3DFLUX: A high-order fully three-dimensional flux integral solver for the scalar transport equation

Emmanuel Germaine, Laurent Mydlarski, Luca Cortelezzi*

Department of Mechanical Engineering, McGill University, Montreal, Quebec, Canada

A R T I C L E   I N F O

Article history:
Received 22 June 2012
Received in revised form 21 December 2012
Accepted 9 January 2013
Available online 25 January 2013

Keywords:
Flux integral method
Multidimensional transport equation
Selective flux-limiters
Monotonicity preservation
Advection–diffusion equation

A B S T R A C T

We present a detailed derivation of a high-order, fully three-dimensional, conservative, monotonicity preserving, flux integral method for the solution of the scalar transport equation. This algorithm, named 3DFLUX, produces highly accurate solutions that are nearly unaffected by numerical dissipation, at a realistic computational cost. The performance of 3DFLUX is characterized by means of several challenging multidimensional tests. 3DFLUX is nominally third-order in space and second-order in time, however, at low Courant numbers, it appears to be superconvergent and, depending on the problem solved, is fourth-order or higher in space. Finally, 3DFLUX is used to simulate advection-diffusion of a complex temperature field in an incompressible turbulent flow of practical relevance, and its results are in excellent agreement with experimental measurements.

© 2013 Elsevier Inc. All rights reserved.

1. Introduction

The scalar transport equation is encountered in a wide range of scientific applications. For example, it models the transport of a scalar (e.g. mass, energy, etc.) in a fluid flow, where it is known as the advection–diffusion equation. In economics, it is used to model the temporal evolution of a financial market, where it is known as the Black–Scholes equation. It constitutes an important tool for medical image analysis, where it takes the form of the Fokker–Planck equation. In biology, it can be used to simulate the dynamics of living species in their environment.

In engineering applications, the scalar transport equation is most often solved by finite difference or finite volume methods. The latter are employed to simulate fluid flows because of their strictly conservative formulation and their flexibility, which enables the solution of problems in complex geometries. Existing finite volume methods [1,2] are very robust, low-order and widely used in commercial software packages. They are reliable for industrial applications, but their accuracy is insufficient for scientific research.

Numerical methods that simulate the transport of a scalar are divided into two classes: Eulerian and Lagrangian. Eulerian methods solve the equation on a spatially fixed grid. Lagrangian methods solve the equation by tracking the path of the particles, and the nodes of the grid (used to discretize the computational domain) move with the particles. Lagrangian methods are more stable than Eulerian ones, but they require a re-meshing procedure that can be complex to implement and computationally expensive.

Semi-Lagrangian methods [3] constitute an appealing alternative approach as they combine the positive aspects of Eulerian and Lagrangian methods. Semi-Lagrangian methods solve the scalar transport equation on a fixed grid by tracking a different set of particles at each time-step. They are widely used in meteorological applications because they are more stable.

* Corresponding author.

E-mail addresses: emmanuel.germaine@mail.mcgill.ca (E. Germaine), laurent.mydlarski@mail.mcgill.ca (L. Mydlarski), luca.cortelezzi@mail.mcgill.ca (L. Cortelezzi).

0021-9991/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2013.01.010
than Eulerian schemes and simpler to implement than Lagrangian ones. The main disadvantage of semi-Lagrangian methods is that, in general, they are not conservative.

A conservative formulation of a semi-Lagrangian method was first presented by Leonard et al. [4] using a flux integral approach. Leonard’s scheme provides an explicit single-step, forward-in-time, conservative control volume update of the unsteady two-dimensional advection-diffusion equation. Other conservative, multidimensional, flux-based semi-Lagrangian methods for purely advective flows have been presented by Leveque [5] and Harris et al. [6]. The advantage of using a multidimensional formulation is to avoid splitting techniques, which solve multidimensional problems as a sequence of one-dimensional schemes [7–9]. Splitting techniques considerably simplify the algorithm on Cartesian grids, at the cost of inducing local splitting errors [10]. However, this advantage is lost on non-Cartesian grids where their implementation is complex compared to a multidimensional scheme [6].

Crucial in the flux integral method is the type of interpolation used to approximate the scalar field within the cells that discretize the domain. High-order schemes require the use of high-order interpolations, which can degrade the monotonicity of the solution where the gradients are poorly resolved. This degradation leads to unphysical over- and undershooting, and oscillations of the solution near its extrema. These detrimental effects can, however, be avoided and monotonicity can be enforced by use of a flux-limiter scheme, for example, the popular flux-corrected transport scheme proposed by Zalesak [11]. Complementing the flux integral method with a flux-corrected transport scheme [12] allows to correct the value of the fluxes after they have been computed and, consequently, enforce the monotonicity of the solution. Thuburn [13] proposed a multidimensional implementation of a flux-limiter scheme, which is computationally less expensive than Zalesak’s method, but has more stringent stability restrictions [14].

The main disadvantage of the flux-limiter schemes is that they tend to smooth the solution, consequently degrading its global accuracy. To overcome this disadvantage, Bloxey and Durran [15] proposed an algorithm that only activates the flux-limiter in regions where the gradients are poorly resolved. Their algorithm preserves the global accuracy of the flux integral method at the expense of negligible violations in the monotonicity of the solution. Recently, Harris et al. [6] confirmed the effectiveness of the selective limiter scheme in conjunction with a two-dimensional conservative semi-Lagrangian method.

We leverage and extend the schemes and results discussed above to derive and implement a highly accurate, fully three-dimensional, cost-effective flux integral method, named 3DFlux, for the solution of the scalar transport equation. 3DFlux is fully explicit and multidimensional. The latter property guarantees that 3DFlux is free of splitting errors and results in a better convergence rate of the numerical errors when compared to widely used one-dimensional techniques, such as the piecewise parabolic method (PPM) [16], the weighted essentially non-oscillatory (WENO) method [17], and the recently-proposed jet scheme [18,19], which is based on the level-set method. 3DFlux is nominally third-order in space and second-order in time. It generates low numerical dissipation and anisotropic distortion [20]. 3DFlux allows the solution of problems of practical interest at a reasonable computational cost.

In the derivation of 3DFlux, we restrict the value of the Courant number to be one or less because our research focuses on the advection–diffusion of the smaller scales of turbulent flows. However, there are no conceptual difficulties in extending 3DFlux to Courant numbers greater than one to target applications in atmospheric science. We characterize the convergence rate, accuracy and cost-efficiency of 3DFlux by running several multidimensional tests: a two-dimensional purely advective mixing/unmixing problem [19], the rotation of a slotted cylinder [21], a two-dimensional purely advective unsteady deformational flow [15], and a three-dimensional advection–diffusion problem. We show that at the highest possible Courant number — one in our implementation — 3DFlux results in a noticeably higher accuracy and better cost-efficiency than the currently available numerical schemes having the same convergence rate. We also show that, at low enough Courant numbers, 3DFlux unexpectedly appears to be superconvergent. Lastly, we complete the validation of 3DFlux by simulating an experiment in which a scalar (temperature) is released from a heated line source in a fully developed turbulent channel flow. The statistics produced by 3DFlux are in remarkable agreement with those measured in the experiment [22].

The remainder of this paper is organized as follows: In Section 2, a detailed derivation of the numerical method is presented. In Section 3, two- and three-dimensional numerical tests are performed to validate the performance of our scheme, including a three-dimensional direct numerical simulation (DNS) of a thermal plume in a fully turbulent channel flow that exemplifies the potential of our scheme to simulate flows of interest in engineering applications. Lastly, conclusions are presented in Section 4.

2. Numerical method

The equation governing the unsteady, three-dimensional transport of a scalar property, $\phi$, can be written in a generic form as

$$\frac{\partial (\rho \phi)}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u} \phi) = \vec{\nabla} \cdot (\Gamma \vec{\nabla} \phi) + S,$$

(1)

where $\rho$ is a physical property of the system, $\vec{u} = (u, v, w)$ is the advecting velocity vector field, $\Gamma$ is the diffusion coefficient of the scalar property, and $S$ represents the source (or sink) terms.
Eq. (1) states that the rate of change of $\phi$ depends on the balance between advective transport, diffusive transport and source or sink terms. For the sake of clarity, we limit our discussion to the well known advection-diffusion equation governing the time evolution of the internal energy of a flow of a simple compressible substance, where $\rho$ is the fluid density, $\phi$ is the temperature, $\Gamma = \kappa/c_p$, where $\kappa$ is the thermal conductivity of the fluid and $c_p$ is its specific heat capacity at constant pressure, and $S = \dot{q}/c_p$, where $\dot{q}$ quantifies the volumetric heat source/sink terms. Obviously, any scalar transport equation that can be cast in the form of Eq. (1) can be solved using the algorithm presented herein.

The density of the fluid, $\rho$, and the coefficient of diffusion, $\Gamma$, are related to the fluid temperature, $\phi$, and the fluid pressure, $p$, by the constitutive equations, i.e. $\rho = \rho(\phi, p)$ and $\Gamma = \Gamma(\phi, p)$. These relations are usually nonlinear and often cause major difficulties in the solution of Eq. (1). Consequently, we restrict our attention to flow where $\rho$ and $\Gamma$ are constant. Under these assumptions, Eq. (1) is fully determined and still correctly models the evolution of a scalar in a wide range of engineering applications.

2.1. Discretization of the scalar transport equation

In the flux integral method, the computational domain is discretized with a number of non-overlapping control volumes, or computational cells, of size $\Delta V = \Delta x \times \Delta y \times \Delta z$. The grid nodes are cell-centered and the value of the scalar property, $\phi$, at a node represents the average value of $\phi$ within the control volume $\Delta V$. The components of the velocity field are represented on three different staggered grids so that the value of the three components of the velocity, $u$, $v$, and $w$, are stored at the center of each side of the control volume. The flux integral method solves Eq. (1) by estimating the fluxes exchanged between adjacent cells. With a staggered arrangement, the velocity components are directly available at the cell-sides, where they are needed for the calculation of the fluxes. This method is a multidimensional finite volume method.

To implement the flux integral method, we need to introduce a nomenclature [1] to designate the location of the cell center and cell sides. Let us focus on a grid point $P$, as shown in Fig. 1. Each neighboring grid point of $P$ is identified by one or more of the following six capital letters: $W$, $E$, $N$, $S$, $R$, $F$. These letters identify the nodes at the west, east, north, south, back and front sides of the computational cell containing $P$, respectively. For example, the node $W$ is the closest neighbor to $P$ located in the negative $x$-direction, the node $SW$ is the closest neighbor to $P$ located in the negative $x$- and $y$-directions, the node $WW$ is the second closest neighbor to $P$ located in the negative $x$-direction, and so on. The same lowercase letters are used to identify the sides of the computational cell, or control volume, containing the grid point $P$. For example, $\phi_w$ and $u_w$ represent, respectively, the average values of the scalar $\phi$ and the $u$-component of the velocity field on the west side of the cell $P$.

