Numerical simulation of boiling on unstructured grids

Guillaume Sahut

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Mécanique des fluides, Procédés, Energétique**

Arrêté ministériel : 25 mai 2016

Présentée par

Guillaume SAHUT

Thèse dirigée par **Philippe MARTY**, Professeur Emérite, Université Grenoble Alpes (UGA), codirigée par **Guillaume BALARAC**, Maître de Conférences, Grenoble INP, et co-encadrée par **Giovanni GHIGLIOTTI**, Maître de Conférences, UGA

préparée au sein du **Laboratoire des Ecoulements Géophysiques et Industriels (LEGI)**

dans **l'Ecole Doctorale I-MEP2 - Ingénierie - Matériaux, Mécanique, Environnement, Energétique, Procédés, Production**

Simulation numérique de l'ébullition sur maillages non structurés

Numerical simulation of boiling on unstructured grids

Thèse soutenue publiquement le **18 décembre 2019**,
devant le jury composé de :

M. Sébastien TANGUY
Maître de Conférences, Université Paul Sabatier (Toulouse), Rapporteur
M. Olivier DESJARDINS
Associate Professor, Cornell University (Ithaca, NY), Rapporteur
M. Emmanuel MAITRE
Professeur des Universités, Grenoble INP, Examinateur, Président du Jury
M. Julien REVEILLON
Professeur des Universités, Université de Rouen, Examinateur
M. François-Xavier DEMOULIN
Professeur des Universités, Université de Rouen, Examinateur
M. Philippe MARTY
Professeur Emérite, Université Grenoble Alpes, Examinateur, Directeur de thèse
M. Guillaume BALARAC
Maître de Conférences, Grenoble INP, Membre invité
M. Giovanni GHIGLIOTTI
Maître de Conférences, Université Grenoble Alpes, Membre invité
M. Vincent MOUREAU
Chargé de Recherche CNRS, Université de Rouen, Membre invité

Remerciements

Je tiens tout d'abord à remercier Sébastien Tanguy et Olivier Desjardins d'avoir accepté de rapporter mes travaux, et d'avoir été disponibles régulièrement au cours de cette thèse pour me transmettre leur grande expérience sur les (nombreux) problèmes liés à la simulation numérique de l'ébullition. Je remercie également Emmanuel Maitre d'avoir présidé mon jury de soutenance, ainsi que l'ensemble de mon jury, composé également de Julien Réveillon, François-Xavier Demoulin, Philippe Marty, Guillaume Balarac, Giovanni Ghigliotti et Vincent Moureau, pour l'intérêt qu'ils ont porté à mon travail.

J'ai effectué ma thèse au LEGI. Je tiens ici à remercier toute l'équipe du laboratoire, à commencer par les directeurs Achim Wirth puis Joël Sommeria. Merci également à Lafiyatou Emeriaud, Cécile Cretin, Julie Germinario et Nathalie Lawson pour leur aide administrative.

Pour m'avoir fait confiance et encadré très assidûment, et toujours avec bienveillance, je remercie très chaleureusement Giovanni Ghigliotti et Guillaume Balarac. Leur implication constante et conséquente (parfois même, nocturne) a grandement contribué à la réussite de cette thèse. Merci ! Un grand merci également à mon directeur de thèse, Philippe Marty, pour ses conseils, sa bonne humeur et sa grande expertise en mécanique des fluides.

Vincent Moureau et Ghislain Lartigue, par leurs formations régulières au code YALES2 et leur grande expertise physique et informatique, mais surtout par leur disponibilité et leur gentillesse, ont également été très importants pour moi.

Une grande chance de l'équipe MoST et du LEGI en général est la très bonne entente de ses membres, ce qui en fait un cadre idéal pour l'entraide, et un endroit où l'on a envie de venir travailler. Cela a été mon cas pendant ces quelques années. Merci donc aux anciens doctorants et post-doctorants : Prasanta Sarkar, Alexandre Sikora, Sylvia Wilhelm, Yves Paquette, Lucas Séguinot, Flavia Turi, François Doussot, Alexandre Simon, Andrey Pushkarev, Roumaïssa Hassaini, Vincent Clary, et toux ceux que j'oublie...

Et pour la jeune génération : Matthieu Guilbot, Matei Badalan, Simon Santoso, Himani Garg, Hugo Frézat, Thomas Fabbri, Pedro Henrique Dias de Veras Sousa, Jérémie Dagaut, Anastasiia Gorbunova, Alexis Barge, Manuel Bernard, Cyril Bozonnet l'Américain, et Savinien Pertant qui aura bientôt l'honneur de réceptionner le solveur Boiling !

Et bien évidemment... un très grand merci à Patrick Bégou, sans qui rien ne va plus ! Ta bonne humeur est aussi source de motivation dans les moments difficiles.

En dehors du labo, merci à Baptiste et Mathieu pour toutes ces années d'amitié, merci à Guillaume (Guimch pour les intimes) pour les mêmes raisons et pour m'avoir un jour, sur le balcon de Baptiste, donné l'idée de faire un tour en physique. Ce ne fut pas une mauvaise idée.

Je remercie mes parents de m'avoir soutenu et encouragé dans cette belle aventure. C'est fait !

Enfin, un énorme merci à Marion pour m'avoir encouragé et supporté pendant tout ce temps... notamment en cas de bulle carrée... ce qui n'a pas été facile tous les jours !

Je suis très heureux d'avoir fait cette thèse. J'espère qu'elle suscitera votre intérêt. Merci encore à tous, et à très bientôt !

Guillaume Sahut,
Mercredi 18 mars 2020

Contents

Chapter 1

Introduction : the boiling phenomenon

In this chapter we give a physical description of the boiling phenomenon.

Outline

1.1 Introduction

1.1.1 Generalities

1.1.1.1 Introduction to boiling

Phase change is the transition of fluid particles from one phase to another. In a fluid, boiling is the phase change from the liquid phase to the vapor phase due to temperature increase. Figure 1.1 shows an example of boiling in everyday life. Boiling is a physical phenomenon which plays an important role in various industrial processes. It occurs especially in steam cycles of thermal power stations, nuclear power plants, two-phase loops used in cooling of electronic components and spray cooling processes. It is also used in the condenser heat exchangers of every refrigerating machine or heat pump. The study of boiling proves to be a major stake not only from a theoretical and academic point of view but also from a technical and industrial point of view, where any improvement of boiling heat transfer would have a major impact given the number of applications.

1.1.1.2 Two-phase flows

The boiling phenomenon implies flows composed of a liquid phase and a vapor phase. In such flows, the separation between the two phases is referred to as interface. Gravitational, inertial and surface tension forces all play a role in the physics of two-phase flows. For large bubbles, the inertial

Fig. 1.1 – Nucleate boiling in a heated tureen. Credit: Roman Sigaev/Fotolia.

forces due to the surrounding flow velocity are dominant. For small bubbles, the surface tension forces, resulting from the intermolecular forces at the interface, are dominant. Inertial forces tend to deform the bubbles, while surface tension forces tend to maintain their spherical shape, as the spatial configuration minimizing their internal energy.

1.1.1.3 Phase change in two-phase flows

Under certain circumstances, part of the liquid phase adjacent to the interface is transformed into vapor. This transition from the liquid phase to the vapor phase can be due to a temperature or pressure change in the fluid. Cavitation is the phase change from the liquid phase to the vapor phase due to a decrease of the liquid pressure below a limit called saturation pressure of the liquid. Below the saturation pressure, the molecular bonds in the liquid phase are broken, leading to the appearance of a vapor phase. Boiling is the phase change from the liquid phase to the vapor phase due to an increase of the liquid temperature above a limit called boiling temperature of the liquid. Similarly to cavitation, above the boiling temperature, molecular bonds in the liquid phase are broken. As a result, the liquid phase in the concerned neighborhood turns into vapor. As shown on the qualitative phase diagram of water in Fig. 1.2, boiling corresponds to an increase of temperature during which the pressure remains constant (isobaric phenomenon). Conversely, cavitation corresponds to a decrease of pressure at constant temperature.

When liquid molecules become vapor, their intermolecular bonding is broken. To do so, energy must be given to the system. This corresponds to the latent heat of vaporization. Therefore, if a thermal flux reaches a vaporizing interface, only a part of it will be able to cross the interface, since

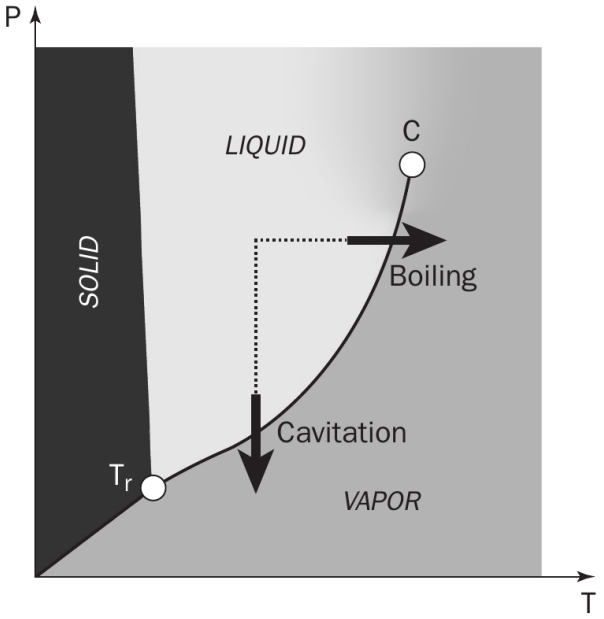


Fig. 1.2 – Pressure-temperature (P-T) phase diagram for water adapted from the book of Franc and Michel [23]. The triple point T_r is the point at which the three phases coexist, the critical point C separates the liquid and vapor domains.

a part is converted into latent heat.

1.1.1.4 Nucleate boiling

This work is the first step of a project aiming at the numerical simulation of two-phase flows with phase change in the nucleate boiling regime. Nucleate boiling occurs when a liquid phase is in contact with a solid phase at a temperature superior to the boiling temperature of the liquid. In this case, a thermal flux is directed from the solid to the fluid, and the liquid close to the solid is transformed into vapor : nucleated bubbles develop along the solid surface. These bubbles are initially attached to the surface. Due to the heat flux from the solid to the liquid phase, the nucleated bubbles grow on the solid surface. Above a critical size, the bubbles detach from the solid surface under buoyancy effects. Nucleate boiling is an efficient heat transfer mode in the sense that it is generally associated to high rates of heat transfer from the heated surface to the fluid for relatively low temperature differences between the solid and liquid phases. Nevertheless, attention has to be paid to the intensity of the heat flux at the solid-liquid interface in order to avoid to damage the solid part, as explained below.

1.1.1.5 The boiling crisis

One major drawback of nucleate boiling appears when the heat flux at the solid surface reaches a limit called Critical Heat Flux (CHF). Above the CHF, bubbles attached to the surface merge and tend to form a vapor film covering the solid surface. Suddenly, the solid is totally separated from

the liquid phase. Due to the weak thermal conductivity of the vapor, as compared to the thermal conductivity of the liquid, the heat transfer coefficient, accounting for heat transfer from the solid phase to the fluid, decreases drastically. As a consequence, the temperature at the solid surface increases dramatically. This increase of the surface temperature can seriously damage and even destroy the solid part if the melting temperature of the solid material is reached. This phenomenon is called boiling crisis. In industrial applications, e.g. in nuclear power plants, the boiling crisis can have destructive consequences.

1.1.1.6 Objective of this thesis

The objective of this thesis is the development of a numerical simulation code for boiling flows. As a first step, the contact with a solid surface is neglected. Therefore, we focus only on the fluid, constituted by liquid and vapor.

1.1.1.7 Numerical developments

The numerical developments have been implemented in the YALES2 code, a community-based code developed by a network of several French labs within the Scientific Interest Group (GIS) "SUCCESS"[1]. YALES2 is a numerical code whose general purpose is the resolution of fluid mechanics problems. It is designed for three-dimensional unstructured grids in order to simulate single-phase and two-phase flows on complex geometries. At the beginning of this thesis, numerical simulations of two-phase flows could be performed with YALES2 but phase change was not implemented. I then dedicated myself to the development and validation of a numerical method suitable for simulations of boiling in two and three dimensions.

Numerical simulations consist in solving on a computer the differential equations describing the evolution of the flow. These equations are solved on a discretized space called grid. Two families of grids can be used : cartesian grids and non-cartesian, or unstructured, grids. Cartesian grids have the advantage of being aligned with the orthogonal directions of the reference frame, thus making various computations easier, as will be largely detailed in this thesis. The drawback of cartesian grids is their limitation to basic computational domains, only delimited by horizontal and vertical lines. Conversely, unstructured grids have the ability to describe more complex geometries. The drawback of unstructured grids is the desordered layout of the points where the equations are solved, thus making accuracy more difficult to be reached in numerical schemes. Our methodology has been designed for unstructured grids, with distributed memory parallelism in order to take advantage of computational power. I focused on the development of all the tools needed to simulate phase change on unstructured grids. Taking into account the triple contact (liquid, vapor, solid) is left to future works.

Our configuration of interest is a vapor bubble surrounded by a liquid phase, as shown in Fig. 1.3. Both the vapor and liquid phases are of the same chemical composition. We focus on the growth of the bubble due to the liquid phase heated above the saturation temperature. The nucleation of the vapor bubble inside the liquid phase is not addressed in this thesis. In addition to the complexity of the physical phenomena encountered at the interface when phase change is not considered (action of surface tension and parameter discontinuity, density and viscosity, as well as pressure discontinuity between the two phases), boiling contributes to the pressure discontinuity and also implies a velocity discontinuity at the interface between the liquid and vapor phases. Classical numerical methods

[1] `https://success.coria-cfd.fr`

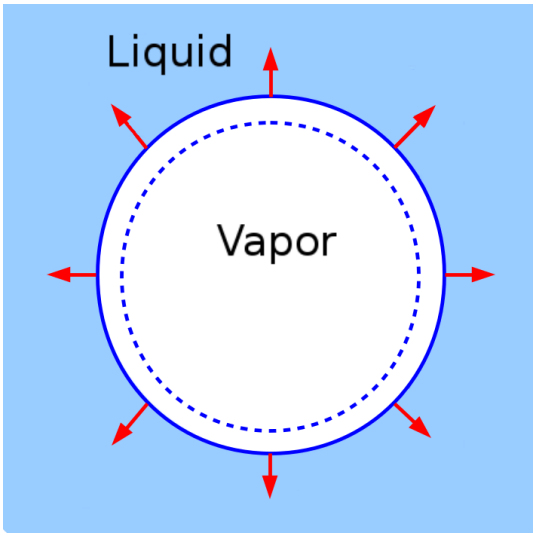


Fig. 1.3 – The boiling phenomenon : growth of a vapor bubble surrounded by a liquid phase. The dashed blue circle represents the location of the liquid-vapor interface at a previous time. The plain blue circle represents the current interface location. The red arrows give the expansion direction of the interface.

for the simulation of two-phase flows are not designed to take into account a discontinuous velocity field whose discontinuity is moreover coupled to the temperature field. Substantial efforts are then required to extend existing algorithms in order to simulate boiling flows, especially on unstructured grids.

In this thesis, we present these developments along with some promising results concerning numerical simulations of boiling flows on unstructured grids.

This work has been conducted at LEGI Laboratory (Laboratoire des Ecoulements Géophysiques et Industriels), Grenoble, France, in the MoST[2] team, and has been funded by the Tec21 LabEx[3] of Université Grenoble Alpes.

1.1.2 Thesis overview

This thesis is composed of six chapters and a conclusion.

Chapter 1 introduces the physics of boiling and the particular regime of nucleate boiling as the motivation of this work.

Chapter 2 presents the state-of-the-art numerical methods used in two-phase flow simulations, with a strong emphasis on the numerical methods commonly used to model the motion of the liquid-vapor interface.

Chapter 3 introduces the YALES2 code suitable for two-phase flow simulations on three-dimensional unstructured grids in which I developed a numerical method to take boiling into account.

[2]Modeling and Simulation of Turbulence

[3]The *laboratoire d'excellence* for mechanical and process engineering is a group of seven research laboratories located in Grenoble, France (`https://www.tec21.fr`).

Chapter 4 details the numerical method developed for simulations of two-phase flows with boiling in one dimension, where phase change is driven by thermal fluxes at the liquid-vapor interface.

Chapter 5 extends the numerical method to multidimensional structured and unstructured grids, where the amount of liquid turning into vapor per surface and time units is imposed. This chapter exposes the difficulties met due to the complex shape of the interface with respect to the one-dimensional case, then introduces and validates the different solutions implemented.

Chapter 6 extends the previous methodology by computing the amount of liquid turning into vapor per surface and time units from the thermal fluxes at the interface (as in Chapter 4 for the one-dimensional case) on two- and three-dimensional unstructured grids.

Finally, a conclusion on the presented work is given and highlights on the perspectives of this work towards numerical simulations of two-phase flows with phase change by nucleate boiling on three-dimensional unstructured grids are mentioned.

1.2 Physics of two-phase flows

In this section, the equations for the fluid motion are given.

1.2.1 Incompressible two-phase flows

Equations describing fluid dynamics are based on the conservation of mass, momentum and energy. The total mass of a fluid is conserved throughout time. Mass conservation reads

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{1.1}$$

where t is the time, ρ is the density and $\mathbf{u}$ the velocity of the fluid. Momentum, or quantity of motion, of the fluid is also conserved. Momentum conservation corresponds to the conservation of the quantity $\rho\mathbf{u}$. The equation describing momentum conservation is the Navier-Stokes equation, given by

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla P + \nabla \cdot \left(\mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right) + \rho \mathbf{g}, \tag{1.2}$$

where P is the fluid pressure, μ is the fluid dynamic viscosity and $\mathbf{g}$ is the gravitational acceleration. If ρ is assumed to be constant and uniform (incompressible flow), Eqs. (1.1) and (1.2) can be simplified. Mass conservation then reads

$$\nabla \cdot \mathbf{u} = 0. \tag{1.3}$$

Similarly, the Navier-Stokes equation reads

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\, \mathbf{u} = -\frac{1}{\rho}\nabla P + \frac{1}{\rho}\nabla \cdot \left(\mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right) + \mathbf{g}. \tag{1.4}$$

1.2.2 Energy conservation

Energy conservation can be formulated by means of different primitive variables as the internal energy or the enthalpy. The internal energy is favored to describe thermodynamics phenomena

in closed systems, and temperature (or enthalpy) is more commonly used for description of quasi-isobaric phenomena (usually open systems) [85]. In this thesis we focus on simulations of boiling on open systems, we then express energy conservation with the temperature variable, i.e.

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \frac{1}{\rho c_{\mathrm{p}}} \nabla \cdot (\lambda \nabla T), \tag{1.5}$$

where T is the temperature, c_{p} is the heat capacity at constant pressure and λ is the thermal conductivity of the fluid. If λ is uniform, then the coefficient $\alpha = \lambda / (\rho c_{\mathrm{p}})$ in the rhs of Eq. (1.5) is the thermal diffusivity of the fluid.

1.2.3 Rankine-Hugoniot jump conditions at the liquid-vapor interface without phase change

In a two-phase flow without phase change, the velocity is continuous across the interface [57], i.e. one has

$$[\mathbf{u}]_\Gamma = \mathbf{0}, \tag{1.6}$$

where $[\cdot]_\Gamma$ is the jump operator defined for a scalar or vector field A as

$$[A]_\Gamma = A_{\mathrm{liq}} - A_{\mathrm{vap}}, \tag{1.7}$$

where A_{liq} is the value of A in the liquid side and A_{vap} is the value of A in the vapor side, both immediately close to the interface. Furthermore, the pressure is in general discontinuous at the interface between the two phases. For a liquid at rest, if it possesses a surface tension, the pressure discontinuity at the interface is given by the Laplace pressure as

$$[P]_\Gamma = \sigma \kappa, \tag{1.8}$$

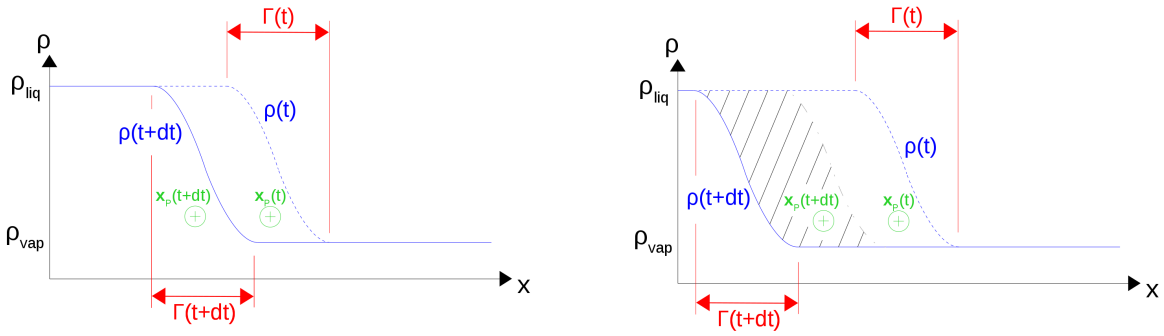
where σ is the surface tension of the fluid, κ is the interface curvature. Moreover, if the flow velocity is non-zero at the interface, the discontinuity of dynamic viscosity at the interface also contributes to $[P]_\Gamma$. The pressure discontinuity at the interface, due to the combined effects of surface tension and shear stress, is then given by the Rankine-Hugoniot jump condition [19, 57]

$$[P]_\Gamma = \sigma \kappa + 2 [\mu]_\Gamma \, \mathbf{n}^{\mathrm{T}} \cdot \nabla \mathbf{u} \cdot \mathbf{n}^{\mathrm{T}}, \tag{1.9}$$

where $\mathbf{n}$ is the interface normal vector and $\mathbf{u}$ the flow velocity at the interface.

1.3 Physics of boiling

Boiling is the phase change characterized by the transition of the fluid particles from the liquid phase to the vapor phase entailed by thermal fluxes at the interface. It exists then a mass transfer between the two phases, characterized by a mass transfer rate $\dot{m}$. Navier-Stokes equations are then coupled to the heat equation by means of the mass transfer rate computed from the thermal fluxes at the interface. Due to this mass transfer rate, and to the fact that vapor and liquid densities are different, we will show in this section that boiling induces a volume expansion at the interface, which in turn leads to a non-zero divergence velocity field at the interface. We then show that this non-zero divergence velocity field, together with the mass transfer rate, create additional terms for the discontinuities of pressure and velocity at the interface. Moreover, we stress here that the mass transfer rate also contributes to the interface motion.



(a) Advection of diffuse interface without phase change. The fluid particle centered at point $\mathbf{x}_P$ is at rest with respect to the interface frame.

(b) Advection of diffuse interface with phase change : the interface motion is faster than the fluid particle motion due to the source term in Eq. (1.17). As a result, the fluid particle exits the interface region. The hatched area represents the contribution of phase change to the interface motion.

Fig. 1.4 – The density ρ describes the interface Γ at a fine scale at times t and $t + \mathrm{d}t$ (a) without phase change, and (b) with phase change.

1.3.1 Phase change in a non-confined area leads to volume expansion

When a fluid particle located in the liquid phase is transformed into vapor, its physical properties become the ones of the vapor phase. As a result, boiling acts on the flow by means of the abrupt transition imposed to the physical properties of the fluid particles that cross the interface. A ratio of three orders of magnitude is typically observed between densities of the liquid and vapor phases of a given chemical compound (e.g. water). Consequently, boiling divides the density of liquid-turned-vapor fluid particles by a factor roughly equal to 10^3. The mass δm of a fluid particle is given by

$$\delta m = \rho \delta V, \tag{1.10}$$

where ρ is the density and δV the volume of the fluid particle. Mass conservation implies that δm is constant, hence a decrease of ρ implies an increase of δV.

1.3.2 Influence on velocity divergence of volume expansion at the interface

In order to examine the physical changes at the interface induced by boiling, it is useful to describe the interface at a scale where its width is non zero. This interface smoothing can be represented by means of the smoothing of a per-phase uniform physical property (density, dynamic viscosity, etc). The following derivation uses the density function. Figure 1.4 represents the density function with a smooth transition between the two phases at times t and $t + \mathrm{d}t$, 1.4(a) without phase change, and 1.4(b) with phase change. The density transition describing the interface (i.e. the area where $\|\nabla\rho\| \neq 0$) used in Fig. 1.4 depends on the model used to represent the molecular dynamics between the two phases. In both cases, the interface is advected from right to left, so the liquid and vapor velocities are not null. In Fig. 1.4(a), the fluid particle located in the diffuse interface and centered at point $\mathbf{x}_P$ is advected at the same velocity as the surrounding interface (in

absence of phase change). Mass conservation applied to the center $\mathbf{x}_P$ of the fluid particle at time t still reads

$$\frac{\mathrm{d}\rho}{\mathrm{d}t}(t,\mathbf{x}_P) = \frac{\partial\rho}{\partial t}(t,\mathbf{x}_P) + \nabla\cdot(\rho(t,\mathbf{x}_P)\mathbf{u}(t,\mathbf{x}_P)) = 0, \tag{1.11}$$

i.e.

$$\nabla\cdot(\rho(t,\mathbf{x}_P)\mathbf{u}(t,\mathbf{x}_P)) = -\frac{\partial\rho}{\partial t}(t,\mathbf{x}_P). \tag{1.12}$$

Since

$$\nabla\cdot(\rho(t,\mathbf{x}_P)\mathbf{u}(t,\mathbf{x}_P)) = \rho(t,\mathbf{x}_P)\nabla\cdot\mathbf{u}(t,\mathbf{x}_P) + \mathbf{u}(t,\mathbf{x}_P)\cdot\nabla\rho(t,\mathbf{x}_P), \tag{1.13}$$

one has

$$\nabla\cdot\mathbf{u}(t,\mathbf{x}_P) = -\frac{1}{\rho(t,\mathbf{x}_P)}\left(\frac{\partial\rho}{\partial t}(t,\mathbf{x}_P) + \mathbf{u}(t,\mathbf{x}_P)\cdot\nabla\rho(t,\mathbf{x}_P)\right), \tag{1.14}$$

where the terms between parentheses are the lhs of the density advection equation given at point $\mathbf{x}_P$ and at time t.

Without phase change, this density advection equation reads

$$\frac{\partial\rho}{\partial t}(t,\mathbf{x}_P) + \mathbf{u}(t,\mathbf{x}_P)\cdot\nabla\rho(t,\mathbf{x}_P) = 0, \tag{1.15}$$

leading, by Eq. (1.14), to

$$\nabla\cdot\mathbf{u}(t,\mathbf{x}_P) = 0. \tag{1.16}$$

In the presence of phase change, the fluid particle, in addition to being advected by the surrounding interface velocity at point $\mathbf{x}_P$, exits the interface region, implying that the interface moves faster than the fluid particle. This increase in the interface velocity in case of phase change is accounted for by a source term $\dot{M}$ in Eq. (1.15) which then reads

$$\frac{\partial\rho}{\partial t}(t,\mathbf{x}_P) + \mathbf{u}(t,\mathbf{x}_P)\cdot\nabla\rho(t,\mathbf{x}_P) = -\dot{M}(t,\mathbf{x}_P), \tag{1.17}$$

resulting, by Eq. (1.14), in

$$\nabla\cdot\mathbf{u}(t,\mathbf{x}_P) = \frac{\dot{M}(t,\mathbf{x}_P)}{\rho(t,\mathbf{x}_P)}. \tag{1.18}$$

The source term $\dot{M}$ is a volumetric mass transfer rate expressed in $\mathrm{kg\,m^{-3}\,s^{-1}}$, so the rhs of Eq. (1.18), expressed in $\mathrm{s^{-1}}$, is the rate of change of the velocity $\mathbf{u}$ at point $\mathbf{x}_P$ and at time t. The effect of $\dot{M}$ is illustrated in Fig. 1.4(b). Integrating the rhs of Eq. (1.17) in the direction along which the interface has been smoothed, one has

$$\dot{m}(t) = \int \dot{M}\,(t,\mathbf{x})\,\mathrm{d}\ell, \tag{1.19}$$

where $\dot{m}$ is the mass transfer rate at the interface accounting for the boiling phenomenon (see Section 1.3.3). An integration by parts along the same direction of the rhs of Eq. (1.18) then gives

$$\int \nabla\cdot\mathbf{u}\,(t,\mathbf{x})\,\mathrm{d}\ell = \dot{m}(t)\left(\frac{1}{\rho_{\mathrm{liq}}} - \frac{1}{\rho_{\mathrm{vap}}}\right), \tag{1.20}$$

which, as stated in [85], in the case of an infinitely thin interface, reads

$$(\nabla \cdot \mathbf{u})|_\Gamma = \dot{m} \left[\frac{1}{\rho}\right]_\Gamma \delta_\Gamma, \tag{1.21}$$

where the temporal dependence is omitted and δ_Γ is the Dirac delta function defined by $\delta_\Gamma(\mathbf{x}) = 0$ everywhere except when $\mathbf{x}$ belongs to the interface, and

$$\int_{\mathbb{R}^n} \delta_\Gamma(\mathbf{x}) \mathrm{d}\mathbf{x} = 1, \tag{1.22}$$

where n is the spatial dimension of the problem.

1.3.3 Mass transfer rate

The mass transfer rate $\dot{m}$ introduced in the previous section is equal to the number of liquid mass units crossing the interface per surface and time units, thus it is expressed in $\mathrm{kg\,m^{-2}\,s^{-1}}$. When there is no phase change, energy conservation implies that the heat flux at the interface is entirely transmitted from one phase to the other. Mathematically, this means that the normal component of the heat flux is continuous at the interface, i.e.

$$\left(-\lambda_{\text{liq}} \nabla T_{\text{liq}}|_\Gamma + \lambda_{\text{vap}} \nabla T_{\text{vap}}|_\Gamma\right) \cdot \mathbf{n}_\Gamma = 0, \tag{1.23}$$

where $\nabla T_{\text{liq}}|_\Gamma$ and $\nabla T_{\text{vap}}|_\Gamma$ are the restrictions to the interface Γ of the liquid and vapor temperature gradients, and $\mathbf{n}_\Gamma$ is the interface normal vector. Equation (1.23) can be shortened to

$$[-\lambda \nabla T \cdot \mathbf{n}]_\Gamma = 0. \tag{1.24}$$

When phase change occurs, the heat flux at the interface is divided in two parts :

- a portion of the heat flux, equal to $[-\lambda \nabla T \cdot \mathbf{n}]_\Gamma$, is absorbed by the interface neighborhood in order to provide the latent heat L_v required to break liquid molecular bonds and transform some quantity $\dot{m}$ of liquid into vapor per interface surface and time units,
- the remaining part of the heat flux is transmitted from one phase to the other.

The energy conservation equation (1.24) is then rewritten

$$[-\lambda \nabla T \cdot \mathbf{n}]_\Gamma = \dot{m} L_v, \tag{1.25}$$

accounting for the latent heat of vaporization as an energy well at the interface. Consequently, the quantity of liquid transformed into vapor per surface and time units, namely the mass transfer rate $\dot{m}$, is given by the amount of thermal energy brought to the interface by the flow dynamics (heat equation), per surface and time units, and absorbed by the interface, i.e. $[-\lambda \nabla T \cdot \mathbf{n}]_\Gamma$ expressed in $\mathrm{W\,m^{-2}}$, divided by the amount of energy needed to transform one mass unit of liquid into vapor, i.e. the latent heat of vaporization L_v of the fluid expressed in $\mathrm{J\,kg^{-1}}$. The mass transfer rate $\dot{m}$ is thus defined as

$$\dot{m} = \frac{1}{L_v} \left[-\lambda \nabla T \cdot \mathbf{n}\right]_\Gamma. \tag{1.26}$$

The discontinuity of the heat flux component normal to the interface is then equivalent to the occurrence of boiling.

1.3.4 Rankine-Hugoniot jump conditions at a liquid-vapor interface with phase change

Following the notations of Nguyen et al. [54] and Gibou et al. [26], we denote the interface normal velocity by $\mathbf{W} = D\mathbf{n}$, where $\mathbf{n}$ is the interface normal vector. Mass conservation across the interface reads (see also [88])

$$\dot{m} = \rho_{\text{liq}} \left(\mathbf{u}_{\text{liq}} \cdot \mathbf{n} - D \right) = \rho_{\text{vap}} \left(\mathbf{u}_{\text{vap}} \cdot \mathbf{n} - D \right). \tag{1.27}$$

As a result, the magnitude of the interface normal velocity D is given by

$$D = \mathbf{u}_{\text{liq}} \cdot \mathbf{n} - \frac{\dot{m}}{\rho_{\text{liq}}} = \mathbf{u}_{\text{vap}} \cdot \mathbf{n} - \frac{\dot{m}}{\rho_{\text{vap}}}, \tag{1.28}$$

leading to the discontinuity of the normal velocity at the interface, given by

$$\left[\mathbf{u} \cdot \mathbf{n} \right]_\Gamma = \dot{m} \left[\frac{1}{\rho} \right]_\Gamma. \tag{1.29}$$

Moreover, in order to specify that the tangential components of the velocity are continuous across the interface, it more convenient to rewrite Eq. (1.29) as [54, 26, 85]

$$\left[\mathbf{u} \right]_\Gamma = \dot{m} \left[\frac{1}{\rho} \right]_\Gamma \mathbf{n}, \tag{1.30}$$

which is the equivalent of Eq. (1.6) in the case of phase change.

Similarly, integrating Eq. (1.4) across the interface and including surface tension effects leads to the pressure discontinuity across the interface given by [85]

$$\left[P \right]_\Gamma = \sigma\kappa + 2 \left[\mu \frac{\partial \mathbf{u}_\mathbf{n}}{\partial \mathbf{n}} \right]_\Gamma - \dot{m}^2 \left[\frac{1}{\rho} \right]_\Gamma, \tag{1.31}$$

where the first term of the rhs accounts for surface tension effects, the second term accounts for viscous effects (only if the flow is in motion), and the third term accounts for the ejection of the liquid from the interface due to the density difference between the liquid and vapor phases. Equation (1.31) is the equivalent of Eq. (1.9) in the case of phase change.

Equations (1.30) and (1.31) imply that the mass transfer rate is of primary importance in the physics of boiling flows since it directly acts on the velocity and pressure of the flow at the interface.

1.3.5 Velocity and temperature of the liquid-vapor interface

The velocity of the interface in absence of phase change is equal to the velocities of the fluids surrounding it. With phase change, these two velocities differ due to Eq. (1.30). The total velocity of the interface $\mathbf{u}_{\Gamma,\text{tot}}$ is then given, using Eq. (1.28), by

$$\mathbf{u}_{\Gamma,\text{tot}} = \mathbf{u}_{\text{liq}} - \frac{\dot{m}}{\rho_{\text{liq}}} \mathbf{n}_\Gamma \tag{1.32}$$

and

$$\mathbf{u}_{\Gamma,\text{tot}} = \mathbf{u}_{\text{vap}} - \frac{\dot{m}}{\rho_{\text{vap}}} \mathbf{n}_\Gamma, \tag{1.33}$$

where the subscript $_\Gamma$ emphasizes that the normal vector is considered at the interface. Equations (1.32) and (1.33) denote that the transport of the interface can be formulated in both phases.

The temperature can usually be assumed continuous at the interface [85] (considerations on the variation of the saturation temperature due to the pressure jump condition and the interface thermal resistance due to molecular effects can be used to question this assumption [42]). In this thesis, the interface temperature is always assumed equal to the saturation temperature, and the saturation temperature is always assumed uniform (mono-component vapor phase and liquid phase) and constant (isobaric approximation), i.e.

$$T_\Gamma = T_{\text{sat}}, \tag{1.34}$$

where T_Γ is the interface temperature and T_{sat} is the saturation temperature of the fluid.

This concludes the physical description of a two-phase boiling fluid without mention of a contact with a heated surface.

1.3.6 Nucleate boiling

We now pass to the discussion of the physical properties of two-phase flows in contact with a solid surface. This part is given for reference, despite the fact that triple contact is not a subject of this thesis.

When a liquid phase is in contact with a solid phase at a temperature above the fluid boiling point, the liquid close to the solid is vaporized and vapor bubbles form on the solid surface. This phenomenon is called nucleate boiling. High rates of heat transfer from the solid to the fluid can be reached, thus making nucleate boiling a widely used heat transfer mode in industrial applications. Appendix A introduces the physics of nucleate boiling.

Chapter 2

State-of-the-art numerical methods for incompressible two-phase flow simulations

A review of some classical numerical methods for incompressible two-phase flow simulations is presented in this chapter. Extensions to phase change are briefly discussed and will be detailed in the next chapters. Finally, a literature review on numerical methods for simulations of two-phase flows with phase change is provided.

Outline

2.1 Generalities

2.1.1 Molecular dynamics versus continuum mechanics

Numerical simulations in fluid dynamics consist in the study of the evolution of some important fluid physical properties. Conservation principles are used to derive partial differential equations in order to model the temporal and spatial evolution of these properties. Typically, the physical properties of interest are the velocity, pressure, density and dynamic viscosity of the fluid. If heat

transfer is considered, the thermal conductivity and capacity at constant pressure or volume, and finally the temperature are also included in the simulation. Some of these properties are generally assumed to exhibit negligible variations in space and time. Such a property is thus modeled using a single uniform and constant value known beforehand. Dynamic viscosity and thermal conductivity are two examples of physical properties which can reasonably be assumed to be uniform and constant in many cases. For liquids, density can somehow be also considered uniform and constant, as detailed in Section 2.1.2. Conversely, velocity, pressure and temperature are not known *a priori* and need be computed. The observed fluid behavior, and thus the modeling of its evolution, depends on the observation scale. At microscopic scale, matter is a discrete set of molecules and is thus discontinuous. At macroscopic scale, the entire space is filled with matter, which is then continuous. The scale-dependent duality between discrete and continuous representation of matter leads to different numerical methods.

When a fluid is described at microscopic scale, space itself can be discretized by means of a lattice, implying that particles can only hop from site to site on the lattice. The particle dynamics is driven by propagation and collision steps in such a way that the particles always stay on the lattice, and mass, momentum and energy are conserved. These fluid models are called *Lattice-Gas Cellular Automata* (LGCA) and the associated equations of propagation and collision, *Lattice Boltzmann Models* (LBM) [12]. Numerical simulations of phase change using LGCA and LBM have been reported in [6, 7, 30], and a methodology for three-phase contact line modeling on curved boundaries is presented in [21]. The foundations of the theory of LGCA and LBM are widely detailed in [69, 96, 82]. The interest for LBM grew in 1986 when Frisch et. al. [24] showed that the molecular dynamics within a fluid, even drastically simplified, still led to a realistic description of fluid mechanics. The ability of LBM to model realistic fluid dynamics has been shown in [20] where von Kármán streets behind a flat obstacle have been observed, in accordance to the results obtained with the Navier-Stokes equations when the fluid is described as a continuum.

At macroscopic scale, since space is a continuum filled with matter, temporal and spatial evolutions of physical properties (velocity, pressure, temperature) are easily modeled with differential operators. The evolution of velocity and pressure is modeled by the Navier-Stokes equations, while the evolution of temperature is modeled using the heat equation. In this thesis, we focus on a macroscopic description of the fluid, hence we solve the Navier-Stokes and heat equations as given in Chapter 1.

2.1.2 Compressible versus incompressible flows

In a fluid, variations of pressure induce variations of density. Due to these variations, the fluid is compressible. The two main consequences of compressibility are the propagation of pressure waves, which are challenging to treat numerically [66], and the non-zero divergence of the velocity field. In order to close the system of Navier-Stokes equations, an equation of state is needed to compute the density from the pressure. However, these density variations are usually neglected when the Mach number (ratio of the flow speed and sound speed in the flow) is smaller than 0.3. Taking flow compressibility into account is only relevant when very high speeds are met (aircraft, rocket engines, atmospheric entries, cavitation in water turbines, etc).

For Mach numbers lower than 0.3, the flow is said incompressible (it is still compressible but its compressibility is neglected). As a result, pressure (sound) travels at infinite speed, i.e. pressure waves immediately vanish. We then emphasize here that purely incompressible flows are only flow models and do not exist in reality. Nevertheless, incompressible flows are sufficiently accurate

models to be widely used in the literature : approximating pressure waves at very high velocity (compared to the fluid velocity) with pressure waves at infinite velocity does not alter the evolution of the flow. One advantage of incompressible flows is that no equation of state is needed to compute the density since it is a constant value known beforehand. In this thesis, only incompressible flows are considered.

2.1.3 Multiphase flows

Numerical simulations of multiphase flows have to account for the different physical properties of each phase. In multiphase flows without phase change, the velocity is continuous at the interface (Eq. (1.6), whereas the pressure is discontinuous at the interface due to the surface tension and the discontinuity of the dynamical viscosity at the interface (Eq. (1.9)). Moreover, numerical methods also have to take into account the discontinuity of density, and, if the heat equation is solved, the discontinuities of thermal conductivity and heat capacity at constant pressure of the two phases. If phase change is considered, the velocity is discontinuous at the interface (Eq. (1.30)) and the pressure discontinuity at the interface is increased by phase change due to the different phase densities. Consequently, numerical simulations of two-phase flows require numerical methods to follow the interface position in order to treat these discontinuities.

2.2 Interface modeling : the Sharp Interface model versus the Continuum Surface Force model

At macroscopic scale, the interface separating the two non-miscible phases is infinitely thin. Consequently, the physical properties, e.g. density and viscosity, are uniform in each phase and admit a mathematical discontinuity at the interface. This assumption is referenced in the literature as the *Sharp Interface* (SI) model. In this model, the interface is a discrete set of points in one dimension, a set of lines in two dimensions and of surfaces in three dimensions. The main difficulty arising with the SI model is the localization of the interface, and more, the computation of quantities defined only at the interface.

In order to alleviate this difficulty, Brackbill et al. [10] proposed the *Continuum Surface Force* (CSF) model which represents the interface as a smoothed transition region of controlled width to ease the computation of surface tension forces, as shown in Fig. 2.1. Instead of applying a pressure jump at the interface (Eq. (1.9)), the CSF model computes a force density distributed in a small region around the interface. Integrating this force density over the small region in the interface normal direction leads to the surface tension force at the interface. The derivatives involved in the calculations are easily computed using smoothly-varying values defined in the smoothed interface. Anderson et al. [2] proposed a review of diffuse-interface models and their application to a wide variety of interfacial phenomena (see also references therein).

2.3 Lagrangian Front Tracking Methods

Two families of numerical methods are used in the literature to account for the two-phase interface movement : lagrangian and eulerian methods. In the context of two-phase flows, lagrangian methods are referred to as Front Tracking Methods, while eulerian methods are referred to as Front Capturing Methods. This section introduces the basics of Front Tracking Methods. The next section addresses Front Capturing Methods.

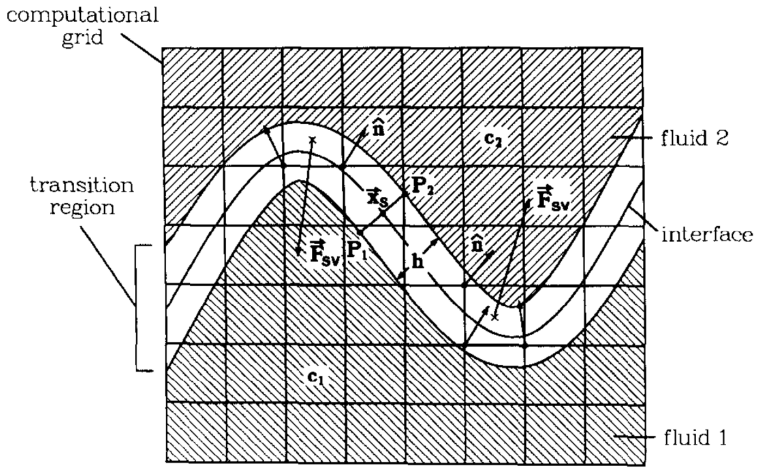


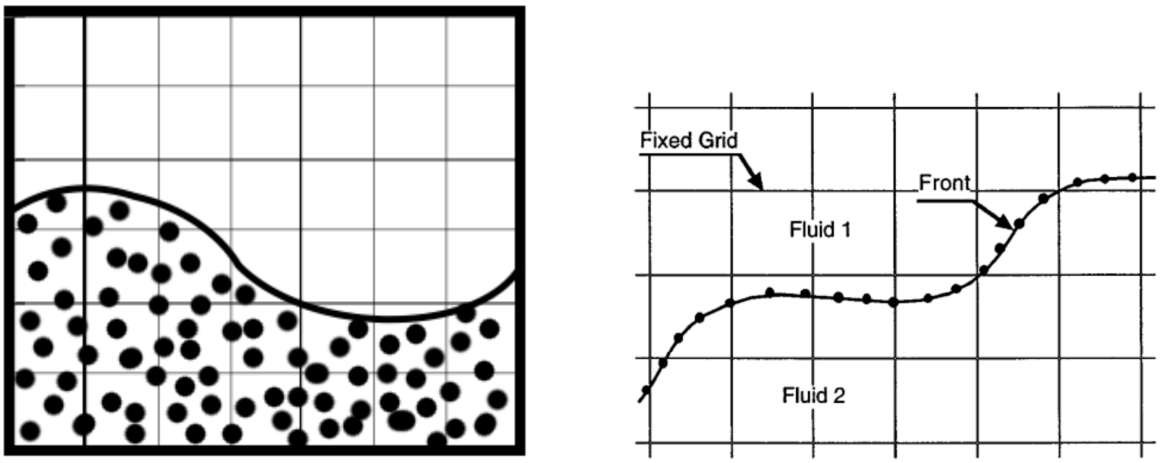
Fig. 2.1 – Sketch of a diffuse interface using the CSF model (excerpt from [10]). A smoothly-varying color function $\tilde{c}$ is defined in the whole computational domain. This function is equal to c_1 in one phase and c_2 in the other phase. The transition region (unshaded) has width h. The interface normal vector $\hat{n}$ is computed at grid nodes located in the transition region by $\hat{n} = \nabla\tilde{c}/\|\nabla\tilde{c}\|$. Surface tension force density, $\mathbf{F}_{\text{sv}}$, is calculated at cell centers from the divergence of $\hat{n}$. The values P_1 and P_2 are local pressure values of fluids 1 and 2, respectively. Point $\mathbf{x}_s$ is lying on the initial interface.

The lagrangian approach of Front Tracking Methods consists in following the interface by means of a distribution of massless particles. These particles are advected by the local fluid velocity field by the advection equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}, \tag{2.1}$$

where $\mathbf{x}$ is the particle location and $\mathbf{u}$ the fluid velocity. Harlow and Welch [32] developed the first front tracking method in 1965, the *Marker-and-Cell* (MAC) method, limited to free surface flows. The phase is seeded with lagrangian markers, as shown in Fig. 2.2(a). Using this method, the geometrical characterization of the interface is quite challenging since markers are not directly located at the interface.

Tryggvason et al. [89, 41, 87] adapted the MAC method by distributing markers only at the interface, as shown in Fig. 2.2(b). After advection by Eq. (2.1), the markers are reconnected to reconstruct the interface. This method is referred to as *the* Front Tracking Method. Two major difficulties arise : one needs to interpolate surface tension forces, known at the markers, to the eulerian grid, and the fluid velocity field, known on the eulerian grid, to the markers to solve Eq. (2.1) ; moreover, marker reconnection can prove highly challenging on complex geometries.



(a) Reproduced from [57]. (b) Reproduced from [87].

Fig. 2.2 – Front Tracking Methods : (a) in the MAC method of Harlow and Welch [32], markers are distributed in all the phase of interest and the interface is defined as the border of the area filled with markers ; (b) Tryggvason et al. [89, 41, 87] adapted the MAC method with markers distributed only at the interface, leading to *the* Front Tracking Method.

2.4 Eulerian Front Capturing Methods

Front Capturing Methods model the interface movement with the transport of a marker function, that identifies one phase w.r.t. the other, by the fluid velocity field using the advection equation

$$\frac{\partial G}{\partial t} + \mathbf{u} \cdot \nabla G = 0, \tag{2.2}$$

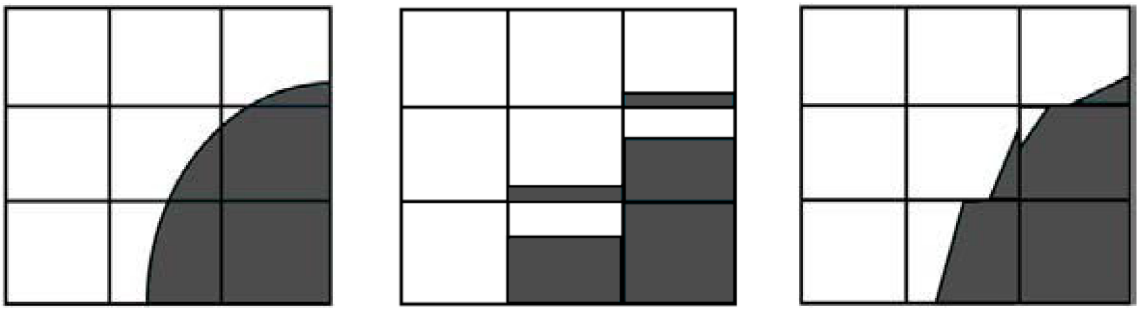
where G is the marker function and $\mathbf{u}$, the fluid velocity.

2.4.1 Volume Of Fluid Method

Hirt and Nichols [35] developed the first eulerian method to capture the interface, namely the *Volume Of Fluid* (VOF) method. The marker function is the liquid volume fraction α, which is substituted to G in Eq. (2.2). The physical quantities of interest are then expressed in terms of α through linear interpolations, i.e.

$$\rho = \alpha\rho_1 + (1-\alpha)\,\rho_2, \tag{2.3}$$
$$\mu = \alpha\mu_1 + (1-\alpha)\,\mu_2, \tag{2.4}$$

where ρ is the density, μ, the dynamic viscosity, and the indices 1 and 2 denote the two phases. Figure 2.3(a) shows the values of α on a two-dimensional cartesian grid. The liquid volume fraction does not give any information about the shape of the interface. The only information available is $0 < \alpha < 1$ in grid cells crossed by the interface. In order to reconstruct the interface, Hirt and Nichols [35] proposed the *Simple Line Interface Calculation* (SLIC) method, in which the interface is reconstructed with per-cell horizontal lines, as shown in Fig. 2.3(b). Youngs [98] improved the



(a) (b) (c)

Fig. 2.3 – The dark area represents the liquid phase, the white area represents the vapor phase. The real interface is shown in (a), its SLIC reconstruction, in (b), and its PLIC reconstruction, in (c).

method and proposed the *Piecewise Linear Interpolation Construction* (PLIC) method, in which the interface orientation is determined by the neighboring values of α, as shown in Fig. 2.3(c). Mass conservation is an intrinsic feature of the VOF method. The major drawback of this method is the difficulty to compute an accurate interface curvature after reconstruction with either the SLIC or PLIC method.

2.4.2 Level Set Methods

Since the interface is a subset of null measure of the set of real numbers, the interface can be seen as a given iso-level of a real function defined in the computational domain.

2.4.2.1 The Signed Distance Function to the interface

Osher and Sethian [60] proposed to use the *Signed Distance Function* (SDF) ϕ to the interface Γ defined for any node $\mathbf{x}$ in the computational domain Ω by

$$\phi\left(t,\mathbf{x}\right)=\min_{\mathbf{x}_\Gamma\in\Gamma(t)}\left\|\mathbf{x}-\mathbf{x}_\Gamma\right\|. \tag{2.5}$$

The interface is identified as the 0 iso-level of the level set function. The signed distance function is advected by solving the standard advection equation

$$\frac{\partial\phi}{\partial t}+\mathbf{u}\cdot\nabla\phi=0, \tag{2.6}$$

where $\mathbf{u}$ is the fluid velocity.

After advection, the error due to the numerical scheme used to solve Eq. (2.6) and the non-uniformity of $\mathbf{u}$ are responsible for the deviation of ϕ from the signed distance function to the new interface position. The function ϕ is then reinitialized as a signed distance function to the interface. The literature on reinitialization methods of level set functions is divided into two families of methods : the *Hamilton-Jacobi* (HJ) equation and the *Fast Marching Method* (FMM). The

Hamilton-Jacobi partial differential equation (PDE) is given by

$$\frac{\partial \phi}{\partial \tau} + S\left(\phi^0\right)\left(\|\nabla \phi\| - 1\right) = 0, \tag{2.7}$$

where τ is a pseudo-time, S is a smoothed sign function typically given by Sussman et al. [83], and ϕ^0 is the previously advected level set that need be reinitialized. Equation (2.7) is solved in pseudo-time until convergence, i.e. until $\|\nabla \phi\| = 1$, which is part of the definition of the signed distance function. In order to avoid perturbations in the interface normal vector and curvature (see below), the term $\nabla \phi$ in Eq. (2.7) is computed using a high-order scheme such as the Fifth-Order WENO schemes developed by Shu [78] or Jiang and Peng [40]. High-order schemes require higher computational cost, are difficult to implement on unstructured grids and may reduce performance in a parallel code.

Another method for the reinitialization of the signed distance function after advection by Eq. (5.28) is to consider the signed distance function to the interface as the solution of an Eikonal equation. As detailed in Appendix A of [46], in the theory of wave propagation, the wave equation

$$\frac{\partial^2 f}{\partial t^2}(t, \mathbf{x}) = c^2 \nabla^2 f(t, \mathbf{x}), \tag{2.8}$$

where $f : [0, \infty[\times \mathbb{R}^n \longrightarrow \mathbb{R}$ is a given function describing a wave and $c > 0$ is the wave speed, can be rewritten, in the *geometric optics approximation*, as the Eikonal equation

$$\|\nabla \phi(t, \mathbf{x})\|^2 = \left(\frac{c_0}{c}\right)^2, \tag{2.9}$$

where the iso-surfaces $\{\phi(t, \mathbf{x}) = \text{constant}\}$ are the wave fronts, c_0 being the velocity in vacuum and c the phase velocity of the wave in the given medium. If the medium is the vacuum, the wave is travelling at speed $c = c_0$, and one has the Eikonal equation

$$\|\nabla \phi\| = 1. \tag{2.10}$$

Note that if boundary condition $\phi = 0$ is imposed at the interface, then ϕ in Eq. (2.10) is the signed distance function to the interface. Eikonal equation (2.10) can be seen as the stationary version of the Hamilton-Jacobi equation (2.7) and can be solved by the Fast Marching Method [76, 44]. The solution of equation (2.10) is based on the propagation of the signed distance function values from the interface along the interface normal direction. The advantage of the Fast Marching Method compared to the resolution of the Hamilton-Jacobi equation (2.7) is that the FMM does not require high-order schemes to compute $\nabla \phi$. The drawback of the FMM is the need to sort node lists efficiently to ensure fast access to specific ϕ values.

The interface normal vector is then computed as

$$\mathbf{n} = \frac{\nabla \phi}{\|\nabla \phi\|}. \tag{2.11}$$

Since ϕ is the signed distance function to the interface, one could expect $\mathbf{n}$ to be defined as $\mathbf{n} = \nabla \phi$. However, while mathematically true, this definition could introduce oscillations in $\mathbf{n}$. It is indeed numerically very difficult to get a normalized gradient of ϕ, as one would expect from a distance function. On one hand, numerical errors in the advection and reinitialization of ϕ are inevitable ;

on the other hand, the numerical scheme used to compute the gradient operator also introduces numerical errors. As a result, it is very unlikely that $\nabla\phi$ is normalized. The interface curvature must be computed to take into account surface tension forces in the pressure jump at the interface given by Eq. (1.31). Mathematically, the interface curvature κ is defined as the divergence of the interface normal vector $\mathbf{n}$, i.e.

$$\kappa = -\nabla \cdot \mathbf{n} \tag{2.12}$$

$$= -\nabla \cdot \left(\frac{\nabla\phi}{\|\nabla\phi\|} \right), \tag{2.13}$$

which, if $\|\nabla\phi\| = 1$, simplifies to

$$\kappa = -\nabla^2\phi, \tag{2.14}$$

where ∇^2 is the laplacian operator. Numerically, this computation requires particular attention. Computing the gradient of ϕ on one node demands access to the ϕ values on the closest neighbors of this node. Then computing the divergence of this gradient demands access to ϕ values on the neighbors of the neighbors of the initial node. This has the side effect of filtering high frequencies of ϕ, leading to a potential smoothing of the curvature.

2.4.2.2 The Conservative Level Set function

In order to improve mass conservation in each phase, Olsson and Kreiss [58] and Olsson et al. [59] proposed the *Conservative Level Set* (CLS) method in which the level set function ψ is a shifted smeared-out Heaviside function defined for $\mathbf{x} \in \Omega$ by

$$\psi(t, \mathbf{x}) = \frac{1}{1 + \exp\left(-\frac{\phi(t,\mathbf{x})}{\epsilon(\mathbf{x})}\right)}, \tag{2.15}$$

where ϕ is the signed distance function to the interface and ϵ is a parameter of the cell size order controlling the thickness of the profile. Equation (2.15) is usually rewritten

$$\psi(t, \mathbf{x}) = \frac{1}{2}\left(1 + \tanh\left(\frac{\phi(t, \mathbf{x})}{2\epsilon(\mathbf{x})}\right)\right). \tag{2.16}$$

The interface is identified as the 0.5 iso-level of ψ, and ψ is uniform in each phase ($\psi = 0$ or $\psi = 1$) starting from a small distance from the interface. The CLS function is advected by the standard conservative advection equation

$$\frac{\partial\psi}{\partial t} + \nabla \cdot (\psi\mathbf{u}) = 0, \tag{2.17}$$

where $\mathbf{u}$ is the fluid velocity. The divergence in the lhs of Eq. (2.17) can be rewritten

$$\nabla \cdot (\psi\mathbf{u}) = \mathbf{u} \cdot \nabla\psi + \psi\nabla \cdot \mathbf{u}, \tag{2.18}$$

and, for an incompressible flow ($\nabla \cdot \mathbf{u} = 0$), simplifies to

$$\nabla \cdot (\psi\mathbf{u}) = \mathbf{u} \cdot \nabla\psi. \tag{2.19}$$

For incompressible flows, Eq. (2.17) is then equivalent to the non-conservative advection equation

$$\frac{\partial\psi}{\partial t} + \mathbf{u} \cdot \nabla\psi = 0. \tag{2.20}$$

One advantage of the CLS is the use of the conservative advection equation (2.17) which implies that the quantity ψ is conserved during the simulation, improving the localization of the interface $\{\psi = 0.5\}$. Another advantage of the CLS is the ability to compute at time t the mass m_+ of the phase in which $\psi > 0.5$ by

$$m_+(t) = \int_\Omega \rho(\mathbf{x})\psi(t,\mathbf{x})\,\mathrm{d}\Omega, \tag{2.21}$$

and the mass m_- of the phase in which $\psi < 0.5$ by

$$m_-(t) = \int_\Omega \rho(\mathbf{x})\left(1-\psi(t,\mathbf{x})\right)\,\mathrm{d}\Omega, \tag{2.22}$$

where Ω is the computational domain. Consequently, the total mass m present at time t in Ω is given by

$$m(t) = m_+(t) + m_-(t). \tag{2.23}$$

The major drawback of level set functions is their need of a reinitialization step after advection. Indeed, the advection equation (2.17) tends to deform the hyperbolic tangent profile of ψ. This deformation can lead to severe deviations of the expected interface shape, and even make the simulation crash if not corrected during the time integration. In order to correct this deformation, a reinitialization step is performed after advection to reinitialize, or reshape, the hyperbolic tangent profile of ψ. This step has no physical meaning and is only a numerical requirement. Special care must be taken in this step which is subject to two constraints :

1. The hyperbolic tangent profile must be recovered.

2. The interface position must not be changed during reinitialization since the advection equation (2.17) has already been solved.

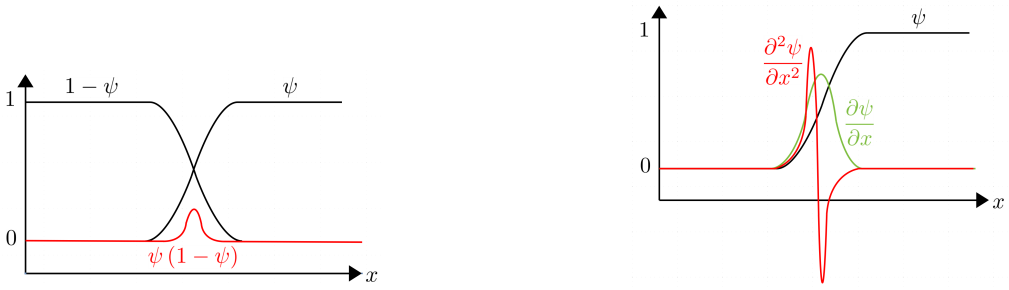
Olsson et al. [58, 59] proposed the reinitialization equation for the CLS function given by

$$\frac{\partial\psi}{\partial\tau}(\tau,\mathbf{x}) + \nabla\cdot\left(\psi(\tau,\mathbf{x})\left(1-\psi(\tau,\mathbf{x})\right)\mathbf{n}(t,\mathbf{x})\right) = \nabla\cdot\left(\epsilon(\mathbf{x})\left(\nabla\psi(\tau,\mathbf{x})\cdot\mathbf{n}(t,\mathbf{x})\right)\mathbf{n}(t,\mathbf{x})\right), \tag{2.24}$$

where the first term of the lhs is the pseudo-temporal variation, the second term is a compression term in the normal direction to reshape the hyperbolic tangent profile, and the rhs is a diffusion term to enforce the characteristic thickness ϵ of the profile. Equation (2.24) is solved in pseudo-time τ, highlighting the fact that the reinitialization step is purely a numerical constraint. This reinitialization slightly changes the position of the interface on a sub-grid scale. In Eq. (2.24), the interface normal vector $\mathbf{n}$ is given by

$$\mathbf{n}(t,\mathbf{x}) = \frac{\nabla\psi(t,\mathbf{x})}{\|\nabla\psi(t,\mathbf{x})\|}. \tag{2.25}$$

As stated in [19], even with an accurate reinitialization algorithm, ψ can locally deviate from an *exact* hyperbolic tangent function, introducing spurious currents in the normal vector. For this reason, in the *Accurate Conservative Level Set* (ACLS) method [19], the authors first recompute the new signed distance function to the interface ϕ by the Fast Marching Method, and then compute the normal vector from ϕ by $\mathbf{n} = \nabla\phi/\|\nabla\phi\|$.



(a) Compression in the lhs.

(b) Diffusion in the rhs.

Fig. 2.4 – The compression and diffusion effects in the reinitialization equation of the Conservative Level Set function proposed by Olsson et al. [58, 59].

In order to illustrate the compression and diffusion effects of respectively the lhs and rhs of Eq. (2.24), we consider the one-dimensional version of Eq. (2.24) given by

$$\frac{\partial\psi}{\partial\tau}+\frac{\partial}{\partial x}\left(\psi\left(1-\psi\right)\right)=\frac{\partial}{\partial x}\left(\epsilon\left(\frac{\partial\psi}{\partial x}\right)\right). \tag{2.26}$$

On a uniform grid, the ϵ parameter can be dropped out of the gradient, leading to

$$\frac{\partial\psi}{\partial\tau}+\frac{\partial}{\partial x}\left(\psi\left(1-\psi\right)\right)=\epsilon\frac{\partial^2\psi}{\partial x^2}. \tag{2.27}$$

Figure (2.4) illustrates the compression effect of the lhs and the diffusion effect of the rhs of Eq. (2.27). Nevertheless, the necessity to set ϵ in Eqs. (2.16) and (2.24) to a small value to improve volume conservation produces sharp gradients in ψ and thus potential oscillations in $\mathbf{n}$ by Eq. (2.25).

In order to avoid this problem, and with the aim of increasing accuracy, Chiodi and Desjardins [13] proposed a new reinitialization method for the Conservative Level Set function on cartesian grids. In this method, Eq. (2.24) is rewritten

$$\frac{\partial\psi}{\partial\tau}\left(\tau,\mathbf{x}\right)=\nabla\cdot\left(\frac{1}{4\cosh^2\left(\dfrac{\phi_{\text{map}}\left(\tau,\mathbf{x}\right)}{2\epsilon\left(\mathbf{x}\right)}\right)}\left(\left|\nabla\phi_{\text{map}}\left(\tau,\mathbf{x}\right)\cdot\mathbf{n}\left(t,\mathbf{x}\right)\right|-1\right)\mathbf{n}\left(t,\mathbf{x}\right)\right), \tag{2.28}$$

where the inverse of the Conservative Level Set function ϕ_{map} is given by

$$\phi_{\text{map}}\left(\tau,\mathbf{x}\right)=\epsilon\left(\mathbf{x}\right)\log\left(\frac{\psi\left(\tau,\mathbf{x}\right)}{1-\psi\left(\tau,\mathbf{x}\right)}\right), \tag{2.29}$$

and the interface normal vector $\mathbf{n}$ is computed as

$$\mathbf{n}\left(t,\mathbf{x}\right)=\frac{\nabla\phi_{\text{FMM}}\left(t,\mathbf{x}\right)}{\left\|\nabla\phi_{\text{FMM}}\left(t,\mathbf{x}\right)\right\|}, \tag{2.30}$$

ϕ_{FMM} being the signed distance function to the interface computed by the Fast Marching Method.

2.4.2.3 Equivalence of the Conservative Level Set and Accurate Conservation Level Set reinitialization equations

We now demonstrate that Eq. (2.28) is equivalent to Eq. (2.24). To this purpose, we first give two useful identities : for all $s \in \mathbb{R}$, one has

$$\tanh^2(s) = 1 - \frac{1}{\cosh^2(s)}, \tag{2.31}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}s}\tanh(s) = \frac{1}{\cosh^2(s)}. \tag{2.32}$$

Then one has

$$\psi(\tau,\mathbf{x})(1-\psi(\tau,\mathbf{x})) = \underbrace{\frac{1}{2}\left(1+\tanh\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)\right)}_{\psi(\tau,\mathbf{x})} \cdot \underbrace{\frac{1}{2}\left(1-\tanh\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)\right)}_{1-\psi(\tau,\mathbf{x})} \tag{2.33}$$

$$= \frac{1}{4}\left(1-\tanh^2\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)\right), \tag{2.34}$$

which by Eq. (2.31) leads to

$$\psi(\tau,\mathbf{x})(1-\psi(\tau,\mathbf{x})) = \frac{1}{4\cosh^2\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)}. \tag{2.35}$$

The compression term in the lhs of Eq. (2.24) is then given by

$$\nabla\cdot\left(\psi(\tau,\mathbf{x})(1-\psi(\tau,\mathbf{x}))\mathbf{n}(t,\mathbf{x})\right) = \nabla\cdot\left(\frac{1}{4\cosh^2\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)}\mathbf{n}(t,\mathbf{x})\right). \tag{2.36}$$

Moreover, Eq. (2.32) implies

$$\nabla\psi(\tau,\mathbf{x}) = \frac{1}{2\cosh^2\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)}\nabla\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right) \tag{2.37}$$

$$= \frac{1}{2\cosh^2\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)}\frac{2\epsilon(\mathbf{x})\nabla\phi(\tau,\mathbf{x}) - 2\phi(\tau,\mathbf{x})\nabla\epsilon(\mathbf{x})}{4\epsilon^2(\mathbf{x})}, \tag{2.38}$$

where $\nabla\epsilon(\mathbf{x}) = \mathbf{0}$ on uniform grids (and may reasonably be considered negligible on weakly stretched grids), leading to

$$\nabla\psi(\tau,\mathbf{x}) = \frac{1}{4\epsilon(\mathbf{x})\cosh^2\left(\frac{\phi(\tau,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)}\nabla\phi(\tau,\mathbf{x}). \tag{2.39}$$

The diffusion term in the rhs of Eq. (2.24) is then given by

$$\nabla \cdot \left(\epsilon \left(\mathbf{x} \right) \left(\nabla \psi \left(\tau, \mathbf{x} \right) \cdot \mathbf{n} \left(t, \mathbf{x} \right) \right) \mathbf{n} \left(t, \mathbf{x} \right) \right) = \nabla \cdot \left(\left\{ \frac{1}{4 \cosh^2 \left(\dfrac{\phi \left(\tau, \mathbf{x} \right)}{2 \epsilon \left(\mathbf{x} \right)} \right)} \nabla \phi \left(\tau, \mathbf{x} \right) \cdot \mathbf{n} \left(t, \mathbf{x} \right) \right\} \mathbf{n} \left(t, \mathbf{x} \right) \right). \tag{2.40}$$

Using Eqs. (2.36) and (2.40), Eq. (2.24) can be rewritten as Eq. (2.28) where the compression term is written in the rhs, ϕ_{map} is substituted to ϕ and the absolute value of $\nabla \phi_{\text{map}} \cdot \mathbf{n}$ is taken into account for the case where ϕ_{FMM} is defined to be positive in the phase where $\psi < 0.5$. Finally, we show that ϕ_{map} defined by Eq. (2.29) is equal to ϕ. At any pseudo-time τ during reinitialization, an approximation of the signed distance function to the interface $\phi \left(\tau, \mathbf{x} \right)$ can be recovered from the (partially) reinitialized conservative level set function $\psi \left(\tau, \mathbf{x} \right)$ by

$$\phi \left(\tau, \mathbf{x} \right) = 2 \epsilon \left(\mathbf{x} \right) \operatorname{atanh} \left(2 \psi \left(\tau, \mathbf{x} \right) - 1 \right). \tag{2.41}$$

Since for $s \in]-1; 1[$, one has

$$\operatorname{atanh} \left(s \right) = \frac{1}{2} \log \left(\frac{1+s}{1-s} \right), \tag{2.42}$$

taking $s = 2 \psi \left(\tau, \mathbf{x} \right) - 1$ and substituting Eq. (2.42) into Eq. (2.41) establishes the equality of ϕ and ϕ_{map}.

The signed distance function is (theoretically only) easier to advect and reinitialize than the Conservative Level Set function, but does not present any conservation property. Conversely, the Conservative Level Set exhibits interesting conservation properties but is challenging to advect and reinitialize accurately.

2.4.2.4 Level Set Methods with phase change

In the presence of phase change, the mass transfer rate $\dot{m}$ contributes to the interface movement (see Section 1.3.5). As such, $\dot{m}$ appears in the advection equation of the level set function. The detailed derivation of the exact advection equation for both the signed distance function and the conservative level set function will be found in Chapter 5.

2.5 Numerical treatment of the discontinuities at the interface and computation of the mass transfer rate with the Ghost Fluid Method

In the context of the Sharp Interface model, Fedkiw et al. [22] proposed the *Ghost Fluid Method* (GFM) to treat the discontinuities at the interface. Each physical field is divided in one field per phase. Each field of each phase is artificially extended (or extrapolated) beyond the interface. The accuracy of the extrapolation depends on the method used (see below) and is independent from the GFM. The superposition at one node of the fields of the two phases enables the computation of a discontinuity, or *jump*, at this node. Nevertheless, this jump, in the formulation of continuum mechanics, is defined only at the interface, and thus has be to interpreted with particular attention. The extended values are called ghost values. These ghost values permit the computation of derivatives across the interface since the discontinuity is artificially removed for the field of a

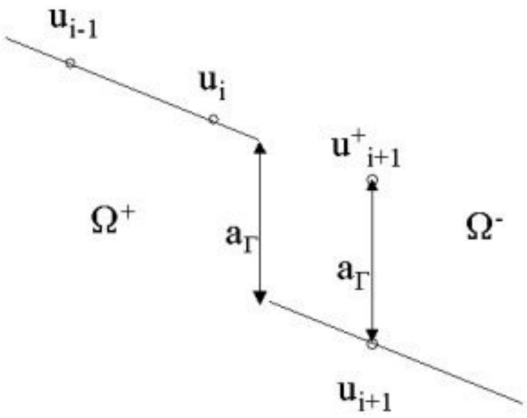


Fig. 2.5 – Illustration of the Ghost Fluid Method in one dimension where the interface Γ separating subdomains Ω^+ and Ω^- is located between nodes i and $i+1$. The physical field u is extrapolated across the interface from node i to node $i+1$, where the ghost value is noted u^+_{i+1}, artificially removing the discontinuity a_Γ of u at the interface.

given phase. Gibou et al. [27, 26] used the GFM with linear extrapolation of the temperature fields to compute the mass transfer rate $\dot{m}$ at the interface. Tanguy et al. [85] extended the accuracy of the method by using high-order extensions performed with the extrapolation technique proposed by Aslam [4] and detailed in Appendix G.

2.6 The Projection Method

Incompressible Navier-Stokes equations (1.4) are classically solved by means of the projection method based on fractional time steps developed by Chorin [14] and improved by Kim and Moin [43]. The velocity is solved at overall iteration times (n, $n+1$, etc) whereas the pressure and the density are solved at half iteration times ($n+\frac{1}{2}$, $n+\frac{3}{2}$, etc). Projection methods are based on Helmholtz decomposition which states that a two- or three-dimensional vector field $\mathbf{u}$ can be expressed as the sum of a solenoidal (divergence-free) component $\mathbf{u}_\Psi$ and an irrotational (curl-free) component $\mathbf{u}_\Phi$,

$$\mathbf{u} = \mathbf{u}_\Psi + \mathbf{u}_\Phi. \tag{2.43}$$

The irrotational component $\mathbf{u}_\Phi$ derives from a scalar potential A, i.e. one has $\mathbf{u}_\Phi = \nabla A$. The divergence operator applied to Eq. (2.43) leads to the Poisson equation

$$\nabla \cdot \mathbf{u} = \nabla \cdot \mathbf{u}_\Phi = \nabla^2 A. \tag{2.44}$$

In the projection method, a velocity predictor $\mathbf{u}^*$ is first computed from Eq. (1.4) from which the pressure gradient at time $n-\frac{1}{2}$ is dropped, i.e.

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\left(\mathbf{u}^n \cdot \nabla\right)\mathbf{u}^n + \frac{1}{\rho^{n-\frac{1}{2}}}\nabla \cdot \left(\mu\left(\nabla \mathbf{u}^n + \left(\nabla \mathbf{u}^n\right)^T\right)\right) + \mathbf{g}. \tag{2.45}$$

The velocity predictor $\mathbf{u}^*$ is *a priori* not divergence-free. Second, the velocity predictor is corrected using the pressure gradient at time $n+\frac{1}{2}$,

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^*}{\Delta t} = -\frac{1}{\rho^{n+\frac{1}{2}}}\nabla P^{n+\frac{1}{2}}. \tag{2.46}$$

Equation (2.46) has two unknowns, $\mathbf{u}^{n+1}$ and $P^{n+\frac{1}{2}}$ ($\rho^{n+\frac{1}{2}}$ only depends on the phase). Since $\mathbf{u}^{n+1}$ is divergence-free, taking the divergence of Eq. (2.46), one obtains the Poisson equation for the updated pressure $P^{n+\frac{1}{2}}$,

$$\nabla\cdot\left(\frac{1}{\rho^{n+\frac{1}{2}}}\nabla P^{n+\frac{1}{2}}\right) = \frac{1}{\Delta t}\nabla\cdot\mathbf{u}^*, \tag{2.47}$$

where $P^{n+\frac{1}{2}}$ is the only unknown. Numerical discretization of the Poisson equation (2.47) leads to a linear system in which the pressure jump at the interface, that will appear as a source term in the rhs, is imposed using the Ghost Fluid Method (see Section 2.5). Once the updated pressure $P^{n+\frac{1}{2}}$ is known, Eq. (2.46) is used to correct $\mathbf{u}^*$ in order to compute the updated velocity $\mathbf{u}^{n+1}$. In this correction, since the pressure gradient is also computed close to the interface, the pressure jump at the interface is again imposed in the discretization of $\nabla P^{n+\frac{1}{2}}$ in Eq. (2.46).

When phase change is considered, one has to account for the velocity jump at the interface (Eq. (1.30)) in the Poisson equation (2.47). As will be detailed in Chapter 4, a two-velocity formulation coupled to the Ghost Fluid Method can be used to impose the velocity jump in the Poisson equation [85].

2.7 Literature review on numerical methods for simulations of two-phase flows with phase change

Abundant literature exists on numerical methods for simulations of two-phase flows with phase change. We mention here some milestones as well as state-of-the-art numerical methods for such simulations.

2.7.1 Early works

Direct Numerical Simulations (DNS) of boiling flows were pioneered by Welch [94] in 1995. The author developed a two-dimensional, moving-mesh finite-volume method for a single, weakly deformable bubble. A simple interface model based on surface tension and surface energy is used. The mass transfer rate at the interface is computed using moving grid triangles. The computations are performed on interface-dependent triangular grids. The results showed basic capabilities to track interfaces with phase change. Son and Dhir [80] used the Continuum Surface Force model (see Section 2.2) to compute the mass transfer rate and the velocity discontinuity at the interface in the context of nucleate boiling to investigate bubble release pattern depending on the heat flux. The interface is captured by a level set method. Juric and Tryggvason [42] presented a numerical method to simulate liquid-vapor phase change in which the interface is tracked by a Front Tracking method (see Section 2.3). Interfacial source terms for surface tension, mass transfer and latent heat are added as indicator functions of the interface. The authors used finite differences on cartesian grids to simulate film boiling and bubble departure. Based on the work of Welch [94], Welch and Wilson [95] presented in 2000 two-dimensional numerical simulations of incompressible two-phase

flows with phase change, using a VOF-based interface tracking method in conjunction with a mass transfer model and a model for surface tension. The authors derived a one-dimensional analytical test problem featuring a thin thermal layer propagating with the moving phase interface. The authors emphasized that this test problem isolates the ability of a method to accurately calculate the thermal layers responsible for driving the mass transfer in boiling flows. A simulation of horizontal film boiling is also provided.

2.7.2 Coupling between Level Set and Ghost Fluid Method

As stated in Chapter 1, boiling is driven by the mass transfer rate at the interface computed from the discontinuity of the thermal fluxes on both sides of the interface (Eq. (1.26)). Numerical methods able to accurately take discontinuities at the interface into account are then of high interest for phase change simulations. In 1999, Fedkiw et al. [22] proposed the Ghost Fluid Method (GFM) to provide a sharp treatment of the discontinuities at the interface. In the GFM, a physical field defined on one phase is continuously extended across the interface to the other phase in order to populate "ghost cells". This technique, applied to both liquid and vapor fields constituting the same physical field, each having a physical meaning only in their respective phase, allows knowledge of one physical value and one extrapolated, or "ghost", value at the same grid node. Consequently, the discontinuity of the physical field at this node is simply given by the substraction of one value from the other one (see also Section 2.5). Since then, the GFM has become a major ingredient in phase change simulations, as detailed below.

In 2007, Tanguy et al. [84] used a level set function to capture the interface coupled to the ghost fluid method to simulate vaporizing flows (liquid-to-vapor phase change with multi-component species) in two-dimensional cartesian grids. The method is validated against various test cases of vaporizing droplets yielding good agreement with analytical results. Also in 2007, Gibou et al. [26] used the same coupling for two-dimensional film boiling simulation, again with good accuracy. These two references demonstrate the efficiency of the level set/ghost fluid method coupling, and thus mark a milestone in two-phase flow simulations with phase change.

In 2014, Tanguy et al. [85] proposed a comparison of different numerical methods suited to simulations of two-phase flows with phase change. The liquid-vapor interface is captured using the level set method. The accuracies of the ghost fluid method (sharp interface) and the delta function method (smooth interface) are compared to compute the normal velocity jump condition. The authors showed that smoothing the velocity jump condition at the interface could lead to a misleading mass prediction, whereas the ghost fluid method performed well that test. Moreover, the authors demonstrated that high order extrapolation methods on the thermal field allowed performing accurate and robust simulations for a thermally controlled bubble growth. Simulations of the growth of static and rising bubbles are presented. The computations are performed on two-dimensional structured cartesian grids.

In order to put into perspective the developments realized in the present thesis, we emphasize that this thesis started in 2015. The references given below have been published since then.

In 2017, Huber et al. [36] applied the method of Tanguy et al. [85] to present a DNS of nucleate boiling on 2D axisymmetric cartesian grids. A single site with a large microscopic contact angle and a high density ratio between the two phases is considered. The authors have studied the influence on the numerical solution of the thermal conduction in the solid heater and concluded that this parameter had no influence when thick and highly conductive materials were considered. The authors also came to the conclusion that the implementation of a micro-region model (see

Fig. A.4) in the neighborhood of the contact line and its coupling with the overall solver was not required to perform well-resolved and accurate numerical simulations in the case of high density ratio, high microscopic contact angle (up to 30°) and moderate Jakob number (ratio of sensible heat to latent heat absorbed, or released, during phase change). Moreover, the authors proposed a new correlation on the bubble detachment radius depending on the Jakob number. Lee et al. [48] proposed to use a smooth distribution of sharp velocity jumps and mass flux within a narrow region surrounding the interface, an improved mass flux projection from the implicit interface onto the uniform cartesian grid and a post-advection velocity correction step to ensure accurate velocity divergence in interfacial cells. A sharp treatment is used to take the discontinuities of pressure and temperature gradients into account. The methodology is validated with results of axisymmetric film boiling. Lee and Son [47] addressed the growth and collapse of a compressible vapor bubble, in which the interface tracking method is extended to include the effects of bubble compressibility and liquid-vapor phase change. To this purpose, the ghost fluid method is used to compute the discontinuities at the interface. Yap et al. [97] proposed numerical simulations of three-phase flows with phase change (two fluids among which only one admits phase change). The authors then used two level set functions to capture the two interfaces involved in the problem, i.e. the interface between two fluids and the interface between two phases of the same fluid. The surface tension is treated using the continuum surface force model. The model is validated against different three-phase flow with phase change problems involving a liquid with a rising condensing vapor bubble and a falling immiscible liquid droplet, and solidification in stratified two-fluid flow with a growing solid layer. In 2018, Anumolu and Trujillo [3] used second-order one-sided differences to discretize the temperature laplacian, i.e. the numerical stencil completely resides within each respective phase, to solve the heat equation in phase change simulations. Perez-Raya and Kandlikar [63] used a single cell around the interface to compute the interfacial temperature gradient, and a linear interpolation normal to the interface to compute the temperature of the cell containing the interface. Shao et al. [77] proposed a DNS of multi-component flows with phase change in the context of combustion. The advection of the level set function is discretized by a semi-lagrangian scheme in which phase change is modelled by a source term. In the context of nucleate pool boiling, Urbano et al. [91] applied the method of Tanguy et al. [85] to investigate the physical mechanisms associated with the evolution of a micro-layer beneath a bubble and the transition between contact line and micro-layer regimes. The authors concluded that neglecting the micro-layer would lead to erroneous results because it has a strong influence on the overall bubble growth. Perez and Kandlikar [64] proposed numerical simulations of nucleate boiling with a sharp interface around which interpolation functions estimate the temperature at points located in a direction normal to the interface. A linear temperature profile at the interface interpolates the temperature of the interface-cells. In 2019, Urbano et al. [90] applied the method of Tanguy et al. [85] in zero gravity conditions, in the context of space applications. The authors have established and verified numerically an analytical model for the equilibrium radius reached by a bubble nucleated in subcooled conditions. To this purpose, DNS of nucleate boiling are proposed in which the heat conduction in the solid wall is taken into account. Also in 2019, Rajkotwala et al. [67] provided a comparison of smooth and sharp interface methods for numerical simulations of two-phase flows with phase change. The authors used a hybrid front tracking method without connectivity, which can easily handle complex topological changes. The expansion due to phase change is incorporated as a non-zero divergence condition at the interface. The energy equation is treated with two different approaches: smooth interface approach and sharp interface approach. In both methods, the saturation temperature is imposed at the interface. The numerical results underline certain advantages of the sharp interface approach over the smooth

interface approach such as better accuracy and convergence rate, reduced fluctuations in the velocity field and a physically bounded temperature field near the interface. Both approaches are validated against the analytical test case of a three-dimensional vapor bubble growing in a superheated liquid, on cartesian grids. The initial bubble radius is 0.15 m and the computational domain is a cube of size 1 m. The results showed that the bubble radius converges with grid refinement in L^2 norm, but no information is provided towards convergence of the bubble radius in L^∞ norm.

