UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

DIPARTIMENTO DI SCIENZE TEORICHE E APPLICATE

PhD Course - XXXVII Cycle

Computer Science and Mathematics of Computation

High-order Semi-Lagrangian schemes and applications to Hamilton-Jacobi equations and Level Set Method

Advisor:
Prof. Matteo Semplice

Candidate:
Silvia Preda

Academic Year:
2023/2024

To my family

Abstract

We present numerical methods based on high-order semi-Lagrangian schemes coupled with Essentially Non-Oscillatory interpolation techniques, in the field of the Level Set Method and Hamilton-Jacobi equations. The first application arises in the context of surface reconstruction from point clouds, where we consider a variational formulation with a curvature constraint that minimizes a suitable functional in order to reconstruct an unknown surface from a set of unorganized points. The level set formulation of this model consists in solving an equivalent advection-diffusion equation that evolves until steady-state an initial surface described implicitly by a level set function, actually a signed distance one. In order not to be prohibitively limited by a parabolic-type time-stepping constraint, the semi-Lagrangian approach is employed, coupled with a multi-linear interpolator and a Weighted Essentially Non-oscillatory one to get a high-quality reconstruction. To concentrate the computational effort, this method is also presented in the framework of Adaptive Mesh Refinement based on octrees, since either the semi-Lagrangian approach and the Level Set Method are apt to be exploited for local refinement. In both cases, an extensive list of numerical tests, in two and three dimensions, is presented. On the other hand, the second application is devoted to high-order numerical schemes for time-dependent first-order Hamilton-Jacobi-Bellman equations. In particular, we propose to resort to a Central Weighted Non-Oscillatory interpolation and we prove a convergence result in the case of state- and time-independent Hamiltonians. Moreover, the expected computational advantages of the central reconstruction, compared to the traditional one, are validated by numerical simulations in one and two dimensions, also for more general state- and time-dependent Hamiltonians. Special attention is paid to the parallel implementation of the algorithms presented in this thesis, especially in the surface reconstruction application, where the three-dimensionality of the problem could make the computations extremely expensive.

Contents

Introduction

This PhD thesis finds its very first motivation in the problem of reconstructing surfaces from a set of unorganized points, which arises in a huge variety of contexts and applications. Essentially, every time a digital representation of a physical shape is not available, one possibility is to obtain a *point cloud* that samples its surface, and from which we can infer the original shape. Fields related to this topic include physics, computer graphics, medical imaging, urban planning and cultural heritage; especially in the latter, real artistic manufacts are obviously not Cartesian domains, instead they usually present complicated geometries and topologies, and their shape must be acquired from the real object itself, e.g. via 3d laser scanning or photogrammetry [117, 98, 99].

In its core, the problem of surface reconstruction can then be stated as follows: one is given a set of points $\mathcal{P} = \{Q_1, \dots, Q_N\}$ in a bounded region of $\mathbb{R}^3$ and the task is to derive from $\mathcal{P}$ a mathematical description of the original shape as a three-dimensional object whose boundary passes through or near the points in $\mathcal{P}$. As a matter of fact, the information carried by the point cloud only reveals the location of the points but does not say anything about the ordering or the connection between them, making the process of surface reconstruction complex and time-consuming. In addition, as an infinite number of surfaces may pass through or near the data points, this problem tuns out to be ill-posed, with no unique solution. The challenge is thus to get a good approximation of the data set that should have some smoothness properties while being able to retain as many details as possible. Moreover, one would like to have a reconstruction which is useful not only for static representations, but also for dynamic operations.

In particular, we are interested in the possibility to produce a high-quality description of complex objects, which can be later used as a domain definition in partial differential equations (PDEs) computations. A prototypical application of this topic in the field of cultural heritage has been presented in [37], where a reaction-diffusion model [39, 37] is applied to a reconstructed domain having the shape of a work of art in order to investigate the evolution of damage caused by the interaction between pollutants in the air and the work of art itself. More complex models including ablation [35] or swelling [36] of the material would require the computational domain to evolve in time.

In general, to represent a surface, one can follow two approaches: the explicit one and the implicit one. The former includes mesh-based techniques, such as Delaunay triangulations and Voronoi diagrams [48, 4, 32] and parametric techniques, e.g. NURBS [93, 60]. Despite its popularity, the main problem with this approach is to ensure a watertight reconstruction in cases when an evolution of the surface occurs. Recent works, for example [44], propose to overcome this difficulty by considering an additional mesh adjustment in order to control the quality of the mesh during the evolution. However, the state of the art agrees that explicit reconstructions are difficult to handle when dealing with moving boundaries or complex and changing

topologies.

On the other hand, using an implicit approach, one has the advantage of simpler handling topological flexibility, while having a representation of the objects that also allows Boolean operations to be easily performed. Among implicit methods, traditional ones enclose interpolation techniques to approximate the desired implicit function as a combination of some smooth basis functions. The problem here is of course related to the large linear system that need to be solved, resulting in high computational costs and, even worse, in ill-conditioned matrices when the number of interpolated points is very large. Radial Basis Functions (RBFs) with global and local support have found many applications along this line [29, 30, 118] and some least-squares-based methods [34] belong to this family, too. Other methods rely instead on local estimators in order to associate an oriented plane to each point in the cloud and hence recover the implicit representation [71]. A different and widespread approach for the processing of point cloud data is based instead on the application of the Level Set Method (LSM) [91]. This last line of research actually constitutes the starting point of this thesis.

For a broader overview about the reconstruction of real objects starting from point clouds, the reader can refer to [14, 13, 73]. Significant advancements in surface reconstruction are also being driven by the integration of machine learning techniques, especially deep learning models, which have shown remarkable results in learning implicit surface representations directly from unorganized point clouds [107, 82].

Since its introduction, the LSM has emerged as a powerful and versatile tool in a wide range of applications [112, 106, 90] which include image processing and surface reconstruction [124, 85]. The core of the method consists in representing both a d-dimensional object $\Omega(t) \subseteq \mathbb{R}^d$ and its boundary Γ(t), which are evolving in time, by a so-called *level set function* $\phi : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}$ such that $\Gamma(t) = \{\vec{x} \in \mathbb{R}^d : \phi(\vec{x}, t) = 0\}$ and the level set function $\phi(\vec{x}, t)$ has negative values inside $\Omega(t)$ and positive outside. The association between the interface $\Gamma(t)$ and the level set function $\phi(\vec{x}, t)$ makes moving the interface equivalent to updating ϕ, which can be done by solving a Hamilton-Jacobi (HJ) type equation of the form

$$\phi_t + V_n \, |\nabla \phi| = 0.$$

In the above equation, the evolution of ϕ is driven in the direction normal to its zero level set and the velocity V_n can be a functional of Γ itself and of possible other unknowns of the problem. The advantages of this capturing approach are well-known by now and we suggest [57, 102] for a recent survey and references. Specific applications to the computation of multiphase flows can be found in [122, 123], while unstable fronts are treated in [64, 63]. A comprehensive review on LSM and its applications can be found in [90].

In this thesis the LSM is applied to the problem of surface reconstruction following the model introduced by Zhao in [124] where an initial surface Γ^0 is evolved until steady-state minimizing a suitable energy functional. The level set formulation derived in [124] contains a hyperbolic term designed to drive $\Gamma(t)$ towards $\mathcal{P}$, but also a second-order parabolic curvature regularization term. This latter allows to trade the exact vanishing of ϕ on $\mathcal{P}$ for the maximum curvature of the resulting zero level set surface.

As a consequence, the numerical solution of this problem via explicit schemes would require to take into account the well-known stability constraints imposed by the Courant-Friedrichs-Levy (CFL) condition, which, given the presence of the parabolic term, would force us to a relation of type $\Delta t = \mathcal{O}(\Delta x^2)$ between the sizes Δx and Δt of the spatio-temporal discretization. One strategy to overcome this issue, while also remaining explicit, is to apply a semi-Lagrangian (SL) scheme for the numerical solution of the second-order level set equation. This will be in fact our choice.

First introduced in [41] for first-order systems of linear equations, SL schemes have been extended to HJ equations and to the treatment of parabolic terms with the main purpose of obtaining methods which are unconditionally stable with respect to the choice of the time step (see [54] for a comprehensive explanation).

In addition, the possibility to overcome the parabolic-type CFL restriction is very tempting in the LSM context since this method usually focuses just on what is happening in the vicinity of the zero level set of ϕ, thus one can naturally be prompted to employ Adaptive Mesh Refinement (AMR) techniques to refine the computational grid in this area. While the CFL condition will frustrate the adaptive approach when a general explicit scheme is employed, this will not be such an issue in the SL framework where one is allowed to work at large Courant numbers with a hyperbolic CFL restriction of type $\Delta t = \mathcal{O}(\Delta x)$.

In the context of LSM, a first application of SL schemes can be found in [110]. Specifically regarding our problem of surface reconstruction, we decide to employ a SL method following the scheme of [26], first presented in [50] for curvature-related equations, but resorting to a local interpolator, instead on a global one, for space reconstructions. In particular, we will employ both a simple multilinear interpolator and third-order Weighted Essentially Non-Oscillatory (WENO) techniques in order to improve the accuracy in the reconstruction. Regarding this application, also other techniques will be put together in other to localize the computational effort [94] and to keep our level set function close to the signed distance function (SDF), usually consisting in the application of a reinitialization procedure [94, 66, 67]. This choice is actually motivated by either the numerical accuracy of the scheme and the further usefulness of the mathematical description of the recovered shape. For instance, the use of the SDF is preferable when dealing with PDEs computations via ghost-cell methods [38, 58].

What had originally started as a surface reconstruction problem, then prompted also an investigation on the coupling of SL schemes with Essentially Non-Oscillatory (ENO) techniques of high-order accuracy. As a matter of fact, when increasing the order of the reconstruction, the unconditional stability that characterizes SL schemes is not trivially guaranteed since monotonicity might be lost. In fact, polynomial interpolation processes that are linear in the data are best avoided in non-smooth contexts due to Gibbs oscillations phenomena, while they can be replaced by ENO and WENO techniques for which some convergence results in the SL framework have been proven; see [56].

To explore this context we focus in particular on the numerical approximation of

the following hyperbolic Hamilton-Jacobi-Bellman (HBJ) equation:

$$\begin{cases} v_t(\vec{x},t) + H(\vec{x},t,\nabla v(\vec{x},t)) = 0, & \text{for } \vec{x},t \in \mathbb{R}^d \times (0,T), \\ v(\vec{x},0) = v_0(\vec{x}), & \text{for } \vec{x} \in \mathbb{R}^d, \end{cases}$$

where $v : \mathbb{R}^d \times (0,T) \to \mathbb{R}$, ∇v is the spatial gradient, $v_0 : \mathbb{R}^d \to \mathbb{R}$ is the initial data and $H : \mathbb{R}^d \times (0,T) \times \mathbb{R}^d \to \mathbb{R}$ is the Hamiltonian function, which will be assumed to be convex with respect to ∇v.

Numerous schemes have been proposed to approximate the solution of this HJB equation, but only a small number are aimed at high-order accuracy. High-order finite difference schemes based on ENO reconstruction, defined in [65], were proposed in [92] and extended to unstructured grids in [1]. Second-order Godunov-type schemes based on global projection operators are discussed in [81]. WENO schemes were introduced in [76, 83] and combined with finite difference schemes for HJ equations in [75]. A fifth-order central scheme based on WENO reconstruction has been developed in [18].

Concerning SL, in [53], a high-order semi-discrete SL scheme is proposed to discretize stationary HJB equations, while in [56], SL schemes are applied to evolutionary HJB equations. A high-order SL scheme for HJB equations is presented in [27] by combining the SL technique with WENO reconstructions.

We will develop our theory for the HJB equation related to a finite horizon optimal control problem, relying on the Dynamic Programming Principle to get the representation formula for $v(\vec{x},t)$, and thus applying a numerical scheme which involves a minimization procedure over the set of controls in order to approximate the solution, which requires, as a consequence, a high number of function evaluations to be performed in the scheme. In this respect, WENO reconstructions, especially in higher space dimensions, are not efficient, due to the dependence of the WENO linear weights on the reconstruction point.

A novel paradigm for non-oscillatory reconstruction operators has been introduced in [80], where the authors suggested to blend, in a WENO-like fashion, polynomials of different degrees, allowing to overcome some difficulties of non-existence, non-positivity and dependence on the reconstruction point of the WENO linear weights. The idea has been further developed into the so-called CWENO reconstruction and exploited in multiple spatial dimensions, also in the case of AMR and non-uniform grids [8, 128, 5, 126, 47]. The technique has also been exploited in finite difference schemes for HJ equations on Cartesian meshes via dimensional splitting [127, 125], as well as on general meshes [129]. General results for establishing the convergence order of a CWENO reconstruction have been presented in [43, 42, 104].

One of the main advantages of CWENO over the traditional WENO is that the central approach provides a reconstruction polynomial that is defined everywhere in the reconstruction cell and that can be later evaluated, with no essential extra cost, at many different reconstruction points. This is guaranteed by the independence of the linear weights from the reconstruction point. Furthermore, as shown in [103, 47, 31, 43], the CWENO approach allows to avoid the dimensional splitting procedure and this is advantageous also on Cartesian meshes when the number of reconstruction points per cell is high.

Relying on this feature, we want to exploit the positive features of CWENO to obtain a high-order SL scheme that is more efficient than the one in [27], which is based on WENO. Also, we prove a convergence result in the framework of [56] for the CWENO reconstruction.

Finally, we have also explored another important feature of the SL approach, namely the potential of this type of approach when employing local grid refinement, referring in particular to the LSM context. In [111] the idea of using tree-based grids for level set calculations was first introduced. Later, it was extended to fluid simulations in [95, 84] and to second-order accuracy in [87]. In particular, the use of adaptive tree-based grids in LSM context is quite advantageous because it gives a fine-grained control over errors in the vicinity of the interface, while also effectively reducing the dimensionality of the discrete problem by focusing most of the grid cells close to the interface. Also, the refinement process is naturally driven by the values of $|\phi|$.

However, even though the use of adaptive grids can dramatically reduce the computational cost, performing high-resolution calculations, especially in three dimensions, could still be prohibitively expensive, or even impossible on serial machine. Thus, in this case like in the Cartesian one, it is of paramount importance to employ an efficient implementation based on parallelization. In fact, parallel implementation constitutes also an important background aspect of most of the computational topics presented in this thesis and its features will be recalled when needed in the course of the dissertation.

Contributions

Summarizing, this thesis revolves around three main topics, which are LSM for surface reconstruction, high-order SL schemes and the application of AMR to LSM. The applications are mainly focused on surface reconstruction from point clouds and HJB equations related to finite horizon optimal control problems. The results obtained within the research presented here have been collected in the following papers:

- [37], *From Point Clouds to 3D Simulations of Marble Sulfation*, collects a preliminary version of the surface reconstruction method described in Chapters 1 and 4, applied to a PDE model simulation in the field of cultural heritage;
- [96], *Surface reconstruction from point cloud using a semi-Lagrangian scheme with local interpolator*, presents the complete workflow, as detailed in Chapter 4, of the surface reconstruction method relying on the WENO interpolator;
- [28], *A CWENO large time-step scheme for Hamilton-Jacobi equations*, is based on the topics presented in Chapters 2 and 3 and presents a convergence result and many tests to validate the computational aspects related to WENO and CWENO in a minimization procedure;
- the materials of Chapter 5 on AMR will be the object of another paper.

All the algorithms that we used in our work have been implemented in C++, relying on the PETSc library [7, 6] for Cartesian grid management, and on the P4EST library [19] for Quadree and Octree grid management. The above libraries, with the support of the Message Passing Interface (MPI) protocol, are also apt for handling the parallel implementation. The tests have been performed on the cluster Galileo 100 hosted at CINECA[1], exploiting the resources assigned to ISCRA-C Projects[2].

Organization

The thesis is organized as follows.

In Chapter 1 we introduce the Level Set Method of Osher and Sethian [91], mainly following the book [90] and focusing on the main ingredients used to deal with the evolution of level set functions: extension of the velocity, reinitialization and localization of the method.

In Chapter 2 we prepare the setting for the SL approximation schemes, starting by introducing first-order schemes for simple hyperbolic equations and then moving to high-order schemes and HJ equations. The convergence theory will be briefly recalled here for the case of WENO interpolator.

In Chapter 3 we present our work devoted to the numerical solution of HJB equations via a SL scheme coupled with CWENO interpolator. A proof for convergence will be given and the dissertation will be accompanied by an extensive number of numerical examples to validate the scheme and the computational advantages of using the CWENO interpolator, rather than the traditional WENO one.

Chapter 4 details our surface reconstruction method from point clouds. Although this was the starting point of our research, it is placed here since it will make use of most of the ingredients presented before. The mathematical model of [124] is recalled, together with its level set formulation. Also, we will recall all the auxiliary techniques needed to complete the algorithm and we will highlight the key points for its parallel implementation. Numerical tests in two and three dimensions will be presented in order to illustrate the performances of the method.

In Chapter 5 we conclude the thesis by presenting some preliminary results of the extension of the surface reconstruction method to AMR based on octrees, emphasizing the potential of the LSM coupled with the SL approach in the adaptive context.

---

[1] https://www.hpc.cineca.it/systems/hardware/galileo100/

[2] *Surface Reconstruction with Level Set Method* (HP10CPQ93M)
Parallel Scalability of Surface Reconstruction with Level Set Method (HP10COSEJL)
Adaptive Mesh Refinement in Level Set Methods (HP10C7HWOL)

Chapter 1

Level Set Method

In this chapter we introduce implicit surfaces and their representation via level set functions and in particular via the signed distance. We discuss the main advantages of this approach starting from the static case and then moving on to the case of evolving interfaces, for the treatment of which a powerful method, the Level Set Method (LSM), has been introduced in 1988 by Osher and Sethian [91]. Our dissertation will start with the general case of d dimensions and will be then focused on curves and surfaces, since our interest will be mainly devoted to modelling problems regarding the real case of two and three dimensional objects. For the different topics presented here we mainly refer to [90, 94, 110].

In what follows, we will indicate with $\vec{x}$ the points $\vec{x} \in \mathbb{R}^d$ and t will represent the time; we will however omit the arguments $\vec{x}$ and t when they can easily be deducted from the context. Unless otherwise specified, the norm used in $\mathbb{R}^d$ will be the Euclidean one, indicated with $|\vec{x}|$, $\vec{x} \in \mathbb{R}^d$. When deriving in space and making explicit the discretization schemes in two or three dimensions, we will use the subscript and superscript notations with indexes x, y, z in order to denote the spatial directions in two or three dimensions.

Finally, in Chapters 1 to 4, we mainly refer to a Cartesian uniform framework, thus the size of the grid will be indicated with Δx and the subscripts i, j, k, $(i, j, k) \in \mathbb{Z}^3$, will be used to index the nodal points of the computational grid. For the discretization in time we will consider a time step of uniform size Δt and denote with the superscript n the temporal evolution step such that, given an initial time t_0 and a final time T, we will denote $t_n = t_0 + n\Delta t$, for $n = 0 \dots N_T$, with $N_T = \lceil \frac{T}{\Delta t} \rceil$. Thus, following this notation, the value of a generic function $\alpha : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ in a point $(\vec{x}, t)$ indexed by (i, j, k) and n, will be denoted by $\alpha^n_{i,j,k}$. Nonetheless we point out that many steps of the algorithms presented in the following sections can be generalized to non-uniform grids by considering Δx to be the local mesh size and a suitable indexing of the nodes.

1.1 Implicit representation

Let us consider a closed region $\Omega \subset \mathbb{R}^d$, such that its boundary $\Gamma = \partial\Omega$ is an interface of codimension one which separates the interior and the exterior parts of the region Ω, denoted respectively with Ω^- and Ω^+. In this thesis we are interested only in the cases $d = 2, 3$, thus in two spatial dimensions our lower-dimensional interface will be a curve, while in three spatial dimensions it will represent a surface.

