AN INTRODUCTION TO THE LEVEL SET METHOD

C. DAPOGNY[1]

[1] *CNRS & Laboratoire Jean Kuntzmann,*

Contents

The Level Set Method was introduced in the seminal paper of Osher and Sethian [45] in 1988; however, related ideas already existed in the literature; see e.g. [24]. It is now acknowledged as one method of choice when dealing with arbitrarily large motions of domains or surfaces (even when they involve topological changes), from both the theoretical and numerical viewpoints.

These notes are a modest and inevitably biased introduction to the basic features of this method, without any ambition of exhaustivity. they are mainly focused on practical and numerical aspects: in order to keep the exposition elementary, insofar as possible, we have decided on several occasions not to insist on the (deep) underlying mathematical theory.

These notes are organized as follows. Section 1 outlines the scope of the study and discusses the types of evolving domains or surfaces we shall be dealing with. In Section 2, the level set framework is introduced, namely the key idea of describing domains in an implicit way, via an auxiliary 'level set' function. At first, Section 2.2 and Section 2.4 deal in a formal way with the connection between the 'intuitive' notion of an evolving domain and the corresponding evolution equations for an associated level set function; Section 2.3 then attempts to provide a glimpse of the underlying mathematical theory; it may be omitted at first reading, or by the uninterested reader.

In Section 3, we describe a typical use of the Level Set Method in the context of the resolution of a concrete physical problem, without yielding to technicality. It is emphasized that, in numerical practice, the Level Set Method relies on two main operations. Section 4 discusses numerical algorithms devoted to the first of these operations, namely the resolution of the level set evolution equation. Then, Section 5 presents numerical methods for the second operation of interest, - that of creating one level set function associated to a given domain or surface -, among which the celebrated *Fast Marching Method*.

Several operations which are easy to perform when a conventional representation of domains is used may result slightly more difficult in the level set framework; in Section 6, we discuss numerical methods for the main such operations. Section 7 is then a pot-pourri of additional practical issues: the need for *redistancing* is discussed, as well as the Narrow Band improvement, etc.

Numerical methods for tracking the evolution of domains are manifold. Section 8 briefly presents other popular methods, together with their main differences with the Level Set Method.

We conclude this chapter with Section 9, in which several applications are referenced where it has been successfully used, e.g. in numerical simulation, image processing, etc.

Before entering the core of the matter, let us mention that very good in-depth monographs exist on the topic: the obvious and perhaps most complete reference is the book of Sethian [55]; see also that of Osher and Fedkiw [44]. These may be fruitfully complemented with the more theoretic treaty [29].

1. Generalities about domain, or surface evolution problems

The Level Set Method is a general framework for the theoretical and numerical study of an evolving domain $\Omega(t) \subset \mathbb{R}^d$ (or equivalently of its boundary $\Gamma(t) := \partial\Omega(t)$) according to a velocity field $(t, x) \mapsto V(t, x) \in \mathbb{R}^d$. The velocity field V may account for motions of different natures, which can roughly be classified into three categories:

(1) V may be externally prescribed, that is, independently from any feature of the domain $\Omega(t)$. For instance, one may imagine that $\Omega(t)$ is a small object, passively transported in a fluid with velocity V. Of course, this coarse model overlooks any physical interaction between the object and the fluid.

(2) V may depend on *local features* of $\Omega(t)$ or $\Gamma(t)$, such as the normal vector field $n_t(x)$ to $\Gamma(t)$ (pointing outward $\Omega(t)$), or the mean curvature $\kappa_t(x)$ of $\Gamma(t)$[1]. Let us present two classical examples of such motions, which will be frequently referred to throughout this chapter:

- In the model describing the propagation of a flame front introduced in [9, 53], $\Omega(t)$ stands for a burnt region, whose frontier $\Gamma(t)$ expands with constant, normal velocity $c > 0$. In other words, V reads:

$$V(t, x) = c\, n_t(x). \tag{1.1}$$

- The *mean curvature flow* features the velocity field

$$V(t, x) = -\kappa_t(x) n_t(x). \tag{1.2}$$

[1]The second fundamental form II_x (resp. the mean curvature $\kappa(x)$) of the boundary Γ of Ω is oriented in the sense that it is positive definite (resp. positive) when Ω is locally convex around x.

Intuitively, $\Gamma(t)$ evolves by resorption of its 'bumpy' regions (i.e. the regions where its mean curvature is positive), and 'corking' of its 'creases' (the regions where its mean curvature is negative). More rigorously, it can be proved that the mean curvature flow is the gradient flow associated to the minimization of the perimeter functional.

(3) V may depend on *global features* of $\Omega(t)$ and $\Gamma(t)$. As an example, when $\Gamma(t)$ represents the interface between two fluids with different physical properties, V arises as $V(t,\cdot) = u_{\Omega(t)}$, the solution to the associated bifluid Navier-Stokes equations (see Section 3).

Motions pertaining to this last class are undoubtedly the most crucial ones from a physical or mechanical perspective; unfortunately, they generally prove far too complicated to lend themselves to a rigorous analysis. Therefore, most of this chapter deals with motions of the first two categories. As we shall see in Section 3 below, this is not a great loss of generality, at least when it comes to numerical applications, since, often, a motion of this third kind is somehow reduced to a series of motions of the first two kinds.

2. The Level Set Method for describing domain or interface evolution

The Level Set Method relies on the representation of a domain $\Omega \subset \mathbb{R}^d$ in *implicit form*, that is, via a scalar 'level set' function $\phi : \mathbb{R}^d \to \mathbb{R}$ satisfying the properties (see Fig. 1 for an illustration):

$$\left\{ \begin{array}{ll} \phi(x) < 0 & \text{if } x \in \Omega, \\ \phi(x) = 0 & \text{if } x \in \Gamma := \partial\Omega, \\ \phi(x) > 0 & \text{if } x \in {}^c\overline{\Omega}. \end{array} \right. \tag{2.1}$$

In other words, Ω is the negative subdomain of ϕ, and Γ coincides with the 0 isovalue of ϕ. Note that there exists an infinity of level set functions associated to a common domain Ω.

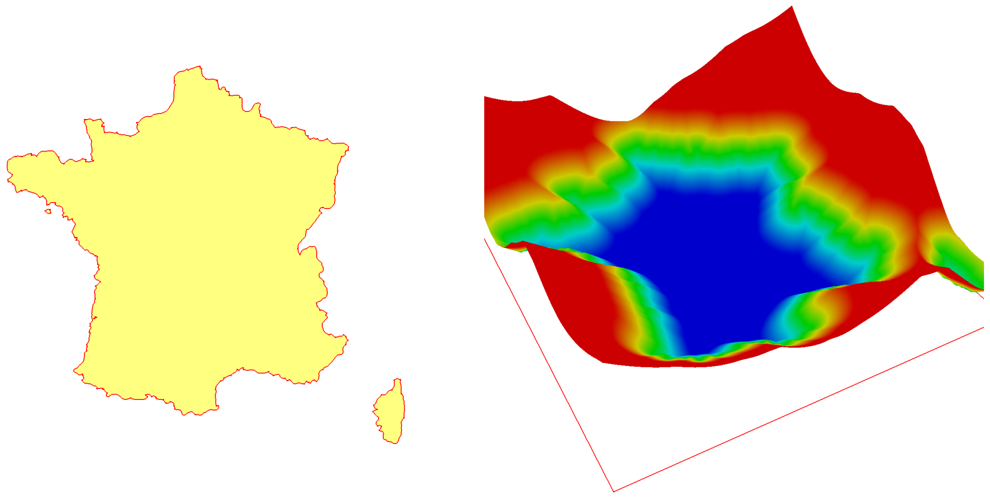

Figure 1. *(Left) A domain $\Omega \subset \mathbb{R}^2$, (right) the graph of an associated level set function ϕ.*

As we are about to see, this change in perspectives allows for a very convenient formulation of the evolution of a domain in terms of a partial differential equation. Before getting into details, let us first discuss about how geometric quantities attached to Ω may be directly calculated from the datum of a level set function ϕ.

2.1. **Level Set framework and geometry**

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain of class $\mathcal{C}^1$, $\phi : \mathbb{R}^d \to \mathbb{R}$ be an associated level set function (i.e. (2.1) holds), such that $\nabla\phi \neq 0$ on a neighborhood of Γ. Note that such a function always exists, as a consequence of a classical argument of partitions of unity, relying on the definition of a domain of class $\mathcal{C}^1$.

The unit normal vector $n(x)$ to Γ, pointing outward Ω can be expressed in terms of ϕ as:

$$\forall x \in \Gamma,\ n(x) = \frac{\nabla\phi(x)}{|\nabla\phi(x)|}. \tag{2.2}$$

Assuming Ω and ϕ to be additionnally of class $\mathcal{C}^2$, the second fundamental form II associated to Γ has the expression:

$$\forall x \in \Gamma, \;\; \text{II}_x = \nabla \left(\frac{\nabla \phi(x)}{|\nabla \phi(x)|} \right). \tag{2.3}$$

In particular, the *mean curvature* κ of Γ, that is, the trace of II reads:

$$\forall x \in \Gamma, \;\; \kappa(x) = \text{div} \left(\frac{\nabla \phi(x)}{|\nabla \phi(x)|} \right). \tag{2.4}$$

2.2. **Surface evolution in the level set framework**

We have hitherto been purposely evasive about what we precisely mean by ‘domain evolution’. In order to appraise the difficulties inherent to this notion, let us start by a formal discussion around a model situation: $\Omega(t) \subset \mathbb{R}^d$ is a smooth bounded domain, evolving over a time period $(0, T)$, according to a smooth velocity field $V(t, x)$, defined for $t \in (0, T)$ and $x \in \mathbb{R}^d$.

By this, it is natural to understand that, for any two times $t_0, t \in (0, T)$, $\Omega(t)$ may be obtained from $\Omega(t_0)$ by transporting its points in the direction pointed by V (see Fig. 2):

$$\Omega(t) = \left\{ \chi(x_0, t, t_0), \; x_0 \in \Omega(t_0) \right\}, \tag{2.5}$$

where $t \mapsto \chi(x_0, t, t_0)$ is the *characteristic* (or *integral*) curve of V emerging from x_0 at t_0, the unique solution of the ordinary differential equation:

$$\begin{cases} \chi'(x_0, t, t_0) = V(t, \chi(x_0, t, t_0)) \text{ for } t > 0, \\ \chi(x_0, t_0, t_0) = x_0. \end{cases} \tag{2.6}$$

Let us point out that this notions is often encountered in the context of fluid mechanics, where the characteristic curve $t \mapsto \chi(x_0, t, t_0)$ emerging from x_0 at time t_0 describes the position of a particle flowing along the velocity field $V(t, x)$, which lies at x_0 at time t_0.

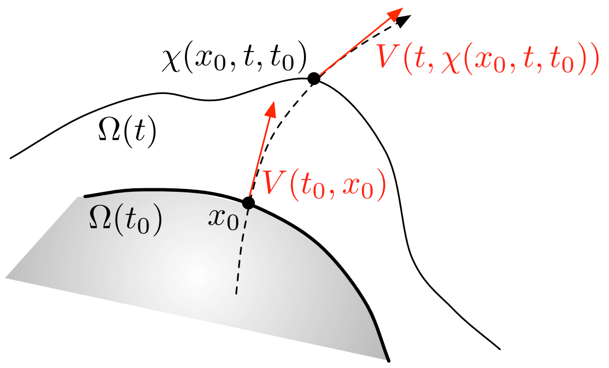

FIGURE 2. *One domain $\Omega(t)$ evolving according to the velocity field $V(t, x)$.*

Now, for each time t, let $\phi(t, \cdot)$ be a smooth level set function for $\Omega(t)$, and let us see how the motion of $\Omega(t)$ translates in terms of ϕ. Let $t_0 \in (0, T)$ and $x_0 \in \Gamma(t_0)$ be given, and let $t \mapsto \chi(x_0, t, t_0)$ be the associated trajectory, defined by (2.6). It stems from the definition (2.5) and (2.1) that:

$$\forall t \in (0, T), \;\; \phi(t, \chi(x_0, t, t_0)) = 0;$$

a direct application of the chain-rule then yields:

$$\frac{d}{dt}(\phi(t, \chi(x_0, t, t_0))) = \frac{\partial \phi}{\partial t}(t, \chi(x_0, t, t_0)) + \chi'(x_0, t, t_0) \cdot \nabla \phi(t, \chi(x_0, t, t_0)) = 0.$$

Evaluating this identity at $t = t_0$, and since the argument holds for any $t_0 \in (0,T)$, $x_0 \in \Gamma(t_0)$, we finally obtain the so-called *level set advection equation*:

$$\forall t \in (0,T),\ x \in \mathbb{R}^d,\ \frac{\partial \phi}{\partial t}(t,x) + V(t,x) \cdot \nabla \phi(t,x) = 0. \tag{2.7}$$

Note that (2.7) is not a usual advection equation, since V generally depends on Ω - thus on ϕ -, except in very specific cases (see the discussion in Section 1).

At this point, several comments are in order:

(1) Strictly speaking, we have only established (2.7) for pairs $(t,x) \in (0,T) \times \mathbb{R}^d$ such that $x \in \Gamma(t)$. However, imposing that *every* level set of $\phi(t,\cdot)$ (and not only the 0 level set) evolves according to $V(t,\cdot)$, the previous argument leads precisely to (2.7).

(2) The normal vector $n_t(x)$ to $\Gamma(t)$ at x, pointing outward $\Omega(t)$, reads $n_t(x) = \frac{\nabla\phi(t,x)}{|\nabla\phi(t,x)|}$. Let $v = V \cdot \frac{\nabla\phi}{|\nabla\phi|}$ be the normal component of V; (2.7) then rewrites as the so-called *level set Hamilton-Jacobi equation*:

$$\forall t \in (0,T),\ x \in \mathbb{R}^d,\ \frac{\partial \phi}{\partial t}(t,x) + v(t,x)|\nabla \phi(t,x)| = 0. \tag{2.8}$$

Again, this terminology is a bit misleading since, in general, (2.8) is not a usual Hamilton-Jacobi equation.

(3) Equation (2.8) expresses the intuitive fact that only the normal component of V plays a role in the evolution of $\Omega(t)$: its tangential part only accounts for a 'convection', or reparameterization of $\Omega(t)$.

(4) The Level Set Method requires the velocity field $V(t,\cdot)$ to be defined on the whole ambient space $\mathbb{R}^d$, whereas, in many applications, it only makes sense on $\Gamma(t)$ (see the examples given in the second and third items in Section 1). When such is the case, $V(t,\cdot)$ must be extended to $\mathbb{R}^d$ (or at least to a neighborhood of $\Gamma(t)$). It is however not difficult to show (at least formally) that the evolution of the 0 level set of $\phi(t,\cdot)$ (and only the 0 level set in this case) does not depend on how this velocity is extended. We will see in Section 7.2 that the same issue arises in the numerical context.

Hence, the Level Set method offers a convenient reformulation of the motion of $\Omega(t)$: this difficult geometric evolution problem translates into the Partial Differential Equations (2.7) and (2.8), posed in terms of the auxiliary function $\phi(t,\cdot)$, which is often simpler to handle - especially from the numerical point of view. This change in points ov view allows to account for rather arbitrary deformations of $\Omega(t)$, including changes in its topology (for instance, the merger of holes); these are indeed naturally addressed in this framework, while they would be difficult to describe with a conventional representation of domains. Consider the example depicted in Fig. 3: a domain $\Omega(t)$ initially composed of two disks evolves by 'inflating' until its two components merge. The instant where $\Omega(t)$ changes topology is very awkward to describe using a classical representation of domains, and it is fairly natural with a level set description. We shall say more about this point in the next Section 2.3.

2.3. A glimpse at the mathematical framework

However appealing, the above considerations are formal. To be more precise, they are legitimate as long as the evolving domain $\Omega(t)$ and the velocity field $V(t,x)$ are smooth. Unfortunately, even in the context of a simple motion, intitiated with an arbitrarily smooth domain $\Omega(0)$, $\Omega(t)$ turns out to develop singularities. Let us illustrate this crucial feature with two examples.

(1) This first example refers to the flame propagation model, presented in Section 1, and is excerpted from [55] §2.3. The situation is that of Fig. 4: a smooth two-dimensional domain $\Omega(0)$, whose boundary $\Gamma(0)$ is locally described by the curve

$$[0,1] \ni s \mapsto \gamma(s) = \left(1 - s, \frac{1+\cos(2\pi s)}{2}\right) \in \mathbb{R}^2,$$

accounts for the initially burnt region. The flame front $\Gamma(t)$ expands with unit normal velocity ($v(t,x) = n_t(x)$ in the notation of the previous section), and several positions of the front are depicted. At some time $t = t_c$, $\Gamma(t)$ becomes singular: it is no longer smooth at the blue dot on Fig. 4.

