General domain FC-based shock dynamics solver

Thesis by
Daniel Victor Leibovici

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

Caltech

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2024
Defended January 22, 2024

ACKNOWLEDGEMENTS

I would like to express my gratitude to all the people who have made this thesis possible and who have guided me through this amazing journey at Caltech, which started nearly a decade ago when I was still a student in France.

First and foremost, I want to thank my advisor, Professor Oscar P. Bruno, for his wonderful supervision over the years. Oscar's contagious enthusiasm for Mathematics, which I discovered when I was a visiting student at Caltech and took ACM 101, helped me decide to pursue a PhD in Applied Mathematics. I will always be grateful for the vast knowledge and the time Oscar imparted with me, which allowed me to learn, grow, and take part in fascinating research projects. But most of all, the patience, encouragement and support he offered me during the challenging times, as well as his excitement for every success obtained along the long path that led to this thesis, were key elements for this PhD, and I hope that our collaboration will continue long after my graduation from Caltech.

I want to thank the members of my candidacy and thesis committee, Professors Dan Meiron, Houman Owhadi and Dale Pullin, for our discussions, insightful comments and guidance regarding my PhD work.

My journey at Caltech started when I was still a student in France, and was made possible by Professor Mathieu Desbrun, who coordinated the exchange visitor program with Ecole des Mines de Paris, and Professor Michael Ortiz, who took a chance in including me in his group while I was a visiting student in 2014 as well as during a SURF in 2015, and gave me an amazing first research experience for which I will always be thankful.

I also want to thank Professor Anima Anandkumar for introducing me to Neural Operators, a novel and promising field in scientific computing, which will constitute a central part in my next research endeavors.

During my PhD, I was fortunate to have the opportunity to intern at SLB, where I could gain some valuable experience in the application of numerical methods in an industry setting as well as in the efficient implementation of these methods, allowing me to become a better researcher. In particular, I would like to thank Dr. Denis Heliot for his valuable insights, and Dr. Gong Li Wang for his kindness and guidance in the project I took part in.

Additionally, I would like to thank fellow members of Professor Bruno's group, past and present: Agustin Fernandez-Lado, Thomas Anderson, Emmanuel Garza, Ambuj Pandey, Martín Maas, Enzo Gaggioli, Sebastian Lamas, Sabhrant Sachan and Manuel Santana. I want to specially thank Jagabandhu Paul, as our conversations allowed me to learn a lot regarding Fourier-Continuation based PDE solvers and their implementation, and were pivotal for the work presented in Chapter 3. I owe a lot to Christoph Bauinger, my dear friend and colleague, with whom I shared the best and also the most difficult times, and who helped me a lot along the way, in particular by sharing with me his vast expertise in high performance computing.

Throughout my years at Caltech, I have received the help from many administrators, at GALCIT and then CMS, as well as the ISP office. I would like to extend my thanks to Lydia Suarez, Sheila Shull and Marta Kahl for their help and guidance during my first visit at Caltech. Special thanks to Diana Bohler, Maria Lopez, Laura Flower Kim and Daniel Yoder for their availability and support.

I want to thank all the friends from my program, who helped make this experience a very pleasant one: Alexander Poremba, Dmitry Burov, Nikola Kovachki (who I will be joining in my next adventure), Lisa Li, Matt Levine, Musab Jilani, Yousuf Soliman, Apurva Badithela, Angel Galvez Merchan, Carmen Amo Alonso, Yasamin Jalalian and Matthieu Darcy. I would also like to thank all my friends from France, who have given me constant support during this long process.

Finally, I want to thank my family for their support and sacrifices, which brought me where I am today. To my late grandfather Mircea. To my loving grandmothers, Titiana and Liliana. To my late father Victor, who taught me to unapologetically follow my own path, and who unfortunately left us too early to see that it led me to a good place. To my mother Rodica, whose strength and love inspires me for all my endeavors.

ABSTRACT

This thesis presents a novel *FC-SDNN* (Fourier Continuation Shock-detecting Neural Network) spectral scheme for the numerical solution of nonlinear conservation laws in general domains and under arbitrary boundary conditions, without the limiting CFL constraints inherent in other spectral schemes for general domains. The approach relies on the use of the Fourier Continuation (FC) method for spectral representation of non-periodic functions in conjunction with smooth artificial viscosity assignments localized in regions detected 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 contour levels. 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, especially in the vicinity of contact discontinuities. The approach, which does not require the use of otherwise ubiquitous positivity-preserving limiters, enjoys a great geometrical flexibility on the basis of an overlapping-patch discretization. This allows its application for the simulation of supersonic and hypersonic flows and shocks, including Euler simulations at significantly higher speeds than previously achieved, such as e.g. Mach 25 re-entry flow speeds, impinging upon complex physical obstacles. This multi-domain approach is suitable for efficient parallelization on large computer clusters, and the MPI implementation proposed in this thesis enjoys high parallel scalability and in particular perfect weak scaling, as demonstrated by simulations on general complex domains. The character of the proposed algorithm is demonstrated through a variety of numerical tests for the linear advection, Burgers and Euler equations in one and two-dimensional non-periodic spatial domains, with results in accordance with physical theory and prior experimental and computational results up to and including both supersonic and hypersonic regimes.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Oscar P Bruno, Jan S Hesthaven, and Daniel V Leibovici. "FC-based shock-dynamics solver with neural-network localized artificial-viscosity assignment". In: *Journal of Computational Physics: X* 15 (2022), p. 100110. DOI: 10.1016/j.jcpx.2022.100110.
D.L. participated in the conception of the project, produced an implementation of the method in the Matlab programming language, generated the data and figures, and participated in the writing of the manuscript. Chapters 1, 2 and 3 are based on this contribution.

[2] Oscar P Bruno and Daniel V Leibovici. *General-domain FC-based shock-dynamics solver*.
D.L. participated in the conception of the project, produced an implementation of the method in the C++ programming language, generated the data and figures, and participated in the writing of the manuscript. Chapter 1, 2 and 4 are based on this contribution. [In Preparation].

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Number *Page*

Chapter 1

INTRODUCTION

This thesis presents a novel *FC-SDNN* (Fourier Continuation Shock-detecting Neural Network) spectral scheme for the numerical solution of nonlinear conservation laws in general domains and under arbitrary boundary conditions, without the limiting CFL constraints inherent in other spectral schemes. The proposed approach relies on use of the FC-Gram Fourier Continuation method [1, 2, 3, 4] for spectral representation of non-periodic functions in conjunction with smooth artificial viscosity assignments localized in regions detected by means of the Shock-Detecting Neural Network (SDNN) method proposed in [5] (in a slightly modified form; cf. Section 4.4.2). The neural network approach introduced in that reference, which utilizes Fourier series to discretize the gas dynamics and related equations in two-dimensional periodic rectangular domains, eliminates Gibbs ringing at shocks (whose locations are determined by means of an artificial neural network) by means of an assignment of artificial viscosity over a small number of discrete points in a close vicinity of the shocks. The use of the classical Fourier spectral method in that contribution restricts the method's applicability to periodic problems (thus precluding, in particular, applicability to geometries containing 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 contour levels in the flow fields. As demonstrated in this thesis via application to a range of well known 1D and 2D shock-wave test configurations, in contrast, the overall FC-SDNN approach yields accurate and essentially oscillation-free solutions for non-periodic problems involving arbitrary boundary conditions in general domains—with illustrations including Euler simulations at re-entry flow speeds past obstacles at e.g. Mach 25 (Section 4.5.2.5).

As indicated above, the FC-SDNN method presented in this thesis provides a general solver for shock-wave problems governed by conservation laws by incorporating 1) The shock-detection methodology mentioned above; in conjunction with 2) The aforementioned rapidly convergent FC-Gram Fourier Continuation representations (which, eliminating the Gibbs phenomenon that would otherwise arise from lack of periodicity, provide rapidly convergent Fourier expansions of non-periodic functions); as well as 3) Smooth artificial viscosity assignments [6] based on use of certain newly-designed *smooth viscosity windows*—which result in (a) smooth flow profiles away from discontinuities as well as (b) sharp shock resolution, and (c) enable applicability to gas-dynamics problems including very strong shocks, without recourse to positivity-preserving limiters for the density and pressure fields (see Remark 13). Unlike polynomial spectral approaches such as the Chebyshev spectral method, the proposed approach does not give rise to quadratically refined meshes near interval endpoints, and it utilizes viscosity assignments that decay proportionally to the spatial mesh size, resulting in 4) CFL restrictions that decay linearly with the spatial meshsize. Finally, in view of its use of an overlapping-patch [1, 2, 7] discretization method 5) the approach enjoys significant geometrical flexibility, afforded by the loose connectivities between overlapping-patch computational subdomains, leading to a lesser geometry-processing overhead than other discretization approaches.

The computational solution of systems of conservation laws has been tackled by means of a variety of numerical methods, including low-order finite volume [8, 9] and finite difference methods equipped with slope limiters [8, 9], as well as higher order shock-capturing methods such as the ENO [10] and WENO schemes [11, 12]. An efficient FC/WENO hybrid solver was proposed in [13]. The use of artificial viscosity as a computational device for conservation laws, on the other hand, was first proposed in [14, 15] and the subsequent contributions [16, 17, 18]. The viscous terms proposed in those papers, which incorporate derivatives of the square of the velocity gradient, may induce oscillations in the vicinity of shocks [18] (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. On the basis of convolution of the spatial derivatives of the unknowns with a compactly supported kernel, the spectral viscosity method [19, 20, 21, 22, 23] similarly acts throughout the computational domain, once again potentially resulting in oversmearing. In the context of a Discontinuous Galerkin scheme, reference [24] proposes the use of a shock-detecting sensor (which relies on problem-dependent parameters) in order to localize the support of the artificial viscosity. In this method a piecewise-constant artificial viscosity profile is used (which vanishes except on "troubled" elements around shocks), leading to a non-smooth overall artificial viscosity profile.

The entropy viscosity method [25] (EV), an approach which is also explored in this thesis in conjunction with an FC-based discretization method, 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 [26] (which additionally introduced a Hermite-based method to discretize the hyperbolic systems) by modifying the EV viscosity term. Like the viscous term introduced by [15], the EV viscosity assignments [25, 26] are themselves discontinuous in the vicinity of shocks, and, thus, their use may introduce spurious oscillations. The C-method [27, 28, 29], 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 [30, 31, 32, 5]. The approach [30, 31, 5] 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 [30, 31] and Fourier spectral schemes [5]. The ML-based approach utilized in [32] for 1D Burgers and Euler problems, in turn, modifies the finite volume coefficients utilized in the WENO5-JS scheme: by learning slightly perturbed values of the coefficients the method may yield improved accuracies, but it could unfortunately also result in decreased accuracies, depending on the discretization sizes used.

The FC-SDNN method proposed in this thesis relies problem-independent Artificial Neural Network based (ANN) approach to discontinuity detection and assignment of localized viscosity $\mu[\mathbf{e}]$ (which, in particular, does not rely on selection of problem-dependent parameters), that was introduced in [5] and was subsequently modified and extended to the non-periodic FC-based context in [6] (Chapter 3 in this thesis). Briefly, the approach relies on a single pre-trained neural network for detection of discontinuities on the basis of Gibbs oscillations in Fourier series, together with the spectral resolution provided by the FC method in the fully general context of non-periodic problems with given boundary conditions—allowing for sharp resolution of flow features such as shocks and contact discontinuities and smooth accurate evaluation of flow fields away from discontinuities. In view of its innovative smooth viscosity assignments, further, this procedure effectively eliminates Gibbs oscillations while avoiding introduction of a type of flow-field roughness that is often evidenced by the serrated contour levels produced by other methods. And, importantly, as discussed in the first paragraph in Section 4, the algorithm does not require use of positivity-preserving limiters to prevent the occurrence of negative densities and pressures.

The artificial-viscosity assignment methods utilized in this thesis are first presented, in Sections 3.1.2 and 3.1.3, in the context of domains that can be discretized by means of a single 1D or 2D Cartesian grid. The capabilities of the proposed algorithm for such simple domains are illustrated in Section 3.2 via a variety of 1D and 2D numerical results for the Linear Advection, Burgers and Euler equations. The (non-trivial) extension of this methodology to the general-domain multi-patch context, in turn, is presented in Section 4.3, and numerical results for the solution of the 2D Euler equations on complex geometries are then presented in Section 4.5, demonstrating the applicability of the method for supersonic and hypersonic flows, and for general non-smooth geometries, and demonstrating solutions for some of the fastest flows and shock-wave problems ever considered.

In order to provide a useful reference point, Chapter 3 in this thesis also presents an FC-based version, termed FC-EV, of the EV algorithm [25]. 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.

A Message Passing Interface (MPI) massive parallel implementation of the FC-SDNN method is introduced in Section 4.4. On the basis of the patch/subpatch general-geometry domain decomposition methodology presented in Section 4.1, the FC-SDNN parallelization strategy enjoys high parallel scalability. In particular, a perfect weak scaling is demonstrated in the context of problem enlargement and mesh refinement, as illustrated by the complex-geometry test problems presented in Section 4.5.1, as the number of cores utilized is

progressively increased up to a maximum of 1620 cores. High quality strong parallel scaling is also demonstrated by increasing the number of computing cores on a geometry containing a fixed number of approximately 8.8 million discretization points, and 35.2 million unknowns.

This thesis is organized as follows. Chapter 2 presents necessary preliminaries concerning the hyperbolic problems under consideration, as well as the Fourier Continuation method, and including basic background on the artificial-viscosity strategies we consider. Chapter 3 then describes the proposed FC-SDNN approach in the context of simple geometries, and demonstrates the algorithm's performance for a variety of non-periodic linear and nonlinear hyperbolic problems. In particular, test cases are considered in that chapter for the linear advection equation, Burgers equation and Euler's equations in one- and two-dimensional rectangular and non-rectangular spatial domains, including cases in which shock waves meet including smooth and non-smooth physical boundaries. The proposed extension of the method to general complex domains via an overlapping-patch approach is presented in Chapter 4, together with the parallel

implementation strategies used and the method's parallel scalability tests. A number of applications presented in that chapter, including hypersonic shocks and flows interacting with physical obstacles, further illustrate the method's performance. Chapter 5 finally presents a number of concluding remarks, and proposes a number of extensions and applications of the present methodology.

Chapter 2

PRELIMINARIES

2.1 Conservation laws

This thesis 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

$$\frac{\partial}{\partial t}\mathbf{e}(\boldsymbol{x},t) + \nabla \cdot \big(f(\mathbf{e}(\boldsymbol{x},t))\big) = 0 \tag{2.1}$$

on a bounded domain $\Omega \subset \mathbb{R}^q$, where $\mathbf{e} : \Omega \times [0,T] \to \mathbb{R}^r$ 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 (2.1), including the Linear Advection equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{2.2}$$

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

$$\frac{\partial u}{\partial t} + \frac{1}{2}\left(\frac{\partial u^2}{\partial x}\right) = 0 \tag{2.3}$$

and

$$\frac{\partial u}{\partial t} + \frac{1}{2}\left(\frac{\partial u^2}{\partial x}\right) + \frac{1}{2}\left(\frac{\partial u^2}{\partial y}\right) = 0, \tag{2.4}$$

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

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho \\ \rho u \\ E\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}\rho u \\ \rho u^2 + p \\ u(E+p)\end{pmatrix} = 0 \tag{2.5}$$

and

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho \\ \rho u \\ \rho v \\ E\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}\rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p)\end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix}\rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p)\end{pmatrix} = 0 \tag{2.6}$$

for each of which we have

$$\mathbf{e} = (\rho, \rho \boldsymbol{u}, E)^T, \quad f(\mathbf{e}) = (\rho \boldsymbol{u}, \rho \boldsymbol{u} \otimes \boldsymbol{u} + p\mathbb{I}, \boldsymbol{u}(E + p))^T, \tag{2.7}$$

where ρ, $\boldsymbol{u}$, p and θ denote the density, velocity vector, pressure and temperature, respectively, and where E denotes the total energy:

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho|\boldsymbol{u}|^2. \tag{2.8}$$

Noting that

$$p = \rho\theta, \tag{2.9}$$

we may also write

$$E = \frac{\rho\theta}{\gamma - 1} + \frac{1}{2}\rho|\boldsymbol{u}|^2. \tag{2.10}$$

The temperature θ, which is not an unknown in equation (2.6), is mentioned here in view of the important role it plays in regard to enforcement of adiabatic boundary conditions.

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 [8]

$$a = \sqrt{\frac{\gamma p}{\rho}} \tag{2.11}$$

for the Euler equations plays important roles in the various artificial viscosity assignments considered in this thesis 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 (2.7), $\nabla \cdot \big(f(\mathbf{e})\big)$ 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 (2.1) 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 (2.1), 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 $\nabla \cdot \big(f_v[\mathbf{e}]\big)$ to the right hand side of (2.1), where the “viscous flux” operator

$$f_v[\mathbf{e}] = \mu[\mathbf{e}]\mathbf{D}[\mathbf{e}], \tag{2.12}$$

(which, for a given vector-valued function $\mathbf{e}(x,t)$, produces a vector-valued function $f_v[\mathbf{e}](x,t)$ defined in the complete computational domain), is given in terms of a certain “viscosity” operator $\mu[\mathbf{e}](x,t)$ (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

$$\frac{\partial \mathbf{e}(\boldsymbol{x},t)}{\partial t} + \nabla \cdot \big(f(\mathbf{e}(\boldsymbol{x},t))\big) = \nabla \cdot \big(f_v[\mathbf{e}](\boldsymbol{x},t)\big) \tag{2.13}$$

results.

Per the discussion in Section 1, this thesis 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 $\mu[\mathbf{e}](x,t)$ 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 $\mu[\mathbf{e}](x,t)$ 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 thesis is based on the neural-network strategy introduced in [5] for detection of discontinuities on the basis of Gibbs oscillations in Fourier series, together with a novel selection of the operator μ in (2.12) that yields spatially localized but smooth viscosity assignments: per the description in Section 3.1.3, the FC-SDNN viscosity $\mu[\mathbf{e}](\boldsymbol{x},t)$ is a smooth function that vanishes except in narrow regions around flow discontinuities. The differential operator $\mathbf{D}$, in turn, is simply given by

$$\mathbf{D}[\mathbf{e}](\boldsymbol{x},t) = \nabla(\mathbf{e}(\boldsymbol{x},t)), \tag{2.14}$$

where the gradient is computed component-wise. As indicated in Chapter 1, the smoothness of the proposed viscosity assignments is inherited by the resulting flows away from flow discontinuities, thus helping eliminate the serrated contour level 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 [25] 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

$$R_{EV}[\mathbf{e}](\boldsymbol{x},t) = \frac{\partial \eta(\mathbf{e}(\boldsymbol{x},t))}{\partial t} + \nabla \cdot \nu(\mathbf{e}(\boldsymbol{x},t)) \tag{2.15}$$

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

In practice, reference [25] proposes $\eta(\mathbf{e}) = \frac{u^2}{2}$, $\nu(\mathbf{e}) = a\frac{u^2}{2}$ and $C(\mathbf{e}) = a$ for the Linear Advection equation (2.2), $\eta(\mathbf{e}) = \frac{u^2}{2}$, $\nu(\mathbf{e}) = \frac{u^3}{3}$ and $C(\mathbf{e}) = u$ for the 1D and 2D Burgers equations (2.3) and (2.4), and $\eta(\mathbf{e}) = \frac{\rho}{\gamma-1}\log(p/\rho^\gamma)$, $\nu(\mathbf{e}) = \boldsymbol{u}\frac{\rho}{\gamma-1}\log(p/\rho^\gamma)$ and $C(\mathbf{e}) = |\boldsymbol{u}| + a$ (where a denotes the speed of sound (2.11)) for the 1D and 2D Euler equations (2.5) and (2.6). As for the normalization operator, reference [25] 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

$$\mu[\mathbf{e}](\boldsymbol{x},t) = \min(\mu_{max}[\mathbf{e}](t), \mu_E[\mathbf{e}](\boldsymbol{x},t)) \tag{2.16}$$

where the maximum viscosity μ_{max} is given by

$$\mu_{max}[\mathbf{e}](t) = c_{max} h \max_{x \in \Omega} |C(\mathbf{e}(\boldsymbol{x}, t))| \tag{2.17}$$

and where

$$\mu_E[\mathbf{e}](\boldsymbol{x}, t) = c_E h^2 \frac{|R_{EV}[\mathbf{e}](\boldsymbol{x}, t)|}{N[\mathbf{e}](\boldsymbol{x}, t)}. \tag{2.18}$$

In particular, the EV viscosity function depends on two parameters, c_{max} and c_E, both of size $\mathcal{O}(1)$, that, following [25], 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

$$\mathbf{D}[\mathbf{e}](\boldsymbol{x}, t) = \nabla(\mathbf{e}(\boldsymbol{x}, t)), \tag{2.19}$$

while for the Euler equations it is given by

$$\mathbf{D}[\mathbf{e}](\boldsymbol{x}, t) = \begin{pmatrix} 0 \\ \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) \\ \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)\boldsymbol{u} + \kappa \nabla(p/\rho) \end{pmatrix} \tag{2.20}$$

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 $\kappa = \frac{\mathcal{P}}{\gamma - 1}$, 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.

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 $[0, 1]$. Thanks to its high order approximation capacity and low dispersion properties (see Section 3.2.1.2), the Fourier continuation method has been used in high order solvers [1, 2, 3, 4, 33, 34], and more recently in the field of ML based PDE

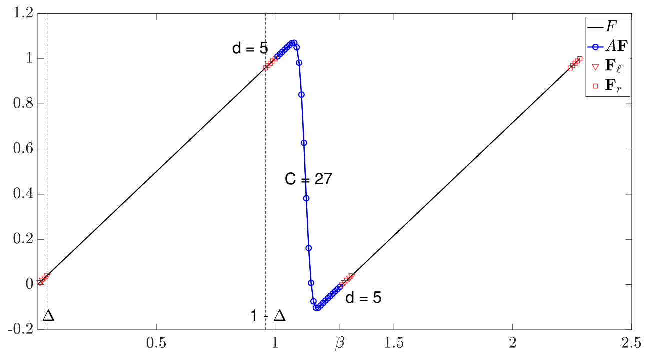


Figure 2.1: Fourier Continuation of the non-periodic function $F(x) = x$ on the interval $[0, 1]$. With reference to the text, the red triangles (resp. squares), represent the $d = 5$ left (resp. right) matching points, while the blue circles represent the $C = 27$ continuation points.

solvers [35, 36]. The algorithm relies on the 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

$$F^c(x) = \sum_{k=-M}^{M} \hat{F}_k^c \exp(2\pi i k x/\beta) \tag{2.21}$$

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 2.1. (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 thesis 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 (2.21).

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, which, using a number d of discretization points near the endpoint intervals, as illustrated in Figure 2.1, 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

$$\mathbf{F}^c = \begin{pmatrix} \mathbf{F} \\ A_\ell Q^T \mathbf{F}_\ell + A_r Q^T \mathbf{F}_r \end{pmatrix} \tag{2.22}$$

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 3.2, 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 thesis, 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 3.2.1.2, for which errors can accumulate on account of the spatio-temporal periodicity, for all cases in Section 3.2 for which both the $d = 2$ and $d = 5$ simulations were performed (which include those presented in Sections 3.2.1.1 (1D linear advection), 3.2.2.2 (2D Burgers equation) and 3.2.3.1 (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 3.7, which displays solutions produced by means of two different methods, namely, the FC-based order-5 FC-SDNN algorithm (Section 3.1) 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`.

Chapter 3

FC-SDNN I: SINGLE-PATCH FC-BASED SHOCK-DYNAMICS SOLVER WITH NEURAL-NETWORK SHOCK DETECTION AND LOCALIZED ARTIFICIAL-VISCOSITY

As indicated in Chapter 1, the FC-SDNN solver [6] detects shocks by exploiting the associated discontinuity-induced Gibbs oscillations in the FC expansions of the flow variables. The method assigns a shock-localized *smooth* viscous term of the form (2.12) on the basis of the viscosity operator $\mu = \mu[\mathbf{e}](x,t)$ developed in Sections 3.1.2 and 3.1.3 below and encapsulated in equations (3.17) (1D) and (3.22) (2D). By relying on smooth viscosity assignments, further, the algorithm effectively eliminates Gibbs oscillations (while still detecting incipient oscillatory behavior in the smooth but sharp-near-shocks flow fields), while avoiding introduction of flow-field roughness—that is often evidenced by the serrated contour levels resulting from the non-smooth viscosities utilized by other methods. In view of its use of the 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 3.7) while enabling solution under general (and, in particular, non-periodic) boundary conditions. Unlike other techniques, finally, the approach does not rely on use of problem-dependent algorithmic parameters.

This chapter presents the FC-SDNN solver in a simple geometrical context, for which the computational domain can be discretized using a single Cartesian grid (a single FC patch). A variety of numerical illustrations produced by the FC-SDNN algorithm in this simple context are presented in Section 3.2—including single-patch results for one- and two-dimensional Linear-Advection, Burgers and Euler-equation problems. (A general-domain multi-patch version of the FC-SDNN algorithm is presented in Chapter 4.) In order to provide a useful reference point, this chapter also presents an FC-based version, termed FC-EV, of the entropy-viscosity (EV) algorithm [25]. (The modified version [26] of the entropy viscosity approach, which was also tested as a possible reference solver, was not found to be completely reliable in the 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 thesis.) We find that the FC-SDNN algorithm generally provides significantly more accurate numerical approximations than the FC-EV approach, as the localized artificial viscosity in the former method induces a much lower dissipation level than the latter.

This chapter consists of two sections: Section 3.1 describes the proposed single-patch FC-SDNN approach, and Section 3.2 then demonstrates the algorithm's performance for a variety of non-periodic linear and nonlinear hyperbolic problems. The numerical results include test cases for one- and two-dimensional rectangular and non-rectangular single-patch spatial domains, as well as cases in which shock waves meet smooth and non-smooth physical boundaries.

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

3.1.1 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

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

and

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

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

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

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 chapter 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 3.21a. Extensions to general domains Ω, which could rely on either an embedded-boundary [3, 4, 37] approach, or an overlapping patch boundary-conforming curvilinear discretization strategy [1, 2, 33], is left for future work.

A spatially-discrete but time-continuous version of the solution vector $\boldsymbol{e}(\mathbf{x}, t)$ considered in Chapter 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 (2.13) becomes

$$\frac{d\boldsymbol{e}_h(t)}{dt} = L[\boldsymbol{e}_h(t)], \tag{3.1}$$

where L denotes a consistent discrete approximation of the spatial operator in (2.13).

The discrete time evolution of the problem, on the other hand, is produced, throughout this thesis, by means of the 4-th order strong stability preserving Runge-Kutta scheme (SSPRK-4) [38]—which, while not providing high convergence orders for the non-smooth solutions considered in this thesis, 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

$$\Delta t = \frac{\text{CFL}}{\pi\left(\frac{\max_{\boldsymbol{x}\in\Omega}|S[\mathbf{e}](\mathbf{x},t))|}{h} + \frac{\max_{\boldsymbol{x}\in\Omega}\mu[\mathbf{e}](\boldsymbol{x},t)}{h^2}\right)}. \tag{3.2}$$

Here CFL is a constant parameter that must be selected for each problem considered (as illustrated by the various selections utilized in Section 3.2), and $\mu[\mathbf{e}](\mathbf{x}, t)$ and $S = S[\mathbf{e}](\mathbf{x}, t)$ denote the artificial viscosity (equations (3.17) and (3.22)) and a *maximum wave speed bound* (MWSB) operator (which must be appropriately selected for each equation; see Section 3.1.3) at the spatio-temporal point $(\mathbf{x}, t)$. (To avoid confusion we emphasize that equation (3.2) utilizes the *maximum* value for all $\mathbf{x} \in \Omega$ of the selected bound $S[\mathbf{e}](\mathbf{x}, t)$ 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

$$(\hat{F}^c)'_k = \frac{2\pi i k}{\beta}\hat{F}^c_k \tag{3.3}$$

of the derivative of the series (2.21) 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 3.1.4.

The boundary conditions of Dirichlet and Neumann considered in this chapter are imposed as part of the differentiation process described above. Dirichlet boundary conditions at time $t_{n,\nu}$ ($t_n < t_{n,\nu} \le t_{n+1}$) 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 $\mathbf{e}_h$ 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 [39], 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 thesis, 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 [39, 40], could also be used in conjunction with the proposed approach. Another alternative, under which no boundary conditions are enforced at intermediate Runge-Kutta stages [41], can also be utilized in our context, but we have found the conventional method leads to smoother solutions near boundaries.

3.1.2 Neural network-induced smoothness-classification

3.1.2.1 Smoothness-classification operator and data pre-processing

The method described in the forthcoming Section 3.1.3 for determination of the artificial viscosity values $\mu[\mathbf{e}](\mathbf{x}, t)$ (cf. also Section 2.2) relies on the "degree of smoothness" of a certain function $\Phi(\mathbf{e})(\mathbf{x}, t)$ (called the "proxy variable") of the unknown solution vector $\mathbf{e}$. In detail, following [5], in this thesis a proxy variable $\Phi(\mathbf{e})$ is used, which equals the velocity u, $\Phi(\mathbf{e}) = u$, (resp. the Mach number, $\Phi(\mathbf{e}) = \|\mathbf{u}\| \sqrt{\frac{\rho}{\gamma p}}$) for equations (2.2) through (2.4) (resp. equations (2.5) and (2.6)). The degree of smoothness of the function $\Phi(\mathbf{e})$ at a certain time t is characterized by a smoothness-classification operator $\tau = \tau[\Phi(\mathbf{e})]$ that analyzes the oscillations in an FC expansion of $\Phi(\mathbf{e})$—which is itself obtained from the discrete numerical values $\phi = \Phi(\boldsymbol{e}_h)$, so that, in particular, $\tau[\Phi(\mathbf{e})] = \tilde{\tau}[\phi]$ 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 3.1.2.2.)

We first describe the operator $\tilde{\tau} = \tilde{\tau}[\phi]$ 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 $(\hat{\phi}^c_{-M}, \ldots, \hat{\phi}^c_M)^T$ of $\Phi(\boldsymbol{e})$ by applying the FC procedure described in Section 2.3 to the column vector $(\phi_0, \ldots, \phi_{N-1})^T$. (Note that in the present 1D case we have $\phi_j = \Phi(\boldsymbol{e}_j)$.)

(ii) For a suitable selected non-negative number $\delta < h$, evaluate the values $\phi_j^{(\delta)}$ $(0 \le j \le N + C - 1\ )$ 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 $\hat{\phi}^\delta = (\hat{\phi}^\delta_{-M}, \dots, \hat{\phi}^\delta_M)$ where $\hat{\phi}^\delta_j = \hat{\phi}^c_j \exp(\frac{2\pi i j\delta}{\beta})$. Here, as in Section 2.3 and equation (2.21), 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 3.1.2.2, different values of δ are used in the training process.

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

$$\phi^{(\delta,j)} = \left(\phi^{(\delta)}_{m(j-3,N+C)}, \dots, \phi^{(\delta)}_{m(j,N+C)}, \dots, \phi^{(\delta)}_{m(j+3,N+C)}\right)^T$$

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 $(x_{j-3}, \dots, x_j, \dots, x_{j+3})$ may not be fully contained within the physical domain.)

(iv) Using the previously selected stencils, obtain the modified stencils $\widetilde{\phi}^{(\delta,j)} = \left(\widetilde{\phi}^{(\delta)}_{m(j-3,N+C)}, \dots, \widetilde{\phi}^{(\delta)}_{m(j,N+C)}, \dots \widetilde{\phi}^{(\delta)}_{m(j+3,N+C)}\right)^T$ given by

$$\widetilde{\phi}^{(\delta)}_{m(j+r,N+C)} = \phi^{(\delta)}_{m(j+r,N+C)} - \ell_{j+r} \quad (-3 \le r \le 3), \tag{3.4}$$

that result by subtracting the "straight line"

$$\ell_{j+r} = \phi^{(\delta)}_{m(j-3,N+C)} + \frac{r+3}{6}(\phi^{(\delta)}_{m(j+3,N+C)} - \phi^{(\delta)}_{m(j-3,N+C)}) \quad (-3 \le r \le 3) \tag{3.5}$$

passing through the first and last stencil points.

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

$$\check{\phi}^{(\delta,j)} = \left(\check{\phi}^{(\delta)}_{m(j-3,N+C)}, \dots, \check{\phi}^{(\delta)}_{m(j,N+C)}, \dots \check{\phi}^{(\delta)}_{m(j+3,N+C)}\right)^T,$$

given by

$$\check{\phi}^{(\delta)}_{m(j+r,N+C)} = \frac{2\widetilde{\phi}^{(\delta)}_{m(j+r,N+C)} - M^{(+)}_j - M^{(-)}_j}{M^{(+)}_j - M^{(-)}_j} \quad (-3 \le r \le 3) \tag{3.6}$$

where

$$M^{(+)}_j = \max_{-3 \le r \le 3} \widetilde{\phi}^{(\delta)}_{m(j+r,N+C)} \quad \text{and} \quad M^{(-)}_j = \min_{-3 \le r \le 3} \widetilde{\phi}^{(\delta)}_{m(j+r,N+C)}. \tag{3.7}$$

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

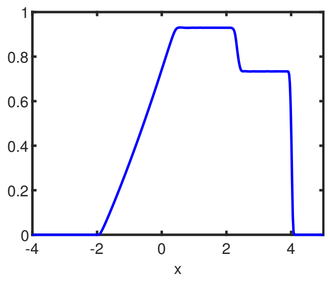


Figure 3.1: Mach number proxy variable for the Sod problem in Section 3.2.3.1 at time $T = 2$.

(vi) Apply the ANN algorithm described in Section 3.1.2.2 to each one of the stencils $\check{\phi}^{(\delta,j)} = \{\check{\phi}^{(\delta)}_{m(j-3,N+C)}, \ldots, \check{\phi}^{(\delta)}_{m(j,N+C)}, \ldots \check{\phi}^{(\delta)}_{m(j+3,N+C)}\}$, to produce a four-dimensional vector w^j of estimated probabilities (EP) for each $j \in [0, \ldots, N-1]$, where w^j_1 is the EP that $\Phi(\mathbf{e})$ is discontinuous on the subinterval $I_j = [x_j - 3h, x_j + 3h]$, where, for $i = 2, 3$, w^j_i equals the EP that $\Phi(\mathbf{e}) \in C^{i-2} \setminus C^{i-1}$ on I_j, and where, for $i = 4$, w^j_i equals the EP that $\Phi(\mathbf{e}) \in C^2$ on I_j. Define $\tau[\phi]_j$ as the index i corresponding to the maximum entry of w^j_i $(i = 1, \ldots 4)$:

$$\tilde{\tau}[\phi]_j = \underset{1\le i\le 4}{\arg\max}(w^j_i) \quad (0 \le j \le N-1). \tag{3.8}$$

(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 $\Phi(\mathbf{e})$ is estimated, can extend beyond the physical domain and into the extended Fourier Continuation region; cf. also point iii above.) As an illustration, Figure 3.1 displays the Mach number proxy variable for the numerical solution of the Sod problem presented Section 3.2.3.1, at time $T = 2$, which includes all major Euler flow features, namely a shock (near $x = 4$), a contact (near $x = 2$), and a rarefaction fan (slope in the interval $-2 \le x \le 0$).

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 $\tau_{xy}[\Phi(\mathbf{e})] = \tilde{\tau}_{xy}[\phi]$, similar to the 1D operator, which classifies the smoothness of the proxy variable $\Phi(\mathbf{e})$ on the basis of its discrete values $\phi = \phi_{ij}$. 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 $\phi = \phi_{ij}$ (see Section 3.1.1). 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 $\phi = \phi_{ij}$, the i-th horizontal section $\phi_{i:}$ (resp. the j vertical section $\phi_{:j}$) of ϕ is defined by $(\phi_{i:})_j = \phi_{ij}, j \in \mathcal{I}_{i:}$ (resp. $\left(\phi_{:j}\right)_i = \phi_{ij}, i \in \mathcal{I}_{:j}$). Utilizing these notations we define

$$\tilde{\tau}_{xy}[\phi]_{ij} = \min\left\{\tilde{\tau}_x[\phi_{i:}]_j, \tilde{\tau}_y[\phi_{:j}]_i\right\}, \tag{3.9}$$

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 (3.8) 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 $\check{\phi}_j^{(\delta,j)}$ for which $M_j^{(+)} - M_j^{(-)} \le \varepsilon$, for a prescribed value of ε, are assigned regularity $\tilde{\tau}[\phi]_j = 4$. Throughout this thesis we have used the value $\varepsilon = 0.01$.

3.1.2.2 Neural network architecture and training

The proposed strategy relies on standard neural-network techniques and nomenclature [42, Sec. 6]. Following common practice in the field of machine learning, the construction of the neural-network shock-detection algorithm used in this thesis, which follows the corresponding shock-detection algorithm introduced in [5], relies on the use of a number of architectural and training-strategy selections, as well as associated parameters, including selection of the number of hidden layers, the activation function, the size of the input stencil of function values, the number of discretization points utilized for the representation of the functions used in the training set, the value of the shift parameter δ, the backward propagation algorithm, the batch size, the learning parameter, and the initialization procedure for the weights and biases. In the spirit of the field, the selections used in this thesis, which are detailed in what follows, are based on the extensive experimentation and resulting recommendations provided in [5], and they additionally include certain adjustments, mentioned in Sections 4.4.2 and below in this section, that were found beneficial in the context of the FC based implementation proposed in this thesis.