The scalar transport equation can be written in integral form as follows

$$\int_M \int_{\Delta V} \frac{\partial (\rho \phi)}{\partial t} dV dt = \int_M \int_{\Delta V} \left[ -\nabla \cdot (\rho \mathbf{u} \phi) + \nabla \cdot (\Gamma \nabla \phi) + S \right] dV dt,$$

(2)

where $\Delta V$ is the volume of the computational cell. Since the value of $\phi$ at the center of a cell is the average of $\phi$ within that cell, the left-hand side of Eq. (2) can be written as

$$\int_M \int_{\Delta V} \frac{\partial (\rho \phi)}{\partial t} dV dt = \rho (\phi^{t-M} - \phi^t) \Delta V.$$

(3)

Fig. 1. (a) Computational cell of volume $\Delta V$. (b) Arrangement of the grid nodes and the staggered velocity components in a two-dimensional view of the three-dimensional grid.
Since the flow is assumed to be of constant density, the first term on the right-hand side of Eq. (2) can be written as

\[
- \int_M \int_{M'} \left[ \frac{\partial (\rho u_w \phi_w)}{\partial x} + \frac{\partial (\rho \phi_w)}{\partial y} + \frac{\partial (\rho w \phi_w)}{\partial z} \right] dV \, dt = \Delta y \Delta z \int_M (\rho u_w \phi_w - \rho u_e \phi_e) \, dt + \Delta x \Delta z \int_M (\rho v_i \phi_i - \rho v_n \phi_n) \, dt
\]

\[
+ \Delta x \Delta y \int_M (\rho w_b \phi_b - \rho w_f \phi_f) \, dt.
\]

Similarly, the second term on the right-hand side of Eq. (2) can be rewritten as

\[
\int_M \int_{M'} \left[ \frac{\partial}{\partial x} \left( \Gamma \frac{\partial \phi}{\partial x} \right) + \frac{\partial}{\partial y} \left( \Gamma \frac{\partial \phi}{\partial y} \right) + \frac{\partial}{\partial z} \left( \Gamma \frac{\partial \phi}{\partial z} \right) \right] dV \, dt = \Delta y \Delta z \int_M \Gamma \left( \frac{\partial \phi}{\partial x} \bigg|_{x_i} \right) \left( \frac{\partial \phi}{\partial y} \bigg|_{y_i} \right) \, dt + \Delta x \Delta z \int_M \Gamma \left( \frac{\partial \phi}{\partial y} \bigg|_{y_i} \right) \left( \frac{\partial \phi}{\partial z} \bigg|_{z_i} \right) \, dt
\]

\[
+ \Delta x \Delta y \int_M \Gamma \left( \frac{\partial \phi}{\partial z} \bigg|_{z_i} \right) \left( \frac{\partial \phi}{\partial y} \bigg|_{y_i} \right) \, dt.
\]

Note that the advective and diffusive terms on the west face of the control volume can be rewritten as

\[
\Delta y \Delta z \int_M (\rho u_w \phi_w) \, dt = (\rho u_w \phi_w) \Delta y \Delta z \Delta t
\]

and

\[
\Delta y \Delta z \int_M -1 \frac{\partial \phi}{\partial x} \bigg|_{x_i} \, dt = \langle \frac{\partial \phi}{\partial x} \rangle \Delta y \Delta z \Delta t,
\]

where angular brackets \( \langle \cdot \rangle \) represent the time average over a time interval \( \Delta t \). Finally, the last term on the right-hand side of Eq. (2), the source term, can be rewritten as

\[
\int_M \int_{M'} S \, dV \, dt = \langle S \rangle \frac{\Delta t}{\rho \Delta V}.
\]

Hence, Eq. (2) can be rewritten in flux form as follows

\[
\phi^{i+1} = \phi^i + f_w - f_e + f_s - f_n + f_b - f_f + \langle S \rangle \frac{\Delta t}{\rho \Delta V},
\]

where the terms \( f_w, f_e, f_s, f_n, f_b \) and \( f_f \), denote the sum of the advective and the diffusive fluxes across the west, east, south, north, front and back sides of the control volume \( P \), respectively. \( f_w, f_e \) and \( f_b \) are defined as

\[
f_w = \frac{\Delta t \Delta y \Delta z}{\rho \Delta V} \left( \langle \rho u_w \phi_w \rangle - \langle \frac{\partial \phi}{\partial x} \bigg|_{x_i} \rangle \right),
\]

\[
f_s = \frac{\Delta t \Delta x \Delta y}{\rho \Delta V} \left( \langle \rho u_s \phi_s \rangle - \langle \frac{\partial \phi}{\partial y} \bigg|_{y_i} \rangle \right),
\]

\[
f_b = \frac{\Delta t \Delta x \Delta y}{\rho \Delta V} \left( \langle \rho u_b \phi_b \rangle - \langle \frac{\partial \phi}{\partial z} \bigg|_{z_i} \rangle \right).
\]

Eq. (9) is strictly conservative when the flux across two adjacent control volumes is the same. Consequently, for a given control volume, only the fluxes \( f_w, f_s \) and \( f_b \) have to be calculated as the three remaining fluxes, i.e. \( f_e, f_n \) and \( f_f \), are directly determined as follows

\[
(f_e)_{ijk} = (f_w)_{ijk},
\]

\[
(f_n)_{ijk} = (f_s)_{ijk},
\]

\[
(f_f)_{ijk} = (f_b)_{ijk},
\]

where the indices \( i, j \) and \( k \) represent the components along the coordinate axes \( x, y \) and \( z \), respectively, and \( i' = i + 1, j' = j + 1 \) and \( k' = k + 1 \). Note that Eq. (9) is equivalent to the two-step Lax–Wendroff scheme, i.e. a second-order in time method, when the fluxes are computed at the intermediate time step \( t + \Delta t/2 \) [23, p. 1040].

Note that in Eqs. (10), the values of the scalar at the side of a cell, i.e. \( \phi_w, \phi_s \) and \( \phi_b \), and the gradients of \( \phi \) normal to the side of a cell, i.e. \( \partial \phi / \partial x|_{x_i}, \partial \phi / \partial y|_{y_i} \) and \( \partial \phi / \partial z|_{z_i} \), have to be computed before performing the time average of the fluxes. The values of \( \phi \) and its gradients are unknown at the sides of the control volume, but they can be interpolated from the values of \( \phi \) known at the center of the cells. The computation of the fluxes is presented in detail in the next subsections.

2.2. Piecewise polynomial interpolation of face quantities

To compute the fluxes, the unknown values of the scalar and its gradients at the sides of a cell can be obtained by interpolating the known values of \( \phi \) at the center of the surrounding cells. One of the most common and simple methods
is piecewise linear interpolation. This interpolation leads to very efficient second-order schemes whose applicability, however, is limited to diffusion-dominated problems. In fact, these second-order schemes generate questionable solutions, which are affected by spurious oscillations when the magnitude of the advective terms is greater than that of the diffusive ones. In fluid mechanics, for example, the Péclet number quantifies the ratio of the advective and diffusive terms. Hence, the validity of such second-order schemes is limited to simulations of fluid flows where the local Péclet number, i.e., Péclet number based on the local velocity, diffusivity, and cell dimensions, is lower than two [1].

The limitations of such second-order schemes can be overcome using upwind schemes, such as the popular first-order upwind schemes that use piecewise constant interpolation, where the unknown value of \( \phi \) at any given side of a cell is set equal to the known value of \( \phi \) at the center of the adjacent upstream cell. In other words, the west value \( \phi_w \) is taken equal to \( \phi_p \), the value at the center of the adjacent upstream cell. Otherwise it is equal to \( \phi_p \) (see Fig. 1). Despite their simplicity, speed and robustness, first-order schemes are usually not adequate for accurate simulations because they exhibit high numerical diffusion and poor convergence properties in addition to producing erroneous solutions when the flow is not aligned with the grid lines [2,24].

Higher-order upwind schemes have subsequently been introduced over the years, such as the popular Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Terms (QUICKEST) scheme, proposed by Leonard [25]. QUICKEST is based on a quadratic polynomial interpolation which leads to a third-order scheme. It has a limitation, however, as the interpolation performed by QUICKEST is locally one-dimensional. As a result, QUICKEST produces anisotropically distorted results when used to solve multidimensional problems [20]. To avoid these detrimental effects, Leonard et al. [4] proposed a Uniformly Third-Order Polynomial Interpolation Algorithm (UTOPIA) implemented with a flux integral method. Leonard et al. [4] characterized the performance of UTOPIA by solving a purely two-dimensional advective problem: the transport of a Gaussian bell in a rotating flow. Subsequently, Leveque [5] proposed a three-dimensional scheme, equivalent to UTOPIA, for solving purely advective problems.

Over the years, flux-based methods have been used for atmospheric transport and chemical models [26]. State-of-the-art solvers, such as COCO (CCSR Ocean Component Model), developed by the Center for Climate System Research [27], combine UTOPIA and QUICKEST to simulate ocean circulation and transport. More specifically, COCO separates the two-dimensional planar advection using UTOPIA, the vertical advection using QUICKEST and diffusion using a second-order, finite central difference scheme. This awkward combination of algorithms has been devised to circumvent the complexity of implementing UTOPIA in three-dimensions.

Leveraging Leonard's work [4], this article provides the reader with a detailed derivation and a careful validation of a new, fully three-dimensional, flux integral solver for the scalar transport equation. Our solver (3DFLUX) is third-order accurate under the assumption that the fluid properties are constant.

The exact solution of the scalar transport Eq. (1), \( \phi = \phi(x, y, z, t) \), is discretized over a number of cells covering the computational domain. At any time \( t \), the value stored at the center of a cell is the average of \( \phi(x, y, z, t) \) over the volume of the cell. A three-dimensional piecewise quadratic interpolation is used to calculate the fluxes of the scalar across the sides of a cell. The exact solution, at each time step, is approximated within each cell by a quadratic polynomial of the form

\[
\psi(x, y, z) = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} x^i y^j z^k,
\]  

where \( c_{ijk} \) are ten unknown coefficients. To determine these coefficients at each time step, we impose that the average value of the polynomial \( \psi \) and its first-order, second-order and cross derivatives are equal to the value of the average solution at the center of the cell and its corresponding derivatives.

As an example, we present the ten conditions used to determine the coefficients, \( c_{ijk} \), of the interpolant, \( \psi_w \), within the west cell (see Fig. 2), where the subscript \( W \) indicates the name of the cell. As a first condition, we impose the average value of the interpolant over the west cell to be equal to the value of the average of the scalar over the west cell.