Instead of explicitly describing all the points that belong to Γ, or resort to a parametrization, an alternative way to distinguish between Ω^- and Ω^+, and describe the interface separating them, is to use a scalar continuous function $\phi : \mathbb{R}^d \to \mathbb{R}$

such that

$$\phi(\vec{x}) \begin{cases} < 0 & \text{for } \vec{x} \in \Omega^-, \\ = 0 & \text{for } \vec{x} \in \Gamma, \\ > 0 & \text{for } \vec{x} \in \Omega^+. \end{cases} \tag{1.1}$$

The interface Γ is thus implicitly defined by the zero level set $\{\vec{x} \in \mathbb{R}^d : \phi(\vec{x}) = 0\}$, and the interior and exterior parts of Ω are easily identified by the sign of ϕ. Such a function is what is usually called a *level set function*.

As a matter of fact, for a fixed interface Γ, there are infinite functions ϕ for which Γ corresponds to their zero level set, but a common choice of level set function is the signed distance function (SDF) to Γ, given by

$$\phi(\vec{x}) = \pm \min_{\vec{y} \in \Gamma} |\vec{x} - \vec{y}|, \tag{1.2}$$

where the sign in (1.2) is chosen accordingly to (1.1). In particular, the SDF is safer to use in numerical algorithms since its gradient $\nabla\phi$ and its zero level set can be readily computed numerically in a stable way.

On the other hand, at first glance, the implicit representation might seem wasteful since the implicit function ϕ is defined on all $\vec{x} \in \mathbb{R}^d$, while the interface has only dimension $d - 1$, but we will see that a number of very powerful tools are easily available when using this type of approach.

1.1.1 Geometric properties

From a geometrical point of view, many properties of Γ have simple expressions in terms of the level set function ϕ. This is because the gradient $\nabla\phi$ is perpendicular to the isocontours of ϕ and points in the direction of increasing ϕ. Also, the direction of the local unit normal $\vec{n}$ to the interface Γ can be recovered from the values of $\nabla\phi$ on the zero isocontour.

Normals Given the level set function ϕ, we can define a vector field $\vec{n} : \mathbb{R}^d \to \mathbb{R}^d$ on the entire domain by

$$\vec{n}(\vec{x}) = \frac{\nabla\phi(\vec{x})}{|\nabla\phi(\vec{x})|}, \tag{1.3}$$

which agrees with the outward unit normal to the interface for points $\vec{x}$ placed on Γ. Singular cases, of course, can occur and need to be treated properly. Nonetheless, if Γ is smooth enough, cases where the gradient vanishes or is undefined usually occur in regions of the domain which are further away from the interface, which are not in our interest when ϕ is used to represent the region Ω or its boundary Γ in a numerical method.

As an example, we can consider the simple two dimensional case of the signed distance function to the circle of radius 1 centered in $(0, 0)$, depicted in Figure 1.1. In this case $\phi(\vec{x}) = |\vec{x}| - 1$ and the gradient is defined everywhere, except for the origin where the tip of the reversed cone is placed. Usually, when one resorts to LSM, he is mainly interested in what happens in the vicinity of the circle, thus the singularity in the center should not pose any problem.

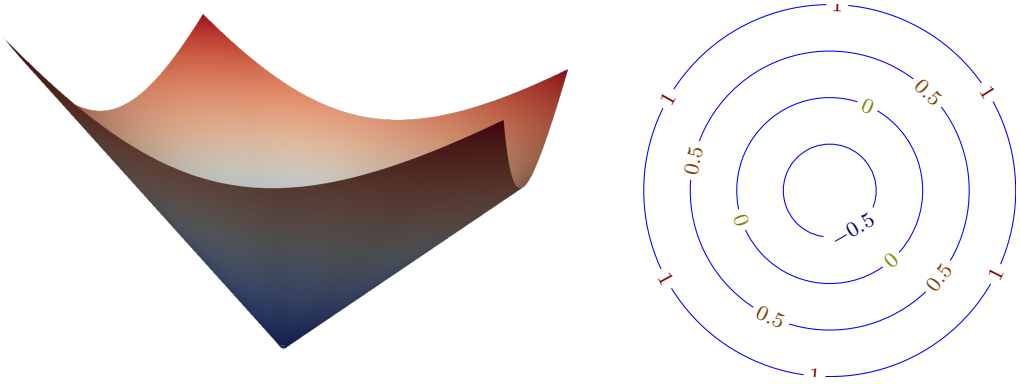

Figure 1.1: A two dimensional example of the level set function $\phi(\vec{x}) = |\vec{x}| - 1$ corresponding to the signed distance function associated to the circle of radius 1, centered in $(0, 0)$. The graph of ϕ is shown on the left, while a few isocontours are depicted on the right.

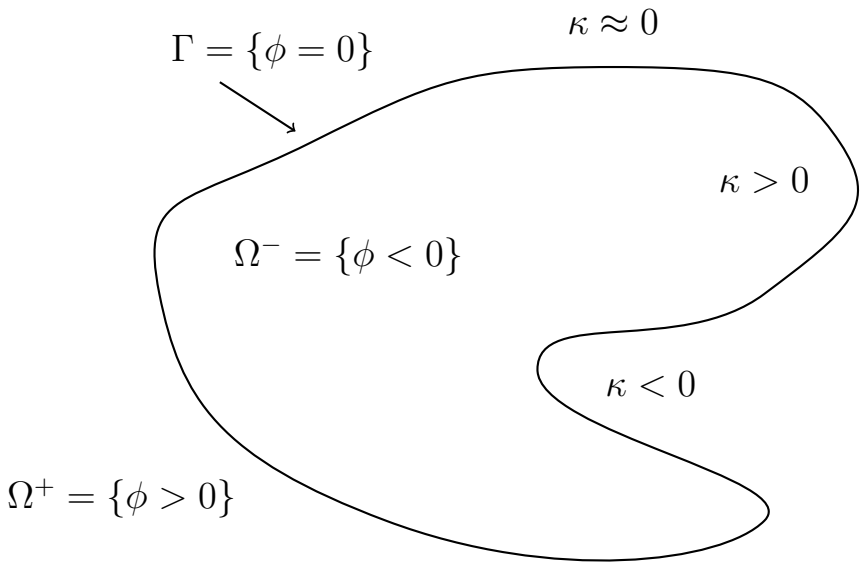

Figure 1.2: The interior and exterior regions of Ω are separated by $\Gamma \equiv \{\phi = 0\}$ and correspond to $\phi < 0$ and $\phi > 0$, respectively. The curvature κ is approximately zero where the curve Γ is becoming straight, while convex regions have $\kappa > 0$ and concave regions have $\kappa < 0$.

In this latter case, and more in general when dealing with SDFs, since $|\nabla\phi| = 1$, the function defining the normals (1.3) simply takes the form

$$\vec{n}(\vec{x}) = \nabla\phi(\vec{x}). \tag{1.4}$$

Furthermore, when ϕ is not a SDF but $\nabla\phi \approx 1$, the formula (1.3) can be computed numerically in a reliable and stable way, since the denominator will be well away from zero.

Curvature The mean curvature of the interface is defined as the divergence of the normals and, also in this case, except for singular cases, can be defined as a function $\kappa : \mathbb{R}^d \to \mathbb{R}$ on the whole domain by

$$\kappa(\vec{x}) = \nabla \cdot \frac{\nabla\phi(\vec{x})}{|\nabla\phi(\vec{x})|}, \tag{1.5}$$

so that $\kappa(\vec{x}) > 0$ for convex regions, $\kappa(\vec{x}) < 0$ for concave regions, and $\kappa(\vec{x}) = 0$ for a plane or a straight line; see Figure 1.2.

Also for the curvature, when ϕ is close to being a SDF the formula (1.5) is numerically stable, and, when ϕ is a SDF, the definition (1.5) takes the simpler form

$$\kappa(\vec{x}) = \nabla \cdot \nabla\phi(\vec{x}). \tag{1.6}$$

Global quantities Other global quantities like the Hausdorff measure $|\Gamma|$ of the interface Γ or the Lebesgue measure $|\Omega|$ of the region Ω can be computed in terms of ϕ. Their expressions read

$$|\Gamma| = \int \delta(\phi)\,|\nabla\phi|\,\mathrm{d}\vec{x}, \tag{1.7}$$

$$|\Omega| = \int H(-\phi)\,\mathrm{d}\vec{x}, \tag{1.8}$$

where $\delta(\phi)$ is the one dimensional δ-function and $H(\phi)$ is the one dimensional Heaviside function which takes the value 0 for $\phi < 0$ and 1 otherwise. In two dimensions, $|\Gamma|$ is simply the arclength of Γ and $|\Omega|$ is the area of Ω, while in three dimensions these quantities represent respectively the area of Γ and the volume of Ω.

1.1.2 Embedding in a larger domain

The key point when dealing with implicit representations is that we are changing our perspective allowing the interface Γ, and the region Ω, to be embedded in a larger domain, in general the space $\mathbb{R}^d$. In this way, it is first of all very easy to detect if a point $\vec{x} \in \mathbb{R}^d$ belongs to the interior or exterior part of the region Ω, by simply checking the sign of $\phi(\vec{x})$. Also, simple Boolean operations such as union of subdomains, their intersection and complement are readily reinterpreted in terms of level set functions resorting to minimum, maximum and change of sign operations.

Above all, the most important aspect is that we can choose to embed the domain Ω into a regular domain $\Omega' \supset \Omega$ on which we can apply common numerical techniques

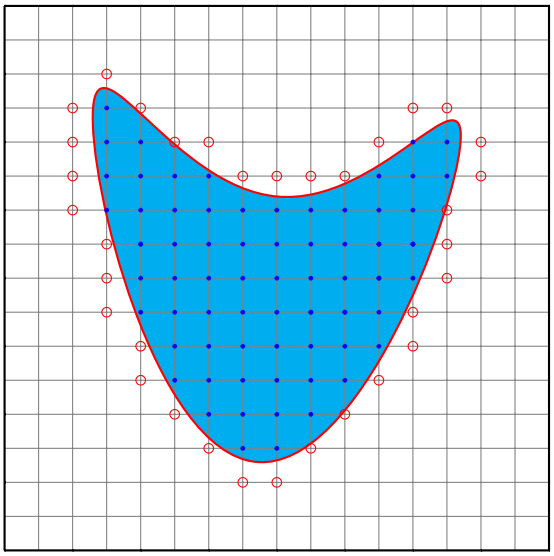

Figure 1.3: An arbitrary domain Ω (light blue area) is embedded in a square domain Ω' discretized by a Cartesian uniform mesh, where the level set function ϕ is given. The boundary Γ is represented by a thick red line. Internal nodal points are represented by blue dots, while ghost points are depicted with red empty circles. The first ones correspond to non-positive values of the level set function ϕ which identifies the domain Ω, the second ones are detected by both being first neighbours of some internal points and corresponding to positive values of ϕ.

based on uniform or, more in general, Cartesian meshes which are easier to handle, see Figure 1.3. The possible complex parametrization or explicit representation of Γ is replaced by the level set function associated to Γ and defined on Ω', and the preferred discretization can be used in the background to work with ϕ instead of Γ, directly. This is often a cheaper alternative to the explicit discretization of Ω. A wide variety of computational physics applications, including free surface flows, multiphase gas or gas-liquid interactions, fluid-solid interactions, moving boundaries in fluid dynamics, etc, are recollected in Part IV of the book [90]. An example in the area of cultural heritage can be found in [40, 39], where the authors, since they were dealing with complex and finely detailed domains, propose to completely avoid the problem of mesh generation by describing the computational domain via a level set function.

Furthermore, the level set approach is suitable for treating boundary conditions, for instance for PDEs computations via ghost-cell methods [38, 59]. See again Figure 1.3 in which the internal points are filled with blue color and the ghost ones are depicted with red empty circles. The difficulties in detecting these points are delegated to the structure of the mesh and to the stencil of the discretization of the differential operator at hand. In general, the ghost points are the points involved in the discretization of the PDE, which are not internal points. In the case of a $\mathcal{P}_1$ Finite Element discretization of an elliptic operator they would be the external vertices connected to internal ones by an edge. The simple case of a 2d domain with the usual 5-point finite-difference discretization of an elliptic operator is depicted in Figure 1.3. Once detected, the ghost points are the ones involved

in the computations of the boundary conditions and, since the projection on Γ will be needed, also these computations take advantages of the level set implicit representation; see [38, 59].

1.2 Moving interfaces

The importance of the implicit approach for interface representation is not just confined to the static case. In their formulation of the LSM [91], the authors highlighted the potential of level set functions for representing evolving boundaries or interfaces, rather than static ones, turning this technique to a powerful and versatile tool in a large variety of applications [90].

A moving front is now identified with a closed interface $\Gamma(t)$ in $\mathbb{R}^d$ with codimension one, enclosing a region $\Omega(t)$, to which we can associate a level set function of both space and time $\phi : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}$, which, similarly to (1.1), fulfills

$$\phi(\vec{x},t) \begin{cases} < 0 & \text{for } \vec{x} \in \Omega(t)^-, \\ = 0 & \text{for } \vec{x} \in \Gamma(t), \\ > 0 & \text{for } \vec{x} \in \Omega(t)^+, \end{cases} \tag{1.9}$$

where $\Omega(t)^-$ and $\Omega(t)^+$ are the interior and the exterior part of the region Ω at time t, respectively.

If we know the level set function ϕ, we can locate the interface by finding the zero level set of ϕ at each time t, that is $\Gamma(t) = \{\vec{x} \in \mathbb{R}^d : \phi(\vec{x},t) = 0\}$. Therefore, adding some dynamics to the interface Γ is equivalent to updating ϕ, which, as we will see, can be done by solving a PDE type equation for ϕ, namely the *level set equation*.

The motion of $\Gamma(t)$ can be prescribed a-priori or even be assigned in terms of the solution of another PDE (typically of fluid-dynamic or elasto-dynamic type) set in $\Omega^+(t)$ or $\Omega^-(t)$.

The power of this approach can be much appreciated especially in cases when $\Omega(t)$ and $\Gamma(t)$ undergo topological changes during their evolution: in this case it would be very cumbersome to represent them via a volume (resp. surface) mesh.

1.3 The level set equation

Suppose now that the velocity of each point on the interface $\Gamma(t)$ is given by $\vec{V}(\vec{x},t)$. Using a Lagrangian approach, one could move all the points $\vec{x}(t) \in \Gamma$ following their particle trajectory, thus solving the ordinary differential equation (ODE)

$$\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = \vec{V}(\vec{x},t). \tag{1.10}$$

However, this kind of approach usually suffers from connectivity changes and distortions that can quickly deteriorate the accuracy of the method. This is why the Lagrangian formulation is often coupled with numerical techniques that include

smoothing, regularization and surgery in order to prevent the solution from becoming completely inaccurate. These techniques, all together, are commonly referred to as *front tracking methods* and the interested reader is referred to [116, 114] for more details.

In order to avoid these repairing issues, in LSM one resorts to the implicit representation via the level set function ϕ both to represent the interface and to evolve it. Since by definition

$$\phi(\vec{x}(t), t) = 0, \tag{1.11}$$

for each particle trajectory $\vec{x}(t)$ lying on $\Gamma(t)$, differentiating (1.11) with respect to t, one gets the usually called the *linear level set equation*

$$\phi_t + \vec{V} \cdot \nabla\phi = 0, \tag{1.12}$$

where the t subscript denotes the partial derivative in the time variable t. Equation (1.12) is a PDE which describes the motion of the interface $\Gamma(t)$ corresponding to $\phi(\vec{x}, t) = 0$ in an Eulerian framework, rather than in a Lagrangian one. As usual, we couple (1.12) with an initial condition, namely we assume that an initial data $\phi^0(\vec{x}) = \phi(\vec{x}, 0)$, associated to the initial interface $\Gamma^0 = \Gamma(0)$, is given.

A non-linear version of the level set equation (1.12) can be obtained observing that the motion of the interface is dictated by the normal component of the velocity field only. Thus, splitting the velocity field $\vec{V}$ in (1.12) in its normal and tangential components, such that $\vec{V} = V_n\vec{n} + V_\tau\vec{\tau}$, one would prescribe the same evolution of the interface, where $\vec{\tau}$ is the direction tangential to the zero level set of ϕ. Since by definition $\vec{\tau} \cdot \nabla\phi = 0$, (1.12) can be rewritten as

$$\phi_t + V_n\, \vec{n} \cdot \nabla\phi = 0. \tag{1.13}$$

Furthermore, since

$$\vec{n} \cdot \nabla\phi = \frac{\nabla\phi}{|\nabla\phi|} \cdot \nabla\phi = \frac{|\nabla\phi|^2}{|\nabla\phi|} = |\nabla\phi|, \tag{1.14}$$

we can rewrite equation (1.13) as

$$\phi_t + V_n\, |\nabla\phi| = 0, \tag{1.15}$$

which is the non-linear version of (1.12). The form (1.15) can be used to evolve the level set function ϕ when one wants to prescribe V_n rather than $\vec{V}$.

Given the two fundamental equations (1.12) and (1.15), essentially the LSM consists in extending them to be valid throughout the domain and recovering the front at all times as the zero level set of ϕ. This approach of capturing, instead of tracking, interfaces by evolving the level set function, describes the topology of Γ via ϕ allowing merging, breaking and other topological changes to be handled automatically.

We point out here some issues of practical importance that will be further addressed later on. First, to extend the level set equation to the whole domain, we need to extend the velocity $\vec{V}$ or V_n, which is not always a trivial task. There exist of course some cases in which $\vec{V} : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^d$ and $V_n : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}$ are

already defined on the entire domain or are known on nodal values, but in general, an extension of these quantities is needed. Second, even if we take as initial data the SDF, the evolution governed by (1.12), (1.15) can produce steep gradients or flat areas that might affect the numerical accuracy of the solution. A procedure is thus needed to keep the level set function well behaved in the sense that, except for isolated points, the norm of the gradient of ϕ need to be bounded by some constants, namely

$$0 < b \leq |\nabla \phi| \leq B, \tag{1.16}$$

for some positive constants b and B. Such a procedure is commonly called *reinitialization*.

Also, using the function ϕ, which is defined globally on $\mathbb{R}^d$, or at least on the larger domain $\Omega' \supset \Omega$, instead of the smaller Ω or the $d-1$-dimensional Γ, might seem a computational overload. Nevertheless, this issue, as the above ones, can be mitigated by localizing the method and focusing our efforts just on a narrow band containing the front, with the bandwidth varying based on the numerical method involved. In particular, the updating of the level set function, the definition of the extended velocities $\vec{V}$, V_n, as well as the condition (1.16), can be all confined to a small neighbourhood of $\Gamma(t)$. From a computational point of view, this means that the computational cost is not $\mathcal{O}(N^d)$ but $\mathcal{O}(N^{d-1})$, for each evolution step on a discretized spatial grid of N points per dimension.

1.3.1 Types of evolution

In the previous section we have introduced the level set equation, in the forms (1.12) and (1.15), as the main ingredient for the evolution of ϕ. This evolution can be significantly different depending on the velocity field $\vec{V}$ which might be not only a function of the space variable $\vec{x}$ and of the time t, but also of the normal direction $\vec{n}$, the local mean curvature κ or a functional of the interface itself. Other cases include the ones in which the velocity depends on other unknowns of the problem, such as free boundary PDE-type models in which the velocity $\vec{V}$, or its projection V_n, depends on the values of the solution on the boundary at each time.

Linear transport Let us suppose that we are given a vector field $\vec{V}(\vec{x}, t)$ defined on $\mathbb{R}^d \times \mathbb{R}^+$, which does not depend on the interface itself. The resulting linear version of the level set equation is the hyperbolic PDE

$$\phi_t + \vec{V} \cdot \nabla \phi = 0. \tag{1.17}$$

For example, in two dimensions, if $\vec{V} = (2\pi(y - \hat{y}), -2\pi(x - \hat{x}))$, the above equation makes a closed curve rotate around the point $(\hat{x}, \hat{y})$. This type of problem occurs when modeling common and important physical situations such as transport, rotation, shearing and stretching in an ambient flow, and is conceptually the simplest to solve because the motion of each point on the interface obeys an ODE with known right-hand-side.

