A fourth-order accurate adaptive solver for incompressible flow problems

van Hooft, J. Antoon; Popinet, Stéphane

DOI
10.1016/j.jcp.2022.111251

Publication date
2022

Document Version
Final published version

Published in
Journal of Computational Physics

Journal of Computational Physics 462 (2022) 111251

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A fourth-order accurate adaptive solver for incompressible flow problems

J. Antoon van Hooft [a,*], Stéphane Popinet [b]

[a] Department of Geoscience and Remote Sensing, Delft University of Technology, 2628 CN Delft, the Netherlands
[b] Institut Jean le Rond d'Alembert, Sorbonne Université, CNRS, 75005 Paris, France

ARTICLE INFO

Article history:
Received 23 September 2021
Received in revised form 7 April 2022
Accepted 20 April 2022
Available online 22 April 2022

Keywords:
Adaptive grid
Fourth order
Navier–Stokes
Flow solver

ABSTRACT

We present a numerical solver for the incompressible Navier–Stokes equations that combines fourth-order-accurate discrete approximations and an adaptive tree grid (i.e. *h*-refinement). The scheme employs a novel compact-upwind advection scheme and a 4th-order accurate projection algorithm whereby the numerical solution exactly satisfies the incompressibility constraint. Further, we introduce a new refinement indicator that is tailored to this solver. We show tests and examples to illustrate the consistency, convergence rate and the application for the adaptive solver. The combination of the solver scheme and the proposed grid adaptation algorithm result in fourth-order convergence rates whilst only tuning a single grid-refinement parameter. The speed performance is benchmarked against a well-established second-order accurate adaptive solver alternative. We conclude that the present 4th order solver is an efficient design for problems with strong localization in the spatial-temporal domain and where a high degree of convergence for the solution statistics is desired.

1. Introduction

Computational analysis of flow problems has become an important piece of equipment in the toolbox for science and engineering [1]. The problem of finding approximate solutions to the equations for fluid flow may be reduced to evaluating a numerical recipe, following from a suitable discretization for the problem. Apart from the prerequisites for the stability and consistency of these formulations, the desire to have a practically applicable solver warrants the method to be time efficient in relation to the fidelity of the approximated solution. Therefore, this works aims to contribute to this purpose by studying a combination of higher-order accurate discrete approximations with adaptive mesh refinement techniques. More specifically, we present and benchmark a 4th-order accurate solver for the Navier–Stokes equations for high Reynolds number incompressible flows on adaptive quadtree grids (or octree-structured grids in 3D). Special emphasis is placed on testing the convergence of the solution (and its statistics) by tuning a single grid-adaptation parameter which is introduced herein.

High Reynolds number flows are characterized by non-linear dynamics, which can give rise to a multi-scale flow structure. Since the seminal works of Marsha Berger [2,3], adaptive grids have proven to be able to efficiently capture the evolution of small-scale flow features that are embedded in a comparatively large spatial-temporal domain (e.g. [4–6]). The ability to focus the computational resources towards the regions in space and time that require it most is an important

* Corresponding author.
E-mail address: j.a.vanhooft@tudelft.nl (J.A. van Hooft).

https://doi.org/10.1016/j.jcp.2022.111251

ingredient for establishing a connection between the complexity of the flow that is being studied and the required computational costs [see e.g. in 7]. Apart from using more refined mesh elements, the accuracy of the discrete approximations can also be increased by using more accurate numerical schemes. By definition, the results from higher-order schemes converge faster towards the asymptotic values compared to their lower-order-accurate counterparts with decreasing grid-element sizes. It is well established that higher-order accurate solvers can be an effective option for error-sensitive studies of flow problems. For example, they are a popular choice for detailed numerical studies on turbulent flows [8,9].

In recent literature, both advances for adaptive, and higher-order methods have been developed and are extensively discussed [10–12]. However, the combination of these methodologies is fairly new. Notable work includes that of Chowdhury [13], who presented a 4th-order accurate adaptive solver for the shallow water equations (Saint-Venant) and many of the ingredients for a 4th-order accurate adaptive flow solver. Further, the innovative work of Kopera and Giraldo [14] has inspired many new research to achieve an adaptive and high-order accurate flow solver using the discontinuous Galerkin formulation [e.g. 15–17]. In this work, we present an alternative to these methods, but a performance benchmark against these methods is outside the scope of this work. In the aforementioned references, it has been explained how the combination of grid adaptivity and higher-order accuracy can be interesting for solving error sensitive problems with a high degree of scale separation in space and time. In fact, the main advantageous feature for each of the approaches is similar: A desired level of accuracy maybe achieved with fewer cells compared to a lower-order or static mesh solver. This in turn can make such solver strategies more efficient, although there is no general consensus on this topic. This works presents and test such a combined adaptive and higher-order-accurate solver that does not use finite element methods and studies the performance by comparing the results against an well-established second-order accurate adaptive flow solver.

This paper is organized as follows: First, the formulation of the solver and its design are presented in Sect. 2. It entails the details of the chosen tree-grid structure, the time and space discretization and a grid adaptation algorithm (i.e. the refinement indicator). Next in Sect. 3, the present solver is tested and exemplified, starting with simple flow problems with a focus on the asymptotic convergence of the numerically obtained solution with increasingly fine grid resolutions. Later in that section, the focus shifts towards use cases in more dynamical flow setups. This includes a benchmark comparison against a second-order accurate adaptive solver. Finally, Sect. 4 briefly discusses the results and concludes the present work.

2. Solver description

This section discusses the design and formulation of the fourth-order accurate adaptive solver. First, the chosen mesh structure is discussed, next the solver methodology itself is presented, and finally, a mesh-adaptation procedure is motivated and discussed.

2.1. The tree-grid structure

The choice of a mesh structure is a delicate issue when designing a numerical solver for partial differential equations. It may influence the numerical schemes, algorithms and areas of application to a great extend. In the absence of a silver-bullet option, we use a tree-structured mesh with square (or cubic in 3D) elementary control volumes for the representation of the discretized fields [18]. It combines the flexibility of a variable-resolution mesh with a locally Cartesian stencil. We use the high-performance and MPI parallel tree-grid implementation provided by the *Basilisk* toolbox (basilisk.fr [19]). The methods presented in this chapter focus on the formulations for two-dimensional (2D) flow. For the details of the (straightforward) extension to three dimensions, the reader is referred to the freely available and documented source code. Scripts and instructions for running the setups presented herein are also available via the Basilisk website (see Table 1).

A tree grid is characterized by a parent-child hierarchy between cells at integer levels of refinement. Consider a square domain of size $[0, L_0] \times [0, L_0]$, discretized with a single cell of size $\Delta_0 = L_0$. In order to increase the resolution, four equidistant child cells of size $\Delta_1 = \frac{L_0}{2}$ can be added. A child cell can become a parent when its children at a higher level of refinement are initialized (e.g. Fig. 1a). As such, a (family) tree can be constructed (Fig. 1b). The child cells at the end of the tree branches are called the leaf cells and these are the cells with the highest resolution at any given point in the domain.

All formulations and examples in this work concern a square domain using a single tree-root cell. This limitation could be overcome to some degree by enabling the stitching of multiple root cells as it would allow for block-based domains. This is part of ongoing developments within the Basilisk framework. Alternatively, the schemes could be extended with an embedded boundary formulation for a more flexible definition of the domain whilst maintaining the square/cubic elementary volumes of the cells. The latter option would require algorithmic changes to the solver presented herein [e.g. 20].

2.1.1. Staggering and boundary conditions

An advantage of the underlying equidistant Cartesian grid structure is that the definition of discrete mathematical operators (gradient, interpolations, etc.) is relatively straightforward and efficient compared to their uneven grid counterparts. Therefore, rather than use formulations that work on irregular meshes, Basilisk chooses to only build operations on regular stencils, and fill the missing values near resolution transitions and the boundaries of the domain. The disadvantage of this choice is that the missing field values must be computed in an additional, potentially expensive, step. In order to minimize this computational burden, only two layers of ghost cells are added to complete the 5×5 stencil (see Fig. 1c). Furthermore, by demanding that the cells at resolution transitions only differ by one level and the usage of a multi grid V-cycle, the

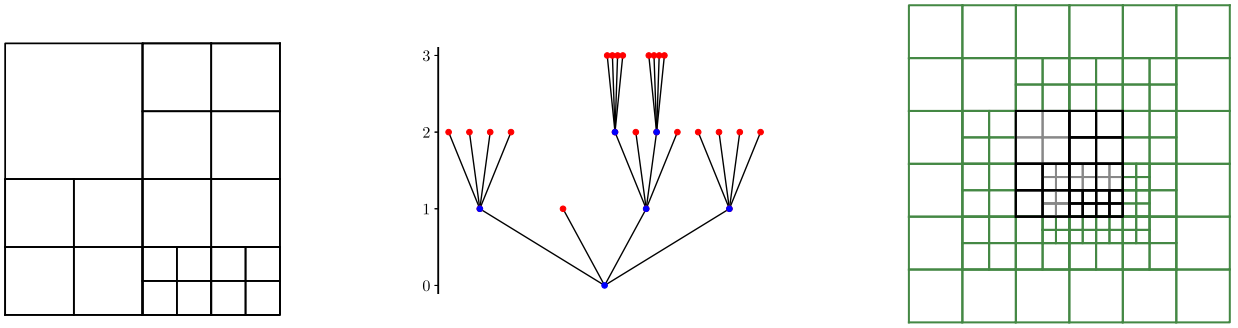


Fig. 1. An example of a two-dimensional tree-grid structure (quadtree) with 19 cells (a). The corresponding tree diagram (b) and the prolongation and domain boundary ghost cells placement in grey and green, respectively (c). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

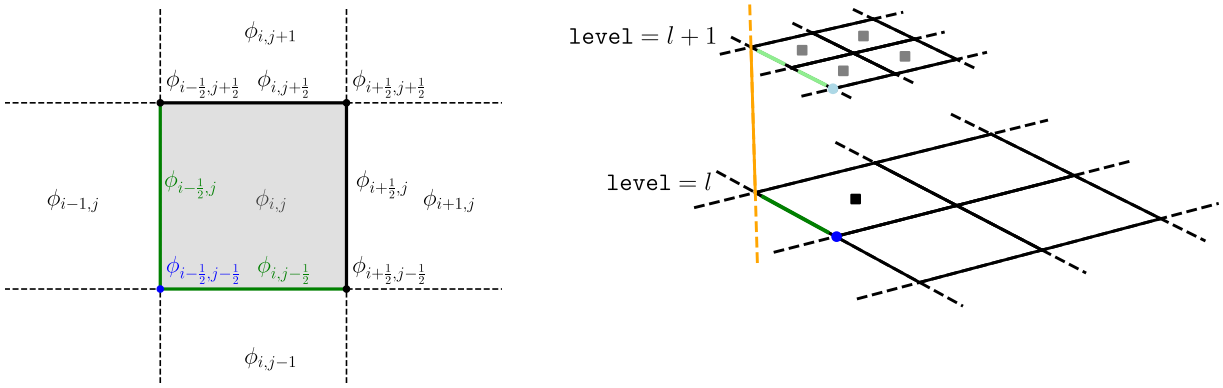


Fig. 2. Relative locations of possible discrete representations of a field $\phi(x, y)$. $\phi_{i,j}$ (grey) represent the cell averaged value, $\phi_{i-\frac{1}{2},j}$ and $\phi_{i,j-\frac{1}{2}}$ (green) represent the face-averaged values and $\phi_{i-\frac{1}{2},j-\frac{1}{2}}$ (blue) the vertex-point value. Further, some neighboring values are shown (black) (a). For restriction, the parent-cell values (saturated colors) are computed from the corresponding child values (light colors) (b).

ghost-cell values can be estimated using regular grid expressions only. The estimation of the ghost-cell values is based on a polynomial approximation with at least 4th order accuracy. For their definition, we distinguish between three types of cell-based discretization styles. For example, a two-dimensional dummy scalar field $\phi(x, y)$ can be discretized on a grid off cells (see Fig. 2). For a square cell $C_{i,j}$ of size $[x_0, x_0 + \Delta] \times [y_0, y_0 + \Delta]$, ϕ can be represented ...

- ... as cell averages; $\phi_{i,j} = \frac{1}{\Delta^2} \iint_{C_{i,j}} \phi(x, y) \mathrm{d}A$,
- ... as face averages; $\phi_{i-\frac{1}{2},j} = \frac{1}{\Delta} \int_{y_0}^{y_0+\Delta} \phi(x_0, y) \mathrm{d}y$ or $\phi_{i,j-\frac{1}{2}} = \frac{1}{\Delta} \int_{x_0}^{x_0+\Delta} \phi(x, y_0) \mathrm{d}x$,
- ... as the vertex-point values; $\phi_{i-\frac{1}{2},j-\frac{1}{2}} = \phi(x_0, y_0)$.

The staggering is chosen so that it is possible to exactly compute the corresponding parent-cell values of ϕ from the child values (see Fig. 2b). In D dimensions,

- The parent-cell average is the average of its 2^D child-cell averaged values,
- The parent-face average is the average of the 2^{D-1} corresponding child-face averaged values,
- The parent vertex-point value is equal to the (single) corresponding child vertex value.

After this so-called *restriction* operation, starting from the finest full level (i.e. the level l at which all 2^{Dl} cells are active), the ghost-cell values at the domain's boundaries are computed using polynomial extrapolation of the field data supplemented with a physical boundary condition. We present the formulation at the left-hand-side boundary (see Fig. 3). The other boundaries are formulated according to a rotation of the indices. For the cell-averages and tangential-face averages a Dirichlet boundary condition with boundary value $\phi_{\text{left}} = f_0$ is expressed as;

$$\phi_{-1,j(-\frac{1}{2})} = 4f_0 + \frac{-13\phi_{0,j(-\frac{1}{2})} + 5\phi_{1,j(-\frac{1}{2})} - \phi_{2,j(-\frac{1}{2})}}{3} + \mathcal{O}(\Delta^4), \tag{1}$$

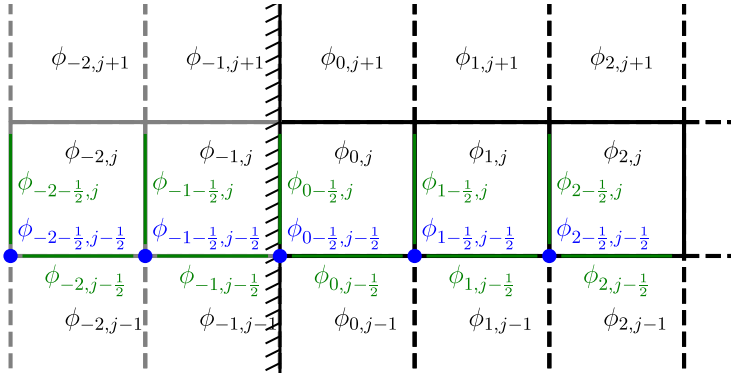


Fig. 3. The stencil indices of the cells and ghost cells near the left-hand-side boundary (stacked diagonal lines).

$$\phi_{-2,j(-\frac{1}{2})} = 16f_0 + \frac{-70\phi_{0,j(-\frac{1}{2})} + 32\phi_{1,j(-\frac{1}{2})} - 7\phi_{2,j(-\frac{1}{2})}}{3} + \mathcal{O}(\Delta^4). \tag{2}$$

A Von Neumann boundary condition, $\left(\frac{\partial\phi}{\partial x}\right)_{\text{left}} = g_x$, is expressed for a cell size Δ as;

$$\phi_{-1,j(-\frac{1}{2})} = \phi_{0,j(-\frac{1}{2})} - g_x\Delta + \mathcal{O}(\Delta^3), \tag{3}$$

$$\phi_{-2,j(-\frac{1}{2})} = \phi_{1,j(-\frac{1}{2})} - 3g_x\Delta + \mathcal{O}(\Delta^3). \tag{4}$$

Although the expression is only third-order accurate, the fourth-order centered derivative at the boundary will exactly yield $(\partial_x\phi)_{\text{left}} = g_x$. For the vertex-point values and normal-face-averaged values a Dirichlet boundary condition with value f_0 is expressed as;

$$\phi_{-\frac{1}{2},j(-\frac{1}{2})} = f_0, \tag{5}$$

$$\phi_{-1-\frac{1}{2},j(-\frac{1}{2})} = 4f_0 - 6\phi_{1-\frac{1}{2},j(-\frac{1}{2})} + 4\phi_{2-\frac{1}{2},j(-\frac{1}{2})} - \phi_{3-\frac{1}{2},j(-\frac{1}{2})} + \mathcal{O}(\Delta^4), \tag{6}$$

$$\phi_{-2-\frac{1}{2},j(-\frac{1}{2})} = 10f_0 - 20\phi_{1-\frac{1}{2},j(-\frac{1}{2})} + 15\phi_{2-\frac{1}{2},j(-\frac{1}{2})} - 4\phi_{3-\frac{1}{2},j(-\frac{1}{2})} + \mathcal{O}(\Delta^4). \tag{7}$$

Herein we do not use the Von-Neumann condition for such fields.

