Available online at www.sciencedirect.com

ScienceDirect

Comput. Methods Appl. Mech. Engrg. 389 (2022) 114250

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

# On the implementation of a robust and efficient finite element-based parallel solver for the compressible Navier–Stokes equations

Jean-Luc Guermond[a,*], Martin Kronbichler[b,c], Matthias Maier[a], Bojan Popov[a], Ignacio Tomas[d]

[a] *Department of Mathematics, Texas A&M University 3368 TAMU, College Station, TX 77843, USA*
[b] *Institute for Computational Mechanics, Department for Mechanical Engineering, Technical University of Munich, Germany*
[c] *Division of Scientific Computing, Department of Information Technology, Uppsala University, Sweden*
[d] *Sandia National Laboratories[1], P.O. Box 5800, MS 1320, Albuquerque, NM 87185-1320, USA*

Received 3 June 2021; received in revised form 6 October 2021; accepted 11 October 2021
Available online 20 November 2021

## Abstract

This paper describes in detail the implementation of a finite element technique for solving the compressible Navier–Stokes equations that is provably robust and demonstrates excellent performance on modern computer hardware. The method is second-order accurate in time and space. Robustness here means that the method is proved to be invariant domain preserving under the hyperbolic CFL time step restriction, and the method delivers results that are reproducible. The proposed technique is shown to be accurate on challenging 2D and 3D realistic benchmarks.

*MSC:* 35L65; 65M60; 65M12; 65N30

*Keywords:* Hyperbolic conservation equations; compressible Navier–Stokes equations; Invariant domains; High-order method; Convex limiting; Finite element method

## 1. Introduction

The objective of the paper is to describe in detail a robust and efficient massively parallel finite element technique for solving the compressible Navier–Stokes equations. This paper is the second part of a research project described in Guermond et al. [1]. The principles of the method have been introduced in [1], but in order to guarantee reproducibility (following the guidelines described in LeVeque et al. [2]), we here describe the implementation details regarding the algorithm *per se* and identify the ingredients that enable efficient execution on large-scale parallel machines. We also explain the implementation of non-reflecting boundary conditions and show that these

* Corresponding author.
*E-mail addresses:* guermond@tamu.edu (J.-L. Guermond), martin.kronbichler@it.uu.se (M. Kronbichler), maier@tamu.edu (M. Maier), popov@tamu.edu (B. Popov), nachotet@gmail.com (I. Tomas).

[1] Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This document describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

https://doi.org/10.1016/j.cma.2021.114250
0045-7825/© 2021 Elsevier B.V. All rights reserved.

conditions are robust, invariant-domain preserving, and accurate. Based on these ingredients, the accuracy of the method is demonstrated on well-documented (i.e., reproducible) non-trivial benchmarks. The robustness of the method and its capability to scale well on large parallel architectures are also demonstrated.

As there is currently a regain of interest for supersonic and hypersonic flight of aircrafts and other devices, there is also a renewed interest for provably robust numerical methods that can solve the compressible Navier–Stokes equations. Here we say that a numerical method is provably robust if it can be unambiguously proved to be invariant domain preserving, i.e., among other things, it ensures positivity of the density, positivity of the internal energy, and preserves a meaningful entropy-dissipation property. There are many papers in the literature addressing this question, but invariant domain properties are available only for very few methods. One notable result in this direction can be found in Grapsas et al. [3] where a first-order staggered approximation using velocity-based upwinding is developed (see Eq. (3.1) therein), and positivity of the density and the internal energy is established (Lem. 4.4 therein). Unconditional stability is obtained by using an implicit time-stepping coupling the mass conservation equation and the internal energy equation. This method is robust, including in the low Mach regime. A similar technique solving the compressible barotropic Navier–Stokes equation is proposed in [4, §3.6]. In the discontinuous Galerkin literature, robustness is established in Zhang [5] for the approximation of the compressible Navier–Stokes equations. The time stepping is explicit though, and this entails a parabolic restriction on the time step that unfortunately makes the method ill-suited for realistic large-scale applications (i.e., $\tau \lesssim \mathcal{O}(h^2)/\mu$, where $\mu$ is some reference viscosity scale, $\tau$ is the time step size and $h$ is the mesh size).

The approximation technique described in the present paper draws robustness from an operator-splitting strategy that uncouples the hyperbolic and the parabolic phenomena. We do not claim originality for this "divide and conquer" strategy since the operator-splitting idea has been successfully used in the CFD literature numerous times in the past. Among the references that inspired the present work in one way or another, we refer the reader to Beam and Warming [6], Bristeau et al. [7], Demkowicz et al. [8]. The key novelties of the paper are as follows: (i) The paper describes an exhaustive and unambiguous (thereby reproducible) robust algorithm for solving the compressible Navier–Stokes using finite elements. Algorithm 1 gives a flow chart that minimizes the complexity of the hyperbolic step; (ii) The implementation of various boundary condition is fully and unambiguously described. In particular, non-reflecting boundary conditions are discussed. Unambiguous, fully discrete, finite-element based algorithms are proposed. These boundary conditions are explicit and are proved to be invariant-domain preserving and to maintain conservation; (iii) The parabolic substep of the algorithm is also fully described and important details regarding its matrix-free implementation are given; (iv) The algorithm is verified against analytical solutions and validated against two challenging benchmarks (one is two-dimensional, the other is three-dimensional). In particular, we provide a reference solution for the benchmark proposed in Daru and Tenaud [9], Daru and Tenaud [10] with an accuracy that has never been matched before (see Table 2 and Fig. 6).

The paper is organized as follows. The problem along with the finite element setting and the principles of the time stepping that are used for the approximation is described in Section 2. As the time stepping is based on Strang's splitting using a hyperbolic substep and a parabolic substep, we describe in Section 3 the full approximation of the hyperbolic step. All the details that are necessary to guarantee reproducibility are given. Key results regarding admissibility and conservation after limiting are collected in Lemma 3.2. Important details regarding the treatment of boundary conditions for the hyperbolic step are reported in Section 4. Key original results regarding admissibility and conservation after boundary postprocessing are collected in Lemmas 4.2 and 4.5, and Corollary 4.4. The full approximation of the parabolic substep is described in Section 5. Here again, all the details that are necessary to guarantee reproducibility are given. The key results of this section regarding admissibility and conservation are stated in Lemma 5.1. The method has been implemented using the finite element library `deal.II` [11,12] and mapped continuous $\mathbb{Q}_1$ finite elements. Our implementation is freely available online[2] [13] under a permissible open source license.[3] The method and its implementation are verified and validated in Section 6. In addition to standard code verifications using analytical solutions (see Section 6.1) and tests on non-reflecting boundary conditions (see Section 6.2), we revisit two benchmarks problems. First, we solve in Section 6.3 a two-dimensional shocktube problem proposed by Daru and Tenaud [9], Daru and Tenaud [10] and demonstrate grid convergence. Following the initiative of [14] and to facilitate rigorous quantitative comparisons with other research codes, we provide very accurate computations of the skin friction coefficient for this problem; these results are freely available at [15]. To

[2] https://github.com/conservation-laws/ryujin
[3] https://spdx.org/licenses/MIT.html

the best of our knowledge, the level of accuracy we achieved for this benchmark has never been matched before. We also demonstrate in Section 6.4 that the proposed method can reliably predict pressure coefficients on the well-studied supercritical airfoil Onera OAT15a in the supercritical regime at Mach 0.73 in three dimensions and at Reynolds number $3\times10^6$ (see [16], Deck and Renard [17], Nguyen et al. [18]). Finally, a series of synthetic benchmarks are presented in Section 6.5 to assess the performance of the compute kernels by investigating the strong and weak scalability of our implementation. Technical details are reported in Appendix A.

## 2. Problem description, finite element setting, time splitting

We briefly introduce relevant notation, recall the compressible Navier–Stokes equations, discuss the finite element setting for the proposed algorithm, and introduce the operator-splitting technique that is used to make the method invariant domain preserving under a standard hyperbolic CFL time step restriction. We follow in large parts the notation introduced in [1].

### 2.1. The model

Given a bounded, polyhedral domain $D$ in $\mathbb{R}^d$, an initial time $t_0$, and initial data $\boldsymbol{u}_0 := (\rho_0, \boldsymbol{m}_0, E_0)$, we look for $\boldsymbol{u} : D\times[t_0, +\infty) \to \mathbb{R}_+\times\mathbb{R}^d\times\mathbb{R}_+$ solving the compressible Navier–Stokes system in some weak sense:

$$\partial_t\rho + \nabla\cdot(\boldsymbol{v}\rho) = 0, \tag{2.1a}$$

$$\partial_t\boldsymbol{m} + \nabla\cdot\big(\boldsymbol{v}\otimes\boldsymbol{m} + p(\boldsymbol{u})\mathbb{I} - \mathbb{s}(\boldsymbol{v})\big) = \boldsymbol{f}, \tag{2.1b}$$

$$\partial_t E + \nabla\cdot\big(\boldsymbol{v}(E + p(\boldsymbol{u})) - \mathbb{s}(\boldsymbol{v})\boldsymbol{v} + \boldsymbol{k}(\boldsymbol{u})\big) = \boldsymbol{f}\cdot\boldsymbol{v}. \tag{2.1c}$$

Here $\rho$ is the density, $\boldsymbol{m}$ is the momentum, $E$ is the total energy, $p(\boldsymbol{u})$ is the pressure, $\mathbb{I}\in\mathbb{R}^{d\times d}$ is the identity matrix, $\boldsymbol{f}$ is an external force, $\mathbb{s}(\boldsymbol{v})$ is the viscous stress tensor and $\boldsymbol{k}(\boldsymbol{u})$ is the heat-flux. The quantity $\boldsymbol{v} := \rho^{-1}\boldsymbol{m}$ is called velocity and $e(\boldsymbol{u}) := \rho^{-1}E - \frac{1}{2}\|\rho^{-1}\boldsymbol{m}\|^2_{\ell^2}$ is called specific internal energy. Given a state $\boldsymbol{u}\in\mathbb{R}^{d+2}$, $\rho(\boldsymbol{u})$ denotes the first coordinate (i.e., density), $\boldsymbol{m}(\boldsymbol{u})$ denotes the $\mathbb{R}^d$-valued vector whose components are the 2-nd up to the $(d+1)$-th coordinates of $\boldsymbol{u}$ (i.e., the momentum), and $E(\boldsymbol{u})$ is the last coordinate of $\boldsymbol{u}$ (i.e., the total energy). Boundary conditions for (2.1) and the implementation of these condition are discussed in detail in Section 4.

To simplify the notation later on, we introduce the flux $\mathbb{f}(\boldsymbol{u}) := (\boldsymbol{m}, \boldsymbol{v}\otimes\boldsymbol{m} + p(\boldsymbol{u})\mathbb{I}_d, \boldsymbol{v}(E+p))^{\mathsf{T}} \in \mathbb{R}^{(d+2)\times d}$, where $\mathbb{I}_d$ is the $d\times d$ identity matrix. Although it is often convenient to assume that the pressure $p(\boldsymbol{u})$ is derived from a complete equation of state, most of what is said here holds true by only assuming that the pressure is given by an oracle (see, e.g., Clayton et al. [19]). In the applications reported at the end of the paper, though, we use the ideal gas law $p(\boldsymbol{u}) = (\gamma-1)\rho e(\boldsymbol{u})$.

The fluid is assumed to be Newtonian and the heat-flux is assumed to follow Fourier's law:

$$\mathbb{s}(\boldsymbol{v}) := 2\mu\,\mathbb{e}(\boldsymbol{v}) + (\lambda - \tfrac{2}{3}\mu)\nabla\cdot\boldsymbol{v}\mathbb{I}, \qquad \mathbb{e}(\boldsymbol{v}) := \nabla^s\boldsymbol{v} := \tfrac{1}{2}\big(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^{\mathsf{T}}\big), \tag{2.2}$$

$$\boldsymbol{k}(\boldsymbol{u}) := -c_v^{-1}\kappa\nabla e, \tag{2.3}$$

where $\mu > 0$ and $\lambda \geq 0$ are the shear and the bulk viscosities, respectively, $\kappa$ is the thermal conductivity, and $c_v$ is the heat capacity at constant volume. For the sake of simplicity, we assume that $\mu, \lambda, \kappa$, and $c_v$ are constant.

Important properties we want to maintain at the discrete level are the positivity of the density and the positivity of the specific internal energy. We formalize these constraints by introducing the set of admissible states:

$$\mathcal{A} := \{\boldsymbol{u} := (\rho, \boldsymbol{m}, E)^{\mathsf{T}} \in \mathbb{R}^{d+2} | \rho > 0,\ e(\boldsymbol{u}) > 0\}. \tag{2.4}$$

We also want that in the inviscid regime limit (i.e., $\lambda\to0$, $\mu\to0$, $\kappa\to0$), the algorithm satisfies the local minimum principle of the specific entropy at each time step.

### 2.2. Finite element setting

Although, as claimed in [20], the proposed approximation technique is discretization agnostic and can be implemented with finite volumes and with discontinuous or continuous finite elements, we restrict ourselves here to continuous finite elements since it greatly simplifies the approximation of the second-order differential operators.

Let $(\mathcal{T}_h)_{h\in\mathcal{H}}$ be a sequence of shape-regular meshes covering $D$ exactly. Here $\mathcal{H}$ is the index set of the mesh sequence, and $h$ is the typical mesh-size. Given some mesh $\mathcal{T}_h$, we denote by $P(\mathcal{T}_h)$ a scalar-valued finite element space with global shape functions $\{\varphi_i\}_{i\in\mathcal{V}}$. Here, the index $i$ is abusively called a degree of freedom, and since we restrict the presentation to continuous Lagrange elements, degrees of freedom are also called nodes. The approximation of the state $\boldsymbol{u} := (\rho, \boldsymbol{m}, E)$ will be done in the vector-valued space $\boldsymbol{P}(\mathcal{T}_h) := (P(\mathcal{T}_h))^{d+2}$. We define the stencil at $i$ by

$$\mathcal{I}(i) := \left\{ j \in \mathcal{V} \;\middle|\; |\mathrm{supp}(\varphi_j) \cap \mathrm{supp}(\varphi_i)| \neq 0 \right\}, \quad \text{and we set} \quad \mathcal{I}^*(i) := \mathcal{I}(i)\backslash\{i\}.$$

We assume that the shape functions are non-negative, i.e., $\varphi_i \geq 0$ for all $i \in \mathcal{V}$ on all of $D$, and satisfy the partition of unity property $\sum_{i\in\mathcal{V}} \varphi_i = 1$.

The concrete implementation used in the verification and benchmark section Section 6 is based on the finite element library `deal.II` [11,12] and uses continuous mapped $\mathbb{Q}_1$ elements. The code called `ryujin` is available online (https://github.com/conservation-laws/ryujin and documented in Maier et al. [13]). We denote by $\mathcal{V}^\partial$ the set of the degrees of freedom whose shape functions are supported on the boundary $\partial D$. The set $\mathcal{V}^\circ := \mathcal{V}\backslash\mathcal{V}^\partial$ is composed of the degrees of freedom whose shape functions are supported in the interior of $D$. They are henceforth called interior degrees of freedom.

The hyperbolic part of the algorithm depends on four mesh-dependent quantities, $m_i$, $m_{ij}$, $\boldsymbol{c}_{ij}$, and $\boldsymbol{n}_{ij}$ defined as follows for all $i \in \mathcal{V}$ and all $j \in (\mathcal{I})$:

$$m_i := \int_D \varphi_i \,\mathrm{d}x, \qquad m_{ij} := \int_D \varphi_i\varphi_j \,\mathrm{d}x, \qquad \boldsymbol{c}_{ij} := \int_D \varphi_i \nabla\varphi_j \,\mathrm{d}x, \qquad \boldsymbol{n}_{ij} := \frac{\boldsymbol{c}_{ij}}{\|\boldsymbol{c}_{ij}\|_{\ell^2}}. \tag{2.5}$$

Here $m_i$ and $m_{ij}$ are the entries of the lumped mass matrix and consistent mass matrix, respectively. The partition of unity property implies the identities $m_i = \sum_{j\in\mathcal{I}(i)} m_{ij}$ and $\sum_{j\in\mathcal{I}(i)} \boldsymbol{c}_{ij} = 0$. The second identity is essential to establish conservation. Using the lumped mass matrix introduces undesirable dispersive errors, whereas using the consistent mass matrix may require global matrix inversions. The present contribution uses the following approximate inverse of the mass matrix with entries defined by

$$\tfrac{1}{m_i}(\delta_{ij} + b_{ij}), \quad \text{where the coefficients} \quad b_{ij} := \delta_{ij} - \tfrac{m_{ij}}{m_j} \quad \text{satisfy} \quad \textstyle\sum_{i\in\mathcal{I}(j)} b_{ij} = 0. \tag{2.6}$$

Using this approximate inverse bypasses the need to invert the mass matrix. These ideas were originally documented in Guermond and Pasquetti [21] and [22, §3.3]. It is also shown therein that this approximate inverse preserves the conservation properties of the scheme. After extensive benchmarking, it is observed in [23, §3.2] that the best parallel performance is achieved by pre-computing and storing on each MPI rank the coefficients $\{m_i\}_{i\in\mathcal{V}}$ and the matrices $\{m_{ij}\}_{i\in\mathcal{V},j\in\mathcal{I}(i)}$, $\{\boldsymbol{c}_{ij}\}_{i\in\mathcal{V},j\in\mathcal{I}(i)}$. The coefficients of the matrices $\{b_{ij}\}_{i\in\mathcal{V},j\in\mathcal{I}(i)}$, $\{b_{ji}\}_{j\in\mathcal{V},i\in\mathcal{I}(j)}$ and $\{\boldsymbol{n}_{ij}\}_{i\in\mathcal{V},j\in\mathcal{I}(i)}$ can be recomputed on the fly from $\{m_{ij}\}_{i\in\mathcal{V},j\in\mathcal{I}(i)}$, $\{\boldsymbol{c}_{ij}\}_{i\in\mathcal{V},j\in\mathcal{I}(i)}$ in each time step. For later reference, $\mathbb{M}_L$, $\mathbb{M}$, and $\mathbb{B}$ denote: the lumped mass matrix, the consistent mass matrix, and the matrix with entries $\{b_{ij}\}_{i\in\mathcal{V},j\in\mathcal{I}(i)}$ respectively.

**Remark 2.1** (*Space Discretization*)**.** The focus of the paper is on continuous Lagrange elements since the discretization of both the hyperbolic and the diffusion operators is relatively natural with these elements. Discontinuous elements can also be used at the expense of additional overhead in the assembly of the diffusion terms [24]. Another space discretization enjoying a straightforward implementation of the diffusion terms is rational barycentric coordinates on arbitrary polygons/polyhedrons. Rational barycentric coordinates satisfy the partition of unity property and can be made globally continuous (i.e., $H^1$-conforming), see Floater [25] and references therein. All the developments presented in this manuscript are directly applicable in that context too.