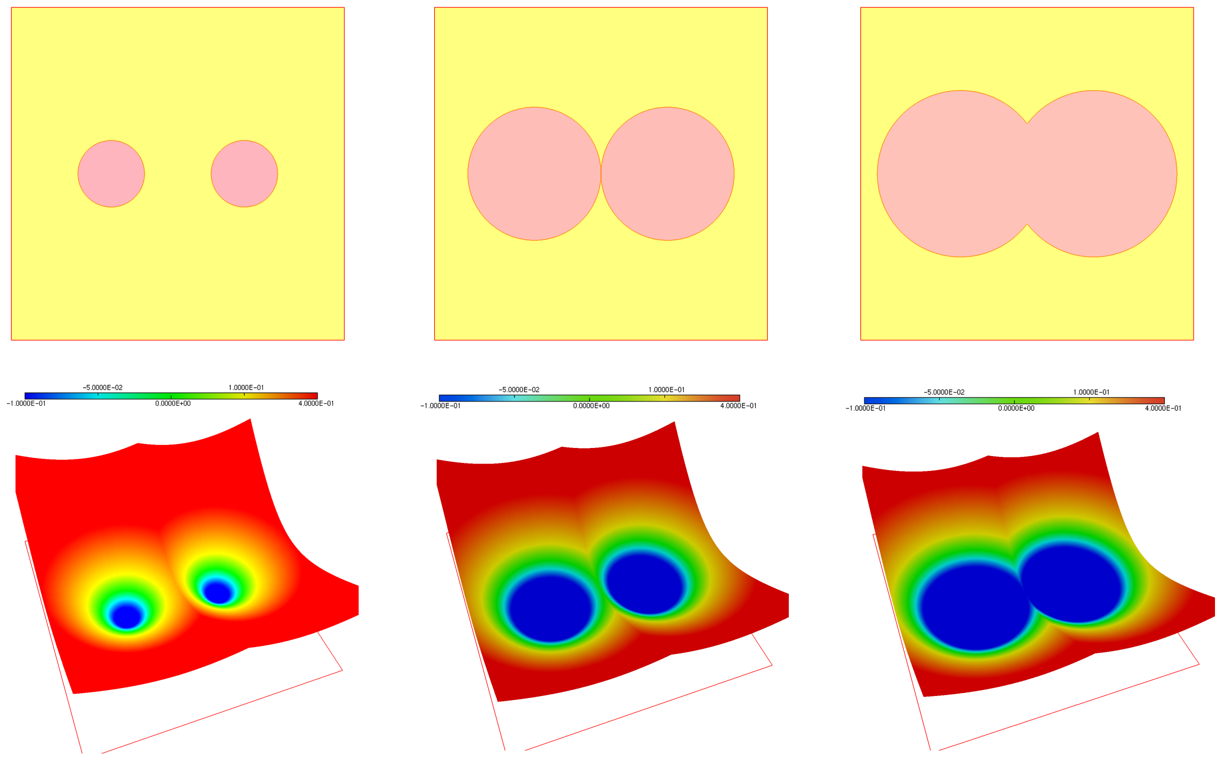

FIGURE 3. *Inflation of a domain* $\Omega(t)$ *initially composed of two disconnected disks; the upper row shows the motion of the boundary* $\Gamma(t)$*, and the lower one, the corresponding evolution of the graph of a corresponding level set function* $\phi(t, \cdot)$.

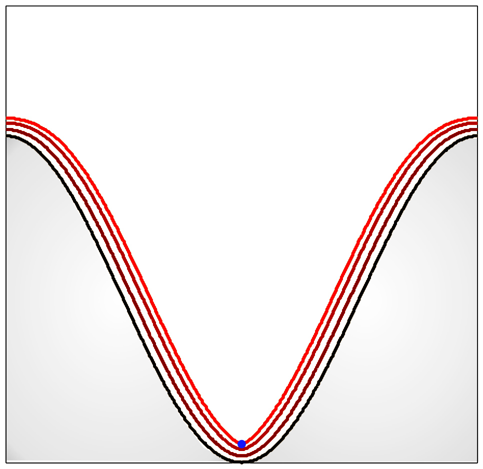

FIGURE 4. *Representation of* $\Omega(t)$ *for* $t = 0, 0.02, 0.04,$ *and* $t = 0.055$ *(from bottom to top), in the flame propagation model. The initially burnt region* $\Omega(0)$ *(in grey) is of class* $\mathcal{C}^\infty$*; however, a singularity (blue dot) appears at approximately* $t_c = 0.055$.

(2) Let us consider the mean curvature flow (1.2), initialized with the 'dumbbell' of Fig. 5 (left). It is observed in [18] that $\Omega(t)$ evolves by shrinking, until two ends of its boundary $\Gamma(t)$ join, as in Fig. 5 (right).

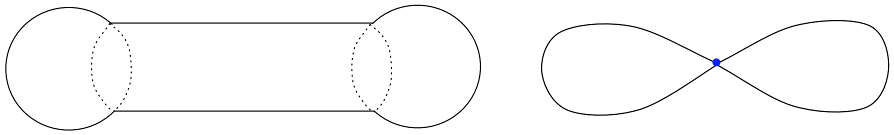

FIGURE 5. *Evolution of a three-dimensional dumbbell under the mean curvature flow (1.2). The central part of the bar ends up pinching.*

After $\Omega(t)$ has developped a singularity, it is unclear how its motion should go on; in both examples discussed above, the normal vector field n_t and mean curvature κ_t of $\Gamma(t)$, are no longer everywhere defined.

What is the 'correct' evolution of $\Omega(t)$ for $t > t_c$ is actually a matter of *definition*, and it is specific to each particular situation. Let us elaborate on the first example discussed above: the perhaps most 'natural' way to characterize the motion of $\Omega(t)$ for $t > t_c$ is to demand that all the points $x \in \Gamma(t)$ where the normal vector $n_t(x)$ is well-defined go on moving independently with unit normal velocity. The subsequent evolution of $\Gamma(t)$ is drawn on Fig. 6 (left): $\Gamma(t)$ is now a self-intersecting curve. Another way to define this subsequent evolution, which relies on the physical meaning of the problem, is to impose an 'entropy criterion': *a point burnt at one time t_0 stays burnt for $t > t_0$*. In more mathematical terms, for any two times $t_1 \leq t_2$, one has $\Omega(t_1) \subset \Omega(t_2)$. The resulting evolution of $\Gamma(t)$ is represented on Fig. 6 (right).

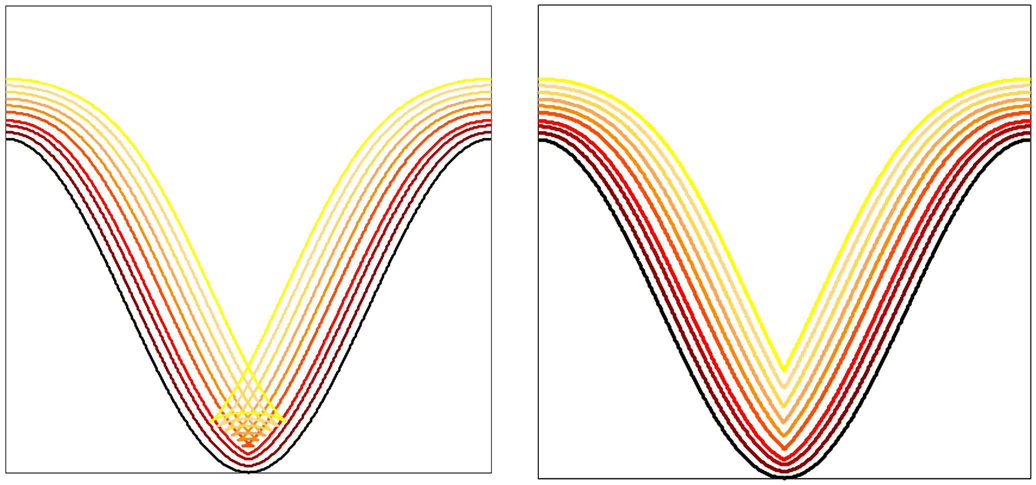

FIGURE 6. *(Left) The 'swallowtail' pattern and (right) the 'cusp' pattern described by $\Omega(t)$, for $t > t_c$ in the flame propagation model.*

Imposing a 'natural' behavior to $\Omega(t)$ once singularities have appeared is not simple in the general case. Historically, several attempts were made to classify all the singularities which may arise in the course of such evolutions, thus allowing to define a subsequent evolution in each case. Unfortunately, this task turned out to be out of reach, especially in the three-dimensional context.

Another idea goes the other way around, and starts from the level set equations: (2.7) and (2.8) are endowed with a generalized notion of 'weak' solutions, that somehow 'encompasses some physics'; if $\phi(t,x)$ is such a weak solution, $\Omega(t)$ is then *defined* as the negative subdomain:

$$\Omega(t) := \left\{ x \in \mathbb{R}^d, \ \phi(t,x) < 0 \right\}.$$

An adapted framework to many such problems turns out to be that of *viscosity solutions*, initially proposed by M. G. Crandall and P.-L. Lions in [20], in a broader context than that of Equations of the form (2.7) and (2.8); the interested reader may find a comprehensive introduction to this rich theory in [19].

Definition 1. *Let $U \subset \mathbb{R}^d$ be an open set and $H : \mathbb{R}^d_x \times \mathbb{R}_u \times \mathbb{R}^d_p \times \mathcal{S}(\mathbb{R}^d)$ be a continuous function (the Hamiltonian). Consider the general second-order Hamilton-Jacobi equation posed on $(0,T) \times U$:*

$$\frac{\partial \phi}{\partial t}(t,x) + H(x, \phi, \nabla\phi, \nabla^2\phi)(t,x) = 0, \tag{2.9}$$

where $\nabla^2\phi$ stands for the Hessian matrix of ϕ.

- *A function ϕ is a viscosity subsolution of (2.9) if it is upper semicontinuous on U, and, for any function φ of class $\mathcal{C}^2$ on U such that $\phi - \varphi$ reaches a local maximum at x,*
$$\frac{\partial \phi}{\partial t}(t,x) + H(x, \phi(x), \nabla\varphi(x), \nabla^2\varphi(x)) \leq 0.$$
- *A function ϕ is a viscosity supersolution of (2.9) if it is lower semicontinuous on U, and, for any function φ of class $\mathcal{C}^2$ on U such that $\phi - \varphi$ reaches a local minimum at x,*
$$\frac{\partial \phi}{\partial t}(t,x) + H(x, \phi(x), \nabla\varphi(x), \nabla^2\varphi(x)) \geq 0.$$
- *ϕ is a viscosity solution of (2.9) if it is both a viscosity subsolution and a viscosity supersolution.*

Remark 1.

- Equations of the form (2.7) and (2.8) directly fall into the framework of the above definition when either the velocity V, or the normal velocity v is independent of ϕ. Such is not the case of the mean curvature flow equation (1.2), which demands a modification of the notion of viscosity solution [25].
- This definition does not allow to deal with a velocity field V which depend on $\Omega(t)$ (i.e. on $\phi(t,\cdot)$) in a non local way, for instance in the case where it involves the solution of some partial differential equation posed on $\Omega(t)$ (see the third category in the classification of Section 1). We mention the work [12] (continued in later articles) in which a notion of solutions to the level set equations is introduced and analyzed qualitatively in cases where the velocity field V arises from shape optimization considerations.

Let us presently list several properties of the viscosity solution ϕ of an equation of the form (2.9), or of the associated negative subdomain $\Omega(t) = \left\{x \in \mathbb{R}^d, \;\; \phi(t,x) < 0\right\}$. These properties attest to the 'physical' meaning of the notion of viscosity solution; note that they are hereafter formulated in a deliberately hazy way, omitting the (very technical) assumptions. Let us solely mention that they hold for fairly general Hamiltonian functions H (see [29], Chap. 4 for precise statements and proofs).

- *Existence and unicity.* The viscosity solution ϕ to (2.9) exists and is unique.
- *Generalization of the notion of classical solution.* If ϕ is of class $\mathcal{C}^2$, then it is also a solution of (2.9) in the classical sense.
- *Vanishing viscosity limit of solutions to 'regular' equations.* For small $\varepsilon > 0$, let $\phi_\varepsilon(t,x)$ be the smooth solution to the equation:
$$\left\{ \begin{array}{l} \frac{\partial \phi_\varepsilon}{\partial t}(t,x) - \varepsilon\Delta\phi_\varepsilon(t,x) + H(x, \phi_\varepsilon, \nabla\phi_\varepsilon, \nabla^2\phi_\varepsilon)(t,x) = 0, \\ \phi_\varepsilon(t=0,\cdot) = \phi(t=0,\cdot) \end{array} \right. ,$$
obtained from (2.9) by adding the regularizing 'vanishing viscosity' term $-\varepsilon\Delta\phi_\varepsilon$. Then, $\phi_\varepsilon \xrightarrow{\varepsilon\to 0} \phi$, uniformly on every compact subset of $[0,T] \times \mathbb{R}^d$.
- *Independence from the level set function characterizing the initial domain.* Let Ω_0 be a domain, and ϕ_0, ψ_0 be two associated level set functions (i.e. (2.1) holds). Let $\phi(t,\cdot)$, $\psi(t,\cdot)$ be the solutions of (2.9) associated to the respective initial data ϕ_0, ψ_0. Then, the domains described by ϕ and ψ coincide, i.e.
$$\left\{x \in \mathbb{R}^d, \;\; \phi(t,x) < 0\right\} = \left\{x \in \mathbb{R}^d, \;\; \psi(t,x) < 0\right\}.$$
- *Monotonicity.* Let $\Omega_0 \subset \widetilde{\Omega_0}$ be two domains of $\mathbb{R}^d$, and $\Omega(t)$, $\widetilde{\Omega}(t)$ be the evolving domains resulting from the process described above; i.e. $\Omega(t)$ (resp. $\widetilde{\Omega}(t)$) is the negative subdomain of $\phi(t,\cdot)$ (resp. $\widetilde{\phi}(t,\cdot)$), the solution to (2.9) with a level set function for $\Omega(0)$ (resp. $\widetilde{\Omega}(0)$) as initial data.
  Then, $\Omega(t) \subset \widetilde{\Omega}(t)$.

Before closing this section, let us appraise the concrete meaning of this notion of solutions on the two examples introduced above.

- As was proved in [9], the evolution of $\Omega(t)$ in the flame propagation model after the first singularity has appeared is that obtained by imposing the so-called 'entropy criterion' (see Fig. 4, right).
- In the case of the mean curvature flow, the two components joined by the singular point separate, and each part shrinks independently to a point. Other interesting examples of domains evolving via the mean curvature flow can be found in [15, 25] and the references therein.

2.4. **Domain evolution as a boundary value problem: Eikonal equations**

An interesting particular case of the general setting of Section 2.2 arises when $\Omega(t)$ *expands* according to a normal velocity, i.e. $V(t,x)$ is of the form:

$$V(t,x) = c(x)n_t(x),$$

with $c(x) > 0$; this holds in particular in the model for flame propagation (1.1). The problem turns out to be equivalently described by the stationary *time function* $T : \mathbb{R}^d \setminus \overline{\Omega(0)} \to \mathbb{R}$, defined by:

$$T(x) = \inf\left\{t \geq 0,\ x \in \Omega(t)\right\},$$

that is, for $x \in \mathbb{R}^d \setminus \overline{\Omega(0)}$, $T(x)$ is the first time t at which $\Omega(t)$ reaches x.

The derivation of a boundary value problem for T follows the same trail as that of the level set equations (2.7) and (2.8): in a first step, it is rigorously established in the regions of space where all the quantities at stake are smooth enough. Then, T is understood as the solution of this partial differential equation in an adequately generalized sense.

Let $x_0 \in \mathbb{R}^d$, and assume that the functions T, c are smooth in a vicinity U of x_0. Using again the intuitive notion of an evolving domain of Section 2.2, let $x(t) := \chi(x_0, t, t_0)$ be the integral curve of V emerging from x_0 at $t = t_0$, which is defined on a neighborhood of t_0 by (2.6); recall that it satisfies the properties:

$$x(t_0) = x_0,\ \ x(t) \in \Gamma(t) \text{ and } x'(t) = c(x(t))n_t(x(t)).$$

On the other hand, it follows from the definition of T that:

$$\Omega(t) = \left\{x \in \mathbb{R}^d,\ T(x) < t\right\},\ \ \Gamma(t) = \left\{x \in \mathbb{R}^d,\ T(x) = t\right\};$$

as a consequence, a level set function associated to $\Omega(t)$ is

$$\forall (t,x) \in [0,T] \times \mathbb{R}^d,\ \ \phi(t,x) := T(x) - t,$$

whence, from formula (2.2), $n_t(x) = \frac{\nabla T(x)}{|\nabla T(x)|}$. Differentiating with respect to t in the relation $T(x(t)) = t$ and incorporating the Dirichlet boundary condition $T(x) = 0$, for all $x \in \Gamma(0)$, we end up with the *Eikonal equation*:

$$\left\{\begin{array}{ll} c|\nabla T| = 1 & \text{in } \mathbb{R}^d \setminus \overline{\Omega(0)} \\ T = 0 & \text{on } \Gamma(0) \end{array}\right. . \tag{2.10}$$

Actually, in the sequel, we will also get interested in the very similar case where $\Omega(t)$ *shrinks* according to a normal velocity c (or $\mathbb{R}^d \setminus \overline{\Omega(t)}$ expands with velocity c):

$$V(t,x) = -c(x)\, n_t(x),$$

with $c(x) > 0$. The time function $T : \Omega(0) \to \mathbb{R}$ is then defined by:

$$T(x) = \inf\left\{t \geq 0,\ x \in \mathbb{R}^d \setminus \overline{\Omega(0)}\right\}.$$

A similar argument reveals that T is now solution to the Eikonal equation:

$$\left\{\begin{array}{ll} c|\nabla T| = 1 & \text{in } \Omega(0) \\ T = 0 & \text{on } \Gamma(0) \end{array}\right. . \tag{2.11}$$

So as to select unambiguously a 'physical' behavior for T, solutions to (2.10) or (2.11) have to be taken into account in a generalized setting, which once again happens to be that of viscosity solutions. Adapting Definition 1 to the present stationary setting (we omit the details, referring again to [19]), the result of interest is now the following (see e.g. [8]):

Theorem 1. *Assume that $\Omega(0) \subset \mathbb{R}^d$ is a bounded domain, and that the normal velocity c is positive and continuous on $\overline{\Omega(0)}$. Then, there exists a unique viscosity solution to the Dirichlet problem (2.11).*

Example 1. Let us briefly look into the interesting particular case of equation (2.11) where the normal velocity field is constant, $c \equiv 1$. The unique viscosity solution to (2.11) is the *Euclidean distance function* $d(.,\Gamma)$ to Γ, defined by:

$$\forall x \in \Omega,\ \ d(x,\Gamma) = \inf_{y \in \Gamma} |x-y|.$$

This fact accounts for the regular spacing out of the isovalues of distance functions, as can be seen on Fig. 7. We will discuss this very important property once again in Section 5. See also the discussion in [50] Chap. 2, about the physical meaning of the viscosity solution in this particular case.