2.7.3 Coupling between VOF and Ghost Fluid Method

In 2018, Zhang and Ni [99] developed a new phase change model for the simulation of incompressible multiphase magnetohydrodynamics (study of the magnetic properties and behavior of electrically conducting fluids) based on the VOF method. In order to decrease the pressure oscillations when large density ratios are present between the liquid phase and the vapor phase, a smooth distribution of sharp mass transfer rate in the narrow band around the interface is adopted, and a ghost fluid approach is used to impose the saturation temperature at the liquid-vapor interface when solving the energy equation.

In 2019, Palmore and Desjardins [61] presented numerical simulations of vaporizing two-phase flows with large density ratios using an unsplit VOF method to transport the interface. The mass transfer rate is computed from the thermal fluxes at the interface. A novel, divergence-free extrapolation technique, similar to that of Tanguy et al. [85], is used to create a velocity field that is suitable for interface transport. Sharp treatments are used for the vapor mass fractions and temperature fields. The Poisson equation for the pressure is treated using the ghost fluid method. The overall method is validated in one dimension and tested in two dimensions against the test case of vaporization of a curved interface. In two dimensions, the authors obtained convergence of the liquid temperature with grid refinement in the L^1 sense but not in the L^∞ sense. The authors then emphasize the difficulty to reach L^∞ convergence and how L^1 convergence, while commonly reported in the literature, can be misleading for numerical analysis of flows with sharp features or discontinuities.

2.7.4 Coupling between Level Set, VOF and Ghost Fluid Method

In 2005, Tomar et al. [86] used a coupled level set and volume-of-fluid (a.k.a. CLSVOF) method for modeling incompressible two-phase flows with surface tension. A simulation of film boiling and bubble formation in refrigerant R134a is provided. The authors studied the effect of saturation pressure on the frequency of bubble formation.

In 2018, Singh and Premachandran [79] proposed the first extension of the CLSVOF method to two-dimensional unstructured grids for simulations of two-phase flows including phase change. The methodology is validated with results on saturated film boiling on a horizontal flat plate (in excellent agreement in comparison with results available in the literature using structured grids), and natural convection film boiling over a horizontal cylinder (also in good agreement with semi-empirical correlations). The boiling simulations presented do not exhibit analytical solutions.

2.7.5 Comments

From this review, one can see that numerical methods for simulations of two-phase flows including phase change constitute a quite active research field. At the time of writing this thesis, in 2019, numerical treatment of phase change in two-phase flows is still a highly challenging task,

as confirmed by the difficulty to reach L^∞ convergence of interface-related quantities (distance, velocity, temperature, etc) with grid refinement.

Accuracy of such numerical methods is even more challenging to reach on unstructured grids, for which the literature is much more thinner.

Moreover, the physical mechanisms involved in nucleate boiling still need to be clarified. To this purpose, numerical simulations can be performed without taking phase change into account. In this context, we mention the work of Guion et al. [29] where the VOF method is used to reproduce the hydrodynamics of hemispherical bubble growth at the wall, in order to resolve the formation of the liquid microlayer beneath the vapor bubble.

Chapter 3

Numerical simulation of two-phase flows without phase change

This chapter presents the starting point of our work. First, a detailed presentation of the YALES2 library is given. Second, we focus on its ability to simulate two-phase flows without phase change and detail all the implemented ingredients to achieve such simulations.

Outline

3.1 Presentation of YALES2

YALES2 is a library of numerical solvers developed for realistic turbulent two-phase flow simulations with low Mach numbers. YALES2 is the acronym of Yet Another LES Solver where LES stands for Large-Eddy Simulations, as explained below. The code is developed and maintained since 2007 by V. Moureau and his research team "Numerical simulation and modeling of turbulent combustion" at CORIA laboratory (CNRS) near Rouen, France. Several research laboratories including LEGI actively contribute to its development. YALES2 originally aims at solving two-phase combustion from primary atomization to pollutant prediction on massive complex meshes. It has also been largely used in hydraulics problems like accurate computing of the flow in hydraulic or wind turbines. It is able to handle efficiently unstructured meshes with several billions of elements, thus enabling the Direct Numerical Simulation (DNS) of laboratory and semi-industrial configurations.

In DNS, all turbulence scales are resolved, hence such simulations are accurate but quite time- and memory-consuming. YALES2 is also able to perform LES, in which only the largest turbulent scales are explicitely resolved the effect of smallest ones are modeled. This allows to predict the most important unsteadiness of the flow (conversely to statistical approaches) with reduced computational cost in comparison with DNS [71]. Turbulence is not considered in this thesis, and then only DNS will be performed.

The Mach number is defined as the ratio of the local fluid velocity by the sound speed in the same fluid. When the Mach number is very small, compressibility of the fluid can be neglected, and Navier-Stokes equations can be solved under incompressible assumption. Low-Mach number flows are encountered for instance in combustion chambers [53]. In boiling simulations, the Mach number is negligible, thus the low-Mach number approach is relevant.

More general information on YALES2 can also be found in Moureau et al. [8].

3.1.1 The Finite Volume Method

In YALES2, the equations of fluid dynamics are solved on grid nodes. In order to ease the resolution of equations, physical quantities are stored on grid nodes. As a result, the derivatives of physical quantities are computed using only physical values stored on grid nodes. YALES2 uses the Finite Volume Method (FVM) for the computation of the differential operators. The principle of the FVM is the partitioning of the computational domain into small volumes over which all differential operators are reconstructed by integration of the fluxes across the volume faces.

3.1.1.1 Mathematical background

The FVM provides discretizations of differential operators based on the following theorems.

Theorem 1 (Divergence theorem, Gauss's theorem, Ostrogradsky's theorem)**.** *Let $\mathcal{V} \subset \mathbb{R}^3$ be a volume delimited by the closed surface $S \subset \mathbb{R}^3$, and let $\mathbf{F} : \mathcal{V} \longrightarrow \mathbb{R}^3$ be a continuously differentiable vector function on $\mathcal{V}$. One has*

$$\iiint_{\mathcal{V}} \nabla \cdot \mathbf{F}\,(v)\,\mathrm{d}^3 v = \oiint_{S} \mathbf{F}\,(\sigma) \cdot \mathbf{n}\,(\sigma)\ \mathrm{d}^2\sigma, \tag{3.1}$$

where the point $v \in \mathcal{V}$ represents successively all points in the volume $\mathcal{V}$, $\mathrm{d}^3 v \subset \mathcal{V}$ is the three-dimensional volume element around point v, $\sigma \in S$ represents successively all points on the surface S, $\mathrm{d}^2\sigma \subset S$ is the two-dimensional surface element around point σ, and $\mathbf{n}\,(\sigma)$ is the outward-pointing normal vector to S at point σ. In two dimensions, i.e. $\mathcal{V}, S \subset \mathbb{R}^2$, $\mathbf{F} \in \mathcal{C}^1\left(\mathcal{V}; \mathbb{R}^2\right)$, Eq. 3.1 simplifies to

$$\iint_{\mathcal{V}} \nabla \cdot \mathbf{F}\,(v)\,\mathrm{d}^2 v = \oint_{S} \mathbf{F}\,(\sigma) \cdot \mathbf{n}\,(\sigma)\ \mathrm{d}^1\sigma. \tag{3.2}$$

Proof. The proof can be found in the book of Spiegel et al. [81]. □

Theorem 2 (Gradient theorem, fundamental theorem for line integrals)**.** *Let $\mathcal{V} \subset \mathbb{R}^3$ be a volume delimited by the closed surface $S \subset \mathbb{R}^3$, and let $F : \mathcal{V} \longrightarrow \mathbb{R}^3$ be a continuously differentiable scalar function on $\mathcal{V}$. One has*

$$\iiint_{\mathcal{V}} \nabla F\,(v)\ \mathrm{d}^3 v = \oiint_{S} F\,(\sigma)\,\mathbf{n}\,(\sigma)\ \mathrm{d}^2\sigma, \tag{3.3}$$

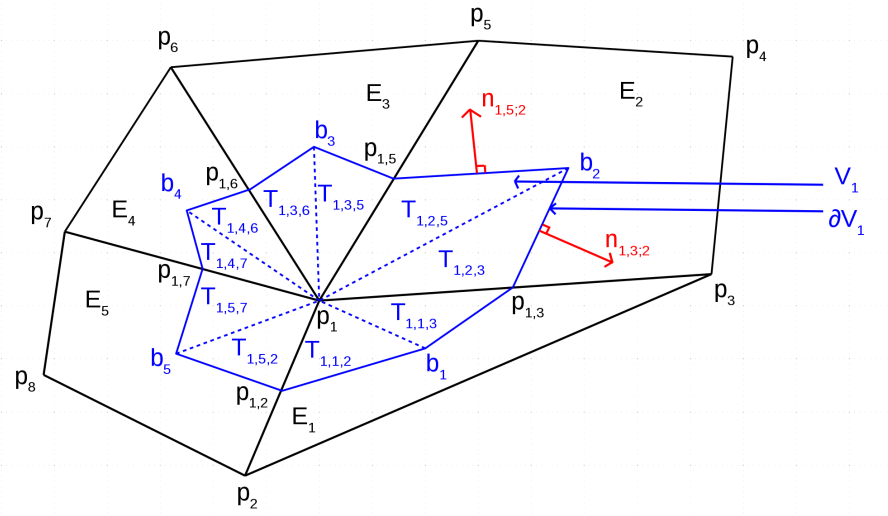


Fig. 3.1 – Example of a two-dimensional control volume used in the Finite Volume Method. The control volume associated to node $\mathbf{p}_1$ is defined by the neighbor grid element barycenters and edge midpoints.

where the point $v \in \mathcal{V}$ represents successively all points in the volume $\mathcal{V}$, $\mathrm{d}^3 v \subset \mathcal{V}$ is the three-dimensional volume element around point v, $\sigma \in S$ represents successively all points on the surface S, $\mathrm{d}^2\sigma \subset S$ is the two-dimensional surface element around point σ, and $\mathbf{n}(\sigma)$ is the outward-pointing normal vector to S at point σ. In two dimensions, i.e. $\mathcal{V}, S \subset \mathbb{R}^2$, $F \in \mathcal{C}^1\left(\mathcal{V}; \mathbb{R}^2\right)$, Eq. 3.3 simplifies to

$$\iint_{\mathcal{V}} \nabla F(v)\, \mathrm{d}^2 v = \oint_S F(\sigma)\, \mathbf{n}(\sigma)\, \mathrm{d}^1\sigma. \tag{3.4}$$

Proof. The proof can also be found in the book of Spiegel et al. [81]. □

3.1.1.2 Construction of control volumes

In order to apply Eqs. (3.1)-(3.4) in numerical simulations, one needs to define the volume $\mathcal{V}$. Since simulations are performed on discrete sets of points (nodes), the volume $\mathcal{V}$ will be restricted to a polygon in two dimensions or a polyhedron in three dimensions. In the terminology of the FVM, this restricted volume is called a *control volume*. In one dimension, control volumes are simply delimited by the midpoints of the grid elements.

Control volumes in two dimensions

Figure 3.1 shows an example of such a two-dimensional control volume used in YALES2. The control volume $\mathcal{V}_1$ associated to node $\mathbf{p}_1$ is built using the following procedure :

1. Identification of the grid elements E_i to which node $\mathbf{p}_1$ is a summit.

2. Computation of the barycenters $\mathbf{b}_i$ of elements E_i.

3. Among the edges of elements E_i, identification of the edges $(\mathbf{p}_1, \mathbf{p}_j)$ linking node $\mathbf{p}_1$ to another node $\mathbf{p}_j$.

4. Computation of the midpoints $\mathbf{p}_{1,j}$ of edges $(\mathbf{p}_1, \mathbf{p}_j)$.

5. For all elements E_i and all edges $(\mathbf{p}_1, \mathbf{p}_j)$ of E_i, computation of the area $\mathcal{A}_{1,i,j}$ of triangle $T_{1,i,j}$ formed by nodes $\mathbf{p}_1$, $\mathbf{p}_{1,j}$ and $\mathbf{b}_i$ denoted here respectively by points $A(x_A, y_A)$, $B(x_B, y_B)$ and $C(x_C, y_C)$ for clarity, as

$$\mathcal{A}_{1,i,j} = \frac{1}{2} \left| (x_B - x_A)(y_C - y_B) - (y_B - y_A)(x_C - x_B) \right|. \tag{3.5}$$

6. Computation of the control volume $\mathcal{V}_1$ associated to node $\mathbf{p}_1$ as the sum of all areas $\mathcal{A}_{1,i,j}$, i.e.

$$\mathcal{V}_1 = \sum_{i \in I^1} \sum_{j \in J_i^1} \mathcal{A}_{1,i,j}, \tag{3.6}$$

where $I^1 = \left\{ i_1^1, \ldots, i_{N_1}^1 \right\}$, with $N_1 \in \mathbb{N}^*$ the number of grid elements having $\mathbf{p}_1$ as summit, and $J_i^1 = \left\{ j_{i,1}^1, j_{i,2}^1 \right\}$ is the set of the two indices $j_{i,1}^1$ and $j_{i,2}^1$ such that grid edges $\left(\mathbf{p}_1, \mathbf{p}_{j_{i,1}^1} \right)$ and $\left(\mathbf{p}_1, \mathbf{p}_{j_{i,2}^1} \right)$ exist.

Applied to node $\mathbf{p}_1$ in Fig. 3.1, the above method yields

$$\begin{cases} I^1 = \{1, 2, 3, 4, 5\}, \\ J_1^1 = \{2, 3\}, J_2^1 = \{3, 5\}, J_3^1 = \{5, 6\}, J_4^1 = \{6, 7\}, J_5^1 = \{2, 7\}. \end{cases} \tag{3.7}$$

Since for all $i \in I^1$, the set J_i^1 is composed of two elements, each grid element E_i contributes with two triangles $T_{1,i,j}$ to the volume $\mathcal{V}_1$. To summarize, in two dimensions, control volumes are built by joining the grid element barycenters to the grid edge midpoints.

Control volumes in three dimensions

In three dimensions, control volumes are delimited by the grid element barycenters and the grid element face barycenters. In order to simplify the visualization, the construction of three-dimensional control volumes is illustrated on a cartesian grid. The same methodology is applied on unstructured tetrahedral grids.

Figure 3.2 shows a hexahedral grid element E delimited by nodes $\mathbf{p}_1, \ldots, \mathbf{p}_8$, as well as the contribution $\mathcal{E}_1$ of E to the control volume associated to node $\mathbf{p}_1$. In the present case, the subvolume $\mathcal{E}_1$ is also a hexahedron. Currently, the computation of $\mathcal{E}_1$ can not be performed as a single operation in YALES2. Indeed, one has to compute independently the volumes of the six tetrahedra constituting $\mathcal{E}_1$ and then compute $\mathcal{E}_1$ as their sum. Figure 3.3 shows the six tetrahedra constituting $\mathcal{E}_1$. Depending on the grid, other grid elements can contribute to the control volume of node $\mathbf{p}_1$. Figure 3.4 shows the whole control volume of node $\mathbf{p}_1$ as the disjoint union of all contributions of grid elements to which $\mathbf{p}_1$ is a summit.

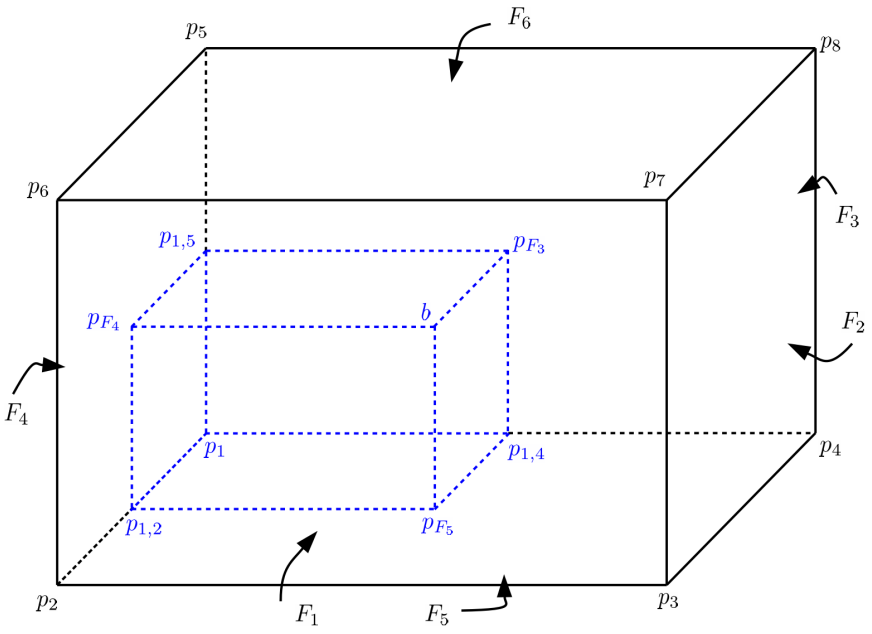


Fig. 3.2 – Contribution $\mathcal{E}_1$ of the hexahedral element E delimited by nodes $\mathbf{p}_1, \ldots, \mathbf{p}_8$ to the control volume associated to node $\mathbf{p}_1$.

3.1.1.3 Discretizations of differential operators on control volumes

We now give the FVM discretizations of the divergence, gradient and laplacian operators used in YALES2. Equations (3.1)-(3.4) are written in continuous form. Under this form, the integrands of both lhs and rhs of each equation are locally defined. For instance, in Eq. (3.2), $\nabla \cdot \mathbf{F}(v)$ depends on $v \in \mathcal{V}$, and $\mathbf{F}(\sigma) \cdot \mathbf{n}(\sigma)$ depends on $\sigma \in S$. These equations need be discretized on control volumes defined in Section 3.1.1.2. Discretizations are only given in two dimensions but their extensions to three dimensions are straightforward.

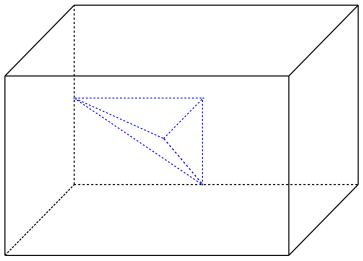
Refering to the control volume $\mathcal{V}_1$ shown in Fig. 3.1 and Eq. (3.2) of Thm. 1, the divergence discretization of flux $\mathbf{F}$ used in YALES2 is based on the following assumptions :

1. The divergence of flux $\mathbf{F}$ (integrand of the lhs) is uniform in the whole control volume $\mathcal{V}_1$.
2. The flux $\mathbf{F}$ at the control volume boundary $\partial\mathcal{V}_1$ is uniform on each piecewise-linear part of $\partial\mathcal{V}_1$.

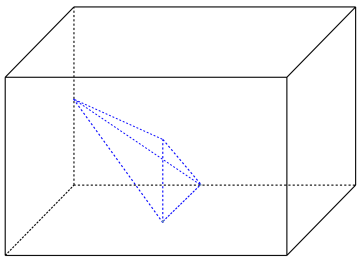
The second assumption needs clarification. One has

$$\partial\mathcal{V}_1 = \bigcup_{i \in I^1} \bigcup_{j \in J_i^1} \left\{ \sigma \in \mathbb{R}^2 : \sigma \in \left\{ \left(\mathbf{b}_i - \mathbf{p}_{1,j}\right) \eta + \mathbf{p}_{1,j} \right\}, \eta \in [0;1] \right\}, \tag{3.8}$$

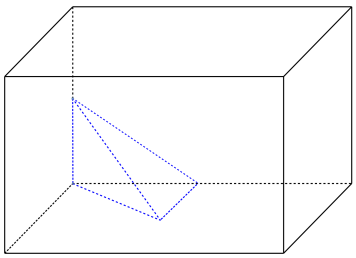
where the sets I^1 and J_i^1 are defined by Eq. (3.7) and $\eta \in [0,1]$ is a real parameter used to describe the segment between $\mathbf{b}_i$ and $\mathbf{p}_{1,j}$. Along each of the segments $\left\{ \left(\mathbf{b}_i - \mathbf{p}_{1,j}\right) \eta + \mathbf{p}_{1,j} \right\}$ where $\eta \in [0,1]$,



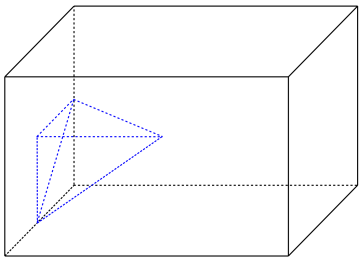
(a) Tetrahedron n°1.



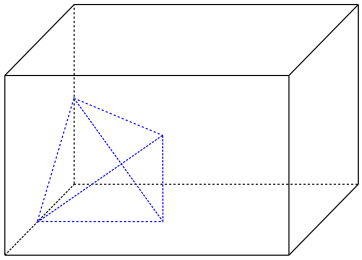
(b) Tetrahedron n°2.



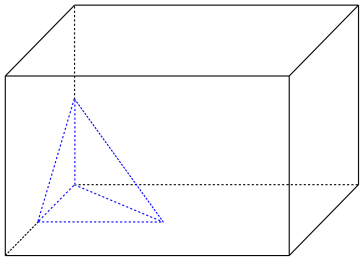
(c) Tetrahedron n°3.



(d) Tetrahedron n°4.



(e) Tetrahedron n°5.



(f) Tetrahedron n°6.

Fig. 3.3 – The six tetrahedra constituting the volume $\mathcal{E}_1$ shown in Fig. 3.2.

the flux $\mathbf{F}$ is then assumed to be uniform. Otherwise stated, $\partial\mathcal{V}_1$ is the union of all blue segments of Fig. 3.1, linking grid element barycenters $\mathbf{b}_i$ to grid edge midpoints $\mathbf{p}_{1,j}$, and $\mathbf{F}$ is assumed to be uniform on each of these segments.

Under these assumptions, a first version of the discretization of Eq. (3.2) on control volume $\mathcal{V}_1$ is given by

$$(\nabla \cdot \mathbf{F})|_{\mathcal{V}_1} \mathcal{V}_1 = \sum_{i \in I^1} \sum_{j \in J_i^1} \mathbf{F}(\sigma) \cdot (\|\mathbf{b}_i - \mathbf{p}_{1,j}\| \, \mathbf{n}_{1,j;i}(\sigma)), \tag{3.9}$$

where $\sigma = (\mathbf{b}_i - \mathbf{p}_{1,j})\,\eta + \mathbf{p}_{1,j}$ for some $\eta \in [0,1]$. To speed up computations, some terms of Eq. (3.9) are grouped and pre-computed. Figure 3.5 shows a zoomed portion of $\mathcal{V}_1$. As previously stated, the flux $\mathbf{F}$ is uniform on segment $S_{3;1,5} = \{(\mathbf{b}_3 - \mathbf{p}_{1,5})\,\eta + \mathbf{p}_{1,5}\}$ and on segment $S_{2;1,5} = \{(\mathbf{b}_2 - \mathbf{p}_{1,5})\,\eta + \mathbf{p}_{1,5}\}$, where $\eta \in [0,1]$. Moreover, $\mathbf{n}$ is also uniform on these two segments. Let $\mathbf{F}_{i;1,j}$ be the value of $\mathbf{F}$ on $S_{i;1,j}$, $\mathcal{L}_{i;1,j}$ be the length of $S_{i;1,j}$, $\mathbf{n}_{i;1,j}$ be the normal vector to $S_{i;1,j}$

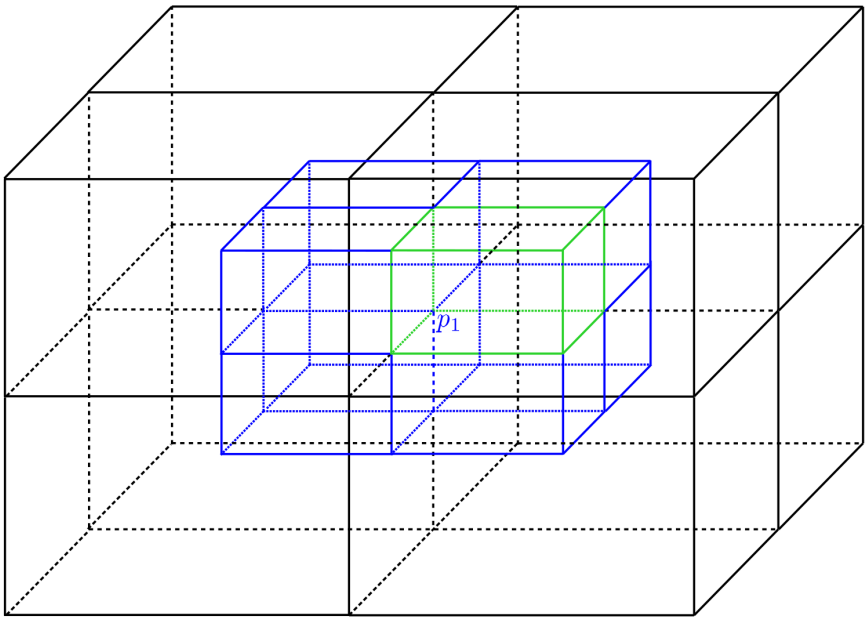


Fig. 3.4 – Total control volume of node $\mathbf{p}_1$. Each contribution is built according to the method illustrated in Figs. 3.2 and 3.3. The contribution shown in Fig. 3.2 is represented in green.

and η be a fixed real number between 0 and 1. Equation (3.9) is rewritten

$$(\nabla \cdot \mathbf{F})|_{\mathcal{V}_1} \mathcal{V}_1 = \sum_{i \in I^1} \sum_{j \in J_i^1} \mathbf{F}_{i;1,j} \cdot (\mathcal{L}_{i;1,j} \mathbf{n}_{i;1,j}). \tag{3.10}$$

The vector $\mathcal{L}_{i;1,j}\mathbf{n}_{i;1,j}$ is equal to the orthogonal vector to $\mathbf{b}_i - \mathbf{p}_{1,j}$ pointing to the outside of $\mathcal{V}_1$. In Fig. 3.5, one has

$$\mathcal{L}_{3;1,5}\mathbf{n}_{3;1,5} = \overline{\overline{R}}\left(-\frac{\pi}{2}\right)(\mathbf{b}_3 - \mathbf{p}_{1,5}), \tag{3.11}$$

and

$$\mathcal{L}_{2;1,5}\mathbf{n}_{2;1,5} = \overline{\overline{R}}\left(\frac{\pi}{2}\right)(\mathbf{b}_2 - \mathbf{p}_{1,5}), \tag{3.12}$$

where the rotation matrix $\overline{\overline{R}}$ of angle $\alpha \in [0, 2\pi[$ is given by

$$\overline{\overline{R}}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}. \tag{3.13}$$

The sum $\mathbf{A}_{1,5}$ of $\mathcal{L}_{3;1,5}\mathbf{n}_{3;1,5}$ and $\mathcal{L}_{2;1,5}\mathbf{n}_{2;1,5}$ is computed and stored at edge $(\mathbf{p}_1, \mathbf{p}_5)$. More generally, for all neighbor nodes $\mathbf{p}_j$ of $\mathbf{p}_1$, we define $\mathbf{A}_{1,j}$ as

$$\mathbf{A}_{1,j} = \mathcal{L}_{i_1;1,j}\mathbf{n}_{i_1;1,j} + \mathcal{L}_{i_2;1,j}\mathbf{n}_{i_2;1,j}, \tag{3.14}$$

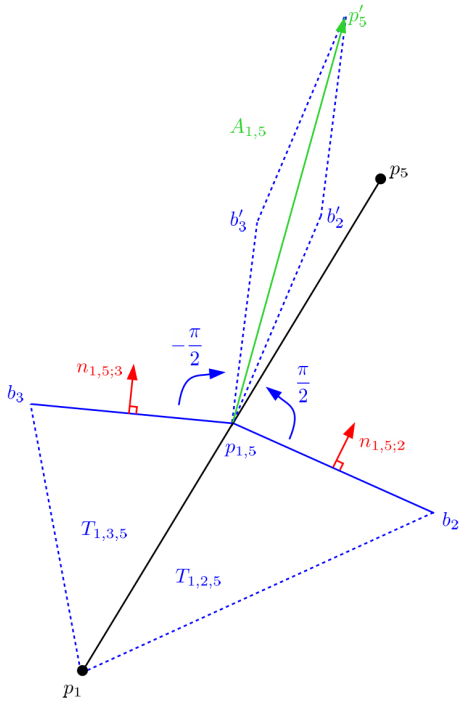


Fig. 3.5 – Definition and storage of vector $\mathbf{A}_{1,5}$ at edge $(\mathbf{p}_1, \mathbf{p}_5)$ as the sum of the oriented surfaces $\|\mathbf{b}_3 - \mathbf{p}_{1,5}\| \, \mathbf{n}_{1,5;3}$ and $\|\mathbf{b}_2 - \mathbf{p}_{1,5}\| \, \mathbf{n}_{1,5;2}$.

where i_1 and i_2 are the indices of the two grid elements to which $\mathbf{p}_1$ and $\mathbf{p}_j$ are two summits. Moreover, we make the assumption that $\mathbf{F}$ on $S_{3;1,5}$ is equal to $\mathbf{F}$ on $S_{2;1,5}$. This common value is given by

$$\mathbf{F}|_{S_{3;1,5}} = \mathbf{F}|_{S_{2;1,5}} = \frac{1}{2} \left(\mathbf{F}\left(\mathbf{p}_1\right) + \mathbf{F}\left(\mathbf{p}_5\right) \right). \tag{3.15}$$

These approximations are used on all grid edges $(\mathbf{p}_1, \mathbf{p}_j)$. Finally, Eq. (3.10) is rewritten

$$\left(\nabla \cdot \mathbf{F}\right)|_{\mathcal{V}_1} = \frac{1}{\mathcal{V}_1} \sum_{j=1}^{M_1} \frac{1}{2} \left(\mathbf{F}\left(\mathbf{p}_1\right) + \mathbf{F}\left(\mathbf{p}_j\right) \right) \cdot \mathbf{A}_{1,j}, \tag{3.16}$$

where M_1 is the number of nodes directly linked to node $\mathbf{p}_1$ ($M_1 = 7$ on Fig. 3.1). Equation (3.16) is the second-order discretization of the divergence operator in two dimensions used in YALES2.

Similarly, the discretized version of Eq. (3.4) reads

$$\left(\nabla F\right)|_{\mathcal{V}_1} = \frac{1}{\mathcal{V}_1} \sum_{j=1}^{M_1} \frac{1}{2} \left(F\left(\mathbf{p}_1\right) + F\left(\mathbf{p}_j\right) \right) \mathbf{A}_{1,j}. \tag{3.17}$$

Equation (3.17) is the second-order discretization of the gradient operator in two dimensions used in YALES2. Fourth-order versions of Eqs. (3.16) and (3.17) are also available.

The FVM makes the flux balance of the physical quantity considered on control volumes, exhbiting inherent conservation properties : a fluid particle leaving a given control volume enters one of its neighbor control volumes, the physical quantities related to this fluid particle are then conserved across control volume boundaries. Numerical simulations in fluid dynamics are based on mass and momentum conservation (Navier-Stokes equations) and energy conservation (heat equation). The FVM is then very well-suited for such simulations.

3.1.2 Temporal integration schemes

Several temporal integration schemes are available in YALES2. The fourth-order Runge-Kutta scheme (RK4) uses the following successive iterative computation to determine the updated variable field ξ^{n+1} :

$$\xi^1 = \xi^n - \frac{1}{4}\Delta t\mathcal{C}\left(\xi^n\right), \tag{3.18}$$

$$\xi^2 = \xi^n - \frac{1}{3}\Delta t\mathcal{C}\left(\xi^1\right), \tag{3.19}$$

$$\xi^3 = \xi^n - \frac{1}{2}\Delta t\mathcal{C}\left(\xi^2\right), \tag{3.20}$$

$$\xi^{n+1} = \xi^n - \Delta t\mathcal{C}\left(\xi^3\right), \tag{3.21}$$

where $\mathcal{C}$ is the discretized spatial operator. This scheme is explicit in the sense that it only uses the known solution at time t^n. In this thesis, otherwise stated, we use the RK4 scheme for temporal integration.

YALES2 also implements the Crank-Nicolson (CN) scheme given as

$$\xi^{n+1} = \xi^n + \Delta t\left[\frac{1}{2}\mathcal{C}\left(\xi^{n+1}\right) + \frac{1}{2}\mathcal{C}\left(\xi^n\right)\right]. \tag{3.22}$$

This scheme is semi-implicit in the sense that the field ξ is advanced using both the known solution at time t^n and also the unknown solution itself at time t^{n+1}. This scheme will be extended to a full implicit resolution of the heat equation in Section 6.2.1.

We also mention the TRK4 scheme developed in YALES2 by Kraushaar [45] and based on RK4. Informations on the TRK4 scheme and other temporal schemes available in YALES2 can also be found in the thesis of Sarkar [74].

3.1.3 A massively parallel code : High-Performance Computing

Numerical simulations in three-dimensional complex geometries, e.g. combustion chambers or rocket nozzles, imply large grids. Simulations on such grids are time- and memory-consuming. In order to save wall-clock time and to overcome memory limitations, YALES2 heavily relies on parallelism. A High-Performance Computing (HPC) approach has been adopted.

From the beginning of YALES2's development, particular attention has been paid to the parallelism of the code. The classical parallelism method consists in splitting the computational domain on subdomains between computer cores. This method is called the Simple Domain Decomposition (SDD) method and is illustrated in Fig. 3.6. In the SDD, each core has knowledge of only one subdomain and MPI communications are performed between subdomain boundaries to share informations when needed (typically to compute derivatives). This technique speeds up simulations

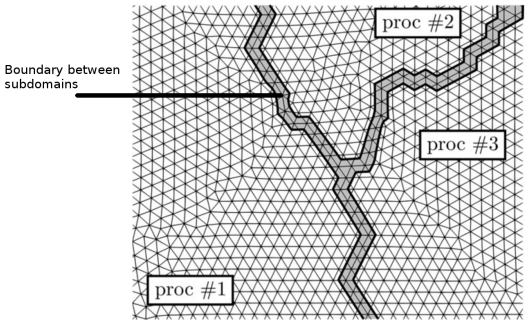
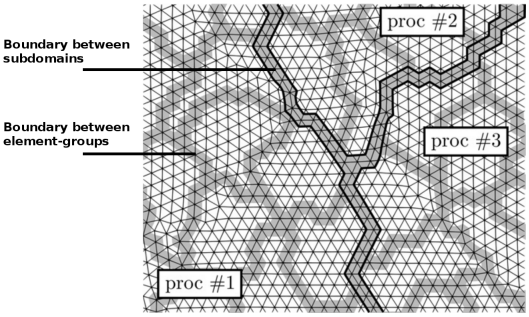
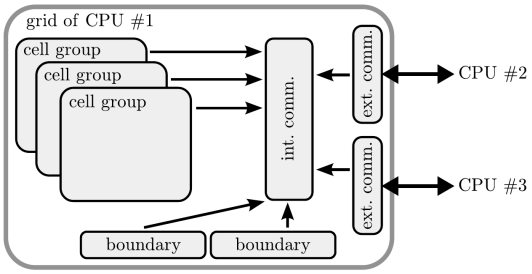


Fig. 3.6 – MPI parallelism using the Simple Domain Decomposition method. Each core has knowledge of only one subdomain.



(a) Each core has knowledge of only one subdomain. Each subdomain is divided into element groups to fill the core-specific L1, L2 or L3 caches (see Section 3.1.3.1).



(b) Communication scheme used in the DDD. In one subdomain (on one CPU), element groups (or cell groups) communicate by means of data structures called internal communicators. Two subdomains (on two different CPUs) communicate by means of data structures called external communicators via MPI (Message Passing Interface).

Fig. 3.7 – Illustration of the Double Domain Decomposition (DDD) used in YALES2 [52].

depending on the scalability of the chosen solver. To further improve the computation speed, a second level of domain decomposition has been adopted in YALES2. Each subdomain defined in the SDD is indeed divided in element groups of customizable number of elements. This method is called the Double Domain Decomposition (DDD) method and is illustrated in Fig. 3.7. The purpose of the DDD is the optimization of data transfer time at the hardware level.

3.1.3.1 The Double Domain Decomposition and the three levels of processor caches

Computer memory can be classified according to a hierarchy defined by data access time. Three main memory types can be distinguished in this hierarchy, from the fastest provided data access time to the slowest one :

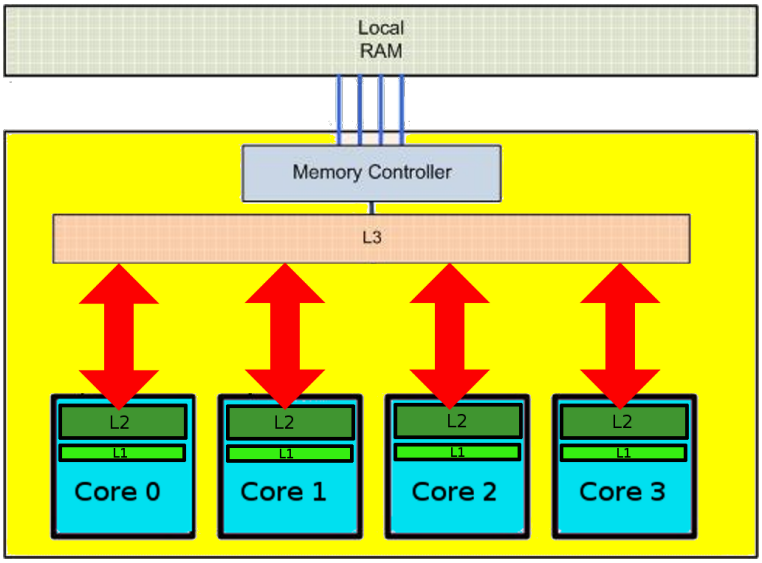


Fig. 3.8 – The three levels of L1, L2 and L3 caches on a quad-core processor.

1. Processor caches
   a) L1
   b) L2
   c) L3
2. Random Access Memory (RAM)
3. Hard disks

When a processor, or CPU, has to perform an operation on given data, it has first to access data. To this purpose, data stored on the Hard disk has to be copied on all memory units listed above until one of the L1, L2 or L3 caches. The two well-known memory types of this list are Hard disks and RAM. A Hard Disk has an important amount of memory but provides the processor with slow data access. To speed up access time, data are copied (*mounted*) to the RAM, where the amount of memory is much smaller but the data access much faster. To further speed operations, processors have their own caches. A processor has typically three caches named L1, L2 and L3. The L1 cache has the smallest amount of memory but provides the processor with a very fast data access. The L2 cache has more memory than the L1 cache but gives slower data access, and the L3 cache has more memory than the L2 cache but also gives slower data access. Figure 3.8 shows the hierarchy of memory units listed above.

In YALES2, data access time is slowered by means of the Double Domain Decomposition. Using this method, operations are performed on a per-element-group basis, where the size of element groups is user-defined. The data associated to each element group are expected to fit in the L1 (or L2) cache memory, thus a single cache filling is theoretically expected when data need to be copied. The element group size is typically set between 1 000 and 2 000, based on empirical tests.

3.2 The Spray solver

In this section we focus on the Spray solver developed for liquid jet atomization simulations but generally valid for two-phase flows. This solver is the base of our Boiling solver detailed in the following chapters. A good understanding of the Spray solver is then needed to understand our developments.

Atomization simulations require a Navier-Stokes solver and a robust and accurate method to simulate the two-phase interface movement. In the Spray solver, the interface is represented by means of the Conservative Level Set method introduced in Section 2.4.2.2. Once the CLS function has been advected and reinitialized, the incompressible Navier-Stokes equations are solved. The Spray solver uses the projection method detailed in Section 2.6 to solve Navier-Stokes equations with a time decoupling between velocity and pressure. The velocity is continuous across the interface. The only discontinuity at the interface that explicitely intervenes in the projection method is the pressure discontinuity due to surface tension.

One temporal iteration of the Spray solver is completed when both the CLS and Navier-Stokes equations are solved. The algorithm of the Spray solver is summarized in Alg. (1), and more details about the ACLS method and the projection method are given in the following sections.

Algorithm 1: One iteration of the Spray solver

```
Prescription of initial and boundary conditions for CLS and velocity or pressure;
while t < t_max do
    Compute SDF function from CLS function, interface normal and curvature;
    Advect CLS function;
    Reinitialize CLS function;
    Solve incompressible Navier-Stokes equations with the projection method;
```

3.3 Interface modeling and localization : the Sharp Interface model versus the Narrow Band Approach

The Continuum Surface Force model presented in Section 2.2 only approximates surface tension forces at the interface. Since simulations of phase change heavily rely on the ability of the numerical method to accurately locate the interface, we made the choice to compute surface tension forces directly at the interface, i.e. to use the SI model. The boiling phenomenon is defined by the transition of fluid particles, due to heat fluxes, from the liquid phase to the vapor phase. Theoretically, the transition of one fluid particle lasts a certain amount of time since, at microscopic scale, the liquid-vapor interface is diffuse and has a non-zero thickness, similar to the macroscopic diffuse interface depicted in Fig. 2.1. Nevertheless, since we want to simulate boiling at macroscopic scale using the SI model, phase change is characterized in our simulations by the abrupt passage of one fluid particle from one phase to the other, i.e. phase change happens instantaneously (at infinite speed).

In two-phase flow simulations, the interface location is a crucial information. Independently of the numerical method used to track or capture the interface, it is also useful for various numerical operations to identify the grid nodes close to the interface. In YALES2, the nodes around the interface are flagged with a signed integer number indicating their degree of remoteness from the interface. This method is called the Narrow Band Approach [1]. These signed integer numbers are called band levels. Sign apart, nodes of band level 1 are the closest nodes to the interface,

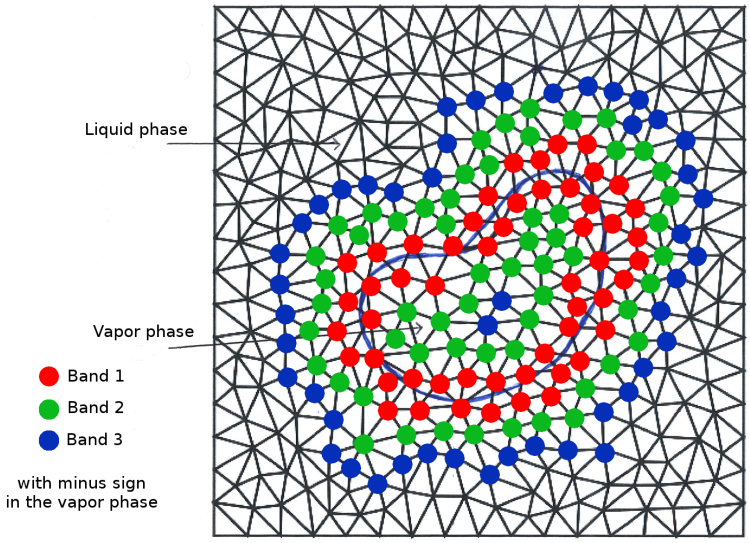


Fig. 3.9 – Narrow band defined around the two-phase interface (in blue) with 3 band levels in each phase : from 1 to 3 in the liquid phase and from -1 to -3 in the vapor phase. In YALES2, the default number of band levels is usually set to 8.

nodes of band level 2 are the direct neighbors of nodes of band level 1, and so on until a chosen maximum band level. In YALES2, the maximum band level is user-defined and typically set to 8. By convention, nodes located in the liquid phase are flagged with positive integer numbers, and nodes located in the vapor phase are flagged with negative integer numbers. Figure 3.9 shows an example of a narrow band definition in two dimensions. In the following parts, the expression "band levels" refers to the band levels defined in Fig. 3.9.

3.4 Accurate Conservative Level Set method

YALES2 uses the Accurate Conservative Level Set method from Desjardins et al. [19] introduced in Section 2.4.2.2. In YALES2, the Fast Marching Method is not implemented for the computation of the signed distance function because this method strongly relies on a cartesian mesh. Unstructured versions of the FMM exist [11], but are difficult to implement[1], even in a sequential algorithm. Instead, YALES2 uses its own distance computation method.

3.4.1 Computation of the Signed Distance Function

In [19], once the CLS is advected and reinitialized, the authors compute the Signed Distance Function with the Fast Marching Method. Instead, YALES2 uses a robust method suitable to unstructured grids. The Spray solver computes the Signed Distance Function the following way : once the CLS is advected and reinitialized, the exact position of the interface is computed on each grid edge by inversion of the CLS function on the closest nodes to the interface. From ψ, one can

[1] See following chapters.

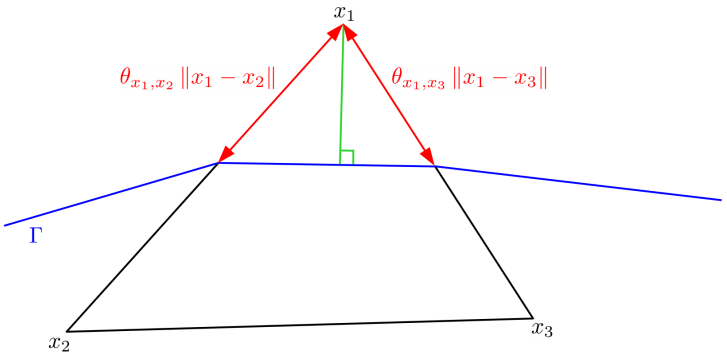


Fig. 3.10 – Computation on node $\mathbf{x}_1$ (band level 1) of the signed distance function ϕ to the interface Γ by Eq. (3.25) where the relative distance $\theta_{\mathbf{x}_1,\mathbf{x}_2}$ is given by Eq. (3.24). The exact value of $\phi(\mathbf{x}_1)$ is given by the length of the green segment.

recover the relative position of the interface $\theta \in [0;1]$ on one grid edge by inversion of the hyperbolic tangent function. Let $(\mathbf{x}_1, \mathbf{x}_2)$ be the grid edge between nodes $\mathbf{x}_1$ and $\mathbf{x}_2$. For $i = 1, 2$, YALES2 computes the signed distance values at $\mathbf{x}_i$ as

$$\phi(t, \mathbf{x}_i) = 2\epsilon(\mathbf{x}_i)\,\mathrm{atanh}\left(2\max\left(\min(\psi(t, \mathbf{x}_i), 1-\delta), \delta\right) - 1\right), \tag{3.23}$$

where the min and max functions are used to avoid under- and overshoots on the argument of the atanh function, and δ is a non-dimensional constant fixed to 10^{-15}. The interface relative distance θ is given by

$$\theta_{\mathbf{x}_1,\mathbf{x}_2}(t) = \frac{\phi(t, \mathbf{x}_1)}{\phi(t, \mathbf{x}_1) - \phi(t, \mathbf{x}_2)}, \tag{3.24}$$

where $\theta = 0$ when the interface is on node $\mathbf{x}_1$ and $\theta = 1$ when on node $\mathbf{x}_2$. From the knowledge of θ, the value of the signed distance function on node $\mathbf{x}_1$ is computed as

$$\phi(t, \mathbf{x}_1) = \min\left\{\theta_{\mathbf{x}_1,\mathbf{x}_2}\,\|\mathbf{x}_1 - \mathbf{x}_2\|, \theta_{\mathbf{x}_1,\mathbf{x}_3}\,\|\mathbf{x}_1 - \mathbf{x}_3\|\right\}, \tag{3.25}$$

as depicted in Fig. 3.10, meaning that the distance of node $\mathbf{x}_1$ is computed with an error $\mathcal{O}(1)$ coming from the simplification of the orthogonal distance (in green) with the distance of node $\mathbf{x}_1$ to the closest intersection of the interface with an edge containing node $\mathbf{x}_1$.

Once a signed distance value is computed on all the closest nodes to the interface (band level 1) by Eq. (3.25), signed distance values are computed for the nodes immediately close to them (band level 2). Figure 3.11 shows an example of signed distance computation for a node $\mathbf{x}_4$ of band level 2. For this node, the signed distance value is computed using Chasles's identity, i.e.

$$\phi(t, \mathbf{x}_4) = \|\mathbf{A} + \mathbf{B}\|, \tag{3.26}$$

where $\mathbf{A}$ is a vector from $\mathbf{x}_4$ to a node of band level 1 and $\mathbf{B}$ is a vector from the same node to the interface, such that $\mathbf{A} + \mathbf{B}$ has the smallest norm among all candidates for $\mathbf{A}$ and $\mathbf{B}$.

As seen in Fig. 3.10, this computation method is a bold approximation of the real signed distance value at node $\mathbf{x}_1$, equal to the length of the segment linking node $\mathbf{x}_1$ to its projection onto

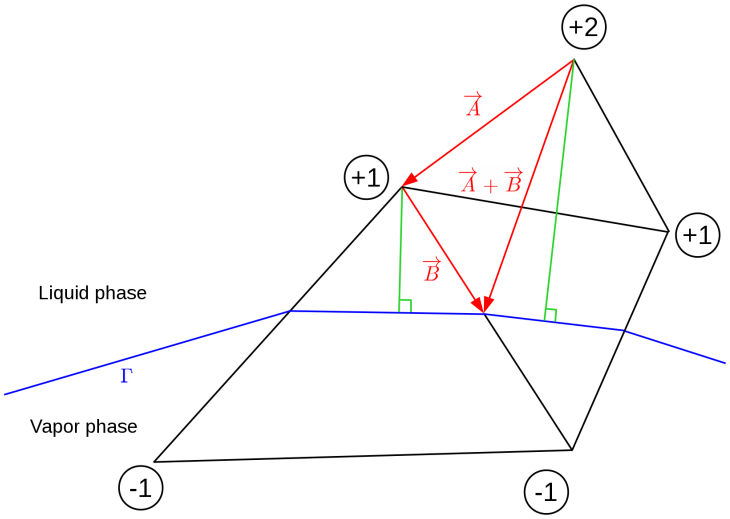


Fig. 3.11 – Computation on a node of band level 2 of the signed distance function ϕ to the interface Γ by Eq. (3.26) where the vector summation is done on the grid edges shown in red. The resulting ϕ value is equal to the norm of vector $\mathbf{A}+\mathbf{B}$. The exact value is given by the length of the green segment.

the piecewise-linear interface Γ, shown in green. Indeed, as will be largely detailed in the following chapters, the lack of precision in the computation of the signed distance function must absolutely be eliminated when dealing with phase change, and more generally, with any numerical method which heavily relies on the interface normal vector. The motivation behind this rough approximation is the robustness and simplicity of the method. It must be said that the error, non-negligible on band level ± 1, becomes increasingly small on outer band levels.

3.4.2 Computation of the interface normal vector

As typically done in the level set formalism, the Spray solver computes the interface normal vector $\mathbf{n}$ as

$$\mathbf{n} = \frac{\nabla \phi}{\|\nabla \phi\|}, \tag{3.27}$$

where ϕ is given by Eq. (3.25). The interface normal vector is used in two different steps of the numerical method of the Spray solver : in the computation of the interface curvature (see Section 3.4.3) and in the reinitialization of the CLS function (see Section 2.4.2.2, Eq. (2.24)).

3.4.3 Computation of the interface curvature

The Spray solver uses Eq. (3.6) from Goldman [28], based on Frenet equations and compatible with Eq. (2.13), to compute the interface curvature. This equation is

$$\kappa = \frac{\nabla \phi^{\mathrm{T}} \cdot H(\phi) \cdot \nabla \phi - \|\nabla \phi\|^2 \operatorname{Tr}\left(H\left(\phi\right)\right)}{\|\nabla \phi\|^3}, \tag{3.28}$$

where $H(\phi)$ is the hessian matrix of ϕ given by

$$H(\phi) = \begin{pmatrix} \phi_{xx} & \phi_{xy} & \phi_{xz} \\ \phi_{yx} & \phi_{yy} & \phi_{yz} \\ \phi_{zx} & \phi_{zy} & \phi_{zz} \end{pmatrix}, \tag{3.29}$$

and Tr is the trace operator given by

$$\mathrm{Tr}(H(\phi)) = \phi_{xx} + \phi_{yy} + \phi_{zz}. \tag{3.30}$$

Until the 2^{nd} *Extreme CFD Workshop* in January 2019, CERFACS, Toulouse, France, Eq. (3.28) was solved using compact schemes (i.e. schemes involving only the closest neghbors of a node) for the discretization of the gradient and hesssian operators. During the workshop, it has been found more accurate on some simulations to use a non-compact scheme to compute the hessian operator in Eq. (3.28). This is now the preferred method to compute the interface curvature in YALES2.

3.5 The Projection Method

The Spray solver uses the projection method detailed in Section 2.6 in which the pressure jump is given by Eq. (1.8). The discretization of the Poisson equation with the application of the discontinuities at the interface in the case of phase change with the Boiling solver will be detailed in Chapter 4 and Appendix B.

3.6 Resolution of the Poisson equation

Poisson equation (2.47) where $P^{n+\frac{1}{2}}$ is the unknown is written as a linear system of the form

$$\overline{\overline{A}}\mathbf{P}^{n+\frac{1}{2}} = \mathbf{B}, \tag{3.31}$$

where $\overline{\overline{A}}$ is a $N \times N$ matrix, N being the total number of grid nodes, only composed of geometrical grid properties (and information on the per-phase uniform density $\rho^{n+\frac{1}{2}}$), $\mathbf{P}^{n+\frac{1}{2}}$ is the vector of unknown pressures $P_i^{n+\frac{1}{2}}$ and $\mathbf{B}$ is the vector of predictor velocity divergences $\frac{1}{\Delta t}\nabla \cdot \mathbf{u}_i^*$, for $i = 1, \ldots, N$, and terms related to the pressure jump at the interface (Eq. (1.8)). The linear system (3.31) is solved using a linear solver. Linear solvers use different iterative methods depending on the properties of $\overline{\overline{A}}$. Since YALES2 is designed to handle grids with billions of elements, the system (3.31) is never solved by direct inversion of the whole matrix $\overline{\overline{A}}$ which would be unfeasible. Moreover, MPI parallelism implies that $\overline{\overline{A}}$ and $\mathbf{B}$ are never entirely known on one given computer core. One core has only knowledge of coefficients of $\overline{\overline{A}}$ and $\mathbf{B}$ concerning nodes located in their assigned subdomain, and MPI communications are performed between core boundaries at each iteration of the linear solver to share information when needed. These communications can occupy up to 80% of the overall computation time [50]. The resolution optimization of Poisson equation (2.47) is of major importance in parallel incompressible flow simulations.

Abundant literature can be found on linear solvers. When matrix $\overline{\overline{A}}$ is symmetric positive definite, the Conjugate Gradient (CG) method can de used [34]. The CG method is iterative and can be time-consuming. A *good* initialization of the method, called preconditioning, significatively lowers the number of iterations. This method is called the Preconditionned Conjugate Gradient

(PCG) [92]. To further speed up computations, a *deflation* operation can be performed. A first set of iterations is performed where the unknown $\mathbf{P}^{n+\frac{1}{2}}$ is assumed to be uniform on predefined grid regions. The resulting field is used as initialization in the PCG solver. The method is called the Deflated PCG (DPCG) [55], [5]. In YALES2, these three solvers are available. In our simulations, Eq. (3.31) is solved using the DPCG solver. The grid regions used in the DPCG solver are the element groups defined by the DDD method detailed in Section 3.1.3 and illustrated in Fig. 3.7(a). Details about the DPCG solver implementation in YALES2 can be found in [49, 50]. Linear systems involving non-symmetric matrices are solved with the BiConjugate Gradient (BiCG) method using the BiCGStab and BiCGStab2 methods [92].

As stated in Section 2.6, once the updated pressure is known, Eq. (2.46) is used to correct the velocity predictor in order to compute the updated velocity.

Chapter 4

Numerical simulation of two-phase flows with phase change in one dimension

A first version of our numerical method is presented in this chapter and evaluated against a one-dimensional stationary test case. Only mandatory ingredients of the method for one-dimensional simulations are presented and used in this chapter.

Phase change naturally arises in real flows, either because of pressure variations, or because of temperature variations. In YALES2, we developed the Boiling solver to enhance the numerical method presented in Chapter 3 in order to simulate phase change due to heat transfer.

Outline

4.1 Introduction

Boiling is a complex physical phenomenon (see Chapter 1 for a description of the physics of boiling). As detailed in Chapter 3, two-phase flows imply the existence of an interface separating the two phases. In two dimensions, the interface is a curve ; in three dimensions, it is a surface, and in one dimension, it is a point. Phase change occurs precisely at the interface. We recall here that the interface whose position changes w.r.t. time, is not necessarily lying on grid nodes. Indeed, the interface is in general located between grid nodes and is thus defined through a set of points at subgrid level. The interface position is then captured using the Conservative Level Set function ψ given by Eq. (2.16). Since the interface normal vector and curvature are needed in the numerical method, a complex machinery is used to compute them from the knowledge of ψ. As will be detailed

in Chapter 5, the methods available in the version of YALES2 of 2015 introduce large deviations in the interface location. This chapter introduces the numerical method used to simulate two-phase flows with phase change. Since our goal is to focus on the numerical ingredients used to take phase change into account in Navier-Stokes equations, the problem of deviations in the interface location is avoided by restricting the presentation and validation of the method to the one-dimensional case. The strategy will be extended to 2D and 3D in Chapters 5 and 6.

The contribution of this thesis is the development of the Boiling solver in the YALES2 library. The Boiling solver has been validated against test cases in one, two and three dimensions, on structured and unstructured grids, in sequential and parallel algorithms. This chapter introduces the main ingredients of the Boiling solver with the restriction to the one-dimensional case. Chapters 5 and 6 detail and validate the numerical method implemented in the Boiling solver in 2D and 3D, with a fixed mass transfer rate (Ch. 5) and a computed mass transfer rate (Ch. 6). The notion of mass transfer rate is explained in this chapter.

We used the Spray solver detailed in Chapter 3 as a base to develop the Boiling solver. The choice of the Spray solver comes from the fact that is was the most advanced two-phase flow solver in YALES2 when we started this work. As a consequence, the Boiling solver implements the same families of methodologies (Level Set function to capture the interface movement and Projection Method to solve Navier-Stokes equations with Ghost Fluid Method to handle discontinuities at the interface). The Boiling solver extends these methodologies and implements new ones to take phase change into account. These improvements are presented in the following sections in which space is one-dimensional.

4.2 Governing equations

4.2.1 Mass conservation

In one dimension, Eq. (1.3) reads

$$\frac{\partial u_i}{\partial x} = 0, \tag{4.1}$$

where the subscript i denotes the phase. The velocity u_i is thus only dependent on time, i.e.

$$u_i = u_i(t). \tag{4.2}$$

Note that different unconnected regions of the same phase i can present different values of u_i (see Section 4.4).

4.2.2 Incompressible Navier-Stokes equation with phase change

Equation (1.21) states that the volume expansion at the interface due to boiling leads to a non zero velocity divergence at the interface, i.e. the incompressible hypothesis, valid for both the vapor and liquid phases, does not hold at the interface. Considering a vaporizing spherical bubble, since volume expansion occurs at all points on the interface, the surrounding liquid is pushed away from the interface. In YALES2 we use the infinitely thin interface model together with an incompressible formalism for Navier-Stokes equations in both phases. Since the infinitely thin interface is a subgrid set of null measure, it would be quite challenging to use a single velocity field and enforce Eq. (1.21). On the contrary, we use the Ghost Fluid Method to enforce the velocity discontinuity $u_{\text{liq}} - u_{\text{vap}}$ at the interface, i.e. to mimic Eq. (1.21), by means of two different velocity fields [84, 85].

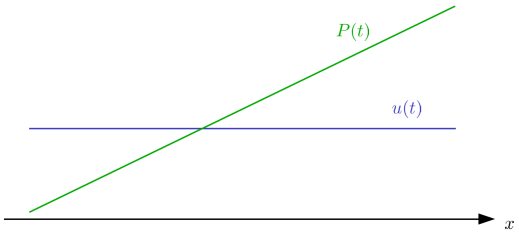


Fig. 4.1 – Velocity and pressure profiles for one phase in one dimension as given by the solution of the incompressible Navier-Stokes equation (4.4).

This methodology is relevant as long as the computational domain does not prevent the liquid to be pushed away from the interface by boiling. In this thesis, we consider boiling in an open system, so the liquid pushed by the vapor volume increase at the interface due to boiling can exit the domain. In a confined area, liquid-to-vapor phase change would lead, due to mass conservation, to the compression of the fluid everywhere in the domain.

The one-dimensional incompressible Navier-Stokes equation is given in each phase i by

$$\frac{\partial u_i}{\partial t} + u_i \frac{\partial u_i}{\partial x} = -\frac{1}{\rho_i}\frac{\partial P}{\partial x} + 2\nu_i \frac{\partial^2 u_i}{\partial x^2}, \tag{4.3}$$

where u_i is the velocity, ρ_i the density and ν_i the kinematic viscosity of the given phase, P is the pressure field (the gravitational forces are neglected). Equation (4.3) simplifies by Eq. (4.1) to

$$\frac{\partial u_i}{\partial t} = -\frac{1}{\rho_i}\frac{\partial P}{\partial x}, \tag{4.4}$$

simply relating the fluid acceleration to its pressure gradient at the same point. Equations (4.2) and (4.4) establish that $\partial P/\partial x$ is only dependent on time. Figure 4.1 shows a sketch of a velocity profile for one phase and a compatible pressure profile.

4.2.2.1 Velocity and pressure jumps at the interface

As a consequence of vapor volume expansion and difference in density between the liquid and vapor phases, the velocity is discontinuous across the interface. This discontinuity is given by Eq. (1.30) recalled here,

$$[\mathbf{u}]_\Gamma = \dot{m}\left[\frac{1}{\rho}\right]_\Gamma \mathbf{n}_\Gamma, \tag{4.5}$$

where $\mathbf{u} = \pm \|\mathbf{u}\| \mathbf{e}_x$ and $\mathbf{n}_\Gamma = \pm \mathbf{e}_x$.

Since in one dimension, the interface is reduced to a disjoint set of points, the interface curvature is not defined, and the pressure jump given by Eq. (1.31) simplifies to

$$[P]_\Gamma = -\dot{m}^2 \left[\frac{1}{\rho}\right]_\Gamma. \tag{4.6}$$

The velocity and pressure jumps are computed from the mass transfer rate and imposed at the interface location when solving Eq. (4.4).

4.2.3 Advection and reinitialization equations of the Accurate Conservative Level Set function with phase change

The Accurate Conservative Level Set Method from Sections 2.4.2.2 and 3.4 is used to capture the interface. In each phase, the advection equation of the Conservative Level Set function ψ is given by

$$\frac{\partial \psi}{\partial t} + \frac{\partial}{\partial x}\left(\psi u_i\right) = 0, \tag{4.7}$$

where i stands for the $_{\text{liq}}$ or $_{\text{vap}}$ subscript, depending on the phase containing the node where the equation is solved, which simplifies by Eq. (4.1) to

$$\frac{\partial \psi}{\partial t} + u_i \frac{\partial \psi}{\partial x} = 0. \tag{4.8}$$

Without phase change, the interface is only advected by the fluid velocity. The density difference between phases implies that the liquid-turned-vapor fluid particles occupy a larger space. As a result, boiling contributes to the interface movement, even for an otherwise quiescent fluid. The total interface velocity in presence of phase change is given by Eqs. (1.32) and (1.33). We adopt the following convention : the interface motion is computed w.r.t. the velocity of the enclosed phase, i.e. a vapor bubble surrounded by liquid has an interface computed w.r.t. the vapor velocity, a liquid droplet surrounded by vapor has an interface computed w.r.t. the liquid velocity. Since we focus this work on boiling, we are then more interested in vapor bubbles surrounding by liquid. As a result, $u_{\text{vap}} - \dot{m}/\rho_{\text{vap}}$ is substituted to u_i in Eq. (4.8) which is rewritten

$$\frac{\partial \psi}{\partial t} + u_{\text{vap}} \frac{\partial \psi}{\partial x} = \frac{\dot{m}}{\rho_{\text{vap}}} \frac{\partial \psi}{\partial x}, \tag{4.9}$$

where the phase change contribution is considered as a source term. The philosophy underlying the choice of this equation form is to consider phase change as an external ingredient in the dynamics of a two-phase flow. If there is no phase change, the rhs of Eq. (4.9) vanishes and this equation defaults to Eq. (4.8). Conversely, in the case of bubbles at rest in a hotter liquid, the advection term in the lhs of Eq. (4.9) vanishes, and the interface motion is only driven by the phase change contribution in the rhs of the same equation. These two observations are also valid in the case of droplets at rest where the liquid velocity and density are substituted in Eq. (4.9). Equation (4.9) is the advection equation of the Accurate Conservative Level Set function solved in the Boiling solver for one-dimensional simulations.

Advection deforms the CLS function, then it is reinitialized using Eq. (2.27).

4.2.4 Temperature equation with immersed Dirichlet boundary condition at the interface

In boiling, the mass transfer rate depends on the heat exchanges across the interface. We recall that the mass transfer rate is defined as the difference between the heat fluxes across the interface divided by the latent heat, as given by Eq. (1.26). The heat equation has thus to be solved in the Boiling solver. We assume that the interface temperature T_Γ is equal to the fluid saturation temperature T_{sat}. The classical explicit advection-diffusion equation at the interface given by

$$\begin{cases} \dfrac{\partial T_i}{\partial t} + u_i \dfrac{\partial T_i}{\partial x} = \dfrac{1}{\rho_i c_{\text{p},i}} \dfrac{\partial}{\partial x}\left(\lambda_i \dfrac{\partial T_i}{\partial x}\right) & (4.10) \\ T_\Gamma = T_{\text{sat}} & (4.11) \end{cases}$$

is solved on both phases, where u_i is the fluid velocity, ρ_i the density, $c_{\mathrm{p},i}$ the heat capacity at constant pressure, λ_i the thermal conductivity of the phase denoted by the subscript $_i$. The difficulty is that condition (4.11) is **not** applied on given nodes, rather on an interface whose position is in general not coincident with grid nodes. We use the terminology *immersed* to state that the interface is a subgrid point set, and Eq. (4.11) is said to be an *immersed boundary condition* [65].

4.2.5 Mass transfer rate equation

Once the liquid and vapor temperature fields have been advanced in time with immersed Dirichlet boundary condition (4.11), one can compute the mass transfer rate. In one dimension, the definition (1.26) of the mass transfer rate $\dot{m}$ becomes

$$\dot{m} = \frac{1}{L_v}\left(-\lambda_{\mathrm{liq}}\left.\frac{\partial T_{\mathrm{liq}}}{\partial x}\right|_\Gamma \mathbf{e}_x + \lambda_{\mathrm{vap}}\left.\frac{\partial T_{\mathrm{vap}}}{\partial x}\right|_\Gamma \mathbf{e}_x\right)\cdot \mathbf{n}_\Gamma, \tag{4.12}$$

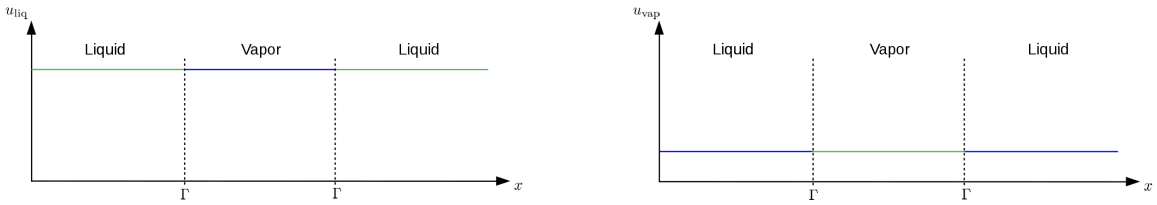
where L_v is the latent heat of the fluid and $\mathbf{n}_\Gamma$ is the interface normal vector, that is trivially computed as $\pm\mathbf{e}_x$ in one dimension.

4.3 Solving the governing equations

We now describe the methodologies used to solve the governing equations presented above in the Boiling solver for the one-dimensional case. These numerical methods will be extended to higher dimensions in Chapters 5 and 6.

4.3.1 Solving the incompressible Navier-Stokes equation with phase change: two-velocity formulation in the projection method

The Boiling solver solves the one-dimensional Navier-Stokes equation (4.4) using the projection method presented in Section 2.6 with some adaptations. The main difference between the projection method used in the Spray solver and the one used in the Boiling solver is the non-zero velocity jump at the interface (Eq. (4.5)) computed in the Boiling solver. The velocity jump cannot be handled as easily as adding a new term in the method used in the Spray solver. Indeed, as for all discontinuities accounted in this methodology, the velocity jump is defined precisely at the subgrid interface location, as denoted by the $_\Gamma$ subscript. As a result, when one has to apply differential operators to the velocity field on nodes close to the interface, one has also to apply the corresponding velocity jump in the operator discretization. This requirement leads to the famous problem of *differentiating across an interface with discontinuities*. It is well-known that classical differential operators act on continuous functions, thus are not designed to differentiate across discontinuities. In terms of two-phase flow dynamics, classical differential operators are not designed to differentiate across the two-phase interface. In order to overcome this difficulty, the Boiling solver uses the Ghost Fluid Method detailed in Chapter 2 : both velocity fields u_{liq} and u_{vap} are defined in the whole domain. Physical values of the liquid (vapor) velocity are contained in the liquid (vapor) part of u_{liq} (u_{vap}), and the vapor (liquid) part of u_{liq} (u_{vap}) can be used to store any working values, with potentially no physical meaning. Figure 4.2 illustrates the definitions of the velocity fields used. These working values are called ghost values and, in the present case, are used to compute the velocity jump at the interface. In the projection method, prior to the computation of the velocity predictor u^*, a constant extrapolation of the liquid velocity is performed in the



(a) Liquid velocity field.

(b) Vapor velocity field.

Fig. 4.2 – Liquid (a) and vapor (b) velocity fields defined in the whole computational domain. For each field, the green part represents the physical field and the blue part is used to store working values for use with the Ghost Fluid Method, as shown in Fig. 4.3.

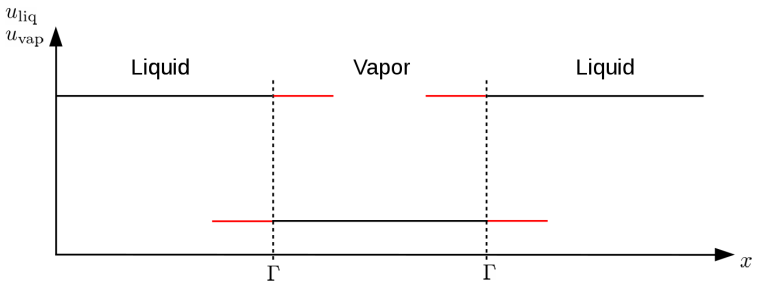


Fig. 4.3 – Constant extrapolations (in red) of the liquid velocity $\mathbf{u}_{\text{liq}}$ in the vapor phase, and of the vapor velocity $\mathbf{u}_{\text{vap}}$ in the liquid phase.

direction of the interface normal vector, from the liquid nodes to the vapor nodes, until the vapor boundary of the narrow band. A similar extrapolation is performed on the vapor velocity u_{vap}, from the vapor nodes to the liquid nodes. Figure 4.3 represents the two velocity fields and their constant extrapolations in the opposite phases. We emphasize that these extrapolations are here particularly easy to perform since in one dimension, both liquid and vapor velocities are uniform due to Eq. (4.1). The extrapolation consists then simply in copying a uniform value on the nodes located on the opposite side of the interface. As will be shown in Chapters 5 and 6, this is not the case in multidimensions. Thanks to the use of two distinct velocity fields and their constant extrapolations across the interface, each velocity field can now be differentiated across the interface. The velocity predictors u^*_{liq} and u^*_{vap} are independently computed by Eq. (2.45), which, in one dimension, simply states that

$$u^*_{\text{liq}} = u^n_{\text{liq}} \tag{4.13}$$

and

$$u^*_{\text{vap}} = u^n_{\text{vap}}, \tag{4.14}$$

since thr rhs of Eq. (2.45) is zero. As opposed to the two velocity fields, our method does not require the definition of two pressure fields. Indeed, only one pressure field is defined in the whole domain and only one Poisson equation (2.47) is solved to compute the new pressure $P^{n+\frac{1}{2}}$ in the whole domain. The velocity and pressure jumps are imposed in the matrix coefficients and in the rhs of the linear system yielded by the Poisson equation in order to compute the divergences

of the predictor velocities u^*_{liq} and u^*_{vap}. Due to Eqs. (4.13) and (4.14) and the incompressible hypothesis, these divergences are null in one dimension. On each grid node crossed by the interface, two different velocity jumps are computed : the velocity predictor jump $[u^*]_\Gamma$ and the updated velocity jump $\left[u^{n+1}\right]_\Gamma$. On one grid edge $(\mathbf{x}_1, \mathbf{x}_2)$ crossed by the interface, the velocity predictor jump is computed by

$$[u^*]_\Gamma = \frac{1}{2}\left(u^*_{\text{liq},1} + u^*_{\text{liq},2}\right) - \frac{1}{2}\left(u^*_{\text{vap},1} + u^*_{\text{vap},2}\right) \tag{4.15}$$

and the updated velocity jump is computed as

$$\left[u^{n+1}\right]_\Gamma = \frac{1}{2}\left(\dot{m}_1^{n+1} + \dot{m}_2^{n+1}\right)\left[\frac{1}{\rho}\right]_\Gamma \mathbf{n}_\Gamma, \tag{4.16}$$

where $\mathbf{n}_\Gamma = \pm\mathbf{e}_x$. The pressure jump is computed as

$$\left[P^{n+\frac{1}{2}}\right]_\Gamma = -\left(\frac{\dot{m}_1^{n+1} + \dot{m}_2^{n+1}}{2}\right)^2 \left[\frac{1}{\rho}\right]_\Gamma. \tag{4.17}$$

The jumps $[u^*]_\Gamma$, $\left[u^{n+1}\right]_\Gamma$ and $\left[P^{n+\frac{1}{2}}\right]_\Gamma$ are imposed in the discretization of the Poisson equation (2.47) as detailed in Appendix B. Once the updated pressure $P^{n+\frac{1}{2}}$ is known, we use Eq. (2.46) to correct the velocity predictors u^*_{liq} and u^*_{vap} in order to obtain the updated velocities u^{n+1}_{liq} and u^{n+1}_{vap}. Using Eq. (2.46) requires the computation of $\nabla P^{n+\frac{1}{2}}$. Close to the interface, valid pressure values are neeeded by the other phase to compute the gradient, as in the case of the velocity for the velocity predictors computations. The computation of such valid pressure values is detailed in Appendix B. The following enumeration summarizes the steps of the specific one-dimensional variant of the projection method implemented in the Boiling solver.

1. **Computation of velocity predictors** Using constant extrapolations on u^n_{liq} and u^n_{vap} across the interface to compute ghost values for nodes close to the interface, computation of u^*_{liq} and u^*_{vap} by
$$u^*_i = u^n_i \tag{4.18}$$
where i denotes the phase (liquid or vapor).

2. **Solving the Poisson equation with discontinuities at the interface** Building of the matrix coefficients of the linear system corresponding to the Poisson equation
$$\frac{\partial}{\partial x}\left(\frac{1}{\rho_i^{n+\frac{1}{2}}}\frac{\partial}{\partial x}P^{n+\frac{1}{2}}\right) = 0 \tag{4.19}$$
imposing the velocity and pressure jumps according to Appendix B.

3. **Computation of the updated pressure gradient** Computation of $\partial P^{n+\frac{1}{2}}/\partial x$ with the Ghost Fluid Method : extrapolation of pressure local ghost values across the interface, based on physical pressure values and velocity and pressure jumps at the interface, according to Appendix B.

4. **Correction of the velocity predictors** Correction of u^*_{liq} and u^*_{vap} to obtain the updated velocities u^{n+1}_{liq} and u^{n+1}_{vap} by
$$u^{n+1}_i = u^*_i - \frac{\Delta t}{\rho_i}\frac{\partial}{\partial x}P^{n+\frac{1}{2}}. \tag{4.20}$$

5. Constant extrapolation of the updated velocities across the interface Constant extrapolation of u_{liq}^{n+1} from the liquid to the vapor phase, and of u_{vap}^{n+1} from the vapor to the liquid phase, to compute updated velocity ghost values, eliminating the velocity jump on each updated velocity field, enabling differentiation across the interface for the next solver iteration.

4.3.2 Solving the Accurate Conservative Level Set equations

In one dimension, the interface normal vector need not be computed from the signed distance function by Eq. (3.27). As a result, one does not need to compute the signed distance function, and the considerations on the potential under- and overshoots of the Accurate Conservative Level Set function due to advection and reinitialization discussed on Section 3.4.1 simply vanish. The one-dimensional case is more permissive concerning *small* interval overshoots of the CLS function outside the interval $[0; 1]$.

4.3.3 Solving the temperature equation with immersed Dirichlet boundary condition at the interface

Enforcing immersed Dirichlet boundary condition (4.11) is challenging. Indeed, because the liquid-vapor interface is not lying on grid nodes but can be located anywhere between them, the immersed boundary condition has to be imposed at a subgrid level corresponding to the exact interface location. Due to the fact that condition (4.11) is of Dirichlet type, the subdomains Ω_1 and Ω_2 corresponding to the liquid and vapor phases can be considered separated from the temperature equation point of view. As a result, the temperature equation can be solved independently on both subdomains, provided that condition (4.11) is enforced at the interface from both phases. Considering the computational domain depicted in Fig. 4.4, we use the Ghost Fluid Method detailed in chapter 2 to fulfill two purposes : first, that condition (4.11) is satisfied from both phase perspectives ; second, that one temperature field is advanced by Eq. (4.10) without the influence of the other temperature field. The method consists in imposing condition (4.11) in the advection term of Eq. (4.10). We recall here the definition of the gradient operator in the finite volume framework as detailed in chapter 2 or chapter 3. In one dimension, the gradient operator $\partial \cdot / \partial \mathbf{x}$ applied to a scalar field T is evaluated at node i by

$$\left.\frac{\partial T}{\partial \mathbf{x}}\right|_i = \frac{T_{i+1} - T_{i-1}}{2\Delta x}\mathbf{e}_x. \tag{4.21}$$

We need to modify the rhs of Eq. (4.21) to satisfy condition (4.11). We detail the procedure for the liquid temperature field. The counterpart with the vapor phase is straightforward.

In Fig. 4.4, the liquid phase is on the left of the interface, and the interface is between nodes i in the liquid and $i+1$ in the vapor. Due to the independence of the two subdomains Ω_1 and Ω_2 with respect to the temperature equation (thanks to the Dirichlet condition imposed on Γ), the value of T_{i+1} must not be used in Eq. (4.21) when solved for the liquid phase. The Ghost Fluid Method is a well-suited method to handle discontinuities in a two-phase flow. It enables the use of two temperature fields, one for each phase. The liquid and vapor temperature fields are denoted T_{liq} and T_{vap} respectively. Both T_{liq} and T_{vap} are defined in the whole domain. The vapor part of T_{liq} is used to store liquid ghost values. The liquid part of T_{vap} is used to store vapor ghost values. Again, this is the principle of the Ghost Fluid Method (see chapter 2 for more details). The liquid

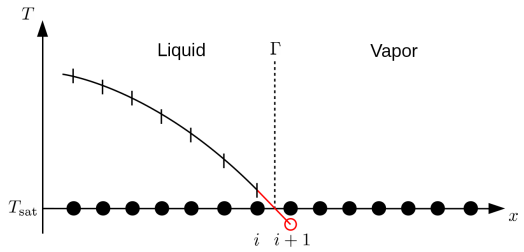


Fig. 4.4 – Representation of a 1D computational domain with liquid on the left of the interface Γ, vapor on the right, and linear extrapolation according to Eq. (4.24) of the liquid temperature field ensuring $T_\Gamma := T_{\text{sat}}$ by the use of the Ghost Fluid Method.

temperature value at node $i+1$ is then a ghost value, and Eq. (4.21) becomes

$$\left.\frac{\partial T}{\partial \mathbf{x}}\right|_i = \frac{T^G_{i+1} - T_{i-1}}{2\Delta x}\mathbf{e}_x, \tag{4.22}$$

where the superscript ^{G} denotes a ghost value. The value T^G_{i+1} has no physical meaning from the liquid phase point of view since it is not in the liquid phase, but permits to enforce condition (4.11) for the liquid phase. To this purpose, several options exist as stated in [27] : constant extrapolation,

$$T^{\text{G}}_{i+1} = T_{\text{sat}}, \tag{4.23}$$

linear extrapolation between node i and the interface,

$$T^{\text{G}}_{i+1} = T_i + \frac{T_{\text{sat}} - T_i}{\theta}, \tag{4.24}$$

linear extrapolation between node $i-1$ and the interface

$$T^{\text{G}}_{i+1} = T_{i-1} + \frac{T_{\text{sat}} - T_{i-1}}{1+\theta}, \tag{4.25}$$

quadratic extrapolation between nodes $i-1$, i and the interface,

$$T^{\text{G}}_{i+1} = \frac{2T_{\text{sat}} + \left(2\theta^2 - 2\right)T_i + \left(1-\theta^2\right)T_{i-1}}{\theta^2 + \theta}, \tag{4.26}$$

cubic extrapolation between nodes $i-2$, $i-1$, i and the interface, etc... where θ is the relative distance detailed in chapter 3. We choose the option that is the most compatible with a multi-dimensional simulation, it being understood that our goal is to extend the methodology to two- and three-dimensional simulations in the next chapters. As it will be explained in chapter 5, the most suitable option is then the linear extrapolation (4.24) between node i and the interface. It is then important to enforce condition (4.11) at the interface and not on the closest nodes to the interface since, in the latter case, the computation of the temperature ghost value T^{G}_{i+1} by linear extrapolation (4.24) would lead to $T^{\text{G}}_{i+1} = T_i$, and to a null temperature gradient between nodes i and $i+1$, which in turn would affect the mass transfer rate $\dot{m}$ (see Sections 4.2.5 and 4.3.4). By definition of θ, the linear extrapolation (4.24) enforces condition (4.11) along the grid edge $[i; i+1]$.