## *2.3. Strang splitting*

The key idea for the time approximation of (2.1) is to use Strang's splitting. As routinely done in the literature, we separate the hyperbolic part and the parabolic parts of the problem (see e.g., Demkowicz et al. [8], we also refer the reader to Beam and Warming [6], [7, §11.2] where other operator-splittings are considered). The hyperbolic part of the problem consists of solving the Euler equations:

$$\partial_t\rho + \nabla\cdot(\boldsymbol{v}\rho) = 0, \tag{2.7a}$$

$$\partial_t \boldsymbol{m} + \nabla\cdot(\boldsymbol{v}\otimes\boldsymbol{m} + p(\boldsymbol{u})\mathbb{I}) = \boldsymbol{0}, \tag{2.7b}$$

$$\partial_t E + \nabla\cdot(\boldsymbol{v}(E + p(\boldsymbol{u}))) = 0, \tag{2.7c}$$

which is formally equivalent to considering the limit for $\mu, \lambda, \kappa \to 0$ in (2.1). The missing dissipative terms in (2.7) compose the parabolic part of the problem:

$$\partial_t \rho = 0, \tag{2.8a}$$

$$\partial_t(\rho\boldsymbol{v}) - \nabla\cdot(\mathbb{s}(\boldsymbol{v})) = \boldsymbol{f}, \tag{2.8b}$$

$$\partial_t(\rho e) - c_v^{-1}\kappa\Delta e = \mathbb{s}(\boldsymbol{v}){:}\mathbb{e}(\boldsymbol{v}). \tag{2.8c}$$

This decomposition is the cornerstone of the operator-splitting scheme considered in this paper. For any admissible state $\boldsymbol{u}_0 \in \mathcal{A}$ at time $t_0$ and any time $t \geq t_0$, we denote by $S_{\mathrm{H}}(t, t_0)(\boldsymbol{u}(t_0)) = \boldsymbol{u}(t)$ the solution map of the hyperbolic system (2.7); that is, $\boldsymbol{u}(t)$ solves (2.7) with appropriate boundary conditions and $\boldsymbol{u}(t_0) = \boldsymbol{u}_0$. The subscript $_{\mathrm{H}}$ is meant to remind us that $S_{\mathrm{H}}$ solves the hyperbolic problem. Similarly, letting $\boldsymbol{u_0} \in \mathcal{A}$ be some admissible state at some time $t_0$, and letting $\boldsymbol{f}$ be some source term, we denote by $S_{\mathrm{P}}(t, t_0)(\boldsymbol{u}_0, \boldsymbol{f}) = \boldsymbol{u}(t)$ the solution map of the parabolic system (2.8). Then, given an admissible state $\boldsymbol{u}_0 \in \mathcal{A}$ at time $t_0$ and given some time step $\tau$, we approximate the solution to the full Navier–Stokes system (2.1) at $t_0 + 2\tau$ by using Strang's splitting technique:

$$\boldsymbol{u}(t_0 + 2\tau) \;\approx\; \big(S_{\mathrm{H}}(t_0 + 2\tau, t_0 + \tau) \circ S_{\mathrm{P}}(t_0 + 2\tau, t_0)(\,.\,, \boldsymbol{f}) \circ S_{\mathrm{H}}(t_0 + \tau, t_0)\big)(\boldsymbol{u}_0). \tag{2.9}$$

In other words, we first perform an explicit hyperbolic update of $\boldsymbol{u}_0$ with step size $\tau$. Then, using this update as initial state at $t_0$ and the source term $\boldsymbol{f}$, we solve the parabolic problem from $t_0$ to $t_0 + 2\tau$. Using in turn this solution as initial state at $t_0 + \tau$, we compute the final update by solving (2.7) from $t_0 + \tau$ to $t_0 + 2\tau$.

## 3. Discretization of $S_{\mathrm{H}}$

In this section we describe the space and time approximation of the hyperbolic operator $S_{\mathrm{H}}$ and summarize important implementation details that make the algorithm efficient and highly scalable. The time approximation is done by using the explicit strong stability preserving Runge–Kutta method SSPRK(3,3), see [26, Eq. (2.18)] and [27, Thm. 9.4]. This method requires three calls to the forward-Euler update discussed in this section. The forward-Euler scheme itself requires the computation of a low-order solution, a provisional high-order solution (possibly constraint violating), and the final flux-limited solution to be returned. The various steps described in Sections 3.1–3.3 are summarized in Algorithm 1 in Appendix B. The implementation of the boundary conditions for $S_{\mathrm{H}}$ is explained in Section 4.

### *3.1. Low-order step*

Let $t^n$ be the current time and let $\boldsymbol{u}_h^n = \sum_{i\in\mathcal{V}} \mathbf{U}_i^n \varphi_i$ be the current approximation which we assume to be admissible, i.e., $\mathbf{U}_i^n \in \mathcal{A}$ for all $i \in \mathcal{V}$. For all $i \in \mathcal{V}$ and for all $j \in \mathcal{I}^*(i)$, we consider the Riemann problem with left state $\mathbf{U}_i^n$, right state $\mathbf{U}_j^n$, and flux $\mathbb{f}(\boldsymbol{w})\boldsymbol{n}_{ij}$. We denote by $\lambda^{\max}(\mathbf{U}_i^n, \mathbf{U}_j^n, \boldsymbol{n}_{ij})$ any upper bound on the maximum wavespeed in this Riemann problem. Iterative techniques to compute the maximum wavespeed are described in Colella and Glaz [28], Toro [29]. In this manuscript we use the inexpensive non-iterative guaranteed upper-bound thoroughly described in [19,30]. With this estimate, we define the graph viscosity coefficient:

$$d_{ij}^{\mathrm{L},n} = \max\big(\lambda^{\max}(\mathbf{U}_i^n, \mathbf{U}_j^n, \boldsymbol{n}_{ij})\|\boldsymbol{c}_{ij}\|_{\ell^2}, \lambda^{\max}(\mathbf{U}_j^n, \mathbf{U}_i^n, \boldsymbol{n}_{ji})\|\boldsymbol{c}_{ji}\|_{\ell^2}\big). \tag{3.1}$$

Noticing that $\boldsymbol{c}_{ij} = -\boldsymbol{c}_{ji}$ if $i$ is an internal node (because $\varphi_{i|\partial D} = 0$ if $i \in \mathcal{V}^\circ$), we infer that $\lambda^{\max}(\mathbf{U}_i^n, \mathbf{U}_j^n, \boldsymbol{n}_{ij})\|\boldsymbol{c}_{ij}\|_{\ell^2} = \lambda^{\max}(\mathbf{U}_j^n, \mathbf{U}_i^n, \boldsymbol{n}_{ji})\|\boldsymbol{c}_{ji}\|_{\ell^2}$ for all $i \in \mathcal{V}^\circ$. This property allows for some computation savings in the construction of $d^{\mathrm{L},n}$. For all $i \in \mathcal{V}^\circ$ and all $i < j \in \mathcal{I}(i)$, one computes $d_{ij}^{\mathrm{L},n} = \lambda^{\max}(\mathbf{U}_i^n, \mathbf{U}_j^n, \boldsymbol{n}_{ij})\|\boldsymbol{c}_{ij}\|_{\ell^2}$, and for all $i \in \mathcal{V}^\partial$ and all $i < j \in \mathcal{I}(i)$, one computes $d_{ij}^{\mathrm{L},n} = \max(\lambda^{\max}(\mathbf{U}_i^n, \mathbf{U}_j^n, \boldsymbol{n}_{ij})\|\boldsymbol{c}_{ij}\|_{\ell^2}, \lambda^{\max}(\mathbf{U}_j^n, \mathbf{U}_i^n, \boldsymbol{n}_{ji})\|\boldsymbol{c}_{ji}\|_{\ell^2})$. Finally one sets $d_{ij}^{\mathrm{L},n} \leftarrow \max(d_{ij}^{\mathrm{L},n}, (d^{\mathrm{L},n})_{ij}^{\mathsf{T}})$ for all $j \in \mathcal{I}^*(i)$, where $(d^{\mathrm{L},n})^{\mathsf{T}}$ is the transpose of $d^{\mathrm{L},n}$. The diagonal entries in $d^{\mathrm{L},n}$ are obtained by setting $d_{ii}^{\mathrm{L},n} := -\sum_{j\in\mathcal{I}^*(i)} d_{ij}^{\mathrm{L},n}$. This technique saves almost half the computing time for $d_{ij}^{\mathrm{L},n}$ [23, §5.2.1]. Once $d^{\mathrm{L},n}$ is known, the time-step size is defined by

$$\tau_n := \mathrm{c}_{\mathrm{cfl}} \times \min_{i\in\mathcal{V}}\Big(-\frac{m_i}{2d_{ii}^{\mathrm{L},n}}\Big), \tag{3.2}$$

where $0 < c_{\text{cfl}} \le 1$ is a user-defined constant. The condition $c_{\text{cfl}} \le 1$ is shown in [31] to be sufficient to guarantee that the low-order method is invariant domain preserving. The time-step size is computed at the first forward-Euler step of the SSPRK(3,3) algorithm, and this time-step size is used for the three stages of the hyperbolic update. Then according to the Strang splitting algorithm (2.9), the time-step size used in the parabolic update is $2\tau_n$, and the time-step size used in the last hyperbolic update is again $\tau_n$. At the end of the entire process the new time level is $t^n + 2\tau_n$.

The low-order update produced by the forward-Euler step as defined in [31] is

$$m_i(\mathbf{U}_i^{\mathrm{L},n+1} - \mathbf{U}_i^n) = \tau_n \sum_{j\in\mathcal{I}(i)} \mathbf{F}_{ij}^{\mathrm{L}}, \qquad \mathbf{F}_{ij}^{\mathrm{L}} := -\mathbb{f}(\mathbf{U}_j^n)\boldsymbol{c}_{ij} + d_{ij}^{\mathrm{L},n}(\mathbf{U}_j^n - \mathbf{U}_i^n). \tag{3.3}$$

The CFL condition $c_{\text{cfl}} \le 1$ guarantees that $\mathbf{U}^{\mathrm{L},n+1}$ remains inside the invariant domain. Consequently, for our choice of Lagrange elements (see Section 2.2) the invariant domain property holds true for the finite element function $\boldsymbol{u}_h^{\mathrm{L},n+1}$ [31, Corollary 4.3].

**Remark 3.1.** [High aspect-ratio meshes] The time-step size $\tau_n$ determined by the theoretical estimate (3.2) decreases significantly as the aspect ratio of the cells increases. This may be problematic when using meshes with high aspect-ratio cells to resolve thin boundary layers (see Section 6.4); in this case the aspect ratios can reach values up to 50:1 or more. For these configurations we have found that a better way to estimate $\tau_n$ is to adaptively increase the value of $c_{\text{cfl}}$ in (3.2) beyond the limit of 1. This requires to additionally check whether the low-order solution $\mathbf{U}^{\mathrm{L},n+1}$ still remains in the admissible set, and if not to restart the time step with a smaller $c_{\text{cfl}}$ number. While this makes each time step slightly more expensive, the significant increase of the CFL number compensates for the otherwise increased cost of using a high aspect-ratio cells.

To save arithmetic operations and prepare the ground for the limiting step, we introduce the auxiliary states $\overline{\mathbf{U}}_{ij}^n$ and rewrite (3.3) as follows:

$$\overline{\mathbf{U}}_{ij}^n := \frac{1}{2}(\mathbf{U}_i^n + \mathbf{U}_j^n) - \frac{1}{2d_{ij}^{\mathrm{L},n}}(\mathbb{f}(\mathbf{U}_j^n) - \mathbb{f}(\mathbf{U}_i^n))\boldsymbol{c}_{ij}, \qquad \forall j \in \mathcal{I}(i), \tag{3.4}$$

$$\mathbf{U}_i^{\mathrm{L},n+1} = \mathbf{U}_i^n + \frac{2\tau_n}{m_i} \sum_{j\in\mathcal{I}(i)} d_{ij}^{\mathrm{L},n} \overline{\mathbf{U}}_{ij}^n. \tag{3.5}$$

The auxiliary states $\overline{\mathbf{U}}_{ij}^n$ are essential to define the bounds that must be guaranteed after limiting. In particular, if one wants to limit some quasi-concave functional $\Psi$, one has to compute the local lower bound $\Psi_i^{\min} := \min_{j\in\mathcal{I}(i)} \Psi(\overline{\mathbf{U}}_{ij}^n)$. In the numerical illustrations reported in the paper, limiting is done with the following two functionals: $\Psi_\flat(\mathbf{U}) = \varrho$ and $\Psi_\sharp(\mathbf{U}) = -\varrho$ (where recalling the notation introduced in Section 2.1 we have set $\varrho := \rho(\mathbf{U})$). One uses $\Psi_\flat$ to enforce a local minimum principle on the density and one uses $\Psi_\sharp$ to enforce a local maximum principle. Additional limiting has to be done to ensure that the specific internal energy is positive. This is done in the case of a $\gamma$-law equation of state by controlling the exponential of the specific entropy, $\Phi(\mathbf{U}) = \varepsilon(\mathbf{U})\varrho^{-\gamma}$ where $\varepsilon(\mathbf{U}) := E - \frac{|\boldsymbol{m}|^2}{2\rho}$ is the internal energy. For theoretical reasons explained in [32, §3.2], the local lower bound for this functional uses the states $\mathbf{U}_j^n$ instead of the auxiliary states $\overline{\mathbf{U}}_{ij}^n$, i.e., one defines $\Phi_i^{\min} := \min_{j\in\mathcal{I}(i)} \Phi(\mathbf{U}_j^n)$.

## *3.2. High-order step*

The computation of the provisional high-order update proceeds as for the low-order update with two exceptions: (i) the graph viscosity is reduced; (ii) the lumped mass matrix is replaced by an approximation of the consistent mass matrix to correct third-order dispersive effects. More precisely the high-order viscosity is defined as follows: $d_{ij}^{\mathrm{H},n} = (\frac{\alpha_i+\alpha_j}{2})d_{ij}^{\mathrm{L},n}$, where $0 \le \alpha_i \le 1$ is an entropy-production indicator (see [32, §3.4] and Maier and Kronbichler [23] for some possible implementations). The key idea is that $\alpha_i + \alpha_j$ is small in regions where the solution is smooth and there is no entropy production. We introduce $\mathbf{F}^{\mathrm{H}}$ to be the vector of the high-order fluxes whose entries are $(d+2)$-valued and defined for every $i \in \mathcal{V}$ by

$$\mathbf{F}_i^{\mathrm{H}} := \sum_{j\in\mathcal{I}(i)} \mathbf{F}_{ij}^{\mathrm{H}} \quad \text{where} \quad \mathbf{F}_{ij}^{\mathrm{H}} := -\mathbb{f}(\mathbf{U}_j^n)\boldsymbol{c}_{ij} + d_{ij}^{\mathrm{H},n}(\mathbf{U}_j^n - \mathbf{U}_i^n). \tag{3.6}$$

Recalling that $\mathbb{M}^{-1} \approx \mathbb{M}_L^{-1}(\mathbb{I}+\mathbb{B})$, the high-order update is obtained by setting

$$\mathbb{M}_L(\mathbf{U}^{\mathrm{H},n+1} - \mathbf{U}^n) := \tau_n(\mathbb{I}+\mathbb{B})\mathbf{F}^{\mathrm{H}}. \tag{3.7}$$

Instead of using this expression, we proceed as in [22, §3.4] to prepare the ground for limiting. Recalling that $b_{ii} = -\sum_{j\in\mathcal{I}^*(i)} b_{ji}$, we rewrite the high-order update (3.7) as follows:

$$m_i(\mathbf{U}_i^{\mathrm{H},n+1} - \mathbf{U}_i^n) = \tau_n\mathbf{F}_i^{\mathrm{H}} + \tau_n \sum_{j\in\mathcal{I}^*(i)} b_{ij}\mathbf{F}_j^{\mathrm{H}} - b_{ji}\mathbf{F}_i^{\mathrm{H}}. \tag{3.8}$$

Now we subtract (3.3) from (3.8) and obtain

$$m_i\mathbf{U}_i^{\mathrm{H},n+1} = m_i\mathbf{U}_i^{\mathrm{L},n} + \tau_n \sum_{j\in\mathcal{I}^*(i)} b_{ij}\mathbf{F}_j^{\mathrm{H}} - b_{ji}\mathbf{F}_i^{\mathrm{H}} + (d_{ij}^{\mathrm{H},n} - d_{ij}^{\mathrm{L},n})(\mathbf{U}_j^n - \mathbf{U}_i^n). \tag{3.9}$$

The state $\mathbf{U}^{\mathrm{H},n+1}$ is a high-order approximation of $\boldsymbol{u}(t^{n+1})$ if the viscosity $d_{ij}^{\mathrm{H},n}$ is indeed small, but it may not be admissible. To save arithmetic operations, one does not compute $\mathbf{U}^{\mathrm{H},n+1}$ since the actual high-order update is obtained after limiting as explained in the next subsection.

### 3.3. Limiting

Recall that, as discussed at the end of Section 3.1, we want the high-order update to satisfy $\Psi_\flat(\mathbf{U}_i^{n+1}) \geq \Psi_{\flat,i}^{\min}$, $\Psi_\sharp(\mathbf{U}_i^{n+1}) \geq \Psi_{\sharp,i}^{\min}$, and $\Phi(\mathbf{U}_i^{n+1}) \geq \Phi_i^{\min}$ for all $i \in \mathcal{V}$. For this purpose we rewrite (3.9) as follows:

$$\mathbf{U}_i^{\mathrm{H},n+1} = \sum_{j\in\mathcal{I}^*(i)} \lambda_i(\mathbf{U}_i^{\mathrm{L},n} + \mathbf{P}_{ij}^n), \tag{3.10}$$

$$\text{with}\quad \mathbf{P}_{ij}^n := \frac{\tau_n}{m_i\lambda_i}\Big(b_{ij}\mathbf{F}_j^{\mathrm{H}} - b_{ji}\mathbf{F}_i^{\mathrm{H}} + (d_{ij}^{\mathrm{H},n} - d_{ij}^{\mathrm{L},n})(\mathbf{U}_j^n - \mathbf{U}_i^n)\Big), \tag{3.11}$$

where $\lambda_i := (\mathrm{card}(\mathcal{I}(i)) - 1)^{-1}$. This motivates computing the final (flux-limited) solution as

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^{\mathrm{L},n+1} + \sum_{j\in\mathcal{I}^*(i)} \lambda_i \ell_{ij}\mathbf{P}_{ij}^n, \tag{3.12}$$

where $\ell_{ij} \in [0,1]$ for all $\{i,j\}$ are the limiters. Observe that if $\ell_{ij} = 1$ for all $j \in \mathcal{I}^*(i)$ then $\mathbf{U}_i^{n+1} = \mathbf{U}_i^{\mathrm{H},n+1}$, and if $\ell_{ij} = 0$ for all $j \in \mathcal{I}^*(i)$ then $\mathbf{U}_i^{n+1} = \mathbf{U}_i^{\mathrm{L},n+1}$. For every $j \in \mathcal{I}^*(i)$, we define $\ell_j^i$ to be the largest number in $[0,1]$ that is such that

$$\Psi_\flat(\mathbf{U}_i^{\mathrm{L},n} + \ell_j^i\mathbf{P}_{ij}) \geq \Psi_{\flat,i}^{\min}, \qquad \Psi_\sharp(\mathbf{U}_i^{\mathrm{L},n} + \ell_j^i\mathbf{P}_{ij}) \geq \Psi_{\sharp,i}^{\min}, \qquad \Phi(\mathbf{U}_i^{\mathrm{L},n} + \ell_j^i\mathbf{P}_{ij}) \geq \Phi_i^{\min}. \tag{3.13}$$

This number always exists since by construction $\ell_j^i = 0$ satisfies the above three constraints. Finding this number (or a very close lower estimate thereof) is quite simple and explained in [20,23]. Then, in order to maintain mass conservation, $\ell_{ij}$ is defined by setting $\ell_{ij} = \min(\ell_j^i, \ell_i^j)$. This symmetry property, together with the identity $m_i\lambda_i\boldsymbol{P}_{ij} = -m_j\lambda_j\boldsymbol{P}_{ji}$, ensures that the mass of the high-order update is unchanged by limiting, i.e.,

$$\sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^{n+1} = \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^{\mathrm{L},n+1}. \tag{3.14}$$

Replacing $\ell_j^i$ by $\min(\ell_j^i, \ell_i^j)$ does not violate the invariant domain properties since $\mathcal{A}$ is convex, [20].

