Symmetry-preserving finite-difference discretizations of arbitrary order on structured curvilinear staggered grids

Bas van 't Hof[*]   Mathea J. Vuik[†]

September 25, 2019

Abstract

Symmetry-preserving (mimetic) discretization aims to preserve certain properties of a continuous differential operator in its discrete counterpart. For these discretizations, stability and (discrete) conservation of mass, momentum and energy are proven in the same way as for the original continuous model.

This paper presents a new finite-difference symmetry-preserving space discretization. Boundary conditions and time integration are not addressed. The novelty is that it combines arbitrary order of convergence, orthogonal and non-orthogonal structured curvilinear staggered meshes, and the applicability to a wide variety of continuous operators, involving chain rules and nonlinear advection, as illustrated by the shallow-water equations. Experiments show exact conservation and convergence corresponding to expected order.

Symmetry-preserving discretizations, Mimetic methods, Finite-difference methods, Mass, momentum and energy conservation, Curvilinear staggered grid

1 Introduction and motivation

Computer simulations of a flow phenomenon require the discretization of the flow properties, reducing the number of values needed to represent the flow state from infinite to some large finite number. In the resulting discrete model of the flow phenomenon, the (continuous) differential operators have been replaced by (discrete) difference operators. Unfortunately, not all properties of the differential operators are automatically inherited by their discrete approximations. The chain and product rules needed in the manipulation of nonlinear equations, for example, do not always work in discrete cases. Moreover, symmetry and positiveness may be lost in the discretization process, mass, momentum, and energy may not be conserved, aliasing errors can occur, and duality and self-adjointness of the differential operators may be violated [2, 17, 21].

Symmetry-preserving methods, or mimetic methods, aim to preserve certain properties of a continuous operator in its discrete counterpart [17]. The use of symmetry-preserving discretizations makes it possible to construct discrete models which allow all the manipulations needed to prove stability and (discrete) conservation in the same way they were proven in the original continuous model. In this paper, we will present a new finite-difference symmetry-preserving discretization of arbitrary order on structured curvilinear staggered grids.

There are a variety of symmetry-preserving discretizations available in the literature. In [35], an exhaustive overview is given of different techniques to obtain mass- or energy-conserving methods. Typically, symmetry properties of differential operators are only automatically preserved in central-difference approximations on uniform, rectilinear grids [17]. Finite-volume methods can be used to construct conservative discretizations for mass and momentum, but it is in general not possible to also obtain energy conservation [35].

---

[*]Corresponding author. bas.vanthof@vortech.nl. VORtech, Westlandseweg 40d, 2624AD Delft.
[†]thea.vuik@vortech.nl.

In [42, 43], a fourth-order symmetry-preserving finite-volume method is constructed using Richardson extrapolation of a second-order symmetry-preserving method [40]. The extension to unstructured collocated meshes is presented in [32], and an application can be seen in [41]. The extension to upwind discretizations was made in [39], and a discretization for the advection operator for curvilinear collocated meshes was found in [16]. In [17], the method is extended to non-uniform structured curvilinear collocated grids by deriving a discrete product rule. Furthermore, a symmetry-preserving method that conserves mass and energy for compressible-flow equations with a state equation is described in [35]. For rectilinear grids, this method works well, but it is challenging to let this method work for unstructured grids [35]. Finally, in [26], a symmetry-preserving discretization for curvilinear collocated and rectilinear staggered meshes is found exploiting the skew-symmetric nature of the advection operator on square-root variables.

Another option to preserve symmetry is to use discrete filters to regularize the convective terms of the equation [33, 19]. The combination of a symmetry-preserving discretization and regularization for compressible flows is studied in [27].

Mimetic finite-difference methods also mimic the important properties of differential operators. An interesting review is given in [21], and recently, a second-order mimetic discretization of the Navier-Stokes equations conserving mass, momentum, and kinetic energy was presented in [23].

Castillo et al. have provided a framework for mimetic operators in [7, 8]. A second- and fourth-order mimetic approach is constructed for non-uniform rectilinear staggered meshes in [1, 3, 9]. In [10] their method is extended to curvilinear staggered meshes, but discrete conservation of mass, momentum and energy is not shown. A second-order mimetic finite-difference method for rectilinear staggered meshes is also constructed in [28].

Other mimetic finite-difference methods use algebraic topology to design and analyze compatible discrete operators corresponding to a continuous formulation [4, 18, 25]. In order to construct a discrete de Rham complex, certain conditions on reconstruction and reduction operators are imposed: they should be conforming, which means that the reconstruction is a right inverse of the reduction [4], they should be constant preserving [5], and the interpolation operator should commute with the differential operator [5]. In [6] a nice overview of mimetic methods is given. Discrete exterior calculus (DEC) is also related to these mimetic approaches [14]. However, in the papers about these mimetic methods, the discrete conservation of mass, momentum and energy is not studied. In [24], a mass, energy, enstrophy and vorticity conserving method is given for unstructured finite-element meshes, using also a conserving time integrator. Note that the method is not applied to staggered meshes, conservation of momentum is not mentioned, and that only incompressible models are used (chain rules are not needed and the advection operator is easier to process). More about mimetic time integration is found in [29].

Another class of symmetry-preserving methods uses the DG method [47, 12, 11]. In these papers, mass, momentum and total energy are conserved on curvilinear meshes, but staggering is not applied. The method is extended to shock capturing and positivity preservation in [48].

All the existing models in the literature have their own advantages and disadvantages. In general, they are not at the same time applicable for arbitrary discretization orders, sophisticated operators such as the advection operator, or the chain rule, and curvilinear staggered grids. The current paper presents a new discretization method that can handle these requirements simultaneously.

In this work, we first introduce some concepts by using a Galerkin-type approach, which is closely related to existing mimetic methods, and was also studied in [37]. Then, we present our new symmetry-preserving finite-difference technique for discretization in space. The novelty of this work is that it combines several important requirements for discretizations: the symmetry-preserving discretization is made for arbitrary order of accuracy; the method works for orthogonal and non-orthogonal structured curvilinear staggered meshes; and the method can be applied to a wide variety of continuous operators, involving chain rules and nonlinear advection, as will be illustrated by the shallow-water equations. The experiments show exact conservation of mass, momentum and energy, and convergence of the approximations corresponding to the expected order. The approach is very similar to the Richardson extrapolation scheme used in [16] and [17], but the current approach leads

to smaller stencils, especially for high orders of accuracy. Apart from uniform, orthogonal grids, the experiments also use nonorthogonal curvilinear grids, in which the angles between grid lines are as small as 15°. The energy equation, derived from the continuity, momentum and state equations, does not have its own discretization. Instead, discrete energy conservation is derived using the symmetries of the discrete operators [38]. One of the symmetries of interest in this paper is that the adjoint of the gradient is minus the divergence. See Table 2, for more details.

The subject of this paper is symmetry-preserving space discretization. In order to focus on this topic, issues concerning boundary conditions and time integration are not addressed. Instead, a periodic domain is used, and a standard time-integration method is used with such a small time step, that the time-integration errors are negligible. Since the shallow-water equations are the main area of interest for the authors, all examples presented in this paper will be 2D, although the method can also be applied in 1D or 3D.

The outline of this paper is as follows: in Section 2, we present the three different models that will be used in the rest of the paper, and in Section 3, we give some information about the curvilinear grids we use. Section 4 contains the desired discretization properties, and the new symmetry-preserving discretization is explained in Section 5. The effectivity of this new method is investigated for the different models in Section 6. We conclude with a discussion of our method and future work in Section 7.

2   Models

In this paper, three different models are used to explain and test the symmetry-preserving ideas. Each of these models is presented in this section with respect to the following aspects:

- The model equations in terms of continuity, momentum, and state equations;

- A compact representation of the discrete model, including discrete operators like DIV, the discrete divergence, and GRAD, the discrete gradient;

- A consistent energy equation, derived from the continuity, momentum and state equations [38].

  The energy density is given by the sum of the kinetic and internal energy densities: $e = e_{kin} + e_{int}$ (see Table 3 for the definition of energy density in each model). The energy equation will be used to show energy conservation of the models.

Section 4 discusses the properties that the discrete operators DIV, GRAD and others are expected to have, for the discrete models to conserve mass, momentum and energy, after which Section 5 discusses how such operators can be constructed.

- **Linear-wave equations**

  The simplest equation that can be used to discuss symmetry preservation, is the linear-wave equation, in which the evolution of the pressure p in a domain V, the flow velocity $\vec{v}$ and the density ρ, are given by the continuity, momentum, and state equations:

$$\frac{\partial \rho}{\partial t} + \rho_0 \nabla \cdot \vec{v} = 0, \quad \frac{\partial \vec{v}}{\partial t} + \frac{1}{\rho_0} \nabla p = 0, \quad p = c^2 \rho, \tag{1}$$

  where ρ_0 is a constant 'reference' density, and c is the speed of sound (i.e. the propagation speed of waves). Initial conditions are specified at initial time $t = 0$.

  The discrete linear-wave equations are given by

$$\frac{\mathrm{d}\,\mathtt{rho}}{\mathrm{d}t} + \rho_0 \, \mathsf{DIV} \, \mathtt{v} = 0, \quad \frac{\mathrm{d}\,\mathtt{v}}{\mathrm{d}t} + \frac{1}{\rho_0} \mathsf{GRAD} \, \mathtt{p} = 0, \quad \mathtt{p} = c^2 \, \mathtt{rho},$$

3

where p, rho and v are the vectors with the discrete pressures, densities and velocities, and where DIV is the discrete divergence and GRAD is the discrete gradient.

The continuity, momentum and state equations can be combined into the following energy equation:

$$\frac{\partial e}{\partial t} + \nabla \cdot p\vec{v} = 0,$$

where e is the energy density.

This model gives us the opportunity to introduce the relation between symmetry preservation and conservation, as well as the curvilinear staggered grid and staggered velocity components. Two approaches are used for the construction of a discretization. First, we use a Galerkin-type approach similar to the one used in [37], and secondly, we apply a finite-difference approach.

- **Compressible-wave equations**

  We introduce a non-linearity into the system by including density variations in the continuity equation, and so obtain compressible-wave equations (without an advection term):

  $$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho\vec{v} = 0, \quad \frac{\partial \vec{v}}{\partial t} + \nabla Q(p) = \frac{\partial \vec{v}}{\partial t} + \frac{1}{\rho}\nabla p = 0, \quad \rho = R(p),$$

  where the pressure-dependent density function R is assumed positive, continuous and monotonically non-decreasing, and where Q is given by $Q(p) := \int^p 1/R(y) \, dy$. The discrete compressible-wave equations are given by

  $$\frac{d\,\mathtt{rho}}{dt} + \mathsf{DIV\tilde{r}}\; \mathtt{v} = 0, \quad \frac{d\,\mathtt{v}}{dt} + \mathsf{GRAD}\; Q(\mathtt{p}) = 0, \quad \mathtt{rho} = R(\mathtt{p}), \qquad (2)$$

  where the operator $\mathsf{DIV\tilde{r}}$ is a discretization of the operator $(\nabla \cdot \rho)$: this means that $(\mathsf{DIV\tilde{r}}\; \mathtt{v})_i$ approximates $(\nabla \cdot \rho\vec{v})(\vec{x}_i)$. In Section 5.2.2, the explicit construction of the operator $\mathsf{DIV\tilde{r}}$ will be discussed.

  The corresponding energy equation is

  $$\frac{\partial e}{\partial t} + \nabla \cdot \left(e_{int} + \rho_0 \int^p \frac{1}{R(q)} \, dq\right)\vec{v} = 0.$$

  The model is analyzed for an arbitrary state equation $\rho = R(p)$, but tests are only conducted for the state equation $p = c^2\rho$, because then an exact solution is available to verify the results.

- **Isentropic compressible Euler equations**

  Finally, after introducing an advection term, a symmetry-preserving discretization is formed for the following equations of isentropic compressible Euler gas dynamics [20]:

  $$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho\vec{v} = 0, \quad \frac{\partial}{\partial t}\rho\vec{v} + \nabla \cdot (\rho\vec{v} \otimes \vec{v}) + \nabla p = 0, \quad \rho = R(p).$$

  The discrete system is given by

  $$\frac{d\,\mathtt{rho}}{dt} + \mathsf{DIVr}\; \mathtt{v} = 0, \quad \frac{d\,\mathtt{rv}}{dt} + \mathsf{ADVEC}\; \mathtt{v} + \mathsf{GRAD}\; \mathtt{p} = 0, \quad \mathtt{rho} = R(\mathtt{p}). \qquad (3)$$

  The vector $\mathtt{rv}$ contains local momentum values, and is defined by $\mathtt{rv} := \mathrm{diag}(\mathsf{Interp}_{v \leftarrow c}\mathtt{rho})\mathtt{v}$. $\mathsf{Interp}_{v \leftarrow c}$ indicates interpolation from the pressure grid points to the velocity grid points in the staggered grid. The operator DIVr is similar to the operator $\mathsf{DIV\tilde{r}}$: it also approximates the operator $(\nabla \cdot \rho)$, but is constructed in a different way (Section 5.4). Section 5.4 also shows the construction of the discrete advection operator ADVEC.

4

The following energy equation can be derived from the continuity, momentum and state equations of the isentropic compressible Euler equations:

$$\frac{\partial e}{\partial t} + \nabla \cdot (e + p)\vec{v} = 0.$$

Choosing the state equation $\rho = R(p) = \sqrt{2p/g}$ changes these equations into the shallow-water equations [20], which are investigated in this paper.

For each model, an exact solution is created using the propagation speeds and Riemann invariants as given in Table 1. For the compressible-wave equations and the isentropic compressible Euler equations, the exact solutions develop shocks after some time. After that moment, the (weak) solution becomes discontinuous, and although mass and momentum are still conserved, energy is not. In Section 6, we show that the symmetry-preserving method conserves mass, momentum and energy until the moment the shock occurs, and that the method converges with the expected order at a time before the shock occurs. At the moment the shock occurs, the convergence is slower. After the appearance of the shock, the discrete solution, which still conserves energy, cannot possibly be an accurate approximation of the correct (weak) solution, in which energy is lost. The comparison of the solutions therefore stops when the shock appears. The method could be extended with artificial viscosity (for instance in the form of flux limiters), which dissipate some of the energy in the system, and allow correct approximation of the discontinuous solution, but these techniques are beyond the scope of this article.

An example with a developing shock was chosen to show the properties of the method, because it shows that the method is accurate when it can be expected to, that it always conserves mass, momentum and energy, and that there are cases (discontinuous solutions) where the method needs an extension before it can be used to calculate meaningful solutions. All these aspects of the method would not be illustrated by an example where shocks do not develop.

Table 1: Propagation speeds and Riemann invariants used to construct exact one-dimensional solutions for the three models [20]. The one-dimensional velocity is given by v.

	Linear-wave eq.	Compressible-wave eq.	Shallow-water eq.
Propagation speed V			
forward V_+	c	$\frac{1}{2}(v + \sqrt{v^2 + 4c^2})$	$v + \sqrt{g\rho}$
backward V_-	$-c$	$\frac{1}{2}(v - \sqrt{v^2 + 4c^2})$	$v - \sqrt{g\rho}$
Riemann invariant F			
forward F_+	$\rho_0 v + c\rho$	$\ln(\rho V_+) - \frac{vV_-}{2c^2}$	$v + 2\sqrt{g\rho}$
backward F_-	$\rho_0 v - c\rho$	$\ln\left(\frac{\rho}{V_+}\right) - \frac{vV_+}{2c^2}$	$v - 2\sqrt{g\rho}$

3 Curvilinear grid

In this section, we introduce a parametrization for curvilinear grids. This is first done for a collocated mesh (used for scalar fields) and then for a staggered grid (used in vector fields).

3.1 Collocated grids and scalar fields

We introduce a uniform grid in computational-grid space, $\vec{\xi}_{c,i}$, that satisfies

$$\vec{\xi}_{c,i+m_x j + m_x m_y k} = (i\Delta\xi, \; j\Delta\eta, \; k\Delta\zeta)^\top,$$

where (m_x, m_y, m_z) relate to the number of cells in each direction, and $(\Delta\xi, \Delta\eta, \Delta\zeta)$ are the corresponding cell widths. The subscript c is chosen because the grid points can be seen as the cell centers of control volumes with vertices $\vec{\xi}_{c,i} + (\pm\Delta\xi, \; \pm\Delta\eta, \; \pm\Delta\zeta)^\top/2$.

The map $\vec{X}$ relates this uniform grid $\{\vec{\xi}_{c,0}, \ldots, \vec{\xi}_{c,m_x m_y m_z-1}\}$ in computational-grid space to a curvilinear grid $\{\vec{x}_{c,0}, \ldots, \vec{x}_{c,m_x m_y m_z-1}\}$ in physical space:

$$\vec{x}_{c,i} = \vec{X}(\vec{\xi}_{c,i}).$$

The grid formed by the points $\xi_{c,i}$ is called the *pressure grid*, because this is the grid used to sample scalar fields, such as the pressure.

The properties of the mapping $\vec{X}$, along with those of the solution, determine the convergence rate of the discretization used. For clarity of presentation, we have only used smooth (C^∞) mappings.

3.2 Staggered grids and vector fields

In a staggered grid, not only the pressure grid is used, but also a velocity grid. The velocity grid consists of grid points that are shifted by half a grid space, and therefore correspond to *cell-face centers*:

$$\vec{x}_{e,i} := \vec{X}\left(\vec{\xi}_{e,i}\right) := \vec{X}\left(\vec{\xi}_{c,i} + \frac{1}{2}\begin{pmatrix}\Delta\xi\\0\\0\end{pmatrix}\right), \qquad \vec{x}_{n,i} := \vec{X}\left(\vec{\xi}_{n,i}\right) := \vec{X}\left(\vec{\xi}_{c,i} + \frac{1}{2}\begin{pmatrix}0\\\Delta\eta\\0\end{pmatrix}\right),$$

$$\vec{x}_{t,i} := \vec{X}\left(\vec{\xi}_{t,i}\right) := \vec{X}\left(\vec{\xi}_{c,i} + \frac{1}{2}\begin{pmatrix}0\\0\\\Delta\zeta\end{pmatrix}\right),$$

where the subscripts e, n and t stand for 'east', 'north' and 'top', respectively. The numbering of the grid points is visualized in Figure 1.

