that boundary. The FC-SDNN algorithm stops assigning viscosity shortly after these induction events, once the associated discontinuities are smeared, as illustrated in Figures 5d-e. Note that the waves exit the domain without producing undesired reflection artifacts around the physical exit boundary.



(a) $t = 0.5$ (b) $t = 1.3$ (c) $t = 2.3$

(d) (e)

Figure 5: Upper row: Solution of the non-periodic one dimensional linear advection problem at three different points in time, using $N = 500$ discretization points. Exact (black solid line), FC-SDNN (blue dashed line). Lower row: Time history of FC-SDNN artificial viscosity assignments. (d) Full space-time range, showing that zero viscosity values are assigned in most of the spatio-temporal domain, and a small rectangular region delimited by red sides, whose detail is shown on panel (e). (e) Zoomed-in range on the small spatio temporal region for which non-zero viscosities are assigned.

4.1.2 Limited dispersion

To demonstrate the limited dispersion inherent in the FC-SDNN algorithm we consider a problem of cyclic advection of a "bump" solution over a bounded 1D spatial domain—thus effectively simulating, in a bounded domain, propagation over arbitrarily extended spatial regions. To do this we

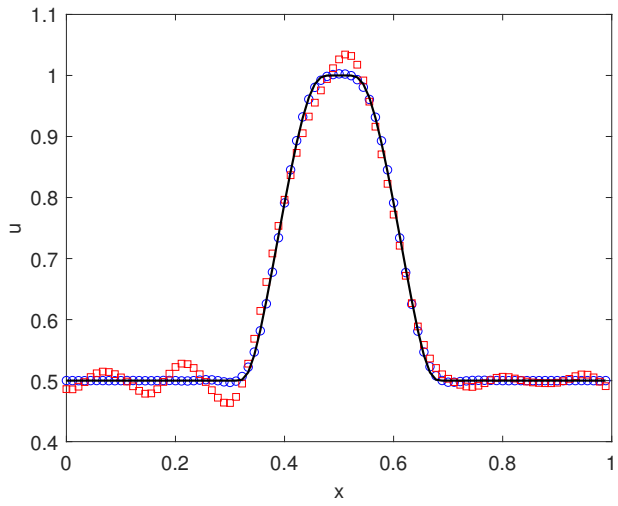


Figure 6: Numerical solutions to the periodic one-dimensional linear advection problem with $a = 1$ up to time $T = 500$: Exact solution (solid black line), fifth order FC-SDNN method (blue circles) and sixth order centered finite-difference scheme (red squares). Both numerical solutions were obtained using $N = 90$ discretization points. In view of its nearly dispersionless character, the FC-based solution remains significantly more accurate than its higher order finite-difference counterpart.

utilize the smooth cut-off "bump" function $\omega = \omega(x, q_1, q_2)$ $(q_1 \neq q_2)$ defined by

$$\omega(x, q_1, q_2) = \begin{cases} \exp(2\frac{e^{-1/\xi}}{\xi - 1}) & \text{if} \quad q_1 \leq |x| \leq q_2 \\ 1 & \text{if} \quad |x| \leq q_1 \\ 0 & \text{if} \quad |x| \geq q_2, \end{cases} \qquad \text{where} \quad \xi = \frac{|x| - q_1}{q_2 - q_1}. \tag{46}$$

For this example we solve the equation (2) with $a = 1$, starting from a smooth initial condition given by $u(x, 0) = \omega(x - 0.5, 0, 0.2)$, over the domain $[0, 1]$ under periodic boundary conditions. In order to enforce such periodic conditions, the FC differentiation scheme is adapted, by using the same precomputed matrices A_ℓ, A_r and Q (see Section 2.3) in conjunction with the "wrapping" procedure described in [1, Sec. 3.3]. Using the FC-SDNN algorithm of order $d = 5$ and SSPRK-4, the solution was evolved for five hundred periods, up to $T = 500$; the resulting $t = 500$ solution $u(x, 500)$ is displayed in Figure 6. For comparison this figure presents numerical results obtained by means of a 6-th order central finite-difference scheme (also with SSPRK-4 time stepping). The finite-difference method uses a constant time-step $\Delta t = 0.0034$, while the FC-SDNN uses the adaptive time step defined in (22) for which, in the present case, with $\mu = 0$ and $S = 1$, we have $\Delta t = 0.0036$. Clearly, the FC-SDNN solution matches the exact solution remarkably well even after very long times, showcasing the low dissipation and dispersion afforded by the FC-based approach. In contrast, the higher-order finite difference solution suffers from noticeable dispersion effects.

4.2 Burgers equation

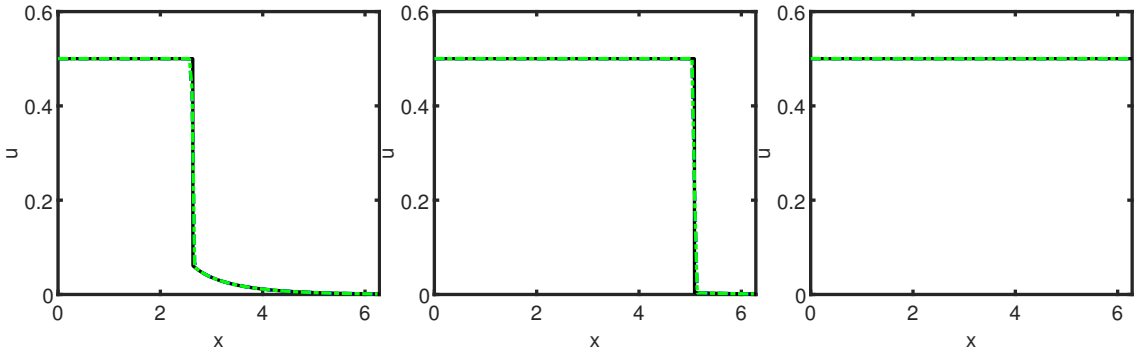
The 1D and 2D Burgers equation tests presented in this section demonstrate the FC-SDNN solver's performance for simple nonlinear non-periodic problems. In particular, the 2D example showcases the ability of the algorithm to handle multi-dimensional problems where shocks intersect domain boundaries (a topic that is also considered in Section 4.3.2 in the context of the Euler equations).

4.2.1 1D Burgers equation

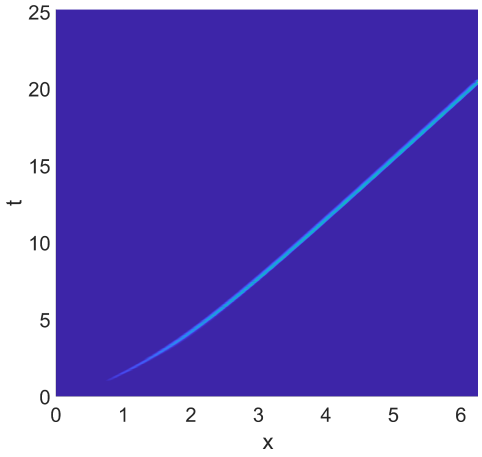
Propagating shock wave Figure 7 displays solutions to the 1D Burgers equation (3), where a shock forms from the sharp features in the initial condition

$$u_0(x) = \frac{1}{\exp(x - \frac{3}{20})[\tanh(10x - 3) + 1] - \tanh(10x - 3) + 1}. \tag{47}$$

Results of simulations produced by means of the FC-SDNN and FC-EV algorithms are presented in the figure. In both cases Dirichlet boundary conditions at the inflow boundary $x = 0$ were used, while the solution at the outflow boundary $x = 2\pi$ was evolved numerically in the same manner as the interior domain points. The FC-EV viscosity parameters defined in equations (15) and (16) were set to $c_{max} = 0.2$ and $c_E = 0.1$; cf. Table 2. As shown in the figure, the shock is sharply resolved by both algorithms, with no visible oscillations. In both cases the shock eventually exits the physical domain without any undesired reflections or numerical artifacts. For this example FC expansions with $d = 2$ were used. The reference solution (black) was computed on a 10000-point domain, with the FC-SDNN method.



(a) $t = 2\pi$ (b) $t = 5\pi$ (c) $t = 8\pi$



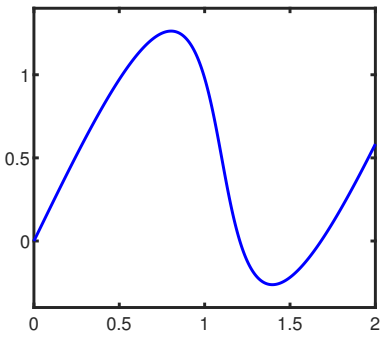
(d) Time history of FC-SDNN artificial viscosity.

Figure 7: Solutions to the non-periodic one dimensional Burgers equation produced by the FC-SDNN and FC-EV algorithms of order $d = 2$ at three different times t. Black solid line: finely resolved FC-SDNN ($N = 10,000$, for reference). Blue dashed line: FC-SDNN with $N = 500$. Green dot-dashed line: FC-EV with $N = 500$. Note that the $N = 500$ FC-SDNN and FC-EV solutions are virtually indistinguishable from each other in this case.

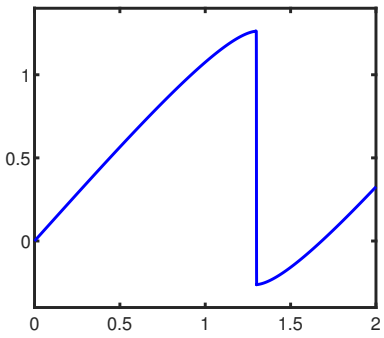
Convergence test To demonstrate the FC-SDNN accuracy for FC expansions of order $d = 5$ before and after the formation of a shock in an initially smooth solution, we consider the 1D Burgers equation (3) in the domain $[0, 2]$, with an initial condition given by

$$u_0(x) = \frac{1}{2}x + \sin(\pi x) \tag{48}$$

and with an identically vanishing Dirichlet boundary condition at $x = 0$. The solution at times $t = 0.3$ (before the formation of the shock) and $t = 0.8$ (after the formation of the shock) are displayed in Figures 8a and b. As demonstrated in Figure 9, a high order of convergence is obtained both in L^1 and L^2 norms before the time $t = \frac{1}{\pi - \frac{1}{2}}$ at which the shock forms. (An order 8.23 is reported in the figure for the pre-shock solution, which is higher than the FC order 5 used in this case. We attribute this discrepancy to the fact that the error is concentrated in the region near the point of highest gradient, where the solution is smooth, and not near the boundary, where the fifth-order FC error is smaller in magnitude than the error near the highest gradient point.) After the shock-formation time, convergence of orders h and $h^{\frac{1}{2}}$ in the L^1 and L^2 norms, respectively, is observed—as expected, in view of the order-1 solution errors that exist in a spatial region of order h around the shock. Figures 8a and b also show that a faster error decay and overall smaller L^1 and L^2 errors are observed in spatial regions away from the shock. Finally, we note that both before and after the formation of the shock, a selection of a higher order $d = 5$ for the FC algorithm yields noticeably smaller errors away from the shock regions than the lower order $d = 2$, thus underscoring the benefit of the use of a higher order scheme, even in the context of an order h (resp. $h^{\frac{1}{2}}$) global accuracy in L^1 norm (resp. L^2 norm).



(a) $t = 0.2$



(b) $t = 0.6$

Figure 8: Solutions to the non-periodic one dimensional Burgers equation produced by the FC-SDNN algorithm of order $d = 5$ at two different times t, with $N = 6400$.