Since $\mathbf{U}_i^{n+1} = \sum_{j\in\mathcal{I}^*(i)} \lambda_i(\mathbf{U}_i^{\mathrm{L},n+1} + \ell_{ij}\mathbf{P}_{ij}^n) + \sum_{j\in\mathcal{I}^*(i)} \lambda_i(1-\ell_{ij})\mathbf{P}_{ij}^n$, and $\sum_{j\in\mathcal{I}^*(i)} \lambda_i(\mathbf{U}_i^{\mathrm{L},n+1} + \ell_{ij}\mathbf{P}_{ij}^n)$ satisfies all the bounds, one can repeat the above process and compute a new set of limiters by replacing $\mathbf{U}_i^{\mathrm{L},n+1}$ by $\sum_{j\in\mathcal{I}^*(i)} \lambda_i(\mathbf{U}_i^{\mathrm{L},n+1} + \ell_{ij}\mathbf{P}_{ij}^n)$ and $\mathbf{P}_{ij}^n$ by $(1-\ell_{ij})\mathbf{P}_{ij}^n$. We have observed that this iterative limiting process must be applied at least two times to reach optimal convergence. All the simulations reported in the paper are done with two passes of limiting.

Let $\boldsymbol{n}$ denote the outward unit normal vector field on $\partial D$. To properly formulate the conservation properties of the method after limiting, we define an approximation of the normal vector and boundary mass at every boundary node $i \in \mathcal{V}^\partial$ by setting

$$\boldsymbol{n}_i := \frac{\int_{\partial D} \varphi_i\,\boldsymbol{n}\,\mathrm{d}s}{m_i{}^\partial}, \qquad m_i{}^\partial := \Big\| \int_{\partial D} \varphi_i\,\boldsymbol{n}\,\mathrm{d}s \Big\|_{\ell^2}. \tag{3.15}$$

**Lemma 3.2** (*Balance of Mass and Admissibility After Limiting*).

(i) *For all* $\boldsymbol{u}_h := \sum_{i\in\mathcal{V}} \mathbf{U}_i\varphi \in \boldsymbol{P}(\mathcal{T}_h)$*, the following holds true:* $\int_D \boldsymbol{u}_h \,\mathrm{d}x = \sum_{i\in\mathcal{V}} m_i \mathbf{U}_i$.

(ii) *Let* $\mathbf{U}^n$ *be a collection of admissible states. Let* $\mathbf{U}^{n+1}$ *be the update after one forward-Euler step and after limiting. Then* $\mathbf{U}^{n+1}$ *is admissible under the condition* $c_{\mathrm{cfl}} \le 1$ *and*

$$\sum_{i\in\mathcal{V}} m_i \mathbf{U}_i^{n+1} + \tau_n \sum_{i\in\mathcal{V}^\partial} m_i^\partial \mathbb{f}(\mathbf{U}_i^n)\boldsymbol{n}_i = \sum_{i\in\mathcal{V}} m_i \mathbf{U}_i^n, \tag{3.16}$$

**Proof.** See A.1 in Appendix A.

**Remark 3.3** (*Literature*). The convex limiting technique is a generalization of the Flux Corrected Transport that accommodates quasi-concave constraints. (Recall that FCT is by design adapted to affine constraints; see e.g., Boris and Book [33], Zalesak [34], Kuzmin et al. [35].) Convex limiting has been introduced in [32], Guermond et al. [20] for the Euler equations and general hyperbolic systems. We refer to Maier and Kronbichler [23], Maier and Tomas [36] for a detailed discussion of a high performance implementation of the hyperbolic solver part of the system.

## 4. Euler boundary conditions

In this section we describe how boundary conditions are enforced in the hyperbolic step. To the best of our knowledge, the implementation details of the various boundary conditions considered in this section for continuous finite elements, and the associated theoretical results regarding conservation and admissibility are original.

### *4.1. Overview*

Since the time stepping is explicit, the boundary conditions are enforced by post-processing the approximation produced at the end of each stage of the SSPRK(3,3) algorithm. The Butcher tableau of the explicit SSPRK(3,3) algorithm is given in the left panel of (4.1). Given some ODE system $\partial_t u = L(t,u)$ and $u^n := u(t^n)$, the steps to approximate the solution to $\partial_t u = L(t,u)$ at $t^{n+1}$ are shown in the right panel of (4.1).

$$\begin{array}{c|ccc} 0 & 0 & & \\ 1 & 1 & 0 & \\ \frac12 & \frac14 & \frac14 & 0 \\ \hline & \frac16 & \frac16 & \frac23 \end{array} \qquad \begin{aligned} w^{(1)} &:= u^n + \tau_n L(t_n, u^n), \\ w^{(2)} &:= \tfrac34 u^n + \tfrac14\big(w^{(1)} + \tau_n L\big(t_n+\tau_n, w^{(1)}\big)\big), \\ u^{n+1} &:= \tfrac13 u^n + \tfrac23\big(w^{(2)} + \tau_n L\big(t_n + \tfrac12\tau_n, w^{(2)}\big)\big). \end{aligned} \tag{4.1}$$

The intermediate stages $w^{(1)}$, $w^{(2)}$ and the final stage $u^{n+1}$ approximate $u$ at $t^{n+1} = t^n + \tau_n$, $t^n + \frac12\tau_n$, and $t^n + \tau_n$, respectively. Hence, the time-dependent boundary conditions have to be enforced on the intermediate stages $w^{(1)}$, $w^{(2)}$ and the final step $u^{n+1}$ (using the corresponding collocation times).

We consider two types of boundary conditions: (i) Slip condition, also called "reflecting": $\boldsymbol{v}\cdot\boldsymbol{n} = 0$; (ii) Non-reflecting condition. Let $\partial D_{\mathrm{s}} \subset \partial D$ be the boundary where one wants to enforce the slip condition, and let $\partial D_{\mathrm{nr}}$ denote the complement of $\partial D_{\mathrm{s}}$ in $\partial D$, i.e., $\partial D\backslash\partial D_{\mathrm{s}}$. The index $_{\mathrm{nr}}$ reminds us that is a non-reflecting boundary (either an inflow or an outflow boundary). Let $\mathcal{V}_{\mathrm{s}}^\partial \subset \mathcal{V}^\partial$ be the collection of all the boundary degrees of freedom $i$ such that $\varphi_{i|\partial D_{\mathrm{s}}} \not\equiv 0$. Similarly, $\mathcal{V}_{\mathrm{nr}}^\partial \subset \mathcal{V}^\partial$ is the collection of all the boundary degrees of freedom $i$ such that $\varphi_{i|\partial D_{\mathrm{nr}}} \not\equiv 0$. We now define the normal vectors associated with the degrees of freedom in $\mathcal{V}_{\mathrm{s}}^\partial$ and $\mathcal{V}_{\mathrm{nr}}^\partial$:

$$\boldsymbol{n}_i^{\mathrm{s}} := \frac{\int_{\partial D_{\mathrm{s}}} \varphi_i \boldsymbol{n}\,\mathrm{d}s}{\|\int_{\partial D_{\mathrm{s}}} \varphi_i \boldsymbol{n}\,\mathrm{d}s\|_{\ell^2}}, \qquad \boldsymbol{n}_i^{\mathrm{nr}} := \frac{\int_{\partial D_{\mathrm{nr}}} \varphi_i \boldsymbol{n}\,\mathrm{d}s}{\|\int_{\partial D_{\mathrm{nr}}} \varphi_i \boldsymbol{n}\,\mathrm{d}s\|_{\ell^2}}. \tag{4.2}$$

Notice that although $\partial D_{\mathrm{s}} \cap \partial D_{\mathrm{nr}} = \emptyset$, the two index sets $\mathcal{V}_{\mathrm{s}}^\partial$ and $\mathcal{V}_{\mathrm{nr}}^\partial$ may not be disjoint. Hence, there may exists two notions of the normal vector at the nodes sitting at the interface between $\partial D_{\mathrm{s}}$ and $\partial D_{\mathrm{nr}}$. Let us set $m_i{}^{\mathrm{s}} := \|\int_{\partial D_{\mathrm{s}}} \varphi_i \boldsymbol{n}\,\mathrm{d}s\|_{\ell^2}$ and $m_i^{\mathrm{nr}} := \|\int_{\partial D_{\mathrm{nr}}} \varphi_i \boldsymbol{n}\,\mathrm{d}s\|_{\ell^2}$. Then (3.15) and (4.2) imply that $m_i{}^\partial \boldsymbol{n}_i = m_i{}^{\mathrm{s}}\boldsymbol{n}_i{}^{\mathrm{s}} + m_i^{\mathrm{nr}}\boldsymbol{n}_i^{\mathrm{nr}}$.

In the following subsections, the symbol $\mathbf{U}$ denotes the state obtained at the end of one forward-Euler step. This state has to be postprocessed to account for the boundary conditions. It could be any one of the three states $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, or $\mathbf{U}^{n+1}$. The postprocessed state is denoted $\mathbf{U}^{\mathcal{P}}$.

### 4.2. Slip boundary condition

We start with the slip boundary condition. Let $i \in \mathcal{V}^{\partial}_{\mathrm{s}}$ and let $\mathbf{U}_i = (\varrho_i, \mathbf{M}_i, \mathsf{E}_i)^{\mathsf{T}}$, i.e., $\varrho_i := \rho(\mathbf{U}_i)$, $\mathbf{M}_i := \boldsymbol{m}(\mathbf{U}_i)$, and $\mathsf{E}_i = E(\mathbf{U}_i)$. We enforce the slip boundary condition at $i$ by setting

$$\mathbf{U}_i^{\mathcal{P}} := (\varrho_i, \mathbf{M}_i - (\mathbf{M}_i{\cdot}\boldsymbol{n}_i{}^{\mathrm{s}})\boldsymbol{n}_i{}^{\mathrm{s}}, \mathsf{E}_i)^{\mathsf{T}}. \tag{4.3}$$

### 4.3. Non-reflecting boundary condition

We now consider non-reflecting boundary conditions at $i \in \mathcal{V}^{\partial}_{\mathrm{nr}}$. To simplify the notation, we omit the node index $_i$ since no argument regarding conservation properties is made. We also write $\boldsymbol{n}$ instead of $\boldsymbol{n}_i^{\mathrm{nr}}$. We propose two post-processing techniques: (i) one based on Godunov's method; (ii) the other uses the characteristic variables (or proxies thereof).

#### 4.3.1. Godunov's method

We assume that on the outer side of the boundary we are given some ideal, admissible state $\mathbf{U}_i^{\mathrm{D}} = (\varrho^{\mathrm{D}}, \mathbf{M}^{\mathrm{D}}, \mathsf{E}^{\mathrm{D}})$ related to the far-field conditions, which we call Dirichlet state. Then we consider the Riemann problem $\partial_t \boldsymbol{v} + \partial_x(\mathbb{f}(\boldsymbol{v})\boldsymbol{n}) = \mathbf{0}$ with left data $\mathbf{U}$ and right data $\mathbf{U}^{\mathrm{D}}$. Let $G(\boldsymbol{n}, \mathbf{U}, \mathbf{U}^{\mathrm{D}})$ denote the value of the solution of the Riemann problem at $x = 0$. The post-processing then consists of setting

$$\mathbf{U}^{\mathcal{P}} = G(\boldsymbol{n}, \mathbf{U}, \mathbf{U}^{\mathrm{D}}). \tag{4.4}$$

Notice that $\mathbf{U}^{\mathcal{P}}$ is automatically admissible. Since this operation may be expensive, we propose an alternative approach in the next section.

#### 4.3.2. Characteristic variables

We now assume that the equation of state is described by the $\gamma$-law and propose a technique based on characteristic variable. The method is loosely based on [37, §3] and has some similarities with [8, §2.6], but instead of working on increments as in [8] we directly work on the characteristic variables. The key results of this section are (4.9)–(4.12)–(4.14)–(4.15).

We define $\varrho := \rho(\mathbf{U})$, $\mathbf{M} := \boldsymbol{m}(\mathbf{U})$, $\mathsf{P} := p(\mathbf{U}) = (\gamma - 1)\varrho e(\mathbf{U})$, $\mathsf{S}(\mathbf{U}) := \varrho^{-\gamma}\mathsf{P}$, and set

$$\mathbf{V} := \varrho^{-1}\mathbf{M}, \qquad \mathsf{V}_n := \mathbf{V}{\cdot}\boldsymbol{n}, \qquad \mathbf{V}^{\perp} := \mathbf{V} - (\mathbf{V}{\cdot}\boldsymbol{n})\boldsymbol{n}, \qquad a := \sqrt{\gamma \mathsf{P}\varrho^{-1}}, \tag{4.5}$$

We start by recalling that, although characteristic variables do not exist in general for the one-dimensional system $\partial_t \boldsymbol{v} + \partial_x(\mathbb{f}(\boldsymbol{v})\boldsymbol{n}) = \mathbf{0}$, characteristic variables and characteristic speeds do exist under the assumption that the flow is locally isentropic. Making this assumption, we obtain

$$\underbrace{\begin{cases}\lambda_1(\mathbf{U}, \boldsymbol{n}) := \mathsf{V}_n - a, \\ C_1(\mathbf{U}, \boldsymbol{n}) := \mathsf{V}_n - \frac{2a}{\gamma-1}\end{cases}}_{\text{mutiplicity } 1} \qquad \underbrace{\begin{cases}\lambda_2(\mathbf{U}, \boldsymbol{n}) := \mathsf{V}_n, \\ \mathsf{S}(\mathbf{U}),\ \mathbf{V}^{\perp}\end{cases}}_{\text{mutiplicity } d} \qquad \underbrace{\begin{cases}\lambda_3(\mathbf{U}, \boldsymbol{n}) := \mathsf{V}_n + a, \\ C_3(\mathbf{U}, \boldsymbol{n}) := \mathsf{V}_n + \frac{2a}{\gamma-1}.\end{cases}}_{\text{mutiplicity } 1} \tag{4.6}$$

Since the eigenvalues are ordered, we distinguish four different cases:

| | | | |
|---|---|---|---|
| (i) | supersonic inflow | $\mathsf{V}_n < 0$ and $a < \lvert\mathsf{V}_n\rvert$ | $\lambda_1(\mathbf{U}, \boldsymbol{n}) \le \lambda_2(\mathbf{U}, \boldsymbol{n}) \le \lambda_3(\mathbf{U}, \boldsymbol{n}) < 0$ |
| (ii) | subsonic inflow | $\mathsf{V}_n < 0$ and $\lvert\mathsf{V}_n\rvert \le a$ | $\lambda_1(\mathbf{U}, \boldsymbol{n}) \le \lambda_2(\mathbf{U}, \boldsymbol{n}) < 0 \le \lambda_3(\mathbf{U}, \boldsymbol{n})$ |
| (iii) | subsonic outflow | $0 \le \mathsf{V}_n$ and $\lvert\mathsf{V}_n\rvert < a$ | $\lambda_1(\mathbf{U}, \boldsymbol{n}) < 0 \le \lambda_2(\mathbf{U}, \boldsymbol{n}) \le \lambda_3(\mathbf{U}, \boldsymbol{n})$ |
| (iii) | supersonic outflow | $0 \le \mathsf{V}_n$ and $a \le \lvert\mathsf{V}_n\rvert$ | $0 \le \lambda_1(\mathbf{U}, \boldsymbol{n}) \le \lambda_2(\mathbf{U}, \boldsymbol{n}) \le \lambda_3(\mathbf{U}, \boldsymbol{n})$. |

We assume that on the outer side of the boundary we are given some Dirichlet state $\mathbf{U}^{\mathrm{D}} := (\varrho^{\mathrm{D}}, \mathbf{M}^{\mathrm{D}}, \mathsf{E}^{\mathrm{D}})$. We are going to postprocess $\mathbf{U}$ such that the characteristic variables of the post-processed state $\mathbf{U}^{\mathcal{P}}$ associated with in-coming eigenvalues match those of the prescribed Dirichlet state, while leaving the out-going characteristics unchanged. More precisely, the proposed strategy consists of seeking $\mathbf{U}^{\mathcal{P}}$ so that the following holds true:

$$C_l(\mathbf{U}^{\mathcal{P}}) = \begin{cases} C_l(\mathbf{U}^{\mathrm{D}}) & \text{if } \lambda_l(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}}) < 0, \\ C_l(\mathbf{U}) & \text{if } 0 \le \lambda_l(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}}), \end{cases} \qquad l \in \{1, 3\}, \tag{4.7}$$

$$S(\mathbf{U}^{\mathcal{P}}) = \begin{cases} S(\mathbf{U}^{\mathrm{D}}) & \text{if } \lambda_2(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}}) < 0, \\ S(\mathbf{U}) & \text{if } 0 \le \lambda_2(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}}), \end{cases} \qquad (\mathbf{V}^{\mathcal{P}})^{\perp} = \begin{cases} (\mathbf{V}^{\mathrm{D}})^{\perp} & \text{if } \lambda_2(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}}) < 0, \\ \mathbf{V}^{\perp} & \text{if } 0 \le \lambda_2(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}}). \end{cases} \tag{4.8}$$

(Note that the condition $\lambda_2(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}}) < 0$ is equivalent to $|\lambda_1(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}})| > |\lambda_3(\mathbf{U}, \boldsymbol{n}^{\mathrm{nr}})|$.) Recalling that we assumed that the evolution of the flow field is locally isentropic, we now solve the above system in the four cases identified above.

