Commun. Comput. Phys.
doi: 10.4208/cicp.260511.050811a

Vol. **12**, No. 1, pp. 1-41
July 2012

High Order Schemes on Three-Dimensional General Polyhedral Meshes — Application to the Level Set Method

Thibault Pringuey* and R. Stewart Cant

CFD Laboratory, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK.

Received 26 May 2011; Accepted (in revised version) 5 August 2011

Available online 16 January 2012

Abstract. In this article, we detail the methodology developed to construct arbitrarily high order schemes — linear and WENO — on 3D mixed-element unstructured meshes made up of general convex polyhedral elements. The approach is tailored specifically for the solution of scalar level set equations for application to incompressible two-phase flow problems. The construction of WENO schemes on 3D unstructured meshes is notoriously difficult, as it involves a much higher level of complexity than 2D approaches. This due to the multiplicity of geometrical considerations introduced by the extra dimension, especially on mixed-element meshes. Therefore, we have specifically developed a number of algorithms to handle mixed-element meshes composed of convex polyhedra with convex polygonal faces. The contribution of this work concerns several areas of interest: the formulation of an improved methodology in 3D, the minimisation of computational runtime in the implementation through the maximum use of pre-processing operations, the generation of novel methods to handle complex 3D mixed-element meshes and finally the application of the method to the transport of a scalar level set.

AMS subject classifications: 65M08, 76-04, 76N99

Key words: WENO scheme, three-dimensional, unstructured mesh, mixed element, polyhedral element, hyperbolic equations, level set.

1 Introduction

The use of unstructured meshes in modern industrial CFD has dramatically increased over the years, following the development of more sophisticated numerical methods. The advantages of the unstructured approach are substantial: better resolution of the geometric details and significant reduction of meshing time when compared to structured-mesh

*Corresponding author. *Email addresses:* `tp299@cam.ac.uk` (T. Pringuey), `rsc10@cam.ac.uk` (R. S. Cant) approaches on complex industrial geometries. The need for fine geometric definition is apparent in the simulation of industrially-relevant two-phase flow problems, for example in simulating primary break-up of the fuel spray in gas turbines. As shown in [13], accurate simulations of the atomisation process require precise numerical representation of the injection device. Modern modelling approaches to two-phase flows make use of level set concepts [4, 6, 9, 16, 21, 25, 27] and require the solution of a partial differential equation that represents the evolution of the level set scalar. As a result, the focus of this work has been placed on the development of a numerical method suitable for the transport of a level set on unstructured meshes.

Weighted Essentially Non-Oscillatory (WENO) schemes have become central to level set methods applied to the resolution of multiphase flows. Indeed, they offer the most efficient way to handle the large gradients incurred by the phase boundaries while maintaining a sharp interface. However, the construction of such schemes is much more complicated on unstructured meshes than on Cartesian grids as no particular direction can be identified in the distribution of the elements.

WENO schemes were originally developed for Cartesian grids [19, 24]. They were an evolution of the Essentially Non-Oscillatory (ENO) schemes introduced by Harten in 1987 [14, 15] to achieve high order accuracy and non-oscillatory properties near discontinuities such as shock waves in high-speed compressible flows. As the interest in unstructured meshes grew, WENO schemes were constructed for triangular [12, 17] and tetrahedral meshes [7, 8, 42].

These numerical schemes cope with the presence of discontinuities in the flow field by considering the solution on N_S stencils distributed around the targeted cell. Whereas the ENO methodology simply selects the stencil on which the solution is the smoothest, the WENO approach combines the data from the different stencils and weights their relative contributions according to the respective smoothness of the N_S datasets. As a result, WENO schemes reach higher orders of accuracy than ENO schemes at equal cost. In particular, Jiang showed that a WENO scheme constructed using r^{th} order ENO schemes could reach $(2r-1)^{\text{th}}$ order of accuracy in smooth regions of the flow [19].

The basic principles of the construction of WENO schemes for triangular meshes were presented by Friedrich [12], based on the work of Abgrall [1]. Later, Dumbser extended these ideas to tetrahedral meshes [7] and defined an approach to devise arbitrarily high order schemes. Titarev [36] improved this approach for two dimensional (2D) computational domains and produced high order schemes on mixed-element 2D meshes. We have extended the approach of Dumbser [7] and Titarev [36] to 3D mixed-element unstructured grids for linear hyperbolic equations [30]. Tsoutsanis [38] describes a similar method developed independently and has applied it to the solution of the Euler equations. In this paper, we present an extension of our scheme [30] to general polyhedral cells and apply it to the solution of the level set equation and the Burgers' equation.

It should be noted that many of the previous WENO schemes have been developed for application to high-speed flows, and specifically for the solution of systems of equations such as the Euler equations. This is not the aim of the present work, where attention is focused instead on the solution of a single level set equation for application to two-phase flows. The coupling to the remainder of the equation set is expected to be problem-specific, and may involve either compressible or incompressible flows, and either Euler or Navier-Stokes type equations.

In this article, we detail the methodology developed, and implemented in parallel, to construct arbitrarily high order schemes — linear and WENO — on 3D mixed-element unstructured meshes, consisting of general convex polyhedra. The linear reconstruction procedure produced for 3D mixed-element unstructured grids is detailed in Section 3.1. Then, in Section 3.2, the main points of the WENO reconstruction are presented. In Section 3.3, the computation of the numerical flux is explained. The application of the technique to the solution of level set equations is considered in Section 4. Results are presented in Section 6 for a set of test cases in 2D and 3D, and conclusions are drawn in Section 7.

2 Overview of the method

The scope of the method is restricted to general hyperbolic systems of first-order partial differential equations [7,8]. Such systems may be expressed in 3D as:

$$\frac{\partial}{\partial t}\mathbf{U}+\frac{\partial}{\partial x}\mathbf{F}(\mathbf{U})+\frac{\partial}{\partial y}\mathbf{G}(\mathbf{U})+\frac{\partial}{\partial z}\mathbf{H}(\mathbf{U})=\mathbf{0}, \tag{2.1}$$

where $\mathbf{U}$ is the vector of conserved variables and $\mathbf{F}(\mathbf{U})$, $\mathbf{G}(\mathbf{U})$ and $\mathbf{H}(\mathbf{U})$ are the vectors of the fluxes respectively in the x, y and z directions.

The computational domain is denoted by Ω and is discretised using conforming elements E_i of volume $|E_i|$ and boundary ∂E_i. Integrating (2.1) over the element E_i leads to:

$$\iiint\limits_{E_i}\frac{\partial}{\partial t}\mathbf{U}\mathrm{d}E_i+\iiint\limits_{E_i}\nabla\cdot\mathcal{A}\mathrm{d}E_i=\mathbf{0} \tag{2.2}$$

with the divergence of the second rank tensor $\mathcal{A}=(\mathbf{F},\mathbf{G},\mathbf{H})$ given by:

$$\nabla\cdot\mathcal{A}=\frac{\partial}{\partial x}\mathbf{F}(\mathbf{U})+\frac{\partial}{\partial y}\mathbf{G}(\mathbf{U})+\frac{\partial}{\partial z}\mathbf{H}(\mathbf{U}).$$

Assuming that the control volume E_i is fixed and therefore independent of time t and introducing $\overline{\mathbf{U}}_i$, the cell average of the conserved variables at time t, we can re-write the first term on the left-hand-side (l.h.s.) of (2.2) as:

$$\iiint\limits_{E_i}\frac{\partial}{\partial t}\mathbf{U}\mathrm{d}E_i=\frac{\mathrm{d}}{\mathrm{d}t}\iiint\limits_{E_i}\mathbf{U}\mathrm{d}E_i=|E_i|\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{U}}_i. \tag{2.3}$$

Inserting (2.3) in (2.2), and applying the divergence theorem to the second term on the l.h.s. of (2.2), leads to the finite volume formulation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{U}}_i+\frac{1}{|E_i|}\iint\limits_{\partial E_i}\mathcal{A}\cdot\mathbf{n}\mathrm{d}(\partial E_i)=\mathbf{0}, \tag{2.4}$$

where $\mathbf{n}=(n_x,n_y,n_z)$ is the outward unit vector normal to the surface ∂E_i and $\mathcal{A}\cdot\mathbf{n}\mathrm{d}(\partial E_i)$ is the flux component normal to ∂E_i.

Splitting the integral over the contour of the element ∂E_i into L_i integrals over the faces F_l of E_i, and introducing the vector $\mathbf{A}_{n_l}=\mathcal{A}\cdot\mathbf{n}_l$ ($\mathbf{n}_l$ being the outward unit vector normal to F_l), we express the second term on the l.h.s. of (2.4) as:

$$\iint\limits_{\partial E_i}\mathcal{A}\cdot\mathbf{n}\mathrm{d}(\partial E_i)=\sum_{l=1}^{Li}\iint\limits_{F_l}\mathbf{A}_{n_l}\mathrm{d}(F_l)=\sum_{l=1}^{Li}\mathbf{A}_{il}. \tag{2.5}$$

The inter-cell flux $\mathbf{A}_{il}$ associated with the face F_l of the element E_i is efficiently calculated with a Gauss Legendre quadrature of appropriate order. Inserting (2.5) into (2.4) leads to:

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{U}}_i+\frac{1}{|E_i|}\sum_{l=1}^{L_i}\mathbf{A}_{il}=\mathbf{0}. \tag{2.6}$$

In order to time-advance the cell-averages of the variables, the finite volume scheme (2.6) requires the determination of the inter-cell fluxes. A reconstruction operator is therefore necessary in each element E_i to provide the point-wise values of the solution as needed by the flux evaluation. By using a polynomial reconstruction operator, the finite volume method can reach arbitrarily high orders of accuracy in space on any type of grid.

The difficulty is to design an efficient polynomial reconstruction that would allow high order accuracy while remaining cost effective. A linear reconstruction operator applied on a single stencil in general will produce spurious oscillations in the vicinity of discontinuities. As multiphase flows systematically involve a severe discontinuity (e.g., density jumps of several orders of magnitude) it is crucial to mitigate this deficiency. For this purpose, we make use of Weighted Essentially Non-Oscillatory (WENO) schemes [7,17–19,24,30,35,38,41,42]. The WENO reconstruction is performed on unstructured meshes by applying a linear reconstruction procedure to the various WENO stencils and combining the polynomials obtained for each stencil, according to the solution-dependent weights.

A feature of the methodology presented here is the tetrahedral decomposition of the mesh to handle the geometrical complexity of 3D mixed-element meshes. This makes our technique quite general and appropriate for most industrial problems.

3 Numerical formulation

3.1 Methodology for the linear reconstruction

The linear reconstruction is presented in this section for a scalar variable, however its extension to vector variables is straightforward. For each cell of the computational domain, the linear reconstruction procedure produces a polynomial representing the variable $u(x,y,z,t)$ everywhere in the cell. The stencil $\mathcal{S}$ of cells E_j is used to generate the interpolating polynomial on the targeted cell E_0. For convenience the first element of the stencil is the targeted cell. Hence:

$$\mathcal{S}=\bigcup_{j=0}^{j\max} E_j. \tag{3.1}$$

For each cell of the mesh, the polynomial $p(x,y,z,t)$ is reconstructed from the cell-averages of the variables in the cells of the associated stencil $\mathcal{S}$. The reconstruction is performed with the constraint that the integral of the polynomial over the targeted cell equals the cell-average value in this cell. This conservation condition is expressed as:

$$\overline{u}_0=\frac{1}{|E_0|}\iiint_{E_0} p(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z. \tag{3.2}$$

The polynomial reconstruction in physical coordinates $\mathbf{x}=(x,y,z)$ on a general unstructured mesh requires the consideration of scaling effects. However, it is crucial for the generality of the method to undertake the reconstruction in a reference space $\boldsymbol{\xi}=(\xi,\eta,\zeta)$ where scaling effects do not apply (see [7]). As well as elegantly simplifying the polynomial reconstruction, working in a reference space prevents the cumbersome introduction of inaccurate scaling factors.

It is necessary to relate the physical coordinates to the reference coordinates by the mapping $\mathbf{x}=\mathbf{x}(\xi,\eta,\zeta)$. This operation requires the coordinates of four different vertices of the targeted element E_i: one for the origin of the frame, and three others — linked to the origin by an edge of E_i — to form the basis. For each cell in the mesh, the inverse of this cell-dependent mapping: $\boldsymbol{\xi}=\boldsymbol{\xi}(x,y,z)$ is applied to the targeted cell and its stencil. This leads to a stencil in the reference space:

$$\mathcal{S}'=\bigcup_{j=0}^{j\max} E'_j. \tag{3.3}$$

Since the spatial average is not affected by the affine transformation $\boldsymbol{\xi}=\boldsymbol{\xi}(x,y,z)$, the conservation condition is also valid in the reference space. With $p(\xi,\eta,\zeta)$ defined as the outcome of the polynomial reconstruction in the reference space, we have:

$$\overline{u}_0=\frac{1}{|E'_0|}\iiint_{E'_0} p(\xi,\eta,\zeta)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta. \tag{3.4}$$

To design a method for the determination of $p(\xi,\eta,\zeta)$, it is convenient to express the polynomial in a basis of polynomial functions: $\{\phi_k(\xi,\eta,\zeta)\}_{k=0,\cdots,K}$. Introducing the degrees of freedom a_k, each associated to a given basis function, we write:

$$p(\xi,\eta,\zeta)=\overline{u}_0+\sum_{k=1}^{K}a_k\phi_k(\xi,\eta,\zeta). \tag{3.5}$$

Therefore, the interpolating polynomial is completely defined by the set of $(K+1)$ basis functions ϕ_k and their associated degrees of freedom a_k. Also, the integer K is related to the degree of the polynomial r by the expression:

$$K=\frac{(r+1)(r+2)(r+3)}{6}-1. \tag{3.6}$$

The basis functions must be constructed so that the conservation condition (3.4) is respected. This implies that the mean value of each basis function over E'_0 is null. As in [36], we choose:

$$\phi_k=\psi_k-\frac{1}{|E'_0|}\iiint\limits_{E'_0}\psi_k\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta, \tag{3.7}$$

where $\{\psi_k\}=\xi,\eta,\zeta,\xi^2,\xi\cdot\eta,\cdots,\zeta^r, k=1,\cdots,K$. Knowing the basis functions, we can calculate the degrees of freedom a_k by requiring that the cell-averages of $p(\xi,\eta,\zeta)$ over each cell E'_j of the stencil $\mathcal{S}'$ is equal to the corresponding cell-average of the variable: $\overline{u}_j$. This is expressed by the formula:

$$\overline{u}_j=\frac{1}{|E'_j|}\iiint\limits_{E'_j}p(\xi,\eta,\zeta)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \qquad j=1,\cdots,j_{\max}, \tag{3.8}$$

$$=\overline{u}_0+\sum_{k=1}^{K}\left(\frac{1}{|E'_j|}\iiint\limits_{E'_j}\phi_k(\xi,\eta,\zeta)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta\right)a_k \qquad j=1,\cdots,j_{\max}, \tag{3.9}$$

$$=\overline{u}_0+\sum_{k=1}^{K}A_{jk}a_k \qquad j=1,\cdots,j_{\max}, \tag{3.10}$$

where A_{jk} is the cell-average of the basis function ϕ_k over the cell E'_j of $\mathcal{S}'$. Introducing $b_j=(\overline{u}_j-\overline{u}_0)$ in (3.10), the system of equations for the degrees of freedom a_k takes the matrix form:

$$\sum_{k=1}^{K}A_{jk}a_k=b_j, \qquad j=1,\cdots,j_{\max}. \tag{3.11}$$

It can be seen from (3.9) and (3.11) that the reconstruction matrix $\mathcal{A}$ is solution independent. This means that the time-consuming operations required for the determination of the a_k can be pre-processed.