Figure 1: Numbering of the grid points for a staggered grid in 2D. The points $\vec{x}_{c,i}$ belong to the pressure grid cells (cell centers), and $\vec{x}_{e,i}$, $\vec{x}_{n,i}$ belong to the velocity grid, with grid points located at the east and north cell faces, respectively.

To find vector-field discretizations in curvilinear spaces, we should first introduce the notation of *grid-aligned velocity components*. A vector field $\vec{v}$ is represented by three scalar functions v_x, v_y and v_z, using a *local grid orientation* $(\vec{r}_x, \vec{r}_y, \vec{r}_z)$,

$$\vec{v}(\vec{x}) = v_x(\vec{x})\vec{r}_x(\vec{x}) + v_y(\vec{x})\vec{r}_y(\vec{x}) + v_z(\vec{x})\vec{r}_z(\vec{x}), \tag{4}$$

where $\vec{r}_x$, $\vec{r}_y$ and $\vec{r}_z$ are orthonormal. The scalar functions v_x, v_y and v_z can be calculated from the vector field $\vec{v}$ using inner products:

$$v_x(\vec{x}) = \vec{r}_x(\vec{x}) \cdot \vec{v}(\vec{x}), \quad v_y(\vec{x}) = \vec{r}_y(\vec{x}) \cdot \vec{v}(\vec{x}), \quad v_z(\vec{x}) = \vec{r}_z(\vec{x}) \cdot \vec{v}(\vec{x}). \tag{5}$$

Typically [34, 46], a combination of *covariant* directions (i.e. parallel to grid lines) and *contravariant* directions (i.e. perpendicular to two of the grid lines) is used to construct a local grid orientation. Often [34], this leads to a different grid orientation in ξ-cell faces than in η- or ζ-cell faces. For the method in this paper, a different kind of grid orientation is needed, because a simple representation of the kinetic energy requires a local grid orientation that is

- available in all points in space, not just in grid points;

- is the same for all quantities and equations;

- consists of three orthogonal directions.

In uniform Cartesian grids, the covariant and contravariant directions are the same, and the local grid orientation is given by $\vec{r}_x = (1, 0, 0)$, $\vec{r}_y = (0, 1, 0)$ and $\vec{r}_z = (0, 0, 1)$. In the general case, the contravariant and covariant directions are not the same, and an orthogonal grid orientation cannot be made from them.

An orthogonal, grid-aligned basis for vectors is based on the singular-value decomposition (SVD) of the grid directions, given by the vector $\mathbf{h}$ with the singular values and unitary matrices $\mathbf{P}$ and $\mathbf{Q}$, such that

$$\left(\begin{array}{ccc} \dfrac{\partial \vec{X}}{\partial \xi} & \dfrac{\partial \vec{X}}{\partial \eta} & \dfrac{\partial \vec{X}}{\partial \zeta} \end{array} \right) = \mathbf{P} \operatorname{diag}(\mathbf{h}) \, \mathbf{Q}. \tag{6}$$

The orientation $(\vec{r}_x, \vec{r}_y, \vec{r}_z)$ is now found by using only the unitary rotation matrices of the singular-value decomposition:

$$[\vec{r}_x, \vec{r}_y, \vec{r}_z] := \mathbf{P} \, \mathbf{Q}.$$

This local-orientation matrix is exactly orthogonal, and forms a compromise between the covariant and contravariant directions. A 2D example of the computation of such local orientation is given in Figure 2.

A discrete vector field is represented by a vector with values for each direction: $\mathbf{v} = (\mathbf{vx}^\top, \mathbf{vy}^\top, \mathbf{vz}^\top)^\top$ contains components $\mathbf{vx}$, $\mathbf{vy}$, and $\mathbf{vz}$ that are located at the grid points $\vec{x}_e$, $\vec{x}_n$, and $\vec{x}_t$, respectively. The components are given by

$$\mathbf{vx}_i = \vec{r}_x(\vec{x}_{e,i}) \cdot \vec{v}(\vec{x}_{e,i}), \quad \mathbf{vy}_i = \vec{r}_y(\vec{x}_{n,i}) \cdot \vec{v}(\vec{x}_{n,i}), \quad \mathbf{vz}_i = \vec{r}_z(\vec{x}_{t,i}) \cdot \vec{v}(\vec{x}_{t,i}). \tag{7}$$

Discrete samplings of the local grid orientation are stored in vectors $\mathbf{r}*\mathbf{\_at\_}*$. As an example, we give the expressions for $\vec{r}_x$ at the cell centers or east cell-face centers, and for $\vec{r}_y$ at the north cell-face centers:

$$\vec{r}_x(\vec{x}_{c,i}) = \left(\begin{array}{c} \mathtt{rxx\_at\_c}_i \\ \mathtt{rxy\_at\_c}_i \\ \mathtt{rxz\_at\_c}_i \end{array} \right), \quad \vec{r}_x(\vec{x}_{e,i}) = \left(\begin{array}{c} \mathtt{rxx\_at\_e}_i \\ \mathtt{rxy\_at\_e}_i \\ \mathtt{rxz\_at\_e}_i \end{array} \right), \quad \vec{r}_y(\vec{x}_{n,i}) = \left(\begin{array}{c} \mathtt{ryx\_at\_n}_i \\ \mathtt{ryy\_at\_n}_i \\ \mathtt{ryz\_at\_n}_i \end{array} \right). \tag{8}$$

Unlike in a uniform Cartesian grid, the discrete representation of the constant vector field $(1, 0, 0)$ is not a constant vector, and neither does it have zeros in the n-points and t-points of the grid. To compute the discrete representations $\mathtt{c100}$, $\mathtt{c010}$ and $\mathtt{c001}$ of the constant fields

$$\vec{v}(\vec{x}) = \vec{c}_{(1,0,0)}(\vec{x}) = (1, 0, 0), \quad \vec{v}(\vec{x}) = \vec{c}_{(0,1,0)}(\vec{x}) = (0, 1, 0), \quad \vec{v}(\vec{x}) = \vec{c}_{(0,0,1)}(\vec{x}) = (0, 0, 1),$$

Figure 2: Example of a 2D curvilinear grid and corresponding relations between the Cartesian grid (x, y) and (ξ, η). The solid thin lines correspond to the orientation of the grid lines. The dashed lines are orthogonal to these grid lines. The solid thick lines form the local grid orientation, computed using the SVD technique. Note that $\vec{r}_x$ is closer to the normal (dashed red line) than to the grid line (thin red line), and $\vec{r}_y$ is farther from the normal. This is because the cells are stretched in the y-direction. The vectors are located at the staggered velocity grid points $\vec{x}_e$ and $\vec{x}_n$.

we use equation (7) to find:

$$\texttt{c100} = \begin{pmatrix} \texttt{rxx\_at\_e} \\ \texttt{ryx\_at\_n} \\ \texttt{rzx\_at\_t} \end{pmatrix}, \quad \texttt{c010} = \begin{pmatrix} \texttt{rxy\_at\_e} \\ \texttt{ryy\_at\_n} \\ \texttt{rzy\_at\_t} \end{pmatrix}, \quad \texttt{c001} = \begin{pmatrix} \texttt{rxz\_at\_e} \\ \texttt{ryz\_at\_n} \\ \texttt{rzz\_at\_t} \end{pmatrix}. \tag{9}$$

4 Desired discretization properties

In this section, we investigate the properties that a symmetry-preserving discretization should satisfy. Therefore, we first introduce the different scalar products used.

4.1 Scalar products

Quantities such as densities and pressures form scalar fields, which are discretely approximated by vectors with a value for each pressure grid point. Let a and b be two continuous scalar fields, that correspond to the discrete scalar fields $\texttt{a}$ and $\texttt{b}$. The integral $\int_V ab \, \mathrm{d}V$ can be approximated by the following scalar product:

$$\langle \texttt{a}, \texttt{b} \rangle_c := \texttt{a}^\top \mathrm{diag}(\texttt{dVc})\texttt{b}, \tag{10}$$

where only real numbers are used, and the vector $\texttt{dVc}$ holds the integration weights for each pressure grid point. For instance, the discrete mass, $\texttt{M}$, is calculated from the vector $\texttt{rho}$ of discrete densities according to $\texttt{M} := \langle \texttt{c1}, \texttt{rho} \rangle_c$, where $\texttt{c1}$ is the (constant) vector of only ones.

The situation is more complicated when vector fields are involved. Since the grid orientations have been chosen orthogonal, the scalar product $\langle \texttt{v}, \texttt{w} \rangle_v$ of two discrete vector fields $\texttt{v}$ and $\texttt{w}$ can be written as the sum of the scalar products of the components:

$$\begin{aligned} \langle \texttt{v}, \texttt{w} \rangle_v &:= \langle \texttt{vx}, \texttt{wx} \rangle_e + \langle \texttt{vy}, \texttt{wy} \rangle_n + \langle \texttt{vz}, \texttt{wz} \rangle_t \\ &:= \texttt{vx}^\top \, \mathrm{diag}(\texttt{dVe}) \, \texttt{wx} + \texttt{vy}^\top \, \mathrm{diag}(\texttt{dVn}) \, \texttt{wy} + \texttt{vz}^\top \, \mathrm{diag}(\texttt{dVt}) \, \texttt{wz}, \end{aligned} \tag{11}$$

8

where `dVe`, `dVn` and `dVt` are the vectors holding the integration weights for the grid points on the three directions of the velocity grid.

As an example, the total momentum $\vec{\mathsf{M}}$ is calculated for the compressible-wave model, using a discrete vector field `rv` of local momentum values:

$$\vec{\mathsf{M}} := \left(\langle \mathtt{c100}, \mathtt{rv} \rangle_v, \ \langle \mathtt{c010}, \mathtt{rv} \rangle_v, \ \langle \mathtt{c001}, \mathtt{rv} \rangle_v \right)^\top .$$

The scalar products introduced in this section determine the corresponding adjoints. For example, for a matrix A that maps values on pressure grid points to values on e-points, the adjoint A^* is given by

$$\mathsf{A}^* = \mathrm{diag}(\mathtt{dVc})^{-1} \ \mathsf{A}^\top \ \mathrm{diag}(\mathtt{dVe}),$$

since in that case, we have for all vectors `x` on e-points and `y` on pressure points:

$$\langle \mathtt{x}, \mathsf{A} \ \mathtt{y} \rangle_e = \mathtt{x}^\top \mathrm{diag}(\mathtt{dVe}) \mathsf{A} \ \mathtt{y} = \mathtt{x}^\top (\mathsf{A}^*)^\top \mathrm{diag}(\mathtt{dVc}) \ \mathtt{y} = \langle \mathsf{A}^*\mathtt{x}, \mathtt{y} \rangle_c.$$

The scalar products and adjoints of this section are used to define the properties that discrete operators should satisfy.

4.2 Operator properties

In this section, we discuss Table 2, that shows some properties of the operators used to define the models from Section 2, along with their discrete equivalents. In [26], symmetry properties are used instead of the null space properties that are given in the table. These symmetry properties are applied to curvilinear collocated grids, and therefore this approach could not be applied to curvilinear staggered grids without the modifications presented in this paper.

The continuous versions of the properties often contain a boundary integral. For example, the change in total mass in the compressible-wave equations equals

$$\frac{\partial M}{\partial t} = \frac{\partial}{\partial t} \int_V \rho \ \mathrm{d}V = - \int_V \nabla \cdot \rho \vec{v} \ \mathrm{d}V = - \oint_{\delta V} \rho \vec{v} \cdot \vec{n} \ \mathrm{d}S. \tag{12}$$

In this paper, we do not focus on boundary conditions, and therefore, we take periodic domains, such that equation (12) equals zero. We could also use closed walls, which satisfy $\vec{v} \cdot \vec{n} = 0$. For other boundary conditions, the total mass, momentum and/or energy may change over time, due to an in- or outflux across the domain boundaries. This is beyond the scope of this paper.

Many discrete equalities in the table are of the same form, containing an adjoint operator and a vector consisting of ones. As an example, the equality for $\mathsf{DIV}^*\mathtt{c1}$ should be read as follows:

$$\mathsf{DIV}^*\mathtt{c1} = 0 \iff \mathtt{f}^\top \ \mathrm{diag}(\mathtt{dVc}) \ \mathsf{DIV}^*\mathtt{c1} = 0 \quad \forall \mathtt{f}$$

$$\iff \mathtt{f}^\top \ \mathrm{diag}(\mathtt{dVc}) \ \left\{ \mathrm{diag}(\mathtt{dVc})^{-1} \ \mathsf{DIV}^\top \ \mathrm{diag}(\mathtt{dVc}) \right\} \ \mathtt{c1} = 0 \quad \forall \mathtt{f}$$

$$\iff \mathtt{f}^\top \ \mathsf{DIV}^\top \ \mathrm{diag}(\mathtt{dVc}) \ \mathtt{c1} = 0 \quad \forall \mathtt{f}$$

$$\iff \mathtt{c1}^\top \ \mathrm{diag}(\mathtt{dVc}) \ \mathsf{DIV} \ \mathtt{f} = 0 \quad \forall \mathtt{f}$$

$$\iff \text{The discrete integral of } \mathsf{DIV} \ \mathtt{f} \text{ equals zero} \quad \forall \mathtt{f},$$

which relates to its continuous equivalent. In Section 5, we will construct the discrete operators that satisfy these properties.

4.3 Conserved quantities

In each model, the conserved quantities (total mass, (total momentum) and total energy) can be identified, which are computed using integrals over the simulation domain V. For instance, for the linear-wave equations, the total mass $M := \int_V \rho \ \mathrm{d}V$ is a conserved quantity. Table 3 displays the

Table 2: Several properties of the operators used in the models. The left null space properties are used to prove conservation of mass and momentum. For energy conservation, all properties are required.

Left null space properties	
Continuous version	**Discrete equivalent**
$\displaystyle\int \nabla \cdot \vec{f} \; \mathrm{d}V = \oint_{\delta V} \vec{f} \cdot \vec{n} \; \mathrm{d}S = 0$	$\mathsf{DIV}^{*}\mathsf{c1} = 0$
$\displaystyle\int \nabla f \; \mathrm{d}V = \oint_{\delta V} f \vec{n} \; \mathrm{d}S = \vec{0}$	$\mathsf{GRAD}^{*}\ \mathsf{c100} = \mathsf{GRAD}^{*}\ \mathsf{c010} = \mathsf{GRAD}^{*}\ \mathsf{c001} = 0$
$\displaystyle\int_{V} \nabla \cdot (\rho \vec{v} \otimes \vec{f}) \; \mathrm{d}V = \oint_{\delta V} \rho \vec{f} \vec{v} \cdot \vec{n} \; \mathrm{d}S = \vec{0}$	$\mathsf{ADVEC}^{*}\mathsf{c1} = 0$
$\displaystyle\int_{V} \nabla \cdot \rho \vec{f} \; \mathrm{d}V = \oint_{\delta V} \rho \vec{f} \cdot \vec{n} \; \mathrm{d}S = 0$	$\mathsf{DIV\tilde{r}}^{*}\ \mathsf{c1} = 0$

Zero derivative for constants (right null space properties)	
Continuous version	**Discrete equivalent**
$\nabla \cdot (1,0,0) = \nabla \cdot (0,1,0) = \nabla \cdot (0,0,1) = 0$	$\mathsf{DIV}\ \mathsf{c100} = \mathsf{DIV}\ \mathsf{c010} = \mathsf{DIV}\ \mathsf{c001} = 0$
$\nabla 1 = \vec{0}$	$\mathsf{GRAD}\ \mathsf{c1} = 0$

Chain-rule properties	
Continuous version	**Discrete equivalent**
$R(p)\nabla S(p) = \nabla Q(p)$	$\mathsf{\tilde{r}GRAD}\ S(\mathsf{p}) = \mathsf{GRAD}\ Q(\mathsf{p})$
$R(p)\nabla Q(p) = \nabla p$	$\mathsf{rGRAD}\ Q(\mathsf{p}) = \mathsf{GRAD}\ \mathsf{p}$
with $Q(p) := \int^{p} \frac{1}{R(q)} \; \mathrm{d}q$ and $S(p) := \int^{p} \frac{1}{R^{2}(q)} \; \mathrm{d}q$	

Symmetry properties	
Continuous version	**Discrete equivalent**
$\displaystyle\int_{V} f \nabla \cdot \vec{g} \; \mathrm{d}V + \int_{V} \vec{g} \cdot \nabla f \; \mathrm{d}V = \oint_{\delta V} f \vec{g} \cdot \vec{n} \; \mathrm{d}S = 0$	$\mathsf{DIV} + \mathsf{GRAD}^{*} = 0$
$\displaystyle\int_{V} f \nabla \cdot \rho \vec{g} \; \mathrm{d}V + \int_{V} \vec{g} \cdot \rho \nabla f \; \mathrm{d}V = \oint_{\delta V} \rho f \vec{g} \cdot \vec{n} \; \mathrm{d}S = 0$	$\mathsf{DIV\tilde{r}} + \mathsf{\tilde{r}GRAD}^{*} = 0$
$\displaystyle\int_{V} f \nabla \cdot \rho \vec{g} \; \mathrm{d}V + \int_{V} \vec{g} \cdot \rho \nabla f \; \mathrm{d}V = \oint_{\delta V} \rho f \vec{g} \cdot \vec{n} \; \mathrm{d}S = 0$	$\mathsf{DIVr} + \mathsf{rGRAD}^{*} = 0$
$\displaystyle\int_{V} \vec{g} \cdot \nabla \cdot \rho \vec{v} \otimes f \; \mathrm{d}V + \int_{V} \vec{f} \cdot \nabla \cdot \rho \vec{v} \otimes g \; \mathrm{d}V$ $\displaystyle= \int_{V} \vec{f} \cdot \vec{g} \nabla \cdot \rho \vec{v} \; \mathrm{d}V + \oint_{\delta V} \rho (\vec{f} \cdot \vec{g})(\vec{v} \cdot \vec{n}) \; \mathrm{d}S = \int_{V} \vec{f} \cdot \vec{g} \nabla \cdot \rho \vec{v} \; \mathrm{d}V$	$\mathsf{ADVEC} + \mathsf{ADVEC}^{*} = \mathrm{diag}(\mathsf{Interp}_{v \leftarrow c}\ \mathsf{DIVr}\ \mathsf{v})$

expressions for each of the conserved quantities in the models used in this paper. The symmetry-preserving discretizations should conserve these quantities in a discrete manner. In this section, we explain how discrete conservation can be proved, thereby using scalar products, and the properties from Table 2. Conservation of discrete mass in the linear-wave equations, for instance, is shown in the following very short proof:

$$\frac{\mathrm{dM}}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\langle \mathtt{c1}, \mathtt{rho}\rangle_c = \left\langle \mathtt{c1}, \frac{\mathrm{d\ rho}}{\mathrm{d}t}\right\rangle_c = -\langle \mathtt{c1}, \rho_0\mathsf{DIV}\ \mathtt{v}\rangle_c = -\langle \mathsf{DIV}^*\mathtt{c1}, \rho_0\ \mathtt{v}\rangle_v = 0.$$

Note the transition from an inner product on the pressure grid to the velocity grid in the second last equality. The velocity v is defined at the cell faces, and $\mathsf{DIV}\ \mathtt{v}$ is located at the cell centers. Therefore, the inner product $\langle \mathtt{c1}, \rho_0\mathsf{DIV}\ \mathtt{v}\rangle_c$ is computed in the pressure grid. Note that ρ_0 is constant, and therefore available both on the pressure and on the velocity grid. All proofs for the conservation of mass and momentum follow these same lines, using the left null space properties of the discretizations.

