A method to solve Hamilton-Jacobi type equation on unstructured meshes

Alexandre Chiapolino, François Fraysse, Richard Saurel

A method to solve Hamilton-Jacobi type equation on unstructured meshes

Alexandre Chiapolino[1b], François Fraysse[2b], Richard Saurel[3a,b]

[a] *Aix Marseille Univ, CNRS, Centrale Marseille, LMA, Marseille, France*
[b] *RS2N, Chemin de Gaumin, Saint-Zacharie 83640, France*

Abstract

A new method is developed to approximate a first-order Hamilton-Jacobi equation in the context of an interface moving along its normal vector field. The interface is tracked with the help of a "Level-Set" function approximated through a finite-volume Godunov-type scheme. Contrarily to most computational approaches that consider smooth Level-Set functions, the present one considers sharp "Level-Set", the numerical diffusion being controlled with the help of the Overbee limiter (Chiapolino et al., 2017). The method requires gradient computation that is addressed through the least squares approximation. Multidimensional results on unstructured meshes are provided and checked against analytical solutions. Geometrical properties such as interfacial area and volume computation are addressed as well. Results show excellent agreement with the exact solutions.

Keywords: Hamilton-Jacobi, Sharp Level-Set, interfaces, interfacial area, unstructured meshes, hyperbolic systems, non-conservative, Godunov, Riemann, MUSCL.

[1] alexandre.chiapolino@rs2n.eu
[2] francois.fraysse@rs2n.eu
[3] richard.saurel@rs2n.eu

July 2020

1. Introduction

Many physical problems involve the motion of an interface moving along its normal vector field. A relevant example is the propagation of a combustion front in a solid propellant. Numerical resolution of such problems has been a topic of rising interest for the last decades and multiple algorithms approximating the equation of motion of propagating fronts have been designed.

More precisely the corresponding problem can be cast as a first-order Hamilton-Jacobi (HJ) equation. When thinking to interface problems, the Level-Set method often comes to mind. This method, developed originally by Osher and Sethian (1988) [1] consists in locating the interface via a signed distance function at any time. The signed distance function provides the closest distance to the interface at every point. In a series of papers, [1], [2], [3], [4], [5], [6] to cite a few, the Level-Set function has been used to solve numerically problems involving a front moving along its normal vector field.

However, the use of a signed distance function may not be trivial with initially complex interface shapes. The present contribution differs from the signed distance approach as a discontinuous "Level-Set" function is considered. With such sharp function, a constant value is assigned initially on either side of the interface separating two corresponding regions. The initialization of the present "Level-Set" function is consequently trivial regardless of the initial shape.

Nevertheless this approach presents drawbacks as well. The first one is related to numerical smearing of the interface that may disappear if the HJ equation is solved with insufficient accuracy. The second one is related to gradient computation of such sharp "Level-Set" function on unstructured meshes. Gradient computation is indeed needed to solve the HJ equation as will be seen further.

Sharp "Level-Set" approach may be used to numerically follow interfaces [7], [8], [9], [10]. However, to the authors' knowledge, the HJ equation has never been addressed with a sharp "Level-Set" function.

The present paper is inspired by techniques from hyperbolic systems and particularly Godunov-type schemes, Riemann solvers and diffuse interface methods widely used in two-phase flow modeling (Saurel and Pantano (2018) [11]).

The artificial dissipation, inherent to all capturing numerical methods, is essential for stability. Its control is of fundamental importance in the present context as excessive smearing may result in geometrical detail loss. The Overbee limiter developed recently in Chiapolino et al. (2017) [12] in the frame of the MUSCL (Monotonic Upwind Scheme for Conservation Laws) numerical scheme is used in this aim. An interface is then diffused on a limited and controlled number of elements.

The present method also takes advantage of the numerical diffusion to ensure differentiability, needed for the numerical approximation of gradients. Gradient computation is performed on unstructured meshes with the help of the least squares approximation. This approach is used as well in Chiapolino et al. (2017) [12] in the frame of diffuse interface methods for two-phase flow systems. However, some modifications are needed in the present HJ context and are addressed in this paper.

The overall method allows to solve the HJ equation on structured and unstructured meshes and provides accurate results with reasonable mesh resolutions. Computation of geometrical properties such as interfacial area and volume is addressed as well.

Interfacial area is of major importance for many applications involving interfaces, combustion, phase change and many other physical effects. However, its accurate computation is an important challenge as well and multiple directions have been investigated in the last decades, [13], [14], [15], [16], [17], [18], [19] to cite a few. In the present contribution, a new method is developed and shows excellent agreement with exact solutions.

This paper is organized as follows. Section 2 presents some background of the Hamilton-Jacobi equation and interfacial area computation. Numerical discretization of the HJ equation is addressed in Section 3, along with a 1D analysis of the corresponding problem and its extension to multidimensional configurations. Numerical accuracy is addressed with the help of the MUSCL-type reconstruction and the Overbee limiter presented in Section 4. Interfacial area and corresponding volume computations are addressed in Section 5. Multidimensional results are then provided in Section 6 and compared to analytical solutions.

2. Hamilton-Jacobi equation

The present contribution addresses numerical treatment of a Hamilton-Jacobi type equation in the context of an interface $\partial\Omega$ separating one region Ω^+ from another Ω^- and propagating along its normal direction. Tangential motion is not considered.

In the present approach, moving interfaces are tracked with the help of a "Level-Set" type function. Many contributions are referred as "Level-Set" in the literature. However, multiple types of equations use that denomination. For instance in Carmouze et al. (2018) [10] a sharp "Level-Set" function is used to follow a solid body. In that case, the velocity of the solid is uniform in space and depends only on time. In this specific context, the transport equation for the surface (and the volume) becomes a conservation equation where a flux can be defined. Its resolution is not trivial due to numerical diffusion. However, very convincing results have been obtained with the help of the gradient limiter Overbee developed recently in Chiapolino et al. (2017) [12].

In the present paper, motion of an interface along its normal vector field is of interest. In that context, the main difficulty dwells in the definition of this normal vector. More precisely, the equation of transport of the Level-Set function ϕ in the 1D configuration reads,

$$\frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} = 0, \tag{2.1}$$

with,

$$\begin{cases} \phi > 0 & \text{in the first region } \Omega^+, \\ \phi < 0 & \text{in the second region } \Omega^-, \\ \phi = 0 & \text{on the interface } \partial\Omega. \end{cases} \tag{2.2}$$

Difficulties appear as soon as the propagation speed u applies along the normal of the interface, this normal being defined as,

$$\vec{n}_F \cdot \vec{i} = \frac{\frac{\partial \phi}{\partial x}}{|\frac{\partial \phi}{\partial x}|}. \tag{2.3}$$

Vector $\vec{i}$ refers to x-axis of the 1D configuration. In this context, Eq. (2.1) becomes,

$$\frac{\partial \phi}{\partial t} + u_0 \frac{\left(\frac{\partial \phi}{\partial x}\right)^2}{|\frac{\partial \phi}{\partial x}|} = 0, \tag{2.4}$$

where u_0 represents the module of the corresponding speed. In the present contribution, interfaces move at a constant speed u_0 normal to themselves. In three dimensions, Eq. (2.4) transforms to,

$$\frac{\partial \phi}{\partial t} + u_0 \frac{\left(\vec{\nabla}\phi\right)^2}{|\vec{\nabla}\phi|} = 0. \tag{2.5}$$

It then appears that this equation is no longer a transport equation nor a conservation equation. It is a first-order Hamilton-Jacobi equation.

Over the years, important efforts have been done to compute interfaces propagating normal to themselves, [1], [2], [3], [4], [5], [6] to cite a few. However, most existing methods are designed only for Cartesian grids. Besides, the Level-Set function ϕ is considered as a signed distance function (Osher and Sethian (1988) [1]) that provides the closest distance to the interface of any mesh point, this interface being defined at $\phi = \phi_I = 0$.

The peculiarity of the signed distance function is the property: $|\vec{\nabla}\phi| = 1$. Let us take a moment to analyze this situation. For more details the reader is referred to Osher and Fedkiw (2003) [15].

The HJ equation may be written under the following form,

$$\frac{\partial \phi}{\partial t} + u_0 |\vec{\nabla}\phi| = 0. \tag{2.6}$$

However, according to the previous property in the context of the signed distance function, this last equation reduces to,

$$\frac{\partial \phi}{\partial t} = -u_0. \tag{2.7}$$

The value of the Level-Set function ϕ either increases or decreases depending on the sign of u_0. A simple forward Euler time discretization yields,

$$\phi^{n+1} = \phi^n - u_0 \Delta t, \tag{2.8}$$

where n denotes the current time iteration. When $u_0 > 0$, the $\phi^n = 0$ isocontour becomes the $\phi^{n+1} = -u_0 \Delta t$ isocontour after one time step Δt. Similarly, the $\phi^n = u_0 \Delta t$ isocontour becomes the $\phi^{n+1} = 0$ isocontour. Consequently, the $\phi = 0$ isocontour covers distance $u_0 \Delta t$ along the normal vector field. Let us now take the gradient of the temporal discretization,

$$\vec{\nabla}\phi^{n+1} = \vec{\nabla}\phi^n - \vec{\nabla}\left(u_0 \Delta t\right). \tag{2.9}$$

Since the product $u_0 \Delta t$ is spatially constant, it involves $\vec{\nabla}\left(u_0 \Delta t\right) = 0$ and consequently,

$$\vec{\nabla}\phi^{n+1} = \vec{\nabla}\phi^n. \tag{2.10}$$

It then appears that when the Level-Set function ϕ is initialized as a signed distance function satisfying $|\vec{\nabla}\phi^n| = 1$, it remains a signed distance function as time goes on. In this specific situation, the corresponding system to be solved reduces to $\frac{\partial \phi}{\partial t} = -u_0$.

This approach is appealing in appearance because this last equation is independent of space. Its numerical resolution is then simple. However, the use of the signed distance function is not trivial for initially complex shapes.

Besides, additional procedures are needed to ensure that the Level-Set function remains a signed distance function over time. Several methods have been developed with this aim such as "Fast Marching" methods (Sethian (1996, 1999) [3], [5]) and reinitializing techniques (Sussman et al. (1994) [20]).

Interfacial area and volume computations have also been addressed with the help of the Level-Set method and the signed distance function. For instance, Osher and Fedkiw (2003) [15] determine the interfacial area A_I as,

$$A_I = \int_V \delta(\phi) |\vec{\nabla}\phi| dV, \tag{2.11}$$

with V the total volume of the domain: $\Omega^+ \bigcup \Omega^-$ and $\delta(\phi)$ the Dirac delta function. In the same reference, the corresponding volumes are computed as,

$$V_{\Omega^+} = \int_V H(\phi)\, dV \quad \text{and} \quad V_{\Omega^-} = \int_V (1 - H(\phi))\, dV, \tag{2.12}$$

where $H(\phi)$ is the Heaviside function. Osher and Fedkiw (2003) [15] use a first-order accurate smeared-out approximation of $\delta(\phi)$ by defining the Heaviside function as,

$$H(\phi) = \begin{cases} 0 & \phi < -\epsilon, \\ \frac{1}{2} + \frac{\phi}{2\epsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\phi}{\epsilon}\right) & -\epsilon \leq \phi \leq \epsilon, \\ 1 & \epsilon < \phi, \end{cases} \tag{2.13}$$

where ϵ is a tunable parameter that determines the size of the bandwidth of numerical smearing. Osher and Fedkiw (2003) [15] suggest $\epsilon = 1.5\Delta x$ as a good value as it makes the interface width equal to three grid cells when ϕ is normalized as a signed distance function with $|\vec{\nabla}\phi| = 1$.

The Dirac delta function is defined as the derivative of the Heaviside function and consequently reads,

$$\delta(\phi) = \begin{cases} 0 & \phi < -\epsilon, \\ \frac{1}{2\epsilon} + \frac{1}{2\epsilon}\cos\left(\frac{\pi\phi}{\epsilon}\right) & -\epsilon \leq \phi \leq \epsilon, \\ 0 & \epsilon < \phi. \end{cases} \tag{2.14}$$

This Dirac delta function allows to evaluate the interfacial area (Eq. (2.11)) using a standard sampling technique such as the midpoint rule [15].

Note that the choice of the discretization of the Dirac delta function is an open research topic and multiple directions are investigated, see for instance Engquist et al. (2005) [21], Smereka (2006) [22].

Other approaches have been investigated to compute the interfacial area with the signed distance function. For instance, Cavallinni (2010) [17] employed isosurface extraction and performed a numerical quadrature of the extracted interface as proposed by Min and Gibou (2007) [23]. A stochastic method has been investigated as well in Sullwald (2014) [18] where the "Monte-Carlo" integration technique is discussed. It is clear that this research area is active and that various directions are still under investigation.

The present contribution differs from the above-mentioned methods, as a sharp "Level-Set" function is used. The integration of the HJ equation (2.5) is considered on fixed unstructured meshes. A tetrahedral mesh allows to reproduce accurately complex initial geometries (shapes). The "fluxes" and non-conservative terms are solutions of Riemann problems at each cell boundary and must be determined.

In the present approach, the initial function ϕ associated with the first region Ω^+ is a constant. Similarly, the initial function ϕ associated with the second region Ω^- is another constant. The interface $\partial\Omega$ is defined by the mean value of the two constants.

The equation to solve numerically is,

$$\frac{\partial \phi}{\partial t} + u_0 \frac{\left(\vec{\nabla}\phi\right)^2}{|\vec{\nabla}\phi|} = 0. \tag{2.15}$$

This equation is hyperbolic [15] and may produce "shocks" or discontinuities even when the initial conditions are smooth [2], [3], [4].

Besides, in 2D and 3D cases, additional difficulties appear regarding normal vector computation. Let us suppose a sharp angle on the interface. On such an angle, the normal vector is not defined. From the definition,

$$\vec{n}_F = \frac{\vec{\nabla}\phi}{|\vec{\nabla}\phi|}, \tag{2.16}$$

an infinite number of values is obtained in the zones where the surface is not regular.

The HJ equation requires consequently serious caution in order to perform its numerical resolution and to overcome the problems that the literature and mathematical analysis reveal.

3. Discretization of the Hamilton-Jacobi equation

3.1. Temporal discretization

In the present approach, a Heaviside profile is used to define (initially) a discontinuous "Level-Set" function, for instance:

$$\begin{cases} \phi = 1 & \text{in the first region } \Omega^+, \\ \phi = 0 & \text{in the second region } \Omega^-, \\ \phi = 0.5 & \text{on the interface } \partial\Omega. \end{cases} \tag{3.1}$$

This type of function simplifies dramatically the initialization of the "Level-Set" function compared to the signed distance function approach. The first region Ω^+ is now defined by $\phi > \phi_I = 0.5$ and the second one Ω^- by $\phi < \phi_I = 0.5$.

Let us denote,

– i the center of the numerical element of interest;

– V_i the volume of this element;

– ij the face separating element i and element j, a neighbor of element i;

– $\vec{n}_{ij}$ the outward normal vector of element i on face ij;

– S_{ij} the surface of face ij;

– $\vec{n}_{F,ij}$ the normal vector of a moving surface defined as $\vec{n}_{F,ij} = \frac{\vec{\nabla}\phi_{ij}}{|\vec{\nabla}\phi_{ij}|}$. Note that a negative sign may be present here depending on the definition of the function ϕ.

The equation to be solved reads,

$$\frac{\partial \phi}{\partial t} + u_0 \left(\vec{n}_F \cdot \vec{\nabla}\phi \right) = 0, \tag{3.2}$$

and can be considered under the following form,

$$\frac{\partial \phi}{\partial t} + u_0 \left(\vec{\nabla} \cdot (\phi \, \vec{n}_F) - \phi \, \vec{\nabla} \cdot \vec{n}_F \right) = 0. \tag{3.3}$$

The integration of Eq. (3.3) with respect to time and space of element i yields,

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t}{V_i} u_0 \sum_{ij} S_{ij} \left(\phi_{ij}^* - \phi_i^n \right) \vec{n}_{F,ij}^* \cdot \vec{n}_{ij}, \tag{3.4}$$

where the superscript ^{n} denotes the current temporal iteration and * represents the solution of the Riemann problem (Fig. 1).

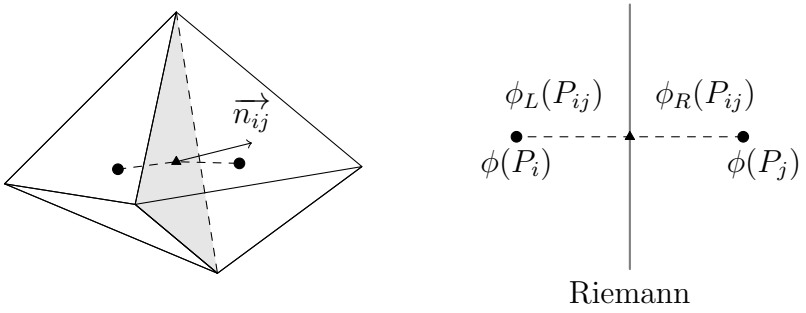

Figure 1: Schematic representation of 3D Godunov-type method applied to tetrahedron meshes. The method is cell-centered (finite volumes) and the Riemann problem is solved at each face. • centers of the elements, ▲ centers of the faces.

Note that the present integration of the non-conservative terms $\phi \, \vec{\nabla} \cdot \vec{n}_F$ supposes ϕ_i^n constant. Note also that this Godunov-type method [24] (Eq. (3.4)) is stable under the conventional CFL condition.

The determination of ϕ_{ij}^* is simple,

$$\phi_{ij}^* = \begin{cases} \phi_i & \text{if} \quad \vec{n}_{F,ij}^* \cdot \vec{n}_{ij} > 0, \\ \phi_j & \text{otherwise.} \end{cases} \tag{3.5}$$

It is based on the Riemann problem as depicted in Fig. 2.