After the domain-boundary ghost values are set, we perform the *prolongation* operation to obtain the values of the ghost cells within the domain by interpolation. At each resolution boundary, we use a 5×5 local Cartesian stencil to estimate the corresponding ghost-cell values at level $l+1$. This is made possible by the requirement that neighboring leaf cells can only differ by one level of refinement. We again distinguish between the discretization styles,

- For the cell averaged quantities, a 4th and 5th order accurate polynomial approximation can be constructed from the one-dimensional 4 and 5 point stencil weights, respectively,

$$\phi_{\text{left child},i,j,4} = \frac{2}{\Delta}\int_{x_0}^{x_0+\frac{\Delta}{2}} \phi(x)\mathrm{d}x \approx \frac{-3\phi_{i-2} + 17\phi_{i-1} + 55\phi_i - 5\phi_{i+1}}{64} + \mathcal{O}(\Delta^4), \tag{8}$$

$$\phi_{\text{left child},i,j,5} = \frac{2}{\Delta}\int_{x_0}^{x_0+\frac{\Delta}{2}} \phi(x)\mathrm{d}x \approx \frac{-3\phi_{i-2} + 22\phi_{i-1} + 128\phi_i - 22\phi_{i+1} + 3\phi_{i+2}}{128} + \mathcal{O}(\Delta^5). \tag{9}$$

  The right-hand-side child value can be approximated by mirroring the stencil weights. This 1D child-average interpolation technique can be applied in each dimension to obtain the stencil weights for children in 2D, and 3D grids. A two-dimensional example is given for the 4th-order accurate approximation of the bottom-left child value in Fig. 4a. It can be easily extended to the 5-point estimation and to three spatial dimensions.
- In order to prolongate the face-average values, we distinguish between $D2^{D-1}$ faces that coincide with the parent cell, and the $D2^{D-1}$ faces that lie along a center-lines/planes of the parent cell. For the coinciding faces with the normal component in the x direction, we use the same weights as in Eq. (9),

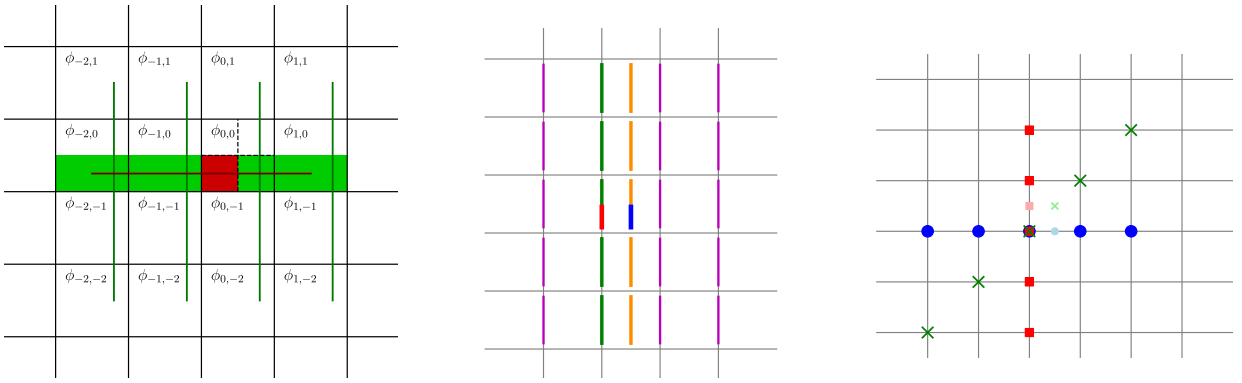


Fig. 4. Various prolongation procedures. Estimation of the bottom left child-averaged value (a); The green rectangle-averages are estimated using the 1D stencil (green lines) and Eq. (8), next, the child-cell value (red) is estimated from the green-rectangle averages, again using Eq. (8). Estimation of the bottom child-faces averages (b): The coinciding (red) face can be estimated from the 1D stencil (Eq. (9)) using the green-marked values, the child face that lies within the parent cell (blue) is computed from the intermediate collocated interpolations (Orange using the green and purple-marked values via Eq. (11)). The child vertex-point values (light blue, light green and pink) are taken from the 1D stencil weights in Eq. (12) using 5 aligned vertex point values.

$$\phi_{\text{bottom left child},i-\frac{1}{2},j} = \int_{y_0+j\Delta}^{y_0+(j+\frac{1}{2})\Delta} \phi(x_0, y)\mathrm{d}y \approx \frac{-3\phi_{i-\frac{1}{2},j-2} + 22\phi_{i-\frac{1}{2},j-1} + 128\phi_{i-\frac{1}{2},j} - 22\phi_{i-\frac{1}{2},j+1} + 3\phi_{i-\frac{1}{2},j+2}}{128} + \mathcal{O}(\Delta^5). \tag{10}$$

Which is conservative. For the child-face values that lies within the parent cell, 5 intermediate 4th-order accurate interpolations are applied (see Fig. 4b).

$$\phi_{\text{mid},j} = \int_{y_0+j\Delta}^{y_0+(j+1)\Delta} \phi\left(x_0 + \frac{\Delta}{2}, y\right)\mathrm{d}y \approx \frac{-\phi_{i-1\frac{1}{2},j} + 9\phi_{i-\frac{1}{2},j} + 9\phi_{i+\frac{1}{2},j} - \phi_{i+1\frac{1}{2},j}}{16} + \mathcal{O}(\Delta^4). \tag{11}$$

From these intermediate values $j \in \{-2, -1, 0, 1, 2\}$, the 5th order accurate interpolation is reused to obtain the child-face values.

- For the prolongation of the vertex-point values, there exists a single child where the parent vertex value can simply be injected. Otherwise, we use a one-dimensional 5-point interpolation with the stencil as shown in Fig. 4c, using the weights,

$$\begin{aligned}\phi_{\text{bottom right child},i-\frac{1}{2},j-\frac{1}{2}} &= \phi\left(x_0 + \frac{\Delta}{2}, y_0\right) \\ &\approx \frac{3\phi_{i-2\frac{1}{2},j-\frac{1}{2}} - 20\phi_{i-1\frac{1}{2},j-\frac{1}{2}} + 90\phi_{i-\frac{1}{2},j-\frac{1}{2}} + 60\phi_{i+\frac{1}{2},j-\frac{1}{2}} - 5\phi_{i+1\frac{1}{2},j-\frac{1}{2}}}{128} + \mathcal{O}(\Delta^4).\end{aligned} \tag{12}$$

The expressions in this section are based on a polynomial reconstruction of the data, and the resulting interpolations are therefore non monotonic. It is well known that such reconstructions may result in non-physical oscillations for non-smooth data. As such, the usage of these higher-order methods comes with a more stringent requirement on the solution smoothness compared to lower-order accurate methods. We will therefore compare the results obtained from the present formulations against those obtained with a second-order adaptive solver. Furthermore, we note that higher-order interpolation methods with limiting have been developed by Chowdhury [13]. These could be used as a replacement of the formulations presented herein. However, these limited methods are more computationally expensive and not conservative for the integral quantities (as opposed to using Eq. (9)). Furthermore, with grid adaptivity, the goal is to capture the sharp transitions in the solution data properly.

With well-timed calls to the restriction, boundary condition and prolongation operator functions, all leaf-cell computations can be formulated using a local 5×5 Cartesian stencil.

2.2. Fourth-order accurate Navier–Stokes solver

The Navier–Stokes equations for incompressible flow are,

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\,\mathbf{u} - \nabla p + \nu\nabla^2\mathbf{u} + \mathbf{a}, \tag{13}$$

$$\nabla\cdot\mathbf{u} = 0. \tag{14}$$

With $\mathbf{u}$ the velocity vector, t the time parameter, ∇ the gradient operator, p is the pressure, ν is the fluid's viscosity and $\mathbf{a}$ is a body-force vector field. For solving the partial differential equation we employ the method of lines, separating the discretization for the spatial dimensions and time.

2.2.1. Temporal discretization

For time advancement of Eq. (13) we use the 4th-order accurate explicit 5-stage low-storage Runge-Kutta method of Carpenter and Kennedy [21]. For stage with index n,

$$\delta\phi_n = f(\phi_n) + a_n(\delta\phi)_{n-1} \tag{15}$$

$$\phi_{n+1} = \phi_n + b_n \mathrm{d}t(\delta\phi)_n. \tag{16}$$

Where f is a function that represents the tendency of a dummy variable ϕ (here the r.h.s. of Eq. (13)). The coefficients a_n and b_n are,

$$a_n = \left\{0, -\frac{567\,301\,805\,773}{1\,357\,537\,059\,087}, -\frac{2\,404\,267\,990\,393}{2\,016\,746\,695\,238}, -\frac{3\,550\,918\,686\,646}{2\,091\,501\,179\,385}, -\frac{1\,275\,806\,237\,668}{842\,570\,457\,699}\right\}, \tag{17}$$

$$b_n = \left\{\frac{1\,432\,997\,174\,477}{9\,575\,080\,441\,755}, \frac{5\,161\,836\,677\,717}{13\,612\,068\,292\,357}, \frac{1\,720\,146\,321\,549}{2\,090\,206\,949\,498}, \frac{3\,134\,564\,353\,537}{4\,481\,467\,310\,338}, \frac{2\,277\,821\,191\,437}{14\,882\,151\,754\,819}\right\}. \tag{18}$$

Noting that for the first stage ($n = 1$), $a_1 = 0$ and hence $\delta\phi_{-1}$ does not need to be defined. The time-step size ($\mathrm{d}t$) is chosen as the minimum of its advection limit and diffusion limit,

$$\mathrm{d}t = \min\left[CFL\left(\frac{\Delta}{u_i}\right)_{\min}, Di\left(\frac{\Delta^2}{\nu}\right)_{\min}\right], \tag{19}$$

with values for the dimensionless quantities; $CFL = 1.3$ and $Di = 0.4$. This means that for small values of Δ, the diffusion limit will be dominant and hamper the speed performance of the solver presented herein. Therefore, we only focus on flow scenarios with a relatively small value for ν, typically corresponding to high-Reynolds-number flows. It would be interesting to overcome this limitation by upgrading to a combined implicit-explicit time-integration scheme [22,23]. By applying an implicit formulation to the treatment of the viscous term, the burden of the $\mathrm{d}t \propto \Delta^2$ scaling can be lifted, as shown by [24–26].

The right-hand-side is computed using the classical operator-splitting method of Chorin [27]. Here a provisional tendency field ($\mathbf{f}^*(\mathbf{u}_n)$) is projected onto the space of divergence free vector fields,

$$\mathbf{f}^*(\mathbf{u}_n) = -(\mathbf{u}_n\cdot\nabla)\,\mathbf{u}_n + \nu\nabla^2\mathbf{u}_n + \mathbf{a}_n, \tag{20}$$

$$\mathbf{f}(\mathbf{u}_n) = \mathbf{f}^*(\mathbf{u}_n) - \nabla p_n. \tag{21}$$

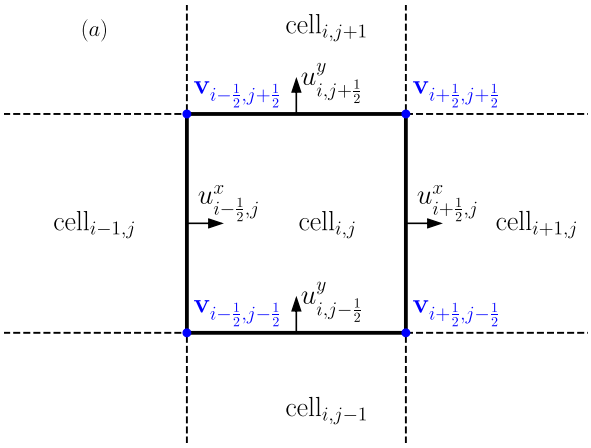
p_n is found by solving the Poisson problem,

$$\nabla\cdot\nabla p_n = \nabla\cdot f^*(\mathbf{u}_n), \tag{22}$$

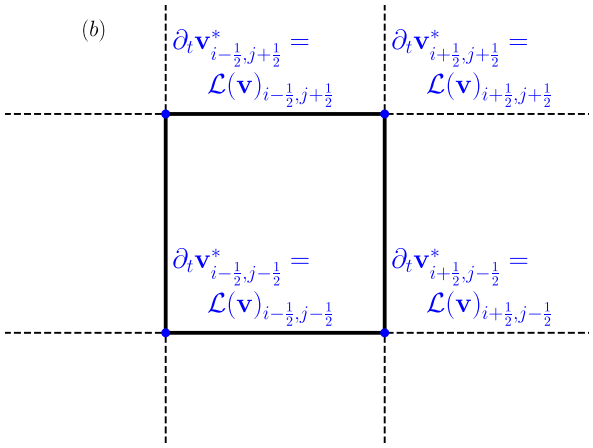
such that $\mathbf{f}(\mathbf{u}_n)$ is solenoidal. Because we will only solve Eq. (22) with a finite tolerance on the residual, these residuals could potentially accumulate and result in a non-governable flow divergence. As such, an additional projection of the solution ($\mathbf{u}$) is performed after each full Runge-Kutta time step (i.e. once every 5 stages).