![Fig. 2. Computational cells W and P.](image)
\[
\frac{1}{\Delta V} \int_V \psi_w \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \phi_w,
\]
\(\text{(13)}\)

where \(\Delta x/2, \Delta y/2, \Delta z/2\) are the coordinates of the center of the west cell (see Fig. 2). Similarly, the remaining nine conditions are obtained imposing that the average value of the first, second and cross derivatives of the interpolant over the west cell are equal to the corresponding derivatives of the average of the scalar over the west cell. We have

\[
\frac{1}{\Delta V} \int_V \frac{\partial \psi}{\partial x} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial \phi}{\partial x}\bigg|_W,
\]

\[
\frac{1}{\Delta V} \int_V \frac{\partial \psi}{\partial y} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial \phi}{\partial y}\bigg|_W,
\]

\[
\frac{1}{\Delta V} \int_V \frac{\partial \psi}{\partial z} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial \phi}{\partial z}\bigg|_W,
\]

\[
\frac{1}{\Delta V} \int_V \frac{\partial^2 \psi}{\partial x^2} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial^2 \phi}{\partial x^2}\bigg|_W,
\]

\[
\frac{1}{\Delta V} \int_V \frac{\partial^2 \psi}{\partial y^2} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial^2 \phi}{\partial y^2}\bigg|_W,
\]

\[
\frac{1}{\Delta V} \int_V \frac{\partial^2 \psi}{\partial z^2} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial^2 \phi}{\partial z^2}\bigg|_W,
\]

\[
\frac{1}{\Delta V} \int_V \frac{\partial^3 \psi}{\partial x \partial y} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial^2 \phi}{\partial x \partial y}\bigg|_W,
\]

\[
\frac{1}{\Delta V} \int_V \frac{\partial^3 \psi}{\partial x \partial z} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial^2 \phi}{\partial x \partial z}\bigg|_W,
\]

\[
\frac{1}{\Delta V} \int_V \frac{\partial^3 \psi}{\partial y \partial z} \, dV = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} c_{ijk} \left( \frac{\Delta x}{2} \right)^i \left( \frac{\Delta y}{2} \right)^j \left( \frac{\Delta z}{2} \right)^k = \frac{\partial^2 \phi}{\partial y \partial z}\bigg|_W,
\]

\(\text{(14)}\)

where \(\frac{\partial \phi}{\partial x}\bigg|_W, \frac{\partial \phi}{\partial y}\bigg|_W, \frac{\partial \phi}{\partial z}\bigg|_W, \frac{\partial^2 \phi}{\partial x^2}\bigg|_W, \frac{\partial^2 \phi}{\partial y^2}\bigg|_W, \frac{\partial^2 \phi}{\partial z^2}\bigg|_W, \frac{\partial^2 \phi}{\partial x \partial y}\bigg|_W, \frac{\partial^2 \phi}{\partial x \partial z}\bigg|_W, \frac{\partial^2 \phi}{\partial y \partial z}\bigg|_W\) are the first-order, second-order and cross-derivatives of the average value of the scalar, \(\phi\).

The ten Eqs. (13) and (14) yield a \(10 \times 10\) non-singular system of equations. This system can be cast in the matrix–vector form \(A \cdot \vec{c} = \vec{b}\), where \(A\) is the Vandermonde matrix, \(\vec{c}\) the coefficients vector, and \(\vec{b}\) the vector of conditions. The inverse coefficients matrix \(A^{-1}\) can be pre-computed. It has the form

\[
\begin{pmatrix}
1 & -\Delta x/4 & -\Delta y/4 & -\Delta z/4 & \Delta x^2/4 & \Delta y^2/4 & \Delta z^2/4 & \Delta x \Delta y/4 & \Delta y \Delta z/4 & \Delta z \Delta x/4 \\
0 & 1 & 0 & 0 & -\Delta x/2 & 0 & 0 & -\Delta x \Delta y/2 & -\Delta x \Delta z/2 & -\Delta x \Delta z/2 \\
0 & 0 & 1 & 0 & 0 & -\Delta y/2 & 0 & -\Delta y \Delta z/2 & -\Delta y \Delta z/2 & -\Delta y \Delta z/2 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -\Delta z/2 & -\Delta z \Delta y/2 & -\Delta z \Delta y/2 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & -\Delta x/2 & -\Delta x \Delta z/2 & -\Delta x \Delta z/2 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\(\text{(15)}\)
The ten coefficients defining the three-dimensional interpolant can then be simply obtained by multiplying the inverse matrix by the vector

\[
\mathbf{\phi}_w \begin{bmatrix} \frac{\partial \phi}{\partial x} |_w, \frac{\partial \phi}{\partial y} |_w, \frac{\partial \phi}{\partial z} |_w, \frac{\partial^2 \phi}{\partial x^2} |_w, \frac{\partial^2 \phi}{\partial x \partial y} |_w, \frac{\partial^2 \phi}{\partial x \partial z} |_w, \frac{\partial^2 \phi}{\partial y^2} |_w, \frac{\partial^2 \phi}{\partial y \partial z} |_w, \frac{\partial^2 \phi}{\partial z^2} |_w \end{bmatrix}^T.
\]

(16)

The components of the above vector, i.e. the derivatives of the average value, \( \phi_w \), at the center of the west cell are approximated using fourth-order centered finite difference formulas. The derivatives of \( \phi_w \) for the west cell are expressed in the following form:

\[
\frac{\partial \phi}{\partial x} |_w = \frac{1}{12 \Delta x} (\phi_{WW} - 8 \phi_{W} + 8 \phi_{E} - \phi_{W}), + O(\Delta x^4),
\]

\[
\frac{\partial \phi}{\partial y} |_w = \frac{1}{12 \Delta y} (\phi_{SS} - 8 \phi_{S} + 8 \phi_{NW} - \phi_{NNW}) + O(\Delta y^4),
\]

\[
\frac{\partial \phi}{\partial z} |_w = \frac{1}{12 \Delta z} (\phi_{BB} - 8 \phi_{B} + 8 \phi_{PW} - \phi_{NPW}) + O(\Delta z^4),
\]

\[
\frac{\partial^2 \phi}{\partial x^2} |_w = \frac{1}{12 \Delta x^2} (\phi_{WWW} - 16 \phi_{WW} + 30 \phi_{W} + 16 \phi_{E} - \phi_{E}), + O(\Delta x^4),
\]

\[
\frac{\partial^2 \phi}{\partial y^2} |_w = \frac{1}{12 \Delta y^2} (\phi_{SSS} + 16 \phi_{SW} - 30 \phi_{W} + 16 \phi_{NW} - \phi_{NNW}) + O(\Delta y^4),
\]

\[
\frac{\partial^2 \phi}{\partial z^2} |_w = \frac{1}{12 \Delta z^2} (\phi_{BBB} + 16 \phi_{BW} - 30 \phi_{PW} + 16 \phi_{PW} - \phi_{PPW}) + O(\Delta z^4),
\]

\[
\frac{\partial^2 \phi}{\partial x \partial y} |_w = \frac{1}{4 \Delta x \Delta y} (\phi_{NNS} + \phi_{SSW} - \phi_{NW} - \phi_{NNW}) + O(\Delta x^2 \Delta y^2),
\]

\[
\frac{\partial^2 \phi}{\partial x \partial z} |_w = \frac{1}{4 \Delta x \Delta z} (\phi_{NNN} + \phi_{SSS} - \phi_{NW} - \phi_{NNW}) + O(\Delta x^2 \Delta z^2),
\]

\[
\frac{\partial^2 \phi}{\partial y \partial z} |_w = \frac{1}{4 \Delta y \Delta z} (\phi_{NNN} + \phi_{SSS} - \phi_{NW} - \phi_{NNW}) + O(\Delta y^2 \Delta z^2).
\]

(17)

The above equations require the values of the scalar at twenty-five distinct nodes, see Fig. 3. Of course, the use of fourth-order formulas does not improve the convergence rate of our quadratic interpolation but, as shown by Russell [28], considerably improves its accuracy.

The stencils shown in Fig. 3 represent the collection of the nodes used to compute the polynomial interpolants \( \psi_w \) and \( \psi_e \). On the one hand, a stencil of twenty-five nodes could be considered as unnecessarily large, making their software implementation complex and computationally intensive. On the other hand, such a stencil leads to high-order interpolations. The recent emergence of affordable multi-core processors and the constantly increasing size of memory chips make the use of large stencils realistic, as our multi-threaded implementation of 3DFlux shows.

Finally, the values of the scalar and its normal gradients, at the side of a cell are estimated by averaging the polynomial interpolant of the upstream cell over the side area of the cell. For example, the value at the west side is evaluated by averaging the polynomial interpolant, \( \psi_w \), over the area of the west side, when \( u_w \) is positive, or by averaging the polynomial interpolant \( \psi_e \), when \( u_w \) is negative, i.e.

\[
\psi_w = \begin{cases} \int_{Ax} \int_{Ay} \psi_w(\Delta x, y, z) \, dy \, dz, & u_w > 0, \\ \int_{Ax} \int_{Ay} \psi_e(\Delta x, y, z) \, dy \, dz, & u_w < 0. \end{cases}
\]

(18)

Note that \( \psi_w \) and \( \psi_e \) are determined using the stencil of nodes shown on Fig. 3(a) and (b), respectively.
Similarly, the gradient normal to the west side is computed as follows

\[ \frac{\partial \phi}{\partial x}\bigg|_w = \frac{-1}{\lambda w} \int_0^\lambda \int_0^{\alpha w(x,y,z)} dy \; dz, \quad u_w > 0, \]

\[ \frac{\partial \phi}{\partial x}\bigg|_w = \frac{-1}{\lambda w} \int_0^\lambda \int_0^{\alpha w(x,y,z)} dy \; dz, \quad u_w < 0. \]

The polynomial interpolants \( \psi_p, \psi_f, \psi_l \) and \( \psi_0 \) are computed following a similar procedure. Note that these polynomials can also be obtained by an appropriate permutation of the inverse coefficient matrix (15) and the components of the vector (17).

2.3. Computation of the time-averaged fluxes

The flux integral method combines both Eulerian and Lagrangian point of views – the solution of the scalar field is discretized in space on an Eulerian grid, whereas it is discretized in time in a Lagrangian frame of reference. This interesting semi-Lagrangian approach was first introduced by Courant et al. [29] and, since then, widely used for the simulation of the transport equations [3,30–32]. Semi-Lagrangian schemes allow for larger time steps and have better stability properties than the classical Eulerian explicit and semi-implicit schemes.