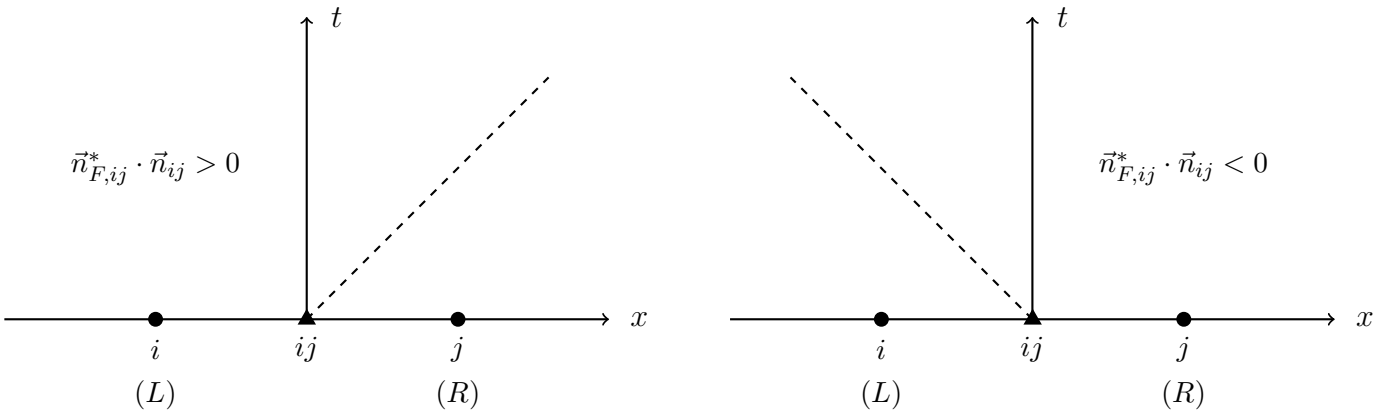

Figure 2: Schematic representation in a (x, t) diagram of the present Riemann problem. • centers of the elements, ▲ centers of the faces. The Riemann problem is solved on each face of the elements. When the dot product between the outward normal vector of the face and the normal vector of the moving surface is positive, the interface moves to the right and the Riemann problem solution is the left state (L). The opposite situation happens when the dot product is negative. The right state (R) is then solution.

The difficulty is then transferred to the determination of the local normal vector of the moving interface,

$$\vec{n}^*_{F,ij} = \frac{\vec{\nabla}\phi^*_{ij}}{|\vec{\nabla}\phi^*_{ij}|}. \tag{3.6}$$

It is important to note that a parameter $\epsilon \to 0$ is introduced in practice. When $|\vec{\nabla}\phi| > \epsilon$, the Riemann problem is solved. Otherwise, the "flux" and the non-conservative term are set to 0, *i.e.* $\left(\phi^*_{ij} - \phi^n_i\right) \vec{n}^*_{F,ij} \cdot \vec{n}_{ij} = 0$ and $\phi^{n+1}_i = \phi^n_i$ is then recovered. In the present work, ϵ is set to $\epsilon = 10^{-6}$.

3.2. Approximation of the normal vectors at cell boundaries

The normal vectors of the moving surfaces must be computed at each face of the elements composing the mesh. As mentioned earlier, the proposed method is of finite-volume type and based on Riemann problems.

The determination of ϕ^*_{ij} is simple and is done via Eq. (3.5) (Fig. 2). It is computed as the Riemann problem solution of a "transport" equation. Nevertheless, $\vec{n}^*_{F,ij} = \frac{\vec{\nabla}\phi^*_{ij}}{|\vec{\nabla}\phi^*_{ij}|}$ depends directly on the gradient $\vec{\nabla}\phi^*_{ij}$ that is still unknown at this level.

We will see in the following that $\vec{\nabla}\phi^*_{ij}$ may also be computed as the Riemann problem solution of a "transport" equation. A robust and accurate method for the computation of gradients is based on least squares approximation. This latter provides the gradient components at the centers of the elements. The least squares method is introduced further (Section 4.1).

Note that the present approach supposes that gradient computation can be performed at initial time, when no numerical diffusion is present. Nevertheless, computed results indicate that this assumption is reasonable as will be seen later.

Note also that a sharp color function may be used to locate the interface by defining two different regions. This sharp color function field may be transformed to a smoothly varying

function within a finite width, see for example [25], [26], [27], [28]. However, the present work attempts to offer a different alternative.

1D analysis

For the sake of clarity, let us analyze the present problem with a 1D configuration. The "transport" equation of the "Level-Set" function in 1D reads,

$$\frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} = 0, \tag{3.7}$$

with,

$$u = u_0 \, \vec{n}_F \cdot \vec{i} = u_0 \frac{\frac{\partial \phi}{\partial x}}{|\frac{\partial \phi}{\partial x}|}. \tag{3.8}$$

As introduced previously, the "Level-Set" function is a discontinuous function and the method is of finite-volume type. On either side of a face of an element, the variables ϕ and u are discontinuous.

On a cell face, ϕ_L and u_L are available on the left side and ϕ_R and u_R are available on the right side. The "speeds" u_L and u_R are defined by Eq. (3.8) and computed with the least squares approximation as will be seen later (Section 4.1).

Let us now consider the gradient of Eq. (3.7),

$$\frac{\partial}{\partial x} \left(\frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} \right) = 0. \tag{3.9}$$

This last equation transforms to,

$$\frac{\partial}{\partial t} \left(\frac{\partial \phi}{\partial x} \right) + \frac{\partial}{\partial x} \left(u \frac{\partial \phi}{\partial x} \right) = 0. \tag{3.10}$$

A conservative equation is then found. Let us now examine the eigenvalues of the following system, made out of the HJ equation and its gradient,

$$\begin{cases} \dfrac{\partial \phi}{\partial t} + u \dfrac{\partial \phi}{\partial x} = 0, \\ \dfrac{\partial}{\partial t} \left(\dfrac{\partial \phi}{\partial x} \right) + \dfrac{\partial}{\partial x} \left(u \dfrac{\partial \phi}{\partial x} \right) = 0. \end{cases} \tag{3.11}$$

As $u = u_0 \frac{\frac{\partial \phi}{\partial x}}{|\frac{\partial \phi}{\partial x}|}$, the second equation of System (3.11) may be written, after some algebraic manipulations, as:

$$\frac{\partial}{\partial t} \left(\frac{\partial \phi}{\partial x} \right) + u_0 \frac{\frac{\partial \phi}{\partial x}}{|\frac{\partial \phi}{\partial x}|} \frac{\partial}{\partial x} \left(\frac{\partial \phi}{\partial x} \right) = 0. \tag{3.12}$$

A "transport-like" equation for the gradient of the "Level-Set" function ϕ consequently appears,

$$\frac{\partial}{\partial t} \left(\frac{\partial \phi}{\partial x} \right) + u \frac{\partial}{\partial x} \left(\frac{\partial \phi}{\partial x} \right) = 0. \tag{3.13}$$

Let us denote,

$$\mathbf{V} = \begin{pmatrix} \phi \\ \frac{\partial \phi}{\partial x} \end{pmatrix}. \tag{3.14}$$

The corresponding system consequently reads,

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{M}(\mathbf{V}) \frac{\partial \mathbf{V}}{\partial x} = 0, \tag{3.15}$$

with,

$$\mathbf{M}(\mathbf{V}) = \begin{pmatrix} u & 0 \\ 0 & u \end{pmatrix}. \tag{3.16}$$

The present system admits u as a double eigenvalue. System (3.15) is consequently considered as a multi-evaluated "hyperbolic" system that admits two values of u on a discontinuity.

This "hyperbolic" system is made out of two "transport-like" equations, one related to the "Level-Set" function ϕ (Eq. (3.7)) and the other to its gradient $\frac{\partial \phi}{\partial x}$ (Eq. (3.13)). Those "transport-like" equations are of great interest. As already mentioned, the "Level-Set" function is solution of the Riemann problem. Its 1D reduction reads,

$$\phi_{ij}^* = \begin{cases} \phi_i & \text{if} \quad \vec{n}_{F,ij}^* \cdot \vec{i} > 0, \\ \phi_j & \text{otherwise.} \end{cases} \tag{3.17}$$

Besides, the gradient $\frac{\partial \phi^*}{\partial x}$, required for the determination of the normal vector $\vec{n}_{F,ij}^* \cdot \vec{i} = \frac{\frac{\partial \phi^*}{\partial x}}{|\frac{\partial \phi^*}{\partial x}|}$, may also be computed as,

$$\frac{\partial \phi_{ij}^*}{\partial x} = \begin{cases} \frac{\partial \phi_i}{\partial x} & \text{if} \quad \vec{n}_{F,ij}^* \cdot \vec{i} > 0, \\ \frac{\partial \phi_j}{\partial x} & \text{otherwise.} \end{cases} \tag{3.18}$$

The determination of the gradient is then itself based on the Riemann problem (Fig. 2).

Nevertheless, $\vec{n}_{F,ij}^* \cdot \vec{i} = \frac{\frac{\partial \phi_{ij}^*}{\partial x}}{|\frac{\partial \phi_{ij}^*}{\partial x}|}$, depends directly on the gradient $\frac{\partial \phi_{ij}^*}{\partial x}$ that is still unknown at this level.

However, Eqs. (3.17) and (3.18) reveal that only the sign of the dot product $\vec{n}_{F,ij}^* \cdot \vec{i}$ is necessary to determine the "Level-Set" function ϕ_{ij}^* and its gradient $\frac{\partial \phi_{ij}^*}{\partial x}$. Indeed, accurate evaluation of the dot product is not crucial to determine the solution of the Riemann problem related to the HJ equation (left or right state, Figs. 1 and 2).

Consequently, the sign of the dot product $\vec{n}_{F,ij}^* \cdot \vec{i} = \frac{\frac{\partial \phi_{ij}^*}{\partial x}}{|\frac{\partial \phi_{ij}^*}{\partial x}|} \cdot \vec{i}$ is sought in the following, in order to determine the "Level-Set" function ϕ_{ij}^* and its gradient $\frac{\partial \phi_{ij}^*}{\partial x}$ based on the Riemann problem (Eqs. (3.17) and (3.18), Figs. 1 and 2) and consequently the normal $\vec{n}_{F,ij}^* \cdot \vec{i}$ (Eq. (3.6)).

HLL solver

As Eq. (3.10) is conservative, it is a good candidate to approximate $\frac{\partial \phi_{ij}^*}{\partial x}$ and consequently the sought-after dot product via techniques from hyperbolic conservation laws (see LeVeque (2002) [29], Toro (2013) [30] for example).

System (3.15) involves another specific Riemann problem, schematically depicted in Fig. 3. Its only purpose is to determine the sign of the dot product $\vec{n}^*_{F,ij} \cdot \vec{i}$, providing consequently knowledge of the solution state (left or right) of the Riemann problem related to the HJ equation (Eqs. (3.17) and (3.18), Fig. 2).

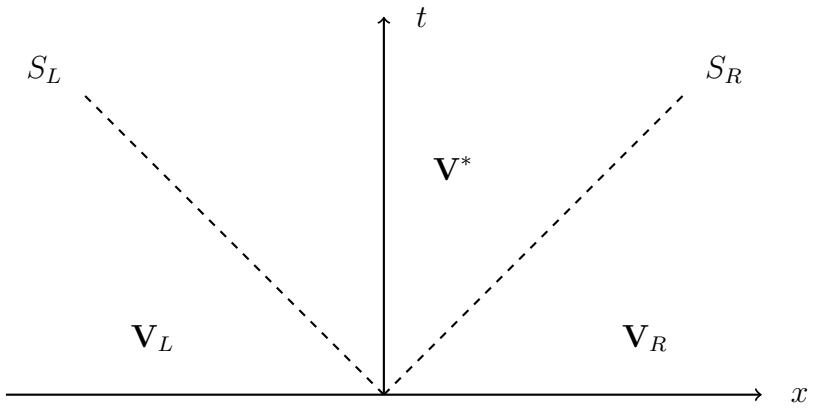

Figure 3: Schematic representation in the (x, t) of the 1D specific Riemann problem, consequence of the eigenvalues of the multi-evaluated "hyperbolic" system (3.15).

This specific Riemann problem may be solved with the help of the conservative form (3.10). Indeed, the HLL solver of Harten et al. (1983) [31] provides the solution state $U^* = \left(\frac{\partial \phi}{\partial x}\right)^*$,

$$U^* = \frac{F_L - F_R + S_R U_R - S_L U_L}{S_R - S_L}, \tag{3.19}$$

with,

$$U = \left(\frac{\partial \phi}{\partial x}\right) \quad \text{and} \quad F = u\frac{\partial \phi}{\partial x}. \tag{3.20}$$

The wave speeds S_L and S_R may be determined with the help of the estimation of Davis (1988) [32],

$$S_L = min\left(u_L, u_R\right) \quad \text{and} \quad S_R = max\left(u_L, u_R\right). \tag{3.21}$$

The conservative equation (3.10) provides $U^* = \left(\frac{\partial \phi}{\partial x}\right)^*$. However, only the sign of U^* is of interest to determine the gradient. Indeed, the sign of U^* provides knowledge of the state (left or right), solution of the gradient "transport-like" equation (3.13). When $U^* > 0$, the gradient solution of the Riemann problem related to the HJ equation is the one of the left state. On the contrary, when $U^* < 0$, the gradient solution is the one of the right state. The corresponding situation is depicted in Fig. 2.

Hereby, the multi-evaluated "hyperbolic" system (3.15) (Fig. 3) provides an estimation of the gradient $\frac{\partial \phi^*_{ij}}{\partial x}$ via Eq. (3.19) and consequently the sign of the dot product $\vec{n}^*_{F,ij} \cdot \vec{i}$. According to the sign of the dot product, the computed gradient is chosen in the left or right state (Fig. 2).

The gradients $\frac{\partial \phi_i}{\partial x}$ and $\frac{\partial \phi_j}{\partial x}$ are determined with the least squares method as will be seen later in Section 4.1. Its 1D analogue corresponds to the centered approximation,

$$\frac{\partial \phi_i}{\partial x} = \frac{1}{2\Delta x} \left(\phi_{i+1} - \phi_{i-1} \right). \tag{3.22}$$

The solution evolves to the next time step with the help of the Godunov-type method, Eq. (3.4), reducing to,

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t}{\Delta x} u_0 \left(\phi^*_{i+1/2} \vec{n}^*_{F,i+1/2} \cdot \vec{i} - \phi^*_{i-1/2} \vec{n}^*_{F,i-1/2} \cdot \vec{i} \right) + \frac{\Delta t}{\Delta x} u_0 \phi_i^n \left(\vec{n}^*_{F,i+1/2} \cdot \vec{i} - \vec{n}^*_{F,i-1/2} \cdot \vec{i} \right), \tag{3.23}$$

with $\vec{n}^*_{F,i\pm+1/2} \cdot \vec{i} = \frac{\frac{\partial \phi^*_{i\pm1/2}}{\partial x}}{\left| \frac{\partial \phi^*_{i\pm1/2}}{\partial x} \right|}$ in the 1D configuration. The indexes i and $i \pm 1/2$ represent the center of element i and its corresponding boundaries. Figure 4 displays the present 1D situation.

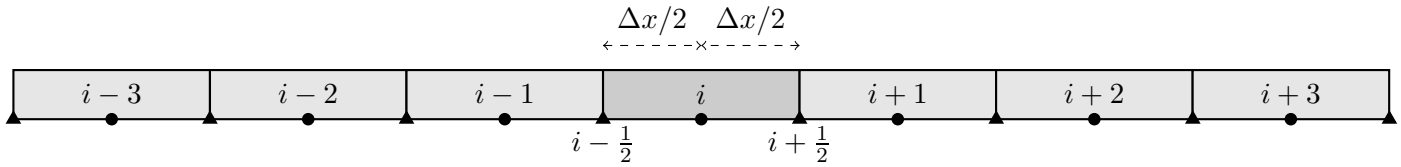

Figure 4: Schematic representation of a 1D mesh. The space step Δx is constant. Symbols • represent the centers of the elements. Symbols ▲ represent the centers of the faces.

The method is now tested on a 1D configuration. The following test consists in a Heaviside function regressing in the normal vector direction of the discontinuities as depicted in Fig. 5.

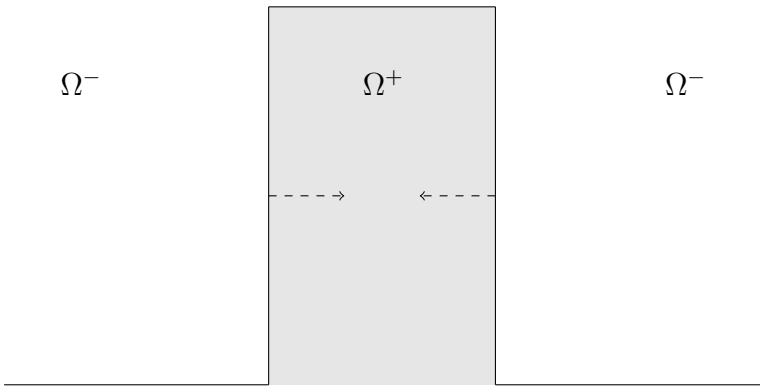

Figure 5: Schematic representation of a simple 1D regression test of a Heaviside profile. This later moves along its normal vector field with speed u_0.

Figure 6 shows the computed results. The exact solution is shown as well.