*Supersonic inflow condition.* Assume that $\lambda_1(\mathbf{U}, \boldsymbol{n}) \le \lambda_2(\mathbf{U}, \boldsymbol{n}) \le \lambda_3(\mathbf{U}, \boldsymbol{n}) < 0$. Since all the characteristics enter the computational domain, the post-processing consists of replacing $\mathbf{U}$ by $\mathbf{U}^{\mathrm{D}}$:

$$\mathbf{U}^{\mathcal{P}} = \mathbf{U}^{\mathrm{D}}. \tag{4.9}$$

*Subsonic inflow boundary.* Assume that $\lambda_1(\mathbf{U}, \boldsymbol{n}) \le \lambda_2(\mathbf{U}, \boldsymbol{n}) < 0 \le \lambda_3(\mathbf{U}, \boldsymbol{n})$. Then, $\mathbf{U}^{\mathcal{P}}$ is obtained by solving the system

$$C_1(\mathbf{U}^{\mathcal{P}}) = C_1(\mathbf{U}^{\mathrm{D}}), \qquad S(\mathbf{U}^{\mathcal{P}}) = S(\mathbf{U}^{\mathrm{D}}), \qquad (\mathbf{V}^{\mathcal{P}})^{\perp} = (\mathbf{V}^{\mathrm{D}})^{\perp}, \qquad C_3(\mathbf{U}^{\mathcal{P}}) = C_3(\mathbf{U}). \tag{4.10}$$

Notice that, as expected, $d + 1$ Dirichlet conditions are enforced. This gives $\mathsf{V}_n^{\mathcal{P}} = \frac{1}{2}(C_1(\mathbf{U}^{\mathrm{D}}) + C_3(\mathbf{U}))$, $\mathsf{P}^{\mathcal{P}} = S(\mathbf{U}^{\mathrm{D}})(\varrho^{\mathcal{P}})^{\gamma}$, and

$$a^{\mathcal{P}} = \tfrac{\gamma-1}{4}(C_3(\mathbf{U}) - C_1(\mathbf{U}^{\mathrm{D}})) = \tfrac{\gamma-1}{4}\mathsf{V}_n + \frac{a}{2} - \tfrac{\gamma-1}{4}\mathsf{V}_n^{\mathrm{D}} + \tfrac{a^{\mathrm{D}}}{2}. \tag{4.11}$$

Notice that $0 < a^{\mathcal{P}}$ if $\gamma \le 3$ and $\frac{\gamma-1}{2}\mathsf{V}_n^{\mathrm{D}} \le a^{\mathrm{D}}$, which is always the case for realistic $\gamma$-laws. (Here $\frac{\gamma-1}{2}\mathsf{V}_n^{\mathrm{D}} \le a^{\mathrm{D}}$ is an admissibility condition on the Dirichlet data.) Using $(a^{\mathcal{P}})^2 = \gamma \mathsf{P}^{\mathcal{P}}(\varrho^{\mathcal{P}})^{-1}$ with $\mathsf{P}^{\mathcal{P}} = S(\mathbf{U}^{\mathrm{D}})(\varrho^{\mathcal{P}})^{\gamma}$, the post-processing for a subsonic inflow boundary condition consists of setting:

$$\begin{cases} \rho^{\mathcal{P}} = \left(\dfrac{1}{\gamma S(\mathbf{U}^{\mathrm{D}})}\left(\dfrac{\gamma-1}{4}\big(C_3(\mathbf{U}) - C_1(\mathbf{U}^{\mathrm{D}})\big)\right)^2\right)^{\frac{1}{\gamma-1}}, \\ \mathbf{M}^{\mathcal{P}} = \rho^{\mathcal{P}}\left((\mathbf{V}^{\mathrm{D}})^{\perp} + \mathsf{V}_n^{\mathcal{P}}\boldsymbol{n}\right), \quad \text{with} \quad \mathsf{V}_n^{\mathcal{P}} = \dfrac{1}{2}\big(C_1(\mathbf{U}^{\mathrm{D}}) + C_3(\mathbf{U})\big), \\ E^{\mathcal{P}} = \dfrac{1}{\gamma-1}\mathsf{P}^{\mathcal{P}} + \dfrac{1}{2}\dfrac{\|\mathbf{M}^{\mathcal{P}}\|_{\ell^2}^2}{\varrho^{\mathcal{P}}}, \quad \text{with} \quad \mathsf{P}^{\mathcal{P}} = S(\mathbf{U}^{\mathrm{D}})(\varrho^{\mathcal{P}})^{\gamma}. \end{cases} \tag{4.12}$$

*Subsonic outflow boundary.* Assume that $\lambda_1(\mathbf{U}, \boldsymbol{n}) < 0 \le \lambda_2(\mathbf{U}, \boldsymbol{n}) \le \lambda_3(\mathbf{U}, \boldsymbol{n})$. Then, $\mathbf{U}^{\mathcal{P}}$ is obtained by solving the system

$$C_1(\mathbf{U}^{\mathcal{P}}) = C_1(\mathbf{U}^{\mathrm{D}}), \qquad (\mathbf{V}^{\mathcal{P}})^{\perp} = \mathbf{V}^{\perp}, \qquad S(\mathbf{U}^{\mathcal{P}}) = S(\mathbf{U}), \qquad C_3(\mathbf{U}^{\mathcal{P}}) = C_3(\mathbf{U}). \tag{4.13}$$

Notice that, as expected, only one Dirichlet condition is enforced. This gives $\mathsf{V}_n^{\mathcal{P}} = \frac{1}{2}(C_1(\mathbf{U}^{\mathrm{D}}) + C_3(\mathbf{U}))$, $\mathsf{P}^{\mathcal{P}} = S(\mathbf{U})(\varrho^{\mathcal{P}})^{\gamma}$, and

$$a^{\mathcal{P}} = \frac{\gamma-1}{4}(C_3(\mathbf{U}) - C_1(\mathbf{U}^{\mathrm{D}})) = \frac{\gamma-1}{4}\mathsf{V}_n + \frac{a}{2} - \frac{\gamma-1}{4}\mathsf{V}_n^{\mathrm{D}} + \frac{a^{\mathrm{D}}}{2}.$$

Here again we have $0 < a^{\mathcal{P}}$ if $\gamma \le 3$ and if the admissibility condition $\frac{\gamma-1}{2}\mathsf{V}_n^{\mathrm{D}} \le a^{\mathrm{D}}$ holds true.

Using $(a^{\mathcal{P}})^2 = \gamma \mathsf{P}^{\mathcal{P}}(\varrho^{\mathcal{P}})^{-1}$ with $\mathsf{P}^{\mathcal{P}} = S(\mathbf{U})(\varrho^{\mathcal{P}})^{\gamma}$, the post-processing consists of setting:

$$\begin{cases} \rho^{\mathcal{P}} = \left(\dfrac{1}{\gamma S(\mathbf{U})}\left(\dfrac{\gamma-1}{4}\big(C_3(\mathbf{U}) - C_1(\mathbf{U}^{\mathrm{D}})\big)\right)^2\right)^{\frac{1}{\gamma-1}}, \\ \mathbf{M}^{\mathcal{P}} = \rho^{\mathcal{P}}\left(\mathbf{V}^{\perp} + \mathsf{V}_n^{\mathcal{P}}\boldsymbol{n}\right), \quad \text{with} \quad \mathsf{V}_n^{\mathcal{P}} = \dfrac{1}{2}\big(C_1(\mathbf{U}^{\mathrm{D}} + C_3(\mathbf{U}))\big), \\ E^{\mathcal{P}} = \dfrac{1}{\gamma-1}\mathsf{P}^{\mathcal{P}} + \dfrac{1}{2}\dfrac{\|\mathbf{M}^{\mathcal{P}}\|_{\ell^2}^2}{\varrho^{\mathcal{P}}}, \quad \text{with} \quad \mathsf{P}^{\mathcal{P}} = S(\mathbf{U})(\varrho^{\mathcal{P}})^{\gamma}. \end{cases} \tag{4.14}$$

*Supersonic outflow boundary condition.* Assume that $0 \le \lambda_1(\mathbf{U}, \boldsymbol{n}) \le \lambda_2(\mathbf{U}, \boldsymbol{n}) \le \lambda_3(\mathbf{U}, \boldsymbol{n})$. Since all the characteristics exit the domain, the post-processing consists of not doing anything:

$$\mathbf{U}^{\mathcal{P}} = \mathbf{U}. \tag{4.15}$$

**Remark 4.1** (*Literature*)**.** It is established in [37, §3] that appropriate non-reflecting boundary conditions for the one-dimensional Riemann problem $\partial_t \boldsymbol{v} + \partial_n(\mathbb{f}(\boldsymbol{v})\boldsymbol{n}) = \mathbf{0}$ are $\partial_t C_1(\mathbf{U}) + \frac{a}{\gamma-1}\partial_t s(\mathbf{U}) = 0$ in the subsonic outflow situation and $\partial_t C_1(\mathbf{U}) = 0$ plus $\partial_t s(\mathbf{U}) = 0$ (and $\partial_t \mathsf{V}^{\perp} = \mathbf{0}$) in the subsonic inflow situation, where $s(\mathbf{U}) = \log(e(\mathbf{U})^{\frac{1}{\gamma-1}}\varrho^{-1})$ is the specific entropy. Assuming that the Dirichlet data are time-independent, these conditions can be rewritten $\partial_t(C_3(\mathbf{U}) - C_3(\mathbf{U}^{\mathrm{D}})) + \frac{a}{\gamma-1}\partial_t(s(\mathbf{U}) - s(\mathbf{U}^{\mathrm{D}})) = 0$ and so on. Then, under the assumption that the flow is locally isentropic at the boundary, these conditions exactly coincide with what is proposed above (notice that in this case $\mathsf{S} = (\gamma-1)e^{s}$). Indeed, by setting $C_1(\mathbf{U})_{|t=0} = C_1(\mathbf{U}^{\mathrm{D}})$, $S(\mathbf{U})_{|t=0} = S(\mathbf{U}^{\mathrm{D}})$, (and $\mathsf{V}^{\perp}(\mathbf{U})_{|t=0} = \mathsf{V}^{\perp}(\mathbf{U}^{\mathrm{D}})$), the condition $\partial_t C_1(\mathbf{U}) = 0$ yields $C_3(\mathbf{U}^{\mathcal{P}}) = C_3(\mathbf{U}^{\mathrm{D}})$ for the subsonic outflow situation (see (4.14)) and it yields $C_1(\mathbf{U}^{\mathcal{P}}) = C_1(\mathbf{U}^{\mathrm{D}})$, $\mathsf{S}(\mathbf{U}^{\mathcal{P}}) = \mathsf{S}(\mathbf{U}^{\mathrm{D}})$, (and $\mathsf{V}^{\perp}(\mathbf{U}^{\mathcal{P}})_{|t=0} = \mathsf{V}^{\perp}(\mathbf{U}^{\mathrm{D}})$) for the subsonic inflow situation (see (4.12)).

No claim is made here about the optimality of the proposed artificial boundary conditions, in particular in regards to their absorbing properties. We refer the reader to Fosso et al. [38] and the abundant literature cited therein for other approaches used in the finite difference context.

### 4.3.3. Conservation and admissibility

We collect in this subsection conservation and admissibility properties of the post-processing method proposed above.

**Lemma 4.2** (*Slip Condition*)**.** *Let $i \in \mathcal{V}^{\partial}_{\mathrm{s}}$, let $\mathbf{U}_i \in \mathcal{A}$, and let $\mathbf{U}^{\mathcal{P}}_i$ as defined in (4.3).*

(i) *Then $\mathbf{U}^{\mathcal{P}}_i$ is also admissible, meaning $\mathbf{U}^{\mathcal{P}}_i \in \mathcal{A}$.*
(ii) *Assume also that the equation of state derives from an entropy $s$. Then $s(\mathbf{U}^{\mathcal{P}}_i) \geq s(\mathbf{U}_i)$.*
(iii) *For all $i \in \mathcal{V}^{\partial}_{\mathrm{s}} \backslash \mathcal{V}^{\partial}_{\mathrm{nr}}$, the mass flux and the total energy flux of the postprocessed solution at $i$ is zero (i.e., $\rho(\mathbb{f}(\mathbf{U}^{\mathcal{P}}_i)\boldsymbol{n}_i) = 0$ and $E(\mathbb{f}(\mathbf{U}^{\mathcal{P}}_i)\boldsymbol{n}_i) = 0$).*

**Proof.** See Lemma A.2 in Appendix A.

**Lemma 4.3** (*Non-reflecting Condition*)**.** *Let $i \in \mathcal{V}^{\partial}_{\mathrm{nr}}$ and let $\mathbf{U}_i \in \mathcal{A}$. Let $\mathbf{U}^{\mathcal{P}}_i$ be defined either by (4.4) or by one the conditions (4.9), (4.12), (4.14), (4.15) (with the $\gamma$-law assumption, $\gamma \in (1,3]$, and the admissibility condition on the Dirichlet data $\frac{\gamma-1}{2}\mathsf{V}^{\mathrm{D}} \leq a^{\mathrm{D}}$). Then $\mathbf{U}^{\mathcal{P}}_i \in \mathcal{A}$.*

**Proof.** Direct consequence of the definitions (4.9), (4.12), (4.14), (4.15).

We obtain the following result by combining Lemmas 4.2 and 4.3.

**Corollary 4.4** (*Admissibility*)**.** *The solution obtained at the end the RKSSP(3,3) algorithm after limiting and post-processing is admissible.*

**Lemma 4.5** (*Global Conservation*)**.** *Assume that $\mathcal{V}_{\mathrm{s}}{}^{\partial} = \mathcal{V}^{\partial}$ and $\mathbf{U}^n$ satisfies the slip boundary condition (i.e., $\mathbf{M}^n_i \cdot \boldsymbol{n}_i = 0$ for all $i \in \mathcal{V}^{\partial}$). Then the solution obtained at the end the RKSSP(3,3) algorithm after limiting and post-processing, say $\mathbf{U}^{n+1}$, satisfies $\sum_{j\in\mathcal{V}} m_j \varrho^{n+1}_j = \sum_{j\in\mathcal{V}} m_j \varrho_j{}^n$ and $\sum_{j\in\mathcal{V}} m_j \mathsf{E}^{n+1}_j = \sum_{j\in\mathcal{V}} m_j \mathsf{E}_j{}^n$.*

**Proof.** See Lemma A.3 in Appendix A.

## 5. Discretization of the parabolic problem

We describe in this section key details involved in the approximation of the parabolic operator $S_{\mathrm{P}}$ (see (2.8)). Given an admissible field $\boldsymbol{u}^n_h = \sum_{i\in\mathcal{V}} \mathbf{U}^n_i \varphi_i$ at some time $t^n$ and given some time step size $\tau_n$, we want to construct an approximation of the solution to (2.8) at $t^{n+1} := t^n + \tau_n$, say $\boldsymbol{u}^{n+1}_h = \sum_{i\in\mathcal{V}} \mathbf{U}^{n+1}_i \varphi_i$. Referring to (2.9), we recall that the time step used in the parabolic problem is twice that used in the hyperbolic step, but to simplify the notation we still call the time step size $\tau_n$ in this entire section. The important point here is that positivity of the internal energy and conservation must be guaranteed. The steps described in Sections 5.1–5.2 are summarized in Algorithm 2 in Appendix C.

### 5.1. Density, velocity, and momentum update

Recalling that in (2.8) the density does not change in time, we set

$$\varrho_i^{n+1} = \varrho_i^n, \qquad \forall i \in \mathcal{V}. \tag{5.1}$$

We use the Crank–Nicolson technique for the time stepping in (2.8b). The approximation in space is done by using the Galerkin technique with the lumped mass matrix. Let $\{\boldsymbol{e}_k\}_{k\in\{1:d\}}$ denote the canonical Cartesian basis of $\mathbb{R}^d$, and for all $\mathbf{X} \in \mathbb{R}^d$, let $\{\mathsf{X}_k\}_{k\in\{1:d\}}$ denote the Cartesian coordinates of $\mathbf{X}$. The discrete problem then consists of seeking $\boldsymbol{v}_h^{n+\frac{1}{2}} := \sum_{i\in\mathcal{V}} \mathsf{V}_{i,k}^{n+\frac{1}{2}} \varphi_i \boldsymbol{e}_k$ so that

$$\varrho_i^n m_i \mathsf{V}_{i,k}^{n+\frac{1}{2}} + \tfrac{1}{2}\tau_n a(\boldsymbol{v}_h^{n+\frac{1}{2}}, \varphi_i \boldsymbol{e}_k) = m_i \mathsf{M}_{i,k}^n + \tfrac{1}{2}\tau_n m_i \mathsf{F}_{i,k}^{n+\frac{1}{2}}, \qquad \forall i \in \mathcal{V},\ \forall k \in \{1:d\}, \tag{5.2}$$

with $a(\boldsymbol{v}, \boldsymbol{w}) := \int_D \mathbb{s}(\boldsymbol{v}){:}\mathbb{e}(\boldsymbol{w})\,\mathrm{d}x$, and $\mathsf{F}_{i,k}^{n+\frac{1}{2}} := \int_D \varphi_i(\boldsymbol{x})\boldsymbol{e}_k{\cdot}\boldsymbol{f}(\boldsymbol{x}, t^{n+\frac{1}{2}})\,\mathrm{d}x$. We consider three types of boundary conditions on the velocity: (i) the no-slip condition $\boldsymbol{v}_{|\partial D} = \boldsymbol{0}$; (ii) the slip condition $\boldsymbol{v}{\cdot}\boldsymbol{n} = 0$ and $\boldsymbol{n}{\times}(\mathbb{s}(\boldsymbol{v})\boldsymbol{n}) = \boldsymbol{0}$; (iii) and the homogeneous Neumann boundary condition $\mathbb{s}(\boldsymbol{v})\boldsymbol{n}_{|\partial D} = \boldsymbol{0}$. The assembling is done in two steps: (1) one first assembles the system with homogeneous Neumann boundary conditions; (2) the correct boundary conditions are implemented by post-processing the linear system. After the essential boundary conditions are enforced, and once this linear system is solved (see Section 5.3), the velocity and the momentum are updated as follows:

$$\mathbf{V}_i^{n+1} := 2\mathbf{V}_i^{n+\frac{1}{2}} - \mathbf{V}_i^n, \qquad \mathbf{M}_i^{n+1} := \varrho_i^{n+1}\mathbf{V}_i^{n+1}, \qquad \forall i \in \mathcal{V}. \tag{5.3}$$

### 5.2. Internal energy and total energy update

The second step of the parabolic solve consists of computing the specific internal energy (or temperature) and updating the total energy. Before going into the details, we discuss the boundary condition. We consider two types of boundary conditions: (i) Dirichlet: $T_{|\partial D} = T^{\partial}$; (ii) homogeneous Neumann: $\partial_n T = 0$. The assembling is done in two steps: (1) one first assembles the system with homogeneous Neumann boundary conditions; (2) the Dirichlet boundary conditions are implemented by post-processing the linear system.

Here again we use the Crank–Nicolson technique for the time stepping in (2.8c), and the approximation in space is done by using the Galerkin technique with the lumped mass matrix. First, we compute the rate of specific internal energy production caused by the viscous stress

$$\mathsf{K}_i^{n+\frac{1}{2}} := \frac{1}{m_i}\int_D \mathbb{s}(\boldsymbol{v}_h^{n+\frac{1}{2}}){:}\mathbb{e}(\boldsymbol{v}_h^{n+\frac{1}{2}})\varphi_i\,\mathrm{d}x, \qquad \forall i \in \mathcal{V}. \tag{5.4}$$

Second, we compute the specific internal energy of $\boldsymbol{u}_h^n$ by setting set $\mathsf{e}_i^n := (\varrho_i^n)^{-1}\mathsf{E}_i^n - \frac{1}{2}\|\mathbf{V}_i^n\|_{\ell^2}^2$ for all $i \in \mathcal{V}$. Note that $\mathsf{e}_i^n > 0$ since we assumed that $\boldsymbol{u}_h^n$ is admissible. Then, we seek the specific internal energy, $e_h^{n+\frac{1}{2}} = \sum_{i\in\mathcal{V}} \mathsf{e}_i^{n+\frac{1}{2}}\varphi_i$, such that the following holds:

$$m_i\varrho_i^n(\mathsf{e}_i^{n+\frac{1}{2}} - \mathsf{e}_i^n) + \tfrac{1}{2}\tau_n b(e_h^{n+\frac{1}{2}}, \varphi_i) = \tfrac{1}{2}\tau_n m_i \mathsf{K}_i^{n+\frac{1}{2}}, \qquad \forall i \in \mathcal{V}, \tag{5.5}$$

with $b(e, w) := c_v^{-1}\kappa\int_D \nabla e{\cdot}\nabla w\,\mathrm{d}x$. Finally, we update the internal energy and the total energy:

$$\mathsf{e}_i^{n+1} = 2\mathsf{e}_i^{n+\frac{1}{2}} - \mathsf{e}_i^n, \qquad \mathsf{E}_i^{n+1} = \varrho_i^{n+1}\mathsf{e}_i^{n+1} + \tfrac{1}{2}\varrho_i^n\|\mathbf{V}_i^{n+1}\|_{\ell^2}^2, \qquad \forall i \in \mathcal{V}. \tag{5.6}$$

The algorithm is second-order accurate in time, but there is no guarantee that the internal energy stays positive because the Crank–Nicolson scheme is not positivity preserving. If it happens that $\min_{i\in\mathcal{V}} \mathsf{e}_i^{n+1} < 0$, then limiting must be applied. This is done as described in [1, §5.3]. We briefly recall the technique. We compute a first-order, invariant-domain-preserving (i.e., positive) update of the internal energy by seeking $e_h^{\mathrm{L},n+1} = \sum_{i\in\mathcal{V}} \mathsf{e}^{\mathrm{L},n+1}\varphi_i$ so that the following holds:

$$m_i\varrho_i^n(\mathsf{e}_i^{\mathrm{L},n+1} - \mathsf{e}_i^n) + \tau_n b(e_h^{\mathrm{L},n+1}, \varphi_i) = \tfrac{1}{2}\tau_n m_i \mathsf{K}_i^{n+\frac{1}{2}}, \qquad \forall i \in \mathcal{V}. \tag{5.7}$$

Let $e_h^{\mathrm{H},n+1} := \sum_{i\in\mathcal{V}} \mathsf{e}^{\mathrm{H},n+1}\varphi_i$ be the solution to (5.5). Subtracting (5.7) from (5.5) yields

$$m_i\varrho_i^n(\mathsf{e}_i^{\mathrm{H},n+1} - \mathsf{e}_i^{\mathrm{L},n+1}) = \sum_{j\in\mathcal{I}^*(i)} A_{ij}, \tag{5.8}$$

$$A_{ij} := -\tfrac{1}{2}\tau_n b(\varphi_j,\varphi_i)(\mathsf{e}_j^{\mathrm{H},n+1} + \mathsf{e}_j^n - 2\mathsf{e}_j^{\mathrm{L},n+1} - \mathsf{e}_i^{\mathrm{H},n+1} - \mathsf{e}_i^n + 2\mathsf{e}_i^{\mathrm{L},n+1}). \tag{5.9}$$

The standard FCT limiting can be applied by setting $m_i\varrho_i^n(\mathsf{e}_i^{\mathrm{H},n+1} - \mathsf{e}_i^{\mathrm{L},n+1}) = \sum_{j\in\mathcal{I}^*(i)} \ell_{ij}A_{ij}$, see e.g., [33–35]. The reader is referred to [1, §5.3] for the computation of $\ell_{ij}$. Once the internal energy is updated, the total energy can also be updated by setting

$$\mathsf{E}_i^{n+1} = \varrho_i^{n+1}\mathsf{e}_i^{n+1} + \tfrac{1}{2}\varrho_i^n\|\mathbf{V}_i^{n+1}\|_{\ell^2}^2, \qquad \forall i\in\mathcal{V}. \tag{5.10}$$

The main properties of the method presented here are collected in the following statement.