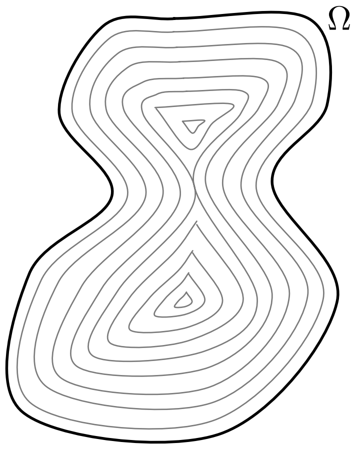

Figure 7. A domain Ω, together with several isolines of the distance function to $\partial\Omega$.

3. A model application of the level set framework

As a preliminary to our discussion of the applications of the Level Set Method in scientific computing, we describe a challenging, real-life problem where it find a natural application. We seize this opportunity to motivate the need for the dedicated algorithms that are presented in the next sections. Throughout the remainder of this chapter, we shall refer to this example to illustrate our purposes.

Let $D \subset \mathbb{R}^d$ be a computational domain, divided into two phases Ω_0 and $\Omega_1 := D \setminus \overline{\Omega_0}$, filled with different fluids with respective (homogeneous) densities and dynamic viscosities ρ_0, ρ_1, and ν_0, ν_1. The interface between the fluids is $\Gamma := \partial\Omega_0 \setminus \partial D$; see Fig. 8. The domains Ω_0, Ω_1 are moving in time, and we aim at solving the bifluid incompressible Navier-Stokes equations over a time period $(0,T)$, that is:

$$(3.1)\qquad \left\{ \begin{array}{cc} \rho_i \left(\frac{\partial u_i}{\partial t} + u_i \cdot \nabla u_i \right) - \nu_i \Delta u_i + \nabla p_i = f_i & \text{for } (t,x) \in (0,T) \times \Omega_i(t), \\ \text{div}(u_i) = 0 & \text{for } (t,x) \in (0,T) \times \Omega_i(t), \\ u_i(t,x) = 0 & \text{for } (t,x) \in (0,T) \times \partial D, \\ u_0(t,\cdot) = u_1(t,\cdot) & \text{on } \Gamma(t), \\ (\sigma_0 - \sigma_1) \cdot n_t = \gamma \kappa_t n_t & \text{on } \Gamma(t), \\ u_i(t=0,\cdot) = u_{i,0}(\cdot) & \text{on } \Omega_i(0). \end{array} \right.$$

In the above system u_i and p_i are the velocity and pressure of the fluid in the phase Ω_i, and $\sigma_i := \nu_i(\nabla u_i + \nabla u_i^T) - p_i I$ is the associated stress tensor. $n_t(\cdot)$ (resp. $\kappa_t(\cdot)$) is the unit normal vector to $\Gamma(t)$, pointing outward $\Omega_0(t)$ (resp. its mean curvature), f_i stands for the applied body force (typically, gravity), and γ is the surface tension coefficient. The interface Γ is consistently moving according to the velocity field $u(t,\cdot)$ (recall that $u_0(t,\cdot)$ and $u_1(t,\cdot)$ coincide on $\Gamma(t)$), and we assume that the initial domains and interface $\Omega_i(0)$, $i = 1,2$, and $\Gamma(0)$ are given.

This example of a moving domain, whose physical and mathematical aspects are broached in Chapter **??**, is one among many where the velocity field depends on global features of this domain.

Let us first sketch the considered numerical setting.

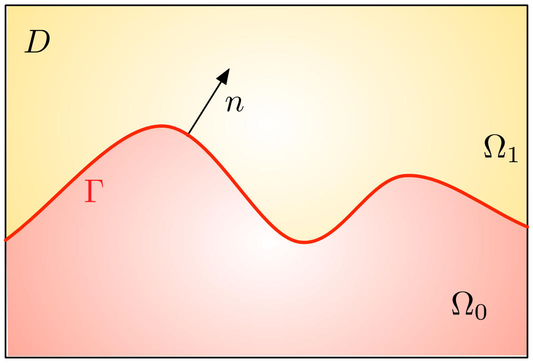

FIGURE 8. Model situation for the bifluid problem (3.1).

Notations: The time interval $(0,T)$ is split into subintervals (t^n, t^{n+1}) of 'small' length $\Delta t = t^{n+1} - t^n$, where $n = 0, ..., T/\Delta t$. When $\phi(t,x)$ is a quantity depending on time and space, $\phi^n(x)$ stands for the sought numerical approximation of $\phi(t^n, x)$, for $x \in D$.

As for the spatial discretization, the computational domain D is discretized with a mesh $\mathcal{T}$, which we do not specify for the time being; for instance, it may be a Cartesian grid, or a simplicial mesh (i.e. composed of triangles in $2d$, tetrahedra in $3d$). This mesh comes with a set of *degrees of freedom*, which we assume to be the vertices $\{x_i\}_{i\in I}$ of $\mathcal{T}$ for simplicity.

The two phases $\Omega_0(t), \Omega_1(t)$ (and thus the interface $\Gamma(t)$) are tracked by means of a Level Set function $\phi(t,\cdot)$ for the domain $\Omega_0(t)$ (i.e. (2.1) holds with $\Omega_0(t)$ instead of Ω). This function is, at all times, numerically discretized at the vertices of $\mathcal{T}$. Hence, the quantity of interest when it comes to ϕ are the $\phi^n(x_i)$, for $n = 0, ..., N$ and $i \in I$.

A tentative outline of an algorithm for the numerical resolution of (3.1) reads as follows:

- **Initialization:** *Create a level set function* ϕ^0 (at the nodes of mesh $\mathcal{T}$) associated to the initial domain $\Omega_0(0)$.
- **Main loop:** for $n = 0, ..., N-1$,
  1. *Motion of the domain:* solve the level set advection equation (2.7) (or the level set Hamilton-Jacobi equation (2.8)) over the time period (t^n, t^{n+1}), using the vector velocity u^n (resp. the scalar, normal velocity $u^n \cdot n$), and the initial condition ϕ^n.
     Implicitely, this relies on the assumption that the time step Δt is small enough so that the velocity of the fluid $u(t,\cdot)$ may be 'frozen' over (t^n, t^{n+1}):
     $$\forall t \in (t^n, t^{n+1}),\ \ x \in D,\ \ u(t,x) \approx u^n(x).$$
  2. *Resolution of the partial differential equation:* Solve the Navier-Stokes Bifluid equation (3.1) over the time period (t^n, t^{n+1}) to obtain the new velocity u^{n+1}.

This discussion highlights the general needs for numerical methods associated to the Level Set Method:

1. We need an algorithm for generating a level set function ϕ^0 associated to the initial domain $\Omega_0(0)$. In Section 5, we present efficient numerical schemes to carry out this initialization step.
2. We also need to solve the level set advection equation (2.7), or the Level Set Hamilton-Jacobi equation (2.8) over a given time period, and with given initial value and (scalar or vector) velocity field. This issue is tackled in Section 4.
3. There are operations that we need to perform on a domain Ω which is only known through an associated level set function ϕ. Here are some of them:

- Calculate the normal vector n and the curvature κ of Ω (which, in the example above, appear in the transmission conditions in (3.1),
- Evaluate the integral of a given function over Ω or Γ (in our context, this is needed for assembling the stiffness matrix associated to (3.1))

These operations will be addressed in Section 6.

(4) The last main interrogation is about the resolution of the Navier-Stokes bifluid equation (3.1). This step is generally hard to perform, mainly because of the complexity of the equation. This issue is not specific to the practice of the Level Set method, and its numerical treatment is specific to each equation; it goes therefore beyond the scope of these notes.

The efficiency, robustness, and relative simplicity in terms of programming effort, of the numerical methods for the aforementioned operations contribute to the popularity of the Level Set Method: it is a fast and accurate means to account for arbitrarily large deformations of a domain, including topological changes. This last point is especially appreciated in the context of bifluid flows, where one may for instance want to track the motion of coalescing bubbles.

4. Solving the Level set Hamilton-Jacobi equation

We have seen that the motion of an domain $\Omega(t)$ driven by a velocity field $V(t,x)$ translates in terms of an associated level set function into the partial differential equations (2.7) and (2.8). Recall that $V(t,x)$ may be of a quite intricate nature (in particular, it may depend on the evolving domain $\Omega(t)$ in a complex way), which makes these equations difficult to solve in the general context.

As we have seen in Section 3, the numerical treatment of (2.7) and (2.8) often requires to split the time interval $(0,T)$ into subintervals (t^n,t^{n+1}) of 'small' length $\Delta t = t^{n+1}-t^n$, and to 'freeze' the velocity field on each subinterval:

$$\forall t \in (t^n,t^{n+1}),\;\; V(t,x) \approx V(t^n,x).$$

Thus, the evolution equation (2.7) becomes a true advection equation on each interval (t^n,t^{n+1}), and can be solved relying on well-established numerical techniques (see e.g. [36]).

A slightly different point of view arises when one wants to preserve the information that the evolution of $\Omega(t)$ is oriented along the normal vector $n_t(x)$, which is a crucial feature in several applications. Then, only the normal component of V is frozen:

$$\forall t \in (t^n,t^{n+1}),\;\; V(t,x) \approx v(t^n,x)n_t(x),$$

and the evolution equation (2.7) becomes a true Hamilton-Jacobi equation. However, the resolution of such equations is not so standard as that of advection equations.

In the present section, we discuss the numerical discretization of the Hamilton-Jacobi equation:

$$(4.1)\qquad \begin{cases} \dfrac{\partial\phi}{\partial t} + v|\nabla\phi| = 0 & \text{on } (0,T)\times\mathbb{R}^d \\ \phi(0,.) = \phi_0 & \text{on } \mathbb{R}^d \end{cases}$$

for given normal velocity field $v(x)$, and initial function ϕ_0. The theory of numerical schemes for (4.1) is part of the more general theory of numerical schemes for first order Hamilton-Jacobi equations of the form:

$$(4.2)\qquad \begin{cases} \dfrac{\partial\phi}{\partial t} + H(x,\nabla\phi) = 0 & \text{on } (0,T)\times\mathbb{R}^d, \\ \phi(0,.) = \phi_0 & \text{on } \mathbb{R}^d, \end{cases}$$

and arises as the particular case when $H(x,p) = v(x)|p|$. The forthcoming discussion takes place in this general context, which encompasses major applications outside from the level set framework, and perhaps allows to better appraise the salient features of the theory.

A wide variety of methods exists to deal with such problems, and we shall only sketch a few basic ones. For the sake of clarity, and without loss of generality, our discussion takes place in the two-dimensional case: $d = 2$.

4.1. **Solving the Level Set Hamilton-Jacobi equation on Cartesian grids**

Notations: Let $N \in \mathbb{N}$, and $\Delta t = \frac{T}{N}$ be a time step, according to which $(0,T)$ is divided into subintervals (t^n, t^{n+1}), $n = 0, ..., N-1$, with $t^n = n\Delta t$. As for the space discretization, in this subsection, the computational support is a Cartesian grid, with step Δx in the x-direction, and Δy in the y-direction. For a quantity ϕ discretized at the vertices of the grid, and for $i,j \in \mathbb{Z}$, ϕ_{ij} denotes the value assigned to the node $x_{ij} := (i\Delta x, j\Delta y)$.

Like in the context of hyperbolic systems of conservation laws, so-called *upwind* discretizations play a key role in the device of numerical schemes for Hamilton-Jacobi equations. Denote the finite difference quantities:

$$D^{+x}_{ij}\phi = \frac{\phi_{i+1j} - \phi_{ij}}{\Delta x} \quad ; \quad D^{-x}_{ij}\phi = \frac{\phi_{ij} - \phi_{i-1j}}{\Delta x},$$

(and likewise for $D^{+y}_{ij}\phi$ and $D^{-y}_{ij}\phi$), which are respectively referred to as *forward* and *backward* finite differences.

Our aim is to compute an approximation to the viscosity solution ϕ of (4.2) as a sequence $\phi^n = \{\phi^n_{ij}\}_{i,j\in\mathbb{Z}}$, with the meaning that ϕ^n_{ij} is an approximation of $\phi(t^n, x_{ij})$. An explicit, first-order numerical scheme which fulfills this role can be written under the general form:

$$\left\{ \begin{array}{ll} \forall i,j \in \mathbb{Z}, & \phi^0_{ij} = \phi_0(i\Delta x, j\Delta y) \\ \forall n \in \mathbb{N}, i,j \in \mathbb{Z}, & \phi^{n+1}_{ij} = \phi^n_{ij} - \Delta t\, \mathcal{H}\left(x_{ij}, D^{-x}_{ij}\phi^n, D^{+x}_{ij}\phi^n, D^{-y}_{ij}\phi^n, D^{+y}_{ij}\phi^n\right) \end{array} \right. , \tag{4.3}$$

where the *numerical Hamiltonian*

$$\mathcal{H}\left(x_{ij}, D^{-x}_{ij}\phi^n, D^{+x}_{ij}\phi^n, D^{-y}_{ij}\phi^n, D^{+y}_{ij}\phi^n\right) \tag{4.4}$$

is intended as an approximation of $H(x_{ij}, \nabla\phi(x_{ij}))$. Two properties are desirable about $\mathcal{H}$:

Definition 2. *An explicit, first-order scheme for (4.2) of the form (4.3) is said to be:*

- *Consistent if, for any $x \in \mathbb{R}^2$ and $p \in \mathbb{R}^2$, $\mathcal{H}(x, p_x, p_x, p_y, p_y) = H(x,p)$. In other words, the finite difference terms in (4.4) stand for the corresponding first-order derivatives of ϕ in $H(x, \nabla\phi)$ where they should.*
- *Monotone if, for any $x \in \mathbb{R}^2$, and any $i,j \in \mathbb{Z}$, the update function*
$$\{\phi_{kl}\}_{k,l\in\mathbb{Z}} \longmapsto \phi_{ij} - \Delta t\, \mathcal{H}\left(x, D^{-x}_{ij}\phi, D^{+x}_{ij}\phi, D^{-y}_{ij}\phi, D^{+y}_{ij}\phi\right)$$
is increasing with respect to each of its arguments.

One can indeed prove that, under 'reasonable' assumptions over the theoretical Hamiltonian H and the initial data ϕ_0, consistent and monotone first-order schemes converge to the viscosity solution of (4.2) [21, 59].

Let us now discuss the construction of a numerical Hamiltonian for the particular Cauchy problem (4.1). The simplest approximation of (4.1) is the following explicit first-order finite difference scheme:

$$\left\{ \begin{array}{ll} \forall n \in \mathbb{N}, i,j \in \mathbb{Z}, & \phi^{n+1}_{ij} = \phi^n_{ij} - \Delta t \left(\max(v_{ij},0)\nabla^+_{ij}\phi^n + \min(v_{ij},0)\nabla^-_{ij}\phi^n\right) \\ \forall i,j \in \mathbb{Z}, & \phi^0_{ij} = \phi_0(i\Delta x, j\Delta y) \end{array} \right. , \tag{4.5}$$

where the discretizations $\nabla^+_{ij}\phi$ and $\nabla^-_{ij}\phi$ of $|\nabla\phi|$ are defined as:

$$\nabla^+_{ij}\phi = \left(\begin{array}{c} \max(\max(D^{-x}_{ij}\phi, 0), -\min(D^{+x}_{ij}\phi, 0))^2 \\ +\max(\max(D^{-y}_{ij}\phi, 0), -\min(D^{+y}_{ij}\phi, 0))^2 \end{array} \right)^{\frac{1}{2}} \tag{4.6}$$

and

$$\nabla^-_{ij}\phi = \left(\begin{array}{c} \max(\max(D^{+x}_{ij}\phi, 0), -\min(D^{-x}_{ij}\phi, 0))^2 \\ +\max(\max(D^{+y}_{ij}\phi, 0), -\min(D^{-y}_{ij}\phi, 0))^2 \end{array} \right)^{\frac{1}{2}} \tag{4.7}$$

The quantity $\nabla^+_{ij}\phi$ (resp. $\nabla^-_{ij}\phi$) is said to be *upwind* (resp. *downwind*), since its calculation only involves values of the $\{\phi_{kl}\}_{k,l\in\mathbb{Z}}$ which are smaller (resp. larger) than ϕ_{ij}.

Likewise, the discretization of the (exact) Hamiltonian $H(x,p) = v(x)|p|$ by the numerical one:

$$\forall n \in \mathbb{N}, i,j \in \mathbb{Z},\; H(x_{ij}, \nabla\phi(x_{ij})) \approx \mathcal{H}_{ij}(\{\phi^n_{kl}\}_{k,l\in\mathbb{Z}}) := \max(v_{ij},0)\nabla^+_{ij}\phi^n + \min(v_{ij},0)\nabla^-_{ij}\phi^n$$

can be deemed *upwind* in the sense that, for given i, j, n, the update $\phi^n_{ij} \to \phi^{n+1}_{ij}$ only involves information coming from smaller values than ϕ^n_{ij} if v_{ij} is positive, and larger values than ϕ^n_{ij} if it is negative.