In detail, the shock-detection algorithm utilized in this thesis is based on use of an ANN with a depth of four layers, including three fully-connected hidden layers of sixteen neurons each, as illustrated in Figure 3.2. 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 [42, Sec. 4.1], the EP mentioned in point (vi) of the previous section, are obtained:

$$w_i = \frac{e^{\widetilde{w}_i}}{\sum_{\ell=1}^{4} e^{\widetilde{w}_\ell}}, \quad 1 \le i \le 4. \tag{3.10}$$

(The values w_i^j $(1 \le i \le 4)$ mentioned in point (vi) result from the expression (3.10) when the overall scheme described above in the present Section 3.1.2.2 is applied to $z = \check{\phi}^{(\delta,j)}$.) The ELU activation function

$$\mathrm{ELU}(x; \alpha_0) = \begin{cases} x & \text{if } x > 0 \\ \alpha_0(e^x - 1) & \text{if } x \le 0, \end{cases} \tag{3.11}$$

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 $\mathcal{F} = \{(F_k, D_k), k = 1, 2, \dots\}$ of all pairs (F_k, D_k), 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

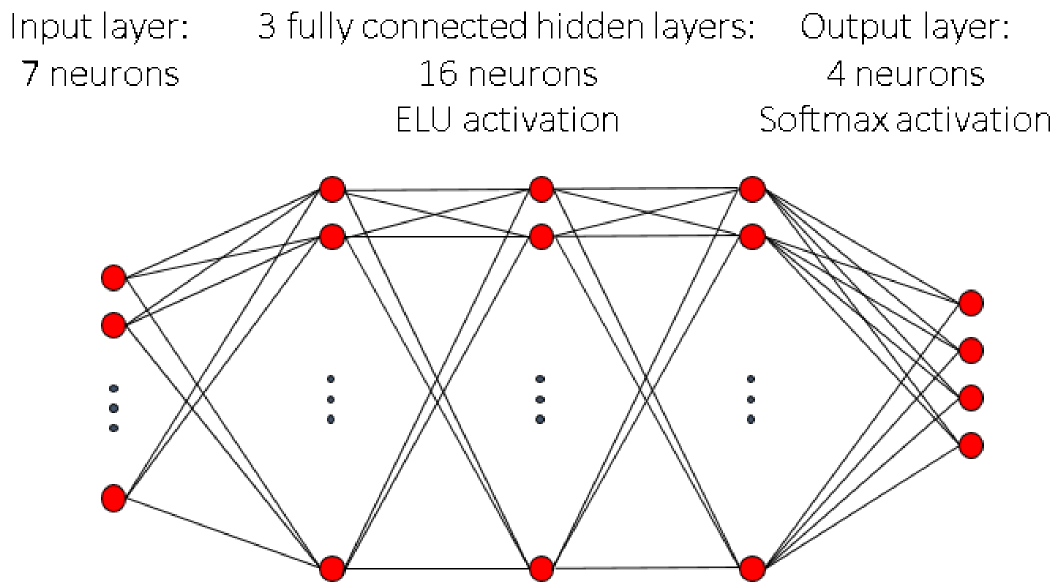


Figure 3.2: Four-layer ANN used in the shock-detection algorithm, including three sixteen-neuron fully-connected hidden layers as well as the ELU an Softmax activation functions.

parameter-dependent analytic expressions

$$
\begin{aligned}
&f_1(x) = \sin(2ax)\\
&f_2(x) = a|x-\pi|\\
&f_3(x) = \begin{cases} a_1 & \text{if} \quad |x-\pi| \le a_3 \\ a_2 & \text{if} \quad |x-\pi| > a_3 \end{cases}\\
&f_4(x) = \begin{cases} a_1|x-\pi| - a_1a_3 & \text{if} \quad |x-\pi| \le a_3 \\ a_2|x-\pi| - a_2a_3 & \text{if} \quad |x-\pi| > a_3 \end{cases}\\
&f_5(x) = \begin{cases} 0.5a_1|x-\pi|^2 - a_1a_3 & \text{if}|x-\pi| \le a_3 \\ a_2|x-\pi|^2 - a_2 - 0.5a_3^2(a_1-a_2) & \text{if}|x-\pi| > a_3 \end{cases}
\end{aligned}
\tag{3.12}
$$

proposed in [5], for each one of the possible selections of the parameters a, a_1, a_2, a_3, as prescribed in Table 3.1. The corresponding parameter dependent restriction domains D_k are also prescribed in Table 3.1; 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 [42, Sec. 7.7] and thus to provide the user with an indication of whether overfitting has occurred.

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 C^2, $C^1 \setminus C^2$, $C^0 \setminus C^1$, 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 preprocessed 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 (3.10), 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 $w_i = A_i(X, z)$ $(1 \le i \le 4)$.

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 [42, Sec. 6.2]

$$\mathcal{L}(X) = -\frac{1}{N_{\mathcal{T}}} \sum_{z \in \mathcal{D}_{\mathcal{F}}^{\mathcal{T}}} \sum_{i=1}^{4} C_i(z) \log(A_i(X, z)) \tag{3.13}$$

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.

$f(x)$	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 3.1: Data set.

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 [42, Secs. 8.1, 8.4] (which was found to yields higher validation accuracy in the context of our FC implementation than the ADAM optimizer used in [5]), 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 [43] (instead of the initialization strategy utilized in [5]). 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 thesis, has a training accuracy of 99.61% and validation accuracy of 99.58%.

3.1.3 SDNN-localized artificial viscosity algorithm

As indicated in Chapter 1, in order to avoid introduction of spurious irregularities in the flow field, the algorithms proposed in this thesis relies on use of smoothly varying artificial viscosity assignments. For a given discrete solution vector $\mathbf{e}_h$, the necessary grid values of the artificial viscosity, which correspond to discrete values of the continuous operator $\mu = \mu[\mathbf{e}]$ in (2.12), are provided by a certain discrete viscosity operator $\tilde{\mu} = \tilde{\mu}[\mathbf{e}_h]$. The discrete operator $\tilde{\mu}$ is defined in terms of a number of flow- and geometry-related concepts, namely the proxy variable ϕ defined in Section 3.1.2.1 and the smoothness-classification operator given by equations (3.8) and (3.9) 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 3.1.1) 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 $\tilde{\mu} = \tilde{\mu}[\mathbf{e}_h]$ is provided in Sections 3.1.3.1 and 3.1.3.2, respectively.

3.1.3.1 One-dimensional case

The proxy variable ϕ and 1D smoothness-classification operator τ that are used in the definition of the 1D artificial viscosity operator have been described earlier in this chapter; 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 thesis 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 $\widetilde{R}[\eta]_i = R(\eta_i)$.

As indicated in Section 3.1.1, the proposed artificial viscosity we use is produced in terms of a Maximum Wave Speed Bound (MWSB) operator $S : \mathcal{G}^q \longrightarrow \mathcal{G}$, which, as is known, plays a significant (albeit different) role in Rusanov's local Lax-Friedrichs flux-splitting method [44]. The MWSB operator S maps the q-dimensional vector grid function $\mathbf{e}_h$ 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 $\boldsymbol{e} = \mathbf{e}_h$, 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 $S(\mathbf{e})$ (resp. the discretized operator $\widetilde{S}[\mathbf{e}_h]$ 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 $S(\mathbf{e}) = a$ (resp $\widetilde{S}[\mathbf{e}_h]_i = a_i$) for the 1D Linear Advection equation (2.2), $S(\mathbf{e}) = |u|$ (resp $\widetilde{S}[\mathbf{e}_h]_i = |u_i|$) for the 1D Burgers equation (2.3), and $S(\mathbf{e}) = |u| + a$ [8] (resp $\widetilde{S}[\mathbf{e}_h]_i = |u_i| + a_i$) in the case of the 1D Euler problem (2.5) (in terms of the sound speed (2.11)).

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

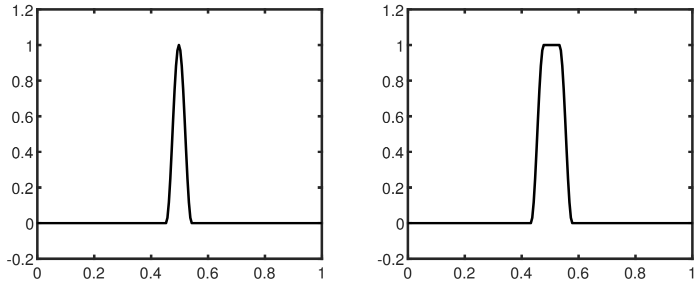


Figure 3.3: Left: Windowing function $W^j(x)$ (with $j = 100$, on the domain $[0,1]$, and using $N = 200$ discretization points) utilized in the definition of the windowed-localization operator Λ. Right: Windowing function $q_{18,9}(x - z)$ utilized for the localized filtering of the initial condition on the domain $[0,1]$ (using $N = 200$ discretization points) for a discontinuity located at $z = 0.5$.

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

$$q_{c,r}(x) = \begin{cases} 1 & \text{if} \quad |x| < ch/2 \\ \cos^2\left(\frac{\pi(|x|-ch/2)}{rh}\right) & \text{if} \quad ch/2 \le |x| \le (c/2+r)h \\ 0 & \text{if} \quad |x| > (c/2+r)h, \end{cases} \tag{3.14}$$

depicted in Figure 3.3 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 $\check{W}^j$ $(0 \le j \le N-1)$, 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

$$\check{W}_i^j = \frac{W_i^j}{\sum_{k=0}^{N-1} W_k^j}, \tag{3.15}$$

and

$$\Lambda[b]_i = \sum_{k=0}^{N-1} \check{W}_i^k b_k, \tag{3.16}$$

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

$$\tilde{\mu}[\mathbf{e}_h]_i = \Lambda[\widetilde{R}(\tilde{\tau}[\phi])]_i \cdot \max_{j \in L^i}(\widetilde{S}[\mathbf{e}_h]_j)h; \tag{3.17}$$

as mentioned in Chapter 1 and demonstrated in Section 3.2, 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 ($0 \le i \le N$) used in equation (3.17) were designed to differ from those defined in point (iii) of Section 3.1.2.1: 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 3.4, 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 3.2.3.1.) The left image displays the viscosity profiles used in [5] (which do not utilize the smoothing windows (3.16)) and the right image presents the window-based viscosity profile (3.17). 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 3.2.3.1. 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" [45, Sec. 5], which are commonly used as frequency filters in the field of signal processing. The normalized form used in this chapter, equation (3.15), 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 3.2.1.2, the localization stencils and the windowing functions are defined by $L^i = \left\{x_{m(i-3,N)}, \dots, x_{m(i,N)}, \dots x_{m(i+3,N)}\right\}$, where the *modulo* function m is defined in Section 3.1.2.1, and where $W_i^j = q_{c,r}(|x_j - x_{\widetilde{m}(i,j,N)}|)$. Here, for $s = \frac{c}{2} + r$ and $0 \le i, j \le N-1$, we have set

$$\widetilde{m}(i,j,N) = \begin{cases} j + N - i & \text{if} \quad j < 2s \text{ and } N - 2s + j \le i \le N - 1 \\ j - 1 - i & \text{if} \quad N - 1 - j < 2s \text{ and } i \le 2s - (N - j) \\ i & \text{else.} \end{cases} \tag{3.18}$$

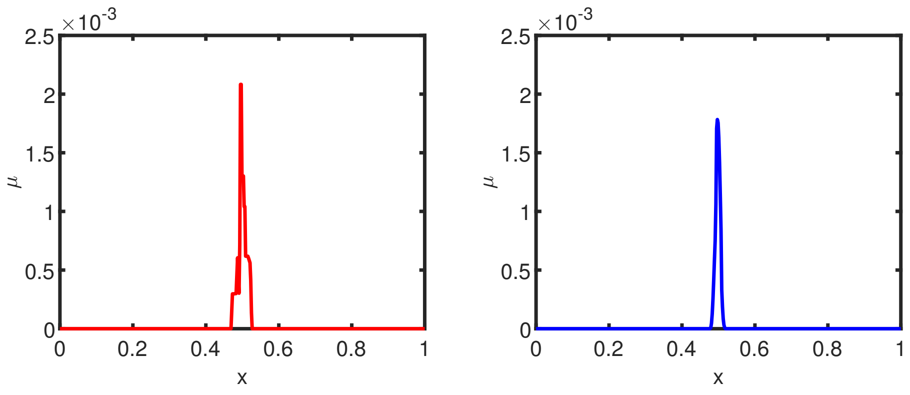


Figure 3.4: Comparison of the viscosity functions arising at the first viscous time-step for the Sod problem, using $N = 500$ discretization points. Left: viscosity used in [5]. Right: viscosity used in the present FC-SDNN method (equation (3.17)).

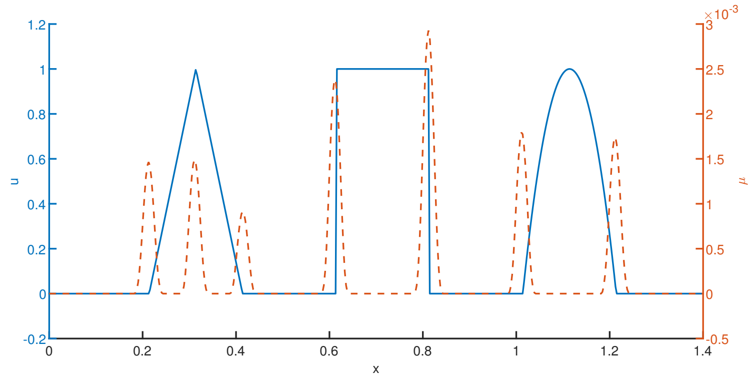


Figure 3.5: Viscosity assignment (orange dashed line) resulting from application of the SDNN-localized artificial viscosity algorithm to the initial condition (3.19) (blue solid line), using $N = 500$ discretization points.

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

As an example, Figure 3.5 displays the viscosity assignments produced, by the method described in this section for the function

$$u(x) = \begin{cases} 10(x-0.2) & \text{if } 0.2 < x \leq 0.3 \\ 10(0.4-x) & \text{if } 0.3 < x \leq 0.4 \\ 1 & \text{if } 0.6 < x \leq 0.8 \\ 100(x-1)(1.2-x) & \text{if } 1 < x \leq 1.2 \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

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.

3.1.3.2 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 3.3). 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 $S[\mathbf{e}]$ (resp. the discrete operator $\widetilde{S}[\mathbf{e}_h]$) is taken to equal the maximum characteristic speed, that is $S[\mathbf{e}] = |u|$ (resp. $\widetilde{S}[\mathbf{e}_h]_{ij} = |u_{ij}|$ for $(i, j) \in \mathcal{I}$). In the case of the 2D Euler problem, the MWSB operator S used in this thesis 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 $\mathbf{u} \cdot \vec{\kappa} + a$ [46, Sec. 16.3 and 16.5]), so that the discrete operator $\widetilde{S}$ we propose is given on the grid by

$$\widetilde{S}[\mathbf{e}_h]_{ij} = |u_{ij}| + |v_{ij}| + a_{ij}. \tag{3.20}$$

It is relevant to note that the MWSB operator (4.44), 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 [25, 26], where the (equivalent) Euclidean-norm of $\mathbf{u}$ was used instead. The two-dimensional local-window operator, finally, is given by

$$\Lambda[b]_{ij} = \sum_{(k,\ell)\in\mathcal{I}} \check{W}_i^k \check{W}_j^\ell b_{k\ell}, \tag{3.21}$$

which, clearly, can be obtained in practice by applying consecutively the 1D local window operator introduced in Section 3.1.3.1 in the horizontal and vertical directions consecutively. As in the 1D case, using these operators and functions, we define the 2D artificial viscosity operator

$$\tilde{\mu}[\mathbf{e}_h]_{ij} = \Lambda[\widetilde{R}(\tilde{\tau}_{xy}[\phi])]_{ij} \cdot \max_{(k,\ell)\in L^{ij}} (S[\mathbf{e}_h]_{k\ell})h. \tag{3.22}$$

3.1.4 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 chapter as well, in conjunction with FC [1, 2], as detailed in Section 4.2.3.1. Additionally, a new "localized" filtering strategy 4.2.3.2 is introduced in this chapter, 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.

3.1.4.1 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

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

the corresponding globally-filtered Fourier Continuation expansion is given by

$$\widetilde{F}_g = \sum_{k=-M}^{M} \hat{F}_k^c \sigma\Big(\frac{2k}{N+C}\Big) \exp(2\pi ikx/\beta) \tag{3.23}$$

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 chapter, 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.

3.1.4.2 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 $F : I \longrightarrow \mathbb{R}$, defined on a one-dimensional interval I, which is discontinuous at a single point $z \in I$. In this case, the smeared-discontinuity function $\widetilde{F}_{\text{sm}}$, which combines the globally filtered function $\widetilde{F}_g$ in a neighborhood of the discontinuity with the unfiltered function elsewhere, is defined by

$$\widetilde{F}_{\text{sm}}(x) = q_{c,r}(x-z)\widetilde{F}_g(x) + (1 - q_{c,r}(x-z))F(x). \tag{3.24}$$

Remark 4. Throughout this chapter, 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 3.3 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 $0 \leq j \leq N_2 - 1$. The resulting “partially” filtered function is then filtered along every vertical line $x = x_i$ for $0 \leq i \leq N - 1$ 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.

3.1.5 Algorithm pseudo-code

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

Algorithm 1 FC-SDNN algorithm

```
 1: \\Initialization.
 2: Input the trained ANN weights and biases (Section 3.1.2.2).
 3: Initialize the unknown solution vector $\mathbf{e}_h$ (Section 2.1) to the given initial-condition
    values over the given spatial grid.
 4: Initialize time: $t = 0$.
 5: **while** $t < T$ **do**
 6:     Evaluate the proxy variable $\phi$ corresponding to $\mathbf{e}_h$ at all spatial grid points.
 7:     Obtain the smoothness classification operator values ($\tilde{\tau}[\phi]$, equation (3.8), in the
    1D case, or $\tilde{\tau}_{xy}[\phi]$, equation (3.9), in the 2D case) at all grid points by applying steps (i)
    through (vi) in Section 3.1.2.1 as required in each case, 1D or 2D.
 8:     Evaluate the MWSB operator $\widetilde{S}[\mathbf{e}_h]$ at all spatial grid points (Section 3.1.3.1 in the
    1D cases, and Section 3.1.3.2 in the 2D cases).
 9:     Determine the artificial viscosity assignments $\tilde{\mu}[\mathbf{e}_h]$ (Equation (3.17) in the 1D case
    or equation (3.22) in the 2D case) at all spatial grid points.
10:     (Case $t = 0$, discontinuous initial data only) Apply localized discontinuity-smearing
    (Section 4.2.3.2) to the solution vector $\mathbf{e}_h$ and overwrite $\mathbf{e}_h$ with the resulting values.
11:     (Case $t > 0$) Apply global filtering (Section 4.2.3.1) to the solution vector $\mathbf{e}_h$ and
    overwrite $\mathbf{e}_h$ with the resulting values.
12:     Evaluate the temporal step-size $\Delta t$ by substituting the discrete version $\widetilde{S}[\mathbf{e}_h]$ and
    $\tilde{\mu}[\mathbf{e}_h]$ of $S[\mathbf{e}]$ and $\mu[\mathbf{e}]$ in equation (3.2).
13:     Perform the FC-based spatial differentiations required for time-stepping for given
    (Dirichlet or Neumann) boundary conditions (Sections 2.3 and 3.1.1) and use them to
    time-step the discrete version of the viscous system of equations (2.12)-(2.13), with
    $\mu[\mathbf{e}]$ substituted by $\tilde{\mu}[\mathbf{e}_h]$, in accordance with the SSPRK-4 time stepping scheme.
14:     Update time: $t = t + \Delta t$
15: **end while**
```

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 3.3. Additionally, we note a point of contrast with the related publication [5]: unlike the algorithm presented in Section 4 of that paper, the FC-SDNN use of the shifting procedure described in point (ii) in Section 3.1.2.1 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 $\mathbf{e}_h$, 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.

3.2 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 3.2.1.2, 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 chapter in view of the inefficiencies associated with the interpreter computer language used but, for reference, we note from [13] that, for the types of equations considered in this chapter, 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 chapter 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 [13], efficient implementations of the proposed FC-SDNN algorithm will prove highly competitive for general configurations.

3.2.1 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 3.2.1.1), and 2) essentially dispersionless character (Section 3.2.1.2). For the examples in this section FC expansions with $d = 5$ were used.

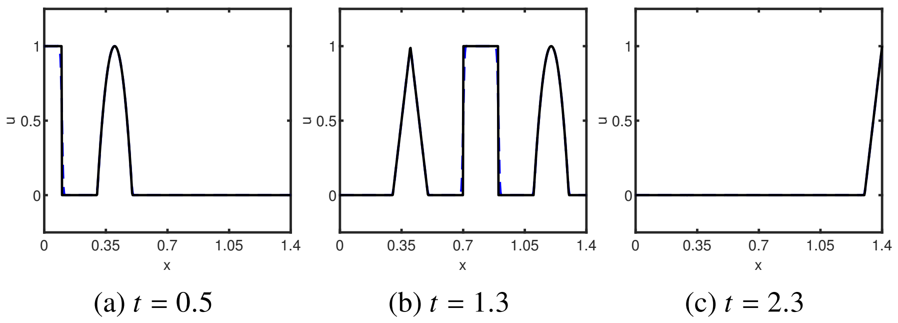
3.2.1.1 Boundary conditions

Figure 3.6 displays the FC-SDNN solution to the linear advection problem (2.2), 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

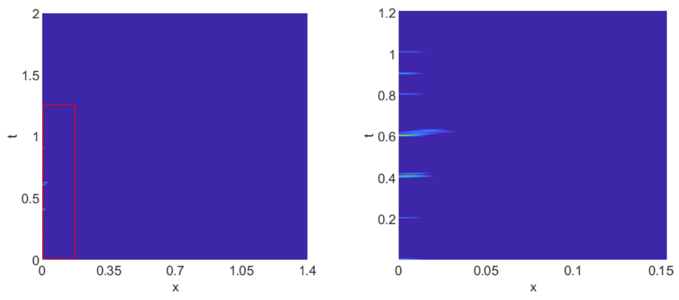
$u(x, 0) = 0$ and boundary condition at $x = 0$ given by

$$u(0,t) = \begin{cases} 100t(t-0.2) & \text{if } 0 < t < 0.2 \\ 1 & \text{if } 0.2 < t < 0.4 \\ 10(t-0.8) & \text{if } 0.8 < t < 0.9 \\ 1 - 10(t-0.9) & \text{if } 0.9 < t < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.25}$$

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 (3.2), with CFL = 2. As shown in Figure 3.6, 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 3.6d-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 3.6: 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.

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

$$\omega(x, q_1, q_2) = \begin{cases} \exp(2\frac{e^{-1/\xi}}{\xi-1}) & \text{if} \quad q_1 \leq |x| \leq q_2 \\ 1 & \text{if} \quad |x| \leq q_1 \\ 0 & \text{if} \quad |x| \geq q_2, \end{cases} \quad \text{where} \quad \xi = \frac{|x| - q_1}{q_2 - q_1}. \tag{3.26}$$

For this example we solve the equation (2.2) 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

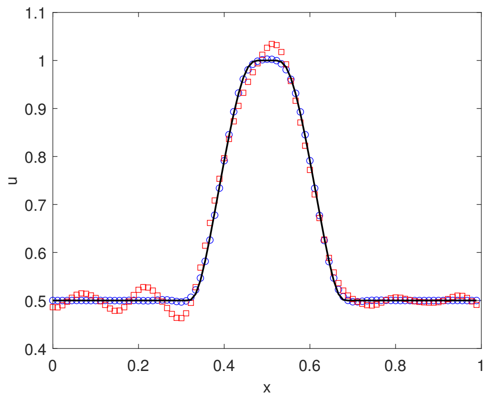


Figure 3.7: Numerical solutions to the periodic one-dimensional linear advection problem with $a = 1$ up to time $T = 500$: Exact solution (solid black line), fifth order FC-SDNN method (blue circles) and sixth order centered finite-difference scheme (red squares). Both numerical solutions were obtained 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.

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 3.7. 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 (3.2) 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.

3.2.2 Burgers equation

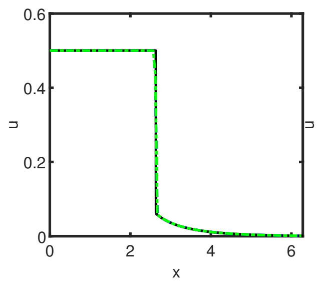
The 1D and 2D Burgers equation tests presented in this section demonstrate the FC-SDNN solver's performance for simple nonlinear non-periodic problems. In particular, the 2D example showcases the ability of the algorithm to handle multi-dimensional problems where shocks intersect domain boundaries (a topic that is also considered in Section 3.2.3.2 in the context of the Euler equations).

3.2.2.1 1D Burgers equation

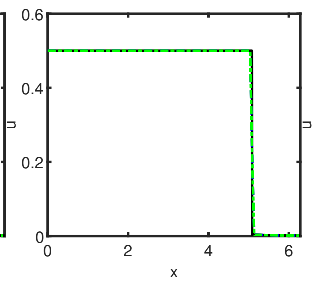
Propagating shock wave. Figure 3.8 displays solutions to the 1D Burgers equation (2.3), where a shock forms from the sharp features in the initial condition

$$u_0(x) = \frac{1}{\exp(x - \frac{3}{20})[\tanh(10x - 3) + 1] - \tanh(10x - 3) + 1}. \tag{3.27}$$

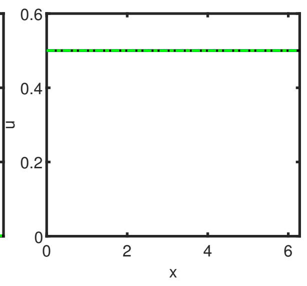
Results of simulations produced by means of the FC-SDNN and FC-EV algorithms are presented in the figure. In both cases Dirichlet boundary conditions at the inflow boundary $x = 0$ were used, while the solution at the outflow boundary $x = 2\pi$ was evolved numerically in the same manner as the interior domain points. The FC-EV viscosity parameters defined in equations (2.17) and (2.18) were set to $c_{max} = 0.2$ and $c_E = 0.1$; cf. Table 3.2. 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.



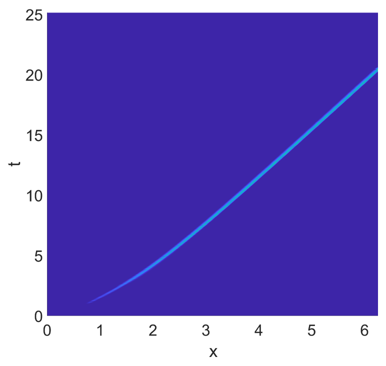
(a) $t = 2\pi$



(b) $t = 5\pi$



(c) $t = 8\pi$



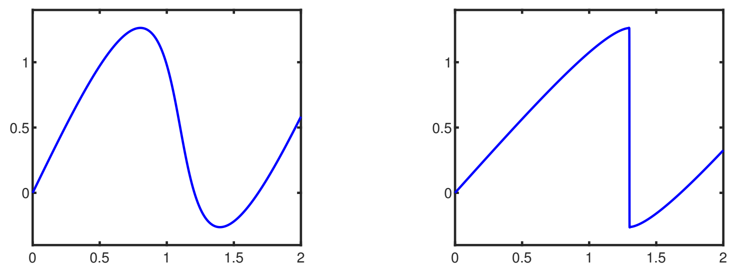
(d) Time history of FC-SDNN artificial viscosity.

Figure 3.8: 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 (2.3) in the domain $[0, 2]$, with an initial condition given by

$$u_0(x) = \frac{1}{2}x + \sin(\pi x) \tag{3.28}$$

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 3.9a and b. As demonstrated in Figure 3.10, 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 3.9a 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 3.9: 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 $N = 6400$.

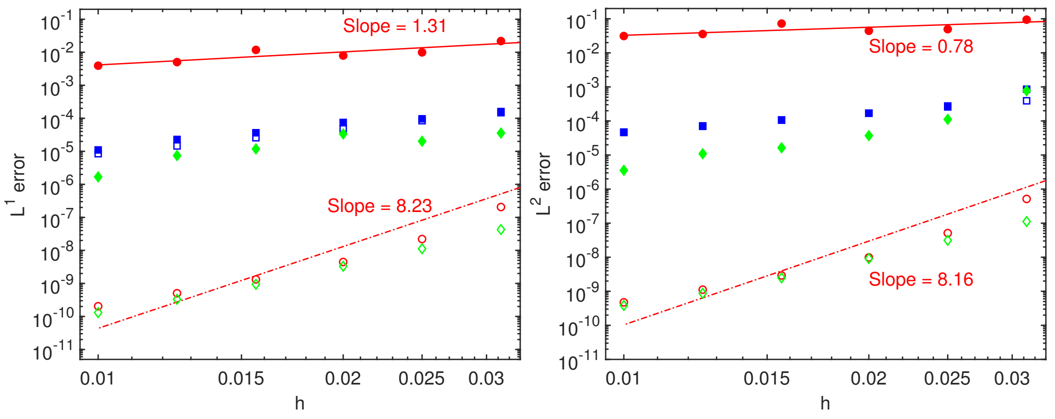


Figure 3.10: Convergence of the FC-SDNN solution for the Burgers 1D test in the L^1 and L^2 norms (left and right panels, respectively), before ($t = 0.1$) and after ($t = 0.6$) the shock forms (empty and full symbols, respectively). The reference data was obtained by solving the exact implicit form of the solution [47, 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 $d = 5$.

3.2.2.2 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 (2.4) on the domain $\mathcal{D} = [0, 1] \times [0, 1]$, with an initial condition given by the function

$$u_0(x) = \begin{cases} -1 & \text{if} \quad x \in [0.5, 1] \text{ and } y \in [0.5, 1] \\ -0.2 & \text{if} \quad x \in [0, 0.5] \text{ and } y \in [0.5, 1] \\ 0.5 & \text{if} \quad x \in [0, 0.5] \text{ and } y \in [0, 0.5] \\ 0.8 & \text{if} \quad x \in [0.5, 1] \text{ and } y \in [0, 0.5], \end{cases} \tag{3.29}$$

and with vanishing normal derivatives at the boundary. This problem admits an explicit solution (displayed in Figure 3.11b 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 3.11c, 3.11d, and 3.11e 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 (3.2) 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 4.2.3.2, 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 3.12, and per the discussion in Section 3.2.2.1, the expected convergence of orders h and $h^{\frac{1}{2}}$ in the L^1 and L^2-norms respectively are obtained.

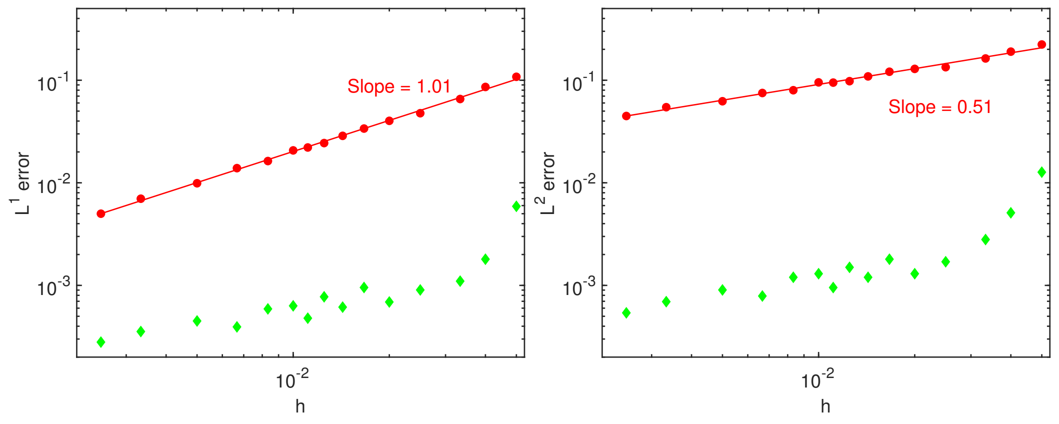
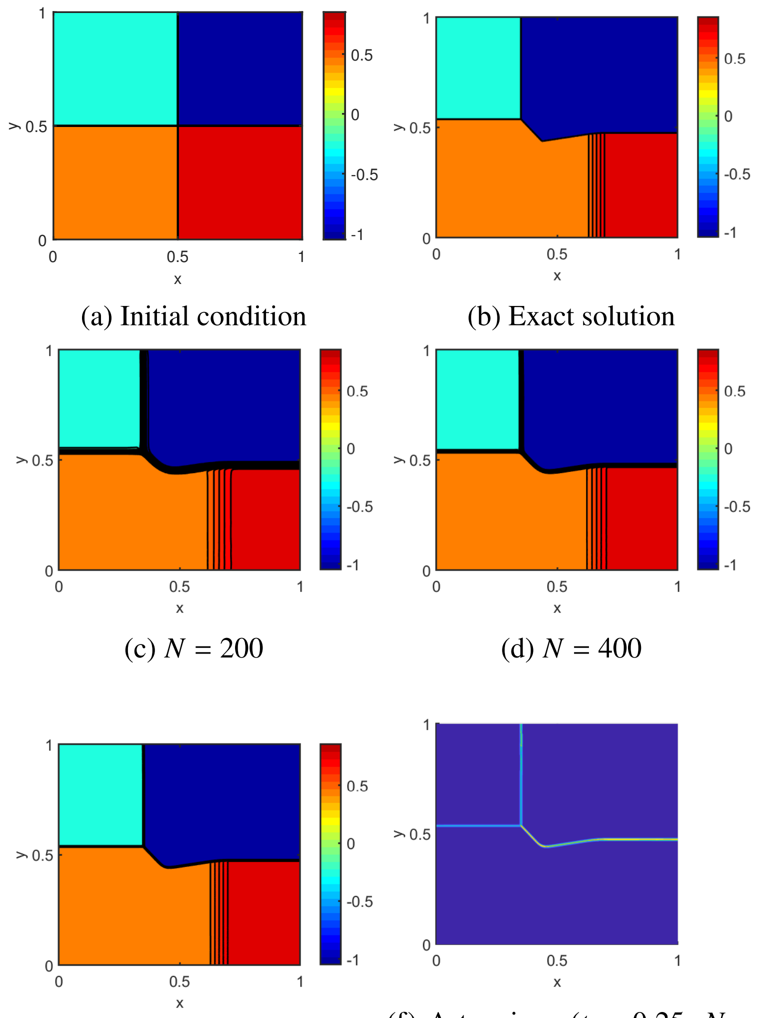


Figure 3.12: Convergence of the FC-SDNN solution for the Burgers 2D test at $t = 0.25$ in the L^1 and L^2 norms (left and right panels, respectively), using FC order $d = 5$. Red dots: errors on the complete domain. Green diamonds: errors away from the shocks ($(x, y) \in ([0, 0.4] \times [0, 0.35]) \cup ([0, 0.25] \times [0, 0.6]) \cup ([0.6, 1] \times [0.6, 1])$.)

3.2.3 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 [48], 2D Riemann problem flow [49], Mach 3 forward facing step [50], and Double Mach reflection [50]. 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.



(a) Initial condition (b) Exact solution

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

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

.

Figure 3.11: 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.

3.2.3.1 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 3.2.2) but also contact discontinuities. Fortunately, in view of the localized spectral filtering strategy used for the initial-data (Section 4.2.3.2), 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 a diverging rarefaction waves [51], the Sod [52], Lax [53], Shu-Osher [54] and Blast Wave [30] 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 [46, 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 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 3.2. For each test case and each algorithm (FC-SDNN and FC-EV), the value of the constant CFL in (3.2), which is included as part of each description, was selected so as to obtain the largest time-step Δt which preserves stability.

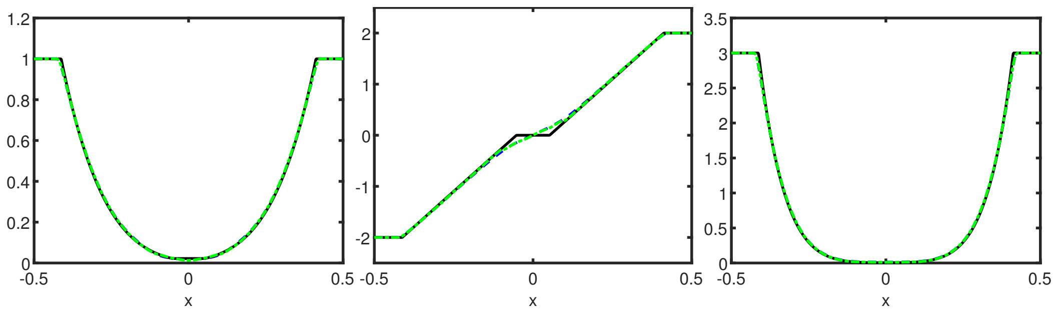
Problem	Burgers 1D	Div. raref.	Sod	Lax	Shu-Osher	Blast Wave
c_{max}	0.2	0.1	0.1	0.15	0.85	1
c_E	0.1	10	15	20	10	0.05

Table 3.2: FC-EV parameters for the 1D Burgers and 1D Euler problems.

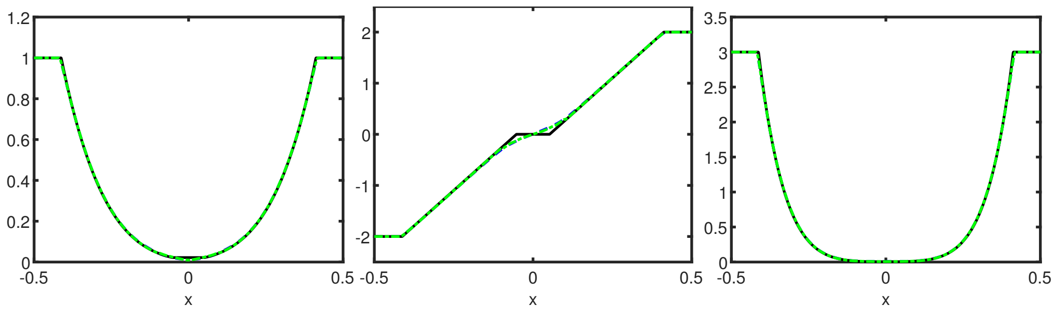
Diverging rarefaction waves. We first consider the problem of two rarefaction waves traveling in opposite directions for the 1D Euler equations (2.5) on the interval $[-0.5, 0.5]$ with initial conditions

$$(\rho, u, p) = \begin{cases} (1, -2, 0.4) & \text{if} \quad x < 0 \\ (1, 2, 0.4) & \text{if} \quad x > 0. \end{cases}$$

This simulation highlights the difficulty that arises as the equal but opposite velocities (i.e., the diverging rarefactions waves) result in a region of a very low density and pressure around the middle of the interval. The solution was computed up to $T = 0.15$ by means of both, the FC-SDNN and the FC-EV solvers—for which



(a) N = 500



(b) N = 1000

Figure 3.13: Solutions to the "diverging rarefaction waves" problem produced by the FC-SDNN and FC-EV algorithms of order $d = 5$ at $t = 0.15$, using $N = 500$ and $N = 1000$ discretization points (upper panels and lower panels, respectively). Left, middle and right panel-pairs display the density, velocity and energy, respectively. Exact solution: solid black line. FC-SDNN solution: Blue dashed-line. FC-EV solution: Green dot-dashed line.

adaptive time steps (3.2) were used with CFL = 3 and CFL = 2, respectively. The results presented in Figure 3.13 show well resolved rarefaction waves, with a low, but positive density and pressure values for the both FC-SDNN and FC-EV algorithms. Notably, the energy component of the solution accurately approximates the exact solution on the complete interval for both the FC-SDNN and FC-EV algorithms. (As is well known, in contrast, the energy values show significant deviations from the exact solution around the middle of the interval for certain types of Godunov schemes (e.g., the one based on certain Riemann solvers such as Osher's (see [51, Sec 2.5]), which result from numerical errors at low density and pressures.)