**Lemma 5.1** (*Positivity and Conservation*)**.** *Let $\mathbf{U}^n$ be an admissible state. Let $\mathbf{U}^{n+1}$ be the state constructed in the parabolic substep. Then, $\mathbf{U}^{n+1}$ is an admissible state, i.e., $\mathbf{U}_i^{n+1} \in \mathcal{A}$ for all $i\in\mathcal{V}$ and all $\tau_n$, and the following holds for all $i\in\mathcal{V}$ and all $\tau_n$:*

$$\varrho_i^{n+1} = \varrho_i^n > 0, \qquad \min_{j\in\mathcal{V}} \mathsf{e}_j^{n+1} \ge \min_{j\in\mathcal{V}} \mathsf{e}_j^n > 0, \qquad \forall i\in\mathcal{V}, \tag{5.11}$$

*Assume that the slip or the no-slip boundary condition is enforced on the velocity everywhere on $\partial D$. Assume that the homogeneous Neumann boundary condition is enforced on the internal energy everywhere on $\partial D$. Then the following holds true for all $\tau_n$:*

$$\sum_{i\in\mathcal{V}} m_i\mathsf{E}_i^{n+1} = \sum_{i\in\mathcal{V}} m_i\mathsf{E}_i^n + \sum_{i\in\mathcal{V}} \tau m_i\mathbf{F}_i^{n+\frac{1}{2}}\cdot\mathbf{V}_i^{n+\frac{1}{2}}. \tag{5.12}$$

**Proof.** See [1, Thm. 5.5].

## 5.3. Matrix-free geometric multigrid

For the `deal.II`-based finite element implementation discussed in Section 6 (see [13]), the linear systems (5.2) and (5.5) are solved iteratively using matrix-free operator evaluations [39]. The *action* of the matrix on a vector is implemented by redundantly computing the information contained in the stencil on the fly through the finite element integrals. On modern hardware, computations are less expensive than data movement [24,40], making a matrix-free evaluation several times faster than a sparse-matrix vector product due to the reduced memory traffic. This is especially relevant for the vector-valued velocity: the block-structured matrix–vector multiplication couples the velocity components, whereas the cell-wise integrals in the matrix-free evaluation only couple these components at the quadrature-point level without additional data transfer. Besides yielding faster matrix–vector products, the matrix-free approach also avoids the cost of matrix assembly. The higher arithmetic intensity in the matrix-free evaluation is leveraged by cross-element SIMD vectorization of the relevant operations [39,41].

Regarding the selection of the iterative solvers, we choose among two options. When the cell-based Reynolds number using the diameter of the cells as length scale is above one, the mass matrix contribution in (2.8b) dominates due to the limit imposed by the hyperbolic problem on the time step. This argument also holds for the internal energy equation provided the Prandtl number is not too large. In that case, the diagonal mass matrix is an optimal preconditioner for the conjugate gradient algorithm, giving iteration counts below 10. If the mesh becomes very fine for a given viscosity level, the elliptic contributions become dominant instead. Then, we equip the conjugate gradient solver with a geometric multigrid preconditioner that steps into successively coarser levels $l \in \{1{:}L\}$ where $l = 1$ refers to the coarsest mesh and $l = L$ refers to the finest mesh, see Kronbichler and Wall [24], Clevenger et al. [42] for details on the parallel scaling and performance. A Chebyshev iteration of degree three (i.e., three matrix–vector products) around the point-Jacobi method is used for pre- and post-smoothing, using parameters to smoothen in a range $[0.08\hat{\lambda}_{\max}, 1.2\hat{\lambda}_{\max}]$ with the maximal eigenvalue estimate $\hat{\lambda}_{\max}$ computed every four time steps by a Lanczos iteration with 12 iterations. In order to improve parallel scaling of the V-cycle, we limit the coarsening to the mesh levels $l \in \{L_{\min}{:}L\}$, with $L_{\min}$ such that the cell-based Reynolds number exceeds unity. On the coarse level, a few iterations suffice to solve the system accurately, which is done by a Chebyshev iteration aiming to reduce the residual by a factor of $10^3$ according to the a-priori error estimate. The multigrid solver typically takes 3 to 5 iterations to converge. To increase the throughput, the multigrid V-cycle is run in single precision [43,44].

**Table 1**
Convergence study of the approximation of the Becker solution [45] on a mesh sequence $\mathcal{H}$ of successively refined uniform meshes with CFL = 0.3. The columns labeled "$\delta_p$" show the $L^p$-norm of the consolidated error [1, Eq. (7.4)].

| gridpoints | $\delta_1$ | rate | $\delta_2$ | rate | $\delta_\infty$ | rate |
|---|---|---|---|---|---|---|
| 4225 | $4.68 \times 10^{-3}$ | – | $5.96 \times 10^{-3}$ | – | $1.29 \times 10^{-2}$ | – |
| 16641 | $3.88 \times 10^{-4}$ | 3.59 | $9.34 \times 10^{-5}$ | 2.68 | $2.89 \times 10^{-3}$ | 2.16 |
| 66049 | $8.73 \times 10^{-5}$ | 2.15 | $2.35 \times 10^{-5}$ | 1.99 | $8.02 \times 10^{-4}$ | 1.85 |
| 263169 | $2.17 \times 10^{-5}$ | 2.01 | $5.90 \times 10^{-5}$ | 1.99 | $2.10 \times 10^{-4}$ | 1.93 |

## 6. Verification, validation and benchmarks

The method described above has been implemented in a code called `ryujin` which is freely available online[4] [13] under a permissible open source license.[5] This code is based on the finite element library `deal.II` [11,12] and uses mapped continuous $\mathbb{Q}_1$ finite elements. We discuss in this section a number of verification and benchmark configurations to demonstrate that the algorithm described herein is robust, accurate and scalable. In particular, we use a 2D shocktube configuration proposed by Daru and Tenaud [9], Daru and Tenaud [10] to demonstrate grid convergence; see Section 6.3. Following [14] and to allow for rigorous quantitative comparisons with other research codes, test vectors obtained from our computation of extrema of the skin friction coefficient are made freely available [15]. We then demonstrate that the method can reliably predict pressure coefficients on the well-studied supercritical airfoil Onera OAT15a [16] in the supercritical regime at Mach 0.73 in three-dimensions; see Section 6.4. Finally, a series of synthetic benchmarks are presented to assess the performance of the compute-kernel and the strong and weak scalability of our implementation (Section 6.5).

### 6.1. *Verification*

The hyperbolic kernel and the parabolic kernel of `ryujin` have been verified on various analytical solutions to ensure the correctness of the implementation; see e.g., [23,36]. We now demonstrate the correctness of the full algorithm in 2D on a viscous shockwave problem that has an exact solution described in Becker [45], see also [1, §7.2] and Johnson [46]. With the parameters given in [1, §7.2; Eqs. (7.1)–(7.4)], we approximate the Becker solution on a mesh sequence $\mathcal{H}$ of successively refined uniform meshes. The computational domain is the unit square with Dirichlet boundary conditions on the left and right boundaries, and periodic boundary conditions on the upper and lower boundaries. We slightly deviate from [1] by choosing the velocity of the Galilean frame to be $v_\infty = 0.125$ and choosing the CFL number 0.3. The source code and the parameter files are archived on the online platform *Zenodo*; see [13,15]. As evidenced in Table 1, we observe second-order convergence in space and time in the maximum norm.

### 6.2. *Non-reflecting conditions*

We now evaluate the performance of the non-reflecting boundary conditions described in Section 4.3 by using a series of tests proposed in Fosso et al. [38]. We solve the Euler equations in the domain $D := [-1, 1]^2$ with the initial data

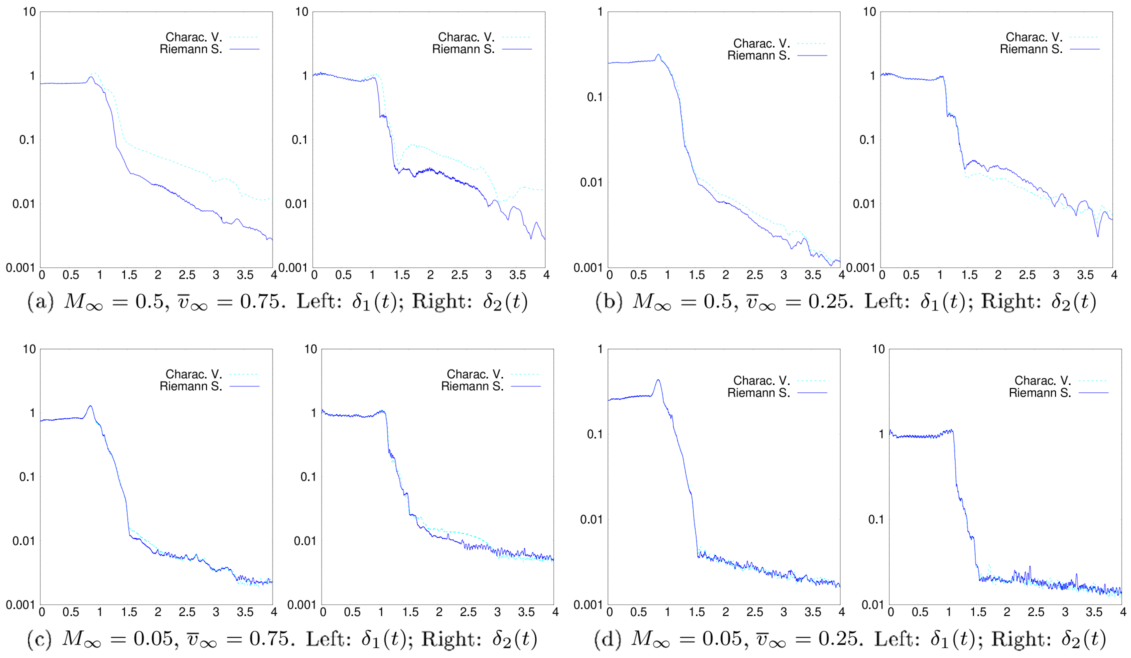
$$\rho_0(\boldsymbol{x}) = \rho_\infty, \tag{6.1}$$

$$\boldsymbol{v}_0(\boldsymbol{x}) = \boldsymbol{v}_\infty + \overline{v}_\infty r_0^{-1} \psi(\|\boldsymbol{x} - \boldsymbol{x}_0\|_{\ell^2}) \mathbb{A}(\boldsymbol{x} - \boldsymbol{x}_0), \tag{6.2}$$

$$p_0(\boldsymbol{x}) = p_\infty - \tfrac{1}{2}\rho_\infty \overline{v}_\infty^2 \psi^2(\|\boldsymbol{x} - \boldsymbol{x}_0\|_{\ell^2}), \tag{6.3}$$

with $\boldsymbol{v}_\infty = (v_\infty, 0)^\mathsf{T}$, $\psi(r) := e^{\frac{1}{2}(1-\frac{r^2}{r_0^2})}$ and $\mathbb{A} := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$. We take $v_\infty = 1$ and $\rho_\infty = 1$. We use the velocity perturbation $\overline{v}_\infty$ and the Mach number $M_\infty$ as parameters. The pressure $p_\infty$ is defined to be $\frac{\rho_\infty}{\gamma} a_\infty^2$ where $a_\infty := \frac{v_\infty}{M_\infty}$

[4] https://github.com/conservation-laws/ryujin.
[5] https://spdx.org/licenses/MIT.html.



(a) $M_\infty = 0.5$, $\overline{v}_\infty = 0.75$. Left: $\delta_1(t)$; Right: $\delta_2(t)$ (b) $M_\infty = 0.5$, $\overline{v}_\infty = 0.25$. Left: $\delta_1(t)$; Right: $\delta_2(t)$

(c) $M_\infty = 0.05$, $\overline{v}_\infty = 0.75$. Left: $\delta_1(t)$; Right: $\delta_2(t)$ (d) $M_\infty = 0.05$, $\overline{v}_\infty = 0.25$. Left: $\delta_1(t)$; Right: $\delta_2(t)$

**Fig. 1.** Tests on the non-reflecting boundary conditions.

is the sound speed. Four cases are considered: (i) $M_\infty = 0.5$, $\overline{v}_\infty = 0.75$; (ii) $M_\infty = 0.5$, $\overline{v}_\infty = 0.25$; (iii) $M_\infty = 0.05$, $\overline{v}_\infty = 0.75$; (iv) $M_\infty = 0.05$, $\overline{v}_\infty = 0.25$. The simulations are done on a $80\times 80$ mesh. We enforce the non-reflecting boundary condition on the four sides of the domain. We test the Riemann solution technique described in Section 4.3.1 and the method based on the characteristic variables described in Section 4.3.2. All the tests are done with CFL $= 0.75$. We show in Fig. 1 the quantities $\delta_1(t) := \frac{\|\boldsymbol{v}(\cdot,t)-\boldsymbol{v}_\infty\|_{L^\infty(D)}}{\|\boldsymbol{v}_\infty\|_{L^\infty(D)}}$ and $\delta_2(t) := \frac{\|\nabla\times\boldsymbol{v}(\cdot,t)\|_{L^\infty(D)}}{\|\nabla\times\boldsymbol{v}_0\|_{L^\infty(D)}}$ as functions of time over the time interval $[0, 4]$. The label “Riemann S.” refers to the method from Section 4.3.1 and the label “Charac. V.” refers to the method from Section 4.3.2. Since the center of the vortex crosses the outflow boundary at $t = 1$, the faster $\delta_1$ and $\delta_2$ go to zero as $t$ grows, the better the non-reflecting properties of the boundary condition are. We observe that the method using the exact solution to a Riemann problem is slightly more efficient than that using the characteristics variables when the velocity perturbation is large ($M_\infty = 0.5$, $\overline{v}_\infty = 0.75$). The method using the characteristic variables performs as well as the method using the Riemann solution in the cases (ii)–(iii)–(iv). Overall the Riemann solution method has properties similar to the method labeled “OC2” in Fosso et al. [38].

## 6.3. 2D shocktube benchmark

We now illustrate the accuracy of the proposed algorithm by testing it against a challenging two-dimensional benchmark problem introduced in the literature by Daru and Tenaud [9], Daru and Tenaud [10]. The configuration is a shocktube problem in a square cavity $D := (0, 1)^2$ where a shock interacts with a viscous boundary layer. A lambda shock is formed as a result of this interaction; see Fig. 2(a). The fluid is initially at rest at $t = 0$. There are two different states separated by a diaphragm at $\{x = \frac{1}{2}\}$. The density, velocity and pressure on the left-hand side of the diaphragm are $\rho_L = 120$, $v_L = 0$, $p_L = \rho_L/\gamma$. The states on the right-hand side are $\rho_R = 1.2$, $v_R = 0$, $p_R = \rho_R/\gamma$. A shock, a contact discontinuity and an expansion wave are created after the diaphragm is broken. The viscous shock and the contact wave move to the right, whereas the expansion wave moves to the left. A thin viscous boundary layer is created on the bottom wall of the cavity as the shock and the contact waves progress toward the right wall. The shock reaches the right wall at approximately $t \approx 0.2$, is reflected, and moves back to the left thereafter. The contact discontinuity remains stationary close to the right wall after it interacted with the

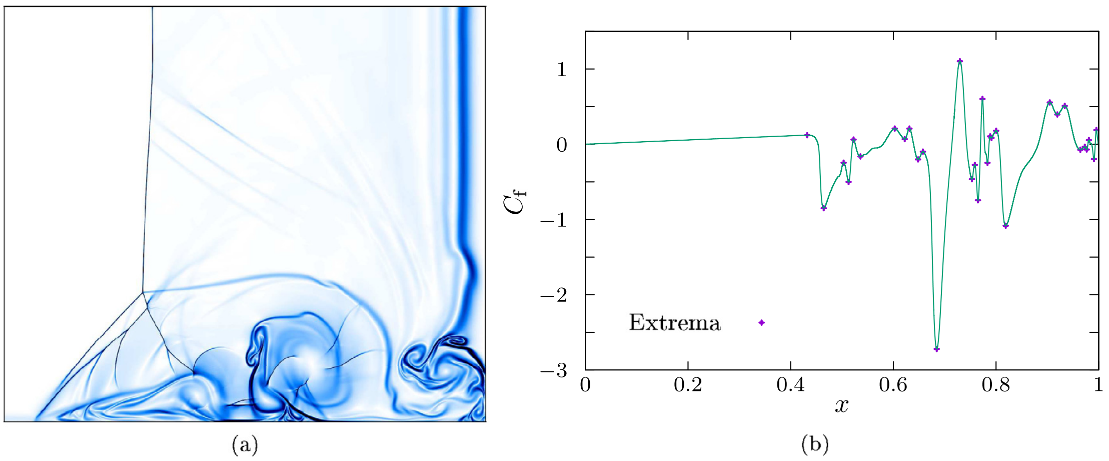


**Fig. 2.** The 2D shocktube benchmark: (a) zoom ($[0.5, 1] \times [0, 0.5]$) of a schlieren plot at $t = 1.00$ showing the interactions of the reflected shock wave with the viscous boundary layer. (b) postprocessed skin friction coefficient $C_f$ at time $t = 1.00$. The continuous green line is the numerical result with the finest level L14. The extrema are computed by means of extrapolation.

shock. A lambda shock is created as the shock interacts with the viscous boundary layer. The various mechanisms at play in this problem are described in [9, §6] and [10, §5 & §6].

The equation of state is $p = (\gamma - 1)\rho e$ with $\gamma = 1.4$. The dynamics viscosity is $\mu = 10^{-3}$ and the bulk viscosity is set to 0. The Prandtl number is $Pr = \frac{\mu c_p}{\kappa} = 0.73$, where $c_p = \frac{\gamma}{\gamma-1}$. The no-slip boundary condition is enforced on the velocity. The homogeneous Neumann boundary condition is enforced on the internal energy, i.e., the tube is thermally insulated. Due to symmetry, the computation is done in the half domain $(0, 1)\times(0, \frac{1}{2})$, and the boundary conditions $\boldsymbol{v}\cdot\boldsymbol{n} = 0$, $(\mathbb{e}(\boldsymbol{u})\boldsymbol{n})\cdot\boldsymbol{\tau} = 0$ are enforced at $\{y = \frac{1}{2}\}$, where $\boldsymbol{\tau}$ is one of the two tangent unit vectors along the boundary $\{y = \frac{1}{2}\}$. Denoting the vertical component of the velocity by $v_y$, the above boundary condition amounts to enforcing $v_y = 0$ and $\partial_y v_y = 0$.