2.2.2. Spatial discretization

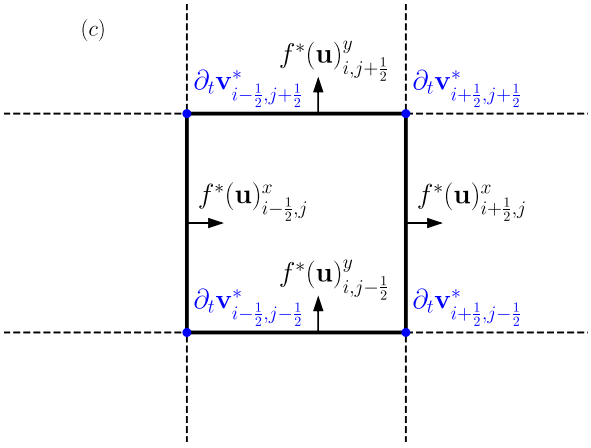
Generally, the discrete solution $\mathbf{u}(\mathbf{x}, t)$ is polluted with truncation errors. However, it is regarded favorable when such solution fields inherit physically consistent properties from the mathematical system *exactly*. For example, a finite-volume method conserves the first-order moments for a flux-divergence evolution equation. Diagnosing such integral quantities is only possible without additional discrete approximation when dealing with cell integrals (e.g. volume averages). For the present solver, we choose to exactly satisfy the incompressible-flow condition of Eq. (2). Given the choices in Sect. 2.2.1, this is only possible when the cell-averaged flow divergence can be computed exactly. It is achieved here by choosing the discrete velocity field to represent the normal-component average along the faces of the grid cells. Following the definitions



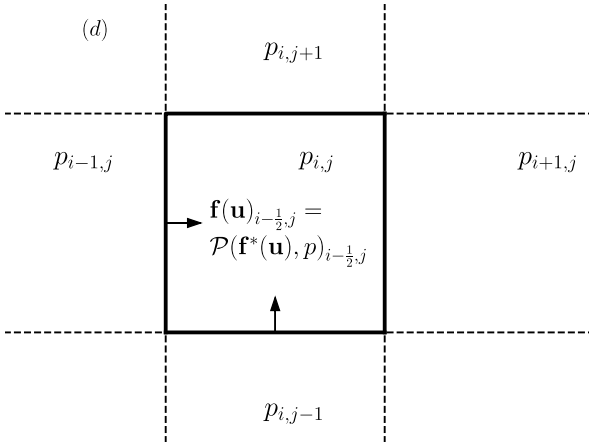
(a) Interpolation of the face-averaged velocities (black) to vertex-point values (blue)



(b) Computation of the provisional tendency vector field at the vertex locations



(c) Interpolation of the provisional tendency from the vertex-point values to face-averaged values.



(d) The (global) solenoidal projection ($\mathcal{P}$) of the provisional tendency using the scalar field p

Fig. 5. The steps taken by the solver to compute $\mathbf{f}(\mathbf{u}_n)$.

given in Fig. 5, the cell-averaged divergence field of a dummy vector field $\mathbf{F} = \{F^x, F^y\}$, discretized as normal-component face averages, for a cell with index i, j ($\text{div}(\mathbf{F})_{i,j}$) can be computed from the discrete solution without discretization error,

$$\frac{\int_{\text{cell}_{i,j}} \nabla \cdot \mathbf{F}\, \mathrm{d}V}{\int_{\text{cell}_{i,j}} \mathrm{d}V} = \text{div}(\mathbf{F})_{i,j} = \frac{1}{\Delta} \Sigma_{\text{dim}} \left(F^x_{i+\frac{1}{2},j} - F^x_{i-\frac{1}{2},j} \right). \tag{23}$$

Where Σ_{dim} is the summation operator for each dimension, applying a rotation among the number of dimensions in the problem (e.g. for two dimensions: $F^x_{i-\frac{1}{2},j} \rightarrow F^y_{i,j-\frac{1}{2}}$).

The general layout of the steps taken to compute $\mathbf{f}(\mathbf{u}_n)$ are depicted in Fig. 5. First, co-located point-values of the provisional tendency field are computed at the vertices of the cells. This choice helps to easily extended the solver to include 4-th order accurate tracer advection and thermodynamics in the Boussinesq limit (see sections 2.3 and 3.6). As such, the discrete velocity field of face-averages $\mathbf{u}_n$ is interpolated to the corresponding values at the vertex locations ($\mathbf{v}_n$). We write for the x component (v^x) in two dimensions;

$$v^x_{i-\frac{1}{2},j-\frac{1}{2}} = \frac{-u^x_{i-\frac{1}{2},j-2} + 7\left(u^x_{i-\frac{1}{2},j-1} + u^x_{i-\frac{1}{2},j} \right) - u^x_{i-\frac{1}{2},j+1}}{12}. \tag{24}$$

At the grid of vertices, finite differencing methods can be readily applied to compute the derivatives that appear in the r.h.s. of Eq. (13) [cf. 28]. In order to estimate the first derivative (for the advection term) with fourth-order accuracy whilst only using a 5-point stencil, we could (in principle) use the fourth-order central approximation. However, it is well known

that this leads to unstable numerical integration. As such, we must resort to compact (implicit) finite difference schemes which can be up to 8th order accurate using the 5 point stencil [29]. However, numerical experimentation revealed that the symmetric compact schemes lead to unstable time integration with the present solver design (not shown). As such, an upwind compact scheme for $\partial_x \phi = \phi'$ is derived using only the following 5-point stencil:

$$\alpha \phi'_{i+1\frac{1}{2}} + \beta \phi'_{i+\frac{1}{2}} + \phi'_{i-\frac{1}{2}} = a\phi_{i-\frac{1}{2}} + b\phi_{i-1\frac{1}{2}} + c\phi_{i-2\frac{1}{2}}. \tag{25}$$

For each term we write the Taylor-series expansion around $i - \frac{1}{2}$ up to 5 terms for the 5 unknown coefficients,

$$\begin{aligned}
a:\ \phi_{i-\frac{1}{2}} &= \phi, \\
b:\ \phi_{i-1\frac{1}{2}} &= \phi - \Delta\phi' + \tfrac{1}{2}\Delta^2\phi'' - \tfrac{1}{6}\Delta^3\phi''' + \tfrac{1}{24}\Delta^4\phi'''' + \mathcal{O}(\Delta^5), \\
c:\ \phi_{i-2\frac{1}{2}} &= \phi - 2\Delta\phi' + \tfrac{4}{2}\Delta^2\phi'' - \tfrac{8}{6}\Delta^3\phi''' + \tfrac{16}{24}\Delta^4\phi'''' + \mathcal{O}(\Delta^5), \\
1:\ \phi'_{i-\frac{1}{2}} &= + \phi' \\
\beta:\ \phi'_{i+\frac{1}{2}} &= + \phi' + \Delta\phi'' + \tfrac{1}{2}\Delta^2\phi''' + \tfrac{1}{6}\Delta^3\phi'''' + \mathcal{O}(\Delta^4), \\
\alpha:\ \phi'_{i+1\frac{1}{2}} &= + \phi' + 2\Delta\phi'' + \tfrac{4}{2}\Delta^2\phi''' + \tfrac{8}{6}\Delta^3\phi'''' + \mathcal{O}(\Delta^4).
\end{aligned} \tag{26}$$

A corresponding linear system can be constructed from Eqs. (25) and (26),

$$\begin{matrix} \phi: \\ \phi': \\ \phi'': \\ \phi''': \\ \phi'''': \end{matrix}
\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & -1 & -2 & -1 & -1 \\ 0 & \frac{1}{2} & 2 & -1 & -2 \\ 0 & \frac{1}{6} & \frac{8}{6} & \frac{1}{2} & 2 \\ 0 & \frac{1}{24} & \frac{16}{24} & -\frac{1}{6} & -\frac{8}{6} \end{pmatrix}
\begin{pmatrix} a\Delta \\ b\Delta \\ c\Delta \\ \beta \\ \alpha \end{pmatrix} =
\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{27}$$

Solving system (27) yields the coefficients $\{a, b, c, \alpha, \beta\}$ resulting in the compact scheme,

$$\frac{17}{197}\phi'_{i+1\frac{1}{2}} - \frac{76}{197}\phi'_{i+\frac{1}{2}} + \phi'_{i-\frac{1}{2}} = \frac{1}{197\Delta}\left(165\phi_{i-\frac{1}{2}} - 192\phi_{i-1\frac{1}{2}} + 27\phi_{i-2\frac{1}{2}}\right). \tag{28}$$

The implicit problem is solved iteratively. This is preferred over a direct linear solver [cf. 30] as the adaptive grid structure would require to continuously update the direct-solver matrix coefficients. Starting from the explicit centralized 5-point fourth-order accurate initial guess, the ϕ' field is relaxed towards the solution using direct replacements,

$$\phi'_{i-\frac{1}{2}} \leftarrow -\frac{17}{197}\phi'_{i+1\frac{1}{2}} + \frac{76}{197}\phi'_{i+\frac{1}{2}} + \frac{1}{197\Delta}\left(165\phi_{i-\frac{1}{2}} - 192\phi_{i-1\frac{1}{2}} + 27\phi_{i-2\frac{1}{2}}\right). \tag{29}$$

Five such relaxation sweeps for all leaf cells proved sufficient for all cases presented in this work. Experimentation revealed that employing a multi-grid strategy [31] did not accelerate these computations, which can likely be attributed to the accurate initial guess. According to the authors best knowledge, this upwind compact scheme, with only a single overlapping stencil between the left and right-hand side ($i - \frac{1}{2}$), has not been applied in previous literature. As such, a brief analysis is presented next. It is clear that the scheme is designed for sufficiently smooth data only. In practice it is hard to guarantee the smoothness of all fields. For example, with artificially initialized fields, or when under resolving small flow features with high gradients. As such the appendix presents a test for the one-dimensional linear advection equation applied to fields with varying degrees of smoothness. It appeared that the advection scheme is not total-variation diminishing, but it is dissipative for the total variance for all test cases. Furthermore, Fig. 6 shows the classical modified wavenumber analysis of this scheme and compares it against alternatives. Although compact schemes can have excellent spectral properties, the present formulation has worse spectral properties than the classic 3rd-order accurate upwind scheme.

For $\nu \neq 0$, the viscous term is computed using the explicit 4th order central estimation of the second derivative in each direction,

$$\partial_{xx}\phi_{i-1/2} = \frac{-\phi_{i-2\frac{1}{2}} - \phi_{i+1\frac{1}{2}} + 16\left(\phi_{i-1\frac{1}{2}} + \phi_{i+\frac{1}{2}}\right) - 30\phi_{i-\frac{1}{2}}}{12\Delta^2}. \tag{30}$$

If the problem is specified with a body-force term ($\mathbf{a}$), it must be defined as a vertex-point vector field such that no additional numerical estimation is required to evaluate $\mathbf{a}$ at the vertex locations.

The last step before projection is to estimate the face-averaged values of $\mathbf{f}^*(\mathbf{u})$ from the vertex point values of the tendency vector ($\partial_t \mathbf{v}^*$). For the x components of $f^*(\mathbf{u})^x$, the corresponding vertex-based tendency is stored in $\partial_t v^{x,*}$. We write,

$$f^*(\mathbf{u})^x_{i-\frac{1}{2},j} = \frac{-\partial_t v^{x,*}_{i-\frac{1}{2},j-1\frac{1}{2}} + 13\left(\partial_t v^{x,*}_{i-\frac{1}{2},j-\frac{1}{2}} + \partial_t v^{x,*}_{i-\frac{1}{2},j+\frac{1}{2}}\right) - \partial_t v^{x,*}_{i-\frac{1}{2},j+1\frac{1}{2}}}{24}. \tag{31}$$

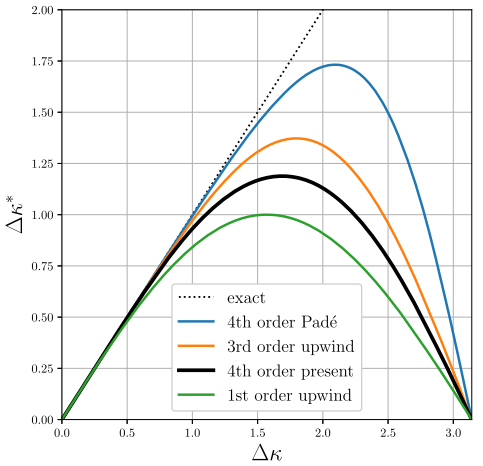


Fig. 6. Modified wavenumber analysis for various finite difference fist derivative schemes, including Eq. (28) (*'4th order present'*).

In order to achieve a 4th-order accurate solenoidal projection of $\mathbf{f}^*(\mathbf{u_n})$, the pressure field ($p$) is discretized with cell-averaged values and its derivative (averaged along a face) is estimated with the 4-point centered expression, cf. Chowdhury [13]. E.g. for the faces with a normal component in the x direction,

$$(\partial_x p)_{i-\frac{1}{2},j} = \frac{p_{i-2,j} - 15p_{i-1,j} + 15p_{i,j} - p_{i+1,j}}{12\Delta}. \tag{32}$$

The components of the gradient field $\nabla p = \{\partial_x p, \partial_y p\}$ are therefore staggered on the faces. Solving the Poisson problem of Eq. (22) is done using the multigrid procedure of Brandt et al. [31] as described in Popinet [19], but it is modified to use the 4th-order-accurate approximations following Chowdhury [13]. The cell-averaged residual field (res) is defined using some initial guess for p, taken from the previous time-integration stage,

$$\mathrm{res}_{i,j} = \mathrm{div}(\nabla p)_{i,j} - \mathrm{div}\left(\mathbf{f}^*(\mathbf{u})\right)_{i,j}. \tag{33}$$

Noting that for this process, the gradient ∇p is first computed on all faces (and coarse-faces via the restriction operator) before the divergence operator can be applied. The 'res' field is then *restricted* to all parent cells. Starting from the coarsest level cells, the relaxation operation is applied to find an update to the solution δp ($p \leftarrow p_o + \delta p$),

$$\delta p_{i,j} \leftarrow \frac{-\Delta^2 \mathrm{res}_{i,j} + \Sigma_{\mathrm{dim}}\left(-\delta p_{i-2,j} + 16\delta p_{i-1,j} + 16\delta p_{i+1,j} - \delta p_{i+2,j}\right)}{15}, \tag{34}$$

with direct reuse replacements and a zero initial guess at the coarsest level. After n_{relax} relaxation sweeps for the cells at level l and the leaf cells at levels lower that l, the field δp at level l is *prolongated* with 4th order accuracy to the active cells at level $l+1$. This process continues until l is equal to the maximum grid level. Then the field p is updated, $p \rightarrow p + \delta p$, and the residual field is re-evaluated. If the maximum absolute residual is more than some tolerance ($\mathrm{res}_{\mathrm{max}} < \epsilon_{\mathrm{tol}}$), the multi grid cycle is repeated.

2.3. Optional scalar tracer fields

The set of equations for incompressible flow (Eq. (13)) can be extended with an advection–diffusion equation for an arbitrary number of scalar fields. E.g. for a scalar field s,

$$\frac{\partial s}{\partial t} = -(\mathbf{u}\cdot\nabla)s + \kappa\nabla^2 s. \tag{35}$$

Where κ is the constant tracer diffusivity of the medium. The time-advancement of Eq. (35) is performed in tandem with the time advancement of $\mathbf{u}$. The scalar field is discretized with vertex-point values, such that the finite difference methods defined in Sect. 2.2.2 can be reused. For the estimation of $\mathbf{u}$ at vertices, the intermediate vertex-vector field values $\mathbf{v}$ are taken (Eq. (24)). Furthermore, the gradient is based on Eq. (28), and the second derivative is computed using Eq. (30). Resulting in a fourth-order accurate discretization in space and time.