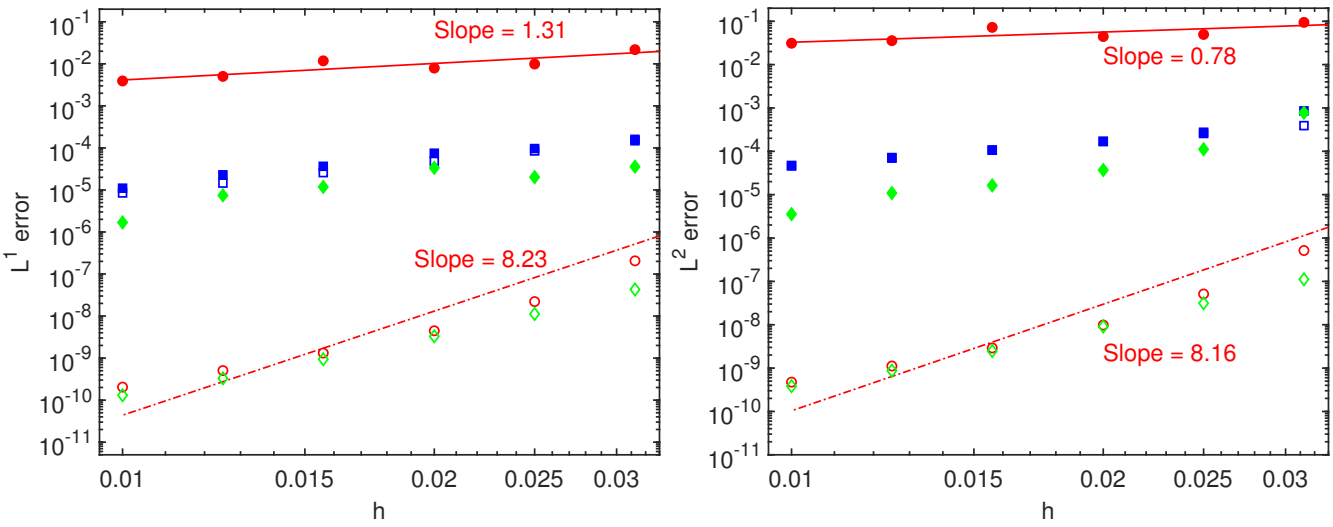


Figure 9: Convergence of the FC-SDNN solution for the Burgers 1D test in the L^1 and L^2 norms (left and right panels, respectively), before ($t = 0.1$) and after ($t = 0.6$) the shock forms (empty and full symbols, respectively). The reference data was obtained by solving the exact implicit form of the solution [46, p.99] by Newton's method, using a fine-grid numerical solution as an initial guess. Full and empty red dots: errors at $t = 0.6$ and $t = 0.1$, respectively, on the complete domain, using FC order $d = 5$. Full and empty blue squares: errors at $t = 0.6$ and $t = 0.1$, respectively, away from highest gradient point ($x \in [0, 1.1] \cup [1.7, 2]$ and $x \in [0, 1] \cup [1.6, 2]$, respectively), using FC order $d = 2$. Full and empty green diamonds: errors at $t = 0.6$ and $t = 0.1$, respectively, away from the highest gradient point ($x \in [0, 1.1] \cup [1.7, 2]$ and $x \in [0, 1] \cup [1.6, 2]$, respectively), using FC order $d = 5$.

4.2.2 2D Burgers equation

To demonstrate the solver's performance and correct handling of shock-boundary interactions for 2D problems, we consider the 2D Burgers scalar equation (4) on the domain $\mathcal{D} = [0, 1] \times [0, 1]$, with an initial condition given by the function

$$u_0(x) = \begin{cases} -1 & \text{if} \quad x \in [0.5, 1] \text{ and } y \in [0.5, 1] \\ -0.2 & \text{if} \quad x \in [0, 0.5] \text{ and } y \in [0.5, 1] \\ 0.5 & \text{if} \quad x \in [0, 0.5] \text{ and } y \in [0, 0.5] \\ 0.8 & \text{if} \quad x \in [0.5, 1] \text{ and } y \in [0, 0.5], \end{cases} \tag{49}$$

and with vanishing normal derivatives at the boundary. This problem admits an explicit solution (displayed in Figure 10b at time $t = 0.25$) which includes three shock waves and a rarefaction wave, all of which travel orthogonally to various straight boundary segments. Figures 10c, 10d, and 10e present the corresponding numerical solutions produced by the FC-SDNN algorithm at time $t = 0.25$ resulting from use of various spatial discretizations and with adaptive time step given by (22) with CFL = 2. Sharply resolved shock waves are clearly visible for the finer discretizations, as is the rarefaction wave in the lower part of the figure. The viscosity assignments, which are sharply concentrated near shock positions as the mesh is refined, suffice to avert the appearance of spurious oscillations. It is interesting to note that non-vanishing viscosity values are only assigned around shock discontinuities. (The SDNN algorithm assigns zero viscosity to the rarefaction wave for all time—as a result of the discontinuity-smearing introduced by the algorithm on the initial

condition for the velocity, described in Section 3.4.2, which the FC-SDNN then preserves for all times in regions near the rarefaction wave on account of the resulting smoothness of the numerical solution in such regions). For this example FC expansions with $d = 5$ were used. As shown in Figure 11, and per the discussion in Section 4.2.1, the expected convergence of orders h and $h^{\frac{1}{2}}$ in the L^1 and L^2-norms respectively are obtained.

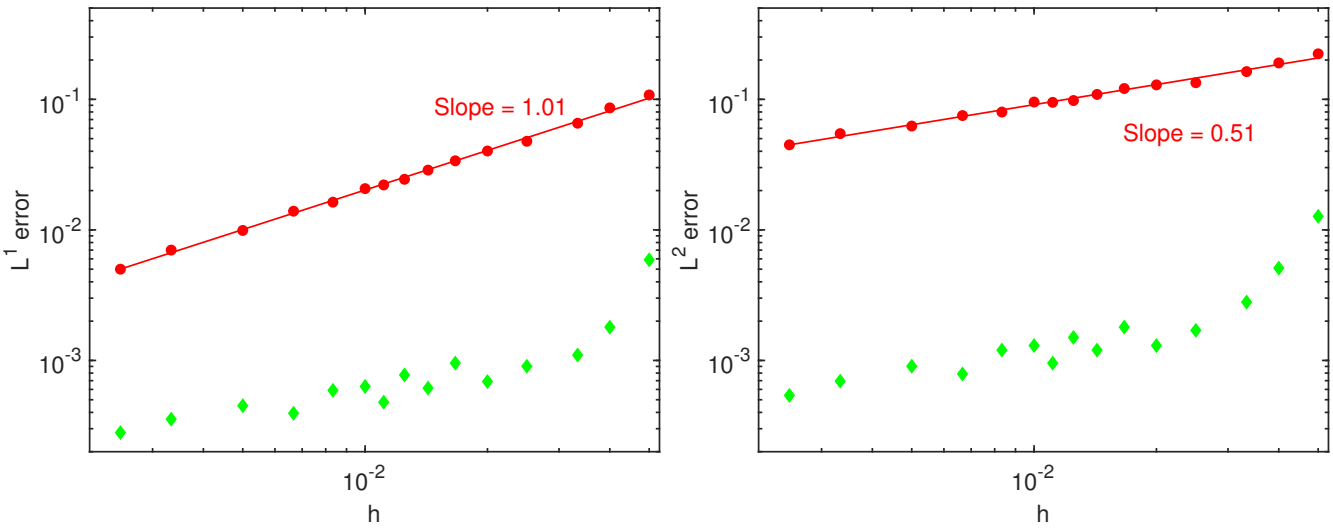


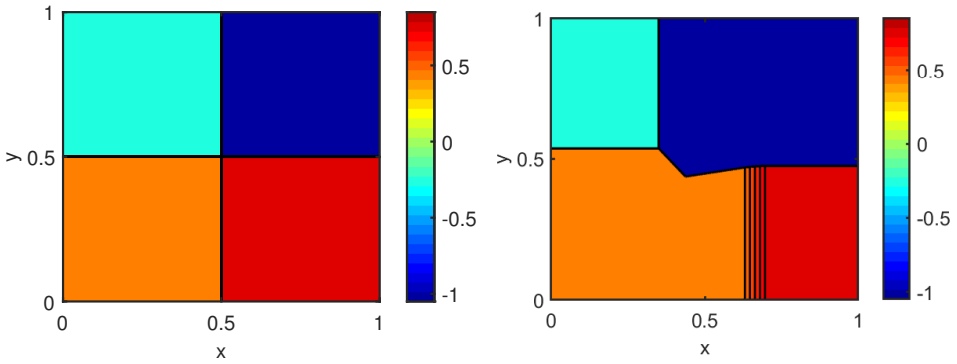
Figure 11: Convergence of the FC-SDNN solution for the Burgers 2D test at $t = 0.25$ in the L^1 and L^2 norms (left and right panels, respectively), using FC order $d = 5$. Red dots: errors on the complete domain. Green diamonds: errors away from the shocks ($(x, y) \in ([0, 0.4] \times [0, 0.35]) \cup ([0, 0.25] \times [0, 0.6]) \cup ([0.6, 1] \times [0.6, 1])$.)

4.3 1D and 2D Euler systems

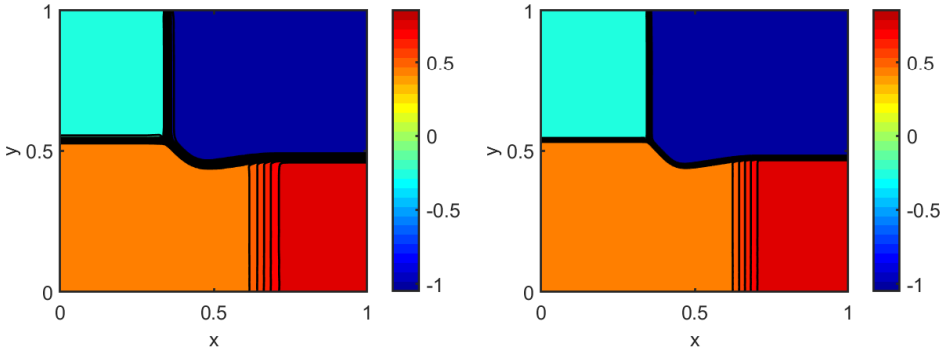
This section presents a range of 1D and 2D test cases for the Euler system demonstrating the FC-SDNN algorithm's performance in the context of a nonlinear systems of equations. The test cases include well-known 2D arrangements, including the shock-vortex interaction example [40], 2D Riemann problem flow [25], Mach 3 forward facing step [47], and Double Mach reflection [47]. In particular the results illustrate the algorithm's ability to handle contact discontinuities and shock-shock interactions as well as shock reflection and propagation along physical and computational boundaries.

4.3.1 1D Euler problems

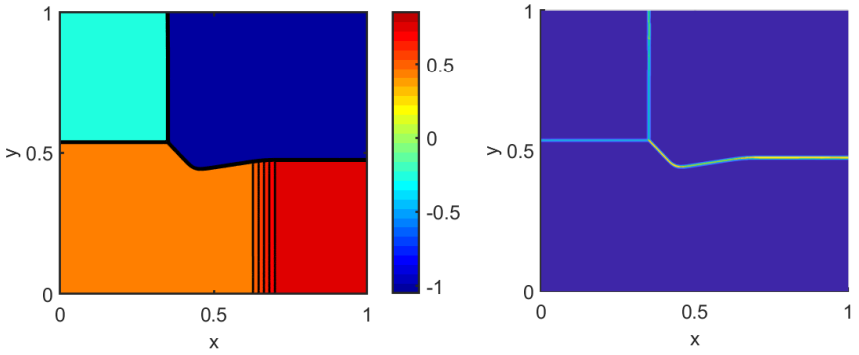
The 1D shock-tube tests considered in this section demonstrate the solver's ability to capture not only shock-wave discontinuities (that also occur in the Burgers test examples considered in Section 4.2) but also contact discontinuities. Fortunately, in view of the localized spectral filtering strategy used for the initial-data (Section 3.4.2), the algorithm completely avoids the use of artificial viscosity around contact discontinuities, and thus leads to excellent resolution of these important flow features. This is demonstrated in a variety of well known test cases, including the Sod [42], Lax [24], Shu-Osher [41] and Blast Wave [35] problems, with flows going from left to right—so that the left boundary point (resp. right boundary point) is the inflow (resp. outflow) boundary. Following [16, Sec. 19], inflow (resp. outflow) boundary condition were enforced at the inflow boundaries (resp. outflow boundaries) by setting ρ and u (resp p) identically equal, for all time, to



(a) Initial condition (b) Exact solution



(c) $N = 200$ (d) $N = 400$



(e) $N = 1000$ (f) Art. visc. ($t = 0.25$, $N = 1000$)

.