We derive the time-averaged advective and diffusive fluxes using a Lagrangian description of the flow. For simplicity, we limit our derivation to the fluxes across the west face of the cell \( P \) when the three components of the velocity field are oriented in their positive directions, i.e. \( u_w > 0, \; v_w > 0 \) and \( w_w > 0 \). Furthermore, we also limit our implementation to Courant numbers \( \{c_x = u_w \Delta t / \Delta x, \; c_y = v_w \Delta t / \Delta y \; \text{and} \; c_z = w_w \Delta t / \Delta z \} \) less than or equal to unity. The time average of the advective and the diffusive fluxes over a time interval \( \Delta t \) and across the west face are expressed as follows

\[ \langle \rho u_w \phi \rangle = \frac{1}{\Delta t} \int_0^\Delta t \rho u_w \phi_w \, dt \]

and

\[ \langle \Gamma \frac{\partial \phi}{\partial x} \rangle = \frac{1}{\Delta t} \int_0^\Delta t \Gamma \frac{\partial \phi}{\partial x} \, dt, \]

where \( \phi_w \) and \( \frac{\partial \phi}{\partial x}\bigg|_w \) are respectively the average value of the scalar and the average of its gradient at the west side of the cell \( P \).

The scalar flux quantifies the rate of advection–diffusion of a scalar transported across the west face, that is, the amount of the scalar that passes through the west face per unit time. Consequently, the average of the advective and diffusive fluxes over a time interval \( \Delta t \) is, respectively, the average of the scalar properties \( \rho u_w \phi \) and \( \Gamma \frac{\partial \phi}{\partial x} \) over the volume of fluid that crosses the west face in the time interval \( \Delta t \).

As the velocity is assumed constant in the vicinity of the west face of the cell \( P \), the volume of fluid that traverses the west face in a time interval \( \Delta t \) is approximated by a rectangular prism \( \Delta V_p = u_w \Delta t \times \Delta y \times \Delta z \). This rectangular prism is a
backwards-in-time projection over an interval $\Delta t$ of the west face of the cell $P$, where the magnitude and the orientation of the projection is given by the vector $-u_w \Delta t$, i.e. $(-u_w \Delta t, -v_w \Delta t, -w_w \Delta t)$, as shown in Fig. 4. Note that, as the transverse velocity components $v_w$ and $w_w$ are non-zero, the prism overlaps with several cells, i.e. cells $W, B, SW$ and $BSW$. The volume fraction of fluid particles transported from each distinct cell into the cell $P$ are the four sub-volumes $V_1$ to $V_4$ illustrated in Fig. 4. The coordinates of the vertices A, B, C and D are determined by integrating backward in time over an interval $\Delta t$ the equation of the motion of the fluid particles located at the four corners of the west face of the cell $P$. The equation of motion of a fluid particle located at $\vec{x}$ is

$$\frac{d\vec{x}}{dt} = \vec{u}(\vec{x}, t).$$  \hspace{1cm} (22)

Integrating it backward in time over a time interval $\Delta t$ with the final conditions, $x(\Delta t) = \Delta x$, $y(\Delta t) = \Delta y$ and $z(\Delta t) = \Delta z$, we have

$$\begin{align*}
x(0) &= \Delta x - u_w \Delta t, \\
y(0) &= \Delta y - v_w \Delta t = \Delta y + v_w \frac{x(0) - \Delta x}{u_w}, \\
z(0) &= \Delta z - w_w \Delta t = \Delta z + w_w \frac{x(0) - \Delta x}{u_w}.
\end{align*}$$  \hspace{1cm} (23)

The integral of the scalar quantity, $\rho u_w \phi(0)$, over the rectangular prism, $\Delta V_p$, can be rewritten as

$$\int_{\Delta V_p} \rho u_w \phi(0) dV_p \approx u_w \int_{x(0)}^{x(0) + \Delta x} \left( \int_{y(0)}^{y(0) + \Delta y} \rho \psi_W dy dz + \int_{y(0)}^{y(0) + \Delta y} \rho \psi_S dy dz + \int_{y(0)}^{y(0) + \Delta y} \rho \psi_{SW} dy dz + \int_{y(0)}^{y(0) + \Delta y} \rho \psi_{BSW} dy dz \right) dx,$$

where the four terms on the right-hand side represent the integration of the interpolants $\psi_W, \psi_S, \psi_{SW}$ and $\psi_{BSW}$ over the volumes $V_1, V_2, V_3$ and $V_4$, respectively, and computed at $t = 0$. Note that the average value of $\rho u_w \phi$ over the prism $\Delta V_p$ represents the average flux across the west face during $\Delta t$, i.e.

$$\langle \rho u_w \phi \rangle = \frac{1}{u_w \Delta t \Delta Y \Delta z} \int_{\Delta V_p} \rho u_w \phi(x, y, z, t) dV_p.$$  \hspace{1cm} (25)

To perform the above integration, we recall that the Lagrangian form of the transport equation (1) is

$$\frac{D(\rho \phi)}{Dt} = \vec{\nabla} \cdot (\Gamma \vec{\nabla} \phi) + S,$$

where $D/Dt$ is the material derivative. Hence, the time evolution of the scalar $\rho \phi$ transported by a fluid particle is obtained by integrating Eq. (26) with respect to time $t$ from 0 to $t$ as follows

$$\rho \phi(x, y, z, t) = \rho \phi(x, y, z, 0) + \int_0^t \left( \vec{\nabla} \cdot (\Gamma \vec{\nabla} \phi) + S \right) dt.$$  \hspace{1cm} (27)
Eq. (27) states that the amount of $\rho \phi$ transported by a particle is not constant when it is subject to diffusion and source terms. Note that the value of $\phi(x, y, z, 0)$ is known, therefore it is approximated by the polynomial interpolant $\psi$ computed at $t = 0$. The integral (27) can be easily computed because $\psi$ is a quadratic polynomial and the source terms are constant within a cell during a time interval $\Delta t$. For instance, the integral over the volume $V_1$ can be approximated as

$$
\int_0^t \left( \nabla \cdot (\Gamma \nabla \psi) + S \right) dt \approx \left( \nabla \cdot (\Gamma \nabla \psi_W) + S \right) t.
$$

(28)

Using Eqs. (24), (27) and (28), we derive the time average of the advective fluxes as follows

$$
\left\{ \frac{1}{\Delta t \Delta y \Delta z} \times \right.
\left\{ \int_{x(0)}^{x(y(0))} \int_{y(0)}^{y(0)} \int_{z(0)}^{z(0)} \left[ \int_{z(0)}^{z(0)} (\rho \phi_W) dx + u_w \frac{\Delta z^2}{2} \left( \nabla \cdot (\Gamma \nabla \psi_W) + S_W \right) \right] dy dz 
+ \int_{x(0)}^{x(y(0))} \int_{y(0)}^{y(y(0))} \int_{z(0)}^{z(0)} \left[ \int_{z(0)}^{z(0)} (\rho \phi_B) dx + u_w \frac{\Delta z^2}{2} \left( \nabla \cdot (\Gamma \nabla \psi_B) + S_B \right) \right] dy dz 
+ \int_{x(0)}^{x(0)} \int_{y(y(0))}^{y(y(0))} \int_{z(0)}^{z(0)} \left[ \int_{z(0)}^{z(0)} (\rho \phi_{SNW}) dx + u_w \frac{\Delta z^2}{2} \left( \nabla \cdot (\Gamma \nabla \psi_{SNW}) + S_{SNW} \right) \right] dy dz 
+ \int_{x(y(0))}^{x(y(0))} \int_{y(0)}^{y(0)} \int_{z(0)}^{z(0)} \left[ \int_{z(0)}^{z(0)} (\rho \phi_{SW}) dx + u_w \frac{\Delta z^2}{2} \left( \nabla \cdot (\Gamma \nabla \psi_{SW}) + S_{SW} \right) \right] dy dz \right\},
$$

(30)

where the subscripts identify the cells in which $\psi$ and $S$ are computed. Note that using Eqs. (22) and (23), we transformed the integrals in $x$ into integrals in $t$ as follows

$$
\int_{x(0)}^{x(y(0))} \left( \nabla \cdot (\Gamma \nabla \psi_W) + S \right) dx = \int_{0}^{\Delta t} u_w \left( \nabla \cdot (\Gamma \nabla \psi_W) + S_W \right) dt = u_w \frac{\Delta t^2}{2} \left( \nabla \cdot (\Gamma \nabla \psi_W) + S_W \right).
$$

(31)

To determine the average of the diffusive flux at the west face, we first take the derivative of Eq. (26) in the direction normal to the west face, i.e. in the $x$-direction, and we multiply it by $\Gamma$ to obtain

$$
\frac{\partial}{\partial x} \left[ \Gamma \left( \frac{D}{D} \phi \right) \right] = D \frac{\partial}{\partial x} \left( \Gamma \frac{\partial \phi}{\partial x} \right) = \frac{\partial}{\partial x} \left[ \Gamma \left( \nabla \cdot (\Gamma \nabla \psi) + S \right) \right].
$$

(32)

The above equation is then integrated with respect to time from 0 to $t$ to express the time evolution of the quantity $\Gamma \frac{\partial \phi(x, y, z, t)}{\partial x}$

$$
\Gamma \left( \frac{\partial \phi(x, y, z, t)}{\partial x} - \frac{\partial \phi(x, y, z, 0)}{\partial x} \right) = \Gamma \int_{0}^{t} \frac{\partial}{\partial x} \left( \nabla \cdot (\Gamma \nabla \psi) + S \right) dt.
$$

(33)

The polynomial interpolant, $\psi$, is used to approximate $\frac{\partial \phi(x, y, z, 0)}{\partial x}$ which is unknown. Hence, noticing that any quadratic polynomial $\psi$ we have

$$
\frac{\partial}{\partial x} \left( \nabla \cdot (\Gamma \nabla \psi) \right) = 0.
$$

(34)

and assuming that the source term is constant within a cell during an interval $\Delta t$, i.e. $\partial S/\partial x = 0$, we derive the time average of the diffusive flux as follows

$$
\left\{ \int_{x(0)}^{x(y(0))} \left[ \int_{y(0)}^{y(y(0))} \left[ \int_{z(0)}^{z(0)} \left( \frac{\partial \phi}{\partial x} \right) \right] dy \right] dz \right\} dx.
$$

(35)

Eqs. (30) and (35), respectively express the time averages of the advective and diffusive fluxes across the west face of the cell $P$. The average fluxes for the other faces and the other velocity orientations are determined using similar derivations.

2.4. Stability

3DFLUX is unconditionally stable when solving pure advection problems because it is strictly a flux integral method (see [33] for details). However, there exists a geometric constraint on the size of the rectangular prism used to perform the backward integration, as it cannot exceed the dimension of one cell (see Fig. 4). This constraint limits the range of the Courant numbers to
\[ |c_x| \leq 1, \quad |c_y| \leq 1, \quad |c_z| \leq 1. \]  

In theory, one could relax the above constraint by allowing larger rectangular prisms when performing the backward integration. However, such a modification would render the implementation of our algorithm extremely complex.