Mean curvature motion More complicated problems arise when the velocity field $\vec{V}$ depends explicitly on the level set function ϕ, $\vec{V} = \vec{V}(\vec{x}, t, \vec{n}, \kappa, ...)$. In such cases the interfacial geometry interacts with the motion, easily producing complex merging shapes and making these models popular in material science.

We consider here the case of mean curvature motion (MCM) in which the interface moves in the normal direction with a velocity proportional to its curvature

$$\vec{V} = -a\kappa\vec{n}, \tag{1.18}$$

where $a > 0$ is a constant and κ is the curvature. When $a > 0$, the interface moves in the direction on concavity, so that circles (in two dimensions) shrink to a single point and disappear. The effect of a curvature-driven flow is a smoothing effect due to the proportionality between mean curvature and velocity. Points with high-curvature move faster than points with lower curvature and the velocity vanishes in flat areas.

The velocity term (1.18) contains a component in the normal direction only, thus from (1.15), using $V_n = -a\kappa$ and substituting the expression for the curvature (1.5), we obtain the model equation for the MCM which is a second-order evolutive HJ equation given by

$$\phi_t = a\,\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right)|\nabla\phi|. \tag{1.19}$$

The right-hand-side term in (1.19) is a parabolic term that imposes a strong time step restriction when solving numerically the MCM equation with an explicit scheme. In fact, if we consider a space discretization of size Δx and a time step Δt, the Courant-Friedrichs-Levy (CFL) condition requires the inequality $\Delta t = \mathcal{O}(\Delta x^2)$ to be satisfied as a necessary condition for the convergence of the scheme, making the numerical computations prohibitively expensive. This time step restriction can be eliminated enforcing the inclusion of the physical domain of dependence into the numerical one, for example by allowing unbounded stencils, or, as we will see later, by shifting them. As a matter of fact, one could also involve implicit time-stepping schemes as they are the usual remedy to overcome the time step restriction. Unfortunately, this approach is often unavailable for level set equations because the complex and problem-dependent relation between V_n and $\Gamma(t)$ frustrates most non-linear equation solvers.

Convection-diffusion equation The convection-diffusion equation mixes both the effects of an advective velocity field and a diffusive term. For the level set function ϕ we have

$$\phi_t + \vec{V} \cdot \nabla\phi = a\kappa|\nabla\phi| \tag{1.20}$$

or equivalently

$$\phi_t + \vec{V} \cdot \nabla\phi = a\,\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right)|\nabla\phi|. \tag{1.21}$$

Some interesting theoretical remarks can be done for the case when ϕ is a SDF, thus $a\kappa|\nabla\phi|$ is identical to $a\Delta\phi$ and (1.20) can be rewritten as

$$\phi_t + \vec{V} \cdot \nabla\phi = a\Delta\phi. \tag{1.22}$$

We point out that this equivalence between equations (1.20) and (1.22) is lost if the level set function ϕ is deformed during the evolution, becoming no longer a SDF. However, replacing (1.20) with (1.22) is still allowed if a reinitialization procedure is performed at each time in order to restore the condition $|\nabla\phi| = 1$.

Despite the apparent simplification, Osher and Sethian pointed out in [91] that equation (1.20) is a more natural choice than (1.22) for dealing with LSM. In fact, supposing that the $\mathcal{O}(1)$ size a term is replaced with an $\mathcal{O}(\Delta x)$ size ϵ term that vanishes as the mesh is refined with $\Delta x \to 0$. Then equation (1.22) becomes

$$\phi_t + \vec{V} \cdot \nabla\phi = \epsilon\Delta\phi, \tag{1.23}$$

which is the same of (1.17) with the addition of an artificial viscosity term $\epsilon\Delta\phi$. This vanishing viscosity picks out the physically correct weak solution of the hyperbolic equation (1.17) when no classical solution exists. In [105], Sethian suggested an entropy condition that required curves to flow into corners, and he provided numerical evidence to show that this entropy condition produced the correct weak solution for self-intersecting curves. Sethian's entropy condition indicates that $\epsilon\kappa|\nabla\phi|$ is a better form for the vanishing viscosity than $\epsilon\Delta\phi$ for dealing with the evolution of lower-dimensional interfaces and this is why we will resort to equation (1.21) as a model for the convection-diffusion of level set functions.

Free boundary problems Another important category of problems that can be treated in a level set framework is free boundary problems. In moving interface problems for PDEs, the interfacial velocity depends on additional fields satisfying algebraic, ordinary differential, partial differential or integral equations on or off the interface. As an example, in volume diffusion [89, 46] the velocity reads

$$\vec{V} = \frac{\partial u(\vec{x}, t)}{\partial \vec{n}}, \tag{1.24}$$

where $\vec{n}$ is the outward normal direction to the interface $\Gamma(t)$, and $u(\vec{x}, t)$ is the solution of the Laplace equation $\Delta u = 0$ outside $\Gamma(t)$ with boundary conditions given on the interface and at ∞. Since the boundary is not fixed in time, we need to track it in order to find the solution u of the Laplace equation, but in the meanwhile the position of the front, and thus the domain of the problem, depends on the solution u itself.

1.4 Extension of a quantity off the interface

In LSM we need the velocity $\vec{V}$ or the normal velocity V_n to be defined in the whole domain or at least in a neighbourhood of the front $\Gamma(t)$. In some applications this velocity is naturally prescribed globally, or at least in all nodal points of the discretization, as it will be for instance in our application to surface reconstruction, while in other applications V_n is naturally prescribed only on $\Gamma(t)$ and we need a suitable method to extend it. In these cases, the usual approach is to extend a quantity q, known a-priori on the interface $\Gamma(t)$, as a constant function along curves

normal to $\Gamma(t)$. This suggests the following hyperbolic PDE,

$$q_\tau + S(\phi)\frac{\nabla\phi}{|\nabla\phi|}\cdot\nabla q = 0, \tag{1.25}$$

whose characteristic curves are normal to the level sets of ϕ and pointing away from $\Gamma(t)$. In (1.25) $S(\phi)$ is the sign function of ϕ defined as

$$S(\phi) = \begin{cases} -1 & \text{if } \phi < 0, \\ 0 & \text{if } \phi = 0, \\ +1 & \text{if } \phi < 0, \end{cases} \tag{1.26}$$

τ is a pseudo-time and the subscript denotes a partial derivative with respect to this pseudo-time variable.

Equation (1.25) is a particular case of the HJ-type equation

$$q_t + H(\nabla q, \vec{x}, t) = 0, \quad \vec{x}\in\mathbb{R}^d, t > 0, \tag{1.27}$$

for which there exist accurate and robust numerical schemes to approximate the solution, even more so when the quantity q is not required to be extended for a significant distance (see [75, 92]).

For the sake of clarity, a numerical solution of (1.25) can be computed resorting for instance to a first-order upwind scheme coupled with a forward Euler time integration, which in two dimensions translates into the updating formula

$$\begin{aligned} q_{i,j}^{\nu+1} = q_{i,j}^{\nu} - \Delta\tau\Big[&(s_{i,j}n_{i,j}^x)^+\frac{q_{i,j}^\nu - q_{i-1,j}^\nu}{\Delta x} + (s_{i,j}n_{i,j}^x)^-\frac{q_{i+1,j}^\nu - q_{i,j}^\nu}{\Delta x} \\ &+(s_{i,j}n_{i,j}^y)^+\frac{q_{i,j}^\nu - q_{i,j-1}^\nu}{\Delta y} + (s_{i,j}n_{i,j}^y)^-\frac{q_{i,j+1}^\nu - q_{i,j}^\nu}{\Delta y}\Big], \end{aligned} \tag{1.28}$$

where the subscripts $i, j,\ i\pm 1, j\pm 1$ denote the nodal values of the quantities involved, the index ν is used to discretize the pseudo-time τ, and $(x)^+ = \max(x, 0)$, $(x)^- = \min(x, 0)$ for a scalar x. Furthermore, $n_{i,j}^x$ and $n_{i,j}^y$ denote the components of the outward unit normal $\vec{n}$ given by the vector $(n^x, n^y) = (\partial_x\phi/|\nabla\phi|, \partial_y\phi/|\nabla\phi|)$ and $s_{i,j}$ denotes the nodal value of $S_\epsilon(\phi)$, where $S_\epsilon(\phi)$ is an approximation of $S(\phi)$ given by

$$S_\epsilon(\phi) = \frac{\phi}{\sqrt{\phi^2+\epsilon^2}} \tag{1.29}$$

and ϵ is a small smoothing parameter which can be taken equal to Δx. Unless otherwise specified, the derivatives are computed by central differencing. Finally, we point out that the scheme (1.28) does not need any internal boundary condition since the characteristics of the PDE (1.25) flow out of the interface Γ and, numerically, the upwind approach only uses the values of ϕ on the nodes biased to Γ.

1.5 Reinitialization

We have already mentioned above the advantages of working with a signed distance function ϕ, both from a geometrical and a numerical point of view. The delicate

issue when dealing with LSM is thus to guarantee that the level set function stays well-behaved during its evolution in the sense of (1.16). For general $\vec{V}$ or V_n, flat and/or steep regions will typically develop at the interface, making further computations and contour plotting highly inaccurate. Therefore, the common practice is to periodically perform a reinitialization step that will eliminate such problems without the explicit knowledge of their location, resetting the level set function ϕ to be a signed distance function in the whole domain or at least in the vicinity of the front, while keeping its zero level set, i.e. the front itself, unchanged.

A first natural idea for such a procedure can be based on first, finding the location of the front $\Gamma(t)$ with some interpolation technique and then, computing the signed distance function to this front [86]. The main problem with this technique is that it usually suffers from its high cost and the likelihood of introducing some spurious irregularities into the data which might require some other smoothing procedure to be coupled with this approach.

A more common and elegant way for reinitializing ϕ was introduced in [112] and resorts again to a HJ-type equation given by

$$\begin{cases} \phi_\tau + S(\tilde{\phi})(|\nabla\phi| - 1) = 0, \\ \phi(\vec{x}, 0) = \phi_0(\vec{x}) = \tilde{\phi}(\vec{x}, t). \end{cases} \tag{1.30}$$

The equation (1.30) is solved to steady state in order to find the desired signed distance function ϕ from the initial data represented by the solution $\tilde{\phi}$ of the level set equation (1.12), (1.15) at a fixed time t, possibly perturbed by the evolution itself. In (1.30) τ is a pseudo-time and the subscript denotes a partial derivative with respect to this pseudo-time variable. The function S is the sign function that is usually approximated by a function S_ϵ which introduces some smoothing effect as in (1.29).

This method works well when $\tilde{\phi}$ is neither too flat nor too step near the interface and, when properly implemented, converges quickly in a neighbourhood of the interface itself. When $\tilde{\phi}$ becomes too flat of too steep, some issues can arise. In the first case the quantity $S_\epsilon(\tilde{\phi})$ is small and the propagation is slowed down, while in the second case things can be worse because ϕ might change sign, thus moving the interface across grid points, which has to be avoided. In [94], the authors give a proper analysis and propose to solve this issue by approximating the nodal values of the function S in the computation with

$$s_{i,j} = \frac{\tilde{\phi}_{i,j}}{\sqrt{\tilde{\phi}_{i,j}^2 + \left|D\tilde{\phi}_{i,j}\right|^2 \Delta x^2}}, \tag{1.31}$$

where $D\tilde{\phi}$ is some discrete operator for $\nabla\tilde{\phi}$ and Δx the size of the spatial discretization involved. This choice of approximation for $S(\tilde{\phi})$ by (1.31) avoids changing the sign of the nodal values of ϕ in the reinitialization step when the initial data is steep and speeds up the convergence when it is flat at the interface.

In what follows we will recall some usual choices to numerically solve (1.30) which are widely used in literature. The most canonical ones are based on an upwind monotone discretization, while the successive one is a constrained version of the

previous. Essentially, in this latter version, an explicit and direct step is applied to reinitialize $\tilde{\phi}$ in the immediate vicinity of the interface, with the aim of better preserving the position of Γ. In Chapter 5 a different technique for adaptive grids will be presented.

1.5.1 Canonical upwind schemes

The first scheme for solving (1.30) have been suggested in [91] and in two dimensions, on a uniform Cartesian grid of size Δx, is described by

$$\begin{aligned}\phi_{i,j}^{\nu+1} = \phi_{i,j}^{\nu} &- \frac{\Delta\tau}{\Delta x} s_{i,j}^{+} \left(\sqrt{(a^{+})^2 + (b^{-})^2 + (c^{+})^2 + (d^{-})^2} - 1 \right) \\ &- \frac{\Delta\tau}{\Delta x} s_{i,j}^{-} \left(\sqrt{(a^{-})^2 + (b^{+})^2 + (c^{-})^2 + (d^{+})^2} - 1 \right)\end{aligned} \tag{1.32}$$

where $s_{i,j}$ is the approximation to $S(\tilde{\phi}(x_{i,j}, t^n)$ with (1.31), $(x)^+ = \max(x, 0)$, $(x)^- = \min(x, 0)$ for a scalar x, and a, b, c, d are defined by

$$\begin{aligned} a &= D_x^{-}\phi_{i,j}^{\nu}, \quad b = D_x^{+}\phi_{i,j}^{\nu}, \\ c &= D_y^{-}\phi_{i,j}^{\nu}, \quad d = D_y^{+}\phi_{i,j}^{\nu}. \end{aligned} \tag{1.33}$$

The scheme (1.32) turns out to be monotone under the condition

$$\frac{\Delta\tau}{\Delta x} |s_{i,j}| \leq \frac{1}{2}. \tag{1.34}$$

The second example is Godunov's scheme

$$\begin{aligned}\phi_{i,j}^{\nu+1} = \phi_{i,j}^{\nu} &- \frac{\Delta\tau}{\Delta x} s_{i,j}^{+} \left(\sqrt{\max[(a^{+})^2, (b^{-})^2] + \min[(c^{+})^2, (d^{-})^2]} - 1 \right) \\ &- \frac{\Delta\tau}{\Delta x} s_{i,j}^{-} \left(\sqrt{\max[(a^{-})^2, (b^{+})^2] + \min[(c^{-})^2, (d^{+})^2]} - 1 \right)\end{aligned} \tag{1.35}$$

described in [92, 11], which is also monotone under the condition (1.34) and where a, b, c, d are still defined by (1.33).

In both cases a more accurate TVD-type Runge-Kutta scheme can be used, instead of the forward Euler time discretiziation, coupled with ENO or WENO techniques for the computation of the one-sided differences $D_x^{\pm}\phi_{i,j}$, $D_y^{\pm}\phi_{i,j}$. More about ENO, WENO and CWENO schemes for Hamilton-Jacobi equations will be detailed in Chapter 2 and 3. For now we just emphasize the fact that these Essentially non-Oscillatory techniques, first introduced in [65] in their ENO original version and then developed in [76, 83, 43, 5] in their weighted and weighted central versions, has been extended with proper modifications to Hamilton-Jacobi type equations [125, 128, 127] in which we seek for solutions that are, in the worst case, continuous with kinks.

1.5.2 The constrained reinitialization scheme

Eq. (1.30) has been proved to yield, for $\tau \to \infty$, the unique viscosity solution of the Eikonal equation $\left|\nabla\tilde{\phi}\right| = 1$ that corrects the perturbed level set function $\tilde{\phi}$ to

become a signed distance function, while also keeping the zero level set invariant because $S(\tilde{\phi}) = 0$. Nevertheless, it has been emphasized by several authors [113, 100] that solving the discretized version of (1.30) considerably displaces the zero level set and thus may lead to substantial errors due to the reinitialization.

More recent techniques have been introduced in [66, 67] by Hartmann et al. with the aim of preventing this kind of displacement, which is indeed the major difficulty in PDE based methods for reinitialization. In fact, the different formulation is derived by explicitly imposing the zero-displacement constraint on the zero level set and this is why we will refer to this scheme as the *constrained reinitialization scheme*.

The idea behind this method is to achieve the least deviation from the fundamental criteria which a reinitialization method should satisfy: the condition on the gradient $|\nabla\phi| = 1$, which is relevant for all the grid points in which we want to reinitialize, and the invariance of the zero level set, which is required only for the points immediately close to the interface Γ. Furthermore, as a starting point for their new formulations, the authors in [66, 67] resort to the already modified reinitialization scheme proposed by Russo and Smereka [100], which makes a distinction between the points nearby Γ and the distant ones, and reads

$$\phi_{i,j}^{\nu+1} = \begin{cases} \phi_{i,j}^{\nu} - \frac{\Delta\tau}{\Delta x}\left(S(\tilde{\phi}_{i,j})\left|\phi_{i,j}^{\nu}\right| - d_{i,j}\right) & \text{for } x_{i,j} \text{ nearby } \Gamma, \\ \phi_{i,j}^{\nu} - \Delta\tau S(\tilde{\phi}_{i,j})\left(\left|\nabla\phi_{i,j}^{\nu}\right| - 1\right) & \text{otherwise,} \end{cases} \tag{1.36}$$

where S is the sign function, $\tilde{\phi} = \phi(\vec{x}, 0)$ is the initial data for the problem (1.30) and

$$d_{i,j} = \frac{\tilde{\phi}_{i,j}}{\sqrt{[D_x\tilde{\phi}_{i,j}]^2 + [D_y\tilde{\phi}_{i,j}]^2}} \tag{1.37}$$

is the target value of the level set function for the point $x_{i,j}$ nearby Γ.

We briefly recall the scheme by Hartmann et al. since it is one of the scheme we have used in our applications and also to give the reader a vertex-centred description of it which is consistent with the rest of the thesis.

Let us consider the set $\mathcal{X}$ of the points close to Γ, which is defined by

$$\mathcal{X} = \left\{ x_{i,j} : \left(\prod_{i',j}^{i,j} \tilde{\phi} \le 0\right) \vee \left(\prod_{i,j'}^{i,j} \tilde{\phi} \le 0\right) \right\}, \tag{1.38}$$

where $i' \in \{i-1, i+1\}$, $j' \in \{j-1, j+1\}$ and $\prod_{i',j}^{i,j} \tilde{\phi} = \tilde{\phi}_{i,j}\tilde{\phi}_{i',j}$. Considering a uniform Cartesian mesh of size Δx, (1.38) states that $\mathcal{X}$ contains all the points which are at a maximum distance of Δx from Γ.

Now, focusing on a grid point $x_{i,j} \in \mathcal{X}$, in two dimensions, and using a linear interpolator to detect the location of Γ, the errors of a reinitialization scheme can be written as

$$\begin{cases} \sigma_{i,j}^{0} = \sqrt{[\partial_x\phi_{i,j}]^2 + [\partial_y\phi_{i,j}]^2} - 1, \\ \sigma_{i,j}^{\alpha} = \phi_{i,j} - r_{i,j}^{\alpha}\phi_{i,j}^{\alpha}, \end{cases} \tag{1.39}$$

where the superscript α denotes quantities computed using the points $x_{i,j}^{\alpha} \in \mathcal{N}_{i,j}$, $\mathcal{N}_{i,j}$ being the set of the neighbouring points of $x_{i,j}$ across the zero set Γ, namely

$\mathcal{N}_{i.j} = \{x^{\alpha}_{i,j} : \phi_{i,j}\phi^{\alpha}_{i,j} < 0\}$ and, by linear interpolation, we have

$$r^{\alpha}_{i,j} = \frac{\tilde{\phi}_{i,j}}{\tilde{\phi}^{\alpha}_{i,j}}. \tag{1.40}$$

Note that in (1.38), (1.39), (1.40) the notations ϕ and $\tilde{\phi}$ are used, as usual, to denote the reinitializing level set function and the perturbed one, that is the initial data for the reinitialization problem (1.30).