The solution of (3.11) provides the polynomial needed for the calculation of the intercell fluxes (see Section 2). The determination of the fluxes depends on the problem considered and on the Riemann solver chosen. The application of this numerical method to the transport of the level set is detailed in Section 4.

3.1.1 Generation of a central stencil

The most compact stencil $\mathcal{S}$ in the physical coordinate system (x,y,z) is first constructed for all the cells of the domain. Then, all the stencils $\mathcal{S}$ are mapped to the reference coordinate system (ξ,η,ζ) to produce the stencils $\mathcal{S}'$.

Number of cells of the stencil The purpose of the stencil is to provide a dataset $(\overline{u}_j,E_j)$ for the polynomial reconstruction. Hence the number cells in the stencil $j_{\max}$ should be greater than or equal to the number of degrees of freedom K (see (3.6)). However, a stencil with the minimum number of cells may lead to unstable schemes on general meshes [7, 36]. Also, in the case $j_{\max}=K$ the square matrix $\mathcal{A}$ may not always be invertible for specific geometrical configurations. As a result, larger stencils are considered to maintain robustness. Typical sizes are: $1.5K$ in 2D and $2K$ in 3D [2,20,26].

Compact stencil in the physical space In preparation for the determination of stencils for high order schemes, we choose to pre-define the list of "point-neighbours" for each cell in the mesh. This list gathers all the cells that share a point with a given cell (see Fig. 1 for an illustration in 2D). Creating such a data structure significantly simplifies the algorithm for stencil generation and increases the speed of the overall process.

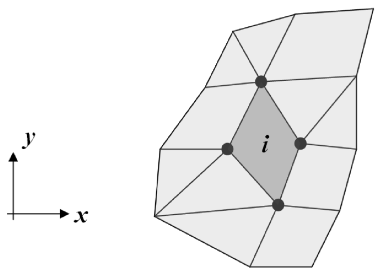


Figure 1: Schematic of the point-neighbours (light grey) of a targeted cell (dark grey) — in 2D.

Starting from the point-neighbours of the targeted cell E_0, the method progresses by an iterative search over the point-neighbours of the cells in a dynamic list. The point-neighbours of all the cells in the list are consequently added to the list at each iteration. The cells added at a given iteration can be perceived as an additional layer of cells surrounding the cells already present in the list. This is illustrated in Fig. 2. The loop stops when the total number of cells in the list is greater than or equal to $j_{\max}$. Then, the cells in the list are sorted according to their centre-to-centre distance from the targeted cell. In order to produce the most compact stencil, the $j_{\max}$ cells closest to E_0 are selected for the stencil.

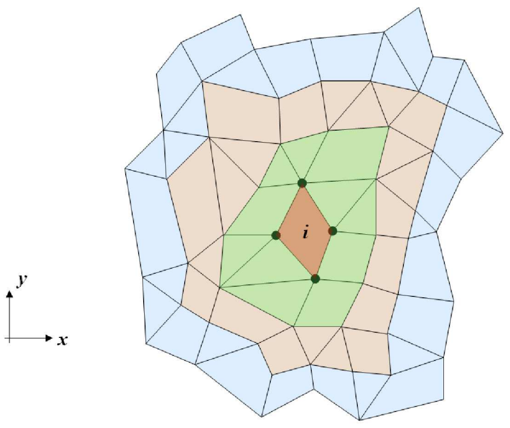


Figure 2: Layers of cells added to the dynamic list at each iteration: in indigo, the cells added at the 1st iteration; in blue, the cells added at the 2nd iteration — in 2D.

Mapping to a reference space Let us consider $M_0=(x_0,y_0,z_0)$ to be any vertex of the targeted element E_0 and $M_1=(x_1,y_1,z_1)$, $M_2=(x_2,y_2,z_2)$, $M_3=(x_3,y_3,z_3)$ to be three different vertices of E_0 linked to M_0 through an edge of E_0 such that the frame of reference $\mathcal{R}_0=(M_0,\overrightarrow{M_0M_1},\overrightarrow{M_0M_2},\overrightarrow{M_0M_3})$ is direct. The mapping $\mathbf{x}=\mathbf{x}(\xi,\eta,\zeta)$ can be expressed as:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \mathcal{J} \begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix}, \tag{3.12}$$

where the Jacobian of the transformation $\mathcal{J}$ is given by:

$$\mathcal{J} = \begin{pmatrix} x_1-x_0 & x_2-x_0 & x_3-x_0 \\ y_1-y_0 & y_2-y_0 & y_3-y_0 \\ z_1-z_0 & z_2-z_0 & z_3-z_0 \end{pmatrix}. \tag{3.13}$$

The algorithm applies the inverse of the mapping to the cell centres and vertices. For a generic point $P_{\mathbf{x}}=(x_P,y_P,z_P)$, the inverse transformation providing $P_{\boldsymbol{\xi}}=(\xi_P,\eta_P,\zeta_P)$ is:

$$\begin{pmatrix} \xi_P \\ \eta_P \\ \zeta_P \end{pmatrix} = \mathcal{J}^{-1} \begin{pmatrix} x_P-x_0 \\ y_P-y_0 \\ z_P-z_0 \end{pmatrix} \tag{3.14}$$

and the volumes of the transformed elements E'_j are given by:

$$|E'_j| = |\mathcal{J}^{-1}|\cdot|E_j|. \tag{3.15}$$

Noting that the mapping gives $\mathrm{d}x\mathrm{d}y\mathrm{d}z=|\mathcal{J}|\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta$ and using (3.15), we can prove that the conservation condition (3.4) is maintained after the affine transformation:

$$\iiint_{E_j} p(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z = |E_j|\cdot\overline{u}_j, \qquad |\mathcal{J}|\iiint_{E'_j} p(\xi,\eta,\zeta)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta = |\mathcal{J}|\cdot|E'_j|\cdot\overline{u}_j. \tag{3.16}$$

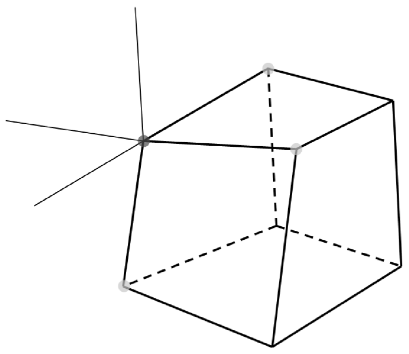


Figure 3: Schematic of the edge-neighbours (light grey vertices) of a given point (dark grey vertex) in a given cell (thick lines) — for a hexahedral cell.

In preparation of the mapping algorithm, we choose to pre-define the list of "edge-neighbours" for each point of each cell in the mesh. This list gathers all the points of a given cell that share an edge with a given point of this cell (see Fig. 3). As with the list of point-neighbours for the stencil generation, this data structure significantly simplifies the implementation and speeds up the overall process.

3.1.2 Determination of the reconstruction matrix

As explained at the beginning of Section 3.1, the determination of the degrees of freedom a_k of the polynomial reconstruction involves the integration of the basis functions ϕ_k over the cells of the transformed stencil E'_j (see (3.9) and (3.10)). Each of these integrations is an element of the reconstruction matrix $\mathcal{A}$ involved in the equation for the degrees of freedom a_k (3.11). Combining (3.7) and (3.9), we get:

$$\begin{aligned} A_{jk} &= \frac{1}{|E'_j|} \iiint\limits_{E'_j} \left(\psi_k - \frac{1}{|E'_0|} \iiint\limits_{E'_0} \psi_k \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta \right) \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta \\ &= \frac{1}{|E'_j|} \iiint\limits_{E'_j} \psi_k \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta - \frac{1}{|E'_0|} \iiint\limits_{E'_0} \psi_k \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta, \qquad j = 1, \cdots, j_{\max}. \end{aligned} \tag{3.17}$$

Therefore, Eq. (3.17) reduces the calculation of A_{jk} to a simple combination of monomial integrations over E'_j and E'_0. According to Stroud [33], the most efficient way of calculating such multiple integrals over simple geometrical domains is to use Gauss-Legendre quadratures. However, Gaussian quadratures are only available over simple geometries such as unit n-simplexes or unit n-cubes. Even in the specific case of an unstructured hexahedral mesh, the trilinear mapping from a unit cube to a hexahedral cell is often non-invertible (see [39] for the conditions of non-degeneracy for hexahedral cells).

As a result, we choose to decompose each element of the mesh systematically into tetrahedra. Since a unit 3-simplex can always be mapped to a tetrahedron through an invertible transformation, the Gaussian quadratures can proceed. The method developed sums the Gaussian quadratures of the monomials calculated over all the tetrahedral sub-elements of a cell and then applies Eq. (3.17) to provide the matrix elements A_{jk}.

Tetrahedralisation of the mesh In order to ensure the convergence of the tetrahedralisation, we chose to split the polyhedral cells into tetrahedra that all have the centre of the original element as a vertex. In this manner, the polyhedron can be decomposed regardless of the way its faces are split and the tetrahedral decomposition is always possible. The faces of the elements (convex polygons) are split into as many triangles as they have sides using the face centre as common vertex. This guarantee the convergence of the tetrahedral decomposition on generic meshes composed of convex polyhedra (see Fig. 4).

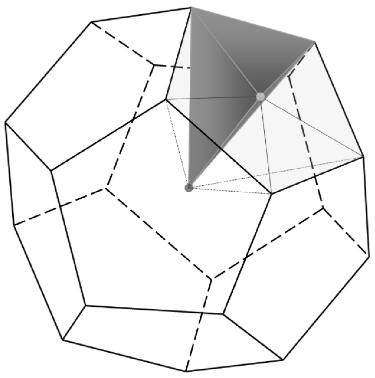


Figure 4: Tetrahedralisation of a convex polyhedron — in solid grey: a sub-element; thin lines: construction lines for the sub-elements; dark grey point: cell centre; shaded face: face decomposed; light grey point: centre of the face considered.

To minimise the number of tetrahedra, the pyramids are still split into two sub-elements and the quadrilateral faces are split in two triangles. This method ensures the convergence of the tetrahedralisation as long as the quadrilateral faces are decomposed before the cells. This decomposition is illustrated for hexahedral and pentahedral cells in Fig. 5.

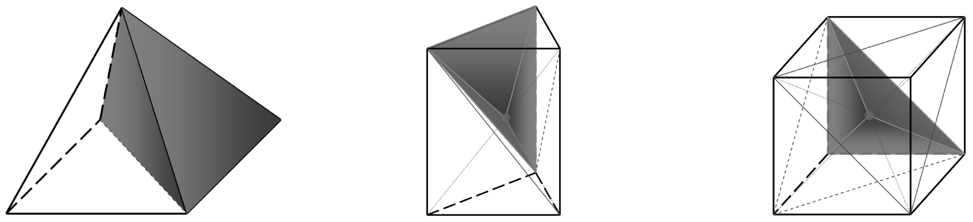

(a) Pyramid into 2 tets (b) Pentahedron into 8 tets (c) Hexahedron into 12 tets

Figure 5: Tetrahedral decomposition ensuring convergence — in solid grey: a sub-element; thin lines: construction lines for the sub-elements; light grey point: cell centre.

3.1.3 Calculation of the degrees of freedom

Inversion of the reconstruction matrix As mentioned in Section 3.1.1, in order to ensure the stability of the scheme on general meshes, the method collects more data than needed for the polynomial reconstruction by constructing a stencil of $j_{\max} > K$ elements. This leads to the over-determination of the system of Eq. (3.11) for the degrees of freedom a_k.

Such a system is generally solved using a least-squares approach [7]. However, Titarev [36] stated that such a method was not optimum for polynomial reconstruction of order higher than three. Indeed, Titarev argues that the condition number of the linear system — produced by the application of the least-squares method for high-order polynomials — is approximately the square of the condition number of the matrix $\mathcal{A}$. This results in a potential lack of accuracy "for smooth problems and very fine meshes" [36].

Consequently, we follow the same approach as Titarev: direct matrix factorisation of $\mathcal{A}$ with a Singular Value Decomposition (SVD) procedure. This is more demanding in computational time and storage than other methods, but it is considered as the most reliable method to compute the degrees of freedom, as it deals better with "errors in data, round-off errors and linear dependence" [10].

With $\mathcal{A}$ a real $j_{\max} \times K$ matrix with $j_{\max} \geq K$, we can write [11]:

$$\mathcal{A} = \mathcal{U}\Sigma\mathcal{V}^T, \tag{3.18}$$

where $\mathcal{U}$ is the $j_{\max} \times K$ matrix of the K orthonormalised eigenvectors associated with the K largest eigenvalues of $\mathcal{A}\mathcal{A}^T$, such that $\mathcal{U}^T\mathcal{U} = \mathcal{I}_K$, $\mathcal{V}$ is the $K \times K$ square matrix of orthonormalised eigenvectors of $\mathcal{A}^T\mathcal{A}$, such that $\mathcal{V}^T\mathcal{V} = \mathcal{I}_K$, and $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_K)$, where σ_i are the non-negative square roots of the eigenvalues of $\mathcal{A}\mathcal{A}^T$. The matrices $\mathcal{U}$ and $\mathcal{V}$ can then be used to transform the Eq. (3.11) into an equivalent diagonal set of equations:

$$\mathcal{A}\mathbf{a} = \mathbf{b}, \qquad (\mathcal{U}\Sigma\mathcal{V}^T)\mathbf{a} = \mathbf{b}, \qquad \Sigma(\mathcal{V}^T\mathbf{a}) = (\mathcal{U}^T\mathbf{b}), \qquad \Sigma\overline{\mathbf{a}} = \overline{\mathbf{b}}. \tag{3.19}$$

In order to illustrate the need to introduce a tolerance τ, below which singular values are neglected, let us assume that $\mathrm{rank}(\mathcal{A}) = K$. Then, none of the singular values σ_k is equal to zero and one may solve Eq. (3.19) by setting:

$$\overline{a}_k = \frac{\overline{b}_k}{\sigma_k}. \tag{3.20}$$

In the case of small σ_k, this may lead to undesirable sensitivity of the a_k to inaccuracies in the data and round-off errors. Hence, in order to compute a robust solution for the degrees of freedom, it is necessary to neglect all the singular values smaller than a given tolerance τ, representative of the accuracy of the data and the floating-point arithmetic. As a consequence, Σ is replaced by $\Sigma_\tau = \mathrm{diag}(\sigma_{\tau,k})$ in (3.19) such that whenever $\sigma_k < \tau$, $\sigma_{\tau,k}$ is set to zero. Since it is always preferable to minimise the coefficients, whenever $\sigma_{\tau,k} = 0$, then $\overline{a}_k = 0$.