Figure 10: Fifth-order ($d = 5$) FC-SDNN numerical solution to the 2D Burgers equation with initial condition displayed on the upper-left panel, whose exact solution at $t = 0.25$ is displayed on the upper-right panel. The middle and left-lower panels display the FC-SDNN numerical solutions at $t = 0.25$ obtained by using $N \times N$ spatial grids with three different values of N, as indicated in each panel. The FC-SDNN numerical viscosity at $t = 0.25$ for the case $N = 1000$ is presented in the lower-right panel.

the corresponding boundary values of these quantities at the initial time $t = 0$. For the examples in this section FC expansions with $d = 5$ were utilized. For each one of these problems, the FC-EV solutions are provided for reference; the selected EV parameters are provided in Table 2. For each test case and each algorithm (FC-SDNN and FC-EV), the value of the constant CFL in (22), which is included as part of each description, was selected so as to obtain the largest time-step Δt which preserves stability.

Problem	Burgers 1D	Sod	Lax	Shu-Osher	Blast Wave
c_{max}	0.2	0.1	0.15	0.85	1
c_E	0.1	15	20	10	0.05

Table 2: FC-EV parameters for the 1D Burgers and 1D Euler problems.

Sod problem. We consider a Sod shock-tube problem for the 1D Euler equations (5) on the interval $[-4, 5]$ with initial conditions

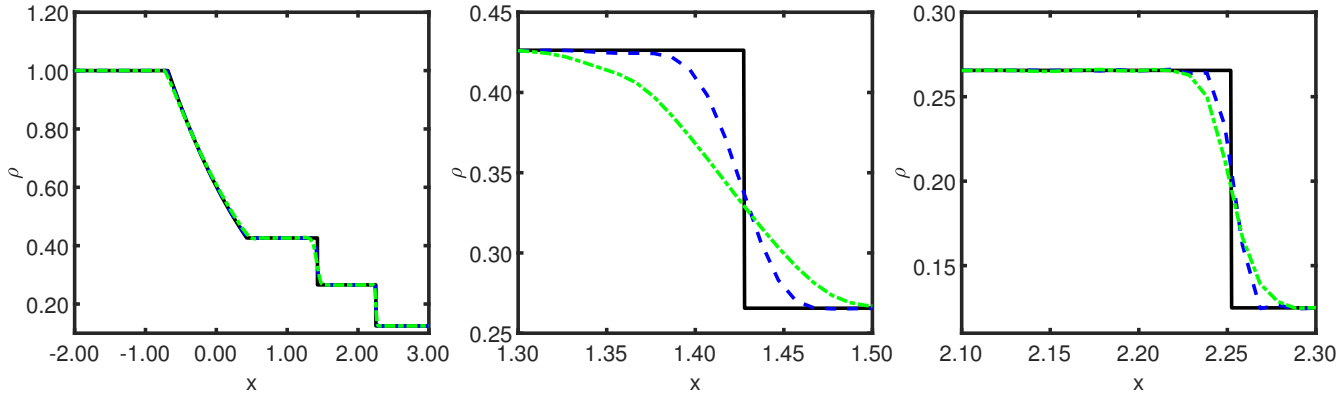
$$(\rho, u, p) = \begin{cases} (1, 0, 1) & \text{if} \quad x < 0.5 \\ (0.125, 0, 0.1) & \text{if} \quad x > 0.5, \end{cases}$$

a setup that gives rise (from right to left) to a right-moving shock wave, a contact discontinuity and a rarefaction wave (left images in Figure 12(a) and (b)). The solution was computed up to time $T = 2$. The results presented in Figure 12 show well resolved shocks (upper and middle right) and contact discontinuities (center images in Figure 12(a) and (b)), with no visible Gibbs oscillations in any case. The FC-SDNN and FC-EV solvers (for which adaptive time steps (22) were used with CFL = 3 and CFL = 2, respectively) demonstrate a similar resolution in a vicinity of the shock, but the FC-SDNN method provides a much sharper resolution of the contact-discontinuity. As shown in Figure (12c), after a short time the FC-SDNN algorithm does not assign artificial viscosity in a vicinity of the contact discontinuity, leading to the significantly more accurate resolution observed for this flow feature.

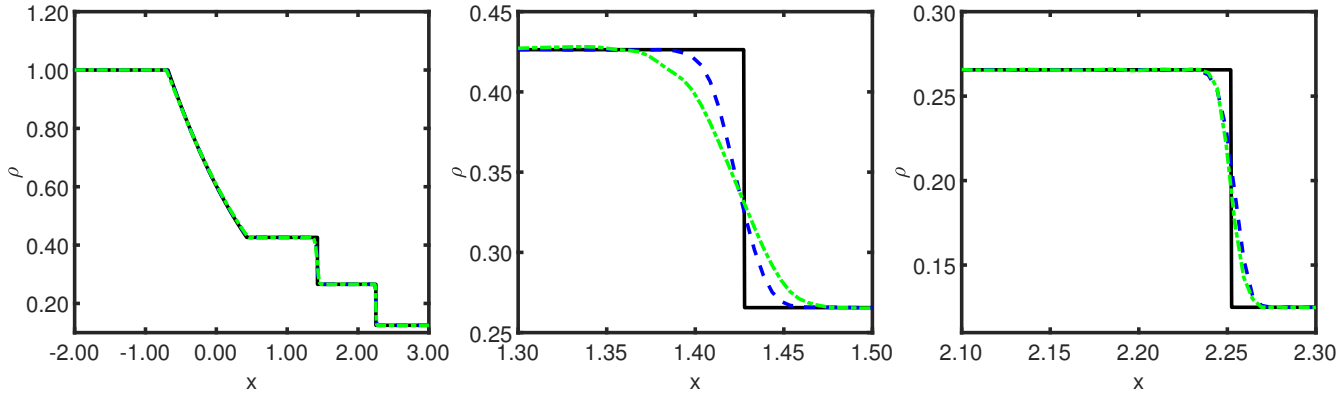
Lax problem. We consider a Lax problem on the interval $[-5, 5]$, with initial condition

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

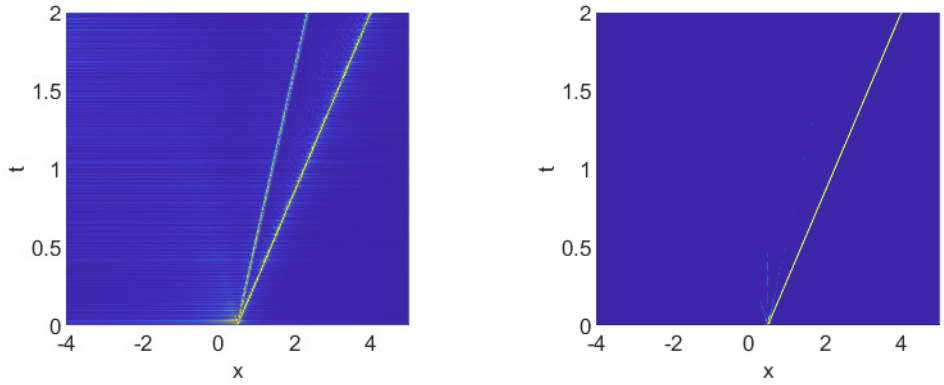
which results in a combination (from right to left) of a shock wave, a contact discontinuity and a rarefaction wave (left images in Figure 13(a) and (b)). The solution was computed up to time $T = 1.3$. The results are presented in Figure 13, which shows well resolved shocks (upper and middle right) without detectable Gibbs oscillations. The viscosity time history displayed in Figure 13c shows that the FC-SDNN method only assigns artificial viscosity in the vicinity of the shock discontinuity but, as discussed above, not around the contact discontinuity, leading to a sharper resolution by the FC-SDNN method in this region (center images in Figure 13(a) and (b)). The FC-EV and FC-SDNN algorithms, (for which adaptive time steps (22) were used with CFL = 2 and CFL = 4, respectively) demonstrate a similar shock resolution, but the latter approach is significantly more accurate around the contact discontinuity.



(a) N = 500

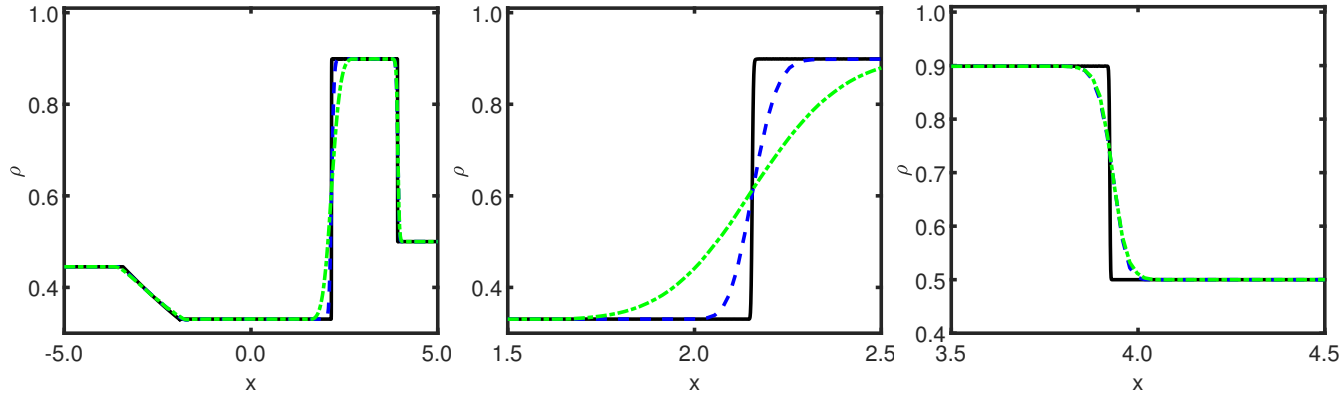


(b) N = 1000

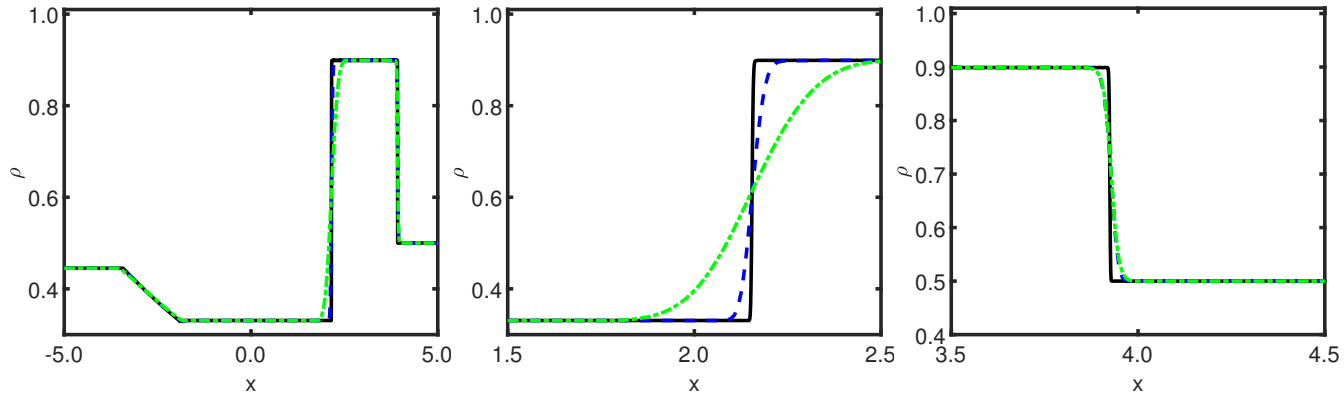


(c) Time history of assigned artificial viscosity for N = 1000, for FC-EV (left) and FC-SDNN (right).