Fig. 4.4 shows the application of Eq. (4.24) to a 1D computational domain. By symmetry, the vapor ghost value at node i is given by

$$T_i^{\mathrm{G}} := T_{i+1} + \frac{T_{\mathrm{sat}} - T_{i+1}}{1-\theta}. \tag{4.27}$$

Substituting Eq. (4.24) in Eq. (4.22) gives a new expression for the gradient operator of T at node i,

$$\left.\frac{\partial T}{\partial \mathbf{x}}\right|_i = \frac{T_{\mathrm{sat}} + (\theta-1)\,T_i - \theta T_{i-1}}{2\theta\Delta x}\mathbf{e}_x. \tag{4.28}$$

Finally, we use Eq. (4.28) to replace the expression of the gradient of T for the nodes close to the interface in Eq. (4.10), leading to the advection-diffusion equation for the liquid temperature with immersed Dirichlet boundary condition (4.11). An obvious problem occurs when the interface is very close to node i. In this case, θ tends to 0 and Eq. (4.24) can no longer be used to compute a ghost value on node $i+1$. A threshold has to be used on the value of θ to avoid division by 0 [27]. In the Boiling solver, if θ is smaller than 10^{-3}, then T_i is set to T_{sat} and Eq. (4.10) need not be solved on this node.

4.3.4 Computing the mass transfer rate

Accounting for the amount of liquid mass which crosses the interface per surface and time units, as detailed in Section 1.3.1, the mass transfer rate $\dot{m}$ has then a physical significance only at the points lying on the interface. Consequently, the temperature gradients in Eq. (4.12) have to be computed at the interface.

As an introduction to our numerical method in one dimension, the temperature gradients at the interface are approximated by their values at the closest nodes to the interface,

$$\dot{m} \cong \frac{1}{L_v}\left(-\lambda_{\mathrm{liq}}\left.\frac{\partial T_{\mathrm{liq}}}{\partial x}\right|_i \mathbf{e}_x\cdot\mathbf{n}_i + \lambda_{\mathrm{vap}}\left.\frac{\partial T_{\mathrm{vap}}}{\partial x}\right|_{i+1}\mathbf{e}_x\cdot\mathbf{n}_{i+1}\right), \tag{4.29}$$

involving the assumption that the variations of the liquid and vapor temperature gradients between node i and the interface, and node $i+1$ and the interface, respectively, are negligible, which is considered acceptable on a well-resolved flow. Using Eq. (4.29) to compute $\dot{m}$ still requires a liquid ghost value on node $i+1$ to compute $\partial T_{\mathrm{liq}}/\partial x|_i$ (and a vapor ghost value on node i to compute $\partial T_{\mathrm{vap}}/\partial x|_{i+1}$). Again, linear extrapolations (4.24) and (4.27) are used to compute these ghost values.

Once the temperature gradients are computed, one can compute the mass transfer rate "at the interface" with Eq. (4.29). This value of $\dot{m}$ is defined at the interface by the approximation (4.29), whereas the Conservative Level Set advection equation (4.9) is solved on grid nodes. A transport step of $\dot{m}$ from the interface to the nodes is then needed. As a first approximation in one dimension, $\dot{m}$ is computed on the closest vapor node to the interface and the obtained value is then copied on all the nodes of the narrow band defined around the interface (a finer computation will be used in two and three dimensions).

4.4 Numerical results

In order to assess the validity of the numerical method detailed in this chapter, the result of a one-dimensional simulation is now presented. Let $\Omega = [-L; L]$ be the computational domain where

ρ_{liq}	ρ_{vap}	ν_{liq}	ν_{vap}	λ_{liq}
$1000\,\text{kg}\,\text{m}^{-3}$	$1\,\text{kg}\,\text{m}^{-3}$	$1\times10^{-3}\,\text{Pa}\,\text{s}^{-1}$	$1.78\times10^{-5}\,\text{Pa}\,\text{s}^{-1}$	$0.73\,\text{W}\,\text{m}^{-1}\,\text{K}^{-1}$

λ_{vap}	$c_{\text{p,liq}}$	$c_{\text{p,vap}}$	L_v	T_{sat}
$2.51\times10^{-2}\,\text{W}\,\text{m}^{-1}\,\text{K}^{-1}$	$4038\,\text{J}\,\text{kg}^{-1}\,\text{K}^{-1}$	$2079\,\text{J}\,\text{kg}^{-1}\,\text{K}^{-1}$	$2.23\times10^{6}\,\text{J}\,\text{kg}^{-1}$	$373\,\text{K}$

Table 4.1 – Physical parameters used in one dimension [85].

$L = 4\times10^{-3}\,\text{m}$. Let $N = 100$ be the number of subdivisions of Ω such that $\Omega = \bigcup_{i=0}^{N-1}[x_i; x_{i+1}]$ where $\{x_i : i \in \{0,\dots,N\}\}$ is the computational node set associated to Ω and all subdivisions $[x_i; x_{i+1}]$ are of equal length Δx. Consider a vapor bubble of radius $R_0 = 1\times10^{-3}\,\text{m}$ centered at the origin of Ω, surrounded by a liquid phase. The physical parameters of interest are listed in Table 4.1. Both phases are initially at rest, i.e. the vapor and liquid velocity fields are initialized to zero. The vapor temperature field T_{vap} is initialized to $T_{\text{sat}} = 373\,\text{K}$ in both phases. The liquid temperature field T_{liq} is initialized to T_{sat} in the vapor phase. In the liquid phase, T_{liq} is initialized at time t_0 with the linear profile given by

$$T_{\text{liq}}(t_0,x) = \begin{cases} T_{\text{sat}} + \dfrac{T_\infty(t_0) - T_{\text{sat}}}{L - R_0}\left(|x| - R_0\right), & \text{if } |x| \geq R_0, \\ T_{\text{sat}}, & \text{otherwise,} \end{cases} \tag{4.30}$$

where $T_\infty(t_0) = T_{\text{liq}}(t_0, x_0) = T_{\text{liq}}(t_0, x_N)$ is used to control the slope of the profile and is set to $T_\infty(t_0) = 573\,\text{K}$. Due to the diffusion of the liquid temperature to the outside of the domain, permitted by the use of outlet boundary conditions on both left and right boundaries, T_∞ decreases during the simulation. The time step Δt is set to $1\times10^{-5}\,\text{s}$. We emphasize here that the purpose of this test-case is only to demonstrate the ability of our numerical method to compute, in one dimension, the interface movement by means of the liquid and vapor thermal fluxes at the interface. Thus we did not derive an analytical solution for T_{liq} at time $t > t_0$, and the analytical liquid temperature profile is known only at initialization by Eq. (4.30). Comparisons with analytical solutions will be provided in Chapters 5 and 6 in 2D and 3D. Figure 4.5 shows the interface motion between time $t_0 = 0\,\text{s}$ and arbitrary time $t_1 = 7.8\times10^{-2}\,\text{s}$ at which the initial bubble radius has been multiplied by a factor roughly equal to three. Figure 4.5(a) shows the evolution of the Accurate Conservative Level Set function by means of Eq. (4.9) in which the interface motion is only driven by the mass transfer rate $\dot{m}$ since the vapor velocity, while also recomputed at each iteration, remains null. On the x-axis, the initial position of the interface is denoted by $x_{\Gamma,0}$, and the position of the interface at time t_1 is denoted by $x_{\Gamma,1}$, both corresponding to the value $\psi(x_\Gamma)$ on the y-axis. Figures 4.5(b) and 4.5(d) show the liquid and vapor temperature profiles at times t_0 and t_1, respectively. Figure 4.5(f) shows the values of the mass transfer rate computed, by Eq. (4.29), from the thermal fluxes at the interface given by λ_{liq}, λ_{vap} and temperature profiles of Figs. 4.5(b) and 4.5(d). The values of the mass transfer rate are then copied on neighbor nodes as stated in Section 4.3.4. Figures 4.5(c) and 4.5(e) show the liquid and vapor velocity profiles at times t_0 and t_1, respectively. The interface motion is only due to phase change since the vapor and liquid phases are globally at rest. Indeed, the vapor velocity, while recomputed at every iteration, remains null. The liquid velocity at the interface is thus equal to the velocity jump at the interface given by Eq. (4.5), and is uniform in the two liquid portions of the domain, in accordance with Eq. (4.1). Figure 4.5(g) shows the pressure profile at times t_0 and t_1. The pressure is uniform in the vapor

phase, and the pressure gradient in the liquid phase is given by Eq. (4.8). Since the mass transfer rate increases with time (see again Fig. 4.5(f)), the liquid velocity also increases with time, by Eq. (4.5), as can be seen in Figs. 4.5(c) and 4.5(e). Equation (4.4) then states that the pressure gradient decreases in the liquid phase, from the interface to the outer of the domain, as can be observed in Fig. 4.5(g). Note that the small values of $\dot{m}$ shown in Fig. 4.5(f) lead, by Eq. (4.6), to a small pressure jump at the interface in Fig. 4.5(g) (in which the pressure field seems continuous at the interface). Due to the scale chosen for the y-axis, this small pressure jump is not visible in Fig. 4.5(g). These results show the ability of the Boiling solver to simulate the interface motion due to phase change in one dimension.

4.5 Conclusion

In this chapter, the numerical method implemented in the Boiling solver has been presented for one-dimensional simulations. We review here the main steps of the algorithm. The Navier-Stokes equation

$$\frac{\partial u_i}{\partial t} = -\frac{1}{\rho_i}\frac{\partial P}{\partial x}, \tag{4.31}$$

on the phase i with jump conditions at the interface

$$[\mathbf{u}]_\Gamma = \dot{m}\left[\frac{1}{\rho}\right]_\Gamma \mathbf{n}_\Gamma, \tag{4.32}$$

where $\mathbf{u} = \pm\|\mathbf{u}\|\,\mathbf{e}_x$ and $\mathbf{n}_\Gamma = \pm\mathbf{e}_x$, and

$$[P]_\Gamma = -\dot{m}^2\left[\frac{1}{\rho}\right]_\Gamma, \tag{4.33}$$

is solved with the two-step projection method

$$\begin{cases} \dfrac{\partial}{\partial x}\left(\dfrac{1}{\rho_i^{n+\frac{1}{2}}}\dfrac{\partial}{\partial x}P^{n+\frac{1}{2}}\right) = 0, \\ u_i^{n+1} = u_i^n - \dfrac{\Delta t}{\rho_i}\dfrac{\partial}{\partial x}P^{n+\frac{1}{2}}. \end{cases} \tag{4.34}$$

The Accurate Conservative Level Set ψ given by

$$\psi(t,x) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{\phi(t,x)}{2\epsilon(x)}\right) \tag{4.35}$$

is advected by

$$\frac{\partial \psi}{\partial t} + u_{\text{vap}}\frac{\partial \psi}{\partial x} = \frac{\dot{m}}{\rho_{\text{vap}}}\frac{\partial \psi}{\partial x}, \tag{4.36}$$

and reinitialized after advection by

$$\frac{\partial \psi}{\partial \tau} + \frac{\partial}{\partial x}\left(\psi\left(1-\psi\right)\right) = \epsilon\frac{\partial^2 \psi}{\partial x^2}. \tag{4.37}$$

The heat equation with immersed Dirichlet boundary condition

$$\begin{cases} \dfrac{\partial T_i}{\partial t} + u_i \dfrac{\partial T_i}{\partial x} = \dfrac{1}{\rho_i c_{\mathrm{p},i}} \dfrac{\partial}{\partial x}\left(\lambda_i \dfrac{\partial T_i}{\partial x}\right) & (4.38) \\ T_\Gamma = T_{\mathrm{sat}} & (4.39) \end{cases}$$

is solved where a ghost temperature value is computed at node x_{i+1} by

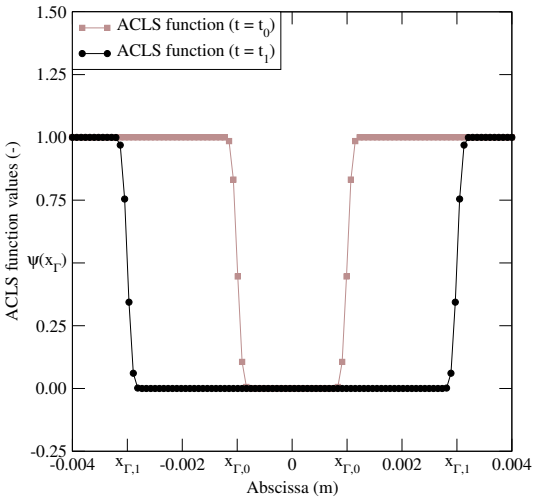
$$T_{i+1}^{\mathrm{G}} = T_i + \frac{T_{\mathrm{sat}} - T_i}{\theta} \tag{4.40}$$

when the interface is located between nodes x_i and x_{i+1}, in order to compute the corresponding temperature gradient at node x_i. The mass transfer rate is computed as

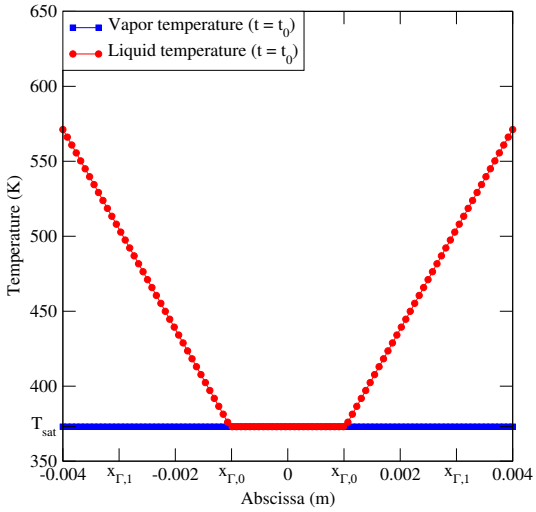
$$\dot{m} = \frac{1}{L_v}\left(-\lambda_{\mathrm{liq}} \left.\frac{\partial T_{\mathrm{liq}}}{\partial x}\right|_i \mathbf{e}_x \cdot \mathbf{n}_i + \lambda_{\mathrm{vap}} \left.\frac{\partial T_{\mathrm{vap}}}{\partial x}\right|_{i+1} \mathbf{e}_x \cdot \mathbf{n}_{i+1}\right), \tag{4.41}$$

where $\mathbf{n}_i$ is the interface normal vector shifted at node x_i.

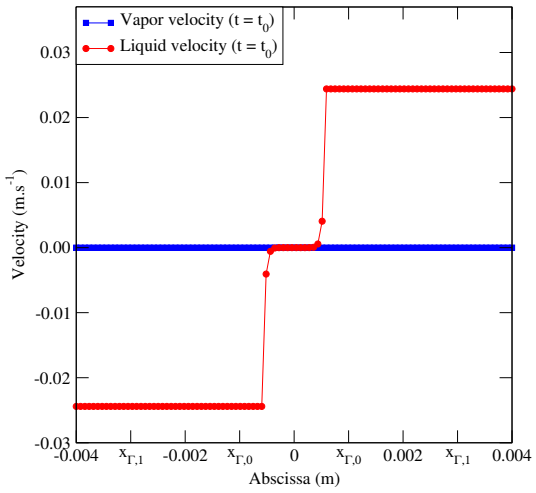
The implementation has been illustrated in one dimension. The next two chapters extend this methodology to two and three dimensions, where several numerical challenges arise mostly from the fact that the normal vector is not aligned with grid nodes.


ACLS function (t = t0)
ACLS function (t = t1)
ACLS function values (-)
ψ(xΓ)
Abscissa (m)

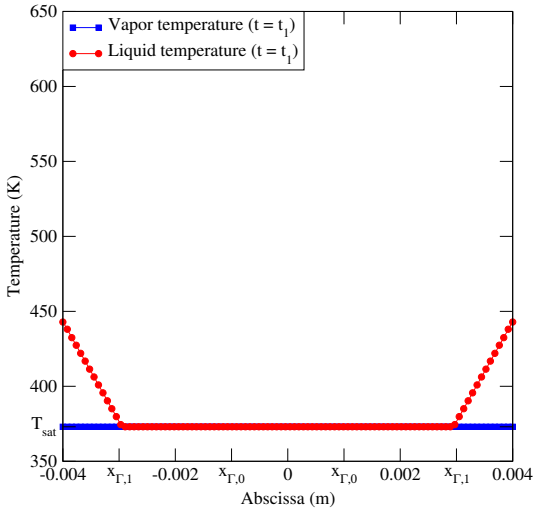
(a)


Vapor temperature (t = t0)
Liquid temperature (t = t0)
Temperature (K)
Tsat
Abscissa (m)

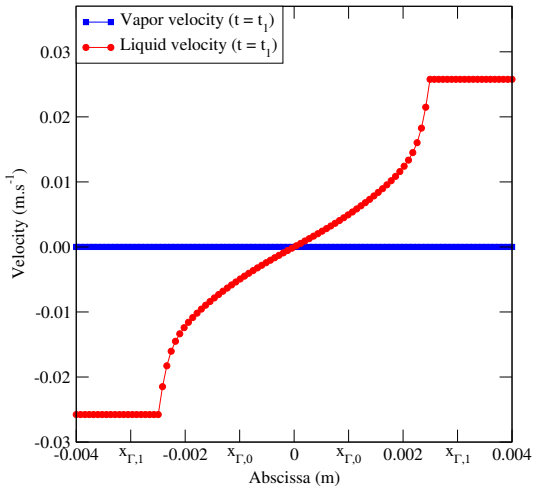
(b)


Vapor velocity (t = t0)
Liquid velocity (t = t0)
Velocity (m.s-1)
Abscissa (m)

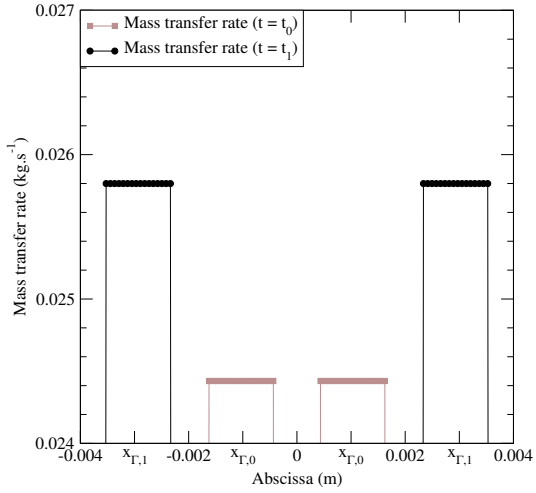
(c)


Vapor temperature (t = t1)
Liquid temperature (t = t1)
Temperature (K)
Tsat
Abscissa (m)

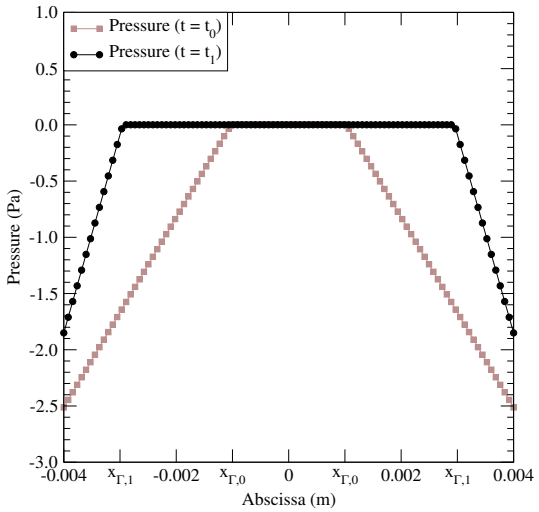
(d)


Vapor velocity (t = t1)
Liquid velocity (t = t1)
Velocity (m.s-1)
Abscissa (m)

(e)


Mass transfer rate (t = t0)
Mass transfer rate (t = t1)
Mass transfer rate (kg.s-1)
Abscissa (m)

(f)



(g)

Fig. 4.5 – Results of the one-dimensional simulation shown at initialization ($t_0 = 0\,\mathrm{s}$) and at time $t_1 = 7.8 \times 10^{-2}\,\mathrm{s}$ where the initial bubble radius has been multiplied by a factor roughly equal to three : (a) the Accurate Conservative Level Set function ψ, (b) and (d) the vapor and liquid temperature fields, (c) and (e) the vapor and liquid velocity fields, (f) the mass transfer rate, (g) the pressure field. On the x-axis, the initial interface location is represented by $x_{\Gamma,0}$, and the interface location at time t_1 is represented by $x_{\Gamma,1}$.

Chapter 5

Numerical simulation of two-phase flows with phase change in two and three dimensions with a fixed mass transfer rate

In this chapter, we focus on the simplified situation of fixed (constant and uniform) mass transfer rate across the two-phase interface, meaning that the mass transfer rate is decoupled from the temperature field. The accuracy of the method is first tested against a two-dimensional case. The interface normal vector and curvature have now to be computed with high precision to avoid too large deformations of the interface, potentially leading to numerical instabilities. The extension to three dimensions is provided at the end of the chapter. Boiling simulations using a mass transfer rate coupled to the temperature field will be addressed in the next chapter.

Outline

5.1 Introduction

Multidimensional simulations of two-phase flows are much more challenging than monodimensional ones. Many of the simplifications used in one dimension would lead to complete aberrations in two or three dimensions, when not mathematically ill-defined. The main difference between one- and two- (three-)dimensional boiling simulations is that, in the first case, the interface is a disjoint point set, whereas in the second case, it is a curve (surface). This difference has one main consequence : the interface normal vector can not be fixed to $\pm\mathbf{e}_x$ anymore and has to be computed. It is indeed a crucial ingredient involved in many steps of our method. The interface normal vector is present in the Accurate Conservative Level Set time advancement and reinitialization equations and in the velocity and pressure jumps at the interface. In this chapter, we show that the interface normal vector is also used in other steps of the method when multidimensional simulations are considered. The interface curvature is well-defined and must be computed to take into account surface tension forces at the interface.

Section 5.2 recalls the governing equations and the numerical method used for two-phase flow simulations with phase change in multidimensions. Section 5.3 presents some preliminary results obtained with this methodology. Section 5.4 explains the lack of accuracy of the previous method. Sections 5.5 to 5.10 detail the implementation of enhanced Level Set method variants developed along this thesis to allow for multidimensional simulations on structured and unstructured grids. Section 5.11 shows the improved accuracy in the interface transport provided by the different methods on structured and unstructured grids. Finally, Section 5.12 concludes this chapter.

5.2 Governing equations

We recall the governing equations for two-phase flow simulations with phase change in two and three dimensions.

5.2.1 Incompressible Navier-Stokes equations with phase change

In this section, we recall the steps of the projection method presented in Section 2.6 with the contribution of phase change detailed in Section 4.3.1, extended to the multidimensional case.

Incompressible Navier-Stokes equations with phase change are given in each phase by

$$\frac{\partial \mathbf{u}_i}{\partial t} + (\mathbf{u}_i \cdot \nabla)\,\mathbf{u}_i = -\frac{1}{\rho_i}\nabla P + \frac{1}{\rho_i}\nabla \cdot \left(\mu_i \left(\nabla \mathbf{u}_i + \nabla \mathbf{u}_i^T\right)\right) \tag{5.1}$$

where $\mathbf{u}$ is the velocity, P the pressure, ρ the density and μ the dynamic viscosity of the phase denoted by the $_i$ subscript. The incompressibility hypothesis reads

$$\nabla \cdot \mathbf{u}_i = 0. \tag{5.2}$$

In the projection method, the velocity predictors are given by

$$\mathbf{u}_i^* = \mathbf{u}_i^n + \Delta t \left(-(\mathbf{u}_i^n \cdot \nabla)\,\mathbf{u}_i^n + \frac{1}{\rho_i^{n-\frac{1}{2}}}\nabla \cdot \left(\mu_i \left(\nabla \mathbf{u}_i^n + (\nabla \mathbf{u}_i^n)^T\right)\right)\right). \tag{5.3}$$

The updated velocities are given by the Helmholtz decomposition applied to $\mathbf{u}_i^*$,

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^* - \frac{\Delta t}{\rho_i^{n+\frac{1}{2}}}\nabla P^{n+\frac{1}{2}}. \tag{5.4}$$

The divergence of Eq. (5.4) and the incompressibility hypothesis (5.2) applied to $\mathbf{u}_i^{n+1}$ lead to the Poisson equation for $P^{n+\frac{1}{2}}$,

$$\nabla \cdot \left(\frac{1}{\rho_i^{n+\frac{1}{2}}} \nabla P^{n+\frac{1}{2}} \right) = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}_i^*. \tag{5.5}$$

The pressure discontinuity at the interface is given by

$$[P]_\Gamma = \sigma\kappa - \dot{m}^2 \left[\frac{1}{\rho} \right]_\Gamma, \tag{5.6}$$

where $\dot{m}$ is the mass transfer rate expressed in $\mathrm{kg\,m^{1-d}\,s^{-1}}$, with d the spatial dimension, and the dynamic viscosity contribution is neglected. The velocity discontinuity at the interface is given by

$$[\mathbf{u}]_\Gamma = \dot{m} \left[\frac{1}{\rho} \right]_\Gamma \mathbf{n}_\Gamma. \tag{5.7}$$

Poisson equation (5.5) is solved using the methodology detailed in Appendix B in which the pressure and velocity discontinuity Eqs. (5.6) and (5.7) are imposed at the exact interface location. The linear system formed from Eq. (5.5) and jump conditions (5.6) and (5.7) is solved using the Deflated Pre-Conjugate Gradient (DPCG) solver [49, 50]. Once $P^{n+\frac{1}{2}}$ is known, the velocity predictors $\mathbf{u}_i^*$ are corrected using Eq. (5.4) to obtain the updated velocities $\mathbf{u}_i^{n+1}$.

5.2.2 Constant extrapolation of the velocities across the interface in the interface normal direction

In Section 4.3.1, constant extrapolations of the velocities across the interface have been introduced to be used in the projection method in one dimension. In multidimensions, such extrapolations of the velocities are still required to apply the projection method described in Section 5.2.1 in order to compute derivatives of the velocities close to the interface in Eqs. (5.3) and (5.5). The major difference with the one-dimensional case is the unknown direction of the interface normal vector $\mathbf{n}_\Gamma$. Indeed, in one dimension, $\mathbf{n}_\Gamma$ is always equal to $\pm\mathbf{e}_x$, whereas in multidimensions, $\mathbf{n}_\Gamma$ can point to any direction and so has to be computed. It is then straightforward to infer that the accuracy with which $\mathbf{n}_\Gamma$ will be computed will strongly influence the overall accuracy of the simulation. Once $\mathbf{n}_\Gamma$ is known, the liquid velocity $\mathbf{u}_{\mathrm{liq}}$ is extrapolated across the interface by solving the equation

$$\frac{\partial \mathbf{u}_{\mathrm{liq}}}{\partial \tau} + \nabla \mathbf{u}_{\mathrm{liq}} \cdot \mathbf{n}_\Gamma = 0 \tag{5.8}$$

only on the vapor nodes of the narrow band around the interface. The same extrapolation is computed for $\mathbf{u}_{\mathrm{vap}}$ on the liquid nodes of the narrow band. More details on Eq. (5.8) will be found in Aslam [4] and in Appendix G.

5.2.3 Accurate Conservative Level Set method

The advection equation of the Accurate Conservative Level Set function described in Section 3.4 and extended to phase change in the previous chapter is given in multidimensions by

$$\frac{\partial \psi}{\partial t} + \nabla \cdot \left(\psi \left(\mathbf{u}_{\mathrm{vap}} + \mathbf{u}_\Gamma \chi_\Gamma \right) \right) = 0, \tag{5.9}$$

where $\mathbf{u}_\Gamma = -\left(\dot{m}/\rho_{\text{vap}}\right)\mathbf{n}_\Gamma$ is the contribution of phase change to the interface velocity, and χ_Γ is the indicator function of the interface defined as

$$\chi_\Gamma\left(t,\mathbf{x}\right) = \begin{cases} 1, & \text{if } \psi\left(t,\mathbf{x}\right) = 0.5, \\ 0, & \text{otherwise.} \end{cases} \tag{5.10}$$

The divergence in the lhs of Eq. (5.9) can not be computed due to the two following incompatibilities :

Finite Volume node-centered discretization The node-centered convention used in YALES2 for the discretization of the differential operators in the finite volume framework implies the creation of a domain partition where each part called control volume is attached to a node. By definition of a partition, there is then no space left on Ω to define a control volume that would be attached to a subgrid point of Ω (the creation of a partition is mandatory in the finite volume method, no control volume overlapping is allowed). The interface Γ is a good example of such a subgrid point set : one can not compute the divergence of a field defined only at the interface in the finite volume framework.

Dimension mismatch A second condition for the usability of the divergence operator (and other differential operators) in the finite volume framework is that the definition set of the vectors of which the divergence is computed must be of the same dimension as the whole domain Ω. Unfortunately, this property is not satisfied by the interface Γ, and one has

$$\dim\Gamma = \dim\Omega - 1. \tag{5.11}$$

Using the identity

$$\nabla\cdot\left(\psi\mathbf{u}_\Gamma\chi_\Gamma\right) = \chi_\Gamma\mathbf{u}_\Gamma\cdot\nabla\psi + \psi\nabla\cdot\left(\chi_\Gamma\mathbf{u}_\Gamma\right), \tag{5.12}$$

$$= \chi_\Gamma\mathbf{u}_\Gamma\cdot\nabla\psi + \psi\left(\chi_\Gamma\nabla\cdot\mathbf{u}_\Gamma + \mathbf{u}_\Gamma\cdot\nabla\chi_\Gamma\right), \tag{5.13}$$

and neglecting the derivatives of quantities defined only at the interface, the interface advection due to phase change is solved under non-conservative form, i.e. $\nabla\cdot\left(\psi\mathbf{u}_\Gamma\chi_\Gamma\right)$ is modified to $\chi_\Gamma\mathbf{u}_\Gamma\cdot\nabla\psi$ in Eq. (5.9) which is rewritten

$$\frac{\partial\psi}{\partial t} + \nabla\cdot\left(\psi\mathbf{u}_{\text{vap}}\right) = -\mathbf{u}_\Gamma\cdot\nabla\psi\chi_\Gamma, \tag{5.14}$$

where phase change is considered as a source term in the advection of the ACLS function. The rhs of Eq. (5.14) still contains a velocity term only defined at the interface, whereas the equation is solved at grid nodes. One needs to make grid nodes aware of $\mathbf{u}_\Gamma$ in a coherent manner. A first observation can be done : only grid nodes relatively close to the interface need to infer the value of $\mathbf{u}_\Gamma$. Attention must be paid to the different node levels that need to be provided a value of $\mathbf{u}_\Gamma$. In order to avoid numerical difficulties, all grid nodes located in the area where $\nabla\psi$ is non zero are provided a value of $\mathbf{u}_\Gamma$. For ease of implementation, this area is extended to the narrow band defined is Section 2.4.2. The contribution of phase change to the interface motion is given by the second term of the rhs of Eq. (1.33), i.e.

$$\mathbf{u}_\Gamma = -\frac{\dot{m}}{\rho_{\text{vap}}}\mathbf{n}_\Gamma. \tag{5.15}$$

In this chapter, the mass transfer rate is imposed to a constant and uniform value on the whole interface. As a result, no special operation is required to populate the narrow band nodes with the value of the mass transfer rate. In this particular case, the imposed $\dot{m}$ value is simply assigned to the concerned nodes. As a result, the Navier-Stokes equations and the heat equation are decoupled. Equation (5.14) is written under conservative form with source term, i.e.

$$\text{temporal variation of } \psi + \text{divergence of the flux } \psi\mathbf{u}_{\text{vap}} = \text{source term}, \tag{5.16}$$

but the actual form of an advection equation with source term for ψ is given by

$$\text{temporal variation of } \psi + \text{advection of } \psi \text{ by the velocity } \mathbf{u}_{\text{vap}} = \text{source term}. \tag{5.17}$$

Since we use the incompressible form of the Navier-Stokes equations, the vapor velocity is theoretically divergence-free. Nevertheless, the constant extrapolation of $\mathbf{u}_{\text{vap}}$ across the interface does not guarantee that the ghost values of $\mathbf{u}_{\text{vap}}$ in the liquid phase are divergence-free. Moreover, the divergence of $\mathbf{u}_{\text{vap}}$ can suffer from numerical errors caused by the numerical schemes used to advect ψ even on nodes far from the interface, due to the limited precision of the differential operators on unstructured grids. In order to minimize these problems, the possibly non-zero divergence of $\mathbf{u}_{\text{vap}}$ is substracted from the lhs of Eq. (5.14), leading to

$$\frac{\partial\psi}{\partial t} + \{\nabla\cdot(\psi\mathbf{u}_{\text{vap}}) - \psi\nabla\cdot\mathbf{u}_{\text{vap}}\} = -\mathbf{u}_\Gamma\cdot\nabla\psi\chi_\Gamma. \tag{5.18}$$

Using Eq. (5.12) transposed to ψ and $\mathbf{u}_{\text{vap}}$, one establishes the mathematical equality of the term between braces in the lhs of Eq. (5.18) to $\mathbf{u}_{\text{vap}}\cdot\nabla\psi$. Consequently, Eq. (5.18) is the actual advection equation of ψ written under form (5.17). In order to benefit from the conservative form, Eq. (5.18) is rewritten

$$\frac{\partial\psi}{\partial t} + \nabla\cdot(\psi\mathbf{u}_{\text{vap}}) = -\mathbf{u}_\Gamma\cdot\nabla\psi\chi_\Gamma + \psi\nabla\cdot\mathbf{u}_{\text{vap}}, \tag{5.19}$$

where $\psi\nabla\cdot\mathbf{u}_{\text{vap}}$ is considered positively in the rhs. Equation (5.19) written under conservative form enforces conservation of the quantity ψ by integration of the flux $\psi\mathbf{u}_{\text{vap}}$ over control volumes to compute its divergence (see Section 3.1.1), whereas it enforces the incompressibility hypothesis ($\nabla\cdot\mathbf{u}_{\text{vap}} = 0$) by removing the divergence of $\mathbf{u}_{\text{vap}}$ which is non zero in the ghost liquid part of $\mathbf{u}_{\text{vap}}$. The conflict between the indicator function of the interface in Eq. (5.19) and the fact that this equation is solved on grid nodes has to be resolved. In the case where the mass transfer rate is imposed, the function χ is simply removed from the equation, resulting by Eq. (5.15) in the approximation

$$\mathbf{n}_\Gamma\cdot\nabla\psi|_\Gamma = \mathbf{n}_\mathbf{p}\cdot\nabla\psi|_\mathbf{p}\,, \tag{5.20}$$

for all nodes $\mathbf{p}$ in the narrow band around the interface. Outside the narrow band, $\dot{m}$ is set to zero. The advection equation of the Accurate Conservative Level Set function is then given by

$$\frac{\partial\psi}{\partial t} + \nabla\cdot(\psi\mathbf{u}_{\text{vap}}) = \frac{\dot{m}}{\rho_{\text{vap}}}\mathbf{n}_\Gamma\cdot\nabla\psi + \psi\nabla\cdot\mathbf{u}_{\text{vap}}, \tag{5.21}$$

where the first term of the rhs vanishes outside the narrow band.

After advection of ψ by Eq. (5.21), the CLS function is reinitialized to the hyperbolic tangent of the new signed distance function by Eq. (2.24) where the interface normal vector from the previous iteration is used. After reinitialization, the updated interface normal vector and curvature are computed by Eqs. (3.27) and (3.28).

5.3 First numerical results

We evaluate the numerical method presented in this chapter on the case of a 2D static growing bubble with a fixed mass transfer rate from [85]. The computational domain is a square of side length $L = 8 \times 10^{-3}\,\mathrm{m}$. The initial bubble radius is $R_0 = 1 \times 10^{-3}\,\mathrm{m}$ and the imposed mass transfer rate is $\dot{m} = 1 \times 10^{-1}\,\mathrm{kg\,m^{-2}\,s^{-1}}$. The simulations are performed until final time $t_\mathrm{f} = 1 \times 10^{-2}\,\mathrm{s}$ needed for the bubble radius to double the initial radius. The other physical parameters of interest are $\rho_\mathrm{liq} = 1 \times 10^{3}\,\mathrm{kg\,m^{-3}}$, $\rho_\mathrm{vap} = 1\,\mathrm{kg\,m^{-3}}$, $\sigma = 7 \times 10^{-2}\,\mathrm{N\,m^{-1}}$, $\mu_\mathrm{liq} = 1 \times 10^{-3}\,\mathrm{kg\,m^{-1}\,s^{-1}}$ and $\mu_\mathrm{vap} = 1.78 \times 10^{-5}\,\mathrm{kg\,m^{-1}\,s^{-1}}$.

5.3.1 Interface displacement due to level set reinitialization numerical errors

Prior to the actual simulation, Eq. (2.24) is solved at initialization in order to highlight the resulting displacement of the liquid-vapor interface, as shown in Fig. 5.1. The implementation of

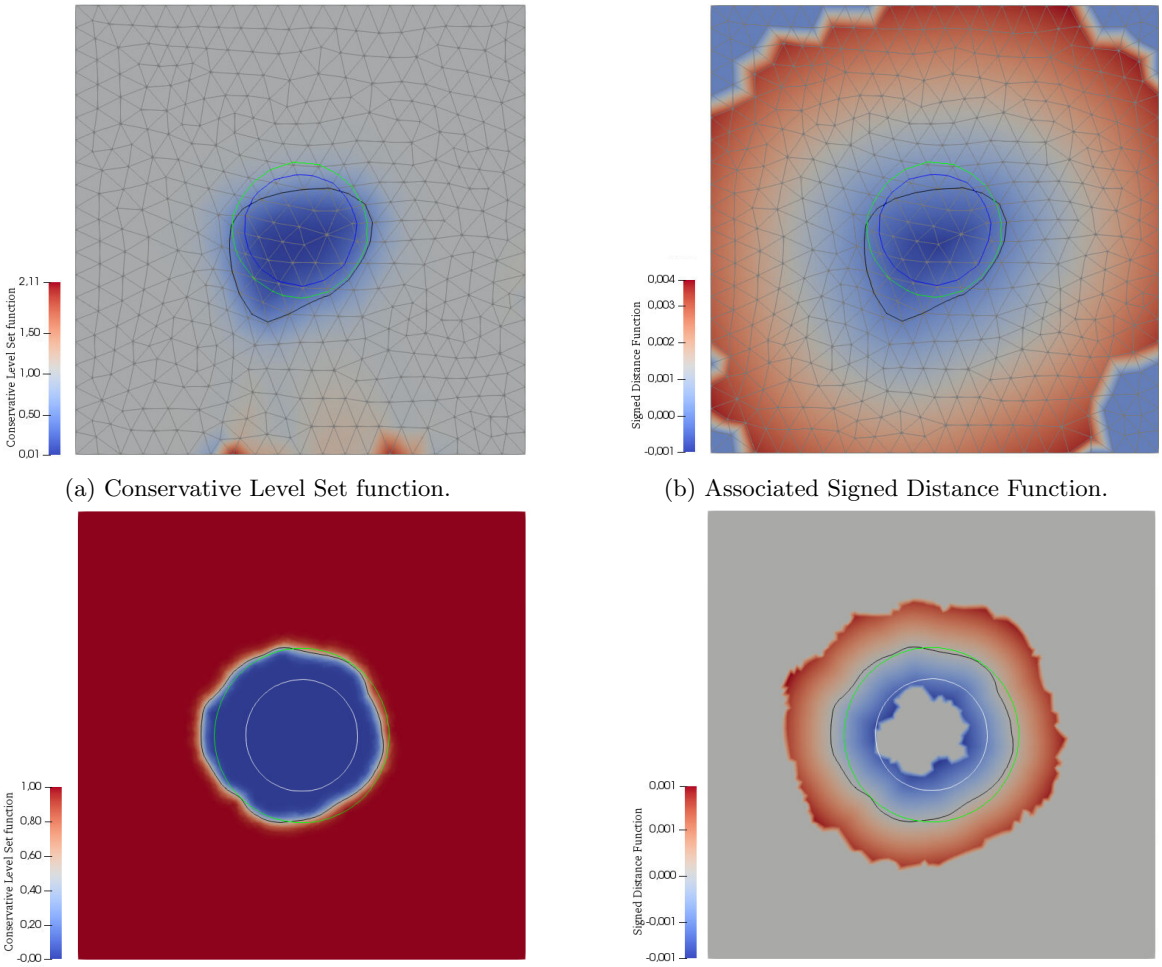
(a) (b)

Fig. 5.1 – Interface displacement due to the reinitialization of the level set function performed at initialization by Eq. (2.24) on two unstructured grids : (a) $\Delta x = 4 \times 10^{-4}\,\mathrm{m}$ and (b) $\Delta x = 1 \times 10^{-4}\,\mathrm{m}$. The initial interface is shown in blue and the interface after reinitialization of the level set function is shown in red. The interface is initially circular, the observed discontinuities at grid edges are due to the interpolation performed by the visualization software in each grid triangle.

the level set reinitialization method then needs to be improved in order to preserve the interface location resulting from the advection of the level set function by Eq. (5.21).

5.3.2 First attempt to validate the numerical method

Figure 5.2 shows the result on two different unstructured grids. One can see the lack of accuracy in the interface localization. Indeed, since the mass transfer rate is constant and uniform, the bubble



(a) Conservative Level Set function.

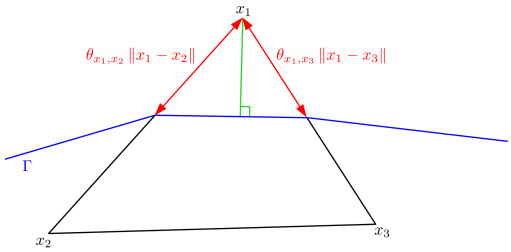
(b) Associated Signed Distance Function.

(c) Conservative Level Set function.

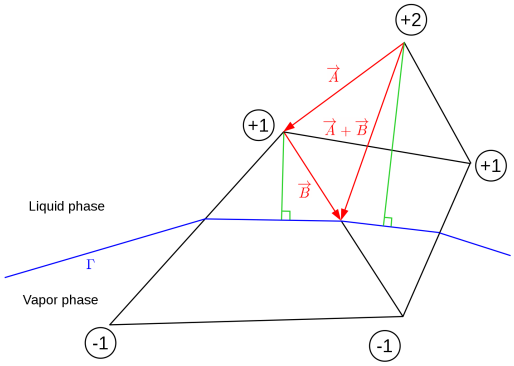
(d) Associated Signed Distance Function.

Fig. 5.2 – Static circular bubble growing by means of a fixed (uniform and constant) mass transfer rate. The bubble shape is expected to be a circle at any time of the simulation. Subfigs. (a) and (b) show the simulation on a grid of cell size $\Delta x = 4 \times 10^{-4}$ m and Subfigs. (c) and (d), on a grid of cell size $\Delta x = 1 \times 10^{-4}$ m. The initial interface is shown in blue in (a) and (b) and in white in (c) and (d). Shortly before the failure of the simulation, the computed interface is shown in black while the corresponding theoretical interface is shown in green.

should remain circular while expanding. In both cases, the simulation failed soon after the physical times of the snapshots. On the coarsest grid (Subfigs. 5.2(a) and 5.2(b)), the bubble radius increased weakly before the failure of the simulation (initial interface in blue and last theoretical interface before failure in green in Subfig. 5.2(a)). On the finest grid (Subfigs. 5.2(c) and 5.2(d)), the bubble radius further increased before failure. While the computed interface shown on the finest grid is globally closer to the corresponding theoretical interface than the computed interface on the



(a) Computation of signed distance values on closest nodes to the interface (band level +1).



(b) Computation of signed distance values on neighbor nodes of the closest nodes to the interface (band level +2).

Fig. 5.3 – Computation of the signed distance function to the interface by the Simple Marker Method in which the distance of a node to the interface is approximated as the distance to the closest intersection of the edge and the piecewise-linear interface. Once all closest nodes to the interface (level 1 in the narrow band) have an updated signed distance value (a), updated values are computed for their closest neighbors (level 2 in the narrow band) (b), and so on until the boundary of the narrow band is reached.

coarsest grid, one has to notice that grid refinement also increases small interface distortions, as can be seen comparing the two computed interfaces shown in black. As a result, the interface curvature given by Eq. (2.12) is highly polluted by these distortions of the signed distance function.

As will be detailed in Section 5.4, these distortions are due to a slight inaccuracy in the signed distance function computation. One has to notice that the interface distortions will in general be higher in the case where the mass transfer rate is computed (see Chapter 6), due to inevitable numerical errors in its computation. This test case is even more challenging in three dimensions where the interface is expected to remain spherical during boiling. The next part of this chapter is devoted to the improvement of the signed distance function computation using different numerical methods.

5.4 Inaccuracy of the Simple Marker Method in the computation of the Signed Distance Function to the interface

At the beginning of this thesis, YALES2 used the method presented in Section 3.4.1 to compute the signed distance function to the interface. In order to briefly recall the principle of this method, Figs. 3.10 and 3.11 are reproduced in Fig. 5.3. The signed distance function is approximated on nodes of band level +1 by their distance to the closest intersection of a grid edge and the piecewise-linear interface, as shown in red in Subfig. 5.3(a), in which the actual distance is given by the length of the green segment. Subfigure 5.3(b) extends this method using Chasles's identity (3.26) to nodes of band level +2. Similar operations are performed on nodes of band levels greater than 2. This method is robust and gives a signed distance value at every node, being on one,

two or three dimensions, on structured or unstructured grids, and on sequential or parallel domain decomposition. In this manuscript, we refer to this method as the Simple Marker Method (SMM). The drawback of this method is its important lack of accuracy. On Fig. 5.3.(a), one can clearly notice the error made on the signed distance values of the closest nodes to the interface. The error depends on the angle between the element edges crossed by the interface and the piecewise-linear interface inside the element. If edge $(\mathbf{x}_1, \mathbf{x}_2)$ is orthogonal to the interface, then the error on the signed distance value on node $\mathbf{x}_1$ is null. In the general case, edge $(\mathbf{x}_1, \mathbf{x}_2)$ is not orthogonal to the interface, and the error is not negligeable. Figure 5.3.(b) shows the same problem for signed distance values on nodes of band level +2. One can not advect the Conservative Level Set function with Eq. (5.21) and reinitialize it with Eq. (2.24) if the interface normal vector $\mathbf{n}_\Gamma$ is computed as the gradient of the signed distance function ϕ reinitialized by the Simple Marker Method.

5.4.1 Collaboration with the CORIA team

Some improvements of the method have been tested in collaboration with Vincent Moureau from the CORIA team responsible for the main development and coordination of YALES2. For instance, the signed distance values on the closest nodes can be corrected by the projection of the nodes onto the interface (see the green segment on Fig. 5.3.(a)).

YALES2 is mainly used to solve the incompressible Navier-Stokes equations on 3D unstructured grids. Single-phase flow simulations are now mature in the CFD community, and a strong interest in two-phase flow simulations has been shown for the last few years to take into account more complex physical phenomena. Among YALES2 users, these phenomena have mainly been liquid jet atomization and combustion. Since two-phase flows imply the presence of an interface separating the two phases, the problem of the signed distance function reinitialization appears, meaning that other research teams using YALES2 also faced this problem on unstructured grids with MPI parallelism. As a consequence, V. Moureau's team also investigated the problem. In collaboration with him and his team, we improved the SMM by computing the signed distance value of a node of band level ± 1 as the distance between this node and its projection onto the closest segment representing the piecewise-linear interface. We only mention here that, while as expected, this correction improved the accuracy of ϕ values on the closest nodes to the interface, it was still insufficient to accurately reinitialize ϕ on farther nodes to the interface in order to compute the interface normal vector by Eq. (3.27). Further improvements were needed and are developed in the following sections.

5.5 The Multiple Marker Method

V. Moureau developed the Multiple Marker Method (MMM) in which the notion of marker is extended to the whole segment representing the piecewise-linear interface inside a given grid element (or the whole polygon representing the piecewise-planar interface inside a given grid element in three dimensions). Let $(\mathbf{x}_1, \mathbf{x}_2)$ be a grid edge crossed by the interface Γ. In the MMM, if $n \in \{1, \ldots, 3\}$ is the spatial dimension, a marker $M_i(\mathbf{x}_1)$ associated to node $\mathbf{x}_1$ is the $(n+1)$-dimensional vector given by

$$M_i(\mathbf{x}_1) = \left(x_{\Gamma_1}, \ldots, x_{\Gamma_n}, \|\mathbf{x}_1 - \mathbf{x}_\Gamma\|\right)^{\mathrm{T}}, \tag{5.22}$$

where $\mathbf{x}_\Gamma = (x_{\Gamma_1}, \ldots, x_{\Gamma_n})^{\mathrm{T}}$ is the coordinate vector of the intersection point between edge $(\mathbf{x}_1, \mathbf{x}_2)$ and the interface Γ, and $i \in I_n$ with

$$I_n = \begin{cases} \{1\}, & \text{if } n = 1, \\ \{1, \ldots, 3\}, & \text{if } n = 2, \\ \{1, \ldots, 6\}, & \text{if } n = 3, \end{cases} \tag{5.23}$$

i.e. one defines one marker in one dimension, three markers in two dimensions and six markers in three dimensions. These values have been adopted based on empirical tests. The signed distance value ϕ on node $\mathbf{x}_1$ is then given by

$$\phi(\mathbf{x}_1) = \min_{i \in I_n} \left\{ (M_i(\mathbf{x}_1))_{n+1} \right\}, \tag{5.24}$$

where $(M_i(\mathbf{x}_1))_{n+1}$ is the last component of vector $M_i(\mathbf{x}_1)$. Otherwise stated, Eqs. (5.22) and (5.23) show that marker lists are composed of one marker in one dimension, three markers in two dimensions and six markers in three dimensions.

In the MMM, the reinitialization of ϕ on the closest nodes to the interface is similar to that of the SMM. The difference concerns the reinitialization of ϕ on the farther nodes. Indeed, in the MMM, a list of markers is built for each node. Each list is ordered in ascending order (in absolute value) of the values of its $(n+1)$-th element components, i.e. for any node $\mathbf{x}$, the first marker $M_1(\mathbf{x})$ in the marker list asssociated to $\mathbf{x}$ is the closest marker to $\mathbf{x}$. Since markers are located at the interface, $\phi(\mathbf{x})$ is updated as the $(n+1)$-th component of marker $M_1(\mathbf{x})$.

Consequently, such marker lists are useless for the closest nodes to the interface since only one marker is needed to update ϕ. The advantage of marker lists appears when ϕ is reinitialized on farther nodes to the interface. When ϕ has been reinitialized on all closest nodes to the interface, then ϕ is reinitialized on nodes of band level ± 2, and so on until the narrow band boundary (nodes of band level ± 8 by default in YALES2). For clarity, only positive values of band levels are considered in the following part. In order to reinitialize ϕ on nodes of band level 2, nodes of band level 1 share their marker list to their neighbor nodes of band level 2. A node $\mathbf{x}$ of band level 2 can be connected to an important number K of nodes $\mathbf{x}_k$ of band level 1. Node $\mathbf{x}$ builds its marker list among markers of nodes $\mathbf{x}_k$ by ordering all these markers in ascending order of the value of their $(n+1)$-th element components. The number of markers retained by node $\mathbf{x}$ is equal to the cardinality of the set I_n given by Eq. (5.23). The signed distance value on node $\mathbf{x}$ is then given by

$$\phi(\mathbf{x}) = \min_{1 \leq k \leq K} \min_{i \in I_n} \left\{ (M_i(x_k))_{n+1} \right\}. \tag{5.25}$$

Unfortunately, the accuracy of the signed distance function reinitialization has not been clearly enhanced using the MMM.

5.5.1 Improvement of the Multiple Marker Method

After discussions with V. Moureau, we tried to improve the method by computing the projection of the closest nodes to the interface onto the per-element segment representing the piecewise-linear interface, as shown in Fig. 5.3(a). For a node $\mathbf{x}$ of band level ± 1 and a segment Γ_h representing the piecewise-linear restriction of the interface Γ to a grid element E_h to which $\mathbf{x}$ is a summit, the orthogonal projection $\mathbf{x}_h$ of $\mathbf{x}$ on Γ_h is computed. The point $\mathbf{x}_h$ is retained only if it is located

in element E_h. If $\mathbf{x}_h \in E_h$, then the first marker of the marker list of $\mathbf{x}_h$ becomes $M_1(\mathbf{x}) = (x_{h_1}, \ldots, x_{h_n}, \|\mathbf{x} - \mathbf{x}_h\|)^{\mathrm{T}}$, and ϕ is updated on $\mathbf{x}$ by

$$\phi(\mathbf{x}) = \pm \|\mathbf{x} - \mathbf{x}_h\|. \tag{5.26}$$

Equation (5.26) gives the exact ϕ value on $\mathbf{x}$ provided that there exists a grid element E_h to which $\mathbf{x}$ is a summit and which contains the orthogonal projection $\mathbf{x}_h$ of $\mathbf{x}$ on Γ_h. The reinitialization method of ϕ on farther nodes to the interface is unchanged but such nodes benefit from these new markers by means of marker list propagation. As for the SMM, this projection step improved the accuracy of ϕ on the closest nodes to the interface, but imprecisions on the farther nodes were still significant. The interface normal vector and curvature then computed from ϕ were still highly polluted by oscillations on ϕ.

5.6 Strategy adopted to improve the reinitialization accuracy of the signed distance function to the interface

While V. Moureau's team kept on working on marker-based methods (see Section 5.9.3 for the most recent developments), we focused on classical reinitialization methods. A large study of the classical and state-of-the-art reinitialization methods for level set functions has been realized, and the classical methods have been implemented and validated. YALES2 uses the CLS since conservation is a needed feature in larger scale two-phase flow simulations, e.g. atomization, where a drop of liquid may be discretized by only a few grid nodes. Nevertheless, it has been shown in Section 5.3 that the overall implementation of the method lacks accuracy in the case of the simulations addressed in this thesis. We then decided to adopt the SDF as level set function. I implemented the corresponding classical reinitialization methods on cartesian grids, with the perspective of understanding their key points and reimplementing it on unstructured grids. I then followed the same methodology for the CLS function. On structured grids, Section 5.7 details the implementation of the signed distance function reinitialization, and Section 5.8 details the implementation of the conservative level set function reinitialization. On unstructured grids, Section 5.9 details the implementation of the signed distance function reinitialization, and Section 5.10 details the implementation of the conservative level set function reinitialization.

This study has been presented at the *Dispersed Two-Phase Flows 2018* Conference in Toulouse, France, and can be found in Sahut et al. [73].

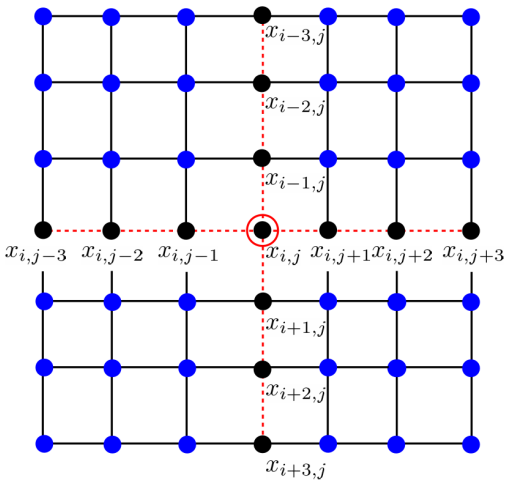
In the following sections, ϕ denotes the signed distance function and ψ, the conservative level set function. In all the implemented methods, the interface normal vector $\mathbf{n}$ and curvature κ are computed as

$$\mathbf{n} = \frac{\nabla\phi}{\|\nabla\phi\|} \quad \text{and} \quad \kappa = \frac{\nabla\phi^{\mathrm{T}} \cdot H(\phi) \cdot \nabla\phi - \|\nabla\phi\|^2 \operatorname{Tr}(H(\phi))}{\|\nabla\phi\|^3}, \tag{5.27}$$

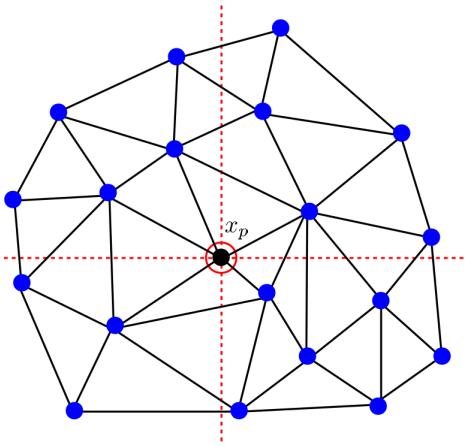
where the equation to compute κ is Eq. (3.6) from Goldman [28], emphasizing again that high precision is needed in the reinitialization of ϕ to avoid perturbations in $\mathbf{n}$ and κ.

5.7 The Signed Distance Level Set function on uniform cartesian grids

The two classical reinitialization methods for the signed distance function, namely the Hamilton-Jacobi equation and the Fast Marching Method, have been implemented for cartesian grids.



(a) Cartesian grid : the neighbor nodes of node $\mathbf{x}_{i,j}$ in the directions $\pm\mathbf{e}_x$ and $\pm\mathbf{e}_y$ are uniquely defined. For instance, the three neighbor nodes of node $\mathbf{x}_{i,j}$ in the direction $-\mathbf{e}_x$ are the nodes $\mathbf{x}_{i,j-1}$, $\mathbf{x}_{i,j-2}$ and $\mathbf{x}_{i,j-3}$.



(b) Unstructured grid : the notion of neighbor nodes of node $\mathbf{x}_p$ in the directions $\pm\mathbf{e}_x$ and $\pm\mathbf{e}_y$ is ill-defined.

Fig. 5.4 – Large orthogonal stencils are uniquely defined on cartesian grids (a) for a node $\mathbf{x}_{i,j}$ using the neighbor nodes located on the red + sign whose origin is at node $\mathbf{x}_{i,j}$, but undefined on unstructured grids (b) since neighbor nodes are not necessarily located on the + sign.

5.7.1 Reinitialization of the Signed Distance Function by the Hamilton-Jacobi equation

In the case of phase change, the signed distance function is advected by solving the standard advection equation with source term

$$\frac{\partial \phi}{\partial t} + \mathbf{u}_{\text{vap}} \cdot \nabla \phi = \frac{\dot{m}}{\rho_{\text{vap}}}, \tag{5.28}$$

where $\mathbf{u}_{\text{vap}}$ is the vapor velocity, ρ_{vap} is the vapor density and the source term is due to phase change. After advection, the function ϕ is reinitialized as a signed distance function to the interface by solving Eq. (2.7). We now describe the procedure used to solve Eq. (2.7). We use a high-order scheme to compute $\nabla\phi$ meaning that large orthogonal stencils are needed for every grid node. This requirement conflicts with the strategy of YALES2 to compute differential operators using only the direct neighbor nodes. Such operators are called compact operators. Indeed, on unstructured grids, neighbor nodes are not located on an orthogonal cross stencil centered on one node. As a consequence, neighbors in the x- and y-directions of a given node are not well-defined, as shown in Fig. 5.4. YALES2 focuses on complex, realistic geometries, so is more devoted to unstructured grids. This implies huge differences in the core of the code structure between YALES2 and classical structured codes. The latter ones usually adopt a simple method to store the grid : an array of dimension of the problem considered is declared and grid nodes are accessed by means of i, j and k indices. The main advantage of this method is the ability from one node $\mathbf{x}_{i,j,k}$ to access its neighbors

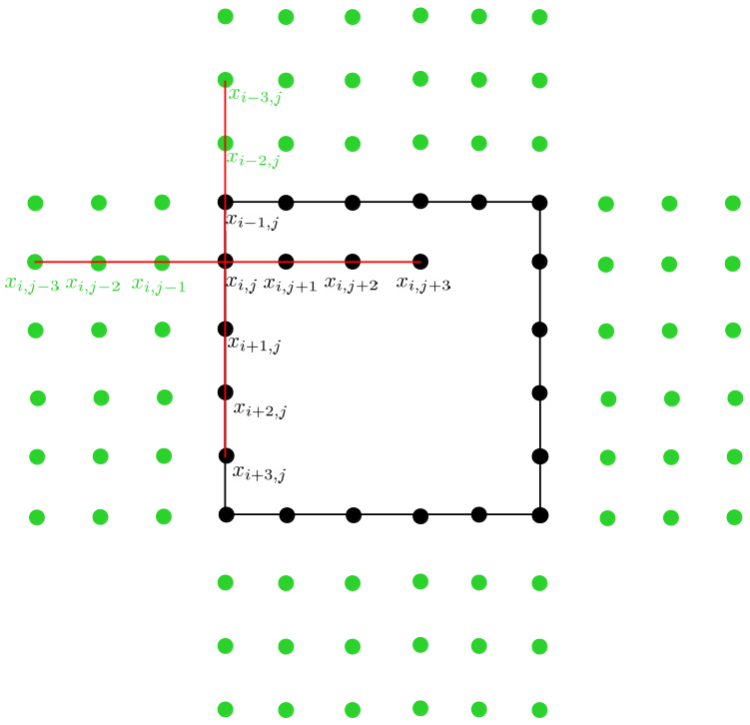


Fig. 5.5 – Addition of a ghost domain (in green) to the initial domain (in black) in order to build neighbor lists composed of three neighbors on top, right, bottom and left of all nodes located on and close to the initial domain boundaries.

with simple operations on the indices. For instance, the right neighbor in the x-direction of node $\mathbf{x}_{i,j,k}$ is accessed *via* $\mathbf{x}_{i,j+1,k}$. In order to mimic this behavior in two dimensions, we first build a large neighbor list for each node composed of three neighbors on top, right, bottom and left of the node by looping over grid edges. Let $(\mathbf{x}_p, \mathbf{x}_q)$ be one grid edge. If $(\mathbf{x}_p - \mathbf{x}_q) \cdot \mathbf{e}_x = 0$, then $(\mathbf{x}_p, \mathbf{x}_q)$ is a vertical grid edge. If $(\mathbf{x}_p)_y < (\mathbf{x}_q)_y$, then $\mathbf{x}_q$ is the neighbor of $\mathbf{x}_p$ in the y-direction of level $+1$. If $(\mathbf{x}_p)_y > (\mathbf{x}_q)_y$, then $\mathbf{x}_q$ is the neighbor of $\mathbf{x}_p$ in the y-direction of level -1. For the nodes close to the domain boundaries, we add a ghost domain to the initial domain in order to build neighbor lists for these nodes. Figure 5.5 illustrates the addition of a ghost domain. The method is iterated until all nodes have a list of three neighbors in the four cardinal directions (in two dimensions). An example of such a neighbor list is shown in Fig. 5.4(a). These neighbor lists are built only once before entering the temporal loop of the Boiling solver.

The first step of the pseudo-temporal loop is the extrapolation of ghost values on the ghost domain from the values in the physical domain. Refering to Fig. 5.5, the ghost value ϕ^{G} of ϕ on node $\mathbf{x}_{i,j-1}$ is given by

$$\phi^{\mathrm{G}}_{i,j-1} = \phi_{i,j} + H^*_h\left(\phi_{i,j}, \ell\right) h, \tag{5.29}$$

where h is the cell size, $\ell = \sqrt{2}h$ is the diagonal of one grid cell, and H^*_h is the shifted smoothed

Heaviside function defined for $x, y \in \mathbb{R}$ by

$$H_h^*(x,y) = 2H_h(x,y) - 1, \tag{5.30}$$

and H_h is the smoothed Heaviside function defined as

$$H_h(x,y) = \begin{cases} 0, & \text{if } x < -y, \\ \dfrac{1}{2}\left(1 + \dfrac{x}{h} + \dfrac{1}{\pi}\sin\left(\pi\dfrac{x}{h}\right)\right), & \text{if } |x| \leq |y|, \\ 1, & \text{if } x > y. \end{cases} \tag{5.31}$$

One can deduce from Eqs. (5.29), (5.30) and (5.31) that if the interface does not cross elements $E_{i-1/2,j+1/2}$ and $E_{i+1/2,j+1/2}$, then the ϕ ghost value on node $\mathbf{x}_{i,j-1}$ is simply given by

$$\phi_{i,j-1}^{\mathrm{G}} = \phi_{i,j} \pm h, \tag{5.32}$$

depending whether $\mathbf{x}_{i,j}$ is located on the liquid or vapor phase (we recall that $\phi > 0$ in the liquid and $\phi < 0$ in the vapor).

The second step is the computation of the Godunov flux $G(\phi_{i,j}) = \|\nabla\phi_{i,j}\| - 1$ for all nodes $\mathbf{x}_{i,j}$ of the initial domain using the Fifth-Order WENO scheme from [78] detailed in Appendix C.

The last step is the actual pseudo-temporal advancement of ϕ given by

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n} - \Delta\tau S(\phi^n) G\left(\phi_{i,j}^{n}\right). \tag{5.33}$$

Equation (5.33) is given for clarity reasons with an explicit Euler temporal integration scheme. The ^{0} superscript in Eq. (2.7) is then replaced by the ^{n} superscript. Nevertheless, Eq. (5.33) is solved in the Boiling solver with a third-order Runge-Kutta scheme.

5.7.2 Reinitialization of the Signed Distance Function by the Fast Marching Method

In order to limit the computational cost needed to perform the Fast Marching Method, we solve it only in the narrow band around the interface, large enough to be able to compute the interface normal vector and curvature by Eqs. (5.27). The implementation of the Fast Marching Method in the Boiling solver is detailed in Appendix D.

5.8 Reinitialization of the Conservative Level Set function on uniform cartesian grids

In the Boiling solver, ψ is advected using Eq. (5.21). Instead of computing $\nabla\psi$ in the rhs of Eq. (5.21) by the classical finite volume gradient operator detailed in Section 3.1.1, we reuse the idea of [13] to express $\nabla\psi$ by Eq. (2.39), replacing ϕ by ϕ_{FMM}, leading to

$$\nabla\psi(t,\mathbf{x}) = \frac{1}{4\epsilon(\mathbf{x})\cosh^2\left(\dfrac{\phi_{\mathrm{FMM}}(t,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)}\nabla\phi_{\mathrm{FMM}}(t,\mathbf{x}), \tag{5.34}$$

where $\nabla\phi_{\mathrm{FMM}}$ is computed by the finite volume gradient operator of Section 3.1.1. Normalizing $\nabla\phi_{\mathrm{FMM}}(t,\mathbf{x})$ by Eq. (2.30) to avoid numerical errors, one has

$$\nabla\psi(t,\mathbf{x}) = \frac{1}{4\epsilon(\mathbf{x})\cosh^2\left(\dfrac{\phi_{\mathrm{FMM}}(t,\mathbf{x})}{2\epsilon(\mathbf{x})}\right)}\mathbf{n}(t,\mathbf{x}). \tag{5.35}$$

5.9 The Signed Distance Level Set function on unstructured grids

In order to tackle complex geometries, the previous algorithms are now extended to unstructured grids.

5.9.1 Reinitialization of the Signed Distance Function by the Hamilton-Jacobi equation on unstructured grids

Section 5.7.1 details the reinitialization of the signed distance function using Hamilton-Jacobi equation (2.7) on cartesian grids. The fifth-order WENO scheme used heavily relies on a cartesian grid to define large stencils for each node. These large stencils are composed of one, two or three neighbors in the x- and y-directions. While particularly efficient on structured cartesian grids, cartesian WENO schemes use large stencils that are undefined on unstructured grids. Moreover, using large stencils is against the global strategy adopted in YALES2 to reconstruct differential operators using compact schemes, i.e. with only the direct neighbor nodes. Unstructured high-order schemes with a compact reconstruction of the differential operators exist [62] but are highly challenging to implement in a parallel algorithm. Since classical cartesian high-order WENO schemes are unusable on unstructured grids, another solution has been found.

In [17], the authors developed a numerical method to solve Hamilton-Jacobi Eq. (2.7) on unstructured grids in two and three dimensions. An implementation is provided by means of the MshDist open source library[1]. This method has two restrictions : 1), the grid must be composed only of simplices (triangles in two dimensions, tetrahedra in three dimensions), and 2), while it can be used with shared memory (OpenMP), the library can not be used with distributed memory (MPI). While using simplicial grids is quite acceptable, the limitation to shared memory must be overcome. YALES2 solvers are designed to launch several processes with distributed memory. If limited to shared memory, solvers can only launch one process (with memory possibly shared by several cores), and the computational time needed for large 3D simulations becomes cumbersome, not to speak about severe memory limitations. First, thanks to fruitful discussions with C. Dapogny[2], the MshDist library has been coupled to the Boiling solver in order to evaluate the accuracy of the reinitialized ϕ values in a sequential boiling simulation. Second, since the resulting ϕ function was highly accurate, the algorithm has been implemented directly in YALES2 where a solution for MPI parallelism has been derived. Appendix E details the implementation of this algorithm in YALES2 with MPI parallelism.

[1] The source code is freely available at `https://github.com/ISCDtoolbox/Mshdist`.
[2] Jean Kuntzmann Lab., Université Grenoble Alpes, France

5.9.2 Reinitialization of the Signed Distance Function by the Fast Marching Method

Section 5.7.2 details the use of the Fast Marching Method (FMM) to reinitialize ϕ on cartesian grids. An extension of the FMM to unstructured grids has recently been implemented in the MshDist library discussed in Section 5.9.1. Currently, the library proposes two methods to reinitialize ϕ : the Hamilton-Jacobi (HJ) equation (Section 5.9.1) and the FMM. The FMM proposed in MshDist has the same limitations as the HJ equation : simplices are required and, above all, the method can not be used with distributed memory (MPI). Tests have been performed using the FMM sequential implementation of MshDist (see Section 5.11 for numerical results).

5.9.3 Reinitialization of the Signed Distance Function by the Geometric Marker Method

Developed by the CORIA team, the Geometric Marker Method (GMM) is based on the Multiple Marker Method (MMM) presented in Section 5.5. Instead of computing the signed distance function to points, as in the MMM, in this new approach, one computes the signed distance function to the piecewise-linear interface in each grid element. As for the MMM, a complex mechanism is used to propagate marker lists such that farther nodes $\mathbf{x}$ to the interface can select the marker minimizing $\phi(\mathbf{x})$ among marker lists of closer nodes to the interface. This method has been implemented in two and three dimensions, and is detailed in Appendix F.

5.10 The Conservative Level Set function on unstructured grids

The reinitialization method for the Conservative Level Set [13] presented in Section 5.8 has been extended to unstructured grids. The function ψ is still reinitialized by Eq. (2.28), but the interface normal vector $\mathbf{n}$ is now computed from the signed distance function ϕ reinitialized using a method suitable to unstructured grids. In Eq. (2.30), ϕ_{FMM}, denoting the function ϕ reinitialized by the cartesian Fast Marching Method detailed in Section 5.7.2, is replaced by $\phi_{\text{HJ}_\text{u}}$, $\phi_{\text{FMM}_\text{u}}$ or ϕ_{GMM}, denoting the signed distance function ϕ reinitialized, using the previously advected ϕ values on the closest nodes to the interface as boundary conditions, respectively by the Hamilton-Jacobi equation (Section 5.9.1), the Fast Marching Method (Section 5.9.2) and the Geometric Marker Method (Section 5.9.3), where the subscript $_\text{u}$ denotes the unstructured versions of the methods (the GMM in only defined for unstructured grids since it requires simplices).

5.11 Improved numerical results

The different level set reinitialization methods presented in the previous sections are validated against the case of a 2D static growing bubble of initial radius $R_0 = 1 \times 10^{-3}\,\text{m}$ with a fixed mass transfer rate presented in Section 5.3. The methods tested are summarized in Table 5.1. The seven cases have been computed on four different grid cell sizes : $4 \times 10^{-4}\,\text{m}$, $2 \times 10^{-4}\,\text{m}$, $1 \times 10^{-4}\,\text{m}$, $5 \times 10^{-5}\,\text{m}$, where the grid cell size is approximated on unstructured grids by the meshing software algorithm. In order to evaluate the accuracy of our different methods, we first plot the relative error on the bubble radius ξ_R at final time. The theoretical bubble radius at final

Grid topology	Structured			Unstructured			
Level set type	SDF		CLS	SDF			CLS
Reinitialization methods	HJ	FMM	Chiodi & Desjardins [13]	FMM from [17]	GMM	HJ // from [17]	Chiodi & Desjardins [13] + FMM from [17]
Cases	1	2	3	4	5	6	7

Table 5.1 – The seven tested combinations : SDF stands for Signed Distance Function, CLS for Conservative Level Set, HJ for Hamilton-Jacobi, FMM for Fast Marching Method and // for parallel.

time is $R^{\text{th}} = 2 \times 10^{-3}\,\text{m}$. One has

$$\xi_R = \frac{1}{R^{\text{th}}} \max_{(\mathbf{x}_1,\mathbf{x}_2)\in\Lambda} \left| R\left(\mathbf{x}_\Gamma\right) - R^{\text{th}} \right|, \tag{5.36}$$

where Λ is the set of grid edges crossed by the interface, the max function defines the L^∞-norm of the error, $\mathbf{x}_\Gamma$ is the interface location interpolated on edge $(\mathbf{x}_1, \mathbf{x}_2) \in \Lambda$ by

$$\mathbf{x}_\Gamma = \left(1 - \theta_{\mathbf{x}_1,\mathbf{x}_2}\right) \mathbf{x}_1 + \theta_{\mathbf{x}_1,\mathbf{x}_2} \mathbf{x}_2, \tag{5.37}$$

with $\theta_{\mathbf{x}_1,\mathbf{x}_2}$ the relative distance defined in Eq. (3.24), and $R\left(\mathbf{x}_\Gamma\right)$ is given by

$$R\left(\mathbf{x}_\Gamma\right) = \left\| \mathbf{x}_\Gamma \right\|, \tag{5.38}$$

since the bubble is centered on the origin of the $(\mathbf{e}_x, \mathbf{e}_y)$ frame. We then plot the L^∞-norm of the error on the x-component of the interface normal vector ξ_N at final time. The error ξ_N is not normalized since normal vectors at the top and bottom of the bubble have an x-component equal to 0. One has

$$\xi_N = \max_{(\mathbf{x}_1,\mathbf{x}_2)\in\Lambda} \left| \left(\mathbf{n}\left(\mathbf{x}_\Gamma\right)\right)_1 - \left(\mathbf{n}^{\text{th}}\left(\mathbf{x}_\Gamma\right)\right)_1 \right|, \tag{5.39}$$

where the interface normal vector $\mathbf{n}\left(\mathbf{x}_\Gamma\right)$ is computed as

$$\mathbf{n}\left(\mathbf{x}_\Gamma\right) = \left(1 - \theta_{\mathbf{x}_1,\mathbf{x}_2}\right) \mathbf{n}\left(\mathbf{x}_1\right) + \theta_{\mathbf{x}_1,\mathbf{x}_2} \mathbf{n}\left(\mathbf{x}_2\right), \tag{5.40}$$

and the theoretical interface normal vector $\mathbf{n}^{\text{th}}\left(\mathbf{x}_\Gamma\right)$ is given by

$$\mathbf{n}^{\text{th}}\left(\mathbf{x}_\Gamma\right) = \frac{\mathbf{x}_\Gamma}{\left\| \mathbf{x}_\Gamma \right\|}. \tag{5.41}$$

Finally we plot the relative error on the interface curvature ξ_κ at final time, given by

$$\xi_\kappa = \frac{1}{\left|\kappa^{\text{th}}\right|} \max_{(\mathbf{x}_1,\mathbf{x}_2)\in\Lambda} \left| \kappa\left(\mathbf{x}_\Gamma\right) - \kappa^{\text{th}} \right|, \tag{5.42}$$

where the interface curvature $\kappa(\mathbf{x}_\Gamma)$ is given by

$$\kappa(\mathbf{x}_\Gamma) = \frac{\kappa(\mathbf{x}_1)\,\kappa(\mathbf{x}_2)}{(1-\theta_{\mathbf{x}_1,\mathbf{x}_2})\,\kappa(\mathbf{x}_2) + \theta_{\mathbf{x}_1,\mathbf{x}_2}\kappa(\mathbf{x}_1)} \tag{5.43}$$

and the theoretical interface curvature κ^{th} is given by

$$\kappa^{\text{th}} = -\frac{1}{R^{\text{th}}} = -5 \times 10^2\,\text{m}^{-1}. \tag{5.44}$$

Figure 5.6 shows the interface location at final time on the finest cartesian grid considered for Cases 1, 2 and 3 ; and Fig. 5.7 shows the relative errors at final time on the bubble radius, the x-component of the interface normal vector and the interface curvature for these cases. Figure 5.8 shows the interface location at final time on the finest unstructured grid considered for Cases 4, 5, 6 and 7 ; and Fig. 5.9 shows the relative errors at final time on the bubble radius, the x-component of the interface normal vector and the interface curvature for these cases. Figure 5.10 summarizes the relative errors on the final bubble radius, x-component of interface normal vector and interface curvature for all cases. For the final bubble radius, the seven methods have a convergence rate close to one with respect to the grid cell size (see Fig. 5.10(a)). For the finest grid, all relative errors on the bubble radius are below 1% (the error of Case 7 being equal to 1.06%). The errors on the normal vector and the curvature are shown in respectively Figs. 5.10(b) and 5.10(c). The results show that the error on the normal vector decreases at order 1 for all methods. Whereas Fig. 5.10(a) shows that Case 7, the Conservative Level Set function on unstructured grids, presents the highest relative error for the final bubble radius on the finest unstructured grid (1.06%), Fig. 5.11 shows that the final interface location is quite satisfactory even for this case on the four unstructured grids. Case 7 is the analogous of the case presented in Section 5.3 and illustrated in Fig. 5.2. One can see the improvement of our implementation in the accuracy of the interface shape comparing to the initial method implemented in YALES2. Further work is needed to improve the convergence of both normal vector and curvature for Case 7.

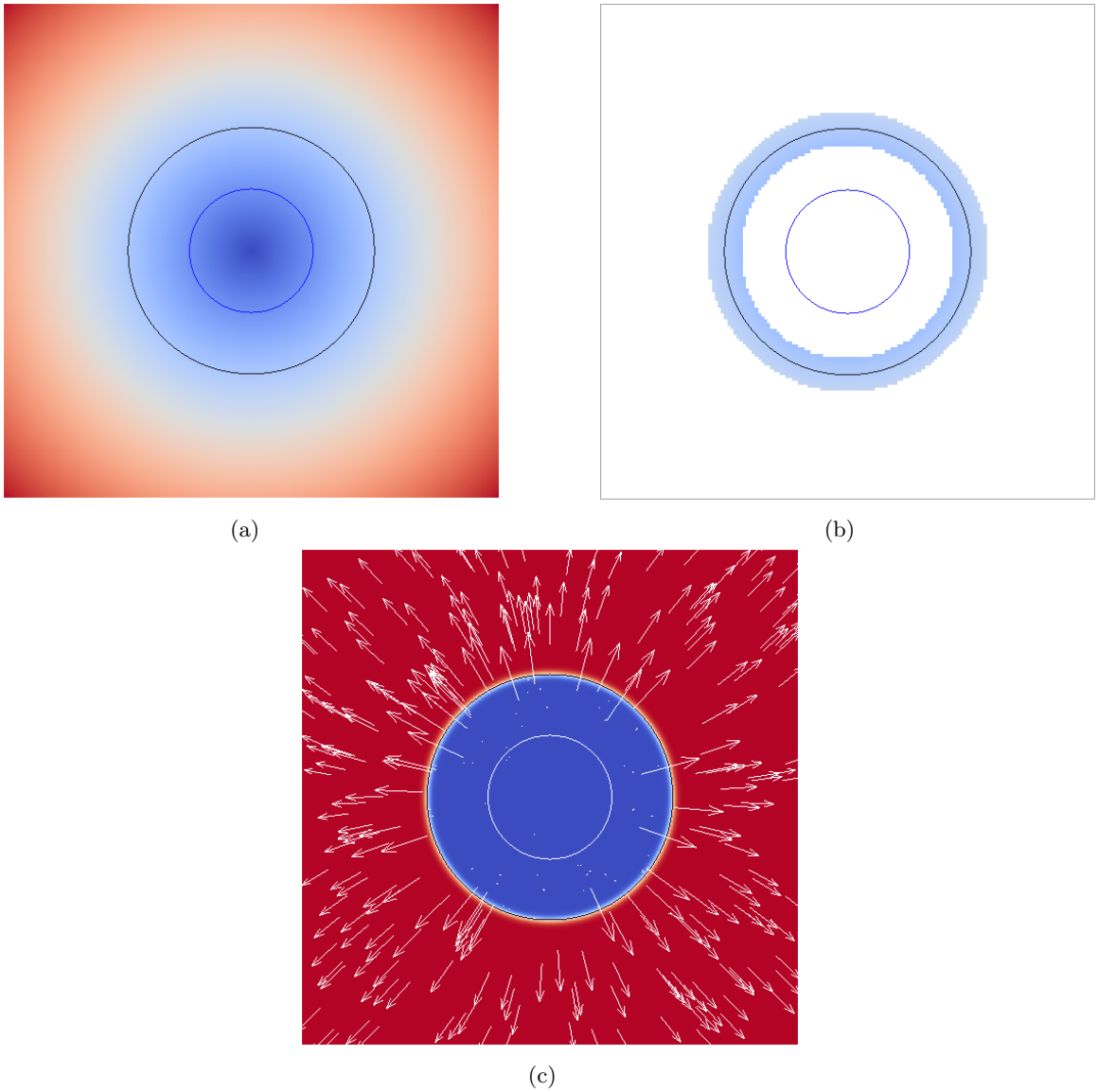
This test case has also been performed on three-dimensional unstructured grids with the GMM, as the method giving the best results for the final bubble radius on two-dimensional unstructured grids. Figure 5.12 shows the interface at final time in three dimensions and Fig. 5.13 shows the corresponding errors on the bubble radius, x-component of the interface normal vector and interface curvature.