However, as the method calculates the degrees of freedom for all the cells of the mesh, storing the $(3\times N_{cells})$ matrices produced by the SVD is memory consuming. To mitigate this, we chose to compute and store the "effective" Moore-Penrose pseudo-inverse $\mathcal{A}_\tau^\dagger$ of $\mathcal{A}$ [10] which also further reduces the number of runtime operations. This matrix can be defined in terms of the tolerance τ as:

$$\mathcal{A}_\tau^\dagger=\mathcal{V}\Sigma_\tau^\dagger\mathcal{U}^T, \tag{3.21}$$

where

$$\Sigma_\tau^\dagger=\text{diag}\left(\sigma_{\tau,k}^\dagger\right) \quad \text{with: } \sigma_{\tau,k}^\dagger=\begin{cases} \dfrac{1}{\sigma_k}, & \text{if } \sigma_k>\tau, \\ 0, & \text{otherwise.}\end{cases}$$

Runtime operations for the degrees of freedom All the above steps of the linear reconstruction are pre-computed and the results are stored. The runtime operations for the degrees of freedom then reduce to two trivial steps for every cell in the mesh:

1. The generation of the vector of data $\mathbf{b}$ required for the calculation of the degrees of freedom in (3.11). The components of $\mathbf{b}$ are computed from the cell-averages of the variable $\overline{u}_j(t)$ in the cells of the stencil $\mathcal{S}$ at a given time t:

$$b_j=\overline{u}_j(t)-\overline{u}_0(t), \qquad j=1,\cdots,j_{\max}. \tag{3.22}$$

2. The determination of the degrees of freedom from the "effective" pseudo-inverse $\mathcal{A}_\tau^\dagger$ and the vector of data $\mathbf{b}$:

$$\mathbf{a}=\mathcal{A}_\tau^\dagger\mathbf{b}. \tag{3.23}$$

When rank$(\mathcal{A})<K$ the equality in (3.11) no longer holds and $\mathcal{A}\mathbf{a}$ is only approximately equal to $\mathbf{b}$. There exists a set of solutions $\mathfrak{S}$, which minimises the Euclidean norm of the residual:

$$\mathfrak{S}=\left\{\mathbf{a}\,\middle|\,|\mathcal{A}\mathbf{a}-\mathbf{b}|_2=\min\right\}. \tag{3.24}$$

It can be shown that the pseudo-inverse provides the shortest vector $\widehat{\mathbf{a}}$ that minimises the norm [10,28]:

$$\widehat{\mathbf{a}}=\mathcal{A}_\tau^\dagger\mathbf{b} \implies \begin{cases} \widehat{\mathbf{a}}\in\mathfrak{S}, \\ \forall\mathbf{a}\in\mathfrak{S}: \ |\widehat{\mathbf{a}}|_2\le|\mathbf{a}|_2. \end{cases} \tag{3.25}$$

Hence, the solution (3.23) is the least-squares solution to the system of Eq. (3.11).

3.2 Methodology for WENO schemes

As in Section 3.1, the WENO reconstruction procedure is presented in this section for a scalar variable, however its extension to vector variables is straightforward. The WENO reconstruction operator is based on several stencils $\mathcal{S}_m$: one central stencil $\mathcal{S}_0$ — generated in the same way as in Section 3.1.1 — and several sectoral stencils that covers all spatial directions in the vicinity of the targeted cell E_i. For convenience, the first element of the stencil is taken as the targeted cell. The appropriate minimal number of one-sided stencils that ensures the self-adaptation of the scheme near discontinuities is obtained by selecting one sectoral stencil per internal face of the cell.

For the cell faces near to the boundaries of the domain, the associated one-sided stencil may have to be discarded as the sector may not encompass enough cells depending on the order of the scheme. As a result, the number of sectoral stencils N_{S_i} per mesh cell E_i varies according to the location of the cell with respect to the boundaries. The set of stencils for a given cell is then:

$$\mathscr{U} = \bigcup_{m=0}^{N_{S_i}} \mathcal{S}_m. \tag{3.26}$$

Once the set of stencils is generated for all the cells of the mesh, the method proceeds with the linear polynomial reconstruction on each stencil $\mathcal{S}_m$ of each cell E_i as described in Section 3.1. The WENO polynomial reconstruction is then given by the convex combination of all the polynomials $p_m(\xi,\eta,\zeta)$ associated with the stencils $\mathcal{S}_m$. Introducing the non-linear WENO weights w_m related to the stencils $\mathcal{S}_m$, we have:

$$p_{WENO}(\xi,\eta,\zeta) = \sum_{m=0}^{N_{S_i}} w_m \cdot p_m(\xi,\eta,\zeta), \tag{3.27}$$

where

$$p_m(\xi,\eta,\zeta) = \overline{u}_0 + \sum_{k=1}^{K} a_k^{(m)} \phi_k(\xi,\eta,\zeta), \tag{3.28a}$$

$$w_m = \gamma_m \Big(\sum_{m=0}^{N_{S_i}} \gamma_m \Big)^{-1} \quad \text{with} \quad \gamma_m = \frac{d_m}{(\varepsilon + IS_m)^p}. \tag{3.28b}$$

The expression for the non-linear weights (3.28b) involves the following parameters:

- d_m, the linear weight. Since the central stencil generally performs better for smooth solutions, following Dumbser [7] we choose to attribute a much larger linear weight to the central stencil.
- IS_m, the oscillation indicator which characterises the level of smoothness of the solution on the stencil $\mathcal{S}_m$. A smooth solution leads to a smaller oscillation indicator and therefore a larger weight.
- ε, a small positive number introduced in γ_m to prevent the denominator from becoming zero.
- p, the exponent of the oscillation indicator, devised to ensure that the contribution of non-smooth stencils vanishes as the cell size tends to zero.

We have chosen the following typical values for the WENO parameters [7,20,36]:

$$d_m = \begin{cases} 10^3, & \text{if } m=0, \\ 1, & \text{otherwise}, \end{cases} \qquad \varepsilon = 10^{-6}, \qquad p=4. \tag{3.29}$$

From the expressions for IS_m given in [7,36], the following matrix formulation can be derived:

$$IS_m = \sum_{p=1}^{K} a_p^{(m)} \cdot \Big(\sum_{q=1}^{K} B_{pq} a_q^{(m)} \Big). \tag{3.30}$$

In (3.30), B_{pq} is an element of the oscillation indicator matrix $\mathcal{B}$. With r denoting the order of the linear polynomial reconstruction, and noting that $\gamma = \Lambda - \alpha - \beta$, the elements of $\mathcal{B}$ are expressed as:

$$B_{pq} = \sum_{\Lambda=1}^{r} \sum_{\alpha=0}^{\Lambda} \sum_{\beta=0}^{\Lambda-\alpha} \iiint_{E_0'} \frac{\partial^\Lambda}{\partial\xi^\alpha \partial\eta^\beta \partial\zeta^\gamma} \phi_p(\xi,\eta,\zeta) \cdot \frac{\partial^\Lambda}{\partial\xi^\alpha \partial\eta^\beta \partial\zeta^\gamma} \phi_q(\xi,\eta,\zeta) \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta. \tag{3.31}$$

As for the reconstruction matrix $\mathcal{A}$, it may be seen from (3.31) that the oscillation indicator matrix $\mathcal{B}$ is solution independent, and hence the time consuming operations required for the determination of IS_m can be pre-processed. In addition, the elements B_{pq} involve the integration of polynomials on the targeted cell in the reference space E_0'. As for the computation of $\mathcal{A}$ we shall apply a Gauss-Legendre quadrature.

The WENO polynomial p_{WENO} can be expressed as a function of the modified degrees of freedom $\tilde{a}_k$ by introducing (3.28a) in Eq. (3.27) and using the condition $\sum_m w_m = 1$ [36]:

$$\begin{aligned} p_{WENO}(\xi,\eta,\zeta) &= \sum_{m=0}^{N_{S_i}} w_m \cdot \Big(\overline{u}_0 + \sum_{k=1}^{K} a_k^{(m)} \phi_k(\xi,\eta,\zeta) \Big) = \overline{u}_0 + \sum_{k=1}^{K} \Big(\sum_{m=0}^{N_{S_i}} w_m a_k^{(m)} \Big) \cdot \phi_k(\xi,\eta,\zeta) \\ &= \overline{u}_0 + \sum_{k=1}^{K} \tilde{a}_k \phi_k(\xi,\eta,\zeta). \end{aligned} \tag{3.32}$$

The combination of the $(N_{S_i}+1)$ set of degrees of freedom $a_k^{(m)}$ into a single set of modified degrees of freedom $\tilde{a}_k$, as in [36], simplifies the algorithm and speeds up the computation of the WENO reconstruction.

3.2.1 Generation of the sectoral stencils

In order to ensure the self-adaptability of the scheme near discontinuities, additional one-sided stencils are associated with the targeted cell. To minimise the number of stencils, Titarev suggested the following guidelines [36]:

1. The stencils are disjoint (apart from the targeted cell).
2. The union of one-sided stencils covers the whole space surrounding the targeted cell.

3. Just like $\mathcal{S}_0$, the stencils are compact (see Section 3.1.1): i.e., in a given sector of the physical space the stencil gathers the $j_{\max}$ elements having the minimum centre-to-centre distance to the targeted cell.

Definition of the sectors In order to remain as general as possible on a 3D mixed-element unstructured mesh, we aim to take as many compact stencils as there are internal faces FI_l of the considered element E_0. For each face FI_l, we choose to define the sector — in which the stencil will be constructed — from the contour of FI_l and the centre C_0 of the cell E_0. As illustrated in Fig. 6, the cells selected to form the sector stencil have their centre in the portion of physical space:

- encompassing the centre of the face FI_l;
- delimited by the surface of the cone $\mathscr{C}_l$ admitting C_0 as apex and the contour of FI_l as directrix.

This approach satisfies all of the three rules given above. In particular, the sectoral stencils cover all the spatial directions in the vicinity of the targeted cell. As a result, the "reverse sectors" of Kaser's approach [7, 20] are unnecessary. This leads to a smaller number of stencils and thus to a faster method.

As mentioned at the beginning of Section 3.1, it is not always possible to find $j_{\max}$ cells in a sector when the internal face considered is close to a boundary (see Fig. 6). The algorithm implemented systematically discards any stencil of less than $j_{\max}$ elements to ensure the robustness of the scheme in the vicinity of a boundary. It follows that the actual number of one-sided stencils of a given cell E_0 is at most the number of its internal faces N_{FI}:

$$N_{S_i} \leq N_{FI}. \tag{3.33}$$

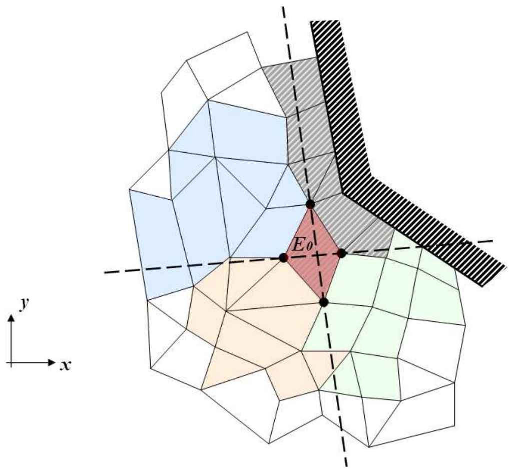


Figure 6: Ten-cells sectoral stencils of E_0 coloured by sector; discarded stencil hatched in grey; domain boundaries hatched in black — as produced by the fast search procedure in 2D.

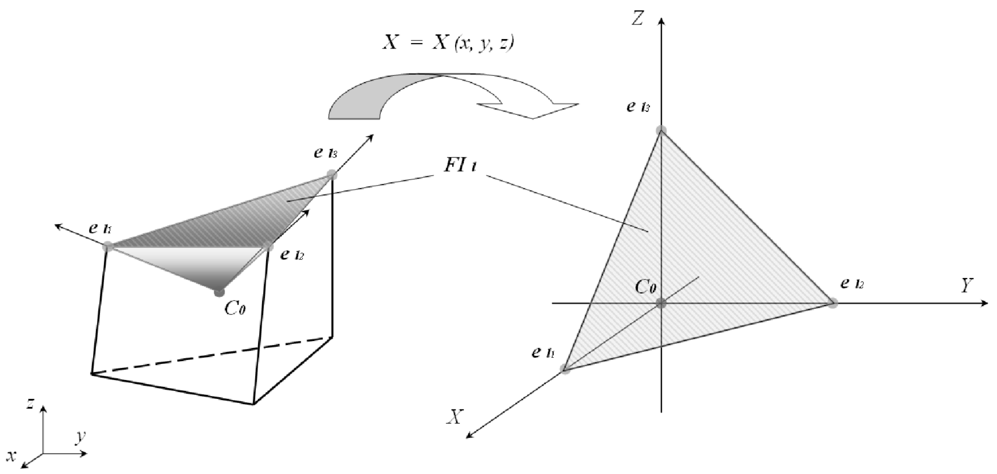


Figure 7: Mapping to the first octant $(+,+,+)$.

Mapping to the first octant In order to assess efficiently which cells have their centre lying in the sector, it is convenient to map the sector to the octant $(+,+,+)$ in a transformed space $\mathbf{X}=(X,Y,Z)$ so that only the cells whose centres have all positive coordinates in $\mathbf{X}$ can be selected. However, such a mapping is only possible if the directrix of the cone $\mathscr{C}_l$ encompassing the geometric sector is a triangle (see Fig. 7). Taking the centre of the targeted cell $C_0=(x_{C_0},y_{C_0},z_{C_0})$ and introducing $\mathcal{J}_Q$ as the Jacobian of the affine transformation that maps the octant $(+,+,+)$ to the sector delimited by $\mathscr{C}_l$, we can write:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \mathcal{J}_Q \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{3.34}$$

where the matrix $\mathcal{J}_Q$ is built from the coordinates of the three vertices of the triangle FI_l, the directrix of $\mathscr{C}_l$.

We extend this methodology to non-tetrahedral cells by splitting, into N_T triangles, all the internal faces FI_l of the mesh. The cells belonging to the sector S_l delimited by $\mathscr{C}_l$ belong to the union of sub-sectors S_{l_i} delimited by the cones $\mathscr{C}_{l_i}$ associated with the N_T triangles:

$$S_l = \bigcup_{i=1}^{N_T} S_{l_i}. \tag{3.35}$$

The sub-sectors S_{l_i} are therefore successively mapped to the octant $(+,+,+)$ and for each potential stencil cell E_k, we produce and test a condition of inclusion of its centre C_k (expressed in $\mathbf{X}^{(i)}$: the transformed space associated to S_{l_i}) in the various sub-sectors. Using the symbol $\sum$ to indicate the repeated use of the logical disjunction, our condition reads:

$$Cond_k = \sum_{i=1}^{N_T} (C_k \in S_{l_i}), \tag{3.36a}$$

$$= \sum_{i=1}^{N_T} \Big(\mathrm{pos}\big(X_{C_k}^{(i)}\big) \cdot \mathrm{pos}\big(Y_{C_k}^{(i)}\big) \cdot \mathrm{pos}\big(Z_{C_k}^{(i)}\big) \Big), \tag{3.36b}$$

where

$$\forall x \in \mathbb{R}: \ \mathrm{pos}(x) = \begin{cases} 1, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0. \end{cases} \tag{3.37}$$

The above test guarantees that cells whose centres belong to several sub-sectors are added just once to the list of sector members. It avoids the need for a cumbersome and time-consuming iterative test against all the cells already identified as sector members. In order to optimise the procedure, the internal faces are split into a minimal number of triangles, such that $N_T = 1$ for triangular faces (no split), $N_T = 2$ for quadrilateral faces and $N_T =$ *number of sides* for polygonal faces. As this split of the internal faces is performed in any case during the tetrahedralisation algorithm (see Section 3.1.2), the tetrahedral decomposition of the mesh is run prior to the generation of the one-sided stencils.

Selection of the stencil cells The cells whose centre belongs to a sector — the "sector members" — are added to the corresponding one-sided stencil if:

- They are not already part of another sectoral stencil of E_0. This ensures that the one-sided stencils remain disjoint when the cell centre of a potential stencil element lies on the boundary of the sector. Such a particular case occurs frequently on Cartesian meshes.
- There are no more than $(j_{\max} - 1)$ cells in this sector with a shorter centre-to-centre distance to E_0. This ensures the compactness of the stencil.

The strategy adopted to produce a fast algorithm is to divide a very large central stencil in as many sectors as appropriate. Therefore, a much bigger temporary central stencil $\mathcal{S}_U$ is produced (see Section 3.1.1), sorted according to the centre-to-centre distance and finally split into the appropriate number of sectoral stencils (see Fig. 8). Therefore, the search procedure starts with building a compact central stencil of N_U cells that should encompass a sufficient number of cells to produce all the one-sided stencils. N_U is defined by:

$$N_U = (N_{FI} + 1) \cdot j_{\max}. \tag{3.38}$$

The extra $j_{\max}$ cells resulting from the $(+1)$ in (3.38) provide a buffer of cells necessary to accommodate the selection of sectoral stencils near convoluted boundaries.