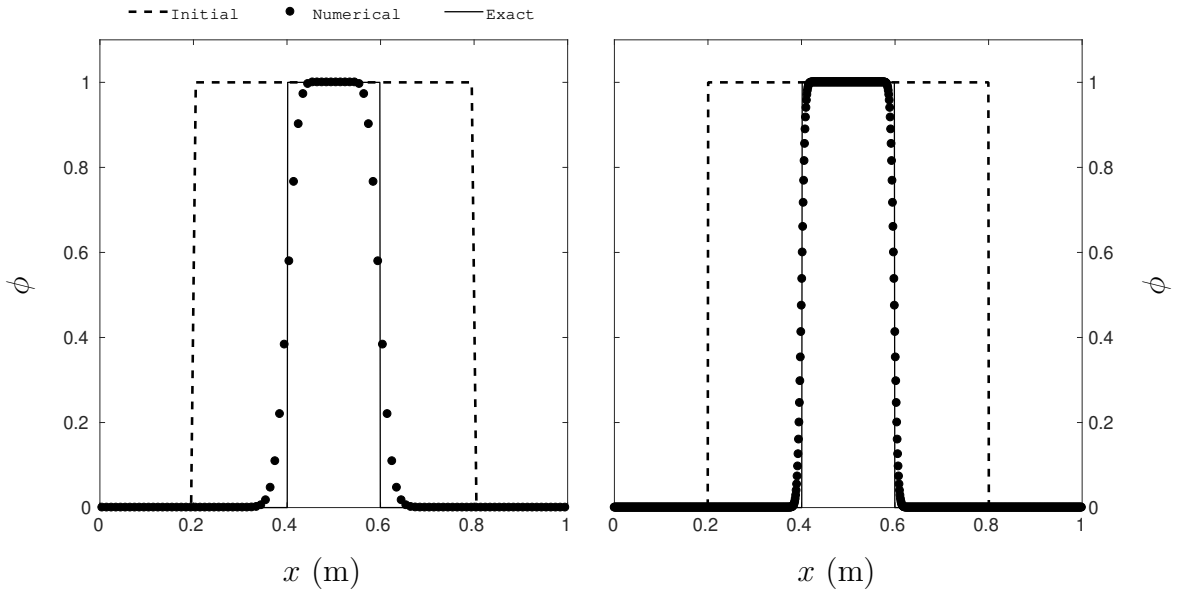

Figure 6: Comparison of the computed solution with the present Godunov-type first-order scheme and corresponding Riemann solver to the exact solution. The speed module along the normal direction of the front is $u_0 = 1$ m.s^{-1}. The final time is $t = 0.2$ s and CFL$= 0.8$ is used. In the figure on the left, 100 cells are used. In the figure on the right, 1000 cells are used. The full circle symbols $\bullet$ represent the corresponding results. The dashed lines represent the initial condition. The full lines represent the exact solution (plotted with 1000 points).

The results are in agreement with the exact solution. The method is now extended to multidimensional configurations on unstructured meshes.

3.3. Multidimensional extension

The method is based on the solution of Riemann problem regarding two "transport-like" equations, one for the "Level-Set" function ϕ and the other for its gradient $\vec{\nabla}\phi$,

$$\phi_{ij}^* = \begin{cases} \phi_i & \text{if} \quad \vec{n}_{F,ij}^* \cdot \vec{n}_{ij} > 0, \\ \phi_j & \text{otherwise,} \end{cases} \qquad \text{and} \quad \vec{\nabla}\phi_{ij}^* = \begin{cases} \vec{\nabla}\phi_i & \text{if} \quad \vec{n}_{F,ij}^* \cdot \vec{n}_{ij} > 0, \\ \vec{\nabla}\phi_j & \text{otherwise,} \end{cases} \tag{3.24}$$

where the gradients $\vec{\nabla}\phi_i$ and $\vec{\nabla}\phi_j$ are determined with the least squares method as will be seen later in Section 4.1. However, Relations (3.24) require the sign of the dot product $\vec{n}_{F,ij}^* \cdot \vec{n}_{ij}$.

The specific Riemann problem of Fig. 3 is now extended to multidimensional configurations on unstructured meshes. Its solution allows to determine the sign of the dot product $\vec{n}_{F,ij}^* \cdot \vec{n}_{ij}$ and consequently to choose the gradient computed in the left or right state, that is the solution of the Riemann problem related to the HJ equation (Figs. 1 and 2).

The "Level-Set" equation now reads,

$$\frac{\partial \phi}{\partial t} + \vec{u} \cdot \vec{\nabla}\phi = 0, \tag{3.25}$$

with,

$$\vec{u} = u_0 \frac{\vec{\nabla}\phi}{|\vec{\nabla}\phi|}. \tag{3.26}$$

Let us consider the gradient of Eq. (3.25),

$$\vec{\nabla}\left(\frac{\partial\phi}{\partial t}+\left(\vec{u}\cdot\vec{\nabla}\phi\right)\right)=0. \tag{3.27}$$

A conservative equation is consequently found,

$$\frac{\partial}{\partial t}\left(\vec{\nabla}\phi\right)+\vec{\nabla}\cdot\left(\vec{u}\cdot\vec{\nabla}\phi\,\underline{\underline{\mathbf{I}}}\right)=0. \tag{3.28}$$

This last equation presents some similarities with the momentum equation in fluid mechanics. Indeed, let us denote,

$$\begin{cases}\vec{\nabla}\phi=\dfrac{\partial\phi}{\partial x}\vec{i}+\dfrac{\partial\phi}{\partial y}\vec{j}+\dfrac{\partial\phi}{\partial z}\vec{k}=m\,\vec{i}+n\,\vec{j}+o\,\vec{k},\\ \vec{u}=u\,\vec{i}+v\,\vec{j}+w\,\vec{k}.\end{cases} \tag{3.29}$$

Thereby Eq. (3.28) may be written as,

$$\frac{\partial\left(m\,\vec{i}+n\,\vec{j}+o\,\vec{k}\right)}{\partial t}+\vec{\nabla}\cdot\left(\left(u\,\vec{i}+v\,\vec{j}+w\,\vec{k}\right)\cdot\left(m\,\vec{i}+n\,\vec{j}+o\,\vec{k}\right)\underline{\underline{\mathbf{I}}}\right)=0. \tag{3.30}$$

Hereby,

$$\begin{cases}\dfrac{\partial m}{\partial t}+\dfrac{\partial\left(mu+nv+ow\right)}{\partial x}=0,\\ \dfrac{\partial n}{\partial t}+\dfrac{\partial\left(mu+nv+ow\right)}{\partial y}=0,\\ \dfrac{\partial o}{\partial t}+\dfrac{\partial\left(mu+nv+ow\right)}{\partial z}=0.\end{cases} \tag{3.31}$$

However, these last relations can only be used on Cartesian grids. On unstructured meshes, the projection of Eq. (3.30) along the normal vector to a face ij reads,

$$\frac{\partial\left(m\,\vec{i}+n\,\vec{j}+o\,\vec{k}\right)\cdot\vec{n}_{ij}}{\partial t}+\frac{\partial\left(mu+nv+ow\right)}{\partial\eta}=0, \tag{3.32}$$

where η is the coordinate of the normal of the face.

The specific Riemann problem (Fig. 3) directly applies to this last equation. The proposed method consists in determining the sign of the dot product $\vec{n}^{*}_{F,ij}\cdot\vec{n}_{ij}$ in order to extract the solution state of the Riemann problem related to the HJ equation (Figs. 1 and 2). With the present notations, the corresponding term is $\left(u\,\vec{i}+v\,\vec{j}+w\,\vec{k}\right)\cdot\vec{n}_{ij}$.

However, the norm of the gradient being necessarily positive (the case $|\vec{\nabla}\phi|=0$ is omitted), the determination of the sign of $\vec{n}^{*}_{F,ij}\cdot\vec{n}_{ij}=u_0\,\frac{\vec{\nabla}\phi^{*}_{ij}}{|\vec{\nabla}\phi^{*}_{ij}|}\cdot\vec{n}_{ij}$ consists in determining the sign of $u_0\,\vec{\nabla}\phi^{*}_{ij}\cdot\vec{n}_{ij}$. With the present notations, the corresponding term is $\left(m\,\vec{i}+n\,\vec{j}+o\,\vec{k}\right)\cdot\vec{n}_{ij}$.

For each cell center, the following vector is available,

$$\vec{u}=u_0\,\vec{n}=u_0\frac{\vec{\nabla}\left(\phi\right)}{|\,\vec{\nabla}\left(\phi\right)\,|}=u\,\vec{i}+v\,\vec{j}+w\,\vec{k}, \tag{3.33}$$

as well as,

$$\vec{\nabla}\phi = \frac{\partial\phi}{\partial x}\,\vec{i} + \frac{\partial\phi}{\partial y}\,\vec{j} + \frac{\partial\phi}{\partial z}\,\vec{k} = m\,\vec{i} + n\,\vec{j} + o\,\vec{k}, \tag{3.34}$$

determined with the help of the least squares method as will be seen in Section 4.1.

For a given face, the states $U_L = \left(\left(m\,\vec{i} + n\,\vec{j} + o\,\vec{k}\right)\cdot\vec{n}_{ij}\right)_L$ and $U_R = \left(\left(m\,\vec{i} + n\,\vec{j} + o\,\vec{k}\right)\cdot\vec{n}_{ij}\right)_R$ are available as well as the fluxes $F_L = (mu + nv + ow)_L$ and $F_R = (mu + nv + ow)_R$.

The HLL relation (3.19),

$$U^* = \frac{F_L - F_R + S_R U_R - S_L U_L}{S_R - S_L},$$

directly applies and provides $\left(\left(m\,\vec{i} + n\,\vec{j} + o\,\vec{k}\right)\cdot\vec{n}_{ij}\right)^* = \left(u_0\,\vec{\nabla}\phi_{ij}\cdot\vec{n}_{ij}\right)^*$. Note that the HLL relation (3.19) does not provide the normal vector $\vec{n}^*_{F,ij}$ nor the gradient $\vec{\nabla}\phi^*_{ij}$ on a face but only the scalar identity $\left(u_0\,\vec{\nabla}\phi_{ij}\cdot\vec{n}_{ij}\right)^*$. The sign of this last term allows to determine the solution state of the Riemann problem related to the HJ equation (Figs. 1 and 2).

In the next section, the proposed numerical scheme is extended to second order, a necessary improvement in the present context. Indeed, numerical dissipation of the "Level-Set" function will make interfaces disappear prematurely. In order to avoid the use of too fine meshes, the MUSCL-type extension used in Chiapolino et al. (2017) [12] is reminded and adapted to the present context.

4. MUSCL-type scheme

The extension to second order is done via the MUSCL-type scheme presented hereafter. Denoting by $V_i(P_i)$ and $V_j(P_j)$ two elements with cell centers P_i and P_j delimited by the boundary S_{ij} (see Fig. 7), the space-time Taylor expansion of the "Level-Set" function ϕ at the point P_{ij}, barycenter of S_{ij}, from the point P_i of the "Level-Set" function ϕ reads,

$$\phi_L(P_{ij}) \simeq \phi(P_i) + \vec{r}_{ij}.\,\vec{\nabla}\phi(P_i) + \Delta t\frac{\partial\phi(P_i)}{\partial t}, \quad \vec{r}_{ij} = \overrightarrow{P_iP_{ij}}. \tag{4.1}$$

Similar expansion at P_{ij} from P_j reads,

$$\phi_R(P_{ij}) \simeq \phi(P_j) + \vec{r}_{ji}.\,\vec{\nabla}\phi(P_j) + \Delta t\frac{\partial\phi(P_j)}{\partial t}, \quad \vec{r}_{ji} = \overrightarrow{P_jP_{ij}}. \tag{4.2}$$

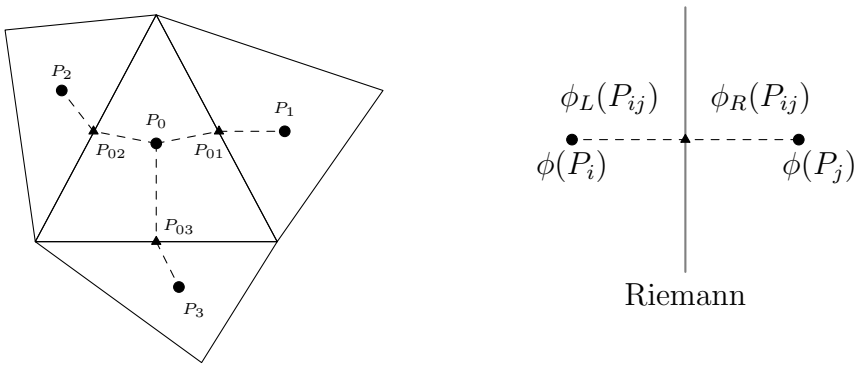

Figure 7: Schematic representation of an unstructured mesh made of triangles. • centers of the elements, ▲ centers of the faces. The Riemann problem is solved on each face of the triangles.

The gradients $\vec{\nabla}\phi_L(P_{ij})$ and $\vec{\nabla}\phi_R(P_{ij})$ are also expressed at the center of the face ij via similar Taylor expansions,

$$\begin{cases} \vec{\nabla}\phi_L(P_{ij}) \simeq \vec{\nabla}\phi(P_i) + \vec{r}_{ij}.\,\vec{\nabla}\left(\vec{\nabla}\phi(P_i)\right) + \Delta t\dfrac{\partial\vec{\nabla}\phi(P_i)}{\partial t}, \qquad \vec{r}_{ij} = \overrightarrow{P_iP_{ij}}. \\ \vec{\nabla}\phi_R(P_{ij}) \simeq \vec{\nabla}\phi(P_j) + \vec{r}_{ji}.\,\vec{\nabla}\left(\vec{\nabla}\phi(P_j)\right) + \Delta t\dfrac{\partial\vec{\nabla}\phi(P_j)}{\partial t}, \qquad \vec{r}_{ji} = \overrightarrow{P_jP_{ij}} \end{cases} \tag{4.3}$$

The reconstructed solutions at left $\phi_L(P_{ij})$, $\vec{\nabla}\phi_L(P_{ij})$ and at right $\phi_R(P_{ij})$, $\vec{\nabla}\phi_R(P_{ij})$ are used as initial conditions for the Riemann problems in order to obtain more accurate numerical results. The MUSCL-type scheme takes into account both data reconstruction and time evolution with the following sequence of computations.

Spatial reconstruction at cell boundaries

The spatial reconstruction step uses the preceding formulas (4.1), (4.2), (4.3) without the time derivative, this one being approximated in the next predictor step,

$$\phi_L^n(P_{ij}) \simeq \phi^n(P_i) + \vec{r}_{ij}.\,\vec{\nabla}\phi^n(P_i), \quad \vec{r}_{ij} = \overrightarrow{P_iP_{ij}}. \tag{4.4}$$

Similar expansion at P_{ij} from P_j reads,

$$\phi_R^n(P_{ij}) \simeq \phi^n(P_j) + \vec{r}_{ji}.\,\vec{\nabla}\phi^n(P_j), \quad \vec{r}_{ji} = \overrightarrow{P_jP_{ij}}. \tag{4.5}$$

Spatial reconstruction of the gradient components are written with the help of the Hessian matrix,

$$\begin{cases} \dfrac{\partial\phi_L^n(P_{ij})}{\partial x} \simeq \dfrac{\partial\phi^n(P_i)}{\partial x} + \left(\vec{r}_{x,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x^2} + \vec{r}_{y,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x\partial y} + \vec{r}_{z,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x\partial z}\right), \\ \dfrac{\partial\phi_L^n(P_{ij})}{\partial y} \simeq \dfrac{\partial\phi^n(P_i)}{\partial y} + \left(\vec{r}_{x,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x\partial y} + \vec{r}_{y,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial y^2} + \vec{r}_{z,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial y\partial z}\right), \\ \dfrac{\partial\phi_L^n(P_{ij})}{\partial z} \simeq \dfrac{\partial\phi^n(P_i)}{\partial z} + \left(\vec{r}_{x,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x\partial z} + \vec{r}_{y,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial y\partial z} + \vec{r}_{z,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial z^2}\right), \end{cases} \tag{4.6}$$

with $\vec{r}_{x,i,ij}$, $\vec{r}_{y,i,ij}$ and $\vec{r}_{z,i,ij}$ the projected distance $\overrightarrow{P_iP_{ij}}$ along axes x, y and z. The same reasoning applies to the right side,

$$\begin{cases} \dfrac{\partial \phi_R^n(P_{ij})}{\partial x} \simeq \dfrac{\partial \phi^n(P_j)}{\partial x} + \left(\vec{r}_{x,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial x^2} + \vec{r}_{y,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial x \partial y} + \vec{r}_{z,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial x \partial z} \right), \\ \dfrac{\partial \phi_R^n(P_{ij})}{\partial y} \simeq \dfrac{\partial \phi^n(P_j)}{\partial y} + \left(\vec{r}_{x,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial x \partial y} + \vec{r}_{y,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial y^2} + \vec{r}_{z,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial y \partial z} \right), \\ \dfrac{\partial \phi_R^n(P_{ij})}{\partial z} \simeq \dfrac{\partial \phi^n(P_j)}{\partial z} + \left(\vec{r}_{x,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial x \partial z} + \vec{r}_{y,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial y \partial z} + \vec{r}_{z,j,ij} \dfrac{\partial^2 \phi^n(P_j)}{\partial z^2} \right). \end{cases} \tag{4.7}$$

Superscript ^{n} denotes the current temporal iteration. During that step, the gradients $\vec{\nabla}\phi^n(P_i)$ and $\vec{\nabla}\phi^n(P_j)$ are determined with the help of the least squares method presented in Section 4.1. The preceding relations yield a second-order-in-space discretization. At this time, reconstructed variables at left $\phi_L^n(P_{ij})$, $\vec{\nabla}\phi_L^n(P_{ij})$ and right $\phi_R^n(P_{ij})$, $\vec{\nabla}\phi_R^n(P_{ij})$ of the cell faces are available.

Half-time step evolution

The cell-center "Level-Set" function ϕ_i^n is evolved during a half-time step with the Godunov method, requiring Riemann problem resolutions at cell faces,

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t}{2V_i} u_0 \sum_{ij} S_{ij} \left(\phi_{ij}^* - \phi_i^n \right) \vec{n}_{F,ij}^* \cdot \vec{n}_{ij} \quad \text{with} \quad \vec{n}_{F,ij}^* = \frac{\vec{\nabla}\phi_{ij}^*}{|\vec{\nabla}\phi_{ij}^*|}. \tag{4.8}$$

Superscript * denotes the solution of the Riemann problem. During this step, the states at left $\phi_L^n(P_{ij})$, $\vec{\nabla}\phi_L^n(P_{ij})$ and right $\phi_R^n(P_{ij})$, $\vec{\nabla}\phi_R^n(P_{ij})$ (Eqs. (4.4), (4.5), (4.6) and (4.7)) of cell faces come from the previous spatial-reconstruction-at-cell-boundary step and are used as initial data of the Riemann problems providing ϕ_{ij}^{*n} at the cell faces. The normal vectors $\vec{n}_{F,ij}^*$ are required as well and are determined with the method presented in the previous section.