Summing up all the squared errors weighted by some quantities $\theta^{\alpha'}$, one gets the least-squares function Θ for each point in $\mathcal{X}$

$$\Theta_{i,j} = \sum_{\alpha'=0}^{M_{i,j}} \theta^{\alpha'}(\sigma^{\alpha'}_{i,j})^2, \tag{1.41}$$

where $M_{i,j}$ is the number of elements in the set $\mathcal{N}_{i,j}$. The problem is thus translated into a minimization problem for the function Θ, which is faced by differentiating (1.41) with respect to the $M_{i,j} + 1$ unknowns, setting all the derivatives to 0 and solving the linear system obtained. For instance, when we differentiate with respect to $\phi_{i,j}$ we get

$$\partial_{\phi_{i,j}}\Theta_{i,j} = \partial_{\phi_{i,j}}\theta^0(\sigma^0_{i,j})^2 + 2\left(\sum_{\alpha'=1}^{M_{i,j}} \theta^{\alpha'}\sigma^{\alpha'}_{i,j}\right) = 0. \tag{1.42}$$

Since the derivative of $(\sigma^0_{i,j})^2$ introduces some non-linearity, the key idea is to allow the level set function to be directly reinitialized via equation (1.37) in $\mathcal{X}$, so that $\sigma^0_{i,j}$ may be assumed very small and the minimization problem for (1.41) can be considered with weights

$$\theta^{\alpha'} = \begin{cases} 0 & \text{if } \alpha' = 0, \\ 1 & \text{if } \alpha' \geq 1. \end{cases} \tag{1.43}$$

The weights (1.43) introduced in Eq. (1.42) leads to solve the problem (1.42)

$$\phi_{i,j} = \frac{1}{M_{i,j}} \sum_{\alpha=1}^{M_{i,j}} r^{\alpha}_{i,j}\phi^{\alpha}_{i,j}, \tag{1.44}$$

which can be taken directly as the final reinitialized value or can be put into the scheme (1.36) by substituting the target function $d_{i,j}$ with the value computed in (1.44).

In practice, the scheme designed by Hartmann et al. is a 2-step correction scheme which in the first step computes the signed distance function in the points in $\mathcal{N}_{i,j}$ using (1.37) and in the second step uses (1.44) to compute $\tilde{d}_{i,j}$ on all the remaining points in $\mathcal{X}$. Both steps are performed only once and sequentially, such that the $\phi^{\alpha}_{i,j}$ required in (1.44) are available from the first step making equation (1.44) explicit and without requiring any iterations between the two steps.

In order to apply this scheme the sets $\mathcal{N}_{i,j}$ should contain as many points as possible. Accordingly, $\mathcal{X}$ is divided into two subset $\mathcal{R}_1$ and $\mathcal{R}_2$ such that

$$\begin{cases} \mathcal{R}_1 = \{x_{i,j} \in \mathcal{X} : \kappa_{i,j}\tilde{\phi}_{i,j} < 0 \vee (\kappa_{i,j} = 0 \wedge \tilde{\phi}_{i,j} < 0)\}, \\ \mathcal{R}_2 = \{\mathcal{X} \setminus \mathcal{R}_1\}, \end{cases} \tag{1.45}$$

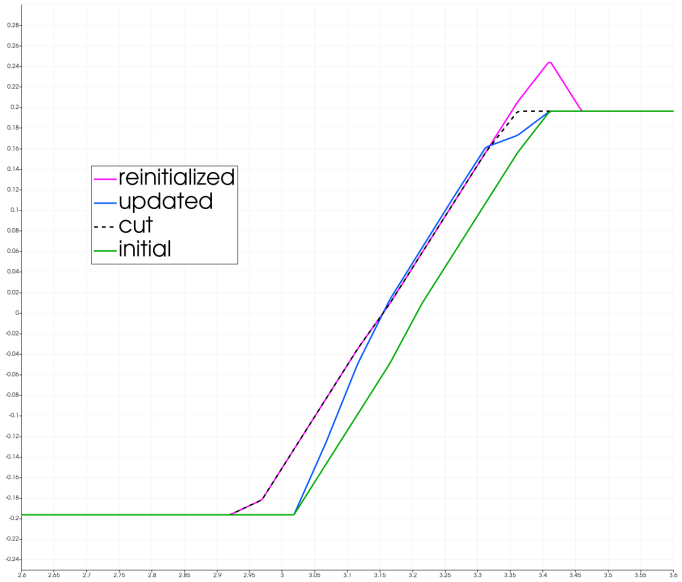

Figure 1.4: Typical evolution of a front in 1d. Starting from the green line, the update makes the front move resulting in the blue line, sharper than the greed one. Reinitialization (pink line) makes the computational narrow band move contextually with the front, fixing the zero level set and resetting the level set function to be a signed distance function with $|\nabla\phi| = 1$. The dotted black line represents the last cutting step necessary to obtain the final update.

where $\kappa_{i,j}$ is the curvature computed in the node $x_{i,j}$. Note that the inequalities in (1.45) are reversed with respect to [66, 67] due to the opposite convention for the sign of the level set function.

Finally, to reinitialize ϕ in the remaining points of the computational domain, or at least in a proper restricted region, the problem (1.30) is solved to steady state resorting to one of the canonical schemes described, such as (1.32) or (1.35).

1.6 Localization

We have already pointed out in the previous sections that, when dealing with LSM, one has to mitigate the extra cost of dealing with ϕ on $\mathbb{R}^d$ instead of the $d-1$-dimensional surface Γ. This is achieved by suitable localization techniques. Namely, the evolution and the subsequent adjustments of the level set function will be restricted at each time step to proper tubes around the front $\Gamma(t)$, as well stated in the work by Peng et al. [94] in 1999.

To this aim, let $0 < \beta < \gamma$ be two constants comparable to Δx and let us consider the modified level set equation

$$\phi_t + c(\phi)\vec{V} \cdot \nabla\phi = 0 \tag{1.46}$$

or

$$\phi_t + c(\phi)V_n \, |\nabla\phi| = 0 \tag{1.47}$$

where $c : \mathbb{R} \to \mathbb{R}$ is a cut-off function, which will be specified later, that has the role of mitigating the evolution at the boundaries of the tubes around $\Gamma(t)$, avoiding the outbreak of spurious oscillations.

Since it is not important to update ϕ far away from the interface, at each discrete time step t_n we choose a narrow band

$$\mathcal{B}^n = \{\vec{x} : |\phi^n(\vec{x})| < \gamma\} \subset \Omega, \tag{1.48}$$

and update ϕ^{n+1} only therein, while outside $\mathcal{B}^n$ we do not update the level set function at all. Moreover, at the end of each evolution step the level set function ϕ is cut as

$$\phi(\vec{x}) = \begin{cases} -\gamma & \text{if } \phi(\vec{x}) < -\gamma, \\ \phi(\vec{x}) & \text{if } |\phi(\vec{x})| \leq \gamma, \\ \gamma & \text{if } \phi(\vec{x}) > \gamma. \end{cases} \tag{1.49}$$

Let us suppose now to start our evolution with an initial guess $\phi^0(\vec{x})$ given by the signed distance function to the front Γ^0. According to (1.48), around Γ^0 we define a narrow band

$$\mathcal{B}^0 = \{x : \left|\phi^0(\vec{x})\right| < \gamma\} \tag{1.50}$$

and cut ϕ^0 as in (1.49). The initial data ϕ^0 is then updated by numerically solving for one time step one of the two equations (1.46), (1.47) to get $\tilde{\phi}^1(\vec{x})$, where the tilde over the letter ϕ is consistent with the notation used to describe the reinitialization procedure and emphasizes the fact that, after the evolution step, $\tilde{\phi}^1(\vec{x})$ is not in general a signed distance function.

The new location of the front is then given by $\Gamma^1 = \{\vec{x} : \tilde{\phi}^1(\vec{x}) = 0\}$, after the first evolution step. To perform a new temporal step we must construct a new level set function $\phi^1(\vec{x})$ from $\tilde{\phi}^1(\vec{x})$ such that

$$\phi^1(\vec{x}) = \begin{cases} -\gamma & \text{if } d^1(\vec{x}) < -\gamma, \\ d^1(\vec{x}) & \text{if } |d^1(\vec{x})| \leq \gamma, \\ \gamma & \text{if } d^1(\vec{x}) > \gamma. \end{cases} \tag{1.51}$$

where $d^1(\vec{x})$ denotes the signed distance function to Γ^1. This can be achieved by applying the reinitialization procedure described in the previous subsection on a larger narrow band $\overline{\mathcal{B}}^1 \supset \mathcal{B}^0$. Since our aim is always to restrict the computations to a neighbourhood of the front, we choose this $\overline{\mathcal{B}}^1$ to be small enough not to increase the computational cost too much, but also large enough to prevent distortions of the level set function.

In other words, let us suppose that in a single time step the front Γ^0 moves less than one grid point, then, as suggested in [94], we can choose the band for the reinitialization to be

$$\overline{\mathcal{B}}^1 = \{\vec{x} : \left|\phi^0(\vec{x} + \vec{y})\right| < \gamma, \text{ for certain } |\vec{y}| < \Delta x\}. \tag{1.52}$$

Otherwise, if our method allows the front to move more than one grid point, say we have a displacement of $k\Delta x$, we will consider

$$\overline{\mathcal{B}}^1 = \{\vec{x} : \left|\phi^0(\vec{x} + \vec{y})\right| < \gamma, \text{ for certain } |\vec{y}| < k\Delta x\}, \tag{1.53}$$

where the value of k depends on the time step restriction of the method, namely the CFL, used to solve the model equation (1.46),(1.47). Note that both the conditions (1.52) and (1.53) are posed on the level set at the previous time.

Once we reinitialize in $\overline{\mathcal{B}}^1$ and $\phi^1(\vec{x})$ is computed, we can consider the new narrow band

$$\mathcal{B}^1 = \{\vec{x} : \left|\phi^1(\vec{x})\right| < \gamma\} \tag{1.54}$$

and iterate this process to evolve the level set function ϕ while also adjusting it in these internal correcting steps, when needed.

It only remains to specify the definition of the cut-off function $c(\phi)$ we use in the modified equations (1.46) and (1.47). To prevent from numerical oscillations at the boundary of $\mathcal{B}^n$ the correct equation is solved only in a tube of radius β, while in the region $\{\vec{x} : \beta < |\phi^n(\vec{x})| \leq \gamma\}$, the motion is modified by the cut-off function and progressively slowed down when approaching the boundary. In our computations we have use the same cut-off function defined in [94], which is given by

$$c(\phi) = \begin{cases} 1 & \text{if } |\phi| \leq \beta, \\ \frac{(|\phi|-\gamma)^2(2|\phi|+\gamma-3\beta)}{(\gamma-\beta)^3} & \text{if } \beta < |\phi| \leq \gamma, \\ 0 & \text{if } |\phi| > \gamma. \end{cases} \tag{1.55}$$

Finally, in order to define the thresholds β and γ, one considers

$$\beta = \hat{\beta}\Delta x \quad \text{and} \quad \gamma = \hat{\gamma}\Delta x, \tag{1.56}$$

where $\hat{\beta}$ and $\hat{\gamma}$ are two positive constant independent of the grid size, instead they can be related to the CFL or to the sizes of stencils employed in the overall numerical scheme. Their values will be specified later in Chapter 4, where the specific application to surface reconstruction is treated.

1.7 Summary of the level set update

In the previous sections we have introduced all the building blocks to numerically compute the evolution of a level set function ϕ, governed by the level set equation (1.12) or (1.15). For the sake of clarity, Figure 1.4 shows the typical evolution of a front in 1d and, in Figure 1.5, a flowchart of the entire procedure is depicted. The scheme is summarized having in mind for instance the case of a level set function ϕ that is evolved until steady-state, hence we design a loop that stops when a proper stopping criteria is satisfied. Alternatively, one can fix a final time for the evolution. Also, we point out that each step of the scheme can itself require some other related computations: as an example, the computation of the velocity field, as explained in Section 1.3, might require an extension procedure or might be even related to a free boundary problem that need to be solve contextually with the updating of ϕ.

Here, referring to Figure 1.5, once the initial signed distance function ϕ^0 is given, we evolve it until steady-state combining all the procedures introduced before. The updating loop requires first of all to localize the update in the narrow band $\mathcal{B}^n$ as in (1.48); in this restricted region we compute the velocity field and, if not given in all $\mathcal{B}^n$, we extend it using equation (1.25); a new level set function $\tilde{\phi}^{n+1}$ is then

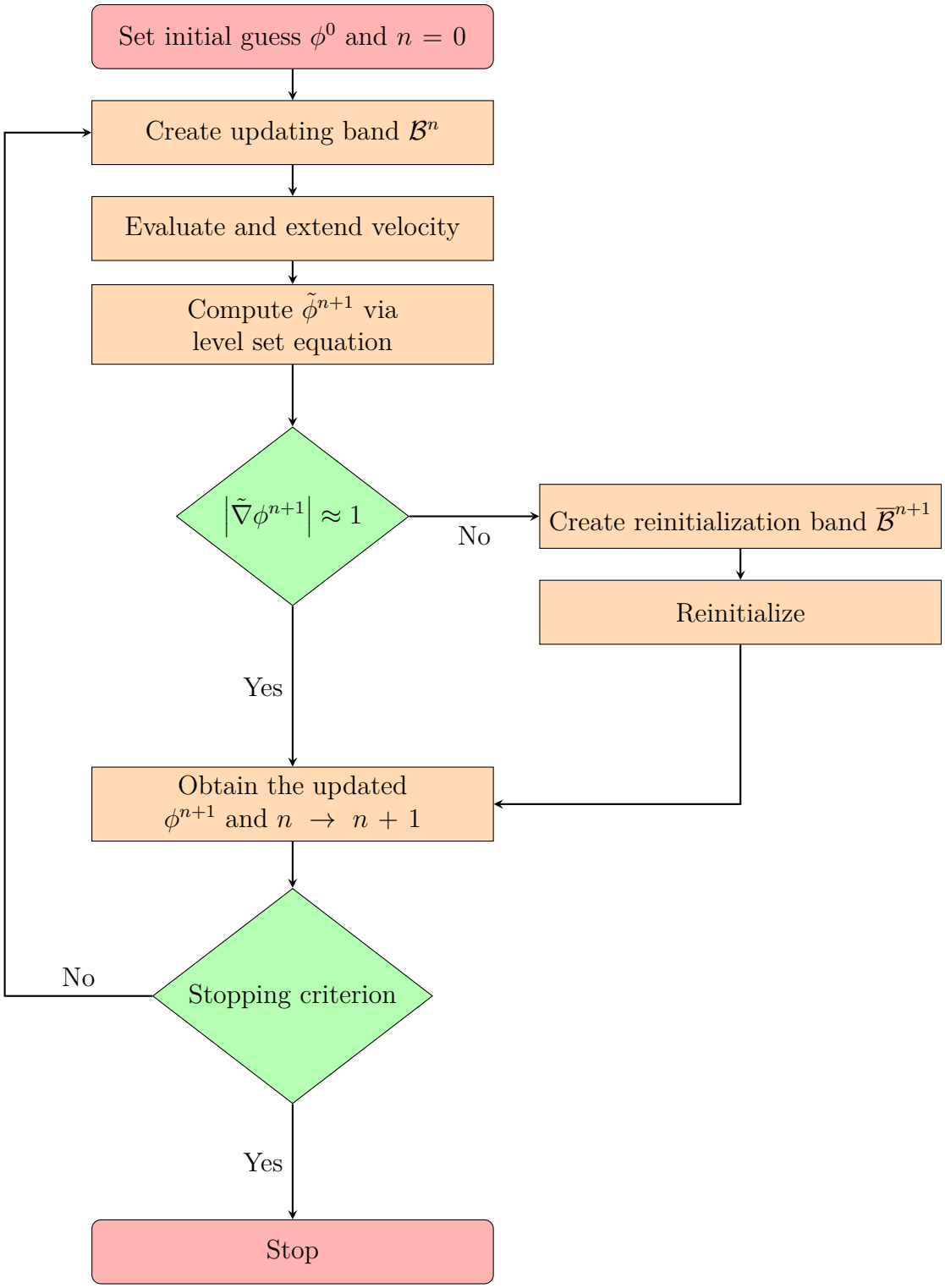

Figure 1.5: Flowchart for the level set update. Once the initial guess in chosen, we perform a loop to evolve the level set function until steady-state, localizing each procedure and checking the condition on the gradient to keep the level set function ϕ well behaved, i.e. close to be a SDF.

obtained via (1.12) or (1.15). Since $\tilde{\phi}^{n+1}$ might no longer be a SDF, we check the condition $\left|\nabla\tilde{\phi}^{n+1}\right| \approx 1$ to decide whether to reinitialize or not. In the first case, localization is again needed, as in (1.53), and (1.30) is solved to reset $\tilde{\phi}^{n+1}$ to be a SDF ϕ^{n+1}. Once the evolution in a time step is completed, we evaluate a proper stopping criterion and, if so, we stop the evolution; otherwise the loop proceeds again from the updating step.

Chapter 2

Semi-Lagrangian schemes

In this chapter we will describe the basics of the SL technique for the approximation of first-order PDEs, focusing for simplicity on the linear case in order to introduce the reader to the next chapters. The more general framework of Hamilton-Jacobi equations will be introduced in Section 2.4 and further developed in Chapter 3, together with the related application.

When dealing with PDEs, the main purpose of the SL approach is to obtain numerical schemes which are unconditionally stable with respect to the choice of the time step, in order to perform stable time updates at large Courant numbers. Introduced in their very first formulation by Courant, Isaacson and Rees in 1952 [41] for first-order systems of linear equations, this type of schemes have gone through a number of improvements and extensions. In particular, the possibility of making them work with large time steps was recognized later, as soon as they were applied to problems with a relevant computational complexity, for which time step restrictions can constitute a real issue.

The core of the method lies in the method of characteristics which accounts for the flow of information in the model PDE. At the numerical level, SL schemes mimic the method of characteristics and follow the characteristic curve pertaining to each node of the space-time discretization backward in time for the time step Δt, namely looking for what is usually referred to as the *foot of the characteristic*. Once the foot is found the solution at time $t + \Delta t$ is updated evaluating the solution at previous time t in the foot itself.

The SL approach thus combines the regular mesh of an Eulerian scheme with the stability properties of a Lagrangian one. In Fig 2.1 this kind of approach is better clarified: to compute the solution in the node (x_i, t_{n+1}), the characteristic curve $y(s)$ is traced backward from (x_i, t_{n+1}) to $(\hat{x}_i, t_n)$, then the neighbouring nodal values information are used to recover the value of the solution in the foot $(\hat{x}_i, t_n)$, for instance by interpolation. The approximated value at $(\hat{x}_i, t_n)$ will be the value of the solution in (x_i, t_{n+1}).

It is clear that this general idea requires several ingredients to be put together in order to design the complete scheme. Mainly, we need an ODEs solver to track the characteristic curves and a reconstruction technique to recover the values of the solution at previous time in the related feet that, in general, do not coincide with the nodes of the spatial discretization, where the solution is known. For more general schemes, we will see that also other ingredients will be needed.

In the next sections we will detail this approach mainly following the book by Falcone and Ferretti [54] in which a comprehensive and full explanation of SL schemes for linear hyperbolic PDEs and HJ equations is given. Along these lines, we will report just the concepts that we need as building blocks for what follows in the thesis.

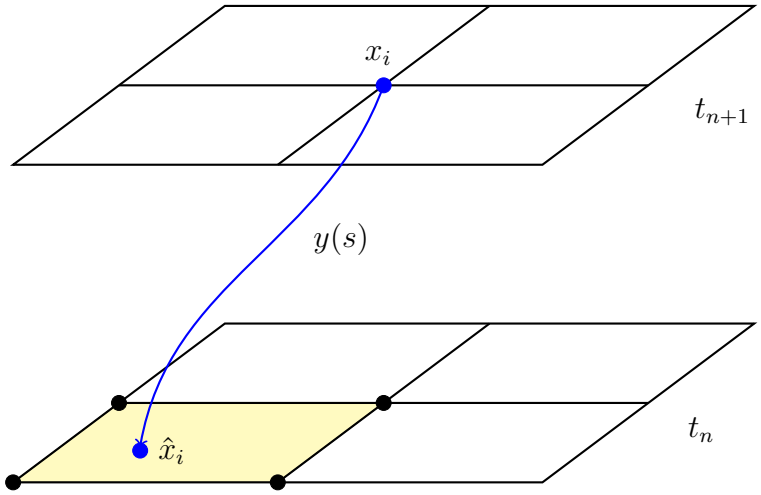

Figure 2.1: Sketch of a Semi-Lagrangian scheme. To compute the solution at (x_i, t_{n+1}), the corresponding characteristic curve $y(s)$ is tracked backward to some point $(\hat{x}_i, t_n)$. This point is located in the yellow cell and the nearby nodal values depicted with black dots (in case of a first-order scheme) are used to interpolate the solution in the foot at time t_n. This interpolated value is then the solution in the original node (x_i, t_{n+1}).