In order to remove the cumbersome and time-consuming test conditions related to the compactness and separation of the stencils, a dynamic list is initially identified to the stencil $\mathcal{S}_U$. Taking advantage of the fact that the cells in $\mathcal{S}_U$ are already sorted according

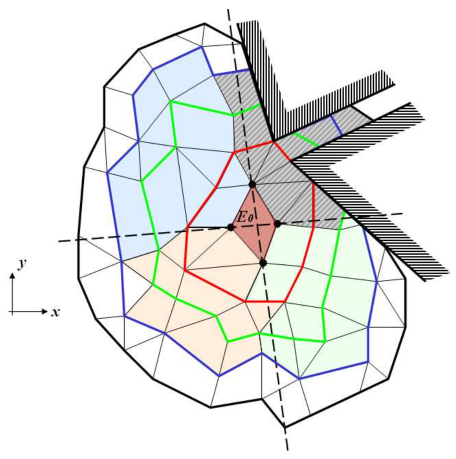


Figure 8: Thirteen-cells sectoral stencils of E_0 coloured by sector: the colored thick lines indicate the progression of the search; discarded stencil hatched in grey; domain boundaries hatched in black-as produced by the fast search procedure in 2D.

to their centre-to-centre distance to E_0, the membership of the cells in a given sector is tested sequentially for each cell in the list. This guarantees the compactness of the one-sided stencils. The procedure stops either when $j_{\max}$ cells have been added to the stencil or when all the cells in the list have been tested. The condition requiring disjoint stencils is ensured by removing the cells selected at each iteration. This fast search algorithm produces the best dataset for an accurate reconstruction of the solution, as the remaining stencils are very compact.

3.2.2 Determination of the oscillation indicator matrix

On a general unstructured mesh, the oscillation indicator matrix $\mathcal{B}$ is solution independent and therefore can be pre-processed. We have derived a computationally convenient expression for the elements of $\mathcal{B}$. Let us first rewrite Eq. (3.31) as:

$$B_{pq}=\sum_{\Lambda=1}^{r}\sum_{\alpha=0}^{\Lambda}\sum_{\beta=0}^{\Lambda-\alpha}\iiint_{E_0'}\frac{\partial^\Lambda}{\partial\xi^\alpha\partial\eta^\beta\partial\zeta^\gamma}\phi_p(\xi,\eta,\zeta)\cdot\frac{\partial^\Lambda}{\partial\xi^\alpha\partial\eta^\beta\partial\zeta^\gamma}\phi_q(\xi,\eta,\zeta)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \tag{3.39}$$

with $\Lambda=\alpha+\beta+\gamma$.

Simplified expression for B_{pq} The term B_{pq} is a triple sum of integrals over the targeted element E_0'. Each integral is calculated through Gauss Legendre quadratures over the N_T tetrahedral sub-elements T_l of E_0', resulting in the quadruple sum:

$$B_{pq}=\sum_{\Lambda=1}^{r}\sum_{\alpha=0}^{\Lambda}\sum_{\beta=0}^{\Lambda-\alpha}\sum_{l=1}^{N_T}\iiint_{T_l}\frac{\partial^\Lambda}{\partial\xi^\alpha\partial\eta^\beta\partial\zeta^\gamma}\phi_p\cdot\frac{\partial^\Lambda}{\partial\xi^\alpha\partial\eta^\beta\partial\zeta^\gamma}\phi_q\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta. \tag{3.40}$$

The integrands involve the product of the partial derivatives of order (α,β,γ), in (ξ,η,ζ), taken on two different polynomial basis functions: ϕ_p and ϕ_q. As the basis functions are constructed to satisfy the conservation condition (3.6), the expression for the basis functions ϕ_k is simply the sum of a monomial ψ_k with a constant $C_{k,i}$ (dependent on the cell E'_i and the monomial ψ_k, see (3.9)):

$$\phi_k = \psi_k + C_{k,i}. \tag{3.41}$$

Since the order of the partial derivative is greater than or equal to one, the constants disappear and we reduce each integrand $ID_{pq}^{\alpha,\beta,\gamma}$ to a product of two monomials:

$$B_{pq} = \sum_{\Lambda=1}^{r} \sum_{\alpha=0}^{\Lambda} \sum_{\beta=0}^{\Lambda-\alpha} \sum_{l=1}^{N_T} \iiint_{T_l} ID_{pq}^{\alpha,\beta,\gamma} \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta \tag{3.42a}$$

$$= \sum_{\Lambda=1}^{r} \sum_{\alpha=0}^{\Lambda} \sum_{\beta=0}^{\Lambda-\alpha} \sum_{l=1}^{N_T} \iiint_{T_l} \frac{\partial^\Lambda}{\partial\xi^\alpha \partial\eta^\beta \partial\zeta^\gamma} \psi_p \cdot \frac{\partial^\Lambda}{\partial\xi^\alpha \partial\eta^\beta \partial\zeta^\gamma} \psi_q \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta. \tag{3.42b}$$

Since a product of monomials is a monomial, the elements of $\mathcal{B}$ are multiple sums of integrals of monomials over E'_0. The integrations of the monomials of the highest degree D clearly occur for the basis functions of the highest degree when the order of the partial derivative is lowest: $\Lambda_{\min} = 1$. As a result, the highest degree D to be considered is expressed by:

$$D = 2 \cdot (r - \Lambda_{\min}) = 2r - 2. \tag{3.43}$$

The associated number of non-constant monomials of degree less than or equal to $(2r-2)$ is then:

$$K_{IS} = \frac{r(4r^2-1)}{3} - 1. \tag{3.44}$$

Derivation of the integrands of B_{pq} Similarly to the determination of the elements of $\mathcal{A}$, the calculation of the B_{pq} involve a combination of integrals of monomials. However, in the computations of $\mathcal{B}$, the degree of the monomials to be integrated reaches $(2r-2)$ and the integrations are only performed over the targeted cell E'_0. In order to perform the integration, we express the integrand $ID_{pq}^{\alpha,\beta,\gamma}$ as a monomial. Introducing $(A,B,C) \in [\![0,r]\!]^3$ such that $\{1 \le A+B+C \le r\}$, we can express the monomial $\psi_k(\xi,\eta,\zeta)$ by:

$$\psi_k(\xi,\eta,\zeta) = \xi^A \eta^B \zeta^C. \tag{3.45}$$

Applying to ψ_k the partial derivative of order (α,β,γ), in (ξ,η,ζ), leads to:

$$\frac{\partial^\Lambda}{\partial\xi^\alpha \partial\eta^\beta \partial\zeta^\gamma} (\psi_k) = K_D \cdot \xi^{(A-\alpha)} \eta^{(B-\beta)} \zeta^{(C-\gamma)}, \tag{3.46}$$

where

$$K_D = \begin{cases} \dfrac{A!}{(A-\alpha)!} \cdot \dfrac{B!}{(B-\beta)!} \cdot \dfrac{C!}{(C-\gamma)!}, & \text{if } (A \ge \alpha) \cap (B \ge \beta) \cap (C \ge \gamma), \\ 0, & \text{otherwise.} \end{cases} \tag{3.47}$$

K_D can be expressed in a more convenient manner by using the function "pos" introduced in Section 3.2.1:

$$K_D = \text{pos}(A-\alpha)\cdot\text{pos}(B-\beta)\cdot\text{pos}(C-\gamma)\cdot\frac{A!}{(A-\alpha)!}\cdot\frac{B!}{(B-\beta)!}\cdot\frac{C!}{(C-\gamma)!}. \tag{3.48}$$

Then, introducing: $(A_1,B_1,C_1,A_2,B_2,C_2)\in[\![0,r]\!]^6$ such that $\{1\le A_i+B_i+C_i\le r,\, i=1,2\}$ and making use of (3.48), we express the integrand $ID_{pq}^{\alpha,\beta,\gamma}$ as a monomial:

$$ID_{pq}^{\alpha,\beta,\gamma} = K_E\cdot\xi^{(A_1+A_2-2\alpha)}\eta^{(B_1+B_2-2\beta)}\zeta^{(C_1+C_2-2\gamma)} \tag{3.49}$$

with

$$\begin{aligned} K_E =& \text{pos}(A_1-\alpha)\cdot\text{pos}(B_1-\beta)\cdot\text{pos}(C_1-\gamma)\cdot\frac{A_1!}{(A_1-\alpha)!}\cdot\frac{B_1!}{(B_1-\beta)!}\cdot\frac{C_1!}{(C_1-\gamma)!} \\ &\times\text{pos}(A_2-\alpha)\cdot\text{pos}(B_2-\beta)\cdot\text{pos}(C_2-\gamma)\cdot\frac{A_2!}{(A_2-\alpha)!}\cdot\frac{B_2!}{(B_2-\beta)!}\cdot\frac{C_2!}{(C_2-\gamma)!}. \end{aligned} \tag{3.50}$$

Efficient computation of $\boldsymbol{\mathcal{B}}$ By inserting (3.49) in (3.42) we derive the final formulation for the elements B_{pq}:

$$B_{pq} = \sum_{\Lambda=1}^{r}\sum_{\alpha=0}^{\Lambda}\sum_{\beta=0}^{\Lambda-\alpha}\sum_{l=1}^{N_T}\iiint_{T_l} K_E\cdot\xi^{(A_1+A_2-2\alpha)}\eta^{(B_1+B_2-2\beta)}\zeta^{(C_1+C_2-2\gamma)}\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta. \tag{3.51}$$

It may be noted that:

- The K integrals of monomials of degrees up to r are readily available from the algorithm for computing the reconstruction matrix $\mathcal{A}$.
- The calculation of the different matrix elements B_{pq} generally involves many of the same monomial integrations over the targeted cell E_0'.

Therefore we choose to store the list of K_{IS} integrals of monomials over E_0' for each cell of the mesh. The elements of $\mathcal{B}$ are then efficiently computed from this list of integrals by applying the formula (3.51). It is worth noting that only the integrals of monomials of degree higher than r have to be calculated, as the first K integrals of the list have already been computed (see Section 3.1.2).

3.2.3 Calculation of the modified degrees of freedom

Since the modified degrees of freedom $\tilde{a}_k$ are a function of both the WENO weights w_m and the degrees of freedom $a_k^{(m)}$ associated to the stencil $\mathcal{S}_m$ (see (3.32)), the calculation of the $\tilde{a}_k$ is performed at runtime. This computation takes the following steps:

1. The generation of the vector of data $\mathbf{b}_m$ for each stencil $\mathcal{S}_m$ of each cell of the mesh. The components $b_j^{(m)}$ are computed from the cell averages of the solution $\overline{u}_j^{(m)}(t)$ in the cells of $\mathcal{S}_m$ at a given time t:

$$b_j^{(m)}=\overline{u}_j^{(m)}(t)-\overline{u}_0^{(m)}(t), \quad j=1,\cdots,j_{\max}, \quad m=0,\cdots,N_S. \tag{3.52}$$

2. The determination of the (N_S+1) vectors of degrees of freedom $\mathbf{a}_m$ from the effective pseudo-inverses $(\mathcal{A}_\tau^\dagger)^{(m)}$ of the reconstruction matrices and the vectors of data $\mathbf{b}_m$:

$$\mathbf{a}_m=(\mathcal{A}_\tau^\dagger)^{(m)}\mathbf{b}_m, \quad m=0,\cdots,N_S. \tag{3.53}$$

3. For each stencil $\mathcal{S}_m$, calculation of the smoothness indicator IS_m from the oscillation indicator matrix $\mathcal{B}$ and the vector of degrees of freedom $\mathbf{a}_m$:

$$IS_m=\mathbf{a}_m^T\cdot(\mathcal{B}\mathbf{a}_m), \quad m=0,\cdots,N_S. \tag{3.54}$$

4. Calculation of the WENO weights w_m from the smoothness indicators IS_m (see (3.30) and (3.31)).

5. For each cell of the mesh, calculation of the vector of modified degrees of freedom $\tilde{\mathbf{a}}$ from the (N_S+1) vector of degrees of freedom $\mathbf{a}_m$ and the WENO weights w_m:

$$\tilde{\mathbf{a}}=\sum_{m=0}^{N_S} w_m\mathbf{a}_m. \tag{3.55}$$

3.3 Determination of the numerical flux

With the details of the polynomial reconstruction procedure in mind, let us derive the flux calculation for the system of Eqs. (2.1). This is a necessary step, since some of the simplifications relevant to tetrahedral grids [7] and 2D mixed-element meshes [36] do not apply here. Recalling the equations derived in Section 2, we can express (2.1) in the following finite volume form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{U}}_i+\frac{1}{|E_i|}\sum_{l=1}^{Li}\iint_{F_l}\mathbf{A}_{n_l}(\mathbf{U}^-,\mathbf{U}^+)\mathrm{d}(F_l)=\mathbf{0}. \tag{3.56}$$

Here $\mathbf{A}_{n_l}(\mathbf{U}^-,\mathbf{U}^+)$ represents the numerical flux in the direction normal to the face F_l ($\mathbf{n}_l$ being the outward unit vector normal to F_l) as a function of the reconstructed solution on either side of F_l: $\mathbf{U}^\pm$. The superscripts "$-$" and "$+$" refer to the spatial limit respectively on the inside and the outside of the targeted cell E_i with respect to its face F_l. In particular, $\mathbf{U}^-$ represents the reconstructed solution calculated on F_l using the polynomial interpolation of the solution in E_i and $\mathbf{U}^+$ represents the reconstructed solution calculated on F_l using the polynomial interpolation of the solution in the neighbouring cell E_{j_l}.