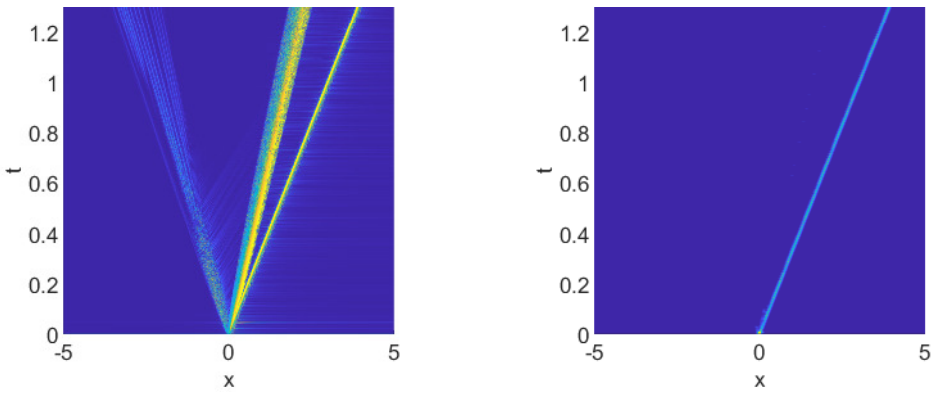
Figure 12: Solutions to the Sod problem produced by the FC-SDNN and FC-EV algorithms of order $d = 5$ at $t = 0.2$. Exact solution: solid black line. FC-SDNN solution: Blue dashed-line. FC-EV solution: Green dot-dashed line. Numbers of discretization points: $N = 500$ in the upper panels and $N = 1000$ in the middle panels. Bottom panels: artificial viscosity assignments.



(a) N = 500



(b) N = 1000



(c) Time history of assigned artificial viscosity for N = 1000, for FC-EV (left) and FC-SDNN (right).

Figure 13: Solutions to the Lax problem produced by the FC-SDNN and FC-EV algorithms of order $d = 5$ at $t = 1.3$. Exact solution: solid black line. FC-SDNN solution: Blue dashed-line. FC-EV solution: Green dot-dashed line. Numbers of discretization points: $N = 500$ in the upper panels and $N = 1000$ in the middle panels. Bottom panels: artificial viscosity assignments.

Shu-Osher problem. We consider the Shu-Osher shock-entropy problem on the interval $[-5, 5]$, with initial condition given by

$$(\rho, u, p) = \begin{cases} (3.857143, 2.6929369, 10.33333) & \text{if} \quad x < -4 \\ (1 + 0.2\sin(5x), 0, 1) & \text{if} \quad x > -4. \end{cases} \tag{50}$$

The solution is computed up to time $T = 1.8$. In this problem, a shock wave encounters an oscillatory smooth wavetrain. This test highlights the FC-SDNN solver's low dissipation, as artificial viscosity is only assigned in the vicinity of the right-traveling shock as long as the waves remain smooth, allowing for an accurate representation of the smooth features. In particular, Figure (14c) shows that the support of the FC-SDNN artificial viscosity is much more narrowly confined around the shock position than the artificial viscosity resulting in the FC-EV approach. As a result, and as illustrated in the right images in Figure 14(a) and (b), the FC-SDNN method provides a more accurate resolution in the acoustic region (behind the main, rightmost, shock). For this problem, the FC-EV and FC-SDNN algorithms, (for which adaptive time steps (22) were used with CFL = 3 and CFL = 4, respectively).

Blast Wave problem. Finally, we consider the Blast Wave problem as presented in [35], on the interval $[0, 1]$, with initial conditions given by

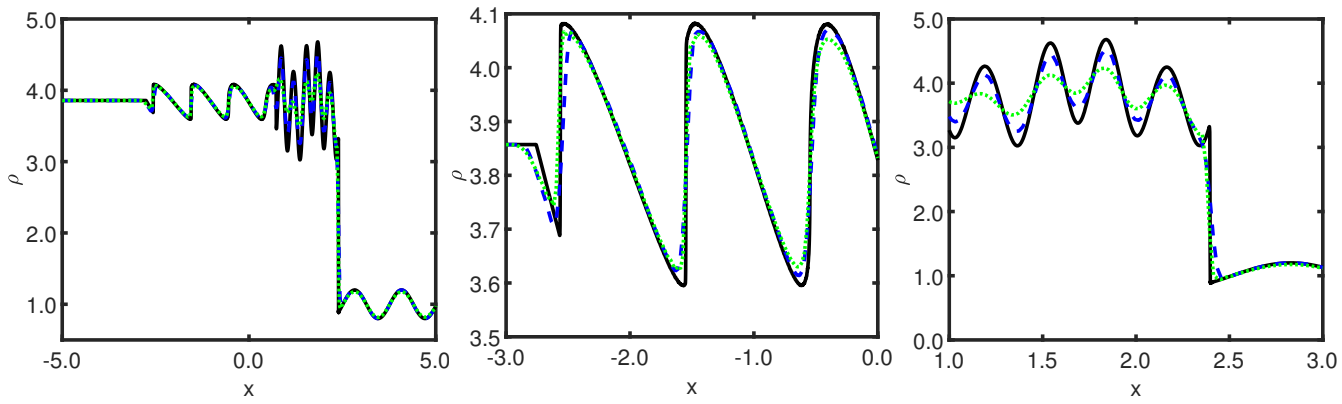
$$(\rho, u, p) = \begin{cases} (1, 0, 1000) & \text{if} \quad x < 0.5 \\ (1, 0, 0.01) & \text{if} \quad x > 0.5, \end{cases} \tag{51}$$

up to time $T = 0.012$. This setup is similar to the one considered in the Sod problem, but with a much stronger right-moving shock. In order to avoid unphysical oscillations at the boundaries, which could result from the presence of the strong shock, the value of the operator τ is set to 1 on the leftmost and rightmost nine points in the domain, thus effectively assigning a small amount of viscosity at the boundaries at every time step of the simulation. The FC-EV and FC-SDNN algorithms, (for which adaptive time steps (22) were used with CFL = 2 and CFL = 2, respectively) provide a sharp resolution of the shock, as shown in Figure 15. As the mesh is refined the contact discontinuity is resolved more sharply by the former method which, as in the previous examples, does not assign viscosity around such features.

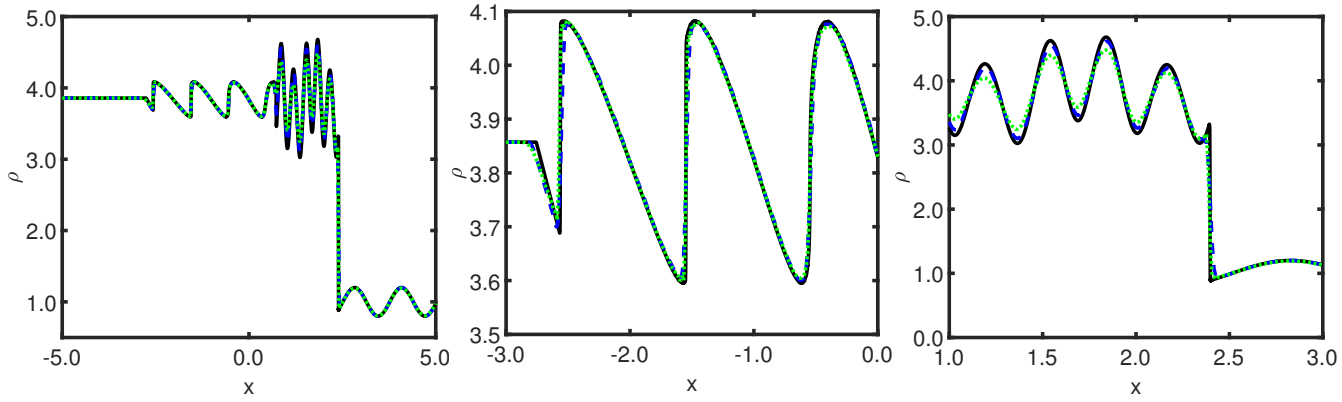
Remark 6. As shown in the left panel of Figure 15, the FC-EV method assigns a non-vanishing artificial viscosity everywhere ahead of the shock for this example. This undesirable feature results from the term $\frac{\partial \eta}{\partial t}$ in the entropy residual (13) which, in view of the relation $\eta(\boldsymbol{e}) = \frac{\rho}{\gamma - 1} \log(p/\rho^\gamma)$ given in Section 2.2.2, contains an additive contribution of the form $\frac{\rho}{p} \frac{\partial p}{\partial t}$—which is large ahead of the shock in view of the small pressure value in that area, together with the numerically non-vanishing values of $\frac{\partial p}{\partial t}$ that result from the global character of the Fourier expansions used and the non-smooth global viscosity profiles that underly the Entropy Viscosity method.

4.3.2 2D Euler problems

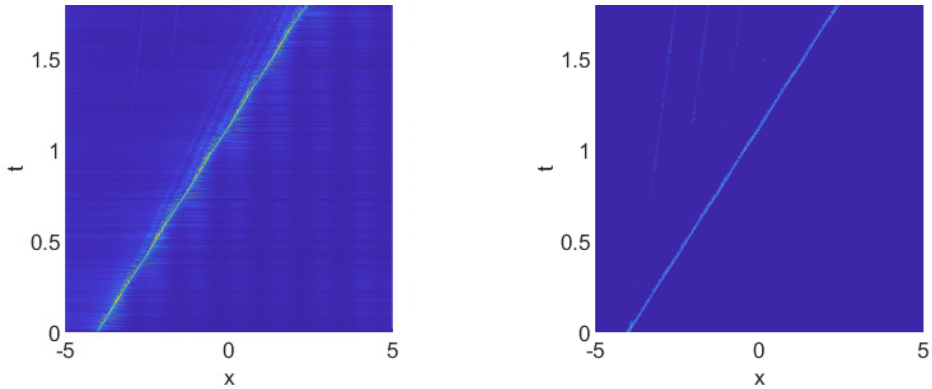
The test cases considered in this section showcase the FC-SDNN method's ability to handle complex shock-shock, as well as shock-boundary interactions, including shocks moving orthogonally to the