3DFLUX is only conditionally stable when solving purely diffusive or advective–diffusive problems. To derive the stability constraint, we perform a von Neumann type of stability analysis for the case of constant coefficients, no source, periodic boundary conditions and homogeneous grid, i.e., \( \Delta x = \Delta y = \Delta z = \Delta h \). Writing the solution at node \( P \) and time \( t \) in a wave form

\[ \phi_P^t = e^{\sigma t} e^{i(k_x x + k_y y + k_z z)}, \]  

where \( \sigma \) is a constant, \( (k_x, k_y, k_z) \) are the wavenumbers in the \( x-, y- \) and \( z \)-directions, respectively, \( i = \sqrt{-1} \), and substituting Eq. (37) into the discretized Eq. (9), we obtain

\[ e^{\rho \Delta t} = 1 + \frac{1}{e^{\rho \Delta t} e^{i(k_x x + k_y y + k_z z)}} \left[ (f_w - f_e) + (f_i - f_n) + (f_b - f_f) \right], \]  

where \( e^{\rho \Delta t} = \frac{\phi_P^t}{\phi_P^0} = G \) is called the complex amplitude ratio. The necessary and sufficient condition for 3DFLUX to be stable is that

\[ |G|_{\text{max}} \leq 1, \]  

where \( |G|_{\text{max}} \) is the maximum magnitude of \( G \).

For a pure diffusion problem, the fluxes in Eq. (38) are determined from Eq. (35). For instance, using Eq. (11), the term \( (f_w - f_e) \) can be written as

\[ f_w - f_e = -\frac{\Gamma \Delta t}{\rho \Delta h^2} \left( -\frac{1}{8} \psi_{xx} + \frac{35}{24} \psi_e - \frac{31}{12} \psi_e + \frac{5}{4} \psi_w + \frac{1}{24} \psi_{ww} - \frac{1}{24} \psi_{www} \right). \]  

Substituting Eq. (37) into Eq. (40), we obtain

\[ \frac{(f_w - f_e)}{e^{\rho \Delta t} e^{i(k_x x + k_y y + k_z z)}} = -\frac{\Gamma \Delta t}{\rho \Delta h^2} \left( -\frac{1}{8} e^{i \theta} - \frac{35}{24} e^{i \theta} - \frac{31}{12} e^{i \theta} + \frac{5}{4} e^{i \theta} + \frac{1}{24} e^{2i \theta} - \frac{1}{24} e^{2i \theta} \right), \]  

where \( \theta = k_1 \Delta h \) is the wavenumber weighted by the grid spacing and \( -\pi \leq \theta \leq \pi \). The other terms on the right hand side of Eq. (38) are determined similarly. Finally, it can be shown that

\[ |G|_{\text{max}} = \left| 1 + \frac{\Gamma \Delta t}{\rho \Delta h^2} \right| \]  

and, consequently, the stability condition is

\[ 0 \leq \frac{\Gamma \Delta t}{\rho \Delta h^2} \leq \frac{1}{8} \]  

where \( \Gamma \Delta t / \rho \Delta h^2 \) is a dimensionless diffusive coefficient.

In addition, this analysis shows that 3DFLUX is, as UTOPIA, only second-order accurate when the problem is purely diffusive. In fact, Eq. (40) can be rewritten as a second-order finite difference approximation of the second order derivative of \( \phi \) with respect to \( h \), i.e.,

\[ f_w - f_e = -\frac{\Gamma \Delta t}{\rho \Delta h^2} \left( \partial_x^2 \phi + \frac{1}{12} \Delta h^2 \partial_x^4 \phi + O(\Delta h^3) \right). \]  

Leonard et al. [4] showed, using a Taylor series analysis, that the coupling between the advective and diffusive terms present in the flux integral method (see Eq. (30)) leads to a third-order advection–diffusion solver. However, when the coupling terms do not exist, i.e., when advection is absent, the diffusion solver is equivalent to a second-order finite difference scheme.

2.5. Monotonicity preservation

A scheme is said to be monotonicity-preserving when it does not generate new local extrema in the solution and the value of a local minimum/maximum is non-decreasing/non-increasing in time [34]. Godunov [35] showed that any linear monotonicity-preserving scheme is, at most, first-order accurate in space. Hence, advection schemes of orders higher than one are not monotonicity-preserving, and generate spurious oscillations near discontinuities or near poorly resolved gradients of the solution. Zalesak [11] was the first to propose a monotonicity-preserving high-order method called the flux-corrected transport (FCT) method.

The FCT method corrects the high-order solution by using a low-order scheme in the vicinity of the poorly resolved gradients and can be decomposed in four steps. In the first step, a solution is computed with a first-order scheme. It is called "transported and diffused" and is identified by the superscript "td". This solution is calculated as follows
\[
(\phi^{t+\Delta t})^{\text{td}} = (\phi^{t+\Delta t})^{\text{corr}} + \Gamma_w^{\text{low}} - \Gamma_e^{\text{low}} + \Gamma_n^{\text{low}} - \Gamma_s^{\text{low}} + \Gamma_b^{\text{low}} - \Gamma_f^{\text{low}} + (S) \frac{\Delta t}{\rho \Delta V},
\]

(45)

where the superscript "low" indicates the fluxes computed with the low-order scheme and \((\phi^{t+\Delta t})^{\text{corr}}\) is the corrected solution computed at previous time steps. In our 3DFLUX solver, the low-order scheme has been implemented by simply replacing the quadratic interpolant polynomial, \(\psi\), in Eq. (12) with a zero-order polynomial. Doing so, we obtain a first-order scheme that is equivalent to the well-known upwind scheme proposed by Patankar [1].

The second and the third steps consist of, respectively, computing the fluxes with the high-order scheme and estimating the values of the corrected fluxes. The low-order scheme is monotonicity preserving, has low accuracy and is highly diffusive. The high-order scheme is not monotonicity preserving, but more accurate and produces low numerical diffusion. Hence, the value of the corrected fluxes has to be an optimal compromise between the values of the low- and high-order fluxes so that the corrected solution benefits from the advantages of both schemes. The corrected fluxes are expressed, for instance at the west face, as

\[
\Gamma_w^{\text{corr}} = \Gamma_w^{\text{low}} + \beta_w A_w,
\]

(46)

where \(A_w = \Gamma_w - \Gamma_w^{\text{low}}\) is called the anti-diffusive flux at the west face and the parameter \(\beta_w \in [0, 1]\) has to be evaluated at the west face. Note that the corrected flux is equal to the low-order flux when \(\beta = 0\) and equal to the high-order flux when \(\beta = 1\). The fourth and final step consists of computing the corrected solution using Eqs. (45) and (46) as follows

\[
(\phi^{t+\Delta t})^{\text{corr}} = (\phi^{t+\Delta t})^{\text{td}} - \beta_w A_w - \beta_e A_e + \beta_n A_n - \beta_s A_s - \beta_b A_b - \beta_f A_f + (S) \frac{\Delta t}{\rho \Delta V}.
\]

(47)

The main difficulty in Zalesak's method stands in correctly estimating the limiting parameters: \(\beta_w, \beta_e, \beta_n, \beta_s, \beta_b, \) and \(\beta_f\). For further details, the reader is referred to [12].

Ideally, a flux-limiting method preserves monotonicity without corrupting the accuracy of the high-order scheme. However, Blossey and Durrant [15] showed that although Zalesak's method correctly damps the over- and under-shoots produced in the vicinity of the poorly resolved gradient, it also incorrectly damps the extrema in the smooth and well-resolved parts of the solution. This results in an undesirable reduction of the overall accuracy of the scheme. For example, Blossey and Durrant [15] showed that their third-order scheme, based on a piecewise parabolic method (PPM) [16], becomes only second–order accurate when it is combined with the FCT method.

Blossey and Durrant [15] therefore modified Zalesak's method to activate the flux–limiter only in the regions of the computational domain affected by unphysical over- and under-shooting. They showed that the PPM solver combined with their selective monotonicity preserving method, remains third-order accurate while quasi-preserving the monotonicity of the solution (i.e. the over-shoots and under-shoots are negligible with respect to the magnitude of the solution). This selective monotonicity preservation introduces a smoothness parameter, \(\lambda\), to detect the presence of the poorly resolved gradients in the solution. The value of \(\lambda\) is computed at each cell and compared to a predefined threshold value, \(\lambda^{\text{MAX}}\). When \(\lambda\) remains below the threshold, the parameter \(\beta\) is set to one, which deactivates the monotonic limiter. Otherwise, the FCT method is used to calculate the value of \(\beta\), i.e.

\[
\beta = \begin{cases} 
\beta & \text{if } \lambda > \lambda^{\text{MAX}}, \\
1 & \text{otherwise}.
\end{cases}
\]

(48)

Blossey and Durrant [15] computed \(\lambda\) at the nodes using the first– and second–order derivatives of the scalar field in the direction normal to the face. Their approach was consistent with the PPM scheme that is based on a dimensional splitting method. Note that the parameter \(\lambda\) is analogous to the one-dimensional smoothness metric used in the WENO scheme [36]. More recently, Harris et al. [6] defined \(\lambda\) using the derivatives of the scalar field in two directions, to be consistent with their two-dimensional solver.

Herein, we propose a three-dimensional formulation of the parameter, \(\lambda\), as follows

\[
\lambda_{ijk} = \frac{\max_{l=1,m=1,k=1} \left\{ \gamma_{l,m,k} \right\}}{\min_{l=1,m=1,k=1} \left\{ \gamma_{l,m,k} \right\} + \varepsilon},
\]

(49)

where

\[
\gamma_{l,m,k} = \frac{1}{2} \left[ \left( 2\Delta x \frac{\partial \phi_{l,m,k}}{\partial x} \right)^2 + \left( 2\Delta y \frac{\partial \phi_{l,m,k}}{\partial y} \right)^2 + \left( 2\Delta z \frac{\partial \phi_{l,m,k}}{\partial z} \right)^2 \right],
\]

(50)

and

\[
\begin{align*}
I &= [i-2,i+1], \\
J &= [j-2,j+1], \\
K &= [k-2,k+1],
\end{align*}
\]

(51)
where \(i, j, k\) and \(\ell, m, n\) are the cell indices in the \(x-, y-, \text{ and } z\)-direction, respectively. The derivatives in (50) are directly computed using Eq. (17) and \(\varepsilon\) is a small parameter required to prevent division by zero. We obtained consistent results using \(\varepsilon = 10^{-4}\) and found that \(\lambda_{\text{Max}} = 60\) is an optimal value to identify the regions of the domain in which the monotonic limiter must be applied. This value is also consistent with the value reported by Harris et al. [6].

2.6. Implementation of the boundary conditions

Our third-order scheme uses a large stencil (see Fig. 3) to compute the fluxes and update the average cell values at each time step. Therefore, near the boundaries of the computational domain, some of the values at neighboring cells required to construct this stencil are not available. To overcome this difficulty, we extend the computational domain by introducing fictitious nodes, also known as “ghost” or external nodes, on the outside of the domain, as shown in Fig. 5. By doing so, the solution is computed at all interior cells using the same stencil and scheme.