2.4. A grid adaptation procedure

The ability of the automatic grid adaptation procedure to refine and coarsen the mesh in the appropriate regions of space and time is a critical ingredient for the performance of an adaptive solver. Here we follow Popinet [19] where the refinement

indicator is based on a so-called multi-resolution analysis of the solution fields [7] (i.e. the velocity components and tracers fields). This analysis characterizes the polynomial content in the solution fields. The tree-grid provides an efficient and convenient method to access this non-trivial variability of the solution field using the underlying multilevel structure. The prolongation methods for the scalar fields can be *reused* in order to check if the solution in the leaf cells can be estimated accurately from the coarser-level data. For each leaf-cell value, the 4 and 5th order accurate prolongation operators can be used to interpolate with said accuracy. For smooth fields that are accurately described with low-order polynomial variation in space (say up to 3rd order), the 4th-order accurate schemes used by the solver will yield accurate discrete approximations and hence, a high resolution would not be required. In general, for a dummy field s, the difference between the prolongation approximation ($\mathcal{P}_{s_i}$) and the actual value s_i ($\chi_i = \|s_i - \mathcal{P}_{s_i}\|$) scales with the higher-order polynomial content of the solution field.

The χ field has successfully been applied as a refinement and coarsening indicator (Ξ) in studies that employ second-order finite-volume methods [32–34]. However, due to the non-symmetric staggering of child faces and vertices with respect to their parent, the resulting χ fields are not smooth for these quantities. As such, for a group of siblings (i.e. child cells that share a common parent cell), the maximum χ value is chosen and smoothed with the neighbor values using a Gaussian kernel. This smoothed χ field ($\hat{\chi}$ is weighted with the grid-cell size (Δ) in order to obtain the refinement indicator:

$$\Xi = \Delta^q \hat{\chi}, \tag{36}$$

where we use $q = 1$, an exponent value that can be tuned to set a penalty on refining fine cells. Increasing $q > 0$ in practice results in a relative decrease of cells at resolution boundaries by effectively removing small (and narrow) high resolution islands. The user can tune the effective mesh resolution by specifying a refinement criterion for each field (ζ), and the algorithm adapts the grid each solver iteration with the following rule:

$$\text{The } i\text{-th cell is } \begin{cases} \text{Too coarse}, & \Xi_i > \zeta \\ \text{Too fine}, & \Xi_i < \frac{2}{3}\zeta \\ \text{Too coarse}, & \text{otherwise}, \end{cases} \tag{37}$$

with the additional constraint being that neighboring cells can only differ by one level of refinement. In case of a dispute, refinement of 'too coarse' cells takes precedence over not refining 'just fine' cells, which in turn has precedence over the coarsening of cells that are marked as being 'too fine'. Upon refinement, the aforementioned prolongation methods can be used to define the values of the variables within the new cells. However, the velocity components warrant special attention, as the prolongated velocity field (using Eqs. (10) and (11)) is not solenoidal. For this purpose we recall that the second-order accurate local projection method of Popinet [35] can be readily applied to project the face-values defined using Eq. (11) onto divergence-free space. The fact that the helper scalar field used in the derivation [see 35] is only defined with second order accuracy is no concern [cf. 13]. Since the prolongation uses a solenoidal velocity field from the coarse grid, the divergence of the prolongated cells can be attributed to the 4th-order prolongation error. The local projection scheme therefore only applies a 4th order correction and hence, the accuracy of the solenoidal refinement inherits the accuracy of the prolongation.

The presented refinement indicator (Ξ) is not rigorously derived from analysis of the discretization errors in the approximations for the various terms in the Navier–Stokes equations, nor how this affects the solution statistics of interest for a given flow configuration [36]. Rather, it should be viewed as a simple solution-field feature-detection algorithm. As such, the ability of the algorithm to represent solution fields with increasing fidelity by decreasing ζ is demonstrated with convergence tests in Sect. 3.1.

3. Tests and examples

This section presents a selection of tests and examples for the present solver. The tests start with a focus on the (asymptotic) convergence in order the verify the order of accuracy for the formulations. In later tests, the focus shifts more to a demonstration of the solver whilst including a comparison against alternative methods. This provides an efficiency benchmark for this solver. All the computer code required to run these tests is freely available, and links to their online locations are presented in Table 1.

3.1. Convergence of the discrete approximations on an adaptive grid

It is not a priori clear that using the proposed grid-adaptation algorithm of Sect. 2.4 will yield improved discrete representation for the flow field evolution with decreasing refinement criteria [11]. Especially, the rate at which the errors in the discrete approximations vanish with an increasing number of automatically placed grid cells is an important characteristic of the solver design presented in Sect. 2. From a user' perspective, it is highly desirable to be able to tune the balance between accuracy and solver speed performance using only a single parameter. As such, we analyze the convergence rates of the various solver steps with an increasingly strict refinement criteria (ζ). We consider a dummy flow field ($\mathbf{u}$) in cylindrical coordinates (r, θ),

Table 1
Overview of tests and examples including links to the documented setup files. The last column indicates if there is additional movie visualization available via the link.

Implementation, Tests and examples			
Name	Section	Web link	Video
The 4th-order solver	2	http://basilisk.fr/sandbox/Antoonvh/nsf4t.h	N.A.
Adaptive Convergence	3.1	http://basilisk.fr/sandbox/Antoonvh/test_conv.c	No
Taylor-Green vortices	3.2	http://basilisk.fr/sandbox/Antoonvh/tg4.c	Yes
3D vortices of Antuono [37]	3.3	http://basilisk.fr/sandbox/Antoonvh/antuono4.c	No
Viscous layer	3.4	http://basilisk.fr/sandbox/Antoonvh/visc_boun.c	Yes
Shear instability	3.5	http://basilisk.fr/sandbox/Antoonvh/doublep.c	Yes
Rising plume	3.6	http://basilisk.fr/sandbox/Antoonvh/rise.c	No
Dipole-wall collision	3.7	http://basilisk.fr/sandbox/Antoonvh/dip.c	Yes
Head-on vortex rings	3.8	http://basilisk.fr/sandbox/Antoonvh/ring4.c	Yes
Non-smooth advection	Appendix	http://basilisk.fr/sandbox/Antoonvh/upat.c	No

$$\mathbf{u} = \{v_r, v_\theta\} = \{0, re^{-\left(\frac{r}{\sigma}\right)^2}\}. \tag{38}$$

The flow field is discretized on a 2D square domain of size $20\sigma \times 20\sigma$, large enough to prevent the boundary conditions to affect the results presented here. For a range of ζ values, the corresponding grid is generated and the errors for the various solver steps are evaluated.

After the initialization of the grid and the face-averaged velocity component values, the convergence of the errors for the face-average to vertex-point interpolator (e.g. (24)) are evaluated as:

$$\epsilon^x_{i,j} = \|u_{x,i-\frac{1}{2},j-\frac{1}{2}} - u_{x,\text{analytical}}\|, \tag{39}$$

$$\epsilon^y_{i,j} = \|u_{y,i-\frac{1}{2},j-\frac{1}{2}} - u_{y,\text{analytical}}\|, \tag{40}$$

$$L_1 = \sum_{i,j} \Delta^2 \left(\epsilon^x_{i,j} + \epsilon^y_{i,j}\right), \tag{41}$$

$$L_\infty = \text{Max}\left(\epsilon^{x,y}_{i,j} \in \text{all cells}\right), \tag{42}$$

where Δ is the local grid-cell size, corresponding to the i, j indexed cell. The errors are plotted against the effective resolution ($N_{\text{effective}}$) which is computed for this two-dimensional test as,

$$N_{\text{effective}} = \sqrt{\#\text{cells}}. \tag{43}$$

The results are shown in Fig. 7a. It appears that over a wide range the errors scale with $N_{\text{effective}}$. A fitted scaling line (using logarithmic weights) is plotted and shows that the error norms vanish slightly faster than the order of accuracy of the approximation in Eq. (24). The errors in the finite difference schemes for the first and second derivatives are tested next (i.e. using Eqs. (28) and (30), respectively). The results show a 4th order convergence rate for the L_1 error norm. For the L_∞ error norm in the second derivatives, the convergence at a slower rate ($L_\infty \propto N^{\approx -2\frac{1}{2}}_{\text{effective}}$). This is due to 1) the regions identified for refinement by the 4th and 5th order accurate prolongation method do not correspond to the regions where the errors in the 4th-order-accurate second derivative are large, and 2) the treatment of resolution boundaries is 5th order accurate and hence pollute the estimate of the second derivative with a 3rd-order error at such boundaries. Face vertex to face interpolation is tested and the results in Fig. 7d show a very similar trend as for the face-to-vertex interpolator.

The last step of the solver is the projection of the provisional tendency field. The only discrete approximation used for this step is the face averaged gradient of the pressure field. As such, the pressure field corresponding to the flow field of Eq. (38) is $p = \frac{1}{4}e^{-2r^2}$ is initialized and the face-averaged gradients are computed using Eq. (32). The results are shown in Fig. 7e, and shows that whilst the corresponding L_1 error norm converges at 4th order with $N_{\text{effective}}$, the L_∞ error converges approximately with 3rd order. This is due to the 4-th order accurate treatment of the prolongation for the pressure field near resolution boundaries, which pollutes the estimation of the gradient with a 3rd order term.

3.2. Advection and diffusion of Taylor-Green vortices

The Taylor-Green vortices are a non-trivial solution to the 2D incompressible Navier–Stokes equations [38].

$$u_x(x, y, t) = -\cos(2\pi x)\sin(2\pi y)e^{-2\nu(2\pi)^2 t}, \tag{44}$$

$$u_y(x, y, t) = \sin(2\pi x)\cos(2\pi y)e^{-2\nu(2\pi)^2 t}. \tag{45}$$

The solver is tested by initializing these vortices on a periodic domain of size $[-0.5, 0.5] \times [-0.5, 0.5]$ in a translating frame of reference with $\mathbf{v}_{\text{ref}} = \{-1, -0.5\}$. When the vortices have returned to their initial position at $t = 2$, the numerically

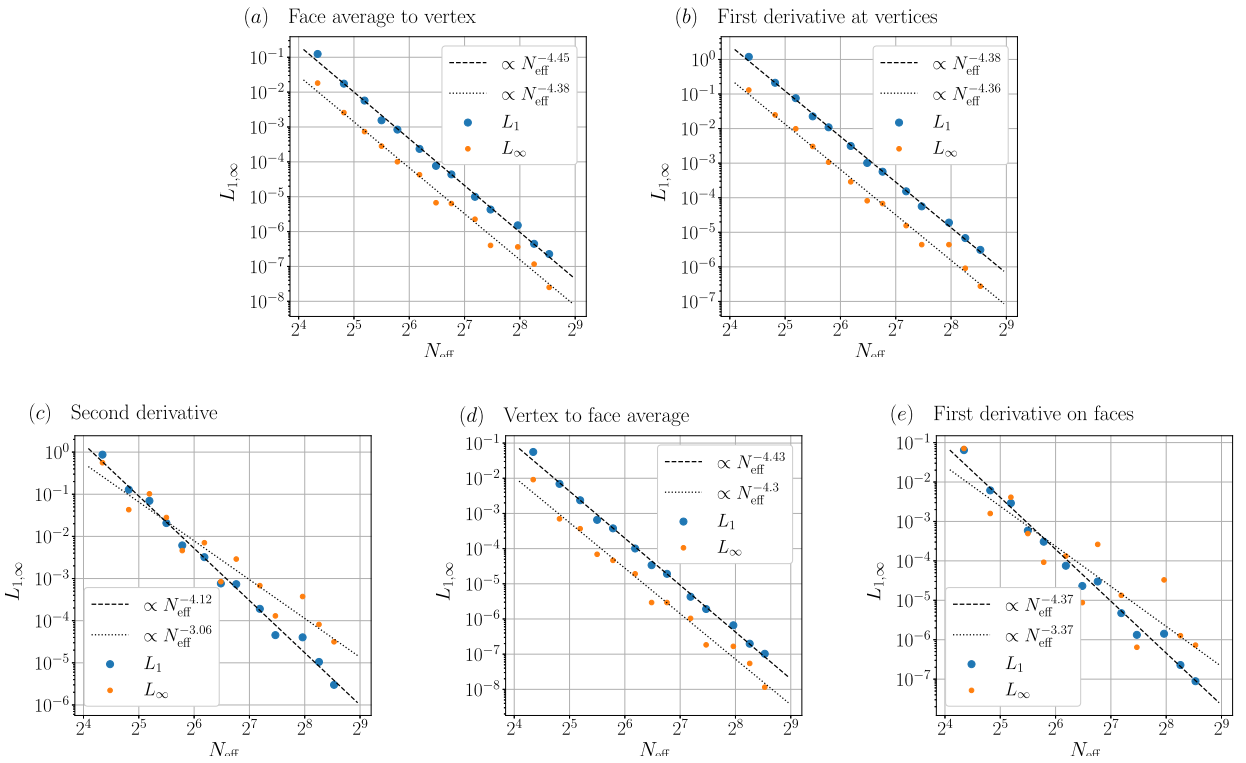


Fig. 7. Adaptive convergence of the error norms with N_{eff} for various discrete approximations. The dashed and dotted lines represent a logarithmic fit.

obtained solution is compared against the analytical solution. The face-averages of the velocity components are computed for initialization and error computation with 6th-order accuracy using the three-point Gaussian quadrature. The test is run for both an inviscid flow ($\nu = 0$) and for a viscous fluid with $\nu = 0.005$. Furthermore, a tracer field with analytical solution,

$$s(x, y, t) = \cos(2\pi x)\cos(2\pi y)e^{-2\kappa(2\pi)^2 t}, \tag{46}$$

is added to test the scalar-tracer schemes for both runs, using $\kappa = \nu$. To study the convergence characteristics, the cases are run on equidistant meshes with $N \times N$ cells, ranging from $N = 8$ to $N = 64$. The L_1 and L_∞ are defined as:

$$\epsilon^x_{i-\frac{1}{2},j} = \|u^x_{i-\frac{1}{2},j} - u^x_{\textbf{analytical}}\|, \tag{47}$$

$$\epsilon^y_{i,j-\frac{1}{2}} = \|u^y_{i,j-\frac{1}{2}} - u^y_{\textbf{analytical}}\|, \tag{48}$$

$$\epsilon^s_{i-\frac{1}{2},j-\frac{1}{2}} = \|s_{i-\frac{1}{2},j-\frac{1}{2}} - s_{\textbf{analytical}}\|, \tag{49}$$

$$L_{1,\mathbf{u}} = \sum_{i,j} \Delta^2 \left(\epsilon^x_{i-\frac{1}{2},j} + \epsilon^y_{i,j-\frac{1}{2}}\right), \tag{50}$$

$$L_{\infty,\mathbf{u}} = \max_{i,j}\left[\epsilon^x_{i-\frac{1}{2},j}, \epsilon^y_{i,j-\frac{1}{2}}\right], \tag{51}$$

$$L_{1,s} = \sum_{i,j} \Delta^2 \epsilon^s_{i-\frac{1}{2},j-\frac{1}{2}}, \tag{52}$$

$$L_{\infty,s} = \max_{i,j}\left[\epsilon^s_{i-\frac{1}{2},j-\frac{1}{2}}\right]. \tag{53}$$

The error norms are plotted in Fig. 8, showing that indeed the numerical solutions converge at a 4th-order rate.

It is known that for sufficiently small values of ν, the array of vortices described by Eqs. (44) and (45) forms an unstable flow configuration [39,40]. The finite tolerated residual for the iterative procedure that solves Eq. (22) introduces a perturbation that breaks the initial flow symmetries. Although these errors are small compared to the discretization errors assessed in this section, it will result in the development of the vortex instability for extended duration [see e.g. 41]. As such, the fourth-order convergence rate can only be expected for a limited simulation time.