Full-time step evolution

The spatial reconstruction step is repeated with the help of the "Level-Set" function in the $^{n+1/2}$ state,

$$\phi_L^{n+1/2}(P_{ij}) \simeq \phi^{n+1/2}(P_i) + \vec{r}_{ij}.\,\vec{\nabla}\phi^n(P_i), \quad \vec{r}_{ij} = \overrightarrow{P_iP_{ij}}. \tag{4.9}$$

Similar expansion at P_{ij} from P_j reads,

$$\phi_R^{n+1/2}(P_{ij}) \simeq \phi^{n+1/2}(P_j) + \vec{r}_{ji}.\,\vec{\nabla}\phi^n(P_j), \quad \vec{r}_{ji} = \overrightarrow{P_jP_{ij}}. \tag{4.10}$$

The gradients $\vec{\nabla}\phi^n(P_i)$ and $\vec{\nabla}\phi^n(P_j)$ come from the first spatial reconstruction step. Note that the gradients at left $\vec{\nabla}\phi_L^n(P_{ij})$ and at right $\vec{\nabla}\phi_R^n(P_{ij})$ may also be evolved into the $^{n+1/2}$ state. However, numerical experiments reveal that a second gradient reconstruction is not necessary. Consequently, only one spatial reconstruction is done for the gradient whereas two are done for the "Level-Set" function. This choice is made for the sake of simplicity. Indeed, computed results with or without this second gradient reconstruction are in close agreement. However, for the sake of space restriction and clarity those results are not presented.

From the extrapolated variables at left $\phi_L^{n+1/2}(P_{ij})$ and right $\phi_R^{n+1/2}(P_{ij})$, a second Riemann problem is solved yielding a more accurate ϕ_{ij}^*. The solution is then evolved during the full-time step with the Godunov method,

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t}{V_i} u_0 \sum_{ij} S_{ij} \left(\phi_{ij}^{*,n+1/2} - \phi_i^n \right) \vec{n}_{F,ij}^* \cdot \vec{n}_{ij} \quad \text{with} \quad \vec{n}_{F,ij}^* = \frac{\vec{\nabla}\phi_{ij}^{*,n}}{|\vec{\nabla}\phi_{ij}^{*,n}|}. \tag{4.11}$$

This MUSCL-type scheme is thus summarized in three steps,

- Spatial reconstruction at cell boundaries;
- Half-time step evolution (prediction);
- Full-time step evolution.

Figure 8 displays a schematic representation of the procedure. The MUSCL-type scheme presented previously requires to solve two Riemann problems per time step but only one gradient computation. This last point is addressed in the following section.

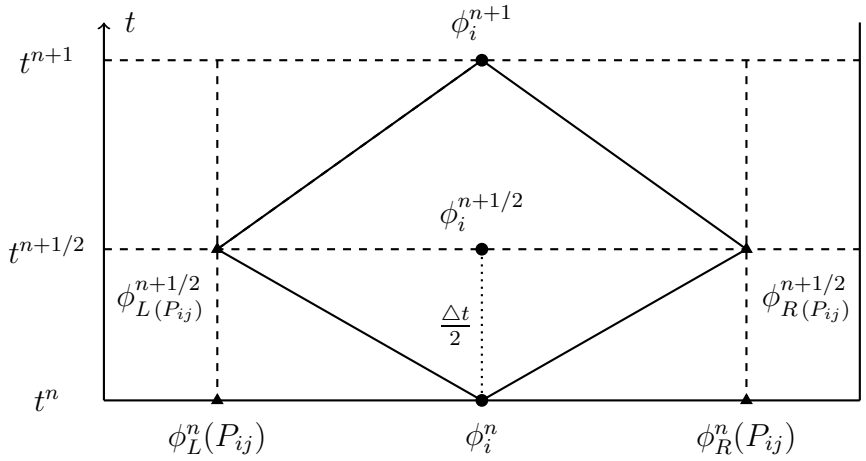

Figure 8: Schematic representation of the MUSCL-type numerical scheme. At time t^n, values at the faces $\phi_L^n(P_{ij})$, $\vec{\nabla}\phi_L^n(P_{ij})$ and $\phi_R^n(P_{ij})$, $\vec{\nabla}\phi_R^n(P_{ij})$ (Eqs. (4.4), (4.6) (4.5), (4.7)) reconstructed via the gradients and Hessian matrix, are used as initial data of a Riemann problem. The solution evolves at time $t^{n+1/2}$ via the Godunov-type scheme (Eq. (4.8)). At this intermediate time, the previous gradients are used to reconstruct the solution at the faces $\phi_L^{n+1/2}(P_{ij})$ and $\phi_R^{n+1/2}(P_{ij})$, (Eqs. (4.9), (4.10)). Those states are used as initial data of a second Riemann problem. Finally, values at cell center ϕ_i^n are updated to ϕ_i^{n+1} with the Godunov-type scheme (Eq. (4.11)).

4.1. Gradients' computation on unstructured meshes

A robust and accurate method for the computation of gradients is based on least squares approximation. This method is more expensive than the direct use of Green-Gauss theorem but

provides more accurate results. It is based on multiple Taylor expansions about P_i and a cloud of neighboring cells,

$$\begin{aligned}\phi_j = \phi_i + \Delta x_{ij}\frac{\partial \phi_i}{\partial x} + \Delta y_{ij}\frac{\partial \phi_i}{\partial y} + \Delta z_{ij}\frac{\partial \phi_i}{\partial z} + \frac{(\Delta x_{ij})^2}{2}\frac{\partial^2 \phi_i}{\partial x^2} + \frac{(\Delta y_{ij})^2}{2}\frac{\partial^2 \phi_i}{\partial y^2} + \frac{(\Delta z_{ij})^2}{2}\frac{\partial^2 \phi_i}{\partial z^2} \\ + \Delta x_{ij}\Delta y_{ij}\frac{\partial^2 \phi_i}{\partial x \partial y} + \Delta x_{ij}\Delta z_{ij}\frac{\partial^2 \phi_i}{\partial x \partial z} + \Delta y_{ij}\Delta z_{ij}\frac{\partial^2 \phi_i}{\partial y \partial y} + \mathrm{O}\left(\|\overrightarrow{P_i P_j}\|^3\right).\end{aligned} \tag{4.12}$$

In the present context, the second partial derivatives are required as present in Eqs. (4.6), (4.7). Using Eq. (4.12) with a set of neighbors results in the following system:

$$\mathbf{AX} = \mathbf{B}, \tag{4.13}$$

with,

$$\mathbf{A} = \begin{pmatrix} \Delta x_{i1} & \Delta y_{i1} & \Delta z_{i1} & \frac{1}{2}(\Delta x_{i1})^2 & \frac{1}{2}(\Delta y_{i1})^2 & \frac{1}{2}(\Delta z_{i1})^2 & \Delta x_{i1}\Delta y_{i1} & \Delta x_{i1}\Delta z_{i1} & \Delta y_{i1}\Delta z_{i1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \Delta x_{iN} & \Delta y_{iN} & \Delta z_{iN} & \frac{1}{2}(\Delta x_{iN})^2 & \frac{1}{2}(\Delta y_{iN})^2 & \frac{1}{2}(\Delta z_{iN})^2 & \Delta x_{iN}\Delta y_{iN} & \Delta x_{iN}\Delta z_{iN} & \Delta y_{iN}\Delta z_{iN} \end{pmatrix},$$

$$\mathbf{X} = \begin{pmatrix} \frac{\partial \phi_i}{\partial x} & \frac{\partial \phi_i}{\partial y} & \frac{\partial \phi_i}{\partial z} & \frac{\partial^2 \phi_i}{\partial x^2} & \frac{\partial^2 \phi_i}{\partial y^2} & \frac{\partial^2 \phi_i}{\partial z^2} & \frac{\partial^2 \phi_i}{\partial x \partial y} & \frac{\partial^2 \phi_i}{\partial x \partial z} & \frac{\partial^2 \phi_i}{\partial y \partial z} \end{pmatrix}^T,$$

$$\mathbf{B} = \begin{pmatrix} \phi_1 - \phi_i \\ \vdots \\ \phi_N - \phi_i \end{pmatrix}, \tag{4.14}$$

where N is the number of neighboring elements.

In three dimensions, a minimum of nine neighboring elements is necessary to solve the system. When the number of available neighbors is greater than nine, the system becomes over-determined and solution of minimum residual $\|\mathbf{AX} - \mathbf{B}\|$ is addressed. A conventional way to solve this over-determined system is to multiply both sides by the transpose matrix. A square system (the so-called normal equations) is obtained: $\mathbf{AX} = \mathbf{B}$ becomes $\mathbf{A}^T\mathbf{AX} = \mathbf{A}^T\mathbf{B}$, and the solution reads, $\mathbf{X} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B}$.

The main issue regarding this methodology is linked with the condition number of the matrix $\mathbf{A}$, $cond(\mathbf{A})$. If it is big (ill-conditioned) then the system of normal equations $\mathbf{A}^T\mathbf{AX} = \mathbf{A}^T\mathbf{B}$ yields a condition number even bigger, $cond(\mathbf{A})^2$. A large condition number is highly undesirable as its numerical solution may be very difficult to achieve accurately. A second approach is to use a **QR** decomposition as will be seen further. Note that other options such as the singular value decomposition (SVD) are possible as well.

The use of a weight matrix $\mathbf{W}$ is necessary to determine the vector $\mathbf{X}$ composed of the partial derivatives. This weight matrix allows to control numerical instabilities (division by small numbers) when the mesh is skewed.

The weight matrix commonly used in least squares problem is a left-preconditioning matrix,

$$\mathbf{W}^{-1}\mathbf{AX} = \mathbf{W}^{-1}\mathbf{B}, \tag{4.15}$$

with

$$\mathbf{W} = \begin{pmatrix} w_1 & & 0 \\ & \ddots & \\ 0 & & w_N \end{pmatrix}, \tag{4.16}$$

and

$$w_j = \frac{1}{\sqrt{\triangle x_{ij}^2 + \triangle y_{ij}^2 + \triangle z_{ij}^2}} \quad j = 1, \cdots, N. \tag{4.17}$$

It corresponds to a square and diagonal weight matrix of dimension corresponding to the number of neighboring element N of the current stencil. The notion of stencil will be introduced later.

This left-preconditioning allows to narrow the amplitude between the lines of the matrix system but not with the columns. This choice is made for instance in Chiapolino et al. (2017) [12]. However, the second derivatives are not required in [12]. The previous weight matrix is nonetheless useful in the present context. Indeed, the system may be under-determined and the second derivatives cannot be computed in that case. This situation may happen for instance on a boundary element of a 2D numerical domain. On such elements, only the "Level-Set" function ϕ is reconstructed at face centers with the least squares method detailed in Chiapolino et al. (2017) [12].

In the present context, first and second partial derivatives are present in Matrix $\mathbf{X}$. Those imply first (Δx) and second $(\Delta x)^2$ order space steps in Matrix $\mathbf{A}$. For this reason, a right-preconditioning is preferred. Hereby, the corresponding system reads,

$$\mathbf{A}\mathbf{W}^{-1}\mathbf{W}\mathbf{X} = \mathbf{B}, \tag{4.18}$$

where $\mathbf{W}$ is the new weight matrix. It is a square matrix of dimension 2 for a 1D problem, 6 for a 2D problem and 9 for a 3D problem. This matrix is yet to be defined.

Let us now consider,

$$\mathbf{Y} = \mathbf{W}\mathbf{X}. \tag{4.19}$$

The system to solve becomes,

$$\mathbf{A}\mathbf{W}^{-1}\mathbf{Y} = \mathbf{B} \quad \Leftrightarrow \quad \widetilde{\mathbf{A}}\mathbf{Y} = \mathbf{B} \quad \text{with} \quad \widetilde{\mathbf{A}} = \mathbf{A}\mathbf{W}^{-1}. \tag{4.20}$$

In this work, the matrix $\widetilde{\mathbf{A}}$ is computed with the help of a $\mathbf{QR}$ decomposition. $\mathbf{Q}$ is an orthogonal matrix ($\mathbf{Q}^T\mathbf{Q} = \underline{\underline{\mathbf{I}}}$) and $\mathbf{R}$ is an upper triangular matrix,

$$\widetilde{\mathbf{A}}\mathbf{Y} = \mathbf{B} \quad \text{becomes} \quad \mathbf{Q}\mathbf{R}\mathbf{Y} = \mathbf{B}, \quad \text{consequently} \quad \mathbf{R}\mathbf{Y} = \mathbf{Q}^T\mathbf{B}, \quad \text{finally} \quad \mathbf{Y} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{B}. \tag{4.21}$$

In this framework, $\mathbf{QR}$ decomposition is performed using Gram-Schmidt algorithm. The equality of Eqs. (4.19) and (4.21) yields,

$$\mathbf{W}\mathbf{X} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{B}, \tag{4.22}$$

and finally,

$$\mathbf{X} = \mathbf{W}^{-1}\mathbf{R}^{-1}\mathbf{Q}^T\mathbf{B}. \tag{4.23}$$

It is important to note that for non-moving meshes, the factors $\mathbf{W}^{-1}\mathbf{R}^{-1}\mathbf{Q}^T$ are computed once for all at the beginning of the computation, so that the whole least squares method only yields one matrix-vector product per element.

The choice of the weight matrix $\mathbf{W}$ is now addressed. Multiple options are possible. In this work, the following form is proposed,

$$\mathbf{W} = \begin{pmatrix} d_i^{max} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & d_i^{max} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & d_i^{max} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (d_i^{max})^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & (d_i^{max})^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & (d_i^{max})^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (d_i^{max})^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (d_i^{max})^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (d_i^{max})^2 \end{pmatrix}, \tag{4.24}$$

with $d_i^{max} = max\,(d_{ij})$, the maximum distance between the current cell center i and the center of the cell j, part of the stencil of i. The following 1D analysis justifies the choice of Matrix $\mathbf{W}$ (4.24).

Let us consider the 1D configuration depicted in Section 3.2, Fig. 5. The element i having only two neighbors $(i \pm 1)$, the corresponding least squares system reduces to,

$$\begin{cases} \phi_{i+1} = \phi_i + \Delta x \dfrac{\partial \phi_i}{\partial x} + \dfrac{(\Delta x)^2}{2}\dfrac{\partial^2 \phi_i}{\partial x^2}, \\ \phi_{i-1} = \phi_i - \Delta x \dfrac{\partial \phi_i}{\partial x} + \dfrac{(\Delta x)^2}{2}\dfrac{\partial^2 \phi_i}{\partial x^2}. \end{cases} \tag{4.25}$$

Matrix $\mathbf{A}$ then reads,

$$\mathbf{A} = \begin{pmatrix} \Delta x & \frac{(\Delta x)^2}{2} \\ -\Delta x & \frac{(\Delta x)^2}{2} \end{pmatrix}, \tag{4.26}$$

and the weight matrix $\mathbf{W}$ is,

$$\mathbf{W} = \begin{pmatrix} \Delta x & 0 \\ 0 & (\Delta x)^2 \end{pmatrix}. \tag{4.27}$$

Its inverse is consequently,

$$\mathbf{W}^{-1} = \begin{pmatrix} \frac{1}{\Delta x} & 0 \\ 0 & \frac{1}{(\Delta x)^2} \end{pmatrix}. \tag{4.28}$$

The matrix product $\mathbf{A}\mathbf{W}^{-1}$ yields,

$$\mathbf{A}\mathbf{W}^{-1} = \begin{pmatrix} \Delta x & \frac{(\Delta x)^2}{2} \\ -\Delta x & \frac{(\Delta x)^2}{2} \end{pmatrix} \begin{pmatrix} \frac{1}{\Delta x} & 0 \\ 0 & \frac{1}{(\Delta x)^2} \end{pmatrix} = \begin{pmatrix} 1 & 0.5 \\ -1 & 0.5 \end{pmatrix}, \tag{4.29}$$

where the components are of the same order of magnitude. No space step is present.

4.2. Notion of stencil

In this work, the stencil of an element gathers all neighboring elements having a common face or vertex with the cell of interest.

When only faces are common between the element and its neighbors, the set will be denoted as "direct" stencil, also known as von Neumann neighborhood of range 1.

When faces and vertices are common between the element and its neighbors, the stencil will be denoted as "extended", also known as Moore neighborhood. This configuration is slightly more complex but is sometimes necessary with unstructured meshes (Chiapolino et al. (2017) [12]). The two configurations are depicted in Fig. 9.