The numerical scheme (4.5) is consistent in the sense of Definition 2. However, it is not unconditionnally monotone; indeed, it is indeed easily seen that the following CFL-like relation connecting Δt and $\Delta x, \Delta y$ must be satisfied for the latter property to hold:

$$\left(\sup_{i,j} v_{ij}\right) \frac{\Delta t}{\min(\Delta x, \Delta y)} \leq 1. \tag{4.8}$$

Roughly speaking, (4.8) means that the information should not propagate over more than one space step Δx or Δy within one time step Δt.

Under this CFL condition, the scheme (4.5) turns out to be convergent. Moreover, an explicit error estimate of the discrepancy between the numerical solution $\{\phi^n_{kl}\}_{k,l\in\mathbb{Z}}$ and the exact viscosity solution $\phi(t, x)$ of (4.1) can be achieved:

$$\forall i, j \in \mathbb{Z},\ \forall n \leq N,\ \ |\phi^n_{ij} - \phi(t^n, x_{ij})| \leq C\sqrt{\Delta t}, \tag{4.9}$$

for a constant C depending only on the data of the problem (4.1) (the initial function ϕ_0, etc.) We refer to [21, 59] for further details.

Unfortunately, this scheme is only first-order accurate, as is reflected by the 'bad' error estimate (4.9); in particular, it turns out to be very *diffusive*, meaning that sharp features of the exact solution ϕ to (4.1) tend to result smeared in the numerical solution $\{\phi_{kl}\}_{k,l\in\mathbb{Z}}$. It is therefore desirable to construct higher-order schemes for solving (4.1) and (4.2). Mimicking conservation laws, it is possible to build high-order, adaptive-stencil schemes, known as *(weighted) Essentially Non Oscillatory* schemes (abridged as (w)ENO); see [31, 46], or the lecture notes [58] for further details.

4.2. **Solving the Level Set Hamilton-Jacobi equation on triangular meshes**

The theory we just skimmed through in the previous subsection can be extended to the case of triangular meshes. Adequate generalizations of the notions of *consistency* and *monotonicity* introduced in Definition 2 make it possible to build a theory for convergent schemes on triangular meshes, a glimpse of which we now provide - see [1] for details.

Notations: As in the last subsection, the time interval $(0, T)$ is still split into subintervals (t^n, t^{n+1}), $n = 0, ..., N-1$, $t^n = n\Delta t$.

The plane $\mathbb{R}^2$ is endowed with a conforming triangular mesh $\mathcal{T}$, and the vertices of $\mathcal{T}$ are denoted by $\{x_i\}_{i\in I}$. If ϕ is a piecewise affine function on $\mathcal{T}$, ϕ_i is the value assigned to the node x_i. In this context, $\nabla\phi$ is a vector-valued function with constant values in restriction to each triangle of $\mathcal{T}$.

At each time t^n, an approximation of the viscosity solution $\phi(t^n, .)$ to (4.2) is sought under the form of a piecewise affine function ϕ^n on $\mathcal{T}$. A general explicit, first-order numerical scheme for (4.2) on $\mathcal{T}$ can be written as:

$$\forall i \in I,\ \ \phi^{n+1}_i = \phi^n_i - \Delta t \mathcal{H}(x_i, \phi^n), \tag{4.10}$$

where $\mathcal{H}(x_i, \phi)$ is the *numerical Hamiltonian*. This notation may seem inappropriate at first glance, since the theoretical Hamiltonian H only depends on ϕ through its gradient, and one could expect the same behavior from $\mathcal{H}(x_i, \phi)$. Actually, $\mathcal{H}(x_i, \phi)$ will depend only on $\nabla\phi$, but it is better expressed in terms of ϕ.

Let us now outline one possible procedure for constructing $\mathcal{H}(x_i, \phi)$, which is very reminiscent of Lax-Friedrichs numerical schemes in the context of hyperbolic systems of conservation laws. Let $C(H)$ be one Lipschitz constant for the theoretical Hamitonian H; if $h > 0$ is smaller than the smallest edge of $\mathcal{T}$, one defines:

$$\forall i \in I,\ \ \mathcal{H}(x_i, \phi) = H(x_i, \overline{\Phi}_i) - \frac{C(H)}{2\pi h} \int_{\mathcal{C}_h(x_i)} (\phi(x) - \phi(x_i))\ d\ell(x),$$

where $\overline{\Phi_i}$ is the mean value of $\nabla\phi$ over the disk $\mathcal{D}_h(x_i)$ of center x_i and radius h (see Figure 9):

$$\overline{\Phi_i} = \frac{1}{\pi h^2}\int_{\mathcal{D}_h(x_i)} \nabla\phi \, dx,$$

and $\mathcal{C}_h(x_i) := \partial\mathcal{D}_h(x_i)$ is the circle of center x_i and radius h.

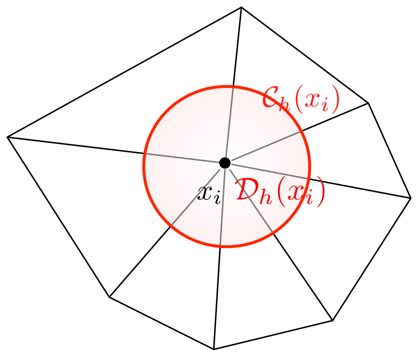

FIGURE 9. *The numerical scheme for triangular meshes of Section 4.2: setting and notations.*

It is possible to prove that the sequence $\{\phi^n\}_{n=0,\dots,N}$ obtained using (4.10) converges (in an appropriate sense) to the viscosity solution of (4.2), provided the following CFL condition holds:

$$\Delta t \leq \frac{h}{C(H)}. \tag{4.11}$$

Remark 2. Other methods exist for solving (4.1) and (4.2) on a triangular mesh; see for instance [10], where, in particular, so-called *Petrov-Galerkin* approximations are introduced, featuring a usual Finite Element formulation which is stabilized by the addition of least-square quantities.

4.3. **Semi-Lagrangian schemes**

So-called *semi-Lagrangian* techniques attempt to compute the solution ϕ to (4.1) or (4.2) by tracking the flow of information attached to the equation. In this section, we illustrate the main idea with a short and formal description of the semi-Lagrangian scheme of Strain [60] for the level set Hamilton-Jacobi equation (4.1); as, in their broad lines, semi-Lagrangian methods are fairly independent of the particular choice of a computational support, we purposely do not specify which space discretization is used for functions.

As usual, let us subdivide the time interval $(0,T)$ into subintervals of the form (t^n, t^{n+1}). Semi-Lagrangian schemes advocate to compute the value $\phi(t^{n+1},x)$ of the solution ϕ at time t^{n+1} and at a (grid) point $x \in \mathbb{R}^d$ by tracking the point $y \in \mathbb{R}^d$ such that the information of $\phi(t^{n+1},x)$ 'comes from' $\phi(t^n,y)$. To be precise, first recall that equation (4.1) stems (at least in a formal way) from the level set advection equation:

$$\left\{ \begin{array}{ll} \frac{\partial\phi}{\partial t} + V(t,x).\nabla\phi = 0 & \text{on } (0,T)\times\mathbb{R}^d, \\ \phi(0,.) = \phi_0 & \text{on } \mathbb{R}^d, \end{array} \right. \tag{4.12}$$

where the vector velocity field $V : (t,x) \mapsto V(t,x) \in \mathbb{R}^d$ depends on ϕ and is defined as:

$$V(t,x) = v(t,x)\,\frac{\nabla\phi(t,x)}{|\nabla\phi(t,x)|}.$$

The nonlinear equation (4.12) is approximated on each subinterval (t^n, t^{n+1}) by the linear advection equation obtained by freezing the value of $V(t,.)$ over (t^n,t^{n+1}), i.e. by setting:

$$\forall t \in (t^n, t^{n+1}), \forall x \in \mathbb{R}^d,\ \ V(t,x) = V(t^n,x) =: V^n(x).$$

In other terms, assuming that an approximation ϕ^n of $\phi(t^n,.)$ has been computed, $\phi(t,.)$ is sought on (t^n,t^{n+1}) as the solution $\psi : (t^n,t^{n+1}) \times \mathbb{R}^d \to \mathbb{R}$ to:

$$\begin{cases} \frac{\partial \psi}{\partial t} + V^n(x).\nabla\psi = 0 & \text{on } (0,T)\times\mathbb{R}^d, \\ \psi(t^n,.) = \phi^n & \text{on } \mathbb{R}^d, \end{cases}$$

whose exact solution can be computed owing to the *method of characteristics* (as is quite customary in the field of computational fluid mechanics):

$$\forall x \in \mathbb{R}^d,\ \ \psi(t^{n+1},x) = \psi(t^n, \chi(x,t^n,t^{n+1})) = \phi^n(\chi(x,t^n,t^{n+1})), \tag{4.13}$$

where $(t^n,t^{n+1}) \ni t \mapsto \chi(x,t,t^{n+1})$ is the *characteristic curve* of V^n, reaching x at time t^{n+1}, defined by (2.6).

With this guidance in hand, the following procedure for approximating the solution ϕ to (4.1) is derived:

- **Initialization:**
  Start with an approximation ϕ^0 of $\phi(0,.)$ on the considered computational support (e.g. at the nodes of a Cartesian grid, or as a piecewise linear function on a simplicial mesh).
- **Loop (for $n = 1,...,N-1$):**
  Loop (for each degree of freedom x of the computational support):
  (1) Calculate $t \mapsto \chi(x,t,t^{n+1})$, and search for the 'foot' $y := \chi(x,t^n,t^{n+1})$ of the characteristic curve reaching x at time t^{n+1} (see Chapter **??** for numerical methods to achieve this).
  (2) Mimicking formula (4.13), the value of ϕ^n at y is computed (using interpolation on the computational support) to produce $\phi^{n+1}(x)$.

The benefits of this approach are numerous: among others, it is easily parallelized since the nodes of the mesh are processed independently from one another. Moreover, the stability of the method does not depend on a CFL condition over the time step Δt such as (4.8) and (4.11). On the other hand, this technique also has some drawbacks, such as its lack of accuracy in regions where the exact solution ϕ is not smooth.

This semi-Lagrangian paradigm (which in the hitherto considered case of Equation (4.1) boils down to solving (2.7)) may be extended to general Hamilton-Jacobi equations of the form (4.2); the particular situation discussed above exemplifies the three typical stages of a general semi-Lagrangian method. Between two consecutive time steps t^n, t^{n+1}, an approximation $\phi^{n+1}(x)$ of the value $\phi(t^{n+1},x)$ is computed using:

(1) A *space-time integration step*, which amounts to searching for the point $y \in \mathbb{R}^d$ such that 'the information about $\phi(t^{n+1},x)$ comes from $\phi(t^n,y)$' (in our case, y is the foot $\chi(x,t^n,t^{n+1})$ of the characteristic curve passing through x at time t^{n+1}).
(2) A *spatial interpolation step*, in which $\phi^n(y)$ is interpolated from the values of ϕ^n at the grid nodes surrounding y.
(3) An *update step*, in which $\phi^{n+1}(x)$ is computed from $\phi^n(y)$ (in the above case, this step is trivial; see Formula (4.13)).

Using this idea in the context of a Hamilton-Jacobi equation of the form (4.2) demands a means to complete the first and third stages of the previous program. This is generally achieved thanks to representation formulae for the exact solutions to (4.2), which generalize the method of characteristics in the case of more general Hamiltonian functions H (e.g. the *Hopf-Lax formula* when H is convex and does not depend on the x variable). See [26] for further details.

5. Initializing level set functions

We have already pointed out the fact that many level set functions can be associated to a domain $\Omega \subset \mathbb{R}^d$. The theoretical framework of Section 2 is independent of which function ϕ is chosen, as long as it fulfills (2.1). Things are very different in numerical practice. It turns out that too steep or too loose variations of ϕ near the boundary Γ may cause instabilities in locating accurately its position, or difficulties in the accurate evaluation of the normal vector or curvatures of Γ by means of formulae such as (2.2) to (2.4) (see e.g. [16]).

This pleads for initializing a level set function for Ω as the distance function - and more precisely (for sign purposes) as the *signed distance function* d_Ω to Ω, defined by:

$$\forall x \in \mathbb{R}^d, \;\; d_\Omega(x) = \begin{cases} -d(x, \partial\Omega) & \text{if } x \in \Omega \\ 0 & \text{if } x \in \Gamma \\ d(x, \partial\Omega) & \text{if } x \in {}^c\overline{\Omega} \end{cases}$$

Indeed, we have seen in Example 1 that the isolines of d_Ω are very 'regularly' spaced out. Besides, among its appealing properties, it turns out that d_Ω is smooth near $\partial\Omega$, provided Γ is a smooth boundary.

In this section, we describe several 'classical' algorithms for calculating the (signed or unsigned) distance function to a domain Ω on a computational support (see also Section 7.1). Most numerical methods for generating d_Ω are more generally devoted to Eikonal equations of the form (2.10) or (2.11). The Fast Marching Method, originally introduced in [54], is no exception in this regard.

5.1. **The Fast Marching Method on Cartesian grids**

Let us briefly present the Fast Marching Method for approximating the solution $T : \mathbb{R}^2 \setminus \overline{\Omega} \to \mathbb{R}$ of the Eikonal equation (2.10), at the nodes of a Cartesian grid of the plane lying outside Ω. Extensions of this discussion to the resolution of (2.11) at the nodes lying inside Ω and to the three-dimensional setting are rather straightforward.

Reusing the notations of Section 4.1, the Fast Marching Method is an iterative process which produces at each step $n \in \mathbb{N}$ a scalar numerical quantity $T^n := (T^n_{ij})_{i,j\in\mathbb{Z}}$ intended as an increasingly accurate approximation of the solution T to (2.10).

One the one hand, the Fast Marching Method relies on an update strategy, whereby the values of T^n are computed in a one-by-one, upwind fashion, mimicking the propagation of a front starting from Γ. More precisely, the nodes x_{ij} of the Cartesian grid are consistently classified into three categories:

- The *accepted* nodes: these are the nodes x_{ij} 'where the front has already passed', i.e. at which the current value T^n_{ij} is considered to have converged. Once a value has become accepted, it is no longer updated.
- The *active* nodes: these are the nodes x_{ij} 'on the front'. One of their four neighbors $x_{i-1j}, x_{i+1j}, x_{ij-1}$ or x_{ij+1} is an accepted node, and a first approximation (*trial value*) T^n_{ij} to the desired solution has been computed, but may still be subject to updates.
- The *far* nodes: these are the nodes 'still far from the front', whose values have not been approximated yet (and are set to ∞).

At each iteration, the algorithms *accepts* one node, to be selected among the active nodes. The set of active nodes is then redefined, and the update procedure begins: the values T^n_{ij} at active nodes are updated according to the new information about the front.

This update procedure $T^n_{ij} \to T^{n+1}_{ij}$ of the values of T at active nodes x_{ij} is the second key feature of the algorithm. At every such node, a trial value $\widetilde{T^n_{ij}}$ is computed as the solution to the following equation, which is an upwind discretization of the Eikonal equation (2.10):

$$\sqrt{\begin{array}{l} \max\left(\max\left(\frac{\widetilde{T^n_{ij}}-T^n_{i-1j}}{\Delta x}, 0\right), -\min\left(\frac{T^n_{i+1j}-\widetilde{T^n_{ij}}}{\Delta x}, 0\right)\right)^2 \\ +\max\left(\max\left(\frac{\widetilde{T^n_{ij}}-T^n_{ij-1}}{\Delta y}, 0\right), -\min\left(\frac{T^n_{ij+1}-\widetilde{T^n_{ij}}}{\Delta y}, 0\right)\right)^2 \end{array}} = \frac{1}{c_{ij}}. \tag{5.1}$$

Note that (5.1) is intrinsically *upwind*, since the resulting value $\widetilde{T^n_{ij}}$ is only influenced by the values of T^n at the four neighbors of x_{ij} that are smaller than $\widetilde{T^n_{ij}}$: this is a means to impose a *causality principle* which is inherent to Hamilton-Jacobi equations. Only the *accepted* values among the set $\left\{T^n_{i-1j}, T^n_{i+1j}, T^n_{ij-1}, T^n_{ij+1}\right\}$ are used in the polynomial equation (5.1), and it must be checked that the obtained solution $\widetilde{T^n_{ij}}$ is larger than those values. In the end, T^{n+1}_{ij} is obtained as:

$$T^{n+1}_{ij} = \min\left(\widetilde{T^n_{ij}}, T^n_{ij}\right).$$

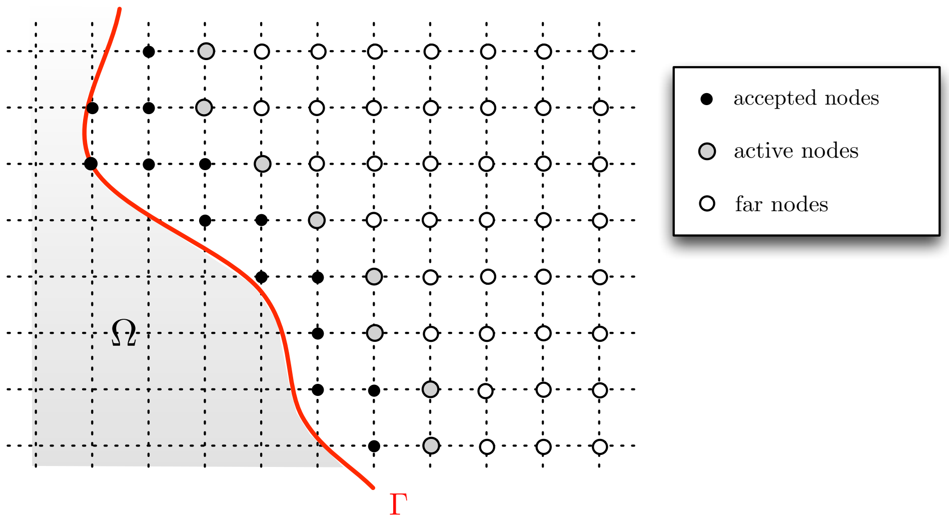

FIGURE 10. *Setting of the Fast Marching Method*

To sum up, the Fast Marching algorithm proceeds along the lines of the following sketch:

- **Initialization:**
  (1) Compute the value of the exact solution (2.10) at the nodes of the cells which intersect Γ (this operation is fast in practice), and mark them as *accepted*.
  (2) Use the local update procedure (5.1) to compute a trial value at the neighbor points to the accepted points which have not been yet accepted, and mark them as *active*.
  (3) Mark all the remaining nodes as *far*, and assign them the value ∞.