(a) N = 500

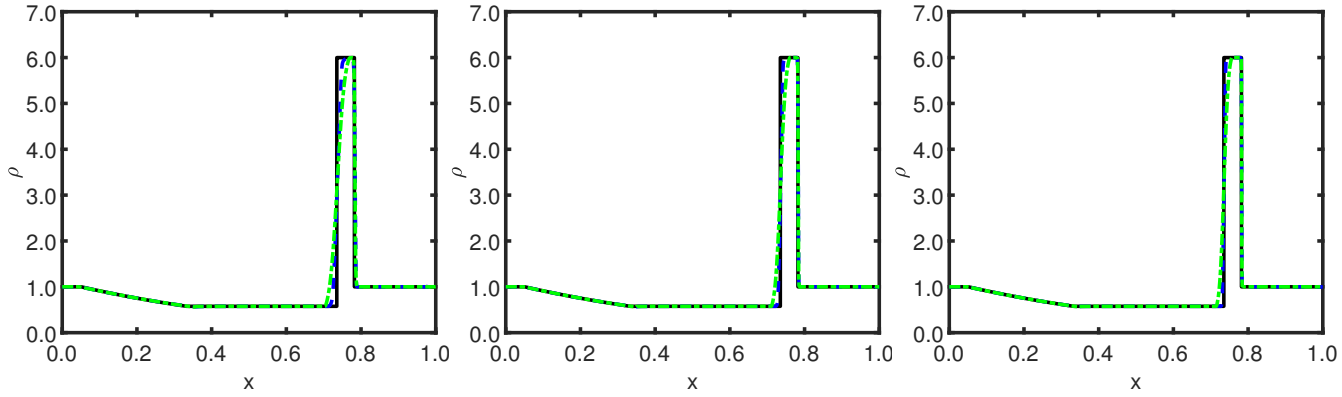


(b) N = 1000

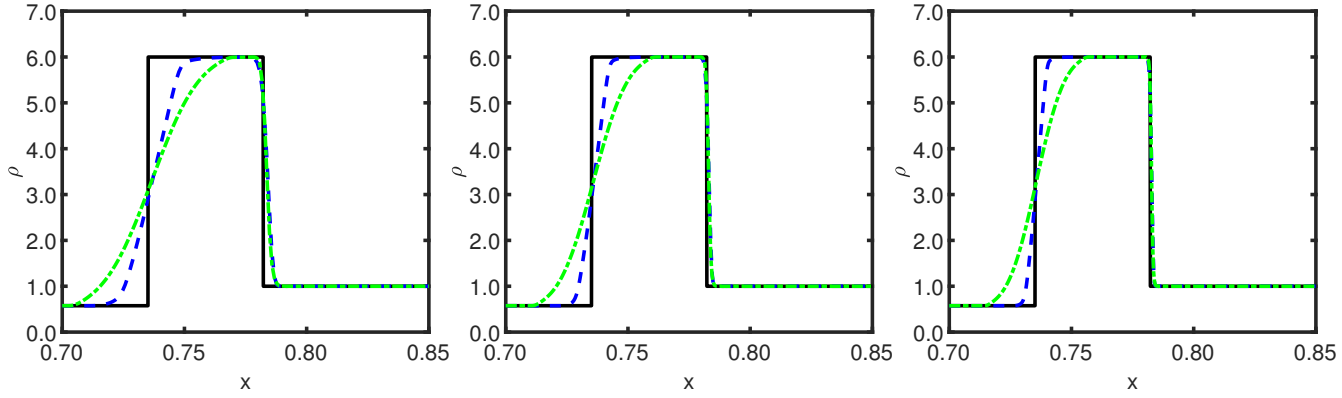


(c) Time history of assigned artificial viscosity for N = 1000, for FC-EV (left) and FC-SDNN (right).

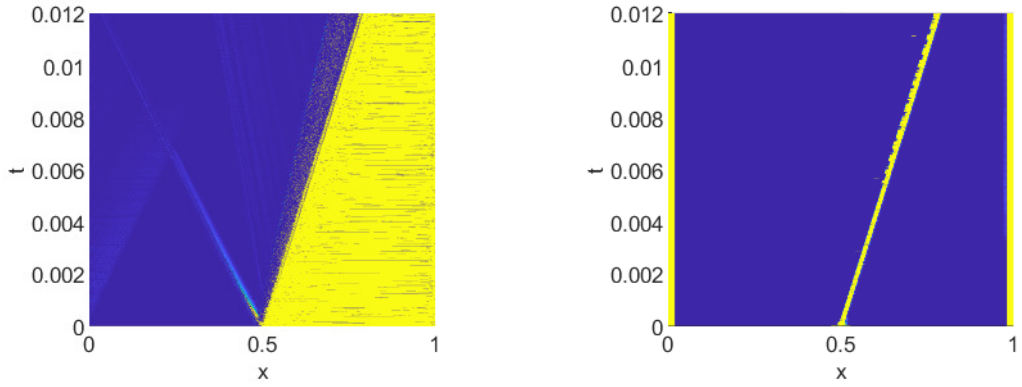
Figure 14: Solutions to the Shu-Osher problem produced by the FC-SDNN and FC-EV algorithms of order $d = 5$ at $t = 1.8$. Solid black line: finely resolved FC-SDNN ($N = 10,000$, for reference). Blue dashed line: FC-SDNN with $N = 500$ and $N = 1000$ (middle panels). Green dot-dashed line: FC-EV with $N = 500$ (upper panels) and $N = 1000$ (middle panels). Bottom panels: artificial viscosity assignments.



(a) "wide"

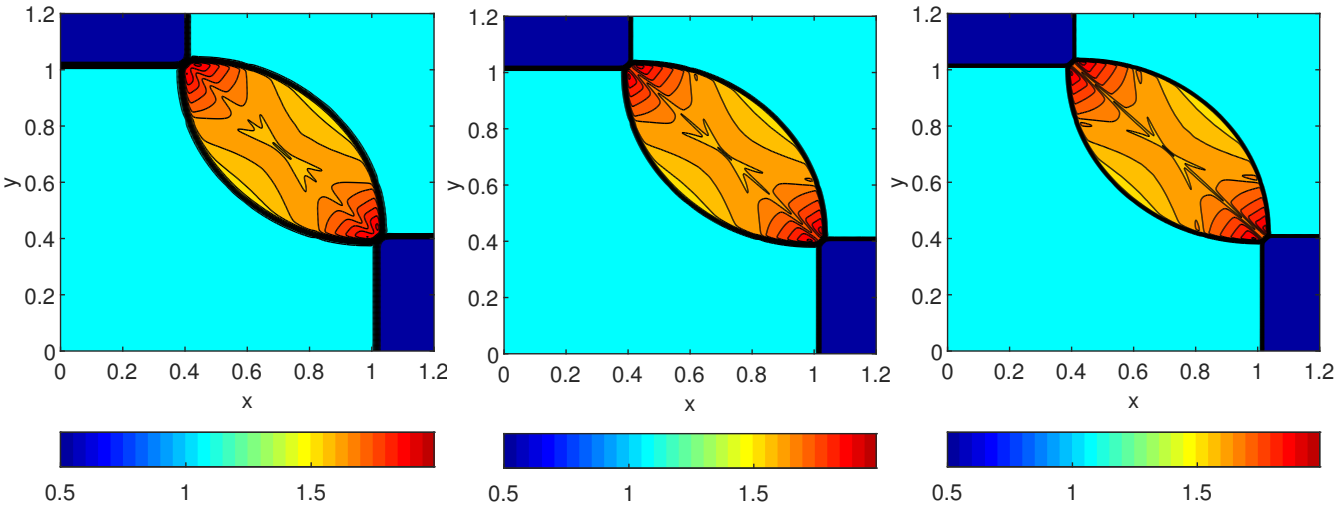


(b) "zoom"



(c) Time history of assigned artificial viscosity for $N = 1000$, for FC-EV (left) and FC-SDNN (right).

Figure 15: Solutions to the Blast wave problem produced by the FC-SDNN and FC-EV algorithms of order $d = 5$ at $t = 0.012$. Solid black line: exact solution. Blue dashed line: FC-SDNN with $N = 1000$ (upper- and middle-left panels), $N = 2000$ (upper- and middle-center panels), and $N = 3000$ (upper- and middle-right panels). Green dot-dashed line: FC-EV with $N = 1000$ (left panels), $N = 2000$ (center panels), and $N = 3000$ (right panels). Bottom panels: artificial viscosity assignments.



(a) N = 400 (b) N = 600 (c) N = 800

Figure 16: Second-order ($d = 2$) FC-SDNN numerical solution to the Euler 2D Riemann problem considered in Section 4.3.2, at $t = 0.25$, obtained by using a spatial discretization containing $N \times N$ grid points with three different values of N, as indicated in each panel. For each discretization, the solution is represented using thirty equispaced contours between $\rho = 0.5$ and $\rho = 1.99$.

boundaries as in the Riemann 2D and the Shock vortex problems, or moving obliquely to the boundary of the domain, after reflecting on a solid wedge, in the Double Mach reflection problem, or reflecting multiple times on the solid walls of a wind tunnel with a step, in the Mach 3 forward facing step problem. For the examples in this section FC expansions with $d = 2$ were used. As a result the FC method can provide significantly improved accuracy over other approaches of the same or even higher accuracy orders. In all cases the solutions obtained are in agreement with solutions obtained previously by various methods [14, 25, 30, 47]. As indicated in the introduction, the proposed approach leads to smooth flows away from shocks, as evidenced by correspondingly smooth level set lines for the various flow quantities, in contrast with corresponding results provided by previous methods.

Riemann4 problem (Riemann problem, configuration 4 in [25]). We consider a Riemann problem configuration on the domain $[0, 1.2] \times [0, 1.2]$, with initial conditions given by

$$(\rho, u, v, p) = \begin{cases} (1.1, 0, 0, 1.1) & \text{if} \quad x \in [0.6, 1.2] \text{ and } y \in [0.6, 1.2] \\ (0.5065, 0.8939, 0, 0.35) & \text{if} \quad x \in [0, 0.6) \text{ and } y \in [0.6, 1.2] \\ (1.1, 0.8939, 0, 0.35) & \text{if} \quad x \in [0, 0.6) \text{ and } y \in [0, 0.6) \\ (0.5065, 0, 0.8939, 0.35) & \text{if} \quad x \in [0.6, 1.2] \text{ and } y \in [0, 0.6), \end{cases} \tag{52}$$

and with vanishing normal derivatives for all variables on the boundary, at all times. The solution is computed up to time $T = 0.25$ by means of the FC-SDNN approach. The initial setting induces four interacting shock waves, all of which travel orthogonally to the straight segments of the domain boundary. The results, presented in Figure 16, show a sharpening of the shocks as the mesh is refined and an absence of spurious oscillations in all cases. As shown on the left image in Figure 17, the FC-SDNN viscosity assignments are sharply concentrated near the shock positions.

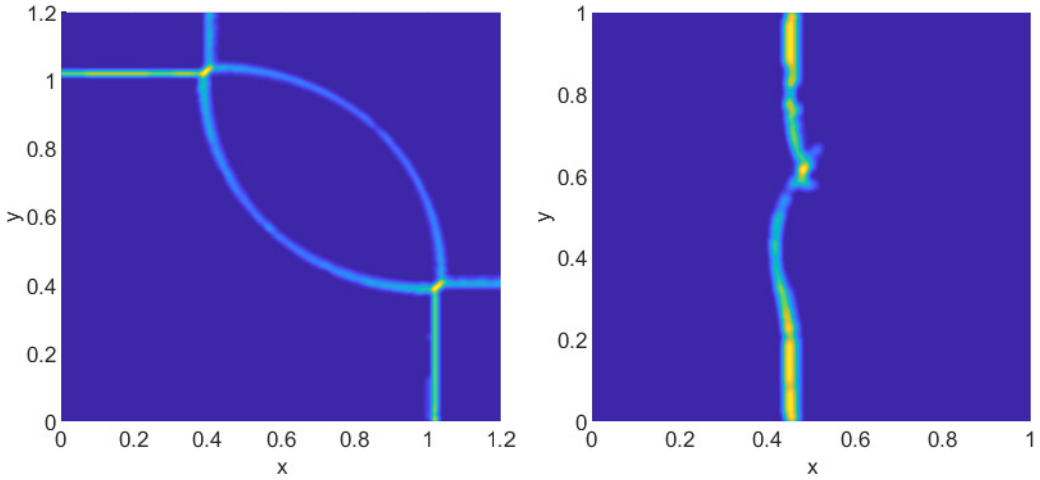


Figure 17: Artificial viscosity profiles for the Euler 2D Riemann problem considered in Figure 16 at $t = 0.25$ (left) and for the Shock vortex problem considered in Figure 18 at $t = 0.35$ (right).