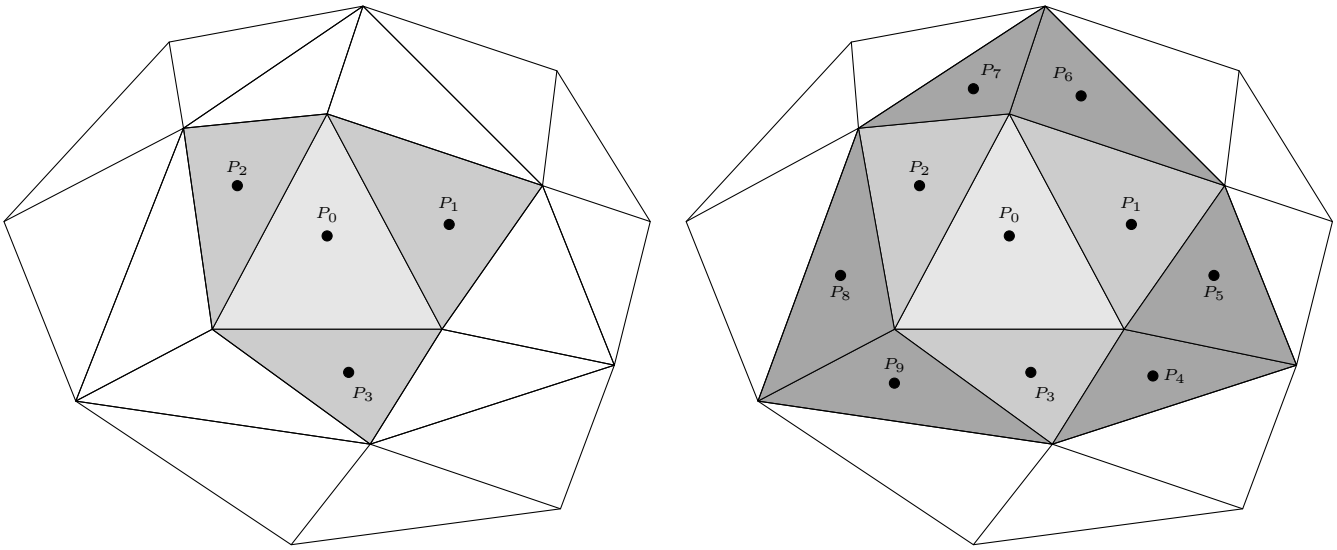

Figure 9: Schematic representation of the direct and extended stencil of the cell P_0 on an unstructured mesh made of triangles, for gradient computation. The cell of interest P_0 is represented as the shaded cell. On the left, only the direct neighbors are represented as the darker cells. On the right, the indirect neighbors are represented in addition as the darkest cells.

In the present context, it is important to note that the extended stencil is mandatory in order to ensure that the least squares system presents at least as many neighboring elements as unknown variables (gradient components of the "Level-Set" function and second partial derivatives).

4.3. Gradient limitation

In the presence of discontinuities, that will be the case with the choice of the present sharp "Level-Set" function, care is needed to avoid oscillations. With this aim the gradients are limited.

In this framework, the Barth and Jespersen (1989) [33] approach is employed. To avoid reconstructed solution at the face exceeding minimum or maximum values at cell centers, the gradient is scaled by factor Θ.

The reconstruction at the center of the face separating P_i and P_j "to the left" becomes,

$$\phi_L(P_{ij}) \simeq \phi(P_i) + \Theta_i \vec{r}_{ij}.\vec{\nabla}\phi(P_i) + \Delta t \frac{\partial \phi(P_i)}{\partial t}, \quad \vec{r}_{ij} = \overrightarrow{P_i P_{ij}}. \tag{4.30}$$

with

$$\Theta_i = min\left(\theta\left(\psi_{ij}\right)\right), \quad j \in neigh(i), \tag{4.31}$$

and

$$\psi_{ij} = \begin{cases} \frac{\phi^{max}-\phi_i}{2\left(\phi_{ij}^{n\,lim}-\phi_i\right)} & \text{if } \left(\phi_{ij}^{n\,lim}-\phi_i\right) > 0, \\ \frac{\phi^{min}-\phi_i}{2\left(\phi_{ij}^{n\,lim}-\phi_i\right)} & \text{if } \left(\phi_{ij}^{n\,lim}-\phi_i\right) < 0, \\ 1 & \text{if } \left(\phi_{ij}^{n\,lim}-\phi_i\right) = 0, \end{cases} \tag{4.32}$$

with $\phi_{ij}^{n\,lim} = \phi_i + \vec{r}_{ij}.\vec{\nabla}\phi_i$, the unlimited reconstruction solution and ϕ^{max}, ϕ^{min} respectively the maximum and minimum value between the current cell and all its direct neighbors. The same reasoning applies to the right side.

$\theta\left(\psi_{ij}\right)$ is limiter dependent. For instance,

$$\theta\left(\psi_{ij}\right) = max\Big[0, min(\beta\psi_{ij}, 1), min(\psi_{ij}, \beta)\Big], \tag{4.33}$$

gives the Minmod limiter [34] for $\beta = 1$ and the Superbee limiter [35] for $\beta = 2$.

However, those limiters result in excessive numerical dissipation. Indeed, the interface becomes significantly diffused, especially on unstructured meshes.

Interface sharpening is then achieved. An efficient method is obtained with the THINC approach, Shyue and Xiao, (2014) [36], Ii et al. (2014) [37]. This technique is based on a hyperbolic tangent reconstruction.

More recently, the Overbee compressible limiter was introduced in Chiapolino et al. (2017) [12] in the frame of two-phase flow modeling and diffuse interface methods. This limiter was precisely designed to lower numerical diffusion of the so-called "diffuse interfaces" by sharpening volume fraction profiles. The Overbee limiter showed enhanced capturing properties with 2-3 cells only in the interface zone when used in the frame of MUSCL-type schemes that are quite simple to implement in codes dealing with unstructured meshes.

The present contribution takes consequently advantage of the Overbee limiter. However, this limiter can only be used with Heaviside-type discontinuities, that is precisely the case with the present sharp "Level-Set" function. In its most general form, the Overbee limiter reads,

$$\theta\left(\psi_{ij}\right) = max\Big[0, min\Big[2, 2\psi_{ij}, max\big[min(2\psi_{ij}, \beta), min\{(2-\beta)\phi_{ij}+2(\beta-1), \phi_{ij}\}\big]\Big]\Big], \quad 1 \leq \beta \leq 2. \tag{4.34}$$

For $\beta = 1$, the Overbee limiter reduces to the upper boundary of the second-order TVD region corresponding to the Superbee limiter. For $\beta = 2$, it increases to the upper boundary of the first-order TVD region (see Fig. 10).

The parameter β controls the amount of artificial compression sharpening the discontinuity while satisfying the TVD constraint. Its maximum value is $\beta = 2$. In this case Eq. (4.34) simplifies to,

$$\theta\left(\phi_{ij}\right) = max\Big[0, min\big[2\phi_{ij}, 2\big]\Big]. \tag{4.35}$$

A graphical representation of the Overbee and Superbee limiters is depicted in Fig. 10. It is based on the Total Variation Diminishing (TVD) theory, which is essential for the design of oscillation free numerical schemes. For details or discussions related to the TVD notion, the reader is referred to [12], [29], [30], [38], [39], [40], [41], [42] for example.

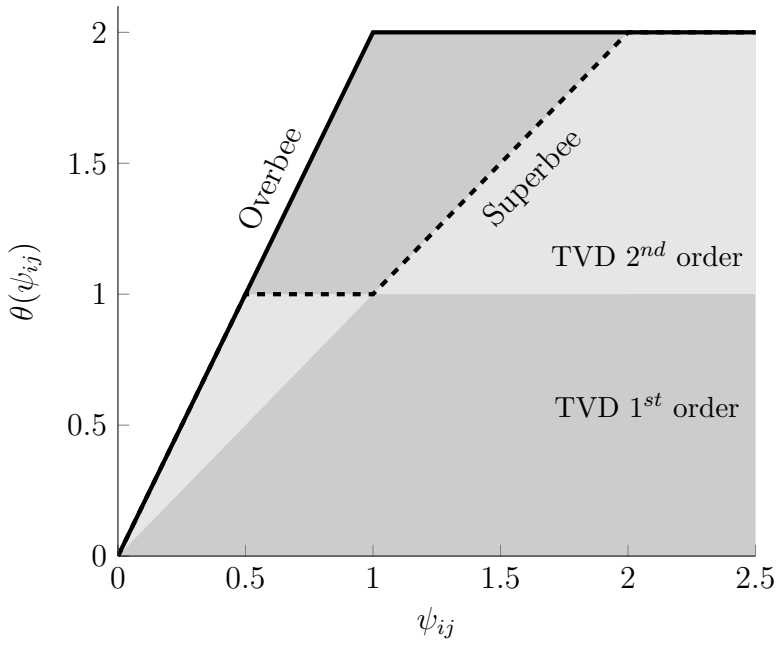

Figure 10: Graphical representation of the Overbee and Superbee limiters. The dark gray shaded region represents the region of first-order TVD methods. The light gray shaded region represents the region of second-order TVD methods. The solid line represents the Overbee limiter that lies along the upper boundary of the first-order TVD zone. The dashed line represents the Superbee limiter that lies along the upper boundary of the second-order TVD zone. The upper first-order region can only be used with Heaviside-type discontinuities. This is exactly the shape of the chosen discontinuous "Level-Set" function. The other regions (second-order and lower first-order) can be used with continuous and discontinuous functions.

The following one-dimensional example of the transport of a Heaviside function, depicted in Fig. 11, shows capabilities of the Overbee limiter. Note that the transport speed u_0 is constant and the 1D "Level-Set" equation transforms consequently to a conservative equation in this specific context,

$$\frac{\partial \phi}{\partial t} + \frac{\partial \left(\phi u_0\right)}{\partial x} = 0. \tag{4.36}$$

Note also that the second partial derivatives are not required in this transport context. Only the first derivatives are needed to perform the MUSCL-type reconstruction at cell faces. The 1D reduction of the least squares method corresponds to the centered approximation,

$$\frac{\partial \phi_i}{\partial x} = \frac{1}{2\Delta x}\left(\phi_{i+1} - \phi_{i-1}\right), \tag{4.37}$$

with i denoting the current element and Δx the space step (see Fig. 4).

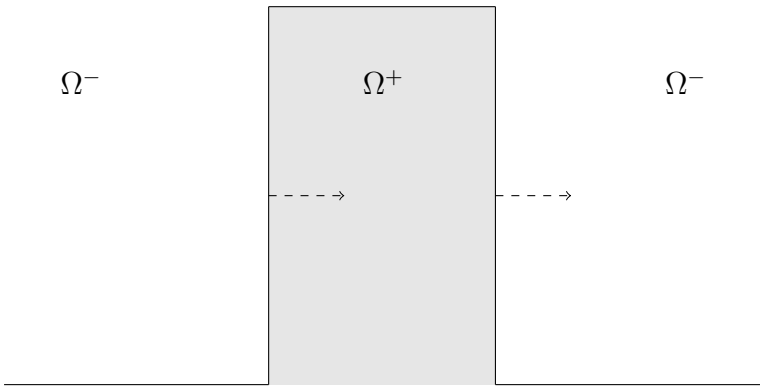

Figure 11: Schematic representation of a simple 1D test of a Heaviside profile, transported to the right with speed u_0. However, numerical diffusion results in a smoothing of the initial profile. The MUSCL-type scheme and Overbee limiter allow to control this artificial dissipation.

Results are provided in Fig. 12 along with the exact solution.

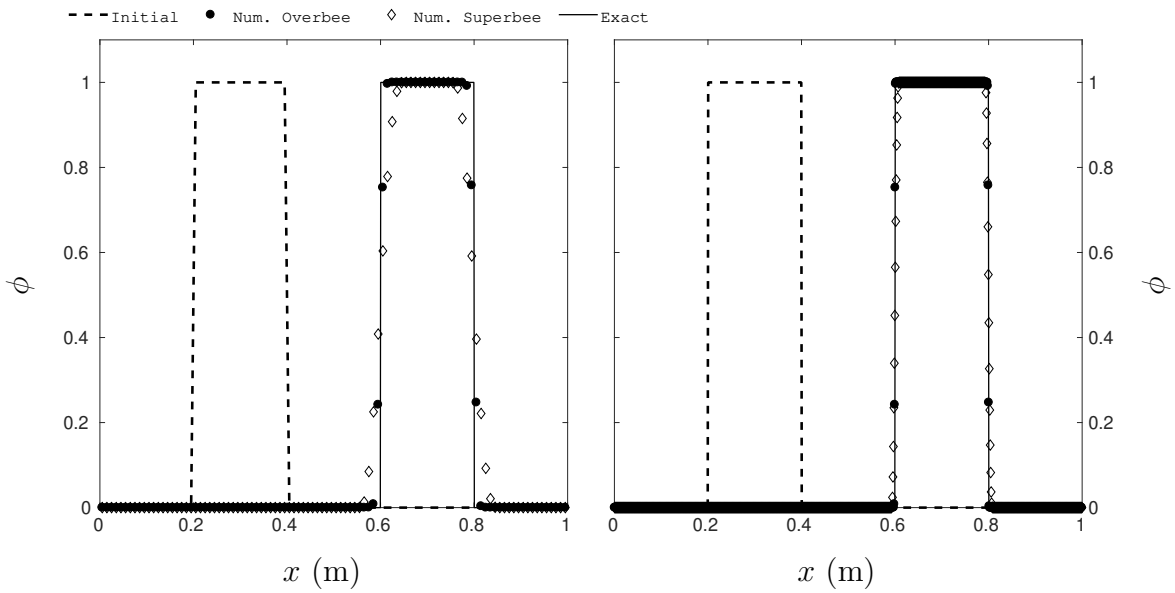

Figure 12: Comparison of the Overbee (full circle symbols •) and Superbee (diamond symbols ⋄) limiters for the transport of a Heaviside function. The advection speed is $u_0 = 100$ m/s. The dashed lines represent the initial condition. The full lines represent the exact solution (plotted with 1000 points). The final time is $t = 4$ ms and CFL= 0.8 is used. In the figure on the left, 100 elements are used. In the figure on the right, 1000 elements are used. Overbee captures the discontinuity with two points only for both mesh resolutions.

The Overbee limiter allows to capture the discontinuities with two mesh points. This is a consequence of the upper first-order TVD zone (see Fig. 10). The Superbee limiter results in much more numerical dissipation around the discontinuities. However, the Superbee limiter can be used with discontinuous and smooth functions unlike the Overbee limiter that is designed only for Heaviside-type functions. For more details and illustrations about the Overbee limiter, the reader is referred to Chiapolino et al. (2017) [12], Carmouze et al. (2018) [10] and Furfaro et al. (2020) [27].

The properties of the Overbee limiter will therefore be used for the resolution of the HJ equation.

The extension of the proposed numerical scheme through the MUSCL-type method provides reconstructed states (left and right) at face centers regarding the "Level-Set" function ϕ resulting in more accurate results.

The gradients $\left(\vec{\nabla}\phi_L, \vec{\nabla}\phi_R\right)$ are also reconstructed at face centers in order to obtain a more accurate normal vector of the moving interface $\vec{n}^*_{F,ij}$. The gradient components are consequently scaled by factors Θ as well.

The reconstruction at the center of the face separating P_i and P_j "to the left" becomes,

$$\begin{cases} \dfrac{\partial \phi_L^n(P_{ij})}{\partial x} = \dfrac{\partial \phi^n(P_i)}{\partial x} + \Theta_{i,x}\left(\vec{r}_{x,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x^2} + \vec{r}_{y,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x\partial y} + \vec{r}_{z,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x\partial z}\right), \\ \dfrac{\partial \phi_L^n(P_{ij})}{\partial y} = \dfrac{\partial \phi^n(P_i)}{\partial y} + \Theta_{i,y}\left(\vec{r}_{x,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x\partial y} + \vec{r}_{y,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial y^2} + \vec{r}_{z,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial y\partial z}\right), \\ \dfrac{\partial \phi_L^n(P_{ij})}{\partial z} = \dfrac{\partial \phi^n(P_i)}{\partial z} + \Theta_{i,z}\left(\vec{r}_{x,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial x\partial z} + \vec{r}_{y,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial y\partial z} + \vec{r}_{z,i,ij}\dfrac{\partial^2\phi^n(P_i)}{\partial z^2}\right), \end{cases} \tag{4.38}$$

with $\vec{r}_{x,i,ij}$, $\vec{r}_{y,i,ij}$ and $\vec{r}_{z,i,ij}$ the projected distance $\overrightarrow{P_iP_{ij}}$ along axes x, y and z. The same reasoning applies to the right side.

Basically, the method of Barth and Jespersen (1989) [33] is used for each component of the gradient. This demands to compute as many Θ factors as dimensions (1, 2 or 3).