The Boiling solver is then able to simulate phase change with high accuracy by means of an imposed mass transfer rate on two- and three-dimensional unstructured grids. Indeed, the proposed implementations of the Level Set method, the Signed Distance Function and the Conservative Level Set function, have demonstrated their ability to accurately model the interface motion and capture the interface location when the mass transfer rate is uniform and constant. In two dimensions, Fig. 5.7 shows a convergence rate of 1 for the final bubble radius and interface normal vector and a convergence rate of 0.5 for the final interface curvature on cartesian grids. Similarly, Fig. 5.9 shows a convergence rate of 1 for the final bubble radius and interface normal vector and a convergence rate between 0 and 0.5 for the final interface curvature (except for Case 7) on unstructured grids. In three dimensions, as shown in Fig. 5.13, the L^∞-norm of the error on the bubble radius at final time decreases at order one with grid refinement. Further work is needed to improve the convergence of the interface normal vector and curvature at final time. Nevertheless, we emphasize that the convergence of the final interface normal vector and curvature are challenging to obtain if the convergence rate of the bubble radius (Level Set function) is not sufficiently high.

5.12 Conclusion

In this chapter, the numerical methodology implemented in the Boiling solver has been presented in multidimensions. The lack of accuracy in the computation of the signed distance function to the interface and its impact on the interface shape have been exposed. Several numerical methods for the advection and reinitialization of the Level Set function have been presented to solve this issue. These methods have been validated and compared on the case of a static growing bubble with a fixed mass transfer rate. All implemented methods present for the final bubble radius a convergence rate of one with grid refinement. Four of the seven methods presented are designed for unstructured grids (Cases 4, 5, 6 and 7). Figure 5.14 shows the improvement in the simulation of a two-dimensional bubble growth with a fixed mass transfer rate w.r.t. the initial simulation performed using the basic implementation of the ACLS function. The Boiling solver is then able to simulate phase change with a fixed mass transfer rate. The bubble radius converges at order one with grid refinement on structured and unstructured two-dimensional grids and unstructured three-dimensional grids. The ability to accurately compute the interface location on unstructured grids opens the path to numerical simulations of liquid-vapor phase change on complex geometries.

In the next chapter, the numerical method of the Boiling solver is enhanced by the computation of the mass transfer rate, and boiling simulations on unstructured grids are presented.



(a) (b)

(c)

Fig. 5.6 – Interface location at final time on the finest structured grid considered ($\Delta x = 5 \times 10^{-5}$ m) for (a) Case 1, (b) Case 2 and (c) Case 3. The initial interface is represented in blue (in white on (c)), the computed interface at final time, in black. Our FMM implementation reinitializes ϕ only in the narrow band around the interface to save computational time. The liquid and vapor velocity fields are represented on (c).

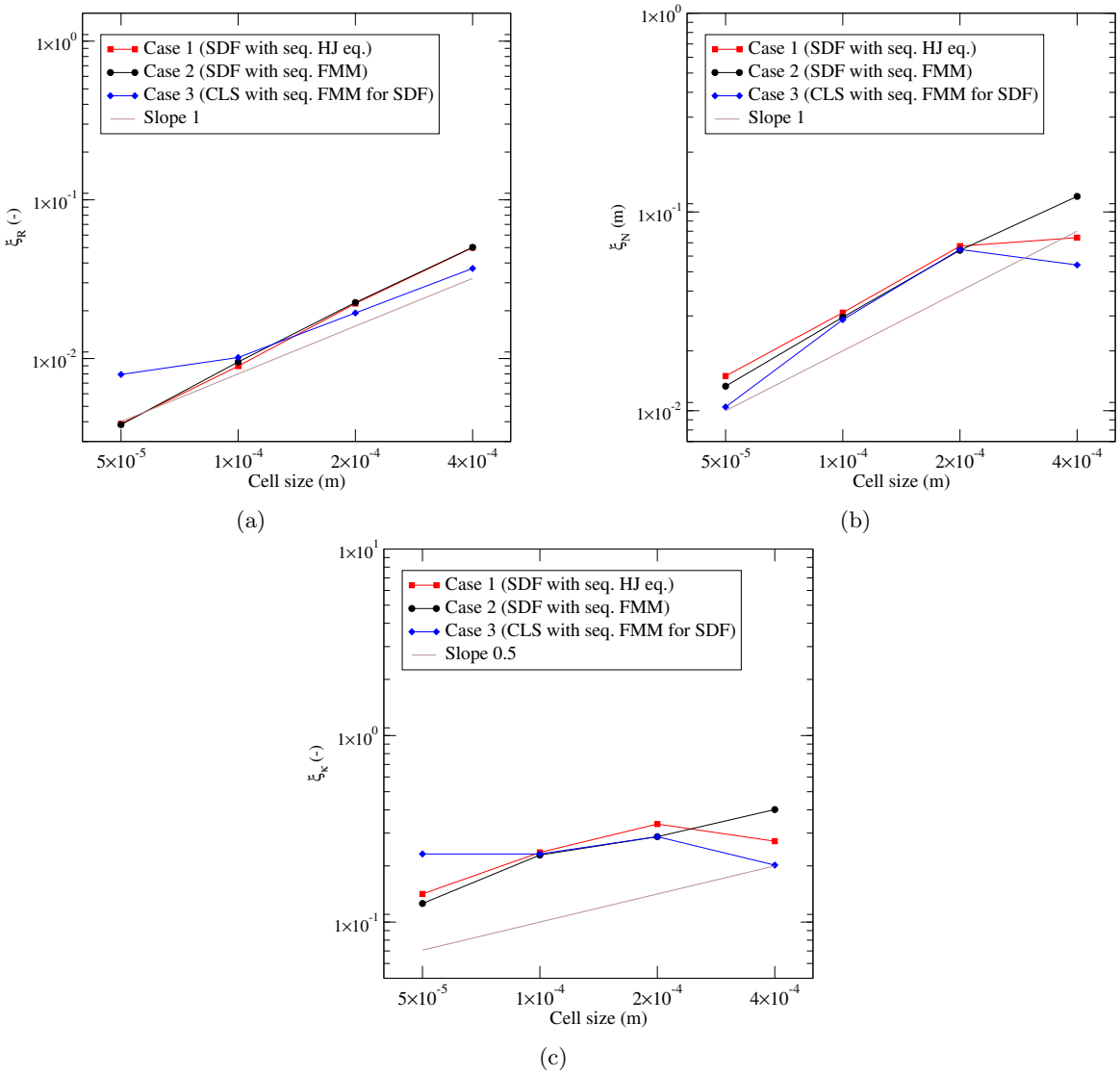
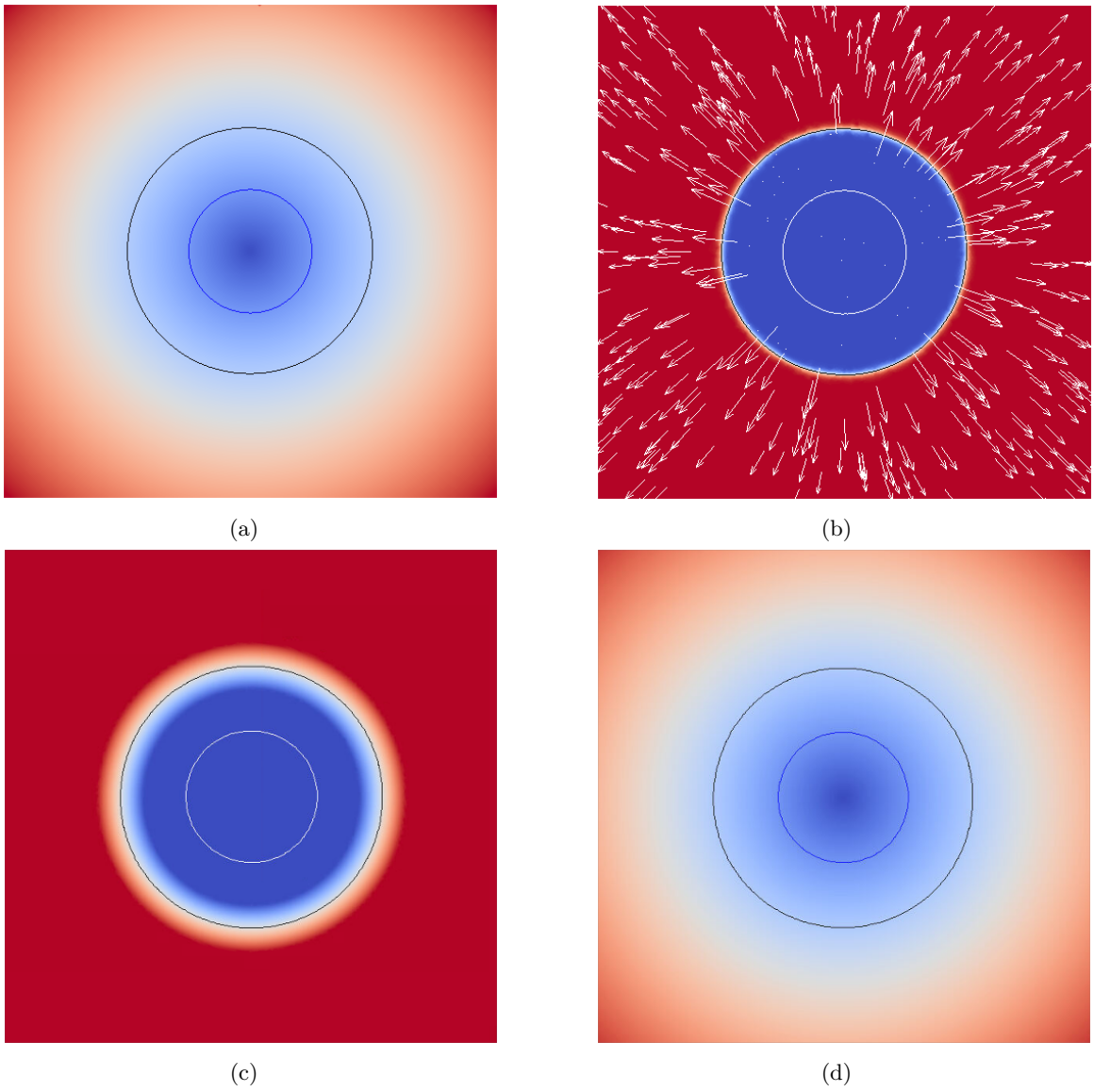


Fig. 5.7 – Errors at final time of (a) bubble radius, (b) x-component of interface normal vector and (c) interface curvature for Cases 1, 2 and 3.



(a) (b)

(c) (d)

Fig. 5.8 – Interface location at final time on the finest unstructured grid considered ($\Delta x = 5 \times 10^{-5}$ m) for (a) Case 4, (b) Case 5, (c) Case 6 and (d) Case 7. The initial interface is represented in blue on (a) and (d) and in white on (b) and (c), the computed interface at final time, in black. The liquid and vapor velocity fields are represented on (b).

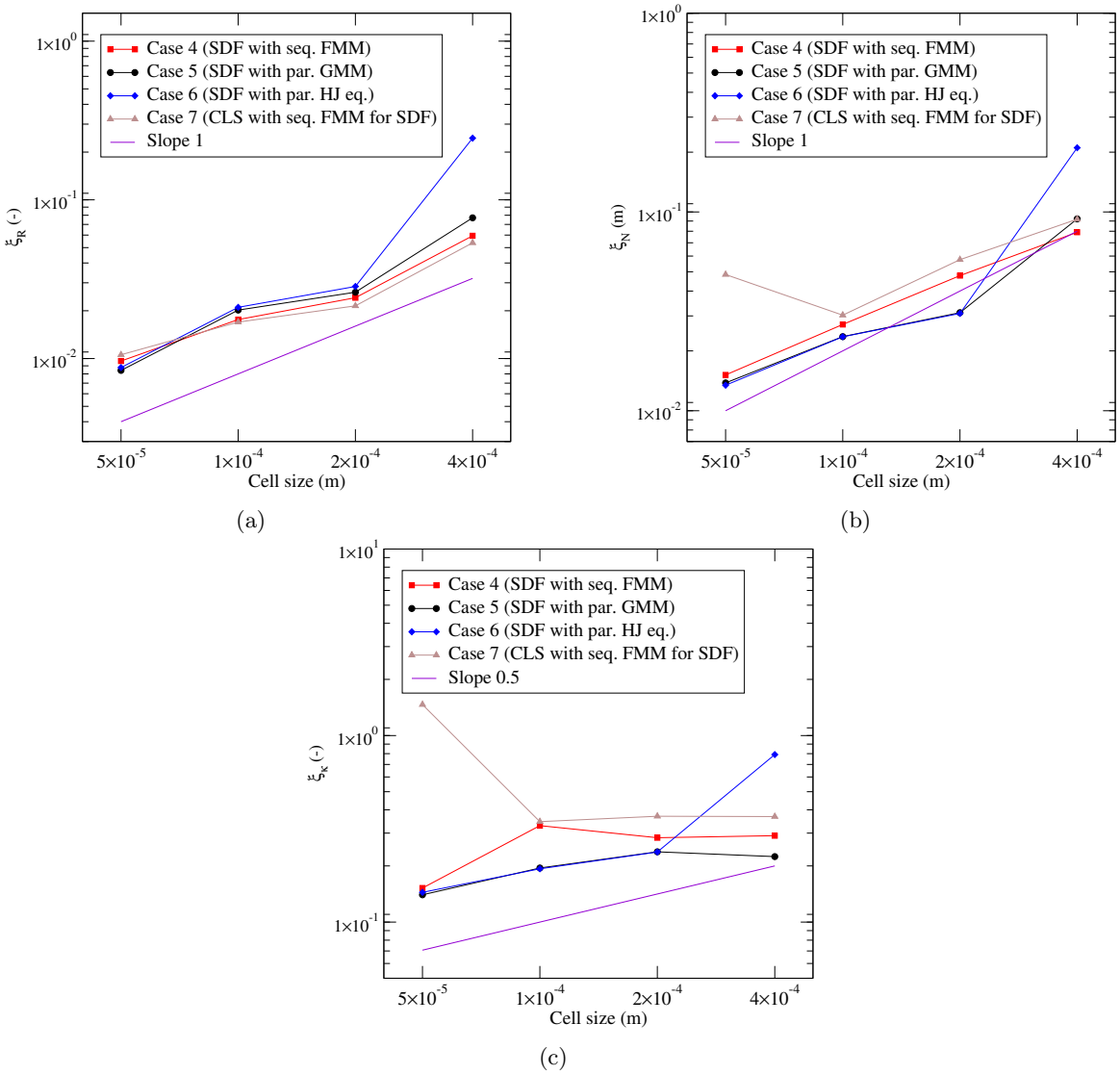


Fig. 5.9 – Errors at final time of (a) bubble radius, (b) x-component of interface normal vector and (c) interface curvature for Cases 4, 5, 6 and 7.

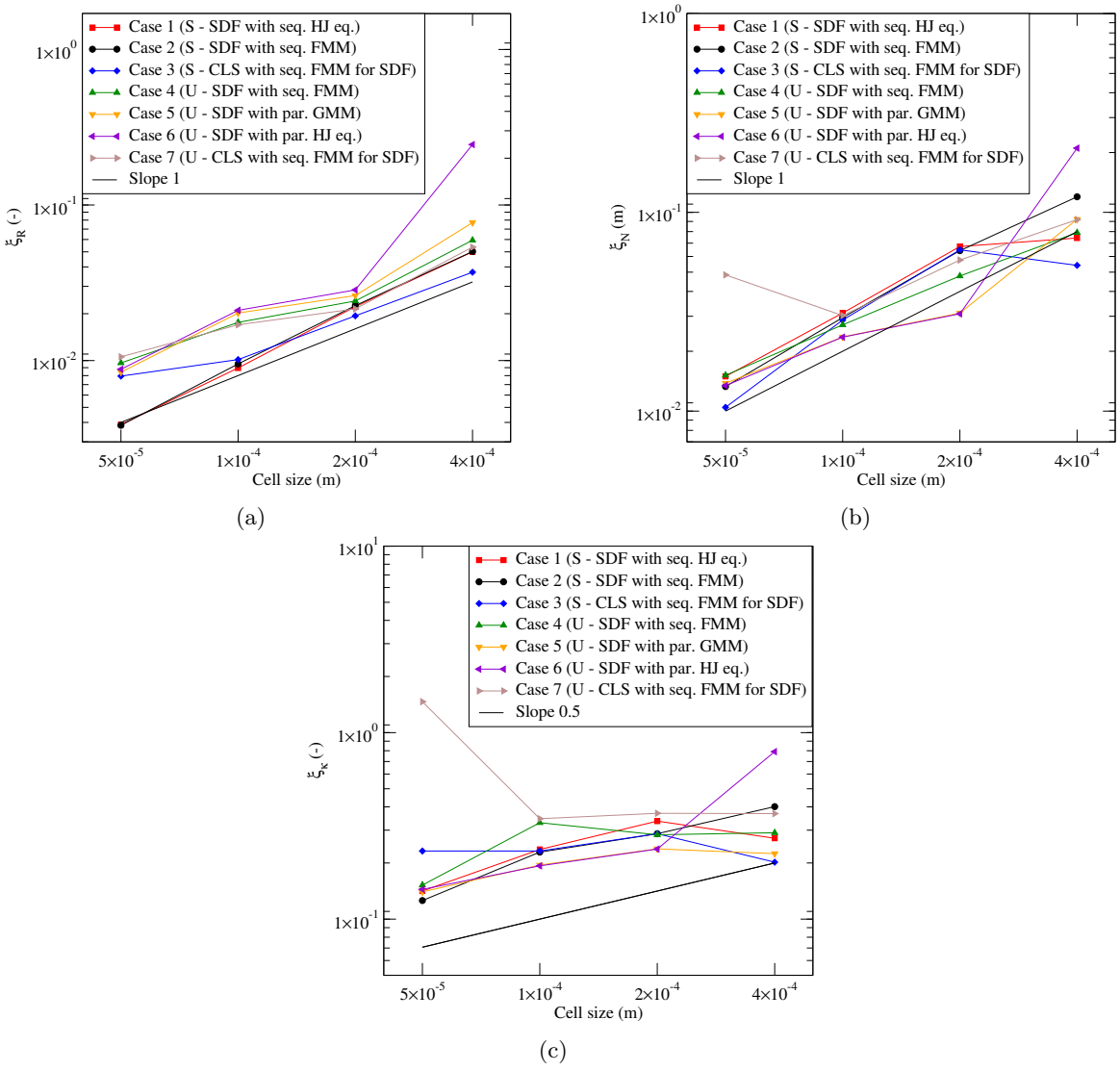
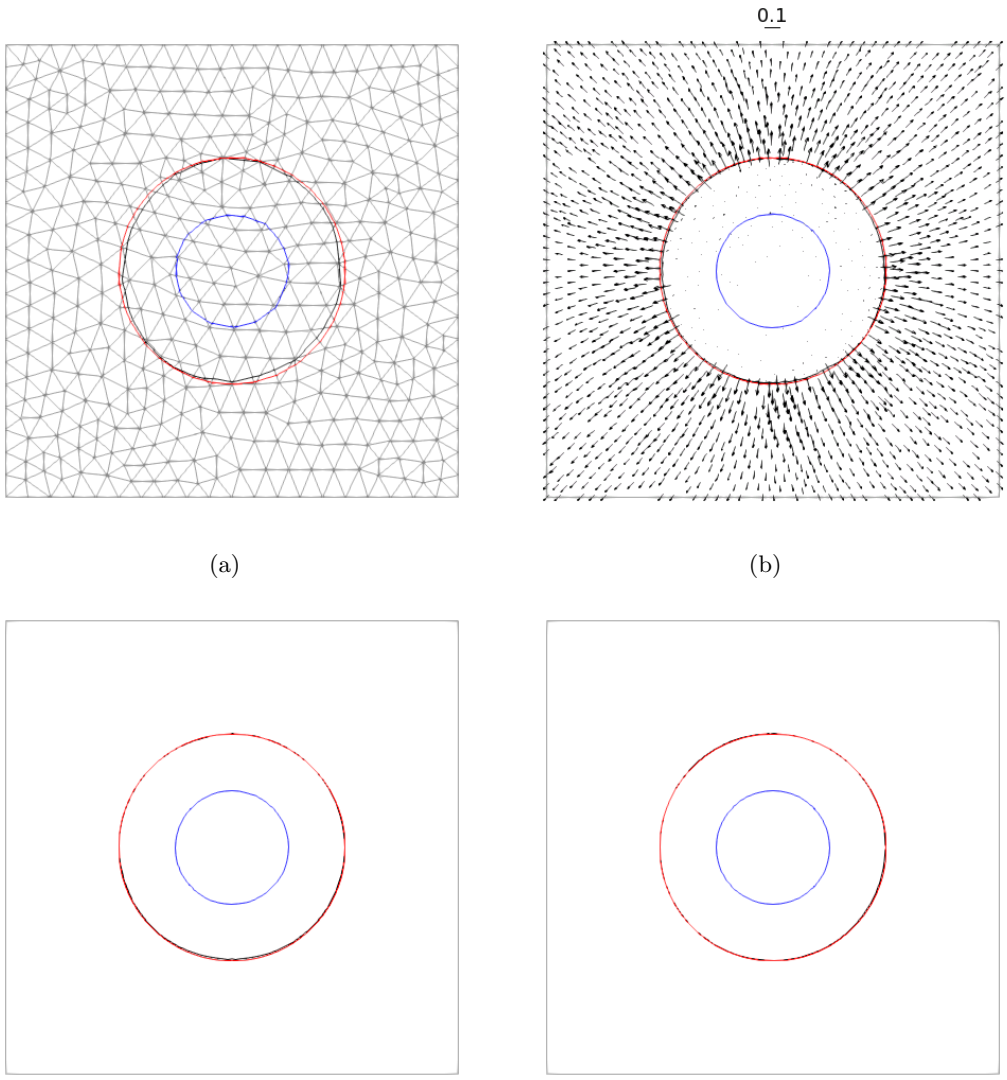


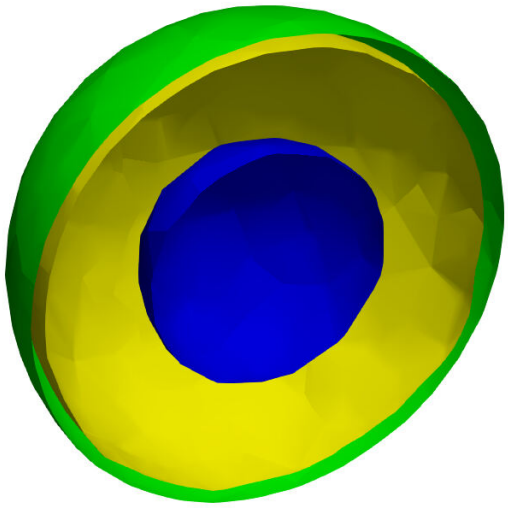
Fig. 5.10 – Errors at final time of (a) bubble radius, (b) x-component of interface normal vector and (c) interface curvature for all Cases of Tab. 5.1. The letters S and U stand respectively for Structured and Unstructured.



(a) (b)

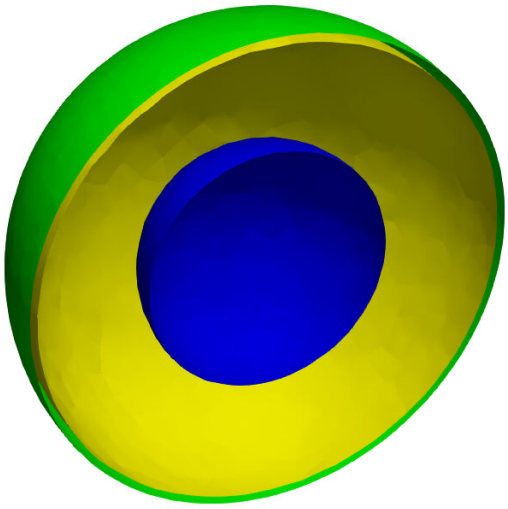
(c) (d)

Fig. 5.11 – The results for the Case 7 on four different unstructured grids are plotted at final time with a characteristic cell size of (a) 4×10^{-4} m, (b) 2×10^{-4} m, (c) 1×10^{-4} m and (d) 5×10^{-5} m. The initial interface is plotted in blue, the computed interface in black, and the theoretical interface in red. For clarity, only the coarsest grid is represented in (a). The computed liquid and vapor velocity fields are plotted in (b).

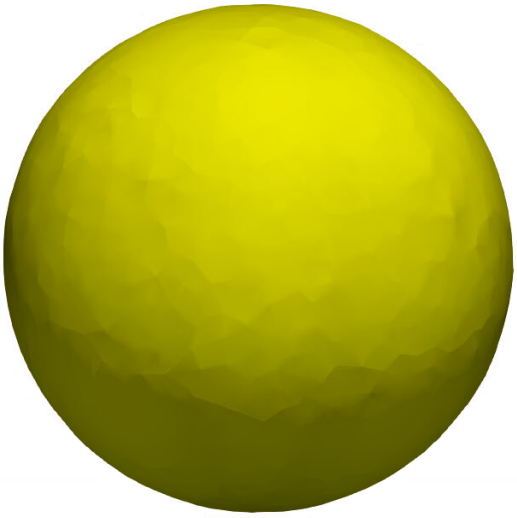


(a)

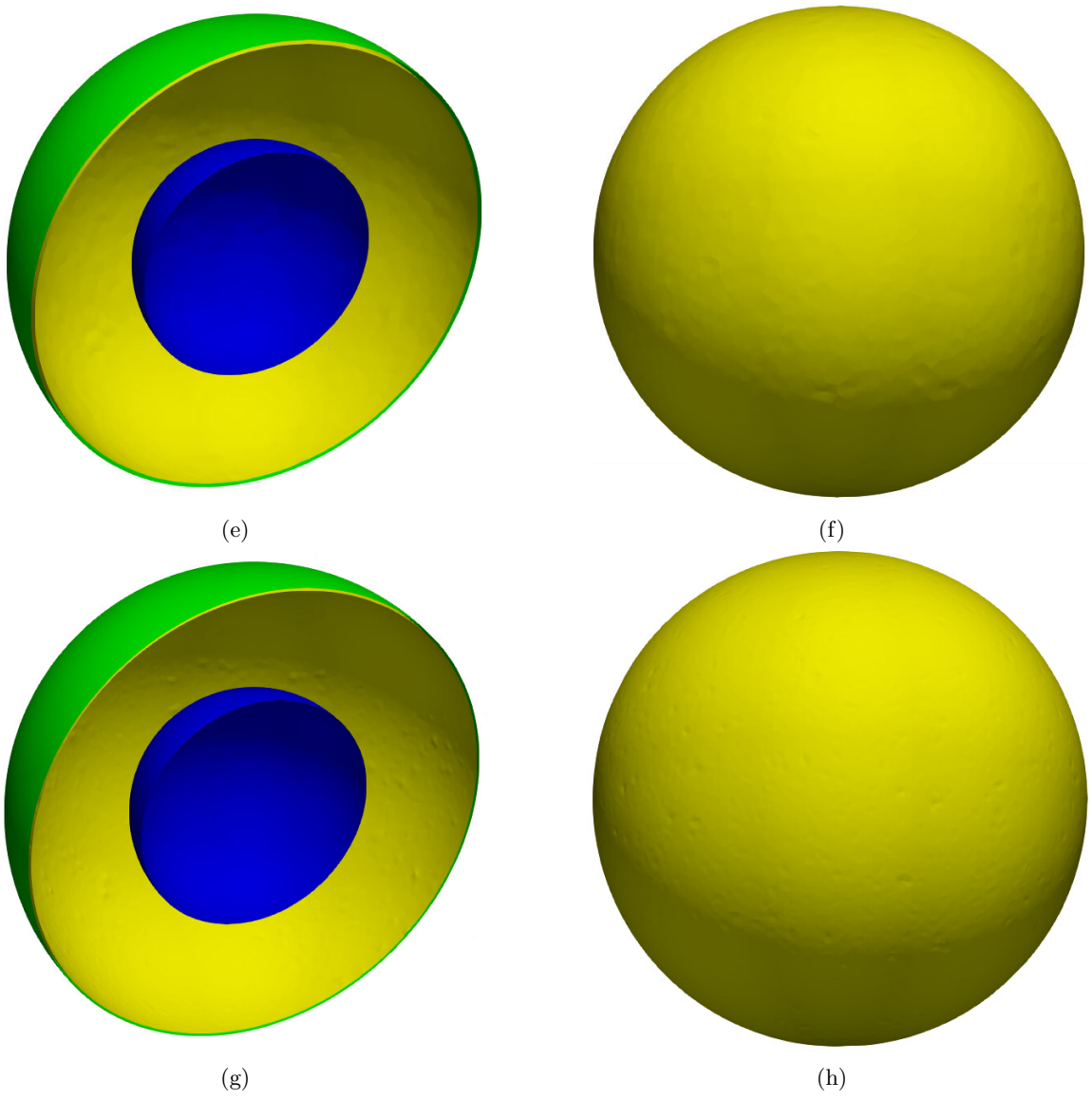
(b)



(c)



(d)



(e) (f)

(g) (h)

Fig. 5.12 – Three-dimensional extension of Case 5 to four tetrahedral unstructured grids : (a)-(b) $\Delta x = 4 \times 10^{-4}$ m, (c)-(d) $\Delta x = 2 \times 10^{-4}$ m, (e)-(f) $\Delta x = 1 \times 10^{-4}$ m and (g)-(h) $\Delta x = 5 \times 10^{-5}$ m. The initial interface is shown in blue, the theoretical and computed interfaces are shown at final time in green and yellow, respectively. We emphasize here the excellent agreement between the theoretical and numerical results.

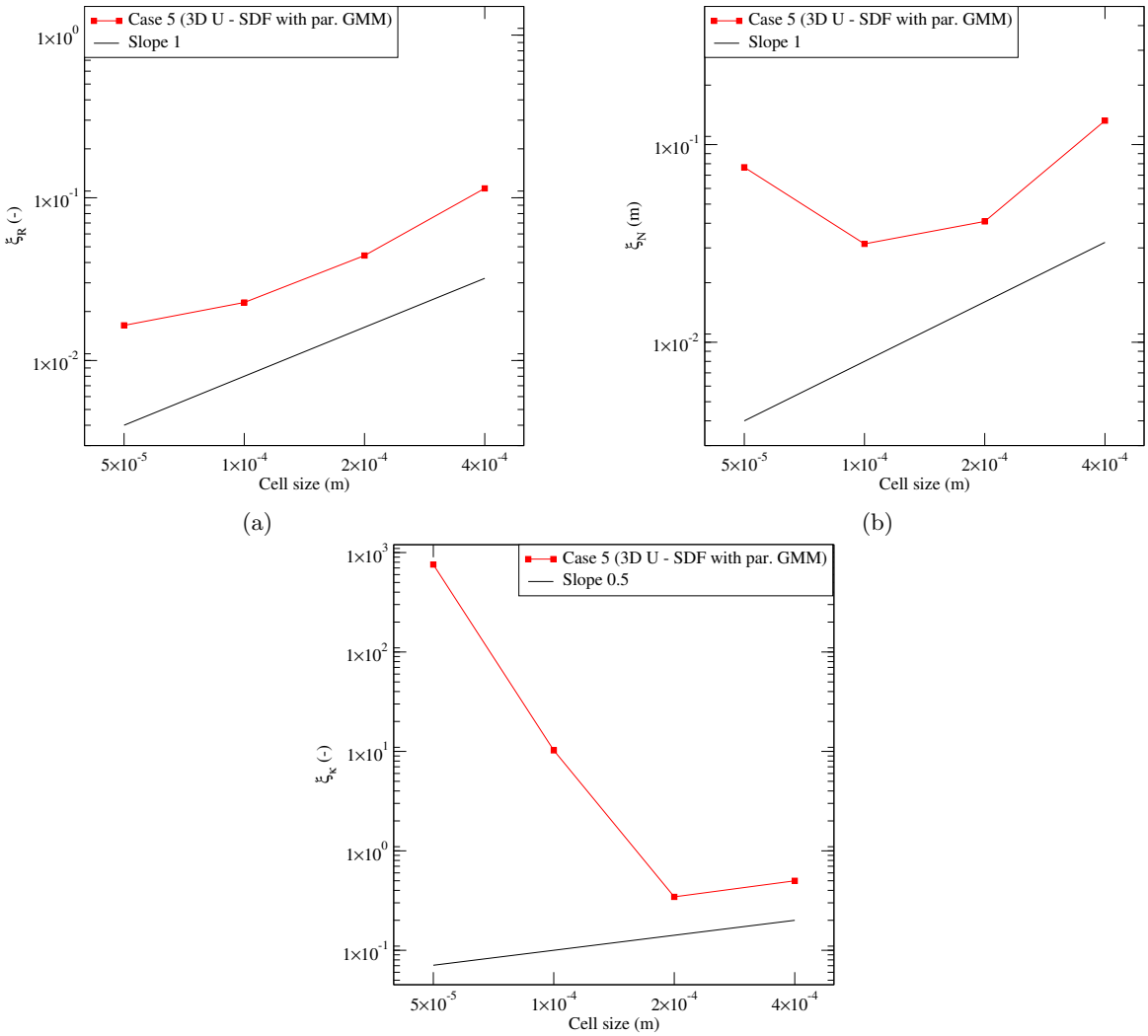
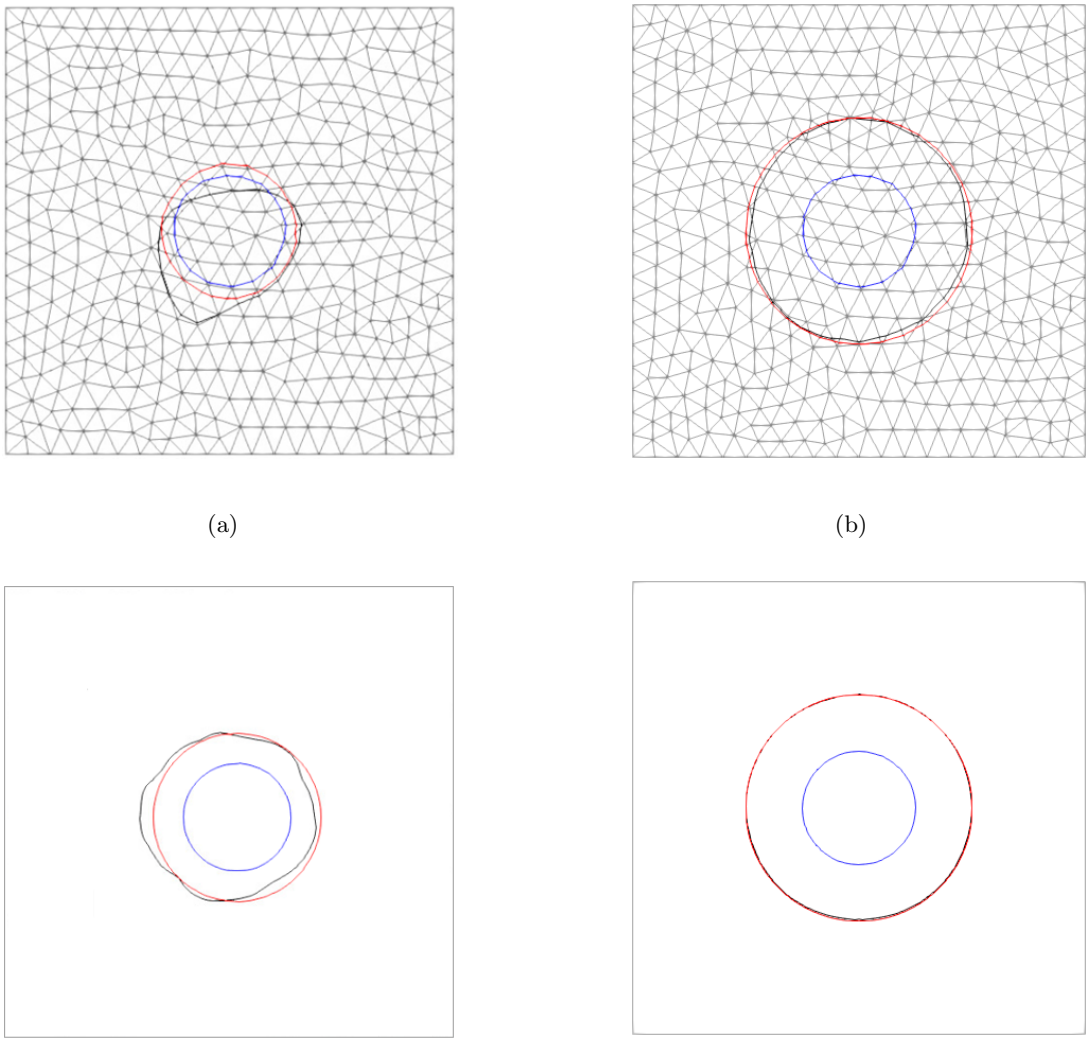


Fig. 5.13 – Error on (a) the bubble radius, (b) the x-component of the interface normal vector and (c) the interface curvature relative to the three-dimensional simulations shown in Fig. 5.12.



(a) (b)

(c) (d)

Fig. 5.14 – Comparison of the two-dimensional bubble shape growing with a fixed mass transfer rate : (a), (c) with the basic ACLS implementation of YALES2 on respectively $\Delta = 4 \times 10^{-4}$ m and $\Delta = 1 \times 10^{-4}$ m (large imprecisions on the interface location led to the failure of the simulation in both cases), reproduced from Figs. 5.2(a) and 5.2(c), and (b), (d) with our implementation denoted as Case 7 in Table 5.1, on grids of same characteristic cell sizes, reproduced from Figs. 5.11(a) and 5.11(c), where the final bubble radius is twice the initial one. The initial interface is colored in blue, the theoretical interface at current time is colored in red and the corresponding computed interface is colored in black.

Chapter 6

Numerical simulation of two-phase flows with phase change in two and three dimensions with a computed mass transfer rate

In this chapter, the resolution of the heat equation and the computation of the mass transfer rate as a function of the temperature gradients across the interface are detailed, and the whole method is validated against the test case of a static bubble, growing under the effect of the surrounding superheated liquid.

Outline

6.1 Introduction

In Chapter 5, the mass transfer rate $\dot{m}$ was imposed to a uniform and constant value. As a result, the flow dynamics was decoupled from the temperature field (the heat equation was not even solved). The Boiling solver has been validated against the case of a two- and three-dimensional static bubble, expanding due to a fixed mass tranfer rate. This chapter deals with the computation of the mass transfer rate as a function of the temperature gradients across the interface and demonstrates

the accuracy of the overall method against the case of a two- and three-dimensional static bubble surrounded by a superheated liquid responsible for phase change.

Section 6.2 presents the methodology used to solve the heat equation with immersed Dirichlet boundary condition in multidimensions. Section 6.3 details the computation of the mass transfer rate in multidimensions. Section 6.4 presents the application of a numerical method detailed in Section 6.2 to the computation of the interface curvature. Section 6.5 introduces the test case and gives the demonstration of its analytical solutions in two and three dimensions. Section 6.6 presents the numerical results and shows the accuracy of the overall numerical method implemented in the Boiling solver to simulate boiling flows on two- and three-dimensional unstructured grids. Finally, Section 6.7 concludes this chapter with a summary of the methodology used for boiling simulations in multidimenssions.

6.2 Solving the heat equation with immersed Dirichlet boundary condition in multidimensions

Under the assumption that boiling occurs at saturation temperature, the condition

$$T_\Gamma = T_{\text{sat}} \tag{6.1}$$

needs to be enforced in order to maintain the interface at saturation temperature. Since this condition is applied at the interface, which can be seen as a boundary immersed in the computational domain, i.e. the interface does not coincide with the grid nodes, this condition is said immersed Dirichlet boundary condition. Section 4.3.3 details a method to satisfy Eq. (6.1) in one dimension. In this method, Eq. (6.1) is satisfied by extrapolating ghost values across the interface using two values along the grid edge crossed by the interface : the physical (computed) temperature on one node and the (imposed) interface temperature. This scheme leads to a linearly extrapolated temperature value across the interface, ensuring that Eq. (6.1) is satisfied. The heat equation (4.10) is then solved explicitly : the temperature gradient and laplacian at time n are used to advance the temperature from time n to time $n+1$. While very simple to implement in one dimension (though attention must be paid to the threshold value used, see Section 4.3.3), this explicit method has proven inaccurate in multidimensions.

6.2.1 Temporal β-scheme

The heat equation (1.5) is solved using the temporal β-scheme [26, 85]

$$\frac{\rho_i c_{\text{p},i}}{\Delta t} T_i^{n+1} - \beta \nabla \cdot \left(\lambda_i \nabla T_i^{n+1}\right) = \frac{\rho_i c_{\text{p},i}}{\Delta t} T_i^n + (1-\beta)\, \nabla \cdot (\lambda_i \nabla T_i^n) - \rho_i c_{\text{p},i} \mathbf{u}_i \cdot \nabla T_i^n, \tag{6.2}$$

where $0 \leq \beta \leq 1$ is the implicitation factor for the diffusion term $\nabla \cdot (\lambda_i \nabla T_i^n)$ and the subscript i denotes the phase. The advection term $\rho_i c_{\text{p},i} \mathbf{u}_i \cdot \nabla T_i^n$ is included in the rhs as a source term. If β is set to 0, Eq. (6.2) defaults to the explicit scheme used in Section 4.3.3 to solve Eq. (4.10). If β is set to 0.5, Eq. (6.2) is the Crank-Nicolson scheme with source term. If β is set to 1, the diffusion term is fully implicit, and Eq. (6.2) is the semi-implicit temporal discretization scheme used in [85], where the term *semi*-implicit refers to the fact that the advection term is still computed explicitly. In order to compute close to the interface all derivatives involved in the temporal β-scheme (6.2), one needs :

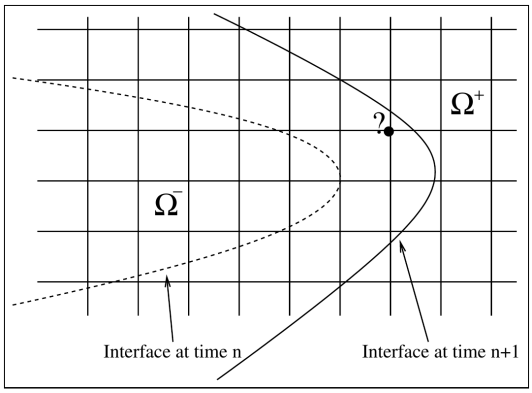


Fig. 6.1 – The node with a question mark is swept over by the interface Γ separating the subdomains Ω^- and Ω^+ between times n and $n+1$. A temperature value needs to be extrapolated at time n from Ω^- in order to advance the temperature in Ω^- with Eq. (6.2). Reproduced from [26].

1. one per-edge ghost temperature value to compute the diffusion term at time $n+1$ (if $\beta > 0$),

2. one per-edge ghost temperature value to compute the diffusion term at time n (if $\beta < 1$),

3. one per-edge ghost temperature value to compute the advection term at time n.

Besides, as stated in [26], due to the interface motion, Eq. (6.2) requires that

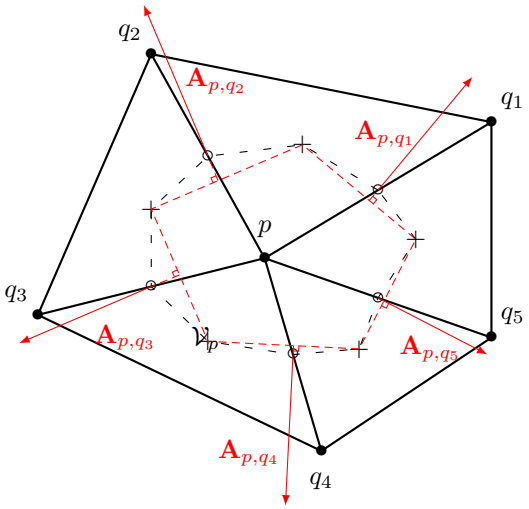
4. nodes that are swept over by the interface need to be provided with valid physical temperature values, as illustrated in Fig. 6.1.

The computation of ghost values is always a source of imprecisions since it is based on extrapolations. In the present case, extrapolations have to be computed across the interface along the direction of the interface normal vector, thus increasing the error magnitude on ghost values due to potential imprecisions on the interface normal vector computation (see Ch. 5). If β is set to 1, requirement 2 disappears. The iterative methods implemented in linear solvers used to solve Eq. (6.2) still converge when computing ghost temperature values for the diffusion term, thus satisfying requirement 1 (see Section 6.2.2). The methods used to satisfy requirements 1, 3 and 4 are discussed below.

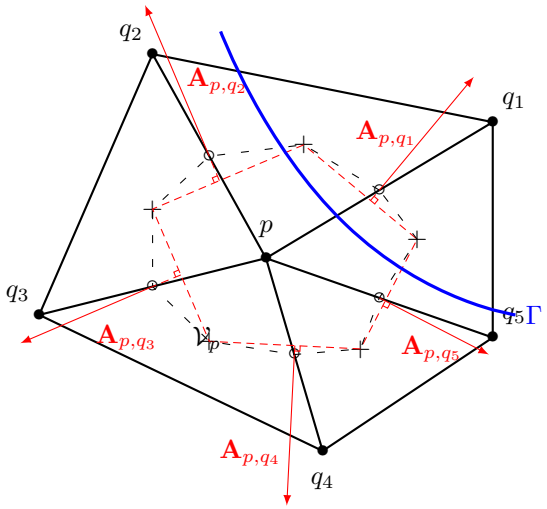
6.2.2 Discretization of the implicit diffusion term with Dirichlet boundary condition at the interface

Refering to Fig. 6.2(a), the diffusion term at time $n+1$ of Eq. (6.2) is given at node $\mathbf{p}$ by Eq. (3.16), i.e.

$$\nabla \cdot (\lambda \nabla T)|_{\mathbf{p}} = \frac{1}{\mathcal{V}_{\mathbf{p}}} \sum_{j=1}^{5} \lambda_{\mathbf{p},\mathbf{q}_j} \, \nabla T|_{\mathbf{p},\mathbf{q}_j} \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_j}, \tag{6.3}$$



(a) Control volume without liquid-vapor interface.



(b) Control volume crossed by the liquid-vapor interface : in order to compute the implicit diffusion term of Eq. (6.2) on node $\mathbf{p}$, a temperature ghost value is computed on node $\mathbf{q}_1$ by Eq. (6.8).

Fig. 6.2 – The presence of the liquid-vapor interface is taken into account in the implicit diffusion term of the lhs of Eq. (6.2). Without interface inside the control volume (a), the implicit diffusion term of Eq. (6.2) is computed by the usual finite volume procedure. When the interface crosses the control volume (b), temperature ghost values are computed by means of Eq. (6.8).

where $\lambda_{\mathbf{p},\mathbf{q}_j} = \left(\lambda_{\mathbf{p}} + \lambda_{\mathbf{q}_j}\right)/2$ and the gradient $\nabla T|_{\mathbf{p},\mathbf{q}_j}$ along the edge $(\mathbf{p},\mathbf{q}_j)$ is computed by

$$\begin{aligned}\nabla T|_{\mathbf{p},\mathbf{q}_j} &= \frac{T_{\mathbf{q}_j} - T_{\mathbf{p}}}{\|\mathbf{q}_j - \mathbf{p}\|}\frac{\mathbf{q}_j - \mathbf{p}}{\|\mathbf{q}_j - \mathbf{p}\|} \\ &= \frac{T_{\mathbf{q}_j} - T_{\mathbf{p}}}{\|\mathbf{q}_j - \mathbf{p}\|^2}\left(\mathbf{q}_j - \mathbf{p}\right). \end{aligned} \tag{6.4}$$

Equations (6.3) and (6.4) form a compact scheme in the sense that, while the laplacian of T at node $\mathbf{p}$ involves second-order derivatives of T, only the values on the closest neighbor nodes $\mathbf{q}_j$ (and on $\mathbf{p}$) are needed.

Note that for a one-dimensional uniform grid of cell size Δx where the indices $i-1$, i and $i+1$ represent nodes $\mathbf{q}_1$, $\mathbf{p}$ and $\mathbf{q}_2$ respectively, one has $\|\mathbf{q}_j - \mathbf{p}\| = \Delta x$, $(\mathbf{q}_j - \mathbf{p})/\|\mathbf{q}_j - \mathbf{p}\| = \pm\mathbf{e}_x$ and $\mathbf{A}_{\mathbf{p},\mathbf{q}_j} = \mathbf{e}_x$. As a result, Eq. (6.4) simplifies to

$$\nabla T|_{\mathbf{x}_{i-1},\mathbf{x}_i} = \frac{T_i - T_{i-1}}{\Delta x}\mathbf{e}_x \tag{6.5}$$

and

$$\nabla T|_{\mathbf{x}_i,\mathbf{x}_{i+1}} = \frac{T_{i+1} - T_i}{\Delta x}\mathbf{e}_x, \tag{6.6}$$

and Eq. (6.3) writes

$$\nabla \cdot (\lambda \nabla T)|_i = \lambda \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}, \tag{6.7}$$

(assuming uniformity of the thermal conductivity λ). Equations (6.6) and (6.7) are respectively the first- and second-order accurate finite difference spatial discretization schemes used for the gradient and laplacian operators in one dimension.

When the interface crosses the control volume, as illustrated in Fig. 6.2(b), temperature ghost values need be computed to avoid using potentially irrelevant temperature values from the other phase, and to ensure saturation temperature at the interface. For instance, in Fig. 6.2(b), when computing the implicit temperature diffusion term of Eq. (6.2) on node $\mathbf{p}$, one needs to compute a temperature ghost value $T^{\mathrm{G}}_{\mathbf{q}_1}$ on node $\mathbf{q}_1$ since this node is not located in the same phase than node $\mathbf{p}$. In multidimensions, Eq. (4.24) applied to edge $(\mathbf{p}, \mathbf{q}_1)$ writes

$$T^{\mathrm{G}}_{\mathbf{q}_1} = T_{\mathbf{p}} + \frac{T_{\mathrm{sat}} - T_{\mathbf{p}}}{\theta_{\mathbf{p},\mathbf{q}_1}}, \tag{6.8}$$

and, by Eq. (6.4), the temperature gradient $\nabla T|_{\mathbf{p},\mathbf{q}_1}$ along grid edge $(\mathbf{p}, \mathbf{q}_1)$ is given by

$$\begin{aligned} \nabla T|_{\mathbf{p},\mathbf{q}_1} &= \frac{T^{\mathrm{G}}_{\mathbf{q}_1} - T_{\mathbf{p}}}{\|\mathbf{q}_1 - \mathbf{p}\|^2} (\mathbf{q}_1 - \mathbf{p}) \\ &= \frac{T_{\mathrm{sat}} - T_{\mathbf{p}}}{\theta_{\mathbf{p},\mathbf{q}_1} \|\mathbf{q}_1 - \mathbf{p}\|^2} (\mathbf{q}_1 - \mathbf{p}). \end{aligned} \tag{6.9}$$

The implicit temperature diffusion term is then computed at node $\mathbf{p}$ by Eqs. (6.3), (6.4) and (6.9), yielding

$$\begin{aligned} \nabla \cdot (\lambda \nabla T)|_{\mathbf{p}} &= \frac{1}{\mathcal{V}_{\mathbf{p}}} \lambda_{\mathbf{p},\mathbf{q}_1} \nabla T|_{\mathbf{p},\mathbf{q}_1} \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_1} + \frac{1}{\mathcal{V}_{\mathbf{p}}} \sum_{j=2}^{5} \lambda_{\mathbf{p},\mathbf{q}_j} \nabla T|_{\mathbf{p},\mathbf{q}_j} \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_j} \\ &= \frac{1}{\mathcal{V}_{\mathbf{p}}} \lambda_{\mathbf{p},\mathbf{q}_1} \frac{T_{\mathrm{sat}} - T_{\mathbf{p}}}{\theta_{\mathbf{p},\mathbf{q}_1} \|\mathbf{q}_1 - \mathbf{p}\|^2} (\mathbf{q}_1 - \mathbf{p}) \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_1} + \frac{1}{\mathcal{V}_{\mathbf{p}}} \sum_{j=2}^{5} \lambda_{\mathbf{p},\mathbf{q}_j} \frac{T_{\mathbf{q}_j} - T_{\mathbf{p}}}{\|\mathbf{q}_j - \mathbf{p}\|^2} (\mathbf{q}_j - \mathbf{p}) \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_j} \end{aligned} \tag{6.10}$$

In one dimension, considering nodes $\mathbf{x}_{i-1}$, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, and assuming that the interface is located between nodes $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, Eq. (6.8) becomes

$$T^{\mathrm{G}}_{i+1} = T_i + \frac{T_{\mathrm{sat}} - T_i}{\theta_{\mathbf{x}_i,\mathbf{x}_{i+1}}}, \tag{6.11}$$

and Eq. (6.9) simplifies to

$$\nabla T|_{\mathbf{x}_i,\mathbf{x}_{i+1}} = \frac{T_{\mathrm{sat}} - T_{\mathbf{i}}}{\theta_{\mathbf{x}_i,\mathbf{x}_{i+1}} \Delta x} \mathbf{e}_x. \tag{6.12}$$

Finally, assuming again uniformity of the thermal conductivity λ in each phase, Eq. (6.10) is rewritten

$$\begin{aligned} \nabla \cdot (\lambda \nabla T)|_i &= \frac{1}{\Delta x} \left(\lambda \frac{T_{\mathrm{sat}} - T_i}{\theta_{\mathbf{x}_i,\mathbf{x}_{i+1}} \Delta x} - \lambda \frac{T_i - T_{i-1}}{\Delta x} \right) \\ &= \lambda \frac{\theta_{\mathbf{x}_i,\mathbf{x}_{i+1}} T_{i-1} - \left(1 + \theta_{\mathbf{x}_i,\mathbf{x}_{i+1}}\right) T_i}{\theta_{\mathbf{x}_i,\mathbf{x}_{i+1}} \Delta x^2} + \lambda \frac{T_{\mathrm{sat}}}{\theta_{\mathbf{x}_i,\mathbf{x}_{i+1}} \Delta x^2}. \end{aligned} \tag{6.13}$$

6.2.3 Discretization of the explicit advection term

6.2.3.1 Interface temperature

The advection term $-\rho c_{\mathrm{p}} \mathbf{u} \cdot \nabla T^n$ is solved explicitly, as noted by the n superscript. As a previously advected field, we do not reimpose the saturation temperature at the interface on T^n when computing ∇T in the advection term, avoiding the abrupt perturbation of the field associated with such linear extrapolation in explicit formalism. Indeed, while using Eq. (6.8) within an implicit formalism (e.g. computation of ghost values in the implicit temperature diffusion of Eq. (6.2)) does not degrade the field, non negligeable perturbations of the temperature fields have been observed when using Eq. (6.8) within an explicit resolution of Eq. (6.2). Moreover, while one could think that reimposing the saturation temperature at the interface could improve accuracy of the method (since boiling is assumed to occur at saturation temperature), it may theoretically be not as crucial since the temperature has already been advected, with the saturation temperature imposed at the interface in the previous computation of the implicit diffusion term (see Section 6.2.2). The interface temperature at time n is then expected to be very close to the saturation temperature.

6.2.3.2 Temperature ghost values close to the interface

When computed on the closest nodes to the interface, the temperature gradient of the explicit advection term also requires valid temperature values on the other side of the interface. This requirement is typically fulfilled by populating ghost nodes close to the interface by ghost values. These ghost values are generally computed by means of extrapolations from the physical values defined in one phase to the other side of the interface. In [85], the authors used the high-order extrapolation method from [4] on two-dimensional axisymmetric cartesian grids. This method has been implemented in the Boiling solver up to order two for arbitrary two- and three-dimensional grids. Nevertheless, the accuracy of the extrapolated temperature values was not sufficient to compute an accurate temperature gradient close to the interface. The method is detailed in Appendix G, where some numerical results are shown.

During this thesis, another methodology to reconstruct differential operators at high order on unstructured simplicial grids has been developed within a collaboration between LEGI and CORIA teams. This new method also enables high-order interpolations and extrapolations, and is presented in the next section.

6.2.4 Differential operator reconstructions and extrapolations : the High-Order Framework

6.2.4.1 Numerical error inherent to the Finite Volume Method

In [9], the authors proposed a new framework to reconstruct differential operators up to a chosen accuracy order within the node-centered finite-volume method on unstructured simplicial grids, in two and three dimensions. This method has been implemented in the Boiling solver and is detailed below.

Let Ω be the computational domain, let $\mathcal{S}$ be one non-moving unstructured simplicial partition of Ω and let $\mathcal{X}$ be the set of nodes delimiting $\mathcal{S}$. Let $Z : \Omega \longrightarrow \mathbb{R}$ be a scalar field. For each node $\mathbf{x}_I \in \mathcal{X}$, let $\Omega_I \subset \Omega$ be the control volume associated to node $\mathbf{x}_I$, and Z_I the value of Z on node $\mathbf{x}_I$. Let assume that the field Z is advected during the simulation by some velocity $\mathbf{u}$ according to

the conservative advection equation

$$\frac{\partial Z}{\partial t} + \nabla \cdot (Z\mathbf{u}) = 0. \tag{6.14}$$

In the rest of this section, let $\mathbf{x}_I$ be one given node of $\mathcal{X}$. At any given time of the simulation, the updated value of Z obtained by solving Eq. (6.14) and stored at node $\mathbf{x}_I$ is not equal to the theoretical new value $Z_I = Z(\mathbf{x}_I)$. This difference does not entirely originate from the numerical error inherent to the advection scheme used to solve Eq. (6.14). Indeed, as will be shown in this section, it is mainly due to the foundations of the finite volume method itself. In order to minimize this difference, the following methodology is applied.

By construction of the finite volume method, the value of Z stored at node $\mathbf{x}_I$ is the average value of Z over Ω_I. As a result, this value will be noted $\overline{Z}^{\Omega_I}$ where the averaging operator $\overline{\cdot}^{\Omega_I}$ applied to Z is given by

$$\overline{Z}^{\Omega_I} = \frac{1}{V_I} \int_{\Omega_I} Z(\omega)\, \mathrm{d}\omega, \tag{6.15}$$

where V_I is the volume of the set Ω_I. Integrating Eq. (6.14) over Ω_I and using Eqs. (6.15) and (3.2) yields

$$\frac{\partial \overline{Z}^{\Omega_I}}{\partial t} + \frac{1}{V_I} \int_{\partial\Omega_I} Z(\sigma)\, \mathbf{u}(\sigma) \cdot \mathbf{n}(\sigma)\, \mathrm{d}\sigma = 0. \tag{6.16}$$

Equation (6.16) is the equation which is actually solved in the finite volume method. The value $\overline{Z}^{\Omega_I}$ is expected to be *close* to $Z_I = Z(\mathbf{x}_I)$. Particularly, $\overline{Z}^{\Omega_I}$ is a second-order accurate approximation of Z_I on regular grids and a first-order accurate approximation of Z_I on unstructured grids. Nevertheless, when high precision is needed on Z (especially to compute its derivatives), this approximation is not sufficient and more work is needed to recover Z_I from $\overline{Z}^{\Omega_I}$. This step is called *deconvolution*. In [9], the authors propose a third-order polynomial reconstruction of Z_I from $\overline{Z}^{\Omega_I}$, together with a second-order polynomial reconstruction of ∇Z_I and a first-order polynomial reconstruction of $\nabla\nabla Z_I$.

6.2.4.2 Computation of control volume first- and second-order moments

Let $\mathbf{x} \in \Omega_I$, not necessarily a node of $\mathcal{X}$, the Taylor expansion of Z at order 2 in point $\mathbf{x}$ w.r.t. node $\mathbf{x}_I$ is given by

$$Z(\mathbf{x}) = Z_I + \nabla Z_I \cdot \boldsymbol{\Delta}_I(\mathbf{x}) + \frac{1}{2} (\nabla\nabla Z_I) : (\boldsymbol{\Delta}_I(\mathbf{x}) \otimes \boldsymbol{\Delta}_I(\mathbf{x})) + \mathcal{O}\left(\Delta^3\right), \tag{6.17}$$

where $\boldsymbol{\Delta}_I = \mathbf{x} - \mathbf{x}_I$, Δ is the characteristic length of grid cells, $\cdot$ denotes the simple dot product, $:$, the double dot product and $\otimes$, the tensor product. Applying the averaging operator of Eq. (6.15) to Eq. (6.17) yields

$$\overline{Z}^{\Omega_I} = Z_I + \nabla Z_I \cdot \overline{\boldsymbol{\Delta}_I}^{\Omega_I} + \frac{1}{2} (\nabla\nabla Z_I) : \left(\overline{\boldsymbol{\Delta}_I \otimes \boldsymbol{\Delta}_I}^{\Omega_I}\right) + \mathcal{O}\left(\Delta^3\right), \tag{6.18}$$

where

$$\overline{\boldsymbol{\Delta}_I}^{\Omega_I} = \frac{1}{V_I} \int_{\Omega_I} (\mathbf{x}(\omega) - \mathbf{x}_I)\, \mathrm{d}\omega = \mathcal{O}(\Delta) \tag{6.19}$$

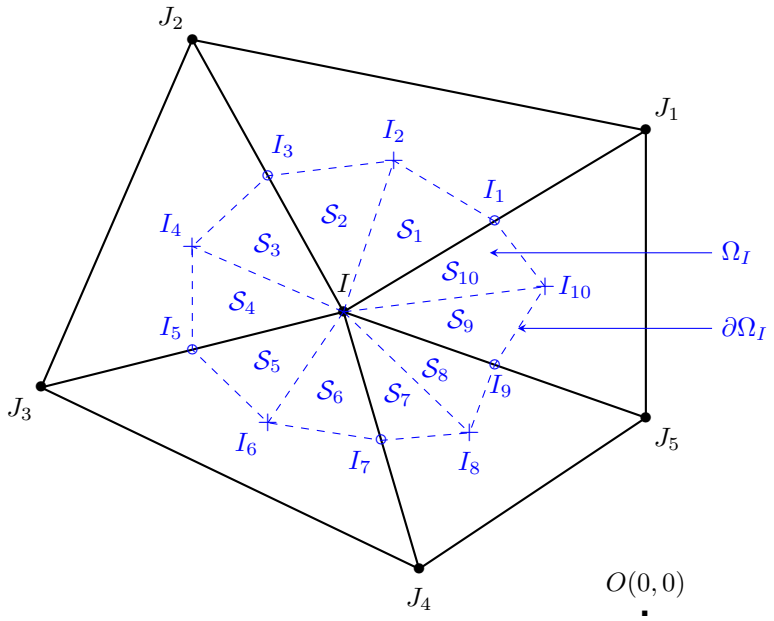


Fig. 6.3 – The control volume Ω_I associated to node $\mathbf{x}_I$ represented by point I is the disjoint union of simplices $\mathcal{S}_1, \ldots, \mathcal{S}_{10}$. The symbol $\bullet$ denotes the grid nodes I and $J_1, \ldots, J_5$, $+$, the grid element barycenters I_2, I_4, I_6, I_8, I_{10}, $\circ$, the grid edge midpoints I_3, I_5, I_7, I_9, and $\blacksquare$, the point O (frame origin) which does not necessarily represent a grid node.

and

$$\overline{\boldsymbol{\Delta}_I \otimes \boldsymbol{\Delta}_I}^{\Omega_I} = \frac{1}{V_I} \int_{\Omega_I} (\mathbf{x}(\omega) - \mathbf{x}_I) \otimes (\mathbf{x}(\omega) - \mathbf{x}_I) \, \mathrm{d}\omega = \mathcal{O}\left(\Delta^2\right) \tag{6.20}$$

are respectively the first- and second-order moments of Ω_I. One can remark that $\overline{\boldsymbol{\Delta}_I}^{\Omega_I} = \mathbf{x}_{\Omega_I} - \mathbf{x}_I$, where $\mathbf{x}_{\Omega_I}$ is the mass center of Ω_I. As a result, $\overline{\boldsymbol{\Delta}_I}^{\Omega_I}$ is null iif node $\mathbf{x}_I$ is the mass center of Ω_I, i.e. when Ω_I is a regular or uniformly stretched control volume. Moreover, the eigenvalues of $\overline{\boldsymbol{\Delta}_I \otimes \boldsymbol{\Delta}_I}^{\Omega_I}$ are the moments of inertia of Ω_I, implying that $\overline{\boldsymbol{\Delta}_I \otimes \boldsymbol{\Delta}_I}^{\Omega_I}$ is never null. Adapting formulas given in [68] for a tetrahedron, the authors have been able to derive the first- and second-order moments of any simplex w.r.t. any point in the computational domain. The computation of the first- and second-order moments is now illustrated on the two-dimensional control volume depicted in Fig. 6.3. The following methodology is equivalently applicable to three-dimensional grids.

The first-order moment $\mathbf{M}_{1,i}^O$ of simplex $\mathcal{S}_i$ w.r.t. the frame origin O is given by

$$\mathbf{M}_{1,i}^O = \frac{1}{d+1} \sum_{i=1}^{d+1} \mathbf{x}_i, \tag{6.21}$$

where d is the simplex dimension and the points $\mathbf{x}_i$ are the simplex vertices. The second-order

moment $\mathbf{M}_{2,i}^{O}$ of simplex $\mathcal{S}_i$ w.r.t. point O is given by

$$\mathbf{M}_{2,i}^{O} = \frac{1}{(d+1)(d+2)} \left[\sum_{i=1}^{d+1} \sum_{j=1}^{d+1} \mathbf{x}_i \otimes \mathbf{x}_j + \sum_{i=1}^{d+1} \mathbf{x}_i \otimes \mathbf{x}_i \right]. \tag{6.22}$$

In the case of the two-dimensional simplices $\mathcal{S}_i$ shown in Figure 6.3, one has $d = 2$ in Eqs. (6.21) and (6.22). Moments can also be computed w.r.t. any point of Ω from moments computed w.r.t. point O using translation formulas. We are interested in the first- and second-order moments of the control volume Ω_I computed w.r.t. node $\mathbf{x}_I$ represented by point I on Fig. 6.3. The first-order moment $\mathbf{M}_{1,i}^{I}$ of simplex S_i w.r.t. point I is given by

$$\mathbf{M}_{1,i}^{I} = \mathbf{M}_{1,i}^{O} - \mathbf{x}_I, \tag{6.23}$$

and the second-order moment $\mathbf{M}_{2,i}^{I}$ of simplex S_i w.r.t. point I is given by

$$\mathbf{M}_{2,i}^{I} = \mathbf{M}_{2,i}^{O} - \mathbf{x}_I \otimes \mathbf{M}_{1,i}^{O} - \mathbf{M}_{1,i}^{O} \otimes \mathbf{x}_I + \mathbf{x}_I \otimes \mathbf{x}_I. \tag{6.24}$$

Finally, using the linearity of the integral, the moments of the whole control volume Ω_I are given by summation of the moments of all simplices constituting Ω_I. In the case depicted in Fig.6.3, Ω_I is composed of ten simplices. The first-order moment $\overline{\boldsymbol{\Delta}_I}^{\Omega_I}$ of Ω_I w.r.t. point I is then given by

$$\overline{\boldsymbol{\Delta}_I}^{\Omega_I} = \sum_{i=1}^{10} \mathbf{M}_{1,i}^{I}, \tag{6.25}$$

and the second-order moment $\overline{\boldsymbol{\Delta}_I \otimes \boldsymbol{\Delta}_I}^{\Omega_I}$ of Ω_i w.r.t. point I, by

$$\overline{\boldsymbol{\Delta}_I \otimes \boldsymbol{\Delta}_I}^{\Omega_I} = \sum_{i=1}^{10} \mathbf{M}_{2,i}^{I}. \tag{6.26}$$

Using the moments of control volumes, the data deconvolution step can now be detailed.

6.2.4.3 Data deconvolution from control volumes

Let $\mathcal{N}_I$ be the direct neighbor node set of node $\mathbf{x}_I$. Let $\mathbb{N}_I \subset \mathbb{N}$ be the set of integers q such that node $\mathbf{x}_{J_q} \in \mathcal{N}_I$. At node $\mathbf{x}_I$, the classical integrated gradient operator $\mathcal{G}_I$ and hessian operator $\mathcal{H}_I$ applied to the field Z lead to

$$\mathcal{G}_I(\mathbf{Z}) = \frac{1}{V_I} \sum_{q \in \mathbb{N}_I} \frac{Z_I + Z_{J_q}}{2} \mathbf{A}_{I,J_q} \tag{6.27}$$

and

$$\mathcal{H}_I(\mathbf{Z}) = \frac{1}{V_I} \sum_{q \in \mathbb{N}_I} \left(\frac{Z_{J_q} - Z_I}{\left\| \mathbf{x}_{J_q} - \mathbf{x}_I \right\|} \right) \frac{1}{2} \left(\mathbf{n}_{I,J_q} \otimes \mathbf{S}_{I,J_q} + \mathbf{S}_{I,J_q} \otimes \mathbf{n}_{I,J_q} \right), \tag{6.28}$$

where $\mathbf{Z}$ is the vector containing all values of Z computed on nodes of $\mathcal{X}$, being pointwise or averaged values, and $\mathbf{n}_{I,J_q} = \left(\mathbf{x}_{J_q} - \mathbf{x}_I \right) / \left\| \mathbf{x}_{J_q} - \mathbf{x}_I \right\|$. Since $\mathcal{G}_I$ is typically second-order accurate

on cartesian grids and first-order accurate on unstructured grids, and since $\mathcal{H}_I$ is typically divergent on unstructured grids, these two operators can not be used to achieve third-order reconstruction of Z_I on unstructured grids by solving Eq. (6.18).

Applying nodal gradient and hessian operators to the averaged quantity of Eq. (6.18) leads to

$$\mathcal{G}_I\left(\overline{Z}^{\Omega}\right) = (\nabla Z_I)\cdot\mathcal{G}_I\left(\overline{\boldsymbol{\Delta}_I}^{\Omega}\right) + \frac{1}{2}(\nabla\nabla Z_I) : \mathcal{G}_I\left(\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega}\right) + \mathcal{O}\left(\Delta^2\right) \tag{6.29}$$

and

$$\mathcal{H}_I\left(\overline{Z}^{\Omega}\right) = (\nabla Z_I)\cdot\mathcal{H}_I\left(\overline{\boldsymbol{\Delta}_I}^{\Omega}\right) + \frac{1}{2}(\nabla\nabla Z_I) : \mathcal{H}_I\left(\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega}\right) + \mathcal{O}\left(\Delta\right). \tag{6.30}$$

Equations (6.18), (6.29) and (6.30) yield the linear system

$$L_I = C_I^{\mathcal{O}_3} H_I + E, \tag{6.31}$$

where

$$L_I = \begin{pmatrix} \overline{Z}^{\Omega_I} \\ \mathcal{G}_I\left(\overline{Z}^{\Omega}\right) \\ \mathcal{H}_I\left(\overline{Z}^{\Omega}\right) \end{pmatrix}, \quad H_I = \begin{pmatrix} Z_I \\ \nabla Z_I \\ \nabla\nabla Z_I \end{pmatrix} \quad \text{and} \quad E = \begin{pmatrix} \mathcal{O}\left(\Delta^3\right) \\ \mathcal{O}\left(\Delta^2\right) \\ \mathcal{O}\left(\Delta\right) \end{pmatrix} \tag{6.32}$$

are respectively the vectors of Low-order approximations, High-order approximations and Errors, and the third-order *convolution matrix* $C_I^{\mathcal{O}_3}$ is given by

$$C_I^{\mathcal{O}_3} = \begin{pmatrix} \overline{1}^{\Omega_I} & \left(\overline{\boldsymbol{\Delta}_I}^{\Omega_I}\right)^T & \frac{1}{2}\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega_I} \\ \mathcal{G}_I\left(\overline{1}^{\Omega}\right) & \mathcal{G}_I\left(\overline{\boldsymbol{\Delta}_I}^{\Omega}\right) & \frac{1}{2}\mathcal{G}_I\left(\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega}\right) \\ \mathcal{H}_I\left(\overline{1}^{\Omega}\right) & \mathcal{H}_I\left(\overline{\boldsymbol{\Delta}_I}^{\Omega}\right) & \frac{1}{2}\mathcal{H}_I\left(\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega}\right) \end{pmatrix}. \tag{6.33}$$

Notice that, in order for the definitions of L_I, H_I and $C_I^{\mathcal{O}_3}$ in Eqs. (6.32) and (6.33) to be consistent, the third-order tensors $\mathcal{H}_I\left(\overline{\boldsymbol{\Delta}_I}^{\Omega}\right)$ and $\frac{1}{2}\mathcal{G}_I\left(\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega}\right)$ as well as the fourth-order tensor $\frac{1}{2}\mathcal{H}_I\left(\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega}\right)$ are transformed to second-order tensors of dimensions 4×2, 2×4 and 4×4, respectively (9×3, 3×9 and 9×9, respectively, in three dimensions). Similarly, the second-order tensors $\mathcal{H}_I\left(\overline{Z}^{\Omega}\right)$, $\nabla\nabla Z_I$, $\mathcal{H}_I\left(\overline{1}^{\Omega}\right)$ and $\frac{1}{2}\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega_I}$ are transformed to first-order tensors of dimensions 4×1, 4×1, 4×1 and 1×4, respectively (9×1, 9×1, 9×1 and 1×9, respectively, in three dimensions). Furthermore, considering the (consistent) differential operators $\mathcal{G}_I$ and $\mathcal{H}_I$ defined in Eqs. (6.27) and (6.28), one has $\mathcal{G}_I\left(\overline{1}^{\Omega}\right) = 0$ and $\mathcal{H}_I\left(\overline{1}^{\Omega}\right) = 0$. In two dimensions, $C_I^{\mathcal{O}_3}$ is then a 7×7 matrix, while in three dimensions, it is a 13×13 matrix[1].

If the convolution matrix was equal to the identity matrix, then the nodal and integrated variables would be equal up to the orders given by E and no deconvolution would be neeeded : operators $\mathcal{G}_I$ and $\mathcal{H}_I$ could directly be used to perform a polynomial reconstruction of Z_I. Unfortunately,

[1]In practice, the symmetry of $\mathcal{H}_I$ and $\overline{\boldsymbol{\Delta}_I\otimes\boldsymbol{\Delta}_I}^{\Omega}$ is actually used to reduce the tensor orders and thus the size of $\mathcal{C}_I^{\mathcal{O}_3}$.

the convolution matrix is always different from the identity matrix, since, as previously stated, $\overline{\boldsymbol{\Delta}_I \otimes \boldsymbol{\Delta}_I}^{\Omega_I}$ is never null. Inverting linear system of Eq. (6.31) yields

$$\begin{pmatrix} Z_I^{\mathcal{O}_3} \\ \mathbf{G}_I^{\mathcal{O}_2}(Z) \\ \mathbf{H}_I^{\mathcal{O}_1}(Z) \end{pmatrix} = \left(C_I^{\mathcal{O}_3}\right)^{-1} \begin{pmatrix} \overline{Z}^{\Omega_I} \\ \mathcal{G}_I\left(\overline{Z}^{\Omega}\right) \\ \mathcal{H}_I\left(\overline{Z}^{\Omega}\right) \end{pmatrix}, \tag{6.34}$$

where $Z_I^{\mathcal{O}_3}$ is a third-order accurate approximation of Z_I, $\mathbf{G}_I^{\mathcal{O}_2}(Z)$ is a second-order accurate approximation of ∇Z_I and $\mathbf{H}_I^{\mathcal{O}_1}(Z)$ is a first-order accurate approximation of $\nabla\nabla Z_I$, respectively.

This method enables a third-order accurate approximation of Z_I even on highly distorted grids and only requires the computation of two compact operators and the inversion of a relatively small matrix at each grid node, avoiding the necessity to build large stencils. As such, this method is very well suited for parallelism. Nevertheless, attention has to be paid prior to the solving of Eq. (6.34) to the invertibility of matrix $C_I^{\mathcal{O}_3}$. This matrix is invertible iif node $\mathbf{x}_I$ has a sufficient number of direct neighbor nodes (see [9] for details). In case of an insufficient number of direct neighbors to perform a third-order reconstruction, a lower-order reconstruction can be applied, or the third-order reconstruction performed on some direct neighbor node can be used to reconstruct Z at order three on the initial node.

Finally, the deconvoluted operators in the lhs of Eq. (6.34) provide a third-order Taylor expansion of Z on node $\mathbf{x}$ around node $\mathbf{x}_I$, given by

$$Z_I^{\mathcal{O}_3}(\mathbf{x}) = Z_I^{\mathcal{O}_3} + \mathbf{G}_I^{\mathcal{O}_2}(Z) \cdot \boldsymbol{\Delta}_I(\mathbf{x}) + \frac{1}{2}\mathbf{H}_I^{\mathcal{O}_1}(Z) : (\boldsymbol{\Delta}_I(\mathbf{x}) \otimes \boldsymbol{\Delta}_I(\mathbf{x})). \tag{6.35}$$

One has to notice that this methodology is still valid if $\mathbf{x} \in \Omega_J$ where $J \neq I$, enabling high-order extrapolations from one node to an arbitrary point in the computational domain. This methodology has been extended to the reconstruction of vector fields, in two and three dimensions.

6.2.5 Extrapolations using the High-Order Framework applied to the computation of the explicit temperature advection term close to the interface

The High-Order Framework can also be used to perform high-order extrapolations. In the case of the explicit temperature advection term computation of Eq. (6.2), we use the High-Order Framework to perform Taylor expansions in order to compute a relevant temperature gradient on nodes of band level ± 1. Refering to Fig. 6.2(b), nodes of band level $+2$ contribute to the liquid temperature gradient on node $\mathbf{p}$ by means of the second-order Taylor expansion given by

$$\mathbf{G}_{\mathbf{q}_j}^{\mathcal{O}_2}(T_{\text{liq}})(\mathbf{p}) = \mathbf{G}_{\mathbf{q}_j}^{\mathcal{O}_2}(T_{\text{liq}}) + \mathbf{H}_{\mathbf{q}_j}^{\mathcal{O}_1}(T_{\text{liq}}) \cdot (\mathbf{p} - \mathbf{q}_j) \tag{6.36}$$

where $j \in \{3, 4\}$. The final value of $\mathbf{G}_{\mathbf{p}}^{\mathcal{O}_2}(T_{\text{liq}})$ is then computed as the average of all Taylor expansions. In the case depicted in Fig. 6.2(b), one has

$$\mathbf{G}_{\mathbf{p}}^{\mathcal{O}_2}(T_{\text{liq}}) = \frac{1}{2}\left(\mathbf{G}_{\mathbf{q}_3}^{\mathcal{O}_2}(T_{\text{liq}})(\mathbf{p}) + \mathbf{G}_{\mathbf{q}_4}^{\mathcal{O}_2}(T_{\text{liq}})(\mathbf{p})\right). \tag{6.37}$$

Similarly, the vapor temperature gradient is computed on nodes of band level -1 by means of Taylor expansions from nodes of band level -2. This method avoids differentiating across the interface, thus does not require the computation of ghost values.

6.2.6 Computation of the temporal temperature variation when the interface crosses the node

When a node is swept over by the interface between times n and $n+1$, the computation of the temporal fluctuation $\left(T^{n+1}-T^n\right)/\Delta t$ requires the knowledge of T^n, as shown in Fig. 6.1. Due to the immersed Dirichlet boundary condition at the interface, it is convenient to consider that the liquid and vapor subdomains are unrelated when solving the heat equation. Then, when a node has been swept over by the interface, e.g. when a node has left the liquid phase to enter the vapor phase, no relevant vapor temperature value was known at time n since the node was in the liquid phase. As a result, one can not compute the temporal fluctuation of the vapor temperature field on this node. One solution to this problem is to perform extrapolation of vapor temperature values from nodes of band level -1 to nodes of band level $+1$. Such extrapolations can be performed using the High-Order Framework by means of Eq. (6.35) where Z stands for T_{vap}, I represents a node of band level -1 and $\mathbf{x}$ is a node a band level $+1$.

6.2.7 Summary of the numerical method used to solve the heat equation with immersed Dirichlet boundary condition at the interface

In this section, we summarize the different steps used to solve the heat equation in the Boiling solver. We refer to Fig. 6.2 where node $\mathbf{p}$ is far from the interface in Subfig. 6.2(a) and close to the interface in Subfig. 6.2(b). Equation (6.2) is solved at node $\mathbf{p}$ with full implicit diffusion ($\beta = 1$), i.e.

$$\frac{\rho(\mathbf{p})c_{\text{p}}(\mathbf{p})}{\Delta t}T^{n+1}(\mathbf{p}) - \nabla\cdot\left(\lambda(\mathbf{p})\nabla T^{n+1}(\mathbf{p})\right) = \frac{\rho(\mathbf{p})c_{\text{p}}(\mathbf{p})}{\Delta t}T^{n}(\mathbf{p}) - \rho(\mathbf{p})c_{\text{p}}(\mathbf{p})\mathbf{u}(\mathbf{p})\cdot\nabla T^{n}(\mathbf{p}). \quad (6.38)$$

For clarity in the notations, the implicit diffusion term will be noted $\nabla\cdot(\lambda\nabla T)|_{\mathbf{p}}^{n+1}$ and the temperature gradient in the explicit advection term will be noted $\nabla T|_{\mathbf{p}}^{n}$. The control volume associated to node $\mathbf{p}$ is denoted $\Omega_{\mathbf{p}}$.

6.2.7.1 Implicit diffusion term

If the interface does not cross the control volume $\Omega_{\mathbf{p}}$, as in Subfig. 6.2(a), the diffusion term is computed as

$$\nabla\cdot(\lambda\nabla T)|_{\mathbf{p}}^{n+1} = \frac{1}{\mathcal{V}_{\mathbf{p}}}\sum_{j=1}^{5}\lambda_{\mathbf{p},\mathbf{q}_j}^{n+1}\frac{T_{\mathbf{q}_j}^{n+1}-T_{\mathbf{p}}^{n+1}}{\|\mathbf{q}_j-\mathbf{p}\|^2}(\mathbf{q}_j-\mathbf{p})\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_j}, \quad (6.39)$$

where $\lambda_{\mathbf{p},\mathbf{q}_j}^{n+1} = \left(\lambda_{\mathbf{p}}^{n+1}+\lambda_{\mathbf{q}_j}^{n+1}\right)/2$ is known (since the interface has already been advected).