Shock vortex problem [40]. We next consider a "shock-vortex" problem in the domain $[0, 1] \times [0, 1]$, in which a shock wave collides with an isentropic vortex. The initial conditions are given by

$$(\rho, u, v, p) = \begin{cases} (\rho_L + \tilde{\rho}, u_L + \tilde{u}, v_L + \tilde{v}, p_L + \tilde{p}) & \text{if} \quad x \in [0, 0.5) \\ (\rho_R, u_R, v_R, p_R) & \text{if} \quad x \in [0.5, 1] \end{cases} \tag{53}$$

where the left state equals the combination of the unperturbed left-state $(\rho_L, u_L, v_L, p_L) = (1, \sqrt{\gamma}, 0, 1)$ in the shock wave with the isentropic vortex

$$\tilde{u} = \frac{x - x_c}{r_c}\Phi(r), \quad \tilde{v} = -\frac{x - x_c}{r_c}\Phi(r), \quad \frac{\gamma - 1}{4\zeta\gamma}\Phi(r)^2 = \frac{p_L}{\rho_L} - \frac{p_L + \tilde{p}}{\rho_L + \tilde{\rho}}, \quad p_L + \tilde{p} = (\rho_L + \tilde{\rho})^\gamma, \tag{54}$$

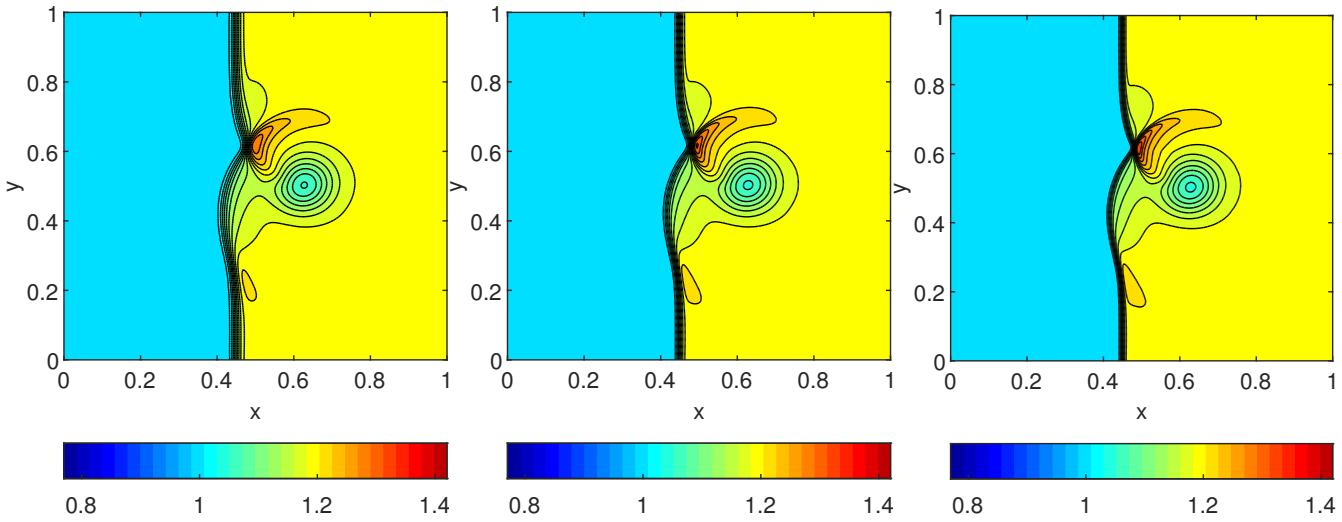
centered at $(x_c, y_c) = (0.25, 0.5)$ (where $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$, $\Phi(r) = \epsilon e^{\zeta(1-(r/r_c)^2)}$). As in previous references for this example we use the vortex parameter values $r_c = 0.05$, $\zeta = 0.204$, and $\epsilon = 0.3$. The initial right state is given by

$$\rho_R = \rho_L \frac{(\gamma + 1)p_R + \gamma - 1}{(\gamma - 1)p_R + \gamma + 1}, \quad u_R = \sqrt{\gamma} + \sqrt{2}\frac{1 - p_R}{\sqrt{\gamma - 1 + p_R(\gamma - 1)}}, \quad v_R = 0, \quad p_R = 1.3.$$

Vanishing normal derivatives for all variables were imposed on the domain boundary at all times.

The solution was obtained up to time $T = 0.35$, for which the vortex has completely crossed the shock. The solutions displayed in Figure 18 demonstrate the convergence of the method as the spatial and temporal discretizations are refined: shocks become sharper with each mesh refinement, while the vortex features remain smooth after the collision with the shock wave—a property that other solvers do not enjoy, and which provides an indicator of the quality of the solution. The right image in Figure 17 shows that, as in the previous examples, the support of the artificial viscosity imposed by the SDNN algorithm is narrowly focused in a vicinity of the shock.

Mach 3 forward facing step [47]. We now consider a "Mach 3 forward facing step problem" on the domain $([0, 0.6] \times [0, 1]) \cup ([0.6, 3] \times [0.2, 1])$, in which a uniform Mach 3 flow streams



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

Figure 18: Second-order ($d = 2$) FC-SDNN numerical solution to the Euler 2D Shock-vortex problem considered in Section 4.3.2, at $t = 0.35$, obtained by using a spatial discretization containing $N \times N$ grid points with three different values of N, as indicated in each panel. For each discretization, the solution is represented using thirty equispaced contours between $\rho = 0.77$ and $\rho = 1.42$.

through a wind tunnel with a forward facing step, of 0.2 units in height, located at $x = 0.6$. The initial condition is given by

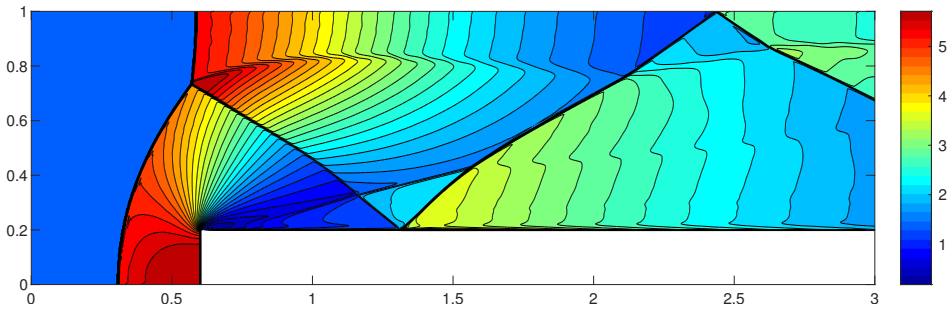
$$(\rho, u, v, p) = \begin{cases} (1.4, 3, 0, 1) & \text{if} \quad (x, y) \in ([0, 0.6) \times [0, 1]) \cup ([0.6, 3] \times [0.2, 1]) \\ (1.4, 0, 0, 1) & \text{if} \quad (x, y) \in \{0.6\} \times [0, 0.2], \end{cases} \tag{55}$$

and the solution is computed up to time $T = 4$. An inflow condition is imposed at the left boundary at all times which coincides with the initial values on that boundary, while the equations are evolved at the outflow right boundary. Reflecting boundary conditions (zero normal velocity) are applied at all the other boundaries. (Following [14, 47], no boundary condition is enforced at the node located at the step corner.) The simulation was performed using the adaptive time step (22) with CFL = 1. The density solution is displayed in Figure 19.

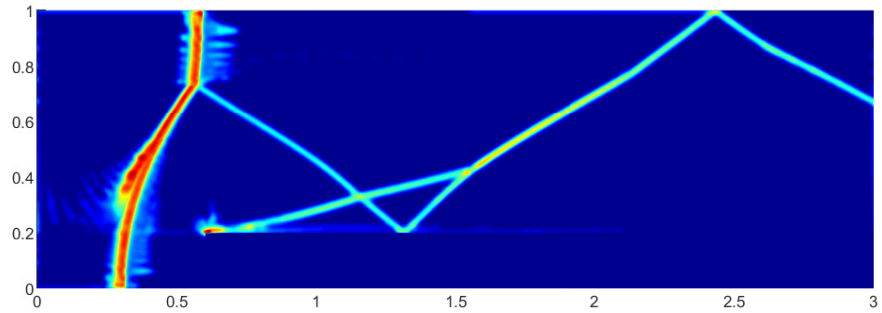
Double Mach reflection [47] We finally consider the "Double Mach reflection problem" on the domain $[0, 4] \times [0, 1]$. This problem contains a reflective wall located on the $x \geq x_r$ part of the bottom boundary $y = 0$ (here we take $x_r = 1/6$), upon which there impinges an incoming shock wave forming a $\theta = \pi/3$ angle with the positive x-axis. The initial condition is given by

$$\mathbf{e}(x, y, 0) = (\rho, \rho u, \rho v, E) = \begin{cases} (8, 57.1597, -33.0012, 563.544) & \text{if} \quad 0 \leq x \leq x_r + \frac{y}{\sin(\theta)} \\ (1.4, 0, 0, 2.5) & \text{if} \quad x_r + \frac{y}{\sin(\theta)} \leq x \leq 4 \end{cases} \tag{56}$$

and the solution is computed up to time $T = 0.2$. This setup, introduced in [47], gives rise to the reflection of a strong oblique shock wave on a wall. (Equivalently, upon counter-clockwise rotation by 30°, this setup can be interpreted as a vertical shock impinging on a 30° ramp.) As a result of the shock reflection a number of flow features arise, including, notably, two Mach stems and two contact discontinuities (slip lines), as further discussed below in this section.



(a) Thirty equispaced ρ contours between 0.18 and 5.71

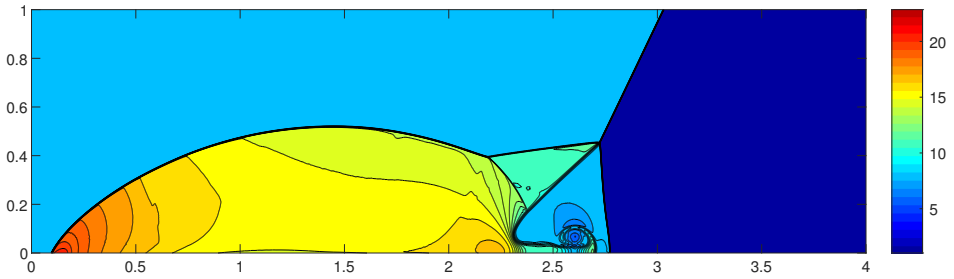


(b) Artificial viscosity at time t=4.

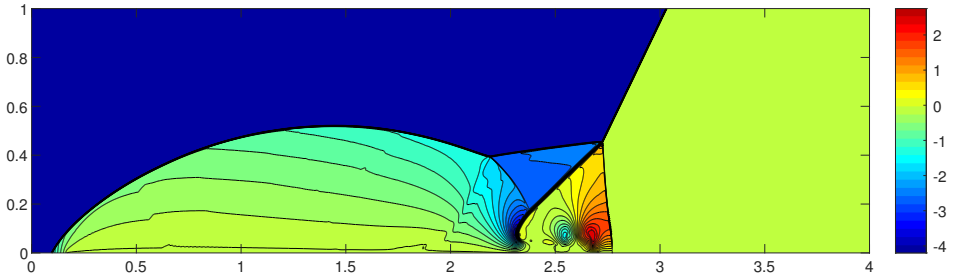
Figure 19: Top panel: Second-order ($d = 2$) FC-SDNN numerical solution to the Mach 3 forward facing-step problem considered in Section 4.3.2, at $t = 4$, obtained by using 1200×400 spatial grid. Bottom panel: Viscosity assignment.