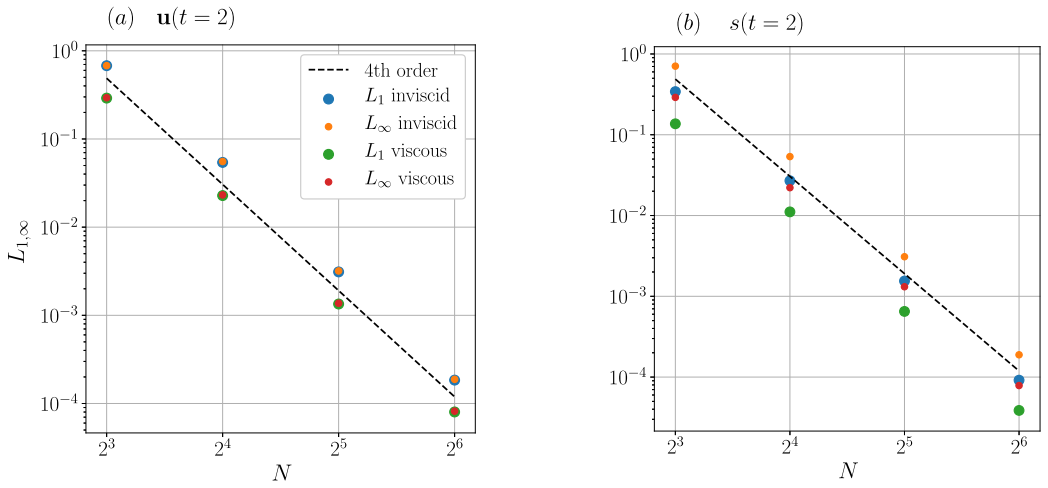


Fig. 8. Taylor-Green vortex advection and diffusion test case: The convergence of the vector field **u** (a) and the scalar-tracer field s (b) at $t = 2$ towards the analytical solution is at fourth order.

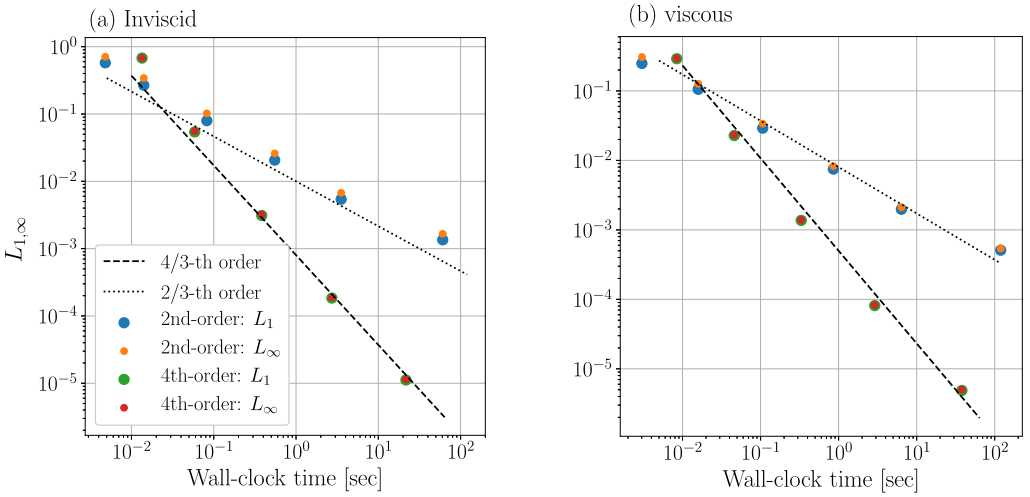


Fig. 9. Taylor-Green vortex advection and diffusion test case: The convergence for the vector field **u** is plotted against the wall-clock time for the runs with varying resolutions. A comparison is presented for the present 4th-order accurate solver and a second-order accurate solver (see text). The Error norms are presented for the inviscid (a) and viscous (b) advection of Taylor-green vortices (Eq. (45)).

3.2.1. Speed performance

It is interesting to compare the efficiency of the present solver against a second-order accurate alternative. As such, we have repeated the numerical experiment using the state-of-the-art second-order-accurate solver of Popinet [42,35], Lagrée et al. [43], which has been employed in many recent works. e.g. [44,7,6]. Fig. 9 compares the error norms for both inviscid and viscous tests against the wall-clock time. It appears that for the low-resolution runs, the 2nd-order solver is able to achieve a more accurate solution for the same wall-clock time. However, as the resolution is increased, the higher convergence rate of the 4th-order accurate solver makes it the more efficient option in the low-error regime.

Although the actual performance may be subject to case-specific optimizations and the details of the used hardware, we highlight the generality of these results. The theoretical scaling lines for a well-behaved 2nd and 4th order accurate CFL-limited solver can be anticipated and are added in Fig. 9. For a n-th order accurate solver the error scales with $L_{1,\infty} \propto N^{-n}$ and the computational effort scales with $t_{\mathbf{wall}} \propto \#\text{cells} \times \#\text{steps} = N^2 \times N = N^3$. Thus the error should scale with the effort according to $L_{1,\infty} \propto t_{\mathbf{wall}}^{-\frac{n}{3}}$. For the inviscid case, both solvers display a slightly favorable scaling. This is likely due to the decreasing relative overhead of steps such as the initialization, the error evaluations and data storage. For the viscous case, the highest-resolution run with the 4th order solver, the solver time step becomes limited by the diffusion criterion rather than the CFL limit (cf. Eq. (19)).

For complex and chaotic flow problems as encountered in science and engineering applications, asymptotic convergence of the numerical solution is impractical. In this perspective, the performance characterization presented in Fig. 9 is not

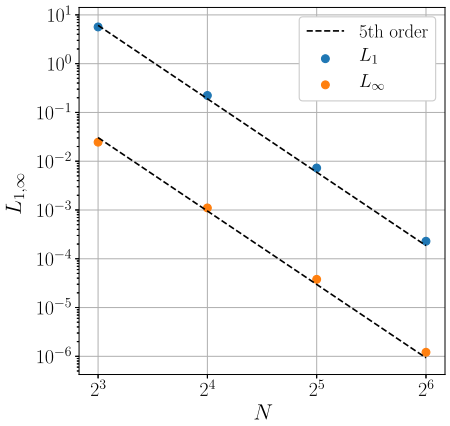


Fig. 10. Convergence for the steady three-dimensional vortices of Antuono (2020), (Eqs. (54) – (56)).

relevant. Rather, the accuracy of the solution is typically only expected in a statistical sense. The fidelity of the obtained statistics is subject to the judgment of experts and maybe based on a grid-convergence study. The formal order of convergence of the solver is then non-trivially related to the convergence of such statistics. Hence, in later Sects. 3.7 and 3.8, we will analyze the efficiency of the solver when it is applied to approximate the evolution of more complex flow setups.

3.3. Steady fully three-dimensional vortices of Antuono (2020)

Antuono [37] presents a steady fully-3D solution to the incompressible Navier–Stokes equations,

$$u_x(x,y,z) = \frac{4\sqrt{2}}{3\sqrt{3}}\left[\sin\left(x-\frac{5\pi}{6}\right)\cos\left(y-\frac{\pi}{6}\right)\sin(z) - \cos\left(z-\frac{5\pi}{6}\right)\sin\left(x-\frac{\pi}{6}\right)\sin(y)\right], \tag{54}$$

$$u_y(x,y,z) = \frac{4\sqrt{2}}{3\sqrt{3}}\left[\sin\left(y-\frac{5\pi}{6}\right)\cos\left(z-\frac{\pi}{6}\right)\sin(x) - \cos\left(x-\frac{5\pi}{6}\right)\sin\left(y-\frac{\pi}{6}\right)\sin(z)\right], \tag{55}$$

$$u_z(x,y,z) = \frac{4\sqrt{2}}{3\sqrt{3}}\left[\sin\left(z-\frac{5\pi}{6}\right)\cos\left(x-\frac{\pi}{6}\right)\sin(y) - \cos\left(y-\frac{5\pi}{6}\right)\sin\left(z-\frac{\pi}{6}\right)\sin(x)\right]. \tag{56}$$

The three-dimensional solver formulation is tested by initializing the solution on a triple-periodic domain of size $[0, 2\pi]^3$ and running the case until $t = 1$. Then the solution is compared against the analytical (steady) solution by computing the L_1 and L_∞ error norms. Fig. 10 shows the results and reveals a 5th-order converge for this three-dimensional test case. This unexpected convergence rate is likely due to the fact that the higher-order derivatives are self-similar versions of the solution itself and thus allowing for compensating errors. We do not investigate this behavior herein, but continue to study the self-convergence of the solver for more complex flows.

3.4. A viscous boundary layer

In order to test the implementation of the no-slip boundary condition we consider a fluid with viscosity $\nu = 1$ moving with a velocity $\mathbf{u} = \{1, 0\}$. At $t = -t_0$ the flow is instantaneously affected by a no-slip top boundary located at $y = L_0$. Here a viscous boundary layer will develop according to,

$$u_x(y,t) = \mathrm{erf}\left(\frac{L_0 - y}{2\sqrt{t+t_0}}\right). \tag{57}$$

Where $\mathrm{erf}(\phi) = \frac{2}{\sqrt{\pi}}\int_0^\phi e^{-\theta}\mathrm{d}\theta$ the Gaussian error function of a dummy variable ϕ. We resolve the flow in a periodic channel of size $[0, 15] \times [0, 15]$ where the top boundary condition is set to no-slip and the bottom boundary condition is free slip. We use $t_o = 1$ and start from $t = 0$. The L_1 and L_∞ error norms in the numerically obtained solution are computed at $t = 1$. The results are plotted is Fig. 11, a show a fourth-order convergence rate. This verifies that the boundary condition for u_x using Eqs. (2) and (7) and the viscous diffusion scheme (cf. Eq. (30)) are indeed implemented correctly with 4th-order accuracy.

3.5. Doubly-periodic shear instability

In this test we consider the Kelvin-Helmholtz instability for two shear layers in a doubly-periodic domain of size $[0, 1]^2$. The flow is initialized according to [45,46],

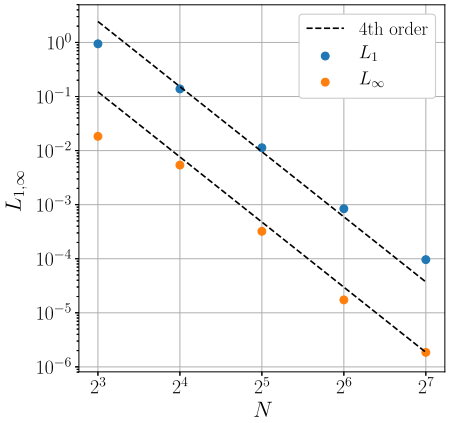


Fig. 11. Convergence for the viscous boundary layer problem (Eq. (57)).

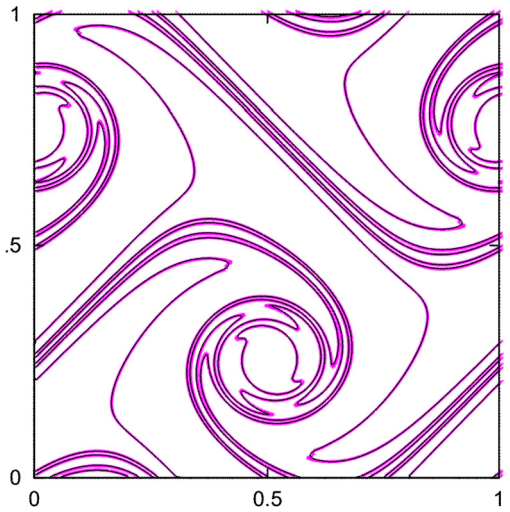


Fig. 12. The vorticity field isolines ($\omega \in \{-24, -18, -12, ..., 24\}$) at $t = 1.2$ for the doubly-period shear instability (see Sect. 3.5). The Magenta lines are obtained with the present solver. The black overlay is a reference solution taken from Hokpunna and Manhart [46] and *their* figure 7f.

$$u_x = \begin{cases} \tanh(\sigma(y - 0.25)) & \text{for } y \leq 0.5, \\ \tanh(\sigma(0.75 - y)) & \text{for } y > 0.5, \end{cases} \tag{58}$$

$$u_y = \xi \sin(2\pi x), \tag{59}$$

with $\sigma = 30$ and $\xi = 0.05$ on a grid with 256^2 cells. The viscosity is set to $\nu = 10^{-4}$ and the flow evolution is computed up to $t = 1.2$. At that stage, each shear layer has rolled-up into a vortex via the Kelvin-Helmholtz instability mechanism. Hokpunna and Manhart [46] showed that the vorticity structure is a sensitive indicator for the fidelity in the representation of the flow. As such, the results from our solver are plotted against those obtained by [46] using their fourth-order accurate finite-volume solver at the same grid resolution. We qualitatively compare the structure of the vorticity field ($\omega = \nabla \times \mathbf{u}$) in Fig. 12 and conclude that the topology of both flows matches to a high degree. Small differences can be found in the regions where the vorticity isolines have a high curvature. The results from this test case gives additional confidence that the present solver is implemented correctly.

3.6. Rise of a buoyant Gaussian plume

The solver is tested for a scenario where a scalar tracer field is coupled to the flow-evolution equation via the body force **a**. Such a buoyancy-force formulation is a popular choice for studying flows with density variations in the Boussinesq limit. In this test, the acceleration vector is set as, $\mathbf{a} = \{0, s\}$ with s the scalar-tracer field. We employ a version of a classical rising buoyant bubble setup [47,48]. A fluid with $\nu = \kappa = 10^{-3}$, is initially at rest and the scalar field is a Gaussian plume placed near the center of the periodic domain of size $[-5, 5] \times [-5, 5]$,

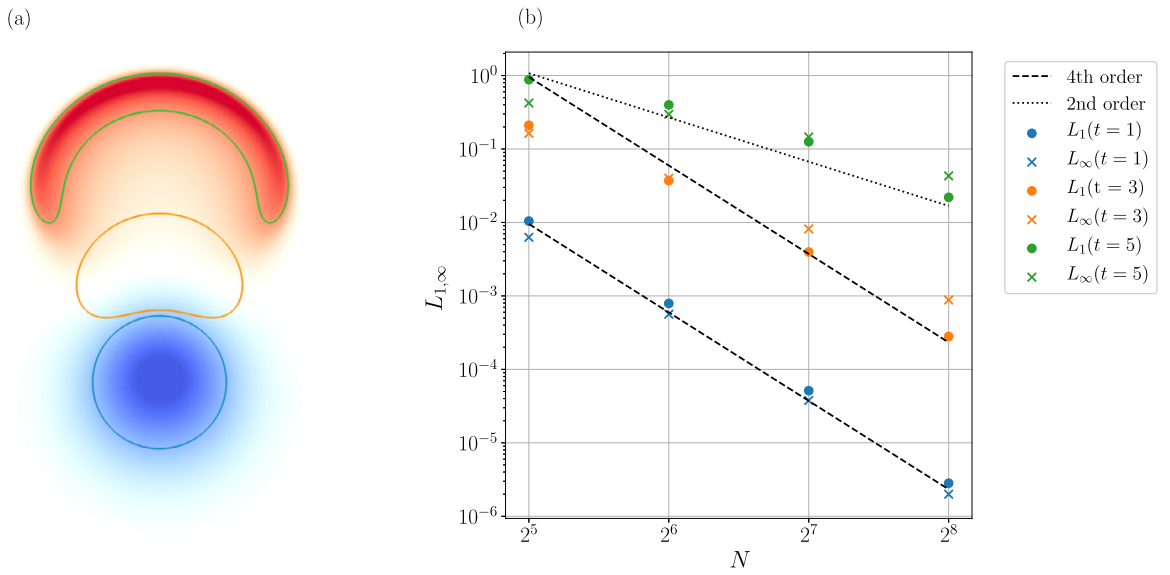


Fig. 13. Rise of a buoyant Gaussian plume test case (Sect. 3.6). (a) Shows the tracer field s in blue and red at $t=1$ and $t=5$, respectively. The blue, orange and green lines are the $s=0.5$ isolines at $t=1, 3$ and 5, respectively. Convergence of the L_1 and L_∞ error norms towards the reference solution (shown in a) are plotted in (*b*).

$$s(x, y, t=0) = e^{-(x-\frac{e}{10})^2-(y+1)^2}. \tag{60}$$

The offset in the x direction is included to prevent symmetries in the discretized fields. Since we do not have an analytical solution for this problem, we check the solver's self convergence. This is not a sufficient condition for the solvers consistency as the solution may converge to an erroneous one. However, it is a necessary condition and it forms an interesting addition to the previous steady analytical-solution tests. As such, the case is run until $t=5$ for grids with $32, 64, 128, 256$ and 512 cells in each direction. The tracer fields obtained at $t=\{1,3,5\}$ from the latter are used as a *reference* solution to compute the errors in the results from the coarser-grid runs. The time window only covers the initial deformation of the plume before shear instabilities are triggered. The reference solution is visualized in Fig. 13a and the convergence of the error is presented in Fig. 13b. We see that for the initial times, the error indeed converges at a 4th order rate. However, at later stages, the errors are non-trivially amplified due the non-linear nature of the system. It is well known that when time-integrating such non-linear systems, accuracy can only be expected in a statistical sense.