The value of the scalar field at the ghost-nodes is extrapolated, at the beginning of each time step, from the boundary conditions and from the value of the scalar field at the internal nodes in the previous time step. Note that the computational scheme at the ghost-nodes can be entirely decoupled from the numerical scheme used at the interior nodes [37]. To compute the fluxes at the boundary with 3Dflux, three ghost-nodes are required. However, when the fluxes are prescribed at the boundary, only the values at two ghost-nodes are required to compute the flux at the face of the first internal cell.

In practical applications, the implementation of the ghost-node technique is problem-dependent, see for example [38–40]. Consequently, we restrict our discussion in this study to the treatment of the boundary conditions implemented in the test cases used to validate our solver, i.e. periodic, inflow, outflow and Neumann boundary conditions.

Periodic boundary conditions are used in all tests presented in Section 3. In this case, the values at the three ghost-nodes are set equal to the values at the corresponding three internal nodes at the opposite side of the domain. In Fig. 5, the value of the nodes \(\phi_W, \phi_{WW}\) and \(\phi_{WWW}\) is respectively equal to the value at the nodes \(\phi_{n-2}, \phi_{n-3}\) and \(\phi_{n-4}\) (not shown).

The inflow and outflow conditions used in the last test of Section 3 are those used to simulate the inlet and outlet conditions of a flow in a channel. The treatment of the inflow boundary conditions is trivial as the values of the scalar field (and by extension the values at the ghost-nodes) are prescribed at the entrance of the channel. However, the treatment of the outflow boundary conditions is more challenging. To illustrate our discussion, we impose an outflow boundary at the node \(n - 1\) in Fig. 5. Among the different techniques that simulate an outflow boundary [41–45], we have chosen the method presented by Ferziger and Pirić [45]. This method extrapolates the value of the scalar field at the ghost-nodes by solving an unsteady convective condition of the form

\[
\frac{\partial \phi}{\partial t} + U \frac{\partial \phi}{\partial x} = 0,
\]

(52)

where \(U\) is the average velocity of the outflow and \(\frac{\partial \phi}{\partial x}\) is the derivative of the scalar at node \(n\) in the outward direction normal to the boundary. Note that \(U\) is constant at the outlet and is chosen so that the outflow mass flux is equal to the incoming mass flux. The partial derivative, \(\frac{\partial \phi}{\partial x}\), is approximated using a third-order backward finite difference scheme [46, p. 319]

\[
\frac{\partial \phi}{\partial x} \Big|_{n} \approx \frac{11}{6} \phi_{n} - 3 \phi_{n-2} + \frac{3}{2} \phi_{n-3} - \frac{1}{3} \phi_{n-4} + O(\Delta x^3)
\]

(53)

and the time integration is performed using the explicit third-order Adams–Bashforth method [47]

\[
\phi^{t+\Delta t} = \phi^t - \frac{1}{12} U \Delta t \left( 23 \frac{\partial \phi}{\partial x} \Big|_{n} + 16 \frac{\partial \phi}{\partial x} \Big|_{n-1} + 5 \frac{\partial \phi}{\partial x} \Big|_{n-2} \right).
\]

(54)

Note that the value at the node \(\phi_{n-1}\) is not used in Eq. (53).

The Neumann boundary condition is used in the first and last tests of Section 3. To implement this condition, we use a local third-order Taylor expansion to extrapolate the value at the ghost-nodes. In the case shown in Fig. 5, the values of \(\phi_W\) and \(\phi_{WW}\) are given by

\[
\phi_W = \phi_p - \Delta x \frac{\partial \phi}{\partial x} \Big|_W + O(\Delta x^3)
\]

(55)

and

\[
\phi_{WW} = \phi_E - 3 \Delta x \frac{\partial \phi}{\partial x} \Big|_W + O(\Delta x^3),
\]

(56)

Fig. 5. Treatment of the boundary using ghost-nodes.
respectively. Note that the no-flux boundary conditions is particularly simple to implement because the values at the ghost nodes are equal to the values at the mirror image internal points. In the case shown in Fig. 5, \( \phi_W = \phi_F \) and \( \phi_{WW} = \phi_K \).

3. Convergence rate and accuracy

The numerical errors that could affect the numerical solutions produced by 3DFLUX are attributable to (i) the spatial interpolation presented in Section 2.2 and (ii) the backwards integration presented in Section 2.3. The overall order of convergence of 3DFLUX in general is

\[
O\left( \Delta t^k, \Delta h^{p+1} \right),
\]

where \( k \) is the order of the time integration scheme, \( p \) is the order of the polynomial interpolant, and \( \Delta t \) and \( \Delta h \) are the temporal and the spatial resolutions of the scheme, respectively. Note that the maximum allowable time step, \( \Delta t \), is controlled by the constraints (36) on the Courant numbers.

To assess the convergence rate and the accuracy of our solver, we present a series of five numerical tests. By construction, 3DFLUX is third-order accurate under constant flow conditions because the polynomial interpolation is quadratic \( (p = 2) \) and there is no truncation error in the time integration \( (\text{i.e. Eq. (23)} \text{ is exact}) \). However, under non-constant flow conditions, the accuracy of the time integration algorithm depends on the accuracy of the approximation of the advective velocity field over the time interval, \( \Delta t \). In 3DFLUX, the components of the velocity field in Eq. (23) are evaluated at the intermediate time step, \( t + \Delta t / 2 \), by an explicit mid-point interpolation \( [5, 48] \) as follows

\[
\bar{u}^{t+\Delta t / 2} = \frac{1}{2} (\bar{u}^{t+\Delta t} + \bar{u}^t).
\]

Note that this temporal staggering of the velocity field makes 3DFLUX second-order accurate in time when solving unsteady problems. However, as we will show in the following tests, the lower accuracy of the time integration scheme has a limited impact in practical applications. In fact, the accuracy of 3DFLUX depends essentially on the form of the spatial integration \( [20] \).

3.1. Test 1: mixing/unmixing problem

We examine the accuracy and the convergence rate of 3DFLUX by solving the mixing/unmixing test proposed by Seibold et al. [19]. In this test, the computational domain is a two-dimensional square, i.e. \( (x, y) \in [0, 1] \times [0, 1] \). The initial condition for the scalar field is

\[
\phi(x, y, t = 0) = \cos(2\pi x) \cos(4\pi y)
\]

and the boundary conditions are doubly periodic. For \( t > 0 \), the initial distribution of the scalar field is purely advected and deformed by the velocity field

\[
\begin{align*}
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} &= 0, \\
\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} &= 0,
\end{align*}
\]

which is unsteady, divergence-free and periodic in time with period \( T \) [5]. After each half-period, the direction of the flow is reversed and the fluid particles return to their initial position.

We perform a series of simulations in which the uniform size, \( \Delta h \times \Delta h \), of the cells is repeatedly halved. We then perform an error analysis by computing the \( L^\infty \) and \( L^2 \) norms of the error, i.e., the difference between the solution after one period \( T \) and the initial scalar field. Recall that the \( L^\infty \) norm of the error is the maximum of the absolute value of the error. In the time integration, we use the largest possible time step allowed by the stability requirement. Note that the value of the maximum velocity over a period is 1.

Fig. 6 reports the convergence rate of the numerical errors for several schemes including the WENO3 third-order scheme, the bi-cubic jet scheme by Seibold et al. [19] and the third-order UTOPIA scheme. All schemes but WENO3 exhibit a third-order convergence rate using the \( L^2 \) and \( L^\infty \) norms. There are, however, significant differences in the accuracy of the schemes. The accuracy of 3DFLUX is comparable to the accuracy of the bi-cubic jet scheme, a recently proposed solver designed for the simulation of purely advective problems. The accuracy of UTOPIA is about an order of magnitude worse than that of 3DFLUX, while WENO3’s accuracy is more than two orders of magnitude worse.

As previously mentioned, Seibold et al. [19] used periodic boundary conditions in this test. However, because the velocity field is zero at any time on the boundaries of the unit square, and because there are no diffusion effects, the computational domain can be considered as a closed-domain. Hence, the solution of the test remains unchanged if we replace the periodic boundary condition with a no-flux boundary condition. We re-computed the solution up to one period \( T \) for different spatial resolutions with the no-flux boundary conditions and obtained the same results as in Fig. 6, therefore validating our implementation of the no-flux boundary conditions. 3DFLUX is nominally second-order in time and third-order in space. This is confirmed by Fig. 6, which shows that 3DFLUX is third-order when using the highest possible Courant number. One might
therefore conclude that the spatial truncation error dominates the overall error. However, a series of simulations in which the time step is repeatedly divided by a factor ten shows that a reduction in the time step leads to a considerable increase in the convergence rate of the $L^1$ and $L^\infty$ norms, see Fig. 7. This result implies that the truncation error due to the time integration is not negligible. In fact, the solution computed at the highest resolution is 20 times more accurate when $\Delta t$ is divided by 100. Fig. 7 shows that the order of the convergence rate can unexpectedly be as high as 4.6, much higher than 3 when the Courant number is equal or lower to 0.01. Similarly, Boslsey and Durran [15] reported an order of convergence higher than three when using a third-order piecewise cubic method.

3.2. Test 2: solid body rotation

This test, first proposed by Orszag [21], examines the robustness of a numerical scheme when the solution contains discontinuities in its derivatives. The test consists in simulating the purely advective transport of a slotted cylinder moving in a rotational motion around the origin, as shown in Fig. 8. The slotted cylinder conserves its original shape while transported, and returns to its initial position after each period, $T$.

The numerical domain is a unit square domain, $(x, y) \in [-0.5, -0.5] \times [-0.5, +0.5]$. The initial condition for the concentration of the scalar field is a slotted cylinder of diameter 0.25 centered at $(x, y) = (0.25, 0)$, i.e.
Fig. 8. Rotation of a slotted cylinder over a period $T$. The position of the slotted cylinder is shown at five different times, $t = 0, T/4, T/2, 3T/4$ and $T$.

$$\phi(x, y, t = 0) = \begin{cases} 1 & \text{if } r \leq 0.125 \land (y > 0.0) \lor (0.225 < x < 0.275), \\ 0 & \text{elsewhere}, \end{cases}$$

where

$$r = \left[ (x - 0.25)^2 + y^2 \right]^{1/2}.$$  

The velocity field,

$$
\begin{align*}
    u(x, y, t) &= -\Omega y, \\
    v(x, y, t) &= \Omega x,
\end{align*}
$$

transports the slotted cylinder over a circular path with constant angular velocity $\Omega$ (positive for counterclockwise rotation). Note that the domain is large enough to prevent the solid body from interacting with the boundaries. The grid is uniform, and the Courant numbers, $c_c$ and $c_s$, are both set to one for each simulation, i.e. $\Delta t = 2\Delta h$.

The gradient of the scalar field at the contour of the slotted cylinder is discontinuous. Without a flux-limiter algorithm, 3DFLUX would generate spurious oscillations that would rapidly deteriorate the shape of the slotted cylinder – see for example [48]. We show that implementing 3DFLUX with a flux-limiter algorithm prevents oscillations from occurring.