The energy-conservation proofs also use the symmetry properties. As an example, we show energy conservation for the isentropic Euler equations. The derivation of the continuous and discrete energy equations for all the models in this paper are presented in [38]. In the isentropic Euler equations, energy conservation is shown by calculating the time derivative of the discrete total energy $\mathtt{E}$, which is the discrete approximation of the (continuous) total energy $E := \int_V edV$:

$$\frac{\mathrm{dE}}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\langle \mathtt{c1}, e_{int}(\mathtt{p})\rangle_c + \frac{1}{2}\langle \mathtt{v}, \mathtt{rv}\rangle_v\right).$$

If we apply the chain rule twice, we find that the first term equals

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle \mathtt{c1}, e_{int}(\mathtt{p})\rangle_c = \left\langle \mathtt{c1}, \frac{e'_{int}(\mathtt{p})}{R'(\mathtt{p})}\frac{\mathrm{d\ rho}}{\mathrm{d}t}\right\rangle_c.$$

For the second term, we use the fact that we only consider real numbers, such that for fields $\mathtt{a}, \mathtt{b}, \mathtt{c}$ we have

$$\langle \mathtt{a}, \mathrm{diag}(\mathtt{b})\mathtt{c}\rangle = \langle \mathtt{b}, \mathrm{diag}(\mathtt{a})\mathtt{c}\rangle = \langle \mathrm{diag}(\mathtt{a})\mathtt{b}, \mathtt{c}\rangle = \langle \mathtt{b}, \mathrm{diag}(\mathtt{c})\mathtt{a}\rangle, \tag{13}$$

since they all satisfy definition (10). If relation (13) and the product rule are applied several times, we find

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\langle \mathtt{v}, \mathtt{rv}\rangle_v = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\langle \mathtt{v}, \mathrm{diag}(\mathsf{Interp}_{v\leftarrow c}\mathtt{rho})\mathtt{v}\rangle_v = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\langle \mathrm{diag}(\mathtt{v})\mathtt{v}, \mathsf{Interp}_{v\leftarrow c}\mathtt{rho}\rangle_v$$

$$= \left\langle \mathrm{diag}(\mathtt{v})\frac{\mathrm{d}\mathtt{v}}{\mathrm{d}t}, \mathsf{Interp}_{v\leftarrow c}\mathtt{rho}\right\rangle_v + \frac{1}{2}\left\langle \mathrm{diag}(\mathtt{v})\mathtt{v}, \mathsf{Interp}_{v\leftarrow c}\frac{\mathrm{d\ rho}}{\mathrm{d}t}\right\rangle_v$$

$$= \left\langle \mathtt{v}, \mathrm{diag}\left(\frac{\mathrm{d}\mathtt{v}}{\mathrm{d}t}\right)\mathsf{Interp}_{v\leftarrow c}\mathtt{rho}\right\rangle_v + \left\langle \mathrm{diag}(\mathtt{v})\mathtt{v}, \mathsf{Interp}_{v\leftarrow c}\frac{\mathrm{d\ rho}}{\mathrm{d}t}\right\rangle_v - \frac{1}{2}\left\langle \mathrm{diag}(\mathtt{v})\mathtt{v}, \mathsf{Interp}_{v\leftarrow c}\frac{\mathrm{d\ rho}}{\mathrm{d}t}\right\rangle_v$$

$$= \left\langle \mathtt{v}, \frac{\mathrm{d\ rv}}{\mathrm{d}t}\right\rangle_v - \frac{1}{2}\left\langle \mathtt{v}, \mathrm{diag}(\mathtt{v})\mathsf{Interp}_{v\leftarrow c}\frac{\mathrm{d\ rho}}{\mathrm{d}t})\right\rangle_v,$$

such that

$$\frac{\mathrm{dE}}{\mathrm{d}t} = \left\langle \mathtt{c1}, \frac{e'_{int}(\mathtt{p})}{R'(\mathtt{p})}\frac{\mathrm{d\ rho}}{\mathrm{d}t}\right\rangle_c + \left\langle \mathtt{v}, \frac{\mathrm{d\ rv}}{\mathrm{d}t}\right\rangle_v - \frac{1}{2}\left\langle \mathtt{v}, \mathrm{diag}(\mathtt{v})\mathsf{Interp}_{v\leftarrow c}\frac{\mathrm{d\ rho}}{\mathrm{d}t}\right\rangle_v.$$

We can move the factor $e'_{int}(\mathtt{p})/R'(\mathtt{p})$ to the left, and use the continuity and momentum equations (3) to replace the time derivatives with spatial derivatives:

$$\frac{\mathrm{dE}}{\mathrm{d}t} = -\left\langle \frac{e'_{int}(\mathtt{p})}{R'(\mathtt{p})}, \mathsf{DIVr}\ \mathtt{v}\right\rangle_c - \langle \mathtt{v}, \mathsf{GRAD}\ \mathtt{p}\rangle_v - \langle \mathtt{v}, \mathsf{ADVEC}\ \mathtt{v}\rangle_v + \frac{1}{2}\langle \mathtt{v}, \mathrm{diag}(\mathtt{v})\mathsf{Interp}_{v\leftarrow c}\mathsf{DIVr}\ \mathtt{v}\rangle_v.$$

11

The fraction e'_{int}/R' is expanded using the definition of e_{int} for the isentropic Euler equations shown in Table 3:

$$\frac{e'_{int}(\mathsf{p})}{R'(\mathsf{p})} = \frac{1}{R'(p)}\frac{\mathrm{d}}{\mathrm{d}p}\int^p \frac{R(p)-R(q)}{R(q)}\,\mathrm{d}q = \frac{1}{R'(p)}\frac{\mathrm{d}}{\mathrm{d}p}\left(R(p)\int^p \frac{1}{R(q)}\,\mathrm{d}q - \int^p \mathrm{d}q\right)$$

$$= \frac{1}{R'(p)}\left(R'(p)\int^p \frac{1}{R(q)}\,\mathrm{d}q + R(p)\frac{1}{R(p)} - 1\right) = \int^p \frac{1}{R(q)}\,\mathrm{d}q = Q(p).$$

Furthermore, the chain rule for rGRAD is applied (see Table 2), and the two advective terms are factorized:

$$\frac{\mathrm{d}\mathsf{E}}{\mathrm{d}t} = -\left\langle Q(\mathsf{p}), \mathsf{DIVr}\ \mathsf{v}\right\rangle_c - \left\langle \mathsf{v}, \mathsf{rGRAD}\ Q(\mathsf{p})\right\rangle_v - \left\langle \mathsf{v}, \left(\mathsf{ADVEC} - \frac{1}{2}\mathrm{diag}\left(\mathsf{Interp}_{v\leftarrow c}\mathsf{DIVr}\ \mathsf{v}\right)\right)\mathsf{v}\right\rangle_v.$$

Finally, the symmetry property of DIVr and rGRAD is applied to the first terms:

$$-\left\langle Q(\mathsf{p}), \mathsf{DIVr}\ \mathsf{v}\right\rangle_c - \left\langle \mathsf{v}, \mathsf{rGRAD}\ Q(\mathsf{p})\right\rangle_v = -\left\langle Q(\mathsf{p}), \mathsf{DIVr}\ \mathsf{v}\right\rangle_c - \left\langle \mathsf{rGRAD}^*\ \mathsf{v}, Q(\mathsf{p})\right\rangle_v$$

$$= -\left\langle Q(\mathsf{p}), \mathsf{DIVr}\ \mathsf{v}\right\rangle_c + \left\langle \mathsf{DIVr}\ \mathsf{v}, Q(\mathsf{p})\right\rangle_v = 0,$$

and also the symmetry property of ADVEC (see Table 2):

$$\left\langle \mathsf{v}, \left(\mathsf{ADVEC} - \frac{1}{2}\mathrm{diag}\left(\mathsf{Interp}_{v\leftarrow c}\mathsf{DIVr}\ \mathsf{v}\right)\right)\mathsf{v}\right\rangle_v = \left\langle \mathsf{v}, \frac{1}{2}\left(\mathsf{ADVEC} - \mathsf{ADVEC}^*\right)\mathsf{v}\right\rangle_v = 0,$$

since it holds that $\langle \mathsf{x}, (\mathsf{A}-\mathsf{A}^*)\mathsf{x}\rangle = \langle \mathsf{x}, \mathsf{A}\mathsf{x}\rangle - \langle \mathsf{x}, \mathsf{A}^*\mathsf{x}\rangle = \langle \mathsf{x}, \mathsf{A}\mathsf{x}\rangle - \langle \mathsf{A}\mathsf{x}, \mathsf{x}\rangle = 0$ for any matrix A. This means that total energy is indeed conserved: $\frac{\mathrm{d}\mathsf{E}}{\mathrm{d}t} = 0$.

Now that we have investigated the properties that a discretization should satisfy to be symmetry preserving, we only need to construct such discretizations. This will cover the rest of this paper.

5 Numerical approaches

In this section, we describe two different techniques to spatially discretize the models: a Galerkin-type approach (which turns out to be very costly), and a new finite-difference technique. The Galerkin-type approach is only used to introduce some of the concepts that are needed for the finite-difference technique. Once the spatial discretization is found, time integration is needed for a simulation. We use the ODEPACK solver `lsode` [13], where it is verified that the time integration is accurate enough such that it has no influence on the results. Following this approach of highly-accurate time integration allows us to study the spatial discretization separately from the time-integration method, which in actual applications should be symmetry-preserving itself to guarantee the conservation of mass, momentum and/or total energy.

5.1 Galerkin-type approach

This section explains an approach that resembles the Galerkin method that is used in finite-element techniques. This technique will be used for the linear-wave model. The method is easy to understand and contains some concepts that help to understand the new finite-difference technique that is explained in the next section.

The traditional Galerkin method samples the solution to obtain a discrete vector of approximations. This discrete approximation is interpolated using a test function, and then the continuous differential operator is applied. The solution is then approximated by the discrete vector for which the inner product of the residual with all test functions equals zero [45].

In order to construct a symmetry-preserving discretization, we follow the same lines. We interpolate the discrete vector of approximations to a continuous function, apply the continuous differential operator, and sample the function back to a discrete vector. If the interpolation and the sampling

Table 3: Expressions for the conserved quantities mass, momentum and energy in the three models. Here, $\rho = R(p)$ for the compressible-wave model and isentropic Euler model.

	Linear-wave	Compressible-wave	Isentropic Euler
Mass			
continuous M	$\int_V \rho \, \mathrm{d}V$	$\int_V \rho \, \mathrm{d}V$	$\int_V \rho \, \mathrm{d}V$
discrete M	$\langle \mathtt{c1}, \mathtt{rho}\rangle_c$	$\langle \mathtt{c1}, \mathtt{rho}\rangle_c$	$\langle \mathtt{c1}, \mathtt{rho}\rangle_c$
Momentum			
continuous $\vec{M}$	$\rho_0 \int_V \vec{v} \, \mathrm{d}V$	$\rho_0 \int_V \vec{v} \, \mathrm{d}V$	$\int_V \rho\vec{v} \, \mathrm{d}V$
discrete $\vec{\mathsf{M}}$	$\rho_0 \begin{pmatrix} \langle \mathtt{c100}, \mathtt{v}\rangle_v \\ \langle \mathtt{c010}, \mathtt{v}\rangle_v \\ \langle \mathtt{c001}, \mathtt{v}\rangle_v \end{pmatrix}$	$\rho_0 \begin{pmatrix} \langle \mathtt{c100}, \mathtt{v}\rangle_v \\ \langle \mathtt{c010}, \mathtt{v}\rangle_v \\ \langle \mathtt{c001}, \mathtt{v}\rangle_v \end{pmatrix}$	$\begin{pmatrix} \langle \mathtt{c100}, \mathtt{rv}\rangle_v \\ \langle \mathtt{c010}, \mathtt{rv}\rangle_v \\ \langle \mathtt{c001}, \mathtt{rv}\rangle_v \end{pmatrix}$
Energy			
continuous E	$\int_V (e_{int} + e_{kin}) \, \mathrm{d}V$	$\int_V (e_{int} + e_{kin}) \, \mathrm{d}V$	$\int_V (e_{int} + e_{kin}) \, \mathrm{d}V$
e_{int}	$\dfrac{c^2 \rho^2}{2\rho_0}$	$\rho_0 \displaystyle\int^p \dfrac{R(p) - R(q)}{R^2(q)} \, \mathrm{d}q$	$\displaystyle\int^p \dfrac{R(p) - R(q)}{R(q)} \, \mathrm{d}q$
e_{kin}	$\frac{\rho_0}{2}|\vec{v}|^2$	$\frac{\rho_0}{2}|\vec{v}|^2$	$\frac{\rho}{2}|\vec{v}|^2$
discrete E	$\langle \mathtt{c1}, e_{int}(\mathtt{p})\rangle_c$ $+ \dfrac{\rho_0}{2}\langle \mathtt{v}, \mathtt{v}\rangle_v$	$\langle \mathtt{c1}, e_{int}(\mathtt{p})\rangle_c$ $+ \dfrac{\rho_0}{2}\langle \mathtt{v}, \mathtt{v}\rangle_v$	$\langle \mathtt{c1}, e_{int}(\mathtt{p})\rangle_c$ $+ \dfrac{1}{2}\langle \mathtt{rv}, \mathtt{v}\rangle_v$

operator are *mutually adjoint*, and the interpolation of constant functions is exact, then all properties in Table 2 (and thereby all conservation laws) are satisfied. The difference between the original Galerkin method and our approach is two-fold: (i) we focus on specific sampling and interpolation operators that are mutually adjoint, and (ii) in the original Galerkin method, the complete residual is sampled (multiplied by a test function), while we keep the original time derivative and only sample the space derivative and right-hand side. This resembles the idea of a lumped mass matrix. Note that the description below is closely related to mimetic discretizations that are defined using *de Rham maps* [31].

Let $\mathcal{J}_c$ be an interpolation operator that maps from the discrete field on the pressure grid to the continuous field, and $\mathcal{S}_c$ be the sampling operator that produces discrete values from a continuous function [44]. The interpolated fields are written using italic letters and sampled fields with truetype letters, for example, $f := \mathcal{J}_c \, \mathtt{f}$, and $\mathtt{g} = \mathcal{S}_c \, g$. The continuous differential operator $\mathcal{A}$ is applied to the continuous field obtained from interpolation of the discrete field using $\mathcal{J}_c$, and the result is mapped back using $\mathcal{S}_c$, which leads to the discrete operator

$$\mathsf{A} := \mathcal{S}_c \, \mathcal{A} \, \mathcal{J}_c. \tag{14}$$

The sampling operator and the interpolation operator are called *mutually adjoint* if $\mathcal{S}_c = \mathcal{J}_c^*$. In the rest of this section, we will see that mutually-adjoint interpolation and sampling, combined with the

exact interpolation of the constant functions, is in general enough to obtain the properties of the discrete operators in Table 2. For collocated grids, exact interpolation for constant functions means that $\mathcal{J}_c$ c1 $= 1$. For staggered grids, we require that the interpolation operator $\vec{\mathcal{J}}_v$ satisfies

$$(\vec{\mathcal{J}}_v \text{c100})(\vec{x}) = (1, 0, 0), \quad (\vec{\mathcal{J}}_v \text{c010})(\vec{x}) = (0, 1, 0), \quad (\vec{\mathcal{J}}_v \text{c001})(\vec{x}) = (0, 0, 1).$$

5.1.1 Linear-wave equations: operators GRAD and DIV

In this section, we first define the mutually-adjoint interpolation and sampling operators that are used. Then, we present the symmetry-preserving operators DIV and GRAD used for the linear-wave equation. Earlier work on the Galerkin-type discretization was presented in [37].