If the interface crosses $\Omega_{\mathbf{p}}$, as in Subfig. 6.2(b), the diffusion term is computed as

$$\nabla\cdot(\lambda\nabla T)|_{\mathbf{p}}^{n+1} = \frac{1}{\mathcal{V}_{\mathbf{p}}}\lambda_{\mathbf{p},\mathbf{q}_1}^{n+1}\frac{T_{\text{sat}}-T_{\mathbf{p}}^{n+1}}{\theta_{\mathbf{p},\mathbf{q}_1}\|\mathbf{q}_1-\mathbf{p}\|^2}(\mathbf{q}_1-\mathbf{p})\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_1} + \frac{1}{\mathcal{V}_{\mathbf{p}}}\sum_{j=2}^{5}\lambda_{\mathbf{p},\mathbf{q}_j}^{n+1}\frac{T_{\mathbf{q}_j}^{n+1}-T_{\mathbf{p}}^{n+1}}{\|\mathbf{q}_j-\mathbf{p}\|^2}(\mathbf{q}_j-\mathbf{p})\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_j}. \quad (6.40)$$

6.2.7.2 Explicit advection term

If the interface does not cross $\Omega_{\mathbf{p}}$, the explicit temperature gradient of the advection term is computed with the classical second-order finite volume gradient given by

$$\nabla T|_{\mathbf{p}}^{n} = \frac{1}{\mathcal{V}_{\mathbf{p}}} \sum_{j=1}^{5} \frac{T_{\mathbf{p}}^{n} + T_{\mathbf{q}_j}^{n}}{2} \mathbf{A}_{\mathbf{p},\mathbf{q}_j}. \tag{6.41}$$

If the interface crosses $\Omega_{\mathbf{p}}$, the explicit temperature gradient of the advection term is computed at $\mathbf{p}$ by means of Taylor expansions performed using the High-Order Framework from the direct neighbor nodes of $\mathbf{p}$ located in the same phase. In the case depicted in Subfig. 6.2(b), one has

$$\begin{aligned} \nabla T|_{\mathbf{p}}^{n} &= \mathbf{G}_{\mathbf{p}}^{\mathcal{O}_2}\left(T^n\right) \\ &= \frac{1}{2}\left(\mathbf{G}_{\mathbf{q}_3}^{\mathcal{O}_2}\left(T^n\right)(\mathbf{p}) + \mathbf{G}_{\mathbf{q}_4}^{\mathcal{O}_2}\left(T^n\right)(\mathbf{p})\right), \end{aligned} \tag{6.42}$$

where the rhs of Eq. (6.42) is given by Eqs. (6.36) and (6.37).

6.2.7.3 Complete linear system

Equation (6.38) leads to the following linear system

$$\overline{\overline{\mathbf{A}}}\mathbf{T}^{n+1} = \mathbf{B}^{n}, \tag{6.43}$$

where $\overline{\overline{\mathbf{A}}}$ is a matrix defined below, $\mathbf{T}^{n+1}$ is the vector of unknown temperature values and $\mathbf{B}^{n}$ is the vector of known temperature values.

Let $N \in \mathbb{N}^*$ be the total node number of the grid and let $p \in \{1, \ldots, N\}$ be the indice of node $\mathbf{p}$.

If the interface does not cross the control volume (Subfig. 6.2(a)), then using Eqs. (6.39) and (6.41), Eq. (6.38) can be rewritten at node $\mathbf{p}$ as

$$\begin{aligned} \left(\frac{\rho c_{\mathrm{p}}}{\Delta t} + \frac{1}{\mathcal{V}_{\mathbf{p}}} \sum_{j=1}^{5} \lambda_{\mathbf{p},\mathbf{q}_j} \frac{\mathbf{q}_j - \mathbf{p}}{\|\mathbf{q}_j - \mathbf{p}\|^2} \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_j}\right) T_{\mathbf{p}}^{n+1} - \frac{1}{\mathcal{V}_{\mathbf{p}}} \sum_{j=1}^{5} \lambda_{\mathbf{p},\mathbf{q}_j} \frac{\mathbf{q}_j - \mathbf{p}}{\|\mathbf{q}_j - \mathbf{p}\|^2} \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_j} T_{\mathbf{q}_j}^{n+1} \\ = \frac{\rho c_{\mathrm{p}}}{\Delta t} T_{\mathbf{p}}^{n} - \rho c_{\mathrm{p}} \mathbf{u}_{\mathbf{p}} \cdot \nabla T|_{\mathbf{p}}^{n}. \end{aligned} \tag{6.44}$$

Let $J \subset \{1, \ldots, N\}$ be the indices of the direct neighbor nodes $\mathbf{q}_j$ of node $\mathbf{p}$. The p-th line $\overline{\overline{A}}_{p,\cdot}$ of matrix $\overline{\overline{\mathbf{A}}}$ is given by

$$\overline{\overline{A}}_{p,\cdot} = \left(A_{p,1}, \ldots, A_{p,N}\right), \tag{6.45}$$

where, for $1 \leq k \leq N$, one has

$$A_{p,k} = \begin{cases} \dfrac{\rho c_{\mathrm{p}}}{\Delta t} + \dfrac{1}{\mathcal{V}_{\mathbf{p}}} \sum_{j \in J} \lambda_{\mathbf{p},\mathbf{q}_j} \dfrac{\mathbf{q}_j - \mathbf{p}}{\|\mathbf{q}_j - \mathbf{p}\|^2} \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_j}, & \text{if } k = p, \\ -\dfrac{1}{\mathcal{V}_{\mathbf{p}}} \lambda_{\mathbf{p},\mathbf{q}_k} \dfrac{\mathbf{q}_k - \mathbf{p}}{\|\mathbf{q}_k - \mathbf{p}\|^2} \cdot \mathbf{A}_{\mathbf{p},\mathbf{q}_k}, & \text{if } k \in J, \\ 0, & \text{otherwise.} \end{cases} \tag{6.46}$$

The p-th component B_p^n of vector $\mathbf{B}^n$ is given by the rhs of Eq. (6.44).

If the interface crosses an edge $(\mathbf{p}, \mathbf{q}_j)$ (Subfig. 6.2(b)), then using Eqs. (6.40) and (6.42), Eq. (6.38) can be rewritten at node $\mathbf{p}$ as

$$\begin{aligned}\left(\frac{\rho c_{\mathrm{p}}}{\Delta t} + \frac{1}{\mathcal{V}_{\mathbf{p}}}\lambda_{\mathbf{p},\mathbf{q}_1}\frac{\mathbf{q}_1-\mathbf{p}}{\theta_{\mathbf{p},\mathbf{q}_1}\left\|\mathbf{q}_1-\mathbf{p}\right\|^2}\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_1} + \frac{1}{\mathcal{V}_{\mathbf{p}}}\sum_{j=2}^{5}\lambda_{\mathbf{p},\mathbf{q}_j}\frac{\mathbf{q}_j-\mathbf{p}}{\left\|\mathbf{q}_j-\mathbf{p}\right\|^2}\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_j}\right)T_{\mathbf{p}}^{n+1} & \\ -\frac{1}{\mathcal{V}_{\mathbf{p}}}\sum_{j=2}^{5}\lambda_{\mathbf{p},\mathbf{q}_j}\frac{\mathbf{q}_j-\mathbf{p}}{\left\|\mathbf{q}_j-\mathbf{p}\right\|^2}\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_j}T_{\mathbf{q}_j}^{n+1} & \\ = \frac{\rho c_{\mathrm{p}}}{\Delta t}T_{\mathbf{p}}^n - \rho c_{\mathrm{p}}\mathbf{u}_{\mathbf{p}}\cdot\nabla T|_{\mathbf{p}}^n + \frac{1}{\mathcal{V}_{\mathbf{p}}}\lambda_{\mathbf{p},\mathbf{q}_1}\frac{\mathbf{q}_1-\mathbf{p}}{\theta_{\mathbf{p},\mathbf{q}_1}\left\|\mathbf{q}_1-\mathbf{p}\right\|^2}\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_1}T_{\mathrm{sat}} &\end{aligned} \tag{6.47}$$

Let $J^* \subsetneq J$ be the indices of the direct neighbor nodes $\mathbf{q}_j$ of node $\mathbf{p}$ which are located in the same phase than $\mathbf{p}$. The p-th line $\overline{\overline{A}}_{p,\cdot}$ of matrix $\overline{\overline{\mathbf{A}}}$ is given by Eq. (6.45) where, for $1 \leq k \leq N$, one has

$$A_{p,k} = \begin{cases} \dfrac{\rho c_{\mathrm{p}}}{\Delta t} + \dfrac{1}{\mathcal{V}_{\mathbf{p}}}\lambda_{\mathbf{p},\mathbf{q}_1}\dfrac{\mathbf{q}_1-\mathbf{p}}{\theta_{\mathbf{p},\mathbf{q}_1}\left\|\mathbf{q}_1-\mathbf{p}\right\|^2}\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_1} + \dfrac{1}{\mathcal{V}_{\mathbf{p}}}\sum_{j\in J^*}\lambda_{\mathbf{p},\mathbf{q}_j}\dfrac{\mathbf{q}_j-\mathbf{p}}{\left\|\mathbf{q}_j-\mathbf{p}\right\|^2}\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_j} & \text{if } k=p, \\ -\dfrac{1}{\mathcal{V}_{\mathbf{p}}}\lambda_{\mathbf{p},\mathbf{q}_k}\dfrac{\mathbf{q}_k-\mathbf{p}}{\left\|\mathbf{q}_k-\mathbf{p}\right\|^2}\cdot\mathbf{A}_{\mathbf{p},\mathbf{q}_k}, & \text{if } k\in J, \\ 0, & \text{otherwise.} \end{cases} \tag{6.48}$$

The p-th component B_p^n of vector $\mathbf{B}^n$ is given by the rhs of Eq. (6.47).

In both cases, the linear system of Eq. (6.43) is solved using the BiConjugate Gradient solver [92]. While the relative distance $\theta_{\mathbf{p},\mathbf{q}_1}$ appears in both the lhs and rhs of Eq. (6.47) in denominators, our simulations did not require the use of a threshold on the value of $\theta_{\mathbf{p},\mathbf{q}_1}$, as opposed to the explicit resolution of the heat equation used in one dimension (see Section 4.3.3).

6.3 Computation of the mass transfer rate

Once the heat equation has been solved in both phases, T_{liq} and T_{vap} are known at time $n+1$ in their respective phases. The mass transfer rate $\dot{m}$ can then be computed at time $n+1$. Since the mass transfer rate has physical meaning only at the interface, special care is taken for its computation. Again, we use the High-Order Framework presented in Section 6.2.4. We recall that the mass tranfer rate $\dot{m}$ is given by

$$\dot{m}_\Gamma = \frac{[-\lambda\nabla T\cdot\mathbf{n}_\Gamma]_\Gamma}{L_v} \tag{6.49}$$

$$= \frac{-\lambda_{\mathrm{liq}}\,\nabla T_{\mathrm{liq}}|_\Gamma + \lambda_{\mathrm{vap}}\,\nabla T_{\mathrm{vap}}|_\Gamma}{L_v}\cdot\mathbf{n}_\Gamma, \tag{6.50}$$

where L_v is the (constant) latent heat of the fluid. In order to compute $\nabla T_{\mathrm{liq}}|_\Gamma$ and $\nabla T_{\mathrm{vap}}|_\Gamma$ precisely at the interface, we use the High-Order Framework to perform high-order extrapolations. First, we compute $\nabla T_{\mathrm{liq}}|_{b=2}$ on the liquid nodes of band level $+2$ and $\nabla T_{\mathrm{vap}}|_{b=-2}$ on the vapor nodes of band level -2 using the second-order accurate nodal deconvoluted gradient operator $\mathbf{G}_I^{\mathcal{O}_2}$ of Eq. (6.34). Then, $\nabla T_{\mathrm{liq}}|_{b=2}$ is extrapolated by Taylor expansion on the liquid nodes of band level

$+1$ using Eqs. (6.36) and (6.37), yielding $\nabla T_{\text{liq}}|_{b=1}$. A similar Taylor expansion of $\nabla T_{\text{vap}}|_{b=-2}$ on the vapor nodes of band level -1 yields $\nabla T_{\text{vap}}|_{b=-1}$. After this step, each grid edge $(\mathbf{p}, \mathbf{q})$ such that node $\mathbf{p}$ is in the liquid phase and node $\mathbf{q}$ is in the vapor phase has knowledge of a value of $\nabla T_{\text{liq}}|_{\mathbf{p}}$ on $\mathbf{p}$ and $\nabla T_{\text{vap}}|_{\mathbf{q}}$ on $\mathbf{q}$. Along every such edge $(\mathbf{p}, \mathbf{q})$, another Taylor expansion is performed for $\nabla T_{\text{liq}}|_{\mathbf{p}}$ from node $\mathbf{p}$ to the interface, yielding $\nabla T_{\text{liq}}|_{\mathbf{p},\Gamma}$. Similarly, $\nabla T_{\text{vap}}|_{\mathbf{q}}$ is extrapolated to the interface location, yielding $\nabla T_{\text{vap}}|_{\mathbf{q},\Gamma}$. The interface normal vector is computed at the same interface location by $\mathbf{n}_{\mathbf{p},\mathbf{q};\Gamma} = (1 - \theta_{\mathbf{p},\mathbf{q}})\,\mathbf{n}_{\mathbf{p}} + \theta_{\mathbf{p},\mathbf{q}}\mathbf{n}_{\mathbf{q}}$. The mass transfer rate $\dot{m}$ is then computed precisely at the interface location along edge $(\mathbf{p}, \mathbf{q})$ by

$$\dot{m}_{\mathbf{p},\mathbf{q};\Gamma} = \frac{-\lambda_{\text{liq}}\ \nabla T_{\text{liq}}|_{\mathbf{p},\Gamma} + \lambda_{\text{vap}}\ \nabla T_{\text{vap}}|_{\mathbf{q},\Gamma}}{L_v} \cdot \mathbf{n}_{\mathbf{p},\mathbf{q};\Gamma}. \tag{6.51}$$

Finally, since values of $\dot{m}$ are needed on nodes to solve the level set advection equation and Navier-Stokes equations, the value of $\dot{m}$ stored on node $\mathbf{p}$ is computed as the simple average of the values $\dot{m}_{\mathbf{p},\mathbf{q};\Gamma}$ computed on edges containing node $\mathbf{p}$.

6.4 Improvement of the interface curvature computation using the High-Order Framework

The deconvoluted operators presented in Section 6.2.4 have been applied in the Boiling solver to the computation of the interface curvature by Eq. (3.28). Let $(\mathbf{x}_I, \mathbf{x}_J)$ be a grid edge crossed by the interface Γ. First, the second-order deconvoluted gradient operator $\mathbf{G}_I^{\mathcal{O}_2}(\phi)$ and first-order deconvoluted hessian operator $\mathbf{H}_I^{\mathcal{O}_1}(\phi)$ are computed on nodes $\mathbf{x}_I$ and $\mathbf{x}_J$, where ϕ is the signed distance function to the interface. Then, since the interface curvature κ has mathematical meaning only at the interface, these operators are extrapolated by Taylor expansion along the edge $(\mathbf{x}_I, \mathbf{x}_J)$ to the exact interface location $\mathbf{x}_\Gamma$. These Taylor expansions are given by

$$\begin{cases} \mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) = \mathbf{G}_I^{\mathcal{O}_2}(\phi) + \boldsymbol{\Delta}_I(\mathbf{x}_\Gamma) \cdot \mathbf{H}_I^{\mathcal{O}_1}(\phi) + \mathcal{O}\left(\Delta^2\right) \\ \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi) = \mathbf{G}_J^{\mathcal{O}_2}(\phi) + \boldsymbol{\Delta}_J(\mathbf{x}_\Gamma) \cdot \mathbf{H}_J^{\mathcal{O}_1}(\phi) + \mathcal{O}\left(\Delta^2\right) \end{cases} \tag{6.52}$$

and

$$\begin{cases} \mathbf{H}_{\Gamma,I}^{\mathcal{O}_1}(\phi) = \mathbf{H}_I^{\mathcal{O}_1}(\phi) + \mathcal{O}(\Delta) \\ \mathbf{H}_{\Gamma,J}^{\mathcal{O}_1}(\phi) = \mathbf{H}_J^{\mathcal{O}_1}(\phi) + \mathcal{O}(\Delta) \end{cases}. \tag{6.53}$$

Equation (3.28) is then rewritten

$$\kappa_\Gamma = \frac{\nabla\phi|_\Gamma^{\mathrm{T}} \cdot H(\phi)|_\Gamma \cdot \nabla\phi|_\Gamma - \|\nabla\phi|_\Gamma\|^2 \operatorname{Tr}\left(H(\phi)|_\Gamma\right)}{\|\nabla\phi|_\Gamma\|^3} \tag{6.54}$$

$$= \frac{\begin{array}{r}\frac{1}{2}\left(\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right)^{\mathrm{T}} \cdot \frac{1}{2}\left(\mathbf{H}_{\Gamma,I}^{\mathcal{O}_1}(\phi) + \mathbf{H}_{\Gamma,J}^{\mathcal{O}_1}(\phi)\right) \cdot \frac{1}{2}\left(\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right) \\ - \left\|\frac{1}{2}\left(\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right)\right\|^2 \operatorname{Tr}\left(\frac{1}{2}\left(\mathbf{H}_{\Gamma,I}^{\mathcal{O}_1}(\phi) + \mathbf{H}_{\Gamma,J}^{\mathcal{O}_1}(\phi)\right)\right)\end{array}}{\left\|\frac{1}{2}\left(\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right)\right\|^3} \tag{6.55}$$

$$= \frac{\begin{array}{r}\left(\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right)^{\mathrm{T}} \cdot \left(\mathbf{H}_{\Gamma,I}^{\mathcal{O}_1}(\phi) + \mathbf{H}_{\Gamma,J}^{\mathcal{O}_1}(\phi)\right) \cdot \left(\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right) \\ - \left\|\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right\|^2 \operatorname{Tr}\left(\mathbf{H}_{\Gamma,I}^{\mathcal{O}_1}(\phi) + \mathbf{H}_{\Gamma,J}^{\mathcal{O}_1}(\phi)\right)\end{array}}{\left\|\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right\|^3}, \tag{6.56}$$

where

$$\nabla\phi|_\Gamma = \frac{1}{2}\left(\mathbf{G}_{\Gamma,I}^{\mathcal{O}_2}(\phi) + \mathbf{G}_{\Gamma,J}^{\mathcal{O}_2}(\phi)\right) \tag{6.57}$$

and

$$H(\phi)|_\Gamma = \frac{1}{2}\left(\mathbf{H}_{\Gamma,I}^{\mathcal{O}_1}(\phi) + \mathbf{H}_{\Gamma,J}^{\mathcal{O}_1}(\phi)\right). \tag{6.58}$$

One can notice that the sums of Eqs. (6.57) and (6.58) can also be weighted by the relative distance $\theta_{\mathbf{x}_I,\mathbf{x}_J}$ in order to improve accuracy. In [9], the interface curvature computed using Eq. (6.56) is shown to be second-order accurate on cartesian grids and first-order accurate on unstructured grids. Convergence of interface curvature on unstructured grids is a very important achievement since the curvature, as a sum of second-order derivatives of the potentially polluted signed distance function to the interface (due to reinitialization), often degrades the accuracy of two-phase flow simulations [19, 13, 51].

Figure 6.4 shows the improvement in the interface curvature computation provided by the High-Order Framework (Eq. (6.56)) w.r.t. the classical finite-volume operators (Eqs. (3.16) and (3.17)).

Equation (6.56) is the preferred method to compute the interface curvature in the Boiling solver.

6.5 Radially symmetric growth of a 3D bubble: a test case

In this section, the test case used to validate the numerical method implemented in the Boiling solver to simulate boiling flows is presented. This test case consists in a vapor bubble surrounded by a superheated liquid. The vapor temperature field is initialized in the vapor phase at saturation temperature, and the liquid temperature field is initialized in the liquid phase by means of a spherically symmetric profile whose complex analytical formulation has been derived in Scriven [75] and is detailed in Appendix H. The liquid temperature gradient is non-zero in the liquid side of the interface, leading to a non-zero mass transfer rate at the interface. Due to the initially stationary bubble and to the symmetry of the problem, the liquid-vapor interface motion is then only due to phase change. The gravitational forces being neglected, the interface is expected to remain spherical for the whole temporal evolution. The duration of the simulation is equal to the

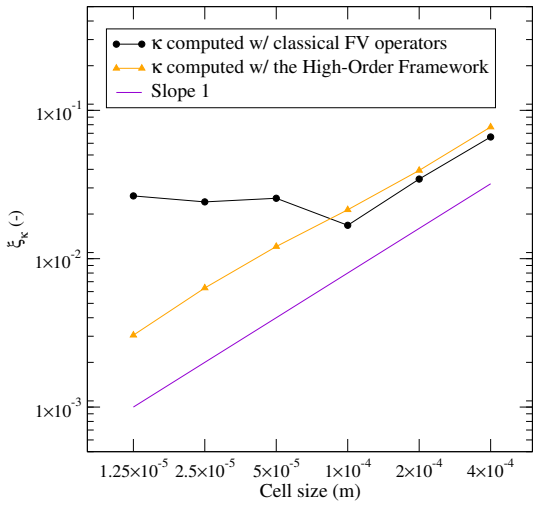


Fig. 6.4 – Relative error of the interface curvature computation on six different two-dimensional simplicial grids using the classical finite-volume operators (Eqs. (3.16) and (3.17)) and the High-Order Framework (Eq. (6.56)). The computation is performed on the analytical expression of the signed distance function ϕ to the interface (no previous reinitialization).

theoretical duration needed for the bubble radius to double. Appendix H also presents a derivation of the initial liquid temperature profile analytical formulation in two dimensions.

6.6 Numerical results

In this section, we present our numerical results in two and three dimensions for the test case introduced in Section 6.5.

6.6.1 Numerical results in 2D

6.6.1.1 On equilateral triangles

In order to avoid the influence of anisotropic control volumes, we first present the results obtained on two-dimensional grids built only with equilateral triangles, leading to regular hexagonal control volumes. The computational domain is a rhombus of side length $L = 8 \times 10^{-3}$ m, as shown in Fig. 6.5. The grids used are listed in Table 6.1. The initial bubble radius is $R_0 = 1 \times 10^{-3}$ m and the simulation is performed until the theoretical bubble radius is twice the initial radius. The physical properties of interest are listed in Table 6.2 where the heat capacity at constant pressure of the liquid has been divided by two orders of magnitude w.r.t. the value used in [85] (see discussion in Section 6.6.2). Figure 6.6 shows the computed interface at final time and the final temperature, velocity and pressure fields for the finest grid. One can notice the high accuracy on the bubble shape at final time : the bubble is perfectly circular and its radius is in excellent agreement with the theoretical one. Due to the symmetry of the problem, the liquid velocity profile is expected to be colinear to the interface normal vector. However, due to the proximity of the domain boundaries

Characteristic grid cell size Δ (m)	Number of cells N (-)
9.375×10^{-5}	14 450
4.688×10^{-5}	57 800
2.344×10^{-5}	231 200

Table 6.1 – Two-dimensional grids composed of equilateral triangles.

ρ_{liq}	ρ_{vap}	σ	μ_{liq}
$958\,\text{kg}\,\text{m}^{-3}$	$0.59\,\text{kg}\,\text{m}^{-3}$	$5.9 \times 10^{-2}\,\text{N}\,\text{m}^{-1}$	$2.82 \times 10^{-4}\,\text{kg}\,\text{m}^{-1}\,\text{s}^{-1}$

μ_{vap}	λ_{liq}	λ_{vap}	$c_{\text{p,liq}}$
$1.23 \times 10^{-6}\,\text{kg}\,\text{m}^{-1}\,\text{s}^{-1}$	$0.6\,\text{W}\,\text{m}^{-1}\,\text{K}^{-1}$	$0.026\,\text{W}\,\text{m}^{-1}\,\text{K}^{-1}$	$42.16\,\text{J}\,\text{kg}^{-1}\,\text{K}^{-1}$

$c_{\text{p,vap}}$	L_v	T_{sat}
$2034\,\text{J}\,\text{kg}^{-1}\,\text{K}^{-1}$	$2.257 \times 10^{6}\,\text{J}\,\text{kg}^{-1}$	373 K

Table 6.2 – Physical properties used in two dimensions. Only $c_{\text{p,liq}}$ differs (by two orders of magnitude) from the value used in Tanguy et al. [85].

to the interface, one can observe anisotropy of the liquid velocity profile, as shown in Fig. 6.6(c). This does not alter the accuracy of the computed interface. Figure 6.7 shows the relative error on the final bubble radius. The relative error on the final bubble radius decreases with grid refinement. A deeper analysis is presented in Fig. 6.8 which shows the error on the x-component of the interface normal vector and the relative error of the interface curvature. The error on the interface normal vector decreases with grid refinement, whereas the error on the interface curvature does not decrease with grid refinement. We emphasize that the errors presented here are given in L^{∞}-norm. Hence one single inaccurate curvature value can lead to a high relative error.

6.6.1.2 On fully unstructured triangles

We now validate our numerical method using unstructured triangles. We have found that large deformations of the interface were observed when the surface tension was considered on arbitrary unstructured triangles of equivalent sizes to the equilateral triangles used in the previous section. These large deformations of the interface are assumed to be due to the lack of accuracy of the interface curvature computation on arbitrary triangles (despite the use of the High-Order Framework), since the product $\sigma\kappa$ is present in the pressure jump at the interface (Eq. (1.31)). As will be shown in Section 6.6.2, this is not the case in three dimensions.

In order to avoid these instabilities, as opposed to the simulations performed on equilateral triangles, the surface tension has been taken equal to zero on unstructured triangles. The grids used are listed in Table 6.3. Figure 6.9 shows the coarsest grid. Figure 6.10 shows the computed interface at final time and the final temperature, velocity and pressure fields for the finest grid. One can observe the high accuracy of the final interface shape and the radial symmetry of the liquid temperature and velocity profiles. Figure 6.11 shows the relative error on the final bubble radius for the three grids. The relative error on the bubble radius at final time is smaller than 5% on the three

Characteristic grid cell size Δ (m)	Number of cells N (-)
1.875×10^{-4}	8 806
9.375×10^{-5}	14 450
4.688×10^{-5}	57 800

Table 6.3 – Two-dimensional grids composed of arbitrary triangles.

grids. The convergence of the bubble radius is less clear than on equilateral triangles (see Fig. 6.7). This is due to the absence of surface tension forces (which would maintain the bubble circular) and to the anisotropy of control volumes (which favors numerical errors on the computation of differential operators). A deeper analysis is presented in Fig. 6.12 which shows the error on the x-component of the interface normal vector and the relative error on the interface curvature. The errors on the interface normal vector and curvature are clearly divergent with grid refinement (Figs. 6.12(a) and 6.12(b)). Again, we emphasize that the errors presented here are given in L^∞-norm. Hence one single inaccurate value can lead to a high relative error.

These results on equilateral and unstructured triangles have been presented in the 4^{th} *World Congress on Momentum, Heat and Mass Transfer* on April 10-12, 2019, in Rome, Italy and can be found in Sahut et al. [72].

6.6.2 Numerical results in 3D

This test case has been simulated by Tanguy et al. [85] on two-dimensional axisymmetric cartesian grids and by Rajkotwala et al. [67] on three-dimensional cartesian grids. Our goal is to reproduce similar results in order to validate our numerical method on three-dimensional unstructured grids. The liquid temperature field is initialized using the procedure described in Section H.1.4. The computational domain is a cube of side length $L = 1.2 \times 10^{-2}\,\mathrm{m}$. The initial bubble radius is $R_0 = 1 \times 10^{-3}\,\mathrm{m}$ and the simulation is performed until the bubble radius is twice the initial one. Equation (H.44) implies that the initial time t_{ini} is not equal to zero but is given by

$$t_{\mathrm{ini}} = \frac{R_0^2}{4K\beta^2}, \tag{6.59}$$

and that the final time t_{fin} is given by

$$t_{\mathrm{fin}} = 4t_{\mathrm{ini}}. \tag{6.60}$$

In Tanguy et al. [85], the simulations are performed for Jakob numbers ranging from 3 to 10, on three two-dimensional axi-symmetric cartesian grids. The authors have simulated the test case using different variants of the numerical method. We chose to compare our method to the variant giving the smallest final relative errors on the bubble radius for the highest tested Jakob numbers, since increasing the Jakob number decreases the width of the thermal boundary layer at the interface, thus making the mass transfer rate more challenging to compute on a given grid [85]. We performed our simulations with Jakob numbers of 3, 5, 7 and 9. Table 6.4 reproduces the errors given for these Jakob numbers in Table 6 of [85]. We performed our simulations on the three three-dimensional unstructured tetrahedral grids listed in Table 6.5. In order to speed up computations, the characteristic cell sizes of these grids are multiplied by a factor 1.6 w.r.t. the characteristic cell

Δ (m)	Ja=3	Ja=5	Ja=7	Ja=9
9.375×10^{-5}	9.3%	21%	30.5%	36.6%
4.688×10^{-5}	1.4%	2.8%	7.9%	14.4%
2.344×10^{-5}	1.0%	1.0%	0.5%	1.0%

Table 6.4 – Relative errors on the bubble radius at final time for different Jakob numbers on three two-dimensional axi-symmetric cartesian grids extracted from Table 6 of Tanguy et al. [85]. The cell sizes in the first column have been computed from the number of cells and the size of the domain given by the authors.

Δ (m)	Number of cells N (-)
1.5×10^{-4}	554 800
7.5×10^{-5}	1 933 621
3.75×10^{-5}	9 186 699

Table 6.5 – Unstructured tetrahedral grids used in three dimensions. The cell sizes correspond to the ones listed in Table 6.4 multiplied by 1.6. The refined area around the interface is shown in Fig. 6.13.

Δ (m)	Ja=3	Ja=5	Ja=7	Ja=9
1.5×10^{-4}	6.8%	20%	34%	47%
7.5×10^{-5}	0.8%	1.7%	7.9%	16%
3.75×10^{-5}	1.7%	0.6%	1.0%	1.9%

Table 6.6 – Relative errors on the bubble radius at final time for different unstructured grids and Jakob numbers in three dimensions.

sizes used in [85] and listed in Table 6.4. Moreover, the liquid heat capacity at constant pressure is divided by two orders of magnitude with respect to the value used in [85] (see Table 6.2). Indeed, Eq. (H.66) shows that if $c_{\mathrm{p,liq}}$ decreases, then T_∞ increases (all other parameters being unchanged). As a result, the liquid temperature field, given at time t by Eq. (H.72), has a steeper initial slope close to the interface. Consequently, the mass transfer rate $\dot{m}$ and so the interface velocity due to phase change (rhs of Eq. (5.28)) are then increased, leading to faster simulations.

Table 6.6 shows the relative errors of the L^∞-norm of the bubble radius at final time on the three grids and for the four Jakob numbers considered. The relative error on the bubble radius at final time decreases with grid refinement for all Jakob numbers (except for Ja = 3 on the finest grid). Moreover, the relative error on the bubble radius at final time decreases when the Jakob number is decreased on all grids considered (except for Ja = 3 on the finest grid). One can observe an error increase for the smallest Jakob number (Ja = 3) on the finest grid. This behavior can be related to the results of Tanguy et al. [85] listed in Table 6.4 in which the error on the bubble radius reaches a minimum value on the finest grid and does not clearly converge with grid refinement below this minimum value. As stated above and in [85], increasing the Jakob number reduces, by Eq. (H.72), the width of the thermal boundary layer at the interface. Consequently, the mass

transfer rate and the interface velocity due to phase change are computed with less accuracy since a reduced number of grids nodes are located in the thermal boundary layer at every iteration. Then, the bubble radius, measured at final time from the signed distance function advected by the interface velocity due to phase change, is also less accurate. The convergence of the bubble radius with grid refinement and decrease of Jakob number has been obtained by Tanguy et al. [85] on two-dimensional axisymmetric cartesian grids, as summarized in Table 6.4. In the present work, this convergence is extended to three-dimensional unstructured grids with comparable results : in our simulations, the bubble radius at final time is generally slightly more accurate for low Jakob numbers (3 and 5) and slightly less accurate for high Jakob numbers (7 and 9). Since our grid cells are larger than the ones used in [85] by a factor 1.6, and since our grids are three-dimensional, which implies that the sizes of the tetrahedra mentioned in Table 6.5 are only approximated by the meshing software, further comparison would not be relevant. Table 6.6 is the main result of this thesis.

For a Jakob number of 5, Eq. (H.73) gives $\beta = 5.304$, Eqs. (6.59) and (6.60) give $t_{\text{ini}} = 5.98 \times 10^{-4}\,\text{s}$ and $t_{\text{fin}} = 2.39 \times 10^{-3}\,\text{s}$, and Eq. (H.66) gives $T_\infty = 538\,\text{K}$. Figure 6.13 shows a two-dimensional slice of the coarsest grid used to validate this test case. Figure 6.14 shows the relative error on the final bubble radius for the three grids used. The final bubble radius converges with grid refinement. On the finest grid, the relative error of the L^∞-norm of the final bubble radius is equal to 0.6%, as reported in Table 6.6. A deeper analysis is presented in Fig. 6.15 which shows the error on the x-component of the interface normal vector and the relative error on the interface curvature at final time. Neither the final interface normal vector nor the final interface curvature converge with grid refinement. We hypothesize that the convergence rate of the bubble radius is not sufficient in order for the normal vector and curvature, respectively first- and second-order derivatives of the signed distance function, to converge at final time with grid refinement. Nevertheless, one has to notice that the overall high accuracy of the method is well established thanks to the convergence of the bubble radius. Indeed, since this convergence is observed at final time, despite non-converging, the interface normal vector and curvature recomputed and used at each iteration of the simulation have permitted to obtain an accurate final bubble radius. Figure 6.16 shows the initial interface and the final computed and theoretical interfaces for the three grids. Again, one can see that on the finest grid, the interface at final time is very close to the theoretical one (Subfig. 6.16(e)). Figure 6.17 shows the liquid temperature at final time on the finest grid. As expected, the liquid temperature is very close to T_∞ in all the liquid phase, except in a region close to the interface, the thermal boundary layer, where a steep gradient responsible for the interface movement is observed. In this area, the liquid temperature decreases from $T_\infty = 538\,\text{K}$ in the liquid phase to $T_{\text{sat}} = 373\,\text{K}$ at the interface, showing that the immersed Dirichlet boundary condition $T_\Gamma = T_{\text{sat}}$ is satisfied. Figure 6.18 shows the liquid velocity field at final time on the finest grid. Also as expected, the liquid is ejected from the interface and the liquid velocity field is thus aligned with the interface normal vector. The velocity jump at the interface due to phase change is responsible for the liquid motion which can exit the domain thanks to the outlet boundary conditions used on all six faces of the cubic domain.

Thanks to the important improvements on the computation of the signed distance function made in this thesis, the Boiling solver is now able to maintain the spherical shape of the bubble throughout the simulation. Since this test case is very severe relatively to the accuracy of the interface capturing or tracking method used, especially on unstructured grids, we believe that the present results have a strong interest for further simulations of two-phase flows with phase change on unstructured grids. These simulations have been performed using the Geometric Marker Method (see Section 5.9.3)

to reinitialize the signed distance function to the interface at each iteration. Similar results are obtained when the signed distance function is reinitialized using the parallel implementation of the Hamilton-Jacobi equation (see Section 5.9.1).

6.7 Conclusion

In this chapter, the numerical method implemented in the Boiling solver for simulations of two-phase flows with phase change in multidimensions has been extended to take into account the computation of the mass transfer rate at the interface from the thermal fluxes on both sides of the interface. To this purpose, the heat equation is solved in both phases and two temperature fields are used to take into account the discontinuity of the heat flux at the interface. An immersed Dirichlet boundary condition is imposed at the interface in order to ensure that boiling always occurs at saturation temperature. The mass transfer rate is computed using a high-order framework to extrapolate the liquid temperature gradient from the liquid phase to the interface, and the vapor temperature gradient from the vapor phase to the interface, in order to compute the mass transfer rate precisely at the interface. The overall implementation has been validated against the test case of a bubble at rest growing in a superheated liquid, in two and three dimensions. The simulations have been performed until the physical time needed for the bubble radius to double. Excellent agreement with the theoretical bubble radius has been obtained at final time on two- and three-dimensional unstructured grids. Moreover, the bubble radius converges with grid refinement in all cases. In three dimensions, the relative L^∞-norm of the error on the bubble radius is below 3% on the finest grid considered. These results demonstrate the ability of the Boiling solver to accurately simulate two-phase flows with phase change where the mass transfer rate is computed from the thermal fluxes at the interface, thus taking a step towards realistic boiling simulations in industrial context.

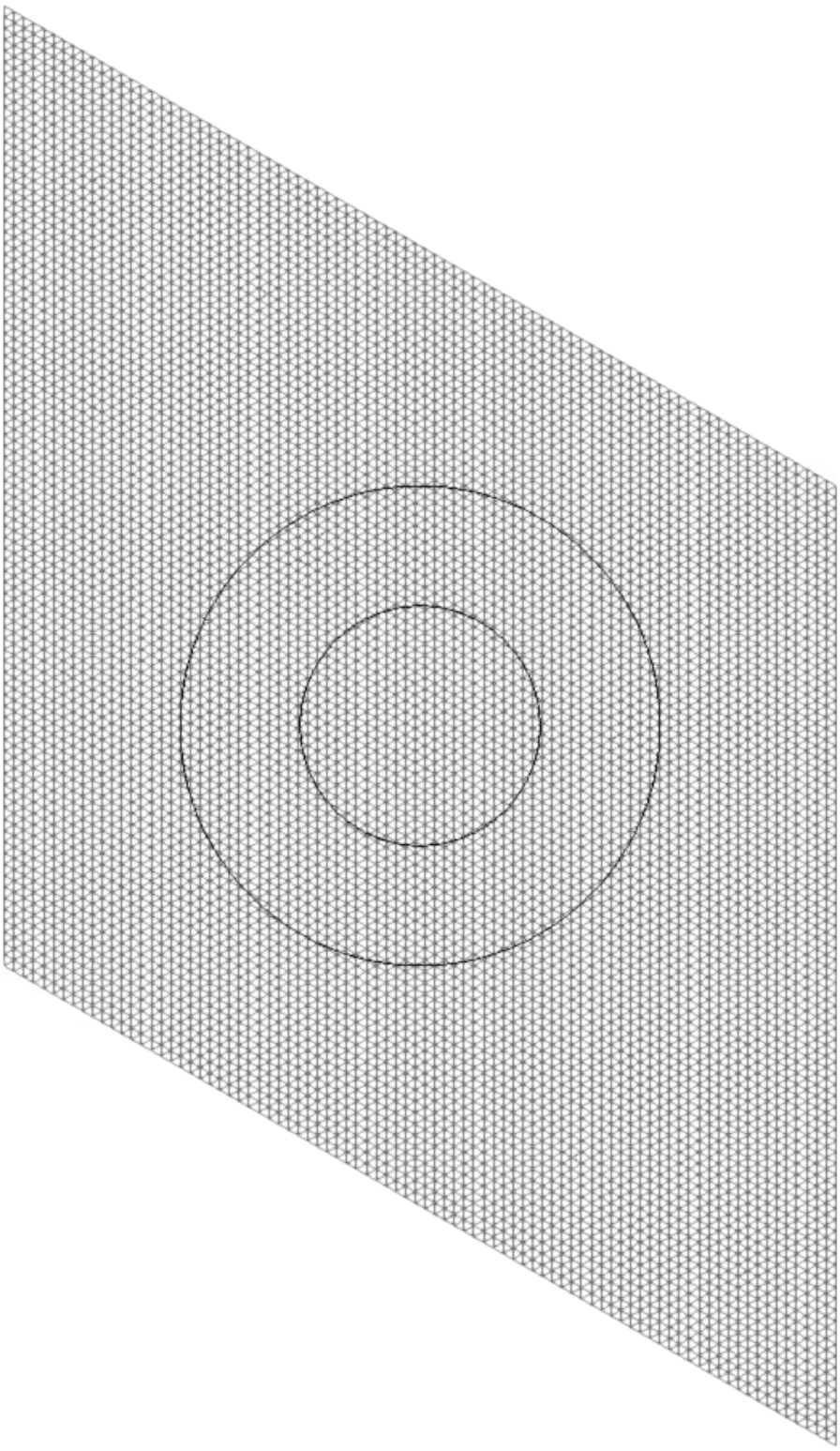
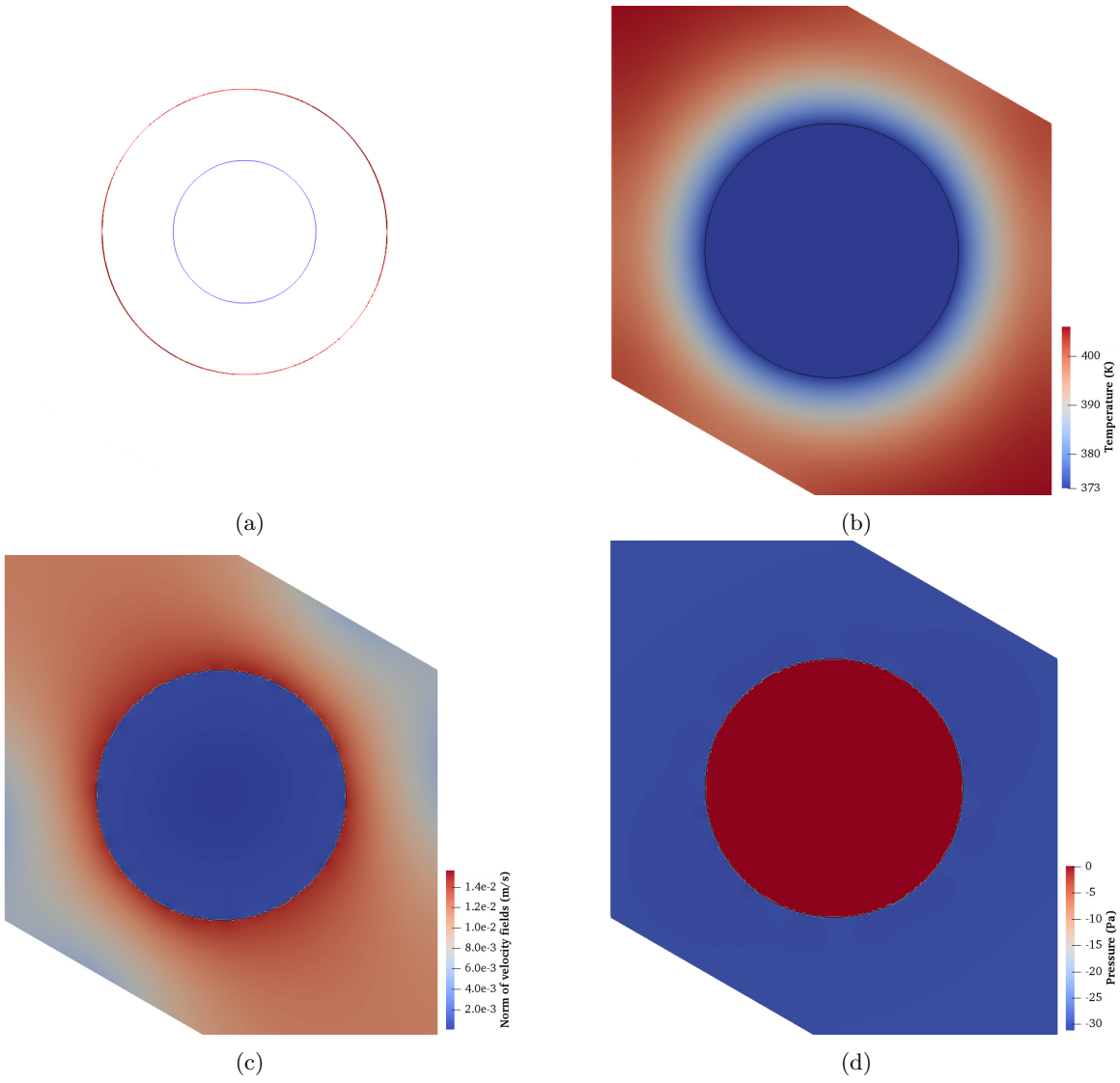


Fig. 6.5 – Coarsest two-dimensional grid ($\Delta = 9.375 \times 10^{-5}\,\mathrm{m}$) composed only of equilateral triangles used is our simulations. The small circle represents the initial interface of radius $R_0 = 1 \times 10^{-3}\,\mathrm{m}$, whereas the bigger circle represents the final theoretical interface (the radius is twice the initial one).



(a) (b) (c) (d)

Fig. 6.6 – Numerical results for the two-dimensional test case on the finest grid (equilateral triangles). On (a), the initial interface is shown in blue, the final computed interface, in black, and the final theoretical interface in red. The final liquid and vapor temperature fields are shown on (b), the final velocity fields are shown on (c) and the final pressure field is shown on (d).

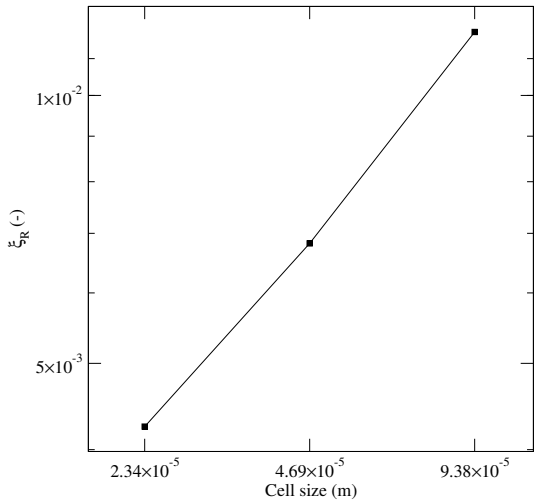
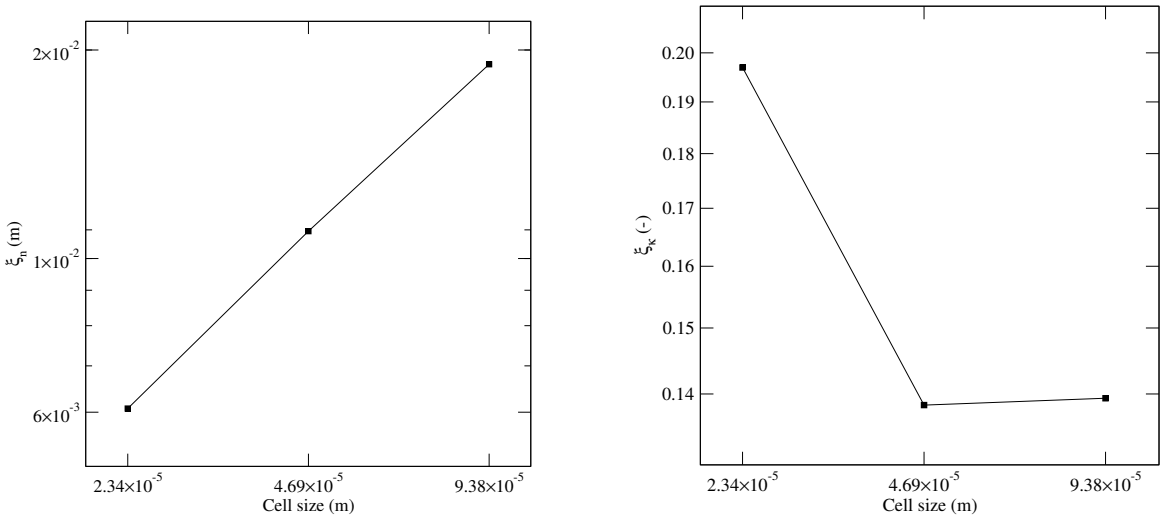


Fig. 6.7 – Relative error on the final bubble radius for the two-dimensional simulations on equilateral triangles presented in Section 6.6.1.1.



(a) (b)

Fig. 6.8 – Error on (a) the final x-component of the interface normal vector and relative error on (b) the final interface curvature for the two-dimensional simulations on equilateral triangles presented in Section 6.6.1.1.

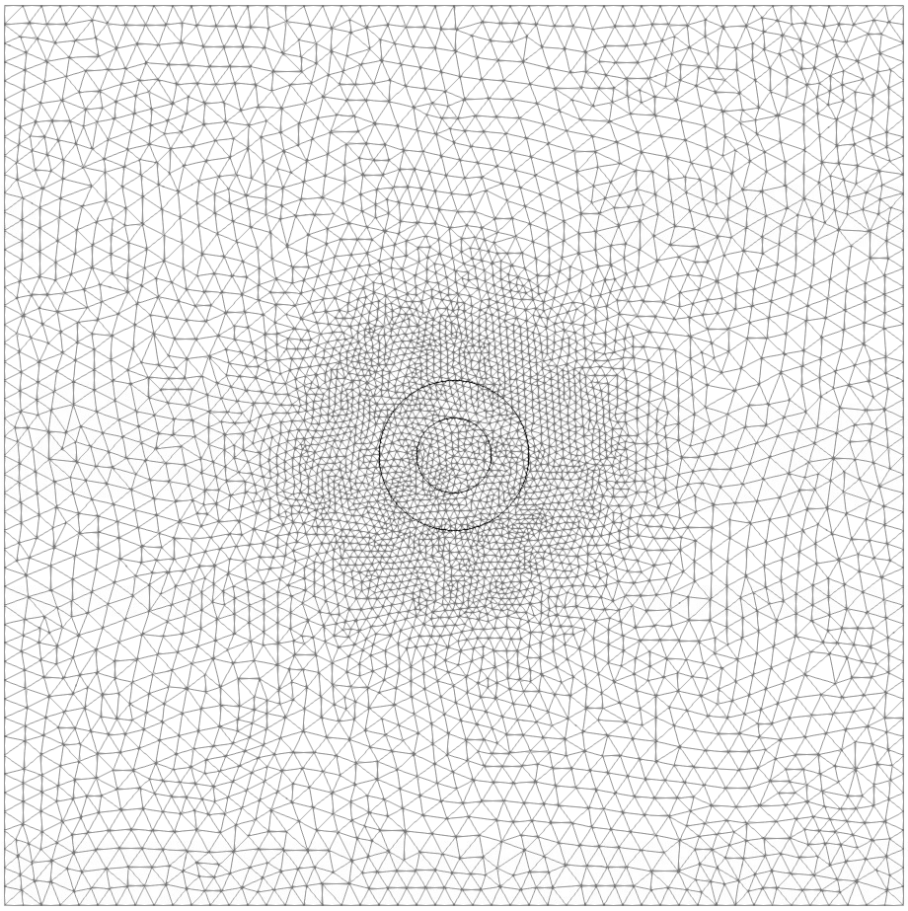
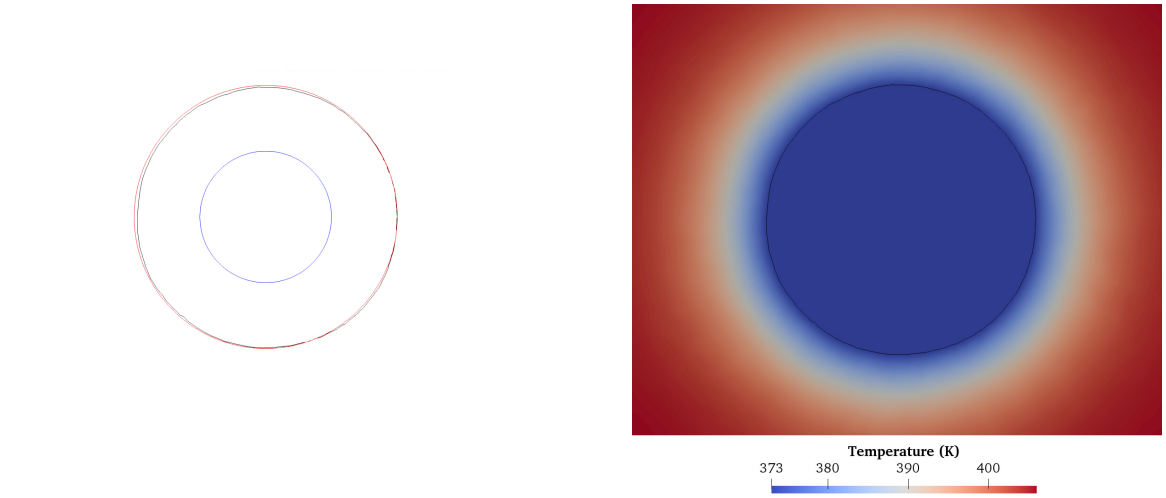
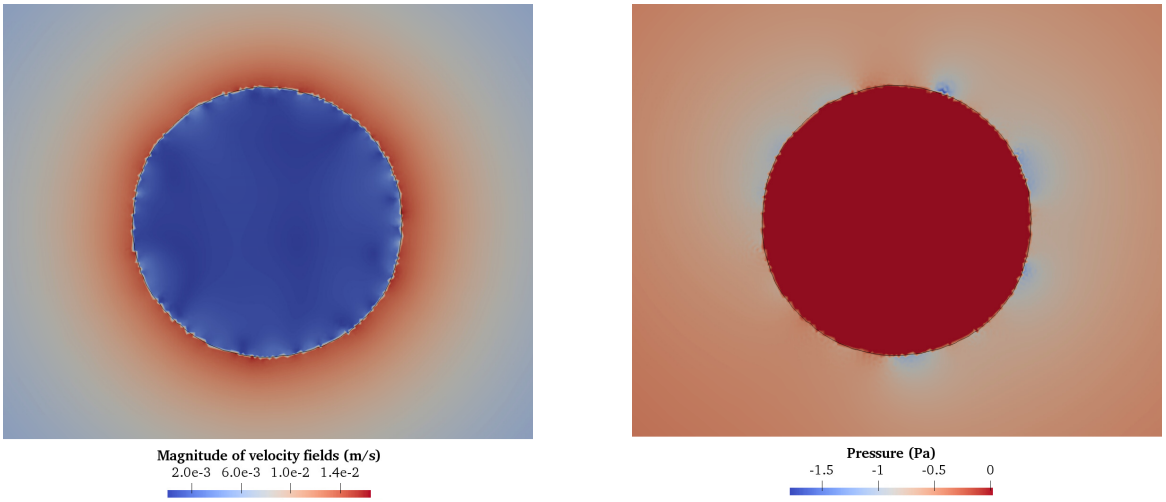


Fig. 6.9 – Coarsest unstructured grid ($\Delta = 1.875 \times 10^{-4}\,\mathrm{m}$) used to perform the test case in two dimensions presented in Section 6.6.1.2. The grid is refined in the area in which the interface will evolve. The small circle represents the initial interface location of radius $R_0 = 1 \times 10^{-3}\,\mathrm{m}$, the bigger circle represents the final theoretical interface location (the radius is twice the initial one).



(a) (b)



(c) (d)

Fig. 6.10 – Numerical results for the two-dimensional test case on the finest grid (unstructured triangles). On (a), the initial interface is shown in blue, the final computed interface, in black, and the final theoretical interface in red. The final liquid and vapor temperature fields are shown on (b), the final velocity fields are shown on (c) and the final pressure field is shown on (d).

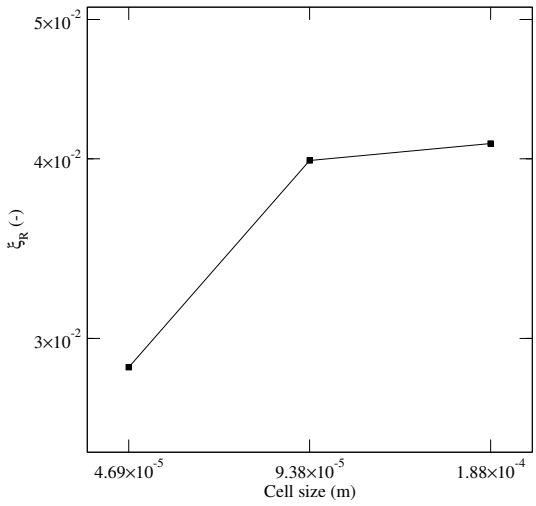


Fig. 6.11 – Relative error on the final bubble radius for the two-dimensional simulations on arbitrary triangles presented in Section 6.6.1.2.

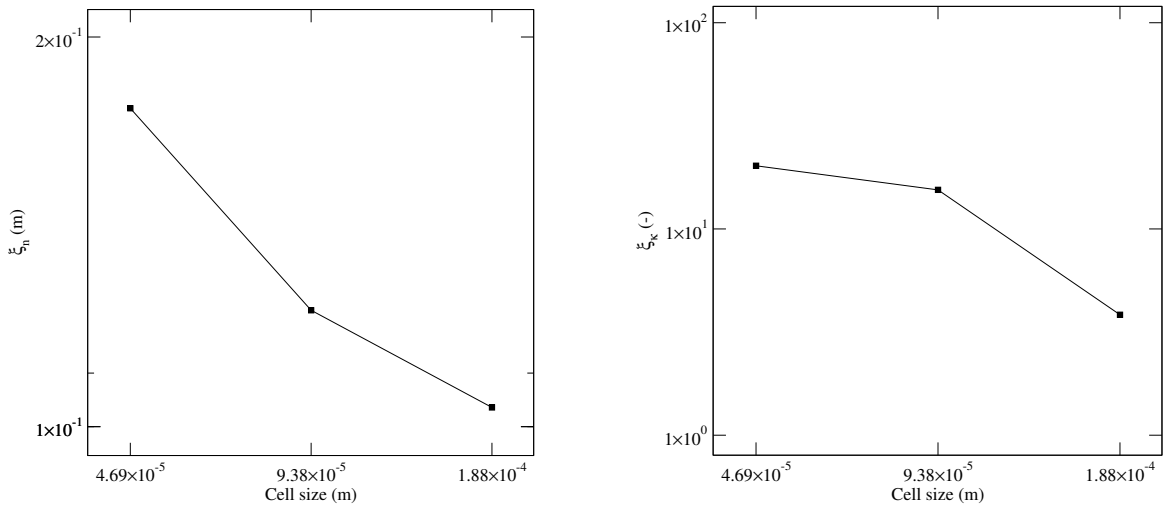


Fig. 6.12 – Error on (a) the final x-component of the interface normal vector and relative error on (b) the final interface curvature for the two-dimensional simulations on arbitrary triangles presented in Section 6.6.1.2.

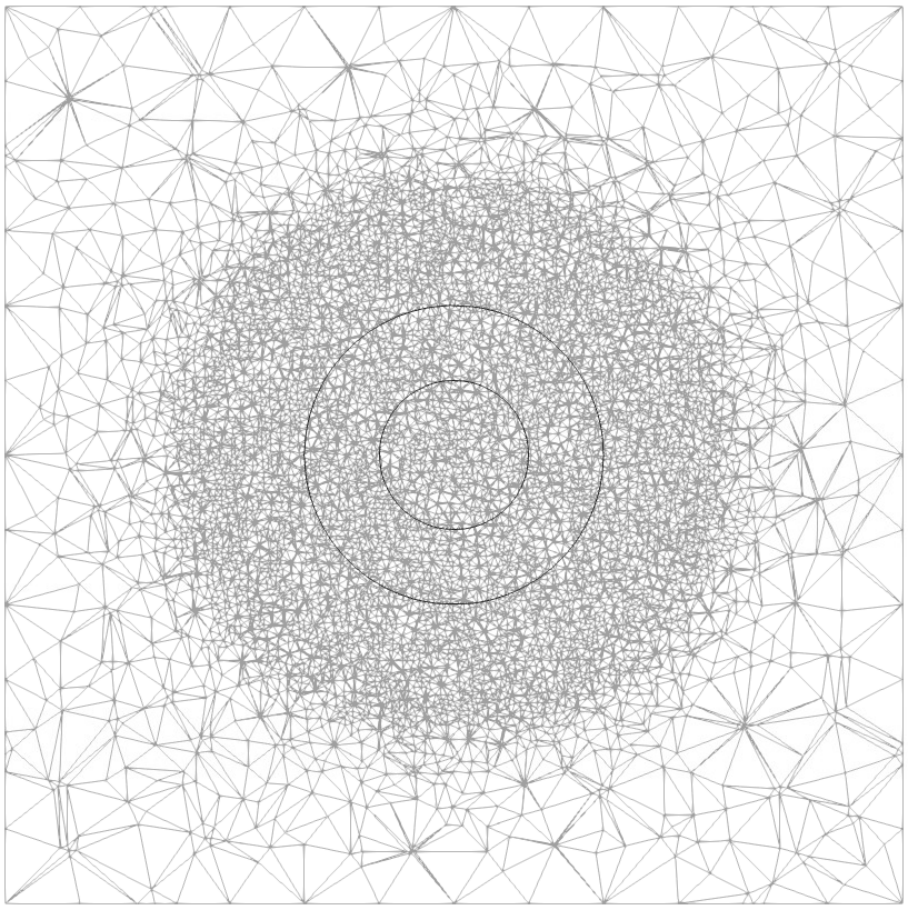


Fig. 6.13 – Refined three-dimensional cubic grid used in our simulations. The small circle represents the initial interface location of radius $R_0 = 1 \times 10^{-3}$ m, the bigger circle represents the final theoretical interface location (the radius is twice the initial one). The grid is refined only in an area englobing the narrow band around the interface in order to speed up computations. This grid has a characteristic cell size around the interface of $\Delta = 1.5 \times 10^{-4}$ m and counts 554 800 grid cells. This is the coarsest grid used in our tests. Outlet boundary conditions are applied on all six faces of the domain in order to permit the evacuation of the liquid.

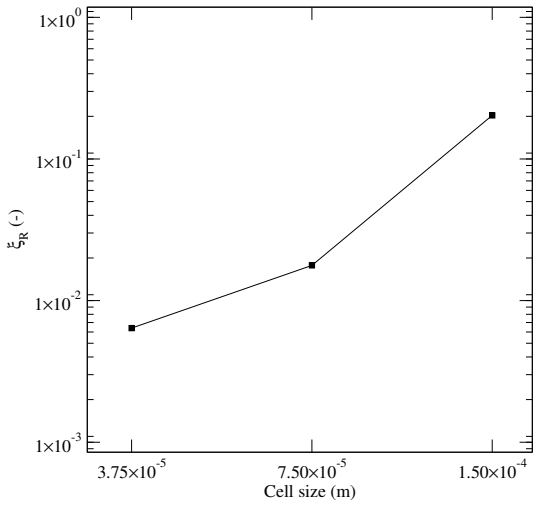


Fig. 6.14 – Relative error on the final bubble radius for the three-dimensional simulations on unstructured grids presented in Section 6.6.2 with a Jakob number of 5.

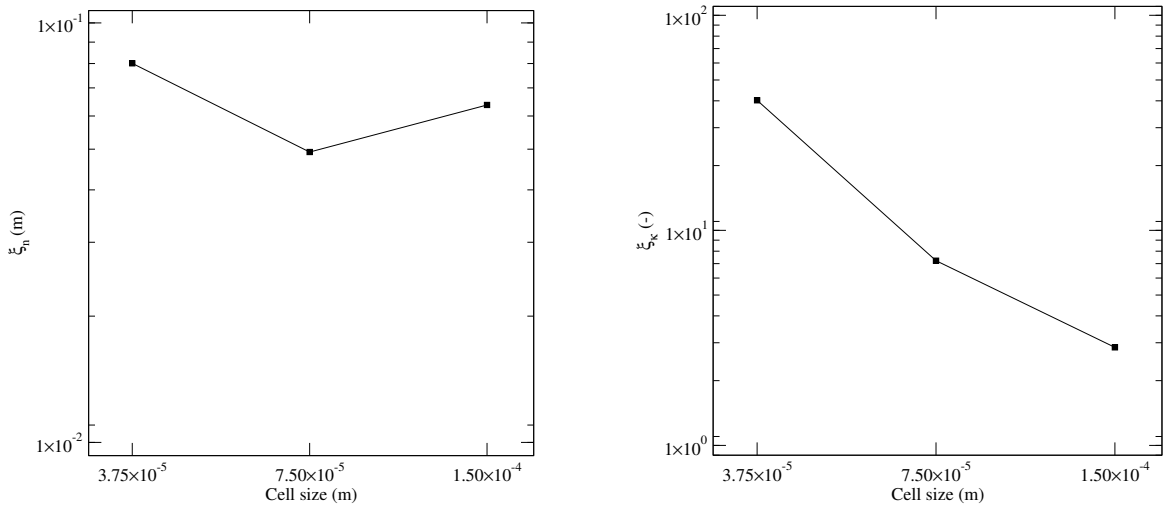
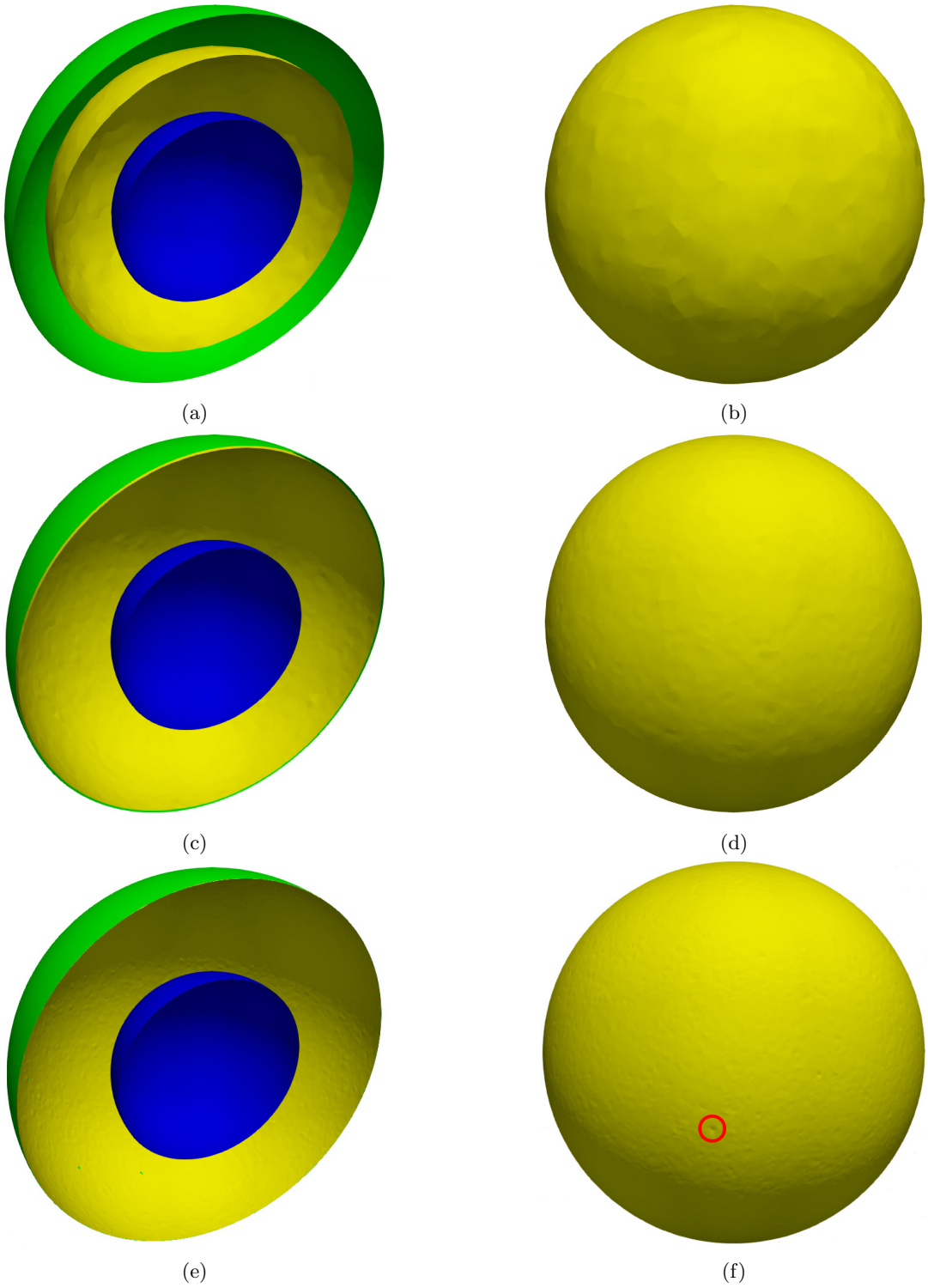


Fig. 6.15 – Error on (a) the final x-component of the interface normal vector and relative error on (b) the final interface curvature for the three-dimensional simulations on unstructured grids presented in Section 6.6.2 with a Jakob number of 5.



(a) (b) (c) (d) (e) (f)

Fig. 6.16 – The left column shows the initial interface in blue, the final computed interface in yellow and the final theoretical interface in green for the three unstructured grids. The right column shows the whole final computed interface. The simulations are shown in sorted lines by ascending order of grid refinement (coarsest grid on the first line, finest grid on the last line). Local errors in the interface position, as shown in red on (f), directly affect the value of the relative L^∞-error on the bubble radius.

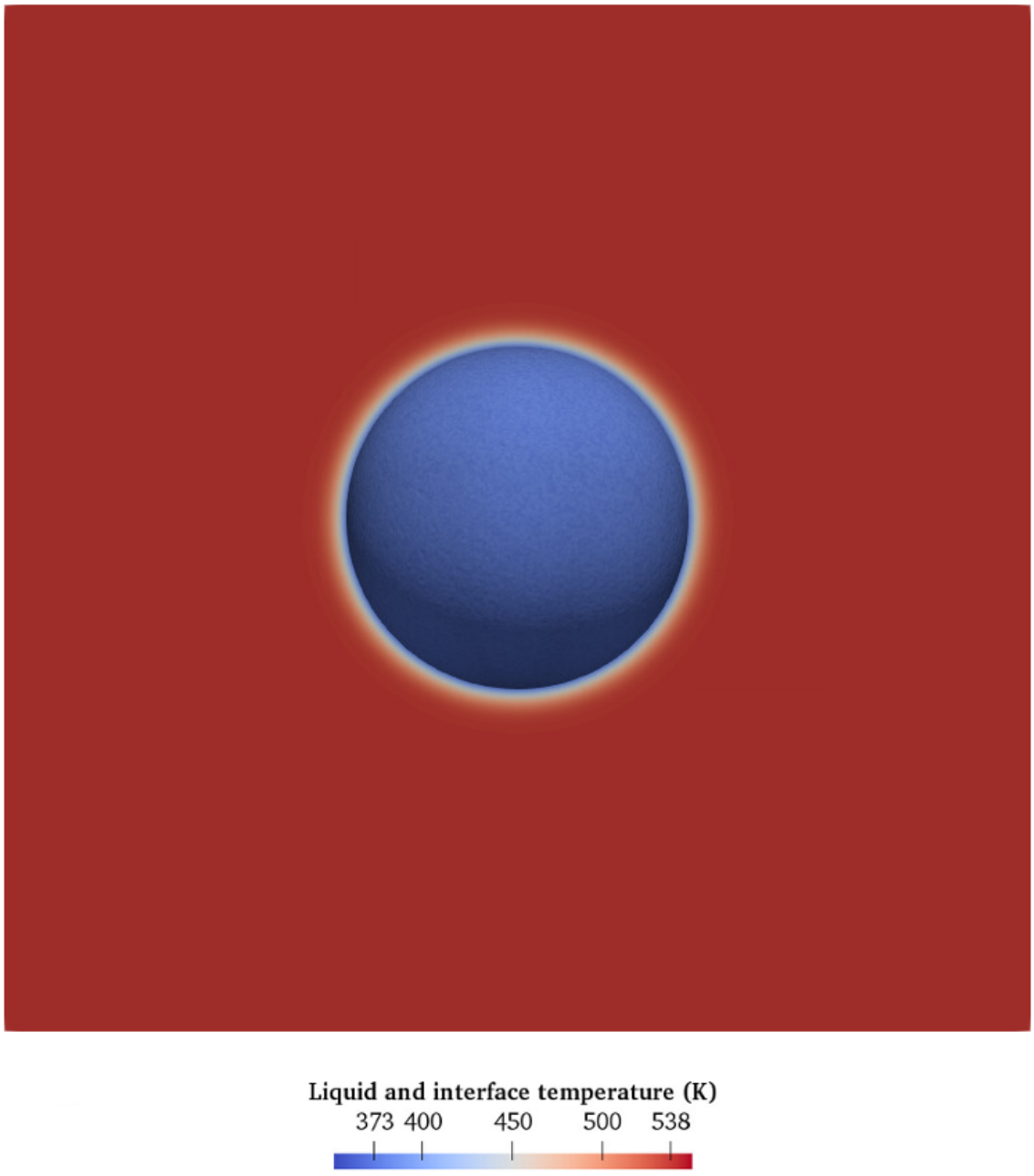


Fig. 6.17 – Liquid temperature field at final time on the finest unstructured grid, corresponding to Subfig. 6.16(f).

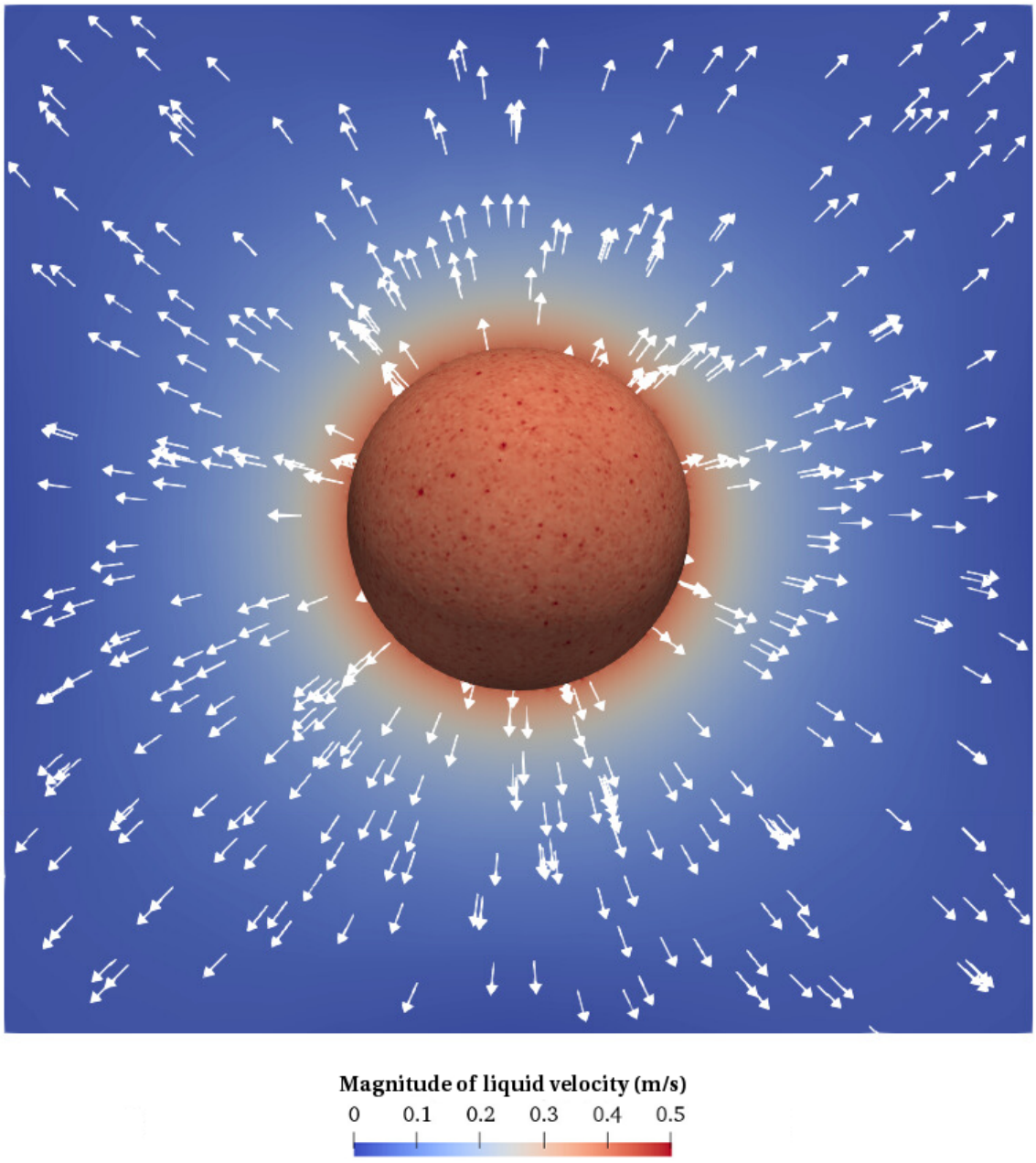


Fig. 6.18 – Liquid velocity field at final time on the finest unstructured grid, corresponding to Subfig. 6.16(f), where the vector field shows the direction of the velocity.

Chapter 7

Conclusion and perspectives

Outline

7.1 Conclusion

In this thesis, a numerical method for simulations of two-phase flows with phase change due to heat transfer (boiling) has been developed within the YALES2 code and validated in one, two and three dimensions.

Chapter 1 introduces the physics of boiling, emphasizing the coupling between the Navier-Stokes equations and the heat equation by means of the velocity and pressure discontinuities at the interface computed as functions of the mass transfer rate. Chapter 2 presents the state-of-the-art numerical methods dedicated to two-phase flow simulations. In Chapter 3, a detailed presentation of the YALES2 code is given, focusing on the Spray solver for two-phase flow simulations without phase change. The numerical developments realized in this thesis to take phase change into account, i.e. the Boiling solver implemented in YALES2, have been presented in Chapters 4, 5 and 6.

Chapter 4 introduces the overall numerical strategy implemented in the Boiling solver for simulations in one dimension. The restriction to one dimension permits to focus on the coupling between the velocity and temperature fields without striving with difficulties involved with a curved interface, namely the computation of the interface normal vector and curvature, needed to compute the velocity and pressure discontinuities at the interface. Performing extrapolations across the interface in one dimension is trivial since the interface normal vector is aligned with the unique spatial dimension. The ability of the Boiling solver to simulate the interface motion due to phase change has been demonstrated in this chapter.

Chapter 5 extends the numerical method to two and three dimensions, starting from the simplified test case of a uniform and constant mass transfer rate not coupled with the temperature field.

The computation of the interface normal vector and curvature from the signed distance function to the interface (the Level Set function) has been largely studied and improved. It has been shown that inaccuracy on the computation of these quantities leads to large numerical instabilities. The state-of-the-art numerical methods to reinitialize the signed distance function to the interface, the Hamilton-Jacobi equation and the Fast Marching Method, have then been implemented and validated on cartesian and unstructured grids against the test case of a two-dimensional bubble growing with an imposed mass transfer rate inside a liquid at rest. Similarly, a state-of-the-art reinitialization methodology for the conservative level set (that has a hyperbolic tangent profile, designed to improve mass conservation) has been implemented and validated on the same test case. The use of the signed distance function or of the conservative level set function both led to a first-order accurate final bubble radius with grid refinement, where the errors are computed at the physical time needed to double the initial bubble radius. For all methodologies presented, the relative error on the final bubble radius is below 1% on the finest grid considered. The simulation has been extended to three dimensions, and the final bubble radius also converges at order one with grid refinement. In three dimensions, the relative error on the final bubble radius is equal to 1.6% on the finest grid. These errors have been computed as the L^∞-norm of the local error normalized by the corresponding theoretical radius. One should stress that the L^∞-norm is the most severe computation method of errors, taking into account the highest local error without any averaging.

Chapter 6 further extends the numerical method of the Boiling solver by introducing the coupling of the mass transfer rate at the interface with the thermal fluxes across the interface. The heat equation is then solved in both liquid and vapor phases, and an immersed Dirichlet boundary condition is used to impose the fluid saturation temperature to the interface at a subgrid level. In order to accurately compute the mass transfer rate from the thermal fluxes at the interface, which represents the thermal coupling of boiling simulations, a new framework has been used to perform high-order extrapolations of temperature gradients precisely at the interface. The overall methodology has been validated against the test case of a three-dimensional bubble growing in a superheated liquid at rest for different Jakob numbers, the heat flux in the liquid side of the interface being responsible for the bubble expansion. Again, the simulation has been performed until the doubling of the bubble radius. Convergence in relative L^∞-norm of the final bubble radius with grid refinement is obtained. Also, the errors decrease with the Jakob number, in accordance with previous results found in the literature for the same test case on two-dimensional axisymmetric cartesian grids. The Boiling solver has thus successfully extended the ability of performing L^∞-convergent simulations on two- and three-dimensional unstructured grids.

To the best of my knowledge, this thesis has provided the first numerical simulations of bubble growth in three dimensions with a bubble radius converging with grid refinement in the L^∞-norm sense. The L^∞ convergence of the radius implies that the position of the bubble surface converges to the exact solution at every surface point. It is then considered as an important ingredient for predictive numerical simulations.

7.2 Perspectives

The results obtained in this thesis open the path to accurate numerical simulations of two-phase flows with phase change on three-dimensional unstructured grids. However, one should note that not all the terms present in the differential equations show an L^∞ convergence in our simulations. Indeed, the normal vector, present in Eqs. (1.26), (1.30), (1.31) and (2.12) in the definitions of the mass transfer rate, the velocity and pressure discontinuities at the interface and the interface

curvature, respectively, does not show a decrease of the error with grid refinement (Fig. 6.15(a)). Worse, the interface curvature diverges with grid refinement (Fig. 6.15(b)). On a highly refined grid, this could lead to the failure of the simulation since an arbitrary elevated curvature value would be used in the computation of the pressure discontinuity at the interface. Convergence in L^∞-norm of the interface curvature is probably necessary in order to maintain the accuracy of the simulation for an arbitrary grid refinement.

The natural extension of this work is the simulation of nucleate boiling, a mode of heat transfer widely used in industrial applications, occurring when a liquid is in contact with a solid whose temperature is above the liquid boiling point, leading to the formation of vapor bubbles on the solid surface. Numerical simulations of nucleate boiling require a methodology to take the motion of the contact line (where the solid, liquid and vapor phases meet) into account, as well as the contact angle existing between the solid surface and the liquid-vapor interface. The numerical simulation of nucleate boiling on unstructured grids is currently under development at LEGI in the MoST team, as an extension of the Boiling solver. The inclusion of the contact line and contact angle in the Boiling solver is expected to enable predictive three-dimensional direct numerical simulations of nucleate boiling at the bubble scale in the coming years.

Appendix A

Introduction to nucleate boiling

Outline

A.1 Contact line dynamics without phase change

When a liquid-vapor interface at rest is in contact with a solid substrate, the differences of physico-chemical properties between the three phases in presence (solid, liquid, vapor) entail interactions which manifest through an energetic equilibrium. The region where the three phases are in contact is called contact line. The aforementioned energetic equilibrium forces the interface to form an angle with the substrate. This angle is generally evaluated in the liquid phase and called contact angle. The contact line and the contact angle have been the subject of multiple studies from the XIX$^{\text{th}}$ century to nowadays. Numerous authors have done experiments and have thus contributed to the understanding of the physics implied in the dynamics of contact line. Thereafter, analytical models have been developed and numerical studies have been carried out. Nevertheless, the knowledge on the subject is not sufficient yet, and still today, research is ongoing on the different spatial scales of such systems.