- **Loop (while the set of active nodes is not empty):**
  (1) Travel the set of active nodes, and identify the one with minimum trial value. This node becomes accepted.
  (2) Identify the new set of active nodes, and compute a new trial value for each one of them, using the local update solver (5.1) for the Eikonal equation.

This algorithm produces a sequence (T^n_{ij}) which converges towards the unique viscosity solution to (2.10); see [22] for a precise statement of this fact and a proof.

Furthermore, it can be seen that, if in practice, the computation is restrained to a large bounding domain (e.g. a box), equipped with a Cartesian mesh consisting of N vertices, the Fast Marching procedure converges within $\mathcal{O}(N \log(N))$ operations, which makes it very efficient in numerical practice.

5.2. **Extension of the Fast Marching Method to triangular meshes**

Let us now give a hint of how the Fast Marching Method can be adapted to the context of a triangular mesh of $\mathbb{R}^2$, using the notations introduced in Section 4.2, and following the work [35].

The general outline of the previous algorithm is unchanged: the vertices of the mesh are still tagged as either *accepted*, *active*, or *far*. At each iteration, the active vertex whose value is minimal becomes accepted once and for all. The set of active vertices is then adequately redefined, and the active values are updated.

The update procedure is the only different feature between both versions of the Fast Marching algorithm. Here, it occurs in a situation where two values of T, say $T_i \leq T_j$ are known at two vertices x_i, x_j of a triangle $K = x_i x_j x_k \in \mathcal{T}$, and a trial value $\widetilde{T_k}$ is sought at x_k.

To this end, T is approximated by its piecewise linear interpolate $\pi_K T$ on T from the accepted values T_i, T_j and the sought trial value $\widetilde{T_k}$ at x_i, x_j, x_k respectively. Let c_K be an approximation of c over K (e.g.

c_K may be the mean value of c over K). $\widetilde{T_k}$ is sought so as to satisfy:

$$|\nabla(\pi_K T)|^2 = c_K^2.$$

More precisely, the desired solution $\widetilde{T_k}$ to this quadratic equation should be larger than T_i and T_j, and satisfy some additional properties (related to the causality inherent to equation (2.10) discussed above), which are omitted here. If such a solution $\widetilde{T_k}$ exists, the value T_k is updated as $T_k = \min(T_k, \widetilde{T_k})$.

Let us mention a potential difficulty in this approach. Depending on the shape of the triangulation $\mathcal{T}$, it may happen that, in the course of the update of the value T_k at x_k, no triangle having x_k as a vertex has accepted values at both other vertices, which makes it impossible to rely on the previous local procedure to compute a trial value $\widetilde{T_k}$ at x_k. This is especially likely to happen when one or several triangles having x_k as a vertex show an obtuse angle at x_k. A special procedure is required in this case to reduce to the previous case.

The exact same construction can be used to generate the distance function to a subset of a triangulated surface in $\mathbb{R}^3$ (this is actually the original setting of the work [35]). Yet, this procedure is more difficult to extend to the case of a computational mesh of $\mathbb{R}^3$ composed of tetrahedra.

Remark 3. Again, many other numerical methods exist to solve Eikonal equations of the form (2.10); for an alternative technique, the so-called *Fast Sweeping Method*, see [67].

6. Operations within the Level Set framework

The Level Set Method features an implicit description of the domain Ω under consideration; however well-suited this framework for tracking its evolution, as we have discussed, the absence of any explicit discretization (e.g. a mesh) of Ω may be prejudicial for certain operations on Ω.

We hereafter list the most common such operations, and popular numerical recipes to carry them out by using only a level set function ϕ for Ω.

Throughout this section, Ω stands for a (smooth) bounded domain in $\mathbb{R}^d$ with boundary Γ, and ϕ is an associated Level Set function.

6.1. Evaluation of the normal vector or the mean curvature of Γ

Relying on the theoretical expression (2.2) of the normal vector n to Γ (pointing outward Ω), a numerical approximation can be achieved by the following formula:

$$n(x) \approx \frac{\nabla\phi(x)}{\sqrt{|\nabla\phi(x)|^2+\varepsilon^2}},$$

in which ε is a 'very small' parameter (typically of the order of 10^{-5}) acting as a safeguard against degeneracy of $\nabla\phi$. The way to evaluate numerically the involved partial derivatives depends on the computational support at hand; for instance, when ϕ is discretized at the vertices of a Cartesian grid, finite difference approximations of various orders may be used, for example, simple first-order finite differences or more sophisticated ENO approximations.

Likewise, an approximation of the mean curvature $\kappa(x)$ of Γ is based on (2.4):

$$\kappa(x) \approx \operatorname{div}\left(\frac{\nabla\phi(x)}{\sqrt{|\nabla\phi(x)|^2+\varepsilon^2}}\right).$$

Note that both formulae actually make sense outside from Γ, so that they actually are approximations to *extended* normal vector and mean curvature fields to Γ.

6.2. Calculating integrals on Ω and Γ.

Let us now address the numerical evaluation of volume or surface integrals of the form:

$$I = \int_\Omega f(x)\,dx, \text{ and } J = \int_{\partial\Omega} g(x)\,ds,$$

where $f, g : \mathbb{R}^d \to \mathbb{R}^d$ are given (smooth) functions. A simple calculation of I relies on the following approximation of the characteristic function χ_Ω of Ω in terms of ϕ:

$$\chi_\Omega(x) \approx H_\varepsilon(\phi(x)) := \frac{1}{2}\left(1 - \frac{\phi(x)}{\sqrt{\phi^2(x) + \varepsilon^2}}\right), \quad x \in \mathbb{R}^d \tag{6.1}$$

where H_ε is a smoothed approximation of the characteristic function of the interval $(-\infty, 0)$ in $\mathbb{R}$, and ε is again a 'small' parameter. In turn, I can be approximated as:

$$I \approx \int_{\mathbb{R}^d} f(x) H_\varepsilon(\phi(x))\, dx,$$

the last integral being straightforward to evaluate relying on appropriate quadrature formulae, and depending on the computational support used for the discretization of ϕ.

The approximation of the integral J is slightly more intricated; it relies on an approximation of the *surface measure* of Γ, that is, of the distribution $\delta_\Gamma \in \mathcal{D}'(\mathbb{R}^d)$ corresponding to the integration over Γ:

$$\forall \varphi \in \mathcal{C}^\infty_c(\mathbb{R}^d), \;\; \langle \delta_\Gamma, \varphi \rangle = \int_\Gamma \varphi \, ds.$$

An application of Green's formula reveals that the following identity holds, in the sense of distributions:

$$\delta_\Gamma = -\frac{\partial \chi_\Omega}{\partial n}, \text{ where } n \text{ is a divergence-free extension of the normal vector to } \Gamma,$$

whence we base the approximation:

$$\delta_\Gamma \approx -\frac{\partial}{\partial n}\left(H_\varepsilon(\phi)\right).$$

Then, the following formula lends itself to an easy numerical evaluation:

$$J \approx -\int_{\mathbb{R}^d} \frac{\partial}{\partial n}\left(H_\varepsilon(\phi(x))\right) g(x)\, ds.$$

6.3. **Algebraic operations over sets**

The main algebraic operations involving domains $\Omega, \Omega_1, \Omega_2 \subset \mathbb{R}^d$ can be achieved by simple manipulations over corresponding level set functions ϕ, ϕ_1, ϕ_2:

- A level set function ϕ_c for the complement ${}^c\overline{\Omega}$ of Ω is obtained as:
$$\phi_c = -\phi.$$
- A level set function ϕ_u for the union $\Omega_1 \cup \Omega_2$ is:
$$\phi_u = \min(\phi_1, \phi_2).$$
- A level set function ϕ_i for the intersection $\Omega_1 \cap \Omega_2$ is:
$$\phi_i = \max(\phi_1, \phi_2).$$

Notice that even if ϕ_1 and ϕ_2 are smooth, ϕ_u and ϕ_i may fail to be so.

6.4. **Solving partial differential operations posed on Ω**

Let Ω be a domain of interest, included in a larger, bounded domain $D \subset \mathbb{R}^d$. In practical situations, D is a computational domain, and it is equipped with a mesh (a Cartesian grid, a triangular mesh, etc.); Ω is solely known via an associated level set function ϕ, which is discretized on the mesh of D; in particular, no mesh of Ω is available. Let us consider the resolution of the following partial differential equation on Ω:

$$\left\{ \begin{array}{cl} -\text{div}(a\nabla u) + u = f & \text{in } \Omega, \\ \frac{\partial u}{\partial n} = 0 & \text{on } \Gamma \end{array} \right. , \tag{6.2}$$

where $a > 0$ and $f \in L^2(\mathbb{R}^d)$. In the present context, it is difficult to rely on classical numerical methods (e.g. with the Finite Element method) since no mesh of Ω is available.

The idea is to approximate the solution u to (6.2) with that u_ε to an approximate problem, defined on the whole working domain D, in such a way that:

$$u_\varepsilon \approx u \text{ on } \Omega.$$

Multiple ways exist to achieve this purpose. One possibility is to consider the following approximate problem:

$$\left\{ \begin{array}{cl} -\text{div}(c_\varepsilon(x) a \nabla u) + c_\varepsilon(x) u = c_\varepsilon(x) f & \text{in } D, \\ \frac{\partial u}{\partial n} = 0 & \text{on } \partial D \end{array} \right., \text{ where } c_\varepsilon(x) = \left\{ \begin{array}{cl} 1 & \text{for } x \in \Omega, \\ \varepsilon & \text{for } x \in D \setminus \Omega \end{array} \right. ; \tag{6.3}$$

i.e. the 'natural' extension of (6.2) to D where the void region $D \setminus \overline{\Omega}$ is filled with a very soft material. It is indeed possible to prove that:

$$||u - u_\varepsilon||_{H^1(\Omega)} \xrightarrow{\varepsilon \to 0} 0.$$

Problem (6.3) is now easier to simulate than (6.2), considering the data at hand: it is posed on the (meshed) domain D, and the cutoff $c_\varepsilon(x) = \varepsilon + \chi_\Omega(x)(1-\varepsilon)$ can be approximated as:

$$c_\varepsilon(x) \approx \varepsilon + H_\varepsilon(\phi(x))(1-\varepsilon),$$

where $H_\varepsilon(\phi)$ is the approximate characteristic function (6.1) of Ω based on the level set function ϕ.

Remark 4.

- The approximate equation (6.3) intrinsically depends on the exact equation (6.2) to be solved. For instance, consider the problem

$$\left\{ \begin{array}{cl} -\text{div}(a \nabla u) + u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{array} \right.$$

  i.e. the version of (6.2) where Neumann boundary conditions on $\partial\Omega$ are replaced with homogeneous Dirichlet boundary conditions. One proves that u is approximated by the solution u_ε of:

$$\left\{ \begin{array}{cl} -\text{div}(a \nabla u) + c_\varepsilon(x) u = f & \text{in } D, \\ u = 0 & \text{on } \partial D \end{array} \right., \text{ where } c_\varepsilon(x) = \left\{ \begin{array}{cl} 1 & \text{for } x \in \Omega, \\ \frac{1}{\varepsilon} & \text{for } x \in D \setminus \Omega. \end{array} \right. ,$$

  instead of that of (6.3).
- Similar approximation processes hold, e.g. for the linear elasticity equations (this is the so-called *Ersatz material approach*), or for the Stokes equations (the so-called *porosity* or *Brinkman approximation*).

7. Additional features

We now describe several additional issues that frequently come up in the numerical implementation of the Level Set Method. In this section, $\Omega(t)$ stands for a domain evolving a period of time $(0,T)$, and $\phi(t,\cdot)$ is an associated level set function. To set ideas, we focus on the situation where the motion is driven by a normal velocity given by the scalar function $v(t,x)$; consequently, ϕ arises as the solution to (4.1).

7.1. Redistancing

We already pointed out at the practical importance of manipulating level set functions that 'resemble' signed distance functions in Section 5. Unfortunately, even if $\phi(0,.)$ is the signed distance function to $\Omega(0)$, the solution $\phi(t,.)$ to the level set equation (4.1) is likely to develop very sharp or loose variations in the vicinity of $\Gamma(t)$ as $t > 0$ increases. This phenomenon may jeopardize the stability of the whole numerical resolution of (4.1).

In practical applications, it is crucial to periodically restore $\phi(t,.)$ as the signed distance function to $\Omega(t)$: this is the purpose of *level set redistancing* (or *re-initialization*).

The situation is as follows: $\Omega \subset \mathbb{R}^d$ is a bounded domain, about which a (possibly very 'irregular') level set function ϕ_0 is available; we aim at generating the signed distance function d_Ω to Ω from these data. The most straightforward idea in this direction consists of course in using one of the algorithms of Section 5. Doing so would be a waste somehow, since we already have one level set function for Ω, and we could hope to take advantage of this knowledge. In such a situation, it has been proposed in [61] to 'regularize' the level set function ϕ_0 into a new one, close to d_Ω. To this end, ϕ_0 is used as the initial state of the *redistancing equation*:

$$\left\{ \begin{array}{ll} \frac{\partial \psi}{\partial t}(t,x) + \text{sgn}(\phi_0(x))\,(|\nabla \psi| - 1) = 0 & \text{for } (t,x) \in (0,\infty) \times \mathbb{R}^d \\ \psi(0,x) = \phi_0(x) & \text{for } x \in \mathbb{R}^d \end{array} \right. . \tag{7.1}$$

The underlying intuition is that, as the stationary state of (7.1) is obtained, the property $|\nabla \psi| = 1$ is restored (which is formally obtained by cancelling the time derivative in (7.1)). The presence of the sign function in (7.1) accounts for the fact that a signed distance function is sought as the stationary state. This very formal explanation is theoretically justified in [6].

In practice, the redistancing equation (7.1) is solved using adequate numerical schemes, in the spirit of those presented in Sections 4 and 5, relying on a smoothing of the discontinuous function $\text{sgn}(\phi_0)$.

Remark 5. However crucial, this redistancing procedure suffers from unwelcome side effects; first, using it 'too often' may undermine the efficiency of the whole process in terms of CPU time (however, in most applications, the cost of redistancing is negligible when compared to e.g., that of a finite element calculation). A more serious issue is that performing this redistancing operation 'too often' may entail a loss of accuracy of the process, since the interface Γ is slightly shifted in solving (7.1) (that is, the 0 level set of $\psi(t,\cdot)$ does not match exactly that of ϕ_0 in numerical practice) Therefore, a balance has to be reached between redistancing 'often', and not 'too often'.

7.2. **Velocity extension**

Another recurring difficuly in the practice of the Level Set Method is that, in a wide class of applications, the (scalar) velocity field $v(t,x)$ is only defined at points x on the boundary $\Gamma(t)$ of the evolving domain. This is a pity, for as we saw in Section 2.2, the Level Set Method requires $v(t,x)$ to be defined over $\mathbb{R}^d$ (at least on a neighborhood of $\Gamma(t)$ if we follow the Narrow Band paradigm of Section 7.3). Actually, even in situations where there is a 'natural' way to extend $v(t,\cdot)$ outside $\Gamma(t)$, it often proves clumsy.

It is thus a key issue to extend $v(t,\cdot)$ into another field $v_{\text{ext}}(t,\cdot)$, which is defined on a neighborhood of $\Gamma(t)$. There is actually a great deal of latitude as for how to do so: the only requirement is that v_{ext} should coincide with $v(t,\cdot)$ on $\Gamma(t)$. This extension procedure can therefore be achieved in many ways, and we briefly outline two popular ones in the literature, with different assets.

7.2.1. *Normal extension of the velocity*

In [4], the authors propose a velocity extension approach which alleviates the need for redistancing the level set function. It relies on the following observation: assume that, for $t \geq 0$, the solution $\phi(t,\cdot)$ to the level set Hamilton-Jacobi equation (4.1) with normal velocity v is the signed distance function to $\Omega(t)$; a formal calculation, blithely ignoring regularity issues reveals that:

$$0 = \frac{\partial}{\partial t}\left(|\nabla \phi|^2\right) = -2|\nabla \phi| \nabla \phi \cdot \nabla v - 2v \nabla \phi \cdot \nabla\left(|\nabla \phi|\right) = -2\nabla \phi \cdot \nabla v \quad \text{on } \mathbb{R}^d,$$

where we have used (4.1) and the property $|\nabla \phi| = 1$. Hence, if we are to select the extended velocity field v_{ext} so that $\phi(t,\cdot)$ stays a signed distance function, it must obey the equation (see the appendix in [68] for further discussion):

$$\nabla v_{\text{ext}}(t,x) \cdot \nabla \phi(t,x) = 0. \tag{7.2}$$

Recalling Formula (2.2) for the (extended) normal vector n_t to $\Gamma(t)$, (7.2) means that, at every time t, $v_{\text{ext}}(t,\cdot)$ should have constant values along the integral curves of n_t.