For a standard uniform one-dimensional grid with $x_i = i$, standard Lagrange interpolation polynomials, w, are used. The corresponding curvilinear interpolation functions are constructed by applying the interpolation function w in each of the three dimensions:

$$w_{c,i+m_x j+m_x m_y k}(\vec{X}(\vec{\xi})) = w\left(\frac{\xi}{\Delta\xi} - i\right) \cdot w\left(\frac{\eta}{\Delta\eta} - j\right) \cdot w\left(\frac{\zeta}{\Delta\zeta} - k\right). \tag{15}$$

The choice of the initial interpolation function w determines the accuracy of the interpolation, as well as the sparseness of the discrete operators.

To see what the mutual adjointness actually means for the sampling and interpolation operators, they are both written in a more explicit form [44], in terms of the *interpolation functions* $w_{c,i}$ (c for cell centers, with point index i). As usual, we use

$$(\mathcal{J}_c \mathbf{f})(\vec{x}) = \sum_i \mathbf{f}_i w_{c,i}(\vec{x}). \tag{16a}$$

Sampling values will be obtained by calculating the integral of the product of a function and a *sampling function* $s_{c,i}$:

$$(\mathcal{S}_c g)_i = \int_V s_{c,i}(\vec{x}) g(\vec{x}) \, \mathrm{d}V. \tag{16b}$$

Mutual adjointness is found using the scalar product on the continuous space (defined by an integral) and the corresponding discrete scalar product on the pressure grid:

$$\langle g, \mathcal{J}_c \mathbf{f} \rangle_{\text{continuous}} = \int_V g(\vec{x})(\mathcal{J}_c \mathbf{f})(\vec{x}) \, \mathrm{d}V = \sum_i \int_V g(\vec{x}) \mathbf{f}_i w_{c,i}(\vec{x}) \, \mathrm{d}V = \sum_i \mathbf{f}_i \mathrm{dVc}_i \int_V g(\vec{x}) \frac{w_{c,i}(\vec{x})}{\mathrm{dVc}_i} \, \mathrm{d}V$$

$$= \sum_i \mathbf{f}_i \mathrm{dVc}_i \int_V s_{c,i}(\vec{x}) g(\vec{x}) \, \mathrm{d}V = \sum_i \mathbf{f}_i \mathrm{dVc}_i (\mathcal{S}_c g)_i = \langle \mathcal{S}_c g, \mathbf{f} \rangle_c,$$

such that the mutually-adjoint sampling functions should satisfy

$$s_{c,i} = \frac{w_{c,i}(\vec{x})}{\mathrm{dVc}_i} = \frac{w_{c,i}(\vec{x})}{\int_V w_{c,i}(\vec{x}) \, \mathrm{d}V}. \tag{17}$$

Here, we have chosen the integration weights dVc_i such that the integral of the sampling function equals 1: in that case, the sampling of a constant field is exact.

As an example, the standard linear interpolation of a one-dimensional function $\mathbf{f}$ between $\mathbf{f}_i$ and $\mathbf{f}_{i+1}$ is given by

$$(\mathcal{J}_c \mathbf{f})(x) = \mathbf{f}_i + \frac{x - x_i}{x_{i+1} - x_i}(\mathbf{f}_{i+1} - \mathbf{f}_i) = \frac{x_{i+1} - x}{x_{i+1} - x_i}\mathbf{f}_i + \frac{x - x_i}{x_{i+1} - x_i}\mathbf{f}_{i+1}, \quad x_i \le x \le x_{i+1}.$$

On the whole domain, the interpolated function satisfies equation (16a), with

$$w_{c,i}(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} \le x \le x_i, \\ \frac{x_{i+1} - x}{x_{i+1} - x_i}, & x_i \le x \le x_{i+1}. \end{cases}$$

14

The corresponding sampling operator follows from equation (17):

$$(\mathcal{S}_c g)_i = \int s_{c,i}(x) g(x) \, \mathrm{d}x = \int g(x) \frac{w_{c,i}(x)}{\int w_{c,i}(x) \, \mathrm{d}x} \, \mathrm{d}x = \frac{2}{x_{i+1} - x_{i-1}} \int g(x) w_{c,i}(x) \, \mathrm{d}x.$$

Similarly, higher-order interpolation and sampling operators can be constructed.

Next, we investigate the Galerkin-type approach for vector fields, in order to define the GRAD and DIV operators for the linear-wave equations. The interpolation of the discrete vector field $\mathtt{v}$ to the continuous vector field $\vec{v} = \vec{\mathcal{J}}_v \mathtt{v}$ is very similar to the interpolation for scalar fields (equation (16a)):

$$(\vec{\mathcal{J}}_v \mathtt{v})(\vec{x}) := \vec{r}_x(\vec{x}) \, (\mathcal{J}_e \mathtt{vx})(\vec{x}) + \vec{r}_y(\vec{x}) \, (\mathcal{J}_n \mathtt{vy})(\vec{x}) + \vec{r}_z(\vec{x}) \, (\mathcal{J}_t \mathtt{vz})(\vec{x}), \tag{18}$$

with

$$(\mathcal{J}_e \mathtt{vx})(\vec{x}) := \sum_i w_{e,i}(\vec{x}) \mathtt{vx}_i, \quad (\mathcal{J}_n \mathtt{vy})(\vec{x}) := \sum_i w_{n,i}(\vec{x}) \mathtt{vy}_i, \quad (\mathcal{J}_t \mathtt{vz})(\vec{x}) := \sum_i w_{t,i}(\vec{x}) \mathtt{vz}_i, \tag{19}$$

where $w_{e,i}$, $w_{n,i}$ and $w_{t,i}$ are interpolation functions similar to equation (15).

Sampling of a continuous vector field $\vec{v}$ to its discrete representation $\mathtt{v} = \mathcal{S}_v \vec{v}$ is similar to the way a continuous scalar field f is sampled to its discrete representation $\mathtt{f} = \mathcal{S}_c f$ (see equations (16b) and (17)), and is given by

$$\mathtt{vx}_i = \int_V \frac{w_{e,i}(\vec{x}) \vec{r}_x \cdot \vec{v}_x(\vec{x})}{\mathtt{dVe}_i} \, \mathrm{d}V, \quad \mathtt{dVe}_i = \int_V w_{e,i}(\vec{x}) \, \mathrm{d}V,$$

and analogously for $\mathtt{vy}$ and $\mathtt{vz}$.

The Galerkin-type approach requires that exact results are obtained when interpolating the discrete representations $\mathtt{c100}$, $\mathtt{c010}$ and $\mathtt{c001}$ of the constant vector fields:

$$(\vec{\mathcal{J}}_v \mathtt{c100})(\vec{x}) = (1, 0, 0), \quad (\vec{\mathcal{J}}_v \mathtt{c010})(\vec{x}) = (0, 1, 0), \quad (\vec{\mathcal{J}}_v \mathtt{c001})(\vec{x}) = (0, 0, 1).$$

Expanding the interpolation operator in $\mathcal{J}_v \mathtt{c100}$ (equation (18)) and taking the scalar product with $\vec{r}_x(\vec{x})$, an equation for the interpolation weights $w_{e,i}(\vec{x})$ is found. Here, we also use the definition of $\mathtt{c100}$ (equation (9)), the fact that the grid orientation is orthonormal, and definition (19):

$$\begin{aligned}
\vec{r}_x(\vec{x}) \cdot (\vec{\mathcal{J}}_v \mathtt{c100})(\vec{x}) &= \vec{r}_x(\vec{x}) \cdot \vec{r}_x(\vec{x}) \, (\mathcal{J}_e \mathtt{rxx\_at\_e})(\vec{x}) + \vec{r}_x(\vec{x}) \cdot \vec{r}_y(\vec{x}) \, (\mathcal{J}_n \mathtt{ryx\_at\_n})(\vec{x}) \\
&\quad + \vec{r}_x(\vec{x}) \cdot \vec{r}_z(\vec{x}) \, (\mathcal{J}_t \mathtt{rzx\_at\_t})(\vec{x}) \\
&= (\mathcal{J}_e \mathtt{rxx\_at\_e})(\vec{x}) = \sum_i w_{e,i}(\vec{x}) \, \mathtt{rxx\_at\_e}_i. \tag{20}
\end{aligned}$$

Combining this with the expected result $\vec{r}_x(\vec{x}) \cdot (\vec{\mathcal{J}}_v \mathtt{c100})(\vec{x}) = \vec{r}_x(\vec{x}) \cdot (1, 0, 0)$, and repeating the process for the constant fields $\mathtt{c010}$ and $\mathtt{c001}$, the following three equations are found for the weights $w_{e,i}(\vec{x})$:

$$\sum_i w_{e,i}(\vec{x}) \mathtt{rxx\_at\_e}_i = \vec{r}_x(\vec{x}) \cdot (1, 0, 0),$$

$$\sum_i w_{e,i}(\vec{x}) \mathtt{rxy\_at\_e}_i = \vec{r}_x(\vec{x}) \cdot (0, 1, 0),$$

$$\sum_i w_{e,i}(\vec{x}) \mathtt{rxz\_at\_e}_i = \vec{r}_x(\vec{x}) \cdot (0, 0, 1).$$

This can be written as a matrix-vector equation with three equations and involving one unknown for each point in the e-grid:

$$\begin{pmatrix} \mathtt{rxx\_at\_e}^\top \\ \mathtt{rxy\_at\_e}^\top \\ \mathtt{rxz\_at\_e}^\top \end{pmatrix} w_{e,:}(\vec{x}) = \vec{r}_x(\vec{x}).$$

15

This is an underdetermined system. We solve for small corrections (in the weighted least-squares sense) to the standard Lagrange interpolation polynomials while making sure that the support of the weight function is no larger than that of the Lagrange polynomials. Similar underdetermined systems have to be solved for w_n and w_t.

The GRAD and DIV operators in the Galerkin-type approach are given by

$$\mathsf{GRAD} = \mathcal{S}_v(\nabla \mathcal{J}_c), \quad \mathsf{DIV} = \mathcal{S}_c(\nabla \cdot \vec{\mathcal{J}}_v).$$

In practice, the divergence operator in $\vec{x}_{c,i}$ is computed as follows:

$$\mathsf{DIV}_i := \sum_j (\mathsf{DIV}_{i,(e,j)} + \mathsf{DIV}_{i,(n,j)} + \mathsf{DIV}_{i,(t,j)}),$$

with

$$\mathsf{DIV}_{i,(e,j)} := \int_V s_{c,i}(\vec{x}) \nabla \cdot w_{e,i}(\vec{x}) \; \mathrm{d}V,$$

and similarly for $\mathsf{DIV}_{i,(n,j)}$ and $\mathsf{DIV}_{i,(t,j)}$, and where $s_{c,i}$ are the sampling functions of equation (17).

5.1.2 Computational costs

In the preceding sections, we have presented the Galerkin-type approach to compute discrete operators. In this section, the computational costs are investigated.

At the start of a simulation, the discrete operators are computed using volume integrals. To guarantee symmetry preservation, these integrals need to be calculated up to machine precision. We achieve high accuracy by using Richardson extrapolation of the composite trapezoidal rule. Although this is a time-consuming process, it is only done once to initialize a simulation. The result is stored in a matrix that corresponds to the discrete operator.

After initialization, the evaluation of the discrete operators is also quite expensive. As an example, we investigate the computational work needed to evaluate the divergence operator, which is related to the number of nonzeros in the DIV matrix. The interpolation function $w_{c,\mathtt{ix,iy,iz}}$ of order $\mathtt{N}$ is zero outside the support $[\mathtt{ix} - \mathtt{N}/2 : \mathtt{ix} + \mathtt{N}/2, \mathtt{iy} - \mathtt{N}/2 : \mathtt{iy} + \mathtt{N}/2, \mathtt{iz} - \mathtt{N}/2 : \mathtt{iz} + \mathtt{N}/2]$. Similarly, the interpolation function $w_{e,\mathtt{jx}+1/2,\mathtt{jy,jz}}$ is zero outside the support
$[\mathtt{jx} + 1/2 - \mathtt{N}/2 : \mathtt{jx} + 1/2 + \mathtt{N}/2, \mathtt{jy} - \mathtt{N}/2 : \mathtt{jy} + \mathtt{N}/2, \mathtt{jz} - \mathtt{N}/2 : \mathtt{jz} + \mathtt{N}/2]$.

Therefore, interpolation functions $w_{c,\mathtt{ix,iy,iz}}$ and $w_{e,\mathtt{jx}+1/2,\mathtt{jy,jz}}$ overlap for all indices $\mathtt{jx,jy,jz}$ for which

$$\mathtt{jx} \in \mathtt{ix} - \mathtt{N} : \mathtt{ix} + \mathtt{N} - 1,$$
$$\mathtt{jy} \in \mathtt{iy} - \mathtt{N} + 1 : \mathtt{iy} + \mathtt{N} - 1,$$
$$\mathtt{jz} \in \mathtt{iz} - \mathtt{N} + 1 : \mathtt{iz} + \mathtt{N} - 1,$$

which is $2 * \mathtt{N} * (2 * \mathtt{N} - 1) * (2 * \mathtt{N} - 1)$ functions. The divergence matrix has three times as many nonzeros (for v_x, v_y and v_z), which is $6 * \mathtt{N} * (2 * \mathtt{N} - 1) * (2 * \mathtt{N} - 1) \approx 24 * \mathtt{N}^3$ nonzeros per matrix row. In 2D, we find $4 * \mathtt{N} * (2 * \mathtt{N} - 1) \approx 8 * \mathtt{N}^2$ nonzeros per matrix row, and in 1D $2 * \mathtt{N}$.

A cheaper approach to arrive at symmetry-preserving discretizations is presented in the next section.

5.2 Finite-difference approach

The Galerkin-type approach described in the previous section has certain disadvantages. For example, the calculation of the discrete operators requires highly-accurate integration, and the resulting matrices have many nonzeros. Therefore, this method leads to relatively long calculation times. Furthermore, the application of the Galerkin-type approach to difficult operators such as the advection operator is not straightforward.

A more efficient and flexible approach is found using a finite-difference strategy. In this section, we explain the finite-difference technique for the different operators needed in the linear-wave, compressible-wave and isentropic Euler models.

5.2.1 Linear-wave equations: operators DIV and GRAD

A higher-order finite-difference approximation DIV for the divergence is constructed from an exact formulation (23) of the divergence in a cell center. Subsequently, standard finite-difference stencils are applied to this formulation, resulting in a higher-order, conservative approximation of the divergence. The resulting discrete operator DIV combines a series of steps such as interpolation and differentiation with respect to $\vec{\xi}$. All of these steps have continuous equivalents, and therefore its negative adjoint GRAD $:= -\text{DIV}^*$ is an accurate approximation of the gradient. The approach is very similar to the Richardson extrapolation scheme used in [16] and [17], but the current approach leads to smaller stencils, especially for high orders of accuracy.

The resulting finite-difference discretizations are locally conservative for the mass balance, because the change in a control volume can be written as the net effect of fluxes, each of which describes the transport between two 'neighbor' grid points. In rectilinear, orthogonal, possibly stretched, grids the momentum equation is locally conservative in the same sense, with momentum control volumes drawn around the cell faces of the mass control volumes. In curvilinear or nonorthogonal grids, the momentum equation cannot be seen as locally conservative, because every sampled velocity contains momentum in a different direction, making it impossible to transfer momentum from one grid point/control volume to another. For similar reasons, the energy balance cannot be seen as locally conservative, because internal energy is sampled in the mass control volumes, and kinetic energy in the cell faces.

For the derivation of the finite-difference technique, a control volume V_i is used, centered around $\vec{x}_{c,i}$. The control volume is an interval (1D), quadrilateral (2D) or hexahedron (3D). The method is described in 2D for simplicity, but it works the same way in 1D and 3D. In 2D, the control volume V_i is given by

$$V_i(\varepsilon) := \vec{x}_{ne,i}(\varepsilon)\vec{x}_{nw,i}(\varepsilon)\vec{x}_{sw,i}(\varepsilon)\vec{x}_{se,i}(\varepsilon),$$

with vertices given by

$$\vec{x}_{ne,i}(\varepsilon) := \vec{X}\left(\vec{\xi}_{c,i} + \varepsilon\left(\begin{smallmatrix}\Delta\xi\\\Delta\eta\end{smallmatrix}\right)\right), \quad \vec{x}_{nw,i}(\varepsilon) := \vec{X}\left(\vec{\xi}_{c,i} + \varepsilon\left(\begin{smallmatrix}-\Delta\xi\\\Delta\eta\end{smallmatrix}\right)\right),$$
$$\vec{x}_{sw,i}(\varepsilon) := \vec{X}\left(\vec{\xi}_{c,i} + \varepsilon\left(\begin{smallmatrix}-\Delta\xi\\-\Delta\eta\end{smallmatrix}\right)\right), \quad \vec{x}_{se,i}(\varepsilon) := \vec{X}\left(\vec{\xi}_{c,i} + \varepsilon\left(\begin{smallmatrix}\Delta\xi\\-\Delta\eta\end{smallmatrix}\right)\right),$$

see Figure 3(a).

The scaled flux of the vector field $\vec{v}$ out of the east cell face is denoted $F_{e,i}$ and given by

$$F_{e,i}(\varepsilon) := \frac{1}{\varepsilon} \int_{\vec{x}_{se,i}(\varepsilon)}^{\vec{x}_{ne,i}(\varepsilon)} \vec{v}(\vec{x}) \cdot \vec{n} \, \mathrm{d}S,$$

and the scaled fluxes $F_{w,i} = -F_{e,i}(-\varepsilon)$, $F_{n,i}$ and $F_{s,i} = -F_{n,i}(-\varepsilon)$ out of the west, north and south cell faces analogously.