A.1.1 Static contact line

When a droplet or bubble lies on a solid surface, the shape of the interface, in the neighborhood of the contact line, is controlled by interactions between the different phases, and forms a contact angle θ, as shown in Fig. A.1. In the static case without phase change, there exists theoretical models enabling the computation of the contact angle, with respect to the surfacic energies generated by the discontinuity of density across the interface. One famous theoretical model is the Young-Dupré

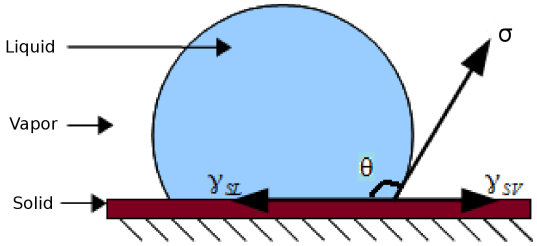


Fig. A.1 – Surfacic energies at contact angle θ formed by a liquid droplet lying on a solid surface, γ_{SL} between solid and liquid phases, γ_{SV} between solid and vapor phases and σ between liquid and vapor phases.

relation, given by

$$\sigma \cos(\theta) = \gamma_{SV} - \gamma_{SL}, \tag{A.1}$$

where γ_{SV}, γ_{SL} and σ represent the surfacic energies respectively of the interface between the solid and vapor, solid and liquid, and liquid and vapor phases, as indicated in Fig A.1. The fluid is said perfectly wetting when the contact angle is equal to zero. In this case, the Young-Dupré relation then becomes $\sigma = \gamma_{SV} - \gamma_{SL}$. Equation (A.1) comes from the balance of forces by unit length acting on the contact line when the three phases are in thermodynamical equilibrium determined by the physico-chemical properties of the three phases.

These properties determine the shape (the contact angle) of a droplet lying on a surface. The tendency of the droplet to spread along the surface is called wetting. Depending on the contact angle, two types of wetting can be observed : total wetting (the liquid phase recovers the whole solid phase, i.e. the vapor phase is not in contact with the solid phase, so there is no contact line) and partial wetting ($0 < \theta < \pi$). These types of wetting are characterized by the spreading parameter S measuring the difference between the surfacic energy of the dry and wet substrate, and defined as

$$S = E_{\text{dry substrate}} - E_{\text{wet substrate}} \tag{A.2}$$

$$= \gamma_{SV} - (\gamma_{SL} + \sigma)\,. \tag{A.3}$$

The value of S gives the type of wetting :

- If $S < 0$, the regime is called partial wetting. The liquid does not spread and forms a spherical cap lying on the substrate with a contact angle at equilibrium θ_e.
- If $S > 0$, the regime is called total wetting. The liquid entirely spreads over the solid in a film of nanoscopic width to lower the interfacial energy γ_{SV}.

Figure A.2 illustrates the two types of wetting according to the spreading parameter S.

A.1.2 Dynamics of contact line

The analysis and modeling of contact angle dynamics has been highly challenging for scientists. Although wetting dynamics plays a role in numerous processes, the understanding of the different physical phenomena implied during the spreading of a liquid droplet over a solid surface is still far

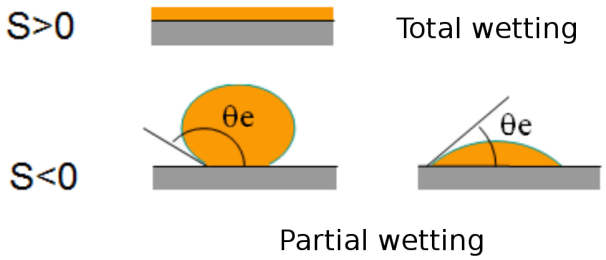


Fig. A.2 – Types of wetting defined by the spreading parameter S.

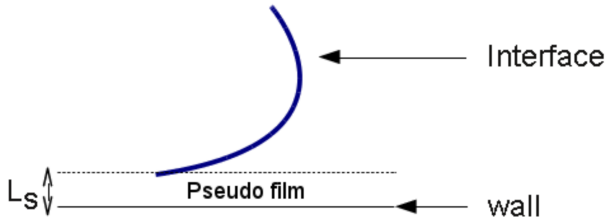


Fig. A.3 – Navier condition : the width L_s of the fictitious film is equivalent to the slipping length of the contact line.

from being complete. Many reasons can explain this lack of understanding. First, the physical phenomena controlling the contact line dynamics intervene at several scales, from microscopic lengths (where molecular mechanics is dominant) to macroscopic lengths (where continuum mechanics is dominant). Nevertheless, observations realized experimentally on velocity or contact angle, do not tackle length scales below a decade of micrometers. In addition to the difficulties encountered to precisely examine phenomena around the contact line, classical boundary conditions applied to Navier-Stokes equations in the presence of a solid surface lead to the appearance of a singularity of viscous constraints.

At macroscopic scales, the behavior of a fluid close to the solid surface is usually described by a no-slip condition. While this condition is often satisfactory in the case of continuous monophasic flows, in other situations, e.g. for rarefied gases, it reveals inadapted. Similarly, in the case of contact lines, the no-slip condition is in contradiction with the hypothesis of contact line displacement. Indeed, a singularity on the viscous constraint due to the no-slip condition has been exposed by Huh and Scriven [37], demonstrating that a gradient of viscous constraints close to a liquid meniscus diverges close to the contact line [70]. In the case of partially wetting droplets, the Navier condition, based on the assumption of the existence of a fictitious liquid film of given width on which the droplet slips, as shown in Fig. A.3, can be used to alleviate the adherence condition of the contact line to the solid surface. Other techniques have been developed for perfectly wetting fluids (see again [70] and the book of de Gennes et al. [18] for more details).

The contact line motion is usually described using its velocity and the contact angle between the liquid-vapor and solid-vapor interfaces. The theoretical approaches developed to describe the hydrodynamics in the neighborhood of the contact line are multi-scale models. The region close to the contact line is thus described at three different scales illustrated in Fig. A.4, the macroscopic,

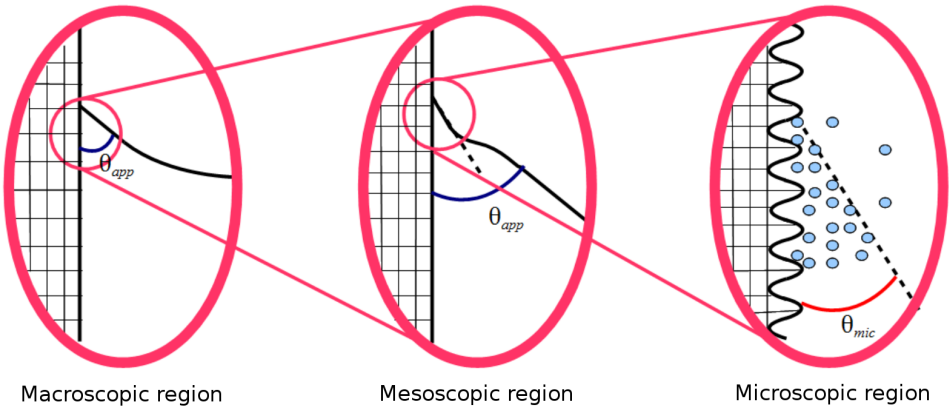


Fig. A.4 – The contact line is described at three different scales : the macroscopic, mesoscopic and microscopic scales. For each scale, a contact angle can be defined [70].

mesoscopic and microscopic scales. One can notice an important element for the study of contact line motion : the macroscopic contact angle differs from the microscopic contact angle.

At macroscopic scale, the flow motion is described by the Navier-Stokes equations, and the apparent contact angle can be more easily evaluated considering the shape of the whole bubble.

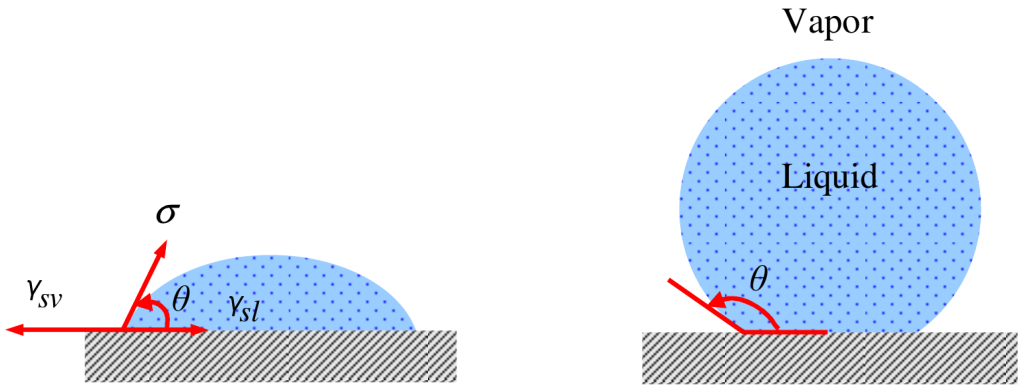
At mesoscopic scale, the liquid-vapor interface in the immediate neighborhood of the contact line is of null curvature. An apparent contact angle can then be defined. Surface tension and viscosity effects compete to reshape the liquid-vapor interface.

At microscopic scale, molecular interactions (e.g. Van der Waals forces) are dominant. The contact angle θ in Eq. (A.1) and the slipping length mentioned above are defined at this scale.

A.1.3 Wettability of hydrophilic and hydrophobic surfaces

If a liquid drop is small enough to neglect the flattening action of gravity, its sticking on a solid surface is referred to as the surface wettability. An important parameter used to characterize the surface wettability is the contact angle θ made by the liquid on the solid. The solid surface is said wetted if $\theta < \pi/2$ (see Fig. A.5(a)) and is said unwetted if $\theta \geq \pi/2$ (cf. Fig. A.5(b)). In the case of water, the wetted surface is called hydrophilic and the unwetted surface is called hydrophobic.

Realistic surfaces are alternatively composed of hydrophilic and hydrophobic regions. In the context of nucleate boiling, hydrophobic regions favor the nucleation of bubbles in cavities since the liquid is repelled by the solid, enabling the vapor phase to develop in the cavities of the solid, but tend to slow down the bubble departure. Conversely, hydrophilic regions favor the departure of bubbles from the nucleation sites by gravity effect since the liquid is attracted by the solid in the region close to the contact line, but slow down the nucleation of bubbles in the cavities of the solid since the liquid easily fills the cavities due to the attraction of the solid phase. In order to optimize the cooling of the solid material, i.e. the heat transfert from the solid to the liquid and vapor phases, it can be useful to find a compromise between the pros and cons in the repartition of hydrophilic and hydrophobic surface regions.



(a) (b)

Fig. A.5 – Surface wettability : (a) wetted or hydrophilic surface ($\theta < \pi/2$), (b) unwetted or hydrophobic surface ($\theta \geq \pi/2$).

A.2 Contact line dynamics with phase change : nucleate boiling

In this section, we introduce the contribution of boiling to the contact line dynamics. Consider a liquid phase on top of a solid phase. If the solid temperature is higher than the liquid temperature, a thermal flux is directed from the solid to the liquid phase. Due to this heat flux, if the liquid temperature reaches the liquid saturation temperature, then molecules gain sufficient kinetic energy to escape the attraction of their neighbors. As a result, the liquid in the concerned area turns into vapor. This physical phenomenon usually occurs at the contact area between the liquid and solid phases, and is called nucleate boiling.

The nucleation of vapor bubbles demands the creation of surface, that is accompanied by a certain energy proportional to surface tension. Nucleation begins therefore in sites that offer the lowest surface per bubble volume, and then the lowest energy barrier, that is in cavities of the solid. Due to the heat flux originating from the solid, a small vapor bubble forms in a cavity. This step is called nucleation and the cavity is called nucleation site. Once nucleation is initiated, the bubble spontaneously grows and then detaches from the solid surface. The complete process of liquid heating, nucleation, bubble growth and release, collectively refers to as the boiling cycle. Three main features of this process that affect the rate of heat transfer are the bubble departure diameter D_d, the bubble emission frequency f and the number of active nucleation sites N_{as}. The bubble emission frequency f at a nucleation site is defined as

$$f = \frac{1}{\tau_{\text{gt}} + \tau_{\text{wt}}}, \tag{A.4}$$

where the growth time τ_{gt} is the duration of the bubble growth and the waiting time τ_{wt} is the duration between the departure of the former bubble and the appearance of the current bubble, as illustrated in Fig. A.6. Over the past eighty years, the bubble departure diameter during nucleate boiling has been the subject of numerous investigations [25]. In experimental studies, it is typically determined from high-speed videos of boiling process. Based on experimental data, a number of correlations were suggested to estimate the bubble departure diameter. Many correlations reflect

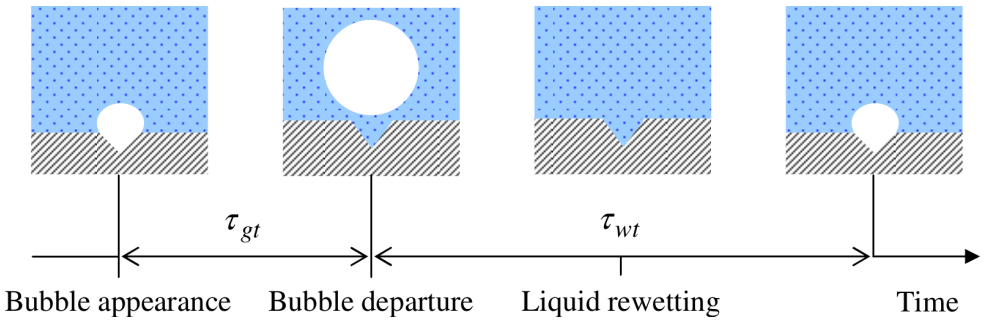


Fig. A.6 – The four steps of the boiling bycle [31].

the role of the capillary length L_c, defined as

$$L_c = \left(\frac{\sigma}{g\left(\rho_{\text{liq}} - \rho_{\text{vap}}\right)} \right)^{\frac{1}{2}}, \tag{A.5}$$

where g is the gravity. Indeed, the capillary length accounts for a simple balance of surface tension force and buoyancy. We only mention here the correlation of Fritz [25] since it takes into account the effect of the surface wettability :

$$D_d = 0.0208 \times \theta \times L_c, \tag{A.6}$$

where θ is given in degrees. Several studies [15, 100, 33, 38] show that the frequency of bubble emission is inversely proportional to the departure diameter. Based on an analogy between the bubble release process and natural convection, Zuber [100] suggested the following relation,

$$f \times D_d = 0.59 \left(\frac{\sigma g \left(\rho_{\text{liq}} - \rho_{\text{vap}}\right)}{\rho_{\text{liq}}^2} \right)^{\frac{1}{4}}. \tag{A.7}$$

The density of active nucleation sites was first estimated by Wang and Dhir [93]. The authors performed experiments using copper heaters with different degrees of oxidation, which resulted in different contact angles. The correlation proposed is given by

$$N_{\text{as}} = N_{\text{ms}} \left(1 - \cos\theta\right) \left(T_w - T_{\text{sat}}\right)^6, \tag{A.8}$$

where N_{ms} is the number of micro-cavities on the heated surface of interest and T_w is the temperature of the solid wall. These classical correlations (Eqs. (A.6), (A.7) and (A.8)) have been widely used as predictive tools. However, they are based on a limited quantity of experimental data and their accuracy has not been extensively verified. Thus, they should be treated as being approximate only.

A.2.1 Pool boiling regimes, critical heat flux and the boiling crisis

The study of boiling is usually divided in two categories : flow boiling and pool boiling. Flow boiling addresses boiling occurring in flows with an overall liquid bulk velocity, whereas pool boiling

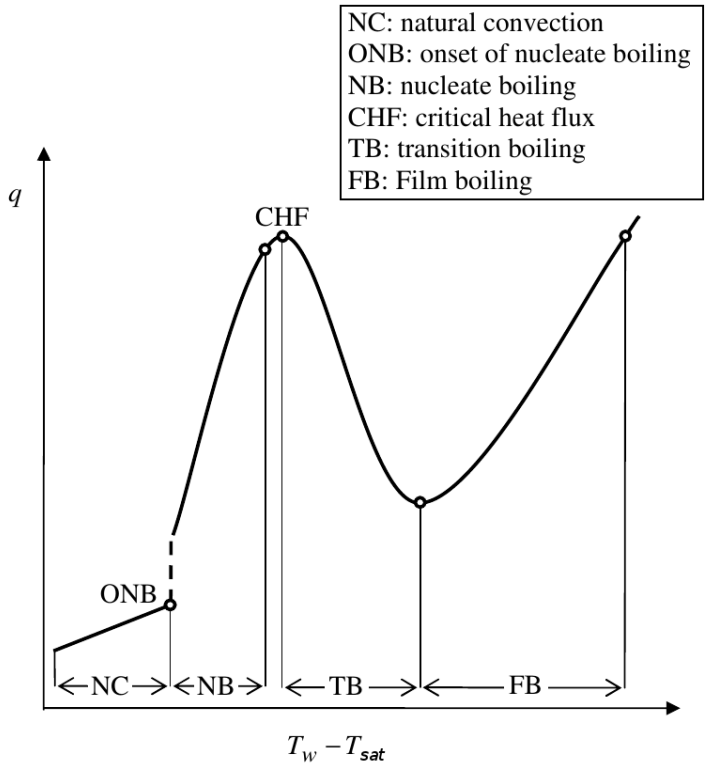


Fig. A.7 – Pool boiling curve of Nukiyama [56] representing the imposed heat flux q at the wall with respect to the wall superheat $T_w - T_{\rm sat}$ [31].

addresses boiling in liquids at rest. In this thesis, we focus on pool boiling since the bulk velocity in flow boiling also influences bubble growth. The different regimes of pool boiling are defined below. A parameter of interest in the study of nucleate pool boiling is the local heat transfer coefficient h given in $\mathrm{W\,m^{-2}\,K^{-1}}$ by

$$h = \frac{q}{T_w - T_{\rm sat}}, \tag{A.9}$$

where q is the imposed heat flux at the wall and $T_w - T_{\rm sat}$ is the wall superheat. The regimes of pool boiling heat transfer are easily understood by referring to the pool boiling curve which is a plot of q versus $T_w - T_{\rm s}$ for the circumstances of interest. Nukiyama [56] is well known for his pool boiling curve based on results from experiment of boiling water at atmospheric pressure. The regimes of pool boiling encountered for a horizontal flat surface are indicated schematically in Fig. A.7. The discussion in this section is limited to pool boiling of wetting liquids. The boiling curve can be conveniently analysed in four different regimes, namely, the natural convection NC, the nucleate boiling NB, the transition boiling TB and the film boiling FB.

Natural convection regime (NC) Due to the temperature gradients, fluid motions are created, removing heat from the heated surface to the free liquid surface. The driving force of natural convection is the buoyancy force, a result of gradients of fluid density.

Nucleate boiling regime (NB) If the superheat (difference between the solid surface temperature and the fluid saturation temperature) is large enough, nucleation is initiated at some cavities on the surface. This stage is called the "onset of nucleate boiling" (ONB). At low wall superheat levels, nucleate boiling is characterized by formation of isolated bubbles. With increasing surface superheat, more and more nucleation sites become active, and the bubble frequency at each site generally increases.

Transition boiling regime (TB) Eventually the active sites are spaced so closely that bubbles from adjacent sites merge together. Bubble coalescence can occur in vertical or horizontal directions. If the superheat still increases, a vapor film is formed along the surface resulting from the replacement of liquid by vapor adjacent to the heated surface. Poor thermal conductivity of vapor phase suddenly decreases the efficiency of heat transfer. The heat flux at this condition is called the critical heat flux (CHF), which describes the thermal limit of boiling phenomenon. Indeed, the CHF is the maximum heat flux that can be transferred by boiling process. A higher heat flux (compared to the CHF) can lead to the burnout of the heated surface. Once the CHF is reached, a large fraction of the surface is covered by a vapor film, boiling becomes unstable and transition boiling occurs. The transition from nucleate boiling to film boiling is known as the boiling crisis. Beyond the CHF point, Fig. A.7 shows that the heat flux **q** decreases when the wall superheat increases, Eq. (A.9) thus states that the heat transfer coefficient decreases once the CHF is reached. As a result, the transition to film boiling is usually inevitable.

Film boiling regime (FB) A further increase in the heat flux causes an insulating film of vapor to fully cover the surface. The heat transfer coefficient decreases significantly, and the surface temperature can then exceed the fusion temperature of the solid material, causing its deterioration.

For nucleate boiling, Cooper [16] developed the following correlation to determine the heat transfer coefficient :

$$h = 55 p^{*0.12-0.2\log_{10} R_p} \left(-\log_{10} p^*\right)^{-0.55} q^{0.67} M^{-0.5}, \tag{A.10}$$

where $p^* = P/P_{\rm cr}$ is the reduced pressure with $P_{\rm cr}$ the critical pressure, R_p is the roughness given in micrometers as defined in German standard DIN 4762/1, q is the heat flux given in $\mathrm{W\,m^{-2}}$ and M is the molecular weight of the fluid given in $\mathrm{g\,mol^{-1}}$. In general, the correlation of Cooper gives a good tendency of the heat transfer coefficient versus the heat flux. However, it should not be used to determine the surface roughness which is an adjustable parameter of this correlation.

In order to avoid the boiling crisis, the physico-chemical properties of the solid surface are also taken into account.

Appendix B

Coupling between velocity and pressure in the Boiling solver

Abstract

This appendix contains the necessary developments to the expression of the discretized pressure laplacian as well as the pressure gradient of each phase used to compute ghost pressures, as they appear in the code of the Boiling solver. The balance equation for the pressure laplacian is also given in 1D.

Outline

We consider the part of a domain Ω represented in Figure B.1.

B.1 Generalities

For all scalar ϕ defined on Ω, we define the liquid gradient operator in p at order 2, denoted ∇_p^l, by

$$\nabla_p^l \phi_p^l := \frac{1}{\mathcal{V}_p} \sum_{i=1}^{N_p} \frac{\phi_p^l + \phi_{q_i}^l}{2} \mathbf{A}_{p,q_i}, \tag{B.1}$$

where N_p is the number of direct neighboring nodes of node p. Similarly, we define the liquid gradient operator on the pair $[p, q_i]$ at order 1, denoted ∇_{p,q_i}^l, by

$$\nabla_{p,q_i}^l \phi_{p,q_i}^l := \frac{\phi_{q_i}^l - \phi_p^l}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i}, \quad \mathbf{\Delta x}_{p,q_i} = \mathbf{x}_{q_i} - \mathbf{x}_p. \tag{B.2}$$

We express $\phi^l_{q_i}$ with a first-order Taylor expansion,

$$\phi^l_{q_i} = \phi^l_p + \mathbf{\Delta x}_{p,q_i} \cdot \nabla^l_p \phi^l_p. \tag{B.3}$$

From Eqs. (B.2) and (B.3) multiplied by $\mathbf{\Delta x}_{p,q_i}$, we have, at order 1 in p,

$$\left.\begin{aligned} \phi^l_{q_i} \mathbf{\Delta x}_{p,q_i} &= \phi^l_p \mathbf{\Delta x}_{p,q_i} + \|\mathbf{\Delta x}_{p,q_i}\|^2 \nabla^l_{p,q_i} \phi^l_{p,q_i} \\ \phi^l_{q_i} \mathbf{\Delta x}_{p,q_i} &= \phi^l_p \mathbf{\Delta x}_{p,q_i} + \left(\mathbf{\Delta x}_{p,q_i} \cdot \nabla^l_p \phi^l_p\right) \mathbf{\Delta x}_{p,q_i} \end{aligned}\right\} \Longrightarrow \nabla^l_{p,q_i} \phi^l_{p,q_i} = \frac{\mathbf{\Delta x}_{p,q_i} \cdot \nabla^l_p \phi^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i}, \tag{B.4}$$

the gradient of ϕ computed on the pair $[p, q_i]$ is then equal to the projection on the pair $[p, q_i]$ of the gradient of ϕ computed on all pairs. The scalar product of Eq. (B.4) and $\mathbf{\Delta x}_{p,q_i}$ gives

$$\nabla^l_{p,q_i} \phi^l_{p,q_i} \cdot \mathbf{\Delta x}_{p,q_i} = \nabla^l_p \phi^l_p \cdot \mathbf{\Delta x}_{p,q_i}. \tag{B.5}$$

Equation (B.3) then becomes

$$\phi^l_{q_i} = \phi^l_p + \mathbf{\Delta x}_{p,q_i} \cdot \nabla^l_{p,q_i} \phi^l_{p,q_i}. \tag{B.6}$$

The same approach on the vapor side leads to

$$\phi^g_{q_i} = \phi^g_p + \mathbf{\Delta x}_{p,q_i} \cdot \nabla^g_{p,q_i} \phi^g_{p,q_i}. \tag{B.7}$$

The difference (B.6) - (B.7) reads

$$[\phi]_{q_i} = [\phi]_p + \mathbf{\Delta x}_{p,q_i} \cdot [\nabla\phi]_{p,q_i}. \tag{B.8}$$

By unicity of the Taylor expansion in p of $[\phi]_{q_i}$ at order1, we have

$$[\nabla\phi]_{p,q_i} = [\nabla\phi]_p. \tag{B.9}$$

Similarly, by unicity of the Taylor expansion in q_i of $[\phi]_p$ at order 1, we have

$$[\nabla\phi]_{p,q_i} = [\nabla\phi]_{q_i}. \tag{B.10}$$

We also have

$$[\phi]_{q_i} = [\phi]_\Gamma + (1-\theta)\mathbf{\Delta x}_{p,q_i} \cdot [\nabla\phi]_\Gamma, \tag{B.11}$$

where $\theta := \theta_{p,q_i}$ is the relative distance to the pair $[p, q_i]$ of the interface Γ to node 1 of the pair $[p, q_i]$ (e.g. p in Figure B.1 as the vectors $\mathbf{A}_{p,q_i}$ are all directed from p to nodes q_i), leading to, by Eqs. (B.8) and (B.11),

$$[\phi]_{q_i} = [\phi]_p + \mathbf{\Delta x}_{p,q_i} \cdot [\nabla\phi]_{p,q_i} = [\phi]_\Gamma + (1-\theta)\mathbf{\Delta x}_{p,q_i} \cdot [\nabla\phi]_\Gamma. \tag{B.12}$$

Similarly, we have

$$[\phi]_p = [\phi]_{q_i} - \mathbf{\Delta x}_{p,q_i} \cdot [\nabla\phi]_{p,q_i} = [\phi]_\Gamma - \theta\mathbf{\Delta x}_{p,q_i} \cdot [\nabla\phi]_\Gamma. \tag{B.13}$$

Moreover,

$$[\phi]_\Gamma = [\phi]_p + \theta\mathbf{\Delta x}_{p,q_i} \cdot [\nabla\phi]_p, \tag{B.14}$$

so, by identification in Eq. (B.12) using Eq. (B.9), we have

$$[\nabla\phi]_p = [\nabla\phi]_\Gamma. \tag{B.15}$$

To summarize, we have, at order 1 in ϕ, for all scalar ϕ defined on Ω, and for all $i \in \{1, \dots, N_p\}$ where N_p is the number of neighboring nodes of p,

$$[\nabla\phi]_p = [\nabla\phi]_{p,q_i} = [\nabla\phi]_\Gamma = [\nabla\phi]_{q_i}\,. \tag{B.16}$$

We set $\phi := P$, the fluid pressure. By definition of the operator $[\cdot]_\Gamma$, we have

$$[\nabla P]_\Gamma := \nabla P_\Gamma^l - \nabla P_\Gamma^g \tag{B.17}$$

and

$$\left[\frac{1}{\rho}\nabla P\right]_\Gamma := \frac{1}{\rho_l}\nabla P_\Gamma^l - \frac{1}{\rho_g}\nabla P_\Gamma^g, \tag{B.18}$$

where ρ_l and ρ_g are the densities of the liquid and vapor phases. We can then rewrite Eq. (B.17) using Eq. (B.18), which gives

$$\begin{aligned}[\nabla P]_\Gamma &= \rho_l\left(\frac{1}{\rho_l}\nabla P_\Gamma^l - \frac{1}{\rho_g}\nabla P_\Gamma^g\right) + \left(\frac{\rho_l}{\rho_g} - 1\right)\nabla P_\Gamma^g \\ &= \rho_l\left[\frac{1}{\rho}\nabla P\right]_\Gamma + \frac{[\rho]_\Gamma}{\rho_g}\nabla P_\Gamma^g,\end{aligned} \tag{B.19}$$

and

$$\begin{aligned}[\nabla P]_\Gamma &= \rho_g\left(\frac{1}{\rho_l}\nabla P_\Gamma^l - \frac{1}{\rho_g}\nabla P_\Gamma^g\right) + \left(1 - \frac{\rho_g}{\rho_l}\right)\nabla P_\Gamma^l \\ &= \rho_g\left[\frac{1}{\rho}\nabla P\right]_\Gamma + \frac{[\rho]_\Gamma}{\rho_l}\nabla P_\Gamma^l.\end{aligned} \tag{B.20}$$

B.2 Without phase change

We make the assumption

$$\left[\frac{1}{\rho}\nabla P\right]_\Gamma = 0. \tag{B.21}$$

Remark : to our knowledge, there does not exist any rigorous demonstration of Eq. (B.21). Without phase change, Eq. (B.19) becomes

$$[\nabla P]_\Gamma = \frac{[\rho]_\Gamma}{\rho_g}\nabla P_\Gamma^g, \tag{B.22}$$

and Eq. (B.20) becomes

$$[\nabla P]_\Gamma = \frac{[\rho]_\Gamma}{\rho_l}\nabla P_\Gamma^l. \tag{B.23}$$

We can then rewrite Eq. (B.12) as

$$[P]_{q_i} = [P]_\Gamma + (1-\theta)\frac{[\rho]_\Gamma}{\rho_l}\boldsymbol{\Delta x}_{p,q_i}\cdot\nabla P_\Gamma^l. \tag{B.24}$$

Similarly, Eq. (B.13) becomes

$$[P]_p = [P]_\Gamma - \theta \frac{[\rho]_\Gamma}{\rho_g} \mathbf{\Delta x}_{p,q_i} \cdot \nabla P^g_\Gamma. \tag{B.25}$$

At order 1 in P, we have

$$\begin{aligned}
\nabla^l_{p,q_i} P^l_{p,q_i} &= \frac{P^l_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \\
&= \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{[P]_{q_i}}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \\
&= \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + (1-\theta) \frac{[\rho]_\Gamma}{\rho_l} \frac{\mathbf{\Delta x}_{p,q_i} \cdot \nabla P^l_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \quad \text{by Eq. (B.24)},
\end{aligned} \tag{B.26}$$

and

$$\begin{aligned}
\nabla^g_{p,q_i} P^g_{p,q_i} &= \frac{P^g_{q_i} - P^g_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \\
&= \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{[P]_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \\
&= \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} - \theta \frac{[\rho]_\Gamma}{\rho_g} \frac{\mathbf{\Delta x}_{p,q_i} \cdot \nabla P^g_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \quad \text{by Eq. (B.25)}.
\end{aligned} \tag{B.27}$$

We define ∇P^l_Γ by

$$\nabla P^l_\Gamma := \nabla^l_{p,q_i} P^l_{p,q_i} \tag{B.28}$$

and ∇P^g_Γ by

$$\nabla P^g_\Gamma := \nabla^g_{p,q_i} P^g_{p,q_i}. \tag{B.29}$$

Then, the projection of ∇P^l_Γ on the pair $[p, q_i]$ is equal to ∇P^l_Γ, idem for ∇P^g_Γ. By Eq. (B.28), Eq. (B.26) then becomes

$$\nabla^l_{p,q_i} P^l_{p,q_i} = \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + (1-\theta) \frac{[\rho]_\Gamma}{\rho_l} \nabla^l_{p,q_i} P^l_{p,q_i}. \tag{B.30}$$

Similarly, by Eq. (B.29), Eq. (B.27) becomes

$$\nabla^g_{p,q_i} P^g_{p,q_i} = \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} - \theta \frac{[\rho]_\Gamma}{\rho_g} \nabla^g_{p,q_i} P^g_{p,q_i}. \tag{B.31}$$

Equations (B.30) and (B.31) are implicit equations. The difference (B.30) - (B.31) gives

$$[\nabla P]_{p,q_i} = (1-\theta) \frac{[\rho]_\Gamma}{\rho_l} \nabla^l_{p,q_i} P^l_{p,q_i} + \theta \frac{[\rho]_\Gamma}{\rho_g} \nabla^g_{p,q_i} P^g_{p,q_i}, \tag{B.32}$$

which gives, factorizing by the gradients,

$$\left(1-(1-\theta)\frac{[\rho]_\Gamma}{\rho_l}\right)\nabla^l_{p,q_i}P^l_{p,q_i}=\left(1+\theta\frac{[\rho]_\Gamma}{\rho_g}\right)\nabla^g_{p,q_i}P^g_{p,q_i}. \tag{B.33}$$

Moreover, the `lhs` and the `rhs` of Eq. (B.33), by respectively Eqs. (B.30) and (B.31), are equal to

$$\left(1-(1-\theta)\frac{[\rho]_\Gamma}{\rho_l}\right)\nabla^l_{p,q_i}P^l_{p,q_i}=\frac{P^g_{q_i}-P^l_p}{\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}+\frac{[P]_\Gamma}{\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}=\left(1+\theta\frac{[\rho]_\Gamma}{\rho_g}\right)\nabla^g_{p,q_i}P^g_{p,q_i}. \tag{B.34}$$

We search ρ_Γ such that

$$\frac{1}{\rho_l}\frac{P^l_{q_i}-P^l_p}{\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}=\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}+\mathbf{f}_l\left(\rho_\Gamma,[\cdot]_\Gamma\right), \tag{B.35}$$

and

$$\frac{1}{\rho_g}\frac{P^g_{q_i}-P^g_p}{\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}=\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}+\mathbf{f}_g\left(\rho_\Gamma,[\cdot]_\Gamma\right), \tag{B.36}$$

where $\rho_\Gamma=\rho_\Gamma(\theta,\rho_l,\rho_g)$ is an intermediate density between ρ_l and ρ_g, and $\mathbf{f}_l$, $\mathbf{f}_g$ two functions of ρ_Γ and of jumps at the interface. Substracting Eq. (B.36) to Eq. (B.35), we obtain

$$\frac{1}{\rho_l}\nabla^l_{p,q_i}P^l_{p,q_i}-\frac{1}{\rho_g}\nabla^g_{p,q_i}P^g_{p,q_i}=\frac{1}{\rho_l}\nabla P^l_\Gamma-\frac{1}{\rho_g}\nabla P^g_\Gamma=\left[\frac{1}{\rho}\nabla P\right]_\Gamma=\mathbf{f}_l\left(\rho_\Gamma,[\cdot]_\Gamma\right)-\mathbf{f}_g\left(\rho_\Gamma,[\cdot]_\Gamma\right). \tag{B.37}$$

Without phase change, Eq. (B.21) imposes that this quantity is zero. The benefit of this formalism appears with phase change. We then determine explicitely the expressions of $\mathbf{f}_l$ and $\mathbf{f}_g$.

B.3 With phase change

B.3.1 Pressure laplacian: $\nabla\cdot\frac{1}{\rho}\nabla P$

We adopt the same methodology. The projection method [14] reads

$$\mathbf{u}^{n+1}=\mathbf{u}^*-\frac{\Delta t}{\rho^{n+\frac{1}{2}}}\nabla P^{n+\frac{1}{2}}, \tag{B.38}$$

i.e.

$$\frac{1}{\rho^{n+\frac{1}{2}}}\nabla P^{n+\frac{1}{2}}=\frac{\mathbf{u}^*-\mathbf{u}^{n+1}}{\Delta t}. \tag{B.39}$$

This equality is verified in the whole domain. At the interface, the `lhs` and the `rhs` of Eq. (B.39) exhibit some jumps due to the density jump between the liquid and the vapor. These two jumps are necessarily equal, i.e.

$$\left[\frac{1}{\rho^{n+\frac{1}{2}}}\nabla P^{n+\frac{1}{2}}\right]_\Gamma=\left[\frac{\mathbf{u}^*-\mathbf{u}^{n+1}}{\Delta t}\right]_\Gamma=\mathbf{f}_l\left(\rho_\Gamma,[\cdot]_\Gamma\right)-\mathbf{f}_g\left(\rho_\Gamma,[\cdot]_\Gamma\right)\quad\text{by Eq. (B.37).} \tag{B.40}$$

In the following part, we will determine the expressions of $\mathbf{f}_l$ and $\mathbf{f}_g$. Replacing $[\nabla P]_\Gamma$ in Eq. (B.12) by its expression in Eq. (B.20), we have

$$\begin{aligned}[P]_{q_i} &= [P]_\Gamma + (1-\theta)\mathbf{\Delta x}_{p,q_i} \cdot [\nabla P]_\Gamma \\ &= [P]_\Gamma + (1-\theta)\mathbf{\Delta x}_{p,q_i} \cdot \left(\rho_g \left[\frac{1}{\rho}\nabla P\right]_\Gamma + \frac{[\rho]_\Gamma}{\rho_l}\nabla P^l_\Gamma\right).\end{aligned} \tag{B.41}$$

Similarly, replacing $[\nabla P]_\Gamma$ in Eq. (B.13) by its expression in Eq. (B.19), we have

$$\begin{aligned}[P]_p &= [P]_\Gamma - \theta\mathbf{\Delta x}_{p,q_i} \cdot [\nabla P]_\Gamma \\ &= [P]_\Gamma - \theta\mathbf{\Delta x}_{p,q_i} \cdot \left(\rho_l \left[\frac{1}{\rho}\nabla P\right]_\Gamma + \frac{[\rho]_\Gamma}{\rho_g}\nabla P^g_\Gamma\right).\end{aligned} \tag{B.42}$$

The `lhs` of Eq. (B.35) then reads

$$\frac{1}{\rho_l}\frac{P^l_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} = \frac{1}{\rho_l}\frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} + \frac{1}{\rho_l}\frac{[P]_{q_i}}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} \tag{B.43}$$

$$\begin{aligned}&= \frac{1}{\rho_l}\frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} + \frac{1}{\rho_l}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} \\ &\quad + \frac{1}{\rho_l}\frac{(1-\theta)\mathbf{\Delta x}_{p,q_i} \cdot \left(\rho_g \left[\frac{1}{\rho}\nabla P\right]_\Gamma + \frac{[\rho]_\Gamma}{\rho_l}\nabla P^l_\Gamma\right)}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}.\end{aligned} \tag{B.44}$$

Replacing ∇P^l_Γ by its expression in Eq. (B.28) (∇P^l_Γ is then equal to its projection on the pair $[p, q_i]$), we have

$$\begin{aligned}\left(1 - (1-\theta)\frac{[\rho]_\Gamma}{\rho_l}\right)\frac{1}{\rho_l}\frac{P^l_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} &= \frac{1}{\rho_l}\frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} + \frac{1}{\rho_l}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} \\ &\quad + (1-\theta)\frac{\rho_g}{\rho_l}\frac{\mathbf{\Delta x}_{p,q_i} \cdot \left[\frac{1}{\rho}\nabla P\right]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\end{aligned} \tag{B.45}$$

We define ρ_Γ such that

$$1 - (1-\theta)\frac{[\rho]_\Gamma}{\rho_l} := \frac{\rho_\Gamma}{\rho_l}, \tag{B.46}$$

i.e.

$$\rho_\Gamma := \rho_l - (1-\theta)[\rho]_\Gamma = \theta\rho_l + (1-\theta)\rho_g. \tag{B.47}$$

Important : ρ_Γ is then not a smoothing of ρ_l and ρ_g ; indeed, if the interface is very close to node p, then $\theta \approx 0$ and $\rho_\Gamma \approx \rho_g$; similarly, if $\theta \approx 1$, $\rho_\Gamma \approx \rho_l$.
Equation (B.45) reads

$$\begin{aligned}\frac{\rho_\Gamma}{\rho_l}\frac{1}{\rho_l}\frac{P^l_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} &= \frac{1}{\rho_l}\frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} + \frac{1}{\rho_l}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} \\ &\quad + (1-\theta)\frac{\rho_g}{\rho_l}\frac{\mathbf{\Delta x}_{p,q_i} \cdot \left[\frac{1}{\rho}\nabla P\right]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\end{aligned} \tag{B.48}$$

which gives, multiplying at left and right by ρ_l/ρ_Γ,

$$\frac{1}{\rho_l}\frac{P^l_{q_i}-P^l_p}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}=\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}$$
$$\underbrace{+\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}+(1-\theta)\,\frac{\rho_g}{\rho_\Gamma}\frac{\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\left[\frac{1}{\rho}\nabla P\right]_\Gamma}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}}_{\mathbf{f}_l\left(\rho_\Gamma,[\cdot]_\Gamma\right)},\quad\text{(B.49)}$$

or equivalently

$$\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}=\frac{1}{\rho_l}\frac{P^l_{q_i}-P^l_p}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}-\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}$$
$$-(1-\theta)\,\frac{\rho_g}{\rho_\Gamma}\frac{\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\left[\frac{1}{\rho}\nabla P\right]_\Gamma}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}.\quad\text{(B.50)}$$

Taking the scalar product of Eq. (B.50) by $+\mathbf{A}_{p,q_i}$, we obtain

$$\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}=\frac{1}{\rho_l}\frac{P^l_{q_i}-P^l_p}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}-\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}$$
$$-(1-\theta)\,\frac{\rho_g}{\rho_\Gamma}\frac{\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\left[\frac{1}{\rho}\nabla P\right]_\Gamma}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}.\quad\text{(B.51)}$$

From Eq. (B.39), we have

$$\nabla\cdot\left(\frac{1}{\rho_l}\nabla^l_p P^l_p\right)=\nabla\cdot\left(\frac{\mathbf{u}^{*,l}_p-\mathbf{u}^{n+1,l}_p}{\Delta t}\right)=\frac{\nabla\cdot\mathbf{u}^{*,l}_p}{\Delta t},\quad\text{(B.52)}$$

then the first term of the `rhs` of Eq. (B.51) reads (computation of divergences before division by $\mathcal{V}_p$)

$$\frac{1}{\rho_l}\frac{P^l_{q_i}-P^l_p}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}=\frac{1}{2}\frac{\mathbf{u}^{*,l}_p+\mathbf{u}^{*,l}_{q_i}}{\Delta t}\cdot\mathbf{A}_{p,q_i},\quad\text{(B.53)}$$

which leads to rewrite Eq. (B.51), replacing $\left[\frac{1}{\rho}\nabla P\right]_\Gamma$ by its expression in Eq. (B.40),

$$\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}=\frac{1}{2}\frac{\mathbf{u}^{*,l}_p+\mathbf{u}^{*,l}_{q_i}}{\Delta t}\cdot\mathbf{A}_{p,q_i}-\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}$$
$$-(1-\theta)\,\frac{\rho_g}{\rho_\Gamma}\frac{\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\dfrac{\left[\mathbf{u}^*\right]_\Gamma-\left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t}}{\left\|\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\right\|^2}\boldsymbol{\Delta}\mathbf{x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}.\quad\text{(B.54)}$$

We make the following assumption:

$$\frac{\mathbf{\Delta x}_{p,q_i} \cdot \dfrac{[\mathbf{u}^*]_\Gamma - \left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t}}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \cdot \mathbf{A}_{p,q_i} = \frac{[\mathbf{u}^*]_\Gamma - \left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t} \cdot \mathbf{A}_{p,q_i}, \tag{B.55}$$

which is true if:

1. $\mathbf{\Delta x}_{p,q_i}$ and $\mathbf{A}_{p,q_i}$ are colinear (i.e. regular grid) ;
2. $\frac{[\mathbf{u}^*]_\Gamma - \left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t}$ is directed toward $\mathbf{\Delta x}_{p,q_i}$.

Equation (B.54) then becomes

$$\begin{aligned}\frac{1}{\rho_\Gamma} \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \cdot \mathbf{A}_{p,q_i} = \frac{1}{2} \frac{\mathbf{u}^{*,l}_p + \mathbf{u}^{*,l}_{q_i}}{\Delta t} \cdot \mathbf{A}_{p,q_i} - \frac{1}{\rho_\Gamma} \frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \cdot \mathbf{A}_{p,q_i} \\ - (1-\theta) \frac{\rho_g}{\rho_\Gamma} \frac{[\mathbf{u}^*]_\Gamma - \left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t} \cdot \mathbf{A}_{p,q_i},\end{aligned} \tag{B.56}$$

which is the **contribution of the pair $[p, q_i]$ to the computation of the laplacian of P in p** before division by $\mathcal{V}_p$: in red, `lhs` of the pair $[p, q_i]$; in blue, contribution of the pair $[p, q_i]$ to the `rhs` of node p, as they appear in the code.

Similarly, the `lhs` of Eq. (B.36) reads

$$\frac{1}{\rho_g} \frac{P^g_{q_i} - P^g_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} = \frac{1}{\rho_g} \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{1}{\rho_g} \frac{[P]_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \tag{B.57}$$

$$\begin{aligned}= \frac{1}{\rho_g} \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{1}{\rho_g} \frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \\ + \frac{1}{\rho_g} \frac{-\theta \mathbf{\Delta x}_{p,q_i} \cdot \left(\rho_l \left[\frac{1}{\rho} \nabla P\right]_\Gamma + \frac{[\rho]_\Gamma}{\rho_g} \nabla P^g_\Gamma \right)}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i}.\end{aligned} \tag{B.58}$$

Then, by Eq. (B.29), we have

$$\begin{aligned}\left(1 + \theta \frac{[\rho]_\Gamma}{\rho_g}\right) \frac{1}{\rho_g} \frac{P^g_{q_i} - P^g_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} = \frac{1}{\rho_g} \frac{P^g_{q_i} - P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} + \frac{1}{\rho_g} \frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i} \\ - \theta \frac{\rho_l}{\rho_g} \frac{\mathbf{\Delta x}_{p,q_i} \cdot \left[\frac{1}{\rho} \nabla P\right]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2} \mathbf{\Delta x}_{p,q_i}.\end{aligned} \tag{B.59}$$

Multiplying at left and right by ρ_g/ρ_Γ, we obtain

$$\frac{1}{\rho_g}\frac{P^g_{q_i}-P^g_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}=\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} + \underbrace{\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}-\theta\frac{\rho_l}{\rho_\Gamma}\frac{\mathbf{\Delta x}_{p,q_i}\cdot\left[\frac{1}{\rho}\nabla P\right]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}}_{\mathbf{f}_g(\rho_\Gamma,[\cdot]_\Gamma)}, \quad \text{(B.60)}$$

or equivalently

$$\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}=\frac{1}{\rho_g}\frac{P^g_{q_i}-P^g_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}-\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} + \theta\frac{\rho_l}{\rho_\Gamma}\frac{\mathbf{\Delta x}_{p,q_i}\cdot\left[\frac{1}{\rho}\nabla P\right]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}. \quad \text{(B.61)}$$

Taking the scalar product of Eq. (B.61) by $-\mathbf{A}_{p,q_i}$, we obtain

$$-\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}=-\frac{1}{\rho_g}\frac{P^g_{q_i}-P^g_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}+\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i} - \theta\frac{\rho_l}{\rho_\Gamma}\frac{\mathbf{\Delta x}_{p,q_i}\cdot\left[\frac{1}{\rho}\nabla P\right]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}. \quad \text{(B.62)}$$

By Eq. (B.52) and Eq. (B.53) applied to vapor, and replacing $\left[\frac{1}{\rho}\nabla P\right]_\Gamma$ by its expression in Eq. (B.40), Eq. (B.62) becomes

$$-\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}=-\frac{1}{2}\frac{\mathbf{u}^{*,g}_p+\mathbf{u}^{*,g}_{q_i}}{\Delta t}\cdot\mathbf{A}_{p,q_i}+\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i} - \theta\frac{\rho_l}{\rho_\Gamma}\frac{\mathbf{\Delta x}_{p,q_i}\cdot\frac{[\mathbf{u}^*]_\Gamma-\left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t}}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}. \quad \text{(B.63)}$$

The hypothesis (B.55) gives

$$-\frac{1}{\rho_\Gamma}\frac{P^g_{q_i}-P^l_p}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}=-\frac{1}{2}\frac{\mathbf{u}^{*,g}_p+\mathbf{u}^{*,g}_{q_i}}{\Delta t}\cdot\mathbf{A}_{p,q_i}+\frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i} -\theta\frac{\rho_l}{\rho_\Gamma}\frac{[\mathbf{u}^*]_\Gamma-\left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t}\cdot\mathbf{A}_{p,q_i}, \quad \text{(B.64)}$$

which is the **contribution of the pair $[p, q_i]$ to the computation of the laplacian of P in q_i** before division by $\mathcal{V}_{q_i}$: in red, `lhs` of the pair $[p, q_i]$; in blue, contribution of the pair $[p, q_i]$ to the `rhs` of node q_i, as they appear in the code.

Note. By Eqs. (B.40) and (B.55), the difference (B.49) - (B.60) gives

$$\frac{1}{\rho_l}\frac{P_{q_i}^l - P_p^l}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} - \frac{1}{\rho_g}\frac{P_{q_i}^g - P_p^g}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i} = \left[\frac{1}{\rho}\nabla P\right]_\Gamma. \tag{B.65}$$

B.3.2 Pressure gradient for the computation of a ghost pressure

Once we have computed $P_p^{l,n+\frac{1}{2}}$, we have to compute $\nabla_p^l P_p^{l,n+\frac{1}{2}}$ in order to correct the velocity. To this purpose, we extrapolate a ghost liquid pressure $P_{q_i}^{l,n+\frac{1}{2},G}$ in q_i using a liquid pressure gradient given by Eqs. (B.51), (B.40) and (B.55),

$$\begin{aligned}\frac{P_{q_i}^l - P_p^l}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i} = \rho_l\Bigg(&\frac{1}{\rho_\Gamma}\frac{P_{q_i}^g - P_p^l}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i} + \frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}\\ &+(1-\theta)\frac{\rho_g}{\rho_\Gamma}\frac{[\mathbf{u}^*]_\Gamma - \left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t}\cdot\mathbf{A}_{p,q_i}\Bigg),\end{aligned} \tag{B.66}$$

and

$$\begin{aligned}P_{q_i}^{l,n+\frac{1}{2},G} &:= P_p^{l,n+\frac{1}{2}} + \left(\frac{P_{q_i}^l - P_p^l}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}\right)\|\mathbf{\Delta x}_{p,q_i}\|\\ &= P_p^{l,n+\frac{1}{2}} + \frac{P_{q_i}^l - P_p^l}{\|\mathbf{\Delta x}_{p,q_i}\|}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}.\end{aligned} \tag{B.67}$$

Similarly, by Eqs. (B.62), (B.40) and (B.55), we have

$$\begin{aligned}\frac{P_{q_i}^g - P_p^g}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i} = \rho_g\Bigg(&\frac{1}{\rho_\Gamma}\frac{P_{q_i}^g - P_p^l}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i} + \frac{1}{\rho_\Gamma}\frac{[P]_\Gamma}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}\\ &-\theta\frac{\rho_l}{\rho_\Gamma}\frac{[\mathbf{u}^*]_\Gamma - \left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t}\cdot\mathbf{A}_{p,q_i}\Bigg),\end{aligned} \tag{B.68}$$

and

$$\begin{aligned}P_p^{g,n+\frac{1}{2},G} &:= P_{q_i}^{g,n+\frac{1}{2}} - \left(\frac{P_{q_i}^g - P_p^g}{\|\mathbf{\Delta x}_{p,q_i}\|^2}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}\right)\|\mathbf{\Delta x}_{p,q_i}\|\\ &= P_{q_i}^{g,n+\frac{1}{2}} + \frac{P_{q_i}^g - P_p^g}{\|\mathbf{\Delta x}_{p,q_i}\|}\mathbf{\Delta x}_{p,q_i}\cdot\mathbf{A}_{p,q_i}.\end{aligned} \tag{B.69}$$

We recover the `rhs` of Eqs. (B.66) and (B.68) in the code.

B.3.3 Balance equation for the pressure laplacian in 1D

We write the finite volume discretization of $\nabla \cdot \left(\frac{1}{\rho}\nabla P\right)$ in p **without considering the interface**. We have

$$\begin{aligned}\nabla \cdot \left(\frac{1}{\rho}\nabla P\right)\Big|_p &= \frac{1}{\mathcal{V}_p}\left(\frac{1}{\rho_l}\frac{P^l_{p+1}-P^l_p}{\|\mathbf{\Delta x}_{p,p+1}\|^2}\mathbf{\Delta x}_{p,p+1}\cdot\mathbf{A}_{p,p+1} + \frac{1}{\rho_l}\frac{P^l_p-P^l_{p-1}}{\|\mathbf{\Delta x}_{p,p-1}\|^2}\mathbf{\Delta x}_{p,p-1}\cdot(-\mathbf{A}_{p,p-1})\right)\\ &= \frac{1}{\mathcal{V}_p}\frac{1}{\rho_l}\left(\frac{P^l_{p+1}-P^l_p}{\|\mathbf{\Delta x}_{p,p+1}\|^2}\mathbf{\Delta x}_{p,p+1}\cdot\mathbf{A}_{p,p+1} - \frac{P^l_p-P^l_{p-1}}{\|\mathbf{\Delta x}_{p,p-1}\|^2}\mathbf{\Delta x}_{p,p-1}\cdot\mathbf{A}_{p,p-1}\right) \quad \text{(B.70)}\end{aligned}$$

The Poisson equation to be solved in s for the pressure is

$$\Delta t \nabla \cdot \left(\frac{1}{\rho}\nabla P\right)\Big|_s = \nabla\cdot\mathbf{u}^*|_s\,, \quad s\in\{2,\dots,N-1\}\,. \tag{B.71}$$

Integrating $\nabla \cdot \left(\frac{1}{\rho}\nabla p\right)$ over the whole domain, we have, by Eq. (B.70),

$$\begin{aligned}\int_\Omega \nabla\cdot\left(\frac{1}{\rho}\nabla P\right)dV &= \sum_{k=2}^{N-1}\mathcal{V}_k\,\nabla\cdot\left(\frac{1}{\rho}\nabla P\right)\Big|_k\\ &= \sum_{k=2}^{N-1}\frac{1}{\rho_\eta}\left(\frac{P^\eta_{p+1}-P^\eta_p}{\|\mathbf{\Delta x}_{p,p+1}\|^2}\mathbf{\Delta x}_{p,p+1}\cdot\mathbf{A}_{p,p+1} - \frac{P^\eta_p-P^\eta_{p-1}}{\|\mathbf{\Delta x}_{p,p-1}\|^2}\mathbf{\Delta x}_{p,p-1}\cdot\mathbf{A}_{p,p-1}\right), \quad \eta = l \text{ or } g\\ &= -\frac{1}{\rho_l}\frac{P^l_2-P^l_1}{\|\mathbf{\Delta x}_{1,2}\|^2}\mathbf{\Delta x}_{1,2}\cdot\mathbf{A}_{1,2} + \frac{1}{\rho_g}\frac{P^g_N-P^g_{N-1}}{\|\mathbf{\Delta x}_{N,N-1}\|^2}\mathbf{\Delta x}_{N,N-1}\cdot\mathbf{A}_{N,N-1}\\ &\qquad + \frac{1}{\rho_l}\frac{P^l_{p+1}-P^l_p}{\|\mathbf{\Delta x}_{p,p+1}\|^2}\mathbf{\Delta x}_{p,p+1}\cdot\mathbf{A}_{p,p+1} - \frac{1}{\rho_g}\frac{P^g_{p+1}-P^g_p}{\|\mathbf{\Delta x}_{p,p+1}\|^2}\mathbf{\Delta x}_{p,p+1}\cdot\mathbf{A}_{p,p+1}\\ &= -\frac{1}{\rho_l}\frac{P^l_2-P^l_1}{\|\mathbf{\Delta x}_{1,2}\|^2}\mathbf{\Delta x}_{1,2}\cdot\mathbf{A}_{1,2} + \frac{1}{\rho_g}\frac{P^g_N-P^g_{N-1}}{\|\mathbf{\Delta x}_{N,N-1}\|^2}\mathbf{\Delta x}_{N,N-1}\cdot\mathbf{A}_{N,N-1} + \left[\frac{1}{\rho}\nabla P\right]_\Gamma\cdot\mathbf{A}_{p,p+1}\end{aligned} \tag{B.72}$$

Moreover, we have

$$\frac{1}{\Delta t}\,\nabla\cdot\mathbf{u}^*|_p = \frac{1}{\Delta t}\frac{1}{\mathcal{V}_p}\left(\frac{\mathbf{u}^{*,l}_p+\mathbf{u}^{*,l}_{p+1}}{2}\cdot\mathbf{A}_{p,p+1} + \frac{\mathbf{u}^{*,l}_p+\mathbf{u}^{*,l}_{p-1}}{2}\cdot(-\mathbf{A}_{p,p-1})\right) \tag{B.73}$$

$$= \frac{1}{\Delta t}\frac{1}{\mathcal{V}_p}\left(\frac{\mathbf{u}^{*,l}_p+\mathbf{u}^{*,l}_{p+1}}{2}\cdot\mathbf{A}_{p,p+1} - \frac{\mathbf{u}^{*,l}_p+\mathbf{u}^{*,l}_{p-1}}{2}\cdot\mathbf{A}_{p,p-1}\right), \tag{B.74}$$

then

$$\begin{aligned}
\int_\Omega \nabla\cdot\left(\frac{\mathbf{u}^*}{\Delta t}\right) dV &= \sum_{k=2}^{N-1} \mathcal{V}_k \left.\nabla\cdot\left(\frac{\mathbf{u}^*}{\Delta t}\right)\right|_k \\
&= \sum_{k=2}^{N-1} \frac{1}{\Delta t}\left(\frac{\mathbf{u}_k^{*,\eta}+\mathbf{u}_{k+1}^{*,\eta}}{2}\cdot\mathbf{A}_{k,k+1} - \frac{\mathbf{u}_{k-1}^{*,\eta}+\mathbf{u}_{k}^{*,\eta}}{2}\cdot\mathbf{A}_{k,k-1}\right) \\
&= -\frac{1}{2}\frac{\mathbf{u}_1^{*,l}+\mathbf{u}_2^{*,l}}{\Delta t}\cdot\mathbf{A}_{1,2} + \frac{1}{2}\frac{\mathbf{u}_{N-1}^{*,g}+\mathbf{u}_N^{*,g}}{\Delta t}\cdot\mathbf{A}_{N-1,N} \\
&\qquad + \frac{1}{2}\left(\frac{\mathbf{u}_p^{*,l}+\mathbf{u}_{p+1}^{*,l}}{\Delta t}\cdot\mathbf{A}_{p,p+1} - \frac{\mathbf{u}_p^{*,g}+\mathbf{u}_{p+1}^{*,g}}{\Delta t}\cdot\mathbf{A}_{p,p+1}\right) \\
&= -\frac{1}{2}\frac{\mathbf{u}_1^{*,l}+\mathbf{u}_2^{*,l}}{\Delta t}\cdot\mathbf{A}_{1,2} + \frac{1}{2}\frac{\mathbf{u}_{N-1}^{*,g}+\mathbf{u}_N^{*,g}}{\Delta t}\cdot\mathbf{A}_{N-1,N} + \frac{1}{2}\frac{[\mathbf{u}^*]_p+[\mathbf{u}^*]_{p+1}}{\Delta t}\cdot\mathbf{A}_{p,p+1} \\
&= -\frac{1}{2}\frac{\mathbf{u}_1^{*,l}+\mathbf{u}_2^{*,l}}{\Delta t}\cdot\mathbf{A}_{1,2} + \frac{1}{2}\frac{\mathbf{u}_{N-1}^{*,g}+\mathbf{u}_N^{*,g}}{\Delta t}\cdot\mathbf{A}_{N-1,N} + \frac{[\mathbf{u}^*]_\Gamma}{\Delta t}\cdot\mathbf{A}_{p,p+1}
\end{aligned} \tag{B.75}$$

supposing that

$$[\mathbf{u}^*]_\Gamma\cdot\mathbf{A}_{p,p+1} = \frac{[\mathbf{u}^*]_p+[\mathbf{u}^*]_{p+1}}{2}\cdot\mathbf{A}_{p,p+1}. \tag{B.76}$$

The difference (B.72) - (B.75) gives

$$\begin{aligned}
0 = \frac{1}{\rho_g}\frac{P_N^g-P_{N-1}^g}{\|\mathbf{\Delta x}_{N,N-1}\|^2}\mathbf{\Delta x}_{N,N-1}\cdot\mathbf{A}_{N,N-1} - \frac{1}{\rho_l}\frac{P_2^l-P_1^l}{\|\mathbf{\Delta x}_{1,2}\|^2}\mathbf{\Delta x}_{1,2}\cdot\mathbf{A}_{1,2} + \left[\frac{1}{\rho}\nabla P\right]_\Gamma\cdot\mathbf{A}_{p,p+1} \\
+ \frac{1}{2}\frac{\mathbf{u}_1^{*,l}+\mathbf{u}_2^{*,l}}{\Delta t}\cdot\mathbf{A}_{1,2} - \frac{1}{2}\frac{\mathbf{u}_{N-1}^{*,g}+\mathbf{u}_N^{*,g}}{\Delta t}\cdot\mathbf{A}_{N-1,N} - \frac{[\mathbf{u}^*]_\Gamma}{\Delta t}\cdot\mathbf{A}_{p,p+1},
\end{aligned} \tag{B.77}$$

which gives, replacing $[\mathbf{u}^*]_\Gamma$ by $[\mathbf{u}^*]_\Gamma - \left[\mathbf{u}^{n+1}\right]_\Gamma + \left[\mathbf{u}^{n+1}\right]_\Gamma$ and moving the pressures to the left,

$$\begin{aligned}
\frac{1}{\rho_l}\frac{P_2^l-P_1^l}{\|\mathbf{\Delta x}_{1,2}\|^2}\mathbf{\Delta x}_{1,2}\cdot\mathbf{A}_{1,2} - \frac{1}{\rho_g}\frac{P_N^g-P_{N-1}^g}{\|\mathbf{\Delta x}_{N,N-1}\|^2}\mathbf{\Delta x}_{N,N-1}\cdot\mathbf{A}_{N,N-1} \\
= \frac{1}{2}\frac{\mathbf{u}_1^{*,l}+\mathbf{u}_2^{*,l}}{\Delta t}\cdot\mathbf{A}_{1,2} - \frac{1}{2}\frac{\mathbf{u}_{N-1}^{*,g}+\mathbf{u}_N^{*,g}}{\Delta t}\cdot\mathbf{A}_{N-1,N} - \frac{\left[\mathbf{u}^{n+1}\right]_\Gamma}{\Delta t}\cdot\mathbf{A}_{p,p+1},
\end{aligned} \tag{B.78}$$

which is the **balance equation for the laplacian of P in 1D with liquid to the left and vapor to the right of the interface**. This equation has then to be verified at every time step. It is the integral over the whole domain of the Poisson equation, which is itself the divergence of the Navier-Stokes equation. This equation then accounts for the conservation of momentum in the whole domain.

Note : We have $\left[\mathbf{u}^{n+1}\right]_\Gamma = \dot{m}^{n+1}\left[\frac{1}{\rho}\right]_\Gamma^{n+1}\mathbf{n}^{n+1} = \frac{[-\lambda\nabla T\cdot\mathbf{n}]_\Gamma^{n+1}}{h_{\mathrm{lv}}}\left[\frac{1}{\rho}\right]_\Gamma^{n+1}\mathbf{n}^{n+1}$. When we compute the pressure, we have already advanced the temperatures and the levelset function, i.e. we already have $T_{l,g}^{n+1}$ and ϕ^{n+1}, we can then use it to compute $\dot{m}^{n+1}$, and use this last term to compute $\left[\mathbf{u}^{n+1}\right]_\Gamma$.

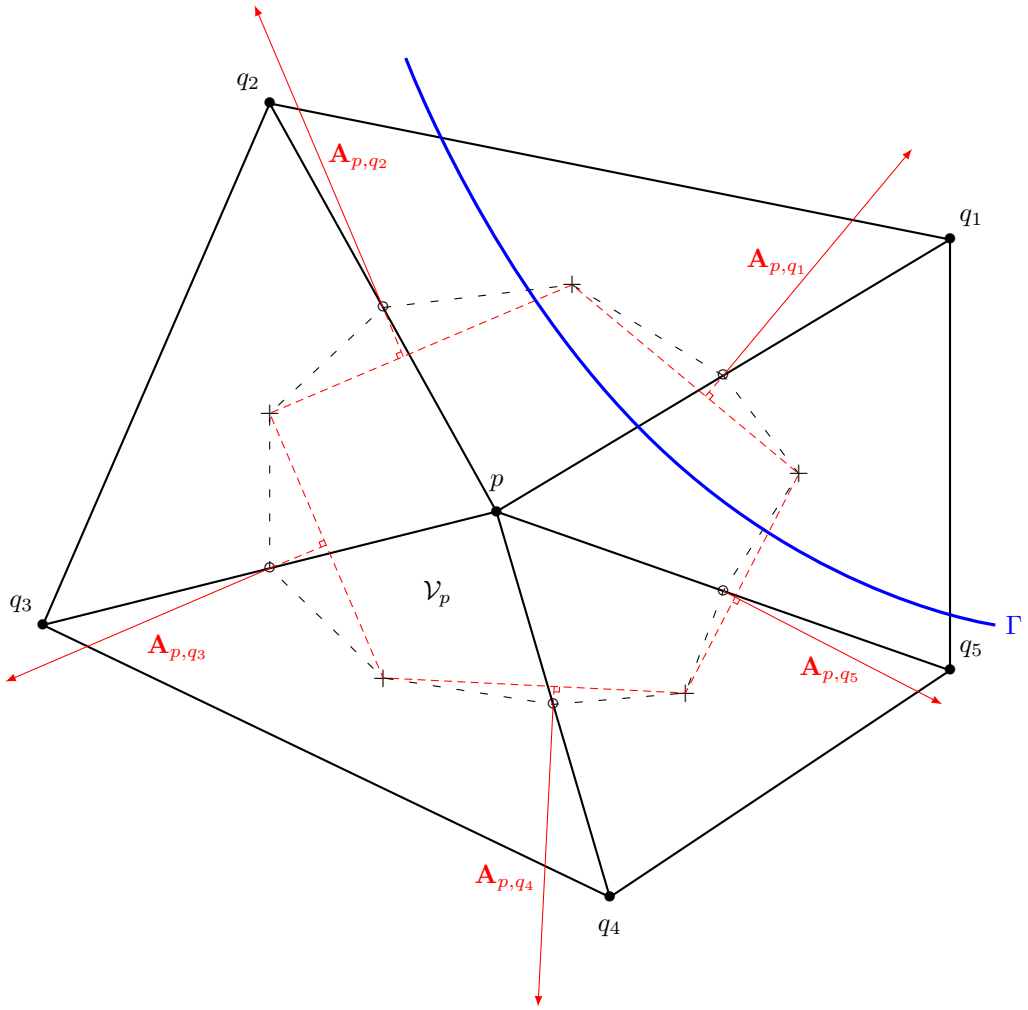


Fig. B.1 – Control volume $\mathcal{V}_p$ associated to node p. The neighboring nodes of p are numbered counterclockwise. The barycenter of each element and the middle of each pair of nodes are respectively represented by the symbols $+$ and $\circ$. The control volume associated to node p is represented in simple black dashes. To each pair of nodes are associated two facets of the control volume, each linking the barycenter of a neighboring element to the middle of the pair. The sum of the fluxes of a physical field at the two facets is equal to the flux of the same physical field at the "face" linking the two barycenters of the neighboring elements to the pair, represented in dense red dashes. If this face is denoted S_i and if $\mathbf{n}_i$ is its outward normal vector to the control volume of p, then the vector $\mathbf{A}_{p,q_i}$ represented in red is equal to $S_i\mathbf{n}_i$. For more clarity, S_i and $\mathbf{n}_i$ are not represented. The interface Γ is represented in blue.

Appendix C

The Fifth-Order Weighted Essentially Non-Oscillatory Scheme for cartesian grids

We present the WENO5 scheme from [78] used in the Hamilton-Jacobi reinitialization equation of the Signed Distance Function on cartesian grids. For clarity reasons, the temporal dependance of ϕ is omitted since reinitialization takes place at fixed physical time. Moreover, only the two-dimensional case is presented, but the extension to three dimensions is straightforward. We suppose a homogeneous cartesian grid of cell size h.

On each grid node $\mathbf{x}_{i,j}$ and ghost node $\mathring{\mathbf{x}}_{i,j}$, we compute first-order upwind gradients in both x and y directions by

$$\left.\frac{\partial \phi}{\partial x}\right|_{i,j} = \frac{\phi_{i,j+1} - \phi_{i,j}}{h} \tag{C.1}$$

and

$$\left.\frac{\partial \phi}{\partial y}\right|_{i,j} = \frac{\phi_{i-1,j} - \phi_{i,j}}{h}, \tag{C.2}$$

where ghost values are used when needed. For simplicity in the notations, theses gradients are denoted $\nabla_x \phi_{i,j}$ and $\nabla_y \phi_{i,j}$.

For each node, we define

$$u^{(1)}_{x,-} = \nabla_x \phi_{j-3,i}, \quad u^{(2)}_{x,-} = \nabla_x \phi_{j-2,i}, \quad u^{(3)}_{x,-} = \nabla_x \phi_{j-1,i}, \quad u^{(4)}_{x,-} = \nabla_x \phi_{j,i}, \qquad u^{(5)}_{x,-} = \nabla_x \phi_{j+1,i}, \tag{C.3}$$

and

$$u^{(1)}_{x,+} = \nabla_x \phi_{j+2,i}, \quad u^{(2)}_{x,+} = \nabla_x \phi_{j+1,i}, \quad u^{(3)}_{x,+} = \nabla_x \phi_{j,i}, \quad u^{(4)}_{x,+} = \nabla_x \phi_{j-1,i}, \quad u^{(5)}_{x,+} = \nabla_x \phi_{j-2,i}. \tag{C.4}$$

The gradients in the y direction are manipulated via

$$u^{(1)}_{y,-} = \nabla_y \phi_{j-3,i}, \quad u^{(2)}_{y,-} = \nabla_y \phi_{j-2,i}, \quad u^{(3)}_{y,-} = \nabla_y \phi_{j-1,i}, \quad u^{(4)}_{y,-} = \nabla_y \phi_{j,i}, \quad u^{(5)}_{y,-} = \nabla_y \phi_{j+1,i}, \tag{C.5}$$

and

$$u_{y,+}^{(1)} = \nabla_y \phi_{j+2,i}, \quad u_{y,+}^{(2)} = \nabla_y \phi_{j+1,i}, \quad u_{y,+}^{(3)} = \nabla_y \phi_{j,i}, \quad u_{y,+}^{(4)} = \nabla_y \phi_{j-1,i}, \quad u_{y,+}^{(5)} = \nabla_y \phi_{j-2,i}. \tag{C.6}$$

We define

$$\begin{aligned} S_{x,-}^{(1)} &= \frac{13}{12}\left(u_{x,-}^{(1)} - 2u_{x,-}^{(2)} + u_{x,-}^{(3)}\right)^2 + \frac{1}{4}\left(u_{x,-}^{(1)} - 4u_{x,-}^{(2)} + 3u_{x,-}^{(3)}\right)^2 \\ S_{x,-}^{(2)} &= \frac{13}{12}\left(u_{x,-}^{(2)} - 2u_{x,-}^{(3)} + u_{x,-}^{(4)}\right)^2 + \frac{1}{4}\left(u_{x,-}^{(2)} - u_{x,-}^{(4)}\right)^2 \\ S_{x,-}^{(3)} &= \frac{13}{12}\left(u_{x,-}^{(3)} - 2u_{x,-}^{(4)} + u_{x,-}^{(5)}\right)^2 + \frac{1}{4}\left(3u_{x,-}^{(3)} - 4u_{x,-}^{(4)} + u_{x,-}^{(5)}\right)^2 \end{aligned} \tag{C.7}$$

and

$$\begin{aligned} S_{x,+}^{(1)} &= \frac{13}{12}\left(u_{x,+}^{(1)} - 2u_{x,+}^{(2)} + u_{x,+}^{(3)}\right)^2 + \frac{1}{4}\left(u_{x,+}^{(1)} - 4u_{x,+}^{(2)} + 3u_{x,+}^{(3)}\right)^2 \\ S_{x,+}^{(2)} &= \frac{13}{12}\left(u_{x,+}^{(2)} - 2u_{x,+}^{(3)} + u_{x,+}^{(4)}\right)^2 + \frac{1}{4}\left(u_{x,+}^{(2)} - u_{x,+}^{(4)}\right)^2 \\ S_{x,+}^{(3)} &= \frac{13}{12}\left(u_{x,+}^{(3)} - 2u_{x,+}^{(4)} + u_{x,+}^{(5)}\right)^2 + \frac{1}{4}\left(3u_{x,+}^{(3)} - 4u_{x,+}^{(4)} + u_{x,+}^{(5)}\right)^2 \end{aligned} . \tag{C.8}$$

The analogous for the y direction gives

$$\begin{aligned} S_{y,-}^{(1)} &= \frac{13}{12}\left(u_{y,-}^{(1)} - 2u_{y,-}^{(2)} + u_{y,-}^{(3)}\right)^2 + \frac{1}{4}\left(u_{y,-}^{(1)} - 4u_{y,-}^{(2)} + 3u_{y,-}^{(3)}\right)^2 \\ S_{y,-}^{(2)} &= \frac{13}{12}\left(u_{y,-}^{(2)} - 2u_{y,-}^{(3)} + u_{y,-}^{(4)}\right)^2 + \frac{1}{4}\left(u_{y,-}^{(2)} - u_{y,-}^{(4)}\right)^2 \\ S_{y,-}^{(3)} &= \frac{13}{12}\left(u_{y,-}^{(3)} - 2u_{y,-}^{(4)} + u_{y,-}^{(5)}\right)^2 + \frac{1}{4}\left(3u_{y,-}^{(3)} - 4u_{y,-}^{(4)} + u_{y,-}^{(5)}\right)^2 \end{aligned} \tag{C.9}$$

and

$$\begin{aligned} S_{y,+}^{(1)} &= \frac{13}{12}\left(u_{y,+}^{(1)} - 2u_{y,+}^{(2)} + u_{y,+}^{(3)}\right)^2 + \frac{1}{4}\left(u_{y,+}^{(1)} - 4u_{y,+}^{(2)} + 3u_{y,+}^{(3)}\right)^2 \\ S_{y,+}^{(2)} &= \frac{13}{12}\left(u_{y,+}^{(2)} - 2u_{y,+}^{(3)} + u_{y,+}^{(4)}\right)^2 + \frac{1}{4}\left(u_{y,+}^{(2)} - u_{y,+}^{(4)}\right)^2 \\ S_{y,+}^{(3)} &= \frac{13}{12}\left(u_{y,+}^{(3)} - 2u_{y,+}^{(4)} + u_{y,+}^{(5)}\right)^2 + \frac{1}{4}\left(3u_{y,+}^{(3)} - 4u_{y,+}^{(4)} + u_{y,+}^{(5)}\right)^2 \end{aligned} . \tag{C.10}$$

Then we define

$$A_{x,-}^{(1)} = \frac{1}{10\left(\epsilon + S_{x,-}^{(1)}\right)^2}, \quad A_{x,-}^{(2)} = \frac{6}{10\left(\epsilon + S_{x,-}^{(2)}\right)^2}, \quad A_{x,-}^{(3)} = \frac{3}{10\left(\epsilon + S_{x,-}^{(3)}\right)^2}, \tag{C.11}$$

and

$$A_{x,+}^{(1)} = \frac{1}{10\left(\epsilon + S_{x,+}^{(1)}\right)^2}, \quad A_{x,+}^{(2)} = \frac{6}{10\left(\epsilon + S_{x,+}^{(2)}\right)^2}, \quad A_{x,+}^{(3)} = \frac{3}{10\left(\epsilon + S_{x,+}^{(3)}\right)^2}. \tag{C.12}$$

For the y direction, we define

$$A_{y,-}^{(1)} = \frac{1}{10\left(\epsilon + S_{y,-}^{(1)}\right)^2}, \quad A_{y,-}^{(2)} = \frac{6}{10\left(\epsilon + S_{y,-}^{(2)}\right)^2}, \quad A_{y,-}^{(3)} = \frac{3}{10\left(\epsilon + S_{y,-}^{(3)}\right)^2}, \tag{C.13}$$

and

$$A_{y,+}^{(1)} = \frac{1}{10\left(\epsilon + S_{y,+}^{(1)}\right)^2}, \quad A_{y,+}^{(2)} = \frac{6}{10\left(\epsilon + S_{y,+}^{(2)}\right)^2}, \quad A_{y,+}^{(3)} = \frac{3}{10\left(\epsilon + S_{y,+}^{(3)}\right)^2}. \tag{C.14}$$

We define the smoothness indicators with the $^{\mathrm{SI}}$ superscript for the x-direction,

$$A_{x,-}^{\mathrm{SI}} = \frac{1}{A_{x,-}^{(1)} + A_{x,-}^{(2)} + A_{x,-}^{(3)}} \quad \text{and} \quad A_{x,+}^{\mathrm{SI}} = \frac{1}{A_{x,+}^{(1)} + A_{x,+}^{(2)} + A_{x,+}^{(3)}}, \tag{C.15}$$

and for the y-direction

$$A_{y,-}^{\mathrm{SI}} = \frac{1}{A_{y,-}^{(1)} + A_{y,-}^{(2)} + A_{y,-}^{(3)}} \quad \text{and} \quad A_{y,+}^{\mathrm{SI}} = \frac{1}{A_{y,+}^{(1)} + A_{y,+}^{(2)} + A_{y,+}^{(3)}}. \tag{C.16}$$

Next we define the weights for the x-direction

$$\omega_{x,-}^{(1)} = A_{x,-}^{(1)} A_{x,-}^{\mathrm{SI}}, \quad \omega_{x,-}^{(2)} = A_{x,-}^{(2)} A_{x,-}^{\mathrm{SI}}, \quad \omega_{x,-}^{(3)} = A_{x,-}^{(3)} A_{x,-}^{\mathrm{SI}}, \tag{C.17}$$

and

$$\omega_{x,+}^{(1)} = A_{x,+}^{(1)} A_{x,+}^{\mathrm{SI}}, \quad \omega_{x,+}^{(2)} = A_{x,+}^{(2)} A_{x,+}^{\mathrm{SI}}, \quad \omega_{x,+}^{(3)} = A_{x,+}^{(3)} A_{x,+}^{\mathrm{SI}}, \tag{C.18}$$

and for the y-direction

$$\omega_{y,-}^{(1)} = A_{y,-}^{(1)} A_{y,-}^{\mathrm{SI}}, \quad \omega_{y,-}^{(2)} = A_{y,-}^{(2)} A_{y,-}^{\mathrm{SI}}, \quad \omega_{y,-}^{(3)} = A_{y,-}^{(3)} A_{y,-}^{\mathrm{SI}}, \tag{C.19}$$

and

$$\omega_{y,+}^{(1)} = A_{y,+}^{(1)} A_{y,+}^{\mathrm{SI}}, \quad \omega_{y,+}^{(2)} = A_{y,+}^{(2)} A_{y,+}^{\mathrm{SI}}, \quad \omega_{y,+}^{(3)} = A_{y,+}^{(3)} A_{y,+}^{\mathrm{SI}}. \tag{C.20}$$

We compute the candidates $S_{x,-}$, $S_{x,+}$, $S_{y,-}$ and $S_{y,+}$ to the components of $\nabla\phi$. We have for the x-direction,

$$\begin{aligned} S_{x,-} = \omega_{x,-}^{(1)} \left(\frac{1}{3}u_{x,-}^{(1)} - \frac{7}{6}u_{x,-}^{(2)} + \frac{11}{6}u_{x,-}^{(3)}\right) + \omega_{x,-}^{(2)} \left(-\frac{1}{6}u_{x,-}^{(2)} + \frac{5}{6}u_{x,-}^{(3)} + \frac{1}{3}u_{x,-}^{(4)}\right) \\ + \omega_{x,-}^{(3)} \left(\frac{1}{3}u_{x,-}^{(3)} + \frac{5}{6}u_{x,-}^{(4)} - \frac{1}{6}u_{x,-}^{(5)}\right), \end{aligned} \tag{C.21}$$

and

$$\begin{aligned} S_{x,+} = \omega_{x,+}^{(1)} \left(\frac{1}{3}u_{x,+}^{(1)} - \frac{7}{6}u_{x,+}^{(2)} + \frac{11}{6}u_{x,+}^{(3)}\right) + \omega_{x,+}^{(2)} \left(-\frac{1}{6}u_{x,+}^{(2)} + \frac{5}{6}u_{x,+}^{(3)} + \frac{1}{3}u_{x,+}^{(4)}\right) \\ + \omega_{x,+}^{(3)} \left(\frac{1}{3}u_{x,+}^{(3)} + \frac{5}{6}u_{x,+}^{(4)} - \frac{1}{6}u_{x,+}^{(5)}\right), \end{aligned} \tag{C.22}$$

and for the y-direction,

$$S_{y,-} = \omega_{y,-}^{(1)} \left(\frac{1}{3} u_{y,-}^{(1)} - \frac{7}{6} u_{y,-}^{(2)} + \frac{11}{6} u_{y,-}^{(3)} \right) + \omega_{y,-}^{(2)} \left(-\frac{1}{6} u_{y,-}^{(2)} + \frac{5}{6} u_{y,-}^{(3)} + \frac{1}{3} u_{y,-}^{(4)} \right) + \omega_{y,-}^{(3)} \left(\frac{1}{3} u_{y,-}^{(3)} + \frac{5}{6} u_{y,-}^{(4)} - \frac{1}{6} u_{y,-}^{(5)} \right), \quad \text{(C.23)}$$

and

$$S_{y,+} = \omega_{y,+}^{(1)} \left(\frac{1}{3} u_{y,+}^{(1)} - \frac{7}{6} u_{y,+}^{(2)} + \frac{11}{6} u_{y,+}^{(3)} \right) + \omega_{y,+}^{(2)} \left(-\frac{1}{6} u_{y,+}^{(2)} + \frac{5}{6} u_{y,+}^{(3)} + \frac{1}{3} u_{y,+}^{(4)} \right) + \omega_{y,+}^{(3)} \left(\frac{1}{3} u_{y,+}^{(3)} + \frac{5}{6} u_{y,+}^{(4)} - \frac{1}{6} u_{y,+}^{(5)} \right). \quad \text{(C.24)}$$

We define

$$\overline{S_x} = S\left(\phi^0\right) \frac{|S_{x,+}| - |S_{x,-}|}{S_{x,+} - S_{x,-}} \quad \text{and} \quad \overline{S_y} = S\left(\phi^0\right) \frac{|S_{y,+}| - |S_{y,-}|}{S_{y,+} - S_{y,-}}. \quad \text{(C.25)}$$

Finally, the components ϕ_x and ϕ_y of $\nabla\phi$ are given by

$$\phi_x = \begin{cases} S_{x,-}, & \text{if } \left\{ S\left(\phi^0\right) S_{x,+} \geq 0 \text{ and } S\left(\phi^0\right) S_{x,-} \geq 0 \right\} \\ & \text{or } \left\{ S\left(\phi^0\right) S_{x,+} < 0 \text{ and } S\left(\phi^0\right) S_{x,-} > 0 \text{ and } \overline{S_x} > 0 \right\} \\ S_{x,+}, & \text{if } \left\{ S\left(\phi^0\right) S_{x,+} \leq 0 \text{ and } S\left(\phi^0\right) S_{x,-} \leq 0 \right\} \\ & \text{or } \left\{ S\left(\phi^0\right) S_{x,+} < 0 \text{ and } S\left(\phi^0\right) S_{x,-} > 0 \text{ and } \overline{S_x} \leq 0 \right\} \\ 0 & \text{otherwise} \end{cases} \quad \text{(C.26)}$$

and

$$\phi_y = \begin{cases} S_{y,-}, & \text{if } \left\{ S\left(\phi^0\right) S_{y,+} \geq 0 \text{ and } S\left(\phi^0\right) S_{y,-} \geq 0 \right\} \\ & \text{or } \left\{ S\left(\phi^0\right) S_{y,+} < 0 \text{ and } S\left(\phi^0\right) S_{y,-} > 0 \text{ and } \overline{S_y} > 0 \right\} \\ S_{y,+}, & \text{if } \left\{ S\left(\phi^0\right) S_{y,+} \leq 0 \text{ and } S\left(\phi^0\right) S_{y,-} \leq 0 \right\} \\ & \text{or } \left\{ S\left(\phi^0\right) S_{y,+} < 0 \text{ and } S\left(\phi^0\right) S_{y,-} > 0 \text{ and } \overline{S_y} \leq 0 \right\} \\ 0 & \text{otherwise} \end{cases}. \quad \text{(C.27)}$$

The Godunov flux G of the signed distance function ϕ at node $\mathbf{x}_{i,j}$ is given by

$$G\left(\phi_{i,j}\right) = \sqrt{\phi_x^2 + \phi_y^2} - 1. \quad \text{(C.28)}$$

Appendix D

The Fast Marching Method for cartesian grids

Outline

D.1 Introduction

We present our implementation of the Fast Marching Method in two-dimensional cartesian grids from [44]. The extension to three dimensions is straightforward. The Fast Marching Method (FMM) is based on the propagation of the signed distance function from the interface in the interface normal direction. To this purpose, the FMM solves the Eikonal equation

$$\|\nabla\phi\| = 1, \tag{D.1}$$

by solving quadratic equations on nodes where ϕ is needed to reconstruct $\nabla\phi$. The starting point of the algorithm is the set of the closest nodes to the interface (band level ± 1) which have ϕ values considered as exact values. From these exact values, updated values can be computed on the neighbor nodes by solving quadratic equations. Among the newly computed values, only the smallest[1] value is considered as exact, and the method iterates until all nodes have an exact signed distance value. The computation is limited to the narrow band around the interface to save computational time. After advection of the interface (using a Level Set method in the case of the Boiling solver), the interface must not be moved anymore during the same temporal iteration. In order to conserve the position of the interface during reinitialization of the signed distance function by the Fast Marching Method, the ϕ values of the nodes of band level ± 1 are considered as exact

[1] or biggest, in the vapor phase

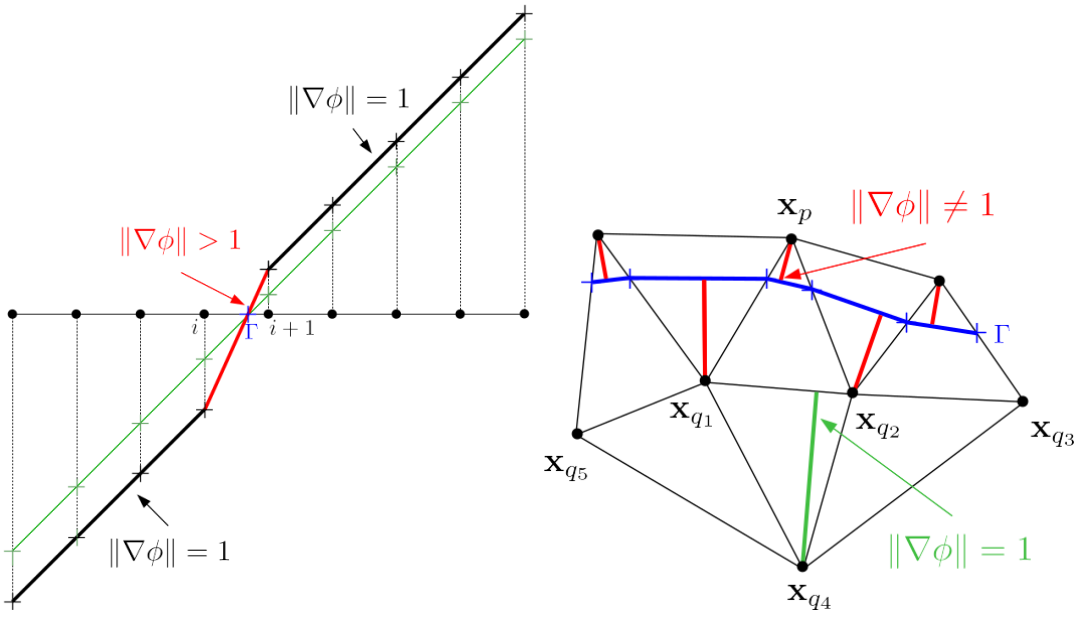


Fig. D.1 – The problem of reinitialization of the signed distance function ϕ on the closest nodes to the interface in multidimensions. In one dimension (a), it is simple to reinitialize ϕ also on the closest nodes $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ without moving the interface. On two dimensions (b), reinitializing the signed distance function on the closest nodes to the interface without moving the interface is much more complex : if one reinitializes $\phi(\mathbf{x}_p)$, $\phi(\mathbf{x}_{q_1})$ and $\phi(\mathbf{x}_{q_2})$, the interface represented with blue crosses on each grid edge will most likely be moved on $(\mathbf{x}_p, \mathbf{x}_{q_1})$ and $(\mathbf{x}_p, \mathbf{x}_{q_2})$ edges.

and are then not updated. As a result, this method does not guarantee $\|\nabla\phi\| = 1$ on these nodes. This drawback is not specific to the FMM and highlights the compromise of all reinitialization methods of level set functions that has to be made : **while we want to ensure $\|\nabla\phi\| = 1$ on the whole domain[2] including the closest nodes to the interface, we do not want to move the interface**. Figure D.1(a) illustrates the problem.