Let us now slip into the numerical setting: recall that the time interval $(0,T)$ is divided into subintervals (t^n, t^{n+1}) of length Δt, $n = 0, ..., N = T/\Delta t$. The Hamilton-Jacobi equation (4.1) is solved on each interval (t^n, t^{n+1}) with initial data $\phi^n := \phi(t^n,\cdot)$, and velocity field $v^n := v(t^n,\cdot)$. This velocity field is defined on $\Gamma(t^n)$ and has to be extended to, say, $\mathbb{R}^d$.

One method to stick with the previous observations - insofar as possible - goes the following way:

(1) Calculate the signed distance function d_{Ω^n} to Ω^n. This can be achieved thanks to one of the methods of Section 5, or, more efficiently, by solving the redistancing equation (7.1) (since we have at hand one level set function ϕ^n for Ω^n).

(2) Calculate v_{ext}^n as the solution to:

$$\begin{cases} \nabla v_{\text{ext}}^n \cdot \nabla d_\Omega^n = 0 & \text{in } \mathbb{R}^d \setminus \Gamma(t^n) \\ v_{\text{ext}}^n = v^n & \text{on } \Gamma(t^n). \end{cases}$$

This system shares many features with boundary-value problems of the form (2.11) and can be solved owing to a procedure very similar to the Fast Marching Method.

We refer to the article [17] for further discussions on this method, and for a similar procedure in the context of Fast Marching Methods.

7.2.2. *PDE-based extension*

A different extension technique allows to extend v into a field v_{ext} which is guaranteed to be smooth. Let us describe this procedure at a fixed iteration of a Level Set evolution procedure: we have a domain Ω (standing for Ω^n at the considered time t^n), and we wish to extend $v \equiv v^n$ to the larger computational domain D.

One possibility is to search for the unique solution $v_{\text{ext}} \in H^1(D)$ to the following equation:

$$\begin{cases} -\alpha \Delta v_{\text{ext}} + v_{\text{ext}} = 0 & \text{in } D, \\ v_{\text{ext}} = v & \text{on } \partial\Omega \end{cases} . \tag{7.3}$$

Doing so guarantees some degree of smoothness for v_{ext}, since it belongs to $H^1(D)$.

This procedure is more expensive than that described in the previous section, since it involves the resolution of the elliptic equation (7.3) (which is typically achieved thanks to a Finite Element calculation). However, it admits other, very interesting applications and variations, for instance if we relax the constraint that $v_{\text{ext}} = v$ on Γ; taking such freedom may be unacceptable in some applications (for instance in the physical simulation outlined in Section 3, but may even be very desirable in some other, for instance in shape optimization.

7.3. **Towards enhancing the efficiency of the Level Set method: the Narrow Band approach**

At this point, it is worth wondering about the efficiency of the Level Set method. Indeed, the problem of evolution of a surface is traded for that of a quantity defined on a space of higher dimension, and one may expect that this entails a significant increase in computational cost.

Several techniques have been thought up to increase this efficiency, among which parallel implementations [37], and mesh adaptation procedures (see e.g. [10], or the recent [2]). However interesting, we do not get into the details of these methods and refer to the aforementionned articles.

Another key improvement of the level set method is the so-called *Narrow Band* approach. In most applications where the evolution of a domain $\Omega(t)$ (or that of its boundary $\Gamma(t)$) is described by that of an associated Level Set function $\phi(t, \cdot)$, only the motion of the 0 level set of ϕ is of interest. As was originally remarked in [16], then systematized in [3], it is therefore sufficient to carry out the operations related to the Level Set Method (initializing the level set function, redistancing it, solving Equation (4.1), extending the velocity field) on a 'narrow band' around $\Gamma(t)$. Here is a very rough sketch of this paradigm, in which we retain the Cartesian setting and the notations of Section 4.1 (see Fig. 11 for an illustration).

Loop (From $n = 0$, while $n < N$):

- **Initialization** of a narrow band $\mathcal{B}$ of 'Near points' around the interface $\Gamma(t^n)$. $\mathcal{B}$ may be constructed by relying on the distance function to $\Gamma(t^n)$, or simply as a tube of k elements around $\Gamma(t^n)$.
- **Loop:**
  (1) Perform an attempt iteration

  $$\forall i, j \in \mathbb{Z} \text{ s.t. } x_{ij} \in \mathcal{B}, \ \left\{\phi_{ij}^n\right\} \mapsto \left\{\phi_{ij,\text{temp}}^{n+1}\right\}$$

  in the resolution of the Level Set Hamilton-Jacobi equation (4.1) using, e.g., the numerical scheme of Section 4.1. Only the nodes of the grid lying inside the narrow band $\mathcal{B}$ are updated, and a special attention must be paid to the calculation of derivatives at the points of $\mathcal{B}$ close to its border. The process can be additionally accelerated by using adapted data structures for the storage of the nodes in the narrow band.

(2) Analyze the position of the attempt front $\Gamma_{\text{temp}}(t^{n+1})$ (this can be done by looking at the sign of the quantity ϕ^{n+1}_{temp} at the nodes lying on the border of the narrow band):
   - **If $\Gamma_{\textbf{temp}}(t^{n+1})$ lies inside the narrow band** $\mathcal{B}$, accept the iteration:

$$n \leftarrow n+1, \ \left\{\phi^n_{ij}\right\} \leftarrow \left\{\phi^{n+1}_{ij,\text{temp}}\right\},$$

     and go back to (1).
   - **Else**, refuse the iteration, interrupt the loop, and go back to the Initialization step.

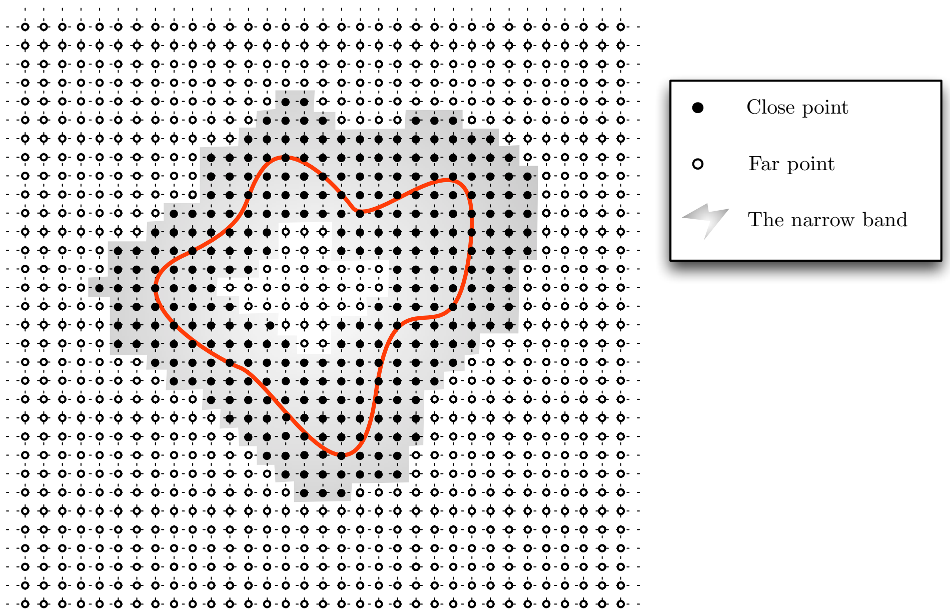

FIGURE 11. Setting of the Narrow Band Method; a band $\mathcal{B}$ (the greyed region) of *close* points (in black) is constructed around the interface $\Gamma(t^n)$ (in red), and the other points of the grid are tagged as *far*. Updates $\phi^n_{ij} \mapsto \phi^{n+1}_{ij}$ are performed only for black points x_{ij} as long as the front $\Gamma(t^n)$ stays inside $\mathcal{B}$.

7.4. Handling multiple phases with the Level Set method

We have hitherto used the Level Set method to account for the motion of one domain, that we shall denote as $\mathcal{O}(t)$ in this section - and only in this section. Equivalently, this amounts to describing the motions of the two phases $\Omega_0 := \mathcal{O}$ and $\Omega_1 := D \setminus \overline{\mathcal{O}}$, where D is the fixed computational domain, and where we dropped the mention to time. Note that this point of view was already used in Section 3.

As was noticed in [64], a small increment in this method allows to describe multiple phases: using m domains $\mathcal{O}_0$,..., $\mathcal{O}_{m-1} \subset D$ it is possible to represent up to 2^m domains $\Omega_0, ..., \Omega_{2^m-1}$, obtained by combining the $\mathcal{O}_i$, (see Fig. 12):

$$(7.4) \qquad \left\{ \begin{array}{ccc} \Omega_0 & = & \mathcal{O}_0 \cap \mathcal{O}_1 \cap ... \cap \mathcal{O}_{m-2} \cap \mathcal{O}_{m-1}, \\ \Omega_1 & = & \mathcal{O}_0 \cap \mathcal{O}_1 \cap ... \cap \mathcal{O}_{m-2} \cap^c \overline{\mathcal{O}_{m-1}}, \\ ... & & \\ \Omega_{2^m-1} & = & {}^c\overline{\mathcal{O}_0} \cap {}^c\overline{\mathcal{O}_1} \cap ... \cap {}^c\overline{\mathcal{O}_{m-2}} \cap^c \overline{\mathcal{O}_{m-1}}. \end{array} \right.$$

Using m Level Set functions ϕ_i, associated to the domains $\mathcal{O}_i$, $i = 0, ..., m-1$, the domains Ω_j, $j = 0, ..., 2^m - 1$, are alternatively defined as:

$$\left\{ \begin{array}{lcl} \Omega_0 & = & \{x \in D,\ \phi_0(x) < 0,\ \phi_1(x) < 0, ..., \phi_{m-2}(x) < 0,\ \phi_{m-1}(x) < 0\}\,, \\ \Omega_1 & = & \{x \in D,\ \phi_0(x) < 0,\ \phi_1(x) < 0, ..., \phi_{m-2}(x) < 0,\ \phi_{m-1}(x) > 0\}\,, \\ ... & & \\ \Omega_{2^m-1} & = & \{x \in D,\ \phi_0(x) > 0,\ \phi_1(x) > 0, ..., \phi_{m-2}(x) > 0,\ \phi_{m-1}(x) > 0\}\,. \end{array} \right.$$

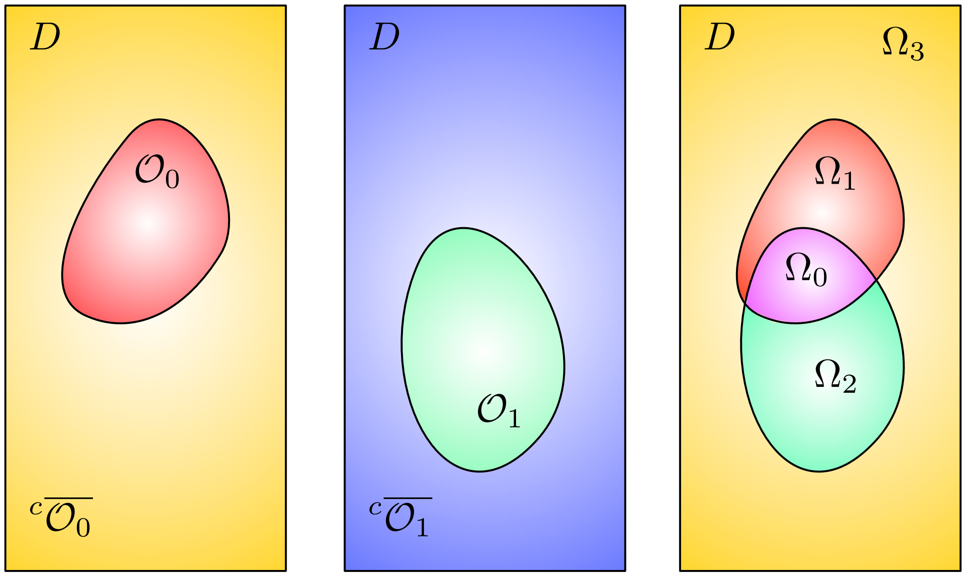

FIGURE 12. *Using two domains $\mathcal{O}_0$ and $\mathcal{O}_1$ yields the four phases Ω_j, $j = 0, ..., 3$.*

When it comes to describing the evolution of each phase Ω_j, the velocity fields driving their motions unambiguously yield those V_i driving the motions of the $\mathcal{O}_i$, $i = 0, ..., m-1$. The Level Set Method can then be independently applied to each domain $\mathcal{O}_i$ along the lines of the previous sections.

Remark 6. Obviously, the number M of described phases is not limited to a power of 2. In the general case, taking m as the unique integer such that $2^{m-1} < M \leq 2^m$, and using m domains $\mathcal{O}_i \subset D$, and m associated level set functions ϕ_i in the way described above, one simply obtains the desired phases as reunions of some of the Ω_j, $j = 0, ..., 2^m - 1$ defined in (7.4).

8. Other domain evolution methods

A wide literature exists for the numerical treatment of problems involving moving domains or interfaces. In this section, we give a rough sketch of the main existing methods, which is deliberately clear-cut, and biased: in particular, this classification and the terminology used here are far from being uniform in the literature.

Throughout this section, $\Omega(t) \subset D$ denotes a moving domain in $\mathbb{R}^d$ with boundary $\Gamma(t)$, included in a larger, fixed computational domain D. Its motion is driven by the (vector) velocity field $V(t, x)$.

In the numerical setting, the time period $(0, T)$ of the evolution is, as usual, discretized into the sequence $t_n = n\Delta t$, $n = 0, ..., N := T/\Delta t$. Sometimes, for brevity, the explicit mention to time is dropped, and we shall simply refer to Ω, Γ, V, etc.

8.1. Lagrangian methods

The first great category is that of *Lagrangian methods*, sometimes also called *moving mesh methods.* They are possibly the most natural ones: at each step t^n of the process, the domain $\Omega(t^n)$ (or, sometimes, its boundary $\Gamma(t^n)$ only) is equipped with a mesh $\mathcal{T}^n$. Hence, such methods are particularly relevant when the velocity field $V(t,x)$ driving the motion is dictated by physical properties, such as a fluid flow, a case in which calculating V demands accurate numerical simulations on Ω (see the bifluid flow example of Section 3).

From one step t^n to the next one t^{n+1}, the mesh $\mathcal{T}^n$ is 'moved' into the new mesh $\mathcal{T}^{n+1}$, according to the velocity field $V^n := V(t^n, \cdot)$. For example, the most naive way to achieve this operation consists in transporting each node x_i^n of $\mathcal{T}^n$ into a node x_i^{n+1} of $\mathcal{T}^{n+1}$ as:

$$x_i^{n+1} = x_i^n + \Delta t \, V(t^n, x_i^n),$$

without altering the connectivity of the mesh. Unfortunately, doing so is likely to yield an invalid mesh, that is, a mesh in which some elements are overlapping; see Fig. 13 for an example.

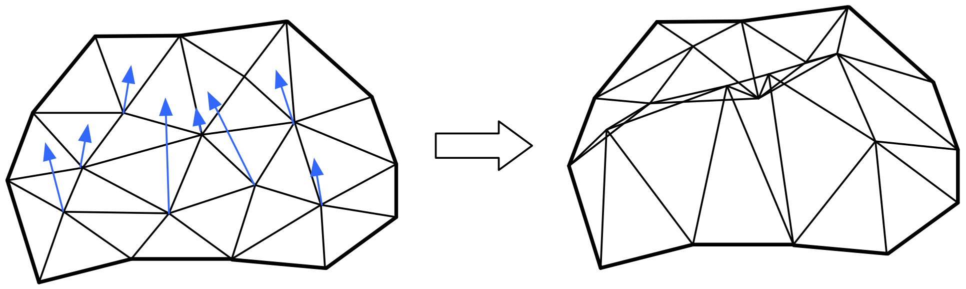

FIGURE 13. *The transport of each node x_i of the mesh via the vector field $V(t,x)$ is likely to cause element overlappings.*

This operation of deforming a mesh according to a velocity field is complex in general (in particular, it is difficult to account for topological changes in the underlying domains in this way). Several heuristics can however help the process; see [28], Chap. 23, for more details around the issue of mesh deformation:

- An occasional *remeshing* of $\mathcal{T}^n$ - e.g. resampling its nodes, modifying its connectivities to eliminate very ill-shaped elements - may postpone the occurrence of difficulties in the mesh deformation process; on the contrary, excessive remeshing could cause a loss of computational efficiency and accuracy.
- The values of the velocity field $V(t,x)$ are only relevant on the boundary of the moving domain $\Omega(t)$; in particular, it is possible to modify $V(t,x)$, while retaining its values on $\Gamma(t)$, so that it entails less distortion in the internal part of $\Omega(t)$ (making the mesh deformation easier). The techniques used to achieve this have a lot in common with the velocity extension procedures described in Section 7.2.

Let us also mention that other Lagrangian techniques represent the evolving domain Ω by means of a set of *particles* instead of a mesh: grossly speaking, Ω is described by a set of points which are not connected to one another. The motion of Ω is considerably simpler under this form. Of course, such techniques must come along with adequate numerical methods to perform operations on Ω, such as solving partial differential equations. See [38] for further details around those so-called *mesh-free* methods.

The reader may consult [39, 62] for an overview and further discussions around Lagrangian methods.

8.2. **Eulerian methods**

Unlike Lagrangian methods, Eulerian methods assume a fixed computational support, that is, a mesh $\mathcal{G}$ (often a Cartesian grid) of a computational domain D. As the evolving domain Ω is not explicitly discretized, some information about it has to be kept somehow. The nature of the retained information allows to distinguish two subclasses, namely *front-tracking* and *interface-capturing* methods.