Gauss's theorem equates the integrated divergence to the net outflux:

$$\int_{V_i(\varepsilon)} \nabla \cdot \vec{v}(\vec{x}) \, \mathrm{d}V = \varepsilon(F_{e,i}(\varepsilon) + F_{w,i}(\varepsilon) + F_{n,i}(\varepsilon) + F_{s,i}(\varepsilon)) = \varepsilon(F_{e,i}(\varepsilon) - F_{e,i}(-\varepsilon) + F_{n,i}(\varepsilon) - F_{n,i}(-\varepsilon)).$$

For very small ε, the volume integral may be approximated by the midpoint rule (note the occurrence

(a) V_i and $\tilde{F}_{e,i}(\varepsilon)$.

(b) $\bar{\mathbb{F}}_{e,i}(\varepsilon)$.

Figure 3: Controle volumes and approximate fluxes (over bold faces). For $\varepsilon = 1/2$ (in Figure 3(a)), the control volume equals the original grid cell, and $\tilde{F}_{e,i}(\varepsilon) = \bar{\mathbb{F}}_{e,i}(\varepsilon)$.

of the Jacobian determinant of the transformation $\vec{X}$):

$$4\varepsilon^2 \Delta\xi\Delta\eta \left|\frac{\partial\vec{X}}{\partial\vec{\xi}}\right| \nabla\cdot\vec{v}(\vec{x}_{c,i}) + \mathcal{O}(\varepsilon^4) = \int_{V_i(\varepsilon)} \nabla\cdot\vec{v}(\vec{x}) \ \mathrm{d}V$$
$$= \varepsilon(F_{e,i}(\varepsilon) - F_{e,i}(-\varepsilon) + F_{n,i}(\varepsilon) - F_{n,i}(-\varepsilon)),$$

and this leads to the following *exact* representation of the divergence $\nabla\cdot\vec{v}$ (when taking $\varepsilon\to 0$):

$$\nabla\cdot\vec{v}(\vec{x}_{c,i}) = \frac{F'_{e,i}(0) + F'_{n,i}(0)}{2\Delta\xi\Delta\eta\left|\frac{\partial\vec{X}}{\partial\vec{\xi}}\right|} = \frac{F'_{e,i}(0) + F'_{n,i}(0)}{2\mathtt{dVc}_i},$$

because the integration weights will be chosen equal to

$$\mathtt{dVc}_i := \Delta\xi\Delta\eta \left|\frac{\partial\vec{X}}{\partial\vec{\xi}}(\vec{x}_{c,i})\right|.$$

Note that the scaled fluxes $F_{e,i}$ and $F_{n,i}$ are given in terms of boundary integrals. They can be replaced by approximations in which only velocity values $\vec{v}$ occur, and no integration is needed. To do this, the cell-face centers $\vec{x}_{e,i}$ and $\vec{x}_{n,i}$ are defined by

$$\vec{x}_{e,i}(\varepsilon) := \vec{X}(\vec{\xi}_{e,i}(\varepsilon)) := \vec{X}\left(\vec{\xi}_{c,i} + \varepsilon\left(\begin{smallmatrix}\Delta\xi\\0\end{smallmatrix}\right)\right), \quad \vec{x}_{n,i}(\varepsilon) := \vec{X}(\vec{\xi}_{n,i}(\varepsilon)) := \vec{X}\left(\vec{\xi}_{c,i} + \varepsilon\left(\begin{smallmatrix}0\\\Delta\eta\end{smallmatrix}\right)\right).$$

The flux $F_{e,i}$ is approximated by the approximate flux $\tilde{F}_{e,i}$, obtained using the midpoint rule:

$$\tilde{F}_{e,i}(\varepsilon) := \frac{1}{\varepsilon}\int_{\vec{x}_{se,i}(\varepsilon)}^{\vec{x}_{ne,i}(\varepsilon)} \vec{n} \ \mathrm{d}S \ \cdot \ \vec{v}(\vec{x}_{e,i}(\varepsilon))$$
$$= \frac{1}{\varepsilon}|\vec{x}_{ne,i} - \vec{x}_{se,i}| \frac{1}{|\vec{x}_{ne,i} - \vec{x}_{se,i}|}\left(\begin{array}{c}y_{ne,i} - y_{se,i}\\-(x_{ne,i} - x_{se,i})\end{array}\right) \ \cdot \ \vec{v}(\vec{x}_{e,i}(\varepsilon))$$
$$= -\frac{(\vec{x}_{ne,i}(\varepsilon) - \vec{x}_{se,i}(\varepsilon))^{\perp}}{\varepsilon} \cdot \vec{v}(\vec{x}_{e,i}(\varepsilon)), \tag{21}$$

where the definition of the outward-directed normal on the east cell face is used, and the *perp operator* $\perp$ is defined as $(x, y)^\perp := (-y, x)$. The location of $\tilde{F}_{e,i}(\varepsilon)$ for two different values of ε is visualized in Figure 3(a).

Similarly, we define

$$\tilde{F}_{n,i}(\varepsilon) := \frac{(\vec{x}_{ne,i}(\varepsilon) - \vec{x}_{nw,i}(\varepsilon))^\perp}{\varepsilon} \cdot v(\vec{x}_{n,i}(\varepsilon)). \tag{22}$$

Since the midpoint rule is at least second-order accurate, the approximate fluxes and their derivatives are exact at $\varepsilon = 0$. Hence, an *exact* formulation for the divergence $\nabla \cdot \vec{v}$ is given by

$$\nabla \cdot \vec{v}(\vec{x}_{c,i}) = \frac{\tilde{F}'_{e,i}(0) + \tilde{F}'_{n,i}(0)}{2\text{dVc}_i}, \tag{23}$$

which indeed does not include any integrals.

The discrete divergence DIV is now found by executing the following steps:

1. Interpolation of the velocities.

   To compute $\tilde{F}_{e,i}(\varepsilon)$ (equation (21)), we need to evaluate the discrete value of

   $$v(\vec{x}_{e,i}(\varepsilon)) = \vec{r}_x(\vec{x}_{e,i}(\varepsilon))v_x(\vec{x}_{e,i}(\varepsilon)) + \vec{r}_y(\vec{x}_{e,i}(\varepsilon))v_y(\vec{x}_{e,i}(\varepsilon)),$$

   see equation (4). Since $\vec{x}_{e,i}(\varepsilon)$ is a point in the e-grid, the discrete y-component of $\vec{v}$ (the sampled velocity components vy) is not available yet. The component vy is first interpolated to the pressure grid (using the *destaggering matrix* N2C) and then to the e-grid (with the *staggering matrix* C2E) using an interpolation procedure similar to the one in Section 5.1. The result will be stored in the vector vy_at_e = C2E N2C vy =: N2E vy.

   Similarly, the components vx are interpolated to vx_at_n = C2N E2C vx =: E2N vx, so that the complete velocity vector is available at e- and n-points.

   In matrix-vector notation, this interpolation step can be written as

   $$\begin{pmatrix} \text{vx} \\ \text{vy\_at\_e} \\ \text{vx\_at\_n} \\ \text{vy} \end{pmatrix} = \begin{pmatrix} \text{I} & 0 \\ 0 & \text{N2E} \\ \text{E2N} & 0 \\ 0 & \text{I} \end{pmatrix} \begin{pmatrix} \text{vx} \\ \text{vy} \end{pmatrix}. \tag{24}$$

2. Computation of the fluxes.

   All the approximate fluxes use the velocity vector in one point of the staggered grid: every $\tilde{F}_{e,i}$ uses the velocity vector at e-points, and $\tilde{F}_{n,i}$ at n-points. Therefore, the flux values are located at these grid points, for instance, the flux at the e-point $\vec{x}_{e,i}$ is

   $$\begin{aligned} \overline{\mathbf{F}}_{e,i}(\varepsilon) &:= -\frac{\left(\vec{X}\left(\vec{\xi}_{e,i} + \varepsilon\begin{pmatrix} 0 \\ \Delta\eta \end{pmatrix}\right) - \vec{X}\left(\vec{\xi}_{e,i} - \varepsilon\begin{pmatrix} 0 \\ \Delta\eta \end{pmatrix}\right)\right)^\perp}{\varepsilon} \cdot (\vec{r}_x(\vec{x}_{e,i})\text{vx}_i + \vec{r}_y(\vec{x}_{e,i})\text{vy\_at\_e}_i) \\ &=: \text{Nx}_{e,i}(\varepsilon)\,\text{vx}_i + \text{Ny}_{e,i}(\varepsilon)\,\text{vy\_at\_e}_i, \end{aligned}$$

   and similar for $\overline{\mathbf{F}}_{n,k}$ for the n-point $\vec{x}_{n,k}$. In matrix-vector form this equals

   $$\begin{pmatrix} \overline{\mathbf{F}}_e(\varepsilon) \\ \overline{\mathbf{F}}_n(\varepsilon) \end{pmatrix} = \begin{pmatrix} \text{diag}(\text{Nx}_e(\varepsilon)) & \text{diag}(\text{Ny}_e(\varepsilon)) & 0 & 0 \\ 0 & 0 & \text{diag}(\text{Nx}_n(\varepsilon)) & \text{diag}(\text{Ny}_n(\varepsilon)) \end{pmatrix} \begin{pmatrix} \text{vx} \\ \text{vy\_at\_e} \\ \text{vx\_at\_n} \\ \text{vy} \end{pmatrix}. \tag{25}$$

19

Note that the use of the interpolated velocity vectors is not the only difference between $\tilde{F}_{e,i}$ and $\overline{\mathrm{F}}_{e,i}$. Also the x-coordinate of

$$\vec{x}_{ne,i}(\varepsilon) - \vec{x}_{se,i}(\varepsilon) = \vec{X}\left(\vec{\xi}_{c,i} + \varepsilon\left(\begin{smallmatrix} \Delta\xi \\ \Delta\eta \end{smallmatrix}\right)\right) - \vec{X}\left(\vec{\xi}_{c,i} + \varepsilon\left(\begin{smallmatrix} \Delta\xi \\ -\Delta\eta \end{smallmatrix}\right)\right),$$

used in $\tilde{F}_{e,i}$ (at distance $\varepsilon\Delta\xi$, see Figure 3(a)), differs from the x-coordinate of

$$\vec{X}\left(\vec{\xi}_{e,i} + \varepsilon\left(\begin{smallmatrix} 0 \\ \Delta\eta \end{smallmatrix}\right)\right) - \vec{X}\left(\vec{\xi}_{e,i} - \varepsilon\left(\begin{smallmatrix} 0 \\ \Delta\eta \end{smallmatrix}\right)\right),$$

used in $\overline{\mathrm{F}}_{e,i}$ (fixed at cell faces, see Figure 3(b)). For constant vector fields, the interpolation is chosen to be exact, and we find the following relation:

$$\tilde{F}_{e,i}(\varepsilon) = \overline{\mathrm{F}}_{e,i+\varepsilon-1/2}(\varepsilon), \quad \overline{\mathrm{F}}_{e,i}(\varepsilon) = \tilde{F}_{e,i-\varepsilon+1/2}(\varepsilon), \quad \varepsilon \in \{\cdots, -3/2, -1/2, 1/2, 3/2, \cdots\}$$

For non-constant vectors, this only approximately holds. Similarly, the following relation for constant vector fields holds:

$$\tilde{F}_{n,i}(\varepsilon) = \overline{\mathrm{F}}_{n,i+(\varepsilon-1/2)m_x}(\varepsilon), \quad \overline{\mathrm{F}}_{n,i}(\varepsilon) = \tilde{F}_{e,i-(\varepsilon+1/2)m_x}(\varepsilon).$$

3. Finite-difference step.

The flux vectors $\overline{\mathrm{F}}_e = (\overline{\mathrm{F}}_{e,j})_j$ and $\overline{\mathrm{F}}_n = (\overline{\mathrm{F}}_{n,k})_k$ are used to calculate the discrete operator DIV corresponding to equation (23). Standard differentiation stencils are applied to calculate DIV (which lives on the pressure points). Therefore, we use the matrices DIFFX and DIFFY, that make sure that central differences are used. In internal pressure points $\vec{x}_{c,i}$, row i of DIFFX contains a -1 at position $(i, i - (\varepsilon + 1/2))$ and a 1 at $(i, i + (\varepsilon - 1/2))$, and DIFFY is defined in a similar way, so that the i-th entry is given by

$$(\mathrm{DIFFX}(\varepsilon)\,\overline{\mathrm{F}}_e(\varepsilon))_i = \overline{\mathrm{F}}_{e,i+\varepsilon-1/2} - \overline{\mathrm{F}}_{e,i-\varepsilon-1/2},$$
$$(\mathrm{DIFFY}(\varepsilon)\,\overline{\mathrm{F}}_n(\varepsilon))_i = \overline{\mathrm{F}}_{n,i+(\varepsilon-1/2)m_x} - \overline{\mathrm{F}}_{n,i-(\varepsilon+1/2)m_x}.$$

The divergence (on the whole domain) can be computed as follows

$$\mathrm{DIV}\ \mathrm{v} = \frac{1}{2}\mathrm{diag}(\mathrm{dVc})^{-1} \sum_{\varepsilon=1/2,\ 3/2,\ \ldots} \left(\mathrm{DIFFX}(\varepsilon)\overline{\mathrm{F}}_e(\varepsilon) + \mathrm{DIFFY}(\varepsilon)\overline{\mathrm{F}}_n(\varepsilon)\right)\alpha(\varepsilon), \tag{26}$$

where α contains the coefficients of the differentiation stencil. For example, for second-order differentiation, $\alpha(1/2) = 1$, and for fourth order, we use [15, 22]

$$\alpha\left(1/2\right) = \frac{9}{8}\quad,\quad \alpha\left(3/2\right) = -\frac{1}{24}. \tag{27}$$

We need to show that this choice for the divergence operator indeed satisfies the null space properties (Table 2). The left null space property holds if $\mathrm{DIV}^*\mathrm{c1} = 0$. Indeed, the integral of a divergence is zero:

$$\langle\mathrm{c1}, \mathrm{DIV}\ \mathrm{v}\rangle_c = \mathrm{c1}^\top\mathrm{diag}(\mathrm{dVc})\mathrm{DIV}\ \mathrm{v} = \frac{1}{2}\mathrm{c1}^\top \sum_{\varepsilon=1/2,\ 3/2,\ \ldots} \left(\mathrm{DIFFX}(\varepsilon)\overline{\mathrm{F}}_e(\varepsilon) + \mathrm{DIFFY}(\varepsilon)\overline{\mathrm{F}}_n(\varepsilon)\right)\alpha(\varepsilon) = 0,$$

because $\mathrm{c1}^\top\mathrm{DIFFX}(\varepsilon) = \mathrm{c1}^\top\mathrm{DIFFY}(\varepsilon) = 0$. Next, we prove that the divergence of a constant vector field is zero. The interpolations used to calculate vx_at_n and vy_at_e are constructed such that they are exact for constant vector fields. Therefore, for a discrete vector field v representing a constant

20

vector field ($\vec{v}(\vec{x}) = \vec{c}$), a zero divergence is found, because at grid point $\vec{x}_{c,i}$, we have (note the exact relation between $\overline{\mathbf{F}}$ and $\tilde{F}$)

$$\left(\mathsf{DIFFX}(\varepsilon)\overline{\mathbf{F}}_e(\varepsilon) + \mathsf{DIFFY}(\varepsilon)\overline{\mathbf{F}}_n(\varepsilon)\right)_i = \tilde{F}_{e,i}(\varepsilon) - \tilde{F}_{e,i}(-\varepsilon) + \tilde{F}_{n,i}(\varepsilon) - \tilde{F}_{n,i}(-\varepsilon)$$

$$= -\frac{(\vec{x}_{ne,i}(\varepsilon) - \vec{x}_{se,i}(\varepsilon))^\perp}{\varepsilon} \cdot \vec{c} + \frac{(\vec{x}_{nw,i}(\varepsilon) - \vec{x}_{sw,i}(\varepsilon))^\perp}{\varepsilon} \cdot \vec{c}$$

$$+ \frac{(\vec{x}_{ne,i}(\varepsilon) - \vec{x}_{nw,i}(\varepsilon))^\perp}{\varepsilon} \cdot \vec{c} - \frac{(\vec{x}_{se,i}(\varepsilon) - \vec{x}_{sw,i}(\varepsilon))^\perp}{\varepsilon} \cdot \vec{c} = 0.$$

The gradient operator GRAD is obtained from the divergence's adjoint and is given by

$$\mathsf{GRAD} := -\mathsf{DIV}^*. \tag{28}$$

Using the notations $\mathtt{Nx}_e$, $\mathtt{Nx}_n$, $\mathtt{Ny}_e$, $\mathtt{Ny}_n$, $\mathsf{E2N}$ and $\mathsf{N2E}$, introduced in this section, and the definition of the adjoint, the gradient operator GRAD is given by

$$\mathsf{GRAD} = \sum_{\varepsilon=1/2,\ 3/2,\ ...} \frac{\alpha(\varepsilon)}{2} \mathsf{GRAD1}(\varepsilon) \begin{pmatrix} -\mathsf{DIFFX}^\top(\varepsilon) \\ -\mathsf{DIFFY}^\top(\varepsilon) \end{pmatrix},$$

where the operator $\mathsf{GRAD1}$ is given by

$$\mathsf{GRAD1}(\varepsilon) = \mathrm{diag}\begin{pmatrix} \mathtt{dVe} \\ \mathtt{dVn} \end{pmatrix}^{-1} \begin{pmatrix} \mathsf{I} & 0 & \mathsf{E2N}^\top & 0 \\ 0 & \mathsf{N2E}^\top & 0 & \mathsf{I} \end{pmatrix} \begin{pmatrix} \mathrm{diag}(\mathtt{Nx}_e(\varepsilon)) & 0 \\ \mathrm{diag}(\mathtt{Ny}_e(\varepsilon)) & 0 \\ 0 & \mathrm{diag}(\mathtt{Nx}_n(\varepsilon)) \\ 0 & \mathrm{diag}(\mathtt{Ny}_n(\varepsilon)) \end{pmatrix}.$$

In general, one cannot expect the (discrete) adjoint $\mathsf{GRAD} = -\mathsf{DIV}^*$ of an accurate discretization of the divergence to be an accurate discretization of the gradient. However, the divergence DIV can be seen as an operator in computational space, discretized on a uniform (ξ, η)-grid. On uniform grids, the adjoint of an interpolation operator is itself an accurate interpolation (in the opposite direction). Similarly, the adjoint of a discrete derivative operator is itself an accurate discrete derivative operator on uniform grids. Therefore, an accurate discretization of the continuous gradient operator is found if adjoint interpolations and adjoint derivative operators are combined to a (discrete) GRAD operator.