Sod problem. We consider a Sod shock-tube problem for the 1D Euler equations (2.5) 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 3.14(a) and (b)). The solution was computed up to time $T = 2$. The results presented in Figure 3.14 show well resolved shocks (upper and middle right) and contact discontinuities (center images in Figure 3.14(a) and (b)), with no visible Gibbs oscillations in any case. The FC-SDNN and FC-EV solvers (for which adaptive time steps (3.2) 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 (3.14c), 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

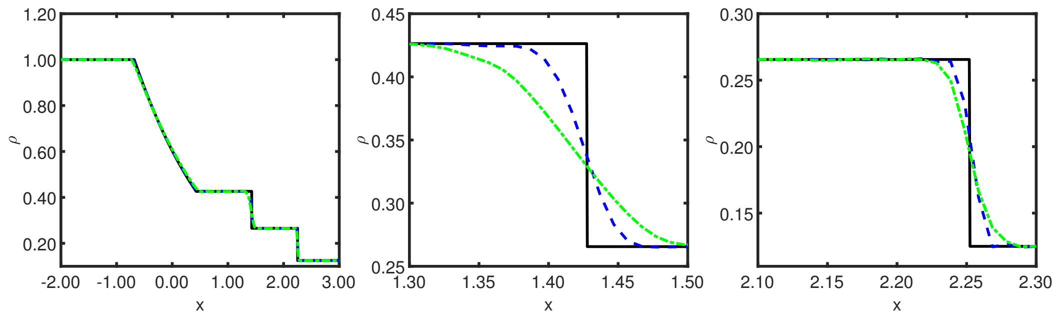
$$(\rho, u, p) = \begin{cases} (0.445, 0.698, 3.528) & \text{if} \quad x < 0 \\ (0.5, 0, 0.571) & \text{if} \quad x > 0 \end{cases}$$

which results in a combination (from right to left) of a shock wave, a contact discontinuity and a rarefaction wave (left images in Figure 3.15(a) and (b)). The solution was computed up to time $T = 1.3$. The results are presented in Figure 3.15, which shows well resolved shocks (upper and middle right) without detectable Gibbs oscillations. The viscosity time history displayed in Figure 3.15c 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 3.15(a) and (b)). The FC-EV and FC-SDNN algorithms, (for which adaptive time steps (4.38) 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.

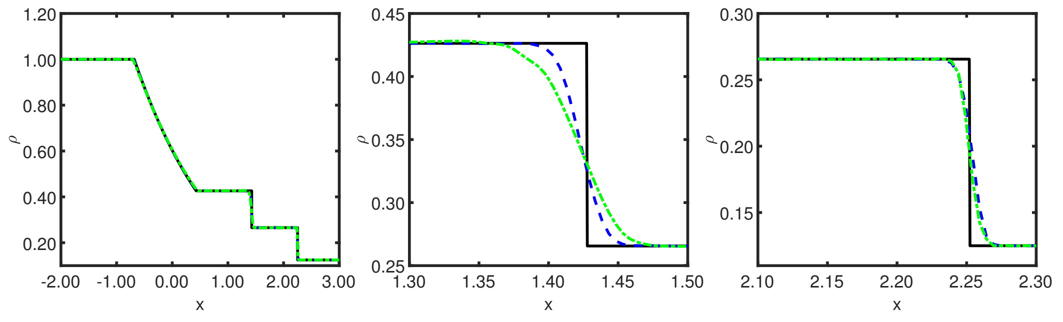
Shu-Osher problem. We consider the Shu-Osher shock-entropy problem on the interval $[-5, 5]$, with initial condition given by

$$(\rho, u, p) = \begin{cases} (3.857143, 2.6929369, 10.33333) & \text{if} \quad x < -4 \\ (1 + 0.2\sin(5x), 0, 1) & \text{if} \quad x > -4. \end{cases} \tag{3.30}$$

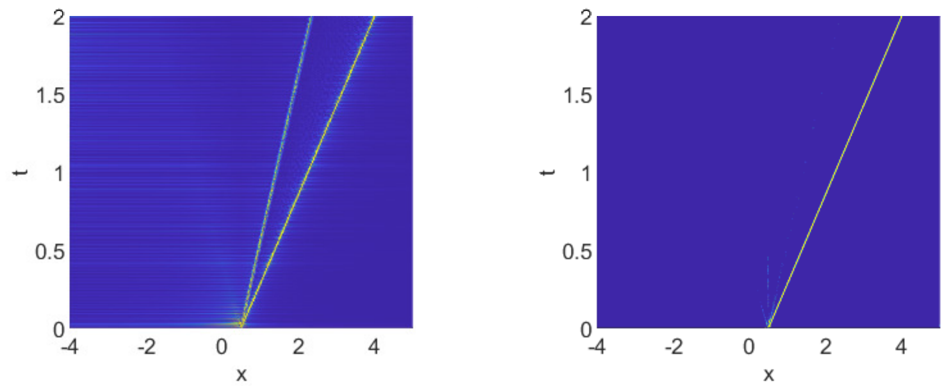
The solution is computed up to time $T = 1.8$. In this problem, a shock wave encounters an oscillatory smooth wavetrain. This test highlights the FC-SDNN



(a) N = 500

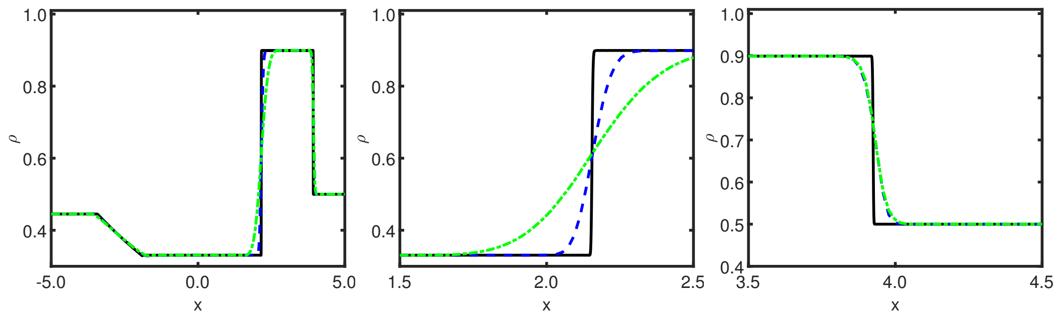


(b) N = 1000

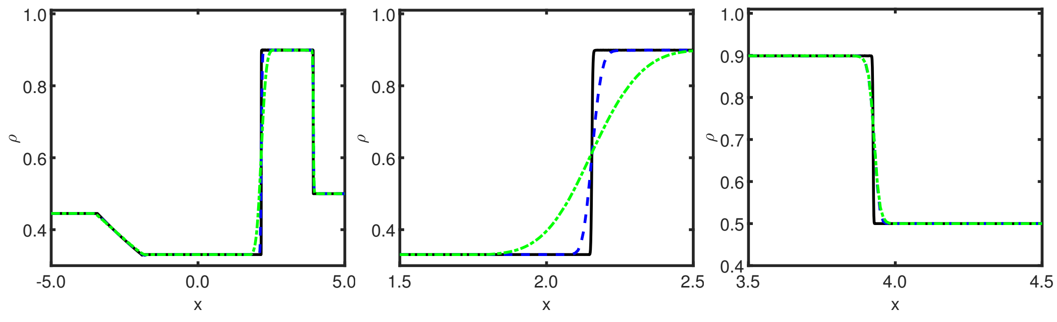


(c) Time history of assigned artificial viscosity for N = 1000, for FC-EV (left) and FC-SDNN (right).

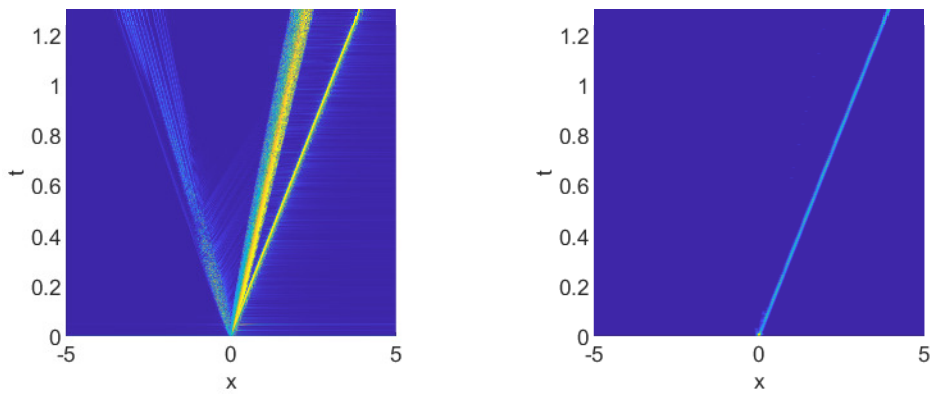
Figure 3.14: Solutions to the Sod problem produced by the FC-SDNN and FC-EV algorithms of order $d = 5$ at $t = 0.2$. Exact solution: solid black line. FC-SDNN solution: Blue dashed-line. FC-EV solution: Green dot-dashed line. Numbers of discretization points: $N = 500$ in the upper panels and $N = 1000$ 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 3.15: 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.

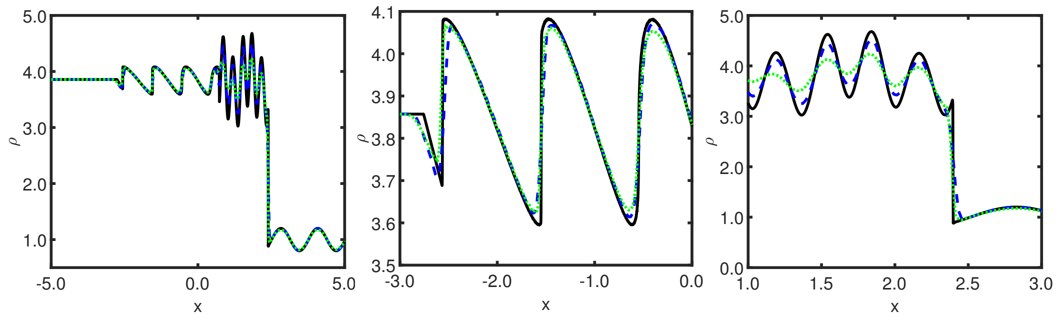
solver's low dissipation, as artificial viscosity is only assigned in the vicinity of the right-traveling shock as long as the waves remain smooth, allowing for an accurate representation of the smooth features. In particular, Figure (3.16c) shows that the support of the FC-SDNN artificial viscosity 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 3.16(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 (4.38) were used with CFL = 3 and CFL = 4, respectively).

Blast Wave problem. Finally, we consider the Blast Wave problem as presented in [30], on the interval $[0, 1]$, with initial conditions given by

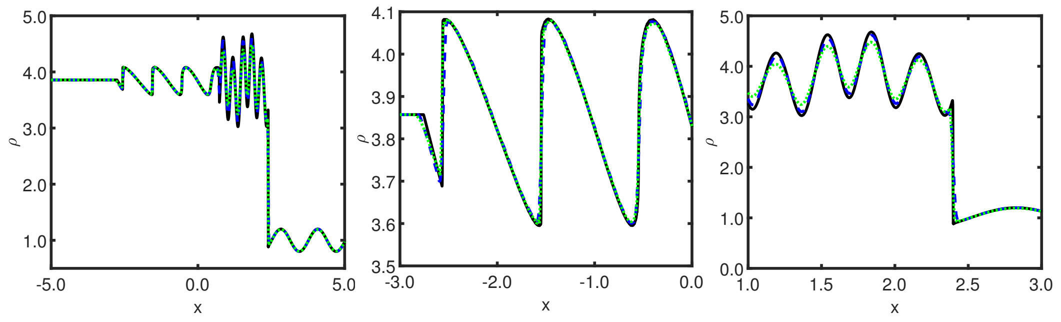
$$(\rho, u, p) = \begin{cases} (1, 0, 1000) & \text{if} \quad x < 0.5 \\ (1, 0, 0.01) & \text{if} \quad x > 0.5, \end{cases} \tag{3.31}$$

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 (3.2) were used with CFL = 2 and CFL = 2, respectively) provide a sharp resolution of the shock, as shown in Figure 3.17. 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.

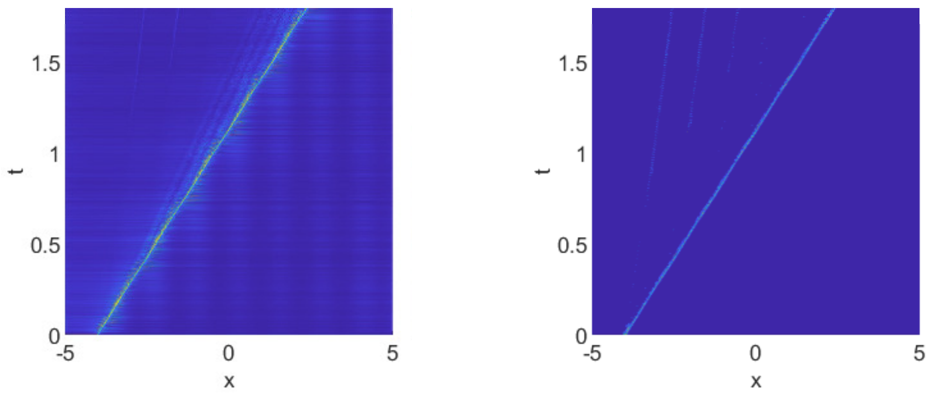
Remark 6. As shown in the left panel of Figure 3.17, the FC-EV method assigns a non-vanishing artificial viscosity everywhere ahead of the shock for this example. This undesirable feature results from the term $\frac{\partial \eta}{\partial t}$ in the entropy residual (2.15) which, in view of the relation $\eta(\mathbf{e}) = \frac{\rho}{\gamma - 1} \log(p/\rho^\gamma)$ 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.



(a) N = 500

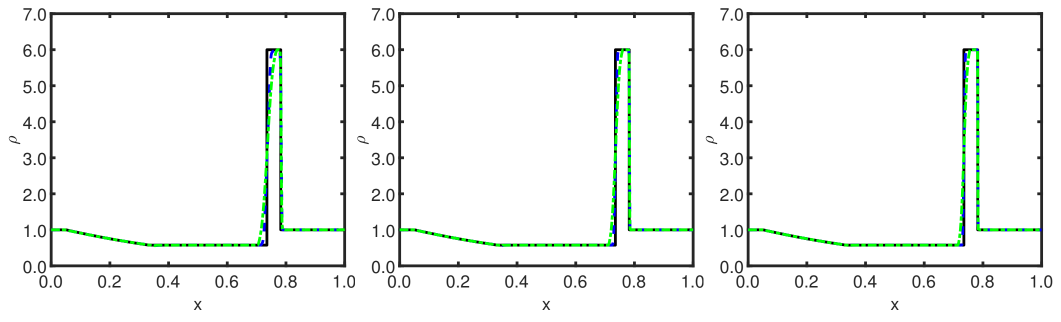


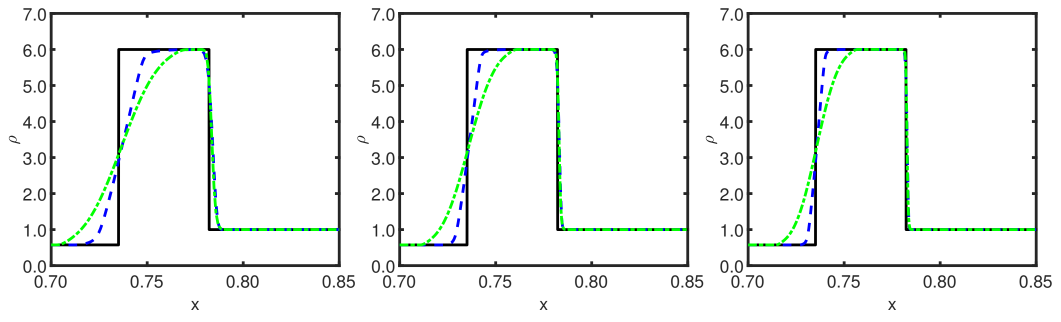
(b) N = 1000



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

Figure 3.16: 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"

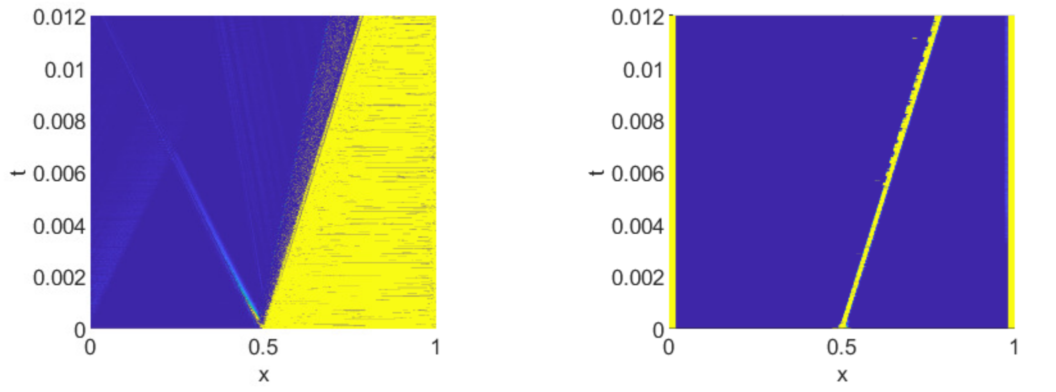

(c) Time history of assigned artificial viscosity for $N = 1000$, for FC-EV (left) and FC-SDNN (right).

Figure 3.17: 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.

3.2.3.2 2D Euler problems

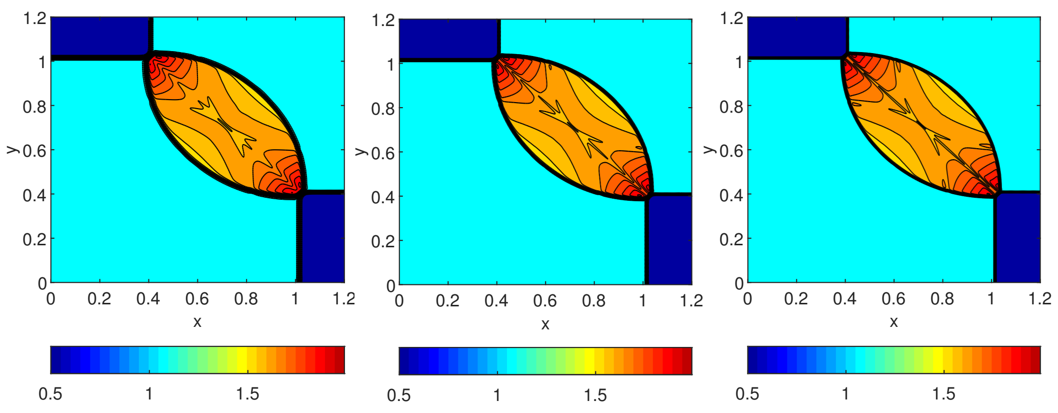
The test cases considered in this section showcase the FC-SDNN method's ability to handle complex shock-shock, as well as shock-boundary interactions, including shocks moving orthogonally to the boundaries as in the Riemann 2D and the Shock vortex problems, or moving obliquely to the boundary of the 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 [50, 49, 25, 55, 56]. As indicated in the introduction, the proposed approach leads to smooth flows away from shocks, as evidenced by correspondingly smooth contour levels for the various flow quantities, in contrast with corresponding results provided by previous methods.

Riemann4 problem (Riemann problem, configuration 4 in [49]). We consider a Riemann problem configuration on the domain $[0, 1.2] \times [0, 1.2]$, with initial conditions given by

$$(\rho, u, v, p) = \begin{cases} (1.1, 0, 0, 1.1) & \text{if} \quad x \geq 0.6 \text{ and } y \geq 0.6 \\ (0.5065, 0.8939, 0, 0.35) & \text{if} \quad x < 0.6 \text{ and } y \geq 0.6 \\ (1.1, 0.8939, 0, 0.35) & \text{if} \quad x < 0.6 \text{ and } y < 0.6 \\ (0.5065, 0, 0.8939, 0.35) & \text{if} \quad x \geq 0.6 \text{ and } y < 0.6, \end{cases} \tag{3.32}$$

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 3.18, 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 3.19, the FC-SDNN viscosity assignments are sharply concentrated near the shock positions.

Shock vortex problem [48]. 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.



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

Figure 3.18: Second-order ($d = 2$) FC-SDNN numerical solution to the Euler 2D Riemann problem considered in Section 3.2.3.2, 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 $\rho = 1.99$.

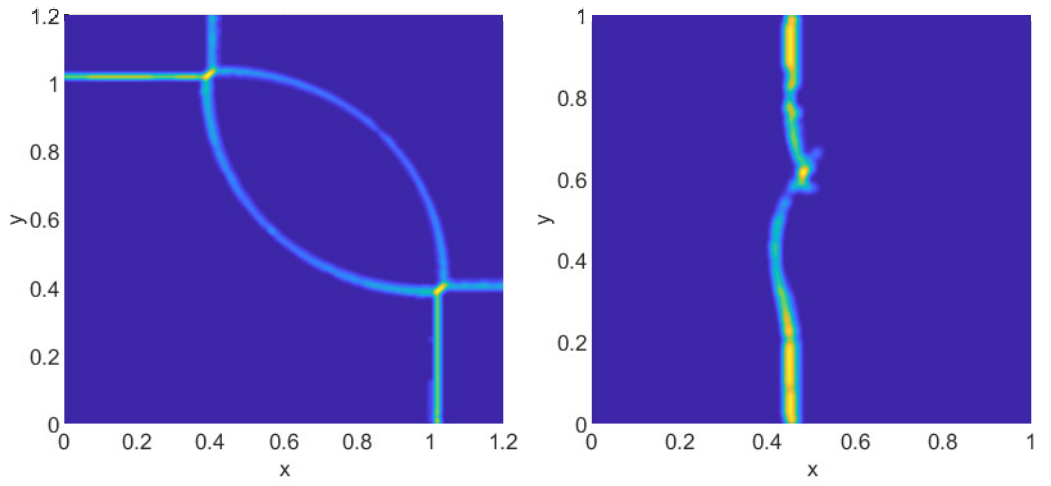


Figure 3.19: Artificial viscosity profiles for the Euler 2D Riemann problem considered in Figure 3.18 at $t = 0.25$ (left) and for the Shock vortex problem considered in Figure 3.20 at $t = 0.35$ (right).

The initial conditions are given by

$$(\rho, u, v, p) = \begin{cases} (\rho_L + \tilde{\rho}, u_L + \tilde{u}, v_L + \tilde{v}, p_L + \tilde{p}) & \text{if} \quad x \in [0, 0.5) \\ (\rho_R, u_R, v_R, p_R) & \text{if} \quad x \in [0.5, 1] \end{cases} \tag{3.33}$$

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

$$\begin{aligned} \tilde{u} &= \frac{x - x_c}{r_c}\Phi(r), & \tilde{v} &= -\frac{x - x_c}{r_c}\Phi(r), \\ \frac{\gamma - 1}{4\zeta\gamma}\Phi(r)^2 &= \frac{p_L}{\rho_L} - \frac{p_L + \tilde{p}}{\rho_L + \tilde{\rho}}, & p_L + \tilde{p} &= (\rho_L + \tilde{\rho})^\gamma, \end{aligned} \tag{3.34}$$

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

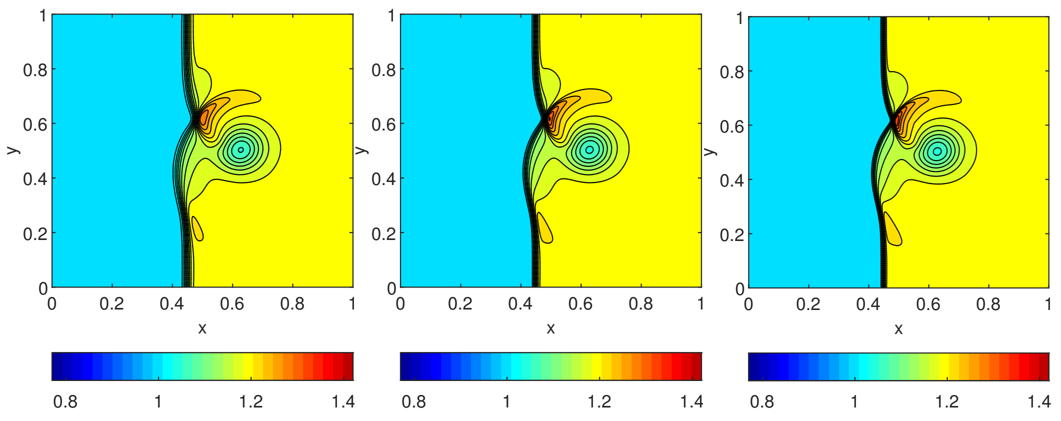
$$\begin{aligned} \rho_R &= \rho_L \frac{(\gamma + 1)p_R + \gamma - 1}{(\gamma - 1)p_R + \gamma + 1}, & u_R &= \sqrt{\gamma} + \sqrt{2}\frac{1 - p_R}{\sqrt{\gamma - 1 + p_R(\gamma - 1)}}, \\ v_R &= 0, & p_R &= 1.3. \end{aligned}$$

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 3.20 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 3.19 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 [50]. We now consider a "Mach 3 forward facing step problem" on the domain $\Omega = ([0, 0.6] \times [0, 1]) \cup ([0.6, 3] \times [0.2, 1])$, in which a uniform Mach 3 flow streams through a wind tunnel with a forward facing step, of 0.2 units in height, located at $x = 0.6$, whose vertical boundary we denote by $\Gamma_s = \{0.6\} \times [0, 0.2]$. The initial condition is given by

$$(\rho, u, v, p) = \begin{cases} (1.4, 3, 0, 1) & \text{if} \quad (x, y) \in \Omega \setminus \Gamma_s \\ (1.4, 0, 0, 1) & \text{if} \quad (x, y) \in \Gamma_s, \end{cases} \tag{3.35}$$



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

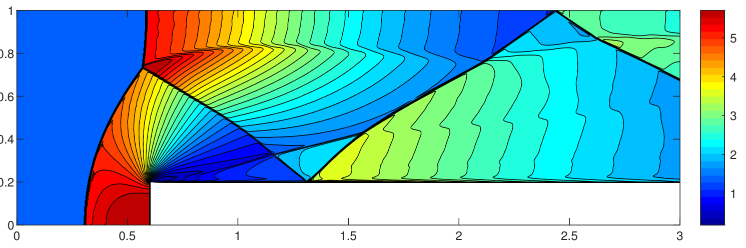
Figure 3.20: Second-order ($d = 2$) FC-SDNN numerical solution to the Euler 2D Shock-vortex problem considered in Section 3.2.3.2, at $t = 0.35$, obtained by using a spatial discretization containing $N \times N$ grid points with three different values of N, as indicated in each panel. For each discretization, the solution is represented using thirty equispaced contours between $\rho = 0.77$ and $\rho = 1.42$.

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 [50, 25], no boundary condition is enforced at the node located at the step corner.) The simulation was performed using the adaptive time step (3.2) with CFL = 1. The density solution is displayed in Figure 3.21.

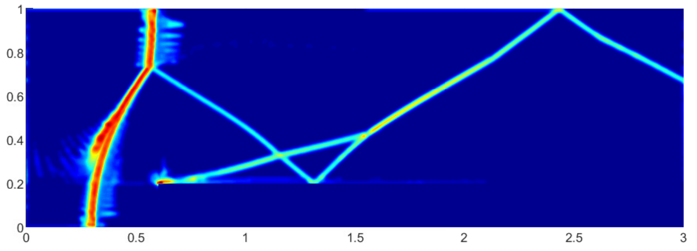
Double Mach reflection [50]. We then consider the "Double Mach reflection problem" on the domain $\Omega = [0, 4] \times [0, 1]$. This problem contains a reflective wall located on the $x \geq x_r$ part of the bottom boundary $y = 0$ (here we take $x_r = 1/6$), upon which there impinges an incoming shock wave forming a $\theta = \pi/3$ angle with the positive x-axis. Denoting by $\Omega_s := \{(x, y) \in \Omega | 0 \leq x \leq x_r + \frac{y}{\sin(\theta)}\}$ (resp. $\Omega_u := \{(x, y) \in \Omega | x_r + \frac{y}{\sin(\theta)} \leq x \leq 4\}$) the shocked (resp. unshocked) region at time $t = 0$, the initial condition $\mathbf{e}(x, y, 0)$ is given by

$$(\rho, \rho u, \rho v, E) = \begin{cases} (8, 57.1597, -33.0012, 563.544) & \text{if} \quad (x, y) \in \Omega_s \\ (1.4, 0, 0, 2.5) & \text{if} \quad (x, y) \in \Omega_u \end{cases} \tag{3.36}$$

and the solution is computed up to time $T = 0.2$. This setup, introduced in [50], 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



(a) Thirty equispaced ρ contours between 0.18 and 5.71



(b) Artificial viscosity at time t=4.

Figure 3.21: Top panel: Second-order ($d = 2$) FC-SDNN numerical solution to the Mach 3 forward facing-step problem considered in Section 3.2.3.2, at $t = 4$, obtained by using 1200×400 spatial grid. Bottom panel: Viscosity assignment.

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.

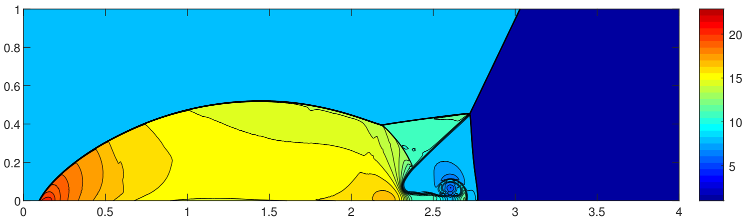
The initial and boundary values used in this context do not exactly coincide with the ones utilized in [50]. Indeed, on one hand, in order to avoid density oscillations near the intersection of the shock and the top computational boundary, we utilize "oblique" Neumann boundary conditions on all flow 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 [57], 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.

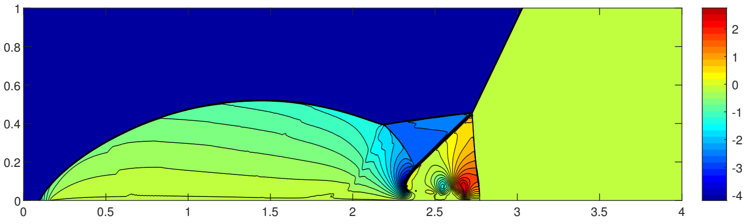
Additionally, following [57], we utilize a numerical viscous incident shock as an initial condition in order to avoid well-known post-shock oscillations, as noted in [58], that result from the use of a sharp initial profile. Such a smeared shock is obtained in our context by applying the FC-SDNN solver 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, letting U_S denote the theoretical value of the incident shock speed ($U_S \approx 10$ for the initial condition (3.36)) considered in the present context) we have set $x_s(y, t) = x_r + \frac{U_S}{\sin(\theta)} t + \frac{y}{\tan(\theta)}$. This shock is followed by some back-trailing oscillations, as has been observed in [58, 57]. In order to eliminate these artifacts from the initial condition, a diagonal strip of N_d points centered around the shock location is utilized and slightly recentered, as described in what follows.

In detail, the shock values on a strip of discretization points that is parallel to the shock and contains N_d discretization points along each discretization line parallel to the x axis around the shock are selected, and the backtrailing oscillatory artifacts are discarded. The integer N_d, which depends on the meshsize h, is taken to be small enough to exclude the aforementioned back-trailing artifacts from the strip, but large enough as to include the complete structure of the actual numerical shock. The resulting discrete shock structure is then used as the incident shock. In order to avoid numerical errors that would arise if the numerical incident shock were placed exactly at the edge of the reflecting wall at the initial time the initial discrete shock structure is taken to equal a translation of the numerical shock to a position slightly behind the initial shock position (by a distance $d_0 = \frac{N_d - 1}{2} h$ in the horizontal direction), so that the initial condition $\mathbf{e}_h$ is taken to equal

$$\begin{aligned} \mathbf{e}_h(x, y, 0) = q_{c,r}(x_s(y, 0) - d_0 - x)\hat{\mathbf{e}}_h(x + d_0 - x_s(y, 0) + x_s(y, T), y, T) \\ + (1 - q_{c,r}(x_s(y, 0) - d_0 - x))\mathbf{e}(x + d_0, y, 0), \end{aligned} \tag{3.37}$$

where $q_{c,r}$ is the window function defined in (4.41). To account for the delay arising from the aforementioned translation in the initial shock location, the final simulation time is taken to be $\tilde{T} = T + t_0$, where $t_0 = \frac{d_0 \sin(\theta)}{U_S}$ denotes the time needed for the translated oblique shock to reach the edge of the reflecting wall. The incident-shock mesh-size dependent parameter values $N_d = \text{floor}(\frac{125}{800h})$, $c = \text{floor}(\frac{25}{800h})$ and $r = \text{floor}(\frac{50}{800h})$ (where for $t \in \mathbb{R}^+$, $\text{floor}(t)$ denotes the integer smaller than t that is closest to t) were used to produce the double-Mach solution depicted in Figures 3.22


(a) Thirty equispaced ρ contours between 1.0 and 22.8


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

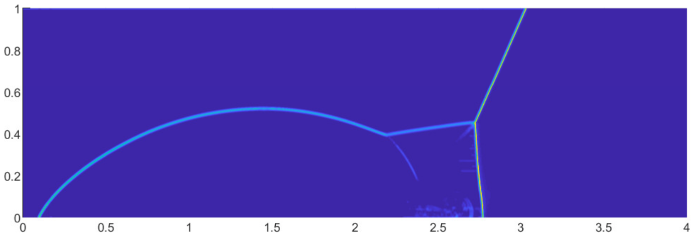

(c) Artificial viscosity at time $t = 0.2$

Figure 3.22: Top and middle panels: Second-order ($d = 2$) FC-SDNN numerical solution to the Double Mach reflection problem considered in Section 3.2.3.2, at $\tilde{T} \approx 0.2065$, obtained by using 3200×800 spatial grid. Bottom panel: Viscosity assignment.

through 3.25. It is to be noted that, in view of these definitions, the time t_0 depends slightly on the mesh discretization. Thus, for the discretizations considered in this thesis, namely 400×1600, 600×2400, 800×3200 and 1200×4800 (see Figure 3.24), the corresponding t_0 values are, respectively, $t_0 \approx 0.0065$, $t_0 \approx 0.0065$, $t_0 \approx 0.0067$, $t_0 \approx 0.0066$.

Among several notable features in the $t = \tilde{T}$ solution (depicted for two different resolutions in Figure 3.22(a) and in the bottom panel of Figure 3.24) we mention the clearly visible density-contour roll-ups of the primary slip line in the density component of the solution (highlighted in Figure 3.23a) as well as the extremely weak secondary slip line, that is most easily noticeable in the variable v (Figure 3.22b) as a dip in the contour lines highlighted in Figure 3.23b but which is also visible in this area as a blip in the density contour lines (Figure 3.22a). The accurate simulation of these two features has typically been found challenging [50, 59, 57]. The elimination of back-trailing shock oscillations, which was accomplished, as

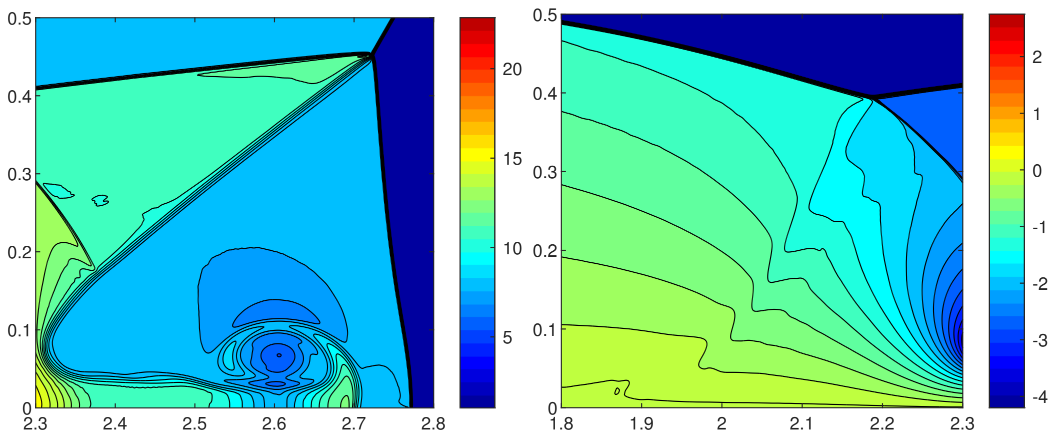


Figure 3.23: Zoom-in on two features displayed in Figure 3.22: (a) Density-contour roll-ups in Figure 3.22(a); and, (b) Secondary slip line in Figure 3.22(b).

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.

Self-similarity and KH instability in the Double Mach reflection problem. It is relevant to highlight two important features in the FC-SDNN solution just considered for the double-Mach problem, namely, its scale invariance and its rendering of the associated Kelvin-Helmholtz instability (KH) along the double-Mach primary slip line and associated roll-up, as discussed in what follows.