As in [9,10], we compute the skin friction coefficient on the bottom wall $[0, 1]\times\{0\}$. We estimate this quantity for all the degrees of freedom $i \in \mathcal{V}^{\partial}_{\mathrm{s}}$ by using the following expression:

$$C_f(\boldsymbol{x}_i) := \frac{1}{\frac{1}{2}\rho_\infty \|\boldsymbol{v}_\infty\|^2_{l^2}} \left( \frac{1}{\tilde{m}^{\partial}_i} \int_{\partial D_{\mathrm{s}}} \varphi_i \, \boldsymbol{\tau}\cdot(\mathbb{s}(\boldsymbol{v}_h)\boldsymbol{n})\,\mathrm{d}s \right)_{\nu}, \qquad \tilde{m}^{\partial}_i := \int_{\partial D_{\mathrm{s}}} \varphi_i \,\mathrm{d}s. \tag{6.4}$$

Our goal is to observe the convergence of this quantity as the meshes are refined. To this end, we compute the local maxima and minima of $C_f$ for a mesh sequence composed of successively refined structured quadrilateral meshes ranging from refinement level L 11 (8 M grid points) to L 14 (512 M grid points). The meshes are graded by transforming a uniform mesh in $y$-direction subject to the mapping $y \mapsto 1 - (1 - y)^{1/5}$. The results for all these grids are shown in Fig. 6 in Appendix D. Convergence is observed as the mesh size goes to zero. The results obtained on the finest mesh level L 14 are shown in Fig. 2(b). The values of the local minima and local maxima of $C_f$ obtained on the grids L 11, L 12, L 13, and L 14 are reported in Table 2 and identified with symbols in Fig. 6 in Appendix D. We also show in this table the extrapolated values of $C_f$ that have been obtained by using the open source software `gnuplot` (see http://www.gnuplot.info/). This is done by fitting the four values of $C_f$ obtained on the grids L 11, L 12, L 13, and L 14 with the linear function $a + b\,h$ using the nonlinear Levenberg–Marquardt least squares technique. The 12th and 13th extrema (global maximum and minimum) have been extrapolated with convergence orders that are slightly higher, i.e., $a + b\,h^{1.5}$ and $a + b\,h^{1.2}$, respectively. We report the fit value $a$ and the asymptotic standard error in the column labeled "extrapolated" in Table 2 in Appendix D. The results suggest that the computation is indeed in the asymptotic regime with the normalized asymptotic standard error $\frac{\Delta C_f}{\Delta C_{f,\mathrm{norm}}}$ smaller than 0.2% where $\Delta C_{f,\mathrm{norm}} := \max C_f(\text{extrap.}) - \min C_f(\text{extrap.})$. We also give in this table the values of $C_f$ reported in [10,14]. These results have been obtained with a 7th-order method called OSMP7 on a 2000×4000 uniform grid. The relative deviation reported in the last column of Table 2 is defined to be $\frac{C_f(\text{OSMP7})-C_f(\text{extrap.})}{\Delta C_{f,\mathrm{norm}}}$. We report generally good agreement with the OSMP7 results with a relative deviation that is generally less below

1%. There are some noticeable differences though on the 13th, 15th, and 17th extrema where the deviations are −2.77%, 2.86%, and 3.99%, respectively. We conjecture however that the results reported here are probably slightly more accurate than those from [10] since our finest grid is significantly finer that those used therein. Following the initiative of [14] and to facilitate rigorous quantitative comparisons with other research codes, we provide more detailed test vectors of the skin friction coefficient at [15]. We also reiterate the suggestion made in [10] that this problem is an interesting benchmark that should be used more systematically in the literature to test compressible Navier–Stokes codes.

## 6.4. Onera OAT15a airfoil

The Onera OAT15a airfoil has been extensively studied in the literature. Experimental results for the supercritical flow regime at Mach 0.73 and at Reynolds number $3\times10^6$ have been published in Jacquin et al. [47], Jacquin et al. [48]. Various numerical simulations of this configuration are also available in the literature, see e.g., Deck [16], Deck and Renard [17], Nguyen et al. [18]. We are interested in predicting the pressure coefficient $C_\mathrm{p}$

$$C_\mathrm{p}(\boldsymbol{x}) := \frac{\langle p(\boldsymbol{x},t)\rangle_t - p_\infty}{\frac{1}{2}\rho_\infty v_\infty^2},$$

at Mach 0.73 with an angle of attack of 3.5° (deg). The symbol $\langle\cdot\rangle_t$ denotes the time average. Here, $\rho_\infty$, $v_\infty$ and $p_\infty$ are the free-stream density, velocity and pressure values. We take $\rho_\infty = 1.225\,\mathrm{kg/m^3}$, $v_\infty = 248.42\,\mathrm{m/s}$, $p_\infty = 1.013\times10^5\,\mathrm{N/m^2}$ in our computation. The ideal gas equation of state is used with $\gamma = 1.401$. We set the shear viscosity to $\mu = 1.789\times10^{-5}\mathrm{Ns/m^2}$ and the bulk viscosity to $\lambda = 0$. The thermal conductivity is set to $c_v^{-1}\kappa = 3.616\times10^{-5}\mathrm{Ns/m^2}$.

The airfoil is shown in Fig. 3. The chord length is 23 cm as in the experiment [47,48]. In order to resolve the boundary layer, a graded mesh is constructed using a “manifold” mapping [49]. As our primary objective is to compute the pressure coefficient, a moderate grading using $x \mapsto x^2$ is chosen. This yields a 2D mesh with about 0.5 million quadrilaterals. The near wall resolution for this mesh is approximately $\Delta x \approx 250\,\mu\mathrm{m}$, $\Delta y \approx 50\,\mu\mathrm{m}$. The 2D mesh is then extruded with 513 repetitions resulting in a 3D mesh composed of about 274 million grid points. The resolution in the $z$-direction is $\Delta z \approx 90\,\mu\mathrm{m}$. The ratio of the extension of the foil in the transversal direction to the chord length is 0.20.

The no-slip boundary condition is enforced on the airfoil. The slip boundary condition is enforced on the two vertical planes bounding the computational domain in the $z$-direction. The non-reflecting boundary condition using the characteristics variables and described in Section 4.3.2 is used for the outer boundary. The fluid is assumed to be thermally insulated on all the boundaries. Notice that deviating from the experimental setup [48, §2] and other numerical configurations [18, Fig. 2], we do not enforce a boundary-layer transition at $x/c = 0.07$ with a (numerical) trip wire.

The $C_\mathrm{p}$ profile reported in Fig. 3 is computed by averaging over 900 temporal snapshots sampled from the computation every $\Delta t = 1.0\times10^{-4}$ and by eventually averaging the results in the $z$-direction. The experimental results have been provided to us by Prof. J. Peraire and have been extracted from [48, Fig. 7] using the online tool https://automeris.io/WebPlotDigitizer. We observe that the agreement with the experimental results is quite good.

## 6.5. Compute-kernel benchmark and 3D strong scaling results

In this section, we report tests assessing the strong and weak scalability of `ryujin`. All the experiments are performed on the supercomputer SuperMUC-NG,[6] using up to 2,048 nodes of 2 Intel Xeon Platinum 8174 CPUs (24 cores each). The CPU cores are operated at a fixed clock frequency of 2.3 GHz. The C++ implementation is compiled with the GNU compiler, using the option `-march=skylake-avx512 -O3 -funroll-loops` to target the specific machine with AVX-512 vectorization. The theoretical peak performance of a node is 3.5 TFlop/s, and the memory bandwidth of a node is 205 GB/s according to the STREAM benchmark. The detailed node-level analysis of Maier and Kronbichler [23] has shown that the memory bandwidth and evaluations of transcendental functions are the dominating costs. In the following experiments, we show that the fast node-level performance comes

[6] https://top500.org/system/179566/, retrieved on April 15, 2021.

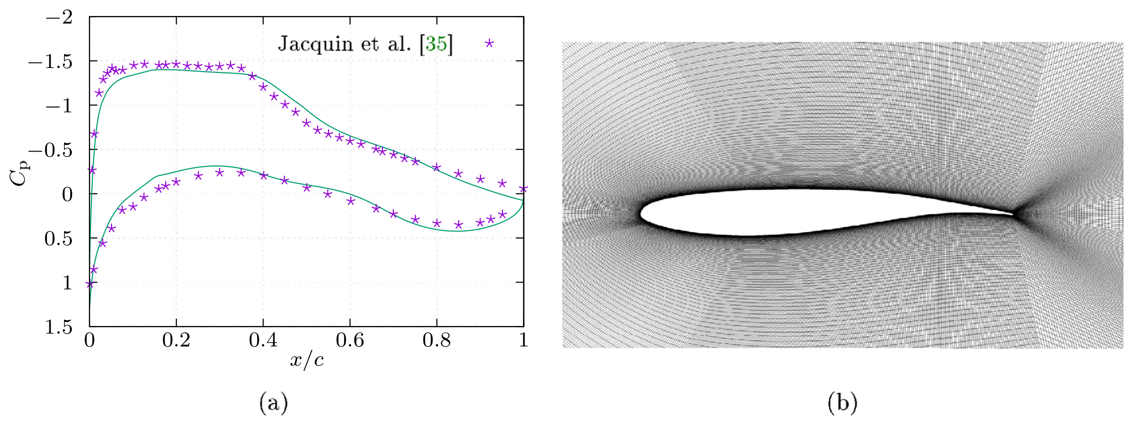


**Fig. 3.** The Onera OAT15a airfoil depicted with a graded mesh used in our computations. The actual hexahedral 3D mesh is created by extruding the quadrilateral 2D mesh in the $z$-direction by a set number of repetition.

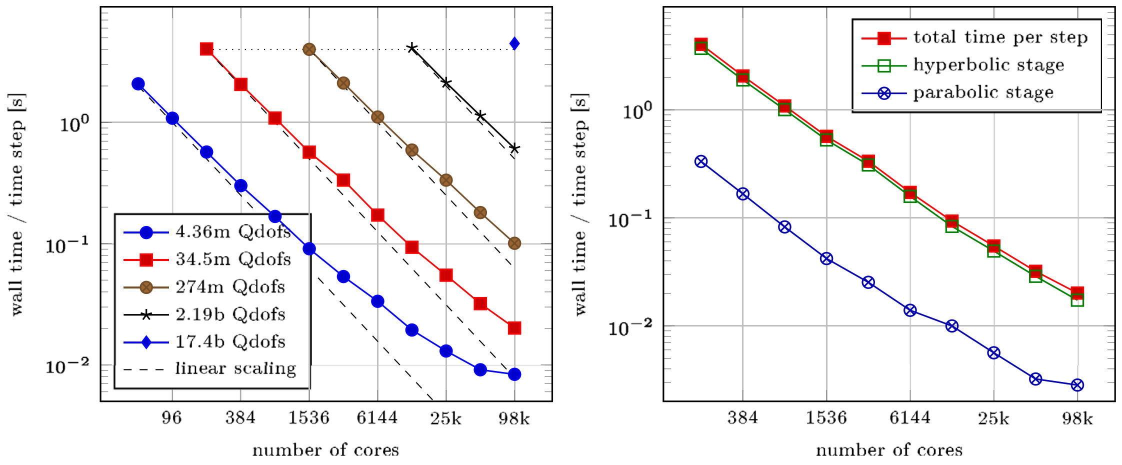


**Fig. 4.** Scaling analysis of the 3D Onera OAT15a airfoil on up to 2,048 nodes (i.e., 98,304 cores) of SuperMUC-NG. In the left panel, the strong and weak scaling of five different problem sizes is assessed, involving up to 17.4 billion grid points (Qdofs) (i.e., 85 billion unknowns). In the right panel, the time per time step for the case with 34.5 million grid points is broken down to show the contributions of the hyperbolic and parabolic parts.

along with optimal scalability to large node counts, where the simulation is partitioned among the participating processes with Morton space filling curves using the `p4est` library by Burstedde et al. [50] wrapped into `deal.II` [51]. Parallelization is based on MPI, using the Intel Omni-Path protocol of SuperMUC-NG for the inter-node communication.

In a first series of tests, we assess the strong scalability performance of `ryujin` on the Onera OAT15a airfoil studied in Section 6.4. The left panel in Fig. 4 shows the time spent per time step versus the number of cores for five different problem sizes ranging from 4.4 million to 17.4 billion grid points (recall that there are 48 cores per node). In order to represent the steady-state performance, the reported time per time step is obtained from an experiment that measures the run time to complete 1000 time steps and then divides this time by the number of time steps. The case with 4.36 million grid points runs into communication latency for 1024 nodes (49k cores). At this point, the number of grid points per node is about 4,257, and the number of grid points per core is about 89. For larger problem sizes per node, the scaling is almost ideal, with a slight loss in efficiency as the share of SIMD-vectorized work in the algorithm by Maier and Kronbichler [23] is reduced near the interfaces between different parallel processes, and by some slight load imbalance in the dynamic steps of the computation as the number of cores increases. The performance achieved is nevertheless excellent. For example, the computation with 274 million grid points

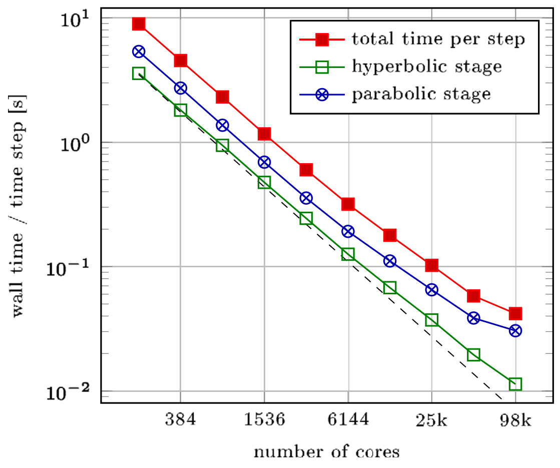


**Fig. 5.** Analysis of strong scaling of 2D shocktube test with 134 million grid points and multigrid solvers for velocity and energy.

on 98k cores achieves a parallel efficiency of 63% when using the computation on 1,536 cores as baseline; the turnaround time is then almost one million time steps per day. The case with 2.2 billion grid points on 98k cores achieves an efficiency of 85% against the computation on 12,288 cores. Note in passing that this series of tests also demonstrates excellent weak scalability. For instance, counting from the top of the panel, the first red square, the first brown circle, the first black star, and the unique blue diamond are perfectly aligned along the horizontal dotted line. This means that the run time per time step remains constant as the number of grid point and the number of cores are both multiplied by 8.

We show in the right panel of Fig. 4 the time per time step for the parabolic step and for the hyperbolic step using the mesh composed of 34.5 millions grid points. This breakdown of the timings reveals that the hyperbolic stage dominates the run time in this case. This is because the Reynolds number in this 3D benchmark is so large that the conjugate gradient solver with a simple inverse mass matrix preconditioner as described in Section 5.3 is efficient. Even for the largest case with 17 billion points, we observe that the velocity solver converges in 2 iterations and the energy solver takes 3 iterations.

Fig. 5 shows a strong scaling experiment done on the 2D shocktube discussed in Section 6.3. The simulations are done on the mesh refinement level L13 with 134 million grid points. Here, the flow is locally dominated by viscous effects, and therefore the geometric multigrid solver described in Section 5.3 is used for optimal performance. While the contribution to the run time per time step of the viscous step is significantly higher when the multigrid solver is activated than when using the plain conjugate gradient solver, the parallel scaling is still excellent in this case. We observe that the strong scaling still holds after multiplying the initial core count by $2^8 = 256$. With $192\times 2^8$ cores, the number of grid points per node is about 131,000. The performance deteriorates though when the initial core count is multiplied by $2^9 = 512$. The excellent strong scaling performance is due to the algorithms developed in Kronbichler and Wall [24], Clevenger et al. [42].

In summary, the above scaling experiments confirm that the proposed algorithms have good node-level performance. They are suitable to solve large-scale problems on large-scale computers in various application scenarios.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Acknowledgments**

The authors thank C. Tenaud and V. Daru for stimulating discussions and for sharing their data [14]. They also thank J. Peraire and his collaborators for sharing the experimental data used in Fig. 3. The authors finally thank S. Deck for providing the OART15a geometry file. They also thank him for providing a file containing experimental data, which, unfortunately, they have not been able to read.

This material is based upon work supported in part by the National Science Foundation, USA via grants DMS 1619892 (BP & JLG), DMS 1620058 (JLG) and DMS 1912847 & 2045636 (MM); the Air Force Office of Scientific Research, USA, USAF, under grant/contract number FA9550-18-1-0397 (JLG&BP); and by the Army Research Office, USA under grant/contract number W911NF-19-1-0431 (JLG&BP). IT was supported by LDRD-21-1189 project #23796 from Sandia National Laboratories, USA. MK was supported by the Bayerisches Kompetenznetzwerk für Technisch-Wissenschaftliches Hoch- und Höchstleistungsrechnen (KONWIHR), Germany. The authors gratefully acknowledge the Gauss Centre for Supercomputing, Germany e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC at Leibniz Supercomputing Centre (LRZ, www.lrz.de) through project id pr83te.

## Appendix A. Boundary fluxes and boundary conditions

We collect in this appendix proofs of results stated in the body of the paper. The statements are repeated for clarity.

**Lemma A.1** (*Balance After Limiting*)**.**

(i) *For all* $\boldsymbol{u}_h := \sum_{i\in\mathcal{V}} \mathbf{U}_i\varphi \in \boldsymbol{P}(\mathcal{T}_h)$*, the following holds true:* $\int_D \boldsymbol{u}_h \,\mathrm{d}x = \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i$.

(ii) *Let* $\mathbf{U}^n$ *be a collection of admissible states. Let* $\mathbf{U}^{n+1}$ *be the update after one forward-Euler step and after limiting. Then the following balance identity holds:*

$$\sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^{n+1} + \tau_n \sum_{i\in\mathcal{V}^\partial} m_i^\partial \mathbb{f}(\mathbf{U}_i^n)\boldsymbol{n}_i = \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^n. \tag{A.1}$$

**Proof.** (i) Using the definition $m_i := \int_D \varphi_i \,\mathrm{d}x$, we have

$$\int_D \boldsymbol{u}_h \,\mathrm{d}x = \sum_{i\in\mathcal{V}} \mathbf{U}_i \int_D \varphi_i \,\mathrm{d}x = \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i.$$

(ii) Recall that the definition of $\mathbf{U}^{\mathrm{L}}$ implies that

$$\sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^{\mathrm{L},n+1} + \tau_n \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{I}(i)} \mathbb{f}(\mathbf{U}_j^n)\int_D \varphi_i\nabla\varphi_j \,\mathrm{d}x - \tau_n \underbrace{\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{I}(i)} d_{ij}^{\mathrm{L},n+1}(\mathbf{U}_j^n - \mathbf{U}_i^n)}_{=0} = \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^n.$$

Using that limiting conserves the total mass, $\sum_{i\in\mathcal{V}} m_i{\mathbf{U}_i}^n = \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^{\mathrm{L},n+1}$, see (3.14), the partition of unity property $\sum_{i\in\mathcal{V}}\varphi_i = 1$, and the definition of $\{{m_j}^\partial\}_{j\in\mathcal{V}^\partial}$ and $\{\boldsymbol{n}_j\}_{j\in\mathcal{V}^\partial}$ yields

$$\begin{aligned}
\sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^n &= \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^{n+1} + \tau_n\sum_{j\in\mathcal{V}} \mathbb{f}(\mathbf{U}_j^n)\int_D \nabla\varphi_j \,\mathrm{d}x \\
&= \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^{n+1} + \tau_n\sum_{j\in\mathcal{V}} \mathbb{f}(\mathbf{U}_j^n)\int_{\partial D} \varphi_j\boldsymbol{n} \,\mathrm{d}s \\
&= \sum_{i\in\mathcal{V}} m_i\mathbf{U}_i^{n+1} + \tau_n\sum_{j\in\mathcal{V}} {m_j}^\partial\mathbb{f}(\mathbf{U}_j^n)\boldsymbol{n}_j.
\end{aligned}$$

The assertion is proved.