We also check the convergence of the adaptive solver with increasingly strict refinement criteria. In contrast to the previous tests, this tests includes the treatment of resolution boundaries, the refinement and coarsening operators and the design of the grid adaptation algorithm. As discussed in Sect. 2.4, the grid adaptation algorithm is not optimized for any specific goal. Also the refinement criterion for the velocity components and the tracer fields are chosen separately, introducing a two-dimensional input parameter space for the resulting grid structure. Here we choose $p=1$ and $\zeta_u = \frac{\zeta_s}{20} = \zeta$. ζ is varied between the runs, using values $\zeta_{\max} = 1.6\times10^{-2}$ down to $\zeta_{\min} = 2.5\times10^{-4}$, so that the maximum used resolution corresponds to 256 cells in each direction at $t=5$. At each time, the effective N is diagnosed as $N_{\text{eff}} = \sqrt{\langle\#\text{cells}\rangle}$, where $\langle\#\text{cells}\rangle$ is the average number of cells in all previous solver iterations. Example solutions field for s and the corresponding grid structures are shown in Fig. 14. The convergence of the error towards the reference solution, is plotted in Fig. 15. For this setup, at $t=1$, hyper convergence is obtained at approximately 5th order for both error norms. At later stages, it appears that the L_1 is well behaved in the studied range. However the L_∞ error is more erratic. Especially at $t=5$, it appears that the L_1 error is not monotonically decreasing with increasing N_{eff}.

3.7. Dipole vortex–wall collision

The dynamical behavior of vortices near a solid object is a relevant sub-problem of wall-bounded flows. For this purpose Orlandi [49] studied the evolution of a dipolar vortex targeted at a no-slip wall using numerical methods and has received considerable attention since [50]. At the wall, a viscous boundary layer is generated with can separate from the wall and roll up into secondary vortices, leading to a rebound of the original dipole. We compute the flow evolution induced by a double vortex with a smooth dipolar vorticity distribution $\omega(x, y)$,

$$\omega = -\omega_0 x e^{\frac{-(x^*)^2-(y^*)^2}{\sigma^2}}. \tag{61}$$

Where ω_0 is a reference vorticity value, controlling the vortex strength, σ is the size of the structure and $x^* = x - \sigma\frac{e}{10}$, $y^* = y - \sigma\frac{\pi}{10}$, are offset coordinates to prevent symmetric alignment of the initial dipole with the grid. The flow in initialized

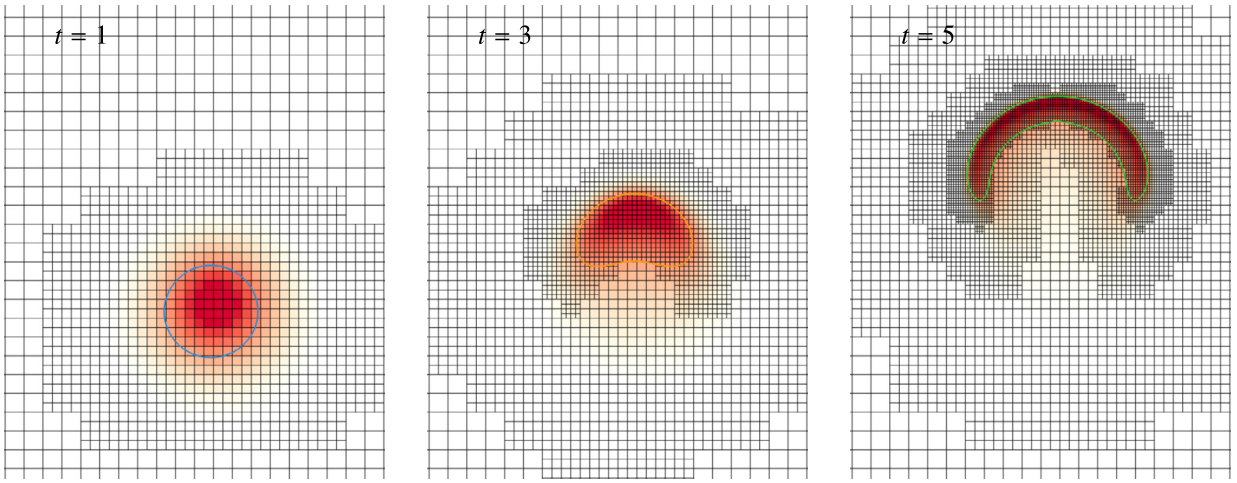


Fig. 14. The evolution of the tracer field s (red), the $s = 0.5$ isoline, and the adaptive tree grid structure. The results are obtained from the run with $\zeta = 2.5 \times 10^{-4}$ for $t = 1, 3$ & 5, left, middle and right, respectively. Note that the images only show a subsection of the domain.

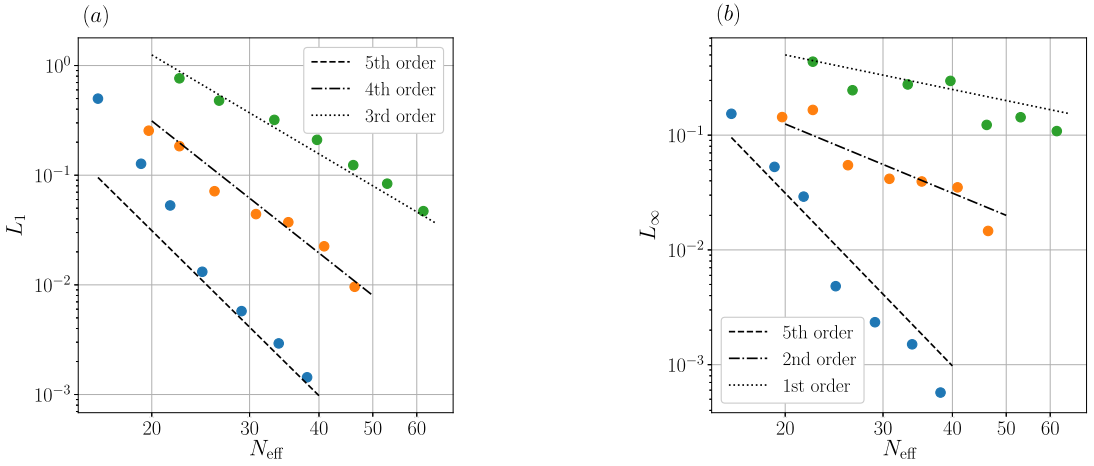


Fig. 15. Error norms for the rising Gaussian plume test using an adaptive grid: The L_1 error for times $t = \{1, 3, 5\}$ in blue, orange and green, respectively (a). The L_∞ error for times $t = \{1, 3, 5\}$ in blue, orange and green, respectively (b).

by solving the Poisson problem for the stream function ($\nabla^2\psi = -\omega$). The dipole is placed in a domain of size $[-15, 15] \times [-15, 15]\sigma^2$, targeted at the top boundary at $y = 15\sigma$. The boundary layer owes its existence to the fluid's viscosity (ν). With parameters ω_0, σ and ν, a Reynolds number (Re) can be defined,

$$Re = \frac{\omega_0^2 \sigma}{\nu}. \tag{62}$$

We set $Re = 10^4$ and compute the evolution up to $t = 400\omega_0^{-1}$. The evolution of an approximate vorticity field and the corresponding grid structure is shown in Fig. 16. Inspired by the work of Kramer et al. [51], Clercx and Van Heijst [50], we are curious about the enhanced dissipation rate due to the vortex-wall collision. The setup is run five times with ζ_u tuned (not shown) so that the wall-clock time is approximately equal to 25, 50, 100, 200 or $400 \pm 10\%$ seconds. This corresponds to $u_e = \{5 \times 10^{-4}, 2.5 \times 10^{-4}\}$. We diagnose the total enstrophy $\Omega = \int \omega^2 \mathrm{d}A$ and plot it as a function of physical time in Fig. 17a. It appears that for the purpose of generating this plot, the 200-second run suffices to approximate the solution of the more expensive simulation that used more cells and more time-integration steps (Fig. 17b), indicating sufficient convergence. Apart from the convergence of the evolution of the enstrophy, the fidelity of the approximate solutions from the cheaper runs is an interesting aspect in less critical applications. It appears that the evolution of the diagnosed enstrophy is non-smooth in the coarse simulations. This is not physical and is likely due to varying representation of the vorticity field on the adaptive grid as is refines and coarsens the flow in the vorticity-rich boundary layer.

The experiment is repeated with the aforementioned second-order accurate adaptive solver and the results are plotted in Fig. 18a. It appears that the statistics do converge towards the results obtained with the fourth-order solver (i.e. in the 400 s run). However, the convergence is slower compared to the 4th-order solver. In the cheaper simulations (25 s, 50 s

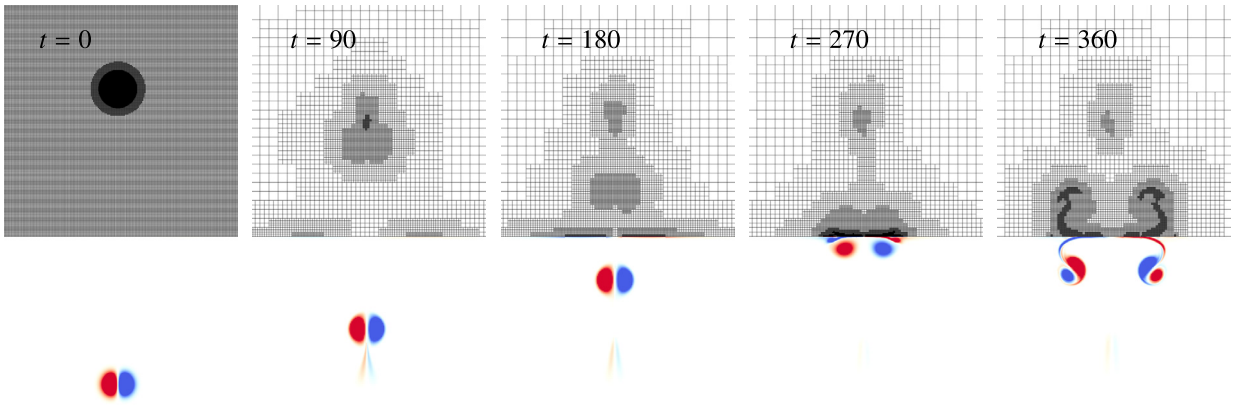


Fig. 16. The evolution of the vorticity field ($\omega = \nabla \times \mathbf{u}$, colors) obtained from the run with wall-clock time of approximately 200 seconds. The adaptive quadtree-grid structure is mirrored at the top boundary. Labels indicate the physical time. Note that the images only show a subsection of the domain.

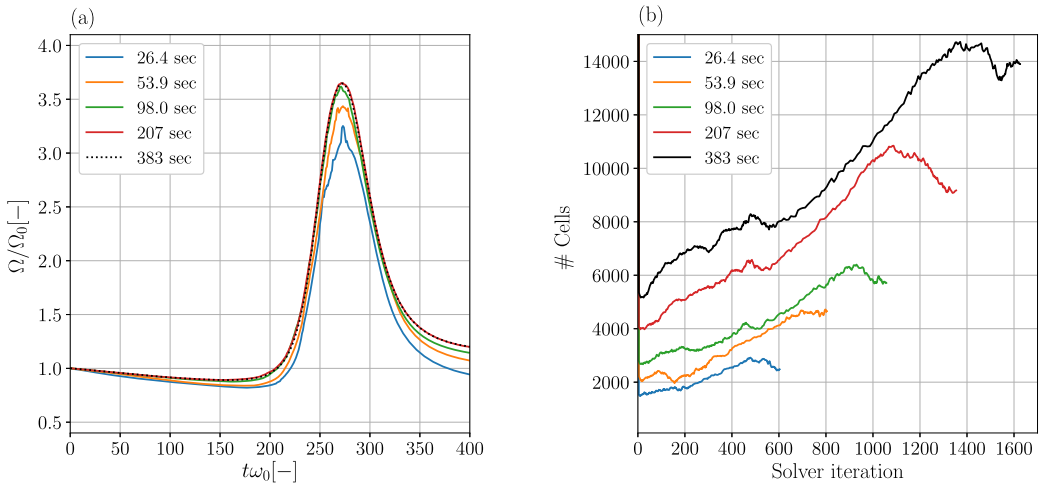


Fig. 17. Results from the dipole-wall collision obtained with the 4th-order accurate solver. The enstrophy is plotted as a function of the physical time (a) and the number of used grid cells for each time-integration step (b). The legend indicates the wall-clock time for each run.

and 100 s), we see that all runs have a more smooth evolution of the enstrophy compared to the 4th order solver. This is a feature that is also present in the converged solution. As such, the 2nd-order solver is a more attractive alternative when convergence for the evolution of the enstrophy is not required.

3.8. Head-on collision of two ring vortices

In their seminal lab experiment, Lim and Nickels [52] targeted two dyed vortex-rings on a head-on collision trajectory. Their video capture shows that during the collision, the vortex rings rapidly expand radially before the vortex rings recombine into multiple smaller vortex rings that are traveling radially outward. The present solver is showcased by reproducing their spectacular results (see also Dorschner et al. [53]) and we hope to set a benchmark for other solvers by defining an unambiguous setup that can be reproduced with other solver strategies.