For certain geometric and initial flow settings the Euler equations admit self-similar solutions [60]. This is the case, in particular, for the Mach and Double Mach reflection problems [61, 62], with self-similarity centered at the wedge's tip $(x_r, 0)$ and at the time t_0 at which the incoming shock impinges upon the tip. More precisely, the change of variables

$$\xi = \frac{x - x_r}{t - t_0} \qquad \eta = \frac{y}{t - t_0}, \tag{3.38}$$

embodies the scale-invariance property of the problem under consideration. As illustrated in Figure 3.25, which presents scaled versions of numerical Double-Mach solutions at various times and for various discretization levels, the FC-SDNN Double-Mach solutions amply satisfy the self-similarity property—within the associated numerical error, which, most notably for the smaller times, is amplified in the self-similar blow-up process—as is evident by consideration of the vast variations

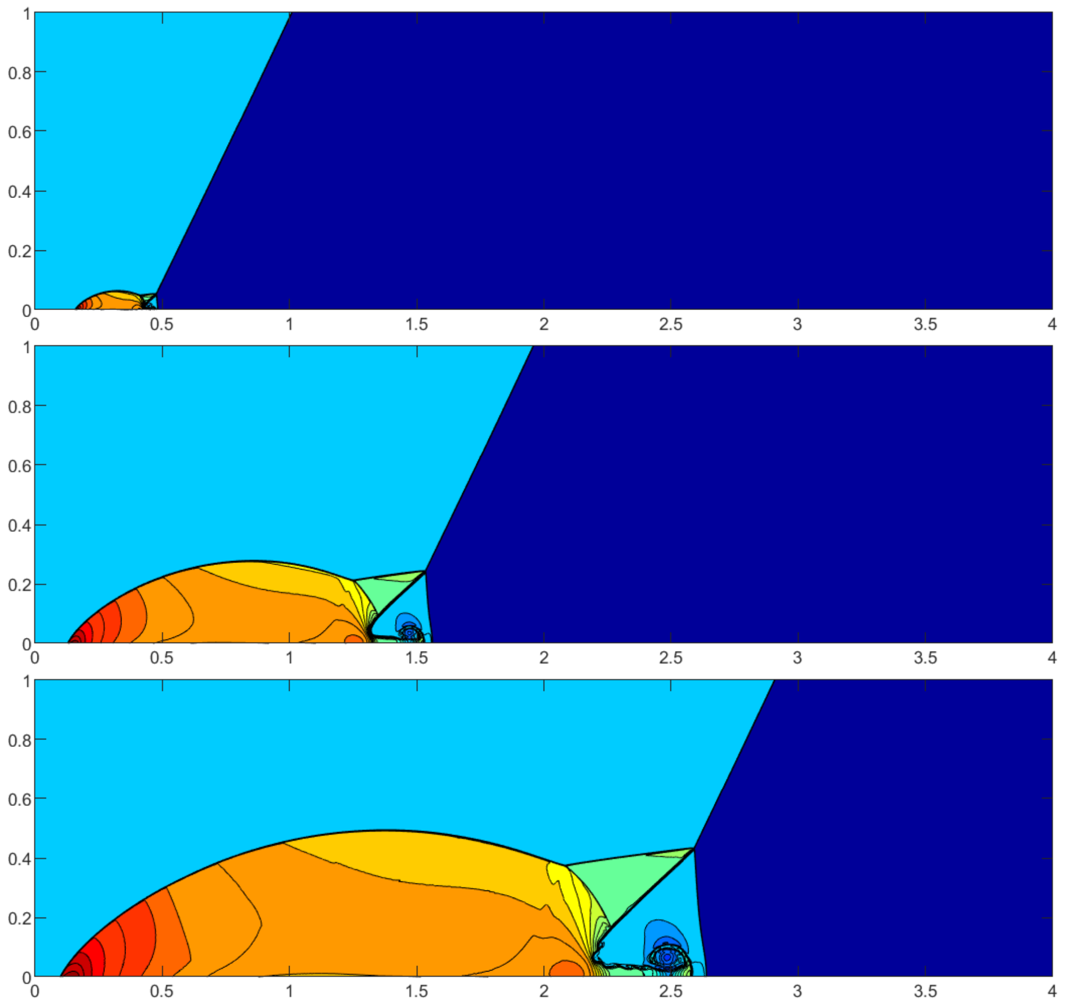


Figure 3.24: Double Mach density profile obtained by means of a 4800×1200 spatial spatial grid, at times $t \approx 0.0207$ (top), $t \approx 0.1033$ (middle) and $t \approx 0.2066$ (bottom). Note that e.g. at the earliest time the self-similar double-Mach features are effectively resolved by a much coarser grid than at later times.

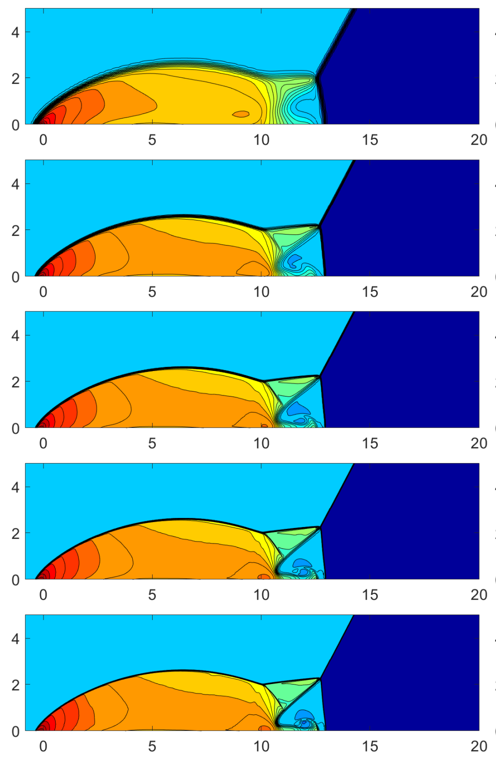
in the shock widths in Figure 3.25, which result from adequate blow-up of images such as the ones presented in Figure 3.24.

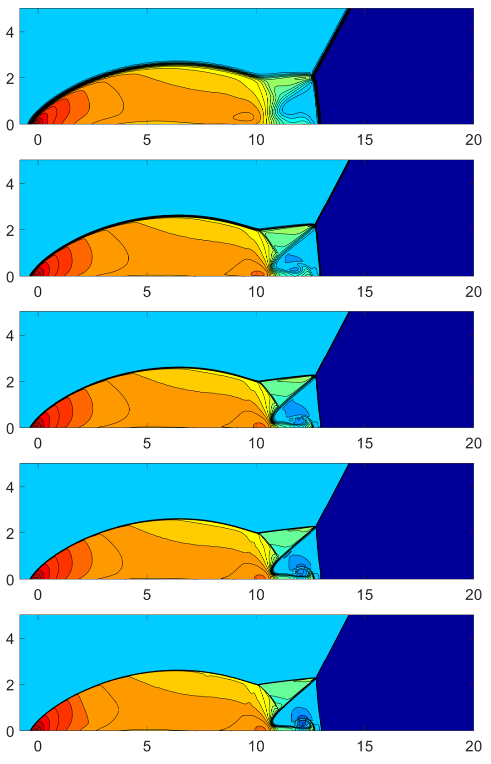
Figure 3.25 illustrates the scale-invariance property of the solution through solution plots at different points in the time, for different levels of discretization but in the same scale-invariant coordinates. The various plots show visibly consistent flow features in all cases, but with various levels of resolution. In particular, for any given mesh resolution used, features such as the shocks and the roll-up are more sharply resolved in the later times plots. This is explained by the fact that, while the solutions display the scale-invariance property, at early times the flow features resulting from the shock reflection on the wedge are concentrated in a small area, and are thus resolved with fewer points than at later times, as illustrated in figure 3.24.

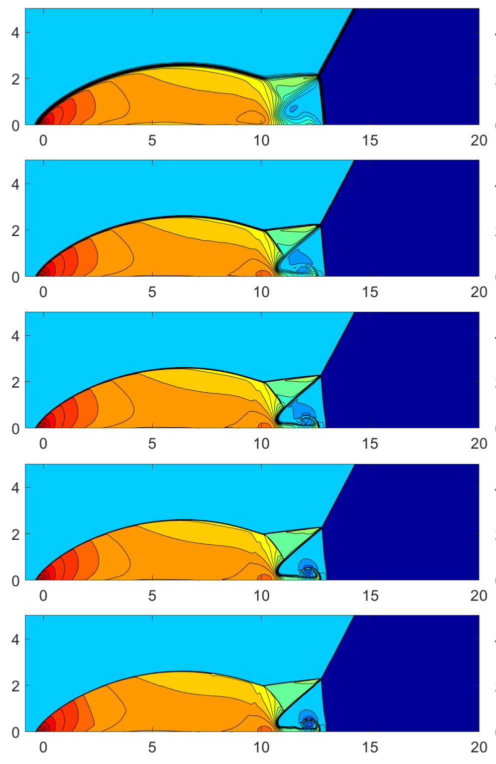
Notably, for later times and on the finest mesh discretizations, we can see that a vortex sheet starts to develop along the primary slip line. This phenomenon, which was previously observed in [55, 60], in different contexts and for the highest mesh refinements, was described in [60] as a deviation (a KH instability) from a purely inviscid solution—which was itself obtained, in time-independent fashion, from self-similarity. This instability may be attributed to a combination of two phenomena: on one hand, inevitable numerical errors, of the order of those observed in Figure 3.12, perturb the numerical solution in the vicinity of the primary slip line by introducing small oscillations in the profile of this contact discontinuity. On the other hand, as the mesh is refined, the magnitudes of the viscosity assignments (3.22) are reduced, thus leading to higher Reynolds numbers, and thus, to flows evolving from a laminar to a more turbulent behavior. In time, the initially small oscillations along the primary slip line grow in amplitude until they enter an initial roll-up phase [63] that can be appreciated in Figures 3.24 and 3.25. These oscillations are not visible in any of the images for coarser discretizations, which present, instead, a clearly laminar character—that as suggested by the numerical experiments reported in [60], would not appear when solving the time-independent 2D Euler equations re-expressed in terms of the scaled variables (3.38).

3.2.4 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 3.2.3.1 and 3.2.3.2. 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 3.3 (resp. Table 3.4), 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 3.5 (resp. Table 3.6). 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


(a) 1600 × 400 spatial grid.


(b) 2400 × 600 spatial grid.


(c) 3200 × 800 spatial grid.

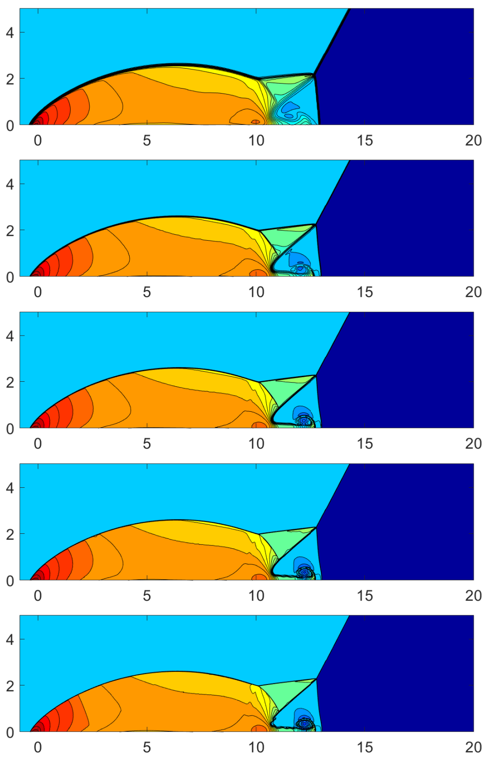

(d) 4800 × 1200 spatial grid.

Figure 3.25: FC-SDNN solutions to the Double Mach reflection problem represented in scale invariant coordinates, at different points in time and for various domain discretization sizes as described in the text.

(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 3.3 to 3.6. 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 3.3 to 3.6 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 3.3 to 3.6.

– 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 3.2.3.1 and 3.2.3.2, 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 3.3: 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 3.4: 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 3.5: 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 3.6: Computing-time components for Euler 2D problems on an $N \times N$-point grid with $N = 400$.

3.3 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 thesis.

FC order. As illustrated in Section 3.2.2, and, in particular, in Figure 3.10, 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, 33], 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 3.7)—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 3.1. On one hand, the FC time-marching scheme presented in Section 3.1.1 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 3.1.2, 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 ϕ 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 3.1.3 for the 1D and 2D cases can be trivially extended to three dimensions. The definition of a 3D artificial viscosity operator (cf. Section 3.1.3), 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 3.1.3, while use of the 1D weight operator yields stable 2D simulations, the modified 2D operator introduced in Section 3.1.3.2 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 ϕ (Section 3.1.2.1). 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 ϕ 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.

Chapter 4

FC-SDNN II: GENERAL-DOMAIN MULTI-PATCH FC-BASED SHOCK DYNAMICS SOLVER

This chapter describes a general-domain multi-patch generalization of the single-patch FC-SDNN method presented in Chapter 3. Based on a multi-domain decomposition and an overlapping patch/subpatch strategy, the proposed discretization method, described in Section 4.1 below, is applicable to general spatial domains, including domains containing smooth and non-smooth boundaries. Importantly, the assignment of artificial viscosity at and around obstacles (which is illustrated in Figure 4.11; see also Remark 13), is closely correlated with the algorithm's enforcement of boundary conditions for both the tangential and normal velocity components as well as the temperature (e.g. adiabatic conditions), and thus the energy. The FC-SDNN algorithm thus produces natively a setting that incorporates both inviscid flow away from boundaries and flow discontinuities, as well as boundary layers as proposed in Prandtl's boundary-layer theory [64]. We have additionally found that, as a significant by-product of the proposed use of boundary conditions on all velocity components and resulting boundary-viscosity assignments and boundary-layers, the algorithm does not require use of limiters to prevent the occurrence of negative densities and pressures and associated numerical instabilities. We hypothesize that the need to utilize positivity limiters in other approaches to ensure positivity and stability may arise from a failure of well-posedness associated with use of incomplete boundary conditions in an effectively (numerically) viscous environment—particularly on the aft obstacle-boundary portions. All of the numerical results presented in this chapter (Sections 4.5.2.4 through 4.5.2.10) enjoy such stability properties without the use of positivity preserving limiters. This include the examples in Section 4.5, which presents test cases at hypersonic speeds, up to and including significantly higher speeds than previously achieved, such as e.g. Mach 25 re-entry flow speeds.

Remark 7. The literature on compressible flows at high Mach number, including simulation and experiment, is somewhat sparse. An early reference is provided by [65]. In particular, this reference mentions that "real-gas effects will affect the shock-wave shape only at Mach numbers 8 and above, and then not too significantly".

But the reference also implies that such real-gas effects do lead to important deviations on physical observables such as density ratios across the shock—including, e.g., a variance from an experimental ratio of 14.6 compared to a perfect-gas ratio of 5.92 for a flow past a blunt body at Mach number 19.25. Similarly, the much more recent computational reference [66] presents numerical computations demonstrating significant differences between perfect-gas solutions and solutions that incorporate certain chemical reactions that are triggered at the high pressures observed in high Mach number configurations (a numerical experiment at Mach 16 on a limited section ahead of the cylindrical obstacle is presented in that reference). Perfect-gas flow simulations at high Mach numbers have nevertheless been considered repeatedly, including the early contribution [67] which uses the method presented in [68] (Navier-Stokes with Mach number 22), as well as [69] (Mach 10), [55] (Mach 10), and [50] (Mach 10). We thus suggest that perfect gas simulations at high Mach numbers remain relevant as mathematical and computational testbeds even if they are not physically accurate. Possibly, for example, an extension of the FC-SDNN solver proposed in this thesis to a multi-species context may provide similar performance (albeit at the increased cost inherent in the multi-species context) as it does in the perfect gas case.

This chapter is organized as follows, Section 4.3 presents an adaptation to the present context of the artificial viscosity method introduced in Chapter 3; of particular note in this connection is the novel windowed-viscosity propagation procedure introduced in Section 4.3.2. The corresponding time-marching algorithm, including necessary inter-patch and intra-patch data exchanges, is described in Section 4.2. A Message Passing Interface (MPI) parallel implementation of the algorithm, which mirrors the spatial patch/subpatch decomposition used, is then proposed in Section 4.4. As suggested in Section 4.5.1, which presents illustrations for which perfect weak parallel scaling was observed, the proposed parallel implementation enjoys high weak and strong parallel scalability. The practical capabilities of the novel multi-patch FC-SDNN strategy are demonstrated in Section 4.5, including test cases demonstrating the parallel performance of the method (Section 4.5.1) as well as illustrations concerning challenging physical problems for a variety of complex geometries (Section 4.5.2)—including cylinders with circular as well as triangular-wedge and prismatic cross-sections, and for test cases involving supersonic and hypersonic flow.

4.1 Overlapping-patch/subpatch geometry description and discretization

4.1.1 Domain decomposition: patches and their parametrizations

Following [7, 1, 2, 33], in order to obtain an FC-based discretization of equation (2.6) for general 2D domains Ω which may feature curved and/or non-smooth boundaries $\Gamma = \partial\Omega$, the proposed method relies on the decomposition of the connected open set Ω as a union of a number $P = P_{\mathcal{S}} + P_{C_1} + P_{C_2} + P_{\mathcal{I}}$ of overlapping patches (see Remark 8), each one of which is an open set, including, a number $P_{\mathcal{S}}$ of $\mathcal{S}$-type patches $\Omega_p^{\mathcal{S}}$ (smooth-boundary patches); a number P_{C_1} of C_1-type patches $\Omega_p^{C_1}$; a number P_{C_2} of C_2-type patches $\Omega_p^{C_2}$; and a number $P_{\mathcal{I}}$ of $\mathcal{I}$-type patches $\Omega_p^{\mathcal{I}}$ (interior patches):

$$\Omega = \left(\bigcup_{p=1}^{P_{\mathcal{S}}} \Omega_p^{\mathcal{S}}\right) \cup \left(\bigcup_{p=1}^{P_{C_1}} \Omega_p^{C_1}\right) \cup \left(\bigcup_{p=1}^{P_{C_2}} \Omega_p^{C_2}\right) \cup \left(\bigcup_{p=1}^{P_{\mathcal{I}}} \Omega_p^{\mathcal{I}}\right). \tag{4.1}$$

Here C_1-type patches (resp. C_2-type patches) are defined to contain a neighborhood within Ω of boundary corner points of “Type C_1” (also called C_1 corner points in what follows), namely corner points formed by two smooth arcs meeting at C with interior angles $\theta > 180°$ (resp. corner points of “Type C_2”, or C_2 corner points, namely corner points with interior angles θ satisfying $0 < \theta < 180°$). $\mathcal{S}$-type patches and $\mathcal{I}$-type patches, on the other hand are used to cover areas along smooth boundary regions and regions away from boundaries, respectively.

Remark 8. As detailed in the following sections, each patch $\Omega_p^{\mathcal{R}}$ ($\mathcal{R} = \mathcal{S}, \mathcal{I}, C_2, C_1$; $p = 1, \ldots, P_{\mathcal{R}}$) is constructed as the image under a smooth parametrization of a polygonal parameter domain $\mathcal{P}$. A requirement imposed on the patches $\Omega_p^{\mathcal{R}}$, which is satisfied in all cases by the patches constructed in the following sections, is that the image of each of the sides of the polygonal parameter domain is either fully contained within Γ or it contains at most one point of intersection with Γ (the latter case occurring as the image of one of the parameter polygon sides meets Γ transversally.)

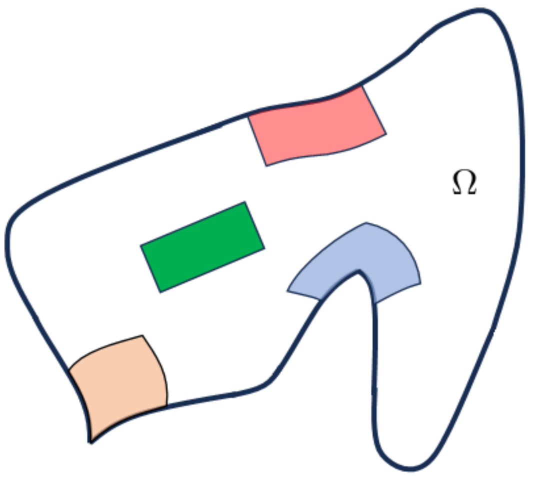


Figure 4.1: Patch types used in the overlapping-patch decomposition of a general domain Ω, including (a) a corner patch of type C_2 (orange); (b) a smooth-boundary patch of type $\mathcal{S}$ (red); (c) an interior patch of type $\mathcal{I}$ (green); and (d) a corner patch of type C_1 (blue). The $\mathcal{Q}$ parameter space is used for (a) through (c) and the $\mathcal{L}$ parameter space is used for (d).

As detailed in what follows, $\mathcal{S}$-type patches, $\mathcal{I}$-type patches and C_2-type patches (resp.C_1-type patches) are discretized on the basis of smooth invertible mappings, each one of which maps a canonical parameter space onto the *closure* of a single patch. For patches of type $\mathcal{S}$, $\mathcal{I}$ and C_2 the canonical parameter space is taken to equal to the unit square

$$\mathcal{Q} := [0,1] \times [0,1], \tag{4.2}$$

whereas for patches of type C_1 the L-shaped parameter space

$$\mathcal{L} := ([0,1] \times [\frac{1}{2}, 1]) \cup ([\frac{1}{2}, 1] \times [0, \frac{1}{2}]) \tag{4.3}$$

is used (Figure 4.1). Simple Cartesian discretizations of $\mathcal{Q}$ and $\mathcal{L}$ induce, via the corresponding parametrizations used, the necessary discretizations of each one of the patches $\Omega_p^{\mathcal{R}}$ for $\mathcal{R} = \mathcal{S}$, C_2, C_1 and $\mathcal{I}$, and for all relevant values of p. The patch overlaps are assumed to be "sufficiently broad"—so that, roughly speaking, except for discretization points near physical boundaries, each discretization point $\mathbf{x}$, associated with a given patch $\Omega_p^{\mathcal{R}}$, ($\mathcal{R} = \mathcal{S}, \mathcal{I}, C_2, C_1$; $p = 1, \dots, P_{\mathcal{R}}$) is contained in a "sufficiently interior" region of at least one other patch, say $\Omega_{p'}^{\mathcal{R}}$, ($p' \neq p$). This overlap condition is imposed so as to ensure adequate interpolation ranges between patches as well as creation of sufficiently smooth viscosity windows near patch boundaries—as detailed in Sections 4.1.2.3 and 4.3.2, respectively.

In order to facilitate parallelization while avoiding the use of Fourier expansions of extremely high order (so as to control differentiation errors in the context of sharp flow features such as shock waves and other solution discontinuities), the proposed algorithm allows for partitioning of the patches $\Omega_p^{\mathcal{R}}$ ($\mathcal{R} = \mathcal{S}, \mathcal{I}, \mathcal{C}_2, \mathcal{C}_1$) into sets of similarly overlapping "subpatches" introduced in Section 4.1.2.

The proposed domain-decomposition algorithm constructs at first boundary-conforming patches $\Omega_p^{\mathcal{R}}, p = 1, \ldots, P_{\mathcal{R}}$ ($\mathcal{C}_1$-type patch), each one of which is mapped via a domain mapping

$$\mathcal{M}_p^{\mathcal{R}} : \widetilde{\mathcal{R}} \to \Omega_p^{\mathcal{R}} \tag{4.4}$$

where, for $\mathcal{R} = \mathcal{S}$, $\mathcal{I}$ and $\mathcal{C}_2$ (resp. $\mathcal{R} = \mathcal{C}_1$) we set $\widetilde{\mathcal{R}} = \mathcal{Q}$ (resp. $\widetilde{\mathcal{R}} = \mathcal{L}$); see equations (4.1.2.1) and (4.3).

Using these notations and conventions the proposed general strategy for construction of patch subdomains can be outlined as a sequence of four steps.

(i) $\mathcal{C}_1$-type patches: Patches around any existing $\mathcal{C}_1$ corner points are first constructed following the procedure described in Section 4.1.1.1.

(ii) $\mathcal{C}_2$-type patches: Patches around any existing $\mathcal{C}_2$ corner points are then constructed following the procedure described in Section 4.1.1.2.

(iii) $\mathcal{S}$-type patches: Patches in the vicinity containing portions of smooth physical boundaries are then constructed, insuring a sufficiently wide overlap with the previously constructed $\mathcal{C}_1$-type and $\mathcal{C}_2$-type patches, as indicated in Section 4.1.1.3.

(iv) $\mathcal{I}$-type patches: Interior patches, away from boundaries are finally constructed so as to cover the remaining interior of the domain Ω, typically using affine mappings, and insuring a sufficiently wide overlap with existing $\mathcal{C}_1$-type, $\mathcal{C}_2$-type and $\mathcal{S}$-type patches designed in steps (i), (ii) and (iii), as detailed in Section 4.1.1.4.

(v) Subpatches: The patches introduced in steps (ii), (iii) and (iv) can subsequently be further decomposed into overlapping subpatches within each patch, as described in Section 4.1.2.4—in order to ensure a sufficiently fine gridsize is achieved in each subpatch while using a constant number of discretization points per subpatch, as discussed in Section 4.1.2.4.

4.1.1.1 C_1- type patches

Let $C \in \Gamma$ denote a C_1 corner point. By definition, in a neighborhood of C the boundary Γ can be represented as the union of two smooth arcs $\overset{\frown}{AC}$ and $\overset{\frown}{BC}$ meeting at C, as illustrated in Figure 4.2. We assume, as we may, that smooth and invertible 1D maps $\ell_A : [0,1] \to \overline{\Omega}$ and $\ell_B : [0,1] \to \mathbb{R}^2$ are available that parametrize suitable extensions of the curves $\overset{\frown}{AC}$ and $\overset{\frown}{BC}$ beyond the point C, in such a way that, with reference to Figure 4.2, we have

$$\begin{aligned} \ell_A([0,\tfrac{1}{2}]) &= \overset{\frown}{AC}, \qquad & \ell_A([\tfrac{1}{2},1]) &= \overset{\frown}{CD}, \\ \ell_B([0,\tfrac{1}{2}]) &= \overset{\frown}{BC}, \qquad & \ell_B([\tfrac{1}{2},1]) &= \overset{\frown}{CE}, \end{aligned} \tag{4.5}$$

(and, thus, in particular, $\ell_A([0,1]) = \overset{\frown}{AD}$ and $\ell_B([0,1]) = \overset{\frown}{BE}$). Using these curves a 2D parametrization $\mathcal{M}_p^{C_1} : \mathcal{L} \to \overline{\Omega_p^{C_1}}$ as in (4.4) for a C_1-type patch Ω^{C_1} around the point C as in Figure 4.2 can be defined by

$$\mathcal{M}_p^{C_1}(\xi,\eta) = \ell_A(\xi) + \ell_B(\eta) - C. \tag{4.6}$$

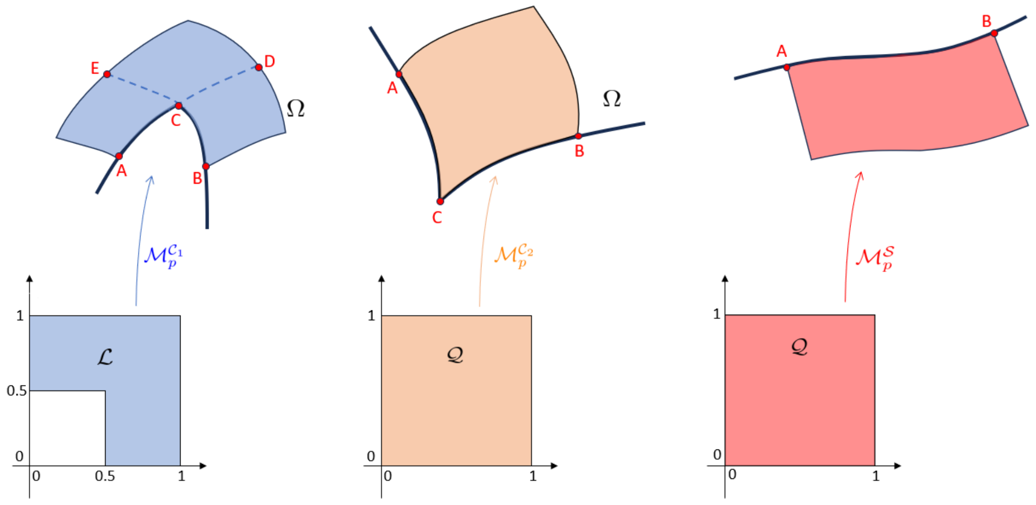


Figure 4.2: Mappings for a C_1-type patch (left image pair), C_2-type patch (middle image pair) and S-type patch (right image pair).

To conclude this section it is relevant to note that a straightforward patching strategy around a C_1 corner point obtained as a union of two patches extruded separately from the curves $\overset{\frown}{AD}$ and $\overset{\frown}{BE}$ results in a pair of patches that cannot satisfy the inter-patch overlap-size conditions required per Section 4.1.2.3 (and motivated by the commentary in Remark 10 and illustrated in Figure 4.6, with additional details in Section 4.1.4.1)—which, relating to the interpatch interpolation algorithm that is a part of the overset decomposition strategy, requires that points in one patch that are recipients of interpolation data from a second patch cannot themselves be donors of interpolation data onto the second patch.

4.1.1.2 $\mathcal{C}_2$-type patches

Let us now consider a $\mathcal{C}_2$ corner point $C \in \Gamma$. By definition, in a neighborhood of C the boundary Γ can be represented as the union of two smooth arcs $\overset{\frown}{AC}$ and $\overset{\frown}{BC}$ meeting at C, which may be parametrized by smooth and invertible 1D maps $\ell_A : [0,1] \to \mathbb{R}^2$ and $\ell_B : [0,1] \to \mathbb{R}^2$ in such a way that, with reference to Figure 4.2, we have

$$\ell_A([0,1]) = \overset{\frown}{AC}, \quad \ell_B([0,1]) = \overset{\frown}{BC}. \tag{4.7}$$

Using these parametrized curves,we obtain the 2D parametrization $\mathcal{M}_p^{\mathcal{C}_2} : Q \to \overline{\Omega_p^{\mathcal{C}_2}}$ given by

$$\mathcal{M}_p^{\mathcal{C}_2}(\xi, \eta) = \ell_A(\xi) + \ell_B(\eta) - C \tag{4.8}$$

for a $\mathcal{C}_2$-type patch $\Omega^{\mathcal{C}_2}$ around the point C.

4.1.1.3 $\mathcal{S}$-type patches

Turning to smooth portions of Γ, for a given smooth arc $\overset{\frown}{AB} \subset \Gamma$ joining two selected points $A, B \in \Gamma$ and parametrized by a smooth and invertible map $\ell_A : [0,1] \to \mathbb{R}^2$ such that

$$\ell_A([0,1]) = \overset{\frown}{AB}, \tag{4.9}$$

a 2D $\mathcal{S}$-patch parametrization along $\overset{\frown}{AB}$ is obtained simply by extruding a patch along the normal direction. In detail, letting $\nu : [0,1] \to \mathbb{R}^2$ denote the unit normal vector to the boundary arc $\overset{\frown}{AB}$, so that, for $t \in [0,1]$, $\nu(t)$ is the unit normal vector at the point $\ell_A(t)$ pointing towards the interior of Ω, and letting $H > 0$ denote an adequately selected parameter, as described below, the desired 2D $\mathcal{S}$-type patch parametrization $\mathcal{M}_p^{\mathcal{S}} : Q \to \overline{\Omega_p^{\mathcal{S}}}$, illustrated in Figure 4.2, is defined by

$$\mathcal{M}_p^{\mathcal{S}}(\xi, \eta) = \ell_A(\xi) + \eta H \nu(\xi). \tag{4.10}$$

4.1.1.4 $\mathcal{I}$-type patches

Finally, we consider $\mathcal{I}$-type patches covering portions of Ω at a positive distance from Γ, which, per point (iv) above, are constructed on the basis of affine parametrizations $\mathcal{M}_p^{\mathcal{I}} : \mathcal{Q} \to \overline{\Omega_p^{\mathcal{I}}}$.

4.1.2 Patches, subpatches and their grids

An overall overlapping-patch decomposition of the form (4.1) for the domain Ω is obtained on the basis of adequately selected sets of patches, including patches of types $\mathcal{C}_1, \mathcal{C}_2, \mathcal{S}$ and $\mathcal{I}$, as described in Section 4.1.1, in such a way that equation (4.1) is satisfied. Since Ω is a connected open set, and since all patches are themselves open sets, each patch $\Omega_p^{\mathcal{R}}$ ($\mathcal{R} = \mathcal{S}$, $\mathcal{C}_1$, $\mathcal{C}_2$, or $\mathcal{I}$ and $1 \le p \le P_{\mathcal{R}}$) must necessarily overlap one or more other patches $\Omega_{p'}^{\mathcal{R}'}$ ($\mathcal{R}' = \mathcal{S}$, $\mathcal{C}_1$, $\mathcal{C}_2$, or $\mathcal{I}$ and $1 \le p' \le P_{\mathcal{R}'}$). In fact, an efficient implementation of the proposed algorithm additionally requires that a "sufficient amount of overlap" exists, in the sense that every point $x \in \Omega_p^{\mathcal{R}}$ that lies "in the vicinity" of the boundary of $\Omega_p^{\mathcal{R}}$ must lie "sufficiently deep" within some patch $\Omega_{p'}^{\mathcal{R}'}$ (i.e., within $\Omega_{p'}^{\mathcal{R}'}$ and away from a vicinity of the boundary of $\Omega_{p'}^{\mathcal{R}'}$) for some $(p', \mathcal{R}') \neq (p, \mathcal{R})$. The vicinity and depth concepts alluded to above are defined and quantified in Sections 4.1.2.3, 4.1.4.1 and 4.1.4.2.

For flexibility in both geometrical representation and discretization refinement, the patching structure we use incorporates a decomposition in "sub-patches" $\Omega_{p,\ell}^{\mathcal{R}}$ of each patch $\Omega_p^{\mathcal{R}}$ ($\mathcal{R} = \mathcal{S}$, $\mathcal{C}_1$, $\mathcal{C}_2$, or $\mathcal{I}$):

$$\Omega_p^{\mathcal{R}} = \bigcup_{\ell=1}^{r_p} \Omega_{p,\ell}^{\mathcal{R}},$$

on each of which a patch-type dependent number of discretization points is used, as indicated in Sections 4.1.2.1 and 4.1.2.2. Like the patches themselves, the subpatches we use are required to display sufficiently large amounts of overlap between neighboring subpatches. Subpatches are defined, simply, via overlapping decomposition of the canonical parameter spaces $\mathcal{P} = \mathcal{Q}$ and $\mathcal{P} = \mathcal{L}$, as described in Sections 4.1.2.1 and 4.1.2.2. The following description utilizes detailed notations for the discretizations used for the various patch and subpatch types. Such level of detail, which involves several sub- and super-indexes, is needed in the context of the description of the multi-patch viscosity assignment and propagation strategy presented in Section 4.3.

4.1.2.1 $\mathcal{Q}$ parameter space discretization (for $\mathcal{S}$-, $\mathcal{I}$- and $\mathcal{C}_2$-type patches)

In order to obtain discretizations for each one of the patches $\Omega_p^{\mathcal{R}}$ ($\mathcal{R} = \mathcal{S}, \mathcal{C}_2, \mathcal{I}$; $1 \le p \le P_{\mathcal{R}}$), we introduce the parameter space grids

$$G_p^{\mathcal{R}} = \mathcal{Q} \cap \left\{ (q_{p,i}^{\mathcal{R},1}, q_{p,j}^{\mathcal{R},2}) \ : \ q_{p,i}^{\mathcal{R},1} = ih_p^{\mathcal{R},1}, q_{p,j}^{\mathcal{R},2} = jh_p^{\mathcal{R},2}, \quad (i,j) \in \mathbb{N}_0^2 \right\}, \qquad (4.11)$$

of grid sizes $h_p^{\mathcal{R},1} = \frac{1}{N_p^{\mathcal{R},\xi}+2n_v-1}$ and $h_p^{\mathcal{R},2} = \frac{1}{N_p^{\mathcal{R},\eta}+2n_v-1}$ along the q^1 (abscissa) and q^2 (ordinate) directions in parameter space, respectively, and denoting $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Here $N_p^{\mathcal{R},\xi}$ (resp. $N_p^{\mathcal{R},\eta}$) denotes the number of discretization points, in each discretization line along the q^1 (resp. q^2) direction, that lie outside a certain "boundary vicinity" layer of discretization points. The boundary vicinity is itself characterized by the positive integer n_v equal to the number of discretization points along the boundary-vicinity width, as illustrated in Figure 4.3. The value $n_v = 9$ was used in all of the implementations presented in this chapter.

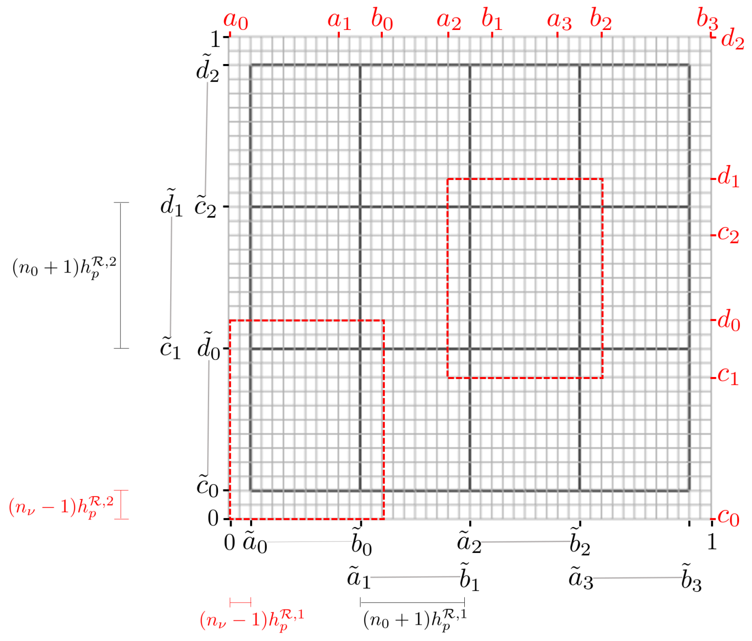


Figure 4.3: Patch decomposition (in $\mathcal{Q}$ parameter-space) for a patch $\Omega_p^{\mathcal{R}}$ (with $\mathcal{R} = \mathcal{S}$, $\mathcal{C}_2$, or $\mathcal{I}$) into preliminary non-overlapping rectangular subpatches (with boundaries shown in black), associated overlapping rectangles (two examples highlighted, with boundaries shown in red) and discretization lines (shown in gray). Geometric parameter values $r_p = 4$, $s_p = 3$, $n_0 = 9$ and $n_v = 3$ were used for this illustration.