**Table 2**
Computed extrema of the skin friction coefficient $C_\mathrm{f}$ for the 2D shocktube configuration for refinement levels L 11 (8 M grid points) through L 14 (512 M grid points). The extrapolated values are computed by fitting the linear function $a + b\,h$ to the four values obtained on the grids L 11 to L 14. We report the coefficient $a$ and the asymptotic standard error in the column labeled "extrapolated". Values for the OSMP7 scheme are taken from [10,14]. The relative deviation reported in the last column is computed by $\big(C_\mathrm{f}(\text{OSMP7}) - C_\mathrm{f}(\text{extrap.})\big)/\big(\max C_\mathrm{f}(\text{extrap.}) - \min C_\mathrm{f}(\text{extrap.})\big)$.

| | **L 11** | **L 12** | **L 13** | **L 14** | **extrapolated** | **OSMP7** |
|---|---|---|---|---|---|---|
| 1 | $1.204{,}09 \times 10^{-1}$ | $1.203{,}77 \times 10^{-1}$ | $1.204{,}23 \times 10^{-1}$ | $1.204{,}34 \times 10^{-1}$ | $1.204{,}24 \times 10^{-1} \pm 2.36 \times 10^{-5}$ | $1.226{,}43 \times 10^{-1}$ ( 0.06%) |
| 2 | $-7.824{,}60 \times 10^{-1}$ | $-8.194{,}50 \times 10^{-1}$ | $-8.408{,}38 \times 10^{-1}$ | $-8.495{,}07 \times 10^{-1}$ | $-8.592{,}32 \times 10^{-1} \pm 9.43 \times 10^{-4}$ | $-8.125{,}68 \times 10^{-1}$ ( 1.21%) |
| 3 | $-2.495{,}33 \times 10^{-1}$ | $-2.490{,}42 \times 10^{-1}$ | $-2.471{,}06 \times 10^{-1}$ | $-2.456{,}95 \times 10^{-1}$ | $-2.459{,}30 \times 10^{-1} \pm 8.50 \times 10^{-4}$ | $-2.597{,}57 \times 10^{-1}$ (-0.36%) |
| 4 | $-4.821{,}52 \times 10^{-1}$ | $-4.956{,}18 \times 10^{-1}$ | $-4.991{,}40 \times 10^{-1}$ | $-5.007{,}07 \times 10^{-1}$ | $-5.044{,}98 \times 10^{-1} \pm 1.42 \times 10^{-3}$ | $-5.069{,}26 \times 10^{-1}$ (-0.06%) |
| 5 | $4.153{,}41 \times 10^{-2}$ | $5.224{,}13 \times 10^{-2}$ | $5.802{,}62 \times 10^{-2}$ | $6.231{,}59 \times 10^{-2}$ | $6.439{,}90 \times 10^{-2} \pm 7.24 \times 10^{-4}$ | $3.578{,}58 \times 10^{-2}$ (-0.74%) |
| 6 | $-1.644{,}34 \times 10^{-1}$ | $-1.653{,}37 \times 10^{-1}$ | $-1.621{,}51 \times 10^{-1}$ | $-1.619{,}35 \times 10^{-1}$ | $-1.620{,}17 \times 10^{-1} \pm 1.25 \times 10^{-3}$ | $-1.694{,}43 \times 10^{-1}$ (-0.19%) |
| 7 | $2.075{,}32 \times 10^{-1}$ | $2.115{,}29 \times 10^{-1}$ | $2.095{,}87 \times 10^{-1}$ | $2.082{,}51 \times 10^{-1}$ | $2.098{,}21 \times 10^{-1} \pm 1.78 \times 10^{-3}$ | $2.329{,}12 \times 10^{-1}$ ( 0.60%) |
| 8 | $5.194{,}67 \times 10^{-2}$ | $4.774{,}62 \times 10^{-2}$ | $6.444{,}45 \times 10^{-2}$ | $6.914{,}07 \times 10^{-2}$ | $6.747{,}67 \times 10^{-2} \pm 7.08 \times 10^{-3}$ | $3.179{,}41 \times 10^{-2}$ (-0.93%) |
| 9 | $1.965{,}33 \times 10^{-1}$ | $1.946{,}37 \times 10^{-1}$ | $2.012{,}83 \times 10^{-1}$ | $2.083{,}07 \times 10^{-1}$ | $2.055{,}60 \times 10^{-1} \pm 4.40 \times 10^{-3}$ | $1.275{,}39 \times 10^{-1}$ (-2.03%) |
| 0 | $-1.874{,}67 \times 10^{-1}$ | $-1.873{,}52 \times 10^{-1}$ | $-1.944{,}32 \times 10^{-1}$ | $-2.012{,}64 \times 10^{-1}$ | $-1.991{,}71 \times 10^{-1} \pm 4.07 \times 10^{-3}$ | $-1.520{,}68 \times 10^{-1}$ ( 1.22%) |
| 11 | $-9.167{,}85 \times 10^{-2}$ | $-8.558{,}52 \times 10^{-2}$ | $-9.308{,}21 \times 10^{-2}$ | $-9.827{,}96 \times 10^{-2}$ | $-9.504{,}32 \times 10^{-2} \pm 4.89 \times 10^{-3}$ | $-6.992{,}61 \times 10^{-2}$ ( 0.65%) |
| 12 | $-2.470{,}24 \times 10^{+0}$ | $-2.636{,}96 \times 10^{+0}$ | $-2.699{,}77 \times 10^{+0}$ | $-2.721{,}51 \times 10^{+0}$ | $-2.735{,}29 \times 10^{+0} \pm 1.31 \times 10^{-3}$ | $-2.730{,}00 \times 10^{+0}$ ( 0.14%) |
| 13 | $9.794{,}93 \times 10^{-1}$ | $1.056{,}81 \times 10^{+0}$ | $1.085{,}12 \times 10^{+0}$ | $1.104{,}35 \times 10^{+0}$ | $1.114{,}16 \times 10^{+0} \pm 7.63 \times 10^{-3}$ | $1.007{,}53 \times 10^{+0}$ (-2.77%) |
| 14 | $-3.904{,}79 \times 10^{-1}$ | $-4.270{,}88 \times 10^{-1}$ | $-4.541{,}09 \times 10^{-1}$ | $-4.638{,}49 \times 10^{-1}$ | $-4.735{,}29 \times 10^{-1} \pm 2.94 \times 10^{-3}$ | $-4.207{,}98 \times 10^{-1}$ ( 1.37%) |
| 15 | $-2.528{,}96 \times 10^{-1}$ | $-2.567{,}42 \times 10^{-1}$ | $-2.656{,}22 \times 10^{-1}$ | $-2.744{,}44 \times 10^{-1}$ | $-2.729{,}34 \times 10^{-1} \pm 4.38 \times 10^{-3}$ | $-1.627{,}05 \times 10^{-1}$ ( 2.86%) |
| 16 | $-6.917{,}56 \times 10^{-1}$ | $-7.172{,}27 \times 10^{-1}$ | $-7.353{,}32 \times 10^{-1}$ | $-7.448{,}41 \times 10^{-1}$ | $-7.504{,}46 \times 10^{-1} \pm 2.41 \times 10^{-3}$ | $-7.409{,}48 \times 10^{-1}$ ( 0.25%) |
| 17 | $6.055{,}57 \times 10^{-1}$ | $6.154{,}61 \times 10^{-1}$ | $6.069{,}03 \times 10^{-1}$ | $6.018{,}52 \times 10^{-1}$ | $6.060{,}99 \times 10^{-1} \pm 5.95 \times 10^{-3}$ | $7.598{,}63 \times 10^{-1}$ ( 3.99%) |
| 18 | $-2.105{,}79 \times 10^{-1}$ | $-2.272{,}79 \times 10^{-1}$ | $-2.421{,}78 \times 10^{-1}$ | $-2.467{,}45 \times 10^{-1}$ | $-2.513{,}36 \times 10^{-1} \pm 2.20 \times 10^{-3}$ | $-2.417{,}54 \times 10^{-1}$ ( 0.25%) |
| 19 | $1.175{,}73 \times 10^{-1}$ | $1.129{,}28 \times 10^{-1}$ | $1.067{,}24 \times 10^{-1}$ | $1.051{,}38 \times 10^{-1}$ | $1.038{,}06 \times 10^{-1} \pm 1.33 \times 10^{-3}$ | $1.159{,}62 \times 10^{-1}$ ( 0.32%) |
| 20 | $1.015{,}99 \times 10^{-1}$ | $9.454{,}46 \times 10^{-2}$ | $8.634{,}67 \times 10^{-2}$ | $8.398{,}69 \times 10^{-2}$ | $8.205{,}06 \times 10^{-2} \pm 1.61 \times 10^{-3}$ | $1.016{,}45 \times 10^{-1}$ ( 0.51%) |
| 21 | $1.717{,}33 \times 10^{-1}$ | $1.789{,}69 \times 10^{-1}$ | $1.794{,}82 \times 10^{-1}$ | $1.796{,}35 \times 10^{-1}$ | $1.818{,}22 \times 10^{-1} \pm 1.35 \times 10^{-3}$ | $1.895{,}60 \times 10^{-1}$ ( 0.20%) |
| 22 | $-1.017{,}07 \times 10^{+0}$ | $-1.059{,}88 \times 10^{+0}$ | $-1.078{,}40 \times 10^{+0}$ | $-1.083{,}24 \times 10^{+0}$ | $-1.095{,}98 \times 10^{+0} \pm 2.82 \times 10^{-3}$ | $-1.128{,}85 \times 10^{+0}$ (-0.85%) |
| 23 | $5.402{,}66 \times 10^{-1}$ | $5.372{,}50 \times 10^{-1}$ | $5.477{,}85 \times 10^{-1}$ | $5.561{,}51 \times 10^{-1}$ | $5.528{,}75 \times 10^{-1} \pm 6.04 \times 10^{-3}$ | $5.869{,}23 \times 10^{-1}$ ( 0.88%) |
| 24 | $3.970{,}91 \times 10^{-1}$ | $3.945{,}69 \times 10^{-1}$ | $3.963{,}79 \times 10^{-1}$ | $3.952{,}94 \times 10^{-1}$ | $3.951{,}08 \times 10^{-1} \pm 9.98 \times 10^{-4}$ | $3.923{,}67 \times 10^{-1}$ (-0.07%) |
| 25 | $4.929{,}19 \times 10^{-1}$ | $5.046{,}19 \times 10^{-1}$ | $5.050{,}90 \times 10^{-1}$ | $5.098{,}68 \times 10^{-1}$ | $5.116{,}00 \times 10^{-1} \pm 1.81 \times 10^{-3}$ | $5.076{,}83 \times 10^{-1}$ (-0.10%) |
| 26 | $-5.703{,}46 \times 10^{-2}$ | $-6.101{,}85 \times 10^{-2}$ | $-7.030{,}93 \times 10^{-2}$ | $-7.318{,}03 \times 10^{-2}$ | $-7.407{,}72 \times 10^{-2} \pm 2.68 \times 10^{-3}$ | $-5.893{,}86 \times 10^{-2}$ ( 0.39%) |
| 27 | $-3.290{,}90 \times 10^{-2}$ | $-3.100{,}79 \times 10^{-2}$ | $-3.301{,}65 \times 10^{-2}$ | $-3.365{,}60 \times 10^{-2}$ | $-3.300{,}18 \times 10^{-2} \pm 1.16 \times 10^{-3}$ | $-3.250{,}20 \times 10^{-2}$ ( 0.01%) |
| 28 | $-6.500{,}36 \times 10^{-2}$ | $-6.827{,}21 \times 10^{-2}$ | $-6.938{,}65 \times 10^{-2}$ | $-6.957{,}24 \times 10^{-2}$ | $-7.059{,}13 \times 10^{-2} \pm 3.44 \times 10^{-4}$ | $-7.872{,}77 \times 10^{-2}$ (-0.21%) |
| 29 | $5.788{,}78 \times 10^{-2}$ | $5.682{,}06 \times 10^{-2}$ | $5.519{,}81 \times 10^{-2}$ | $5.571{,}25 \times 10^{-2}$ | $5.504{,}54 \times 10^{-2} \pm 4.42 \times 10^{-4}$ | $4.684{,}84 \times 10^{-2}$ (-0.21%) |
| 30 | $-1.746{,}15 \times 10^{-1}$ | $-1.915{,}37 \times 10^{-1}$ | $-1.957{,}48 \times 10^{-1}$ | $-1.972{,}36 \times 10^{-1}$ | $-2.021{,}70 \times 10^{-1} \pm 1.98 \times 10^{-3}$ | $-2.064{,}24 \times 10^{-1}$ (-0.11%) |
| 31 | $1.563{,}19 \times 10^{-1}$ | $1.781{,}50 \times 10^{-1}$ | $1.859{,}10 \times 10^{-1}$ | $1.890{,}11 \times 10^{-1}$ | $1.951{,}16 \times 10^{-1} \pm 1.57 \times 10^{-3}$ | $2.072{,}32 \times 10^{-1}$ ( 0.31%) |

**Lemma A.2** (*Slip Condition*). *Let $i \in \mathcal{V}^\partial_\mathrm{s}$, let $\mathbf{U}_i \in \mathcal{A}$, and let $\mathbf{U}^\mathcal{P}_i$ as defined in (4.3).*

(i) *Then $\mathbf{U}^\mathcal{P}_i$ is also admissible, meaning $\mathbf{U}^\mathcal{P}_i \in \mathcal{A}$.*
(ii) *Assume also that the equation of state derives from an entropy $s$. Then $s(\mathbf{U}^\mathcal{P}_i) \geq s(\mathbf{U}_i)$.*
(iii) *For all $i \in \mathcal{V}^\partial_\mathrm{s} \backslash \mathcal{V}^\partial_\mathrm{nr}$, the mass flux and the total energy flux of the postprocessed solution at $i$ is zero (i.e., $\rho(\mathbb{f}(\mathbf{U}^\mathcal{P}_i)\boldsymbol{n}_i) = 0$ and $E(\mathbb{f}(\mathbf{U}^\mathcal{P}_i)\boldsymbol{n}_i) = 0$).*

**Proof.** (i) Let $\mathbf{U}^\mathcal{P}_i =: (\varrho^\mathcal{P}_i, \mathsf{M}^\mathcal{P}_i, \mathsf{E}^\mathcal{P}_i)$. Then $\varrho^\mathcal{P}_i = \varrho_i$, which implies that $\varrho^\mathcal{P}_i > 0$ since $\mathbf{U}_i \in \mathcal{A}$. We also have

$$\varepsilon(\mathbf{U}^\mathcal{P}_i) = \mathsf{E}^\mathcal{P}_i - \tfrac{1}{2\varrho^\mathcal{P}_i}(\mathsf{M}^\mathcal{P}_i)^2 = \mathsf{E}_i - \tfrac{1}{2\varrho_i}((\mathsf{M}_i)^2 - (\mathsf{M}_i \cdot \boldsymbol{n}_i)^2) \geq \mathsf{E}_i - \tfrac{1}{2\varrho_i}(\mathsf{M}_i)^2 = \varepsilon(\mathbf{U}_i).$$

That is $\varepsilon(\mathbf{U}^\mathcal{P}_i) \geq \varepsilon(\mathbf{U}_i) > 0$ because $\mathbf{U}_i \in \mathcal{A}$. Since $\rho(\mathbf{U}^\mathcal{P}_i) = \rho(\mathbf{U}_i)$ and, as proved above, the internal energy $\varepsilon(\mathbf{U})$ stays positive, we infer that the specific internal energy $e(\mathbf{U}) = \varepsilon(\mathbf{U})/\rho$ remains positive too. This proves the first assertion.

(ii) Let us make the change of variable $\sigma(\rho(\mathbf{U}), e(\mathbf{U})) := s(\mathbf{U})$. Using fundamental theorem of calculus

$$s(\mathbf{U}^\mathcal{P}_i) = \sigma(\varrho_i, e(\mathbf{U}^\mathcal{P})) = \sigma(\varrho_i, e(\mathbf{U})) + \int_{e(\mathbf{U})}^{e(\mathbf{U}^\mathcal{P})} \partial_e \sigma(\varrho_i, e)\,\mathrm{d}e.$$

But, in order for $\sigma$ to be physically realistic it has to satisfy $\partial_e \sigma(\rho, e) > 0$. Hence $s(\mathbf{U}^\mathcal{P}_i) \geq \sigma(\varrho_i, e(\mathbf{U})) := s(\mathbf{U}_i)$.

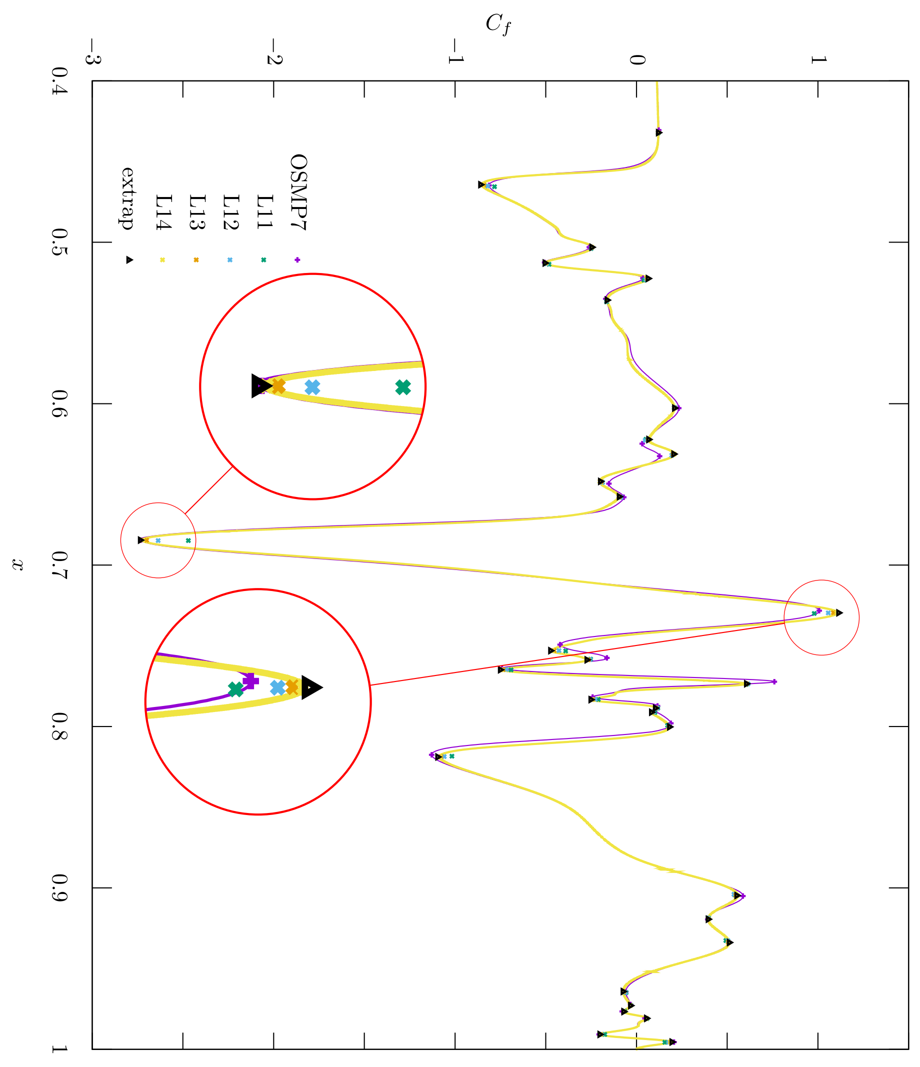


**Fig. 6.** The 2D shocktube benchmark: detailed plot of the skin friction coefficient $C_\mathrm{f}$ at time $t = 1.00$. The continuous lines are for the finest level (L 14) of our computation and the OSMP7 scheme as reported in [10,14]. The two insets show the convergence behavior in the global maximum and minimum, respectively.