As an introduction, we start considering the simplest case of the one dimensional advection equation, first with constant, and then with variable coefficients. After introducing the basic first-order scheme we will move to the high-order SL approximation. A more general case for HJ equations will be treated successively.

In what follows the unknown function will be denoted by letter the v in order to distinguish this general framework from the level set one, introduced in the previous chapter. The discretized values of v will be indicated with the letter u. Moreover, since we will still consider a Cartesian uniform mesh, as in Chapter 1, but we will lighten the notation using $x_i = i\Delta x$, for a multi-index $i \in \mathbb{Z}^d$, to denote a node in the grid. To be consistent with this notation the discretized values at time n will be denoted by u_i^n, $i \in \mathbb{Z}^d$. When the subscript is dropped and the capital letter is used, e.g. writing U^n, or simply U, we mean to consider all the discretized values u_i^n, $i \in \mathbb{Z}^d$, or a finite subset of them, at a fixed time.

2.1 The constant-coefficient case

Let us consider first the initial value problem (IVP) for linear advection

$$\begin{cases} v_t(x,t) + cv_x(x,t) = 0, \\ v(x,t_0) = v_0(x), \end{cases} \tag{2.1}$$

where $(x,t) \in \mathbb{R} \times (t_0, T)$, c is a positive constant. The solution of (2.1) is given by the representation formula

$$v(x,t) = v_0(x - c(t - t_0)), \tag{2.2}$$

for each $(x,t) \in \mathbb{R} \times (t_0, T)$, which states that the solution is advected with constant velocity c, namely along parallel straight lines having slope c.

Considering a discretized grid and making the scheme evolve on a single time step, from time t_n to t_{n+1}, in the node x_i, (2.2) becomes

$$v(x_i, t_{n+1}) = v(x_i - c\Delta t, t_n), \tag{2.3}$$

where no approximation has yet been introduced.

Since the point $x_i - c\Delta t$ is not in general a grid point, in the SL approach we must introduce a numerical approximation in (2.3) by replacing its right-hand-side with an interpolation operator, thus obtaining the scheme

$$u_i^{n+1} = I[U^n](x_i - c\Delta t). \tag{2.4}$$

In (2.4), $I[U^n](x)$ denotes the interpolation at x of the data $\{u_i^n\}_{i\in\mathbb{Z}}$. In the simplest case, when I is the piece-wise linear interpolator, later on denoted by I_1, and we assume that $c\Delta t < \Delta x$, so that $x_i - c\Delta t \in (x_{i-1}, x_i]$, setting $\lambda = c\Delta t/\Delta x$, the scheme (2.4) becomes

$$u_i^{n+1} = \lambda u_{i-1}^n + (1-\lambda)u_i^n, \tag{2.5}$$

which precisely coincides with the well-known first-order upwind discretization.

However, the true potential of the SL approach is based on the validity of the representation formula (2.2) for any Δt, meaning that we are allowed to use Courant numbers beyond unity in the computations. Thus, for any $\lambda > 1$, we can rewrite $\lambda = \lambda' + \lfloor\lambda\rfloor$ and obtain the scheme

$$u_i^{n+1} = \lambda' u_{i-\lfloor\lambda\rfloor-1}^n + (1-\lambda')u_{i-\lfloor\lambda\rfloor}^n, \tag{2.6}$$

relying on the fact that, even when the characteristic crosses more than one cell, its foot falls in the interval $(x_{i-\lfloor\lambda\rfloor-1}, x_{i-\lfloor\lambda\rfloor}]$.

The scheme (2.6) is a first formulation of the CIR scheme introduced by Courant, Isaacson and Rees [41], which is reduced to the first-order upwind scheme when applied to the advection equation. Note also that, despite the possible large time step, the stencil of the scheme (2.6) involves a very low number of grid points.

2.2 The variable-coefficient case

This idea can be now generalized considering the IVP

$$\begin{cases} v_t(x,t) + f(x,t)v_x(x,t) = g(x,t), \\ v(x,t_0) = v_0(x), \end{cases} \tag{2.7}$$

where the velocity field f and the source term g are both scalar functions defined on $(x,t) \in \mathbb{R}\times(t_0,T)$. In this case, the more general form of the representation formula reads

$$v(x,t) = v_0(y(x,t;t-t_0),t_0) + \int_0^{t-t_0} g(y(x,t;s),t-s)\,ds, \tag{2.8}$$

where $y(x,t;s)$ is the solution at time $s \in [0, t-t_0]$ of the Cauchy problem, called *system of base characteristics*

$$\begin{cases} \dot{y}(x,t;s) = f(y(x,t;s), t-s), \\ y(x,t;0) = x. \end{cases} \tag{2.9}$$

We point out that the form (2.8)-(2.9) of the representation formula emphasizes the fact that the characteristics are integrated backward: as time goes from t_0 to $t > t_0$, the variable s goes in the opposite direction from 0 to $t - t_0$.

The function v defined by (2.8)-(2.9) solves (2.7) if v_0 is smooth. Its value in (x,t) is the sum of two contributions: the value of the initial data at the foot of the characteristic curve, which might no longer be a straight line, as in the constant coefficient case, and the integral of the source term along the characteristic itself.

Considering a single time step for the node x_i, yet not introducing numerical approximations, the representation formula (2.8) becomes

$$v(x_i, t_{n+1}) = v(y(x_i, t_{n+1}; \Delta t), t_n) + \int_0^{\Delta t} g(y(x_i, t_{n+1}; s), t_{n+1} - s)\, ds, \tag{2.10}$$

where $y(x_i, t_{n+1}, s)$ solves, for $s \in [0, \Delta t]$,

$$\begin{cases} \dot{y}(x_i, t_{n+1}; s) = f(y(x_i, t_{n+1}; s), t_{n+1} - s), \\ y(x_i, t_{n+1}; 0) = x_i. \end{cases} \tag{2.11}$$

In (2.10), in addition to replacing $v(y(x_i, t_{n+1}; \Delta t), t_n)$ with a reconstruction, two more approximations are required. First, the point $y(x_i, t_{n+1}, \Delta t)$ should be approximated, since the exact solution of (2.11) is in general not explicitly known. Second, the integral should be evaluated by some quadrature formula.

For instance, employing the explicit Euler approximation for the computation of the foot and the rectangle quadrature rule, and coupling them with a piecewise linear interpolator, one gets the CIR scheme

$$\begin{cases} u_i^{n+1} = I_1[U^n](x_i - \Delta t f(x_i, t_{n+1})) + \Delta t\, g(x_i, t_{n+1}), \\ u_i^0 = v_0(x_i), \end{cases} \tag{2.12}$$

where U^n is a proper subset of all the nodal values $\{u_i^n\}_{i \in \mathbb{Z}}$, used for the interpolation.

In order to show the properties of the SL approach, we will take the scheme (2.12) as an example of how the various ingredients of this technique interplay with each other in terms of consistency, stability, convergence and numerical viscosity.

2.2.1 Consistency

The analysis of the consistency of the CIR scheme (2.12) is made comparing it with the representation formula (2.8)-(2.9), rather than the model PDE (2.7). One starts by taking into account the following error estimates for the building blocks of the scheme:

$$|x_i - \Delta t f(x_i, t + \Delta t) - y(x_i, t + \Delta t; \Delta t)| = \mathcal{O}(\Delta t^2), \tag{2.13}$$

$$\left|\Delta t\, g(x_i, t+\Delta t) - \int_0^{\Delta t} g(y(x_i, t+\Delta t; s), t+\Delta t - s)ds\right| = \mathcal{O}(\Delta t^2), \tag{2.14}$$

$$\|v - I_1[V]\|_\infty = \mathcal{O}(\Delta x^2), \tag{2.15}$$

respectively for the ODE method, the quadrature rule and the interpolation operator.

From the bounds above we can deduce the following estimate for the local truncation error at node x_i:

$$\left|L_i^{CIR}(\Delta; t, V(t))\right| = \frac{1}{\Delta t}\left[\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta t^2) + L_v\mathcal{O}(\Delta x^2)\right], \tag{2.16}$$

where L_v is the Lipschitz constant of v, which itself bounds the Lipschitz constant of the reconstructed solution $I_1[V]$ and Δ in the argument denotes the dependence of the consistency error of the scheme on the spatio-temporal discretization sizes. Thus from (2.16) we can get the estimate

$$\left\|L^{CIR}(\Delta; t, V(t))\right\| \leq C\left(\Delta t + \frac{\Delta x^2}{\Delta t}\right), \tag{2.17}$$

which states that consistency is satisfied under the condition $\Delta x^2 = o(\Delta t)$. Note also that, unlike the usual bound for upwind schemes which takes into account the velocity of characteristics, here C is a positive constant completely independent of the velocity field f.

To include the case of small Courant number and verify consistency under any $\Delta x/\Delta t$ relationship, a more accurate estimate of the interpolation error needs to be considered. In [54], the authors use the fact that

$$\min_{m\in\mathbb{Z}}\left(\frac{\hat{x}_i - x_m}{\Delta x}\right) \leq \min\left(\frac{1}{2}, \frac{\hat{x}_i - x_i}{\Delta x}\right) \tag{2.18}$$

where $\hat{x}_i = y(x_i, t+\Delta t; t) = x_i + \mathcal{O}(\Delta t)$ is the point at which interpolation is performed, to improve the local error estimate for the linear interpolator by

$$|v(y(x_i, t+\Delta t; t)) - I_1[V](y(x_i, t+\Delta t; t))| \leq C\Delta x^2 \min\left(1, \frac{\Delta t}{\Delta x}\right). \tag{2.19}$$

Thus the consistency estimate can be rewritten as

$$\left\|L^{CIR}(\Delta; t, V(t))\right\| \leq C\left(\Delta t + \min\left(\Delta x, \frac{\Delta x^2}{\Delta t}\right)\right). \tag{2.20}$$

Finally, we point out for now that, from (2.17) and (2.20), the consistency rate is maximized when $\Delta t \sim \Delta x$.

2.2.2 Stability

The main property of the SL schemes is their unconditional stability, which allows for large Courant numbers. In fact, the restrictions introduced by the CFL condition are overcome, without introducing too much numerical diffusion, by employing

a small reconstruction stencil, which is shifted along characteristics towards the neighbourhood of the foot. Thus, the additional cost of tracking the characteristic curve is counterbalanced by first, a sort of self-adaptation of the numerical domain of dependence, and second, a reduction in the number of grid points involved in the numerical approximation, which in principle yields to a lower numerical viscosity.

However, the unconditional stability of SL-type schemes can be affected by the degree of the interpolation operator chosen to replace the value of the solution at the foot of the characteristic, since spurious oscillations might be introduced by enlarging the stencil of the reconstruction and the monotonicity might be lost. In fact, the stability of the SL approach is guaranteed only for interpolation techniques which do not increase the norm too much, in a sense that will be specified later, which is of course the case of the linear interpolation involved in (2.12), but can be an issue when high-order polynomial interpolation techniques are involved.

Monotonicity In order to check the monotonicity of the scheme (2.12) we need to rewrite it in a different form. Let $\{\varphi_j^{[1]}\}_{j\in\mathbb{Z}}$ be the basis functions of the interpolation operator $I_1[V]$ which are, in each cell $[x_j, x_{j+1}]$ polynomials of first degree such that $\varphi_j^{[1]}(x_k) = \delta_{j,k}$. Then, for any function $v(x)$ the linear interpolator is defined by $I_1[V](x) = \sum_{j\in\mathbb{Z}} v_j \varphi_j^{[1]}(x)$. Thus the scheme (2.12) can be rewritten as

$$u_i^{n+1} = \sum_{j\in\mathbb{Z}} v_j \varphi_j^{[1]}(x_i - \Delta t f(x_i, t_{n+1})) + \Delta t\, g(x_i, t_{n+1}), \tag{2.21}$$

from which it is easy to inspect monotonicity by checking if all the entries $\varphi_{ij} = \varphi_j^{[1]}(x_i - \Delta t f(x_i, t_{n+1}))$ are positive. Since $\varphi_j^{[1]}(x) \geq 0$ for each $j \in \mathbb{Z}$ and $x \in \mathbb{R}$, monotonicity is satisfied and, moreover, since $\sum_{j\in\mathbb{Z}} \varphi_j^{[1]}(x) \equiv 1$, the scheme is also invariant under the addition of constants.

Von Neumann analysis As usual, to perform a Von Neumann analysis, we refer to the scheme in the form (2.6), considering the simplest case of the linear advection equation. Vectors of the form $w_j = e^{\mathbf{i}j\omega}$, $j \in \mathbb{Z}$, are eigenvectors of the scheme (2.6) with eigenvalues ρ such that

$$\rho e^{\mathbf{i}j\omega} = \lambda' e^{\mathbf{i}(j-\lfloor\lambda\rfloor-1)\omega} + (1-\lambda') e^{\mathbf{i}(j-\lfloor\lambda\rfloor)\omega}, \tag{2.22}$$

where we recall that the Courant number λ has been written in the form $\lambda = \lambda' + \lfloor\lambda\rfloor$ in order to distinguish between its fractional and integer part. In a compact form, equation (2.22) reads

$$\rho = e^{\mathbf{i}\lfloor\lambda\rfloor\omega}\Big[\lambda' e^{-\mathbf{i}\omega} + (1-\lambda')\Big], \tag{2.23}$$

which states that ρ is given by the sum of a pure phase term and a second term, relevant for stability, depending only on the fractional part λ'. Focusing on just this second term one obtains the equality

$$|\rho| = \left|(1-\lambda') + \lambda'\cos\omega + \mathbf{i}\lambda'\sin\omega\right|, \tag{2.24}$$

which, since $\lambda' \in [0,1]$ locates the eigenvalues in a circle centered at $1-\lambda'$, with radius λ', thus guaranteeing that the Von Neumann condition is satisfied.

2.2.3 Convergence

To summarize this analysis for the CIR scheme (2.12), we report here the convergence result, as stated in [54], in the form of the following

Theorem 2.1. *Let $f, g \in W^{1,\infty}(\mathbb{R})$, let v be the solution of (2.7), and let u_i^n be defined by (2.10). Then, for any $i \in \mathbb{Z}$ and $n = 0 \dots N$,*

$$|u_i^n - v(x_i, t_n)| \to 0 \tag{2.25}$$

as $\Delta \to 0$. Moreover, if $v \in L^\infty([0,T], W^{s,\infty}(\mathbb{R}))$ $(s = 1, 2)$, then

$$\|U^n - V(t_n)\|_\infty \leq C \left(\Delta t + \min \left(\Delta x^{s-1}, \frac{\Delta x^s}{\Delta t} \right) \right). \tag{2.26}$$

2.2.4 Numerical viscosity

Performing a numerical viscosity analysis on the constant-coefficient advection case discretized by (2.6), if we consider the numerical solution $u(x,t)$ defined on $\mathbb{R} \times [t_0, T]$, obtained by a first-order local interpolation of the nodal values u_i^n, such that $u_i^n = u(x_i, t_n)$, we can deduce that this function $u(x,t)$ solves the modified equation

$$v_t + c v_x = \nu \, v_{xx} + o(\nu), \tag{2.27}$$

with the viscosity coefficient ν expressed by

$$\nu = \frac{\lambda'(1 - \lambda')\Delta x^2}{2\Delta t}. \tag{2.28}$$

This coefficient is positive, depends on λ', and vanishes if $\lambda' = 0$, which is consistent with the fact that, if the feet of characteristics coincide with grid nodes, namely $\lambda = \lfloor \lambda \rfloor$, then the solution is advected without errors. On the other hand, the largest value for (2.28) is attained for $\lambda' = 1/2$. In this case, equation (2.27) takes the form

$$u_t + c u_x = \frac{\Delta x^2}{8\Delta t} u_{xx} + o\left(\frac{\Delta x^2}{\Delta t} \right), \tag{2.29}$$

which shows that the viscous term, generated by the accumulation of interpolation errors, can be reduced, as we expected, by involving large Courant numbers $\lambda = c\Delta t / \Delta x$.

2.3 High-order approximation

The first-order discretization of the representation formula we have seen in the previous section is suitable for the homogeneous, constant-coefficient case, where it provides the exact upwinding along characteristics. However, in non-homogeneous problems, as well as in the variable-coefficient case, it could lead to an undesired error in time discretization. In practice, in order to balance the errors, the scheme could be forced again to work at small Courant numbers requiring an additional

computational complexity and against the possibility to reduce the numerical viscosity, as shown by (2.29).

Therefore, in order to get a better approximation for the solution of (2.7), we introduce a more accurate scheme for the tracking of characteristics and for the computation of the integral of the source term, without affecting the stability properties of the scheme. Moreover, recalling that the consistency rate in (2.17) is affected by the lowest of the rates in (2.13) and (2.14), one will need to employ both improved schemes in order to obtain higher accuracy.

Let us introduce a more general notation to rewrite the scheme (2.12) in the simplified form

$$\begin{cases} u_i^{n+1} = G_i^n + I_1[U^n](y_i^n), \\ u_i^0 = v_0(x_i), \end{cases} \tag{2.30}$$

where y_i^n and G_i^n denote respectively approximations of order p of the foot of the characteristic and of the integral of the source term along the characteristic itself. In the scheme (2.30) the interpolation operator is still the piecewise linear one, denoted by I_1.

Since the linear interpolator is still involved, stability is not affected, and one is just interested in the consistency analysis for which now we have

$$|y_i^n - f(x_i, t_{n+1}; t_n)| = \mathcal{O}(\Delta t^{p+1}), \tag{2.31}$$

$$\left| G_i^n - \int_{t_n}^{t_{n+1}} g(y(x_i, t_{n+1}; s), s) ds \right| = \mathcal{O}(\Delta t^{p+1}) \tag{2.32}$$

and thus

$$\left\| L^{CIR}(\Delta; t, V(T)) \right\| \leq C \left(\Delta t^p + \frac{\Delta x^2}{\Delta t} \right). \tag{2.33}$$

Consequently, with proper assumptions on the functions f and g, one can reformulate Theorem 2.1 in the more general form

Theorem 2.2. *Let $f, g \in W^{p,\infty}(\mathbb{R})$, let v be the solution of (2.7), and let u_i^n be defined by (2.30). Then, for any $i \in \mathbb{Z}$ and $n = 0 \dots N$,*

$$|u_i^n - v(x_i, t_n)| \to 0 \tag{2.34}$$

as $\Delta \to 0$. Moreover, if $v \in L^\infty([0,T], W^{s,\infty}(\mathbb{R}))$ $(s = 1, 2)$, then

$$\|U^n - V(t_n)\|_\infty \leq C \left(\Delta t^p + \min \left(\Delta x^{s-1}, \frac{\Delta x^s}{\Delta t} \right) \right). \tag{2.35}$$

Under large Courant numbers, introducing the relationship $\Delta t = \Delta x^\alpha$, the consistency error is $\mathcal{O}(\Delta x^{\alpha p} + \Delta x^{2-\alpha})$, thus the optimal choice for α in order to balance the two terms is given by $\alpha = 2/(p+1)$.

The last most delicate step is the one regarding the high-order interpolator. Let's say we are interested in an interpolation operator of order r, which will be denoted

below by I_r. With this further improvement, the SL scheme (2.30) for the problem (2.7) can be rewritten as

$$\begin{cases} u_i^{n+1} = G_i^n + I_r[U^n](y_i^n), \\ u_i^0 = v_0(x_i). \end{cases} \tag{2.36}$$

The consistency analysis carried out for the scheme (2.36) takes into account the estimates (2.31), (2.32) and replaces (2.33) with

$$\|u - I_r[U]\|_\infty = \mathcal{O}(\Delta x^{r+1}) \tag{2.37}$$

in order to get the consistency estimate for large Courant numbers

$$\left\| L^{SL}(\Delta; t, U(t)) \right\| \leq C \left(\Delta t^p + \frac{\Delta x^{r+1}}{\Delta t} \right), \tag{2.38}$$

and its improved version for all Courant numbers

$$\left\| L^{SL}(\Delta; t, U(t)) \right\| \leq C \left(\Delta t^p + \min \left(\Delta x^r, \frac{\Delta x^{r+1}}{\Delta t} \right) \right). \tag{2.39}$$

While high-order characteristics tracking does not change the stability properties of the CIR scheme, we know that the introduction of a space reconstruction of degree $r > 1$ does. A short dissertation on the stability of schemes involving high-order interpolation will be given in Subsection 2.2.3, where a sufficient condition for the case of non-oscillatory reconstructions will be illustrate.