As indicated in Section 4.1.1, the proposed discretization algorithm additionally relies on use of subpatches to limit the order of the FC expansions used. To describe the sub-patch decomposition approach for of $\mathcal{S}$-, $\mathcal{I}$- and $\mathcal{C}_2$-type patches, we first introduce, for a given assignment of positive integers $r_p^{\mathcal{R}}$ and $s_p^{\mathcal{R}}$ for each $(\mathcal{R}, p)$, a set of non-overlapping rectangles strictly contained within Q:

$$\bigcup_{(r,s)\in\Theta_p^{\mathcal{R}}} [\tilde{a}_r, \tilde{b}_r] \times [\tilde{c}_s, \tilde{d}_s] \subsetneq Q \tag{4.12}$$

where we define

$$\Theta_p^{\mathcal{R}} = \{(r,s) \in \mathbb{N}_0^2 \mid 0 \le r \le r_p^{\mathcal{R}} - 1 \text{ and } 0 \le s \le s_p^{\mathcal{R}} - 1\}. \tag{4.13}$$

In what follows we select $N_p^{\mathcal{R},\xi} = r_p^{\mathcal{R}}(n_0+1)-1$ and $N_p^{\mathcal{R},\eta} = s_p^{\mathcal{R}}(n_0+1)-1$, where $r_p^{\mathcal{R}}$ and $s_p^{\mathcal{R}}$ are given positive integers, and where n_0 denotes the number of points used in the discretization of certain preliminary non-overlapping subpatches introduced below along both the parameter space q^1 and q^2 directions. The procedure used for selection of the integers $r_p^{\mathcal{R}}$ and $s_p^{\mathcal{R}}$ is described in Section 4.1.2.4. The ($(\mathcal{R}, p)$-dependent interval endpoints $\tilde{a}_r$, $\tilde{b}_r$, $\tilde{c}_s$ and $\tilde{d}_s$ are selected, using certain integer parameters n_0 (number of discretization points along both the parameter space q^1 (abscissa) and q^2 (ordinate) directions in the discretization of certain preliminary non-overlapping subpatches introduced below) and n_v (number of discretization points along both the parameter space q^1 (abscissa) and q^2 (ordinate) directions contained in the discretization of a certain "vicinity" of the preliminary subpatches), in accordance with the relations

$$\begin{aligned}
&\tilde{a}_0 = (n_v - 1)h_p^{\mathcal{R},1}, && \\
&\tilde{b}_r - \tilde{a}_r = (n_0 + 1)h_p^{\mathcal{R},1}, && (0 \le r \le r_p^{\mathcal{R}} - 1),\\
&\tilde{a}_{r+1} = \tilde{b}_r, && (0 \le r < r_p^{\mathcal{R}} - 1),\\
&\tilde{c}_0 = (n_v - 1)h_p^{\mathcal{R},2}, && \\
&\tilde{d}_s - \tilde{c}_s = (n_0 + 1)h_p^{\mathcal{R},2}, && (0 \le s \le s_p^{\mathcal{R}} - 1),\\
&\tilde{c}_{s+1} = \tilde{d}_s, && (0 \le s < s_p^{\mathcal{R}} - 1).
\end{aligned} \tag{4.14}$$

In all of the implementations presented in this chapter the value $n_0 = 83$ was used.

The overlapping subpatches associated with a given patch are characterized, via the patch parametrization, by the decomposition of Q as the union

$$Q = \bigcup_{(r,s)\in\Theta_p^{\mathcal{R}}} [a_r, b_r] \times [c_s, d_s], \tag{4.15}$$

of overlapping rectangles, whose vertices are given by

$$\begin{aligned}
a_r &= \tilde{a}_r - (n_v - 1)h_p^{\mathcal{R},1}, && (0 \le r \le r_p^{\mathcal{R}} - 1),\\
b_r &= \tilde{b}_r + (n_v - 1)h_p^{\mathcal{R},1}, && (0 \le r \le r_p^{\mathcal{R}} - 1),\\
c_s &= \tilde{c}_s - (n_v - 1)h_p^{\mathcal{R},2}, && (0 \le s \le s_p^{\mathcal{R}} - 1),\\
d_s &= \tilde{d}_s + (n_v - 1)h_p^{\mathcal{R},2}, && (0 \le s \le s_p^{\mathcal{R}} - 1).
\end{aligned} \tag{4.16}$$

Note that for the patches of type $\mathcal{R} = \mathcal{S}$, $\mathcal{C}_2$ and $\mathcal{I}$ considered in this Section, all of whose subpatches utilize $\mathcal{Q}$ parameter-space discretizations, each subpatch contains a total of

$$N_{\mathcal{Q}} = (n_0 + 2n_v)^2 \tag{4.17}$$

discretization points.

For $\ell = (r, s) \in \Theta_p^{\mathcal{R}}$, we denote by $G_{p,\ell}^{\mathcal{R}}$ the parameter space grid for the ℓ-th subpatch of the patch $\Omega_p^{\mathcal{R}}$; we clearly have

$$G_{p,\ell}^{\mathcal{R}} = G_p^{\mathcal{R}} \cap ([a_r, b_r] \times [c_s, d_s]) .$$

Using the mappings $\mathcal{M}_p^{\mathcal{R}}$ ($\mathcal{R} = \mathcal{C}_2$, $\mathcal{S}$, $\mathcal{I}$) described in Sections 4.1.1.2, 4.1.1.3 and 4.1.1.4, respectively, we obtain the desired subpatches $\Omega_{p,\ell}^{\mathcal{R}} = \mathcal{M}_p^{\mathcal{R}}([a_r, b_r] \times [c_s, d_s])$ ($\ell = (r, s)$) and the physical grids

$$\mathcal{G}_p^{\mathcal{R}} = \mathcal{M}_p^{\mathcal{R}}\left(G_p^{\mathcal{R}}\right) \quad \text{and} \quad \mathcal{G}_{p,\ell}^{\mathcal{R}} = \mathcal{M}_p^{\mathcal{R}}\left(G_{p,\ell}^{\mathcal{R}}\right). \tag{4.18}$$

The set of indices $\mathcal{D}_{p,\ell}^{\mathcal{R}}$ of parameter space grid points contained in $G_{p,\ell}^{\mathcal{R}}$, finally, is denoted by

$$\mathcal{D}_{p,\ell}^{\mathcal{R}} = \{(i, j) \in \mathbb{N}_0^2 \mid (ih_p^{\mathcal{R},1}, jh_p^{\mathcal{R},2}) \in G_{p,\ell}^{\mathcal{R}}\}, \quad \mathcal{R} = \mathcal{C}_2, \mathcal{S}, \mathcal{I}. \tag{4.19}$$

4.1.2.2 $\mathcal{L}$ parameter space discretization (for $\mathcal{C}_1$-type patches)

In analogy to the approach utilized in the previous section, in order to obtain discretizations for each one of the patches $\Omega_p^{\mathcal{C}_1}$; $1 \le p \le P_{\mathcal{C}_1}$, we introduce the parameter space grids

$$G_p^{\mathcal{C}_1} = \mathcal{L} \cap \left\{(q_{p,i}^{\mathcal{C}_1,1}, q_{p,j}^{\mathcal{C}_1,2}) \;:\; q_{p,i}^{\mathcal{C}_1,1} = ih_p^{\mathcal{C}_1,1}, q_{p,j}^{\mathcal{C}_1,2} = jh_p^{\mathcal{C}_1,2}, \quad (i, j) \in \mathbb{N}_0^2\right\}, \tag{4.20}$$

which, in the present case, we take to be *equiaxed*—that is, with parameter spacegrid sizes along the q^1 and q^2 directions given

$$h_p^{\mathcal{C}_1,1} = h_p^{\mathcal{C}_1,2} = \frac{1}{N_p^{\mathcal{C}_1} + 2n_v - 1}, \tag{4.21}$$

and denoting $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ Here $N_p^{C_1}$ denotes the number of discretization points, in each discretization line along either the q^1 or q^2 directions, that lie outside the n_v-point wide patch boundary vicinity, where n_v is the parameter introduced in Section 4.1.2.1 (as indicated in that section, the value $n_v = 9$ is used throughout this chapter).

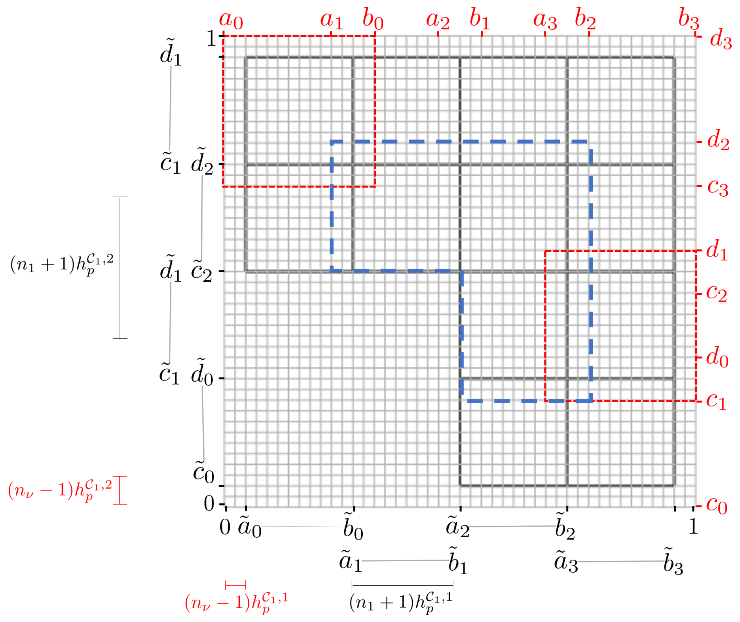


Figure 4.4: Patch decomposition (in $\mathcal{L}$ parameter-space) for a patch $\Omega_p^{\mathcal{R}}$ with $\mathcal{R} = C_1$ into preliminary non-overlapping rectangular subpatches (with boundaries shown in black), associated overlapping rectangles (two examples highlighted, with boundaries shown in red), corner set $H_0 = L_p^{C_1}$ (shown as a blue dashed polygon), and discretization lines (shown in gray). Geometric parameter values $r_p = 4$, $s_p = 4$, $n_1 = 9$ and $n_v = 3$ were used for this illustration.

Remark 9. To enable use of a single $\mathcal{L}$ parameter space for all C_1-type patches, as in the approach used for the $\mathcal{Q}$ parameter space in the previous section, it is assumed here that each C_1-type patch is constructed in such a way that its two arms are approximately equal in width and length. Under this assumption it is reasonable to define a single grid size parameter $h_p^{C_1}$ for C_1-type patches, as opposed to the two grid size parameters $h_p^{\mathcal{R},1}$ and $h_p^{\mathcal{R},2}$ used in section 4.1.2.1 for $\mathcal{S}$-, $\mathcal{I}$- and C_2-type patches.

The strategy used for sub-patch decomposition of C_1-type patches, in turn, is based on a decomposition of the corresponding parameter space $\mathcal{P} = \mathcal{L}$, and is illustrated in Figure 4.4. Following the lines of the $\mathcal{P} = \mathcal{Q}$ decomposition presented in the previous section here we utilize, for a given even integer $r_p^{C_1}$, a union of non-overlapping squares strictly contained within the canonical L-shaped domain $\mathcal{L}$:

$$\bigcup_{(r,s)\in\widetilde{\Theta}_p^{C_1}} [\widetilde{a}_r, \widetilde{b}_r] \times [\widetilde{c}_s, \widetilde{d}_s] \subsetneq \mathcal{L}, \qquad (\widetilde{\Theta}_p^{C_1} = \widetilde{\Theta}_p^{C_1,1} \cup \widetilde{\Theta}_p^{C_1,2} \cup \widetilde{\Theta}_p^{C_1,3}), \quad (4.22)$$

where

$$\begin{aligned}
\widetilde{\Theta}_p^{C_1,1} &= \{(r,s) \in \mathbb{N}_0^2 \mid 0 \le r \le \frac{r_p^{C_1}}{2} - 1 \text{ and } \frac{s_p^{C_1}}{2} \le s \le s_p^{C_1} - 1\},\\
\widetilde{\Theta}_p^{C_1,2} &= \{(r,s) \in \mathbb{N}_0^2 \mid \frac{r_p^{C_1}}{2} \le r \le r_p^{C_1} - 1 \text{ and } \frac{s_p^{C_1}}{2} \le s \le s_p^{C_1} - 1\},\\
\widetilde{\Theta}_p^{C_1,3} &= \{(r,s) \in \mathbb{N}_0^2 \mid \frac{r_p^{C_1}}{2} \le r \le r_p^{C_1} - 1 \text{ and } 0 \le s \le \frac{s_p^{C_1}}{2} - 1\}.
\end{aligned} \quad (4.23)$$

In what follows we select $N_p^{C_1} = r_p^{C_1}(n_1 + 1) - 1$, where $r_p^{C_1}$ is a given positive integer, and where n_1 denotes the number of discretization points along both the parameter space q^1 and q^2 directions in the discretization of certain preliminary non-overlapping subpatches introduced below.

The (p-dependent) interval endpoints $\widetilde{a}_r$, $\widetilde{b}_r$, $\widetilde{c}_s$ and $\widetilde{d}_s$ are selected, using the integer parameters n_1 and n_v (number of discretization points along either the parameter space q^1 and q^2 directions contained in the discretization of a "vicinity" of the preliminary subpatches, which coincides with the integer n_v used in the previous section), in accordance with the relations

$$\begin{aligned}
&\tilde{a}_0 = (n_v - 1)h_p^{C_1,1}, &&\\
&\tilde{b}_r - \tilde{a}_r = (n_0 + 1)h_p^{C_1,1}, && (0 \le r \le r_p^{C_1} - 1),\\
&\tilde{a}_{r+1} = \tilde{b}_r, && (0 \le r < r_p^{C_1} - 1),\\
&\tilde{c}_0 = (n_v - 1)h_p^{C_1,2}, &&\\
&\tilde{d}_s - \tilde{c}_s = (n_0 + 1)h_p^{C_1,2}, && (0 \le s \le s_p^{C_1} - 1),\\
&\tilde{c}_{s+1} = \tilde{d}_s, && (0 \le s < s_p^{C_1} - 1).
\end{aligned} \quad (4.24)$$

Typically the selection $n_1 \approx \frac{1}{2}n_0$ is utilized, where n_0 denotes the integer parameter used in the previous section; in particular, the value $n_1 = 43 \approx 83/2$ is used throughout the test cases considered in this chapter.

As for the previously considered patch types, the overlapping subpatches associated with a given C_1 patch are characterized, via the patch parametrization, by an overlapping decomposition of $\mathcal{L}$ as the union of closed subpatches. In the present case, in addition to overlapping rectangular subpatches an "L-shaped" subpatch $L_p^{C_1}$ is utilized. In detail, the overlapping sub-patch decomposition of a C_1 patch is constructed on the basis of the parameter-space decomposition

$$\mathcal{L} = \bigcup_{q \in \Theta_p^{C_1}} H_q, \qquad (\Theta_p^{C_1} = \Theta_p^{C_1,1} \cup \Theta_p^{C_1,2} \cup \Theta_p^{C_1,3} \cup \{0\}) \tag{4.25}$$

where

$$\begin{aligned} H_q &= [a_r, b_r] \times [c_s, d_s] \quad \text{for} \quad q = (r,s) \in \Theta_p^{C_1,1} \cup \Theta_p^{C_1,2} \cup \Theta_p^{C_1,3} \\ H_q &= L_p^{C_1} \qquad\qquad\qquad \text{for} \quad q = 0, \end{aligned} \tag{4.26}$$

where the (p-dependent) set of vertices a_r, b_r, c_s and d_s is given by

$$\begin{aligned} a_r &= \tilde{a}_r - (n_v - 1)h_p^{C_1}, && (0 \le r \le r_p^{C_1} - 1), \\ b_r &= \tilde{b}_r + (n_v - 1)h_p^{C_1}, && (0 \le r \le r_p^{C_1} - 1), \\ c_s &= \tilde{c}_s - (n_v - 1)h_p^{C_1}, && (0 \le s \le s_p^{C_1} - 1), \\ d_s &= \tilde{d}_s + (n_v - 1)h_p^{C_1}, && (0 \le s \le s_p^{C_1} - 1), \end{aligned} \tag{4.27}$$

and where the index sets

$$\begin{aligned} \Theta_p^{C_1,1} &= \widetilde{\Theta}_p^{C_1,1} \setminus \{(\frac{r_p^{C_1}}{2} - 1, \frac{s_p^{C_1}}{2})\}, \\ \Theta_p^{C_1,2} &= \widetilde{\Theta}_p^{C_1,2} \setminus \{(\frac{r_p^{C_1}}{2}, \frac{s_p^{C_1}}{2})\}, \\ \Theta_p^{C_1,3} &= \widetilde{\Theta}_p^{C_1,3} \setminus \{(\frac{r_p^{C_1}}{2}, \frac{s_p^{C_1}}{2} - 1)\}, \end{aligned} \tag{4.28}$$

where used. The corner set $H_0 = L_p^{C_1}$ in (4.26), finally, is defined by

$$L_p^{C_1} = L_{p,1}^{C_1} \cup L_{p,2}^{C_1} \cup L_{p,3}^{C_1}, \tag{4.29}$$

where

$$\begin{aligned} L_{p,1}^{C_1} &= [a_{\frac{r_p^{\mathcal{R}}}{2}-1}, b_{\frac{r_p^{\mathcal{R}}}{2}-1}] \times [c_{\frac{r_p^{\mathcal{R}}}{2}}, d_{\frac{r_p^{\mathcal{R}}}{2}}], \\ L_{p,2}^{C_1} &= [a_{\frac{r_p^{\mathcal{R}}}{2}}, b_{\frac{r_p^{\mathcal{R}}}{2}}] \times [c_{\frac{r_p^{\mathcal{R}}}{2}-1}, d_{\frac{r_p^{\mathcal{R}}}{2}-1}], \\ L_{p,2}^{C_1} &= [a_{\frac{r_p^{\mathcal{R}}}{2}-1}, b_{\frac{r_p^{\mathcal{R}}}{2}-1}] \times [c_{\frac{r_p^{\mathcal{R}}}{2}}, d_{\frac{r_p^{\mathcal{R}}}{2}}]. \end{aligned} \tag{4.30}$$

Note the three single-index sets subtracted in (4.28), which correspond to three squares whose union makes up the corner subpatch $H_0 = L_p^{C_1}$.

For $\ell \in \Theta_p^{C_1}$, we denote by $G_{p,\ell}^{C_1}$ the parameter space grid for the ℓ-th subpatch of the patch $\Omega_p^{C_1}$:

$$G_{p,\ell}^{C_1} = G_p^{C_1} \cap H_\ell.$$

Using the mapping $\mathcal{M}_p^{C_1}$ described in Section 4.1.1.1 we obtain the desired subpatches $\Omega_{p,\ell}^{C_1} = \mathcal{M}_p^{C_1}(H_\ell)$ as well as the physical grids

$$\mathcal{G}_p^{C_1} = \mathcal{M}_p^{C_1}\left(G_p^{C_1}\right) \quad \text{and} \quad \mathcal{G}_{p,\ell}^{C_1} = \mathcal{M}_p^{C_1}\left(G_{p,\ell}^{C_1}\right), \tag{4.31}$$

with subpatch parameter space grid points $\mathcal{D}_{p,\ell}^{C_1}$ is denoted by

$$\mathcal{D}_{p,\ell}^{C_1} = \{(i,j) \in \mathbb{N}_0^2 \mid (ih_p^{C_1,1}, jh_p^{C_1,2}) \in G_{p,\ell}^{C_1}\}. \tag{4.32}$$

4.1.2.3 Minimum subpatch overlap condition and special $\mathcal{S}$-C_1 patch overlaps

As motivated in Remark 10 below and detailed in in Sections 4.1.4.1 and 4.1.4.2, and as alluded to previously in Section 4.1.1, in order to properly enable inter-patch data communication and multi-patch viscosity assignment propagation, the overlap between patches $\Omega_p^{\mathcal{R}}$ must be sufficiently broad. The required overlap breadth is quantified in terms of a parameter associated with the selected subpatches of each patch (namely, the integer parameter n_v introduced in Sections 4.1.2.1 and 4.1.2.2) as well as the patch-boundary "sides". (The "sides" of a patch (resp. of a subpatch) are defined as the images under the patch parametrization of each one of the straight segments that make up the boundary of the corresponding parameter polygon $\mathcal{Q}$ or $\mathcal{L}$, cf. Remark 8 (resp. each one of the sides of the rectangles in equations (4.15) and (4.26), or the L-shaped polygons (4.29)). Utilizing these concepts, the minimum-overlap condition is said to be satisfied for a given patch $\Omega_p^{\mathcal{R}}$ if and only if for each side of $\Omega_p^{\mathcal{R}}$ that is not contained in $\partial\Omega$ (cf. Remark 8), a $(2n_v + 1)$-point wide layer of grid points adjacent to that side in the $\Omega_p^{\mathcal{R}}$ grid is also included in a union of one or more patches $\Omega_{p'}^{\mathcal{R}'}$ with $(p', \mathcal{R}') \neq (p, \mathcal{R})$. (Note that, per the construction in Sections 4.1.2.1 and 4.1.2.2 at the parameter space level, subpatches of a single patch satisfy the minimum overlap condition embodied in equations (4.27) and (4.24): neighboring subpatches share a $(2n_v + 1)$-point wide overlap.)

A particular note is necessary in regard to the overlap of C_1- and $\mathcal{S}$-type patches. Numerical experiments have shown that spurious oscillations may emanate from boundary regions in the C_1 patch near the corner point unless the overlap with $\mathcal{S}$-patch is sufficiently broad. Specifically, numerical experimentation for the wedge and prism geometries (Figures 4.13 and 4.14) suggests that spurious oscillations may be eliminated provided the $\mathcal{S}$-type patch extends along the boundary up to and including the corner point (e.g. including the complete arcs $\overset{\frown}{AC}$ and $\overset{\frown}{BC}$ in the left panel of Figure 4.2).

4.1.2.4 Subpatches and grid refinement

Very general overlapping patch decompositions, satisfying the minimum overlap condition introduced in Section 4.1.2.3, can be obtained on the basis of the procedures described in Section 4.1.1. As briefly discussed below in this section, further, the strategies presented in Sections 4.1.2.1 and 4.1.2.2 can be used to produce sets of overlapping subpatches, each one endowed with a subpatch grid, in such a way that, 1) The set of all patches (namely, all ($\mathcal{S}$-, C_1-, C_2- and $\mathcal{I}$-type patches) satisfies the overlap conditions introduced in Section 4.1.2.3; 2) Each subpatch contains no more than $F = (n_0 + 2(n_v - 1))^2$ discretization points; and, 3) All of the subpatch parameter space grid sizes $h_p^{\mathcal{R},1}$ and $h_p^{\mathcal{R},2}$ ($\mathcal{R} = \mathcal{S}, C_1, C_2, \mathcal{I}$; $1 \le p \le P_{\mathcal{R}}$) are such that the resulting physical grid sizes (defined as the maximum distance between two consecutive grid points in physical space) are less than or equal to a user-prescribed upper bound $\overline{h} > 0$.

As indicated in Section 4.1.2.3, the sets of patches and subpatches constructed per the descriptions in Sections 4.1.2.1 and 4.1.2.2 satisfy points 1) and 2), but, clearly, they may or may not satisfy point 3). In the latter case, the integers $r_p^{\mathcal{R}}$ and $s_p^{\mathcal{R}}$ ($\mathcal{R} = \mathcal{S}, C_2, \mathcal{I}$; $1 \le p \le P_{\mathcal{R}}$) and $r_p^{C_1}$ ($1 \le p \le P_{C_1}$) are increased by a uniform constant factor—thus proportionally increasing the numbers of subpatches and the number of gridpoints while leaving the overall patch decomposition and mappings $\mathcal{M}_p^{\mathcal{R}}$ unchanged—until the physical grid size upper bound condition in point 3) is satisfied, as desired.

4.1.3 Governing equations in curvilinear coordinates

In the multi-patch setting described in this section, the solution vector $\mathbf{e}(\mathbf{x}, t)$ can be viewed as a family of vectors $\mathbf{e}_{p,\ell}^{\mathcal{R}}(\mathbf{x}, t)$ (with $\mathcal{R} = \mathcal{S}, C_1, C_2$, or $\mathcal{I}$, $1 \le p \le P_{\mathcal{R}}$ and $\ell \in \Theta_p^{\mathcal{R}}$), defined on $\Omega_{p,\ell}^{\mathcal{R}} \times \mathbb{R}^+$. To evolve the solution on the curvilinear grids $\mathcal{G}_{p,\ell}^{\mathcal{R}}$ associated to the smooth invertible mapping $\mathcal{M}_p^{\mathcal{R}}$, the Jacobian matrix $J_{\mathbf{q}}^{\mathcal{R},p}(\mathbf{x})$ of the inverse mapping $\left(\mathcal{M}_p^{\mathcal{R}}\right)^{-1}$ is utilized. Thus, using the chain rule expression

$\nabla_{\mathbf{x}} = [J_{\mathbf{q}}^{\mathcal{R},p}(\mathbf{x})]^T \nabla_{\mathbf{q}}$ for $\mathbf{x} \in \Omega_{p,\ell}^{\mathcal{R}}$, we obtain the curvilinear form

$$\frac{\partial \mathbf{e}_{p,\ell}^{\mathcal{R}}(\mathbf{x},t)}{\partial t} + \left[J_{\mathbf{q}}^{\mathcal{R},p}(\mathbf{x})\right]^T \nabla_{\mathbf{q}} \cdot \left(f(\mathbf{e}_{p,\ell}^{\mathcal{R}}(\mathbf{x},t))\right) = \left[J_{\mathbf{q}}^{\mathcal{R},p}(\mathbf{x})\right]^T \nabla_{\mathbf{q}} \cdot \left(f_v\left(\mathbf{e}_{p,\ell}^{\mathcal{R}}(\mathbf{x},t)\right)\right) \quad (4.33)$$

of the artificial viscosity-augmented hyperbolic system (2.13).

4.1.4 Patch/subpatch communication of solution values

The overlap built into the patch/subpatch domain decomposition approach described in the previous sections enables the exchange, or communication, of subpatch grid-point values of the solution $\mathbf{e}$ in the vicinity of boundaries of subpatches, which is a necessary element in the overlapping-patch time-stepping algorithm introduced in Section 4.2; see Remark 10 below. The communication of solution values relies on the concept of "n-point fringe region" of a subpatch. To introduce this concept we consider the sides of any subpatch $\Omega_{p,\ell}^{\mathcal{R}}$ (defined in Section 4.1.2.3), and we note that, per the constructions in that section, each side of a subpatch is either contained within Γ or it intersects Γ at most at one point. In the first (resp. the second) case we say that the side is "external" (resp. "internal"). Then, denoting by $\mathcal{I}_{p,\ell}^{\mathcal{R}}$ the set of all grid points in $\mathcal{G}_{p,\ell}^{\mathcal{R}}$ that are contained in the internal sides of $\Omega_{p,\ell}^{\mathcal{R}}$, we define the n-point fringe region $\mathcal{F}_{p,\ell,n}^{\mathcal{R}}$ of $\Omega_{p,\ell}^{\mathcal{R}}$ as the set of all grid points $\mathbf{x}_{ij} = (x_i, y_j) \in \mathcal{G}_{p,\ell}^{\mathcal{R}}$ whose distance to $\mathcal{I}_{p,\ell}^{\mathcal{R}}$ (in the sense of the "maximum index norm" distance d defined below) is less than or equal to n. In detail, the fringe region

$$\mathcal{F}_{p,\ell,n}^{\mathcal{R}} := \{\mathbf{x}_{ij} \in \mathcal{G}_{p,\ell}^{\mathcal{R}}, \quad d(\mathbf{x}_{ij}, \mathcal{I}_{p,\ell}^{\mathcal{R}}) < n\} \quad (4.34)$$

where the maximum index norm distance $d(\mathbf{x}_{ij}, \mathcal{I}_{p,\ell}^{\mathcal{R}})$ from the point $\mathbf{x}_{ij} \in \mathcal{G}_{p,\ell}^{\mathcal{R}}$ to the set $\mathcal{I}_{p,\ell}^{\mathcal{R}}$ is defined by

$$d(\mathbf{x}_{ij}, \mathcal{I}_{p,\ell}^{\mathcal{R}}) := \min_{\mathbf{x}_{rs} \in \mathcal{I}_{p,\ell}^{\mathcal{R}}} \max \{|i-r|, |j-s|\}, \quad (4.35)$$

is illustrated in Figure 4.5.

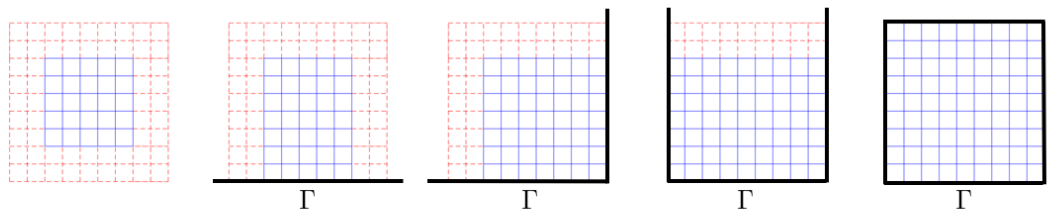


Figure 4.5: Illustration of subpatches (union of blue and red) and their fringes (red) for various kinds of subpatches. For simplicity this illustration uses the value $n_f = 2$ ($n_f = 5$ was used for all the numerical examples presented in this Chapter). Left panel: all the sides of the subpatch are internal sides. Four rightmost panels, from left to right: subpatches containing one, two, three and all four external sides, all of which, by design, are contained in the domain boundary Γ (black).

As described over the next few sections, as part of the time-stepping process each subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ receives solution data (from neighboring subpatches) on its n_f-fringe region $\mathcal{F}^{\mathcal{R}}_{p,\ell,n_f}$. In all implementations presented in this chapter the value $n_f = 5$ was used. In fact, the time-stepping method we use utilizes two different algorithms for the communication of solution data between pairs of subpatches, including "inter-patch" communication (that is, communication between two subpatches of different patches, which requires polynomial *interpolation* of grid-point values of the solution $\mathbf{e}$) and "intra-patch" communication (which only involves *exchange* of grid-point values of $\mathbf{e}$ between two subpatches of the same patch). Details concerning these two procedures are presented in Sections 4.1.4.1 and 4.1.4.2, respectively. The minimum overlap condition is required to ensure that donor points are not recipients of solution data.

Remark 10. The overlapping patch decomposition and discretization should be set up in such a way that data donor grid points are not themselves receivers of data from other patches. If this condition is not satisfied then the well-posedness of the problem and, thus, stability of the time stepping method, are compromised [7]. This donor-receiver condition is generally satisfied by requiring that donor grid points are located sufficiently deeply within the subpatch, while receiving grid points are located next to the boundary of the subpatch, as detailed in Section 4.1.4.1 for the case of solution-value communication between subpatches of different patches, and in Section 4.1.4.2 for the case of communication between subpatches of a single patch. Additionally, use of donor data from regions well separated from subpatch boundaries is advantageous since, as is expected from any discretization scheme used, the FC representations of the solution are more accurate and smoother away from the boundary of the representation region—which in this case are the subpatch boundaries themselves.

4.1.4.1 Inter-patch data communication

As indicated above, the proposed time-stepping algorithm requires communication of values of the solution vector $\mathbf{e} = \mathbf{e}(\mathbf{x}, t)$ at grid points in certain boundary-vicinity regions of each subpatch $\Omega^{\mathcal{R}}_{p,\ell}$; in this section we consider only communication between subpatches $\Omega^{\mathcal{R}}_{p,\ell}$ and $\Omega^{\mathcal{R}'}_{p',\ell'}$ with $(\mathcal{R}', p') \neq (\mathcal{R}, p)$ (that is, between subpatches of different patches). In detail, for every subpatch $\Omega^{\mathcal{R}}_{p,\ell}$, an "$n_f$-point fringe region", namely, a n_f-point wide layer of points adjacent to each one of the four sides of $\Omega^{\mathcal{R}}_{p,\ell}$ (with the exception of sides adjacent to Γ), is set to receive solution data from neighboring subpatches. (The concepts of side of a subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ is defined in Section 4.1.2.3.) In all the implementations presented in this chapter the value $n_f = 5$ was used. By construction, each point located in a subpatch fringe region must be contained in at least one other donor subpatch, and, as indicated above we consider here the case the donor subpatch is contained in different patch. Prior to every time step (or, more precisely, prior to each time-stage in the SSPRK-4 time-stepping method we use, see Section 4.2.1), for every point $\mathbf{x}$ in the fringe region of the subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ the solution value $\mathbf{e}(\mathbf{x})$ is overwritten with the value obtained for the same quantity by interpolation from an adjacent donor subpatch $\Omega^{\mathcal{R}'}_{p',\ell'}$. (Among the possibly multiple neighboring donor subpatches $\Omega^{\mathcal{R}'}_{p',\ell'}$ containing $\mathbf{x}$, with $(\mathcal{R}', p') \neq (\mathcal{R}, p)$, a donor subpatch is selected for which $\mathbf{x}$ is the farthest from the boundary of $(\mathcal{R}', p')$.) A 3×3-point stencil of $\mathcal{G}^{\mathcal{R}'}_{p',\ell'}$ grid points surrounding $\mathbf{x}$ and iterated 1D second order polynomial interpolation [70, Sec. 3.6], enacted on the basis of the Neville algorithm, is used to produce the necessary interpolated values, as illustrated in Figure 4.6.

With reference to Remark 10, the algorithm developed in this chapter includes a testing phase which, for a given patch/subpatch decomposition and grid assignments checks whether, as indicated in that remark, any data donor points in any patch are themselves receivers of solution data. If this donor-receiver condition is not satisfied, possible remedies include 1) Refining the grids by proportionally increasing the number of subpatches for all patches while keeping fixed the number of grid points per subpatch, as described in Section 4.1.2.4, or 2) Modifying the offending patches so as to adequately increase the sizes of the relevant overlap regions. An initial assignment of grid sizes aiming at satisfying the donor-receiver condition might be selected by ensuring that for each side of each patch $\Omega_p^{\mathcal{R}}$ that is not contained in Γ, a $2n_v + 1$-point wide layer of grid points adjacent to that side is also included in a union of one or more patches $\Omega_{p'}^{\mathcal{R}'}$ (with $(\mathcal{R}', p') \neq (\mathcal{R}, p)$.

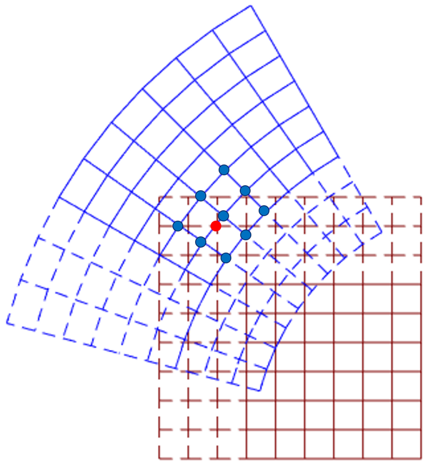


Figure 4.6: Illustration (with $n_f = 3$, for graphical simplicity) of the inter-patch data communication effected via exchanges between the blue and brown subpatches (which, as befits inter-patch communications, are subpatches of different patches). Dashed-line regions: receiving fringe regions for each subpatch. Red dot: a receiver point in the fringe region of the brown subpatch, set to receive solution data from the blue subpatch, via interpolation. Blue dots: the 3×3 donor-point stencil in the blue subpatch used for interpolation to the red receiver point. As indicated in Remark 10, none of the points in the donor stencil are located in the fringe region of the blue subpatch.

4.1.4.2 Intra-patch communication

Unlike the communication between pairs of adjacent patches considered in the previous Section 4.1.4.1, which proceeds via interpolation from one subpatch fringe region to another, the "intra-patch" communication, namely, communication of data among pairs of neighboring subpatches of a single patch, relies on the set of fringe points that are *shared* between such neighboring subpatches—in such a way that communication amounts merely to exchange of solution values as shown in Figure 4.7. As in the inter-patch case, intra-patch solution communications are subject to the donor-receiver condition outlined in Remark 10. Clearly, the donor-receiver condition is satisfied for a given patch $\Omega_p^{\mathcal{R}}$ provided every pair $\Omega_{p,\ell}^{\mathcal{R}}$ and $\Omega_{p,\ell'}^{\mathcal{R}}$ $((\ell, \ell') \in \Theta_p^{\mathcal{R}})$ of adjacent subpatches shares a $2n_f$-point wide layer of grid points along the side common to both subpatches, where the notion of side of a subpatch is introduced in Section 4.1.2.3. We note that, in view of the patch decomposition procedures described in Sections 4.1.2.1 and 4.1.2.2 (according to which subpatches of the same patch share a $(2n_v + 1)$-point wide overlap region with their neighbors), the subpatch donor-receiver condition is satisfied for the parameter values $n_f = 5$ and $n_v = 9$ used in this chapter.

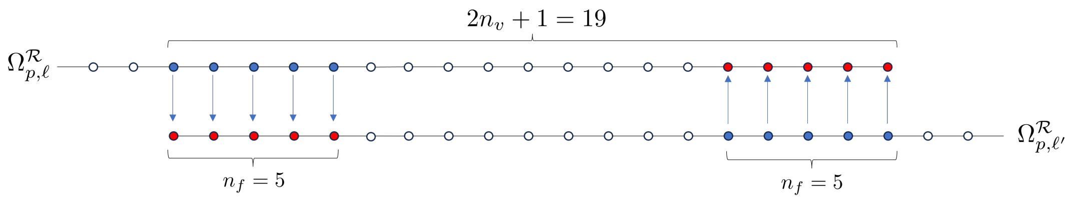

$2n_v + 1 = 19$
$\Omega^{\mathcal{R}}_{p,\ell}$
$\Omega^{\mathcal{R}}_{p,\ell'}$
$n_f = 5$
$n_f = 5$

Figure 4.7: One dimensional illustration of intra-patch communication between two subpatches $\Omega^{\mathcal{R}}_{p,\ell}$ and $\Omega^{\mathcal{R}}_{p,\ell'}$ of the same patch . Red dots: receiver points in the fringe regions of subpatches $\Omega^{\mathcal{R}}_{p,\ell}$ and $\Omega^{\mathcal{R}}_{p,\ell'}$. Blue dots: donor points in subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ (resp. $\Omega^{\mathcal{R}}_{p,\ell'}$) used for solution data communication with the receiving points of subpatch $\Omega^{\mathcal{R}}_{p,\ell'}$ (resp. $\Omega^{\mathcal{R}}_{p,\ell}$).