The finite-difference approach is computationally more efficient than the Galerkin-type approach. Similar to Section 5.1.2, we investigate the computational work needed to evaluate the divergence operator. Here, we assume interpolation and differentiation of order $\mathtt{N}$. We start with the interpolation of v_x, v_y and v_z to the pressure grid points. Each interpolation involves $\mathtt{N}$ grid points, so this step is a matrix-vector multiplication with $3*\mathtt{N}$ nonzeros per grid point. Next, v_y and v_z are interpolated to the points of the e-grid, v_x and v_z to the points of the n-grid, and v_x and v_y to the points of the t-grid. This is three matrix-vector multiplications, each involving $2*\mathtt{N}$ nonzeros. Finally, the evaluation of the divergence involves an x-, y- and z-derivative, each of which involves $\mathtt{N}$ nonzeros. Therefore, the evaluation of the divergence requires $3*\mathtt{N}+3*2*\mathtt{N}+3*\mathtt{N} = 12*\mathtt{N}$ nonzeros per grid point. In 2D, the calculation involves $2*\mathtt{N}+2*\mathtt{N}+2*\mathtt{N} = 6*\mathtt{N}$ nonzeros per grid point, and in 1D $\mathtt{N}$ nonzeros. Indeed, the calculation of the divergence operator is much cheaper than when using a Galerkin-type approach. This observation generally holds for all the operators that are designed in this paper.

5.2.2 Compressible-wave equations: operators $\tilde{r}\mathsf{GRAD}$ and $\mathsf{DIV}\tilde{r}$

The discretization of the compressible-wave equations requires the operator $\tilde{r}\mathsf{GRAD}$, which is an approximation of the operator $(\tilde{r}\mathsf{GRAD}\ \mathtt{f})_i \approx (\rho\nabla f)(\vec{x}_{c,i})$. The operator $\tilde{r}\mathsf{GRAD}$ is obtained by

applying the same interpolations used in the GRAD-operator,

$$\tilde{r}\mathsf{GRAD} := \sum_{\varepsilon=1/2,\ 3/2,\ \dots} \frac{\alpha(\varepsilon)}{2} \mathsf{GRAD1}(\varepsilon)\ \mathrm{diag} \left(\begin{array}{c} \widetilde{\mathtt{rho}}_e(\varepsilon) \\ \widetilde{\mathtt{rho}}_n(\varepsilon) \end{array} \right) \left(\begin{array}{c} -\mathsf{DIFFX}^\top(\varepsilon) \\ -\mathsf{DIFFY}^\top(\varepsilon) \end{array} \right),$$

$$\mathsf{DIV\tilde{r}} := -\tilde{r}\mathsf{GRAD}^*,$$

where the intermediate densities $\widetilde{\mathtt{rho}}_e$ and $\widetilde{\mathtt{rho}}_n$ are given by

$$\mathrm{diag} \left(\begin{array}{c} \widetilde{\mathtt{rho}}_e(\varepsilon) \\ \widetilde{\mathtt{rho}}_n(\varepsilon) \end{array} \right) := \mathrm{diag} \left(\begin{array}{c} \mathsf{DIFFX}^\top(\varepsilon)\ Q(\mathtt{p}) \\ \mathsf{DIFFY}^\top(\varepsilon)\ Q(\mathtt{p}) \end{array} \right) \mathrm{diag} \left(\begin{array}{c} \mathsf{DIFFX}^\top(\varepsilon)\ S(\mathtt{p}) \\ \mathsf{DIFFY}^\top(\varepsilon)\ S(\mathtt{p}) \end{array} \right)^{-1}.$$

Here, the functions Q and S that were introduced in Table 2 are applied. Obviously, the chain rule for $\tilde{r}\mathsf{GRAD}$ holds: $\tilde{r}\mathsf{GRAD}\ S(\mathtt{p}) = \mathsf{GRAD}\ Q(\mathtt{p})$. Of course, special care is necessary for the numerically-stable calculation of the intermediate densities, especially when elements of the diagonal matrix $\mathrm{diag}(\mathsf{DIFF}\ S(\mathtt{p}))$ are very small.

5.3 Isentropic Euler equations: operators rGRAD and DIVr

The techniques presented in the preceding sections are almost sufficient for a symmetry-preserving discretization of the isentropic Euler equations. Only the advection operator is not yet available. The advection operator is very similar to the divergence operator, on which it will be based. Here, we do not use the operators $\mathsf{DIV\tilde{r}}$ and $\tilde{r}\mathsf{GRAD}$, but the very similar operators DIVr and rGRAD, given by

$$\mathsf{rGRAD} := \sum_{\varepsilon=1/2,\ 3/2,\ \dots} \frac{\alpha(\varepsilon)}{2} \mathsf{GRAD1}(\varepsilon)\ \mathrm{diag} \left(\begin{array}{c} \mathtt{rho}_e(\varepsilon) \\ \mathtt{rho}_n(\varepsilon) \end{array} \right) \left(\begin{array}{c} -\mathsf{DIFFX}^\top(\varepsilon) \\ -\mathsf{DIFFY}^\top(\varepsilon) \end{array} \right),$$

$$\mathsf{DIVr} := -\mathsf{rGRAD}^*,$$

where the intermediate densities $\mathtt{rho}_e$ and $\mathtt{rho}_n$ are given by

$$\mathrm{diag} \left(\begin{array}{c} \mathtt{rho}_e(\varepsilon) \\ \mathtt{rho}_n(\varepsilon) \end{array} \right) := \mathrm{diag} \left(\begin{array}{c} \mathsf{DIFFX}^\top(\varepsilon)\ \mathtt{p} \\ \mathsf{DIFFY}^\top(\varepsilon)\ \mathtt{p} \end{array} \right) \mathrm{diag} \left(\begin{array}{c} \mathsf{DIFFX}^\top(\varepsilon)Q(\mathtt{p}) \\ \mathsf{DIFFY}^\top(\varepsilon)Q(\mathtt{p}) \end{array} \right)^{-1}.$$

5.4 Isentropic Euler equations: operator ADVEC

A symmetry-preserving approximation for the advection term, which also preserves momentum, is constructed using a 'splitting' strategy very similar to the one presented in [48], [47] and [35], and classical results of Tadmor [30]. For the construction of the advection operator ADVEC, we start with a discrete advection operator ADVEC_s that works on a scalar field $\mathtt{f}_v$. Unlike a 'normal' scalar field $\mathtt{f}$, which is sampled at the cell centers, the scalar field $\mathtt{f}_v = (\mathtt{f}_e^\top, \mathtt{f}_n^\top)^\top$ is sampled at the cell faces. The scalar advection operator ADVEC_s approximates the continuous advection operator: $(\mathsf{ADVEC}_s \mathtt{f}_v)_{c,i} \approx (\nabla \cdot \rho \vec{v} f)(\vec{x}_{c,i})$. This operator ADVEC_s has the following properties compared to ADVEC:

- **Input field of the operator**:

  ADVEC_s works on scalar fields sampled at the cell faces, and not, like ADVEC, on a staggered representation of a vector field;

- **Output field of the operator**:

  The result $\mathsf{ADVEC}_s\ \mathtt{f}_v$ lives on cell centers, and is not, like the result $\mathsf{ADVEC}\ \mathtt{v}$, a staggered representation of a vector field;

22

- **Applying the operator to a constant field**:

  When applied to a constant field, the operators ADVEC_s and ADVEC should both return the mass flux divergence $\mathsf{DIVr}\ \mathtt{v}$.

- **Left null space**:

  All constant fields should be in the null space of the adjoint operators ADVEC_s^* and ADVEC^*.

- **Meaningful adjoint**:

  ADVEC_s's adjoint is an accurate approximation of the adjoint of the continuous advection operator, like ADVEC's adjoint.

Because of the differences between ADVEC_s and ADVEC in the input and output spaces, certain interpolations and rotations are necessary which are not necessary in [48], [47] and [35].

The scalar advection operator ADVEC_s is defined as (cf. equation (26))

$$\mathsf{ADVEC}_s\ \mathbf{f}_v := \frac{1}{2}\mathrm{diag}(\mathtt{dVc})^{-1} \sum_{\varepsilon = 1/2,\ 3/2,\ \ldots} \big(\mathsf{DIFFX}(\varepsilon)\mathrm{diag}(\mathtt{rho}_e(\varepsilon))\mathrm{diag}(\mathbf{f}_e)\overline{\mathbf{F}}_e(\varepsilon)$$
$$+\ \mathsf{DIFFY}(\varepsilon)\mathrm{diag}(\mathtt{rho}_n(\varepsilon))\mathrm{diag}(\mathbf{f}_n)\overline{\mathbf{F}}_n(\varepsilon)\big)\,\alpha(\varepsilon). \qquad (29)$$

In this computation, $\overline{\mathbf{F}}_e$ and $\overline{\mathbf{F}}_n$ are computed using $\vec{v}$ and equations (24) and (25). This advection operator ADVEC_s returns values at the pressure points of the grid. It satisfies the left null space property and the 'advection of a constant function is the divergence'-property

$$\langle \mathtt{c1}, \mathsf{ADVEC}_s\ \mathbf{f}_v \rangle_c = 0 \quad \text{and} \quad \mathsf{ADVEC}_s\ \mathtt{c1} = \mathsf{DIVr}\ \mathtt{v}. \qquad (30)$$

The 'advection of a constant function is the divergence'-property is closely related to ADVEC's symmetry property in Table 2.

In combination with the interpolation matrices $\mathsf{C2E}$ and $\mathsf{C2N}$, the scalar advection operator can be used in models in which a scalar quantity is advected or convected, like temperature, or the concentration of a dissolved substance. The models discussed in this paper do not have such an advected/convected quantity.

The advection needed in the momentum equation of the isentropic Euler equations is not applied to a scalar field $\mathbf{f}_v$, but to a vector field $\mathbf{w}$. This is done by applying the scalar advection operator to the vector field's components separately. To make sure the operator is momentum preserving, this should not be done in the local grid orientation, but in the Cartesian orientation. The discretization process below is described in detail for 2D for simplicity, but all steps have also been worked out for 3D.

1. cell-face interpolation and transformation to Cartesian orientation.

   The interpolation operators $\mathsf{E2N}$ and $\mathsf{N2E}$ of Section 5.2.1 are needed to calculate the complete vectors, represented by $\mathtt{wx}$, $\mathtt{wy\_at\_e}$, $\mathtt{wx\_at\_n}$ and $\mathtt{wy}$. Next, the vector field is transformed into the Cartesian grid. As an example, the vector field $\vec{w}_{e,i} = \vec{r}_x(\vec{x}_{e,i})\mathtt{wx}_i + \vec{r}_y(\vec{x}_{e,i})\mathtt{wy\_at\_e}_i$ is written as follows:

   $$\vec{w}_{e,i} = \begin{pmatrix} \mathtt{rxx\_at\_e}_i \\ \mathtt{rxy\_at\_e}_i \end{pmatrix} \mathtt{wx}_i + \begin{pmatrix} \mathtt{ryx\_at\_e}_i \\ \mathtt{ryy\_at\_e}_i \end{pmatrix} \mathtt{wy\_at\_e}_i = \begin{pmatrix} \mathtt{rxx\_at\_e}_i\mathtt{wx}_i + \mathtt{ryx\_at\_e}_i\mathtt{wy\_at\_e}_i \\ \mathtt{rxy\_at\_e}_i\mathtt{wx}_i + \mathtt{ryy\_at\_e}_i\mathtt{wy\_at\_e}_i \end{pmatrix}.$$

   This means that the component of $\vec{w}_{e,i}$ in the Cartesian direction $(1,0)$ equals

   $$\mathtt{w}_{e,(1,0),i} = (1,0) \cdot \vec{w}_{e,i} = \mathtt{rxx\_at\_e}_i\mathtt{wx}_i + \mathtt{ryx\_at\_e}_i\mathtt{wy\_at\_e}_i,$$

   such that

   $$\mathtt{w}_{e,(1,0)} = \mathrm{diag}(\mathtt{rxx\_at\_e})\mathtt{wx} + \mathrm{diag}(\mathtt{ryx\_at\_e})\mathtt{wy\_at\_e}.$$

Similarly, the representation of $\vec{w}_{n,i}$ in the Cartesian basis vector $(1,0)$ can be computed:

$$\mathtt{w}_{n,(1,0)} = \operatorname{diag}(\mathtt{rxx\_at\_n})\mathtt{wx\_at\_n} + \operatorname{diag}(\mathtt{ryx\_at\_n})\mathtt{wy}.$$

The scalar field $\mathtt{w}_{v,(1,0)} = (\mathtt{w}_{e,(1,0)}^\top, \mathtt{w}_{n,(1,0)}^\top)^\top$ combines the two results. This scalar field has values in the complete velocity grid, like the scalar field $\mathtt{f}_v$ to which the advection operator ADVEC_s was applied earlier. It contains the horizontal component of the vector field $\mathtt{w}$.

Using the same procedure, the vertical component is found:

$$\mathtt{w}_{v,(0,1)} = \left(\begin{array}{c} \mathtt{w}_{e,(0,1)} \\ \mathtt{w}_{n,(0,1)} \end{array} \right) := \left(\begin{array}{c} \operatorname{diag}(\mathtt{rxy\_at\_e})\mathtt{wx} + \operatorname{diag}(\mathtt{ryy\_at\_e})\mathtt{wy\_at\_e} \\ \operatorname{diag}(\mathtt{rxy\_at\_n})\mathtt{wx\_at\_n} + \operatorname{diag}(\mathtt{ryy\_at\_n})\mathtt{wy} \end{array} \right).$$

2. ADVEC_c : application of ADVEC_s to each component.

The advection operator ADVEC_s can be applied to $\mathtt{w}_{v,(1,0)}$ and $\mathtt{w}_{v,(0,1)}$, and the result will be the operator ADVEC_c (advection at pressure points):

$$\mathsf{ADVEC}_c \ \mathtt{w} := \left(\begin{array}{c} \mathsf{ADVEC}_s \ \mathtt{w}_{v,(1,0)} \\ \mathsf{ADVEC}_s \ \mathtt{w}_{v,(0,1)} \end{array} \right) := \left(\begin{array}{c} \mathtt{a}_{c,(1,0)} \\ \mathtt{a}_{c,(0,1)} \end{array} \right).$$

This operator approximates the continuous advection operator: $\vec{a}(\vec{x}_{c,i}) \approx ((\mathtt{a}_{c,(1,0)})_i, (\mathtt{a}_{c,(0,1)})_i)$. This operator satisfies the left null space property given by

$$\left\langle \mathtt{c1}, \mathtt{a}_{c,(1,0)} \right\rangle_c = \left\langle \mathtt{c1}, \mathtt{a}_{c,(0,1)} \right\rangle_c = 0,$$

and the 'advection of a constant function is the divergence'-property given by

$$\mathsf{ADVEC}_c(\mathtt{c10}, \mathtt{c01}) = \left(\begin{array}{cc} \mathsf{DIVr} \ \mathtt{v} & 0 \\ 0 & \mathsf{DIVr} \ \mathtt{v} \end{array} \right).$$

3. ADVEC_a: interpolation to the velocity grid and transformation to the curvilinear grid.

The result of the advection operator ADVEC_c consists of standard Cartesian components at the pressure points of the grid, whereas the advection operator is needed for the components in local grid orientation at the velocity grid points. For this, we need to interpolate the components and transform the results. Interpolation is done by applying the adjoint operators $\mathsf{E2C}^*$ and $\mathsf{N2C}^*$, such that for each grid point $\vec{x}_{c,i}$, we compute

$$\vec{a}(\vec{x}_{e,i}) \approx \left(\begin{array}{c} (\mathsf{E2C}^* \mathtt{a}_{c,(1,0)})_i \\ (\mathsf{E2C}^* \mathtt{a}_{c,(0,1)})_i \end{array} \right), \quad \vec{a}(\vec{x}_{n,i}) \approx \left(\begin{array}{c} (\mathsf{N2C}^* \mathtt{a}_{c,(1,0)})_i \\ (\mathsf{N2C}^* \mathtt{a}_{c,(0,1)})_i \end{array} \right).$$

Note that the interpolation from the cell centers to the cell faces is done with the adjoint interpolations $\mathsf{E2C}^*$ and $\mathsf{N2C}^*$ and not with the forward interpolations $\mathsf{C2E}$ and $\mathsf{C2N}$, as might have been expected. This is needed in order to satisfy the left null space property (32). In general, the adjoint of an accurate interpolation cannot be expected to be an accurate interpolation, but in the current case, the interpolations $\mathsf{E2C}$ and $\mathsf{N2C}$ are standard destaggering operations working on the computational grid, and their adjoints are accurate interpolations.