The initial and boundary values used in this context do not exactly coincide with the ones utilized in [47]. Indeed, on one hand, in order to avoid density oscillations near the intersection of the shock and the top computational boundary, we utilize "oblique" Neumann boundary conditions on all flow variables $\mathbf{e} = (\rho, \rho u, \rho v, E)$. More precisely, we enforce zero values on the derivative of $\mathbf{e}$ with respect to the direction parallel to the shock, along both the complete upper computational boundary and the region $0 \leq x \leq x_r$ (that is, left of the ramp) on the lower computational boundary. This method can be considered as a further development of the approach proposed in [44], wherein an extended domain in the oblique direction was utilized in conjunction with Neumann conditions along the normal direction to the oblique boundary. In order to incorporate the oblique Neumann boundary condition we utilized the method described in [2], which, in the present application, proceeds by obtaining relations between oblique, normal and tangential derivatives of the flow variables $\mathbf{e}$. Inflow boundary conditions were used which prescribe time-independent values of ρ, u and v on the left boundary; outflow conditions on the right boundary, in turn, enforce a time independent value of p.

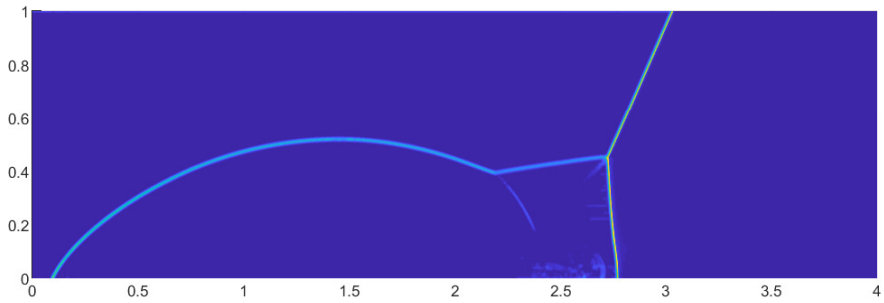
Additionally, following [44], we utilize a numerical viscous incident shock as an initial condition in order to avoid well-known post-shock oscillations, as noted in [18], that result from the use of a sharp initial profile. Such a smeared shock is obtained in our context by applying the FC-SDNN solver to the propagation of an oblique flat shock on all of space (without the ramp) up to time $T = 0.2$, including imposition of oblique Neumann boundary conditions throughout the top and bottom boundary. The solution $\hat{\mathbf{e}}_h$ amounts to a smeared shock profile on the (x, y) plane which, at time t, is centered on the straight line $x = x_s(y, t)$ where $x_s(y, t) = x_r + \frac{U_s}{\sin(\theta)}t + \frac{y}{\tan(\theta)}$. This shock is followed by some back-trailing oscillations, as has



(a) Thirty equispaced ρ contours between 1.0 and 22.8



(b) Thirty equispaced v contours between -4.2 and 2.7



(c) Artificial viscosity at time $t = 0.2$

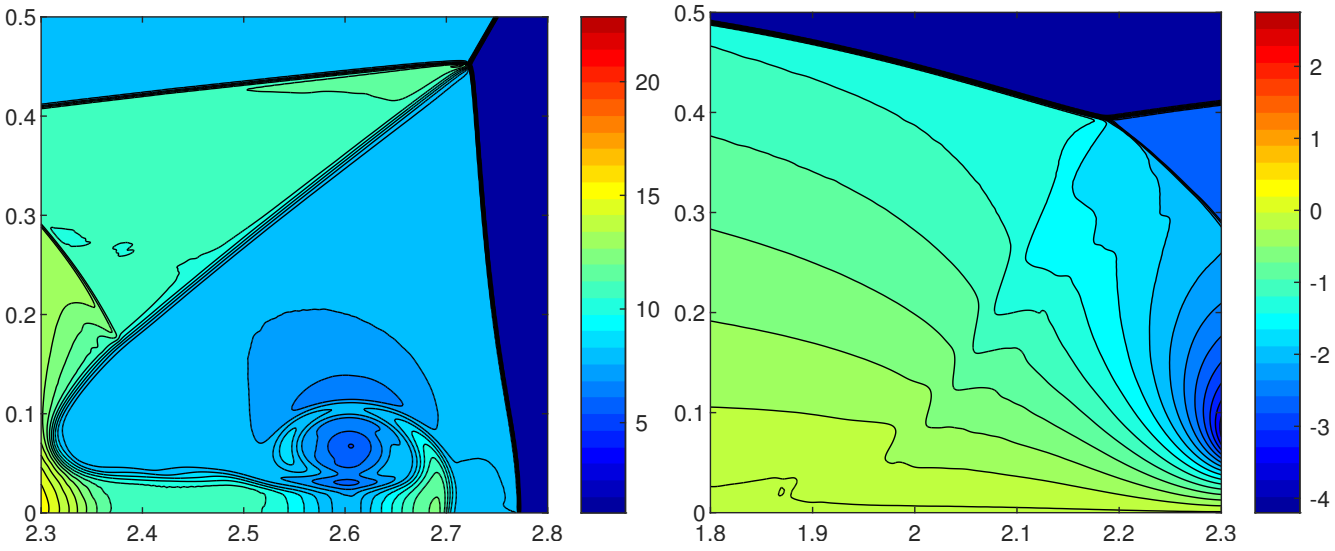
Figure 20: Top and middle panels: Second-order ($d = 2$) FC-SDNN numerical solution to the Double Mach reflection problem considered in Section 4.3.2, at $t = 0.2$, obtained by using 3200×800 spatial grid. Bottom panel: Viscosity assignment.

been observed in [18, 44]. In order to eliminate these artifacts from the initial condition, a diagonal strip of N_d points centered around the shock location is selected. Then, on a strip of N_d points surrounding the initial position of the shock, the initial condition $\mathbf{e}_h$ to the Double Mach reflection problem is defined as

$$\mathbf{e}_h(x, y, 0) = q_{c,r}(x_s(y, 0) - x)\hat{\mathbf{e}}_h(x + x_s(y, T), y, T) + (1 - q_{c,r}(x_s(y, 0) - x))\mathbf{e}(x, y, 0)$$

where $q_{c,r}$ is the window function defined in (34). N_d is taken to be small enough to exclude the back-trailing oscillations from the strip, and large enough as to provide a relatively smooth profile for the initial condition. For the double-Mach solution depicted in Figure 20, which was obtained on the basis of a 3200×800-point grid, the incident-shock parameter values $N_d = 125$, $c = 25$ and $r = 50$ were used.

Among several notable features in the $t = 0.2$ solution we mention the density-contour roll-ups of the primary slip line that are clearly visible in the density component of the solution presented in Figure 20a and highlighted in Figure 21a as well as the extremely weak secondary



(a) (b)

Figure 21: Zoom-in on two features displayed in Figure 20: (a) Density-contour roll-ups in Figure 20(a); and, (b) Secondary slip line in Figure 20(b).

slip line, that is most easily noticeable in the variable v (Figure 20b) as a dip in the contour lines highlighted in Figure 21b but which is also visible in this area as a blip in the density contour lines (Figure 20a). The accurate simulation of these two features has typically been found challenging [19, 44, 47]. The elimination of back-trailing shock oscillations, which was accomplished, as discussed above, by resorting to use of a numerically viscous incident shock, is instrumental in the resolution of the secondary slip line—which might otherwise be polluted by such oscillations to the point of being unrecognizable.

4.4 Computational cost

This section presents an analysis of relative computing costs, averaged over many time steps, that are required by the various elements of the FC-SDNN and FC-EV algorithms for a sample of tests cases—namely the four Euler 1D problems and the 2D Euler Riemann and Shock-vortex problems considered in Sections 4.3.1 and 4.3.2. All simulations were run and timed using Matlab scripts on an Intel(R) Core(TM) i7-9750H @ 2.60GHz computer. Relative computing costs for the Euler 1D problems, using $N = 500$ spatial points (resp. $N = 1000$ spatial points) are presented in Table 3 (resp. Table 4), while those for the Euler 2D problems, simulated on square $N \times N$-point grids with $N = 200$ (resp. $N = 400$) are presented in Table 5 (resp. Table 6). These tables display the fractions of the computing times required by the code elements in FC-SDNN Algorithm 1 that incur the main computational costs of the method, namely, (a) The FC-based differentiations and other operations unrelated to viscosity calculations that are necessary to time-step the algorithm according to the SSPRK-4 time-stepping scheme (line 13); (b) The filtering procedure (line 11); and, (c) The artificial viscosity-assignment procedure (lines 6 through 9). For reference, the tables also present computing costs incurred by related elements in the FC-EV algorithm. Additionally, in view of potential interest, the tables list the relative time required for evaluation of the ANN smoothness classification procedure (point (vi) in line 7 of Algorithm 1); inspection of the various tables shows

this portion of the algorithm generally requires a small fraction of the artificial viscosity-assignment cost.

The following computing-time components are listed in Tables 3 to 6. Times reported for each problem are computed as the times required by various sections of Algorithm 1 for the problem considered, per time-step, and relative to the time required by one FC-SDNN time-step for the problem considered. The computing time per time-step for each algorithmic section was computed as the quotient of the total time required by that section divided by the number of time steps. To reduce the impact of random timing fluctuations, each algorithmic-section time provided in Tables 3 to 6 was obtained as the average of each quantity over twenty separate runs.

- Tot: total computational time.
- AV: Computing time required by the Artificial Viscosity calculation: Lines 6 through 9 in Algorithm 1 for FC-SDNN, and Section 2.2.2 for FC-EV.
- Tot - AV: Self explanatory. Lines 11 through 14 in Algorithm 1 for FC-SDNN.
- ANN: Computing time required for evaluation of the ANN smoothness classification procedure (for the FC-SDNN method only): Point (vi), line 7 in Algorithm 1.
- TS: number of time steps required by the simulation.

A number of general observations emerge from the results presented in Tables 3 to 6.

- For each of the examples considered in these tables, the required FC-EV and FC-SDNN computing times per time-step are essentially the same. A slight but systematic higher cost is incurred per time-step by the FC-SDNN method—which can be attributed to a somewhat larger value of the AV cost inherent in this method.
- Importantly, however, the number of time-steps necessary to complete the simulation can be much lower for the FC-SDNN approach, which results in significantly faster overall FC-SDNN performance. (The significantly smaller time-steps required by the FC-EV method result from this method's use of larger viscosity values—which are needed to smooth oscillations arising, precisely, from the non-smoothness of the FC-EV viscosity distribution used.)
- It should be noted that the relative cost of the SDNN artificial viscosity method is problem dependent: it is higher for problems with multiple shocks (such as the Shu-Osher problem and the Riemann 2D problem presented in Sections 4.3.1 and 4.3.2, respectively) which require larger numbers of neural network evaluations. The FC-EV cost, in contrast, is not affected by the number of shocks involved.
- The various timings show that the relative cost of the artificial viscosity algorithm does not vary substantially as the gridsizes grow.
- The relative cost of the neural network evaluations is modest, less than 10%, resp. 2% for the 1D, resp. 2D problems, and this relative cost diminishes as the gridsize is increased.

Problem	Method	Tot	AV	Tot - AV	ANN	TS
Sod	SDNN	1.00	0.23	0.77	0.01	317
	EV	0.98	0.09	0.89	-	433
Shu Osher	SDNN	1.00	0.21	0.79	0.09	432
	EV	0.94	0.09	0.85	-	1387
Lax	SDNN	1.00	0.19	0.81	0.06	284
	EV	0.92	0.09	0.83	-	668
Blast Wave	SDNN	1.00	0.22	0.78	0.08	613
	EV	0.99	0.09	0.90	-	2346

Table 3: Computing-time components for Euler 1D problems on an N-point grid with $N = 500$.

Problem	Method	Tot	AV	Tot - AV	ANN	TS
Sod	SDNN	1.00	0.18	0.82	0.04	634
	EV	0.95	0.04	0.91	-	865
Shu Osher	SDNN	1.00	0.15	0.85	0.05	864
	EV	0.96	0.05	0.92	-	2774
Lax	SDNN	1.00	0.14	0.86	0.04	568
	EV	0.95	0.04	0.91	-	1337
Blast Wave	SDNN	1.00	0.16	0.84	0.06	1224
	EV	0.94	0.04	0.90	-	4736

Table 4: Computing-time components for Euler 1D problems on an N-point grid with $N = 1000$.