2.3.1 High-order time integration

As the first ingredient for the construction of high-order SL schemes, we will consider high-order time integration to better track the characteristic curves that locate the values to be propagated. This task consists in accurately solving the system

$$\begin{cases} \dot{y}(x, t; s) = f(y(x, t; s), t - s), \quad s \in [0, t - t_0], \\ y(x, t; 0) = x, \end{cases} \tag{2.40}$$

where t_0 is the initial time and we suppose that the vector field $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is globally Lipschitz continuous with respect to its first argument.

Such a task can be numerically accomplished by different techniques, and, in particular, we will resort to a ν-stage explicit Runge-Kutta (RK) scheme. Thus, considering one time step and the usual notation for the space-time discretization, we look for the numerical solution of the system

$$\begin{cases} \dot{y}(x_i, t_{n+1}; s) = f(y(x_i, t_{n+1}; s), t_{n+1} - s), \quad s \in [0, \Delta t], \\ y(x_i, t_{n+1}; 0) = x_i, \end{cases} \tag{2.41}$$

considering a set of nodes t_k on the time axis and denoting with Y_k the corresponding approximations of $y(x_i, t_{n+1}; t_k)$.

Given the coefficients b_k, c_k, A_{kj} (for $k, j = 1, \dots, \nu$ and A strictly lower triangular) of the Butcher tableau, we can compute the solution of (2.41), denoted as y_i^n, by the scheme

$$\begin{aligned} y_i^n &= x_i + \Delta t \sum_{k=1}^{\nu} b_k K_k, \\ K_k &= f\left(Y_k, t_k\right), \qquad\qquad k = 1, \dots, \nu, \\ Y_k &= x_i + \Delta t \sum_{j=1}^{k-1} A_{kj} K_j, \qquad k = 1, \dots, \nu, \end{aligned} \tag{2.42}$$

where $t_k = t_{n+1} - c_k \Delta t$ are the abscissae of the RK scheme.

For the purposes of this thesis, we will resort to the Forward Euler method

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array} \tag{2.43}$$

for first-order schemes, to Heun's method

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array} \tag{2.44}$$

to get second-order accuracy, and to the RK method with tableau

$$\begin{array}{c|ccc} 0 & & & \\ 1/2 & 1/2 & & \\ 1 & -1 & 2 & \\ \hline & 1/6 & 2/3 & 1/6 \end{array} \tag{2.45}$$

for third-order schemes.

The schemes reported above allow us to approximate with the desired accuracy the solution of (2.41) at time t_n, namely the location of the foot of the characteristic in which we will perform the spatial interpolation.

Furthermore, high-order time integration is needed in order to track not only the foot, but also the intermediate values of $y(x_i, t_{n+1}; s)$, required to compute some quadrature formula to get an approximation G_i^n of the integral of the source term. Considering $\tilde{\nu}$ nodes ξ_l and weights w_l we compute

$$\begin{aligned} &\int_{t_n}^{t_{n+1}} g(y(x_i, t_{n+1}; s), s) \mathrm{d}s \\ &= \Delta t \sum_{l=1}^{\tilde{\nu}} w_l \, g(y(x_i, t_{n+1}, t_{n+1} - \xi_l \Delta t; t_{n+1} - \xi_l \Delta t) + \mathcal{O}(\Delta t^{q+1}) \\ &= \Delta t \sum_{l=1}^{\tilde{\nu}} w_l \, g(\tilde{Y}_l, t_{n+1} - \xi_l \Delta t) + \mathcal{O}(\Delta t^{q+1}) + \mathcal{O}(\Delta t^{s+1}), \end{aligned} \tag{2.46}$$

where q is the accuracy of the quadrature rule and s describes the extra error introduced by the approximation of $y(x_i, t_{n+1}; t_{n+1} - \xi_l \Delta t)$ with some proper values $\tilde{Y}_l$, for $l = 1, \dots, \tilde{\nu}$ within the quadrature rule.

Since we have already computed the intermediate stage values Y_k of the RK scheme, one is tempted to choose a quadrature rule such that $\tilde{\nu} = \nu$ and the nodes ξ_l coincide with the abscissae c_k of the RK scheme, in order to replace, in the last approximation of (2.46), $\tilde{Y}_l$ with the stage values Y_k of the RK scheme and the weights w_l with the coefficients b_k, for $l = k = 1, \dots, \nu$. We just need to check that this choice doesn't affect the accuracy of the approximation, ensuring that the balance between q and s is coherent also with the order of accuracy p of the RK scheme, and thus with the consistency error (2.33).

For first-order schemes, the foot is computed with first-order accuracy by (2.43) and the coupling with the rectangle rule gives the desired accuracy with $p = q = r = 1$.

For the Heun method, the numerical computation of G_i^n would be performed with the trapezoidal rule, whose nodes are at $\xi_l = 0, 1$; for this method $\tilde{Y}_1 = x_i$ and $\tilde{Y}_2 = y_i^n$ are computed with the correct accuracy by the RK scheme (2.44), thus the balance between the RK scheme and the quadrature rule gives the proper second-order accuracy.

For the third-order RK scheme (2.45), whose nodes are at $c = \{0, 1/2, 1\}$, one can choose the Simpson's quadrature rule and the approximations $\tilde{Y}_1 = x_i$, $\tilde{Y}_2 = Y_2$ and $\tilde{Y}_3 = Y_3$. In this case, even if both $\tilde{Y}_2 - Y_2$ and $\tilde{Y}_3 - Y_3$ are $\mathcal{O}(\Delta t^2)$, these errors cancel each other out when computing the linear combination of the Simpson's quadrature rule, resulting in a third-order accuracy for the approximation of G_i^n, as detailed for a general case in the paragraph below.

Quadrature rule accuracy In order to demonstrate the accuracy of the quadrature rule when relying on the stage values Y_k, $k = 1, \dots, 3$ computed resorting to an explicit third-order RK scheme, we consider the ODE equation (2.9) rewritten in the usual form

$$\begin{cases} \dot{y}(t) = f(y(t), y), \\ y(t_n) = y^n, \end{cases} \tag{2.47}$$

considering the interval $t \in [t_n, t_{n+1}]$ and f sufficiently smooth both in y and t.

We compute the numerical solution y^{n+1} of (2.47), at time t_{n+1}, as

$$\begin{aligned} y^{n+1} &= y^n + \Delta t \sum_{k=1}^{3} b_k K_k, \\ K_k &= f\left(Y_k, t_n + c_k \Delta t\right), \qquad k = 1, \dots, 3, \\ Y_k &= y_n + \Delta t \sum_{j=1}^{k-1} A_{kj} K_j, \qquad k = 1, \dots, 3, \end{aligned} \tag{2.48}$$

from which we can explicitly write the expressions for the stage values Y_k, for $k = 1, \dots, 3$, given by

$$\begin{aligned} &Y_1 = y^n, \\ &Y_2 = y^n + A_{21} \Delta t \dot{y}^n, \\ &Y_3 = y^n + (A_{31} + A_{3,2}) \Delta t \dot{y}^n + (A_{32} c_2 f_t^n + A_{32} A_{21} \dot{y}^n f_y^n) \Delta t^2 + \mathcal{O}(\Delta t^3), \end{aligned} \tag{2.49}$$

where $\dot{y}^n = f(y^n, t_n)$, f_y^n and f_t^n corresponds to the partial derivatives of the function f evaluated in the point (y^n, t_n).

The stage values (2.49) have been obtained by performing a Taylor expansion of the function f around the point (y^n, t_n) and one can compare these Y_k to the exact values of the solution $y_k = y(t_n + c_k\Delta t)$. When expanding the y_k around the point t_n, one gets the following expressions for the errors computed at each time t_k:

$$\begin{aligned} e_1 &= 0, \\ e_2 &= \frac{c_2^2}{2}\ddot{y}^n\Delta t^2 + \mathcal{O}(\Delta t^3), \\ e_3 &= \left(\frac{c_3^2}{2} - A_{32}A_{21}\right)\ddot{y}^n\Delta t^2 + \left(A_{32}A_{21} - A_{32}c_2\right) f_t^n\Delta t^2 + \mathcal{O}(\Delta t^3). \end{aligned} \tag{2.50}$$

These errors are the ones we need to take into account when applying a 3-points quadrature formula using the stage values Y_k instead of the exact values y_k.

In other words, let us denote with Q the integral of a sufficiently regular scalar function $q(y(t), t)$ over the interval $[t_n, t_{n+1}]$, where the function $y(t)$ solves (2.47). Applying a third-order quadrature rule with nodes t_k and weights w_k we get an approximation for Q given by

$$\overline{Q} = \Delta t \sum_{k=1}^{3} w_k q(y_k, t_k), \tag{2.51}$$

which is itself approximated by $\overline{\overline{Q}}$ if we use the Y_k instead of the y_k, thus getting

$$\begin{aligned} \overline{\overline{Q}} &= \Delta t \sum_{k=1}^{3} w_k q(Y_k, t_k) \\ &= \overline{Q} + \Delta t\Big[- w_2 e_2 q_y^2 - w_3 e_3 q_y^3\Big]. \end{aligned} \tag{2.52}$$

In equation (2.52) we have computed the expansions of the function q around the points (y_k, t_k) and we have denoted by q_y^k the partial derivative with respect to the variable y of q, evaluated in the point y_k, namely at time t_k.

Moreover, expanding these q_y^k around the same point, lets say y_2, one finally gets

$$\overline{\overline{Q}} = \overline{Q} + \Delta t\Big[- (w_2 e_2 + w_3 e_3) q_y^2 + \mathcal{O}(\Delta t^3)\Big], \tag{2.53}$$

which states that $\overline{\overline{Q}}$ is still a third-order approximation for Q, provided that the condition

$$w_2 e_2 + w_3 e_3 = 0 \tag{2.54}$$

holds.

Using the expression of the errors (2.50) and the coefficients b_k instead of the w_k, one finally concludes that the condition (2.54) is equivalent to having the equalities

$$A_{21} = c_2 \quad \text{and} \quad \frac{b_2 c_2^2 + b_3 c_3^2}{2} - b_3 A_{32} A_{21} = 0, \tag{2.55}$$

which are always satisfied for an explicit third-order RK scheme, since the third-order conditions for a RK process are

$$\sum_{k=1}^{3} b_k c_k^2 = \frac{1}{3} \quad \text{and} \quad \sum_{k,j=1}^{3} b_k A_{kj} c_j = \frac{1}{6}. \tag{2.56}$$

2.3.2 Continuous extensions of RK schemes

The RK schemes involved in this thesis, coupled with the related quadrature rules, give the desired accuracy in order to be coherent with the consistency error analyzed in (2.33). For more general schemes in which we may not enforce the equality between the ξ_l and the c_k, with $\tilde{\nu} = \nu$, or if we wanted to compute the intermediate stage values with the highest possible accuracy, a solution for the evalutation of the $\tilde{X}_l$ is to resort to the Natural Continuous Extensions (NCEs) of the RK methods [119].

We refer again to the IVP problem written in the simplified form (2.47) where $t \in (t_0, T)$ is a real variable, y_0, y and f are scalar functions defined on $\mathbb{R}$, with f being smooth enough both in t and y.

If a RK method of order $p \geq 1$ is applied, one gets information about the solution y on a discrete set of points such that

$$\max_{0 \leq n \leq N_T} |y_n - y(t_n)| = \mathcal{O}(\Delta t^p), \tag{2.57}$$

where the $\{y_n\}$ are the approximate values of the solution at time t_n. Nonetheless, one could be interested in obtaining a continuous approximate solution available, possibly avoiding any extra evaluations of the function f.

To this purpose, the author in [119] aims to give such a solution, namely the NCE, which, considering a single time step of size Δt, is defined as:

Definition 2.1. *The ν-stage RK process of order p has a NCE u of degree d if there exist ν polynomials $b_i(\theta)$, $i = 1, \ldots, \nu$, of degree $\leq d$, independent of the function f, such that, by putting*

$$u(t_0 + \theta \Delta t) \coloneqq y_0 + \Delta t \sum_{i=1}^{\nu} b_i(\theta) g^{(i)}, \quad 0 \leq \theta \leq 1, \tag{2.58}$$

the following statements hold:

$$u(t_0) = y_0 \quad \textit{and} \quad u(t_0 + \Delta t) = \overline{y}; \tag{2.59}$$

$$\max_{t_0 \leq t \leq t_0 + \Delta t} |y'(t) - u'(t)| = \mathcal{O}(\Delta t^d); \tag{2.60}$$

$$\int_{t_0}^{t_0 + \Delta t} G(t)[y'(t) - u'(t)] \mathrm{d}t = \mathcal{O}(\Delta t^{p+1}) \tag{2.61}$$

for every sufficiently smooth function G.

In the definition above $\overline{y}$ denotes the approximate value of $y(t_0 + \Delta t)$ given by the RK scheme and it is also easy to see that condition (2.60) implies the following error bounds for the higher derivatives of the NCEs:

$$\max_{t_0 \leq t \leq t_0 + \Delta t} \left| y^{(k)}(t) - u^{(k)}(t) \right| = \mathcal{O}(\Delta t^{d-k+1}), \tag{2.62}$$

for $k = 2, \ldots, d$, and $u^{(k)}(t) \equiv 0$ for $k \geq d + 1$. Therefore, all the derivatives of the NCE u are uniformly bounded as $\Delta t \to 0$.

Moreover, by integrating (2.60) and by (2.59), we have also

$$\max_{t_0 \le t \le t_0+\Delta t} |y(t) - u(t)| = \mathcal{O}(\Delta t^{d+1}), \tag{2.63}$$

while simple integration by parts, together with (2.61) and (2.59), yelds

$$\int_{t_0}^{t_0+\Delta t} G(t)[y(t) - u(t)]\mathrm{d}t = \mathcal{O}(\Delta t^{p+1}) \tag{2.64}$$

for every sufficiently smooth function G.

We point out that the condition (2.63) is crucial when one has some issues in the computation of the integral (2.46). In practice, once the degree d of the NCE is fixed, (2.63) guarantees a certain accuracy for all the possible intermediate values needed by the quadrature formula.

In [119] the existence of NCEs for all RK processes is proven and the NCEs for some of the most popular explicit RK processes are given. The following ones are sufficient for the topics of this thesis.

- 1-stage RK process of order 1 ($d = 1$)

$$b_1(\theta) \equiv \theta.$$

- 2-stage RK process of order 2 ($d = 1$)

$$b_i(\theta) \equiv b_i\theta, \quad i = 1, 2.$$

- 2-stage RK process of order 2 ($d = 2$)

$$\begin{cases} b_1(\theta) \equiv (b_1 - 1)\theta^2, \\ b_2(\theta) \equiv b_2\theta^2. \end{cases}$$

- 3-stage RK process of order 3 with $c_2, c_3 \neq 0$ ($d = 2$)

$$b_i(\theta) \equiv w_i\theta^2 + (b_i - w_i)\theta, \quad i = 1, 2, 3,$$

where

$$\begin{cases} w_1 \coloneqq -[\mu(c_3 - c_2) + c_2]/2c_2c_3, \\ w_2 \coloneqq \mu/2c_2, \\ w_1 \coloneqq (1 - \mu)/2c_3, \end{cases}$$

and $\mu \in \mathbb{R}$. In particular Theorem 7 in [119] provides the choice

$$\mu = 6c_2(2c_2 - 1)b_2.$$

Finally, we point out that in the case of 3-stage RK processes of order 3, no NCE of degree $\nu = p$ exists. Nonetheless, this issue doesn't pose any problem in our applications since, for the computation of the integral with a quadrature formula (2.46) we just need to have the intermediate values $\tilde{X}_l$ computed with an error of order $\mathcal{O}(\Delta t^p)$ in order to give a final error of order $\mathcal{O}(\Delta t^{p+1})$.

2.3.3 High-order non-oscillatory interpolator

As pointed out before, polynomial interpolations processes that are linear in the data show Gibbs oscillations when treating solutions with singularities, and therefore are best avoided in non-smooth contexts. This is the reason why essentially non-oscillatory techniques as ENO, WENO and successive central versions, come into play in order to reduce this effect.

Nonetheless, when involving a high-order interpolator one has to deal with stability issues and it can be really troublesome to prove convergence via classical results. For instace, bounds on the Lipschitz norm, which can be proved with reasonable effort in the case of low-order and usually monotone schemes, are not easy to prove in the high-order case, and also the classical result of Barles and Souganidis [12] which, roughly speaking, states that consistency, monotonicity and L^∞ stability imply convergence, does not hold due to the lack of monotonicity. Thus, we will resort here to some results based on uniform Lipschitz continuity for SL high-order schemes, along the lines of [56].

In what follows we will introduce the WENO reconstruction, as it will be used also in the next chapters. A sufficient condition for convergence, as presented in [56], will be explored at the end of this chapter in the framework of more general first-order HJ equations. This condition will be also investigated in Chapter 3 where we prove the convergence of the high-order SL scheme coupled with the CWENO reconstruction operator.

2.3.4 WENO reconstruction

We recall here the definition of the WENO operator, as it will be also used in our applications. This operator has been firstly introduced in [76] and has then gone through numerous extensions including the application to HJ equations [127, 125, 129, 97, 27]; in this presentation we will follow in particular the description given in [27].

For a point $\vec{x} \in \mathbb{R}^d$, let Ω be the grid cell containing it and $\mathcal{S}_\Omega$ be the set of its vertices. In order to achieve better than first-order accuracy, we need to consider stencils $\mathcal{S} \supset \mathcal{S}_\Omega$. We associate to any set of vertices $\mathcal{S}$ a polynomial of degree r', $P_{\mathcal{S}}^{(r')}(\vec{x}) \in \mathbb{P}_d^{r'}$, which interpolates the data in $\mathcal{S}$, i.e., such that $P_{\mathcal{S}}^{(r')}(\vec{x}_i) = v_i \ \forall i \in \mathcal{S}$, where the v_i are the nodal values of a given function v. Following the usual notation we will indicate with V the set of all nodal values v_i, or a finite subset of them.

As already pointed out, taking larger symmetric stencils could pose a serious issue since this choice leads to define highly accurate interpolators, which are however also very oscillatory when the data in $\mathcal{S}$ represent a non-smooth function. On the other hand, polynomials associated to smaller stencils, when biased in a specific direction, could avoid interpolating across the discontinuities in the data and show less spurious oscillations.

To face this drawback, ENO or WENO methods (see [65, 76]) tend to choose interpolation points on the smooth side of the function. ENO schemes tries to choose the best candidate stencil among all the possible ones. WENO instead weights the different polynomials, exploiting information from the whole global stencil, namely from both sides of the cell Ω. Briefly speaking, when constructing a WENO inter-

polator, one searches for a convex combination of low-degree polynomials, designed in such a way that a high-order reconstruction is computed in regions associated to smooth data, while non-oscillatory properties are guaranteed in presence of a discontinuity.

The selection or blending of such polynomials is performed in a non-linear way relying on oscillation indicators OSC[P], which are in general scalar quantities associated to a polynomial P, designed in such a way that OSC[P] $\to 0$ under grid refinement, if P is associated to smooth data, and OSC[P] $\asymp 1$ (i.e., the indicator is asymptotically a non-zero constant), in presence of a discontinuity within the stencil of P.

In this thesis, we rely on the classical Jiang-Shu oscillation indicators [76], suitably modified to evaluate the regularity of the solution of HJB problems under consideration [75, 52].

Construction of the WENO operator For practical purposes, here we will give some details of this procedure focusing on the one-dimensional reconstruction in the cell $\Omega_i = [x_i, x_{i+1}]$, considering a uniform Cartesian mesh of width Δx. To treat the multi-dimensional case, one only needs to iterate this procedure along each dimension.