Similarly to the reconstruction of the "Level-Set" function, the limitation factors are,

$$\Theta_{i,x,y,z} = min\left(\theta\left(\psi_{ij,x,y,z}\right)\right), \quad j \in neigh(i), \tag{4.39}$$

and

$$\psi_{ij,x} = \begin{cases} \dfrac{\frac{\partial\phi}{\partial x}^{max} - \frac{\partial\phi_i}{\partial x}}{2\left(\frac{\partial\phi_{ij}^{n\,lim}}{\partial x} - \frac{\partial\phi_i}{\partial x}\right)} & \text{if } \left(\dfrac{\partial\phi_{ij}^{n\,lim}}{\partial x} - \dfrac{\partial\phi_i}{\partial x}\right) > 0, \\ \dfrac{\frac{\partial\phi}{\partial x}^{min} - \frac{\partial\phi_i}{\partial x}}{2\left(\frac{\partial\phi_{ij}^{n\,lim}}{\partial x} - \frac{\partial\phi_i}{\partial x}\right)} & \text{if } \left(\dfrac{\partial\phi_{ij}^{n\,lim}}{\partial x} - \dfrac{\partial\phi_i}{\partial x}\right) < 0, \\ 1 & \text{if } \left(\dfrac{\partial\phi_{ij}^{n\,lim}}{\partial x} - \dfrac{\partial\phi_i}{\partial x}\right) = 0, \end{cases} \tag{4.40}$$

$$\psi_{ij,y} = \begin{cases} \dfrac{\frac{\partial\phi}{\partial y}^{max} - \frac{\partial\phi_i}{\partial y}}{2\left(\frac{\partial\phi_{ij}^{n\,lim}}{\partial y} - \frac{\partial\phi_i}{\partial y}\right)} & \text{if } \left(\dfrac{\partial\phi_{ij}^{n\,lim}}{\partial y} - \dfrac{\partial\phi_i}{\partial y}\right) > 0, \\ \dfrac{\frac{\partial\phi}{\partial y}^{min} - \frac{\partial\phi_i}{\partial y}}{2\left(\frac{\partial\phi_{ij}^{n\,lim}}{\partial y} - \frac{\partial\phi_i}{\partial y}\right)} & \text{if } \left(\dfrac{\partial\phi_{ij}^{n\,lim}}{\partial y} - \dfrac{\partial\phi_i}{\partial y}\right) < 0, \\ 1 & \text{if } \left(\dfrac{\partial\phi_{ij}^{n\,lim}}{\partial y} - \dfrac{\partial\phi_i}{\partial y}\right) = 0, \end{cases} \tag{4.41}$$

$$\psi_{ij,z} = \begin{cases} \frac{\frac{\partial \phi}{\partial z}^{max} - \frac{\partial \phi_i}{\partial z}}{2\left(\frac{\partial \phi_{ij}^{n\,lim}}{\partial z} - \frac{\partial \phi_i}{\partial z}\right)} & \text{if } \left(\frac{\partial \phi_{ij}^{n\,lim}}{\partial z} - \frac{\partial \phi_i}{\partial z}\right) > 0, \\ \frac{\frac{\partial \phi}{\partial z}^{min} - \frac{\partial \phi_i}{\partial z}}{2\left(\frac{\partial \phi_{ij}^{n\,lim}}{\partial z} - \frac{\partial \phi_i}{\partial z}\right)} & \text{if } \left(\frac{\partial \phi_{ij}^{n\,lim}}{\partial z} - \frac{\partial \phi_i}{\partial z}\right) < 0, \\ 1 & \text{if } \left(\frac{\partial \phi_{ij}^{n\,lim}}{\partial z} - \frac{\partial \phi_i}{\partial z}\right) = 0, \end{cases} \tag{4.42}$$

with

$$\begin{cases} \dfrac{\partial \phi_L^{n\,lim}(P_{ij})}{\partial x} = \dfrac{\partial \phi^n(P_i)}{\partial x} + \left(\vec{r}_{x,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial x^2} + \vec{r}_{y,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial x \partial y} + \vec{r}_{z,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial x \partial z} \right), \\ \dfrac{\partial \phi_L^{n\,lim}(P_{ij})}{\partial y} = \dfrac{\partial \phi^n(P_i)}{\partial y} + \left(\vec{r}_{x,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial x \partial y} + \vec{r}_{y,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial y^2} + \vec{r}_{z,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial y \partial z} \right), \\ \dfrac{\partial \phi_L^{n\,lim}(P_{ij})}{\partial z} = \dfrac{\partial \phi^n(P_i)}{\partial z} + \left(\vec{r}_{x,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial x \partial z} + \vec{r}_{y,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial y \partial z} + \vec{r}_{z,i,ij} \dfrac{\partial^2 \phi^n(P_i)}{\partial z^2} \right), \end{cases} \tag{4.43}$$

the unlimited reconstruction solutions and $\frac{\partial \phi^{max}}{\partial x,y,z}$, $\frac{\partial \phi^{min}}{\partial x,y,z}$ respectively the maximum and minimum value between the current cell and all its direct neighbors.

This corresponds to Barth and Jespersen (1988) [33] approach applied to the gradient components.

The present Heaviside "Level-Set" function ϕ takes advantage of the Overbee limiter. However, the gradient does not present a Heaviside profile. Consequently, the Overbee limiter cannot be used.

The choice of the limiter function is then reduced to the second-order TVD zone but not the upper region of the first-order zone, see Fig. 10. The lower region of the first-order zone can be used as well but provides too much numerical diffusion.

Hereby, the Superbee limiter is used for the gradient component reconstruction only. This limiter lies along the upper boundary of the second-order TVD region as seen in Fig. 10.

The use of the compressive Overbee function on variables presenting a Heaviside profile and a conventional limiter such as Superbee on other variables does not cause specific difficulties. This particular treatment has been used in Carmouze et al. (2018) [10] where transport of a discontinuous "Level-Set" function is addressed in a two-phase flow context. In this framework, the Overbee limiter is used for the discontinuous "Level-set" function only and another limiter is used for the other flow variables such as the density for example.

The 1D test case of fronts propagating along the normal vector depicted in Section 3.2 (Fig. 5) is now repeated with the present MUSCL-type method. The least squares method for gradient computation reduces to the centered approximations,

$$\begin{cases} \dfrac{\partial \phi_i}{\partial x} = \dfrac{\phi_{i+1} - \phi_{i-1}}{2\Delta x}, \\ \dfrac{\partial^2 \phi_i}{\partial x^2} = \dfrac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2}. \end{cases} \tag{4.44}$$

Results are shown in Fig. 13 as well as the exact solution.

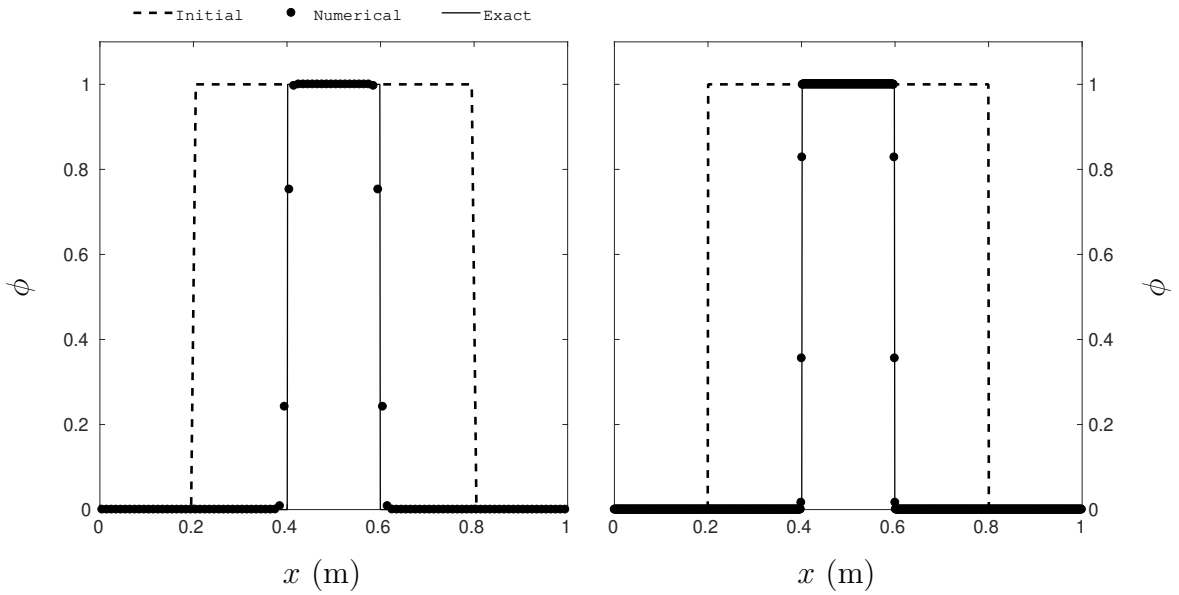

Figure 13: Comparison of the computed solution with the present Godunov-type second-order scheme and corresponding Riemann solver to the exact solution. The Overbee limiter is used for the "Level-Set" function ϕ and the Superbee limiter is used for its gradient. The speed module along the normal direction of the front is $u_0 = 1$ m.s^{-1}. The final time is $t = 0.2$ s and CFL= 0.8 is used. In the figure on the left, 100 elements are used. In the figure on the right, 1000 elements are used. The full circle symbols • represent the corresponding results. The dashed lines represent the initial condition. The full lines represent the exact solution (plotted with 1000 points).

Results are in excellent agreement with the exact solution and the Overbee limiter, used with the discontinuous "Level-Set" function ϕ, allows to capture the discontinuities with 2 mesh points only regardless of the mesh resolution.

5. Computation of the interfacial area and corresponding volume

Interfacial area computation is important for many applications. The proposed method may be used to determine this interfacial area as well as the corresponding volume with the help of the interfacial normal vector $\vec{n}_F$.

5.1. Conventional relation

The determination of the interfacial area A_I is a key problem in combustion and two-phase flow modeling (Drew and Passman (2006) [43]) as well as many other physical and technical areas. It is usually defined as,

$$A_I = -\int_V \vec{\nabla}\chi \cdot \vec{n}_F dV, \tag{5.1}$$

where χ is a characteristic function defining presence of the media, *i.e.* $\chi = 1$ in a given medium and $\chi = 0$ outside. Figure 14 presents a very simple 2D configuration allowing for easier analyses.

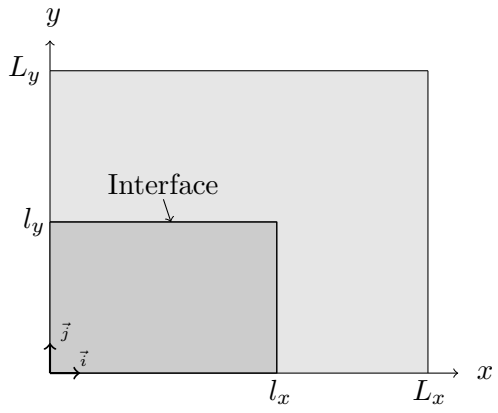

Figure 14: Analysis of the interfacial area computation on a simple example. The domain is a square of dimensions (L_x, L_y). An interface separates two media represented as the light and dark zones.

In the present context, Eq. (5.1) transforms to,

$$\int_V \vec{\nabla}\chi \cdot \vec{n}_F dV = \int_0^{L_y} \int_0^{L_x} \left(\frac{\partial \chi}{\partial x}\vec{i} + \frac{\partial \chi}{\partial y}\vec{j} \right) \cdot \vec{n}_F dx dy. \tag{5.2}$$

After some algebraic manipulations, the interfacial area, corresponding to the perimeter in the present 2D context, reads:

$$-\int_V \vec{\nabla}(\chi) \cdot \vec{n}_F dV = l_x + l_y = A_I. \tag{5.3}$$

Equation (5.1) is thus analytically correct. Note that in a two-phase flow context, the interfacial normal reads $\vec{n}_F = -\frac{\vec{\nabla}\chi}{|\vec{\nabla}\chi|}$. Consequently, Eq. (5.1) may be rewritten under the following form,

$$A_I = \int_V |\vec{\nabla}\chi| dV. \tag{5.4}$$

A numerical approximation of Eq. (5.4) is,

$$A_I \simeq \sum_i |\vec{\nabla}\chi_i| V_i, \tag{5.5}$$

where i denotes the index of the elements on the whole numerical domain.

In the present context, the "Level-Set" function ϕ is the analogue of the characteristic function χ. The corresponding interfacial area consequently reads,

$$A_I = \int_V |\vec{\nabla}\phi| dV \simeq \sum_i |\vec{\nabla}\phi_i| V_i. \tag{5.6}$$

5.2. Another formulation

Another formulation is worth considering, more suitable for discrete approximations,

$$A_I = \sum_{interfacial\ faces} |\vec{n}^*_{F,ij} \cdot \vec{n}_{ij}| \times S_{ij} = \sum_{interfacial\ faces} \left| \frac{\vec{\nabla}\phi^*_{ij}}{|\vec{\nabla}\phi^*_{ij}|} \cdot \vec{n}_{ij} \right| \times S_{ij}. \tag{5.7}$$

Relation (5.7) consists in a projection of cell's face's surface S_{ij} onto the interface $\partial\Omega$, as shown in Fig. 15. The dot product $\vec{n}^*_{F,ij} \cdot \vec{n}_{ij} = \frac{\vec{\nabla}\phi^*_{ij}}{|\vec{\nabla}\phi^*_{ij}|} \cdot \vec{n}_{ij}$ corresponds to the dot product of the interface vector $\vec{n}^*_{F,ij}$ and the cell face vector $\vec{n}_{ij}$.

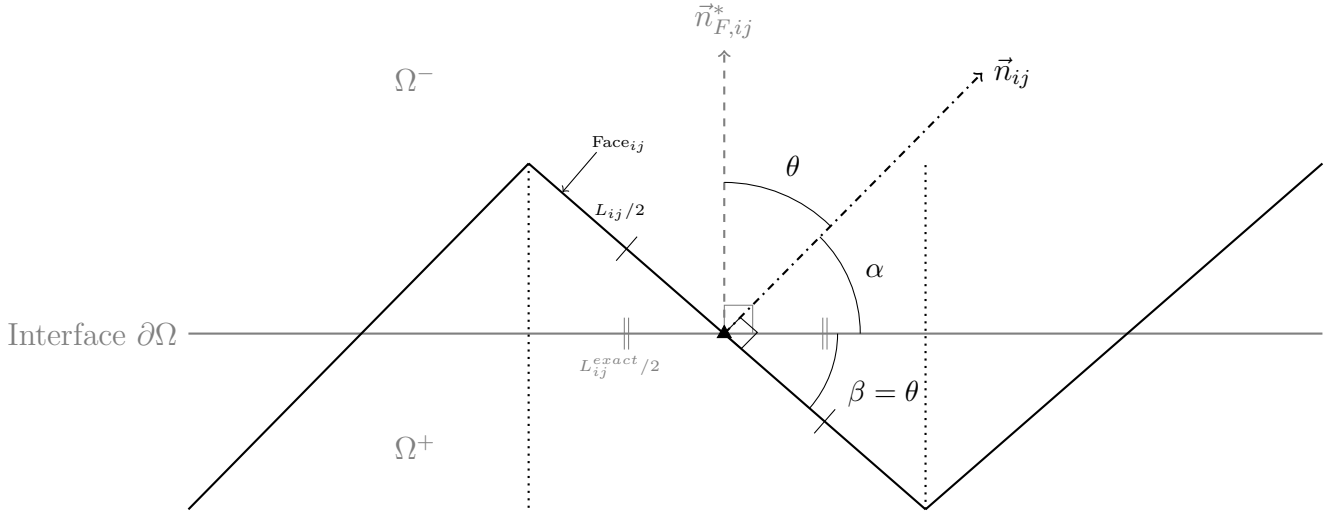

Figure 15: Analysis of the interfacial area computation on a simple example. The interface is represented as the light solid line. The faces of the elements separating one region from another are interfacial faces and are represented as dark solid lines. The center of the face ij is represented by the symbol ▲. The dot product $|\vec{n}^*_{F,ij} \cdot \vec{n}_{ij}|$ projects the cell's face's surface S_{ij} (L_{ij} in 2D) onto the interface $\partial\Omega$.

In Fig. 15, face ij separates an element i presenting a "Level-Set" function ϕ_i corresponding to the fist region (Ω^+) and an element j presenting a "Level-Set" function ϕ_j corresponding to the second one (Ω^-). Face ij is consequently an "interfacial" face.

The situation depicted in Fig. 15 involves $\alpha = \frac{\pi}{2} - \theta$ and $\beta = \frac{\pi}{2} - \alpha = \theta$. In the present 2D configuration, the dot product multiplied by the surface of the face ij, corresponding to its length L_{ij}, reads:

$$\left(\vec{n}^*_{F,ij} \cdot \vec{n}_{ij}\right) \times S_{ij} = |\vec{n}^*_{F,ij}| \times |\vec{n}_{ij}| \times \cos(\theta) \times S_{ij} = \cos(\theta) \times L_{ij}. \tag{5.8}$$

Besides, the cosine function expresses in the present situation as,

$$\cos(\theta) = \cos(\beta) = \frac{L_{ij}^{exact}/2}{L_{ij}/2} = \frac{L_{ij}^{exact}}{L_{ij}}. \tag{5.9}$$

Inserting the cosine function (5.9) into the dot product relation (5.8) yields,

$$(\vec{n}_{F,ij} \cdot \vec{n}_{ij}) \times S_{ij} = L_{ij}^{exact}. \tag{5.10}$$

The exact interfacial area corresponding to the perimeter in the 2D configuration is then recovered. Note that the present situation (Fig. 15) is schematic and tends to an ideal configuration (the center of the face ij separates half of the exact length). However, practical situations are expected to tend to Fig. 15, especially under mesh refinement. As it is shown in the following section, results obtained with this approach are in excellent agreement with analytical solutions.

We then have in hands two relations (Eqs. (5.6) and (5.7)) to compute the interfacial area. The determination of the volume of the medium of interest is easier to compute and may

be obtained as the sum of all the volumes of the elements presenting a "Level-Set" function corresponding to the region of interest.

Indeed, when medium 1 is the one of interest, its volume is given by:

$$V_1 = \int_V \chi_1 dV. \tag{5.11}$$

The direct application of this last relation with the situation depicted in Fig. 14 yields,

$$V_1 = \int_{V_1} 1\, dV + \int_{V_2} 0\, dV = V_1. \tag{5.12}$$

In the present HJ context, the "Level-Set" function ϕ is initialized as a Heaviside function. The corresponding volume of interest is consequently directly approximated as,

$$V_{\Omega^\pm} \simeq \sum_{\phi_i \in\, \Omega^\pm} V_i. \tag{5.13}$$

In relation (5.13), the volume Ω^+ is determined with the help of the elements presenting $\phi_i > \phi_I$. This relation is then an approximation of the exact volume.

Before providing computed results, it is worth highlighting the analogy between the surface and volume relations, Eqs. (5.4) and (5.11), and those used in the signed-distance-function context, Eqs. (2.11) and (2.12), see Section 2. Despite the advantage of the signed-distance function related to gradient computations, a Heaviside and Dirac delta functions are necessary to compute the corresponding interfacial area and volume.

6. Multidimensional results

The proposed method is now tested on both 2D and 3D test cases. A common example of an interface moving along its normal vector appears in the propagation of a combustion front. Among the straightforward examples is the regression of a propellant grain, reminiscent of interior ballistic or rocket engineering. Knowledge of its burning surface is of utmost importance for the aforementioned applications [44].

Grain regressions appear as excellent tests to assess the present method, solving the HJ equation and computing the interfacial area and volume. Indeed, propellant grains are made from very simple geometries [45] and analytical interfacial areas and volumes are available [46]. Besides, time evolution of the interfacial area and volume presents multiple and different stages, making relevant benchmarks to assess.

The following multidimensional test cases consist in a cylinder of initial radius R_0 presenting multiple perforations. The perforations are cylindrical as well and of initial radius r_0. Such geometries are known as the B7T (7 perforations) and B19T (19 perforations) solid propellants [47], [48]. The web thickness, that defines the smallest thickness of the initial propellant grain reads: $w_0 = (R_0 - 3r_0)/2$ for the B7T and $w_0 = \left(1 + \sqrt{2} - \sqrt{3}\right)(R_0 + r_0)/2 - 2r_0$ for the B19T.