8.2.1. *Front-tracking methods*

Front-tracking methods rely on an explicit discretization $\mathcal{T}^n$ of the interface - or *front* - $\Gamma(t^n)$ at every step t^n of the numerical process. This mesh may be obtained in several different ways:

- It may be maintained from the beginning, and transported from one step t^n to the next t^{n+1}. This operation is achieved by relying on techniques very similar to those described in Section 8.1, in the case of Lagrangian methods. Like then, it may be accompanied with remeshing algorithms so that $\mathcal{T}^n$ stays well-shaped, and with some heuristics (which may use the computational grid $\mathcal{G}$) to detect and resolve possible self-intersections.
- It may be reconstructed at each step t^n, for instance from the datum of a set of markers (or particles) whose evolution is calculated.

Contrary to the case of Lagrangian methods, the velocity field V characterizing the motion is calculated by using the computational mesh $\mathcal{G}$ of D, and not directly the mesh $\mathcal{T}^n$.

To set ideas, let us focus the discussion on the case where the calculation of V demands the resolution of a PDE posed on Ω (we have in mind the example of Section 3). This PDE is solved on the Cartesian grid $\mathcal{G}$ at each step t^n; hence, ways have to be found to account for the communication between this grid and the mesh $\mathcal{T}^n$ of the front. This can be done in several ways:

- Interpolation procedures may be devised from $\mathcal{T}^n$ into $\mathcal{G}$, to locate the position of $\Omega(t^n)$ on $\mathcal{G}$; this allows for the calculation of quantities of interest (integrals on $\Omega(t^n)$, curvature of $\Gamma(t^n)$, etc...).
- A whole new mesh of $\Omega(t^n)$ may be generated from $\mathcal{G}$ and $\mathcal{T}^n$. Of course, this procedure is quite costly, and very dependent on the robustness of the used meshing algorithm.

The possibly best known implementations of front-tracking algorithms are those detailed in the articles [30, 63]. See also [34] for an interesting overview and complementary features.

Without getting into details, let us mention the *Immersed Boundary method*, introduced by C. Peskin in [48] (see also [42, 49] for overviews), which can be viewed as a front-tracking method in the particular context of fluid-structure interactions. In a nutschell, the idea is to describe an elastic structure immersed in a fluid by using Lagrangian variables for the former part (a part which is numerically discretized and tracked thanks to Lagrangian techniques: moving mesh or particles), and Eulerian variables for the latter (a part which is numerically discretized on a fixed grid); the coupling between both parts is achieved owing to operations which are, in spirit, similar to those used in classical front-tracking methods.

8.2.2. *Interface-capturing methods*

Interface-capturing methods account for the moving domain Ω via auxiliary quantities, which are discretized on the fixed mesh $\mathcal{G}$ of D. The Level Set Method falls into this class, the evolution of Ω being described by that of a Level Set function ϕ; at least two other techniques are worth mentioning.

8.2.2.1. Volume of Fluid Methods.

The Volume of Fluid Method (often abridged as VOF) was pioneered in the article [32], and has long been a reference method to account for the motion of a fluid domain, or that of several fluid phases.

The basic idea is to represent a fluid domain Ω by a scalar function C, which is constant by element $T \in \mathcal{G}$ - T may be a e.g. a triangle, a square, depending on the nature of $\mathcal{G}$. The quantity C_T stands for the 'proportion of fluid' present in the element T, that is:

$$C_T = \frac{1}{|T|} \int_{C_T} \chi_\Omega(x)\, dx,$$

where χ_Ω is the characteristic function of Ω. In other words, C_T equals 1 if T is completely occupied by the fluid, 0 if it does not carry any fluid, and it takes an intermediate value if it is crossed by the interface Γ.

A formal analysis reveals that this function C satisfies the evolution equation:

$$\frac{\partial C}{\partial t} + V \cdot \nabla C = 0, \tag{8.1}$$

which is to be understood in a very loose sense since C is constant par element! For this reason (and others), the numerical resolution of (8.1) is difficult and requires adequate numerical schemes. Most of those involve a *reconstruction* step, where the boundary of the fluid domain Ω is inferred from the datum of C.

As any interface-capturing method, the Volume of Fluid method naturally adresses the problem of topological changes. Furthermore, it has proved particularly successful in preserving the volume of Ω in the case of an incompressible flow - a key feature in fluid simulations. However, the quality of the description of the boundary Γ is inherently less accurate than that featured by the Level Set method.

See [51, 52] for reviews of several aspects of the Volume of Fluid Method.

8.2.2.2. Phase-Field Methods.

The Phase Field Method was originally introduced in the context of two phases Ω_0, Ω_1, separated by an interface Γ. The situation is described by a so-called *order parameter*, or *Phase Field* scalar function ϕ, which is discretized on the mesh of the computational domain D in the numerical setting. This function ϕ assumes constant values in the bulk phases Ω_0, Ω_1 (typically -1 and 1), and shows steep, yet smooth, variations in a thin tubular neighborhood of Γ.

Instead of giving an abstruse presentation of the theoretical aspects of the method, let us concentrate the discussion on the particular example of the bifluid flow model of Section 3, whose notations are reused.

The various quantities characterizing the model (in our case, the density ρ and the viscosity ν of the fluid) which are discontinuous at the interface Γ, are approximated by quantities featuring a sharp, but smooth transition across this interface, for instance:

$$\forall x \in D,\ \nu(x) \approx \nu(\phi(x)) := \frac{\nu_1 - \nu_0}{2}\phi(x) + \frac{\nu_1 + \nu_0}{2}. \tag{8.2}$$

This phase field model is a smoothed approximations of the *sharp-interface*, exact model (3.1).

As far as it is concerned, the function ϕ is assumed to minimize a so-called *free energy* functional $J_\varepsilon(\phi)$, which is, for example, of the form:

$$J_\varepsilon(\phi) = \int_D \left(\frac{1}{\varepsilon} W(\phi) + \frac{\varepsilon}{2} |\nabla \phi|^2 \right) \, dx. \tag{8.3}$$

In this formula, W is a so-called *double-well* potential, which assumes two minima at ± 1. The meaning of this description is the following: ϕ is in competition between minimizing the double-well $W(\phi)$ (which drives it towards a function taking only the values ± 1, regardless of having sharp variations), and the gradient term, which constrains ϕ to retain some smoothness. The parameter ε balancing both trends can be shown to be a measure of the thickness ε of the smeared interface thus imposed by the method. Notice that, as $\varepsilon \to 0$, the constraint on the smoothness of the variations of ϕ is loosened in the expression (8.3).

The evolution of ϕ in time is driven by the minimization of the free energy $J_\varepsilon(\phi)$. Writing conservation laws to express this feature, in connection with the fact that ϕ is dragged by the fluid velocity $u(t, \cdot)$ of the phases leads to a diffusion equation for ϕ, either of the *Allen-Cahn* or of the *Cahn-Illiard* type.

Depending on the point of view of the user, the Phase-Field Method can be seen as a theoretical and numerical artifice to mimic the sharp-interface formulation of the evolution problem at stake. It is then worth ensuring that the different choices (the free energy $J_\varepsilon(\phi)$, the evolution equation for ϕ, the coupling between ϕ and u expressed in equations such as (8.2), etc.) allow to retrieve the original model in some appropriate sense when the thickness parameter ε goes to 0.

On the other hand, the Phase-Field Method can be thought off as a way to incorporate a complex, microstructural behaviors of the interface Γ in a macroscopic model. In this case, the phase-field function ϕ is given a physical meaning, and the choices of the free energy and the diffusion equation for ϕ are based upon physical (notably thermodynamical) considerations.

From the numerical point of view, the Phase-Field Method has been successfully applied to several evolution problems such as, for instance, solidification [66] or multiphase flow [7]. It shares a lot of aspects with the Level Set Method: as a purely Eulerian method, it handles large deformations (including topological changes) in a natural way; it may imply a stronger, physical coupling between the phase representation ϕ and the physical equations which lends itself to theoretical convergence analyses. However, the geometrical description of the interface Γ is less accurate than in the context of the Level Set Method.

See [11, 14] for introductions to the Phase-Field Method.

8.3. **Hybrid methods**

Numerous hybrid algorithms exist between Lagrangian and Eulerian methods, the perhaps most famous of which are *Arbitrary Eulerian-Lagrangian* methods (ALE in short).

In Lagrangian methods, the computational mesh of the considered domain $\Omega(t)$ is transported in time according to the velocity field $V(t,x)$ of the motion. Doing so potentially offers maximum accuracy regarding the location of $\Omega(t)$, but this procedure may dramatically degrade the quality of the mesh, compromising the accuracy of computations. On the other hand, Eulerian methods use a fixed mesh of a computational domain D at every step of the process.

As their name suggest, ALE methods go halfway between both points of view: the domain where calculations are carried out (and with it, the computational mesh) is moved in an arbitrary way, that is, independent from the exact position of $\Omega(t)$. The choice of a motion for this so-called *ALE domain* $\Omega_R(t)$, is dictated by a tradeoff between staying close from the deformed domain $\Omega(t)$, without jeopardizing with the quality of the attached mesh. The efficiency of an ALE method strongly depends on the devised strategy for the evolution of the ALE domain, and their description goes beyond the scope of this introduction.

To sum up, at a given time $t > 0$, three domains are considered:

- The initial domain $\Omega(0)$; in mechanical modeling, $\Omega(0)$ is referred to as the *material domain*, since it stands for the undeformed configuration, and contains the 'particles' x of Ω at the state of rest.
- The deformed domain $\Omega(t)$, called *spatial domain* in mechanical modelling.
- The ALE domain $\Omega_R(t)$, or *reference domain*, which is the support of numerical computations.

The correspondance between the material and spatial domains $\Omega(0)$ and $\Omega(t)$ is described by the mapping:

$$\Omega(0) \ni x_0 \mapsto \varphi(t, x_0) = \chi(x_0, t, 0) \in \Omega(t)$$

where $t \mapsto \chi(x_0, t, 0)$ is the solution to (2.6) and represents the trajectory of a particle driven by the velocity $V(t,x)$, and lying at position x_0 at time $t = 0$ (see Section 2.2).

In turn, the motion of the ALE domain is described by the mappings

$$\Phi(t, \cdot) : \Omega_R(t) \to \Omega(0), \text{ and } \Psi(t, \cdot) : \Omega_R(t) \to \Omega(t),$$

which are explicitly given, once the moving strategy for the ALE domain has been set; see Fig. 14.

In practice, all the theoretical and numerical calculations are performed on the ALE domain and mesh; hence, quantities of interest attached to $\Omega(t)$ (normal vector, curvature, etc) must be transferred to $\Omega_R(t)$. Likewise, a partial differential equation posed on $\Omega(t)$ - for instance, in the model of Section 3, the bifluid Navier-Stokes equations - must be recast on the domain $\Omega_R(t)$: these are the so-called ALE formulations of these equations.

As far as ALE methods are concerned, we refer to the nice presentation [23] on this topic, and references therein for historical references and further discussions, or to the book [33].

9. Applications of the Level Set Method

Since its inception, the Level Set Method has become a reference method for capturing the motion of an evolving domain or interface, and it has found applications in diverse fields. In this section, we mention some (actually very few) of these applications.

9.1. **Applications in Computational Fluid Dynamics**

The Level Set Method appeared in Computational Fluid Dynamics with the study of the motion of two compressible gases, separated by a sharp interface [43]. Soon after, it was used in [61] for describing the interface between two immiscible, incompressible fluids, driven by the Navier-Stokes bifluid equations. Since these seminal works, it has been a popular way for describing boundaries of domains filled with fluids, and many improvements and extensions of the original techniques have come out; see for instance the review article [56] for a more in-depth discussion, and Fig. 15 below for an example.

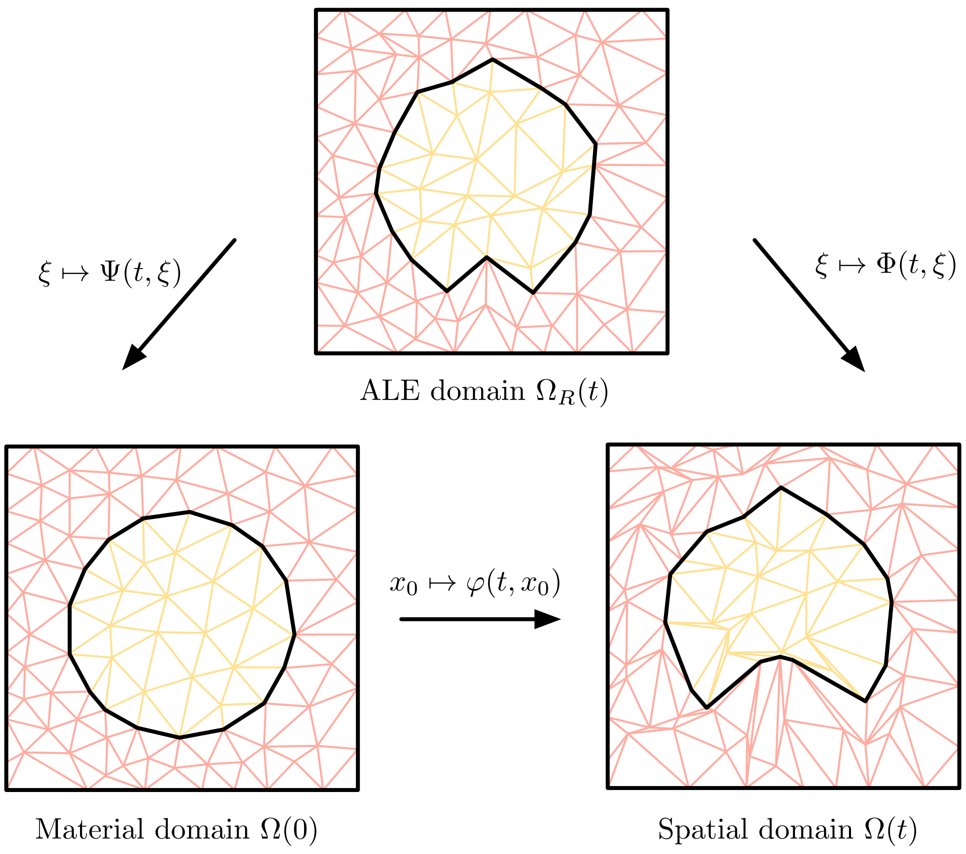
$\xi \mapsto \Psi(t,\xi)$
$\xi \mapsto \Phi(t,\xi)$
ALE domain $\Omega_R(t)$
$x_0 \mapsto \varphi(t,x_0)$
Material domain $\Omega(0)$
Spatial domain $\Omega(t)$

FIGURE 14. *The three domains and meshes involved in an Arbitrary Lagrangian-Eulerian procedure, and the mappings for passing from one to another.*

9.2. **Applications in shape optimization**

Shape optimization aims at finding the 'best' shape (or domain) $\Omega \subset \mathbb{R}^d$, i.e. that which minimizes a given functional $J(\Omega)$ depending on the domain itself. In the context where Ω stands for an elastic structure, on may imagine that $J(\Omega)$ is a measure of the stress within the structure, or (the opposite of) its fundamental frequency. If Ω is a pipe in which a fluid flows, $J(\Omega)$ may be for instance the dissipation of energy in the medium due to viscous effects.

Theoretical analyses make it possible to calculate explicitly a vector field V_Ω defined on the boundary Γ of Ω - a so-called *shape gradient* -, such that, the deformed shape $(\mathrm{Id} + V_\Omega)(\Omega)$, obtained by deforming Ω in the direction V_Ω gives rise to a new shape with a smaller value of J; see Fig. 16 for an illustration.

Accordingly, a possible strategy for the minimization of $J(\Omega)$ consists in starting from an initial domain $\Omega(0)$, then tracking the evolution of the domain $\Omega(t)$, driven by this shape gradient $V(t,x) = V_{\Omega(t)}(x)$.

The Level Set method was introduced to represent such shape optimization problems in the articles [5, 57, 65], and some application examples are reported on Fig. 17.

9.3. **Applications in image processing**

In the simplest mathematical models, a grey-scale image is represented by an intensity function $I : D \to \mathbb{R}$, where the working domain D is the unit square $(0,1)^2$.

One key issue in image processing is that of *segmentation*: the image I is generally composed of several areas with different contrasts, representing distinct objects. For instance, a medical image obtained by a process such as MRI identifies different tissues or organs by different intensities. Segmentation consists in capturing one of these particular regions, say $\Omega_T \subset D$, corresponding to a given intensity value I_0.