In our implementation of the Fast Marching Method, we made the choice not to reinitialize the signed distance function on the closest nodes to the interface.

D.2 Implementation in the Boiling solver

The FMM relies on node lists named *Alive*, *Close* and *Far*. The Alive nodes have ϕ values considered as exact, the Close nodes are the nodes which have at least one Alive node as neighbor. The Far nodes are the nodes which do not belong to the Alive or Close node lists. Figure D.2 illustrates this domain partition. The signed distance function is reinitialized in both phases in the same pseudo-temporal loop. This feature is available due to the independence of the two phases

[2]at least on a narrow band around the interface

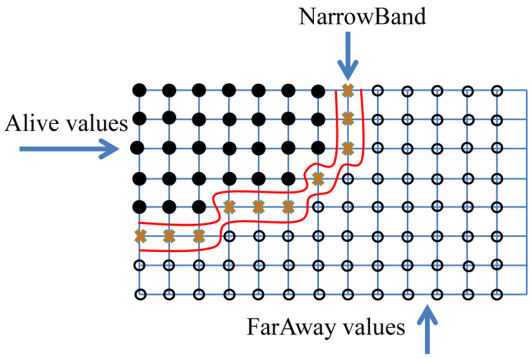


Fig. D.2 – Illustration of the domain partition used in the Fast Marching Method.

with respect to the Fast Marching Method : the previously advected ϕ values on nodes of band level $+1$ are Dirichlet boundary conditions for ϕ in the liquid phase, and the previously advected ϕ values on nodes of band level -1 are Dirichlet boundary conditions for ϕ in the vapor phase. In the following equations, the "$\pm$" sign becomes "$+$" in the liquid phase and "$-$" in the vapor phase. Moreover, the term *neighbor* is understood as one of the four "direct" (top, bottom, left and right) neighbors in two dimensions.

Step 1 : Initialization The first step is the initialization of the Alive, Close and Far node lists. The Alive node list is initialized as the set of all nodes of band level ± 1. The Close node list is initialized as the set of all nodes of band level ± 2. The Far node list is initialized as the set of all nodes of the narrow band which do not belong to the Alive and Close node lists.

Step 2 : Gradient reconstruction by solving quadratic equations For all Close nodes, we solve quadratic equations depending on the number of Alive neighbors. Let $\mathbf{x}$ be a grid node where the indices i and j are omitted for clarity. We use the following method to solve the relevant quadratic equations in order to compute intermediate ϕ^* values.

- If $\mathbf{x}$ has one Alive neighbor $\mathbf{x}_1$, the equation solved is

$$\phi^*(\mathbf{x}) = \phi^{n+1}(\mathbf{x}_1) \pm h. \tag{D.2}$$

- If $\mathbf{x}$ has two Alive neighbors $\mathbf{x}_1$ and $\mathbf{x}_2$, the equation solved is

$$\phi^*(\mathbf{x}) = \frac{1}{2}\left(\phi^{n+1}(\mathbf{x}_1) + \phi^{n+1}(\mathbf{x}_2) \pm \sqrt{2h^2 - \left(\phi^{n+1}(\mathbf{x}_1) - \phi^{n+1}(\mathbf{x}_2)\right)^2}\right). \tag{D.3}$$

- If $\mathbf{x}$ has three Alive neighbors $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$, and
  - if $\mathbf{x}$, $\mathbf{x}_1$ and $\mathbf{x}_2$ are aligned, the equation solved is

$$\phi^*(\mathbf{x}) = \phi^{n+1}(\mathbf{x}_3) \pm \sqrt{h^2 - \frac{1}{4}\left(\phi^{n+1}(\mathbf{x}_1) - \phi^{n+1}(\mathbf{x}_2)\right)^2}; \tag{D.4}$$

  - if $\mathbf{x}$, $\mathbf{x}_1$ and $\mathbf{x}_3$ are aligned, the equation solved is

$$\phi^*(\mathbf{x}) = \phi^{n+1}(\mathbf{x}_2) \pm \sqrt{h^2 - \frac{1}{4}\left(\phi^{n+1}(\mathbf{x}_1) - \phi^{n+1}(\mathbf{x}_3)\right)^2}; \tag{D.5}$$

– if $\mathbf{x}$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are aligned, the equation solved is

$$\phi^*(\mathbf{x}) = \phi^{n+1}(\mathbf{x}_1) \pm \sqrt{h^2 - \frac{1}{4}\left(\phi^{n+1}(\mathbf{x}_2) - \phi^{n+1}(\mathbf{x}_3)\right)^2}. \tag{D.6}$$

- If $\mathbf{x}$ has four Alive neighbors $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ and $\mathbf{x}_4$, the equation solved is

$$\phi^*(\mathbf{x}) = \begin{cases} \min\limits_{p\in\{1,\dots,4\}} \left\{\phi^{n+1}(\mathbf{x}_p)\right\} + h & \text{if } \mathbf{x} \text{ is in the liquid phase} \\ \max\limits_{p\in\{1,\dots,4\}} \left\{\phi^{n+1}(\mathbf{x}_p)\right\} - h & \text{if } \mathbf{x} \text{ is in the vapor phase} \end{cases}. \tag{D.7}$$

Step 3 : Definition of the Trial nodes and node list update Among all Close nodes, we define the liquid and vapor Trial nodes $\mathbf{x}_{\text{Trial,liq}}$ and $\mathbf{x}_{\text{Trial,vap}}$ as the nodes which have respectively the smallest and biggest ϕ^* values, i.e. $\mathbf{x}_{\text{Trial,liq}}$ is defined such that

$$\phi^*(\mathbf{x}_{\text{Trial,liq}}) = \min_{\mathbf{x}\in\mathcal{C}} \{\phi^*(\mathbf{x}) : \phi^*(\mathbf{x}) \geq 0\}, \tag{D.8}$$

and $\mathbf{x}_{\text{Trial,vap}}$ is defined such that

$$\phi^*(\mathbf{x}_{\text{Trial,vap}}) = \max_{\mathbf{x}\in\mathcal{C}} \{\phi^*(\mathbf{x}) : \phi^*(\mathbf{x}) < 0\}, \tag{D.9}$$

where $\mathcal{C}$ denotes the Close node list. The Trial nodes $\mathbf{x}_{\text{Trial,liq}}$ and $\mathbf{x}_{\text{Trial,vap}}$ are transferred to the Alive node list. As a consequence, the reinitialized signed distance values on $\mathbf{x}_{\text{Trial,liq}}$ and $\mathbf{x}_{\text{Trial,vap}}$ are given by

$$\phi^{n+1}(\mathbf{x}_{\text{Trial,liq}}) = \phi^*(\mathbf{x}_{\text{Trial,liq}}), \tag{D.10}$$

and

$$\phi^{n+1}(\mathbf{x}_{\text{Trial,vap}}) = \phi^*(\mathbf{x}_{\text{Trial,vap}}). \tag{D.11}$$

Moreover, the neighbors of $\mathbf{x}_{\text{Trial,liq}}$ and $\mathbf{x}_{\text{Trial,vap}}$ located in the Far node list, if any, are transferred to the Close node list.

Step 4 : Exit criterium The number of Close nodes is recomputed, and the method stops if there is no more Close nodes. Otherwise, the loop goes back to Step 2.

Appendix E

MPI parallelism of the Hamilton-Jacobi equation for the reinitialization of the signed distance function on arbitrary simplicial unstructured grids in two and three dimensions

Outline

E.1 The causality principle on distributed memory

Computing the signed distance function to an interface using distributed memory is highly challenging. Numerical methods designed to build such signed distance functions have to respect the *causality principle* : information originates from the interface and propagates away from it, implying that **the signed distance value to the interface can be computed on one node only if some signed distance values on closer nodes to the interface have already been computed**. The immediate consequence of the causality principle is the major difficulty arising with the parallelism of signed distance function computation algorithms on multiple cores. Indeed, as opposed to classical parallel operations such as computing the gradient of a scalar field known

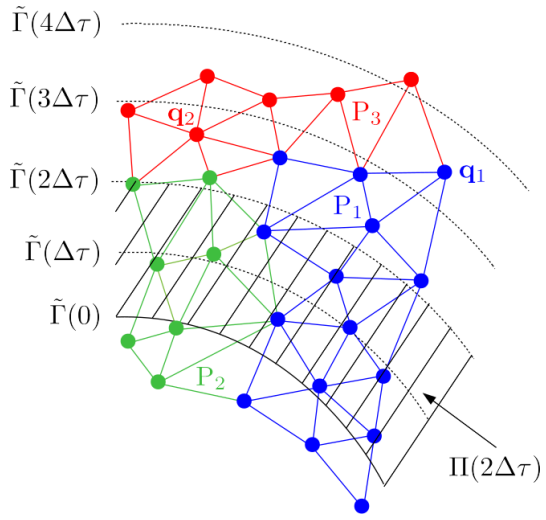


Fig. E.1 – The causality principle in the reinitialization of the signed distance function to the interface with distributed memory. The domain is decomposed on three processes P_1, P_2 and P_3 where only P_1 and P_2 contain a portion of the interface $\Gamma = \widetilde{\Gamma}(0)$. The fictitious interface $\widetilde{\Gamma}(\tau)$ propagates from Γ in its normal direction at speed 1. The width of the area $\Pi(n\Delta\tau)$ delimited by $\widetilde{\Gamma}(0)$ and $\widetilde{\Gamma}(n\Delta\tau)$ is the distance on which ϕ has been reinitialized after n pseudo-temporal iterations of Eq. (E.1). As opposed to *true* parallelism where a given operation is performed simultaneously on all nodes with the same accuracy, the propagation of $\widetilde{\Gamma}$ imposes an order in the computational times needed to reinitialize ϕ. The ϕ value on node $\mathbf{q}_2 \in \Pi(3\Delta\tau)$ will then be reinitialized faster than the ϕ value on node $\mathbf{q}_1 \in \Pi(4\Delta\tau)$.

in the whole domain, parallel operations needed to reinitialize the signed distance function to an interface are performed on information that comes from other processes. Figure E.1 shows a portion of a domain partition composed of three processes P_1, P_2 and P_3, suitable for MPI parallelism.

The interface is located on processes P_1 and P_2. The resolution of Hamilton-Jacobi Eq. (2.7) is based on the following idea. Let Γ be the interface after advection of the signed distance function, and $\widetilde{\Gamma}(0)$ a fictitious copy of Γ. Solving Eq. (2.7) is equivalent to propagating $\widetilde{\Gamma}(0)$ in the normal direction of Γ at speed 1. Let $\Pi(\tau)$ be the area delimited by Γ and $\widetilde{\Gamma}(\tau)$ where $\tau \in [0, c]$ for some constant $c > 0$ and $\widetilde{\Gamma}(\tau)$ is the interface Γ propagated along its normal direction at speed 1 during the time interval $[0, \tau]$. Then, at any given instant τ during reinitialization, where $\tau = 0$ is the beginning of reinitialization, $\Pi(\tau)$ is the area where the signed distance function ϕ has been reinitialized. Outside $\Pi(\tau)$, ϕ is not reinitialized yet. The area $\Pi(\tau)$ is fully characterized by the interface location Γ and the value of time τ, and is thus not related to the domain partition used for MPI parallelism. As a consequence, algorithms have to be designed to update ϕ values on nodes according to their distance to the interface and not on a per-process basis. For instance, on Fig. E.1, process P_1 does contain a portion of the interface but process P_3 does not. One could argue that, in this case, it would be easier to reinitialize ϕ on all nodes of process P_1, and then use ϕ values on common nodes between processes P_1 and P_3 (along with ϕ values on common nodes between processes P_2 and P_3) as boundary conditions to reinitialize ϕ on process P_3. This method would

violate the causality principle because ϕ on node $\mathbf{q}_1$ in P_1 would be reinitialized before ϕ on node $\mathbf{q}_2$ in P_3, whereas $\mathbf{q}_1 \in \Pi(4\Delta\tau)$ and $\mathbf{q}_2 \in \Pi(3\Delta\tau)$. The causality principle imposes to reinitialize $\phi(\mathbf{q}_2)$ before $\phi(\mathbf{q}_1)$.

E.2 Algorithm

We use the algorithm proposed by Dapogny and Frey [17] to solve Hamilton-Jacobi Eq. (2.7) on arbitrary unstructured grids, in two and three dimensions. The method restricts to grids composed only of simplices[1]. This algorithm is reproduced in Alg. 2 with adapted notations where $\mathcal{K} := \{K \in \mathcal{T} : K \cap \Gamma \neq \emptyset\}$ is the set of simplices $K \in \mathcal{T}$ intersecting the interface Γ, $\mathcal{T}$ is the simplicial grid, and $\phi_{\max}$ is a characteristic length of the domain typically set to the length of the largest domain bounding box diagonal. Figure E.2 shows a scheme of the notations used in Alg. 2.

Algorithm 2: Reinitialization of the signed distance function by Hamilton-Jacobi Eq. (2.7) on arbitrary simplicial unstructured grids [17].

Initialize the signed distance function ϕ^0 with:

$$\phi^0(\mathbf{x}) = \begin{cases} \text{exact signed distance function to } \Gamma, & \text{if } \mathbf{x} \text{ belongs to a simplex of } \mathcal{K}, \\ \phi_{\max}, & \text{otherwise.} \end{cases}$$

for $n = 1$ **to** convergence **do**
  $\phi^n(\mathbf{x}) = \phi^{n-1}(\mathbf{x})$ for each node x of $\mathcal{T}$
  for each simplex T of $\mathcal{T}$ **do**
    for each node $\mathbf{x}$ of T *which does not belong to a simplex in* $\mathcal{K}$ **do**
      if $\mathbf{x} \in \Omega_{\text{liq}}$ **then**

$$\phi^n(\mathbf{x}) = \min\left(\phi^n(\mathbf{x}), \phi^{n-1}\left(\mathbf{x} - \frac{\nabla\left(\phi^{n-1}|_T\right)}{\|\nabla\left(\phi^{n-1}|_T\right)\|}\Delta\tau\right)\right) + \Delta\tau, \tag{E.1}$$

      else

$$\phi^n(\mathbf{x}) = \max\left(\phi^n(\mathbf{x}), \phi^{n-1}\left(\mathbf{x} + \frac{\nabla\left(\phi^{n-1}|_T\right)}{\|\nabla\left(\phi^{n-1}|_T\right)\|}\Delta\tau\right)\right) - \Delta\tau. \tag{E.2}$$

return ϕ^n

This algorithm is implemented in the open source[2] MshDist library [17]. The library has been coupled to the Boiling solver, and provided very accurate results in two and three dimensions, as will be shown in Section 5.11. While the library can be used on multiple cores to speed up computations by means of shared memory (OpenMP), it cannot be used with multiple processes and distributed memory (MPI). Indeed, the MshDist library requires that one core has knowledge of all ϕ values

[1] Simplices are the non-flat elements with the smallest number of edges for a given spatial dimension. If $k \in \mathbb{N}$ is the dimension, a k-simplex is defined as the convex hull of its $k+1$ vertices : a 2-simplex is a triangle, a 3-simplex is a tetrahedron.

[2] The source code of the MshDist library is freely available at `https://github.com/ISCDtoolbox/Mshdist`.

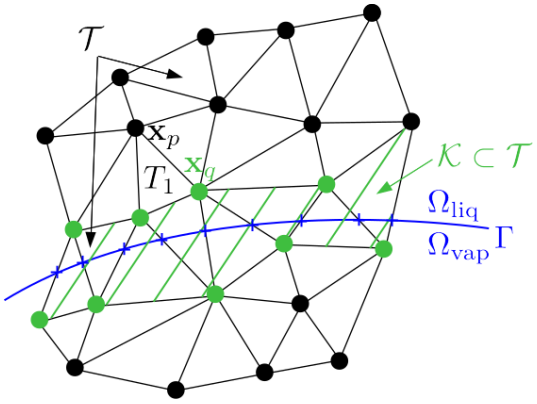


Fig. E.2 – Notations of Alg. 2 : $\mathcal{T}$ is the simplicial grid, $\mathcal{K} \subset \mathcal{T}$ is the simplicial subset of $\mathcal{T}$ intersecting the interface Γ, $T_1 \in \mathcal{T}$ is a simplex, nodes $\mathbf{x}_p$ and $\mathbf{x}_q$ are summits of T_1, and Ω_{liq} is the region occupied by the liquid phase. As opposed to node $\mathbf{x}_p$, node $\mathbf{x}_q$ belongs to simplices of $\mathcal{K}$.

on the whole computational domain. Since our goal is to tackle complex geometries, the Boiling solver is designed to run on multiple processes using distributed memory and MPI parallelism. Simulations performed with the Boiling solver coupled to the MshDist library are limited to one single process. To alleviate this problem, parallelism using MPI and distributed memory of Alg. 2 has been implemented. We present the problems that arose with the parallelism of this algorithm and the solutions found.

E.3 Pre-computation of grid-related informations

Prior to the actual parallelism of Alg. 2 using MPI, grid connectivity-related informations are pre-computed to ease and speed up the search of neighbor nodes or elements during reinitialization. For each node $\mathbf{x}$, the two following lists are pre-computed :

Neighbor element list $\mathcal{L}_1(\mathbf{x})$ The list $\mathcal{L}_1(\mathbf{x})$ of elements to which $\mathbf{x}$ is a summit. In Fig. E.2, the neighbor element list $\mathcal{L}_1(\mathbf{x}_p)$ of node $\mathbf{x}_p$ is given by

$$\mathcal{L}_1(\mathbf{x}_p) = \{T_1, T_2, T_3, T_4, T_5\}. \tag{E.3}$$

Neighbor element edge list $\mathcal{L}_2(\mathbf{x})$ The list $\mathcal{L}_2(\mathbf{x})$ of element edges for all elements in $\mathcal{L}_1(\mathbf{x})$. In Fig. E.2, the neighbor element edge list $\mathcal{L}_2(\mathbf{x}_p)$ of node $\mathbf{x}_p$ is given by

$$\begin{aligned}\mathcal{L}_2(\mathbf{x}_p) = \big\{&(\mathbf{x}_p, \mathbf{x}_{q_1}), (\mathbf{x}_p, \mathbf{x}_{q_2}), (\mathbf{x}_{q_1}, \mathbf{x}_{q_2}), (\mathbf{x}_p, \mathbf{x}_{q_3}), (\mathbf{x}_{q_2}, \mathbf{x}_{q_3}),\\ &(\mathbf{x}_p, \mathbf{x}_{q_4}), (\mathbf{x}_{q_3}, \mathbf{x}_{q_4}), (\mathbf{x}_p, \mathbf{x}_{q_5}), (\mathbf{x}_{q_4}, \mathbf{x}_{q_5}), (\mathbf{x}_{q_5}, \mathbf{x}_{q_1})\big\}.\end{aligned} \tag{E.4}$$

These two lists are local to one element group (see Chapter 3 for details on the double domain decomposition used in YALES2).

The final time of reinitialization $\tau_{\max}$ is given by

$$\tau_{\max} = \sqrt{x_{\text{BB}}^2 + y_{\text{BB}}^2}, \tag{E.5}$$

where x_{BB} and y_{BB} are the dimensions of the whole domain bounding box. Equation (E.5) expresses time $\tau_{\max}$ with respect to distances x_{BB} and y_{BB}. Indeed, thanks to the propagation speed of $\widetilde{\Gamma}$ equal to 1, the analogy between times and distances is valid and common in the research field of viscosity solutions to Hamilton-Jacobi equations.

As will be explained in Section E.5.2.3, the smallest height of all grid simplices is also precomputed.

E.4 Initialization

Line 1 of Alg. 2 initializes the signed distance function to be reinitialized, denoted ϕ^0. Here, ϕ has already been advected, so the interface location $\Gamma = \{x \in \Omega : \phi(\mathbf{x}) = 0\}$ is known and has to be maintained during reinitialization of ϕ. To this purpose, ϕ values on the closest nodes to the interface are not reinitialized. As a result, interface locations on grid edges crossed by the interface, which can be computed by linear interpolations on these edges, are maintained. With the notations of [17], this step is described in the first case of line 1 : by definition of $\mathcal{K}$, "*if* $\mathbf{x}$ *belongs to a simplex in* $\mathcal{K}$" means "*if* $\mathbf{x}$ *is one of the closest nodes to the interface, a node of band level* ± 1", such as node $\mathbf{x}_q$ in Fig. E.2. In this case, the "*exact*" ϕ^0 value is assumed to be the previously advected ϕ value, and one has

$$\phi^0(\mathbf{x}) = \phi(\mathbf{x}). \tag{E.6}$$

As a drawback of maintaining the interface location, we recall that if one does not reinitialize ϕ on the closest nodes to the interface, one does not guarantee $\|\nabla\phi\| = 1$ on these nodes, see Appendix D. If the band level of $\mathbf{x}$ is different from ± 1, as for node $\mathbf{x}_p$ in Fig E.2, then one has

$$\phi^0(\mathbf{x}) = \pm\phi_{\max}. \tag{E.7}$$

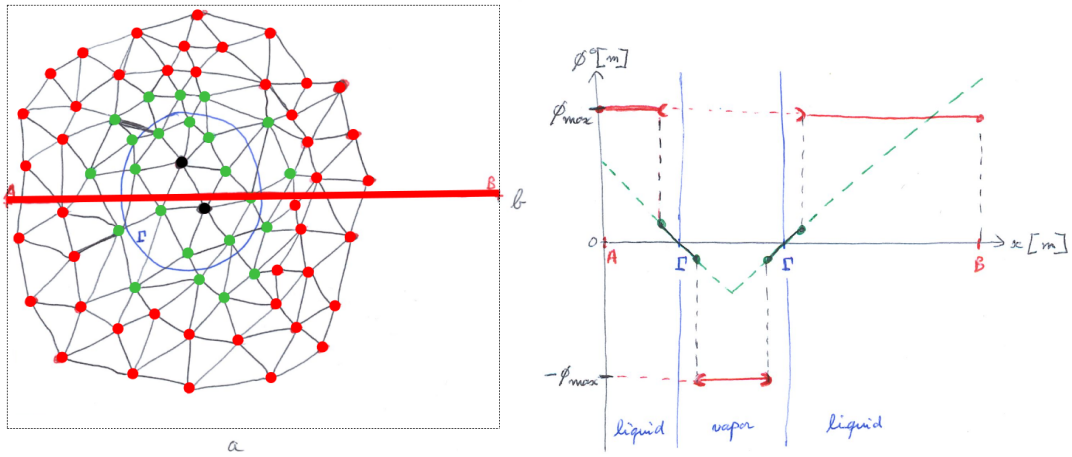
The idea consists in setting $\phi_{\max}$ to a value that will never be reached by the reinitialized signed distance function. In the liquid phase, $\phi_{\max}$ is set to a big, positive value ; whereas in the vapor phase, $\phi_{\max}$ is set to a small, negative value. One can set $\phi_{\max}$ to "$\pm\infty$" where ∞ is understood as a constant real value typically of the order of 10^{15}. Nevertheless, with such an initialization, the algorithm will take longer time to converge. It has been found sufficient to set $\phi_{\max}$ to a characteristic length of the domain such as the length of the largest domain bounding box diagonal. The analogy between times and distances discussed above then gives

$$\phi_{\max} = \tau_{\max}. \tag{E.8}$$

Figure E.3 shows the initialization step for a two-dimensional simulation.

E.5 Implementation in two dimensions

In two dimensions, simplices are triangles. In the first iteration, only the processes containing a portion of the interface, or some closest nodes to the interface, on their subdomain can start computing new relevant values for the closest neighbors (nodes of band level ± 2). The other processes have no knowledge of the interface and their computed ϕ values will be not be accurate until information has propagated to their subdomain boundary. Since information needed to reinitialize the signed distance function *propagates*, sufficiently accurate information is not available for all processes at the same time. This propagation apparently conflicts with *true* parallelism in the



(a) The interface Γ is colored in blue. The nodes of band level ± 1 have exact ϕ values, i.e. values resulting from the previous advection of ϕ, and are colored in green. These values will not be updated during reinitialization. The nodes of band level < -1 located in the vapor phase are set to $-\phi_{\max}$, and are colored in black. The nodes of band level > 1 located in the liquid phase are set to $+\phi_{\max}$, and are colored in red. The horizontal line AB is used to plot interpolated values of ϕ^0 on Subfig. (b).

(b) Interpolated ϕ^0 values on line AB. One can see the discontinuity of ϕ^0 between nodes of band level ± 1 and ± 2. The objective of Alg. 2 is to reinitialize ϕ as the theoretical signed distance function to the interface shown in green.

Fig. E.3 – Initialization of ϕ^0 by line 1 of Alg. 2 for a vapor bubble in two dimensions ($\phi > 0$ in the liquid and $\phi < 0$ in the vapor). Subfig. (a) shows a portion of the domain containing the interface Γ with initialized values of ϕ^0, and Subfig. (b) shows a plot of ϕ^0 along the red line linking points A and B. The $\phi_{\max}$ constant is set to the largest domain bounding box diagonal, i.e. $\phi_{\max} := \sqrt{a^2 + b^2}$.

sense that, even if signed distance function reinitialization algorithms can be designed to run on multiple cores, all cores cannot compute relevant ϕ values at the same time. Figure E.1 illustrates the causality principle and its obstacle to true parallelism.

The two main difficulties arising when solving Eq. (E.1) are the computation of $\nabla\left(\phi^{n-1}\big|_T\right)$ for some simplex T, and the evaluation of ϕ^{n-1} on $\mathbf{x} - \Delta\tau\nabla\left(\phi^{n-1}\big|_T\right) / \left\|\nabla\left(\phi^{n-1}\big|_T\right)\right\|$. These key points are detailed in Sections E.5.1 and E.5.2. The workaround of the obstacle to true parallelism due to the causality principle is detailed in Section E.5.2.3.

E.5.1 Computation of the per-triangle gradient

In this section, the computation of $\nabla\left(\phi|_T\right)$ is presented where the temporal exponent $^{n-1}$ is omitted.

In YALES2, physical fields are stored on grid nodes, or vertices, enabling differentiation in the finite volume method by fluxes integration over vertex-centered control volumes. This convention is

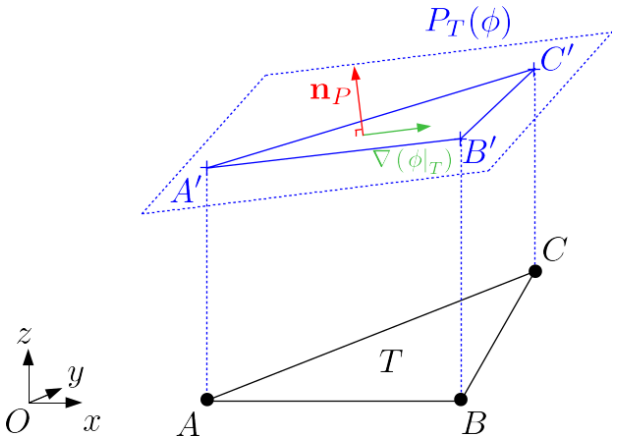


Fig. E.4 – Representation in three dimensions of a two-dimensional element and its associated plane. The triangular grid element T of summits $A\,(x_A, y_A, 0)$, $B\,(x_B, y_B, 0)$ and $C\,(x_C, y_C, 0)$ lies in the plane Oxy (in black), the plane P is the plane containing points $A'\,(x_\mathrm{A}, y_\mathrm{A}, \phi_\mathrm{A})$, $B'\,(x_\mathrm{B}, y_\mathrm{B}, \phi_\mathrm{B})$ and $C'\,(x_\mathrm{C}, y_\mathrm{C}, \phi_\mathrm{C})$ (in blue), and $\mathbf{n}$ is the normal vector to plane P (in red). The gradient of ϕ restricted to T, i.e. $\nabla\,(\phi|_T)$, is computed from $\mathbf{n}$ (in green), and represents the tilt direction of plane P with respect to triangle ABC.

also used for the signed distance function ϕ. Then, on a triangle T of summits $A(x_A, y_A)$, $B(x_B, y_B)$ and $C(x_C, y_C)$, only ϕ values computed on A, B and C, namely ϕ_A, ϕ_B and ϕ_C, are known. Let $(O, \mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ be a three-dimensional cartesian frame. If triangle T lies in the plane generated by $\mathbf{e}_x$ and $\mathbf{e}_y$, then the notation $\phi|_T$ denotes the plane P containing points $A'\,(x_\mathrm{A}, y_\mathrm{A}, \phi_\mathrm{A})$, $B'\,(x_\mathrm{B}, y_\mathrm{B}, \phi_\mathrm{B})$ and $C'\,(x_\mathrm{C}, y_\mathrm{C}, \phi_\mathrm{C})$, as shown in Fig. E.4. The gradient of ϕ on plane P is computed using the normal vector $\mathbf{n} = (n_x, n_y, n_z)^\mathrm{T}$ of plane P, given by

$$\mathbf{n} = \begin{pmatrix} x_B - x_A \\ y_B - y_A \\ \phi_B - \phi_A \end{pmatrix} \times \begin{pmatrix} x_C - x_A \\ y_C - y_A \\ \phi_C - \phi_A \end{pmatrix}, \tag{E.9}$$

where $\times$ denotes the cross product in $\mathbb{R}^3$. Once $\mathbf{n}$ is known, one has

$$\nabla\,(\phi|_T) = \begin{pmatrix} -n_x/n_z \\ -n_y/n_z \end{pmatrix}. \tag{E.10}$$

One can observe that Eq. (E.10) is valid only if $n_z \neq 0$. The case $n_z = 0$ never occurs since it would imply orthogonality between plane P and triangle T, and alignment of nodes A, B and C. The gradient of ϕ restricted to one triangle is computed for all triangles T having at least one summit of band level different from ± 1, i.e. for all triangles $T \in \mathcal{T} \backslash \mathcal{K}$.

E.5.2 Subgrid interpolation of the signed distance function

This section details the evaluation of ϕ at point $\mathbf{x} - \Delta\tau \nabla\,(\phi|_T) \,/\, \|\nabla\,(\phi|_T)\|$ for a given node $\mathbf{x}$.

E.5.2.1 Selecting the per-triangle gradient with the highest norm among neighbor element list

Once $\nabla\left(\phi|_T\right)$ has been computed for all triangles $T \in \mathcal{T}$, one needs to retain for node $\mathbf{x}$ the vector $\nabla\left(\phi|_{T^*(\mathbf{x})}\right)$ having the highest norm among all vectors $\nabla\left(\phi|_T\right)$ where $T \in \mathcal{L}_1(\mathbf{x})$. To this purpose, the euclidean norm $N_T(\phi)$ of $\nabla\left(\phi|_T\right)$ is computed for all triangles $T \in \mathcal{L}_1(\mathbf{x})$ by

$$N_T(\phi) = \sqrt{\frac{n_x^2 + n_y^2}{n_z^2}}, \tag{E.11}$$

where the normal vector $\mathbf{n} = (n_x, n_y, n_z)^{\mathrm{T}}$ is given by Eq. (E.9). The triangle $T^*(\mathbf{x})$ is identified by its norm $N_{T^*(\mathbf{x})}(\phi)$ given by

$$N_{T^*(\mathbf{x})}(\phi) = \max\left\{N_T(\phi) : T \in \mathcal{L}_1(\mathbf{x})\right\}. \tag{E.12}$$

The selection of $\nabla\left(\phi|_{T^*(\mathbf{x})}\right)$ on node $\mathbf{x}$ using Eq. (E.12) requires a special type of el_grp-to-el_grp (internal communicators) and process-to-process (MPI) communications : the selection by one node of a vector having the highest norm among a set of vectors defined on the neighor elements of this node. To this purpose, the communication type `COMM_MAX_NORM` has been implemented in the communication type list of YALES2[3]. Similar operations are performed on all other nodes. Hereinafter, the dependence on $\mathbf{x}$ of $T^*(\mathbf{x})$ is omitted.

E.5.2.2 Shoot in the direction of the steepest gradient

In Alg. 2, the main idea used to reinitialize the signed distance function ϕ to the interface Γ is contained in Eqs. (E.1) and (E.2), recalled hereafter. If node $\mathbf{x}$ is in the liquid phase, then ϕ is updated on $\mathbf{x}$ by

$$\phi^n(\mathbf{x}) = \min\left(\phi^{n-1}(\mathbf{x}), \phi^{n-1}\left(\mathbf{x} - \frac{\nabla\left(\phi^{n-1}|_{T^*}\right)}{\|\nabla\left(\phi^{n-1}|_{T^*}\right)\|}\Delta\tau\right)\right) + \Delta\tau, \tag{E.13}$$

and, if $\mathbf{x}$ is in the vapor phase, then ϕ is updated on $\mathbf{x}$ by

$$\phi^n(\mathbf{x}) = \max\left(\phi^{n-1}(\mathbf{x}), \phi^{n-1}\left(\mathbf{x} + \frac{\nabla\left(\phi^{n-1}|_{T^*}\right)}{\|\nabla\left(\phi^{n-1}|_{T^*}\right)\|}\Delta\tau\right)\right) - \Delta\tau, \tag{E.14}$$

where T^* is the triangle maximizing $\left\|\nabla\left(\phi^{n-1}|_T\right)\right\|$ defined by Eq. (E.12), and $n \in \mathbb{N}^*$.

In this section, the iterative method is illustrated for some node $\mathbf{x}$ located in the liquid phase, i.e. using Eq. (E.13). The transposition to the vapor phase is straightforward (Eq. (E.14)). Let P be the following proposition :

$$\phi^{n-1}(\mathbf{x}) \geq \phi^{n-1}\left(\mathbf{x} - \frac{\nabla\left(\phi^{n-1}|_{T^*}\right)}{\|\nabla\left(\phi^{n-1}|_{T^*}\right)\|}\Delta\tau\right). \tag{P}$$

[3]YALES2 uses a wrapper for the MPI library providing homogeneity in the communication commands between el_grp-to-el_grp and process-to-process communications.

If P is verified, then Eq. (E.13) rewrites

$$\phi^n(\mathbf{x}) = \phi^{n-1}\left(\mathbf{x} - \frac{\nabla\left(\phi^{n-1}\big|_{T^*}\right)}{\|\nabla\left(\phi^{n-1}|_{T^*}\right)\|}\Delta\tau\right) + \Delta\tau, \tag{E.15}$$

otherwise,

$$\phi^n(\mathbf{x}) = \phi^{n-1}(\mathbf{x}) + \Delta\tau. \tag{E.16}$$

Let $\mathbf{s}^{n-1}(\mathbf{x})$ be the *shoot vector* of node $\mathbf{x}$ at iteration $n-1$, given by

$$\mathbf{s}^{n-1}(\mathbf{x}) = \frac{\nabla\left(\phi^{n-1}\big|_{T^*}\right)}{\|\nabla\left(\phi^{n-1}|_{T^*}\right)\|}\Delta\tau, \tag{E.17}$$

implying

$$\left\|\mathbf{s}^{n-1}(\mathbf{x})\right\| = \Delta\tau. \tag{E.18}$$

Equation (E.15) rewrites

$$\phi^n(\mathbf{x}) = \phi^{n-1}\left(\mathbf{x} - \mathbf{s}^{n-1}(\mathbf{x})\right) + \Delta\tau. \tag{E.19}$$

Let $\mathbf{x}_{\text{target}}^{n-1} \in \mathbb{R}^2$ be the *target point* of node $\mathbf{x}$ defined as

$$\mathbf{x}_{\text{target}}^{n-1} = \mathbf{x} - \mathbf{s}^{n-1}(\mathbf{x}). \tag{E.20}$$

Equation (E.19) finally rewrites

$$\phi^n(\mathbf{x}) = \phi^{n-1}(\mathbf{x}_{\text{target}}^{n-1}) + \Delta\tau. \tag{E.21}$$

To summarize, Eq. (E.13) is equivalent to

$$\phi^n(\mathbf{x}) = \begin{cases} \phi^{n-1}\left(\mathbf{x}_{\text{target}}^{n-1}\right) + \Delta\tau, & \text{if Proposition P is true,} \quad \text{(E.22)} \\ \phi^{n-1}\left(\mathbf{x}\right) + \Delta\tau, & \text{otherwise.} \quad \text{(E.23)} \end{cases}$$

The key idea of the method is to iteratively build $\nabla\left(\phi|_{T^*}\right)$ such that Proposition P is verified, i.e. such that $\mathbf{s}(\mathbf{x})$ points to the direction of the interface normal vector $\mathbf{n}_\Gamma$ at node $\mathbf{x}$, $\mathbf{n}_\Gamma(\mathbf{x})$. Equations (E.18) and (E.20) imply that point $\mathbf{x}_{\text{target}}^{n-1} \in \mathcal{C}\left(\mathbf{x}, \Delta\tau\right)$ where $\mathcal{C}\left(\mathbf{x}, \Delta\tau\right)$ is the circle of center $\mathbf{x}$ and radius $\Delta\tau$. Figure E.5 gives a first resolution example of Eq. (E.13) in sequential computing.

In the first iterations of Eq. (E.13), $\nabla\left(\phi^{n-1}\big|_{T^*}\right)$ potentially points to a random direction, as in Fig. E.5(a). As a result, $\mathbf{x}_{\text{target}}^{n-1}$ is potentially farther to the interface than $\mathbf{x}$. In this case, Proposition P is false, and $\phi^n(\mathbf{x})$ is updated by Eq. (E.23), favoring triangle ABC to be, in the next iteration, the triangle T^* maximizing $\|\nabla\left(\phi^n|_T\right)\|$ among all triangles $T \in \mathcal{L}_1(\mathbf{x})$, as in Fig. E.5(b).

Consequently, the next target points of $\mathbf{x}$ are most likely closer to the interface than $\mathbf{x}$ (Fig. E.5(b)), and the direction of $\nabla\left(\phi^{n-1}\big|_{T^*}\right)$ converges to the direction of the interface normal vector at node $\mathbf{x}$, $\mathbf{n}_\Gamma(\mathbf{x})$. In this case, Proposition P is true, and ϕ is updated on $\mathbf{x}$ by Eq. (E.22).

After convergence of Eq. (E.13), the direction of $\nabla\left(\phi|_{T^*}\right)$ is equal to the direction of $\mathbf{n}_\Gamma(\mathbf{x})$, and $\mathbf{x}_{\text{target}}^{n-1}$ is closer to the interface than $\mathbf{x}$, as shown in Fig. E.5(c). Again, Proposition P is true, and ϕ is accurately updated on $\mathbf{x}$ by Eq. (E.22).

Equation (E.22) solved only on node $\mathbf{x}$ is not sufficient to update ϕ on $\mathbf{x}$. Indeed, except if $\mathbf{x}_{\text{target}}^{n-1}$ is located in a triangle in $\mathcal{K}$, as in Fig. E.5(c), if Eq. (E.22) has not previously been solved until convergence on the three summits of the triangle containing $\mathbf{x}_{\text{target}}^{n-1}$, as could be the case in Fig. E.5(b), then Eq. (E.22) solved on $\mathbf{x}$ can set $\phi^n(\mathbf{x})$ to an inaccurate value. This problem is avoided by enforcing the causality principle. Theoretically, the pseudo-time $\tau^{\text{exact}}(\mathbf{x})$ needed for Alg. 2 to reinitialize ϕ on node $\mathbf{x}$ is equal to $|\phi^{\text{exact}}(\mathbf{x})|$, since the fictitious interface $\widetilde{\Gamma}$ propagates at speed 1 away from the interface Γ, along the interface normal vector $\mathbf{n}_\Gamma$ (see Section E.1). The causality principle is enforced in the design of Alg. 2 by the two following key points :

- The main loop is the pseudo-temporal loop for pseudo-time advancement, and the nested loop is a loop on grid nodes in which ϕ^{n-1} is updated to ϕ^n. This loop order implies that when $\phi^n(\mathbf{x})$ is computed on node $\mathbf{x}$, shown in Fig. E.5, using ϕ^{n-1} on the summits D, E and F of triangle T_{target}^{n-1} containing $\mathbf{x}_{\text{target}}^{n-1}$, ϕ^{n-1} has already been computed on D, E and F in the previous pseudo-temporal iteration.

- At pseudo-time τ, ϕ^n has been reinitialized on all nodes located in $\Pi(\tau)$, where $\Pi(\tau)$ is the area of thickness τ delimited by Γ and $\widetilde{\Gamma}(\tau)$ (see Section E.1). Figure E.6 reproduces Fig. E.5(c) where different positions of $\widetilde{\Gamma}$ are represented. Outside $\Pi(\tau)$, ϕ^n has not been reinitialized yet.

These two properties ensure that, for one grid node $\mathbf{p}$ and one point $\mathbf{q} \in \mathbb{R}^2$ located in the same phase, if $\mathbf{q}$ is closer to the interface than $\mathbf{p}$, then Alg. 2 will take less computational time (less pseudo-temporal iterations) to reinitialize ϕ on the summits of the triangle containing point $\mathbf{q}$ than on node $\mathbf{p}$. As a consequence, at any given pseudo-temporal iteration n (or any given pseudo-time τ), the interpolated reinitialized ϕ^n value on $\mathbf{q}$ will be more or equally accurate than the reinitialized ϕ^n value on $\mathbf{p}$: $\phi^n(\mathbf{q})$ will be more accurate than $\phi^n(\mathbf{p})$ if $\mathbf{q} \in \Pi(\tau)$ and $\mathbf{p} \notin \Pi(\tau)$, and $\phi^n(\mathbf{q})$ will be equally accurate as $\phi^n(\mathbf{p})$ if both $\mathbf{p}, \mathbf{q} \in \Pi(\tau)$. As a result, the causality principle is enforced by the update of $\phi^n(\mathbf{p})$ by means of $\phi^{n-1}(\mathbf{q})$, i.e. $\phi^n(\mathbf{p}) = \phi^{n-1}(\mathbf{q}) + \Delta\tau$. Transposed to Figs. E.5(b) and E.5(c), the causality principle leads to Eq. (E.22).

E.5.2.3 Modification of the pseudo-time step for MPI parallelism

Since at any given pseudo-time τ, the accuracy of the reinitialized ϕ nodal values depends on the node distance to the interface ($\Pi(\tau - 2\Delta\tau) \subset \Pi(\tau - \Delta\tau) \subset \Pi(\tau) \subset \dots$), the convergence of Alg. 2 on a given node $\mathbf{x}$ can be accelerated if the target points $\mathbf{x}_{\text{target}}$ of $\mathbf{x}$ are chosen close to the interface. For instance, in Fig. E.5, the convergence speed of Alg. 2 on node $\mathbf{x}$ is increased by fixing $\Delta\tau$ such that $\mathbf{x}_{\text{target}}$ is expected to be close to $\mathbf{x}'$. Since this remark is valid for each node $\mathbf{x}$, the overall convergence of Alg. 2 can be accelerated if the pseudo-time step $\Delta\tau$ is node-dependent, i.e. if $\Delta\tau = \Delta\tau(\mathbf{x})$. However, if $\Delta\tau$ is node-dependent, for a given node $\mathbf{x}$, $\Delta\tau(\mathbf{x})$ must be adjusted with caution. Indeed, if the shoot vectors $\mathbf{s}(\mathbf{x})$ cross the interface, Eq. (E.22) can provide inaccurate values for $\phi^n(\mathbf{x})$, as shown in Fig. E.7. In [17], the authors use another strategy :

> [...] the time step [$\Delta\tau$] must be chosen small enough at the beginning of the process, so that going back along the characteristic lines does not lead to crossing the interface [Γ] and picking irrelevant values. But after a certain amount of iterations, we can obviously increase this time step. [...]

As a result, the different areas $\Pi^n(\tau)$ shown in Fig. E.6, where ϕ^n has been reinitialized, do not have the same increment thickness. For *small* values of τ, $\Pi(\tau)$ is quite thin, and gets thicker for larger values of τ, in which case attention is still paid to build the target points in the same phase. Propagation of $\widetilde{\Gamma}$ along characteristics of Γ at speed 1 is also used to speed up computations :

> [...] we decided to fix the values computed at a node $\mathbf{x}$ when these values are smaller than the current total time of the propagation (with a security margin).

Indeed, if node $\mathbf{x}$ belongs to $\Pi(\tau)$, then the propagating fictitious interface $\widetilde{\Gamma}(\tau)$ has already crossed node $\mathbf{x}$, so the reinitialized signed distance value on node $\mathbf{x}$ can be considered accurate and need not be reinitialized anymore.

In [17], the authors developed the method for use with shared memory only (OpenMP). Since our goal is to simulate boiling on unstructured grids in multidimensions, we need an algorithm compatible with distributed memory for execution on multiple cores (MPI). The next part of this section presents the modification adopted for the pseudo-time step to enable MPI parallelism.

In parallel computing, if the boundary between two processes separates node $\mathbf{x}$ from point $\mathbf{x}_{\text{target}}^{n-1}$, then a procedure is needed for $\mathbf{x}$ on one process to :

- locate the triangle $T_{\text{target}}^{n-1}(\mathbf{x})$ containing $\mathbf{x}_{\text{target}}^{n-1}$ on the other process,
- interpolate ϕ^{n-1} on $\mathbf{x}_{\text{target}}^{n-1}$ from ϕ^{n-1} values in the plane associated to $T_{\text{target}}^{n-1}(\mathbf{x})$,
- send the interpolated $\phi^{n-1}\left(\mathbf{x}_{\text{target}}^{n-1}\right)$ value to the process containing node $\mathbf{x}$ to solve Eq. (E.13).

Such communication scheme is highly complex to implement in an efficient way. To simplify the problem, we impose that $\mathbf{x}_{\text{target}}^{n-1}$ is located in a triangle to which $\mathbf{x}$ is a summit, i.e. a triangle $T \in \mathcal{L}_1(\mathbf{x})$. This restriction is imposed via the pseudo-time step $\Delta\tau$ which is set to

$$\Delta\tau = \eta_{\text{CFL}} \times h_{\min}, \tag{E.24}$$

where η_{CFL} is the non-dimensional Courant-Friedrichs-Lewy (CFL) number set to 0.99 and $h_{\min}$ is the smallest height of all grid triangles. Figure E.8 illustrates this restriction on $\Delta\tau$ in a domain decomposed for MPI parallelism. Equation (E.24) ensures that the distance between $\mathbf{x}$ and $\mathbf{x}_{\text{target}}^{n-1}$ is always smaller than the smallest height of all triangles to which $\mathbf{x}$ is a summit, i.e. $\mathbf{x}_{\text{target}}^{n-1}$ is always located in a triangle $T_{\text{target}}^{n-1}(\mathbf{x}) \in \mathcal{L}_1(\mathbf{x})$. As a result, for a given node $\mathbf{x}$, since $\mathcal{L}_1(\mathbf{x})$ is pre-computed, the identification of $T_{\text{target}}^{n-1}(\mathbf{x})$ is much easier and suitable to MPI parallelism. In Fig. E.8, since node $\mathbf{x}$ is not located on a process boundary, $T_{\text{target}}^{n-1}(\mathbf{x})$ is identified among the elements of $\mathcal{L}_1(\mathbf{x})$. Conversely, for node $\mathbf{y}$ located on the boundary of processes P_1 and P_2, since neighbor element lists are local to one process, only process P_2 containing $\mathbf{y}_{\text{target}}^{n-1}$ will identify the appropriate triangle in its list $\mathcal{L}_1^{P_2}(\mathbf{y})$. Process P_2 will compute the new value $\phi^n(\mathbf{y})$ by Eq. (E.13) and the result will be communicated to Process P_1, so that both processes P_1 and P_2 have the same $\phi^n(\mathbf{y})$ value. The same method is used if node $\mathbf{y}$ belongs to more than two processes. In all cases, one has

$$\mathcal{L}_1(\mathbf{x}) = \bigcup_i \mathcal{L}_1^{P_i}(\mathbf{x}), \tag{E.25}$$

where $i \in \mathbb{N}^*$ is the number of processes containing node $\mathbf{x}$.

E.5.2.4 Linear interpolation of ϕ^{n-1} on the target point

Equation (E.13) requires the knowledge of ϕ^{n-1} on point $\mathbf{x}_{\text{target}}^{n-1}$. Once the triangle T_{target}^{n-1} containing $\mathbf{x}_{\text{target}}^{n-1}$ is identified, a linear interpolation is used to evaluate ϕ^{n-1} at $\mathbf{x}_{\text{target}}^{n-1}$ from the ϕ^{n-1} values at the summits of T_{target}^{n-1}.

In order to identify the triangle T_{target}^{n-1} containing $\mathbf{x}_{\text{target}}^{n-1}$, the following test is performed on each triangle $T \in \mathcal{L}_1(\mathbf{x})$. Let A, B and C be the three summits of T. For clarity, we denote $\mathbf{x}_{\text{target}}^{n-1}$ by M. If one has

$$\begin{cases} (\mathbf{AB} \times \mathbf{AM}) \cdot (\mathbf{AB} \times \mathbf{AC}) > 0, \\ (\mathbf{BC} \times \mathbf{BM}) \cdot (\mathbf{BC} \times \mathbf{BA}) > 0, \\ (\mathbf{CA} \times \mathbf{CM}) \cdot (\mathbf{CA} \times \mathbf{CB}) > 0, \end{cases} \tag{E.26}$$

then M belongs to triangle T. Figure E.9 shows a case where this test can detect point M in triangle ABC. It is well-known that, in the case where M is *very* close to one edge, this test will not detect the triangle containing M due to numerical errors at the machine precision level. For instance, if M is inside triangle T and very close to edge AB, then $\mathbf{AB} \times \mathbf{AM}$ will be almost equal to $\mathbf{0}$, and in turn $(\mathbf{AB} \times \mathbf{AM}) \cdot (\mathbf{AB} \times \mathbf{AC})$ will be equal to $\epsilon > 0$ where ϵ is arbitrarily close to 0. While mathematically well-defined, this equality is very challenging to detect numerically since $\epsilon > 0$ can be much smaller than the smallest real number representable on the current machine, i.e. the machine precision is potentially insufficient to represent ϵ. In such case, while mathematically true, the numerical test $(\mathbf{AB} \times \mathbf{AM}) \cdot (\mathbf{AB} \times \mathbf{AC}) > 0$ will randomly return `true` or `false`. The problem cannot be solved using a more precise machine since ϵ is arbitrarily close to 0. The operator "$>$" in the test cannot be replaced by "$\geq$" since testing the equality of two real numbers is subjected to the same problem. As a result, point M can be detected in none, one or more than one triangles. Figure E.10 illustrates the difficulty in locating the target point M of node C in triangle ABC, in two different cases : (a), when M is close to an edge linked to C, and (b), when M is close to an edge not linked to C. For instance, in the case of Fig. E.10(a), test (E.26) can detect point M :

- only in triangle ABC,
- only in triangle ACD,
- in both triangles ABC and ACD,
- in none triangle.

If test (E.26) detects M in one single triangle T of summits A, B and C, then one can interpolate ϕ on M using ϕ values on A, B and C, denoted ϕ_A, ϕ_B and ϕ_C. Let P be the plane in $\mathbb{R}^3$ defined by the three points $A'(x_A, y_A, \phi_A)$, $B'(x_B, y_B, \phi_B)$ and $C'(x_C, y_C, \phi_C)$, and $\mathbf{n}_P(n_x, n_y, n_z)$ the normal vector to plane P given by Eq. (E.9), as shown in Fig. E.4. Since $A', B', C' \in P$, one has

$$\overrightarrow{M'A'} \cdot \mathbf{n}_P = \overrightarrow{M'B'} \cdot \mathbf{n}_P = \overrightarrow{M'C'} \cdot \mathbf{n}_P = 0, \tag{E.27}$$

where $M' \in \mathbb{R}^3$ is the point of coordinates $(x_M, y_M, \phi_M)^{\mathrm{T}}$, and the only unknown is ϕ_M. The interpolated ϕ value on M is given by

$$\phi_M = \phi_A + \frac{(x_A - x_M)\, n_x + (y_A - y_M)\, n_y}{n_z}, \tag{E.28}$$

where A can be replaced by B or C, and n_z is always non zero.

If test (E.26) fails to find M in all triangles in $\mathcal{L}_1(\mathbf{x})$, then, due to the restriction on $\Delta\tau$ defined by Eq. (E.24), one can conclude at least that M is *infinitely close* to one grid edge $P_1P_2 \in \mathcal{L}_2(\mathbf{x})$, as in Fig. E.10. In this case, the edge P_1P_2 is identified, and ϕ is interpolated on M using ϕ_{P_1} and/or ϕ_{P_2}. In order to identify P_1P_2, a loop over the elements of $\mathcal{L}_2(\mathbf{x})$ is used. This list contains the identifiers of all edges forming triangles to which $\mathbf{x}$ is a summit. For instance, in Fig. E.10, the neighbor element edge list $\mathcal{L}_2(\mathbf{x})$, where node $\mathbf{x}$ is identified as point C, contains the identifiers of edges AB, BC, AC, CD, AD, BE, CE and DE. Let P_1P_2 be the edge of $\mathcal{L}_2(\mathbf{x})$ to which point M is the closest. The edge P_1P_2 is defined by

$$\frac{\|P_1M\| + \|P_2M\|}{\|P_1P_2\|} = \min_{PQ \in \mathcal{L}_2(\mathbf{x})} \left\{ \frac{\|PM\| + \|QM\|}{\|PQ\|} \right\} \geq 1. \tag{E.29}$$

In Fig. E.10(a), Eq. (E.29) sets P_1P_2 to AC, and in Fig. E.10(b), to AB. The interpolated ϕ value on M is then given by

$$\phi_M = \begin{cases} \phi_{P_1}, & \text{if } |\|P_1P_2\| - \|P_2M\|| < \epsilon, \\ \phi_{P_2}, & \text{if } |\|P_1P_2\| - \|P_1M\|| < \epsilon, \\ (1-\theta)\,\phi_{P_1} + \theta\phi_{P_2}, & \text{otherwise,} \end{cases} \tag{E.30}$$

where ϵ is the smallest real number representable on the current machine, typically of the order of 10^{-15}, and $\theta = \|P_1M\| \,/\, \|P_1P_2\|$ is the relative distance of M to P_1 on edge P_1P_2.

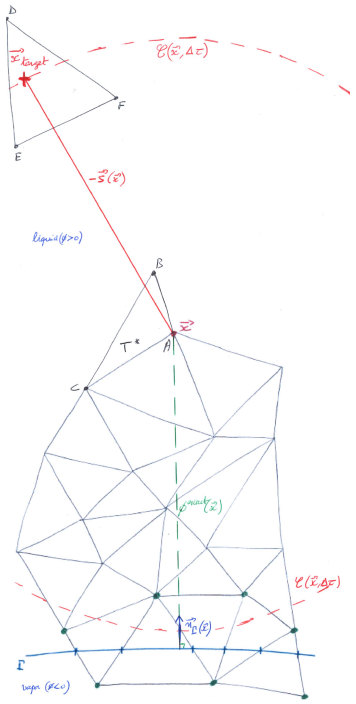
Finally, the signed distance function ϕ is updated on node C by Eq. (E.1) which rewrites

$$\phi^n(\mathbf{x}) = \min\left\{\phi^{n-1}(\mathbf{x}), \phi_M^{n-1}\right\} + \Delta\tau. \tag{E.31}$$

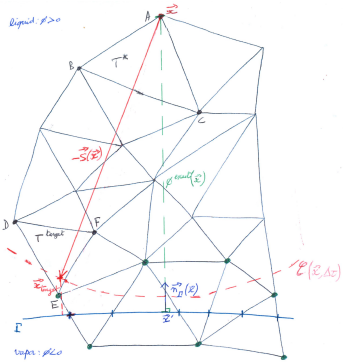
Figure E.11 shows the pseudo-temporal iterations needed to reinitialize the signed distance function ϕ to the interface Γ on node $\mathbf{x}$, by means of the different tilt directions of the associated planes.

E.6 Implementation in three dimensions

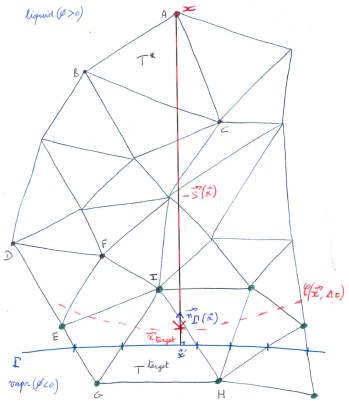
The method has been extended to three dimensions. Similarly to the two-dimensional case which involves geometric operations in $\mathbb{R}^3$ (see Fig. E.4), the three-dimensional case involves geometric operations in $\mathbb{R}^4$. Consequently, operations based on the three-dimensional vector product operator are adapted in four dimensions. The overall logic of the method remains unchanged.



(a) First iteration of Eq. (E.13) solved on node $\mathbf{x}$. The shoot vector $\mathbf{s}^{n-1}(\mathbf{x})$ of node $\mathbf{x}$ points to a random direction. As a result, Proposition P is false, and $\phi^n(\mathbf{x})$ is updated by Eq. (E.23).



(b) Unsteady state of Eq. (E.13) solved at node $\mathbf{x}$. The shoot vector $\mathbf{s}^{n-1}(\mathbf{x})$ of node $\mathbf{x}$ is not yet colinear to the interface normal vector on $\mathbf{x}$, $\mathbf{n}_\Gamma(\mathbf{x})$, but Proposition P is true : $\phi^n(\mathbf{x})$ is updated by Eq. (E.22) but the obtained value is not yet accurate.



(c) Steady state of Eq. (E.13) solved on node $\mathbf{x}$. The shoot vector $\mathbf{s}^{n-1}(\mathbf{x})$ of node $\mathbf{x}$ is colinear to the interface normal vector on $\mathbf{x}$, $\mathbf{n}_\Gamma(\mathbf{x})$. Proposition P is true. Since $\mathbf{x}'^{,n-1}$, $\mathbf{x}^{n-1}_{\text{target}}$ and $\mathbf{x}$ are aligned, Eq. (E.22) provides an accurate value for $\phi^n(\mathbf{x})$.

Fig. E.5 – Equation (E.13) is solved on node $\mathbf{x}$. The plane of ϕ^{n-1} values associated to the triangle T^* maximizes $\left\|\nabla\left(\phi^{n-1}\big|_T\right)\right\|$ among all planes associated to triangles $T \in \mathcal{L}_1(\mathbf{x})$ (see Fig. E.4). The target point $\mathbf{x}^{n-1}_{\text{target}}$ of node $\mathbf{x}$ is defined by Eq. (E.20), and the distance between node $\mathbf{x}$ and point $\mathbf{x}^{n-1}_{\text{target}}$ is equal to $\Delta\tau$. The point $\mathbf{x}^{n-1}_{\text{target}}$ is not a grid node, so there is no defined ϕ^{n-1} value on $\mathbf{x}^{n-1}_{\text{target}}$, but ϕ^{n-1} can be linearly interpolated on $\mathbf{x}^{n-1}_{\text{target}}$ from ϕ^{n-1} values computed at the summits of triangle $T^{n-1}_{\text{target}}(\mathbf{x})$, leading to $\phi^{n-1}\left(\mathbf{x}^{n-1}_{\text{target}}\right)$. The exact, unknown ϕ^{exact} value on $\mathbf{x}$ can be expressed as $\phi^{\text{exact}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}'\|$, where $\mathbf{x}'$ is the intersection point between the interface Γ and the line generated by $\mathbf{x}_{\text{target}} - \mathbf{x}$ when $\mathbf{x}_{\text{target}}$ has reached steady state. Subfigs. E.5(a) and E.5(b) show two different iterations of Eq. (E.13), and subfig. E.5(c) shows its steady state.

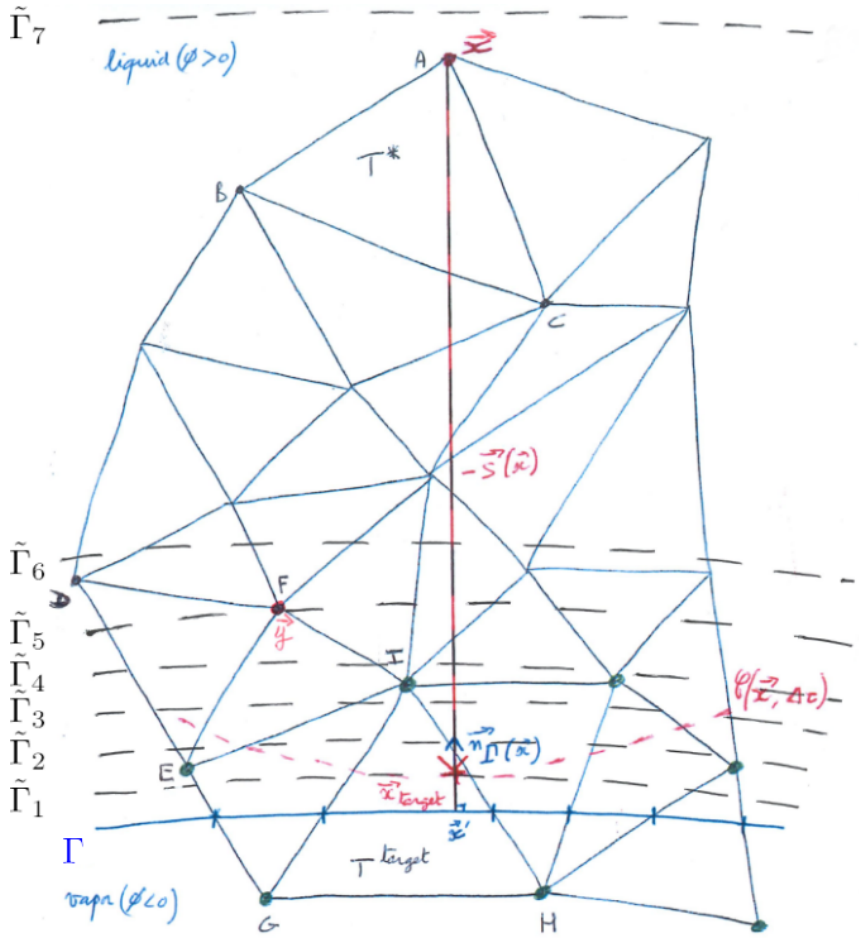


Fig. E.6 – Steady state of Eq. (E.13) solved on node $\mathbf{x}$. Reproduction of Fig. E.5(c) where different positions of the fictitious interface $\widetilde{\Gamma}$ are represented. In the first iterations, the pseudo-time step $\Delta\tau$ is small enough to ensure that target points $\mathbf{y}_{\text{target}}$ and node $\mathbf{y}$ are located in the same phase, resulting in thin Π areas. After a certain number of iterations, $\Delta\tau$ is increased to speed up convergence by enabling far nodes to the interface, like node $\mathbf{x}$, to shoot at target points $\mathbf{x}_{\text{target}}$ closer to the interface.

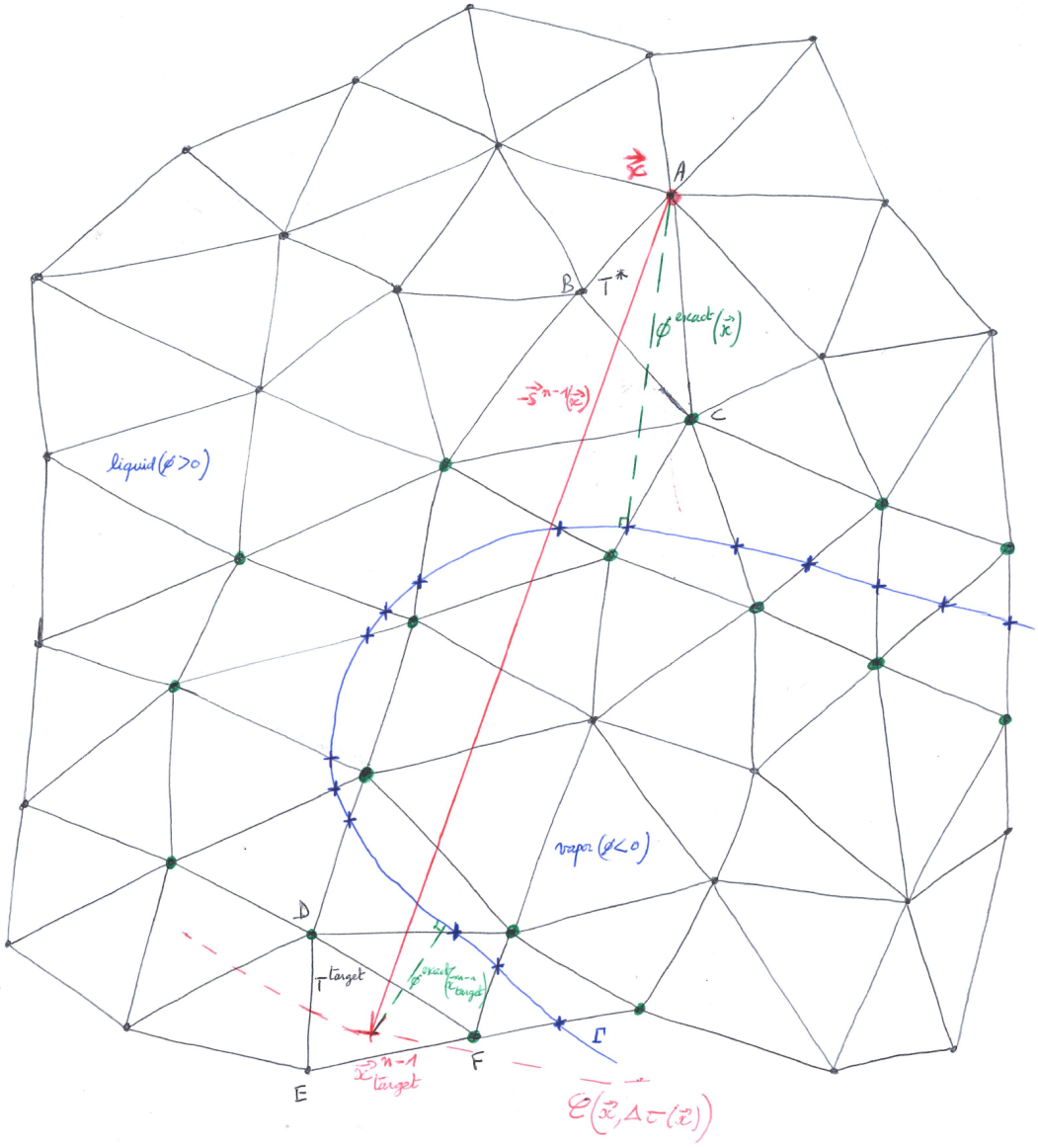


Fig. E.7 – In this example, the pseudo-time step $\Delta\tau(\mathbf{x})$ is node-dependent. The shoot vector $\mathbf{s}^{n-1}(\mathbf{x})$ is almost colinear to the interface normal vector $\mathbf{n}_\Gamma(\mathbf{x})$ but crosses the interface. Proposition P is true but Eq. (E.22) fails to update $\phi^n(\mathbf{x})$ accurately.

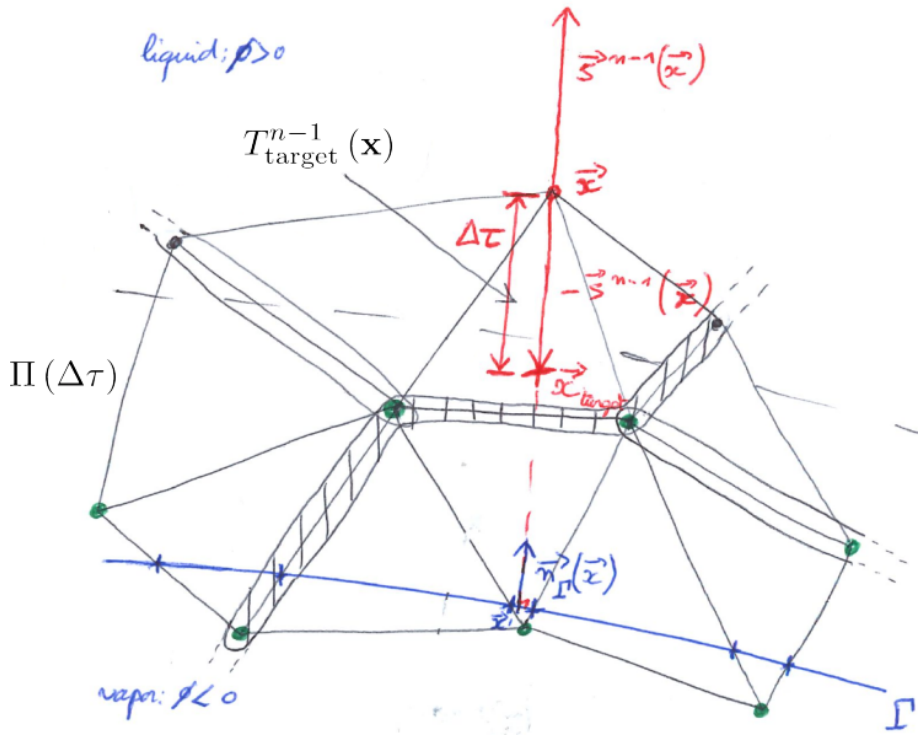


Fig. E.8 – Double decomposition of the computational domain on two processes. The hatched area is the boundary between the processes and the circled areas are the boundaries between element groups of the same process. The restriction on $\Delta\tau$ imposed by Eq. (E.24) implies that target point $\mathbf{x}^{n-1}_{\text{target}}$ of node $\mathbf{x}$ is contained in one triangle $T^{n-1}_{\text{target}}(\mathbf{x}) \in \mathcal{L}_1(\mathbf{x})$. From node $\mathbf{x}$, it is then easier to access triangle $T^{n-1}_{\text{target}}(\mathbf{x})$ (looping only over elements of $\mathcal{L}_1(\mathbf{x})$ and not over all grid elements), interpolate ϕ^{n-1} on $T^{n-1}_{\text{target}}(\mathbf{x})$, and update $\phi^n(\mathbf{x})$ by Eq. (E.13).

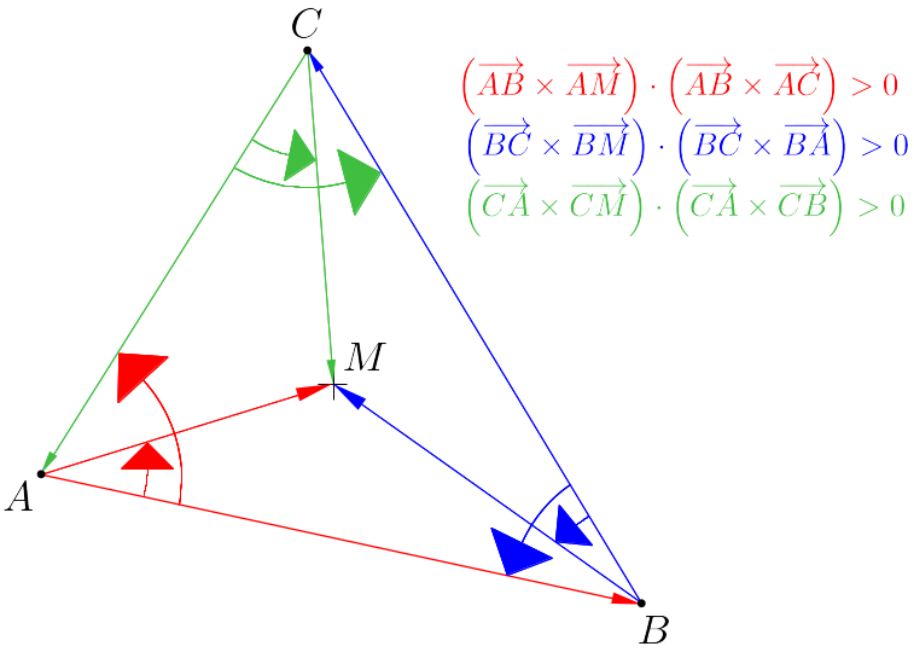
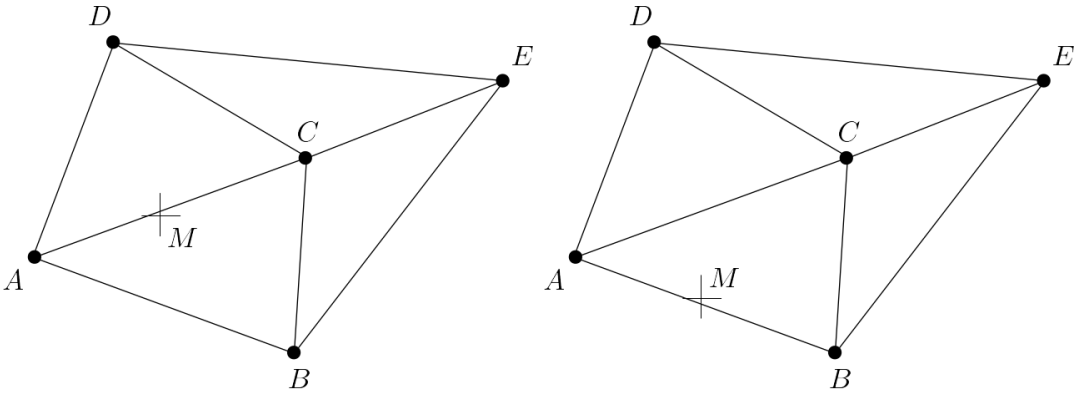


Fig. E.9 – Point M is easily detected in triangle ABC by test (E.26). The three scalar products of test (E.26) are evaluated positive without numerical problems since point M is *sufficiently* far from the edges of triangle ABC.



(a) Point M is close to edge AC, directly related to node $\mathbf{x}_C$.

(b) Point M is close to edge AB, not directly related to node $\mathbf{x}_C$.

Fig. E.10 – In cases (a) and (b), point C represents a computational node $\mathbf{x}_C$. Test (E.26) will most likely fail to identify triangle ABC as the triangle containing the target point M of node $\mathbf{x}_C$, computed using Eq. (E.20). Case (b) shows that $\mathcal{L}_2(\mathbf{x}_C)$ must also contain edges not directly related to $\mathbf{x}_C$.

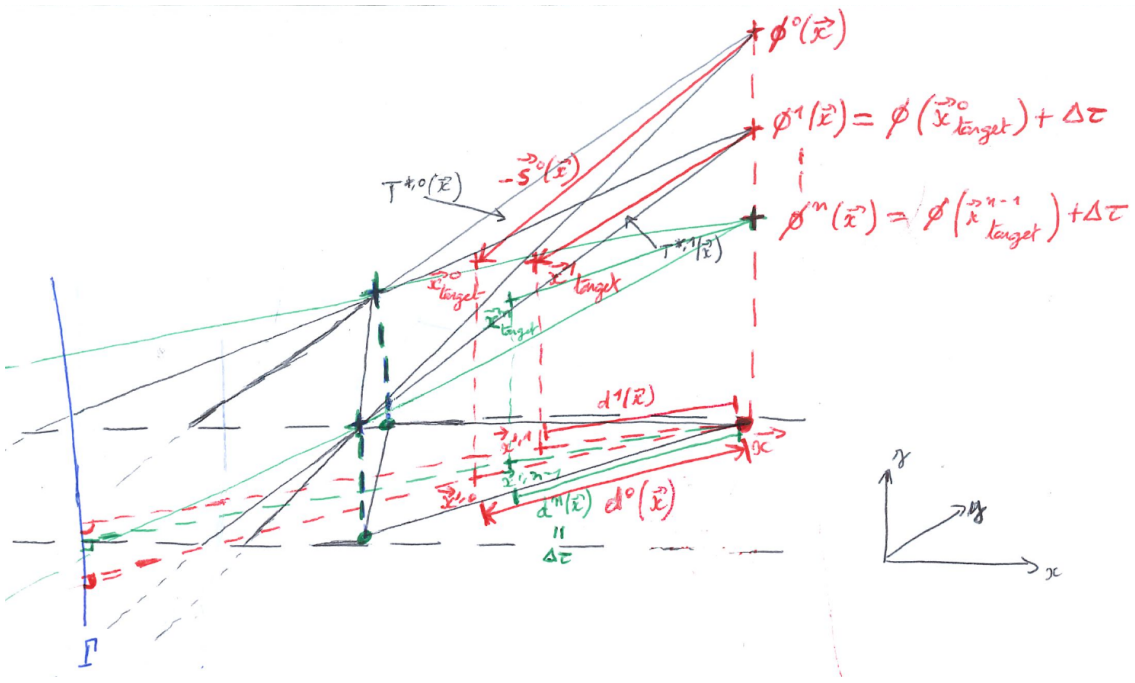


Fig. E.11 – For each iteration n, the distance between node $\mathbf{x}$ and point $\mathbf{x}^{n-1}_{\text{target}}$ is equal to $\Delta\tau$. After the last iteration N, one has $\nabla\left(\phi|_{T^*(\mathbf{x})}\right) = \mathbf{n}_\Gamma(\mathbf{x})$, i.e. the tilt direction of the final plane P^N is equal to $\mathbf{n}_\Gamma(\mathbf{x})$, and one has $\phi^N(\mathbf{x}) = \phi^{N-1}(\mathbf{x}^{N-1}_{\text{target}}) + \Delta\tau$ by Eq. (E.22).