The respective geometries are generated with Gmsh [49], a software provided under the terms of the GNU General Public License (GPL). Unstructured meshes made of triangular or tetrahedral elements are used. The interface speed remains $u_0 = 1$ m.s^{-1} and the CFL number is 0.8. The computations are done with the MUSCL-type scheme. The Overbee limiter is used for the Level-function ϕ and the Superbee limiter is used for its gradient $\vec{\nabla}\phi$.

6.1. Two-dimensional results

In Fig. 16, the method is applied to a 2D regression/combustion of the B7T grain. The computed interfacial areas (Eq. (5.7) referred as "Geometry" and Eq. (5.6) referred as "Conventional") are compared to the analytical solution (referred as "Exact") in Fig. 17. The computed solid volume (Eq. (5.13)) is compared to the exact solution as well.

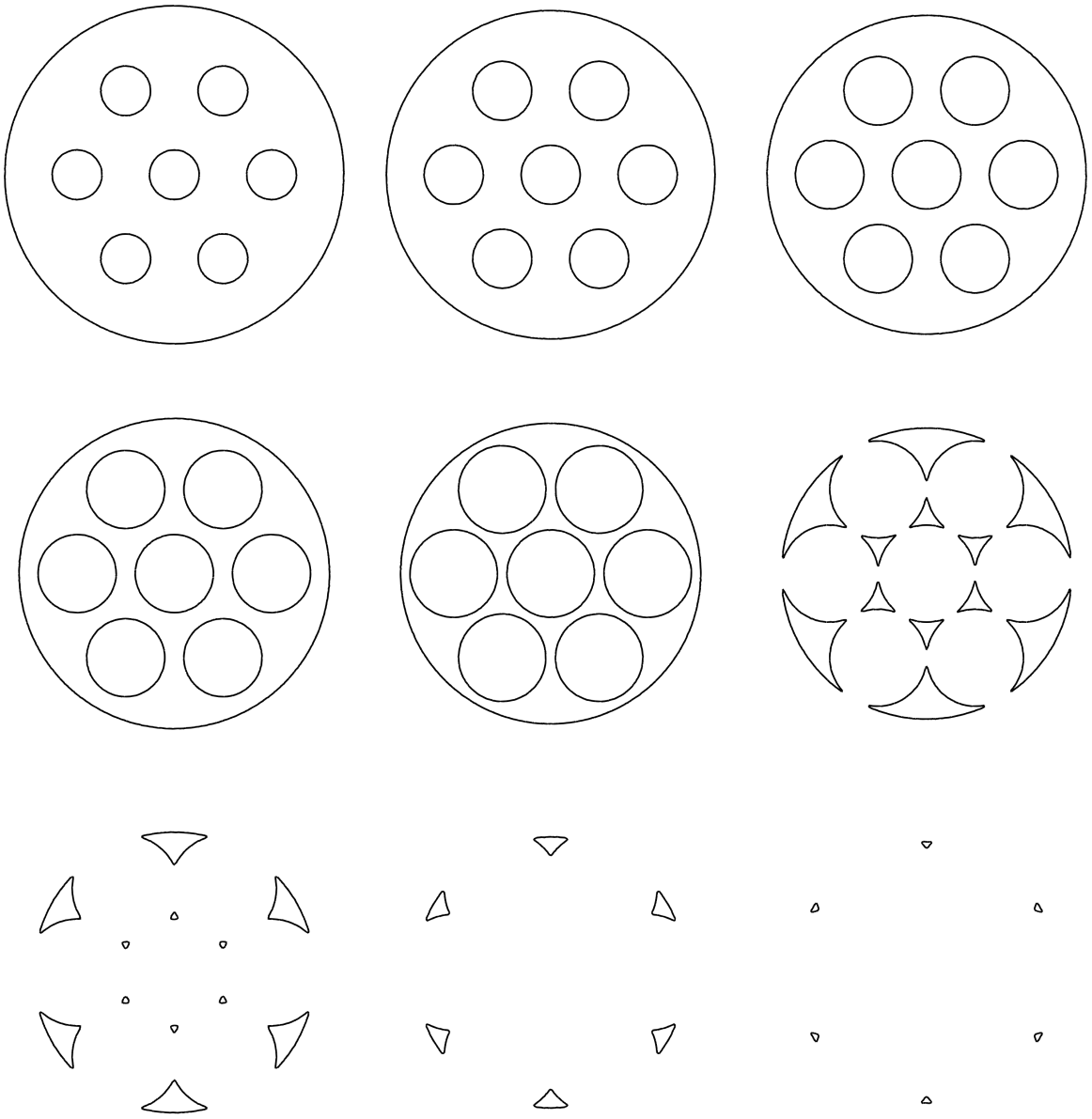

Figure 16: 2D regression of the B7T grain. The $\phi = 0.5$ isocontour corresponding to the interface level ϕ_I is presented. The front propagates at speed $u_0 = 1$ m.s^{-1} along its normal vector field. The grain consists in a cylinder of initial radius $R_0 = 1.75$ mm. The perforations are cylindrical as well and of initial radius $r_0 = 0.2$ mm. The initial web is $w_0 = 0.575$ mm. The solutions are given at times $t = 0.05$ ms, $t = 0.0976$ ms, $t = 0.145$ ms, $t = 0.193$ ms, $t = 0.240$ ms, $t = 0.288$ ms, $t = 0.338$ ms, $t = 0.386$ ms and $t = 0.433$ ms. The mesh is composed of about 50.000 triangular elements in the grain, generated by the "Delaunay" algorithm of the Gmsh software. The computation is done with the MUSCL-type scheme. The Overbee limiter is used for the Level-Set function ϕ and the Superbee limiter is used for its gradient $\vec{\nabla}\phi$. The CFL number is 0.8.

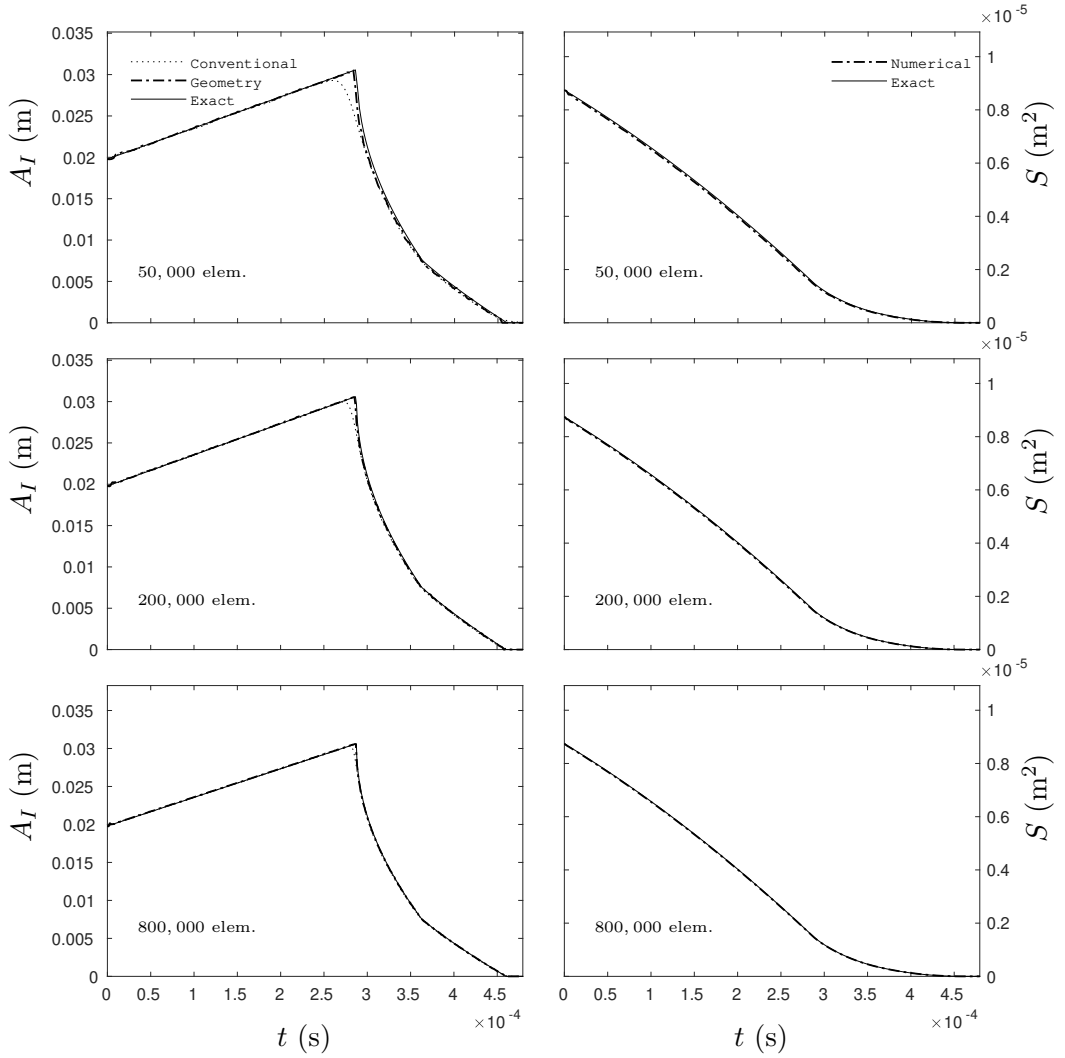

Figure 17: Comparison of the computed interfacial area (perimeter in 2D) and volume versus the exact solutions during the regression of the 2D B7T grain (Fig. 16). Three mesh resolutions are used in this figure, generated by the "Delaunay" algorithm of the Gmsh software. The volume (surface in 2D) computed with Eq. (5.13) is presented in the graphs on the right. In the graphs on the left, the interfacial area computed with the geometry-based relation (Eq. (5.7)) and with the conventional relation (Eq. (5.6)) is presented.

Figure 17 indicates that the proposed method and corresponding interfacial area and volume computations converge towards the analytical solutions. The results are indeed in excellent agreement with the exact solutions. However, interfacial area computation converges much faster with the geometry-based relation (Eq. (5.7)). Indeed, computed interfacial area from the convention of Relation (5.6) converges slowly while the interfacial area computed by the geometry-based relation (Eq. (5.7)) yields excellent results with the coarsest mesh. This feature is important with 3D computations, reducing significantly the number of elements required to obtain reasonable results.

6.2. Three-dimensional results

In Fig. 18, 3D results are provided. Those consist in the regression of the B19T solid propellant grain.

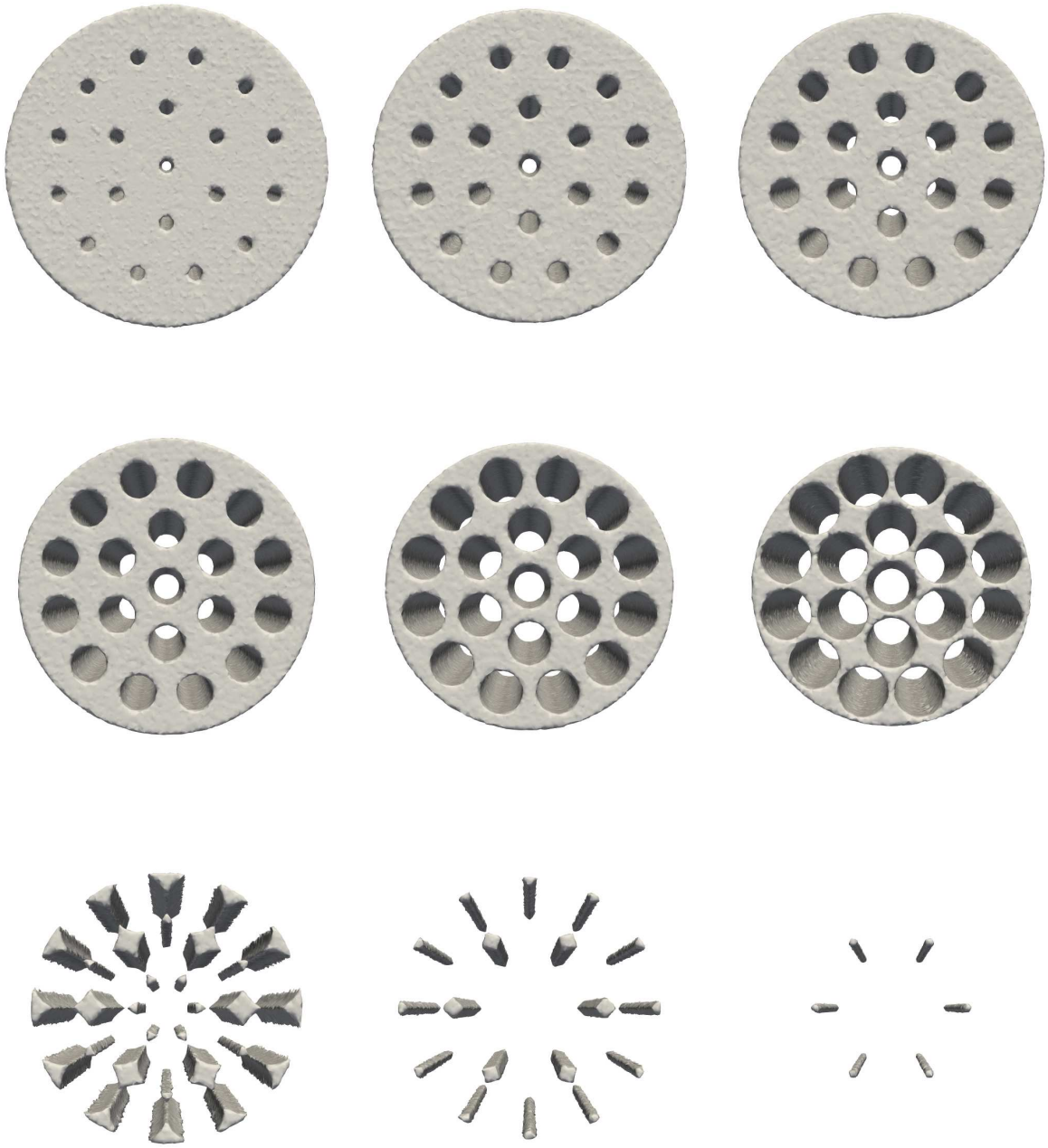

Figure 18: Regression of the B19T grain. The $\phi = 0.5$ isocontour corresponding to the interface level ϕ_I is presented. The front propagates at speed $u_0 = 1$ m.s^{-1} along its normal vector field. The grain consists in a cylinder of initial radius $R_0 = 7.155$ mm. The perforations are cylindrical as well and of initial radius $r_0 = 0.155$ mm. The grain is initially 16.1 mm long. The initial web is $w_0 = 2.18$ mm. The solutions are given at times $t = 0.16$ ms, $t = 0.32$ ms, $t = 0.48$ ms, $t = 0.64$ ms, $t = 0.80$ ms, $t = 0.96$ ms, $t = 1.12$ ms, $t = 1.28$ ms and $t = 1.44$ ms. The mesh is composed of about 2.5 million tetrahedral elements in the grain, generated by the "Frontal" algorithm of the Gmsh software. The computation is done with the MUSCL-type scheme. The Overbee limiter is used for the Level-Set function ϕ and the Superbee limiter is used for its gradient $\vec{\nabla}\phi$. The CFL number is 0.8.

Figure 19 provides the corresponding burning surface and solid volume over time.

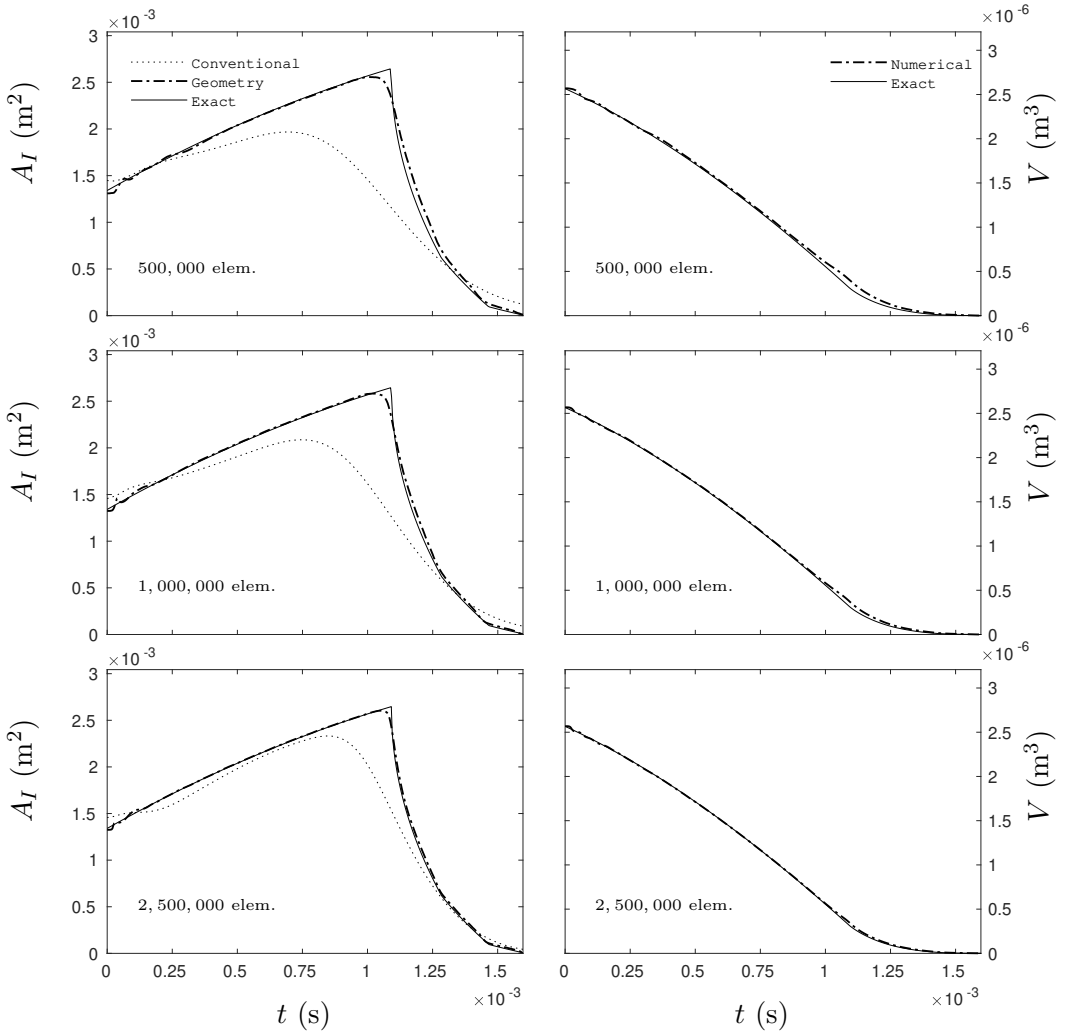

Figure 19: Comparison of the computed interfacial area and volume versus the exact solutions during the regression of the B19T grain (Fig. 18). Three mesh resolutions are used in this figure, generated by the "Frontal" algorithm of the Gmsh software. The volume computed with Eq. (5.13) is presented in the graphs on the right. In the graphs on the left, the interfacial area computed with the geometry-based relation (Eq. (5.7)) and with the conventional relation (Eq. (5.6)) is presented.