4.2 FC-based time marching

4.2.1 Spatio-temporal discretization

The proposed algorithm spatially discretizes the solution vector $\mathbf{e} = \mathbf{e}(\mathbf{x}, t)$ on the basis of q-dimensional families of vector grid functions

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

with $q = 4$ (other values of q are used in what follows to discretize other quantities). Here, with reference to (4.18) and (4.31), the overall spatial computational grid is given by

$$\mathcal{G} = \bigcup_{\mathcal{R}\in\mathbb{T}} \; \bigcup_{1\le p\le P_{\mathcal{R}}} \; \bigcup_{\ell\in\Theta^{\mathcal{R}}_p} \mathcal{G}^{\mathcal{R}}_{p,\ell}, \qquad \mathbb{T} = \{\mathcal{S}, C_1, C_2, \mathcal{I}\}. \tag{4.36}$$

For each vector grid function b, further, we write

$$b = (b^{\mathcal{R}}_{p,\ell}) \quad \text{and} \quad b^{\mathcal{R}}_{p,\ell,i,j} = b^{\mathcal{R}}_{p,\ell}(\mathbf{x}_{ij}), \quad \text{where } \mathbf{x}_{ij} = \mathcal{M}^{\mathcal{R}}_p(q^{\mathcal{R},1}_{p,i}, q^{\mathcal{R},2}_{p,j}),\ (i,j) \in \mathcal{D}^{\mathcal{R}}_{p,\ell},$$

in terms of the parameter space grid points $q^{\mathcal{R},1}_{p,i}$ and $q^{\mathcal{R},2}_{p,j}$ defined in (4.11) and (4.20), where $\mathcal{D}^{\mathcal{R}}_{p,\ell}$ is defined in (4.19) and (4.32). In particular, a spatially-discrete but time-continuous version $\mathbf{e}_h = \mathbf{e}_h(t)$ of the solution vector $\mathbf{e}(\mathbf{x}, t)$ is obtained as a $q = 4$ vector grid-function family $\mathbf{e}_h(t) = (\mathbf{e}^{\mathcal{R}}_{h,p,\ell}(t))$ over all relevant values of $\mathcal{R}$, p and ℓ; in accordance with the conventions above we may write $\mathbf{e}^{\mathcal{R}}_{p,\ell,i,j}(t) = \mathbf{e}^{\mathcal{R}}_{h,p,\ell}(\mathbf{x}_{ij}, t)$.

Like [6, Sec. 3.1], the proposed algorithm produces the necessary spatial derivatives on the basis of FC-based differentiation. In detail, approximations of the parameter-space partial derivatives of $\mathbf{e}$ are obtained via the sequential application of one-dimensional FC-differentiation (Section 2.3) in the q^1 or q^2 parameter variables (see Sections 4.1.2.1 and 4.1.2.2) on the basis of the point values $\mathbf{e}^{\mathcal{R}}_{p,\ell,i,j}(t) = \mathbf{e}^{\mathcal{R}}_{h,p,\ell}(\mathcal{M}^{\mathcal{R}}_p(q^{\mathcal{R},1}_{p,i}, q^{\mathcal{R},2}_{p,j}), t)$. Using such 1D discrete spatial differentiation operators to obtain each one of the $\mathbf{q}$ derivatives in the curvilinear equation (4.33), the resulting discrete equation may be expressed in the form

$$\frac{d\mathbf{e}_h(t)}{dt} = L[\mathbf{e}_h(t)], \tag{4.37}$$

where $L = L^{\mathcal{R}}_{p,\ell}$ denotes a family of discrete operators that incorporate the product of the Jacobian matrices and the difference of the divergences of the viscous and convective fluxes in that equation.

Following [5, 6], on the other hand, the algorithm's time-stepping proceeds via the 4-th order strong stability preserving Runge-Kutta scheme [38] (SSPRK-4). This scheme, which leads to low temporal dispersion and diffusion in smooth flow regions, is employed in conjunction with an adaptive time-step Δt selected at each time $t = t_n$ according to the expression

$$\Delta t = \frac{1}{\pi} \min_{(\mathcal{R},p)} \min_{\ell \in \Theta^{\mathcal{R}}_p} \frac{\mathrm{CFL}}{\max_{\mathbf{x}\in\mathcal{G}^{\mathcal{R}}_{p,\ell}} \left\{ |S[\mathbf{e}](\mathbf{x},t)| / \widetilde{h}_{\mathcal{R},p} \right\} + \max_{\mathbf{x}\in\mathcal{G}^{\mathcal{R}}_{p,\ell}} \left\{ \mu[\mathbf{e}]\mathbf{x},t) / (\widetilde{h}_{\mathcal{R},p})^2 \right\}}, \tag{4.38}$$

which generalizes the corresponding single patch version used in [5, 6]. Here $\widetilde{h}_{\mathcal{R},p}$ denotes the minimum mesh size in the patch $\Omega^{\mathcal{R}}_p$ (a lower-bound close to that minimum can be used, which can be obtained via consideration of the patch parametrization gradients), CFL denotes a problem-dependent constant parameter selected to ensure stability, and $S[\mathbf{e}](\mathbf{x},t)$ and $\mu[\mathbf{e}](\mathbf{x},t)$ denote the *maximum wave speed bound* (MWSB) and *artificial viscosity* operators introduced in Sections 4.3.1 and 4.3.2.

4.2.2 Enforcement of boundary conditions and overlap communications

Following the "conventional method" described in [39], boundary conditions are enforced at each intermediate stage of the Runge-Kutta scheme, and both Dirichlet and Neumann boundary conditions are incorporated as part of the differentiation process described in the previous section. Dirichlet boundary conditions at the time-stage $t_{n,\nu}$ ($t_n < t_{n,\nu} \leq t_{n+1}$) for the ν-th SSPRK-4 stage ($\nu = 1, \ldots, 4$) of the time-step starting at $t = t_n$, are simply imposed by overwriting the boundary values of the unknown solution vector $\mathbf{e}_h$ 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, in turn, are enforced by constructing appropriate Fourier continuations after each stage of the SSPRK-4 scheme on the basis of the modified pre-computed matrix $\widetilde{Q}$ mentioned in Section 2.3; see also Remark 11 below. Similarly, patch/subpatch communication is enforced at every SSPRK-4 stage, by enacting both the inter- and intra-patch data exchanges described in Section 4.1.4 and overwriting the corresponding solution values. In particular, we have found that imposing patch/subpatch communication at every stage leads to smoother numerical solutions, particularly at patch-boundary regions, than are obtained from a single enforcement at the end of every SSPRK-4 time step.

Remark 11. The proposed algorithm enforces adiabatic boundary conditions (vanishing of the normal gradient of the temperature T) at all obstacle boundaries (as illustrated in Section 4.5.2). Since the temperature is not one of the unknowns utilized in the equation (2.6), the enforcement of an adiabatic condition requires consideration of (2.10). The proposed algorithm thus proceeds as follows: 1) at the beginning of each SSPRK-4 time stage, the solution $\mathbf{e}_h$ is used in conjunction with equation (2.10) to obtain the grid values of T at all grid points in the interior of the computational domain, and then the adiabatic boundary condition is employed to obtain the values of T at the physical boundaries by producing, on the basis of the method described in [2, Sec. 6.3], a Fourier Continuation expansion that matches the interior data and is consistent with a vanishing normal gradient of the temperature T; 2) the Fourier Continuation expansion of T obtained per point 1 is then used together with the values of the density ρ and the horizontal and vertical momenta ρu and ρv that are part of the existing vector $\mathbf{e}_h$ (but ignoring the the values of E that are also part of $\mathbf{e}_h$) to evaluate the derivatives of E and p with respect to ξ and η by differentiation of equations (2.9) and (2.10) on the basis of the sum and product differentiation rules. Having obtained spatial derivatives of E and p with respect to ξ and η that are consistent with the adiabatic boundary conditions, the time stage proceeds by 3) evaluating the spatial derivatives of the remaining components of $\mathbf{e}_h$ (namely, ρ, ρu, ρv) on the basis Fourier Continuations of the existing values of these quantities under the existing boundary conditions for u and v (no boundary conditions are imposed on ρ at physical boundaries where, in accordance with the physics of the problem, this quantity is evolved in the same manner as at the interior grid points).

4.2.3 Spectral filtering

Spectral methods regularly utilize filtering strategies in order to control the error growth in the unresolved high frequency modes, and the present method is not an exception: like the approaches [1, 2], the proposed algorithm incorporates a multipatch spectral filter in conjunction with the Fourier Continuation expansions at the level of each subpatch. Additionally, a localized initial-data filtering strategy (which is only applied at the initial time, and then only at subpatches that contain discontinuities in the initial data), is used to regularize discontinuous initial conditions, while avoiding over-smearing of smooth flow-features. (The localized filtering approach was introduced in [6] in the context of a single-grid FC-based shock-dynamics solver.) Details regarding the aforementioned multipatch and localized filtering strategies are provided in Sections 4.2.3.1 and 4.2.3.2.

4.2.3.1 Subpatch-wise filtering strategy

To introduce the subpatch-wise filtering method we use we first consider filtering of a one-dimensional Fourier Continuation expansion of the form

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

for which the corresponding filtered expansion is given by

$$\widetilde{F} = \sum_{k=-M}^{M} \hat{F}_k^c \sigma\Big(\frac{2k}{N+C}\Big) \exp(2\pi i k x/\beta) \tag{4.39}$$

where

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

with adequately chosen values of the positive integer p_f and the real parameter $\alpha_f > 0$. In order to filter a function defined on a two-dimensional subpatch of the type that underlies the discretizations considered in this chapter, the spectral filter is applied sequentially, one dimension at a time—thus filtering in accordance with (4.39), in the q^1 subpatch parameter variable for each value of q^2, and then applying the reverse scheme to the resulting function, filtering with respect to q^2 for each value of q^1.

In the strategy proposed in this chapter, Fourier Continuation expansions of the density ρ, the horizontal and vertical momenta ρu and ρv and the temperature T are computed and filtered independently on each grid $G^{\mathcal{R}}_{p,\ell}$, at every time step following the initial time (but not at individual SSRPK-4 stages), and using the parameter values $\alpha_f = 10$ and $p_f = 14$. The energy E component of the solution vector $\mathbf{e}_h$ is not directly filtered: per Remark 11, the quantities E and p are re-expressed in terms of the filtered versions of ρ, ρu and ρv and T, and the necessary derivatives of E and p are obtained via direct differentiation of equations (2.9) and (2.10). Thus, the quantities E and p filtered indirectly, by means of the filters applied to the other flow variables. This approach makes it possible to incorporate the adiabatic boundary conditions (while simultaneously filtering E and p) even in absence of the temperature T in the vector $\mathbf{e}$ of unknowns.

4.2.3.2 Localized discontinuity-smearing for initial data

To avert the formation of spurious oscillations originating from a discontinuous initial conditions without unduly smearing smooth flow features, before the initiation of the time-stepping procedure the initial data is filtered (using a strong form of the spectral filter (4.39)) in all subpatches where initial data discontinuities exist. This procedure, which generalizes the localized initial-data filtering approach introduced in [6, Sec. 3.4.2] to the present multi-patch context, proceeds as follows.

For each subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ within which the initial condition $\mathbf{e}(\mathbf{x}, 0)$ is discontinuous, the localized filtering is performed one dimension at a time in the parameter-space grid $G^{\mathcal{R}}_{p,\ell}$ and for the physical quantities $F \in \{\rho, \rho u, \rho v, T\}$, in an approach similar, but different, to the one used in the previous section. In detail, considering each restriction of F to the parameter-space direction q^i $(i = 1, 2)$, containing a discontinuity at a single point z, the smeared-discontinuity function $\widetilde{F}_{\mathrm{sm}}$ combines the 1D filtered function $\widetilde{F}$ in a neighborhood of the discontinuity with the unfiltered function elsewhere by means of the expression

$$\widetilde{F}_{\mathrm{sm}}(x) = w^{\mathcal{R},m}_{p,c,r}(x - z)\widetilde{F}(x) + (1 - w^{\mathcal{R},m}_{p,c,r}(x - z))F(x) \quad , m = 1, 2. \tag{4.40}$$

Here, the filtered function $\widetilde{F}$ is obtained by applying (4.39) with filter parameters $\alpha_f = 10$ and $p_f = 2$, and by using, for $m = 1, 2$, the window functions $w^{\mathcal{R},m}_{p,c,r}$ given by

$$w^{\mathcal{R},m}_{p,c,r}(x) = \begin{cases} 1 & \text{if } |x| < \frac{c}{2}h^{\mathcal{R},m}_p \\ \cos^2\left(\frac{\pi(|x| - \frac{c}{2}h^{\mathcal{R},m}_p)}{rh}\right) & \text{if } \frac{c}{2}h^{\mathcal{R},m}_p \leq |x| \leq (\frac{c}{2} + r)h^{\mathcal{R},m}_p \\ 0 & \text{if } |x| > (\frac{c}{2} + r)h^{\mathcal{R},m}_p, \end{cases} \tag{4.41}$$

with $c = 18$ and $r = 9$. As in [6, Sec. 3.4.2], in case multiple discontinuities exist along a given 1D parameter direction in a given subpatch, 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.

4.3 Multi-patch artificial viscosity assignment

The proposed strategy for assignment of artificial viscosity values $\mu[\mathbf{e}] = \mu[\mathbf{e}](\mathbf{x}, t)$ relies on an extension of the method described in [6, Secs. 3.2, 3.3] to the multi-patch setting utilized in the present contribution. The proposed artificial-viscosity algorithm proceeds by producing, at first, a preliminary subpatch-wise viscosity assignment, on the basis of a variation of the aforementioned method [6]; the details of the new multi-patch algorithm are presented in Section 4.3.1. In order to ensure smoothness of the artificial viscosity assignment across subpatches (and, in particular, at the interfaces of neighboring subpatches), a novel windowed-viscosity propagation procedure introduced in Section 4.3.2 is used in conjunction with the preliminary viscosity assignment mentioned above.

4.3.1 Subpatch-wise preliminary viscosity assignment

The subpatch-based viscosity assignment procedure described in what follows, which determines preliminary artificial viscosity values $\widehat{\mu}[\mathbf{e}](\mathbf{x}, t)$ independently for each subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ (with $\mathcal{R} = \mathcal{S}$, C_1, C_2, or $\mathcal{I}$, $1 \le p \le P_{\mathcal{R}}$ and $\ell \in \Theta^{\mathcal{R}}_p$), proceeds on the basis of the "degree of smoothness" of a certain "proxy variable" function $\Phi(\mathbf{e})(\mathbf{x}, t)$ of the unknown solution vector $\mathbf{e}$ on $\Omega^{\mathcal{R}}_{p,\ell}$. Following [6] and [5] we use the proxy variable $\Phi(\mathbf{e})$ equal to the Mach number $\Phi(\mathbf{e}) = \|\mathbf{u}\| \sqrt{\frac{\rho}{\gamma p}}$. As detailed in what follows, utilizing this proxy variable, a smoothness-classification operator $\tau = \tau[\Phi(\mathbf{e})]$ characterizes the degree of smoothness of the function $\Phi(\mathbf{e})$ at a certain time t and over each subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ by analyzing the oscillations of restrictions of $\Phi(\mathbf{e})$ to certain subsets of the subpatch grid $\mathcal{G}^{\mathcal{R}}_{p,\ell}$ (where the grid subsets used are contained in regions including interior portions of the subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ as well as certain subpatch regions near physical domain boundaries).

The smoothness-classification operator τ for each such region is obtained via consideration of approximations of FC expansions of $\Phi(\mathbf{e}) = \Phi(\mathbf{e}^{\mathcal{R}}_{p,\ell})$ over $\Omega^{\mathcal{R}}_{p,\ell}$ (see Section 4.1.3) that are obtained from the discrete numerical solution values

$\phi^{\mathcal{R}}_{p,\ell} = \Phi(\mathbf{e}^{\mathcal{R}}_{h,p,\ell})$ on $\mathcal{G}^{\mathcal{R}}_{p,\ell}$—and we thus use the approximation $\tau[\mathbf{e}^{\mathcal{R}}_{p,\ell}] \approx \tilde{\tau}[\phi^{\mathcal{R}}_{p,\ell}]$ that defines the discrete operator $\tilde{\tau}$. For each grid $\mathcal{G}^{\mathcal{R}}_{p,\ell}$ the discrete operator $\tilde{\tau}$ is produced on the basis of an application of the 2D algorithm presented in [6, Sec. 3.2.1]. When applied to the subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ in the present multi-patch context, however, the previous algorithm [6, Sec. 3.2.1] is used to obtain smoothness classification of the solution $\mathbf{e}^{\mathcal{R}}_{p,\ell}$ only for a certain "viscosity-generation subgrid" $\widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell} \subset \mathcal{G}^{\mathcal{R}}_{p,\ell}$ described in what follows. The fact that smoothness classification is only carried out over the subgrid $\widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell}$ relates to the overlap of patches and subpatches in the discretization structure we use: in the present context, in order to properly account for overlaps, the algorithm [6, Sec. 3.2.1] is only applied to stencils of points centered at grid points in the set $\widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell}$. Per the prescriptions in [6, Sec. 3.2.1], the values of the operator $\tilde{\tau}$ are produced numerically on the basis of an Artificial Neural Network (ANN) whose architecture and training procedure are described in [6, Sec. 3.2.2]. (The ANN weights and biases are loaded from disc at the FC-SDNN initialization stage; see Algorithm (5).)

The motivation for the introduction of the viscosity-generation subgrid $\widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell}$ is twofold. On one hand smoothness classification values, which must be produced for all patch discretization points $\mathcal{G}^{\mathcal{R}}_{p}$ (equations (4.18) and (4.31)), are produced on the basis of FC expansions at the level of subpatch grids $\mathcal{G}^{\mathcal{R}}_{p,\ell}$, and, since subpatches overlap, a decision must be made as to which classification value is used at a grid point in $\mathcal{G}^{\mathcal{R}}_{p}$ that belongs to the sets $\mathcal{G}^{\mathcal{R}}_{p,\ell}$ for two or more values of ℓ. The decision is dictated on the basis of accuracy: since the FC approximation, and, thus, the smoothness classification algorithm, provide more accurate results at interior points of the subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ than at points near its boundary, for $\mathcal{R} = \mathcal{S}$, C_1, C_2 and $\mathcal{I}$ we define the subgrid $\widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell}$ as the set of all points in $\mathcal{G}^{\mathcal{R}}_{p,\ell}$ that are not contained in the fringe region $\mathcal{F}^{\mathcal{R}}_{p,\ell,n_v}$ defined in (4.34):

$$\widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell} \coloneqq \mathcal{G}^{\mathcal{R}}_{p,\ell} \setminus \mathcal{F}^{\mathcal{R}}_{p,\ell,n_v}. \tag{4.42}$$

The integer n_v used here equals the one utilized in Sections 4.1.2.1 and 4.1.2.2; as indicated in those sections, the value $n_v = 9$ is used throughout this chapter. The set of all indices of grid points in $\widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell}$ is denoted by $\widetilde{\mathcal{D}}^{\mathcal{R}}_{p,\ell}$: using the notations (4.11) and (4.20) we have

$$\widetilde{\mathcal{D}}^{\mathcal{R}}_{p,\ell} = \{(i,j) \in \mathcal{D}^{\mathcal{R}}_{p,\ell} \mid \mathcal{M}^{\mathcal{R}}_{p}(q^{\mathcal{R},1}_{p,i}, q^{\mathcal{R},2}_{p,j}) \in \widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell}\}. \tag{4.43}$$

The introduction of the preliminary subpatch-wise viscosity assignment mentioned in the first paragraph of Section 4.3, which is the main objective of the present section and is given in (4.45), is based on use of the viscosity-generation subgrid introduced above together with slight variants, summarized in what follows, of a number of functions and operators introduced in [6, Sec. 3.3]. In detail, the definition (4.45) utilizes

1. The weight function R, which assigns viscosity weights to the smoothness classification, and is given by $R(1) = 1.5$, $R(2) = 1$, $R(3) = 0.5$, and $R(4) = 0$;

2. The grid function operator $\widetilde{R}$ defined by $(\widetilde{R}[\eta]_{ij} = R(\eta_{ij}))$;

3. The MWSB operator $S[\mathbf{e}]$ defined as the upper bound $S(\mathbf{e}) = |u| + |v| + a$ 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 $\mathbf{u} \cdot \vec{\kappa} + a$ [46, Sec. 16.3 and 16.5]);

4. The discrete version $\widetilde{S}[\mathbf{e}_h]$ of the operator introduced in pt. 3 above, defined by
$$\widetilde{S}[\mathbf{e}^{\mathcal{R}}_{h,p,l}]_{ij} = |u^{\mathcal{R}}_{p,i,j}| + |v^{\mathcal{R}}_{p,i,j}| + a^{\mathcal{R}}_{p,i,j}; \tag{4.44}$$

5. The 7×7 localization stencils $L^{i,j}$ $((i,j) \in \tilde{\mathcal{D}}^{\mathcal{R}}_{p,\ell})$ (with $(\mathcal{R}, p, \ell)$-dependence suppressed in the notation) defined as the sets of points in $\mathcal{G}^{\mathcal{R}}_{p,\ell}$ that surround the grid point with index (i,j) in $\mathcal{G}^{\mathcal{R}}_{p,\ell}$. Their definition is identical to the localization stencils defined in [6, Sec. 3.3];

6. The discretization parameter $\widehat{h}_{\mathcal{R},p}$ equal to the maximum spacing between two consecutive discretization points in the patch $\Omega^{\mathcal{R}}_p$ (an upper-bound close to that maximum can be used, which can be obtained via consideration of the patch parametrization gradients).

Using these operators and functions, the preliminary artificial viscosity operator $\widehat{\mu}[\mathbf{e}^{\mathcal{R}}_{p,\ell}]$ is defined by

$$\widehat{\mu}[\mathbf{e}^{\mathcal{R}}_{p,\ell}]_{i,j} = \widetilde{R}(\tilde{\tau}[\phi^{\mathcal{R}}_{p,\ell}]_{i,j}) \cdot \max_{(k,\ell)\in L^{i,j}} (S[\mathbf{e}^{\mathcal{R}}_{p,\ell}]_{k\ell})\widehat{h}_{\mathcal{R},p}, \quad (i,j) \in \tilde{\mathcal{D}}^{\mathcal{R}}_{p,\ell}. \tag{4.45}$$

4.3.2 Overall viscosity operator $\mu^{\mathcal{R}}_{p,\ell} = \mu^{\mathcal{R}}_{p,\ell}[\mathbf{e}]$

The overall multi-patch artificial viscosity operator used in this chapter is obtained by exploiting a certain smoothing-blending (SB) operator Λ that smoothes the viscosity values provided by the subpatch-wise preliminary artificial viscosity operator (4.45), and that additionally combines the smooth values thus obtained in the various subpatches. The application of the SB operator ensures a well-defined and spatially smooth time-dependent artificial viscosity profile over the complete multi-patch computational domain Ω.

To introduce the SB operator Λ we first utilize the window functions $w_{p,c,r}^{\mathcal{R},m}$ (4.41) to define, for $\mathcal{R} = \mathcal{S}, \mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{I}$, and for each $(i,j) \in \widetilde{\mathcal{D}}_{p,\ell}^{\mathcal{R}}$ (that is to say, for each point $\mathcal{M}_p^{\mathcal{R}}(q_{p,i}^{\mathcal{R},1}, q_{p,j}^{\mathcal{R},2})$ in the viscosity-generation subgrid $\widetilde{\mathcal{G}}_{p,\ell}^{\mathcal{R}}$ of $\mathcal{G}_{p,\ell}^{\mathcal{R}}$), the family of subpatch windowing functions

$$W_{p,\ell,i,j}^{\mathcal{R}}(\mathbf{x}) = w_{p,0,9}^{\mathcal{R},1}(q^1 - q_{p,i}^{\mathcal{R},1}) w_{p,0,9}^{\mathcal{R},2}(q^2 - q_{p,j}^{\mathcal{R},2}) \tag{4.46}$$

where $\mathbf{q} = (q^1, q^2) = (\mathcal{M}_p^{\mathcal{R}})^{-1}(\mathbf{x})$ for $\mathbf{x} \in \mathcal{G}_{p,\ell}^{\mathcal{R}}$ We then introduce the family of multi-patch windowing functions

$$\mathcal{W}_{p,\ell,i,j}^{\mathcal{R}}(\mathbf{x}) = \begin{cases} W_{p,\ell,i,j}^{\mathcal{R}}(\zeta_{p,\ell}^{\mathcal{R}}(\mathbf{x})) & \text{if} \quad \mathbf{x} \in \Omega_{p,\ell}^{\mathcal{R}} \\ 0 & \text{if} \quad \mathbf{x} \notin \Omega_{p,\ell}^{\mathcal{R}} \end{cases}, \tag{4.47}$$

where $\zeta_{p,\ell}^{\mathcal{R}}(\mathbf{x})$, denotes the point in the real-space grid $\mathcal{G}_{p,\ell}^{\mathcal{R}}$ that is “closest” to $\mathbf{x}$ in the following sense: $\zeta_{p,\ell}^{\mathcal{R}}(\mathbf{x})$ equals the image of the point in the grid $G_{p,\ell}^{\mathcal{R}}$ which is the closest to the pre-image of $\mathbf{x}$ under the patch mapping $\mathcal{M}_p^{\mathcal{R}}$. In some cases this definition requires disambiguation, which is achieved as follows: letting $\mathbf{q} = (q^1, q^2) = (\mathcal{M}_p^{\mathcal{R}})^{-1}(\mathbf{x})$ we define

$$\zeta_{p,\ell}^{\mathcal{R}}(\mathbf{x}) = \mathcal{M}_p^{\mathcal{R}}(\mathrm{rnd}(q^1/h_p^{\mathcal{R},1})h_p^{\mathcal{R},1}, \mathrm{rnd}(q^2/h_p^{\mathcal{R},2})h_p^{\mathcal{R},2}), \quad \mathbf{x} \in \Omega_{p,\ell}^{\mathcal{R}}, \tag{4.48}$$

where for $t \in \mathbb{R}^+$, $\mathrm{rnd}(t)$ denotes the integer that is closest to t, disambiguated by $\mathrm{rnd}(n + \frac{1}{2}) = n + 1$ for every integer n.

Using these notations, finally, the normalized multi-patch windowing functions $\widetilde{\mathcal{W}}_{p,\ell,i,j}^{\mathcal{R}}$, and, generalizing the windowed-localization operator (3.21) and the viscosity operator (3.22), the multi-patch windowing operator Λ, and the overall viscosity operator $\mu_{p,\ell}^{\mathcal{R}}$ are defined by

$$\widetilde{\mathcal{W}}_{p,\ell,i,j}^{\mathcal{R}}(\mathbf{x}) = \frac{\mathcal{W}_{p,\ell,i,j}^{\mathcal{R}}(\mathbf{x})}{\sum_{\mathcal{R}'} \sum_{1 \le p' \le P_{\mathcal{R}}} \sum_{\ell' \in \Theta_{p'}^{\mathcal{R}'}} \sum_{(i',j') \in \widetilde{\mathcal{D}}_{p',\ell'}^{\mathcal{R}'}} \mathcal{W}_{p',\ell',i',j'}^{\mathcal{R}'}(\mathbf{x})}, \quad \mathbf{x} \in \Omega, \tag{4.49}$$

$$\Lambda[b](\mathbf{x}) = \sum_{\mathcal{R}} \sum_{1 \le p \le P_{\mathcal{R}}} \sum_{\ell \in \Theta_p^{\mathcal{R}}} \sum_{(i,j) \in \widetilde{\mathcal{D}}_{p,\ell}^{\mathcal{R}}} \widetilde{\mathcal{W}}_{p,\ell,i,j}^{\mathcal{R}}(\mathbf{x}) b_{p,\ell,i,j}^{\mathcal{R}} \quad \text{and} \tag{4.50}$$

$$\mu_{p,\ell}^{\mathcal{R}}[\mathbf{e}](\mathbf{x}) = \Lambda[\widehat{\mu}(\mathbf{e}](\mathbf{x}), \quad \mathbf{x} \in \mathcal{G}_{p,\ell}^{\mathcal{R}}, \tag{4.51}$$

respectively.

Remark 12. We note that the viscosity operator $\mu^{\mathcal{R}}_{p,\ell}$ is constructed on the basis of the preliminary viscosity operator $\widehat{\mu}$, which relies on an associated smoothness classification operator $\tilde{\tau}$— that is computed over grids $\widetilde{\mathcal{G}}^{\mathcal{R}}_{p,\ell}$ contained in subpatch interiors, *except for subpatches that are adjacent to the boundary of* Ω, for which smoothness classification near subpatch boundaries cannot be avoided. Numerical experiments have shown that favoring smoothness classification values produced by stencils located in subpatch interiors helps eliminate spurious oscillations that would otherwise arise as shocks or contact discontinuities travel from one subpatch to the next. Clearly, the use of such interior values requires reliance on window functions with a relatively large support, as well as a sufficiently large overlap region between neighboring subpatches.

4.4 Multi-domain strategy: Parallelization

Sections (4.4.1) and (4.4.2) describe our parallel implementation of the FC-SDNN algorithm presented above in this chapter. The description assumes the algorithm is run in a number N_r of parallel MPI ranks, where each rank (that is to say, each individual parallel MPI process) is assumed to be pinned to a single compute core. Of course the code can be run in serial mode, simply by selecting $N_r = 1$.

4.4.1 Parallelization strategy

The structure of the proposed algorithm, which is based on use of multiple patches and subpatches, as outlined in Section 4.1.2 (with potentially arbitrarily large allowable numbers of subpatches for a fixed patch decomposition of a given geometry), wherein each subpatch is essentially independent of all other subpatches (with exception of a minimal amount of required communication between neighboring subpatches), was designed with a triple goal of 1) Enabling applicability to general geometries, and 2) Limiting the size of the required Fourier-continuation expansions, while 3) Lending itself to effective parallelization in a distributed parallel computing environment. Concerning point 3), in particular, relying on the underlying overlapping patch-decomposition, the main algorithmic components of the method are embarrassingly parallel, including:

(i) The computation of the subpatch-wise preliminary viscosity $\widehat{\mu}[\mathbf{e}^{\mathcal{R}}_{p,\ell}]$ (equation (4.45)) described in Algorithm 2 below, which requires the evaluation of the proxy variable $\phi^{\mathcal{R}}_{p,\ell}$, the smoothness classification values $\tilde{\tau}[\phi^{\mathcal{R}}_{p,\ell}]$, and the MWSB operator $\widetilde{S}[\mathbf{e}_h]$ (see Section 4.3.1 and, in particular, equation (4.44)).

(ii) The computation and filtering of the FC-expansions of the solution followed by subpatch-wise filtering (Section 4.2.3.1) required by the time-stepping procedure (Section (4.2)), and the computation, filtering and windowing of the FC-expansions of initial data for localized discontinuity smearing (Section 4.2.3.2), as detailed in Algorithm 3.

(iii) The evaluation of each stage of the SSPRK-4 time stepping scheme (Section 4.2.1) using FC-based spatial differentiations incorporating the required Dirichlet and/or Neumann boundary conditions (Section 4.2.2), as indicated in Algorithm 4.

These parallel segments are delimited by the necessary MPI inter-rank data communications, namely, the communication of solution values (Sections 4.1.4.1 and 4.1.4.2) and the communication of viscosity data (Section 4.3.2)— which allow for the parallelization of the multi-patch FC-SDNN procedure in a distributed parallel computing environment.

Algorithm 2 Parallel evaluation of the preliminary artificial viscosity operator

```
1: for every MPI rank $n_r$ do
2:     for every subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ distributed to rank $n_r$ do
3:         Evaluate the proxy variable $\phi$ corresponding to $\mathbf{e}_h$ at all spatial grid points (Section 4.3.1).
4:         Obtain the smoothness classification values ($\tilde{\tau}[\phi]$ (Section 4.3.1).
5:         Evaluate the MWSB operator $\widetilde{S}[\mathbf{e}_h]$ at all spatial grid points (Equation (4.44)).
6:         Evaluate the artificial viscosity operator $\widehat{\mu}[\mathbf{e}^{\mathcal{R}}_{p,\ell}]$ (Equation (4.45)).
7:     end for
8: end for
```

Algorithm 3 Parallel filtering of the solution vector $\mathbf{e}^{\mathcal{R}}_{h,p,\ell}$

```
1: for every MPI rank $n_r$ do
2:     for every subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ distributed to rank $n_r$ do
3:         (Case $t = 0$) Apply localized discontinuity-smearing (Section 4.2.3.2) to the solution vector $\mathbf{e}^{\mathcal{R}}_{h,p,\ell}$ and overwrite $\mathbf{e}^{\mathcal{R}}_{h,p,\ell}$ with the resulting values.
4:         (Case $t > 0$) Apply the subpatch-wise filtering strategy (Section 4.2.3.1) to the solution vector $\mathbf{e}^{\mathcal{R}}_{h,p,\ell}$ and overwrite $\mathbf{e}^{\mathcal{R}}_{h,p,\ell}$ with the resulting values.
5:     end for
6: end for
```

Algorithm 4 Parallel stage evolution and enforcement of boundary conditions for the solution vector $\mathbf{e}^{\mathcal{R}}_{h,p,\ell}$

```
1: for every MPI rank $n_r$ do
2:     for every subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ distributed to rank $n_r$ do
3:         Compute the spatial gradient of the viscosity $\mu^{\mathcal{R}}_{p,\ell}$ (Equation (4.51)) via
           FC-based differentiation (Section 2.3).
4:         Compute the first- and second-order spatial partial derivative of the
           density $\rho$, horizontal and vertical momenta $\rho u$ and $\rho v$ and temperature $T$ via
           FC-based differentiation while enforcing the relevant Dirichlet or Neumann
           boundary conditions (Remark 11).
5:         Using the derivatives obtained in lines 3 and 4 together with the Jacobian
           and the product differentiation rule, evaluate the discrete differential operator
           $L^{\mathcal{R}}_{p,\ell}$ (Equation (4.37)) by combining these derivatives and the Jacobian $J^{\mathcal{R},p}_{\mathbf{q}}$ of
           the inverse mapping $(\mathcal{M}^{\mathcal{R}}_p)^{-1}$ (Equation (4.33)).
6:         Evaluate the next stage of the SSPRK-4 scheme.
7:     end for
8: end for
```

4.4.2 Initialization and overall multi-patch FC-SDNN algorithm pseudo-codes

A pseudo-code for the complete multi-patch FC-SDNN algorithm is presented in Algorithm 6, with a preliminary initialization stage outlined in Algorithm 5.

4.5 Numerical results

This section presents computational results produced by means of the multi-patch FC-SDNN method in a number of challenging test cases, including problems involving interactions between supersonic/hypersonic flow and obstacles, and including multiple moving-shocks and contact discontinuities, as well as shock collisions with obstacles, domain boundaries, contacts, and other shocks, etc. The efficiency of the parallel implementation of the algorithm in terms of weak scaling is demonstrated in Section 4.5.1, and a number of illustrative numerical examples produced by the multi-patch FC-SDNN algorithm are presented in Section 4.5.2.

The numerical tests presented in this chapter were run on our *Wavefield* cluster, which consists of 30 dual-socket nodes connected via HDR Infiniband. Each socket incorporates two 28-core Intel Xeon Platinum 8273 processors, for a total of 56 cores per node, and 384 GB of GDDR4 RAM per node. While supported by the Xeon processors, the hyper-threading capability was not utilized in any of the numerical examples presented in this chapter.

Algorithm 5 Multi-patch FC-SDNN initialization

```
1: for every MPI rank $n_r$ do
2:     Initialize the complete patch/subpatch structure, detailing explicit mapping functions, the subpatch decomposition, the subpatch discretization sizes, and the subpatch numbering, but not including assignments of arrays of discretization points. (The resulting data repetition across ranks is not memory-demanding and is thus used.)
3:     Assign the subpatches $\Omega^{\mathcal{R}}_{p,\ell}$ to the various ranks so as to ensure as close to equidistribution of subpatches per rank as possible.
4:     for every subpatch $\Omega^{\mathcal{R}}_{p,\ell}$ distributed to rank $n_r$ do
5:         Allocate the data arrays necessary for storage of discretization points, viscosity values, stage solution values, and solution and viscosity data communication.
6:         Compute the physical grids $\mathcal{G}^{\mathcal{R}}_{p,\ell}$ (Sections 4.1.2.1 and 4.1.2.2) using the mappings on the parameter grid points $(q^1_i, q^2_j) \in G^{\mathcal{R}}_{p,\ell}$ (that are solely determined by the patch/subpatch decomposition and the discretization sizes initialized in line 2).
7:         Allocate, compute and store the values of the Jacobian $J^{\mathcal{R},p}_{\mathbf{q}}$ of the inverse mapping $\left(\mathcal{M}^{\mathcal{R}}_p\right)^{-1}$ (Section 4.1.3) over all grid points in $\mathcal{G}^{\mathcal{R}}_{p,\ell}$.
8:         Load the trained ANN weights and biases (Section 4.3.1).
9:         Initialize the unknown solution vector $\mathbf{e}_h = \mathbf{e}^{\mathcal{R}}_{h,p,\ell}$ (Section 2.1) to the given initial-condition values over all spatial discretization grid $\mathcal{G}^{\mathcal{R}}_{p,\ell}$.
10:        Compute (through communication with the other subpatches) the values of the normalized multi-patch windowing function $\widetilde{\mathcal{W}}^{\mathcal{R}}_{p,\ell,i,j}(\mathbf{x})$ (equation (4.49)) for all $\mathbf{x} \in \mathcal{G}^{\mathcal{R}}_{p,\ell}$.
11:    end for
12: end for
```

4.5.1 Parallel performance: Weak and strong scaling

This section demonstrates the weak and strong parallel scalability enjoyed by our implementation of the multi-patch FC-SDNN algorithm. In particular, Sections 4.5.1.3 and 4.5.1.4 present weak scaling results: in Section 4.5.1.3 the sizes of the problems are increased by increasing the size and complexity of the physical problems considered, whereas in Section 4.5.1.4 the problem sizes are increased via discretization refinement for a fixed physical problem. Section 4.5.1.5, in turn, demonstrates the strong scaling of the algorithm using the physical problem utilized in Section 4.5.1.4. The two types of physical problems considered are described in Section 4.5.1.1, and parallel scalability metrics used are introduced in Section 4.5.1.2.