The flow is initialized inside a $[-20R, 20R]^3$ domain with two vortex rings (major radius R), which are defined using a generalization of the Lamb-Oseen vorticity distribution,

$$\boldsymbol{\omega}(x, y, z) = \oint_{\mathcal{C}_1} \frac{\Gamma}{\pi^{3/2} a^3} e^{-\frac{d^2}{a^2}} \mathrm{d}\mathbf{l}. + \oint_{\mathcal{C}_2} \frac{\Gamma}{\pi^{3/2} a^3} e^{-\frac{d^2}{a^2}} \mathrm{d}\mathbf{l}. \tag{63}$$

Here $\boldsymbol{\omega}$ is the vorticity vector corresponding to vortices with center lines $\mathcal{C}_1$ and $\mathcal{C}_2$, both of equal strength Γ, d is the distance between the point with coordinates $\{x, y, z\}$ and the line element vector $\mathrm{d}\mathbf{l}$ and a is the vortex core size. In the limiting case of a straight vortex core, the resulting vorticity distribution represents the 2D Lamb-Oseen vortex with circulation Γ and size a. The curves are defined as parametric curves,

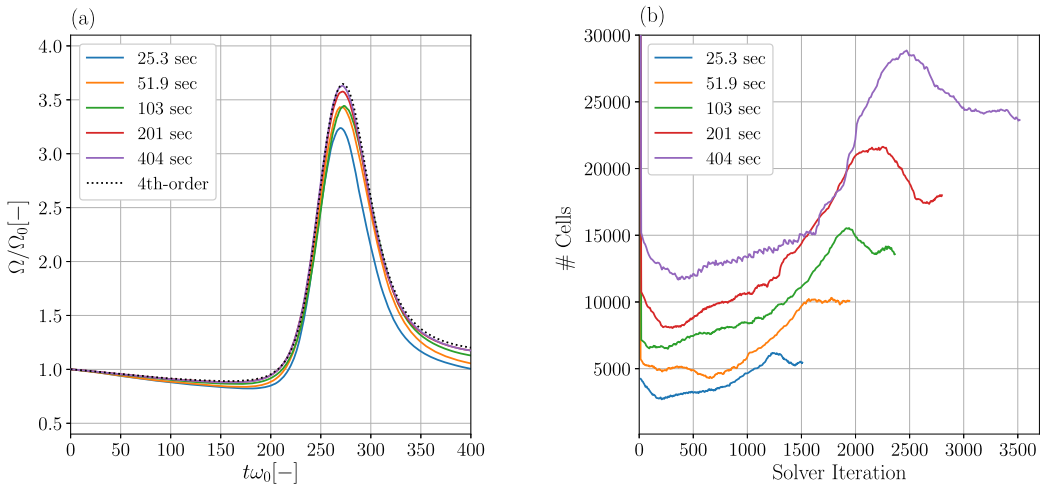


Fig. 18. Results from the dipole-wall collision obtained with the 2nd-order accurate solver and a reference result from the 4th order solver (383 sec). The enstrophy is plotted as a function of the physical time (a) and the number of used grid cells for each time-integration step (b). The legend indicates the wall-clock time for each run.

$$\mathcal{C}_1(\tau) = \begin{pmatrix} R\cos(\tau) \\ R\sin(\tau) \\ d_z + A\cos(n\,\tau) \end{pmatrix}, \tag{64}$$

$$\mathcal{C}_2(\tau) = \begin{pmatrix} R\cos(\tau) \\ -R\sin(\tau) \\ -d_z + A\cos(n\,\tau) \end{pmatrix}, \tag{65}$$

with $\tau \in [0, 2\pi]$, A and n the amplitude and mode of the perturbation, respectively. The vector $\mathrm{d}\mathbf{l}$ is defined as,

$$\mathrm{d}\mathbf{l} = \frac{\partial \mathcal{C}}{\partial \tau}. \tag{66}$$

In this work we set unitary values for a and Γ and choose $R = 3a$, $d_z = 10a$, $A = 0.05a$, $n = 9$ and $\nu = \frac{\Gamma}{2500}$. The initial cell-averaged vorticity field is approximated using the 8-point 4th-order accurate Gauss–Legendre quadrature rule and the integrals are discretized with 1000 segments for each ring. The velocity-potential vector $\boldsymbol{\Psi}$ is computed as a solution to the 3D Poisson problem $\nabla^2\boldsymbol{\Psi} = -\omega$, which is in turn used to compute the initial flow field, $\mathbf{u} = \nabla \times \boldsymbol{\Psi}$.

The runs with different values of the refinement criteria for the velocity component fields are performed using the 6 cores of a workstation equipped with an AMD Ryzen 5 3500X CPU and 32 GB of memory. Fig. 19 visualizes the evolution of the flow field as computed by the most accurate simulation (see below), it shows a combination of snapshots at different times. (See also Fig. 20.) After the initial approach of the rings, they combine and move radially outward. Then an mode-2n instability occurs, resulting in 18 smaller vortex rings that follow a radially outward trajectory. In order to benchmark the solver, we characterize the convergence of the solution statistics obtained with increasingly more refinement. Fig. 21a shows the evolution of the number of cells as a function of the solver time step for runs with different refinement criteria. The legend shows the wall-clock time for each run. Next, the evolution of the so-called dissipation timescale (t_d) is computed as the ratio of the viscous dissipation and the kinetic energy within the domain,

$$t_d = \frac{\nu \int_{\mathcal{D}} J(\mathbf{u})^2 \mathrm{d}V}{\frac{1}{2}\int_{\mathcal{D}} \mathbf{u}^2 \mathrm{d}V}, \tag{67}$$

where $J(\mathbf{u})$ is the Jacobian matrix of $\mathbf{u}$. Presenting this ratio is preferred over plotting the evolution of the dissipation rate and energy separately as its value has a smaller range over the course of the simulation. Especially at the final stages of the simulation, where both dissipation rate and kinetic energy are small. The evolution of t_d^{-1} is plotted as a function of time in Fig. 21b. Here we see that the run that takes approximately 10 hours seems to correspond to the more refined runs, indicating its convergence. These results can be compared against those obtained with the aforementioned second-order accurate solver with the data in Fig. 22. Here the maximum level of refinement needed to be limited to 11 levels, as otherwise, the second-order grid-adaptation procedure increased the cell count so that the run exceeded the available memory space. As such, the results do not quite convergence towards the 4th-order accurate solution, even for equally expensive runs. The fact that the 4th-order accurate solver is able to achieve converged results, without the additional burden of selecting a maximum level of refinement shows its efficiency for solving these types of flow problems.

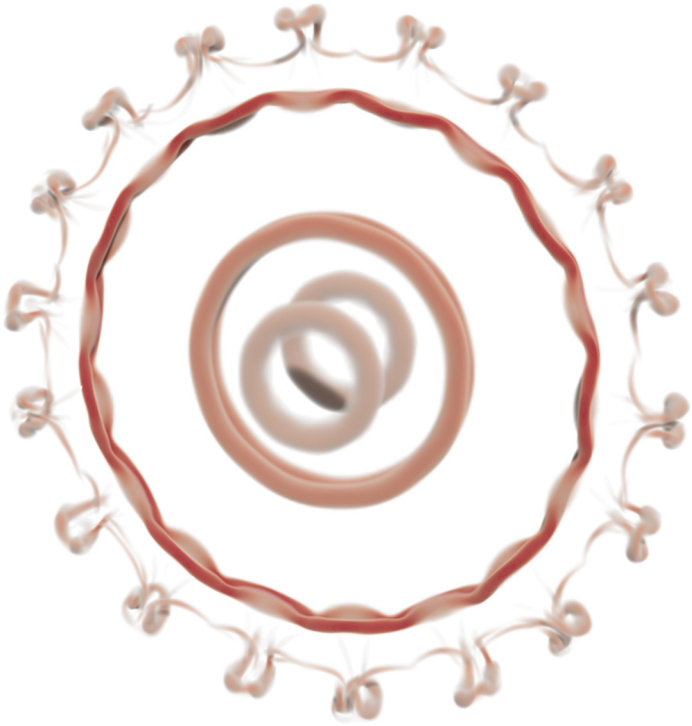


Fig. 19. Visualization of the head-on collision of two vortex rings using volumetric renderings of the λ_2 vortex detection field of Jeong and Hussain [54]. The image shows snapshots taken at $t = \{100, 200, 300, 400\}$. These depict the stages of the initial approach of the rings, their rapid radial expansion, the development of the instability and the emergence of many small vortex rings, respectively.

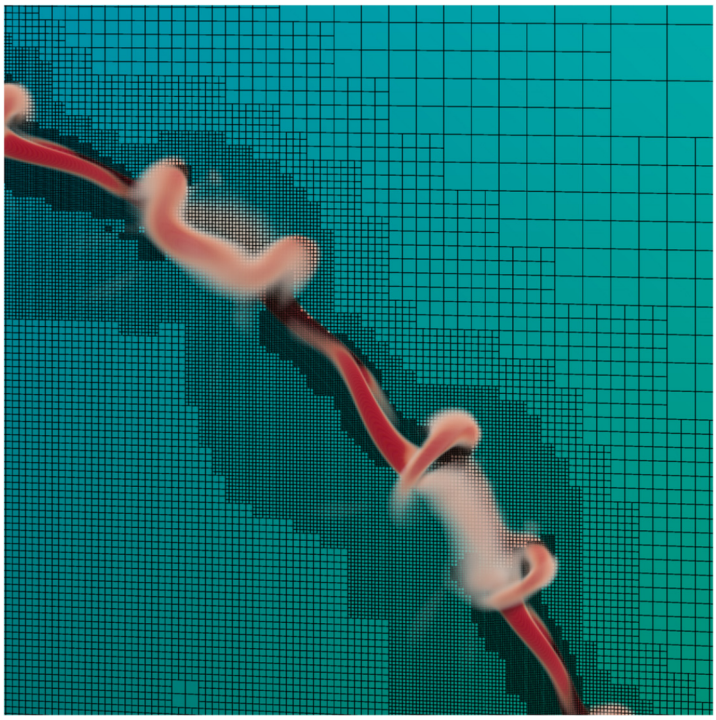


Fig. 20. A zoom in on a volumetric rendering of the λ_2 vortex detection field of Jeong and Hussain [54] with a slice of the adaptive octree grid and a blue and green background. The snapshot it taken at $t = 350$. The maximum level of refinement is 12, which corresponds to an equidistant mesh with 4096^3 cells.

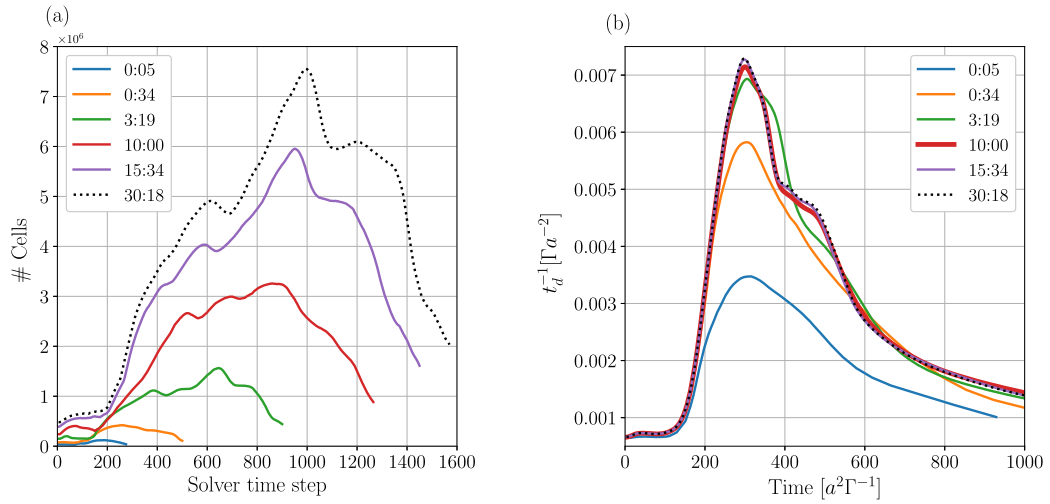


Fig. 21. Results from the head-on vortex ring collision obtained with the 4th-order accurate solver. The number of used grid cells for each time-integration step (a) and the inverse dissipation timescale (t_d^{-1}, see Eq. (67)) is plotted as a function of the physical time (b). The legends indicate the wall-clock time for each run (HH:MM).

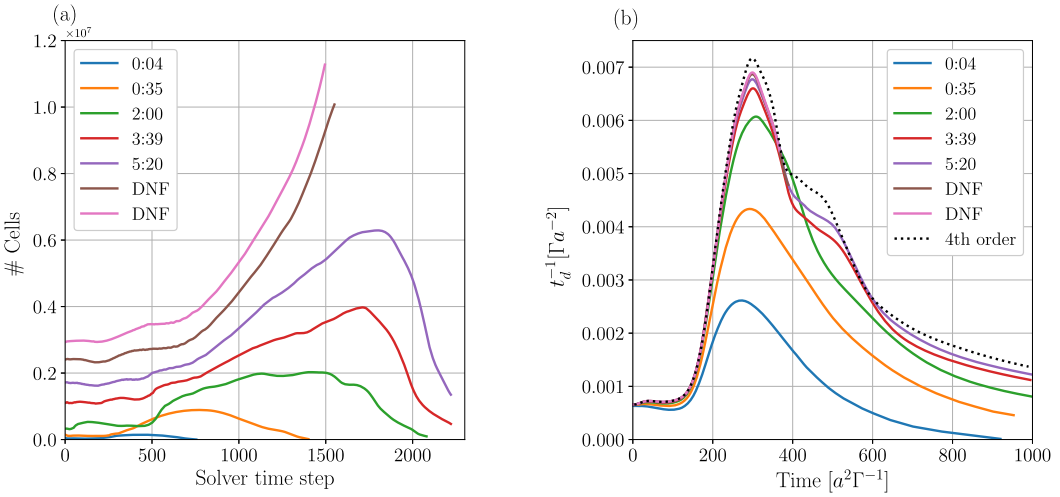


Fig. 22. Results from the head-on vortex ring collision obtained with the 2nd-order accurate solver. The number of used grid cells for each time-integration step (a) and the inverse dissipation timescale (t_d^{-1}, see Eq. (67)) is plotted as a function of the physical time (b). The legends indicate the wall-clock time for each run (HH:MM) or if they have stopped due to the limited memory (DNF, see text).

4. Summary and conclusion

We have presented an adaptive solver for incompressible flow problems that uses fourth-order accurate discrete approximations. The adaptive grid structure is based on the adaptive quad/octree toolbox provided by the Basilisk code as free software. The primary variables are the velocity components which are discretized as face averages. The tendencies due to advection and diffusion of these quantities are computed using finite difference schemes at the co-located grid of vertices. For this step, we have derived a 4th-order accurate compact upwind scheme. The projection step exploits the fact that the velocities are defined as face averages, and hence, the cell-averaged divergence condition of the numerical solution can be satisfied exactly. The solution is advanced in time using a 5-stage 4th-order accurate Runge-Kutta method. Between each solver step the numerical mesh can be adapted based on the higher-order polynomial content of the solution fields in conjunction with a single user-defined refinement criterion. Further, the formulations allow active and passive tracers to be advected and diffused alongside the evaluation of the flow evolution. The implementation of the solver, the setup files for the cases presented herein and additional tests are freely available via the links to their online locations.