3.3.1 Hyperbolic systems of linear PDE

The exact Riemann solver As in [7], we choose to use the exact Riemann solver to express the numerical flux between E_i and E_{j_l}. In order to express this flux, we first introduce the Jacobian matrix of the system in the normal direction $\mathcal{J}_{N_l}$. Noting that $\mathcal{J}_X$ is the Jacobian of the vector of fluxes in the x direction $\mathbf{F}(\mathbf{U})$ we have (see [37]):

$$\frac{\partial}{\partial x}\mathbf{F}(\mathbf{U})=\frac{\partial \mathbf{F}}{\partial \mathbf{U}}\cdot\frac{\partial \mathbf{U}}{\partial x}=\mathcal{J}_X\cdot\frac{\partial \mathbf{U}}{\partial x}. \tag{3.57}$$

With $\mathcal{J}_Y$ and $\mathcal{J}_Z$ as the Jacobian matrices of the vectors of fluxes $\mathbf{G}(\mathbf{U})$ and $\mathbf{H}(\mathbf{U})$, we can re-write (2.1) as:

$$\frac{\partial}{\partial t}\mathbf{U}+\mathcal{J}_X\cdot\frac{\partial}{\partial x}\mathbf{U}+\mathcal{J}_Y\cdot\frac{\partial}{\partial y}\mathbf{U}+\mathcal{J}_Z\cdot\frac{\partial}{\partial z}\mathbf{U}=\mathbf{0}. \tag{3.58}$$

Recalling $\mathbf{n}_l=\left(n_{lx},n_{ly},n_{lz}\right)$ from Section 2, $\mathcal{J}_{N_l}$ is expressed as:

$$\mathcal{J}_{N_l}=\mathcal{J}_X n_{lx}+\mathcal{J}_Y n_{ly}+\mathcal{J}_Z n_{lz}. \tag{3.59}$$

In order to express the numerical flux $\mathbf{A}_{n_l}$, we introduce (after [37]) the diagonal matrix of eigenvalues $\Lambda_{N_l}=\mathrm{diag}(\lambda_{l1},\lambda_{l2},\cdots,\lambda_{lP})$ of $\mathcal{J}_{N_l}$ and the left and right eigenvector matrices: $\mathcal{L}_{N_l}$ and $\mathcal{R}_{N_l}$, so that $\mathcal{J}_{N_l}=\mathcal{L}_{N_l}\cdot\Lambda_{N_l}\cdot\mathcal{R}_{N_l}$. Defining the matrix absolute value operator: $|\mathcal{J}_{N_l}|=\mathcal{L}_{N_l}\cdot|\Lambda_{N_l}|\cdot\mathcal{R}_{N_l}$ with $|\Lambda_{N_l}|=\mathrm{diag}(|\lambda_{l1}|,|\lambda_{l2}|,\cdots,|\lambda_{lP}|)$, the numerical flux — associated with the exact Riemann solver — between E_i and E_{j_l} becomes [7,37]:

$$\mathbf{A}_{n_l}(\mathbf{U}^-,\mathbf{U}^+)=\frac{1}{2}\left((\mathcal{J}_{N_l}+|\mathcal{J}_{N_l}|)\mathbf{U}^-+(\mathcal{J}_{N_l}-|\mathcal{J}_{N_l}|)\mathbf{U}^+\right). \tag{3.60}$$

Computation of the numerical flux Introducing the expression (3.60) in the finite volume formulation (3.56) leads to:

$$\mathbf{0}=\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{U}}_i+\frac{1}{|E_i|}\sum_{l=1}^{Li}\frac{1}{2}(\mathcal{J}_{N_l}+|\mathcal{J}_{N_l}|)\iint_{F_l}\mathbf{U}^-\mathrm{d}(F_l)+\frac{1}{|E_i|}\sum_{l=1}^{Li}\frac{1}{2}(\mathcal{J}_{N_l}-|\mathcal{J}_{N_l}|)\iint_{F_l}\mathbf{U}^+\mathrm{d}(F_l). \tag{3.61}$$

Since the reconstruction procedure relies on a mapping to avoid introducing cumbersome scaling factors, the reconstruction is performed on a reference space $\boldsymbol{\xi}$. As a result, the polynomial representing the solution $\mathbf{U}$ is only known as a function of $\boldsymbol{\xi}$. Therefore, we relate the integrals in (3.61) to their counterparts over the faces F'_l of E'_i in the reference spaces $\boldsymbol{\xi}^-$ and $\boldsymbol{\xi}^+$ associated respectively to E'_i and E'_{j_l}:

$$\iint_{F_l}\mathbf{U}^\pm\mathrm{d}(F_l)=\frac{|F_l|}{|F'_l|}\iint_{F'_l}\mathbf{U}^\pm(\boldsymbol{\xi}^\pm,t)\mathrm{d}(F'_l). \tag{3.62}$$

As the transformation from the physical space to the reference space is affine, the ratio of square roots of the Gram determinants involved (see [22]) is constant and may be taken out of the integrals.

In the case of a vector variable $\mathbf{U}$, exactly the same basis functions ϕ_k are used for the polynomial reconstruction of all the components u_p of $\mathbf{U}$, since the functions ϕ_k are only dependent on the mesh. As each component of the vector variable can be considered as a scalar variable, a set of k modified degrees of freedom $\tilde{a}_k$ is calculated at each time step for every scalar variable u_p. Therefore there are as many modified degrees of freedom associated with ϕ_k as there are components in $\mathbf{U}$, and these are gathered in the vector $\tilde{\mathbf{a}}_k$.

Recalling the equation for the polynomial reconstruction of a scalar variable (3.32), the formulae for $\mathbf{U}^-(\boldsymbol{\xi}^-,t)$ and $\mathbf{U}^+(\boldsymbol{\xi}^+,t)$, at time $t=n\Delta t$, are:

$$\begin{cases} \mathbf{U}^-(\boldsymbol{\xi}^-,t)=(\overline{\mathbf{U}}_i)^{(n)}+\sum\limits_{k=1}^{K}(\tilde{\mathbf{a}}_k)_i^{(n)}(\phi_k)_i(\boldsymbol{\xi}^-), \\ \mathbf{U}^+(\boldsymbol{\xi}^+,t)=(\overline{\mathbf{U}}_{j_l})^{(n)}+\sum\limits_{k=1}^{K}(\tilde{\mathbf{a}}_k)_{j_l}^{(n)}(\phi_k)_{j_l}(\boldsymbol{\xi}^+). \end{cases} \tag{3.63}$$

Replacing $\mathbf{U}^-(\boldsymbol{\xi}^-,t)$ and $\mathbf{U}^+(\boldsymbol{\xi}^+,t)$ in (3.62) by their expressions in (3.63) leads to:

$$\begin{cases} \iint\limits_{F_l}\mathbf{U}^-\mathrm{d}(F_l)=|F_l|(\overline{\mathbf{U}}_i)^{(n)}+\frac{|F_l|}{|F_l'|}\sum\limits_{k=1}^{K}\Big((\tilde{\mathbf{a}}_k)_i^{(n)}\iint\limits_{F_l'}(\phi_k)_i(\boldsymbol{\xi}^-)\mathrm{d}(F_l')\Big), \\ \iint\limits_{F_l}\mathbf{U}^+\mathrm{d}(F_l)=|F_l|(\overline{\mathbf{U}}_{j_l})^{(n)}+\frac{|F_l|}{|F_l'|}\sum\limits_{k=1}^{K}\Big((\tilde{\mathbf{a}}_k)_{j_l}^{(n)}\iint\limits_{F_l'}(\phi_k)_{j_l}(\boldsymbol{\xi}^+)\mathrm{d}(F_l')\Big). \end{cases} \tag{3.64}$$

In (3.64), the integrals of the basis functions $(\phi_k)_i(\boldsymbol{\xi}^-)$ and $(\phi_k)_{j_l}(\boldsymbol{\xi}^+)$ are independent of the solution $\mathbf{U}$. Hence the integrals of the k basis functions $(\phi_k)_i(\boldsymbol{\xi}^-)$ are pre-computed for all the faces F_l' of all the transformed elements E_i'. As a result, the integrals of $(\phi_k)_i(\boldsymbol{\xi}^-)$ and $(\phi_k)_{j_l}(\boldsymbol{\xi}^+)$ are readily available for the runtime calculation of the inter-cell flux between any neighbouring cells.

Surface integrals of the basis functions Recalling that the basis functions ϕ_k are constructed from the monomials ψ_k (see (3.7)), the expression for the kth basis function of the element E_i: $(\phi_k)_i$ is:

$$(\phi_k)_i=\psi_k-\frac{1}{|E_i'|}\iiint\limits_{E_i'}\psi_k\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta. \tag{3.65}$$

Therefore, the integral of the basis function $(\phi_k)_i$ is expressed as:

$$\iint\limits_{F_l'}(\phi_k)_i(\boldsymbol{\xi}^-)\mathrm{d}(F_l')=\iint\limits_{F_l'}\psi_k(\boldsymbol{\xi}^-)\mathrm{d}(F_l')-\frac{|F_l'|}{|E_i'|}\iiint\limits_{E_i'}\psi_k\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta. \tag{3.66}$$

In (3.66), the triple integral of the monomials ψ_k over the volume E_i' is calculated during the linear reconstruction (see Section 3.1.2) and is therefore readily available. As in Section 3.1.2, in order to maximise the efficiency of the method, we choose to calculate the surface integral of the monomial ψ_k over the face F_l' with a Gaussian quadrature. Since the affine transformation of a convex polyhedral does not generally result in the standard domain required by Gauss-Legendre quadratures, we take advantage of the tetrahedralisation of the mesh and, for each face F'_l, sum the Gaussian quadratures produced for the N_{T_l} triangles T'_{lm} constituting F'_l:

$$\iint_{F'_l} \psi_k(\boldsymbol{\xi}^-)\mathrm{d}(F'_l) = \sum_{m=1}^{N_{T_l}} \iint_{T'_{lm}} \psi_k(\boldsymbol{\xi}^-)\mathrm{d}(T'_{lm}). \tag{3.67}$$

3.3.2 Hyperbolic systems of non-linear PDE

In the general case, the flux $\mathbf{A}_{n_l}(\mathbf{U}^-,\mathbf{U}^+)$ (see (3.56)) varies along the face F_l. Consequently, as we perform the integration of the flux with a Gauss-Legendre quadrature, $\mathbf{A}_{n_l}(\mathbf{U}^-,\mathbf{U}^+)$ is evaluated at each Gauss point $\mathbf{x}_\beta$ of each triangle T_{lm} constituting the face F_l.

The solution being reconstructed in the mapped space, the integral of the flux are calculated over the transformed face F'_l. Re-writing (3.56) with the integral expressed over F'_l leads to:

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{U}}_i + \frac{1}{|E_i|}\sum_{l=1}^{Li}\frac{|F_l|}{|F'_l|}\iint_{F'_l}\mathbf{A}_{n_l}(\mathbf{U}^-,\mathbf{U}^+)\mathrm{d}(F'_l) = \mathbf{0}. \tag{3.68}$$

For each point of the Gaussian quadrature, the reconstructed solutions on either side of the face, $\mathbf{U}^-_\beta$ and $\mathbf{U}^+_\beta$, are evaluated at the mapped Gauss point in their respective transformed space: $\boldsymbol{\xi}^-_\beta$ and $\boldsymbol{\xi}^+_\beta$. Introducing the weights of the Gaussian quadrature k_β and the Riemann solver $\widehat{\mathbf{A}}$, the integral of the flux can be expressed as follows:

$$\iint_{F'_l}\mathbf{A}_{n_l}(\mathbf{U}^-,\mathbf{U}^+)\mathrm{d}(F'_l) = \sum_{m=1}^{N_{T_l}}\iint_{T'_{lm}}\mathbf{A}_{n_l}(\mathbf{U}^-,\mathbf{U}^+)\mathrm{d}(F'_l) = \sum_{m=1}^{N_{T_l}}\sum_{\beta=1}^{N_\beta} k_\beta\widehat{\mathbf{A}}(\mathbf{U}^-_\beta,\mathbf{U}^+_\beta). \tag{3.69}$$

As suggested by the equations above, the integrals of the basis functions can no longer be pre-processed when solving non-linear PDE. However, the values of the basis functions on each Gauss point can be pre-computed to save computational time.

4 Application to the level set equation

4.1 Finite volume formulation of the level set equation

The WENO scheme presented in the previous section is to be used in the transport of the level set scalar function φ for application to multiphase flows. Introducing the velocity $\mathbf{u} = (u,v,w)$, the transport equation for φ may be expressed as:

$$\frac{\partial\varphi}{\partial t} + \mathbf{u}\cdot\nabla\varphi = 0. \tag{4.1}$$

Assuming an incompressible flow we have $\nabla u = 0$, so that (4.1) can be re-written as a hyperbolic conservation law:

$$\frac{\partial \varphi}{\partial t} + \nabla \cdot (\varphi \mathbf{u}) = 0. \tag{4.2}$$

In terms of the coordinates (x,y,z), Eq. (4.2) becomes

$$\frac{\partial \varphi}{\partial t} + \frac{\partial}{\partial x}(u\varphi) + \frac{\partial}{\partial y}(v\varphi) + \frac{\partial}{\partial z}(w\varphi) = 0. \tag{4.3}$$

Comparing Eq. (4.3) to Eq. (2.1) provides the equalities:

$$\begin{cases} F(\varphi) = u\varphi, \\ G(\varphi) = v\varphi, \\ H(\varphi) = w\varphi, \end{cases} \Longrightarrow \begin{cases} \mathbf{A} = (F,G,H) = (u\varphi, v\varphi, w\varphi), \\ A_{n_l}(\varphi^-, \varphi^+) = \mathbf{A} \cdot \mathbf{n}_l. \end{cases} \tag{4.4}$$

In (4.4), φ^- represents the reconstructed level set function calculated on F_l using polynomial interpolation of the solution in E_i, while φ^+ represents the reconstructed level set function calculated on F_l using polynomial interpolation of the solution in the neighbouring cell E_{j_l}.

From (4.4) and (3.56), we express the transport equation for the level set in the finite volume form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{\varphi}_i + \frac{1}{|E_i|}\sum_{l=1}^{Li} \iint_{F_l} A_{n_l}(\varphi^-, \varphi^+) \mathrm{d}(F_l) = 0. \tag{4.5}$$

The normal component of the flux A_{n_l} for the level set equation can be obtained by expressing (4.1) in terms of the coordinates to give:

$$\frac{\partial \varphi}{\partial t} + u\frac{\partial \varphi}{\partial x} + v\frac{\partial \varphi}{\partial y} + w\frac{\partial \varphi}{\partial z} = 0. \tag{4.6}$$

Identifying Eq. (4.6) with Eq. (3.58) provides the equalities:

$$\begin{cases} J_X = u, \\ J_Y = v, \\ J_Z = w, \end{cases} \Longrightarrow \quad J_{N_l} = un_{lx} + vn_{ly} + wn_{lz} = \mathbf{u} \cdot \mathbf{n}_l = u_{n_l}. \tag{4.7}$$

Using (4.7) in (3.60), the normal component of the flux may be expressed as:

$$A_{n_l}(\varphi^-, \varphi^+) = \frac{1}{2}\big((u_{n_l} + |u_{n_l}|)\varphi^- + (u_{n_l} - |u_{n_l}|)\varphi^+\big). \tag{4.8}$$

4.2 The Riemann problem for the level set equation

Simple manipulations of (4.6) demonstrates its rotational invariance (see Chapter 3 of [37]) according to:

$$\mathbf{A}\cdot\mathbf{n}_l=(F,G,H)\cdot\mathbf{n}_l=n_{lx}F+n_{ly}G+n_{lz}H=\widehat{F}, \tag{4.9}$$

where $\widehat{F}$ is the flux vector expressed in the direction n_l, the first axis of the rotated Cartesian frame (n_l,s_l,t_l). The expression for $\widehat{F}$ is:

$$\widehat{F}=(\mathbf{u}\cdot\mathbf{n}_l)\varphi=u_{n_l}\varphi. \tag{4.10}$$

In our three dimensional space, the rotated Cartesian frame is obtained by applying successively two rotations around the angles $\theta_{l1}\in[0,2\pi]$ and $\theta_{l2}\in[0,\pi]$. The Eq. (4.9) is valid $\forall(\theta_{l1},\theta_{l2})\in[0,2\pi]\times[0,\pi]$ (see Fig. 9 for an illustration in 2D). Since we are dealing with a single scalar equation, the rotation matrix $\mathcal{T}_l(\theta_{l1},\theta_{l2})$ (see [3, 37]) and its inverse both reduce to the scalar unity. From (4.4) and (4.5), the finite volume formulation of (4.6) can be expressed as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{\varphi}_i+\frac{1}{|E_i|}\sum_{l=1}^{Li}\iint_{F_l}(F,G,H)\cdot\mathbf{n}_l\mathrm{d}(F_l)=\mathbf{0}. \tag{4.11}$$

Using the rotational invariance of (4.6) we re-write (4.11) in the rotated Cartesian frame (n_l,s_l,t_l), where the direction n_l is normal to the inter-cell boundary F_l. Noting $\widehat{F}$, $\widehat{G}$ and $\widehat{H}$, the flux vectors respectively in the directions n_l, s_l and t_l, we have:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\overline{\varphi}_i&=-\frac{1}{|E_i|}\sum_{l=1}^{Li}\iint_{F_l}(\widehat{F},\widehat{G},\widehat{H})\cdot\mathbf{n}_l\mathrm{d}(F_l)=-\frac{1}{|E_i|}\sum_{l=1}^{Li}\iint_{F_l}(\widehat{F},\widehat{G},\widehat{H})\cdot(1,0,0)^T\mathrm{d}(F_l)\\&=-\frac{1}{|E_i|}\sum_{l=1}^{Li}\iint_{F_l}\widehat{F}\mathrm{d}(F_l).\end{aligned} \tag{4.12}$$

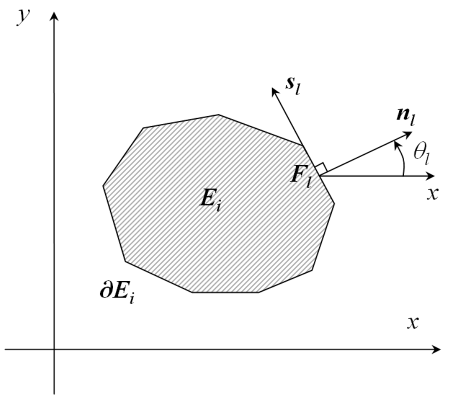


Figure 9: Rotated Cartesian frame in 2D: (n_l,s_l) — the first axis n_l is orthogonal to the arbitrary boundary ∂E_i of the control volume E_i.