Finally, the results are transformed to the basis spanned by the local grid orientation vectors: inner products with $r_x(\vec{x}_{e,i})$ and $r_y(\vec{x}_{n,i})$ are taken. Doing so defines the advection operator ADVEC_a, which has almost all the properties listed in Table 2:

$$\mathsf{ADVEC}_a = \left(\begin{array}{cc} \operatorname{diag}(\mathtt{rxx\_at\_e}) \ \mathsf{E2C}^* & \operatorname{diag}(\mathtt{rxy\_at\_e}) \ \mathsf{E2C}^* \\ \operatorname{diag}(\mathtt{ryx\_at\_n}) \ \mathsf{N2C}^* & \operatorname{diag}(\mathtt{ryy\_at\_n}) \ \mathsf{N2C}^* \end{array} \right) \mathsf{ADVEC}_c. \tag{31}$$

In Section 5.2.1, the interpolations E2C and N2C were required to be exact for the constant vector fields c10 and c01. The construction of the advection operator ADVEC_a also requires products of such fields to be exact, so (see equation (9) for the definition of c10 and c01)

$$\mathsf{E2C} \ \texttt{rxx\_at\_e} = \texttt{rxx\_at\_c} \quad , \quad \mathsf{E2C} \ \texttt{rxy\_at\_e} = \texttt{rxy\_at\_c},$$
$$\mathsf{E2C} \ \mathrm{diag}(\texttt{rxx\_at\_e})\texttt{rxx\_at\_e} = \mathrm{diag}(\texttt{rxx\_at\_c})\texttt{rxx\_at\_c},$$
$$\mathsf{E2C} \ \mathrm{diag}(\texttt{rxx\_at\_e})\texttt{rxy\_at\_e} = \mathrm{diag}(\texttt{rxx\_at\_c})\texttt{rxy\_at\_c},$$
$$\mathsf{E2C} \ \mathrm{diag}(\texttt{rxy\_at\_e})\texttt{rxy\_at\_e} = \mathrm{diag}(\texttt{rxy\_at\_c})\texttt{rxy\_at\_c},$$

and similar for N2C.

This advection operator ADVEC_a also satisfies the left null space property. This can be seen by computing

$$\begin{aligned}
\langle \texttt{c10}, \mathsf{ADVEC}_a \ \texttt{w}\rangle_v = {} & \langle \texttt{rxx\_at\_e}, \mathrm{diag}(\texttt{rxx\_at\_e}) \ \mathsf{E2C}^*\mathsf{ADVEC}_s \ \texttt{w}_{v,(1,0)}\rangle_e \\
& + \langle \texttt{rxx\_at\_e}, \mathrm{diag}(\texttt{rxy\_at\_e}) \ \mathsf{E2C}^*\mathsf{ADVEC}_s \ \texttt{w}_{v,(0,1)}\rangle_e \\
& + \langle \texttt{ryx\_at\_n}, \mathrm{diag}(\texttt{ryx\_at\_n}) \ \mathsf{N2C}^*\mathsf{ADVEC}_s \ \texttt{w}_{v,(1,0)}\rangle_n \\
& + \langle \texttt{ryx\_at\_n}, \mathrm{diag}(\texttt{ryy\_at\_n}) \ \mathsf{N2C}^*\mathsf{ADVEC}_s \ \texttt{w}_{v,(0,1)}\rangle_n \\
= {} & \langle \mathrm{diag}(\texttt{rxx\_at\_c})\texttt{rxx\_at\_c} + \mathrm{diag}(\texttt{ryx\_at\_c})\texttt{ryx\_at\_c}, \mathsf{ADVEC}_s \ \texttt{w}_{v,(1,0)}\rangle_c \\
& + \langle \mathrm{diag}(\texttt{rxy\_at\_c})\texttt{rxx\_at\_c} + \mathrm{diag}(\texttt{ryy\_at\_c})\texttt{ryx\_at\_c}, \mathsf{ADVEC}_s \ \texttt{w}_{v,(0,1)}\rangle_c.
\end{aligned}$$

Due to orthonormality of the local grid orientation, we have

$$\begin{aligned}
\begin{pmatrix} \texttt{rxx\_at\_c} & \texttt{rxy\_at\_c} \\ \texttt{ryx\_at\_c} & \texttt{ryy\_at\_c} \end{pmatrix} \begin{pmatrix} \texttt{rxx\_at\_c} & \texttt{ryx\_at\_c} \\ \texttt{rxy\_at\_c} & \texttt{ryy\_at\_c} \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \texttt{rxx\_at\_c} & \texttt{ryx\_at\_c} \\ \texttt{rxy\_at\_c} & \texttt{ryy\_at\_c} \end{pmatrix} \begin{pmatrix} \texttt{rxx\_at\_c} & \texttt{rxy\_at\_c} \\ \texttt{ryx\_at\_c} & \texttt{ryy\_at\_c} \end{pmatrix},
\end{aligned}$$

such that

$$\texttt{rxx\_at\_c}_i^2 + \texttt{ryx\_at\_c}_i^2 = 1, \quad \texttt{rxx\_at\_c}_i\texttt{rxy\_at\_c}_i + \texttt{ryx\_at\_c}_i\texttt{ryy\_at\_c}_i = 0.$$

Therefore,

$$\langle \texttt{c10}, \mathsf{ADVEC}_a \ \texttt{w}\rangle_v = \langle \texttt{c1}, \mathsf{ADVEC}_s \ \texttt{w}_{v,(1,0)}\rangle_c + \langle 0, \mathsf{ADVEC}_s \ \texttt{w}_{v,(0,1)}\rangle_c = 0, \qquad (32)$$

and similar for $\langle \texttt{c01}, \mathsf{ADVEC}_a \ \texttt{w}\rangle_v$.

The advection operator ADVEC_a also has the following 'advection of a constant function is the divergence'-property:

$$\mathsf{ADVEC}_a \ \texttt{c} = \mathrm{diag}\left(\left(\begin{array}{c} \mathsf{E2C}^* \\ \mathsf{N2C}^* \end{array}\right) \mathsf{DIVr} \ \texttt{v}\right) \texttt{c},$$

for any discrete vector field c that is the discrete representation of a constant vector field.

4. ADVEC: construction of the symmetry-preserving advection operator.

The only property from Table 2 that the advection operator ADVEC_a does not have, is the symmetry property. We obtain the advection operator ADVEC that has the symmetry property (as well as the left null space property) by combining the operator ADVEC_a and its adjoint [35]:

$$\mathsf{ADVEC} := \frac{1}{2}\left(\mathsf{ADVEC}_a - \mathsf{ADVEC}_a^* + \mathrm{diag}\left(\left(\begin{array}{c} \mathsf{E2C}^* \\ \mathsf{N2C}^* \end{array}\right) \mathsf{DIVr} \ \texttt{v}\right)\right).$$

The advection operator ADVEC has precisely the symmetry property for the advection operator, provided that the interpolation operator $\mathsf{Interp}_{v \leftarrow c}$ used in the table is given by

$$\mathsf{Interp}_{v \leftarrow c} := \left(\begin{array}{c} \mathsf{E2C}^* \\ \mathsf{N2C}^* \end{array}\right).$$

25

Negative densities may be the result of this interpolation, in cases where the density rapidly approaches zero, for instance in cases of wetting and drying, and this may affect the accuracy and stability of the numerical scheme. In such cases, the order of interpolation and discretization may be locally reduced (in a symmetry-preserving way). Alternatively, techniques like artificial porosity may be used to regularize the solution [36]. Such situations did not occur in the examples shown in this paper, and the techniques mentioned (local order reduction and artificial porosity) are left outside the scope of this paper.

All required symmetry-preserving discrete operators are now defined, and we can investigate the performance of these operators for the three models. This will be done in the next section.

6   Numerical results

In this section, the numerical results for the three different models are studied. The calculations for the linear-wave equations are done both with the Galerkin-type approach (Section 5.1) and with the finite-difference technique (Section 5.2). This makes it possible to compare the results and computation times. The more advanced models for compressible waves and shallow waves are only investigated with the finite-difference technique.

All calculations are done both on a uniform and on a periodic, non-orthogonal, curvilinear mesh. The periodicity of the curvilinear mesh gives us the opportunity to study symmetry preservation without focusing on boundary conditions (Section 4.2). An example of the uniform and curvilinear grids used for 20×20 grid points is given in Figure 4.

(a) Uniform mesh   (b) Curvilinear mesh   (c) Detail of curvilinear mesh

Figure 4: 2D grids used in the experiments. An impression of the periodicity is given by extending the mesh on each boundary. The detail shows that the grid is non-orthogonal: in parts of the grid, the angle between grid lines are as small as 15°.

6.1   Linear-wave equations

Choosing $p = \rho$ ($c = 1$ in equation (1)), an exact solution for the linear-wave equation is discussed in Section 2. Since the propagation speed $V_+ = 1$ only has one value, the solution is a traveling wave, which does not change its shape, but only its position, traveling in 'southeast' direction through the infinite periodic domain, given by

$$p(x,y,t) = \exp\left(-\frac{2}{9\pi^2}\sin^2(\sqrt{2}\pi(x-y-\sqrt{2}t))\right) \quad , \quad \vec{v}(x,y,t) = \frac{1}{2}\sqrt{2}\begin{pmatrix} 1 \\ -1 \end{pmatrix}p(x,y,t). \quad (33)$$

In fact, the solution on the uniform mesh is a 1D function of $x - y$, so the results at time $T = 10$ can be represented in a 1D plot (Figure 5). The numerical approximation, shown in red in the figure,

is close to the exact solution, shown in blue. The relative errors of the Galerkin-type approximations are shown in Table 4 for different grid sizes and interpolation orders, and for the finite-difference approach they are shown in Table 6. It is seen that the solution becomes more accurate as more grid points are used and/or higher-order interpolation are used. The order of the approximation is equal to the interpolation order. Note that the results for our curvilinear grid are about the same as for the uniform grid. This is expected, since the exact solution, as well as the grid transformation $\vec{X}$ are infinitely smooth.

The effect of symmetry preservation is, that, assuming exact time integration, the total discrete mass M, momentum $\vec{\text{M}}$ and total energy E are constant up to machine precision, even in the most inaccurate solutions shown in Table 4. In Tables 5 and 7, the mass, momentum and energy losses are given for a fourth-order discretization on a 20×20 grid. Independently of the time-integration tolerance used, the mass is always conserved up to a very high precision, which depends on how accurately the matrix entries were calculated (using Richardson extrapolation of the composite trapezoidal rule). For the finite-difference models, where the discrete operators require no numerical integration, the error for mass and momentum is really in the order of machine accuracy. Considering energy conservation, the error depends on the accuracy of the time integration.

For linear models, the implicit midpoint rule and its higher-order Gauss-Runge-Kutta generalizations will conserve energy (in fact all quadratic and linear invariants) up to machine precision. For the nonlinear models, dedicated time-integration methods are needed. The current paper leaves time-integration methods outside the scope, because a standard time-integration method is used with such a small time step, that the time-integration errors are negligible.

Figure 5: Linear-wave equation: exact solution and numerical approximation of the density on a uniform mesh at $T = 10$ (1D solution). The interpolation order is 4, 20×20 grid points.

Discrete mass, momentum and total energy are conserved up to machine precision, even in the most inaccurate solutions. The losses for different time-integration tolerances for the Galerkin-type approach are shown in Table 5. Similar results obtained in the finite-difference approach are shown in Table 7.

Because both the Galerkin-type approach and the finite-difference technique are available for the linear-wave equations, their relative computational performance can be compared. The Fortran code was compiled using gfortran version 7.3.0, and simulations were run on a desktop computer running Ubuntu 18.04 on an Intel i5-7400 CPU with 4 cores. Both methods use the same shared memory parallelization technique (OpenMP). We compare the results for interpolation (and derivative) order 8 and 40×40 grid points.

The Galerkin-type approach needs 118 seconds for the computation, thereby doing 21016 time steps and computing 27751 time derivatives. This means that this implementation computes 178 time steps per second, and 234 time-derivative evaluations per second.

On the other hand, the finite-difference approach needs 14 seconds for the total computation, in which it computes 26901 time steps and 31199 time derivatives. This implementation is able to compute 1922 time steps per second, and 2230 evaluations per second. We clearly see that

27

the finite-difference approach is about ten times faster than the Galerkin-type approach. This also corresponds to the theory (Sections 5.1.2 and 5.2.1): for this 2D problem with 8th order interpolation, the Galerkin-type approach results in 480 nonzeros per matrix row, whereas the finite-difference approach only uses 48 nonzeros per grid point (a factor 10 difference).

Table 4: Relative errors (in 2-norm) for pressure in the linear-wave equations (Galerkin-type approach) at time $T = 10$, for various interpolation orders, using reltol equal to `1e-11` in `lsode`. For 160×160 grid points on the uniform grid and 8th order, reltol=`1e-13` is used.

Uniform grid

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	1.12e+00		7.64e-01		1.68e-01		4.71e-01	
20×20	1.43e+00	−0.35	4.10e-01	0.90	1.43e-01	0.23	4.49e-02	3.39
40×40	4.95e-01	1.53	7.85e-02	2.38	6.86e-03	4.38	7.82e-04	5.84
80×80	4.02e-01	0.30	6.34e-03	3.63	1.22e-04	5.82	3.55e-06	7.78
160×160	1.46e-01	1.46	4.01e-04	3.98	1.93e-06	5.98	1.43e-08	7.96

Curvilinear grid

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	8.23e-01		1.06e+00		6.83e-01		3.57e+00	
20×20	9.76e-01	−0.25	2.53e-01	2.07	4.29e-01	0.67	1.73e-01	4.36
40×40	8.67e-01	0.17	1.85e-01	0.45	4.61e-02	3.22	1.73e-02	3.33
80×80	5.22e-01	0.73	2.31e-02	3.00	1.49e-03	4.95	1.60e-04	6.75
160×160	2.03e-01	1.36	1.57e-03	3.88	2.51e-05	5.89	7.17e-06	7.80

Table 5: Relative losses for linear-wave equations (Galerkin-type approach) at time $T = 10$, for varying tolerances reltol in the time-integration method `lsode`. Fourth-order discretization on a 20×20 grid.

	Uniform grid			Curvilinear grid		
reltol	mass loss	mom. loss	energy loss	mass loss	mom. loss	energy loss
1e-07	5.29e-14	1.20e-13	9.94e-07	4.14e-15	6.33e-15	1.60e-06
1e-08	5.27e-14	1.20e-13	6.97e-08	2.83e-15	4.78e-15	1.56e-08
1e-09	5.16e-14	1.21e-13	6.39e-08	3.92e-15	6.57e-15	3.25e-09
1e-10	5.29e-14	1.20e-13	1.15e-09	3.92e-15	5.58e-15	1.50e-11
1e-11	5.36e-14	1.17e-13	5.54e-11	2.61e-15	7.73e-15	1.87e-11
1e-12	5.23e-14	1.18e-13	4.74e-11	1.96e-15	4.43e-15	5.84e-12

6.2 Compressible-wave equations

Unlike the solutions in the previous numerical examples, the solution of the compressible-wave equations has variable propagation speed. This means that the crests of the wave travel faster than the troughs. This effect causes the wave to steepen over time, and finally to become discontinuous. In Figure 6, the initial solution and the numerical approximation at the time t_N (when the solution becomes discontinuous) are shown. Obviously, the solution becomes more difficult to approximate for a numerical method as the moment t_N approaches. However, we see that away from the shock, the approximations are still accurate. Because the solution becomes increasingly difficult to approximate

Table 6: Relative errors (in 2-norm) for pressure in the linear-wave equations (finite-difference approach) at time $T = 10$, for various (same) orders for derivatives and interpolations, using reltol equal to `1e-11` in `lsode`. For 160×160 grid points and 8th order, reltol=`1e-13` is used.

Uniform grid

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	9.29e-01		4.52e-01		1.33e-01		1.62e-01	
20×20	6.22e-01	0.58	8.90e-02	2.34	1.88e-02	2.82	5.54e-03	4.87
40×40	2.58e-01	1.27	7.68e-03	3.53	4.14e-04	5.51	3.59e-05	7.27
80×80	7.67e-02	1.75	4.91e-04	3.97	6.77e-06	5.94	1.49e-07	7.92
160×160	1.96e-02	1.97	3.08e-05	4.00	9.79e-08	6.11	6.00e-10	7.96

Curvilinear grid

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	1.14e+00		5.08e-01		2.92e-01		4.76e-01	
20×20	4.93e-01	1.21	1.65e-01	1.62	6.65e-02	2.13	2.29e-02	4.37
40×40	3.31e-01	0.57	2.13e-02	2.95	2.58e-03	4.69	5.36e-04	5.42
80×80	1.13e-01	1.55	1.43e-03	3.90	4.47e-05	5.85	2.60e-06	7.69
160×160	3.03e-02	1.91	9.01e-05	3.99	7.17e-07	5.96	1.05e-08	7.95

Table 7: Relative losses for linear-wave equations (finite-difference approach) at time $T = 10$, for varying tolerances reltol in the time-integration method `lsode`. Fourth-order discretization on a 20×20 grid.

	Uniform grid			Curvilinear grid		
reltol	mass loss	mom. loss	energy loss	mass loss	mom. loss	energy loss
1e-07	2.18e-16	2.39e-15	1.94e-06	2.18e-16	1.36e-14	3.29e-08
1e-08	1.09e-15	1.57e-15	1.22e-06	2.18e-16	1.43e-14	4.61e-09
1e-09	4.35e-16	1.12e-15	1.78e-09	4.35e-16	1.15e-14	4.33e-10
1e-10	1.09e-15	1.66e-15	2.98e-09	1.52e-15	1.10e-14	7.85e-11
1e-11	0.00e+00	9.86e-16	1.35e-09	6.53e-16	1.52e-14	1.28e-11
1e-12	2.18e-16	1.66e-15	3.16e-11	1.31e-15	1.27e-14	8.80e-13

as time $t = t_N$ approaches, results are shown for $t = 1.09 = t_N/2$ and for $t = 2.18 = t_N$ in Table 8. Halfway the simulation, the convergence order approaches the theoretical order; for the final time, only order 1 can be achieved.

Table 9 shows that even the most inaccurate solutions at the final time conserve discrete mass, momentum and energy up to machine precision. However, the physically-relevant solution should have lower total energy after the shock has appeared. Therefore, numerical viscosity can be added, for instance in the form of flux limiters, but such techniques are beyond the scope of this paper.

6.3 Shallow-water equations

Like the compressible-wave equations, the variable propagation speed leads to a steepening wave in the exact solution. The results at time $t = t_N$ (the moment that the solution becomes discontinuous) are visualized in Figure 7. The accuracies at time $t = t_N/2$ are shown in Table 10: the accuracy orders are approaching the theoretical order. At the final time, only an order 1 can be achieved. Mass, momentum and energy are again conserved up to machine precision, as shown in Table 11.