More in details, to construct a WENO interpolation of degree $2n-1$ on the interval $[x_i, x_{i+1}]$, we start considering the Lagrange polynomial $Q(x) \in \mathcal{P}_{2n-1}$ built on the stencil $S = \{x_{i-n+1}, \dots, x_{i+n}\}$, written in the form

$$Q(x) = \sum_{k=1}^{n} C_k(x) P_k(x), \tag{2.65}$$

where the C_k are polynomials of degree $n-1$ and the P_k are polynomials of degree n interpolating on the stencil $\mathcal{S}_k = \{x_{i-n+k}, \dots, x_{i+k}\}$, $k = 1, \dots, n$. Then, we proceed as follows:

1. compute suitable regularity indicators

$$\mathrm{OSC}_k = \mathrm{OSC}[P_k], \quad k = 1, \dots, n; \tag{2.66}$$

2. define the quantities

$$\alpha_k(x) = \frac{C_k(x)}{(\mathrm{OSC}_k + \epsilon)^2}, \tag{2.67}$$

with $\epsilon = \Delta x^2$;

3. compute the non-linear weights $\{w_k\}_{k=1}^n$ as

$$w_k(x) = \frac{\alpha_k(x)}{\sum_l \alpha_l(x)}; \tag{2.68}$$

4. finally, define the reconstruction at x by

$$I[V](x) = \sum_{k=1}^{n} w_k(x) P_k(x). \tag{2.69}$$

Differently from [27], we consider regularity indicators defined as

$$\mathrm{OSC}_k = \mathrm{OSC}[P_k] = \sum_{\beta \geq 2} \Delta x^{2\beta-3} \int_{x_j}^{x_{j+1}} \left(\frac{d^{(\beta)} P_k}{dx^\beta} \right)^2 dx. \tag{2.70}$$

We point out that the above is similar to the classical definition of the oscillation indicators for the WENO reconstruction as given in [76], except that the first derivative is not included in the sum, thus implying a different scaling given by $\Delta x^{2\beta-3}$ to ensure the expected behaviours of the indicators. This choice is justified by the fact that, since we will employ this interpolator in the HJ framework, the function that we will interpolate can be at worst continuous with kinks.

In fact, let us consider a generic polynomial P_k of degree $n-1$ having a discontinuity in its first derivative in the stencil $\mathcal{S}_k$. When we consider its β-derivative, for $\beta \geq 2$, we get a polynomial of degree $n-\beta-1$ whose coefficients of degree $m = 0, \dots, n-\beta-1$ are given by the Newton divided differences with $m+\beta+1$ points that scale as $\mathcal{O}(\Delta x^{1-m-\beta})$ due to the discontinuity. Now, computing the integral in (2.70), we get a sum of terms, each one scaling at the same rate, given by $\mathcal{O}(\Delta x^{3-2\beta})$. Is it thus clear that the exponent $2\beta-3$ is apt to get $\mathrm{OSC}[P_k] \asymp 1$ in this non-regular case. In case of smoothness, instead the Newton divided differences would be $\mathcal{O}(1)$ and $\mathrm{OSC}[P_k] \to 0$ as expected.

Third-order reconstruction Focusing on the third-order case, we need to find a proper expression for the linear weights C_k considering $n = 2$ to get a formally third-order reconstruction. The whole stencil will be given by $\mathcal{S} = \{x_{i-1}, x_i, x_{i+1}, x_{+2}\}$ and the only two substencils to consider are $\mathcal{S}_L = \{x_{i-1}, x_i, x_{i+1}\}$ and $\mathcal{S}_R = \{x_i, x_{i+1}, x_{i+2}\}$. Since the polynomials P_L and P_R interpolate the function v respectively on $\mathcal{S}_L$ and $\mathcal{S}_R$, each C_k should vanishes outside $\mathcal{S}_k$, namely $C_L(x_{i+2}) = 0$ and $C_R(x_{i-1}) = 0$, and the condition $\sum_{k=L,R} C_k(x_j) = 1$ must hold for every node $x_j \in \mathcal{S}$. Thus, we have two conditions for each polynomial C_k, which, in our case leads to a unique definition of these one degree polynomials that we can write in the form

$$C_k(x) = \gamma_k \frac{x - \hat{x}_k}{\Delta x}, \tag{2.71}$$

where $k = L, R$ and $\hat{x}_k \in \mathcal{S} \setminus \mathcal{S}_k$.

To get the coefficients γ_L and γ_R we start considering the first node of the stencil x_{i-1}, on which we have $C_L(x_{i-1}) = 1$ and $C_R(x_{i-1}) = 0$, so that

$$Q(x_{i-1}) = C_L(x_{i-1}) P_L(x_{i-1}) = C_L(x_{i-1}) v_{i-1}, \tag{2.72}$$

thus inferring that $C_L(x_{i-1}) = 1$ and therefore, using (2.71) with $\hat{x}_k = x_{i+2}$,

$$\gamma_L = -\frac{1}{3}. \tag{2.73}$$

Analogously, we can proceed considering the node x_i for which we must have $C_L(x_i) + C_R(x_i) = 1$. Since the expression for C_L is known, we can compute

$$C_R(x_i) = 1 - C_L(x_i) = 1 - \frac{2}{3} = \frac{1}{3} \tag{2.74}$$

and therefore, from $\frac{1}{3} = \gamma_R[(x_i - x_{i-1})/\Delta x]$ in (2.71), we get

$$\gamma_R = \frac{1}{3}. \tag{2.75}$$

Summarizing, we have the following expressions:

$$C_L(x) = \frac{x_{i+2} - x}{3\Delta x}, \quad C_R(x) = \frac{x - x_{i-1}}{3\Delta x}. \tag{2.76}$$

It only remains to find a proper expression of the regularity indicators OSC_L and OSC_R for the two degree polynomials P_L and P_R. Let us consider a polynomial of degree at most two written in the form $P(x) = \sum_{j=0}^{2} z_j[(x - x_j)/\Delta x]^j$. Its indicator is a quadratic form of its coefficients and is given by

$$\mathrm{OSC}[P] = \frac{1}{\Delta x^2} 4 z_2^2$$

or equivalently, denoting by U the vector of data $(u_{i-1}, u_i, u_{i+1}, u_{i+2})^T$, since the coefficients of the polynomial linearly depends on U, we can express the regularity indicators as a quadratic form of the data being interpolated. For our specific case we obtain

$$\mathrm{OSC}[P_k] = \frac{1}{\Delta x^2} U^T A_k U \quad (k = L, R),$$

with

$$A_L = \begin{pmatrix} 1 & -2 & 1 & 0 \\ -2 & 4 & -2 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad A_R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & -2 & 4 & -2 \\ 0 & 1 & -2 & 1 \end{pmatrix}.$$

Finally, we illustrate briefly the two-dimensional reconstruction. Let (x, y) be the reconstruction point, located in the cell $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$. One first performs four one-dimensional WENO interpolations to compute auxiliary data $w_{i-1}, w_i, w_{i+1}, w_{i+2}$. Each w_k is the interpolation in the y direction of the data $u_{k,j-1}, \ldots, u_{k,j+2}$ evaluated at y. Finally a WENO interpolation in the x direction of $w_{i-1}, w_i, w_{i+1}, w_{i+2}$ evaluated at x will be the reconstructed value. The three-dimensional case can be treated analogously.

Note that, due to the dimensional splitting employed in the two and three dimensional procedures, the WENO interpolator is globally continuous on the edges of the cell in which the reconstruction is performed.

Now that we have defined the high-order interpolator, but still we need to investigate the possible issues related to high-order interpolation and stability. In the next section we will recall the basic concepts of the convergence theory developed in [56, 27], in the framework of HJ equations, in order to give a sufficient condition for the WENO, and successively for the CWENO, case.

2.4 First-order HJ equations

In order to generalize the construction of the SL scheme for PDEs, we consider now a case of first-order HJ equation in a multi-dimensional framework, assuming that the Hamiltonian function does not explicitly depend on space and time. In the construction of SL schemes for HJ equations, a central role is played by the Lax-Hopf formula and, as we will see, by the Legendre transform of the Hamiltonian.

Let us consider the following model problem:

$$\begin{cases} v_t(\vec{x},t) + H(\nabla v(\vec{x},t)) = 0, \\ v(\vec{x},t_0) = v_0(\vec{x}), \end{cases} \tag{2.77}$$

for $(\vec{x},t) \in \mathbb{R}^d \times (t_0,T)$, $\vec{x} \in \mathbb{R}^d$ and where $H : \mathbb{R}^d \to \mathbb{R}$ is the Hamiltonian function which is assumed to be convex and satisfying the coercivity condition

$$\lim_{|\vec{p}|\to+\infty} \frac{H(\vec{p})}{\vec{p}} = +\infty. \tag{2.78}$$

The assumption above allows us to give the following definition.

Definition 2.2. *Let H be convex and coercive in the sense of* (2.78)*. We define the* Legendre-Fenchel conjugate (or Legendre-Fenchel transform) of H for $\vec{q} \in \mathbb{R}^d$ as

$$H^*(\vec{q}) = \sup_{\vec{p}\in\mathbb{R}^d} \{\vec{p}\cdot\vec{q} - H(\vec{p})\}. \tag{2.79}$$

In the general case, an explicit computation of H^* is not available and a numerical procedure might be needed to approximate it. Also, it is worth noting that the assumptions made on H, convexity and coercivity in the sense of (2.78), respectively imply that H is continuous and that the sup in (2.79) is in fact a maximum.

In general, the Legendre-Fenchel transform may not allow for an explicit computation, thus in this case it should be approximated with some numerical procedure. Anyway, an useful analytical example is the one related to the quadratic Hamiltonian

$$H_2(\vec{p}) = \frac{|\vec{p}|^2}{2}, \tag{2.80}$$

for which an easy computation gives

$$H_2^*(\vec{q}) = \frac{|\vec{q}|^2}{2}. \tag{2.81}$$

Finally, an important results about the Legendre transform is given by the following

Theorem 2.3. *Let H be convex and coercive in the sense of* (2.78)*. Then, the function H^* has the following properties:*

(i) $H^* : \mathbb{R}^d \to \mathbb{R}$ is convex and

$$\lim_{|\vec{p}|\to\infty} \frac{H^*(\vec{p})}{|\vec{p}|} = +\infty;$$

(ii) $H(\vec{p}) = H^{**}(\vec{p})$ *for any* $\vec{p} \in \mathbb{R}^d$.

We reported here the definition of the Legendre transform due to its importance for characterizing the unique solution of (2.77), given by the so called *Hopf-Lax representation formula* (see [70]), as stated by the following Theorems [54].

Theorem 2.4. *The function* v *defined by the following Hopf-Lax formula:*

$$v(\vec{x}, t) = \sup_{\vec{y} \in \mathbb{R}^d} \left\{ v_0(\vec{y}) + tH^* \left(\frac{\vec{x} - \vec{y}}{t} \right) \right\} \tag{2.82}$$

is Lipschitz continuous, is differentiable a.e. in $\mathbb{R}^d \times (0, +\infty)$*, and solves in the viscosity sense the IVP* (2.77).

Proof The proof of this result can be found in [54].

Theorem 2.5. *The unique viscosity solution of* (2.77) *is given by the Hopf-Lax representation formula* (2.82).

Proof The proof of this result can be found in [49].

Finally, we note that setting

$$\vec{a} := \frac{\vec{x} - \vec{y}}{t}, \tag{2.83}$$

so that $\vec{y} = \vec{x} + \vec{a}t$, we can exchange the maximization with respect to $\vec{y}$ with a minimization with respect to $\vec{a}$, rewriting the Hopf-Lax formula as

$$v(\vec{x}, t) = \inf_{\vec{a} \in \mathbb{R}^d} \left\{ v_0(\vec{x} - \vec{a}t) + tH^*(\vec{a}) \right\}, \tag{2.84}$$

which recalls the representation formula used for the linear advective equation in order to get the SL discretization.

2.4.1 SL discretization

The construction of the SL scheme for HJ equation (2.77) follows the same steps as for the advection equation and is again obtained by discretizing the representation formula (2.84) rather than the original equation.

Considering the usual spatio-temporal discretization, in a node (x_i, t_{n+1}) of the grid, the solution is given by

$$\begin{aligned} v(x_i, t_{n+1}) &= \min_{\underline{a} \in A} \left\{ v(x_i - \underline{a}\Delta t, t_n) + \Delta t H^*(\underline{a}) \right\} \\ &= v(x_i - \underline{a}_i^* \Delta t, t_n) + \Delta t H^*(\underline{a}_i^*), \end{aligned} \tag{2.85}$$

where $\underline{a} \in A \subset \mathbb{R}^d$ is a discretized version of the variable $\vec{a} \in \mathbb{R}^d$ and $\underline{a}_i^*$ is the optimal value found for the node x_i. The set $A \subset \mathbb{R}^d$ is a compact set in which we look for this minimizer.

Excluding the difficulties in carrying out the minimization procedure, the same considerations made for the linear advection case hold in this framework, too. In

this particular case, since the Hamiltonian function does not explicitly depend on space and time, the characteristics are straight lines and high-order integration in time is not required to improve the accuracy when locating the points $x_i - \underline{a}\Delta t$. On the other hand, the evaluation of the function v at time t_n in the foot might require a high-order interpolator to be involved.

In this first case the SL scheme takes the form

$$\begin{cases} u_i^{n+1} = \min\limits_{\underline{a}\in\mathbb{R}^d} \left\{ I_r[U^n](x_i - \underline{a}\Delta t) + \Delta t H^*(\underline{a}) \right\}, \\ u_i^0 = v_0(x_i). \end{cases} \tag{2.86}$$

which introduces a spatial reconstruction of degree r.

Despite its apparent simplicity, the Hopf-Lax formula, and the scheme (2.86), introduce the important role of the minimization process for the computation of the solution of (2.77). The procedures of ODE integration, quadrature and interpolation, that, in the previouos case, were just performed once, now need to be repeated for a discretized set of values $\underline{a} \in A$ until the optimum $\underline{a}^*$ is found, or at least well approximated. This clearly poses a real issue when considering the computational effort of such an algorithm. In Figure 2.2 a sketch of the scheme is depicted: to compute the solution in the node (x_i, t_{n+1}), different values of $\underline{a} \in A$ are considered, each one leading to a different location of the foot and a different evaluation of the Legendre transform H^*.

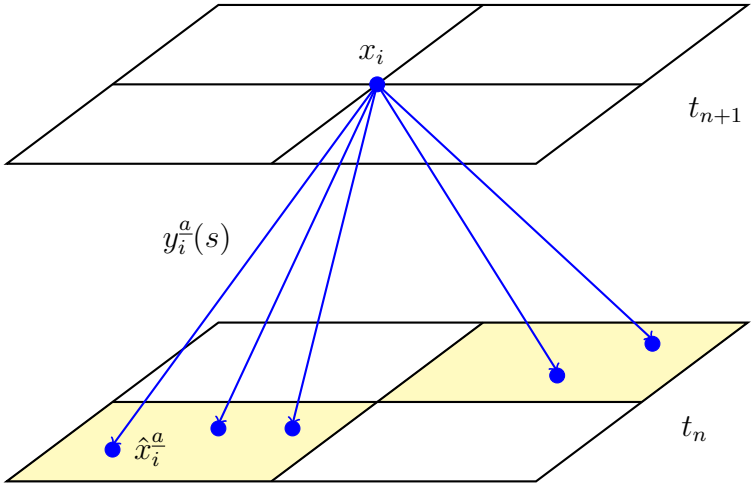

Figure 2.2: Sketch of a Semi-Lagrangian scheme for HJ equations. To compute the solution at (x_i, t_{n+1}), the corresponding characteristic curve $y^{\underline{a}}(s)$ is tracked backward to some point $(\hat{x}_i^{\underline{a}}, t_n)$ depending on the value of $\underline{a}$. The update is performed minimizing the sum of the interpolation in the foot and of the term depending on the Legendre transform evaluated in $\underline{a}$, according to (2.86).

In the next Chapter, a more general case for HJ equations will be investigated and an accent will be posed on the computational cost of the scheme. In fact, things might become worse when the Hamiltonian is a function that depends on both space and time, since this requires to reintroduce in the scheme the high-order time integration, with the consequent increase in the computational effort.

2.4.2 Convergence theory

To analyze the convergence for a SL, high-order scheme we follow the lines of [56, 54], considering the model problem (2.77) is one dimension, namely

$$\begin{cases} v_t(x,t) + H(v_x(x,t)) = 0, & (x,t) \in \mathbb{R} \times [0,t_0], \\ v(x,t_0) = v_0(x), & x \in \mathbb{R}, \end{cases} \tag{2.87}$$

and assuming that v_0 is a Lipschitz continuous function and that the function H is strictly convex, and more precisely

$$H''(p) \geq m_H > 0, \tag{2.88}$$

which implies the condition on the Legendre transform

$$H^{*''}(\alpha) \leq \frac{1}{m_H}. \tag{2.89}$$

The convergence theory we recall here overcomes the non-monotonicity of the numerical scheme by using a weaker condition, that is, that (2.86) should be monotone up to a term $o(\Delta t)$, to apply the Barles-Souganidis Theorem. This condition turns out be valid under increasing Courant numbers if the scheme is Lipschitz stable, so this will be the core of the stability issue.

More specifically, the key assumption for the convergence result is that for any Lipschitz continuous function $v(x)$, once defined the sequence $V = \{v_i\}_{i\in\mathbb{Z}} = \{v(x_i)\}_{i\in\mathbb{Z}}$, the interpolation operator $I_r[V](x_k)$ satisfies

$$I_r[V](x_k) = v(x_k) \tag{2.90}$$

and, for some constant $C_r < 1$,

$$|I_r[V](x) - I_1[V](x)| \leq C_r \max_{x_k \in B(x)} |v_{k+1} - 2v_k + v_{k+1}| \tag{2.91}$$

where by $I_1[V]$ we denote the piecewise liner interpolation on the sequence V, and with $B(x) = (x - h_-\Delta x, x + h_+\Delta x)$ we denote the interval containing the stencil of the reconstruction $I[V](x)$. For example, in the case of WENO, the reconstruction in performed taking two nodes on the left and two nodes on the right of the point x, so that $h_- = h_+ = 2$.

The key point of assumption (2.91) consists in requiring that $C_r < 1$, which amounts to assuming that I_r is not "too sensitive" with respect to large second increments which can occur on singularities of the solution.

In [56, 54] the authors give a bound on the second increment of the numerical solution which is globally one-sided, but becomes two-sided at the foot of characteristics, namely in a neighbourhood $\overline{B}(x_i + \alpha_i^*\Delta t)$, with $\overline{B}$ defined as

$$\overline{B}(x) = (x - h\Delta x, x + h\Delta x), \tag{2.92}$$

with a fixed $h > \max(h_-, h_+)$, so that $B(x) \subset \overline{B}(x)$.

Thus the first technical result is given by the following

Lemma 2.6. *Let U^n be defined by the scheme* (2.86)*. If* (2.89) *holds, then, for any $k \in \mathbb{Z}$ and $n \geq 1$,*

$$u_{k+1}^n - 2u_k^n + u_{k-1}^n \leq \frac{\Delta x^2}{m_H \Delta t}. \tag{2.93}$$

Moreover, assuming in addition that (2.91) *holds, then*

$$\max_{x_{k-1},x_k,x_{k+1} \in \overline{B}(x_i + \alpha_i^* \Delta t)} \left| u_{k+1}^n - 2u_k^n + u_{k-1}^n \right| \leq C \frac{\Delta x^2}{\Delta t} \tag{2.94}$$

with $\overline{B}$ defined by (2.92) *and for some positive constant C depending on C_r, h and m_H.*

Proof The proof of this result can be found in [56, 54].

The result obtained with Lemma 2.6 allows us to state the result of Lipschitz stability.