Therefore, as mentioned in [37], the numerical fluxes across the L_i faces F_l of the element E_i result from the equation written in the rotated frame (n_l,s_l,t_l). Consequently, the flux across F_l is given by the one-dimensional equation:

$$\frac{\partial\varphi}{\partial t}+\frac{\partial\widehat{F}}{\partial n_l}=0. \tag{4.13}$$

Eq. (4.13) leads to the Riemann problem:

$$\left.\begin{array}{ll}\text{PDE:} & \dfrac{\partial\varphi}{\partial t}+u_{n_l}\dfrac{\partial\varphi}{\partial n_l}=0,\\ \text{IC:} & \varphi(n_l,0)=\varphi_0(n_l)=\begin{cases}\varphi^-, & \text{if } n_l<0,\\ \varphi^+, & \text{if } n_l>0.\end{cases}\end{array}\right\} \tag{4.14}$$

Eq. (4.14) admits an exact solution:

$$\varphi(n_l,t)=\varphi_0(n_l-u_{n_l}t)=\begin{cases}\varphi^-, & \text{if } n_l-u_{n_l}t<0,\\ \varphi^+, & \text{if } n_l-u_{n_l}t>0,\end{cases} \tag{4.15}$$

so that the flux A_{n_l} (see (4.4)) across F_l (i.e., at $n_l=0$ with $t>0$), reads:

$$A_{n_l}(\varphi^-,\varphi^+)=\begin{cases}u_{n_l}\varphi^-, & \text{if } u_{n_l}>0,\\ u_{n_l}\varphi^+, & \text{if } u_{n_l}<0.\end{cases} \tag{4.16}$$

The formulation of the flux in (4.16) is equivalent to the expression for $A_{n_l}(\varphi^-,\varphi^+)$ given in (4.8). Thus, the finite volume formulation of the level set equation on 3D unstructured grids has been related to its associated Riemann problem.

5 Application to the Burgers′ equation

5.1 Finite volume formulation of the Burgers′ equation

In this section we extend the application of the WENO scheme to the solution of the Burgers′ equation. Considering the variable φ and introducing the vector $\mathbf{v}=(a,b,c)$ fixed in $\mathbb{R}^d$, the Burgers′ equation for φ may be expressed as:

$$\frac{\partial\varphi}{\partial t}+\mathbf{v}\cdot\nabla\Big(\frac{\varphi^2}{2}\Big)=0. \tag{5.1}$$

As $\mathbf{v}$ is fixed in $\mathbb{R}^d$, (5.1) can be re-written as a hyperbolic conservation law:

$$\frac{\partial\varphi}{\partial t}+\nabla\cdot\Big(\frac{\varphi^2}{2}\mathbf{v}\Big)=0. \tag{5.2}$$

In terms of the coordinates (x,y,z), (5.2) becomes

$$\frac{\partial\varphi}{\partial t}+\frac{\partial}{\partial x}\Big(a\frac{\varphi^2}{2}\Big)+\frac{\partial}{\partial y}\Big(b\frac{\varphi^2}{2}\Big)+\frac{\partial}{\partial z}\Big(c\frac{\varphi^2}{2}\Big)=0. \tag{5.3}$$

Comparing Eq. (5.3) to Eq. (2.1) provides the equalities:

$$\left\{\begin{array}{l} F(\varphi)=a\dfrac{\varphi^2}{2}, \\ G(\varphi)=b\dfrac{\varphi^2}{2}, \\ H(\varphi)=c\dfrac{\varphi^2}{2}, \end{array}\right. \Longrightarrow \left\{\begin{array}{l} \mathbf{A}=(F,G,H)=\Big(a\dfrac{\varphi^2}{2},b\dfrac{\varphi^2}{2},c\dfrac{\varphi^2}{2}\Big), \\ A_{n_l}(\varphi^-,\varphi^+)=\mathbf{A}\cdot\mathbf{n}_l. \end{array}\right. \tag{5.4}$$

The Burgers' equation can then be expressed in the following finite volume form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{\varphi}_i+\frac{1}{|E_i|}\sum_{l=1}^{Li}\iint_{F_l}A_{n_l}(\varphi^-,\varphi^+)\mathrm{d}(F_l)=0. \tag{5.5}$$

5.2 The Riemann problem for the Burgers' equation

As in Section 4, simple manipulations of (5.3) demonstrates its rotational invariance according to:

$$\mathbf{A}\cdot\mathbf{n}_l=(F,G,H)\cdot\mathbf{n}_l=n_{lx}F+n_{ly}G+n_{lz}H=\widehat{F}, \tag{5.6}$$

where $\widehat{F}$ is the flux vector expressed in the direction n_l, the first axis of the rotated Cartesian frame (n_l,s_l,t_l). The expression for $\widehat{F}$ is:

$$\widehat{F}=(\mathbf{v}\cdot\mathbf{n}_l)\frac{\varphi^2}{2}=v_{n_l}\frac{\varphi^2}{2}. \tag{5.7}$$

Therefore, the flux across F_l is given by the one-dimensional equation:

$$\frac{\partial\varphi}{\partial t}+\frac{\partial\widehat{F}}{\partial n_l}=0. \tag{5.8}$$

Eq. (5.8) leads to the Riemann problem:

$$\left.\begin{array}{ll} \text{PDE:} & \dfrac{\partial\varphi}{\partial t}+\dfrac{\partial}{\partial n_l}\Big(v_{n_l}\dfrac{\varphi^2}{2}\Big)=0, \\ \text{IC:} & \varphi(n_l,0)=\varphi_0(n_l)=\left\{\begin{array}{ll}\varphi^-, & \text{if } n_l<0, \\ \varphi^+, & \text{if } n_l>0.\end{array}\right. \end{array}\right\} \tag{5.9}$$

For the PDE (5.9), the characteristic speed $\lambda(\varphi)$ is given by:

$$\lambda(\varphi)=\frac{\mathrm{d}\widehat{F}}{\mathrm{d}\varphi}=v_{n_l}\varphi. \tag{5.10}$$

Eq. (5.9) admits an exact solution:

$$\left.\begin{array}{ll}
\text{If } \lambda(\varphi^-)>\lambda(\varphi^+): & \varphi(n_l,t)=\begin{cases}\varphi^-, & \text{if } n_l-St<0,\\ \varphi^+, & \text{if } n_l-St>0;\end{cases}\\
\quad\text{with: } S=\dfrac{\Delta\widehat{F}}{\Delta\varphi}=\dfrac{v_{n_l}}{2}(\varphi^-+\varphi^+) & \\
\text{If } \lambda(\varphi^-)\leq\lambda(\varphi^+): & \begin{cases}\varphi(n_l,t)=\varphi^-, & \text{if } \dfrac{n_l}{t}\leq\lambda(\varphi^-),\\ \lambda(\varphi)=\dfrac{n_l}{t}, & \text{if } \lambda(\varphi^-)<\dfrac{n_l}{t}<\lambda(\varphi^+),\\ \varphi(n_l,t)=\varphi^+, & \text{if } \dfrac{n_l}{t}\geq\lambda(\varphi^+),\end{cases}
\end{array}\right\} \tag{5.11}$$

so that the flux A_{n_l} (see (5.4)) across F_l (i.e., at $n_l=0$ with $t>0$), reads:

$$\left.\begin{array}{ll}
\text{If } \lambda(\varphi^-)>\lambda(\varphi^+): & A_{n_l}(\varphi^-,\varphi^+)=\begin{cases}v_{n_l}\dfrac{(\varphi^-)^2}{2}, & \text{if } S>0,\\ v_{n_l}\dfrac{(\varphi^+)^2}{2}, & \text{if } S<0;\end{cases}\\
\quad\text{with: } S=\dfrac{v_{n_l}}{2}(\varphi^-+\varphi^+) & \\
\text{If } \lambda(\varphi^-)\leq\lambda(\varphi^+): & A_{n_l}(\varphi^-,\varphi^+)=\begin{cases}v_{n_l}\dfrac{(\varphi^-)^2}{2}, & \text{if } 0\leq v_{n_l}\varphi^-,\\ 0, & \text{if } v_{n_l}\varphi^-<0<v_{n_l}\varphi^+,\\ v_{n_l}\frac{(\varphi^+)^2}{2}, & \text{if } 0\geq v_{n_l}\varphi^+.\end{cases}
\end{array}\right\} \tag{5.12}$$

6 Results

The approach described above was implemented in parallel and in C++ using the framework provided by the open source CFD toolkit OpenFOAM. In order to reach the appropriate level of accuracy in parallel, the size of the halo surrounding each processor's subdomain varies with the order of the polynomial reconstruction.

In all the simulations presented, the time discretisation is performed with a third order Runge-Kutta scheme [32]. As we test WENO schemes of different order, let us introduce WENO*r*, the WENO scheme based on a polynomial reconstruction of order *r*.

6.1 Level set test cases

In all of the test cases considered in this section, we are advecting the level set scalar function φ according to Eq. (4.1) only. No re-distancing is applied as we are concerned mainly with assessing the performance of the WENO scheme.

6.1.1 Two-dimensional test cases

Two-dimensional meshes For these simulations, the mesh is made up of a single layer of three dimensional elements: wedges for the "triangular" mesh, hexahedra for the Cartesian mesh and both wedges and hexahedra for the hybrid mesh. As a result, the mesh is in fact three-dimensional as the code employed manages only 3D elements.

A front view of the three types of mesh used is given in Fig. 10 for the resolution $L/64$, with L being the length of the domain. Each two-dimensional mesh is shown on the left with a section of the corresponding three-dimensional mesh on the right. Three different resolutions have been considered: $L/64$, $L/128$ and $L/256$. The unstructured meshes have been generated so that — for a given resolution — the sides of the elements have the same length. As a result, the mesh density is higher for the "triangular" mesh and in the triangular region of the hybrid mesh.

Translation of a slotted disk This test assesses the performance of the scheme in advecting a complicated shape in a solid body rotation field. This problem, first set by Zalesak [40], is particularly difficult as it involves sharp corners and a thin slot within a solid geometrical shape. Depending on the quality of the scheme, this latter feature may disappear resulting in a modified topology. The computational domain is a square box delimited by the points $(-1;-1)$ and $(1;1)$. The shape is made of a disk of diameter 0.6, centred on $(0;0.5)$ from which a vertical rectangle of 0.1×0.5 is subtracted (see Fig. 11). The slotted disk is then translated along a circular trajectory ($C_{traj}=(0;0)$, $D_{traj}=1$). The simulation settings for this problem are given in Fig. 11. The end of the simulation is reached after one revolution for $t=1$.

The results for the advection of the slotted disk are given in Fig. 13. This figure provides the solution obtained with the WENO3 scheme after one revolution for three different resolutions of $L/64$ (red line), $L/128$ (green line) and $L/256$ (blue line) on three types of grids: Cartesian, triangular and hybrid (top to bottom). The results illustrate the convergence of the solution under grid refinement regardless of the type of mesh. Indeed, for all test cases, the level set contour for the resolution $L/256$ is closer to the initial contour in both the smooth circular part of the disk and in the region of the slot where both the width of the slot and the sharp corners are well captured.

The results are comparable across mesh types and, as intended, no significant degradation was observed when running the test case on general unstructured grids. On the triangular grid, the accuracy of the scheme is slightly better as this mesh is characterised by a higher density of elements for the same edge length.

Disk in a deformation field This test assesses the ability of the method to represent thin ligaments on coarse grids and to avoid the generation of "flotsam". During the simulation a disk of radius 0.3, centred on $(0.5;0.75)$ in a square box delimited by the points $(0;0)$ and $(1;1)$, is deformed into a spiral by a prescribed velocity field (see Fig. 12). The simulation settings for this problem are given in Fig. 12. The end of the simulation is reached at $t=3s$.

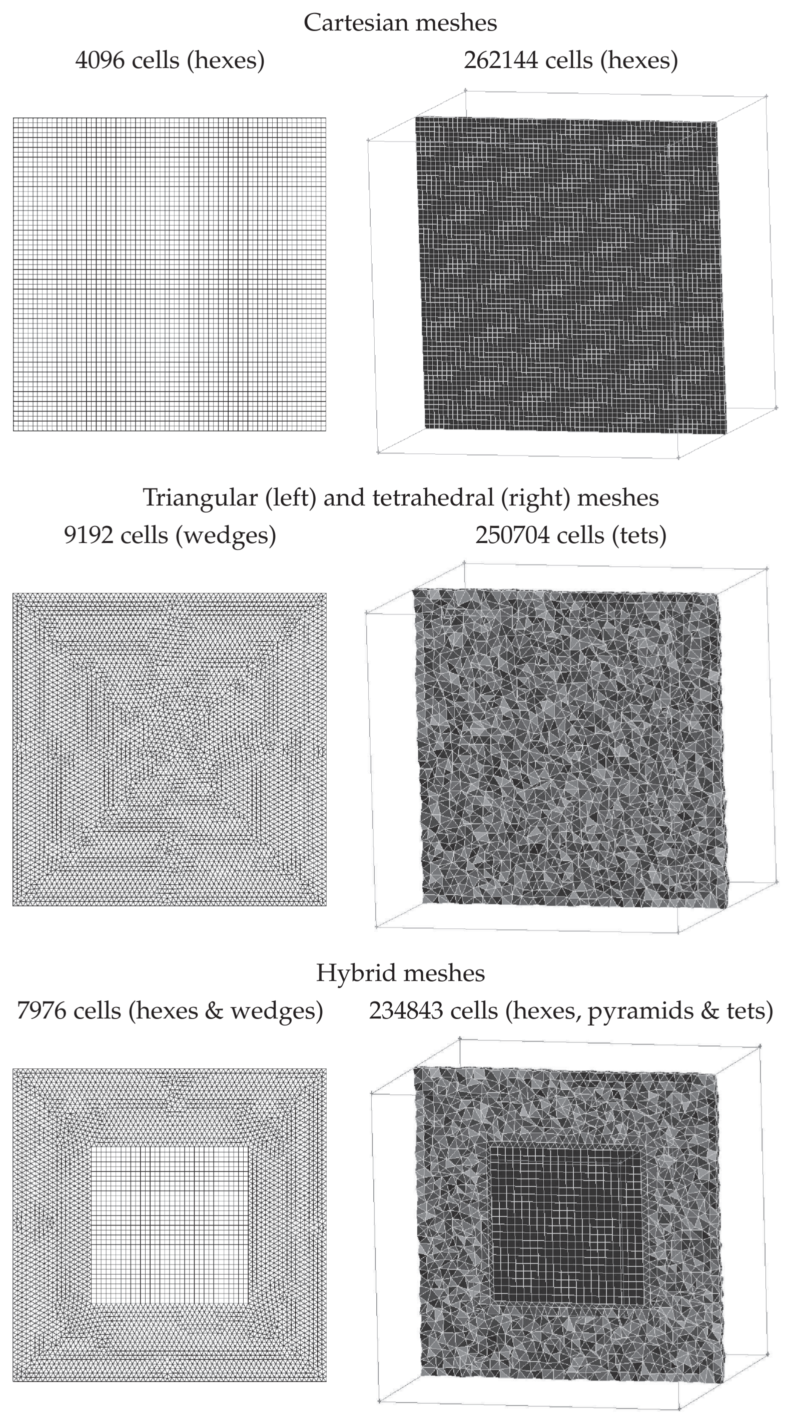

Cartesian meshes
4096 cells (hexes)
262144 cells (hexes)
Triangular (left) and tetrahedral (right) meshes
9192 cells (wedges)
250704 cells (tets)
Hybrid meshes
7976 cells (hexes & wedges)
234843 cells (hexes, pyramids & tets)

Figure 10: Meshes for the level set test cases. Left column: 2D cases; right column: 3D cases.