The solver design has been applied to a set of flow problems ranging from simple setups with analytical solutions to a 3D flow scenario with developing sharp gradients in the flow field and vortex instabilities. The analysis shows that indeed, the approximated solution converges more quickly to the asymptotic solution than the 2nd-order accurate alternative. Further,

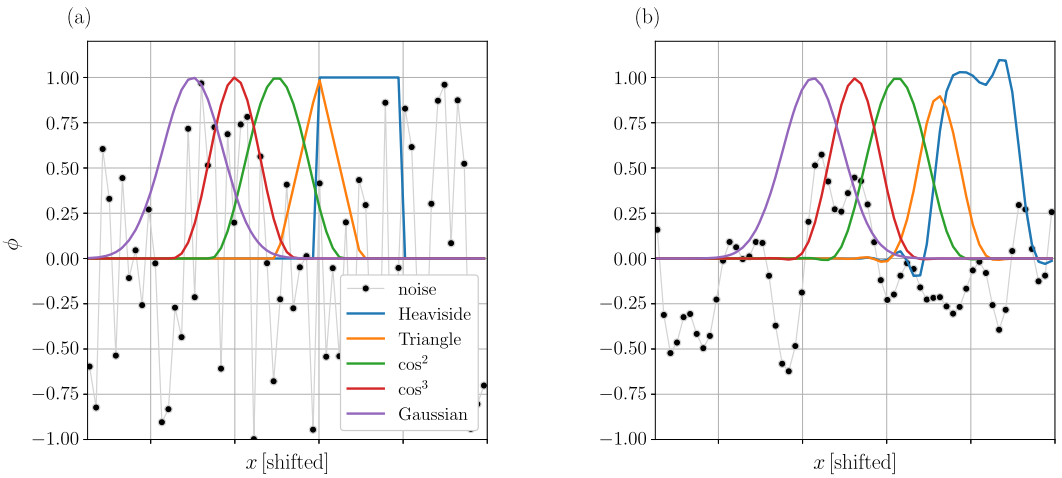


Fig. 23. A plot of the Initialized test functions (see Eqs. (68) – (73)) in a) and in b) the solution for the advection problem (cf. (74)) at $t = 1$. The data of the various test functions are shifted along the x direction for clarity of presentation.

the refinement indicator appears to consistently refine and coarsen the mesh corresponding to the flow evolution. As such, to proposed solver is attractive to model flow configurations with a high degree of localization in space and time. This is showcased in the final example concerning the head-on collision of two vortex rings in Sect. 3.8.

CRediT authorship contribution statement

J. Antoon van Hooft: Conceptualization, Methodology, Software, Validation, Visualization, Writing – original draft. **Stéphane Popinet:** Conceptualization, Methodology, Software, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare no competition of interest.

Acknowledgements

J.A. van Hooft received funding from the Plantenna project which was initiated by the 4TU federation.

Appendix A. 1D advection

The advection scheme (Eq. (28)) was derived using the assumption of sufficiently smooth data. In addition to the convergence tests of Sect. 3, the advection scheme is tested for 1D functions which are initialized with varying degrees of 'smoothness' (cf. Fig. 23a),

$$\text{Noise}: \quad n(x, t=0) \in [-1, 1] \text{ (randomized)} \tag{68}$$

$$\text{Heaviside}: \quad H(x, t=0) = \begin{cases} 1 & \text{for } \|x\| \leq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{69}$$

$$\text{Triangle}: \quad T(x, t=0) = \begin{cases} 1+x & \text{for } -1 < x \leq 0, \\ 1-x & \text{for } \ \ 0 \leq x < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{70}$$

$$\text{Squared cosine sect.}: \quad C_2(x, t=0) = \begin{cases} \cos^2(x) & \text{for } \|x\| \leq \frac{\pi}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{71}$$

$$\text{Cubed cosine sect.}: \quad C_3(x, t=0) = \begin{cases} \cos^3(x) & \text{for } \|x\| \leq \frac{\pi}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{72}$$

$$\text{Gaussian}: \quad G(x, t=0) = e^{-x^2}. \tag{73}$$

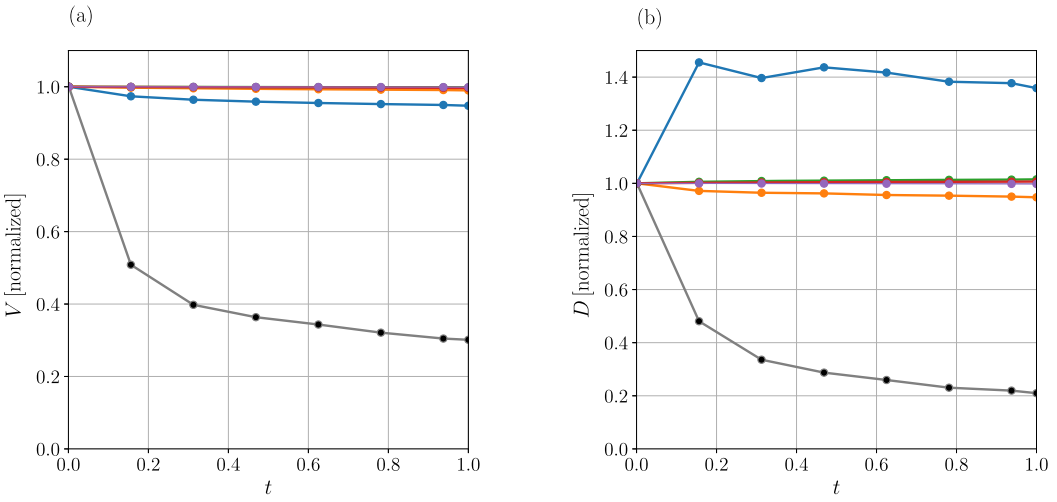


Fig. 24. The evolution of the normalized variance (V, a) and total variation (D, b) for the various test functions (see text). The data is normalized with their initial values and the color coding is the same as in Fig. 23.

The test functions are 'advected' by solving for a dummy variable $\phi(x,t)$,

$$\frac{\partial \phi}{\partial t} = -\frac{\partial \phi}{\partial x}, \tag{74}$$

using the discretizations of Eqs. (15) and (28) for time and space, respectively. The test functions are initialized in a periodic domain of size $[-5,5]$ using $N = 64$ grid points and the time step size is based on the CFL condition (cf. Eq. (19)). Fig. 23 shows the initialized data and the solution at $t = 1$. We see that the numerical errors can introduce new extrema in the data. Further, the noise, Heaviside and triangle functions have become notably smoothed. This motivates to plot the variance (i.e. $V_\phi = \sum_{i=0}^{N} \phi_i^2$) and total variation (i.e. $D_\phi = \sum_{i=0}^{N} \|\phi_i - \phi_{i+1}\|$) over time. The data is plotted in Fig. 24. The scheme appears to be dissipative for the second-order moment (V) but is not always total-variation diminishing.

References

[1] J.H. Ferziger, M. Perić, R.L. Street, Computational Methods for Fluid Dynamics, vol. 3, Springer, 2002.
[2] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[3] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[4] S. Portegies Zwart, Computational astrophysics for the future, Science 361 (2018) 979–980.
[5] N.K.-R. Kevlahan, T. Dubos, Wavetrisk-1.0: an adaptive wavelet hydrostatic dynamical core, Geosci. Model Dev. 12 (2019) 4901–4921.
[6] A. Berny, S. Popinet, T. Séon, L. Deike, Statistics of jet drop production, Geophys. Res. Lett. (2021) e2021GL092919.
[7] J.A. van Hooft, S. Popinet, C.C. van Heerwaarden, S.J. van der Linden, S.R. de Roode, B.J. van de Wiel, Towards adaptive grids for atmospheric boundary-layer simulations, Bound.-Layer Meteorol. 167 (2018) 421–443.
[8] D. Donzis, P. Yeung, Resolution effects and scaling in numerical simulations of passive scalar mixing in turbulence, Physica D 239 (2010) 1278–1287, At the boundaries of nonlinear physics, fluid mechanics and turbulence: where do we stand? Special issue in celebration of the 60th birthday of K.R. Sreenivasan.
[9] J.P. Mellado González, The evaporatively driven cloud-top mixing layer, J. Fluid Mech. 660 (2010) 5–36.
[10] J. Behrens, Adaptive atmospheric modeling scientific computing at its best, Comput. Sci. Eng. 7 (2005) 76–83.
[11] F. Naddei, M. de la Llave Plata, V. Couaillier, F. Coquel, A comparison of refinement indicators for p-adaptive simulations of steady and unsteady flows using discontinuous Galerkin methods, J. Comput. Phys. 376 (2019) 508–533.
[12] M.O. Deville, P.F. Fischer, P.F. Fischer, E. Mund, et al., High-Order Methods for Incompressible Fluid Flow, vol. 9, Cambridge University Press, 2002.
[13] R.R. Chowdhury, Higher-order adaptive methods for fluid-dynamics, Ph.D. thesis, Sorbonne Université, 2018.
[14] M.A. Kopera, F.X. Giraldo, Analysis of adaptive mesh refinement for IMEX discontinuous Galerkin solutions of the compressible Euler equations with application to atmospheric simulations, J. Comput. Phys. 275 (2014) 92–117.
[15] O. Zanotti, F. Fambri, M. Dumbser, A. Hidalgo, Space–time adaptive ADER discontinuous Galerkin finite element schemes with a posteriori sub-cell finite volume limiting, Comput. Fluids 118 (2015) 204–224.
[16] B. Bonev, J.S. Hesthaven, F.X. Giraldo, M.A. Kopera, Discontinuous Galerkin scheme for the spherical shallow water equations with applications to tsunami modeling and prediction, J. Comput. Phys. 362 (2018) 425–448.
[17] F. Fambri, Discontinuous Galerkin methods for compressible and incompressible flows on space–time adaptive meshes: toward a novel family of efficient numerical methods for fluid dynamics, Arch. Comput. Methods Eng. 27 (2020) 199–283.
[18] D. Meagher, Geometric modeling using octree encoding, Comput. Graph. Image Process. 19 (1982) 129–147.
[19] S. Popinet, A quadtree-adaptive multigrid solver for the Serre–Green–Naghdi equations, J. Comput. Phys. 302 (2015) 336–358.
[20] M. Ben-Artzi, J.-P. Croisille, D. Fishelov, A cartesian compact scheme for the Navier–Stokes equations in streamfunction formulation in irregular domains, J. Sci. Comput. 81 (2019) 1386–1408.
[21] M.H. Carpenter, C.A. Kennedy, Fourth-order 2N-storage Runge–Kutta schemes, NASA LangleyResearch Center Tech. Rep. TM-109112, 1994.
[22] M. Crouzeix, Une méthode multipas implicite-explicite pour l'approximation des équations d'évolution paraboliques, Numer. Math. 35 (1980) 257–276.
[23] U.M. Ascher, S.J. Ruuth, B.T. Wetton, Implicit-explicit methods for time-dependent partial differential equations, SIAM J. Numer. Anal. 32 (1995) 797–823.

[24] G.E. Karniadakis, M. Israeli, S.A. Orszag, High-order splitting methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 97 (1991) 414–443.
[25] J. Verwer, J. Blom, W. Hundsdorfer, An implicit-explicit approach for atmospheric transport-chemistry problems, Appl. Numer. Math. 20 (1996) 191–209.
[26] M. Ben-Artzi, J.-P. Croisille, D. Fishelov, A high order compact scheme for the pure-streamfunction formulation of the Navier–Stokes equations, J. Sci. Comput. 42 (2010) 216–250.
[27] A.J. Chorin, The numerical solution of the Navier–Stokes equations for an incompressible fluid, Bull. Am. Math. Soc. 73 (1967) 928–931.
[28] E.M. De Angelis, G. Coppola, F. Capuano, L. De Luca, Derivation of new staggered compact schemes with application to Navier–Stokes equations, Appl. Sci. 8 (2018) 1066.
[29] S.K. Lele, Compact finite difference schemes with spectral-like resolution, J. Comput. Phys. 103 (1992) 16–42.
[30] C. Chen, X. Zhang, Z. Liu, Y. Zhang, A new high-order compact finite difference scheme based on precise integration method for the numerical simulation of parabolic equations, Adv. Differ. Equ. 2020 (2020) 1–28.
[31] A. Brandt, S. McCormick, J. Ruge, Multigrid methods for differential eigenproblems, SIAM J. Sci. Stat. Comput. 4 (1983) 244–260.
[32] P.K. Farsoiya, S. Popinet, L. Deike, Bubble-mediated transfer of dilute gas in turbulence, J. Fluid Mech. 920 (2021).
[33] W. Aniszewski, Y. Saade, S. Zaleski, S. Popinet, Planar jet stripping of liquid coatings: numerical studies, Int. J. Multiph. Flow 132 (2020) 103399.
[34] G. Kirstetter, F. Bourgin, P. Brigode, O. Delestre, Real-time inundation mapping with a 2D hydraulic modelling tool based on adaptive grid refinement: the case of the October 2015 French riviera flood, in: Advances in Hydroinformatics, Springer, 2020, pp. 335–346.
[35] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, J. Comput. Phys. 228 (2009) 5838–5866.
[36] K.J. Fidkowski, D.L. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, AIAA J. 49 (2011) 673–694.
[37] M. Antuono, Tri-periodic fully three-dimensional analytic solutions for the Navier–Stokes equations, J. Fluid Mech. 890 (2020).
[38] G.I. Taylor, A.E. Green, Mechanism of the production of small eddies from large ones, Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci. 158 (1937) 499–521.
[39] A. Lifschitz, E. Hameiri, Local stability conditions in fluid dynamics, Phys. Fluids A, Fluid Dyn. 3 (1991) 2644–2651.
[40] D. Sipp, L. Jacquin, Elliptic instability in two-dimensional flattened Taylor–Green vortices, Phys. Fluids 10 (1998) 839–849.
[41] T.K. Sengupta, N. Sharma, A. Sengupta, Non-linear instability analysis of the two-dimensional Navier–Stokes equation: the Taylor-Green vortex problem, Phys. Fluids 30 (2018) 054105.
[42] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2003) 572–600.
[43] P.-Y. Lagrée, L. Staron, S. Popinet, The granular column collapse as a continuum: validity of a two-dimensional Navier–Stokes model with a μ (I)-rheology, J. Fluid Mech. 686 (2011) 378–408.
[44] A. Castillo-Castellanos, A. Sergent, M. Rossi, Reversal cycle in square Rayleigh—Bénard cells in turbulent regime, J. Fluid Mech. 808 (2016) 614–640.
[45] D.L. Brown, Performance of under-resolved two-dimensional incompressible flow simulations, J. Comput. Phys. 122 (1995) 165–183.
[46] A. Hokpunna, M. Manhart, Compact fourth-order finite volume method for numerical solutions of Navier–Stokes equations on staggered grids, J. Comput. Phys. 229 (2010) 7545–7570.
[47] A. Robert, Bubble convection experiments with a semi-implicit formulation of the Euler equations, J. Atmos. Sci. 50 (1993) 1865–1873.
[48] F.X. Giraldo, M. Restelli, A study of spectral element and discontinuous Galerkin methods for the Navier–Stokes equations in nonhydrostatic mesoscale atmospheric modeling: equation sets and test cases, J. Comput. Phys. 227 (2008) 3849–3877.
[49] P. Orlandi, Vortex dipole rebound from a wall, Phys. Fluids A, Fluid Dyn. 2 (1990) 1429–1436.
[50] H. Clercx, G. Van Heijst, Two-dimensional Navier–Stokes turbulence in bounded domains, Appl. Mech. Rev. 62 (2009).
[51] W. Kramer, H. Clercx, G. van Heijst, Vorticity dynamics of a dipole colliding with a no-slip wall, Phys. Fluids 19 (2007) 126603.
[52] T. Lim, T. Nickels, Instability and reconnection in the head-on collision of two vortex rings, Nature 357 (1992) 225–227.
[53] B. Dorschner, K. Yu, G. Mengaldo, T. Colonius, A fast multi-resolution lattice Green's function method for elliptic difference equations, J. Comput. Phys. 407 (2020) 109270.
[54] J. Jeong, F. Hussain, On the identification of a vortex, J. Fluid Mech. 285 (1995) 69–94.