Problem	Method	Tot	AV	Tot - AV	ANN	TS
Shock-vortex	SDNN	1.00	0.18	0.82	0.01	585
Riemann4	SDNN	1.00	0.19	0.81	0.02	650

Table 5: Computing-time components for Euler 2D problems on an $N \times N$-point grid with $N = 200$.

Problem	Method	Tot	AV	Tot - AV	ANN	TS
Shock-vortex	SDNN	1.00	0.20	0.80	0.01	1417
Riemann4	SDNN	1.00	0.21	0.79	0.01	1594

Table 6: Computing-time components for Euler 2D problems on an $N \times N$-point grid with $N = 400$.

5 Algorithm design considerations

In order to facilitate consideration of the proposed algorithms in related but different application areas, including, for example, applications concerning systems of nonlinear conservation laws in 1D, 2D and 3D, this section briefly presents a few rationales inherent in the design of the proposed methods, whose validity should extend beyond the test cases considered in this paper.

FC order. As illustrated in Section 4.2, and, in particular, in Figure 9, use of a high FC order d, say $d = 5$, yields increased accuracy, *even after the formation of shocks,* even though limited asymptotic L^1 and L^2 convergence rates, of order h and $h^{\frac{1}{2}}$, respectively, are observed after shock formation. It has been found, however, that, for problems where shocks impact upon domain boundaries, use of an FC order $d > 2$ may lead to development of spurious oscillations near the impact points, and, even, to overall numerical instability and blow-up, and therefore should be avoided for such applications. The higher-order FC methods are valuable, nevertheless. For example, ongoing efforts concern use of an overlapping-patch setup [1, 5], which could allow the application FC procedures of lower (resp. higher) order in boundary (resp. interior) patches, thus enabling high accuracy-order and improved dispersion character in the domain interior and away from shocks (cf. Figure 6)—which could be exploited, by using different resolutions near and away from boundaries, to obtain low dispersion and uniformly small errors throughout the computational domain at reduced computational cost.

Higher spatial dimensionality. Higher-dimensional FC-SDNN solvers can be obtained by straightforward extension of the 1D and 2D algorithms and operators presented Section 3. On one hand, the FC time-marching scheme presented in Section 3.1 can simply be adapted to the 3D context by emulating the algorithmic prescriptions along a third spatial coordinate direction. The determination of a 3D smoothness classification operator, in turn, could be based on use of one-dimensional partial discrete classification operators, such as those introduced in Section 3.2, on a three-dimensional stencil of points; as in that section, the overall smoothness-classification operator at a given point, for a given proxy variable $\boldsymbol{\phi}$ used, would then be defined as the lowest degree of smoothness among those resulting from the partial classification operators. Similarly, the localization stencils and windowed localization operators defined in Section 3.3 for the 1D and 2D cases can be trivially extended to three dimensions. The definition of a 3D artificial viscosity operator (cf. Section 3.3), finally, requires the empirical selection of values of the weight function R and associated weight operator $\tilde{R}$ appropriate for hyperbolic systems in the 3D context. In this regard we recall that, as indicated in Section 3.3, while use of the 1D weight operator yields stable 2D simulations, the modified 2D operator introduced in Section 3.3.2 leads to smoother flow profiles away from shocks for all the 2D problems considered. We therefore suggest use of the 2D R function values as a starting point for an empirical selection of suitable values of the function R in the 3D case.

Proxy variable. The extension of the FC-SDNN algorithm to other systems of conservation laws requires, in each case, identification of an appropriate proxy variable $\boldsymbol{\phi}$ (Section 3.2.1). The proxy variable should be a dimensionless quantity which is discontinuous whenever any one of the system's conservative variables is discontinuous. As an example, a reasonable candidate for the Magnetohydrodynamic (MHD) system could be a function $\boldsymbol{\phi}$ of both, the fluid-dynamic Mach number and the Magnetic Mach number. The empirical selection of this function should probably be conducted in tandem with the aforementioned selection of the function R and associated operator $\tilde{R}$, to achieve overall stability as well as desired levels of smoothness and accuracy.

6 Conclusions

This paper introduced the FC-SDNN method, a neural network-based artificial viscosity method for evaluation of shock dynamics in non-periodic domains. In smooth flow areas the method enjoys the essentially dispersionless character inherent in the FC method. The smooth but localized viscosity assignments allow for a sharp resolution of shocks and contact discontinuities, while yielding smooth flow profiles away from jump discontinuities. An efficient implementation for general 2D and 3D domains, which could be pursued on the basis of an overlapping-patch setup [1,4], is left for future work.

Acknowledgments. OB and DL gratefully acknowledges support from NSF under contracts DMS-1714169 and DMS-2109831, from AFOSR under contract FA9550-21-1-0373, and from the NSSEFF Vannevar Bush Fellowship under contract number N00014-16-1-2808. Initial conversations with J. Paul are gratefully acknowledged.

References

[1] N. Albin and O. P. Bruno. A spectral FC solver for the compressible Navier–Stokes equations in general domains I: Explicit time-stepping. *Journal of Computational Physics*, 230:6248–6270, 2011.

[2] F. Amlani and O. P. Bruno. An FC-based spectral solver for elastodynamic problems in general three-dimensional domains. *Journal of Computational Physics*, 307:333–354, 2016.

[3] R. B. Blackman and J. W. Tukey. The measurement of power spectra from the point of view of communications engineering—part i. *Bell System Technical Journal*, 37(1):185–282, 1958.

[4] D. L. Brown, W. D. Henshaw, and D. J. Quinlan. Overture: An object-oriented framework for solving partial differential equations. In *International Conference on Computing in Object-Oriented Parallel Environments*, pages 177–184. Springer, 1997.

[5] O. P. Bruno, M. Cubillos, and E. Jimenez. Higher-order implicit-explicit multi-domain compressible Navier-Stokes solvers. *Journal of Computational Physics*, 391:322–346, 2019.

[6] O. P. Bruno and M. Lyon. High-order unconditionally stable FC-AD solvers for general smooth domains I. Basic elements. *Journal of Computational Physics*, 229(6):2009–2033, 2010.

[7] O. P. Bruno and J. Paul. Two-dimensional Fourier Continuation and applications. *arXiv:2010.03901*, 2020.

[8] M. H. Carpenter, D. Gottlieb, S. Abarbanel, and W.-S. Don. The theoretical accuracy of Runge–Kutta time discretizations for the initial boundary value problem: a study of the boundary error. *SIAM Journal on Scientific Computing*, 16(6):1241–1252, 1995.

[9] N. Discacciati, J. S. Hesthaven, and D. Ray. Controlling oscillations in high-order Discontinuous Galerkin schemes using artificial viscosity tuned by neural networks. *Journal of Computational Physics*, 409:109304, 2020.

[10] R. A. Gentry, R. E. Martin, and B. J. Daly. An Eulerian differencing method for unsteady compressible flow problems. *Journal of Computational Physics*, 1(1):87–118, 1966.

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *JMLR Workshop and Conference Proceedings*, pages 249–256, 2010.

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[13] S. Gottlieb. On high order strong stability preserving Runge-Kutta and multi step time discretizations. *Journal of Scientific Computing*, 25(1):105–128, 2005.

[14] J.-L. Guermond, R. Pasquetti, and B. Popov. Entropy viscosity method for nonlinear conservation laws. *Journal of Computational Physics*, 230(11):4248–4267, 2011.

[15] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, III. In *Upwind and high-resolution schemes*, pages 218–290. Springer, 1987.

[16] C. Hirsch. Numerical computation of internal and external flows. Vol. 2-Computational methods for inviscid and viscous flows. *John Wiley & Sons, 1990, 708*, 1990.

[17] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996.

[18] E. Johnsen and S. Lele. Numerical errors generated in simulations of slowly moving shocks. *Center for Turbulence Research, Annual Research Briefs*, pages 1–12, 2008.

[19] F. Kemm. On the proper setup of the double Mach reflection as a test case for the resolution of gas dynamics codes. *Computers & Fluids*, 132:72–75, 2016.

[20] D. A. Kopriva. *Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers*. Springer Science & Business Media, 2009.

[21] A. Kornelus and D. Appelö. Flux-conservative Hermite methods for simulation of nonlinear conservation laws. *Journal of Scientific Computing*, 76(1):24–47, 2018.

[22] A. Lapidus. A detached shock calculation by second-order finite differences. *Journal of Computational Physics*, 2(2):154–177, 1967.

[23] P. Lax. Systems of conservation laws. Technical report, LOS ALAMOS NATIONAL LAB NM, 1959.

[24] P. D. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Communications on pure and applied mathematics*, 7(1):159–193, 1954.

[25] P. D. Lax and X.-D. Liu. Solution of two-dimensional Riemann problems of gas dynamics by positive schemes. *SIAM Journal on Scientific Computing*, 19(2):319–340, 1998.

[26] R. J. LeVeque. *Numerical methods for conservation laws*, volume 132. Springer, 1992.

[27] R. J. LeVeque et al. *Finite volume methods for hyperbolic problems*, volume 31. 2002.

[28] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994.

[29] M. Lyon and O. P. Bruno. High-order unconditionally stable FC-AD solvers for general smooth domains II. Elliptic, parabolic and hyperbolic PDEs; theoretical considerations. *Journal of Computational Physics*, 229(9):3358–3381, 2010.

[30] A. Mazaheri, C.-W. Shu, and V. Perrier. Bounded and compact weighted essentially nonoscillatory limiters for discontinuous Galerkin schemes: Triangular elements. *Journal of Computational Physics*, 395:461–488, 2019.

[31] D. Pathria. The Correct Formulation of Intermediate Boundary Conditions for Runge–Kutta Time Integration of Initial Boundary Value Problems. *SIAM Journal on Scientific Computing*, 18(5):1255–1266, 1997.

[32] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 112, 2006.

[33] R. Ramani, J. Reisner, and S. Shkoller. A space-time smooth artificial viscosity method with wavelet noise indicator and shock collision scheme, Part 1: The 1-D case. *Journal of Computational Physics*, 387:81–116, 2019.

[34] R. Ramani, J. Reisner, and S. Shkoller. A space-time smooth artificial viscosity method with wavelet noise indicator and shock collision scheme, Part 2: the 2-D case. *Journal of Computational Physics*, 387:45–80, 2019.

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

[36] J. Reisner, J. Serencsa, and S. Shkoller. A space–time smooth artificial viscosity method for nonlinear conservation laws. *Journal of Computational Physics*, 235:912–933, 2013.

[37] R. Richtmyer. Proposed numerical method for calculation of shocks. *Los Alamos Report*, 671, 1948.

[38] L. Schwander, D. Ray, and J. S. Hesthaven. Controlling oscillations in spectral methods by local artificial viscosity governed by neural networks. *Journal of Computational Physics*, 431:110144, 2021.

[39] K. Shahbazi, N. Albin, O. P. Bruno, and J. S. Hesthaven. Multi-domain Fourier-continuation/WENO hybrid solver for conservation laws. *Journal of Computational Physics*, 230(24):8779–8796, 2011.

[40] C.-W. Shu. High order ENO and WENO schemes for computational fluid dynamics. In *High-order methods for computational physics*, pages 439–582. Springer, 1999.

[41] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. In *Upwind and High-Resolution Schemes*, pages 328–374. Springer, 1989.

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

[43] B. Stevens and T. Colonius. Enhancement of shock-capturing methods via machine learning. *Theoretical and Computational Fluid Dynamics*, 34:483–496, 2020.

[44] U. Vevek, B. Zang, and T. H. New. On alternative setups of the double Mach reflection problem. *Journal of Scientific Computing*, 78(2):1291–1303, 2019.

[45] J. VonNeumann and R. D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21(3):232–237, 1950.

[46] G. B. Whitham. *Linear and nonlinear waves*. John Wiley & Sons, 2011.

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