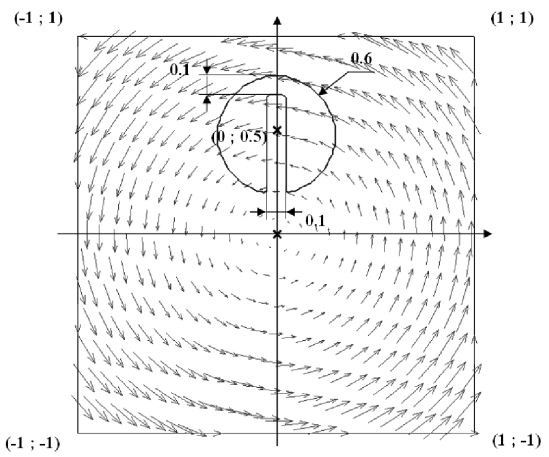


Figure 11: Simulation settings for the advection of a slotted disk. Velocity field: $u_x = -2\pi \cdot y$; $u_y = 2\pi \cdot x$.

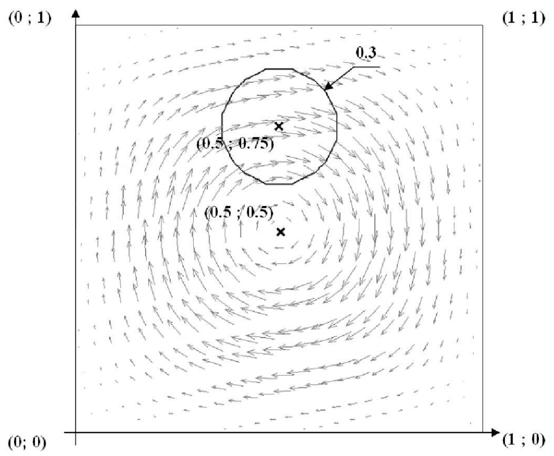


Figure 12: Simulation settings for the disk in a deformation field. Velocity field for $t \in [0,3](s)$: $u_x = -\partial\psi/\partial y$; $u_y = \partial\psi/\partial x$, with: $\psi = \pi^{-1}\sin^2(\pi x)\sin^2(\pi y)$.

The results for the disk in the deformation field are given in Fig. 13. This figure provides the solution obtained with the WENO3 scheme at $t = 3s$ for three different resolutions of $L/64$ (red line), $L/128$ (green line) and $L/256$ (blue line) on three types of grids: Cartesian, triangular and hybrid (top to bottom). The solution obtained with the WENO3 scheme on the tetrahedral grid (highest mesh density) of resolution $L/256$ has been taken as the reference case (black line), as it provides the results closest to the "exact solution" as obtained with the marker particle method by Rider [31] or the hybrid particle level set method by Enright [9].

For all test cases, under grid refinement the scheme demonstrates a greater ability to capture thin ligaments, regardless of the type of mesh. Indeed, when the mesh resolution

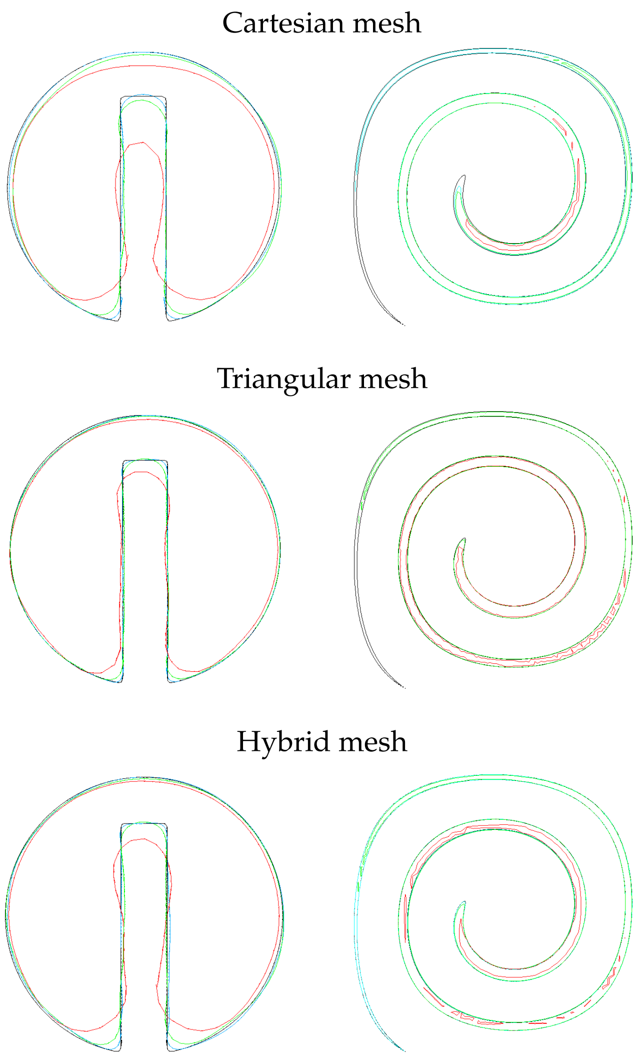


Figure 13: Zero level set for the translation of a slotted disk and the disk in a deformation field — in black: the reference; in red: solution for $L/64$; in green: solution for $L/128$; in blue: solution for $L/256$. Left column: translation of slotted disk; Right column: disk in a deformation field.

is increased, the tail of the spiral becomes systematically longer. As with the previous test case, no significant degradation of the results were observed on general unstructured grids.

6.1.2 Three-dimensional test case

Three-dimensional meshes For the three dimensional test case, the meshes have been designed to maintain similar mesh density on the three types of mesh considered (see Fig. 10, right column, top to bottom): Cartesian mesh (262144 hexes), tetrahedral mesh

(250704 tets) and hybrid mesh (234843 elements). With such meshes, we extend our comparison of the performance of the scheme to 3D Cartesian grids and unstructured meshes.

Sphere in a deformation field LeVeque extended the case presented in Section 6.1.1 (see [23]) by considering a sphere in a three dimensional deformation field given by:

$$\begin{cases} u(x,y,z)=2\cdot\sin^2(\pi x)\cdot\sin(2\pi y)\cdot\sin(2\pi z), \\ v(x,y,z)=-\sin(2\pi x)\cdot\sin^2(\pi y)\cdot\sin(2\pi z), \\ w(x,y,z)=-\sin(2\pi x)\cdot\sin(2\pi y)\cdot\sin^2(\pi z). \end{cases} \tag{6.1}$$

In this test, the domain is delimited by the points (0;0;0) and (1;1;1) and the simulation follows a sphere of radius 0.15, centred on (0.35;0.35;0.35).

The results obtained with the WENO3 scheme for the sphere in the deformation field are given in Fig. 14. This figure provides the initial level set field (top row) together with

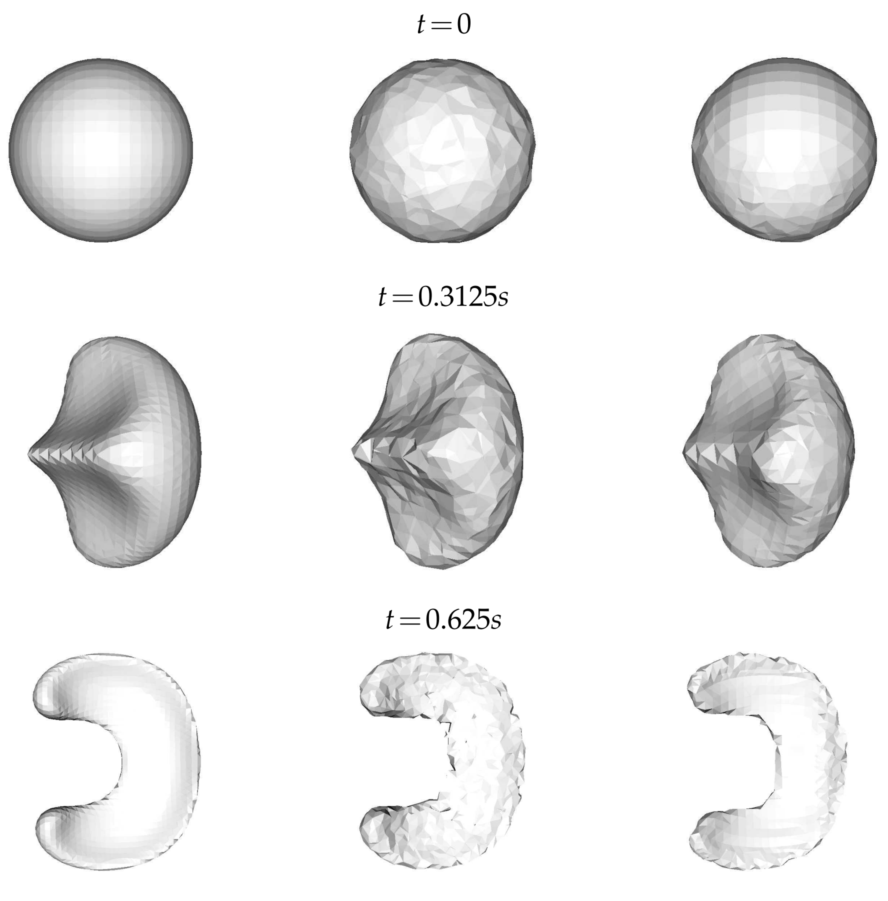


Figure 14: Zero level set for the sphere in a deformation field — Time $t=0s$, $0.3125s$, $0.625s$. Left: Cartesian mesh; Middle: Tetrahedral mesh; Right: Hybrid mesh.

the solution obtained at $t=0.3125s$ (middle row) and $t=0.625s$ (bottom row) on the three types of grids considered: Cartesian (left), tetrahedral (middle) and hybrid (right).

As for the two dimensional test cases, the results are comparable for all three types of mesh. The results obtained on the Cartesian mesh are slightly better in terms of smoothness. However, it is worth noting that the Cartesian mesh contains more elements than the tetrahedral mesh (by 5%) and the hybrid mesh (by 12%). It is clear from the results that the present approach works well on fully 3D structured, unstructured and mixed-element meshes.

6.2 Numerical convergence study

As in [42], we have chosen to demonstrate the numerical convergence of the WENO scheme on the advection of a sine function. The scalar field is transported in a cube with periodic boundary conditions, such that the solution at $t=1$ should match the initial field. The linear equation solved and the initial field are given below:

$$\begin{cases} \dfrac{\partial u}{\partial t}+2\dfrac{\partial u}{\partial x}+2\dfrac{\partial u}{\partial y}+2\dfrac{\partial u}{\partial z}=0, \quad (x,y,z)\in[-1;1]^3, \\ u(x,y,z,0)=\sin(\pi(x+y+z))+\sin(2\pi(x+y+z)). \end{cases} \tag{6.2}$$

We have solved this equation using both WENO3 and WENO4 schemes on Cartesian, tetrahedral and hybrid meshes. Three different level of refinement have been considered for this study. Similarly to the 3D meshes in Section 6.1, the unstructured meshes have been generated so that — for a given resolution — the different types of mesh have roughly the same number of elements.

The results of the numerical convergence study are presented in Tables 1 to 3. These tables provide the numerical error in the L^1 and L^2 norms and their associated convergence rates calculated using the number of cells N_c in the domain. Introducing the error in the L^p norm $\mathcal{E}_{L^p}$ and the level of grid refinement k, the formula for the order $\mathcal{O}_{L^p}^{(k)}$ reads:

$$\mathcal{O}_{L^p}^{(k)}=\ln\left(\frac{\mathcal{E}_{L^p}^{(k-1)}}{\mathcal{E}_{L^p}^{(k)}}\right)\left[\ln\left(\sqrt[3]{\frac{N_c^{t(k)}}{N_c^{(k-1)}}}\right)\right]^{-1}. \tag{6.3}$$

As expected, these results illustrate that both WENO3 and WENO4 systematically reach a convergence rate significantly higher than the order r of their respective polynomial interpolations. As in [7], we even observe that the order reached by the WENO schemes tends to $r+1$ regardless of the type of mesh.

In the case of the WENO4 scheme and for the maximum level of resolution considered, the full order of $r+1$ is not yet reached when calculating the error with the L^2 norm. However, the convergence rate seems to increase much faster for the WENO4 scheme as the mesh is refined. A similar apparent loss of relative accuracy has been observed by Pilliod [29] with high order schemes on coarse grids.

Table 1: Numerical convergence study for the Cartesian meshes – Error and associated order given for both WENO3 and WENO4.

Scheme	Number of cells	$\mathcal{E}_{L^1}$	$\mathcal{O}_{L^1}$	$\mathcal{E}_{L^2}$	$\mathcal{O}_{L^2}$
WENO3	4096	4.8322×10^{-1}	–	5.321×10^{-1}	–
	32768	3.5328×10^{-2}	3.77	3.9349×10^{-2}	3.76
	262144	2.7226×10^{-3}	3.70	3.1798×10^{-3}	3.63
WENO4	4096	5.5752×10^{-1}	–	6.0496×10^{-1}	–
	32768	5.4309×10^{-2}	3.36	6.1376×10^{-2}	3.3
	262144	1.8432×10^{-3}	4.88	3.2140×10^{-3}	4.26

Table 2: Numerical convergence study for the tetrahedral meshes – Error and associated order given for both WENO3 and WENO4.

Scheme	Number of cells	$\mathcal{E}_{L^1}$	$\mathcal{O}_{L^1}$	$\mathcal{E}_{L^2}$	$\mathcal{O}_{L^2}$
WENO3	3511	4.5490×10^{-1}	–	5.1014×10^{-1}	–
	27983	3.1000×10^{-2}	3.88	3.5946×10^{-2}	3.83
	251906	1.7644×10^{-3}	3.91	2.4047×10^{-3}	3.69
WENO4	3511	5.2429×10^{-1}	–	5.8559×10^{-1}	–
	27983	5.2578×10^{-2}	3.32	5.8891×10^{-2}	3.32
	251906	1.8486×10^{-3}	4.57	2.5985×10^{-3}	4.26

Table 3: Numerical convergence study for the hybrid meshes – Error and associated order given for both WENO3 and WENO4.

Scheme	Number of cells	$\mathcal{E}_{L^1}$	$\mathcal{O}_{L^1}$	$\mathcal{E}_{L^2}$	$\mathcal{O}_{L^2}$
WENO3	2945	5.6115×10^{-1}	–	6.2401×10^{-1}	–
	27234	6.5903×10^{-2}	2.89	8.8502×10^{-2}	2.63
	201293	4.4836×10^{-3}	4.03	6.1391×10^{-3}	4.00
WENO4	2945	5.8506×10^{-1}	–	6.5202×10^{-1}	–
	27234	9.5779×10^{-2}	2.44	1.2464×10^{-1}	2.23
	201293	5.4088×10^{-3}	4.31	7.7781×10^{-3}	4.16

A plot of the error in the L^2 norm against the normalised computational time is given for both the WENO3 and the WENO4 scheme in Figs. 15 and 16 respectively. These graphs illustrate once again that the convergence rates reached by the WENO schemes are independent of the type of grid.