Algorithm 6 Multi-patch FC-SDNN algorithm

```
1: begin
2:     \\Initialization.
3:     Initialize the mesh and the multi-patch FC-SDNN solver. (Algorithm 5.)
4:     Initialize time: $t = 0$.
5:     \\Time stepping.
6:     while $t < T$ do
7:         In parallel, evaluate the preliminary artificial viscosity operator $\widehat{\mu}[\mathbf{e}^{\mathcal{R}}_{p,\ell}]$ (Algorithm 2.)
8:         In parallel, communicate the preliminary artificial viscosity values obtained in line 7 to all patches and subpatches, as needed for evaluation of the overall artificial viscosity $\mu^{\mathcal{R}}_{p,\ell}[\mathbf{e}]$ (equation (4.51)) in each subpatch grid,
9:         In parallel, combine, for each subpatch grid, the values communicated in line 8 to produce $\mu^{\mathcal{R}}_{p,\ell}[\mathbf{e}]$ on the subpatch grid.
10:        Filter the solution vector $\mathbf{e}^{\mathcal{R}}_{h,p,\ell}$ in parallel. (Algorithm 3.)
11:        Evaluate of the temporal step-size $\Delta t$ (equation (4.38)).
12:        for each stage of the SSPRK-4 time stepping scheme do
13:            In parallel, evolve the solution vector $\mathbf{e}^{\mathcal{R}}_{h,p,\ell}$ for the SSPRK-4 stage and enforce boundary conditions. (Algorithm 4.)
14:            In parallel, communicate the solution values between neighboring patches and subpatches via exchange and interpolation, as relevant (Sections 4.1.4.1 and 4.1.4.2).
15:        end for
16:        Update time: $t = t + \Delta t$
17:        Write solution values to disk at specified time steps $t$.
18:    end while
19: end
```

4.5.1.1 Scaling I: Test problems

Test Problem 1: Mach 10 shock mitigation by matrices of cylindrical obstacles. Test Problem 1 actually comprises a set of problems in which a Mach 10 shock-wave (a shock wave traveling at 10 times the speed of sound of the unperturbed fluid) impinges upon a rectangular array of circular cylinders containing n_{col} columns and n_{row} rows of cylinders, as depicted in the left panel of Figure 4.8. The computational domain considered is divided into three zones, namely, an obstacle-free front region ahead of the rectangular array; a middle region containing the rectangular array of obstacles; and an obstacle-free wake region. The middle $n_{\text{col}} \times n_{\text{row}}$-cylinder region is constructed as a corresponding array of identical overlapping rectangular subdomains of horizontal and vertical sides of lengths $\ell_x = 3$ and $\ell_y = 2$, respectively, each one of which consists of 54 subpatches arranged around a single cylinder of radius 0.25. One such rectangular subdomain is depicted in the right panel of Figure 4.8. The front and wake region comprise a vertical array of n_{row} rows of rectangular cylinder-free patches spanning an area of length $L_x = 2.5$ and $L_y = \ell_y = 2$, each one of which once again contains 54 rectangular subpatches.

The problem prescriptions are completed by incorporating the following initial conditions and boundary conditions. The initial conditions utilized are given by

$$(\rho, u, v, p) = \begin{cases} (4.0816, 8.25, 0, 116.5) & \text{if} \quad x \le 0.6 \\ (1.4, 0, 0, 1) & \text{if} \quad x > 0.6. \end{cases} \tag{4.52}$$

An inflow condition with (ρ, u, v, p) values coinciding with the $x \le 0.6$ initial values, on the other hand, are imposed on (ρ, u, v, p) at the left boundary at all times. Further, outflow conditions consisting of the time-independent pressure value $p = 1$ is imposed at the right boundary. Slip-wall (zero-normal velocity) boundary conditions are imposed at the bottom and top walls. No slip (zero velocity) and adiabatic (zero normal component of the gradient of the temperature) boundary conditions, finally, are imposed at the boundaries of the cylinders at all times—since, as detailed as part of the Test-Problem-2 description below, a viscous problem is solved near the cylinder boundaries whenever non-zero velocities occur at the cylinder boundaries.

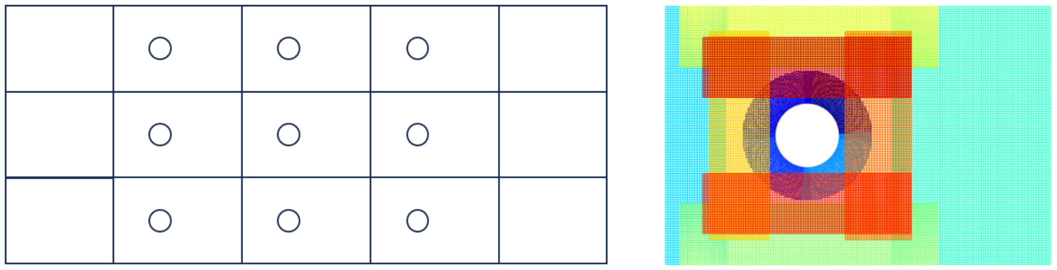


Figure 4.8: Illustration of the types of geometries used for the test problems in Section 4.5.1.1. Left panel: domain decomposition utilized in Test Problem 1 in the case $n_{\text{row}} = 3$ and $n_{\text{col}} = 3$. Right panel: patch decomposition of each cylinder-containing subdomain in the left panel. The right panel also depicts the complete computational domain used for Test problem 2.

Test Problem 2: Hypersonic flow past a cylindrical obstacle. Test Problem 2 concerns a hypersonic (Mach 10) flow past a cylinder of diameter 0.25 and centered at $(x_c, y_c) = [1.1, 0]$, where the computational domain corresponds exactly to a single instance of a subdomain containing a cylinder as previously described as part of Test Problem 1 and shown in the right panel of Figure 4.8. An initial configuration with 54 subpatches is considered, which is then refined in accordance with the method described in Section 4.1.2.4 in order to perform weak and strong scaling tests. The initial condition considered is a Mach 10 flow given by

$$(\rho, u, v, p) = (1.4, 10, 0, 1). \tag{4.53}$$

As in Test Problem 1, an inflow condition with (ρ, u, v, p) values coinciding quantitatively with the initial values is imposed at the left boundary at all times, and no boundary conditions are imposed on the right outflow boundary, as befits a supersonic outflow. Reflecting boundary conditions, corresponding to zero-normal velocity, are imposed at the bottom ($y = -1$) and top ($y = 1$) walls. No slip (zero velocity) and adiabatic (zero normal component of the gradient of the temperature) boundary conditions are imposed at the boundaries of the cylinder at all times—as befits the viscous-like problem that is solved in a neighborhood of cylinder on account of the artificial viscosity which, as illustrated in Figure 4.11, is assigned by the artificial-viscosity algorithm in that region.

4.5.1.2 Scaling II: Parallel performance metrics

In order to quantify the parallel scaling efficiencies of the FC-SDNN algorithm in multi-patch settings, for given numbers N of discretization points and N_C of compute cores (or N_C^0 of reference compute cores), we denote by $T_S(N_C, N)$ and $T_S(N_C^0, N)$ as the number of seconds required by the FC-SDNN to advance $4N$ unknowns for one time-step using N_C cores or N_C^0 cores, respectively. The strong (resp. weak) scaling efficiencies $E^s_{N_C^0,N_C}$ (resp. $E^w_{N_C^0,N_C}$) are then defined by

$$E^s_{N_C^0,N_C} = \frac{T_S(N_C^0, N)N_C^0}{T_S(N_C, N)N_C}, \qquad E^w_{N_C^0,N_C} = \frac{T_S(N_C^0, N)}{T_S(N_C, NN_C/N_C^0)}. \tag{4.54}$$

An alternative measure of the parallel computing time required by the algorithm to advance the $4N$ unknowns associated with an N-point discretization grid in a given computational experiment is provided by the number S_{N_C} of CPU-seconds required to advance the simulation for one time-step in a number N_C of CPU cores per 10^6 unknowns, that is,

$$S_{N_C,N} = \frac{N_C \times (\text{total computation time}) \times 10^6}{4 \times N \times (\text{Time steps})}. \tag{4.55}$$

In particular, the strong and weak efficiencies can be re-expressed as a function of S_{N_C}:

$$E^s_{N_C^0,N_C} = \frac{S_{N_C^0,N}N_C^0}{S_{N_C,N}N_C}, \qquad E^w_{N_C^0,N_C} = \frac{S_{N_C^0,N}}{S_{N_C,N_c/N_C^0N}}. \tag{4.56}$$

Section 4.5.1.3 uses the Test Problem 1 described in Section 4.5.1.1 to illustrate the weak scaling in a context where the number of discretization points is increased so as to tackle increasingly larger physical problems—in that case, through the progressive addition of rows and/or columns of cylindrical obstacles in a rectangular array of cylinders. Section 4.5.1.4 then utilizes Test Problem 2 in Section 4.5.1.1 to illustrate the weak scaling properties of the algorithm in the context in which increasing discretizations result from mesh refinement. Finally, Section 4.5.1.5 uses Test Problem 2 once again to study strong scalability—by solving a fixed physical problem defined on a fixed mesh on the basis of increasingly larger numbers of computing cores.

The very high, essentially perfect, weak scalability demonstrated by the tests presented in Sections 4.5.1.3 and 4.5.1.4 indicate that both, the communication and the average computing-cost per subpatch required by the algorithm remain essentially fixed as the numbers of cores and the sizes of the problems increase proportionally. The high but less-than-perfect strong scalability illustrated in Section 4.5.1.5, on the other hand, reflects the fact that the computing time per subpatch varies between subpatches—owing to differences in the communication costs required to various subpatches. The strong scaling could be further improved by incorporating an algorithm that distributes a combination of subpatch types to each computer node, including adequate proportions high and low communication-cost subpatches. Such additional algorithmic developments are beyond the scope of this thesis, however, and are left for future work.

4.5.1.3 Scaling III: Weak scaling under problem enlargement

This section illustrates the weak scalability of the FC-SDNN algorithm by enlarging the problem size via a progressive addition of columns and/or rows of cylindrical obstacles in rectangular array of cylinders of the type described in Test Problem 1, Section 4.5.1.1, and advancing the solution up to time T = 0.1 (with space and time units such that the speed of sound in the unperturbed flow state is $a = 1$, and the distance between the centers of two vertically consecutive cylinders is 1.25). As showcased in Table 4.1, the FC-SDNN algorithm enjoys essentially perfect weak scaling as the number of discretization points and cores are proportionally increased ($N/N_C = N_Q$ with $N_Q = 101^2$, see (4.17)), starting with an $N_C = 9 \times 54 = 486$ computing-core initial configuration (at 54 cores per node) for a Test Problem-1 array containing a single three-cylinder column in addition to the front and wake regions.

N	n_{row}	n_{col}	N_C	Nodes	$T(s)$	$S_{N_C}(s)$	E^w_{486,N_C} (%)
4,957,686	3	1	486	9	74.4	0.97	
6,610,248	3	2	648	12	74.4	0.97	100
6,610,248	4	1	648	12	75.0	0.98	99
8,262,810	3	3	810	15	74.2	0.97	100
8,262,810	5	1	810	15	74.4	0.97	100
8,813,664	4	2	864	16	74.8	0.98	99
9,915,372	3	4	972	18	74.0	0.97	101
11,017,080	4	3	1080	20	74.0	0.97	101
11,017,080	5	2	1080	20	74.8	0.98	99
11,567,934	3	5	1134	21	74.0	0.97	101
13,220,496	4	4	1296	24	73.9	0.97	101
13,771,350	5	3	1350	25	74.5	0.97	100
15,423,912	4	5	1512	28	74.8	0.98	99
16,525,620	5	4	1620	30	75.0	0.98	99

Table 4.1: FC-SDNN weak parallel scaling for various rectangular arrays of cylinders (Test Problem 1 in Section 4.5.1.1), containing n_{row} rows and n_{col} columns of cylindrical obstacles, and using N_C cores, with N_C ranging from 486 to 1620 (9 to 30 computer nodes).

4.5.1.4 Scaling IV: Weak scaling under mesh refinement

This section once again illustrates the weak scalability of the proposed FC-SDNN parallel implementation, but this time in the context of mesh refinement—wherein, as described in Section 4.1.2.4, decreases in discretization sizes are obtained via corresponding increases in the number of subpatches used while simultaneously and proportionally increasing the number N_C of cores—in a setting where the number of cores equals the number of subpatches. We do this here by solving Test Problem 2 up to time $T = 0.1$ (with space and time units such that the speed of sound in the unperturbed flow state is $a = 1$, and the distance between the top and bottom boundaries of the computational domain equals 2). As the mesh is refined the time-step decreases in an approximately proportional fashion (following equation (4.38)), thereby increasing in a (roughly) linear manner the number of simulation time-steps—as illustrated in Table 4.2.

Table 4.2 displays the runtimes T, the numbers S_{N_C} and the efficiencies E^w_{54,N_C} as the total number of cores and discretization points are increased proportionally, as noted in the table, using 54 cores per computer node and increasing the number of discretization points N proportionally to the number of computer nodes used.

N	N_C	Nodes	$T(s)$	Time steps	$S_{N_C}(s)$	$E^w_{54,N_C}(\%)$
550,854	54	1	23.0	563	1.00	
2,203,416	216	4	44.41	1106	0.98	102
4,957,686	486	9	64.96	1650	0.96	104
8,813,664	864	16	86.20	2194	0.96	104
13,771,350	1350	25	107.46	2738	0.96	104

Table 4.2: FC-SDNN weak parallel scaling in a mesh refinement context (Test Problem 2 in Section 4.5.1.1) using N_C cores, with N_C ranging from 54 to 1350 (1 to 25 computer nodes).

As illustrated in Table 4.2, the weak scaling efficiency of the procedure in the present mesh-refinement context is excellent, steadily rising above 100% with respect to the coarsest mesh before stabilizing for finer meshes. We attribute this better than perfect scalability to slightly diminishing workloads associated to the ranks carrying the heaviest workloads as the mesh is refined. In detail, for initial, coarse, subpatch partitions, a number of boundary subpatches play the roles of both donors and recipients of interpolated solution data. As the subpatch refinements take place, donors tend not to be receivers and viceversa, and thus the maximum interpolation-data workload per MPI rank that occurs in such patches is decreased. This process eventually stabilizes, as illustrated in Table 4.2, once most subpatches engaged in communication of interpolated data only play the roles of either donor or recipient.

4.5.1.5 Scaling V: Strong scaling

This section illustrates the strong parallel scalability of the FC-SDNN algorithm in a context in which the number of allocated processing cores is increased for a given physical problem and for fixed patch/subpatch decomposition of the problem—namely Test Problem 2 discretized on the basis of 864 subpatches, which corresponds to a number $N = 8,813,664$ of discretization points. (This example thus utilizes the same physical problem considered in Section 4.5.1.4, using the next-to-finest discretization considered in that section, but assigning multiple subpatches to each computer core, as needed for each given number of computers cores used to deal with the fixed number of subpatches considered.) Strong scaling is studied in the Wavefield cluster by progressively increasing the number of nodes used to solve the problem up to time $T = 0.1$, and using a constant number of 54 cores per node. The associated times $T(s)$ in seconds and parallel efficiencies E^s_{54,N_C} with respect to a run of 54 cores are displayed in Table 4.3. The observed decrease in the strong scaling efficiency as the number of cores is increased may be attributed to differences in communication costs and associated computational workloads assigned to various subpatches.

N	N_C	Nodes	$T(s)$	$S_{N_C}(s)$	$E^s_{54,N_C}(\%)$	$E^s_{N_C/2,N_C}(\%)$
	54	1	1126.0	0.79		
	108	2	598.8	0.84	94	94
8,813,664	216	4	330.1	0.92	85	91
	432	8	167.0	0.93	84	99
	864	16	86.4	0.96	81	97

Table 4.3: FC-SDNN strong parallel scaling (Test Problem 2 in Section 4.5.1.1) using N_C cores, with N_C ranging from 54 to 864 (1 to 16 computer nodes).

4.5.2 Applications

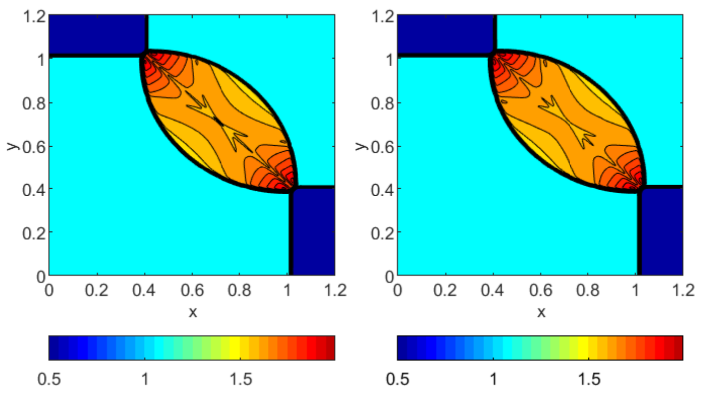
This section presents numerical results of the application of the FC-SDNN to a number of physical problems on various degrees of geometric complexity, and involving strong shocks, as well as supersonic and hypersonic flows. The new results clearly accord with previous numerical simulations, theory and experimental data, and they include simulations significantly beyond regimes previously tackled computationally, such as is the case for e.g. Mach 10 shocks and flows—for which, nevertheless, good agreement is observed with certain flow details predicted by previous theory as well as extrapolations from experimental data such as e.g. the distance between an obstacle and a reflected bow shock. To facilitate visualization of the position of shock waves, following [71] and [72] displays of the Schlieren diagram of the quantity

$$\sigma = \exp\left(-\beta \frac{|\nabla\rho(\mathbf{x})| - \min_{\mathbf{x}\in\Omega}|\nabla\rho(\mathbf{x})|}{\max_{\mathbf{x}\in\Omega}|\nabla\rho(\mathbf{x})| - \min_{\mathbf{x}\in\Omega}|\nabla\rho(\mathbf{x})|}\right) \tag{4.57}$$

with $\beta = 10$ are presented for each test case. The time-stepping constant CFL = 0.25 in (4.38) was used for all the simulations involving C_1-type corners and patches, as in Section 4.5.2.7 and 4.5.2.10 (which, for the problems considered, involve rather stretched meshes and thus small distances between spatial discretization points), while the value CFL = 0.5 was used in all other cases.

4.5.2.1 Multi-patch method validation

In order to validate the accuracy of the multi-patch method presented in this chapter we consider once again the test case presented in Section 3.2.3.2. The left panel in



(a) Single subpatch, $N = 600$ (b) 7×7 subpatches

Figure 4.9: Second-order ($d = 2$) Multi-patch FC-SDNN numerical solution to the Euler 2D Riemann problem considered in Section 3.2.3.2, at $t = 0.25$. The solution is represented using thirty equispaced contours between $\rho = 0.5$ and $\rho = 1.99$.

Figure 4.9 reproduces the single-patch 600×600-point solution previously presented in the middle panel of Figure 3.18. The right panel in Figure 4.9, in turn, displays the solution produced, for the same problem, on the basis of a 7×7 subpatch decomposition (the subpatch concept is introduced in Section 4.1.2) with a similar meshsize: $h \approx 0.002$. The two images in Figure 4.9 present closely matching flow features, with errors consistent with the underlying error level—which may be gleaned from the red error curves in Figure 3.12. In particular, the smoothness of the contour levels is maintained, and the shocks are equally well resolved in both cases.

4.5.2.2 Energy conservation.

Several mechanisms used by the FC SDNN algorithm, such as the introduction of artificial viscosity (Section 4.3) to avoid spurious oscillations, and the use of a spectral filtering at every time-step (Section 4.2.3) to control the error growth in unresolved high frequency modes, are dissipative in nature, but, as demonstrated below, do not lead to excessive energy loss. To quantify the effect we studied a 2D multi-patch version of the Sod Problem (Section 3.2.3.1), in the domain $[0, 1] \times [0, 0.25]$, with initial conditions given by

$$(\rho, u, v, p) = \begin{cases} (1, 0, 0, 1) & \text{if} \quad x < 0.5 \\ (0.125, 0, 0, 0.1) & \text{if} \quad x \geq 0.5. \end{cases} \tag{4.58}$$

As no energy enters or exits the domain via inflow or outflow, the exact value $\bar{E}_{\text{exact}} = \int_0^{0.25} \int_0^1 E_{\text{exact}}(x, y, t)dxdy$ of the energy contained in the complete domain is constant—as it can be easily verified e.g. by differentiation with respect to time of the integral of E followed by use of the energy equation (2.6) together with the velocity vanishing boundary values; it is easy to check that for the exact solution we have $\bar{E}_{\text{exact}} = 1.375$. Thus any observed departures $\Delta\bar{E}$ of the computed energy $\bar{E}$ from this exact value provide an indication of the dissipative character of the algorithm.

To test such dissipative effects we discretized the domain by using a unique patch decomposed into an arrangement of 4 horizontal subpatches, and reflecting boundary conditions $u = 0$ were imposed at $x = 0$ and $x = 1$. Vanishing normal derivatives for all variables were enforced on the top and bottom boundaries at all times so as to simulate a vertically infinite domain. The solution was computed up to time $T = 100$, during which the shock, contact discontinuity and rarefaction wave were reflected 60 times per boundary, for a total of 120 wall reflections. As showcased in Figure 4.10, the numerical values $\bar{E}$ of the complete energy content remain essentially constant, $\bar{E} \approx 1.37 \pm 0.01$, where the errors are consistent with the underlying error level indicated by the red error curves in Figure 3.12. Noticeable features of the energy and energy error curves include an early-time increase in the energy values (with a total increase that diminishes in size as the discretization are refined, as it was found on the basis of an additional set of tests not included in this thesis for brevity) as well as a slow energy decrease attributable to the dissipative processes mentioned above in this section.

4.5.2.3 Flow initialization

In the remainder of this chapter we consider two types of application problems, arising from two distinct types of initial conditions. The first type is the interaction of a supersonic/hypersonic "Mach M" flow with an obstacle (for M values such as 1.4, 3.5, 10 etc.), which is considered in Sections 4.5.2.4 through 4.5.2.7. The initial condition, given by

$$(\rho, u, v, p) = (1.4, M, 0, 1), \tag{4.59}$$

indeed corresponds to a uniform Mach M flow with speed of sound $a = 1$. The second type of problems, discussed in Sections 4.5.2.8 through 4.5.2.10, corresponds to the interaction of a shock initially located at $x = x_s$ and traveling toward the region $x \geq x_s$ at a speed M that is supersonic/hypersonic with respect to the speed of sound

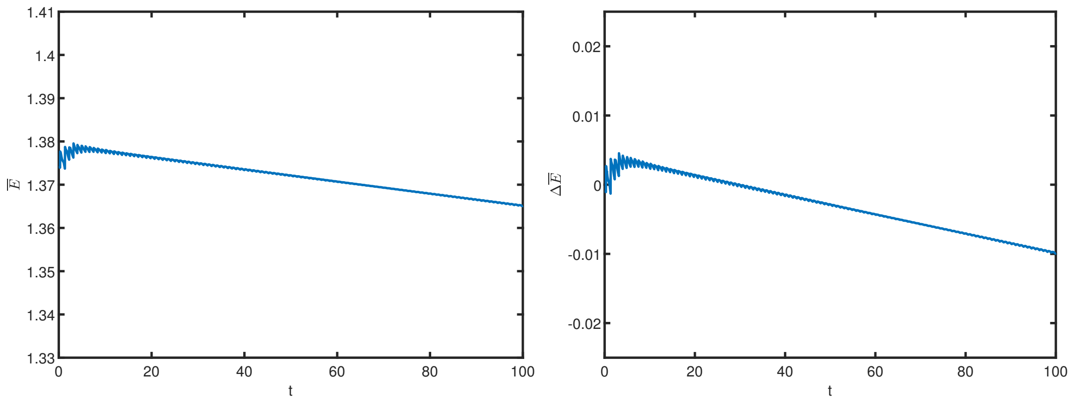


Figure 4.10: Numerical values of the total energy $\bar{E}$ (left panel) and its defect $\Delta\bar{E}$ from the exact value $\bar{E}_{\text{exact}} = 1.375$ (right panel) as functions of time in the time interval $0 \leq t \leq 100$—within which the shock, contact discontinuity and rarefaction wave were reflected 60 times per boundary, for a total of 120 wall reflections. The figures show that the numerical energy $\bar{E}$ remains essentially constant ($\bar{E} \approx 1.37 \pm 0.01$) for the duration of the time-interval considered, with defect values consistent with the algorithm's error levels illustrated in Figure 3.12.

$a = 1$ ahead of the shock. For such a Mach M shock the initial condition is given by

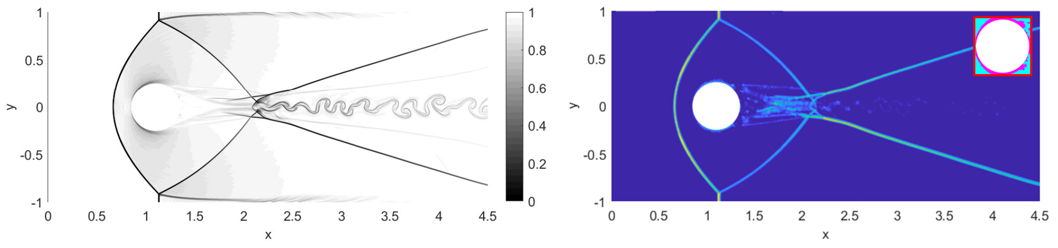
$$(\rho, u, v, p) = \begin{cases} \left(\frac{(\gamma+1)M^2}{(\gamma-1)M^2+2}, \frac{\zeta-1}{\gamma M}, 0, \zeta\right) & \text{if} \quad x < x_s \\ (1.4, 0, 0, 1) & \text{if} \quad x \geq x_s, \end{cases} \tag{4.60}$$

where $\zeta = \frac{2\gamma M^2-\gamma+1}{\gamma+1}$ denotes the strength of the shock; see e.g. [73].

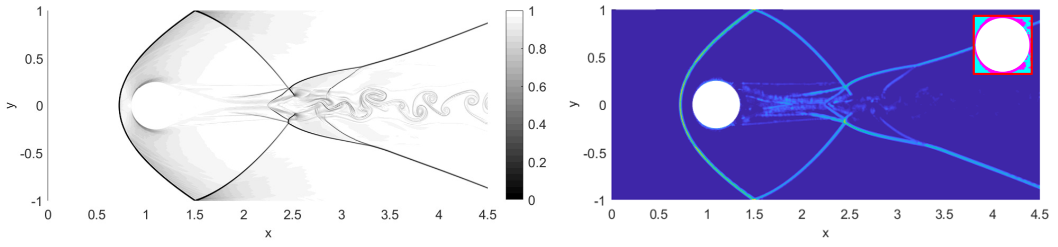
4.5.2.4 Supersonic flow past cylinder in a wind tunnel

We consider a set of problems involving supersonic and hypersonic flows of varying Mach numbers, in a wind tunnel of dimensions $[0, 4.5] \times [-1, 1]$ over a stationary cylinder of radius $R_c = 0.25$ and centered at the point $(x_c, y_c) = (1.25, 0)$. The geometric setting is close to the one considered in Test Problem 2 in Section 4.5.1.1, but here additional empty front and wake regions are included. As in that section, reflecting boundary conditions are imposed at the bottom and top walls, together no-slip and adiabatic boundary conditions on the cylinder. Initial conditions (4.59) with Mach numbers $M = 3$, $M = 6$ and $M = 10$ are considered.

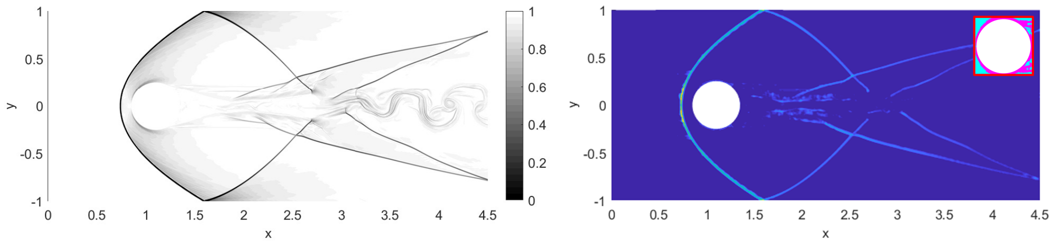
Schlieren diagrams of the solutions at $T = 2$ are displayed in Figure 4.11. Particular challenges posed by the configuration include the strong shock (with very high pressure next the front of the cylinder) as well as the possible formation of vacuum states in the wake region, particularly for high Mach number flows. Mach 2, Mach 3, and Mach 3.5 simulations for similar geometric setting were considered



(a) Mach 3 flow.



(b) Mach 6 flow.



(c) Mach 10 flow.

Figure 4.11: Supersonic flows past a cylindrical obstacle at various Mach numbers, at time $T = 2$, obtained on a $N = 6.2M$-point mesh. Left panels: density Schlieren images. Right panels: Artificial viscosity profiles. The insets on the right panels display the artificial viscosity profiles (in an appropriate color scale) in the immediate vicinity of the cylindrical obstacle—which illustrates the boundary-layer resolution provided by the method, and which motivates the use of no-slip/adiabatic boundary conditions, as discussed in the Test Problem 2 description presented in Section 4.5.1.1.

in [72] and [74] and [73], respectively. It is worthwhile to emphasize here that, as noted in Chapter 4 and as illustrated in the present section and, indeed, in all of the numerical-example Sections 4.5.2.4 through 4.5.2.10, and unlike the methods [72, 74, 73], the present approach does not incorporate limiters or other artificial devices to prevent the formation of negative-density regions, which, in the present case, could manifest themselves in areas behind the cylinder. Generally the solver has lead to non-negative (possibly quite small albeit numerically positive) values of the density ρ; cf. e.g. the density plot in Figure 3.13. Additional considerations in this regard are presented as part of the first paragraph in Chapter 4.

Remark 13. As mentioned in Chapter 1 and further discussed in the first paragraph of Chapter 4, and as illustrated in Figure 4.11, for flow-past-obstacle problems the vanishing-velocity boundary condition at obstacle boundaries gives rise to sharp boundary layers—which causes the SDNN algorithm to assign artificial viscosity at boundaries at all times. As suggested in the first paragraph of Chapter 4 the presence of such boundary viscosity may explain the preservation of positivity of the density and pressure without use of positivity limiters that is observed to result from the FC-SDNN algorithm.

4.5.2.5 $M = 25$ hypersonic flow past a cylinder

We next consider a set of problems involving flows at supersonic (Mach 3.5), and re-entry (Mach 25) speeds past a cylindrical obstacle of radius $R_c = 0.25$ and centered at the point $(x_c, y_c) = (1, 0)$. For the supersonic (resp. the re-entry) velocity problem, the computational domain considered consists of the portion of the rectangle $[0, 2] \times [-1.75, 1.75]$ (resp. $[0, 2.5] \times [-1.5, 1.5]$) located outside the cylinder. Initial conditions (4.59) with Mach numbers $M = 3.5$ and $M = 25$ are considered. An inflow condition with (ρ, u, v, p) values coinciding quantitatively with the initial values is imposed at the left boundary at all times, and no boundary conditions are imposed on the right boundary, as befits a supersonic outflow. No-slip and adiabatic boundary conditions are imposed on the cylinder, while vanishing normal derivatives for all variables are imposed at the top on bottom domain boundaries. Schlieren diagrams of the solutions at $T = 2.0$ are displayed in Figure 4.12.

A notable feature in these simulations is the formation of a reflected bow shock ahead of the cylinder. An existing parametric fit to experiment [65] of the distance $d_0 = d_0(M)$ between the bow shock and the cylinder's leading edge as a function of the Mach number M is used in what follows (as was used previously [75, 73]) to validate, at least partially, the quality of the proposed gas-dynamics solver. According to [65] the distance d_0 is given by

$$\frac{d_0}{2R_c} \approx 0.193 \exp\left(\frac{4.67}{M^2}\right). \tag{4.61}$$

Numerical values of the ratio $\frac{d_0}{2R_c}$ produced by our solver for various Mach numbers are provided in Table 4.4; these data show that, for each Mach number considered, the numerical value of d_0 converges towards the empirical value given in (4.61) as the mesh is refined.

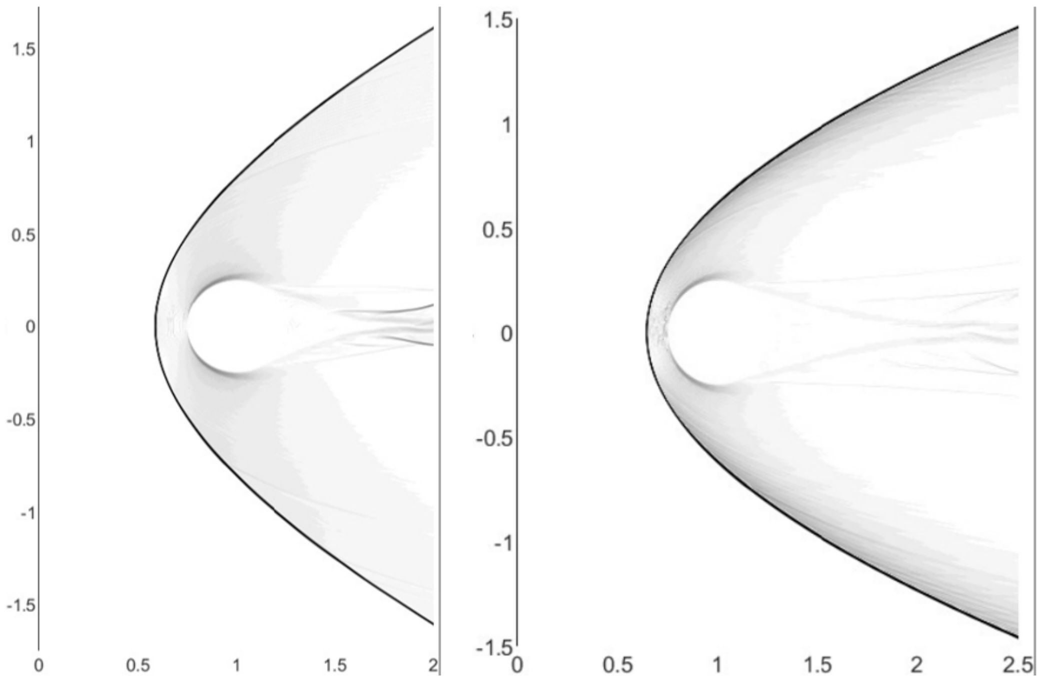


Figure 4.12: Supersonic flow past a cylindrical obstacle in a wide channel, at time $T = 2.0$, for two different Mach numbers, Mach 3.5 (left) and Mach 25 (right), obtained on meshes containing $N \approx 8.8M$ and $N \approx 9.8M$ discretization points, respectively.

$M = 3.5$	$\frac{d_0}{2R_c}$	$M = 25$	$\frac{d_0}{2R_c}$
Exper. parametric fit	0.283	Exper. parametric fit	0.194
$N = 2,203,416$	0.32	$N = 2,448,240$	0.21
$N = 8,813,664$	0.30	$N = 9,792,960$	0.20
$N = 19,830,744$	0.30	$N = 22,034,160$	0.20

Table 4.4: Bow-shock/cylinder distance in the supersonic flow past a cylinder in a wide channel, including data corresponding to the experimental parametric fit [65, Fig. 2] (eq. (4.61) above) and computational results for various values of N. The differences between the simulations and the experimental parametric fit appear consistent with experimental error levels reported in [65].

4.5.2.6 Supersonic flow past a triangular wedge

We next consider supersonic and hypersonic flows past the triangular wedge depicted in Figure 4.13, whose tip is located at the point $(x_t, y_t) = (0.013, 0.015)$, and whose interior angle equals $\alpha_t = 40°$. The computational domain consists of the portion of the rectangle $[0, 0.024] \times [0, 0.03]$ located outside the wedge. Initial conditions given by equation (4.59) with supersonic Mach number $M = 3.5$ and hypersonic Mach number $M = 10$ are considered in what follows. An inflow condition with

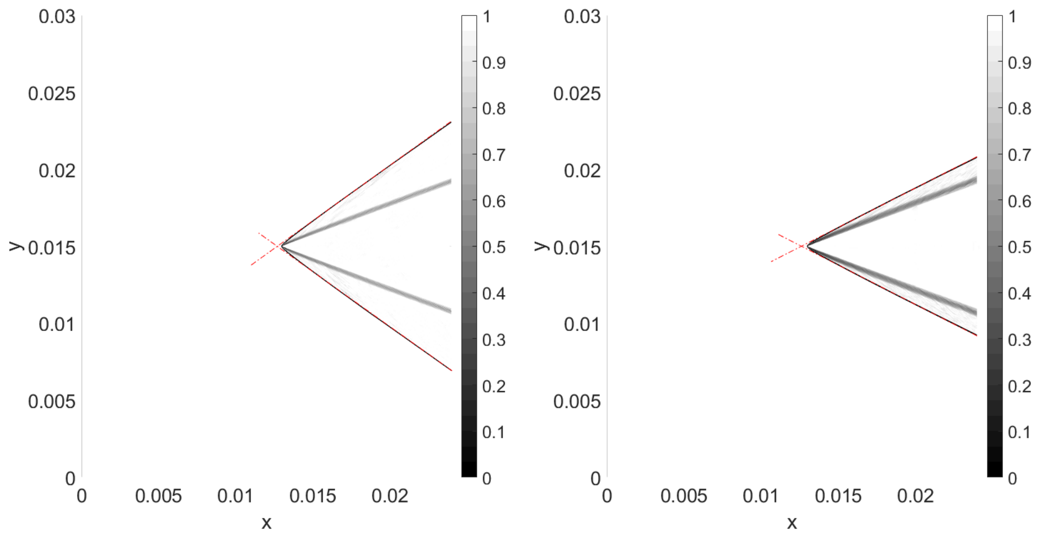


Figure 4.13: Supersonic flow past a triangular wedge, at time $T = 0.02$, for two different Mach numbers, Mach 3.5 (left) and Mach 10 (right), obtained on an $N = 31.5M$-point mesh.