Table 8: Relative errors (in 2-norm) for the densities in the compressible-wave equations, for various (same) orders for derivatives and interpolations, using reltol equal to `1e-11` in `lsode`.

Uniform grid, $t = t_N/2$

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	3.03e-01		6.65e-02		3.52e-02		2.48e-02	
20×20	1.04e-01	1.54	1.24e-02	2.42	5.29e-03	2.74	3.49e-03	2.83
40×40	3.05e-02	1.77	1.32e-03	3.24	3.13e-04	4.08	1.53e-04	4.51
80×80	7.98e-03	1.93	9.50e-05	3.79	8.55e-06	5.19	2.29e-06	6.06
160×160	2.01e-03	1.99	6.10e-06	3.96	1.59e-07	5.75	1.56e-08	7.20

Curvilinear grid, $t = t_N/2$

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	4.35e-01		1.38e-01		9.38e-02		7.94e-02	
20×20	1.55e-01	1.48	2.95e-02	2.23	1.67e-02	2.49	1.26e-02	2.65
40×40	4.75e-02	1.71	4.67e-03	2.66	1.90e-03	3.14	1.22e-03	3.37
80×80	1.26e-02	1.91	4.14e-04	3.49	8.85e-05	4.43	4.10e-05	4.89
160×160	3.15e-03	2.00	2.75e-05	3.91	2.12e-06	5.39	5.13e-07	6.32

Uniform grid, $t = t_N$

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	5.42e-01		2.62e-01		2.20e-01		1.96e-01	
20×20	3.25e-01	0.74	1.44e-01	0.86	1.16e-01	0.92	1.07e-01	0.87
40×40	1.82e-01	0.84	7.63e-02	0.92	5.92e-02	0.97	5.24e-02	1.03
80×80	1.04e-01	0.82	4.06e-02	0.91	3.14e-02	0.91	2.81e-02	0.90
160×160	6.09e-02	0.77	2.10e-02	0.95	1.49e-02	1.08	1.28e-02	1.14

Curvilinear grid, $t = t_N$

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	7.41e-01		3.19e-01		2.24e-01		2.15e-01	
20×20	4.13e-01	0.84	1.83e-01	0.80	1.43e-01	0.64	1.30e-01	0.73
40×40	2.33e-01	0.83	1.02e-01	0.84	8.10e-02	0.82	7.29e-02	0.83
80×80	1.39e-01	0.75	5.98e-02	0.77	4.63e-02	0.81	4.10e-02	0.83
160×160	8.34e-02	0.74	3.36e-02	0.83	2.51e-02	0.88	2.20e-02	0.90

(a) Initial solution

(b) Final-time approximation

Figure 6: Exact solution and numerical approximation for the compressible-wave equations on a uniform, 40×40 mesh (1D solution), with interpolation order 4. The final-time approximation belongs to the moment when the solution becomes discontinuous.

Table 9: Relative losses for the compressible-wave equations, for varying tolerances reltol in the time-integration method `lsode`, at the time when the solution becomes discontinuous. Fourth-order discretization on a 20×20 grid.

	Uniform grid			Curvilinear grid		
reltol	mass loss	mom. loss	energy loss	mass loss	mom. loss	energy loss
1e-05	6.82e-16	1.85e-14	6.53e-09	1.02e-15	1.67e-13	6.41e-10
1e-06	3.41e-16	1.94e-14	7.99e-10	1.19e-15	1.69e-13	1.57e-10
1e-07	1.02e-15	2.14e-14	4.18e-11	8.52e-16	1.64e-13	6.64e-12
1e-08	3.41e-16	2.39e-14	1.68e-13	0.00e+00	1.68e-13	6.49e-13
1e-09	6.82e-16	1.75e-14	3.30e-13	8.52e-16	1.69e-13	5.20e-14
1e-10	1.70e-16	1.81e-14	3.33e-14	5.11e-16	1.66e-13	1.09e-15
1e-11	3.41e-16	1.97e-14	2.90e-15	2.56e-15	1.66e-13	1.45e-15
1e-12	1.02e-15	2.10e-14	1.81e-16	1.88e-15	1.66e-13	1.81e-16

Figure 7: Shallow-water equations: exact solution and numerical approximation at $t = t_N$ on a uniform mesh (1D solution). The interpolation order is 4, and the meshes contain 80×80 grid points.

Table 10: Relative errors (in 2-norm) for the densities in the shallow-water equations at time $t = t_N/2$, for various (same) orders for derivatives and interpolations, using reltol equal to `1e-11` in `lsode`.

Uniform grid

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	7.07e-02		2.23e-02		1.33e-02		1.10e-02	
20×20	2.62e-02	1.43	4.60e-03	2.28	2.31e-03	2.53	1.65e-03	2.74
40×40	8.16e-03	1.68	6.00e-04	2.94	1.89e-04	3.61	1.05e-04	3.97
80×80	2.21e-03	1.89	4.81e-05	3.64	6.40e-06	4.89	2.03e-06	5.69
160×160	5.61e-04	1.98	3.15e-06	3.93	1.24e-07	5.69	1.46e-08	7.12

Curvilinear grid

	order=2		order=4		order=6		order=8	
	error	order	error	order	error	order	error	order
10×10	8.48e-02		4.35e-02		3.51e-02		3.49e-02	
20×20	3.34e-02	1.34	1.17e-02	1.90	7.73e-03	2.18	6.16e-03	2.50
40×40	1.25e-02	1.42	2.19e-03	2.42	1.06e-03	2.87	7.32e-04	3.07
80×80	3.71e-03	1.75	2.34e-04	3.22	6.26e-05	4.08	3.12e-05	4.55
160×160	9.73e-04	1.93	1.68e-05	3.80	1.64e-06	5.26	3.99e-07	6.29

7 Conclusion

This paper describes the construction of an arbitrary-order symmetry-preserving finite-difference technique on structured curvilinear staggered grids, offering flexibility and accuracy of the numerical approximations. The numerical examples presented in this work show that the method leads to results in which a high accuracy can be obtained, while the discrete mass, momentum and energy are all preserved. Also, the computation times are investigated for the linear-wave model, and the method is shown to be faster than when using a symmetry-preserving Galerkin-type approach. We have shown that the finite-difference method is able to handle difficult operators such as the advection operator in the shallow-water equations.

This paper does not address the application of energy-conserving (symplectic) time integration. This is left outside the scope of this paper, and so are the discussion of boundary conditions and the use of numerical viscosity (for instance using flux limiters), which is necessary when shocks occur in the solution. Finally, future work includes handling local grid refinements.

References

[1] E. D. Batista and J. E. Castillo. Mimetic Schemes on Non-Uniform Structured Meshes. *Electronic Transactions on Numerical Analysis*, 34:152–162, 2009.

[2] G. A. Blaisdell, E. T. Spyropoulos, and J. H. Qin. The effect of the formulation of nonlinear terms on aliasing errors in spectral methods. *Applied Numerical Mathematics*, 21:207–219, 1996.

Table 11: Relative losses for the shallow-water equations, for varying tolerances reltol in the time-integration method `lsode`, at the time when the solution becomes discontinuous. Fourth-order discretization on a 20×20 grid.

reltol	Uniform grid			Curvilinear grid		
	mass loss	mom. loss	energy loss	mass loss	mom. loss	energy loss
1e-05	8.07e-16	1.49e-15	1.16e-05	2.02e-16	5.40e-14	3.35e-06
1e-06	1.41e-15	9.39e-16	8.07e-07	4.04e-16	5.55e-14	4.85e-08
1e-07	0.00e+00	7.04e-16	2.07e-07	8.07e-16	5.56e-14	1.59e-09
1e-08	0.00e+00	1.37e-15	5.05e-09	2.02e-16	5.55e-14	7.83e-10
1e-09	1.01e-15	5.25e-16	1.36e-10	4.04e-16	5.44e-14	1.05e-11
1e-10	0.00e+00	1.11e-15	3.91e-11	0.00e+00	5.68e-14	3.12e-12
1e-11	6.06e-16	9.39e-16	2.12e-12	1.01e-15	5.48e-14	5.79e-13
1e-12	4.04e-16	8.30e-16	5.61e-13	6.06e-16	5.53e-14	4.02e-14

[3] J. Blanco, O. Rojas, C. Chacón, J. M. Guevara-Jordan, and J. Castillo. Tensor formulation of 3-D mimetic finite differences and applications to elliptic problems. *Electronic Transactions on Numerical Analysis*, 45:457–475, 2016.

[4] P. B. Bochev and J. M. Hyman. Principles of Mimetic Discretizations of Differential Operators. In D. N. Arnold, P. B. Bochev, R. B. Lehoucq, R. A. Nicolaides, and M. Shashkov, editors, *Compatible Spatial Discretizations*, volume 142 of *The IMA Volumes in Mathematics and its Applications*, pages 89–119. Springer New York, 2006.

[5] F. Brezzi and A. Buffa. Innovative mimetic discretizations for electromagnetic problems. *Journal of Computational and Applied Mathematics*, 234(6):1980–1987, 2010.

[6] F. Brezzi, A. Buffa, and G. Manzini. Mimetic scalar products of discrete differential forms. *Journal of Computational Physics*, 257:1228–1259, 2014.

[7] J. E. Castillo and R. D. Grone. A Matrix Analysis Approach to Higher-Order Approximations for Divergence and Gradients Satisfying a Global Conservation Law. *SIAM Journal on Matrix Analysis and Applications*, 25(1):128–142, 2003.

[8] J. E. Castillo and G. F. Miranda. *Mimetic Discretization Methods*. CRC Press, Taylor & Francis Group, 2013.

[9] J. Corbino and J. E. Castillo. High Order Mimetic Finite Difference Operators Satisfying a Gauss Divergence Theorem. *Journal of Applied & Computational Mathematics*, 7(1), 2018.

[10] J. de la Puente, M. Ferrer, M. Hanzich, J. E. Castillo, and J. M. Cela. Mimetic seismic wave modeling including topography on deformed staggered grids. *GEOPHYSICS*, 79(3):T125–T141, 2014.

[11] D. C. Del Rey Fernández, J. E. Hicken, and D. W. Zingg. Review of Summation-By-Parts Operators with Simultaneous Approximation Terms for the Numerical Solution of Partial Differential Equations. *Computers and Fluids*, 95:171 – 196, 2014.

[12] G. J. Gassner, A. R. Winters, and D. A. Kopriva. Split form nodal discontinuous Galerkin schemes with summation-by-parts property for the compressible Euler equations. *Journal of Computational Physics*, 327:39–66, 2016.

[13] A. C. Hindmarsh. ODEPACK, A Systematized Collection of ODE Solvers. *Scientific Computing*, 1:55–64, 1983. IMACS Transactions on Scientific Computation.

[14] A. N. Hirani. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology, 2003.

[15] M. Kindelan, A. Kamel, and P. Sguazzero. On the construction and efficiency of staggered numerical differentiators for the wave equation. *GEOPHYSICS*, 55(1):107–110, 1990.

[16] J. C. Kok. A symmetry and dispersion-relation preserving high-order scheme for aeroacoustics and aerodynamics. Technical Report NLR-TP-2006-525, National Aerospace Laboratory NLR, 2006.

[17] J. C. Kok. A high-order low-dispersion symmetry-preserving finite-volume method for compressible flow on curvilinear grids. *Journal of Computational Physics*, 228:6811–6832, 2009.

[18] J. Kreeft, A. Palha, and M. Gerritsma. Mimetic framework on curvilinear quadrilaterals of arbitrary order, 2011. arXiv:1111.4304 [math.NA].

[19] O. Lehmkuhl, R. Borrell, I. Rodríguez, C. D. Pérez-Segarra, and A. Oliva. Assessment of the symmetry-preserving regularization model on complex flows using unstructured grids. *Computers & Fluids*, 60:108–116, 2012.

[20] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, New York, sixth edition, 2002.

[21] K. Lipnikov, G. Manzini, and M. Shashkov. Mimetic finite difference method. *Journal of Computational Physics*, 257:1163–1227, 2014.

[22] Y. Liu and M. K. Sen. An implicit staggered-grid finite-difference method for seismic modelling. *Geophysical Journal International*, 179:459–474, 2009.

[23] G. T. Oud, D. R. van der Heul, C. Vuik, and R. A. W. M. Henkes. A fully conservative mimetic discretization of the Navier-Stokes equations in cylindrical coordinates with associated singularity treatment. *Journal of Computational Physics*, 325:314–337, 2016.

[24] A. Palha and M. Gerritsma. A mass, energy, enstrophy and vorticity conserving (MEEVC) mimetic spectral element discretization for the 2D incompressible Navier-Stokes equations. *Journal of Computational Physics*, 328:200–220, 2017.

[25] N. Robidoux and S. L. Steinberg. A Discrete Vector Calculus in Tensor Grids. *Computational Methods in Applied Mathematics*, 11:23 – 66, 2011.

[26] W. Rozema, R. W. C. P. Verstappen, J. C. Kok, and A. E. P. Veldman. Discretizations and Regularization Models for Compressible Flow that Preserve the Skew-Symmetry of Convective Transport. In E. Oñate, X. Oliver, and A. Huerta, editors, *Proceedings of the jointly organized WCCM XI - ECCM V - ECFD VI*, pages 4652–4663, 2014.

[27] W. Rozema, R. W. C. P. Verstappen, J. C. Kok, and A. E. P. Veldman. A Symmetry-Preserving Discretization and Regularization Subgrid Model for Compressible Turbulent Flow. In J. Fröhlich, H. Kuerten, B.J. Geurts, and V. Armenio, editors, *Direct and Large-Eddy Simulation IX*, volume 20 of *ERCOFTAC Series*, pages 319–325, 2015.

[28] F. Solano-Feo, J. Guevara-Jordan, C. González-Ramirez, O. Rojas-Ulacio, and B. Otero-Calvinyo. Modeling seismic wave propagation using staggered-grid mimetic finite differences. *Bulletin of Computational Applied Mathematics*, 5(2):9–28, 2017.

[29] S .L. Steinberg. Explicit Time Mimetic Discretizations, 2016. arXiv:1605.08762v4 [math.NA].

[30] E. Tadmor. Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems. *Acta Numerica*, 12:451–512, 2003.

[31] T. Tarhasaari, L. Kettunen, and A. Bossavit. Some realizations of a discrete Hodge operator: A reinterpretation of finite element techniques. *IEEE Transactions on Magnetics*, 35(3):1494–1497, 1999.

[32] F. X. Trias, O. Lehmkuhl, A. Oliva, C. D. Pérez-Segarra, and R. W. C. P. Verstappen. Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids. *Journal of Computational Physics*, 258:246–267, 2014.

[33] F. X. Trias and R. W. C. P. Verstappen. On the construction of discrete filters for symmetry-preserving regularization models. *Computers & Fluids*, 40:139–148, 2011.

[34] P. van Beek, R. R. P. van Nooyen, and P. Wesseling. Accurate Discretization of Gradients on Non-uniform Curvilinear Staggered Grids. *Journal of Computational Physics*, 117(2):364–367, 1995.

[35] B. van 't Hof and A. E. P. Veldman. Mass, momentum and energy conserving (MaMEC) discretizations on general grids for the compressible Euler and shallow water equations. *Journal of Computational Physics*, 231:4723–4744, 2012.

[36] B. van 't Hof and E. A. H. Vollebregt. Modelling of wetting and drying of shallow water using artificial porosity. *International Journal for Numerical Methods in Fluids*, 48(11):1199–1217, 2005.

[37] B. van 't Hof and M. J. Vuik. Symmetry-preserving discretizations of arbitrary order on structured curvilinear grids, 2017. arXiv:1710.07149 [math.NA].

[38] B. van 't Hof and M. J. Vuik. Derivations of continuous and discrete energy equations in wave and shallow-water equations, 2019. arXiv:1905.04085 [math.NA].

[39] A. E. P. Veldman and K.-W. Lam. Symmetry-preserving upwind discretization of convection on non-uniform grids. *Applied Numerical Mathematics*, 58:1881–1891, 2008.

[40] A. E. P. Veldman and K. Rinzema. Playing with nonuniform grids. *Journal of Engineering Mathematics*, 26:119–130, 1992.

[41] R. W. C. P. Verstappen and R. M. van der Velde. Symmetry-preserving discretization of heat transfer in a complex turbulent flow. *Journal of Engineering Mathematics*, 54:299–318, 2006.

[42] R. W. C. P. Verstappen and A. E. P. Veldman. Spectro-consistent discretization of Navier-Stokes: a challenge to RANS and LES. *Journal of Engineering Mathematics*, 34:163–179, 1998.

[43] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187:343–368, 2003.

[44] M. Vetterli, J. Kovačević, and V. K. Goyal. *Foundations of Signal Processing*. Cambridge University Press, 2014.

[45] M. J. Vuik and J. K. Ryan. Multiwavelet troubled-cell indicator for discontinuity detection of discontinuous Galerkin schemes. *Journal of Computational Physics*, 270:138–160, 2014.

[46] P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer-Verlag Berlin Heidelberg, 2001.

[47] N. Wintermeyer, A. R. Winters, G. J. Gassner, and D. A. Kopriva. An entropy stable nodal discontinuous Galerkin method for the two dimensional shallow water equations on unstructured curvilinear meshes with discontinuous bathymetry. *Journal of Computational Physics*, 340:200–242, 2017.

[48] N. Wintermeyer, A. R. Winters, G. J. Gassner, and T. Warburton. An entropy stable discontinuous Galerkin method for the shallow water equations on curvilinear meshes with wet/dry fronts accelerated by GPUs. *Journal of Computational Physics*, 375:447–480, 2018.

Code Availability

The source code used for the experiments presented in this article can be obtained from:

$$\text{https://gitlab.com/VORtechBV/mamec}$$