Fig. 9 shows, for three spatial resolutions, three isocontours of the scalar field, $\phi = 0.01, 0.5$ and 0.99, after one and five revolutions of the slotted cylinder. In addition, Fig. 10 shows, after one revolution, a one-dimensional cross-section of the slotted cylinder along the $x$-axis and at $y = 0.05$. The $\phi = 0.5$ isocontour reproduces very well the slotted cylinder profiles, even after several rotations. The band delimited by the $\phi = 0.01$ and $\phi = 0.99$ isocontours illustrates clearly the effects of the numerical diffusion produced by the flux-limiter in the vicinity of the discontinuity. The flux-limiter diffuses the discontinuity of the scalar field over three nodes on each side of it, apparently independently of the spatial resolution (Fig. 10). Hence, length scales smaller than six cells, centered on the discontinuity, are not fully resolved by 3DFLUX.

3.3. Test 3: two-dimensional unsteady deformational flow

The pure advection test proposed by Blochey and Durran [15] consists of simulating the stretching and rotation of a cosine bell advected by a deformational velocity field. The computational domain is the unit square $(x, y) \in [0, 1] \times [0, 1]$. The initial condition for the scalar field is defined as

$$\phi(x, y, t = 0) = \begin{cases} \frac{1 + \cos(\pi r)}{2} & \text{if } r \leq 0, \\ 0 & \text{elsewhere}, \end{cases}$$

where

$$r = 5[ (x - 0.3)^2 + (y - 0.5)^2 ]^{1/2}.$$  

The unsteady, divergence free velocity field that advects the scalar field is defined as

$$
\begin{align*}
    u(x, y, t) &= (y - \frac{1}{2})\Omega, \\
    v(x, y, t) &= -(x - \frac{1}{2})\Omega,
\end{align*}
$$

where the angular velocity, $\Omega = \Omega(x, y, t)$, is
Fig. 9. The slotted cylinder after one (a) and five (b) periods of rotation, respectively, for three different grid resolutions ($N_x \times N_y = 200 \times 200$, $400 \times 400$ and $800 \times 800$). The isocontours corresponding to $\phi = 0.01$, $0.5$ and $0.99$, are compared to the exact solution (bold dashed line).

Fig. 10. The 1D cross-section of the slotted cylinder at $y = -0.05$ for two mesh resolutions, $N_x \times N_y = 200 \times 200$ (■) and $N_x \times N_y = 800 \times 800$(○), are compared to the exact solution (solid line).

$$\Omega(x, y, t) = \frac{4\pi}{T} \times \left\{ 1 + \left[ 1 - \frac{2}{(256\zeta(x,y)^2 - 16\zeta(x,y) + 1)(16\zeta(x,y) + 1)} \right] \cos \left( \frac{2\pi t}{T} \right) \right\}, \quad (67)$$

where $\zeta(x, y)$ is

$$\zeta(x, y) = \left( x - \frac{1}{2} \right)^2 + \left( y - \frac{1}{2} \right)^2. \quad (68)$$
Fig. 11 presents the evolution of the cosine bell in the clockwise direction at seven different time instants, $t = 0, T/4, 2T/5, T/2, 3T/5, 3T/4$ and $T$. During the first half period (Fig. 11(a)), the velocity field deforms the cosine bell and generates a long tail that stretches away from the center of the bell. Sharp gradients appear on the lateral sides of the tail. At $t = T/2$, the bell recovers its initial shape and position. In the second half period (Fig. 11(b)), the orientation of the stretching is reversed while the orientation of the circular motion remains in the clockwise direction. Finally, at $t = T$, the cosine bell returns to its initial position. The exact solution to this problem after one period is its initial condition, i.e. the scalar field defined in Eq. (64).

The time evolution of the cosine bell has been computed using the largest time step possible corresponding to the largest Courant numbers admissible, i.e. $c_x = c_y = 1$. Note that in contrast with the mixing/unmixing test shown in Section 3.1, where the transition from mixing to unmixing was obtained by reversing the flow, the velocity field defined in the present test never reverses. Blossey and Durran [15] recommended avoiding tests with inversions of the velocity field because they can lead to phase error cancelation, masking computational inefficiencies.

To quantify the numerical error of 3DFlux, we compare the initial condition (the exact solution) with the numerical solution obtained after one period. Fig. 12 plots the contours of the exact and numerical solutions after one period for three different spatial resolutions, $N_x \times N_y = 107 \times 107$, $207 \times 207$, and $407 \times 407$. The coarser resolution does not accurately resolve the small scales induced by the intensive stretching and, consequently, the solution is deformed in the $y$-direction and the center of the cosine bell is not correctly recovered. However, the monotonicity of the solution is preserved. Moreover, 3DFlux does not generate spurious oscillations despite the presence of sharp gradients. The results obtained with finer resolutions ($207 \times 207$ and $407 \times 407$) preserve well the symmetry of the cosine bell and its peak locations.

To characterize the convergence rate and accuracy of 3DFlux, we calculate the $L^2$ and $L^\infty$ norms of the error while halving the grid resolution. In Fig. 13, we compare the performance of 3DFlux to those of UTOPIA and a second-order in time and

![Fig. 11. Advection of a cosine bell at different time intervals: $t = 0$ (A), $t = T/4$ (B), $t = 2T/5$ (C), $t = T/2$ (D), $t = 3T/5$ (E), $t = 3T/4$ (F) and $t = T$ (G). The isocontours are plotted from 0.05 to 0.95 in increments of 0.1.](image)

![Fig. 12. Comparison of the numerical solution after one period (solid lines) computed with 3DFlux and the exact solution (dashed lines) for three different grid resolutions: $N_x \times N_y = 107 \times 107$ ($\Delta h = 0.01$), $207 \times 207$ ($\Delta h = 0.005$) and $407 \times 407$ ($\Delta h = 0.0025$) where $\Delta h = \Delta x = \Delta y$. The isocontours are plotted from 0.05 to 0.95 in increments of 0.05.](image)
third-order in space PPM scheme [15], Fig. 13 shows that, on average, at the highest resolution (807 × 807), 3DFLUX is twice as accurate as PPM and an order of magnitude more accurate than UTOPIA.

In addition, Fig. 13 confirms the results obtained in test 1. 3DFLUX is about one order of magnitude more accurate than UTOPIA in reconstructing a smooth function that is subjected to substantial stretching. Apparently, the simplified polynomial interpolant used in UTOPIA, where the cross-terms are missing [4], induces a substantial loss of accuracy when the transport in the transverse directions is significant (see Fig. 13).

Although, 3DFLUX is nominally second-order in time and third-order in space, this test confirms that the convergence rate and accuracy of 3DFLUX improve at smaller time steps and, consequently, depend on the Courant numbers chosen. As shown in Fig. 13, the convergence rate of 3DFLUX increases and tends to be third-order when the time step is reduced by an order of magnitude. In fact, the slope of the $L^\infty$ and $L^2$ norms for the two finest spatial resolutions shown in Fig. 13 are respectively equal to 2.39 and 2.48 when $c_x = c_y = 1.0$ and they are equal to 3.00 and 3.39, respectively, when $c_x = c_y = 0.1$. Note also the substantial improvement in accuracy when the Courant number is 0.1.

It is important to note that the accuracy and the convergence rate of 3DFLUX remain unchanged with or without the activation of the flux limiter presented in Section 2.5. To obtain better insight into the efficiency of the selective monotonic limiter, we report the values of the extrema generated by the deformational flow for the different grid resolutions in Table 1. As expected, the flux limiter considerably damps the undershoots that appears in a non-monotonicity preserving solution, and the remaining negative values are so small that they can be neglected. Note that when even small negative values are undesirable, a positivity preserving correction can be implemented (see [15] for details). Finally, we remark that the effect of the flux limiter becomes unobservable at the finest resolution reported in Table 1.

3.4. Test 4: three-dimensional advection–diffusion problem

This test investigates the convergence rate of 3DFLUX when solving a three-dimensional advection–diffusion problem. The test simulates the transport of a Gaussian sphere. The initial scalar field is

$$
\phi(x, y, z, t = 0) = \exp \left[ \frac{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}{-4t} \right],
$$

(69)

Table 1

<table>
<thead>
<tr>
<th>$\Delta h$</th>
<th>3DFLUX with selective monotonic flux-limiter</th>
<th>3DFLUX without flux-limiter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\phi_{\text{max}}$</td>
<td>$\phi_{\text{min}}$</td>
</tr>
<tr>
<td>0.0400</td>
<td>0.583</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.0181</td>
<td>0.957</td>
<td>0.0002</td>
</tr>
<tr>
<td>0.0095</td>
<td>1.004</td>
<td>0.0002</td>
</tr>
<tr>
<td>0.0048</td>
<td>0.999</td>
<td>0.0001</td>
</tr>
<tr>
<td>0.0024</td>
<td>0.999</td>
<td>0.0001</td>
</tr>
<tr>
<td>0.0012</td>
<td>1.000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Extrema and $L^\infty$ norm computed with and without the flux-limiters after one period of the deformational flow. $\Delta h$ is the uniform spatial resolution and the number of nodes, $N_x \times N_y$, are 27 × 27, 57 × 57, 107 × 107, 207 × 207, 407 × 407 and 807 × 807.
where \((x_0, y_0, z_0) = (0.8, 0.8, 0)\) are the coordinates of the center of the Gaussian sphere. The velocity field

\[
\begin{align*}
    u(x, y, z) &= \Omega \left[-y \cos\left(\frac{\pi}{2}\right) - z \sin\left(\frac{\pi}{2}\right)\right], \\
    v(x, y, z) &= \Omega \left[x \cos\left(\frac{\pi}{2}\right) - z \sin\left(\frac{\pi}{2}\right) \cos\left(\frac{\pi}{2}\right)\right], \\
    w(x, y, z) &= \Omega \left[x \sin\left(\frac{\pi}{2}\right) + y \sin\left(\frac{\pi}{2}\right) \cos\left(\frac{\pi}{2}\right)\right],
\end{align*}
\]

(70)

advepts the Gaussian sphere along a circular trajectory lying on a diagonal plane of the cubic domain at a constant angular velocity, \(\Omega = 2\pi\). After a complete rotation, i.e., at time \(t = 1\), the exact solution is

\[
\phi(x, y, z, t = 1) = \frac{1}{2^{3/2}} \exp\left(\frac{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}{-8\Delta t}\right)
\]

(71)

We simulate the evolution of the Gaussian sphere (69) within a cube, \((x, y, z) \in [-2, 2] \times [-2, 2] \times [-2, 2]\), using a uniform spatial discretization, i.e., \(\Delta x = \Delta y = \Delta z = \Delta h\), and a time step corresponding to a Courant number of 0.9. We estimate the numerical error by comparing our solution with the exact solution (71).