Appendix F

Reinitialization of the Signed Distance Function by the Geometric Marker Method in two and three dimensions with MPI parallelism

Abstract

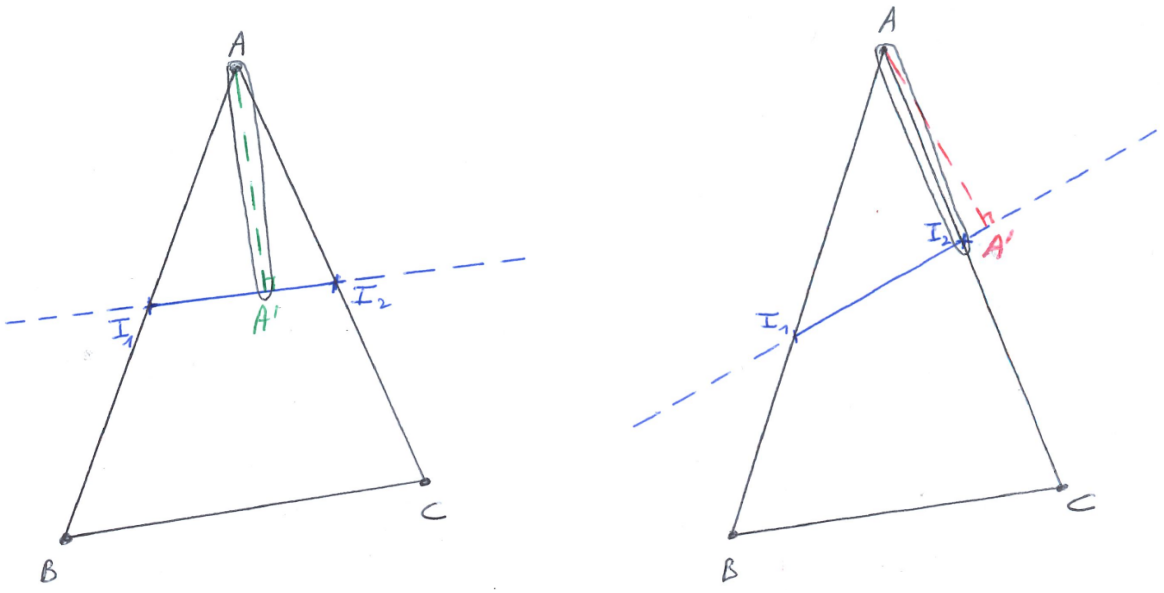
This Appendix details a new marker-based reinitialization method for the signed distance function on unstructured grids. This method is an extension of the Multiple Marker Method presented in Section 5.5.

Outline

The Geometric Marker Method (GMM), developed by Janodet et al. [39], is based on the Multiple Marker Method (MMM) presented in Section 5.5. The GMM extends the notion of marker from point to segment.

F.1 The Geometric Marker Method in two dimensions

Let $\mathcal{T}$ be a triangle of summits $A\left(x_A, y_A\right)$, $B\left(x_B, y_B\right)$ and $C\left(x_C, y_C\right)$. Points A, B and C represent grid nodes $\mathbf{p}_A$, $\mathbf{p}_B$ and $\mathbf{p}_C$ respectively, and segments AB, AC and BC represent grid edges $(\mathbf{p}_A, \mathbf{p}_B)$, $(\mathbf{p}_A, \mathbf{p}_C)$ and $(\mathbf{p}_B, \mathbf{p}_C)$ respectively. Assume that the piecewise-linear interface Γ crosses triangle $\mathcal{T}$ on segments AB and AC at points $I_1\left(x_{I_1}, y_{I_1}\right)$ and $I_2\left(x_{I_2}, y_{I_2}\right)$ respectively. This configuration is shown in Fig. F.1. In the GMM, markers are first defined for the closest nodes to the interface, such as node $\mathbf{p}_A$. In order to define a marker for node $\mathbf{p}_A$, one first computes the



(a) A' is inside the triangle. (b) A' is outside the triangle.

Fig. F.1 – Computation of the orthogonal projection A' of point A onto the piecewise-linear interface delimited by I_1 and I_2.

projection A' of point A onto the piecewise-linear interface Γ. Let $d_{\mathcal{T}}(\mathbf{p}_A)$ be the distance of node $\mathbf{p}_A$ to the piecewise-linear interface part contained in triangle $\mathcal{T}$ given by

$$d_{\mathcal{T}}(\mathbf{p}_A) = \begin{cases} \left\| \overrightarrow{AA'} \right\|, & \text{if } A' \in \mathcal{T} \text{ (Fig. F.1(a))}, \\ \min\left\{ \left\| \overrightarrow{AI_1} \right\|, \left\| \overrightarrow{AI_2} \right\| \right\}, & \text{otherwise (Fig. F.1(b))}. \end{cases} \tag{F.1}$$

Triangle $\mathcal{T}$ enables the definition of one marker $M_{\mathcal{T}}(\mathbf{p}_A)$ associated to node $\mathbf{p}_A$, given by the 5-dimensional vector

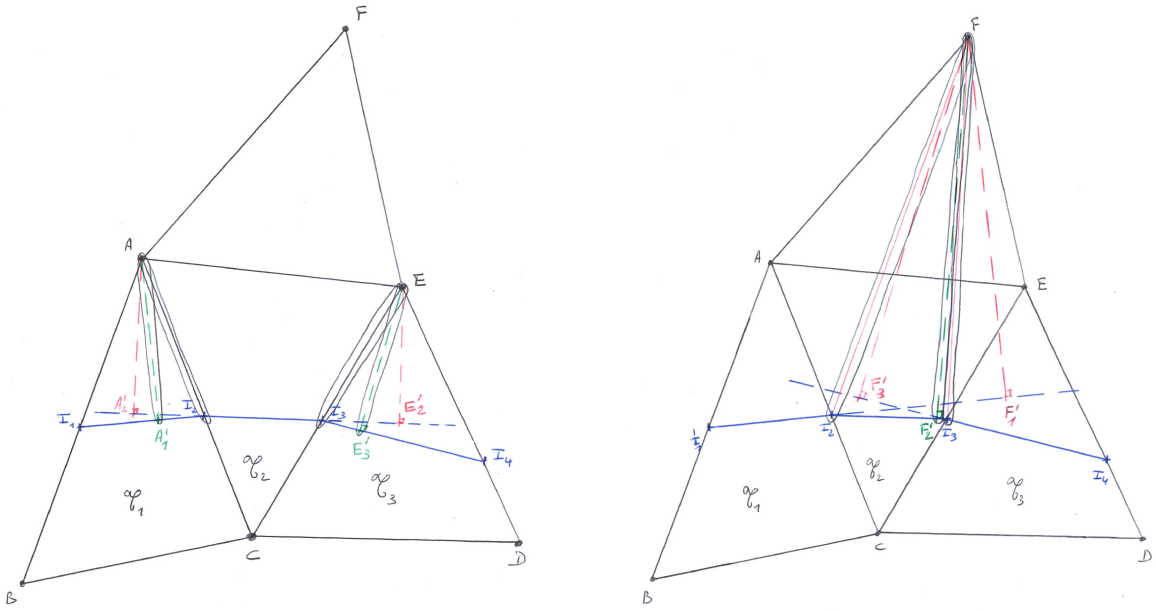
$$M_{\mathcal{T}}(\mathbf{p}_A) = (x_{I_1}, y_{I_1}, x_{I_2}, y_{I_2}, d_{\mathcal{T}}(\mathbf{p}_A))^{\mathrm{T}}. \tag{F.2}$$

The signed distance function ϕ on node $\mathbf{p}_A$ is then given by

$$\phi(\mathbf{p}_A) = \min_{i \in \{1,\dots,N\}} d_{\mathcal{T}_i}(\mathbf{p}_A), \tag{F.3}$$

where $N \in \mathbb{N}^*$ is the number of triangles crossed by the interface and to which point A is a summit. In two dimensions, the marker list $\mathcal{M}(\mathbf{p}_A)$ associated to node $\mathbf{p}_A$ is composed of three markers. In order to select these three markers, all markers $M_{\mathcal{T}_i}(\mathbf{p}_A)$ are sorted in ascending order of their fifth component. The first three markers of the sorted list compose the list $\mathcal{M}(\mathbf{p}_A)$. As a result, the fifth component $d_{\mathcal{T}_j}(\mathbf{p}_A)$ of the sorted list first marker $M_{\mathcal{T}_j}(\mathbf{p}_A)$ is equal to the reinitialized ϕ value on $\mathbf{p}_A$ given by Eq. (F.3), for some $j \in \{1, \dots, N\}$.

Once ϕ has been reinitialized on all the closest nodes to the interface, these nodes propagate their marker list in the narrow band, as shown in Fig. F.2. In Fig. F.2(a), triangles $\mathcal{T}_1$ and $\mathcal{T}_2$ each



(a) (b)

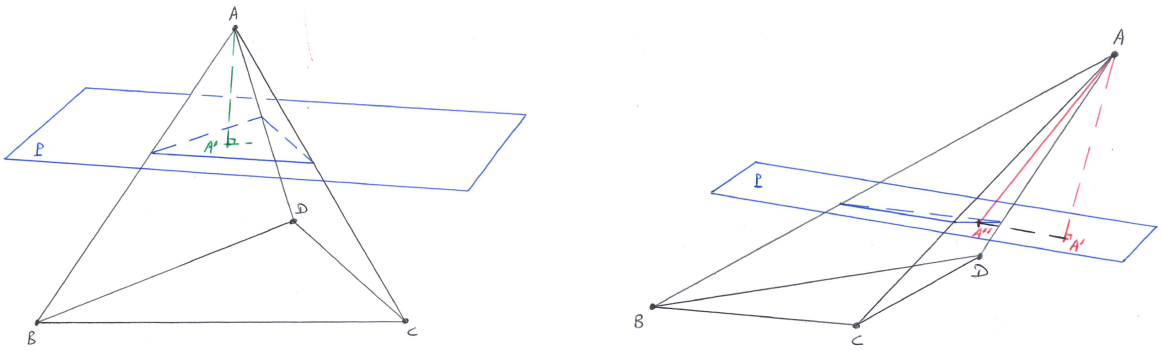
Fig. F.2 – Reinitialization of the signed distance function by the GMM on (a) nodes of band level 1, and (b) nodes of band level 2.

give one marker to node $\mathbf{p}_A$, and triangles $\mathcal{T}_2$ and $\mathcal{T}_3$ each give one marker to node $\mathbf{p}_E$. In this case, all components of the third markers associated to nodes $\mathbf{p}_A$ and $\mathbf{p}_E$ are set to $+\infty$. In turn, node $\mathbf{p}_F$ builds its list of three markers among the six propagated markers from nodes $\mathbf{p}_A$ and $\mathbf{p}_E$, as shown in Fig. F.2(b). Orthogonal projections F' of point F are still computed onto lines generated by linear-piecewise interface segments I_1I_2, I_2I_3 and I_3I_4, and only the three markers minimizing $d_{\mathcal{T}}(\mathbf{p}_F)$ are selected (with a unicity test on marker segments, since for instance triangle $\mathcal{T}_2$, by means of nodes $\mathbf{p}_A$ and $\mathbf{p}_E$, will provide node $\mathbf{p}_F$ with two different markers related to segment $[I_2, I_3]$).

The propagation of marker lists relies on a complex MPI communication scheme originally developed for the MMM.

F.2 The Geometric Marker Method in three dimensions

In three dimensions, the same methodology is applied for tetrahedra instead of triangles. Let $\mathcal{T}$ be a tetrahedron of summits $A(x_A, y_A, z_A)$, $B(x_B, y_B, z_B)$, $C(x_C, y_C, z_C)$ and $D(x_D, y_D, z_D)$, and Γ be the piecewise-planar interface cutting $\mathcal{T}$. The intersection between Γ and $\mathcal{T}$ is either a triangle or a quadrangle. In the latter case, the quadrangle is split into two triangles. As in Fig. F.1 for the two-dimensional case, the signed distance function is reinitialized on the closest nodes to the interface by computing their projections onto the piecewise-planar interface, as shown in Fig. F.3. The reinitialization of ϕ on the farther nodes is then similar to the method used for the two-dimensional case shown in Fig. F.2(b).



(a) A' is inside the tetrahedron.

(b) A' is outside the tetrahedron.

Fig. F.3 – Computation of the orthogonal projection A' of point A onto the piecewise-planar interface.

Appendix G

High-order extrapolation in the interface normal direction

Outline

G.1 Introduction

In order to satify requirements 3 and 4 of Section 6.2.1, we use Aslam's extrapolation technique [4]. This extrapolation method is mentioned in [26] and has been used in [85] to extrapolate ghost temperature values across the interface. The method is based on a PDE resolution and builds a continuous extension of a discontinuous field. Consider a scalar field f defined on one side of the interface. One can continuously extend f on the other side of the interface by solving

$$\frac{\partial f}{\partial \tau} \pm \mathbf{n} \cdot \nabla f = 0, \tag{G.1}$$

where τ is a pseudo-time, $\mathbf{n}$ is the interface normal vector and the $\pm$ sign depends on which phase f is defined. Equation (G.1) performs a constant (order 0) extrapolation of f across the interface. The method has been extended to preserve the continuity of normal derivatives up to an arbitrary order. The chosen order gives the number of supplementary equations to solve. In [85], the authors have performed extrapolations of the temperature fields up to order 2. The three equations solved

to extend the vapor temperature field to the liquid phase are then given by

$$\frac{\partial \left(T_{\text{vap}}\right)_{\mathbf{nn}}}{\partial \tau} + H_{-2}\left(b\right)\mathbf{n}\cdot\nabla\left(T_{\text{vap}}\right)_{\mathbf{nn}} = 0, \tag{G.2}$$

$$\frac{\partial \left(T_{\text{vap}}\right)_{\mathbf{n}}}{\partial \tau} + H_{-1}\left(b\right)\left(\mathbf{n}\cdot\nabla\left(T_{\text{vap}}\right)_{\mathbf{n}} - \left(T_{\text{vap}}\right)_{\mathbf{nn}}\right) = 0, \tag{G.3}$$

$$\frac{\partial T_{\text{vap}}}{\partial \tau} + H_{0}\left(b\right)\left(\mathbf{n}\cdot\nabla T_{\text{vap}} - \left(T_{\text{vap}}\right)_{\mathbf{n}}\right) = 0, \tag{G.4}$$

where the normal derivative $\left(T_{\text{vap}}\right)_{\mathbf{n}}$ is computed as

$$\left(T_{\text{vap}}\right)_{\mathbf{n}} = \nabla T_{\text{vap}}\cdot\mathbf{n}. \tag{G.5}$$

Similarly, the three equations solved to extend the liquid temperature field to the vapor phase are given by

$$\frac{\partial \left(T_{\text{liq}}\right)_{\mathbf{nn}}}{\partial \tau} - H_{2}\left(b\right)\mathbf{n}\cdot\nabla\left(T_{\text{liq}}\right)_{\mathbf{nn}} = 0, \tag{G.6}$$

$$\frac{\partial \left(T_{\text{liq}}\right)_{\mathbf{n}}}{\partial \tau} - H_{1}\left(b\right)\left(\mathbf{n}\cdot\nabla\left(T_{\text{liq}}\right)_{\mathbf{n}} - \left(T_{\text{liq}}\right)_{\mathbf{nn}}\right) = 0, \tag{G.7}$$

$$\frac{\partial T_{\text{liq}}}{\partial \tau} - H_{0}\left(-b\right)\left(\mathbf{n}\cdot\nabla T_{\text{liq}} - \left(T_{\text{liq}}\right)_{\mathbf{n}}\right) = 0. \tag{G.8}$$

In Eqs. (G.2)-(G.4) and (G.6)-(G.8), $H_{b_0}(b)$ is the shifted Heaviside function defined for node $\mathbf{p}$ of band level $b \neq 0$, i.e. node $\mathbf{p}$ belongs to the narrow band around the interface, and $b_0 \in \mathbb{Z}$ by

$$H_{b_0}\left(b\right) = \begin{cases} 1 & \text{if } (b_0 < 0 \text{ and } b \geq b_0) \text{ or } (b_0 > 0 \text{ and } b \leq b_0) \text{ or } (b_0 = 0 \text{ and } b > 0), \\ 0 & \text{otherwise.} \end{cases} \tag{G.9}$$

We now give more detail about the method. We focus on the extension of the vapor temperature field to the liquid phase. The counterpart is of course equivalent. Solving only Eq. (G.4) where the source term $\left(T_{\text{vap}}\right)_{\mathbf{n}}$ is set to 0 performs a constant (order 0) extrapolation, solving Eqs. (G.3) and (G.4) where the source term $\left(T_{\text{vap}}\right)_{\mathbf{nn}}$ is set to 0 performs a linear (order 1) extrapolation, solving Eqs. (G.2), (G.3) and (G.4) performs a quadratic (order 2) extrapolation.

G.2 Constant extrapolation

As stated above, a constant extrapolation can be performed solving the equation

$$\frac{\partial T_{\text{vap}}}{\partial \tau} + H_{0}\left(b\right)\mathbf{n}\cdot\nabla T_{\text{vap}} = 0. \tag{G.10}$$

In order to solve Eq. (G.10), one needs to compute ∇T_{vap}. The temperature gradient cannot be computed using the centered scheme given in Eq. (3.17) : in order to extrapolate values originating from one given direction, one needs to use an upwind scheme where the upwinding is performed w.r.t. the said direction. Refering to Fig. 6.2(b) where node $\mathbf{q}_1$ is the only node located in the vapor phase, Eq. (3.17) would state that the contribution of grid edge $(\mathbf{p}, \mathbf{q}_3)$ to $\nabla T_{\text{vap}}|_{\mathbf{p}}$ is

$\frac{1}{2\mathcal{V}_{\mathbf{p}}}\left(T_{\text{vap}}(\mathbf{p})+T_{\text{vap}}(\mathbf{q}_3)\right)\mathbf{A}_{\mathbf{p},\mathbf{q}_3}$. Instead, since node $\mathbf{p}$ is closer to the vapor phase than node $\mathbf{q}_3$, the upwinding used states that the contribution of grid edge $(\mathbf{p},\mathbf{q}_3)$ to $\nabla T_{\text{vap}}\big|_{\mathbf{p}}$ is

$$\frac{1}{\mathcal{V}_{\mathbf{p}}}T_{\text{vap}}(\mathbf{p})\mathbf{A}_{\mathbf{p},\mathbf{q}_3}, \tag{G.11}$$

and the contribution of grid edge $(\mathbf{p},\mathbf{q}_3)$ to $\nabla T_{\text{vap}}\big|_{\mathbf{q}_3}$ is

$$-\frac{1}{\mathcal{V}_{\mathbf{p}}}T_{\text{vap}}(\mathbf{p})\mathbf{A}_{\mathbf{p},\mathbf{q}_3}. \tag{G.12}$$

The upwinding given by Eqs. (G.11) and (G.12) is performed on all grid edges having at least one node in the liquid part of the narrow band around the interface. The term $H_0(b)$ vanishes in the vapor phase to avoid polluting the vapor part of the vapor temperature field.

G.3 Higher-order extrapolation

Linear extrapolation is achieved in three steps. One first computes the first normal derivative of the vapor temperature for the nodes located in the vapor phase, i.e.

$$\left(T_{\text{vap}}\right)_{\mathbf{n}}=\mathbf{n}\cdot\nabla T_{\text{vap}}, \tag{G.13}$$

where $\mathbf{n}$ is the interface normal vector. Equation (G.13) is applied in all the vapor part of the narrow band around the interface except on the nodes of band level -1, where one or more vapor temperature values would be needed in the liquid part. The first normal derivative of the vapor temperature is then extended into the liquid part by the equation

$$\frac{\partial\left(T_{\text{vap}}\right)_{\mathbf{n}}}{\partial\tau}+H_{-1}\left(b\right)\mathbf{n}\cdot\nabla\left(T_{\text{vap}}\right)_{\mathbf{n}}=0, \tag{G.14}$$

where the gradient of the first normal derivative is computed using the upwind scheme detailed in Section G.2. The function $H_{-1}(b)$ is equal to 1 in the liquid part and also on the nodes of band level -1 in order to extrapolate the first normal derivative which could not be directly computed on these nodes. Once all nodes of the narrow band have a physical or extrapolated value of first normal derivative, the vapor temperature itself is extrapolated into the liquid part of the narrow band by the equation

$$\frac{\partial T_{\text{vap}}}{\partial\tau}+H_0\left(b\right)\left(\mathbf{n}\cdot\nabla T_{\text{vap}}-\left(T_{\text{vap}}\right)_{\mathbf{n}}\right)=0, \tag{G.15}$$

where the first normal derivative is used as a source term and the temperature gradient is again computed using the upwind scheme detailed in Section G.2. Solving Eq. (G.15) until the pseudo-temporal fluctuation term vanishes ensures that the newly extrapolated field T_{vap} and the previously extrapolated field $\left(T_{\text{vap}}\right)_{\mathbf{n}}$ respect Eq. (G.13) on all nodes for which $H_0(b)=1$.

Quadratic extrapolation is based on the same principle : one first computes the first normal derivative by Eq. (G.13) and the second derivative by

$$\begin{aligned}\left(T_{\text{vap}}\right)_{\mathbf{nn}}&=\mathbf{n}\cdot\nabla\left[\left(T_{\text{vap}}\right)_{\mathbf{n}}\right]\\&=\mathbf{n}\cdot\nabla\left(\mathbf{n}\cdot\nabla T_{\text{vap}}\right)\end{aligned} \tag{G.16}$$

on nodes of band level strictly smaller than two, and then solves Eqs. (G.2)-(G.4).

One can remark that it is useless to perform an extrapolation of a field f at an order superior to the order of the polynomial function describing f. For instance, a second-order extrapolation of the signed distance function to the interface, other than for testing purposes, is useless since the second normal derivative of ϕ is zero.

G.4 Numerical results

In order to assess the accuracy of the method, we use the following two-dimensional configuration. The unstructured computational domain is a square of side length $L = 6 \times 10^{-3}\,\mathrm{m}$, with a characteristic cell size $\Delta = 2.5 \times 10^{-5}\,\mathrm{m}$. The point $O(0,0)$ is at the center of the domain. A circular bubble is centered at point O, and the bubble radius is $R_0 = 1 \times 10^{-3}\,\mathrm{m}$. Around 85 nodes are present along the bubble diameter. The liquid temperature profile is defined for all nodes $\mathbf{p}$ of coordinate vector $(x, y)^{\mathrm{T}}$ in the liquid part of the narrow band by

$$T_{\mathrm{liq}}(R) = T_{\mathrm{sat}} + a\,(R - 2R_1 + R_0)\,(R - R_0)\,, \tag{G.17}$$

where $R = \sqrt{x^2 + y^2}$, $T_{\mathrm{sat}} = 373\,\mathrm{K}$, $a = 1 \times 10^7\,\mathrm{K\,m^{-2}}$ and $R_1 = 0.95 \times R_0$. The extrapolation of the liquid temperature is computed in the vapor part of the narrow band at orders 0, 1 and 2. From Eq. (G.17), the theoretical extrapolated value at order 0, $T_{\mathrm{liq}}^{\mathrm{G}^0,\mathrm{th}}$, is given by

$$T_{\mathrm{liq}}^{\mathrm{G}^0,\mathrm{th}} = T_{\mathrm{liq}}(R = R_0) = T_{\mathrm{sat}}. \tag{G.18}$$

Moreover, one has

$$\nabla T_{\mathrm{liq}}(R) = a\,(2R - R_0 - 2R_1 + R_0)\,\nabla R \tag{G.19}$$

$$= 2a\,(R - R_1)\,\mathbf{n}, \tag{G.20}$$

(the gradient of the bubble radius being equal to the interface normal vector), which, for $R = R_0$, yields

$$\nabla T_{\mathrm{liq}}(R_0) = 2a\,(R_0 - R_1)\,\mathbf{n}. \tag{G.21}$$

As a result, the theoretical extrapolated values at order 1, $T_{\mathrm{liq}}^{\mathrm{G}^1,\mathrm{th}}$, are given by

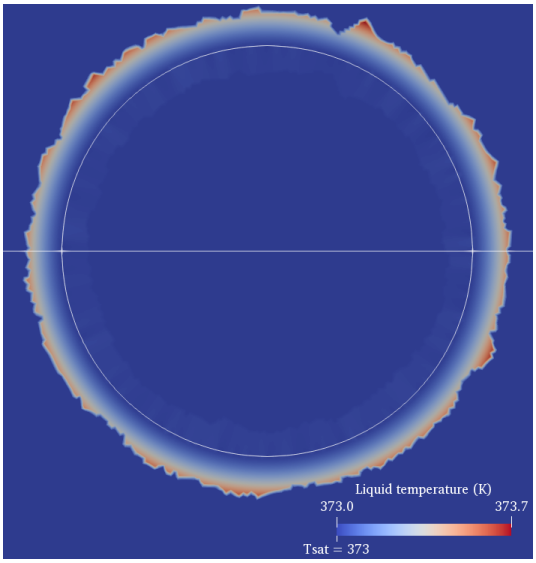
$$T_{\mathrm{liq}}^{\mathrm{G}^1,\mathrm{th}}(R) = T_{\mathrm{sat}} + (R - R_0)\,\mathbf{n} \cdot \nabla T_{\mathrm{liq}}(R_0) \tag{G.22}$$

$$= T_{\mathrm{sat}} + 2a\,(R - R_0)\,(R_0 - R_1)\,, \tag{G.23}$$

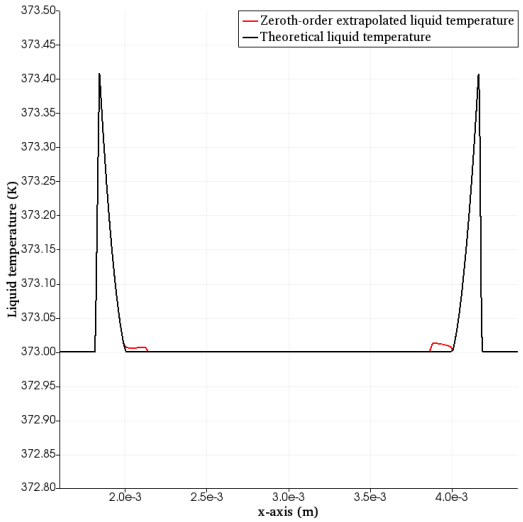
where one can notice that $R - R_0 = \phi(\mathbf{p})$. Since Eq. (G.17) defines T_{liq} as a second-order polynomial function of R, the theoretical extrapolated values at order 2, $T_{\mathrm{liq}}^{\mathrm{G}^2,\mathrm{th}}$, are simply given by

$$T_{\mathrm{liq}}^{\mathrm{G}^2,\mathrm{th}}(R) = T_{\mathrm{liq}}(R). \tag{G.24}$$

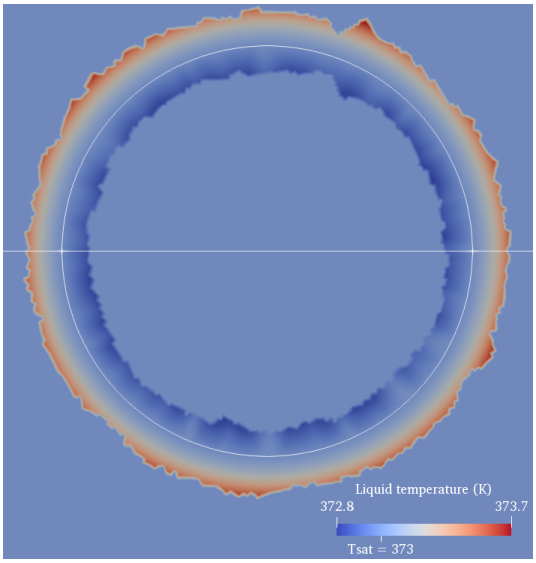
Figure G.1 shows the computed and theoretical extrapolated liquid temperature values for the three extrapolation orders 0, 1 and 2. In the subfigures of the right column, one can see that the extrapolated liquid temperature values, shown in red, do not fit well the theoretical extrapolated values in all tested configurations. For instance, while on the right part of Fig. G.1(d), the first-order extrapolated values are quite accurate, on the left part of Fig. G.1(f), the second-order extrapolated values do not match the theoretical ones. The method has been tested with different grid cell sizes and no accuracy improvement has been observed.


Liquid temperature (K)
373.0
373.7
Tsat = 373

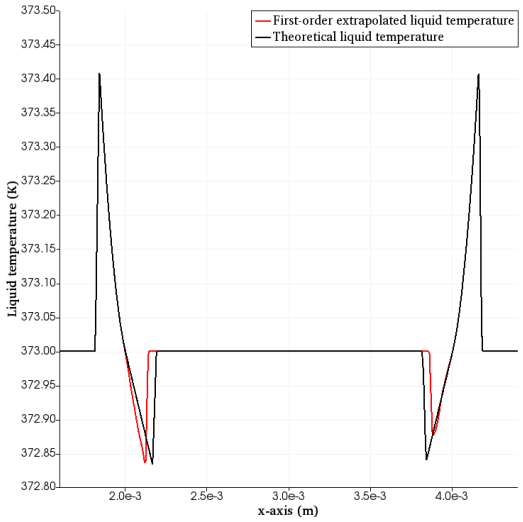
(a)


Zeroth-order extrapolated liquid temperature
Theoretical liquid temperature
Liquid temperature (K)
373.50
373.45
373.40
373.35
373.30
373.25
373.20
373.15
373.10
373.05
373.00
372.95
372.90
372.85
372.80
2.0e-3
2.5e-3
3.0e-3
3.5e-3
4.0e-3
x-axis (m)

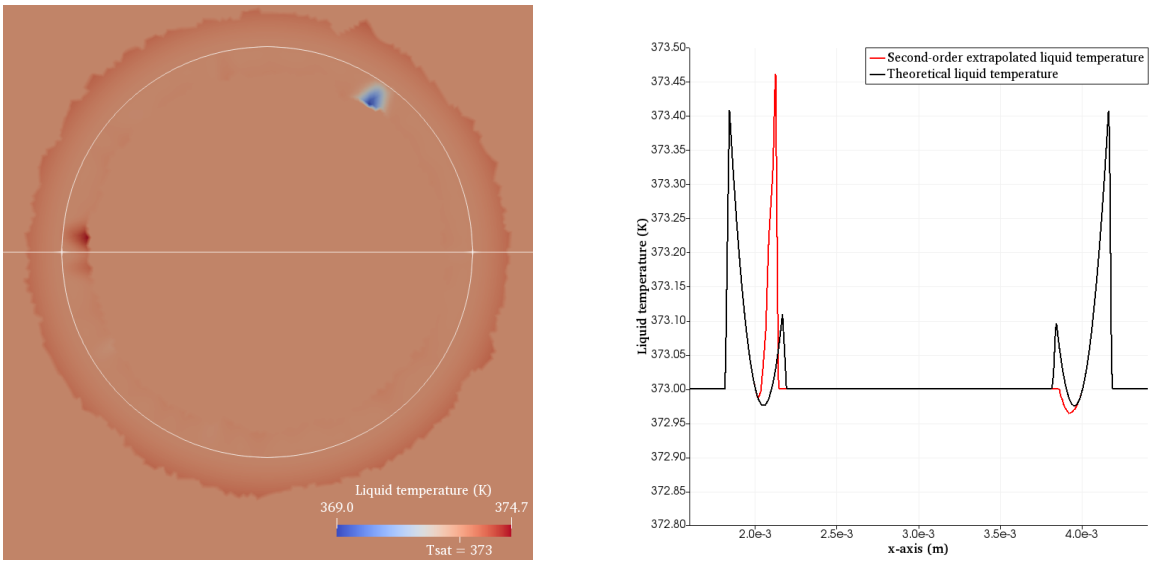
(b)


Liquid temperature (K)
372.8
373.7
Tsat = 373

(c)


First-order extrapolated liquid temperature
Theoretical liquid temperature
Liquid temperature (K)
373.50
373.45
373.40
373.35
373.30
373.25
373.20
373.15
373.10
373.05
373.00
372.95
372.90
372.85
372.80
2.0e-3
2.5e-3
3.0e-3
3.5e-3
4.0e-3
x-axis (m)

(d)



(e) (f)

Fig. G.1 – Extrapolation across the interface of the liquid temperature field along the interface normal vector direction using the method proposed in [4]. Subfigs. (a) and (b) show the constant extrapolation (order 0), Subfigs. (c) and (d) show the linear extrapolation (order 1) and Subfigs. (e) and (f) show the quadratic extrapolation (order 2) of the liquid temperature field in the vapor part of the narrow band around the interface. In the left column, the white circle is the liquid-vapor interface, and the white horizontal line is used to extract liquid temperature values and plot it in the corresponding subfigure of the right column.

Appendix H

Three-dimensional bubble growth: a test case

Abstract

In this appendix, we use the notations adopted in Scriven [75]. The derivations are given in spherical coordinates where the spherical symmetry of the liquid temperature field T is used to cancel components of the differential operators relative to the two angular spatial coordinates.

Outline

H.1 In three dimensions

For simplicity, the liquid temperature will be denoted T where the $_{\text{liq}}$ subscript is omitted. Following the notations of [75], θ will denote the temporal coordinate, r the radial spatial coordinate, and R the bubble radius at time θ. The reference frame origin is assumed to be located at the bubble center.

H.1.1 Some definitions

We first introduce some dimensionless numbers for use in the next sections. Consider

$$\epsilon = 1 - \frac{\rho_{\text{vap}}}{\rho_{\text{liq}}} \tag{H.1}$$

$$= -\rho_{\text{vap}} \left[\frac{1}{\rho}\right]_{\Gamma}, \tag{H.2}$$

$$\nu = \frac{c_{\mathrm{p,liq}} - c_{\mathrm{p,vap}}}{c_{\mathrm{p,liq}}}, \tag{H.3}$$

$$\tau = \frac{T_\infty - T_{\mathrm{sat}}}{T_\infty}, \tag{H.4}$$

$$\xi = \frac{\rho_{\mathrm{vap}} \overline{L}}{\rho_{\mathrm{liq}} c_{\mathrm{p,liq}} T_\infty}, \tag{H.5}$$

where $\overline{L}$ is the mass-averaged latent heat (expressed in $\mathrm{J\,kg^{-1}}$), and

$$\omega = 1 - \epsilon \tag{H.6}$$

$$= \frac{\rho_{\mathrm{vap}}}{\rho_{\mathrm{liq}}}. \tag{H.7}$$

H.1.2 Derivation of the heat equation written in Scriven's formalism

The heat equation written in spherical coordinates is given by

$$\frac{\partial T}{\partial \theta} = K \left(\frac{\partial T}{\partial r^2} + \frac{2}{r} \frac{\partial T}{\partial r} \right) - \mathbf{u} \cdot \frac{\partial T}{\partial r} \mathbf{e}_r, \tag{H.8}$$

where K is the liquid thermal diffusivity and $\mathbf{u}$ is a velocity vector to be determined. A first hypothesis on $\mathbf{u}$ can be assumed : since, as the bulk velocity is zero, $\mathbf{u}$ is only due to phase change, $\mathbf{u}$ is also spherically symmetric, i.e. $\mathbf{u} = u\mathbf{e}_r$. Then, Eq. (H.8) is rewritten

$$\frac{\partial T}{\partial \theta} = K \left(\frac{\partial T}{\partial r^2} + \frac{2}{r} \frac{\partial T}{\partial r} \right) - u \frac{\partial T}{\partial r}. \tag{H.9}$$

The divergence of $\mathbf{u}$ is given by

$$\nabla \cdot \mathbf{u} = \frac{1}{r^2} \frac{\partial \left(r^2 u \right)}{\partial r}. \tag{H.10}$$

As a consequence, the incompressible hypothesis ($\nabla \cdot \mathbf{u} = 0$) implies

$$u(r, \theta) = \frac{C(\theta)}{r^2}, \tag{H.11}$$

where $C(\theta)$ is a constant w.r.t. r. Then, for $r = R(\theta)$, one has

$$u\left(R(\theta), \theta\right) = \frac{C(\theta)}{R^2(\theta)}. \tag{H.12}$$

Moreover, since the vapor velocity is zero everywhere, the liquid velocity at the interface is equal to the velocity jump, i.e.

$$u\left(R(\theta), \theta\right) = \left[u\right]_\Gamma (\theta) \tag{H.13}$$

$$= \dot{m}(\theta) \left[\frac{1}{\rho} \right]_\Gamma. \tag{H.14}$$

Equations (H.12) and (H.14) lead to

$$C(\theta) = \dot{m}(\theta) \left[\frac{1}{\rho}\right]_\Gamma R^2(\theta) \tag{H.15}$$

$$= -\frac{\dot{m}(\theta)}{\rho_{\text{vap}}} \epsilon R^2(\theta). \tag{H.16}$$

The first term in the rhs of Eq. (H.16), $-\dot{m}(\theta)/\rho_{\text{vap}}$, is the interface velocity due to phase change. This term can also be seen as the temporal variation of the bubble radius, $\dot{R}(\theta)$, i.e.

$$\dot{R}(\theta) = -\frac{\dot{m}(\theta)}{\rho_{\text{vap}}}, \tag{H.17}$$

leading to

$$C(\theta) = \epsilon R^2(\theta) \dot{R}(\theta). \tag{H.18}$$

The norm of the velocity is then given by

$$u(r, \theta) = \frac{\epsilon R^2(\theta) \dot{R}(\theta)}{r^2}, \tag{H.19}$$

and Eq. (H.9) is rewritten

$$\frac{\partial T}{\partial \theta} = K \left(\frac{\partial T}{\partial r^2} + \frac{2}{r} \frac{\partial T}{\partial r} \right) - \frac{\epsilon R^2 \dot{R}}{r^2} \frac{\partial T}{\partial r}, \tag{H.20}$$

where the dependence of R to θ is omitted, which is the heat equation as expressed in [75].

H.1.3 Derivation of the temperature analytical expression

We now establish the solution of Eq. (H.20). The boundary conditions are the temperature value at the interface, $T(R, \theta)$, and the temperature value infinitely far from the interface, $T(\infty, \theta)$. We define

$$T(R, \theta) = T_{\text{sat}}, \tag{H.21}$$

and

$$T(\infty, \theta) = T_\infty, \tag{H.22}$$

where T_∞ is to be determined. The initial condition is given by

$$T(r, 0) = T_0, \tag{H.23}$$

where T_0 is a uniform value. As stated in [75], since heat generation is not considered in Eq. (H.20), one has

$$T_0 = T_\infty, \tag{H.24}$$

hence

$$T(r, 0) = T_\infty. \tag{H.25}$$

Let t be the reduced temperature defined by

$$t(r, \theta) = \frac{T(r, \theta) - T_\infty}{T_\infty}. \tag{H.26}$$

Equivalently, one has

$$T(r,\theta) = T_\infty t(r,\theta) + T_\infty. \tag{H.27}$$

The reduced boundary conditions are given by

$$\begin{aligned} t(R,\theta) &= \frac{T(R,\theta) - T_\infty}{T_\infty} \\ &= \frac{T_{\text{sat}} - T_\infty}{T_\infty} \\ &= -\tau \end{aligned} \tag{H.28}$$

and

$$\begin{aligned} t(\infty,\theta) &= \frac{T(\infty,\theta) - T_\infty}{T_\infty} \\ &= \frac{T_\infty - T_\infty}{T_\infty} \\ &= 0, \end{aligned} \tag{H.29}$$

and the reduced initial condition, by

$$\begin{aligned} t(r,0) &= \frac{T(r,0) - T_\infty}{T_\infty} \\ &= \frac{T_\infty - T_\infty}{T_\infty} \\ &= 0. \end{aligned} \tag{H.30}$$

Equation (H.20) applied to the rhs of Eq. (H.27) reads

$$T_\infty \frac{\partial t}{\partial \theta}(r,\theta) = K\left(T_\infty \frac{\partial^2 t}{\partial r^2}(r,\theta) + \frac{2}{r} T_\infty \frac{\partial t}{\partial r}(r,\theta)\right) - \frac{\epsilon R^2 \dot{R}}{r^2} T_\infty \frac{\partial t}{\partial r}(r,\theta). \tag{H.31}$$

Simplifying Eq. (H.31) by T_{sat} leads to

$$\dot{t}(r,\theta) = K\left(t_{rr}(r,\theta) + \frac{2}{r} t_r(r,\theta)\right) - \frac{\epsilon R^2 \dot{R}}{r^2} t_r(r,\theta), \tag{H.32}$$

where the temporal derivatives are noted with a dot and spatial derivatives are noted with an indice representing the direction of differentiation. Equation (H.32) is the heat equation written in spherical coordinates for the reduced temperature t.

Under the assumptions listed in Section G of [75] and using Eqs. (H.21) and (H.25), the energy balance written for the open system represented by the bubble reads

$$\rho_{\text{vap}} \dot{R} \left\{\overline{L} + c_{\text{p,vap}} \left[T_{\text{sat}} - T_\infty\right]\right\} = \rho_{\text{liq}} c_{\text{p,liq}} \left[T_{\text{sat}} - T_\infty\right] (1-\epsilon)\, \dot{R} + k \left(\frac{\partial T}{\partial r}\right)_{r=R}, \tag{H.33}$$

where $k = \rho_{\text{liq}} c_{\text{p,liq}} K$ is the liquid thermal conductivity. Isolating the thermal conduction flux at the interface in Eq. (H.33) leads to

$$
\begin{aligned}
k \left(\frac{\partial T}{\partial r} \right)_{r=R} &= \dot{R} \left(\left\{ \overline{L} + c_{\text{p,vap}} \left[T_{\text{sat}} - T_\infty \right] \right\} \rho_{\text{vap}} - \rho_{\text{liq}} c_{\text{p,liq}} \left[T_{\text{sat}} - T_\infty \right] (1 - \epsilon) \right) && \text{(H.34)} \\
&= \dot{R} \left(\overline{L} \rho_{\text{vap}} + \left(\rho_{\text{vap}} c_{\text{p,vap}} - \rho_{\text{liq}} c_{\text{p,liq}} (1 - \epsilon) \right) \left[T_{\text{sat}} - T_\infty \right] \right) && \text{(H.35)} \\
&= \dot{R} \left(\overline{L} \rho_{\text{vap}} + \left(\rho_{\text{vap}} c_{\text{p,vap}} - \rho_{\text{liq}} c_{\text{p,liq}} \omega \right) \left[T_{\text{sat}} - T_\infty \right] \right) && \text{(H.36)} \\
&= \dot{R} \left(\xi \rho_{\text{liq}} c_{\text{p,liq}} T_\infty + \left(\rho_{\text{vap}} c_{\text{p,vap}} - c_{\text{p,liq}} \rho_{\text{vap}} \right) \left[T_{\text{sat}} - T_\infty \right] \right) && \text{(H.37)} \\
&= \dot{R} \left(\xi \rho_{\text{liq}} c_{\text{p,liq}} T_\infty + \underbrace{\rho_{\text{vap}} \left(c_{\text{p,vap}} - c_{\text{p,liq}} \right)}_{= -\omega \nu \rho_{\text{liq}} c_{\text{p,liq}}} \underbrace{\left[T_{\text{sat}} - T_\infty \right]}_{= -T_\infty \tau} \right) && \text{(H.38)} \\
&= \dot{R} \left(\xi \rho_{\text{liq}} c_{\text{p,liq}} T_\infty + \omega \nu \tau \rho_{\text{liq}} c_{\text{p,liq}} T_\infty \right). && \text{(H.39)}
\end{aligned}
$$

Using Eq. (H.27), the analogous of Eq. (H.39) for the reduced temperature is given by

$$
\left(\frac{\partial t}{\partial r} \right)_{r=R} = k^{-1} \dot{R} \left(\xi \rho_{\text{liq}} c_{\text{p,liq}} + \omega \nu \tau \rho_{\text{liq}} c_{\text{p,liq}} \right), \tag{H.40}
$$

which, simplifying by $\rho_{\text{liq}} c_{\text{p,liq}}$ can also be written

$$
t_r(R, \theta) = K^{-1} \dot{R} \left(\xi + \omega \nu \tau \right). \tag{H.41}
$$

Our goal is the derivation of a self-similar solution of Eq. (H.20). Let s be the reduced spatial variable defined by

$$
s = \frac{r}{2\sqrt{K\theta}}, \tag{H.42}
$$

where $\sqrt{K\theta}$ is the thermal diffusion length of the liquid. Let β be the dimensionless positive real number defined as

$$
r = R(\theta) \Longrightarrow s = \beta, \tag{H.43}
$$

yielding the expression of the bubble radius at time θ,

$$
R(\theta) = 2\beta\sqrt{K\theta}, \tag{H.44}
$$

and its temporal derivative,

$$
\dot{R}(\theta) = \frac{\beta\sqrt{K}}{\sqrt{\theta}}. \tag{H.45}
$$

The spatial differentiation operator $\partial / \partial r$ can then be expressed by

$$
\begin{aligned}
\frac{\partial}{\partial r} &\equiv \frac{\partial s}{\partial r} \frac{\partial}{\partial s} \\
&\equiv \frac{1}{2\sqrt{K\theta}} \frac{\partial}{\partial s},
\end{aligned} \tag{H.46}
$$

and the temporal differentiation operator $\partial/\partial\theta$, by

$$\begin{aligned}\frac{\partial}{\partial\theta} &\equiv \frac{\partial s}{\partial\theta}\frac{\partial}{\partial s}\\ &\equiv \frac{\partial}{\partial\theta}\left(\frac{r}{2\sqrt{K\theta}}\right)\frac{\partial}{\partial s}\\ &\equiv -\frac{r}{4\sqrt{K}\theta^{3/2}}\frac{\partial}{\partial s}.\end{aligned} \tag{H.47}$$

Let $\bar{t}$ be the reduced temperature taking s as argument, i.e.

$$\bar{t}(s) = t(r,\theta). \tag{H.48}$$

Then, the operators in the lhs of Eqs. (H.46) and (H.47) apply to t, and the operators in the rhs of the same equations apply to $\bar{t}$. Equations (H.46) and (H.48) lead to

$$t_r(r,\theta) = \frac{1}{2\sqrt{K\theta}}\bar{t}(s), \tag{H.49}$$

and

$$t_{rr}(r,\theta) = \frac{1}{4K\theta}\bar{t}_{ss}(s). \tag{H.50}$$

Similarly, Eqs. (H.47) and (H.48) lead to

$$\dot{t}(r,\theta) = -\frac{r}{4\sqrt{K}\theta^{3/2}}\bar{t}_s(s). \tag{H.51}$$

Equation (H.32) is then rewritten

$$-\frac{r}{4\sqrt{K}\theta^{3/2}}\bar{t}_s(s) = K\left(\frac{1}{4K\theta}\bar{t}_{ss}(s) + 2r^{-1}\frac{1}{2\sqrt{K\theta}}\bar{t}_s(s)\right) - r^{-2}R^2\dot{R}\frac{1}{2\sqrt{K\theta}}\bar{t}_s(s)\epsilon, \tag{H.52}$$

which, isolating $\bar{t}_{ss}(s)$, is equivalent to

$$\bar{t}_{ss}(s) = 2\bar{t}_s(s)\left(-s - s^{-1} + s^{-2}\beta^3\epsilon\right). \tag{H.53}$$

Dividing Eq. (H.53) by $\bar{t}_s(s)$ and integrating both lhs and rhs yields

$$\bar{t}_s(s) = As^{-2}\exp\left(-s^2 - 2s^{-1}\beta^3\epsilon\right), \tag{H.54}$$

where A is the constant of integration to be determined. Integrating Eq. (H.54) yields

$$\bar{t}(s) = \int_0^s Ax^{-2}\exp\left(-x^2 - 2x^{-1}\beta^3\epsilon\right)\mathrm{d}x. \tag{H.55}$$

Using Eq. (H.29), one has $\bar{t}(\infty) = 0$. Consequently, Eq. (H.55) can be rewritten

$$\bar{t}(s) = \int_0^{+\infty} Ax^{-2}\exp\left(-x^2 - 2x^{-1}\beta^3\epsilon\right)\mathrm{d}x - \int_s^{+\infty} Ax^{-2}\exp\left(-x^2 - 2x^{-1}\beta^3\epsilon\right)\mathrm{d}x \tag{H.56}$$

$$= \bar{t}(\infty) - A\int_s^{+\infty} x^{-2}\exp\left(-x^2 - 2x^{-1}\beta^3\epsilon\right)\mathrm{d}x \tag{H.57}$$

$$= -A\int_s^{+\infty} x^{-2}\exp\left(-x^2 - 2x^{-1}\beta^3\epsilon\right)\mathrm{d}x. \tag{H.58}$$

In order to find the value of A, Eq. (H.46) is used to rewrite Eq. (H.41) as

$$\frac{1}{2\sqrt{K\theta}}\bar{t}_s(s=\beta) = K^{-1}\dot{R}\left(\xi+\omega\nu\tau\right), \tag{H.59}$$

which, substituting $\bar{t}_s(s)$ in the lhs by its expression in the rhs of Eq. (H.54), gives

$$\frac{1}{2\sqrt{K\theta}}As^{-2}\exp\left(-s^2-2s^{-1}\beta^3\epsilon\right) = K^{-1}\dot{R}\left(\xi+\omega\nu\tau\right). \tag{H.60}$$

Isolating A in Eq. (H.60) yields

$$A = 2\beta^3\left(\xi+\omega\nu\tau\right)\exp\left(\beta^2+2\epsilon\beta^2\right), \tag{H.61}$$

which, substituting in Eq. (H.58), leads to

$$\bar{t}(s) = -\left(\xi+\omega\nu\tau\right)2\beta^3\exp\left(\beta^2+2\epsilon\beta^2\right)\int_s^{+\infty} x^{-2}\exp\left(-x^2-2\epsilon\beta^3x^{-1}\right)\mathrm{d}x, \tag{H.62}$$

which is the formulation of the analytical solution of Eq. (H.20) as expressed in [75]. The last step is the determination of the value of β. We recall that $t(R,\theta)=\bar{t}(\beta)$. Using Eq. (H.28), one has

$$\bar{t}(\beta) = -\tau. \tag{H.63}$$

Moreover, for $s=\beta$, Eq. (H.62) yields the self-similar solution of Eq. (H.20) given by

$$\bar{t}(\beta) = -\left(\xi+\omega\nu\tau\right)2\beta^3\exp\left(\beta^2+2\epsilon\beta^2\right)\int_\beta^{+\infty} x^{-2}\exp\left(-x^2-2\epsilon\beta^3x^{-1}\right)\mathrm{d}x. \tag{H.64}$$

Equations (H.63) and (H.64) lead to the implicit equation for β given by

$$\frac{\tau}{\xi+\omega\nu\tau} = 2\beta^3\exp\left(\beta^2+2\epsilon\beta^2\right)\int_\beta^{+\infty} x^{-2}\exp\left(-x^2-2\epsilon\beta^3x^{-1}\right)\mathrm{d}x. \tag{H.65}$$

Equations (H.62) and (H.65) are written with the notations from [75]. In [85], these equations are expressed by means of the Jakob number Ja given by

$$Ja = \frac{\rho_{\mathrm{liq}}c_{\mathrm{p,liq}}\left(T_\infty-T_{\mathrm{sat}}\right)}{\rho_{\mathrm{vap}}L_v}. \tag{H.66}$$

One can obtain en equivalent formula for Ja using the notations of [75], i.e.

$$Ja = \frac{\tau}{\xi+\omega\nu\tau} \tag{H.67}$$

$$= \frac{\rho_{\mathrm{liq}}c_{\mathrm{p,liq}}\left(T_\infty-T_{\mathrm{sat}}\right)}{\rho_{\mathrm{vap}}\left(\overline{L}+\left(c_{\mathrm{p,liq}}-c_{\mathrm{p,vap}}\right)\left(T_\infty-T_{\mathrm{sat}}\right)\right)}. \tag{H.68}$$

Equations (H.66) and (H.68) lead to

$$L_v = \overline{L}+\left(c_{\mathrm{p,liq}}-c_{\mathrm{p,vap}}\right)\left(T_\infty-T_{\mathrm{sat}}\right). \tag{H.69}$$

Using the expression of the Jakob number given in Eq. (H.67), Eq. (H.62) can be divided by $-\tau$, yielding

$$-\frac{1}{\tau}\bar{t}(s) = \frac{2\beta^3}{Ja}\exp\left(\beta^2 + 2\epsilon\beta^2\right)\int_s^{+\infty} x^{-2}\exp\left(-x^2 - 2\epsilon\beta^3 x^{-1}\right)\mathrm{d}x. \tag{H.70}$$

Finally, let $\overline{T}$ be the temperature taking the reduced variable s as argument, i.e. $\overline{T}(s) = T(r,\theta)$, yielding

$$\bar{t}(s) = \frac{\overline{T}(s) - T_\infty}{T_\infty}. \tag{H.71}$$

Using Eqs. (H.4) and (H.71), Eq. (H.70) is rewritten

$$\frac{\overline{T}(s) - T_\infty}{T_{\mathrm{sat}} - T_\infty} = \frac{2\beta^3}{Ja}\exp\left(\beta^2 + 2\epsilon\beta^2\right)\int_s^{+\infty} x^{-2}\exp\left(-x^2 - 2\epsilon\beta^3 x^{-1}\right)\mathrm{d}x, \tag{H.72}$$

which is the formulation of the analytical solution of Eq. (H.20) as expressed in [85]. In [85], Eq. (H.67) is used to rewrite the implicit equation for β, Eq. (H.65), as

$$Ja = 2\beta^3\exp\left(\beta^2 + 2\epsilon\beta^2\right)\int_\beta^{+\infty} x^{-2}\exp\left(-x^2 - 2\epsilon\beta^3 x^{-1}\right)\mathrm{d}x. \tag{H.73}$$

Equations (H.72) and (H.73) are the formulations used in our simulations to compute the theoretical liquid temperature at nodes of radius r and at time θ.

H.1.4 Computation of the theoretical temperature profile

The liquid temperature field T_{liq} is initialized in the liquid phase with the analytical expression given in Eq. (H.72). In order to compute this analytical expression, one first needs to specify ρ_{liq}, $c_{\mathrm{p,liq}}$, ρ_{vap}, L_v and T_{sat}. Then, by specifying the Jakob number Ja, the temperature at infinity T_∞ is computed using Eq. (H.66) as

$$T_\infty = T_{\mathrm{sat}} + \frac{\rho_{\mathrm{vap}} L_v}{\rho_{\mathrm{liq}} c_{\mathrm{p,liq}}} Ja. \tag{H.74}$$

Using Eq. (H.1), one then computes the value of the growth rate β by solving the implicit equation (H.73) where β is the only unknown. In our simulations, we used the bisection method to solve Eq. (H.73). An initial interval $\left[\beta_0^0, \beta_1^0\right]$ of β values is chosen. The rhs of Eq. (H.73) is computed for both β_0^0 and β_1^0, leading to Ja_0^0 and Ja_1^0. If the order relation

$$Ja_0^0 \leq Ja \leq Ja_1^0 \tag{H.75}$$

holds, then, for $\beta_{1/2}^0 = \left(\beta_0^0 + \beta_1^0\right)/2$, if $Ja_{1/2}^0 < Ja$, then we set $\beta_0^1 = \beta_{1/2}^0$ and $\beta_1^1 = \beta_1^0$, otherwise, we set $\beta_0^1 = \beta_0^0$ and $\beta_1^1 = \beta_{1/2}^0$. The method is iterated N times until $\left|Ja_0^N - Ja\right| < \alpha$ or $\left|Ja_1^N - Ja\right| < \alpha$ where α is the accepted tolerance. The retained value for β is either β_0^N or β_1^N, depending on which one among Ja_0^N and Ja_1^N is the closest to Ja. Since the rhs of Eq. (H.73) involves a term of the form $\exp\left(\beta^2\right)$, the initial upper limit β_1^0 must be chosen small enough in order to avoid numerical errors due to the limited size of the numbers storable on a given machine. We set β_0^0 to 10^{-15}, β_1^0 to 15 and α to 10^{-8}. Once β is known, Eq. (H.72) can be solved. Both Eqs. (H.72) and (H.73) involve the improper integral of a fast decreasing function. This integral is computed using the midpoint rule within which the integration is performed until $x = 20$.

H.2 In two dimensions

In order to speed up computations, we derived an analogous expression of the solution given in Eq. (H.72) in two dimensions. To the best of our knowledge, such two-dimensional derivation does not exist in the literature.

Under the notations used in the previous section, the heat equation written for the reduced temperature t in polar coordinates reads

$$\dot{t}(r,\theta) = K\left(t_{rr}(r,\theta) + r^{-1}t_r(r,\theta)\right) - \frac{\epsilon R\dot{R}}{r}t_r(r,\theta), \tag{H.76}$$

with boundary conditions

$$t(R,\theta) = -\tau \tag{H.77}$$

and

$$t(\infty,\theta) = 0, \tag{H.78}$$

and initial condition

$$t(r,0) = 0. \tag{H.79}$$

Equations (H.46) and (H.47) are still valid in two dimensions. Thus, Eq. (H.32) is rewritten

$$-\frac{r}{4\sqrt{K}\theta^{3/2}}\bar{t}_s(s) = K\left(\frac{1}{4K\theta}\bar{t}_{ss}(s) + r^{-1}\frac{1}{2\sqrt{K\theta}}\bar{t}_s(s)\right) - r^{-1}R\dot{R}\frac{1}{2\sqrt{K\theta}}\bar{t}_s(s)\epsilon, \tag{H.80}$$

which, isolating $\bar{t}_{ss}(s)$, is equivalent to

$$\bar{t}_{ss}(s) = 2\bar{t}_s(s)\left(-s - \frac{1}{2}s^{-1} + s^{-1}\beta^2\epsilon\right). \tag{H.81}$$

Dividing Eq. (H.81) by $\bar{t}_s(s)$ and integrating both lhs and rhs yields

$$\bar{t}_s(s) = As^{-1}\exp\left(-s^2 + 2\beta^2\epsilon\ln s\right), \tag{H.82}$$

Integrating Eq. (H.82) gives

$$\bar{t}(s) = A\int_0^s x^{-1}\exp\left(-x^2 + 2\beta^2\epsilon\ln x\right)\mathrm{d}x. \tag{H.83}$$

Similarly to Eqs. (H.56)-(H.58), where we used $\bar{t}(\infty) = 0$, Eq. (H.83) can be rewritten

$$\begin{aligned}\bar{t}(s) &= -A\int_s^{+\infty} x^{-1}\exp\left(-x^2 + 2\beta^2\epsilon\ln x\right)\mathrm{d}x \\ &= -A\int_s^{+\infty} x^{2\beta^2\epsilon-1}\exp\left(-x^2\right)\mathrm{d}x.\end{aligned} \tag{H.84}$$

Let $p \in \mathbb{R}$ and $x \in \mathbb{R}_+$. The primitive of $x^p\exp\left(-x^2\right)$ is given by

$$\int x^p\exp\left(-x^2\right) = -\frac{1}{2}\Gamma\left(\frac{p+1}{2}, x^2\right) + C, \tag{H.85}$$

where C is a constant and Γ is the incomplete gamma function defined by

$$\Gamma(a,s) = \int_s^{+\infty} x^{a-1} \exp(-x)\,\mathrm{d}x. \tag{H.86}$$

In order to facilitate the calculations, we restrain the derivation to the class of solutions for which $p = 2\beta^2\epsilon - 1 = 1$ in Eq. (H.84), i.e.

$$\beta^2\epsilon = 1. \tag{H.87}$$

Equation (H.85) written for $p = 1$ yields

$$\int x \exp\left(-x^2\right) = -\frac{\exp\left(-x^2\right)}{2} + C. \tag{H.88}$$

Consequently, Eq. (H.84) is rewritten

$$\bar{t}(s) = -\frac{A}{2}\exp\left(-s^2\right). \tag{H.89}$$

The value of A is derived from Eq. (H.77), stating that $\bar{t}(\beta) = -\tau$. As a result, Eq. (H.89) is rewritten

$$-\tau = -\frac{A}{2}\exp\left(-s^2\right), \tag{H.90}$$

yielding

$$A = 2\tau\exp\left(\beta^2\right). \tag{H.91}$$

Finally, Eq. (H.89) reads

$$\bar{t}(s) = -\tau\exp\left(\beta^2 - s^2\right). \tag{H.92}$$

Using Eqs. (H.4) and (H.71), Eq. (H.92) is rewritten

$$\frac{\overline{T}(s) - T_\infty}{T_{\mathrm{sat}} - T_\infty} = \exp\left(\beta^2 - s^2\right), \tag{H.93}$$

where, as opposed to the three-dimensional case, no implicit equation similar to Eq. (H.73) has to be solved to find β, which is indeed directly given by Eq. (H.87). Moreover, in Eq. (H.93), T_∞ is given by Eq. (H.66) for a fixed Jakob number.

Bibliography

[1] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys.*, 118(2):269 – 277, 1995.

[2] D. Anderson, G. McFadden, and A. Wheeler. Diffuse-interface methods in fluid mechanics. *Annu. Rev. Fluid Mech.*, 30, 06 1997.

[3] L. Anumolu and M. F. Trujillo. Gradient augmented level set method for phase change simulations. *J. Comput. Phys.*, 353:377 – 406, 2018.

[4] T. D. Aslam. A partial differential equation approach to multidimensional extrapolation. *J. Comput. Phys.*, 193(1):349 – 355, 2004.

[5] R. Aubry, F. Mut, R. Löhner, and J. R. Cebral. Deflated preconditioned conjugate gradient solvers for the pressure–poisson equation. *J. Comput. Phys.*, 227(24):10196 – 10208, 2008.

[6] A. Begmohammadi, M. Farhadzadeh, and M. H. Rahimian. Simulation of pool boiling and periodic bubble release at high density ratio using lattice boltzmann method. *Int. Commun. Heat Mass Transf.*, 61:78 – 87, 2015.

[7] A. Begmohammadi, M. H. Rahimian, M. Farhadzadeh, and M. A. Hatani. Numerical simulation of single- and multi-mode film boiling using lattice boltzmann method. *Comput. Math. Appl.*, 71(9):1861 – 1874, 2016.

[8] P. Benard. *Analysis and improvement of a centimetric burner by large-eddy simulations*. Theses, INSA de Rouen, October 2015.

[9] M. Bernard, G. Lartigue, G. Balarac, V. Moureau, and G. Puigt. A framework to perform high-order deconvolution for finite-volume method on simplicial meshes. *J. Comput. Phys.*, To be submitted.

[10] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comput. Phys.*, 100(2):335 – 354, 1992.

[11] E. Carlini, M. Falcone, and P. Hoch. A generalized fast marching method on unstructured triangular meshes. *SIAM J. Numer. Anal*, 51(6):2999–3035, 2013.

[12] S. Chen and G. D. Doolen. Lattice boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30(1):329–364, 1998.

[13] R. Chiodi and O. Desjardins. A reformulation of the conservative level set reinitialization equation for accurate and robust simulation of complex multiphase flows. *J. Comput. Phys.*, 343:186–200, 2017.

[14] A. J. Chorin. Numerical solution of the navier-stokes equations. *Math. Comp.*, 22:745–762, 1968.

[15] R. Cole. A photographic study of pool boiling in the region of the critical heat flux. *AIChE Journal*, 6(4):533–538, 1960.

[16] M. G. Cooper. Heat flow rates in saturated nucleate pool boiling-a wide-ranging examination using reduced properties. volume 16 of *Advances in Heat Transfer*, pages 157 – 239. Elsevier, 1984.

[17] C. Dapogny and P. Frey. Computation of the signed distance function to a discrete contour on adapted triangulation. *Calcolo*, 49(3):193–219, 2012.

[18] P.-G. de Gennes, F. Brochart-Wyart, and D. Quéré. *Gouttes, bulles, perles et ondes*. 2002.

[19] O. Desjardins, V. Moureau, and H. Pitsch. An accurate conservative level set/ghost fluid method for simulating turbulent atomization. *J. Comput. Phys.*, 227(18):8395 – 8416, 2008.

[20] D. d'Humières. Simulation d'allées de von karman bidimensionelles à l'aide d'un gaz sur réseau. *C. R. Acad. Sc.*, 301:1391 – 1394, 1985.

[21] A. Fakhari and D. Bolster. Diffuse interface modeling of three-phase contact line dynamics on curved boundaries: A lattice boltzmann model for large density and viscosity ratios. *J. Comput. Phys.*, 334:620 – 638, 2017.

[22] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152(2):457 – 492, 1999.

[23] J.-P. Franc and J.-M. Michel. *Fundamentals of Cavitation*. Fluid Mechanics and Its Applications. Springer Netherlands, 2006.

[24] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equation. *Phys. Rev. Lett.*, 56:1505–1508, Apr 1986.

[25] W. Fritz. Maximum volume of vapour bubbles. *Physik Zeitschr*, 36:379 – 384, 1935.

[26] F. Gibou, L. Chen, D. Nguyen, and S. Banerjee. A level set based sharp interface method for the multiphase incompressible navier–stokes equations with phase change. *J. Comput. Phys.*, 222(2):536 – 555, 2007.

[27] F. Gibou, R. P. Fedkiw, L.-T. Cheng, and M. Kang. A second-order-accurate symmetric discretization of the poisson equation on irregular domains. *J. Comput. Phys.*, 176(1):205 – 227, 2002.

[28] R. Goldman. Curvature formulas for implicit curves and surfaces. *Comput. Aided Geom. D.*, 22(7):632 – 658, 2005. Geometric Modelling and Differential Geometry.

[29] A. Guion, S. Afkhami, S. Zaleski, and J. Buongiorno. Simulations of microlayer formation in nucleate boiling. *Int. J. Heat Mass Transf.*, 127:1271 – 1284, 2018.

[30] R. Haghani-Hassan-Abadi and M. H. Rahimian. Hybrid lattice boltzmann finite difference model for simulation of phase change in a ternary fluid. *Int. J. Heat Mass Transf.*, 127:704 – 716, 2018.

[31] P. Hai Trieu. *Effects of Nano- and Micro-surface Treatments on Boiling Heat Transfer*. Theses, Institut National Polytechnique de Grenoble - INPG, September 2010. Septembre 2010.

[32] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8(12):2182–2189, 1965.

[33] A. P. Hatton and I. S. Hall. Photographic study of boiling on prepared surfaces. *Third International Heat Transfer Conference*, 4:24–37, 1966.

[34] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.*, 49(6):409 – 436, 1952.

[35] C. W. Hirt and B. D. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *J. Comput. Phys.*, 39(1):201 – 225, 1981.

[36] G. Huber, S. Tanguy, M. Sagan, and C. Colin. Direct numerical simulation of nucleate pool boiling at large microscopic contact angle and moderate jakob number. *Int. J. Heat Mass Transf.*, 113:662 – 682, 2017.

[37] C. Huh and L. E. Scriven. Hydrodynamic model of steady movement of a solid/liquid/fluid contact line. *J. Colloid Interface Sci.*, 35(1):85 – 101, 1971.

[38] H. J. Ivey. Relationships between bubble frequency, departure diameter and rise velocity in nucleate boiling. *Int. J. Heat Mass Transf.*, 10(8):1023 – 1040, 1967.

[39] R. Janodet, G. Vaudor, G. Lartigue, P. Benard, V. Moureau, and R. Mercier. An unstructured conservative level-set algorithm coupled with dynamic mesh adaptation for the computation of liquid-gas flows. In *29th European Conference on Liquid Atomization and Spray Systems (ILASS Europe)*, Paris, France, September 2019.

[40] G. Jiang and D. Peng. Weighted eno schemes for hamilton–jacobi equations. *SIAM J. Sci. Comput.*, 21(6):2126–2143, 2000.

[41] D. Juric and G. Tryggvason. A front-tracking method for dendritic solidification. *J. Comput. Phys.*, 123(1):127 – 148, 1996.

[42] D. Juric and G. Tryggvason. Computations of boiling flows. *Int. J. Multiph. Flow*, 24(3):387 – 410, 1998.

[43] J. Kim and P. Moin. Application of a fractional-step method to incompressible navier-stokes equations. *J. Comput. Phys.*, 59(2):308 – 323, 1985.

[44] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci.*, 95(15):8431–8435, 1998.

[45] M. Kraushaar. *Application of the compressible and low-mach number approaches to large-eddy simulation of turbulent flows in aero-engines*. PhD thesis, 2011. Thèse de doctorat dirigée par Gicquel, Laurent et Moureau, Vincent Dynamique des fluides Toulouse, INPT 2011.

[46] V. Lakshminarayanan, A. K. Ghatak, and K. Thyagarajan. *Lagrangian Optics*. 2002.

[47] J. J. Lee and G. Son. A sharp-interface level-set method for compressible bubble growth with phase change. *Int. Commun. Heat Mass Transf.*, 86:1 – 11, 2017.

[48] M. S. Lee, A. Riaz, and V. Aute. Direct numerical simulation of incompressible multiphase flow with phase change. *J. Comput. Phys.*, 344:381 – 418, 2017.

[49] M. Malandain. *Simulation massivement parallèle des écoulements turbulents à faible nombre de Mach*. PhD thesis, Institut National des Sciences Appliquées de Rouen, 2013.

[50] M. Malandain, N. Maheu, and V. Moureau. Optimization of the deflated conjugate gradient algorithm for the solving of elliptic equations on massively parallel machines. *J. Comput. Phys.*, 238:32 – 47, 2013.

[51] E. Marchandise, P. Geuzaine, N. Chevaugeon, and J.-F. Remacle. A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics. *J. Comput. Phys.*, 225(1):949–974, 2007.

[52] V. Moureau, P. Domingo, and L. Vervisch. Design of a massively parallel cfd code for complex geometries. *CR Mécanique*, 339(2):141 – 148, 2011. High Performance Computing.

[53] V. Moureau, P. Domingo, and L. Vervisch. From large-eddy simulation to direct numerical simulation of a lean premixed swirl flame: Filtered laminar flame-pdf modeling. *Combust. Flame*, 158(7):1340 – 1357, 2011.

[54] D. Q. Nguyen, R. P. Fedkiw, and M. Kang. A boundary condition capturing method for incompressible flame discontinuities. *J. Comput. Phys.*, 172(1):71 – 98, 2001.

[55] R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM J. Numer. Anal.*, 24(2):355 – 365, 1987.

[56] S. Nukiyama. The maximum and minimum values of the heat q transmitted from metal to boiling water under atmospheric pressure. *Int. J. Heat Mass Transf.*, 9(12):1419 – 1433, 1966.

[57] N. Odier. *Numerical simulation of liquid jets sheared by a high-speed stream : flapping dynamics and interaction with vortical structures*. Theses, Université de Grenoble, December 2014.

[58] E. Olsson and G. Kreiss. A conservative level set method for two phase flow. *J. Comput. Phys.*, 210(1):225 – 246, 2005.

[59] E. Olsson, G. Kreiss, and S. Zahedi. A conservative level set method for two phase flow ii. *J. Comput. Phys.*, 225(1):785 – 807, 2007.

[60] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12 – 49, 1988.

[61] J. Palmore and O. Desjardins. A volume of fluid framework for interface-resolved simulations of vaporizing liquid-gas flows. *J. Comput. Phys.*, 399:108954, 2019.

[62] J. Pan, Q. Wang, Y. Zhang, and Y. Ren. High-order compact finite volume methods on unstructured grids with adaptive mesh refinement for solving inviscid and viscous flows. *Chinese Journal of Aeronautics*, 31(9):1829 – 1841, 2018.

[63] I. Perez-Raya and S. G. Kandlikar. Discretization and implementation of a sharp interface model for interfacial heat and mass transfer during bubble growth. *Int. J. Heat Mass Transf.*, 116:30 – 49, 2018.

[64] I. Perez-Raya and S. G. Kandlikar. Numerical models to simulate heat and mass transfer at sharp interfaces in nucleate boiling. *Numerical Heat Transfer, Part A: Applications*, 74(10):1583–1610, 2018.

[65] C. S. Peskin. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.*, 10(2):252 – 271, 1972.

[66] T. J. Poinsot and S. K. Lelef. Boundary conditions for direct simulations of compressible viscous flows. *J. Comput. Phys.*, 101(1):104 – 129, 1992.

[67] A. H. Rajkotwala, A. Panda, E. Peters, M. W. Baltussen, C. Van der Geld, J. Kuerten, and J. Kuipers. A critical comparison of smooth and sharp interface methods for phase transition. *Int. J. Multiph. Flow*, 120:103093, 2019.

[68] H. T. Rathod and H. S. G. Rao. Integration of polynomials over an arbitrary tetrahedron in euclidean three-dimensional space. *Comput. Struct.*, 59(1):55 – 65, 1996.

[69] D. H. Rothman and S. Zaleski. *Lattice-Gas Cellular Automata: Simple Models of Complex Hydrodynamics*. Collection Alea-Saclay: Monographs and Texts in Statistical Physics. Cambridge University Press, 1997.

[70] M. S. Sagan. *Simulation numérique directe et étude expérimentale de l'ébullition nucléée en microgravité : application aux réservoirs des moteurs d'Ariane V*. PhD thesis, 2013. Thèse de doctorat dirigée par Colin, Catherine et Tanguy, Sébastien Dynamique des fluides Toulouse, INPT 2013.

[71] P. Sagaut and C. Meneveau. *Large Eddy Simulation for Incompressible Flows: An Introduction*. Scientific Computation. Springer, 2006.

[72] G. Sahut, G. Ghigliotti, P. Bégou, P. Marty, and G. Balarac. Numerical simulation of boiling on 3D unstructured grids. *Proceedings of the 4th World Congress on Momentum, Heat and Mass Transfer (MHMT'19)*, 2019.

[73] G. Sahut, G. Ghigliotti, P. Marty, and G. Balarac. Evaluation of Level Set reinitialization algorithms for phase change simulation on unstructured grids. Submitted, October 2019.

[74] P. Sarkar. *Simulation of cavitation erosion by a coupled CFD-FEM approach*. Theses, Université Grenoble Alpes, March 2019.

[75] L. E. Scriven. On the dynamics of phase growth. *Chem. Eng. Sci.*, 10(1):1 – 13, 1959.

[76] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci.*, 93(4):1591–1595, 1996.

[77] C. Shao, K. Luo, M. Chai, H. Wang, and J. Fan. A computational framework for interface-resolved dns of simultaneous atomization, evaporation and combustion. *J. Comput. Phys.*, 371:751 – 778, 2018.

[78] C.-W. Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Technical report, ICASE Report No. 97-65, Institute for Computer Applications in Science Engineering, NASA/CR-97-206253, 1997.

[79] N. K. Singh and B. Premachandran. A coupled level set and volume of fluid method on unstructured grids for the direct numerical simulations of two-phase flows including phase change. *Int. J. Heat Mass Transf.*, 122:182 – 203, 2018.

[80] G. Son and V. K. Dhir. Numerical Simulation of Film Boiling Near Critical Pressures With a Level Set Method. *J. Heat Transfer*, 120(1):183–192, 02 1998.

[81] M. R. Spiegel and S. Lipschutz. *Schaum's Outline of Vector Analysis, 2ed.* Schaum's Outline Series. McGraw-Hill Education, 2009.

[82] S. Succi. *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond.* Numerical Mathematics and Scientific Computation. Clarendon Press, 2001.

[83] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114(1):146 – 159, 1994.

[84] S. Tanguy, T. Ménard, and A. Berlemont. A level set method for vaporizing two-phase flows. *J. Comput. Phys.*, 221(2):837 – 853, 2007.

[85] S. Tanguy, M. Sagan, B. Lalanne, F. Couderc, and C. Colin. Benchmarks and numerical methods for the simulation of boiling flows. *J. Comput. Phys.*, 264:1–22, 2014.

[86] G. Tomar, G. Biswas, A. Sharma, and A. Agrawal. Numerical simulation of bubble growth in film boiling using a coupled level-set and volume-of-fluid method. *Phys. Fluids*, 17, 11 2005.

[87] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, 169(2):708 – 759, 2001.

[88] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct Numerical Simulations of Gas–Liquid Multiphase Flows.* Cambridge University Press, 2011.

[89] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *J. Comput. Phys.*, 100(1):25 – 37, 1992.

[90] A. Urbano, S. Tanguy, and C. Colin. Direct numerical simulation of nucleate boiling in zero gravity conditions. *Int. J. Heat Mass Transf.*, 143:118521, 2019.

[91] A. Urbano, S. Tanguy, G. Huber, and C. Colin. Direct numerical simulation of nucleate boiling in micro-layer regime. *Int. J. Heat Mass Transf.*, 123:1128 – 1137, 2018.

[92] H. A. Van Der Vorst. Parallel iterative solution methods for linear systems arising from discretized pde's. *Lecture Notes on Parallel Iterative Methods for discretized PDE's. AGARD Special Course on Parallel Computing in CFD*, 1995.

[93] C. H. Wang and V. K. Dhir. Effect of Surface Wettability on Active Nucleation Site Density During Pool Boiling of Water on a Vertical Surface. *J. Heat Transfer*, 115(3):659–669, 08 1993.

[94] S. W. J. Welch. Local simulation of two-phase flows including interface tracking with mass transfer. *J. Comput. Phys.*, 121(1):142 – 154, 1995.

[95] S. W. J. Welch and J. Wilson. A volume of fluid based method for fluid flows with phase change. *J. Comput. Phys.*, 160(2):662 – 682, 2000.

[96] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Lecture Notes in Mathematics, 1725. Springer, 2000.

[97] Y. F. Yap, H. Y. Li, J. Lou, L. S. Pan, and Z. Shang. Numerical modeling of three-phase flow with phase change using the level-set method. *Int. J. Heat Mass Transf.*, 115:730 – 740, 2017.

[98] D. Youngs. *Time-Dependent Multi-material Flow with Large Fluid Distortion*, volume 24, pages 273–285. 01 1982.

[99] J. Zhang and M.-J. Ni. Direct numerical simulations of incompressible multiphase magneto-hydrodynamics with phase change. *J. Comput. Phys.*, 375:717 – 746, 2018.

[100] N. Zuber. Nucleate boiling. the region of isolated bubbles and the similarity with natural convection. *Int. J. Heat Mass Transf.*, 6(1):53 – 78, 1963.

Résumé

Cette thèse a pour objectif la simulation numérique du phénomène d'ébullition sur maillages non structurés. L'ébullition est le changement de phase des particules fluides de la phase liquide vers la phase vapeur sous l'action des flux thermiques à l'interface séparant les deux phases. Il s'agit donc d'un phénomène rencontré au sein d'écoulements diphasiques et piloté par le taux de transfert de masse à l'interface. Ce taux de transfert de masse est calculé à partir des flux thermiques de part et d'autre de l'interface. Cela implique donc la nécessité d'adopter une méthode de suivi d'interface très précise pour localiser l'interface à tout instant de la simulation. Les équations de Navier-Stokes sont alors couplées à l'équation de la chaleur par l'intermédiaire du taux de transfert de masse à l'interface. De telles simulations ont été menées par Tanguy et al. (J. Comput. Phys., 2014) sur des maillages cartésiens axisymétriques en deux dimensions. Dans cette thèse, nous étendons cette méthodologie à des maillages non structurés en trois dimensions (maillages composés de tétraèdres non réguliers utiles pour décrire des géométries complexes). Pour ce faire, nous avons développé un solveur spécifique dans le code YALES2 (code diphasique basé sur la méthode des volumes finis pour des maillages 3D non structurés). Le suivi de l'interface est assuré par la méthode Level Set. Le changement de phase engendre des discontinuités de vitesse et de pression à l'interface qui dépendent notamment du taux de transfert de masse. Ces discontinuités sont prises en compte par la méthode Ghost Fluid à l'aide de deux champs de vitesse et deux champs de température. Cette méthodologie étant déjà bien établie pour des maillages structurés cartésiens, l'apport de cette thèse réside dans la possibilité de simuler le changement de phase par ébullition sur des maillages non structurés en trois dimensions. Les spécificités des maillages non structurés ont nécessité de nombreux développements pour la réinitialisation de la fonction Level Set après advection, ainsi que l'utilisation d'opérateurs d'ordres élevés pour le calcul du taux de transfert de masse à l'interface. L'ensemble des développements proposés est finalement validé sur maillages non structurés à l'aide du cas-test analytique d'une bulle 3D en expansion dans un liquide surchauffé au repos.

Mots-clés : écoulements diphasiques, ébullition, calcul haute performance, maillages non structurés

Abstract

The objective of this thesis is the numerical simulation of the boiling phenomenon on unstructured grids. Boiling is the phase change of fluid particles from the liquid phase to the vapor phase under the action of thermal fluxes at the interface separating the two phases. Boiling is thus encountered in two-phase flows and driven by the mass transfer rate at the interface. This mass transfer rate is computed from the thermal fluxes on both sides of the interface. Consequently, a highly accurate numerical method is needed to locate the interface throughout the simulation. The Navier-Stokes equations are then coupled to the heat equation by means of the mass transfer rate at the interface. Such simulations have been performed by Tanguy et al. (J. Comput. Phys., 2014) on two-dimensional axisymmetric cartesian grids. In this thesis, we extend this methodology to three-dimensional unstructured grids (composed of irregular tetrahedra, useful to describe complex geometries). We then developed a specific solver in the YALES2 code (finite-volume-based code for simulations of two-phase flows on 3D unstructured grids). The interface motion is captured by the Level Set method. Phase change implies velocity and pressure discontinuities at the interface which especially depend on the mass transfer rate. These discontinuities are taken into account by the Ghost Fluid Method, with two velocity fields and two temperature fields. This methodology being already well established for structured cartesian grids, the contribution of this thesis relies on the ability to simulate phase change by boiling on three-dimensional unstructured grids. The particularities of unstructured grids have demanded numerous developments for the reinitialization of the Level Set function after advection, as well as the use of high-order operators for the computation of the mass transfer rate at the interface. The proposed developments are finally validated on unstructured grids against the analytical test-case of a 3D bubble expanding inside a superheated quiescent liquid.

Keywords : two-phase flows, boiling, high-performance computing, unstructured grids