(ρ, u, v, p) values coinciding quantitatively with the initial values is imposed at the left boundary at all times, and no boundary conditions are imposed on the right boundary, as befits a supersonic outflow. Vanishing normal derivatives for all variables are imposed at the top on bottom domain boundaries. No slip and adiabatic boundary conditions are imposed on the wedges at all times. The results at time $T = 0.02$ for Mach $M = 3.5$ and $M = 10$ flows and for a mesh containing approximately 31 million points are presented in Figure 4.13.

Notably, a straight oblique shock forms starting at the tip of the triangular wedge, with a deflection angle α_d with respect to the horizontal. A closed form relation between α_d, the wedge angle α_t and the Mach number M of the incoming flow can be obtained (for an infinite wedge) in closed form on the basis of the Rankine-Hugoniot jump conditions [73]; the result is

$$\tan(\frac{\alpha_t}{2}) = 2\cot(\alpha_d)\frac{M^2\sin^2(\alpha_d) - 1}{M^2(\gamma + \cos(2\alpha_d)) + 2}. \tag{4.62}$$

The numerical results presented in Figure 4.11 show a deflected shock which starts slightly ahead of the tip of the wedge, and it slightly curves around it to eventually form a straight shock. The slightly-curved near-tip shock region can be attributed to the formation of a fine viscous boundary layer at the wedge boundaries (in line with our use of artificial viscosity (4.51) and associated no-slip and adiabatic boundary conditions at the obstacle, as prescribed above in this section). The deflection angle

	$M = 3.5$	$M = 10$
Inviscid theory	34.6°	25.8°
$N = 520,251$	36.8°	21.9°
$N = 2,356,431$	36.0°	23.0°
$N = 10,007,181$	35.7°	23.8°
$N = 31,500,688$	35.2°	24.3°

Table 4.5: Deflected shock angle for the problem of a supersonic flow past a wedge, including inviscid-theory values and computational results for various values of N.

corresponding to the computational simulation was calculated by fitting a straight line to the shock starting from $x = 0.024$ (the rightmost abscissa shown in the figure) which, in view of the aforementioned slight near-tip curvature, meets the x axis at an abscissa x_d slightly ahead of the tip abscissa x_t. Numerical values of the deflection angles for various meshes used are compared in Table 4.5 to the corresponding theoretical values. As demonstrated in that table, the deflection angle approaches closely the expected theoretical value as the discretization is refined. The $M = 3.5$ shock problem was previously considered in [73]; that contribution calculates a deflection angle of 34.5° on the basis of a mesh of approximately one million discretization points, which more closely approximates the theoretical deflection angle than any of the predictions presented in Table 4.5. We suggest that further work is warranted in this connection however, as the coarser discretization used in [73, Fig. 4] results in a wider shock profile and an associated imperfect match to a straight line showing noticeable departures from the shock for extended shock sections near the wedge vertex.

4.5.2.7 Supersonic flow past triangular prism

In this section we consider flow problems in a wind tunnel similar to the one considered in [73], of dimensions $[0, 0.06] \times [0, 0.03]$, over a stationary triangular prism of length $\ell = 0.011$ from left to right, whose front vertex has coordinates $(x_t, y_t) = (0.013, 0.015)$ and front angle $\alpha_t = 40°$. Two different initial value problems are considered for this geometry, namely those resulting from supersonic- and hypersonic-flow initial conditions given by equation (4.59) with Mach numbers $M = 3.5$ and $M = 10$, respectively. Reflecting boundary conditions are imposed at the bottom and top walls, together with no slip and adiabatic boundary conditions at the prism's boundaries. An inflow condition with (ρ, u, v, p) values coinciding quantitatively with the initial values is imposed at the left boundary at all times, and no boundary conditions are imposed on the right boundary, as befits a supersonic outflow. Schlieren images as well as a contour plots of the density ρ for both test cases at time $T = 0.02$ are presented in Figure 4.14. As in Section 4.5.2.6, oblique shocks are reflected off the front facing sides of the triangular prism. As shown in the figures, these shocks are reflected by the top and bottom wind-tunnel boundaries, and the reflected shocks, in turn, interact with the wake at the back of the prism. Clearly, the FC-SDNN method allows for a fine resolution of the shock structures, and as demonstrated by the contour plots in Figures 4.14, smooth density profiles are preserved away from shocks. This problem presents certain challenges concerning the possible formation of vacuum states in the wake of the prism, in particular for high Mach number flows, but, unlike previous spectral and finite-element approaches, no density limiters were used to preserve density positivity. To the best of the author's knowledge, the fastest wind-tunnel flows past a prism are the Mach 3.5 problems considered in [73, 76].

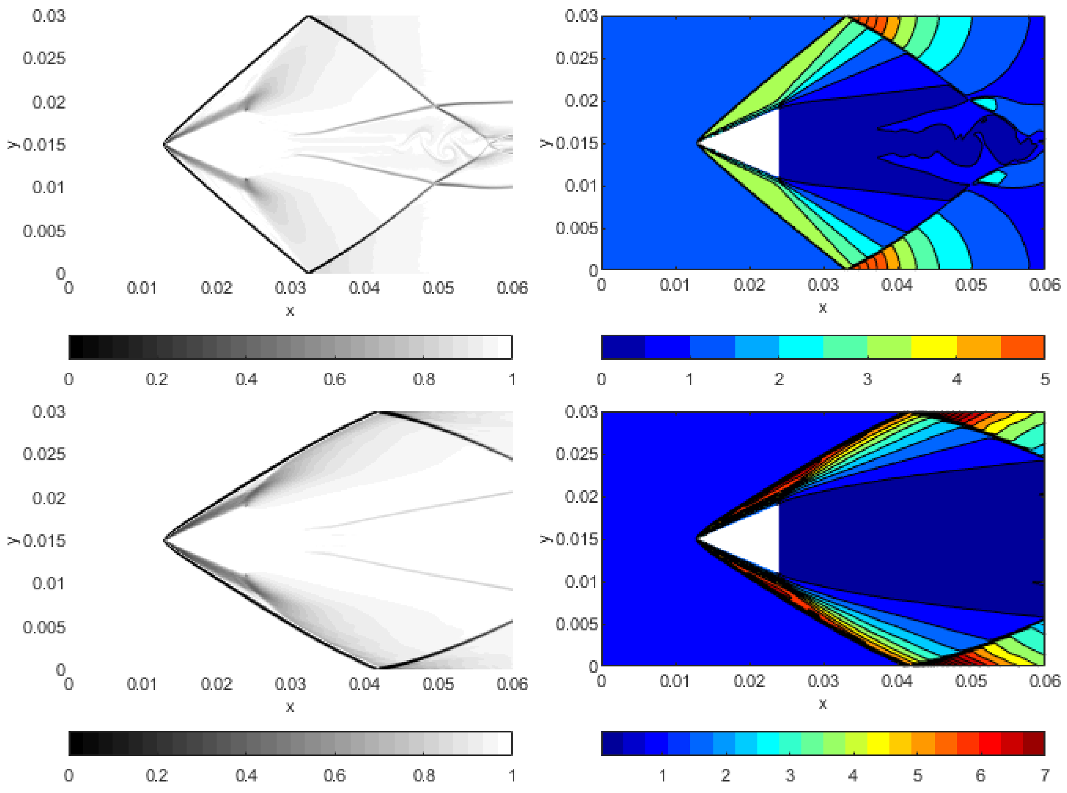


Figure 4.14: Supersonic flows past a triangular prism, at time $T = 0.02$, for two different Mach numbers, Mach 3.5 (top row) and Mach 10 (bottom row), obtained on a $N = 10.5M$-point mesh in both cases. Left column: density Schlieren images. Right column: density contour plots.

4.5.2.8 Shock-cylinder interaction

This section concerns shock-cylinder interaction problems for the geometry utilized in Section 4.5.2.4 with two different shock speeds of Mach numbers $M = 3$ and $M = 10$. The corresponding initial conditions are given by (4.60), with initial shock position at $x_s = 0.5$. An inflow condition with (ρ, u, v, p) values coinciding with the $x \leq x_s$ initial values is imposed at the left boundary at all times, and an outflow condition consisting of the time-independent pressure value $p = 1$ is imposed at the right boundary, also at all times. Slip-wall boundary conditions are imposed at the bottom and top walls, while no-slip and adiabatic boundary conditions, finally, are imposed at the boundaries of the cylinders at all times.

Schlieren and density contour plots for the Mach 3 solution (resp. the Mach 10 solution) at time $T = 0.45$ (resp. $T = 0.15$) are displayed in Figure 4.15. The Mach 3 Schlieren diagrams display flow features in close agreement with the experimental and numerical results presented in [77, Fig. 4] and [73, Fig. 17], respectively, including the reflected shock and symmetric Mach shocks and contact discontinuities starting at the back of the cylinder and intersecting the incident shock at two triple points. Similar features are observed for the Mach 10 problem that is also illustrated in Figure 4.15. (A single shock-cylinder interaction problem, with Mach 2.8 shock speed, is presented in [73].) The contour plots in Figure 4.15 display smooth density contours, without spurious oscillations.

4.5.2.9 Shock-wave mitigation by a matrix of solid cylindrical obstacles

This section presents results for a Mach 10 shock propagating through an array of cylindrical obstacles—a significantly stronger shock than previously considered for this type of problem: Mach 1.8 and Mach 3 problems for similar geometries are considered in [78, 79]. The study of such interactions of shocks mitigation is one of paramount importance for the design of shielding systems [78]. The geometry considered, depicted in Figure 4.8 results as a slight modification of the one used in Test Problem 1 (Section 4.5.1.1) with $n_{\text{row}} = 3$ and $n_{\text{col}} = 3$; the modifications introduced here concern the front and the wake region, both of which have been extended: the new front (resp. wake) region considered here contains a 3×3 array of rectangular cylinder-free patches (resp. a 4×3 array of cylinder-free patches), instead of the corresponding 1×3 front and wake arrays considered in Section 4.5.1.1. The initial conditions considered here are given by equation (4.60) with shock traveling at Mach numbers $M = 3$ and $M = 10$, and starting at the

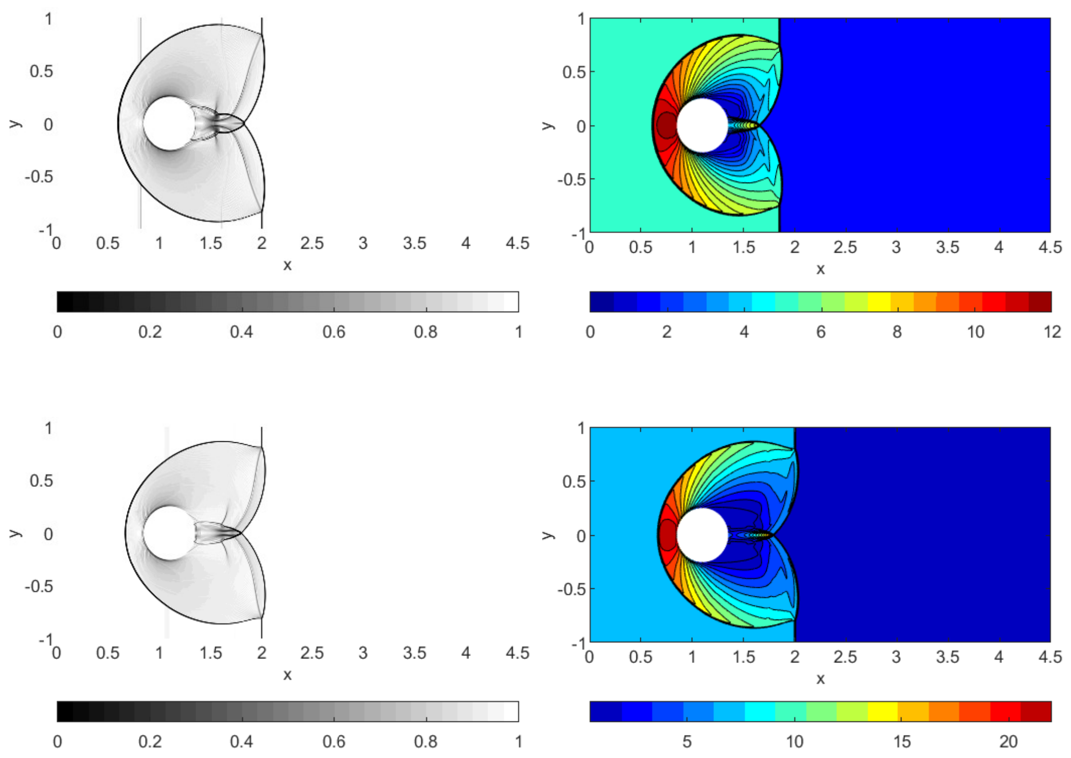


Figure 4.15: Shock-cylinder interaction for two different shock speeds and at two different times, namely, Mach 3.5 at time $T = 0.45$ (top) and Mach 10 at time $T = 0.15$ (bottom), obtained on an $N \approx 6.2M$-point mesh. Left panels: density Schlieren images. Right panels: density contour plots.

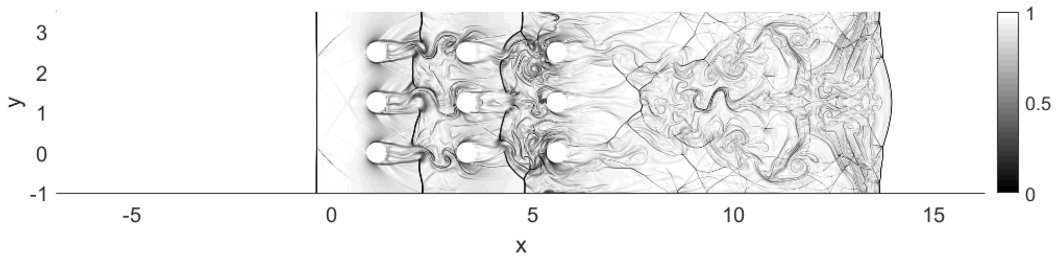


Figure 4.16: Mach 10 shock-wave mitigation by a 3x3 matrix of cylindrical obstacles at time $T = 1.4$, obtained on $N \approx 16.5M$-point mesh. Density Schlieren image.

position $x_s = 0.5$; the boundary conditions used are identical to those described in Test Problem 1. Figure 4.16 presents a Schlieren diagram of the solution at time $T = 1.4$.

4.5.2.10 Shock-prism interaction

We finally consider the interaction of a shock with a stationary triangular prism in a $[0, 0.08] \times [0, 0.06]$ wind tunnel. In this case an equilateral triangular prism of length $\ell = 0.011$ from left to right is considered, with a front vertex at coordinates $(x_t, y_y) = (0.013, 0.03)$ and front angle $\alpha_t = \frac{\pi}{3}$. The initial conditions for the two tests considered here are given by equation (4.60) with $M = 1.5$ and $M = 10$, respectively, and with $x_s = 0.007$. A test for a similar geometry, the *Schardin* problem, was introduced in [80] for a Mach 1.34 shock, and studied numerically in [81] and in [73]. An inflow condition with (ρ, u, v, p) values coinciding with the $x \leq x_s$ initial values are imposed at the left boundary at all times. Further, outflow conditions consisting of the time-independent pressure value $p = 1$ are imposed at the right boundary. Slip-wall boundary conditions are imposed at the bottom and top walls, while no-slip and adiabatic boundary conditions are imposed at the boundaries of the prism at all times. Solutions at times $T = 0.025$ for the Mach 1.5 shock and $T = 0.00375$ for the Mach 10 shock are displayed in Figure 4.17. The numerical solution correctly displays transmitted and reflected shocks, as well as slip lines and vortices at the back of the prism. As in previous examples, the density contour plots in Figure 4.17 displays smooth contour levels away from flow discontinuities (such as, e.g., the contact discontinuities that start at the the upper and lower triple points and that, in the $M = 1.5$ images, continue up to the upper and lower vortices (as evidenced also in the $M = 1.5$ Schlieren diagram).

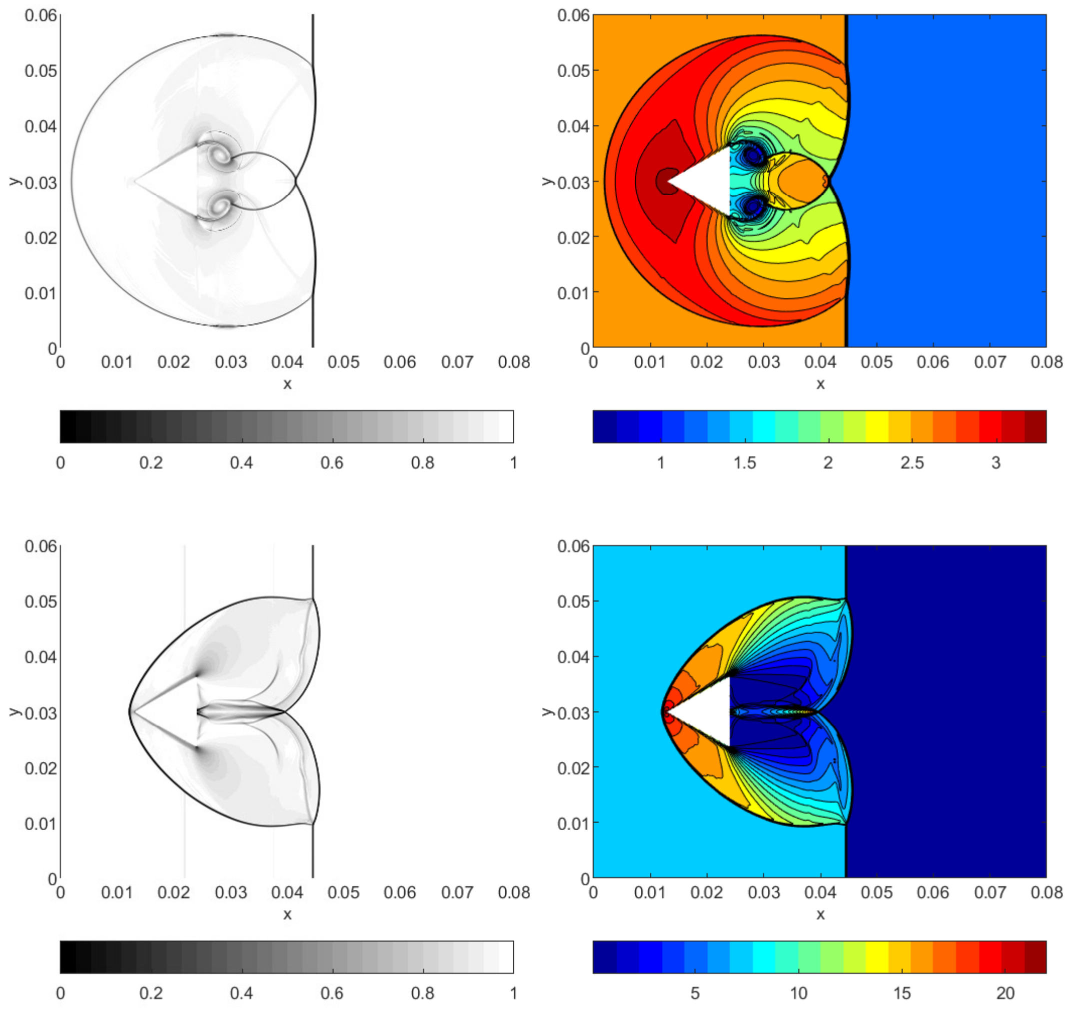


Figure 4.17: Shock-prism interaction for two different shock speeds and at two different times, namely, Mach 1.5 at time $T = 0.02$ (top) and Mach 10 at time $T = 0.00375$ (bottom), obtained on an $N \approx 10.5M$-point mesh. Left panels: density Schlieren images. Right panels: density contour plots.

Chapter 5

CONCLUSIONS

This thesis introduced a novel computational algorithm for general gas-dynamics problems, namely, the FC-SDNN *spectral* shock-dynamics solver which, without the taxing CFL constraints inherent in other spectral schemes, is applicable to *general domains*, up to and including *supersonic and hypersonic* flows and *non-smooth* domain boundaries. Relying on the Fourier Continuation method (FC) for accurate Fourier expansion of non-periodic functions, together with a neural network-based shock-detection methodology, as well as an innovative smooth artificial-viscosity concept and a multi-patch discretization strategy, the proposed methodology delivers accurate numerical simulations, with sharp definition of shocks and other flow discontinuities, as well as accurate and smoothly contoured flow profiles away from flow discontinuities, and including shock-wave simulations at significantly higher Mach numbers than previously reported for certain important experimental configurations and physical obstacles. In particular, test cases presented in this thesis illustrate the method's accurate shock detecting capability and concentration of artificial viscosity near shocks, while essentially avoiding viscosity assignments away from shocks, up to and including at contact discontinuities—with significant impact on the resulting resolution of the latter often-oversmeared flow features.

1) The new multi-patch approach enables the application of the FC-SDNN method for evaluation of flows and shock-dynamics in general 2D domains, including boundaries containing corners, and in the context of both supersonic and hypersonic flows. Further, the new method 2) introduces a smooth and localized artificial-viscosity quantity of the type utilized as part of a previously existing, single-domain FC-SDNN strategy, that is additionally compatible with the multi-patch general-domain discretization strategy used presently, and 3) incorporates an MPI-based parallel implementation based on the multi-domain discretization used, which enjoys high weak and strong parallel scaling in large present-day computer clusters. And, finally, 4) the proposed multi-patch implementation exhibits numerical results in close accordance with physical theory and prior experimental and computational results up to an including both the supersonic and hypersonic regimes.

Chapter 6

FUTURE WORK

Significant extensions of the FC-SDNN methodology are envisioned in a number of promising directions. On one hand, the implementation of the methodology in the 3D context, which could be highly impactful, can be pursued by appropriately generalizing the multi-domain decomposition strategy described in Section 4.1, and by adapting the associated overall smooth artificial viscosity assignment procedure (Section 4.3), shock-detection and time-marching schemes. Similarly, the application of the FC-SDNN method to other systems of conservation laws, particularly the Magnetohydrodynamics (MHD) equations [34], which requires an adaptation of the choice of regularity proxy (cf. Section 3.3), might provide a significant impact in the area of energy research. An additional important extension concerns the design of an efficient hybrid implicit/explicit scheme to facilitate the discretization of domains including irregular boundaries (which require fine discretizations) and/or sharp angles (which give rise to stretched meshes and thus small distances between spatial discretization points, cf. Section 4.5.2) without taxing CFL time-step constraints, while maintaining a less costly explicit capability to manage simulations away from challenging boundary regions. Further, the development of an adaptive mesh refinement procedure for shock dynamics on the basis of the multi-domain strategy employed in Chapter 4, which has previously been considered in the context of the overset grid method [82], would allow for a finer resolution of shocks and other flow discontinuity features, as well as an increased accuracy of the solution in regions behind shocks, without requiring use of fine meshes throughout the computational domain. Finally, the implementation of the proposed methods on computer architectures that incorporate graphical processing units (GPUs) is highly promising. In particular, the GPU distribution of the numerous but relatively small FFT calls associated with the FC discretization should result in massively accelerated computations for both highly challenging 2D and (specially) 3D contexts, with highest promise associated with localization of large subdomain regions within single GPUs in a multi-GPU system, with limited interdomain solution communication.

BIBLIOGRAPHY

[1] Nathan Albin and Oscar P Bruno. “A spectral FC solver for the compressible Navier–Stokes equations in general domains I: Explicit time-stepping”. In: *Journal of Computational Physics* 230 (2011), pp. 6248–6270.

[2] Faisal Amlani and Oscar P Bruno. “An FC-based spectral solver for elastodynamic problems in general three-dimensional domains”. In: *Journal of Computational Physics* 307 (2016), pp. 333–354.

[3] Oscar P Bruno and Mark Lyon. “High-order unconditionally stable FC-AD solvers for general smooth domains I. Basic elements”. In: *Journal of Computational Physics* 229.6 (2010), pp. 2009–2033.

[4] Mark Lyon and Oscar P Bruno. “High-order unconditionally stable FC-AD solvers for general smooth domains II. Elliptic, parabolic and hyperbolic PDEs; theoretical considerations”. In: *Journal of Computational Physics* 229.9 (2010), pp. 3358–3381.

[5] Lukas Schwander, Deep Ray, and Jan S Hesthaven. “Controlling oscillations in spectral methods by local artificial viscosity governed by neural networks”. In: *Journal of Computational Physics* 431 (2021), p. 110144.

[6] Oscar P Bruno, Jan S Hesthaven, and Daniel V Leibovici. “FC-based shock-dynamics solver with neural-network localized artificial-viscosity assignment”. In: *Journal of Computational Physics: X* 15 (2022), p. 100110.

[7] G Chesshire and William D Henshaw. “Composite overlapping meshes for the solution of partial differential equations”. In: *Journal of Computational Physics* 90.1 (1990), pp. 1–64.

[8] Randall J LeVeque. *Numerical methods for conservation laws*. Vol. 132. Springer, 1992.

[9] Randall J LeVeque et al. *Finite volume methods for hyperbolic problems*. Vol. 31. 2002.

[10] Ami Harten et al. “Uniformly high order accurate essentially non-oscillatory schemes, III”. In: *Upwind and high-resolution schemes*. Springer, 1987, pp. 218–290.

[11] Xu-Dong Liu, Stanley Osher, and Tony Chan. “Weighted essentially non-oscillatory schemes”. In: *Journal of Computational Physics* 115.1 (1994), pp. 200–212.

[12] Guang-Shan Jiang and Chi-Wang Shu. “Efficient implementation of weighted ENO schemes”. In: *Journal of Computational Physics* 126.1 (1996), pp. 202–228.

[13] Khosro Shahbazi et al. “Multi-domain Fourier-continuation/WENO hybrid solver for conservation laws”. In: *Journal of Computational Physics* 230.24 (2011), pp. 8779–8796.

[14] RD Richtmyer. “Proposed numerical method for calculation of shocks”. In: *Los Alamos Report* 671 (1948).

[15] John VonNeumann and Robert D Richtmyer. “A method for the numerical calculation of hydrodynamic shocks”. In: *Journal of Applied Physics* 21.3 (1950), pp. 232–237.

[16] Richard A Gentry, Robert E Martin, and Bart J Daly. “An Eulerian differencing method for unsteady compressible flow problems”. In: *Journal of Computational Physics* 1.1 (1966), pp. 87–118.

[17] Arnold Lapidus. “A detached shock calculation by second-order finite differences”. In: *Journal of Computational Physics* 2.2 (1967), pp. 154–177.

[18] Peter Lax. *Systems of conservation laws*. Tech. rep. LOS ALAMOS NATIONAL LAB NM, 1959.

[19] Yvon Maday and Eitan Tadmor. “Analysis of the spectral vanishing viscosity method for periodic conservation laws”. In: *SIAM Journal on Numerical Analysis* 26.4 (1989), pp. 854–870.

[20] Eitan Tadmor. “Convergence of spectral methods for nonlinear conservation laws”. In: *SIAM Journal on Numerical Analysis* 26.1 (1989), pp. 30–44.

[21] Eitan Tadmor. “Shock capturing by the spectral viscosity method”. In: *Computer Methods in Applied Mechanics and Engineering* 80.1-3 (1990), pp. 197–208.

[22] Eitan Tadmor. “Super viscosity and spectral approximations of nonlinear conservation laws”. In: *Numerical Methods for Fluid Dynamics* 4 (1993), pp. 69–81.

[23] Yvon Maday, Sidi M Ould Kaber, and Eitan Tadmor. “Legendre pseudospectral viscosity method for nonlinear conservation laws”. In: *SIAM Journal on Numerical Analysis* 30.2 (1993), pp. 321–342.

[24] Per-Olof Persson and Jaime Peraire. “Sub-cell shock capturing for discontinuous Galerkin methods”. In: *44th AIAA Aerospace Sciences Meeting and Exhibit*. 2006, p. 112.

[25] Jean-Luc Guermond, Richard Pasquetti, and Bojan Popov. “Entropy viscosity method for nonlinear conservation laws”. In: *Journal of Computational Physics* 230.11 (2011), pp. 4248–4267.

[26] Adeline Kornelus and Daniel Appelö. “Flux-conservative Hermite methods for simulation of nonlinear conservation laws”. In: *Journal of Scientific Computing* 76.1 (2018), pp. 24–47.

[27] Jon Reisner, Jonathan Serencsa, and Steve Shkoller. “A space–time smooth artificial viscosity method for nonlinear conservation laws”. In: *Journal of Computational Physics* 235 (2013), pp. 912–933.

[28] Raaghav Ramani, Jon Reisner, and Steve Shkoller. “A space-time smooth artificial viscosity method with wavelet noise indicator and shock collision scheme, Part 1: The 1-D case”. In: *Journal of Computational Physics* 387 (2019), pp. 81–116.

[29] Raaghav Ramani, Jon Reisner, and Steve Shkoller. “A space-time smooth artificial viscosity method with wavelet noise indicator and shock collision scheme, Part 2: the 2-D case”. In: *Journal of Computational Physics* 387 (2019), pp. 45–80.

[30] Deep Ray and Jan S Hesthaven. “An artificial neural network as a troubled-cell indicator”. In: *Journal of Computational Physics* 367 (2018), pp. 166–191.

[31] Niccolo Discacciati, Jan S Hesthaven, and Deep Ray. “Controlling oscillations in high-order Discontinuous Galerkin schemes using artificial viscosity tuned by neural networks”. In: *Journal of Computational Physics* 409 (2020), p. 109304.

[32] Ben Stevens and Tim Colonius. “Enhancement of shock-capturing methods via machine learning”. In: *Theoretical and Computational Fluid Dynamics* 34 (2020), pp. 483–496.

[33] Oscar P Bruno, Max Cubillos, and Edwin Jimenez. “Higher-order implicit-explicit multi-domain compressible Navier-Stokes solvers”. In: *Journal of Computational Physics* 391 (2019), pp. 322–346.

[34] Mauro Fontana et al. “Vector potential-based MHD solver for non-periodic flows using Fourier continuation expansions”. In: *Computer Physics Communications* 275 (2022), p. 108304.

[35] Haydn Maust et al. “Fourier Continuation for Exact Derivative Computation in Physics-Informed Neural Operators”. In: *arXiv preprint arXiv:2211.15960* (2022).

[36] Colin White et al. “Physics-informed neural operators with exact differentiation on arbitrary geometries”. In: *The Symbiosis of Deep Learning and Differential Equations III*. 2023.

[37] Oscar P Bruno and Jagabandhu Paul. “Two-dimensional Fourier Continuation and applications”. In: *arXiv:2010.03901* (2020).

[38] Sigal Gottlieb. “On high order strong stability preserving Runge-Kutta and multi step time discretizations”. In: *Journal of Scientific Computing* 25.1 (2005), pp. 105–128.

[39] Mark H Carpenter et al. “The theoretical accuracy of Runge–Kutta time discretizations for the initial boundary value problem: a study of the boundary error”. In: *SIAM Journal on Scientific Computing* 16.6 (1995), pp. 1241–1252.

[40] D Pathria. “The Correct Formulation of Intermediate Boundary Conditions for Runge–Kutta Time Integration of Initial Boundary Value Problems”. In: *SIAM Journal on Scientific Computing* 18.5 (1997), pp. 1255–1266.

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

[42] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[43] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *JMLR Workshop and Conference Proceedings*. 2010, pp. 249–256.

[44] Vladimir Vasil’evich Rusanov. “The calculation of the interaction of non-stationary shock waves and obstacles”. In: *USSR Computational Mathematics and Mathematical Physics* 1.2 (1962), pp. 304–320.

[45] Ralph Beebe Blackman and John Wilder Tukey. “The measurement of power spectra from the point of view of communications engineering—Part I”. In: *Bell System Technical Journal* 37.1 (1958), pp. 185–282.

[46] Charles Hirsch. “Numerical computation of internal and external flows. Vol. 2-Computational methods for inviscid and viscous flows”. In: *John Wiley & Sons, 1990, 708* (1990).

[47] Gerald Beresford Whitham. *Linear and nonlinear waves*. John Wiley & Sons, 2011.

[48] Chi-Wang Shu. “High order ENO and WENO schemes for computational fluid dynamics”. In: *High-order methods for computational physics*. Springer, 1999, pp. 439–582.

[49] Peter D Lax and Xu-Dong Liu. “Solution of two-dimensional Riemann problems of gas dynamics by positive schemes”. In: *SIAM Journal on Scientific Computing* 19.2 (1998), pp. 319–340.

[50] Paul Woodward and Phillip Colella. “The numerical simulation of two-dimensional fluid flow with strong shocks”. In: *Journal of Computational Physics* 54.1 (1984), pp. 115–173.

[51] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[52] Gary A Sod. “A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws”. In: *Journal of Computational physics* 27.1 (1978), pp. 1–31.

[53] Peter D Lax. “Weak solutions of nonlinear hyperbolic equations and their numerical computation”. In: *Communications on pure and applied mathematics* 7.1 (1954), pp. 159–193.

[54] Chi-Wang Shu and Stanley Osher. “Efficient implementation of essentially non-oscillatory shock-capturing schemes, II”. In: *Upwind and High-Resolution Schemes*. Springer, 1989, pp. 328–374.

[55] Alireza Mazaheri, Chi-Wang Shu, and Vincent Perrier. “Bounded and compact weighted essentially nonoscillatory limiters for discontinuous Galerkin schemes: Triangular elements”. In: *Journal of Computational Physics* 395 (2019), pp. 461–488.

[56] Jan S Hesthaven and Fabian Mönkeberg. “Two-dimensional RBF-ENO method on unstructured grids”. In: *Journal of scientific computing* 82.3 (2020), p. 76.

[57] US Vevek, B Zang, and Tze How New. “On alternative setups of the double Mach reflection problem”. In: *Journal of Scientific Computing* 78.2 (2019), pp. 1291–1303.

[58] Eric Johnsen and SK Lele. “Numerical errors generated in simulations of slowly moving shocks”. In: *Center for Turbulence Research, Annual Research Briefs* (2008), pp. 1–12.

[59] Friedemann Kemm. “On the proper setup of the double Mach reflection as a test case for the resolution of gas dynamics codes”. In: *Computers & Fluids* 132 (2016), pp. 72–75.

[60] Ravi Samtaney and DI Pullin. “On initial-value and self-similar solutions of the compressible Euler equations”. In: *Physics of Fluids* 8.10 (1996), pp. 2650–2655.

[61] David L Book et al. “Power-Series Solutions of the Gasdynamic Equations for Mach Reflection of a Planar Shock by a Wedge.” In: *Shock Tubes and Waves (eds. RD Archer and BE Milton), Proceedings of the l^ th International Symposium on Shock Tubes and Waves, held in Sydney, Australia*. 1983, pp. 166–174.

[62] Hans Hornung. “Regular and Mach reflection of shock waves”. In: *Annual review of fluid mechanics* 18.1 (1986), pp. 33–58.

[63] AT Dinh et al. “Simulation of viscous stabilization of Kelvin-Helmholtz instability”. In: *WIT Transactions on Engineering Sciences* 29 (1970).

[64] Ludwig Prandtl. “Zur berechnung der grenzschichten”. In: *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* 18.1 (1938), pp. 77–82.

[65] Frederick S Billig. “Shock-wave shapes around spherical-and cylindrical-nosed bodies.” In: *Journal of Spacecraft and Rockets* 4.6 (1967), pp. 822–823.

[66] Paul Novello et al. “Accelerating hypersonic reentry simulations using deep learning-based hybridization (with guarantees)”. In: *Journal of Computational Physics* 498 (2024), p. 112700.

[67] Robert L Calloway. *Pressures, forces, moments and shock shapes for a geometrically matched sphere-cone and hyperboloid at Mach 20.3 in helium*. Tech. rep. 1983.

[68] Ajay Kumar and R GRAVES JR. “Numerical solution of the viscous hypersonic flow past blunted conesat angle of attack”. In: *15th Aerospace Sciences Meeting*. 1977, p. 172.

[69] Jan S Hesthaven and Fabian Mönkeberg. “Hybrid high-resolution RBF-ENO method”. In: *Journal of Computational Physics: X* 12 (2021), p. 100089.

[70] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[71] Jeffrey W Banks et al. “A study of detonation propagation and diffraction with compliant confinement”. In: *Combustion Theory and Modelling* 12.4 (2008), pp. 769–808.

[72] Murtazo Nazarov and Aurélien Larcher. “Numerical investigation of a viscous regularization of the Euler equations by entropy viscosity”. In: *Computer Methods in Applied Mechanics and Engineering* 317 (2017), pp. 128–152.

[73] Arnab Chaudhuri, Abdellah Hadjadj, and Ashwin Chinnayya. “On the use of immersed boundary methods for shock/obstacle interactions”. In: *Journal of Computational Physics* 230.5 (2011), pp. 1731–1748.

[74] Jean-Luc Guermond et al. “Second-order invariant domain preserving approximation of the Euler equations using convex limiting”. In: *SIAM Journal on Scientific Computing* 40.5 (2018), A3211–A3239.

[75] Olivier Boiron, Guillaume Chiavassa, and Rosa Donat. “A high-resolution penalization method for large Mach number flows in the presence of obstacles”. In: *Computers & fluids* 38.3 (2009), pp. 703–714.

[76] Saikat Das et al. “A PRELIMINARY NUMERICAL STUDY OF HIGH SPEED COMPRESSIBLE FLOW AROUND A SOLID TRIANGULAR PRISM”. In: ().

[77] AE Bryson and RWF Gross. “Diffraction of strong shocks by cones, cylinders, and spheres”. In: *Journal of Fluid Mechanics* 10.1 (1961), pp. 1–16.

[78] A Chaudhuri et al. “Numerical study of shock-wave mitigation through matrices of solid obstacles”. In: *Shock Waves* 23 (2013), pp. 91–101.

[79] Y Mehta et al. “Numerical investigation of shock interaction with one-dimensional transverse array of particles in air”. In: *Journal of Applied Physics* 119.10 (2016).

[80] H Schardin. “High frequency cinematography in the shock tube”. In: *The Journal of Photographic Science* 5.2 (1957), pp. 17–19.

[81] Se-Myong Chang and Keun-Shik Chang. “On the shock–vortex interaction in Schardin’s problem”. In: *Shock Waves* 10.5 (2000), pp. 333–343.

[82] William D Henshaw and Donald W Schwendeman. “Moving overlapping grids with adaptive mesh refinement for high-speed reactive and non-reactive flow”. In: *Journal of Computational Physics* 216.2 (2006), pp. 744–779.