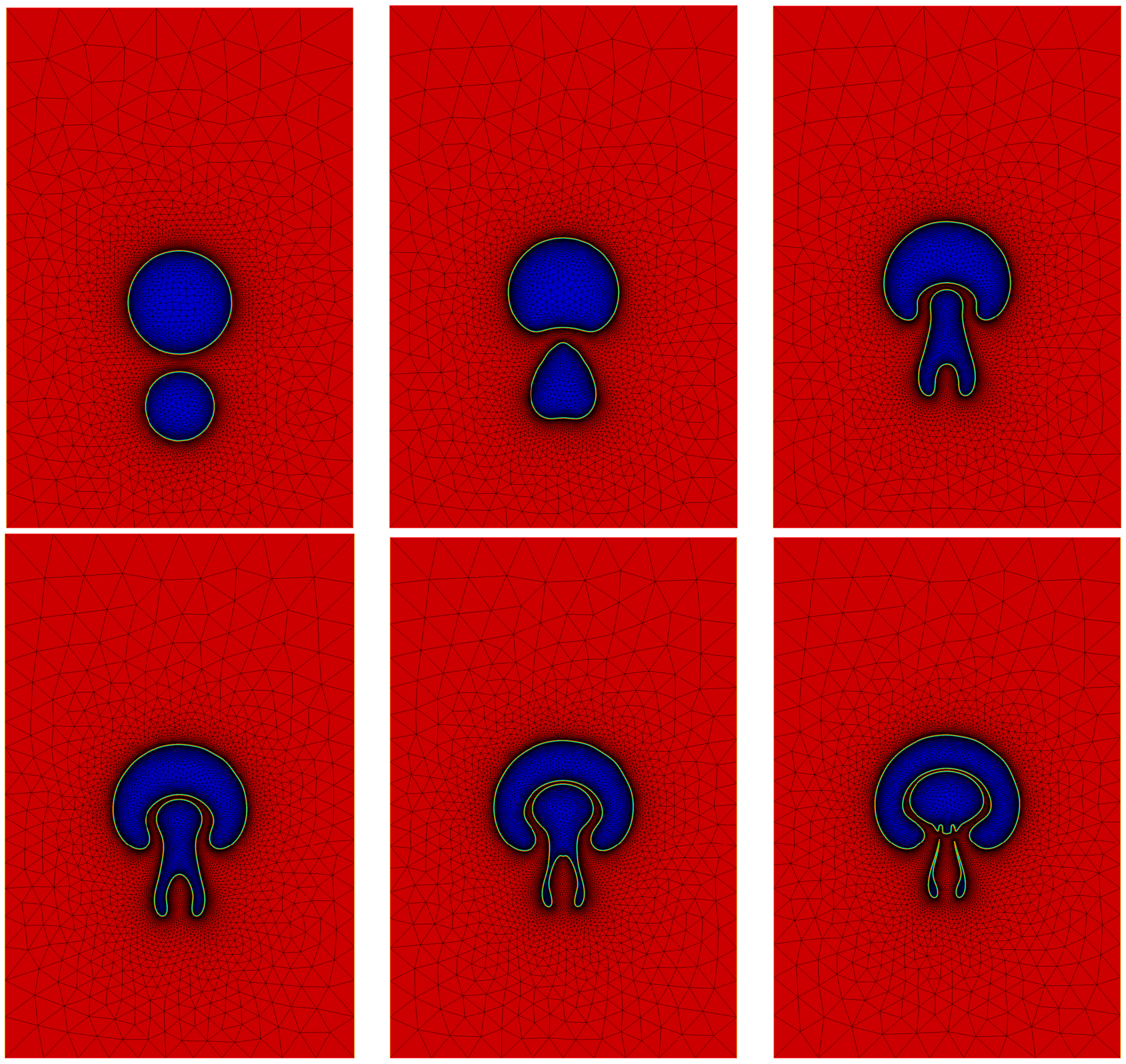

FIGURE 15. *Rise of two bubbles of a light fluid, immersed in a heavier one: interface at times (from left to right, top to bottom)* $t = 0, 0.1, 0.175, 0.2, 0.225,$ *and* 0.25.

One method to achieve this was introduced in [13, 40], as a variant of the *active contour model*: a domain $\Omega(t)$ evolves in time (from an arbitrary initial position $\Omega(0)$) according to a velocity field $V(t,x)$ of the form:

$$V(t,x) = (f_0(x) + f_1(\kappa_t(x)))n_t(x),$$

where f_0 and f_1 are two scalar functions with the following meanings: f_0 'attracts' the domain $\Omega(t)$ towards Ω_T, while $f_1(\kappa_t(x))$ only involves the mean curvature $\kappa_t(x)$ of $\partial\Omega(t)$ and compels $\Omega(t)$ to stay 'smooth enough'. In both aforementioned references, this evolution problem is numerically tackled with the Level Set Method.

Another major topic in image processing is *denoising*: in practice, the intensity function I is often plagued with noise, and it is desirable to replace it with another one, say $\widetilde{I}$, which is a smoothed version of I, but also retains the interfaces between the regions of different intensities. Several models have been proposed to give a mathematical flavour to this problem, among which the celebrated *Mumford-Shah* and *Rudin-Osher-Fatemi* (ROF) models. Both of them express the new image $\widetilde{I}$ as the minimizer of an energy functional where the

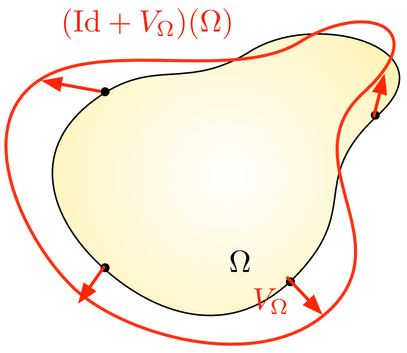

FIGURE 16. *The deformed version* $(\mathrm{Id} + V_\Omega)(\Omega)$ *of a shape* Ω *has a lower value of the performance criterion* $J(\Omega)$.

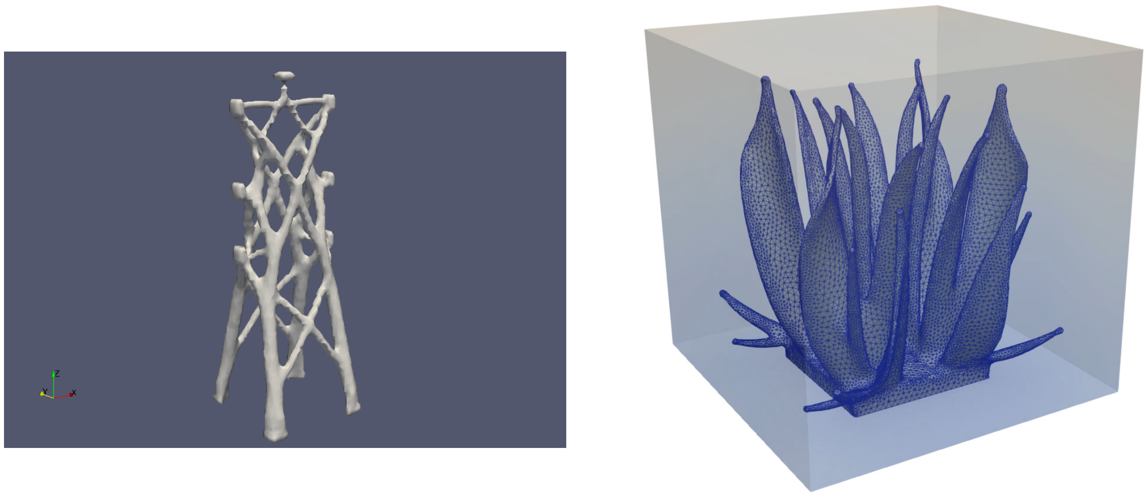

FIGURE 17. *(Left) Optimized shape of an electric pylon withstanding wind; (right) optimized shape of a heat exchanger device.*

contours of the regions with different intensities appear explicitly. Again, the Level Set Method is a natural ingredient in a numerical method dedicated to such problems.

Let us eventually mention the work [27], which uses the level set method to tackle the *stereo problem*, that is the problem of reconstructing a three-dimensional scene, from the data of several two-dimensional views.

More details around these issues, and many other examples of such problems in image processing may be found in the article [41] and in the monograph [47].

REFERENCES

[1] R. ABGRALL, *Numerical Discretization of the First-Order Hamilton-Jacobi Equation on Triangular Meshes*, Com. Pure Applied Math. XLIX, (1996) pp. 1339–1373.

[2] R. ABGRALL, H. BEAUGENDRE AND C. DOBZRYNSKI, *An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques*, J. Comput. Phys., 257 (2014) pp. 83–101.

[3] D. ADALSTEINSSON AND J.A. SETHIAN, *A fast Level Set Method for propagating interfaces*, J. Comput. Phys., 118 (1995) pp. 269–277.

[4] D. ADALSTEINSSON AND J.A. SETHIAN, *The Fast Construction of Extension Velocities in Level Set Methods*, J. Comput. Phys., 148 (1997) pp. 2–22.

[5] G. ALLAIRE AND F. JOUVE AND A.M. TOADER, *Structural optimization using shape sensitivity analysis and a level-set method*, J. Comput. Phys., 194 (2004) pp. 363–393.

[6] J.F. Aujol and G. Auber, *Signed distance functions and viscosity solutions of discontinuous Hamilton-Jacobi Equations*, INRIA Technical Report, 4507 (2002).

[7] V.E. Badalassi, H.D. Ceniceros and S. Banerjee, *Computation of multiphase systems with phase field models*, J. Comput. Phys., 190, (2003), pp. 371–397.

[8] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, 2nd printing, Modern Birkhäuser Classics, (2008).

[9] G. Barles, *Remarks on a flame propagation model*, INRIA: Technical Report, 451 (1985).

[10] T.J. Barth and J.A. Sethian, *Numerical Schemes for the HamiltonJacobi and Level Set Equations on Triangulated Domains*, J. Comput. Phys., 145 (1998) pp. 1–40.

[11] W. J. Boettinger, J. A.Warren, C. Beckermann and A. Karma,*Phase-Field Simulation of Solidification*, Annu. Rev. Fluid Mech., 32, (2002), pp. 163–194.

[12] P. Cardaliaguet and O. Ley, *Some flows in shape optimization*, Arch. Ration. Mech. Anal., 183, (2007), pp. 21–58.

[13] V. Caselles, F. Catté, B. Coll and F. Dibos, *A geometric model for edge detection*, Num. Mathematik, 66, (1993), pp. 1–31.

[14] L.-Q. Chen,*Phase-Field Models for Microstructure evolution*, Annu. Rev. Fluid Mech., 32, (2002), pp. 113–140.

[15] Y. G. Chen, Y. Giga, and S. Goto, *Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations*, J. Differential Geom., 33 (3), (1991), pp. 749–786.

[16] D. Chopp, *Computing minimal surfaces via level-set curvature flow*, J. Comput. Phys., 106, pp. 77-91 (1993).

[17] D. Chopp, *Another look at velocity extensions in the Level Set Method*, SIAM J. Sci. Comput., vol. 31, no. 5, (2009), pp. 3255–3273.

[18] D. L. Chopp and J. A. Sethian, *Flow under curvature: singularity formation, minimal surfaces, and geodesics*, Experiment. Math. 2, 4 (1993), pp. 235–255.

[19] M.G. Crandall, H.Ishii and P.L. Lions, *User's guide to viscosity solutions of second order partial differential equations*, Bulletin of the American Mathematical Society, 27 (1992), pp. 1–67.

[20] M. G. Crandall and P.-L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc. 277 (1983), pp. 1–42.

[21] M. G. Crandall and P.-L. Lions, *Two Approximations of Solutions of Hamilton-Jacobi Equations*, Math. Comput., 167, Vol. 43, (1984), pp. 1–19.

[22] E. Cristiani and M. Falcone, *Fast semi-Lagrangian schemes for the Eikonal equation and applications*, SIAM J. Numer. Anal. Vol. 45, No. 5, (2007) pp. 1979–2011.

[23] J. Donea, A. Huerta, J.-P. Ponthot and A. Rodríguez-Ferran, *Arbitrary Lagrangian-Eulerian Methods*, chap. 14 in Encyclopedia of Computational Mechanics, (2004).

[24] A. Dervieux and F. Thomasset, *A finite element method for the simulation of Raleigh-Taylor instability*, Springer Lect. Notes in Math., 771, (1979), pp. 145–158.

[25] L.C. Evans and J. Spruck, *Motion of level sets by mean curvature. I*, J. Differential Geom. Volume 33, Number 3 (1991), pp 635–681.

[26] M. Falcone and R. Ferretti, *Semi-Lagrangian Schemes for Hamilton-Jacobi Equations, Discrete Representation Formulae and Godunov Methods*, J. Comput. Phys., 175, (2002), pp. 559-575.

[27] O. Faugeras and R. Keriven, *Variational principles, surface evolution, pdes, level set methods and the stereo problem*, IEEE Transactions on Image Processing, 7, 3 (1998), pp. 336–344.

[28] P.J. Frey and P.-L. George, *Mesh Generation : Application to Finite Elements*, Wiley, 2nd Edition, (2008).

[29] Y. Giga, *Surface Evolution Equations, a Level Set Approach*, Monographs in Mathematics, 99. Birkhäuser, Basel-Boston-Berlin, (2006).

[30] J. Glimm, J. W. Grove, X. L. Li, K.-M. Shyue, Y. Zeng and Q. Zhang, *Three Dimensional Front Tracking*, SIAM J. Sci. Comput., 19, (1995), pp. 703–727.

[31] E. Harten and S. J. Osher, *Uniformly High-Order Accurate Nonoscillatory Schemes. I*, SIAM J. Numer. Anal., 24, 2, (1987), pp. 279–309.

[32] C.W. Hirt and B.D. Nichols, *?Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries*, J. Comput. Phys., 39, (1981), pp. 201-225.

[33] W. Huang and R. D. Russel, *Adaptive Moving Mesh Methods*, Applied Mathematical Sciences, Vol. 174, (2011).

[34] J. M. Hyman, *Numerical Methods for Tracking Interfaces*, Physica, 12D, (1984), pp.396–407.

[35] R. Kimmel and J.A. Sethian, *Computing geodesic paths on manifolds*, Proc. Natl. Acad. Sci. USA, vol.95 (1998), pp. 8431–8435.

[36] R.J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAM, (2007).

[37] X.L. Li, *Study of three dimensional Rayleigh-Taylor instability in compressible fluids through level set method and parallel computation*, Phys. Fluids A 5(1), 1904 (1993).

[38] S. Li and W.K. Liu, *Meshfree Particle Methods*, Springer, (2004).

[39] D.R. Lynch, *Unified approach to simulation on deforming elements with application to phase change problems*, J. Comput. Phys., 47, (1982) pp. 387–411.

[40] M.Malladi, J.A. Sethian and B.C. Vemuri, *A Fast Level Set based Algorithm for Topology-Independent Shape Modeling*, J. Math. Imaging and Vision, 6,2 (1996), pp. 269–290.

[41] M.Malladi and J.A. Sethian, *Level Set and Fast Marching Methods in Image Processing and Computer Vision*, Proceedings of the IEEE International Conference on Image Processing, (1996), pp. 489–492.

[42] R. Mittal and G. Iaccarino,*Immersed Boundary Methods*, Annu. Rev. Fluid Mech., 37, (2005), pp. 239–261.

[43] W. Mulder, S. Osher and J.A. Sethian, *Computing interface Motion in Compressible Gas Dynamics*, J. Comput. Phys., 100, (1992), pp. 209-228.

[44] S.J. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer Verlag, (2003).

[45] S.J. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.

[46] S.J. Osher and C.-W. Shu, *High order essentially non-oscillatory schemes for Hamilton-Jacobi equations*, SIAM J. Numer. Anal., 28, 4, (1991), pp. 907–922.

[47] S.J. Osher and N. Paragios (Eds.), *Geometric Level Set Methods in Imaging, Vision, and Graphics*, Springer-Verlag, (2003).

[48] C.S. Peskin, *Flow patterns around heart valves: a digital computer method for solving the equations of motion*, PhD thesis. Physiol., Albert Einstein Coll. Med., Univ. Microfilms. (1972).

[49] C.S. Peskin, *The Immersed Boundary Method*, Acta Numerica, (2002), pp. 479–517.

[50] E. Prados, O. Faugeras and E. Rouy, *Shape-from-Shading and Viscosity Solutions*, INRIA: Technical Report, 4638 (2002).

[51] W.J. Rider and D.B. Kothe, *Reconstructing Volume Tracking*, J. Comput. Phys., 141, (1998), pp. 112-152.

[52] R. Scardovelli and S. Zaleski,*Direct Numerical Simulation of Free-Surface and Interfacial Flow*, Annu. Rev. Fluid Mech., 31, (1999), pp. 567–603.

[53] J.A. Sethian, *An Analysis of Flame Propagation*, Ph.D. Dissertation, Lawrence Berkeley Laboratory, University of California, (1982).

[54] J.A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA Vol. 93, (1996), pp. 1591–1595.

[55] J.A. Sethian, *Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry,Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, (1999).

[56] J.A. Sethian and P. Smereka, *Level Set Methods for Fluid Interfaces*, Annual Review of Fluid Mechanics, 35 (2003), pp. 341–372.

[57] J. A. Sethian and A. Wiegmann, *Structural boundary design via level set and immersed interface methods*, J. Comput. Phys., 163, 2 (2000), pp. 489-528.

[58] C.-W. Shu, *Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, A. Quarteroni, Editor, Lecture Notes in Mathematics, CIME subseries, Springer-Verlag, ICASE Report 97-65 (1997).

[59] P. E. Souganidis, *Approximation Schemes for Viscosity Solutions of Hamilton-Jacobi Equations*, J. Differential Equations, 59, no. 1 (1985) pp. 1–43.

[60] J. Strain, *Semi-Lagrangian Methods for Level Set Equations*, J. Comput. Phys., 151 (1999) pp. 498–533.

[61] M. Sussman, P. Smereka and S. Osher, *A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow*, J. Comput. Phys., 114, 1 (1994), pp. 146-159.

[62] T.E. Tezduyar,*Finite element methods for flow problems with moving boundaries and interfaces*, Archives of Computational Methods in Engineering, vol. 8, (2001), pp 83–130.

[63] 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, pp. 708-759 (2001).

[64] L.A. Vese and T.F. Chan, *A multiphase level set framework for image segmentation using the Mumford and Shah model*, Int. J. Comput. Vision 50 (3) (2002), pp. 271–293.

[65] M.Y. Wang, X. Wang, D. Guo, *A level set method for structural topology optimization,* Comput. Methods Appl. Mech. Engrg., 192, 227–246 (2003).

[66] J.A. Warren and W.J. Boettinger, *Prediction of dendritic growth and microsegregation patterns in a binary alloy using the phase-field method*, Acta Metall., 43, (1995), pp. 689–703.

[67] H.-K. Zhao, *A Fast Sweeping Method for Eikonal Equations*, Math. Comp., 74 (2005), pp. 603–627.

[68] H.-K. Zhao, T. Chan, B. Merriman and S.J. Osher, *A Variational Level Set Approach to Multiphase Motion*, J. Comput. Phys., 127 (1996) pp. 179–195.