The geometry-based relation (Eq. (5.7)) provides one more time very reasonable results with the coarsest mesh (top graphs of Fig. 19). The agreement with the analytical solution becomes very good when the mesh is refined (middle and bottom graphs of Fig. 19), illustrating the capabilities of the present method and interfacial area computation. The volume is also in very good agreement with the exact solution.

However, surface computation with the conventional relation (Eq. (5.6)) presents convergence difficulties.

In Fig. 20, the mesh is refined two more times.

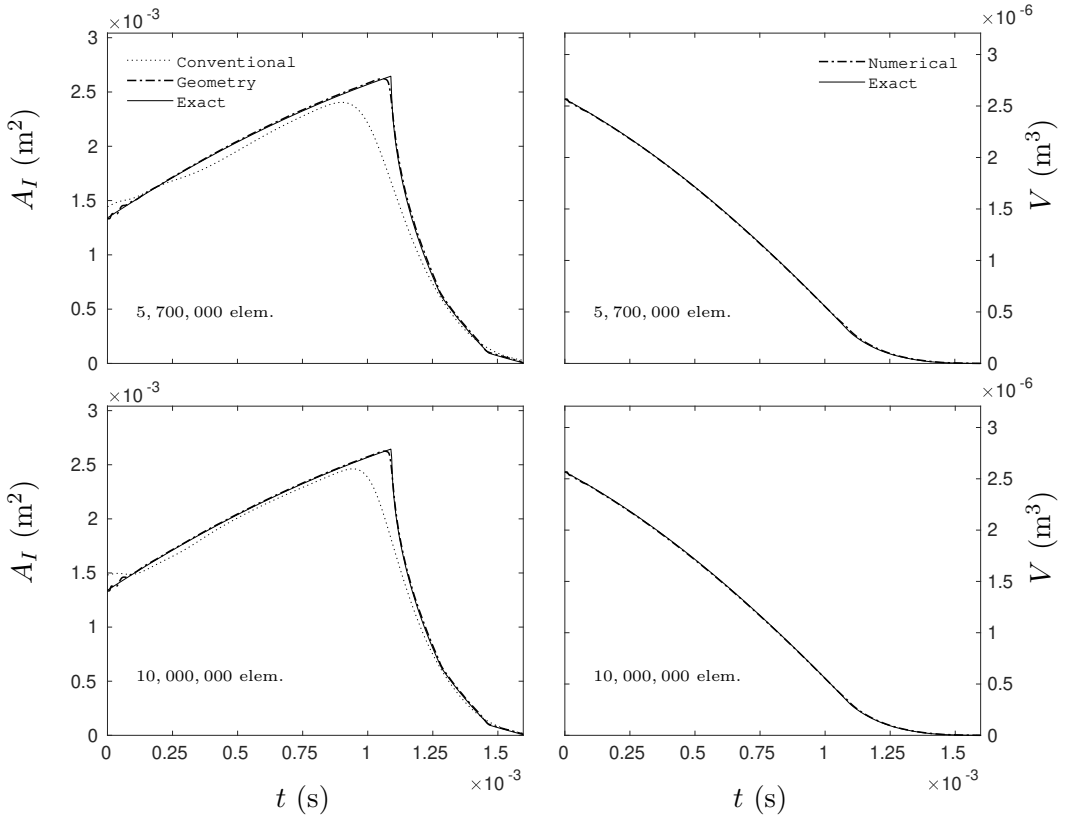

Figure 20: Comparison of the computed interfacial area and volume versus the exact solutions during the regression of the B19T grain (Figs. 18, 19). The mesh is refined two more times. The interfacial area computed via the geometry-based relation (Eq. (5.7)), and volume computed with Eq. (5.13) are in excellent agreement with the exact solution. The conventional interfacial area computation (Eq. (5.6)) tends to converge towards the analytical solution but is still perfectible despite the fine mesh resolutions.

The interfacial area computed via the geometry-based relation (Eq. (5.7)) is in excellent agreement with the exact solution. The volume, computed with Eq. (5.13), is excellent as well.

640 The interfacial area provided by conventional relation (Eq. (5.6)) tends to converge towards the analytical solution but is still perfectible despite the fine mesh resolutions. The proposed geometry-based relation (Eq. (5.7)) is consequently of great interest.

Mesh convergence analysis has been carried out with meshes ranging from $500,000$ elements to 10 million elements. The error is defined by Relation (6.1),

$$\overline{\text{Error }\%} = \frac{1}{t_{end}} \int_0^{t_{end}} (\text{Error }\%)\, dt = \frac{1}{t_{end}} \int_0^{t_{end}} \left(100 \times \frac{|\text{Numerical} - \text{Exact}|}{\text{Exact}} \right) dt, \qquad (6.1)$$

645 where t_{end} is the last simulation time considered (slightly below the final time in order to ensure a defined error percentage (Error %)). The numerical integration is done with the help of the trapezoidal rule. Corresponding results are reported in Fig. 21.

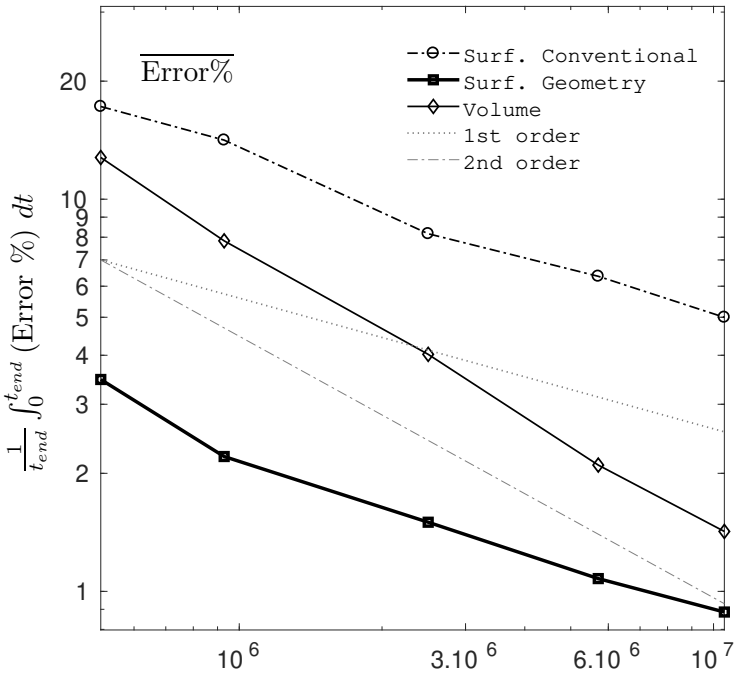

Figure 21: Mean error percentages of the computed interfacial areas (Eq. (5.6), ∘ and Eq. (5.7) ■) and volume (Eq. (5.13) ⋄) for the regression of the B19T grain (Figs. 19 , 20). The first and second order theoretical convergence rates are plotted in dotted and dash dotted lines respectively for easier analysis. The error related to the interfacial area decreases for both formulations (conventional and geometrical) when the mesh is refined and the corresponding convergence rates are first-order. However, errors obtained with the geometry-based relation (Eq. (5.7)) are systematically smaller than the ones provided by Eq. (5.6). The mean error percentage related to the volume computation (Eq. (5.13), ⋄) is superior to the interfacial area (Eq. (5.7)) but remains quite reasonable. Besides, the observed order of accuracy is two.

Figure 21 indicates that the proposed interfacial area computation with the geometry-based relation (Eq. (5.7)) provides a mean error percentage of about 3.5% with the coarsest mesh whereas the conventional relation (Eq. (5.6)) yields a mean error percentage of about 17% for the same mesh resolution.

With the finest mesh, the geometry-based relation (Eq. (5.7)) leads to an error of about 0.88%. The error related to the conventional relation (Eq. (5.6)) is around 5%.

The error decreases for both formulations when the mesh is refined and the corresponding convergence rates are first-order.

However, errors obtained with the geometry-based relation (Eq. (5.7)) are systematically smaller than the ones provided by Eq. (5.6) as already observed in Figs. 17, 19 and 20. Those results show one more time the capabilities of the proposed geometry-based relation (Eq. (5.7)).

The mean error percentage related to the volume computation (Eq. (5.13)) is superior to the interfacial area (Eq. (5.7)) but remains quite reasonable. The error is around 12% with the coarsest mesh and is about 1.4% with the finest mesh. Besides, the observed order of accuracy is two.

7. Conclusion

A "Level-Set" type method has been developed to approximate a first-order Hamilton-Jacobi equation on unstructured meshes in the context of an interface moving along its normal vector field.

The present "Level-Set" function is sharp and the method relies on the resolution of the Riemann problem and a Godunov-type scheme. Numerical diffusion is used to compute the corresponding gradients with the help of the least squares approximations. However, thanks to the Overbee limiter, this artificial smearing is controlled and limited, providing results of high accuracy.

The method is also used to compute the corresponding interfacial area. A new formulation is proposed and multidimensional computations show results in very good agreement with analytical solutions with reasonable mesh resolutions.

Acknowledgments

The authors are very grateful to Jeaniffer Vides for numerous helpful discussions that definitely helped to improve the quality of this work.

This work has been funded by Eurenco that is gratefully acknowledged.

References

[1] S. Osher, J. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics 79 (1) (1988) 12–49.

[2] J. Sethian, Curvature and the evolution of fronts, Communications in Mathematical Physics 101 (4) (1985) 487–499.

[3] J. Sethian, A fast marching level set method for monotonically advancing fronts, Proceedings of the National Academy of Sciences 93 (4) (1996) 1591–1595.

[4] J. Sethian, Numerical algorithms for propagating interfaces: Hamilton-Jacobi equations and conservation laws, Journal of differential geometry 31 (1) (1990) 131–161.

[5] J. Sethian, Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, Vol. 3, Cambridge university press, 1999.

[6] N. Morgan, J. Waltz, 3D level set methods for evolving fronts on tetrahedral meshes with adaptive mesh refinement, Journal of Computational Physics 336 (2017) 492–512.

[7] E. Olsson, G. Kreiss, A conservative level set method for two phase flow, Journal of Computational Physics 210 (1) (2005) 225–246.

[8] E. Olsson, G. Kreiss, S. Zahedi, A conservative level set method for two phase flow II, Journal of Computational Physics 225 (1) (2007) 785–807.

[9] R. Shukla, C. Pantano, J. Freund, An interface capturing method for the simulation of multi-phase compressible flows, Journal of Computational Physics 229 (19) (2010) 7411–7439.

[10] Q. Carmouze, F. Fraysse, R. Saurel, B. Nkonga, Coupling rigid bodies motion with single phase and two-phase compressible flows and unstructured meshes, Journal of Computational Physics 375 (2018) 1314–1338.

[11] R. Saurel, C. Pantano, Diffuse Interfaces and Capturing Methods in Compressible Two-Phase Flows, Annual Review of Fluid Mechanics 50 (2018) 105–130.

[12] A. Chiapolino, R. Saurel, B. Nkonga, Sharpening diffuse interfaces with compressible fluids on unstructured meshes, Journal of Computational Physics 340 (2017) 389–417.

[13] E. Dalla, M. Hilpert, C. Miller, Computation of the interfacial area for two-fluid porous medium systems, Journal of Contaminant Hydrology 56 (1-2) (2002) 25–48.

[14] F. Dauch, D. Ribereau, A software for SRM grain design and internal ballistics evaluation, PIBAL, in: 38th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, 2002, p. 4299.

[15] S. Osher, R. Fedkiw, Level set methods and dynamic implicit surfaces, Vol. 153, Springer Science & Business Media, 2003.

[16] M. Willcox, M. Brewster, K. Tang, D. Stewart, Solid propellant grain design and burnback simulation using a minimum distance function, Journal of Propulsion and Power 23 (2) (2007) 465–475.

[17] E. Cavallini, Modeling and numerical simulation of solid rocket motors internal ballistics, Ph.D. thesis, Sapienza Università di Roma (2010).

[18] W. Sullwald, Grain regression analysis, Ph.D. thesis, University of Stellenbosch (2014).

[19] D. Gueyffier, F. Roux, Y. Fabignon, G. Chaineray, N. Lupoglazoff, F. Vuillot, J. Hijlkema, F. Alauzet, Accurate computation of grain burning coupled with flow simulation in rocket chamber, Journal of Propulsion and Power 31 (6) (2015) 1761–1776.

[20] M. Sussman, P. Smereka, S. Osher, A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow, Journal of Computational Physics 114 (1) (1994) 146–159.

[21] B. Engquist, A. Tornberg, R. Tsai, Discretization of Dirac delta functions in level set methods, Journal of Computational Physics 207 (1) (2005) 28–51.

[22] P. Smereka, The numerical approximation of a delta function with application to level set methods, Journal of Computational Physics 211 (1) (2006) 77–90.

[23] C. Min, F. Gibou, Geometric integration over irregular domains with application to level-set methods, Journal of Computational Physics 226 (2) (2007) 1432–1443.

[24] S. Godunov, A finite difference scheme for numerical computation of the discontinuous wave solutions of equations of fluid dynamics, Math. Sb. 47 (1959) 271–306.

[25] J. Brackbill, D. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (2) (1992) 335–354.

[26] S. Le Martelot, R. Saurel, B. Nkonga, Towards the direct numerical simulation of nucleate boiling flows, International Journal of Multiphase Flow 66 (2014) 62–78.

[27] D. Furfaro, R. Saurel, L. David, F. Beauchamp, Towards sodium combustion modeling with liquid water, Journal of Computational Physics 403 (2020) 109060.

[28] F. Denner, B. van Wachem, Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fractions, Numerical Heat Transfer, Part B: Fundamentals 65 (3) (2014) 218–255.

[29] R. LeVeque, Finite volume methods for hyperbolic problems, Vol. 31, Cambridge University Press, 2002.

[30] E. Toro, Riemann solvers and numerical methods for fluid dynamics: A practical introduction, Springer Science & Business Media, 2013.

[31] A. Harten, P. Lax, B. van Leer, On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws, SIAM Review 25 (1) (1983) 35–61.

[32] S. Davis, Simplified second-order Godunov-type methods, SIAM Journal on Scientific and Statistical Computing 9 (3) (1988) 445–473.

[33] T. Barth, D. Jespersen, The design and application of upwind schemes on unstructured meshes, Proceedings of the AIAA 27th Aerospace Science Meeting (Reno, Nevada), (1989).

[34] P. Sweby, M. Baines, Convergence of Roe's scheme for the general non-linear scalar wave equation, University of Reading. Department of Mathematics, 1981.

[35] P. Roe, Some contributions to the modelling of discontinuous flows, in: Large-scale computations in fluid mechanics, 1985, pp. 163–193.

[36] K. Shyue, F. Xiao, An Eulerian interface sharpening algorithm for compressible two-phase flow: The algebraic THINC approach, Journal of Computational Physics 268 (2014) 326–354.

[37] S. Ii, B. Xie, F. Xiao, An interface capturing method with a continuous function: The THINC method on unstructured triangular and tetrahedral meshes, Journal of Computational Physics 259 (2014) 260–269.

[38] A. Harten, High resolution schemes for hyperbolic conservation laws, Journal of Computational Physics 49 (3) (1983) 357–393.

[39] P. Sweby, High resolution schemes using flux limiters for hyperbolic conservation laws, SIAM Journal on Numerical Analysis 21 (5) (1984) 995–1011.

[40] A. Harten, On a class of high resolution total-variation-stable finite-difference schemes, SIAM Journal on Numerical Analysis 21 (1) (1984) 1–23.

[41] B. van Leer, Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme, Journal of Computational Physics 14 (4) (1974) 361–370.

[42] E. Tadmor, Convenient total variation diminishing conditions for nonlinear difference schemes, SIAM Journal on Numerical Analysis 25 (5) (1988) 1002–1014.

[43] D. Drew, S. Passman, Theory of multicomponent fluids, Vol. 135, Springer Science & Business Media, 2006.

[44] Z. Xiao, W. He, F. Xu, Emulation and Calculation of the Burning Surface of 3D Grains of Partially Cut Multi-Perforated Stick Propellant using the Level Set Method, Propellants, Explosives, Pyrotechnics 41 (1) (2016) 148–153.

[45] J. Horst, W. Albert, F. Robbins, Programmed-splitting solid propellant grain for improved ballistic performance of guns, US Patent 4,581,998 (1986).

[46] J. Stals, Form-Functions for Multi-Component Propellant Charges Including Inhibited Grains and Sliver Burn, Tech. rep., MATERIALS RESEARCH LABS ASCOT VALE (AUSTRALIA) (1975).

[47] H. Krier, M. Summerfield, Interior ballistics of guns, Vol. 66, Progress in astronautics and aeronautics, 1979.

[48] D. Carlucci, S. Jacobson, Ballistics: theory and design of guns and ammunition, CRC Press, 2013.

[49] C. Geuzaine, J. Remacle, P. Dular, Gmsh: a three-dimensional finite element mesh generator, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331.