The \(L^2\) and \(L^\infty\) norms of the errors are reported in Fig. 14. 3DFLUX is third-order accurate when solving the advection-diffusion problems. Note that for the coarser spatial resolution, the Péclet number \((\rho u_{\text{max}} \Delta h / \Gamma)\), i.e., the ratio of the Courant number over the diffusive stability constraint (43), is 233, whereas for the finer spatial resolution it is 19. This is a considerable improvement with respect to second-order central difference schemes, which generate spurious oscillations for Péclet cell numbers higher than two, see Section 2.2.

Furthermore, Fig. 14 shows the numerical errors obtained in two additional tests: one where the diffusive terms are zero, i.e., \(\Gamma = 0\), and the other where the velocity field (70) is replaced by a zero velocity field. The pure advective solution is compared to the initial solution after one full rotation whereas the pure diffusive solution is compared to the exact solution (71) at \(t = 1\). Fig. 14 shows that 3DFLUX is third-order when solving the purely advective problem, however, it becomes second-order when solving the pure diffusive problem, as noted in Section 2.4.

To characterize the computational cost and efficiency of our code, we compare the CPU time required by 3DFLUX, UTOPIA and a traditional first-order discretization scheme to calculate the solution of the present test up to time \(t = 1\). The low-order scheme is the hybrid scheme described in Spalding [49] and Patankar [1] which combines upwind and central differencing for the advective and diffusive terms, respectively, and use a second-order Runge–Kutta method for the time integration. 3DFLUX, UTOPIA and the hybrid scheme are all second-order in time. The hybrid scheme is a simple and fast algorithm that is implemented in most commercial CFD codes. For all computations, we set the Courant numbers to 0.45 because the hybrid scheme becomes unstable for higher Courant numbers.

The computation was performed using two standard high-end multi-core CPUs (Intel Xeon (R) E5645) running at 2.40 GHz with 48 GB of RAM. We leveraged Intel’s Thread Building Blocks (TBB) library to exploit the capabilities of the multi-core processor. TBB is an object oriented library for the C++ language, which allows programmers to decide the extent of parallelism.

Fig. 15(a) show the CPU time (in seconds) needed for the computation as a function of the spatial resolution. As expected, the first-order scheme is significantly faster than the two other schemes because it performs a substantially smaller number of flops and does not require a flux limiter. We observe that UTOPIA is roughly 20% faster than 3DFLUX. Fig. 15(b) plot the CPU time as a function of the \(L^\infty\) norm of the numerical error. 3DFLUX is clearly the most efficient scheme. For a given value

![Fig. 14. Error analysis for the three-dimensional advection-diffusion problem. Convergence of (a) \(L^2\) and (b) \(L^\infty\) norms as a function of the cell size, \(\Delta h\), when solving the pure advection (○), the pure diffusion (□) and the advection-diffusion (•) test with 3DFLUX. The dashed and solid lines show the slopes of second- and third-order convergence rates, respectively.](image)
of the $L^\infty$ norm, 3DFLUX is about 10 times faster than UTOPIA. The poor efficiency of the hybrid scheme is due to its high numerical diffusion that affects considerably the solution of the multidimensional problems.

3.5. Test 5: turbulent mixing in a channel flow

This test illustrates the potential of 3DFLUX to simulate complex flows of practical relevance. To this end, we simulated a real experiment where a scalar, temperature, is released from a heated line source in a fully developed turbulent channel flow [22,50]. Note that heat is released at a sufficiently small rate so that it has no effect on the dynamics of the fluid motion, i.e. the problem is one-way coupled. For this reason, we pre-compute the turbulent flow with a spectral solver and, once the velocity field is available, we compute the evolution of the scalar field a posteriori using 3DFLUX.

We simulate the experiment [22,50] in a numerical channel of size $(x,y,z) \in (0, 4\pi] \times [-1, 1] \times [0, \pi]$, oriented so that $x$ is the streamwise direction, $y$ is the wall-normal direction, and $z$ is the spanwise direction of the flow. The channel is delimited by two parallel flat walls located at $y = \pm 1$. The scalar field is injected in the flow at constant power by a line source oriented in the spanwise direction located at $(x,y) = (0.24, 0)$, as shown in Fig. 16.
The turbulent velocity field is computed using a spectral code [51] and [52, http://channelflow.org/], which uses a spectral discretization in space (Fourier × Chebyshev × Fourier) and a third-order Runge–Kutta time integrator to solve the incompressible Navier–Stokes equations. The boundary conditions are periodic in the streamwise and spanwise directions and consist of no-penetration and no-slip conditions at the walls. The pressure gradient that drives the flow in the positive x-direction is adjusted dynamically to maintain a constant mass flux through the channel. The code is initialized with a parabolic velocity profile, $\bar{u} = (1 - y)^2, 0, 0$ plus a random disturbance to accelerate the onset of turbulence. The simulations are performed at a Reynolds number $Re = \langle U \rangle h / v = 3475$, where $\langle U \rangle$ is the mean velocity at the centerline of the channel, where the brackets indicate averaged values, $h$ is the channel half-width and $v$ is the kinematic viscosity.

3DFLUX uses the precomputed velocity field as an input to solve the transport equation for the temperature $\phi$. We discretized the channel with a grid $N_x \times N_y \times N_z = 771 \times 195 \times 194$ to resolve both the large and small scales of the turbulent scalar mixing. The boundary conditions for the scalar field are (i) periodic in the spanwise direction, (ii) inflow/outflow at the inlet/outlet of the channel, and (iii) adiabatic (no-flux) at the walls. The line source is simulated by a sequence of nodes having a constant source term $S$.

The temperature field forms a plume that is transported in the streamwise and wall-normal directions, as shown in Fig. 16. The turbulent field induces mixing over a wide range of length scales. A statistical description of the scalar field enables an analysis of the dynamics of mixing. In this test, we compare statistics of the numerical simulations with the ones obtained in our experiments. In addition, we also report the data of Lavertu and Mydlarski [22], where available. Since the Reynolds number of the experiments, $Re = 10350$, is higher than that used in the simulation, we use an appropriate normalization to make a meaningful comparison between the experimental and numerical results.

![Fig. 17. Non-dimensionalized mean temperature excess profiles at different downstream locations. Results from our experiments (a) and 3DFLUX (b) are reported for (×), $x/h = 2.0$; (x), $x/h = 5.0$; (□), $x/h = 10.0$; (□□), $x/h = 12.0$; (+), $x/h = 15.0$. The gray line indicates the transverse location of the source ($y_s/h = 0.0$) and the black lines are Gaussian curve fits to the data.](image1)

![Fig. 18. Non-dimensionalized r.m.s. temperature profiles at different downstream locations. Results from our experiments (a), and 3DFLUX (b) are reported for (×), $x/h = 2.0$; (x), $x/h = 5.0$; (□), $x/h = 10.0$; (□□), $x/h = 12.0$; (+), $x/h = 15.0$. The gray line indicates the transverse location of the source ($y_s/h = 0.0$) and the black lines are Gaussian curve fits to the data.](image2)
Fig. 19. Non-dimensionalized standard deviation of the mean (a) and r.m.s. (b) temperature profiles at different downstream locations. Our experiments (■), 3DFLUX (×) and experimental results of [22] (□). \( \bar{U} \) and \( \bar{u}_{rms} \) are the mean and r.m.s. velocities at the centerline of the channel, respectively.

We first analyze the mean and root-mean-square (r.m.s.) temperature profiles and their peaks and widths. To this end, we define the scalar excess as \( \Delta \phi = \phi - \phi_0 \), where \( \phi_0 \) is the ambient temperature in the channel before the injection of the scalar, and then we decompose it into a mean, \( \langle \Delta \phi \rangle = \phi - \phi_0 \), and a fluctuating part \( \theta \), i.e. \( \Delta \phi = \langle \Delta \phi \rangle + \theta \). The average value of the scalar field, \( \langle \phi \rangle \), at a given location \( (x,y) \) can be expressed as

\[
\phi = \langle \phi(x,y) \rangle = \frac{1}{N_t} \sum_{t=1}^{N_t} \frac{1}{N_x} \sum_{x=1}^{N_x} \phi(x,y,z,t),
\]

where \( N_t \) is the number of time steps required to obtain converged statistics. Since the scalar field is periodic, and thus statistically homogeneous in the spanwise direction, the statistics are computed by averaging the scalar field in the spanwise direction. We obtained converged statistics by running 3DFLUX for \( 2.2h/u_t \), which is equivalent to \( N_t = 5000 \) with \( \Delta t = 4.39 \times 10^{-4}h/u_t \), where \( u_t = \sqrt{-\langle \nabla \rho \cdot \nabla \rho \rangle} \) is the friction velocity and \( \nabla \rho / \nabla x \) is the mean axial pressure gradient.

The mean and the r.m.s. temperature profiles normalized by their respective peak values are plotted as a function of the wall-normal locations at several locations downstream from the line source, see Figs. 17 and 18. Both experimental and numerical mean and r.m.s. profiles agree very well. The symmetry of the Gaussian distribution of the numerical profiles suggests that the flow statistics are correctly converged. Fig. 19 shows the standard deviation of the mean and r.m.s. profiles normalized by the channel half-width at several stations downstream the line source. All these results show an excellent agreement between the numerical simulations performed with 3DFLUX and the experiments.

4. Conclusions

We have presented 3DFLUX, a high-order, fully multidimensional, conservative, monotonicity preserving numerical scheme for the solution of the scalar transport equation. 3DFLUX is nominally third-order in space and second-order in time. Its convergence rate and accuracy have been characterized via several multidimensional tests, both purely advective and advective–diffusive problems. In general, when using the highest possible Courant number (one in our implementation), 3DFLUX presents a noticeably higher accuracy than the currently available numerical schemes having the same convergence rate. This is owed to the stencil presented in this article – a stencil that allows an accurate three-dimensional interpolation of the solution over a computational cell. Unexpectedly, at low Courant numbers, 3DFLUX appears to be superconvergent and, depending on the problem solved, is fourth-order or higher in space.

3DFLUX is very attractive for research-oriented or high-end engineering applications because it does not require dimensional splitting and generates highly accurate solutions at a realistic computational cost. We successfully tested 3DFLUX’s potentials by simulating a complex flow where a scalar is released from a line source in a fully turbulent channel flow. We showed a remarkable agreement between the statistics produced by 3DFLUX and those measured in an experiment.

Acknowledgments

The authors thank P. Blossey and J.F. Gibson for the helpful discussions regarding selective flux-limiters and turbulent channel flow solvers, respectively. Funding was provided by NSERC under Contracts No. RGPIN217169 and No. RGPIN217184.
References


[27] H. Hasumi, T. Daigaku, CCSR ocean component model (CCOC) Version 4.0, Center for Climate System Research, University of Tokyo, 2006.