(iii) Let $i \in \mathcal{V}_\mathrm{s}^\partial \backslash \mathcal{V}_\mathrm{nr}^\partial$, then ${\boldsymbol{n}_i}^\mathrm{s} = \boldsymbol{n}_i$. Recall from (A.1) that the boundary flux induced by $\mathsf{U}_i^\mathcal{P}$ is ${m_i}^\partial \mathbb{f}(\mathsf{U}_i^\mathcal{P})\boldsymbol{n}_i$. Let $\mathsf{V}_i^\mathcal{P}$ be the velocity of the post-processed state $\mathsf{U}_i^\mathcal{P}$. By definition, $\rho(\mathbb{f}(\mathsf{U}_i^\mathcal{P})\boldsymbol{n}_i) = \varrho_i^\mathcal{P}\mathsf{V}_i^\mathcal{P}\cdot\boldsymbol{n}_i$, $E(\mathbb{f}(\mathsf{U}_i^\mathcal{P})\boldsymbol{n}_i) = \mathsf{V}_i^\mathcal{P}\cdot\boldsymbol{n}_i(\mathsf{E}_i^\mathcal{P} + \mathsf{P}_i^\mathcal{P})$ and $\mathsf{V}_i^\mathcal{P}\cdot\boldsymbol{n}_i = \mathsf{V}_i^\mathcal{P}\cdot{\boldsymbol{n}_i}^\mathrm{s} = 0$, whence the assertion.

**Lemma A.3** (*Global Conservation*)**.** *Assume that* ${\mathcal{V}_\mathrm{s}}^\partial = \mathcal{V}^\partial$ *and* $\mathsf{U}^n$ *satisfies the slip boundary condition (i.e.,* $\mathsf{M}_i^n\cdot\boldsymbol{n}_i = 0$ *for all* $i \in \mathcal{V}^\partial$*). Then the solution obtained at the end the RKSSP(3,3) algorithm after limiting and post-processing, say* $\mathsf{U}^{n+1}$*, satisfies* $\sum_{j\in\mathcal{V}} m_j\varrho_j^{n+1} = \sum_{j\in\mathcal{V}} m_j{\varrho_j}^n$ *and* $\sum_{j\in\mathcal{V}} m_j\mathsf{E}_j^{n+1} = \sum_{j\in\mathcal{V}} m_j{\mathsf{E}_j}^n$.

**Proof.** Let us assume now that $\mathcal{V}_{\text{s}}{}^{\partial} = \mathcal{V}^{\partial}$ and $\mathbf{U}^n$ satisfies the slip boundary condition at every boundary node. Referring to (4.1) for the notation, let us denote $\boldsymbol{w}_h^{(1)} := \sum_{i\in\mathcal{V}} \mathbf{W}_i^{(1)}$ the update obtained after the first forward-Euler step (high-order plus limiting). Then using the identity (A.1), we infer that $\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{W}_i^{(1)}) = \sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^n)$. Similarly, after post-processing $\boldsymbol{w}_h^{(1)}$, the update $\boldsymbol{w}_h^{(2)}$ satisfies

$$\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{W}_i^{(2)}) = \tfrac{3}{4}\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^n) + \tfrac{1}{4}\sum_{i\in\mathcal{V}} m_i \rho((\mathbf{W}_i^{(1)})^{\mathcal{P}}) = \tfrac{3}{4}\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^n) + \tfrac{1}{4}\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{W}_i^{(1)}),$$

i.e., $\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{W}_i^{(2)}) = \sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^n)$. $\boldsymbol{w}_h^{(3)}$ be update obtained at the final stage. After post-processing $\boldsymbol{w}_h^{(2)}$ we obtain

$$\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{W}_i^{(3)}) = \tfrac{1}{3}\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^n) + \tfrac{2}{3}\sum_{i\in\mathcal{V}} m_i \rho((\mathbf{W}_i^{(2)})^{\mathcal{P}}) = \tfrac{1}{3}\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^n) + \tfrac{2}{3}\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{W}_i^{(2)}),$$

i.e., $\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{W}_i^{(3)}) = \sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^n)$. Let $\boldsymbol{u}_h^{n+1} := \sum_i \mathbf{U}_i^{n+1}\varphi_i$ be the final update after post-processing, i.e., $\boldsymbol{u}_h^{n+1} = \mathcal{P}(\boldsymbol{w}_h^{(3)})$. Then, $\sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^{n+1}) = \sum_{i\in\mathcal{V}} m_i \rho(\mathbf{W}_i^{(3)}) = \sum_{i\in\mathcal{V}} m_i \rho(\mathbf{U}_i^n)$. The argument for the conservation of the total energy is identical. Actually, the argument holds for any explicit SSPRK technique.

## Appendix B. Hyperbolic step

```
// Step 1: compute off-diagonal $d_{ij}^{L,n}$ and $\alpha_i$
for $i \in \mathcal{V}^\circ$ do
    compute indicator $\alpha_i$
    for $j \in \mathcal{I}(i),\ j > i$ do
        $d_{ij}^{L,n} \leftarrow \lambda_{\max}(\boldsymbol{n}_{ij}, \mathbf{U}_i^n, \mathbf{U}_j^n)\, \|\boldsymbol{c}_{ij}\|_{\ell^2}$
for $i \in \mathcal{V}^\partial$ do
    compute indicator $\alpha_i$
    for $j \in \mathcal{I}(i),\ j > i$ do
        $d_{ij}^{L,n} \leftarrow \max\Big(\lambda_{\max}(\boldsymbol{n}_{ij}, \mathbf{U}_i^n, \mathbf{U}_j^n)\, \|\boldsymbol{c}_{ij}\|_{\ell^2}, \lambda_{\max}(\boldsymbol{n}_{ji}, \mathbf{U}_j^n, \mathbf{U}_i^n)\, \|\boldsymbol{c}_{ji}\|_{\ell^2}\Big)$
// Step 2: fill lower-diagonal part and compute $d_{ii}^{L,n}$ and $\tau_n$
$\tau_n \leftarrow +\infty$
for $i = 1, \dots, \mathcal{N}$ do
    for $j \in \mathcal{I}(i),\ j < i$ do
        $d_{ij}^{L,n} \leftarrow d_{ji}^{L,n}$
    $d_{ii}^{L,n} \leftarrow -\sum_{j \in \mathcal{I}(i), j \neq i} d_{ij}^{L,n}$ ;   $\tau_n \leftarrow \min\left(\tau_n, -c_{\text{cfl}} \frac{m_i}{2 d_{ii}^{L,n}}\right)$
// Step 3: low-order update, compute $\mathbf{F}_i^{\text{H}}$ and accumulate limiter bounds
for $i \in \mathcal{V}$ do
    $\boldsymbol{U}_i^{n+1} \leftarrow \boldsymbol{U}_i^n$,   $\mathbf{F}_i^{\text{H}} \leftarrow \mathbf{0}$
    for $j \in \mathcal{I}(i)$ do
        $d_{ij}^{H,n} \leftarrow d_{ij}^{L,n} \frac{\alpha_i^n + \alpha_j^n}{2}$ ;   $\mathbf{F}_i^{\text{H}} \leftarrow \mathbf{F}_i^{\text{H}} - \mathbb{f}_j \cdot \boldsymbol{c}_{ij} + d_{ij}^{H,n}(\mathbf{U}_j^n - \mathbf{U}_i^n)$
        $\overline{\mathbf{U}}_{ij}^n \leftarrow \frac{1}{2}(\mathbf{U}_i^n + \mathbf{U}_j^n) - \frac{1}{2 d_{ij}^{L,n}}(\mathbb{f}_j - \mathbb{f}_i) \cdot \boldsymbol{c}_{ij}$
        $\mathbf{U}_i^{n+1} \leftarrow \mathbf{U}_i^{n+1} + \frac{2\tau_n}{m_i} d_{ij}^{L,n} \overline{\mathbf{U}}_{ij}^n$
        accumulate local bounds from $\overline{\mathbf{U}}_{ij}^n$
// Step 4: compute $\mathbf{P}_{ij}$ and $\ell_{ij}$:
for $i \in \mathcal{V}$ do
    for $j \in \mathcal{I}(i)$ do
        $\mathbf{P}_{ij} \leftarrow \frac{\tau_n}{\lambda_i m_i}\Big((d_{ij}^{H,n} - d_{ij}^{H,n})(\mathbf{U}_j^n - \mathbf{U}_i^n) + b_{ij}\mathbf{F}_j^{\text{H}} - b_{ji}\mathbf{F}_i^{\text{H}}\Big)$
        compute $l_{ij}$ from $\boldsymbol{U}_i^{n+1}$, $\mathbf{P}_{ij}$ and local bounds
$\ell \leftarrow \min(\ell, \ell^{\mathsf{T}})$
for $pass = 1, \dots,$ number of limiter passes do
    // Step 5, 6, ...: high-order update and recompute $l_{ij}$:
    for $i \in \mathcal{V}$ do
        for $j \in \mathcal{I}(i)$ do
            $\boldsymbol{U}_i^{n+1} \leftarrow \boldsymbol{U}_i^{n+1} + \lambda_i \ell_{ij} \mathbf{P}_{ij}^n$
    if last round then
        break
    for $i \in \mathcal{V}$ do
        for $j \in \mathcal{I}(i)$ do
            $\mathbf{P}_{ij} \leftarrow (1 - \ell_{ij})\, \mathbf{P}_{ij}$
            compute $l_{ij}$ from $\boldsymbol{U}_i^{n+1}$, $\mathbf{P}_{ij}$ and local bounds
```

**Algorithm 1:** High-order forward Euler step.

## Appendix C. Parabolic step

```
// Step 1: momentum update
assemble right-hand side in (5.2)
solve (5.2)
update momentum, (5.3)
// Step 2: internal energy and total energy update
assemble $\mathsf{K}^{n+\frac{1}{2}}$, (5.4)
solve (5.5)
update internal energy, (5.10)
// Step 3: check bounds and limit if bounds are violated
if $\min_{i\in\mathcal{V}} \mathsf{e}_i^{n+1} < 0$ then
    solve for low-order solution $e_h^{\mathrm{L},n+1}$, (5.7)
    compute limiting matrix $A_{ij}$, (5.9)
    Compute limiters $\ell_{ij}$ using FCT
update total energy, (5.10)
return
```

**Algorithm 2:** High-order parabolic step.

## Appendix D. 2D shocktube benchmark

See Fig. 6 and Table 2.

## References

[1] J.-L. Guermond, M. Maier, B. Popov, I. Tomas, Second-order invariant domain preserving approximation of the compressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 375 (2021) 113608.
[2] R.J. LeVeque, I.M. Mitchell, V. Stodden, Reproducible research for scientific computing: Tools and strategies for changing the culture, Comput. Sci. Eng. 14 (4) (2012) 13–17.
[3] D. Grapsas, R. Herbin, W. Kheriji, J.-C. Latché, An unconditionally stable staggered pressure correction scheme for the compressible Navier-Stokes equations, SMAI J. Comput. Math. 2 (2016) 51–97.
[4] T. Gallouët, L. Gastaldo, R. Herbin, J.-C. Latché, An unconditionally stable pressure correction scheme for the compressible barotropic Navier-Stokes equations, M2AN Math. Model. Numer. Anal. 42 (2) (2008) 303–331.
[5] X. Zhang, On positivity-preserving high order discontinuous Galerkin schemes for compressible Navier-Stokes equations, J. Comput. Phys. 328 (2017) 301–343.
[6] R.M. Beam, R.F. Warming, An implicit factored scheme for the compressible Navier-Stokes equations, in: 3rd Computational Fluid Dynamics Conference, 1977, pp. 130–140.
[7] M.O. Bristeau, R. Glowinski, J. Périaux, Numerical methods for the Navier-Stokes equations. Applications to the simulation of compressible and incompressible viscous flows, in: Finite Elements in Physics, Lausanne, 1986, North-Holland, Amsterdam, 1987, pp. 73–187.
[8] L. Demkowicz, J.T. Oden, W. Rachowicz, A new finite element method for solving compressible Navier-Stokes equations based on an operator splitting method and $h$-$p$ adaptivity, Comput. Methods Appl. Mech. Engrg. 84 (3) (1990) 275–326.
[9] V. Daru, C. Tenaud, Evaluation of TVD high resolution schemes for unsteady viscous shocked flows, Comput. Fluids 30 (1) (2001) 89–113.
[10] V. Daru, C. Tenaud, Numerical simulation of the viscous shock tube problem by using a high resolution monotonicity-preserving scheme, Comput. & Fluids 38 (3) (2009) 664–676.
[11] D. Arndt, W. Bangerth, B. Blais, T.C. Clevenger, M. Fehling, A.V. Grayver, T. Heister, L. Heltai, M. Kronbichler, P. Munch, M. Maier, J.-P. Pelteret, R. Rastak, B. Turcksin, Z. Wang, D. Wells, The deal.II library, Version 9.2, J. Numer. Math. 28 (3) (2020) 131–146.
[12] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, D. Wells, The deal.II finite element library: design, features, and insights, Comput. Math. Appl. 81 (1) (2021) 407–422.
[13] M. Maier, M. Kronbichler, I. Tomas, E. Tovar, ryujin: High-performance second-order collocation-type finite-element scheme for solving the compressible Navier-Stokes and Euler equations, 2021, URL https://zenodo.org/record/4577766.
[14] V. Daru, C. Tenaud, Shock tube problem data, 2020, URL https://zenodo.org/record/4182716.
[15] J.-L. Guermond, M. Kronbichler, M. Maier, B. Popov, I. Tomas, Shock tube problem data, 2021, URL https://zenodo.org/record/4895237.
[16] S. Deck, Numerical simulation of transonic buffet over a supercritical airfoil, AIAA J. 43 (7) (2005) 1556–1566.

[17] S. Deck, N. Renard, Towards an enhanced protection of attached boundary layers in hybrid RANS/LES methods, J. Comput. Phys. 400 (2020) 108970, 36.

[18] C. Nguyen, S. Terrana, J. Peraire, Wall-resolved implicit large eddy simulation of transonic buffet over the OAT15A airfoil using a discontinuous Galerkin method, http://dx.doi.org/10.2514/6.2020-2062.

[19] B. Clayton, J.-L. Guermond, B. Popov, Invariant domain preserving approximations for the Euler equations with tabulated equation of state, SIAM J. Sci. Comput. (2021) In review.

[20] J.-L. Guermond, B. Popov, I. Tomas, Invariant domain preserving discretization-independent schemes and convex limiting for hyperbolic systems, Comput. Methods Appl. Mech. Engrg. 347 (2019) 143–175.

[21] J.-L. Guermond, R. Pasquetti, A correction technique for the dispersive effects of mass lumping for transport problems, Comput. Methods Appl. Mech. Engrg. 253 (2013) 186–198.

[22] J.-L. Guermond, M. Nazarov, B. Popov, Y. Yang, A second-order maximum principle preserving Lagrange finite element technique for nonlinear scalar conservation equations, SIAM J. Numer. Anal. 52 (4) (2014) 2163–2182.

[23] M. Maier, M. Kronbichler, Efficient parallel 3D computation of the compressible Euler equations with an invariant-domain preserving second-order finite-element scheme, ACM Trans. Parallel Comput. 8 (3) (2021) 16/1–30.

[24] M. Kronbichler, W.A. Wall, A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers, SIAM J. Sci. Comput. 40 (5) (2018) A3423–A3448.

[25] M.S. Floater, Generalized barycentric coordinates and applications, Acta Numer. 24 (2015) 161–214.

[26] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (2) (1988) 439–471.

[27] J.F.B.M. Kraaijevanger, Contractivity of Runge-Kutta methods, BIT 31 (3) (1991) 482–528.

[28] P. Colella, H.M. Glaz, Efficient solution algorithms for the Riemann problem for real gases, J. Comput. Phys. 59 (2) (1985) 264–289.

[29] E.F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics, third ed., Springer-Verlag, Berlin, 2009, p. xxiv+724, A practical introduction.

[30] J.-L. Guermond, B. Popov, Fast estimation from above of the maximum wave speed in the Riemann problem for the Euler equations, J. Comput. Phys. 321 (2016) 908–926.

[31] J.-L. Guermond, B. Popov, Invariant domains and first-order continuous finite element approximation for hyperbolic systems, SIAM J. Numer. Anal. 54 (4) (2016) 2466–2489.

[32] J.-L. Guermond, M. Nazarov, B. Popov, I. Tomas, Second-order invariant domain preserving approximation of the Euler equations using convex limiting, SIAM J. Sci. Comput. 40 (5) (2018) A3211–A3239.

[33] J.P. Boris, D.L. Book, Flux-corrected transport. I. SHASTA, a fluid transport algorithm that works [J. Comput. Phys. **11** (1973), no. 1, 38–69], J. Comput. Phys. 135 (2) (1997) 170–186, With an introduction by Steven T. Zalesak, Commemoration of the 30th anniversary of J. Comput. Phys.

[34] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (3) (1979) 335–362.

[35] D. Kuzmin, R. Löhner, S. Turek, Flux–Corrected Transport, in: Scientific Computation, Springer, 2005, 3-540-23730-5.

[36] M. Maier, I. Tomas, The step-69 tutorial program: implementation of a graph-based scheme for Euler's equation of compressible gas dynamics. deal.II Library, 2020, URL https://www.dealii.org/developer/doxygen/deal.II/step_69.html.

[37] G.W. Hedstrom, Nonreflecting boundary conditions for nonlinear hyperbolic systems, J. Comput. Phys. 30 (2) (1979) 222–237.

[38] P.A. Fosso, H. Deniau, N. Lamarque, T. Poinsot, Comparison of outflow boundary conditions for subsonic aeroacoustic simulations, Internat. J. Numer. Methods Fluids 68 (10) (2012) 1207–1233.

[39] M. Kronbichler, K. Kormann, A generic interface for parallel cell-based finite element operator application, Comput. Fluids 63 (2012) 135–147.

[40] P. Fischer, M. Min, T. Rathnayake, S. Dutta, T. Kolev, V. Dobrev, J.-S. Camier, M. Kronbichler, T. Warburton, K. Świrydowicz, J. Brown, Scalability of high-performance PDE solvers, Int. J. High Perf. Comput. Appl. 34 (5) (2020) 562–586.

[41] M. Kronbichler, K. Kormann, Fast matrix-free evaluation of discontinuous Galerkin finite element operators, ACM Trans. Math. Software 45 (3) (2019) 29/1–40.

[42] T.C. Clevenger, T. Heister, G. Kanschat, M. Kronbichler, A flexible, parallel, adaptive geometric multigrid method for FEM, ACM Trans. Math. Softw. 47 (1) (2021) 7/1–27.

[43] W. Gropp, D. Kaushik, D. Keyes, B. Smith, Performance modeling and tuning of an unstructured mesh CFD application, in: SC'00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, 2000, pp. 34–34.

[44] M. Kronbichler, K. Ljungkvist, Multigrid for matrix-free high-order finite element computations on graphics processors, ACM Trans. Parallel Comput. 6 (1) (2019) 2/1–32.

[45] R. Becker, Stoßwelle und Detonation, Z. Phys. 8 (1) (1922) 321–362.

[46] B.M. Johnson, Analytical shock solutions at large and small Prandtl number, J. Fluid Mech. 726 (2013) R4, 12.

[47] L. Jacquin, P. Molton, S. Deck, B. Maury, D. Soulevant, An experimental study of shock oscillation over a transonic supercritical profile, http://dx.doi.org/10.2514/6.2005-4902.

[48] L. Jacquin, P. Molton, S. Deck, B. Maury, D. Soulevant, Experimental study of shock oscillation over a transonic supercritical profile, AIAA J. 47 (9) (2009) 1985–1994.

[49] L. Heltai, W. Bangerth, M. Kronbichler, A. Mola, Propagating geometry information to finite element computations, ACM Trans. Math. Software 47 (4) (2021) 32/1–30.

[50] C. Burstedde, L.C. Wilcox, O. Ghattas, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (3) (2011) 1103–1133.

[51] W. Bangerth, C. Burstedde, T. Heister, M. Kronbichler, Algorithms and data structures for massively parallel generic adaptive finite element codes, ACM Trans. Math. Software 38 (2011) 14/1–28.