It is interesting to note that Figs. 15 and 16 also suggest that — when applying the WENO schemes to the transport of a smooth field — the tetrahedral grids are more computationally efficient than the hybrid and Cartesian grids. This can be explained by the considering the number of side stencils per element and the linear weight given to the side stencils (see Section 3.2). The WENO reconstruction method attributes to an element as many side stencils as it has faces. As a result, for a given number of cells, the Cartesian mesh would involve more side stencils per element than the tetrahedral mesh. The average number of side stencil per element of the hybrid mesh (as constructed in Fig. 10)

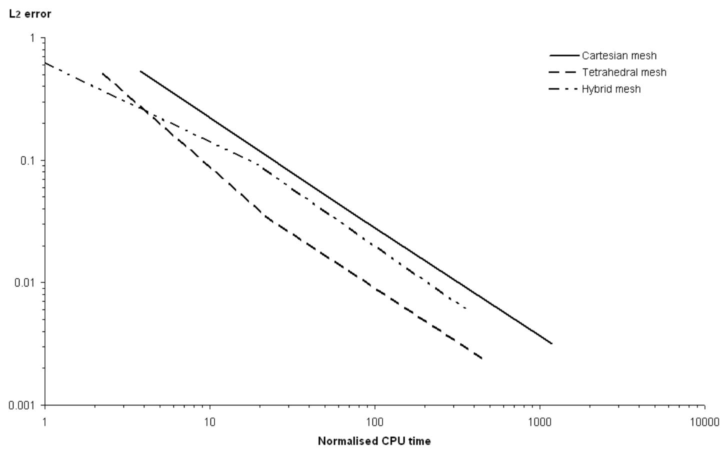


Figure 15: L^2 error vs. normalised CPU time for the WENO3 applied to the linear equation.

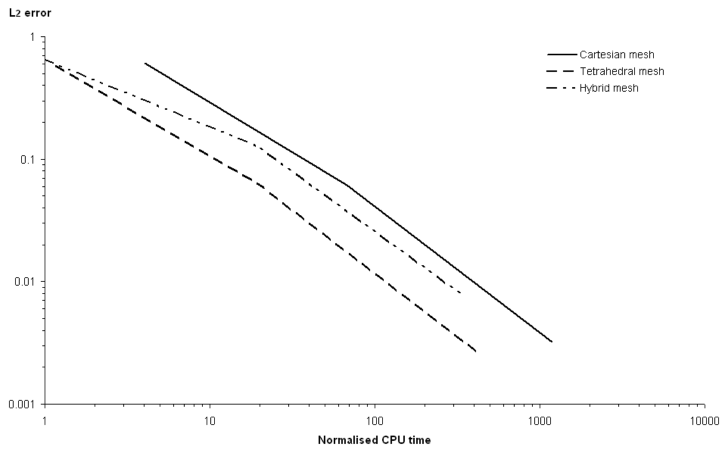
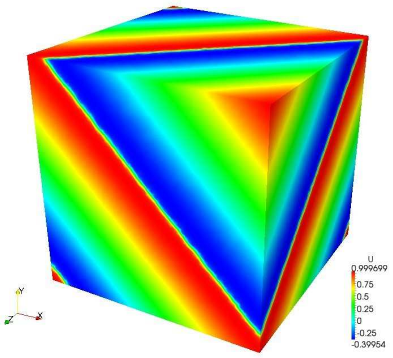


Figure 16: L^2 error vs. normalised CPU time for the WENO4 applied to the linear equation.

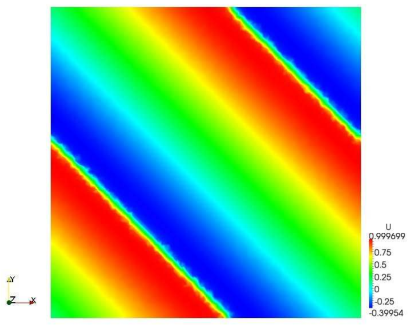
would lie in between the two. Therefore, the Cartesian mesh requires the largest amount of CPU time followed by the hybrid mesh and then by the tetrahedral mesh. As we chose to give much smaller linear weights to the side stencils than to the central stencil (see Section 3.2), for smooth solutions, the contribution of the additional side stencils to the accuracy of the calculation is not sufficient to offset the additional computational cost incurred.

6.3 Extension to a non-linear PDE

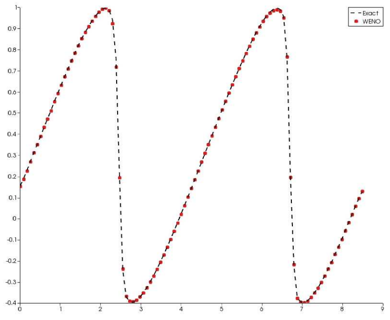
In order to demonstrate the capability of the WENO scheme on a non-linear PDE, we choose to solve the 3D Burgers′ equation on a hybrid mesh of 1.73×10^6 cells. The equation is solved with the WENO3 scheme in a cubical domain with periodic boundary con-



(a)



(b)



(c)

Figure 17: Solution for the 3D Burgers' equation on the hybrid mesh at $t=5/\pi^2$. (a) Contour plot of the surface of the domain; (b) contour plot on the cut at $z=0$; (c) exact solution (dashed line) and numerical solution (circles) along the line $x=y$ in the plane $z=0$.

ditions. We use the same initial condition as in [42], such that the shock occurs at time $t=5/\pi^2$. The settings of the simulation are given below:

$$\begin{cases} \dfrac{\partial u}{\partial t}+\dfrac{\partial}{\partial x}\Big(\dfrac{u^2}{2}\Big)+\dfrac{\partial}{\partial y}\Big(\dfrac{u^2}{2}\Big)+\dfrac{\partial}{\partial z}\Big(\dfrac{u^2}{2}\Big)=0, \quad (x,y,z)\in[-3;3]^3, \\ u(x,y,z,0)=0.3+0.7\sin\Big(\dfrac{\pi}{3}(x+y+z)\Big). \end{cases} \tag{6.4}$$

The results of the calculation are given in Fig. 17. A contour plot of the solution on the surface of the domain is shown in the top left corner of Fig. 17. In the top right corner, we show a contour plot of the solution on the cut at $z=0$. Finally, in the bottom part of the figure, we compare the numerical solution to the exact solution along the line $x=y$ in the plane $z=0$.

As intended, the scheme resolve the shock sharply with no trace of oscillatory behaviour. In addition, as suggested by the bottom picture of Fig. 17, the numerical solution matches very well the exact solution all along the line considered.

7 Conclusions

In this paper we have presented a methodology — implemented in parallel — for the construction of arbitrarily high order WENO schemes on general polyhedral unstructured meshes. The intended application is the solution of scalar level set transport equations in two-phase flow problems. Our method improves and extends the approach of Dumbser [7] — generated for tetrahedral meshes — to polyhedral meshes through a more general derivation of the reconstruction operator and the inter-cell fluxes. In addition, we have handled efficiently the notorious complexity of high order schemes on 3D mixed-element grids by generating novel algorithms.

Principally, these algorithms include the tetrahedralisation of the mesh, which allows generality of the approach while remaining efficient and affordable, together with a novel approach to stencil generation and faster interpolation of the solution. The general method for tetrahedralisation of the mesh is presented for convex polyhedral cells with convex polygonal faces. Also, we have ensured that as much as possible of the computational work is done in pre-processing steps, in order to reduce the work done at run-time.

Finally, the derivation of the resulting inter-cell fluxes is given in the case of convex polyhedral cells for linear hyperbolic systems of equations. The application of the method to the level set equation is also given, with an interpretation of the Riemann problem in such frameworks. The performance of the scheme presented has been demonstrated on typical two dimensional and three dimensional test cases of the level set method. The results obtained with the WENO3 scheme without re-distancing compare very well with existing methods that use WENO schemes of order five together with re-distancing (see [5,34]). Besides, the numerical convergence studies conducted on various types of mesh and the extension of the method to the solution of a non-linear hyperbolic PDE have demonstrated the expected performance of the scheme.

As WENO schemes are essential to level set methods applied to the simulation of multiphase flows, this work paves the way for the implementation of the level set method on 3D general unstructured meshes. This is of particular relevance for industrial CFD of multiphase flows. We are currently undertaking the development of a multiphase flow simulation tool on general unstructured meshes based on the scheme presented in this paper. Further work will consist in the efficient coupling of the level set with the incompressible Navier-Stokes equations and the implementation of the Continuum Surface Force formulation to handle the variation of properties across the interface.

Acknowledgments

This work has been funded by Rolls-Royce Group plc.

References

[1] R. Abgrall, On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation, J. Comput. Phys., 144 (1994), 45–58.

[2] T. Barth and P. Frederickson, High order solution of the Euler equations on unstructured grids using quadratic reconstruction, Tech. Rep., 90-0013, American Institute of Aeronautics and Astronautics, 1990.

[3] S. Billett and E. Toro, Numerical Methods for Wave Propagation, Chap: unsplit WAF-type schemes for three-dimensional hyperbolic conservation laws, Kluwer Academic Publishers, 1998, pp. 75–124.

[4] Y. Chang, T. Hou, B. Merriman and S. Osher, A level-set formulation of Eulerian capturing methods for incompressible fluid flows, J. Comput. Phys., 124 (1996), 449–464.

[5] F. Couderc, Développement d'un code de calcul pour la simulation d'écoulements de fluides non miscibles, Application à la désintegration assistée d'un jet liquide par un courant gazeux, Ph.D. thesis, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace- Toulouse, 2007.

[6] O. Desjardins, V. Moureau, E. Knudsen, M. Herrmann and H. Pitsch, Conservative level set/ghost fluid method for simulating primary atomization, Proceedings of the 20th Annual Conference of the Institute for Liquid Atomization and Spray Systems-Americas.

[7] M. Dumbser and M. Käser, Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems, J. Comput. Phys., 221 (2007), 693–723.

[8] M. Dumbser, M. Käser, V. Titarev and E. Toro, Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems, J. Comput. Phys., 226 (2007), 204–243.

[9] D. Enright, R. Fedkiw, J. Ferziger and I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys., 183 (2002), 83–116.

[10] G. Forsythe, M. Malcolm and C. Moler, Computer Methods for Mathematical Computations, NJ: Prentice-Hall, 1977.

[11] G. Forsythe and C. Moler, Computer Solution of Linear Algebraic Systems, NJ: Prentice-Hall, 1967.

[12] O. Friedrich, Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids, J. Comput. Phys., 144 (1998), 194–212.

[13] D. Fuster, A. Bague, T. Boeck, L. Le Moyne, A. Leboissetier, S. Popinet, P. Ray, R. Scardovelli and S. Zaleski, Simulation of primary atomization with an octree adaptive mesh refinement and VOF method, Int. J. Multiphase Flow, 35 (2009), 550–565.

[14] A. Harten, B. Engquist, S. Osher and S. Chakravarthy, Uniformly high order accurate essentially non-oscillatory schemes iii, J. Comput. Phys., 71 (1987), 231–303.

[15] A. Harten and S. Osher, Uniformly high order accurate non-oscillatory schemes i, SIAM J. Numer. Anal., 24 (1987), 279–309.

[16] M. Herrmann, A Eulerian level set/vortex sheet method for two-phase interface dynamics, J. Comput. Phys., 203 (2005), 539–571.

[17] C. Hu and C.-W. Shu, Weighted essentially non-oscillatory schemes on triangular meshes, J. Comput. Phys., 150 (1999), 97–127.

[18] G.-S. Jiang and D. Peng, Weighted ENO schemes for Hamilton-Jacobi equations, SIAM J. Sci. Comput., 21 (2000), 2126–2143.

[19] G.-S. Jiang an C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys., 126 (1996), 202.

[20] M. Käser and A. Iske, ADER schemes on adaptive triangular meshes for scalar conservation laws, J. Comput. Phys., 205 (2005), 486–508.

[21] D. Kim, O. Desjardins, M. Herrmann and P. Moin, The primary breakup of a round liquid jet by a coaxial flow of gas, Proceedings of the 20th Annual Conference of the Institute for

Liquid Atomization and Spray Systems-Americas.
[22] A. Krommer and C. Ueberhuber, Computational Integration, Society for Industrial and Applied Mathematics, 1998.
[23] R. LeVeque, High-resolution conservative algorithm for advection in incompressible flow, SIAM J. Numer. Anal., 33 (1996), 627–665.
[24] X.-D. Liu, S. Osher and T. Chan, Weighted essentially non-oscillatory schemes, J. Comput. Phys., 115 (1994), 200.
[25] T. Menard, S. Tanguy and A. Berlemont, Coupling level set/VOF/ghost fluid methods: validation and application to 3D simulation of the primary break-up of a liquid jet, Int. J. Multiphase Flow, 33 (2007), 510–524.
[26] C. Ollivier-Gooch and M. Van Altena, A high-order-accurate unstructured mesh finite-volume scheme for the advectiondiffusion equation, J. Comput. Phys., 181 (2002), 729–752.
[27] E. Olsson and G. Kreiss, A conservative level set method for two phase flow, J. Comput. Phys., 210 (2005), 225–246.
[28] R. Penrose and A. Todd, On best approximate solutions of linear matrix equations, Math. Proc. Cambridge, 52 (1956), 17–19.
[29] J. Pilliod and E. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, J. Comput. Phys., 199 (2004), 465–502.
[30] T. Pringuey and R. Cant, High order schemes on 3D mixed-element unstructured meshes, Tech. Rep. CUED/A-AERO/TR29, University of Cambridge, Department of Engineering (Sept. 2010).
[31] W. Rider and D. Kothe, Stretching and tearing interface tracking methods, AIAA Computational Fluid Dynamics Conference, 12th, San Diego CA, June 1995, Collection of Technical Papers (AIAA-1995-1717) (1995), 806–816.
[32] C. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes II, J. Comput. Phys., 83 (1989), 32–78.
[33] A. Stroud, Approximate Calculation of Multiple Integrals, NJ: Prentice-Hall, 1971.
[34] S. Tanguy, Développement d'une méthode de suivi d'interface, Applcations aux écoulements diphasiques, Ph.D. thesis, Université de Rouen, 2004.
[35] V. Titarev and D. Drikakis, Uniformly high order schemes on arbitrary unstructured meshes for advection-diffusion equations, Comput. Fluids, 46 (2011), 467–471.
[36] V. Titarev, P. Tsoutsanis and D. Drikakis, WENO schemes for mixed-element unstructured meshes, Commun. Comput. Phys., 8 (2010), 585–609.
[37] E. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction, Springer, Verlag, 1965.
[38] P. Tsoutsanis, V. Titarev and D. Drikakis, WENO schemes on arbitrary mixed-element unstructured meshes in three space dimensions, J. Comput. Phys., 230 (2011), 1585–1601.
[39] O. Ushakova, Conditions of nondegeneracy of three-dimensional cells: a formula of a volume of cells, SIAM J. Sci. Comput., 23 (2001), 1274–1290.
[40] S. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys., 31 (1979), 335–362.
[41] Y.-T. Zhang and C.-W. Chu, High order WENO schemes for Hamilton-Jacobi equations on triangular meshes, SIAM J. Sci. Comput., 24 (2003), 1005–1030.
[42] Y.-T. Zhang and C.-W. Chu, Third order WENO scheme on three dimensional tetrahedral meshes, Commun. Comput. Phys., 5 (2009), 836–848.