Theorem 2.7. *Let U^n be defined by the scheme* (2.86)*. Assume that* (2.88)*,* (2.91) *hold, that $\Delta x = \mathcal{O}(\Delta t^2)$, and that v_0 is Lipschitz continuous with Lipschitz constant L_0. Then U^n satisfies, for any i and j, the discrete Lipschitx estimate*

$$\frac{\left| u_i^n - u_j^n \right|}{x_i - x_j} \leq L \tag{2.95}$$

for a constant L independent of Δx and Δt, and for $0 \leq n \leq N_T$, as $\Delta t \to 0$.

Last, under such assumptions it is possible (see [56, 54]) to prove the main convergence result for the scheme (2.86).

Theorem 2.8. *Consider the scheme* (2.86) *applied to equation* (2.87)*, and assume that* (2.88) *and* (2.91) *hold, that $\Delta x = \mathcal{O}(\Delta t^2)$ and that v_0 is Lipschitz continuous. Then, the numerical solution $U^n = \{u_i^n\}_i$ (with u_i^n defined by* (2.86)*) satisfies*

$$\| I_r[U^n] - v(n\Delta t) \|_\infty \to 0$$

where v is the solution of (2.87)*, for $0 \leq n \leq N_T$, as $\Delta t \to 0$.*

Proof The proof of this result can be found in [56, 54]. The key point for proving convergence consists in obtaining a scheme which is monotone up to a term $\mathcal{O}(\Delta x)$ (therefore $o(\Delta t)$) such that a generalized monotonicity condition is satisfied and the Barles-Souganidis Theorem can be applied.

Remark 2.9. *In this theorem, the condition $\Delta x = \mathcal{O}(\Delta t^2)$ is required only to ensure Lipschitz stability. The proof itself requires the weaker condition $\Delta x = o(\Delta t)$, which is in turn related to consistency. Since the scheme experimentally appears to be convergent under any $^{\Delta t}/_{\Delta x}$ relationship, we should infer that this convergence result in not optimal.*

Convergence for the WENO case It is proved in [56] that condition (2.91) is satisfied for Lagrange reconstructions up to the fifth-order, if the reconstruction stencil includes the interval $[x_i, x_{i+1}]$, namely the interval in which the foot of characteristic falls and where we want to interpolate. Since WENO interpolation is performed by taking a convex combination of polynomials which satisfy (2.91) up the degree 5 for the partial polynomials (and therefore, up to the degree 9 for the global interpolant), this fact is used in [27] to obtain convergence via Theorem 2.8. This essentially boils down to proving that all the function appearing in the convex combination defining the interpolant I satisfy (3.31), and we plan to apply the same principle in the next chapter in order to prove the convergence for the CWENO case. In particular, for the third-order WENO reconstruction in which we start from polynomials of degree 2, the condition (2.91) holds with $C_2 = 1/8$.

Chapter 3

High-order CWENO for HJ equations

In the previous chapter all the building blocks for the construction of a SL scheme for linear first-order hyperbolic PDEs and simple cases of HJ type equations have been presented. Now we show how to adapt this type of scheme to more complex HJ equations, considering in particular the case of a Hamiltonian function that might explicitly depends on both space and time. The presentation in this chapter follows closely that of [28].

In order to lighten the notation, even if we treat multi-dimensional case, we will simply use the notation x for a point $x \in \mathbb{R}^d$ and we will also switch the order of the arguments x and t due to the dependence of the spatial variable on time, when tracking the characteristics.

In our study we consider the following hyperbolic Hamilton-Jacobi-Bellman (HJB) equation:

$$\begin{cases} v_t(t,x) + H(t,x,\nabla v(t,x)) = 0, & \text{for } t,x \in (0,T)\times\mathbb{R}^d, \\ v(0,x) = v_0(x), & \text{for } x \in \mathbb{R}^d, \end{cases} \tag{3.1}$$

where $v : (0,T)\times\mathbb{R}^d \to \mathbb{R}$, ∇v stands for the spatial gradient, $v_0 : \mathbb{R}^d \to \mathbb{R}$ denotes the initial data and $H : (0,T)\times\mathbb{R}^d\times\mathbb{R}^d \to \mathbb{R}$ is the Hamiltonian function, which will be assumed to be convex with respect to ∇v.

Numerous schemes have been proposed to approximate (3.1), but only a small number of them is aimed at high-order accuracy. To recall some main results in literature devoted to the SL framework, we find in [53] a high-order semi-discrete SL scheme proposed to discretize stationary HBJ euqations, while in [55], the evolutionary case is studied. A high-order SL scheme for HJB quations is presented in [27] and is particularly interesting since it combines the SL technique with WENO reconstructions.

In fact, as a next step, the aim of this work is to present a new scheme that couples the SL technique with CWENO reconstructions exploiting the positive features of CWENO to obtain a high-order SL scheme that is more efficient than the one in [27], which is based on WENO.

The results obtained in this study have been collected in a paper [28] and constitute part of the novelties of our research.

3.1 Representation formula for HJB equation

We will develop our theory for the HJB equation (3.1) related to a finite horizon optimal control problem. More precisely, given a compact set $A \subset \mathbb{R}^m$, we will denote by $\mathcal{A} = \{\alpha : [0,T] \to A, \text{ measurable}\}$ the set of admissible controls and, according to the usual terms adopted in this framework, we will refer to the velocity field and the source term of the model problem (2.7) respectively as the dynamics $f_D : (0,T) \times \mathbb{R}^d \times A \to \mathbb{R}^d$ and the running cost $f_C : (0,T) \times \mathbb{R}^d \times A \to \mathbb{R}$, which are both now functions that depend also on the value $\alpha(t) \in A$ of the control.

The ODE system for the characteristics then reads

$$\begin{cases} \dot{y}(s) = f_D(t-s, y(s), \alpha(s)), \quad s \in (0,t], \\ y(0) = x, \end{cases} \tag{3.2}$$

and we consider Hamiltonian given by

$$H(t,x,p) = \max_{a \in A} \{-f_D(t,x,a) \cdot p - f_C(t,x,a)\}. \tag{3.3}$$

Let us assume that:

(i) f_C, f_D are continuous and bounded. Moreover, for every $a \in A$, the functions $f_C(\cdot,\cdot,a)$, $f_D(\cdot,\cdot,a)$ are Lipschitz continous, with Lipschitz constants; independent of $a \in A$.

(ii) the initial data v_0 is Lipschitz continous and bounded.

Under Assumptions (i)-(ii), the problem (3.1) admits a unique viscosity solution v, which is Lipschitz continuous and bounded. Moreover, the Dynamic Programming Principle holds (see [10]), i.e., for any $\Delta t > 0$

$$v(t,x) = \inf_{\alpha \in \mathcal{A}} \left\{ \int_0^{\Delta t} f_C(t-s, y^\alpha(x,t;s), \alpha(s)) \mathrm{d}s + v(y^\alpha(x,t;\Delta t), t - \Delta t) \right\}, \tag{3.4}$$

where $y^\alpha(x,t;s)$ denotes the solution to (3.2) at time s and the superscript α emphasizes the dependence of the characteristics on the control.

As usual when dealing with the SL approach, the numerical scheme will be derived from the representation formula (3.4), rather than the original PDE (3.1). Thus, similarly to (2.86), the numerical scheme will involve a minimization procedure over the set of controls; in turn, each function evaluation requires to approximate the cost integral and to interpolate the solution at the previous time step at the foot $y^\alpha(x,t;\Delta t)$ of the characteristics. In this respect, the efficiency of the numerical method plays a crucial role, especially when treating the higher dimensional case.

As already stated, aiming to design a high-order numerical scheme, one could resort to the common choice of ENO interpolation techniques. In particular WENO reconstructions could be involved but at the price of their efficiency. In fact, the dependence of the WENO linear weights on the reconstruction point makes this reconstruction technique quite expensive and thus not very efficient for the discretization of (3.4). On the other hand one of the main advantages of CWENO over

the traditional WENO is that the central approach provides a reconstruction polynomial that is defined everywhere in the reconstruction cell and that can be evaluated later, with no essential extra cost, at many different reconstruction points. This is guaranteed by the independence of the linear weights from the reconstruction point. Furthermore, as shown in [103, 47, 31, 43], the CWENO approach allows to avoid the dimensional splitting procedure and this is advantageous on Cartesian meshes when the number of reconstruction points per cell is high and paves the way towards the employment of non-Cartesian meshes.

3.2 Numerical scheme

To obtain an approximate solution for (3.1), we need first to discretize the control problem in space and time. As usual we consider a space-time uniform grid of size Δx and Δt, respectively, an initial time t_0 and a final time T, so that $t_n = t_0 + n\Delta t$, $n = 0 \dots N_T$, with $N_T = \lceil \frac{T}{\Delta t} \rceil$, and $x_i = i\Delta x = (i_1\Delta x, \dots, i_d\Delta x)$, for $i \in \mathbb{Z}^d$. The Dynamic Programming Principle is thus considered on a single time step $[t_n, t_{n+1}]$, by choosing in (3.4) $x = x_i$ and $t = t_{n+1}$.

A discretization of the control function α is also required. In particular, when using a ν-stages RK scheme for (3.2), we discretize an admissible control $\alpha \in \mathcal{A}$ via a sequence $\underline{a} = (a_1, a_2, \dots, a_\nu) \in A^\nu$ representing the values $\alpha(t_n + c_k\Delta t)$ used in each RK stage. Then the foot of the characteristic emanating from the node (x_i, t_{n+1}) and corresponding to the control $\underline{a}$ is

$$y_i^n(\underline{a}) \simeq y^\alpha(x_i, t_{n+1}; \Delta t) \tag{3.5}$$

for any $n = 0, \dots, N_T - 1$, $i \in \mathbb{Z}^d$ and $\underline{a} \in A^\nu$.

Moreover, a suitable quadrature formula based on ν quadrature nodes is introduced to discretize the integral term related to the running cost,

$$C_i^n(\underline{a}) \simeq \int_0^{\Delta t} f_C(t_{n+1} - s, y^\alpha(x_i, t_{n+1}; s), \alpha(s))\, \mathrm{d}s, \tag{3.6}$$

for any $n = 0, \dots, N_T - 1$, $i \in \mathbb{Z}^d$ and $\underline{a} \in A^\nu$.

We approximate the solution $v(x,t)$ of (3.1) by a discrete function $u_i^n \simeq v(x_i, t_n)$ defined on the space-time grid, computed by the following iterative scheme, for $n = 0, \dots, N_T - 1$

$$\begin{cases} u_i^{n+1} = \min\limits_{\underline{a} \in A^\nu} \{ I[U^n](y_i^n(\underline{a})) + C_i^n(\underline{a}) \} & i \in \mathbb{Z}^d, \\ u_i^0 = v_0(x_i), & i \in \mathbb{Z}^d, \end{cases} \tag{3.7}$$

where $I[U^n](x)$ denotes the spatial reconstruction at $x \in \mathbb{R}^d$ of the numerical solution $U^n = \{u_i^n\}_{i \in \mathbb{Z}^d}$.

The building blocks composing the scheme (3.7) are same as the ones introduced in Chapter 2, with the only differences given by the possible explicit dependence on time and space, and the crucial dependence on the control function. In Figure 3.1 the dependence on the control is emphasized by the presence of multiple characteristic lines emanating from the same node; these curves might no longer be straight lines

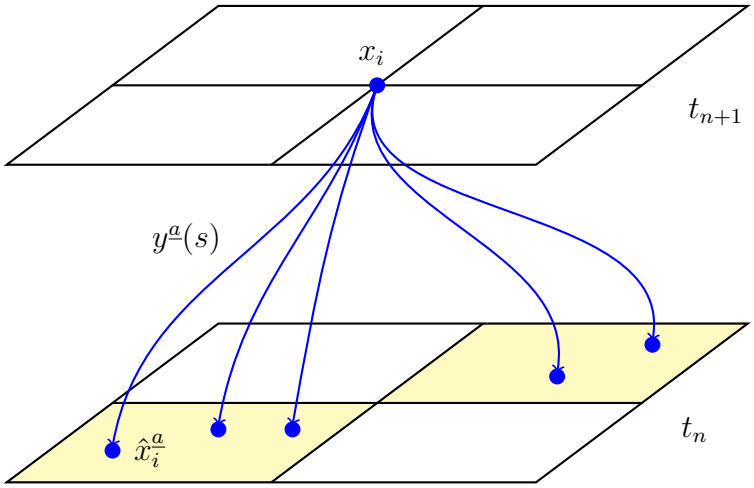

Figure 3.1: Sketch of the Semi-Lagrangian scheme for HJB equation (3.1). To update the solution at (x_i, t_{n+1}), a set characteristic curves $y^{\underline{a}}(s)$ are tracked backward to some points $(\hat{x}_i^{\underline{a}}, t_n)$ depending on the value of $\underline{a}$. The update is performed minimizing the sum of the interpolation in the feet and the integral of the cost function evaluated along the line $y^{\underline{a}}(s)$, according to (3.7). Note that for the problem (3.1) the characteristics might no longer be straight lines.

due to the possibile explicit dependence of the Hamiltonian function, in particular of the dynamics, on space and time.

Going into the details of the scheme, for a given $\underline{a} \in A^\nu$, $y_i^n(\underline{a})$ we rewrite the ν-stage Runge-Kutta (RK) method used for the computation as follows:

$$\begin{aligned} y_i^n(\underline{a}) &= x_i + \Delta t \sum_{k=1}^{\nu} b_k K_k(\underline{a}), \\ K_k(\underline{a}) &= f_D\left(t_{n+1} - c_k \Delta t, X_k(\underline{a}), a_k\right), \qquad k = 1, \dots, \nu, \\ X_k(\underline{a}) &= x_i + \Delta t \sum_{j=1}^{k-1} A_{kj} K_j(\underline{a}), \qquad k = 1, \dots, \nu, \end{aligned} \tag{3.8}$$

where b_k,c_k,A_{kj} are the coefficients of the Butcher tableau defining the explicit RK method used to solve (3.2).

Similarly, with the introduction of the control function, the computation of $C_i^n(\underline{a})$ can be rewritten as:

$$\begin{aligned} &\int_0^{\Delta t} f_C(t_{n+1} - s, y(x_i, t_{n+1}; s), \alpha(s)) ds \\ \approx &\Delta t \sum_k w_k f_C(t_{n+1} - \xi_k \Delta t, y(x_i, t_{n+1}; \xi_k \Delta t), \alpha(\xi_k \Delta t)) \\ \approx &\Delta t \sum_k w_k f_C(t_{n+1} - \xi_k \Delta t, \tilde{y}(x_i, t_{n+1}; \xi_k \Delta t), \alpha(\xi_k \Delta t)) \end{aligned} \tag{3.9}$$

where ξ_k, w_k are the nodes and the weights of the quadrature rule, respectively, and $\tilde{y}(x_i, t_{n+1}; \xi_k \Delta t)$ approximates the numerical solution of $y(x_i, t_{n+1}; s)$ for $s \in [0, \Delta t]$.

We have already discussed in the previous chapter the balance between the orders of accuracy of the approximations introduced in (3.9). We will see in the numerical

examples that with the choices (2.43), (2.44), (2.45) for the RK methods we will be allowed to choose a quadrature rule such that $\xi_k = c_k$ and thus replace in the last approximation of (3.9), $\tilde{y}(x_i, t_{n+1}; \xi_k \Delta t)$ with the stage values of the RK scheme X_k and $\alpha(\xi_k \Delta t)$ with a_k, for $k = 1, \dots, \nu$, without affecting the global accuracy of the scheme. For different choices, one could always resort to the NCEs recalled in Section 2.3.2.

Summarizing, denoting with $\mathcal{F}(\underline{a}) = I[u^n](y_i^n(\underline{a})) + C_i^n(\underline{a})$ the function to be minimized in (3.7), to compute the solution of (3.7), one has to find the optimum discrete control $\underline{a}^* = \arg\min \mathcal{F}(\underline{a})$ over $\underline{a} \in A^\nu$ and set $u_i^{n+1} = \mathcal{F}(\underline{a}^*)$.

Thus, the minimization problem constitutes another important building block of the scheme. Since in this research we are not focused on the minimization procedure, our aim is to combine different techniques to first, find the optimum on a coarser grid, and then refine the search imposing a very strong threshold for detecting the final optimum. In particular, we employ tabulation on the points $\hat{\underline{a}}_j = (a_j, a_j, \dots, a_j) \in A^\nu$ for a_j in a coarsened grid in A, followed by a Nelder-Mead algorithm, adjusted so that no vertex of the simplices could exit the compact set A.

The core of this work then lays in the following argument: at each step of the minimization we need to evaluate the reconstruction at the feet of the characteristics $y_i^n(\underline{a})$ for each node x_i and for each discrete control $\underline{a}$, thus an efficient and accurate reconstruction operator is crucial to obtain a robust and fast numerical scheme. The typical non-smoothness of the solutions of (3.1) discourages the use of standard high-order interpolation, since they might give rise to oscillations in the numerical solution, and motivates the use of an essentially non-oscillatory approach. Since during the minimization we expect to evaluate the reconstruction in a cell in a lot of points we will investigate the application of CWENO interpolation, in opposition to the traditional WENO one.

The construction of the WENO interpolator has been recalled in Section 2.3.4, in what follows the procedure for CWENO, and CWENOZ, is recalled following the presentation of [43, 42], and modified to be applied to reconstructions from point values.

3.3 CWENO reconstruction

As already pointed out for the WENO reconstruction we work in a finite-difference setting, rather than a finite-volume one, starting from node values in place of cell averages. Nonetheless, we will still be able to exploit the general theorems for the accuracy of the CWENO and CWENOZ reconstructions on smooth solutions proven in [43, 42].

For a point $x \in \mathbb{R}^d$, let Ω be the grid cell containing it and $\mathcal{S}_\Omega$ be the set of its vertices. In order to achieve better than first-order accuracy, we need to consider stencils $\mathcal{S} \supset \mathcal{S}_\Omega$. We associate to any $\mathcal{S}$ a polynomial $P_{\mathcal{S}}^{(r')}(x) \in \mathbb{P}_d^{r'}$ which interpolates the data in $\mathcal{S}$, i.e., such that $P_{\mathcal{S}}^{(r')}(x_i) = v_i \ \forall i \in \mathcal{S}$.

In order to overcome the well-know issues arising from the enlarging of the stencils, a non-linear selecting and blending procedure is performed, similarly to the WENO case, but taking also into account the high-order polynomial interpolating v

in the whole stencil $\mathcal{S}$. In opposition to WENO, the use of this high-order polynomial together with the usual set of lower-order ones, allow to produce an essentially non-oscillatory reconstruction polynomial for the cell Ω that can be evaluated at any point in $x \in \Omega$ with negligible cost. About the non-linear procedure that allow to select or blend polynomials, this relies on the same oscillation indicators $\mathrm{OSC}[P]$ introduced for WENO in Section 2.3.4.

We recall now the definition of the CWENO and CWENOZ reconstructions given in [42].

Definition 3.1. *Given a stencil $\mathcal{S}_{opt}$, including $\mathcal{S}_\Omega$, let $P_{opt} \in \mathbb{P}_n^G$ (optimal polynomial) be the polynomial of degree G, associated to $\mathcal{S}_{opt}$. Further, let $P_1, P_2, ..., P_m$ be a set of $m \geq 1$ polynomials of degree g with $g < G$, associated to substencil $\mathcal{S}_k$ such that $\mathcal{S}_\Omega \subset \mathcal{S}_k \subset \mathcal{S}_{opt}$ $\forall k = 1, ..., m$. Let also $\{d_k\}_{k=0}^m$ be a set of strictly positive real coefficients such that $\sum_{k=0}^m d_k = 1$.*

The CWENO *and* CWENOZ *operators compute a reconstruction polynomial*

$$\begin{aligned} P_{rec}^{CW} &= \mathsf{CWENO}(P_{opt}, P_1, ..., P_m) \in \mathbb{P}_n^G, \\ P_{rec}^{CWZ} &= \mathsf{CWENOZ}(P_{opt}, P_1, ..., P_m) \in \mathbb{P}_n^G, \end{aligned} \tag{3.10}$$

as follows:

1. *First, introduce the polynomial P_0 defined as*

$$P_0(x) = \frac{1}{d_0}\left(P_{opt}(x) - \sum_{k=1}^m d_k P_k(x)\right) \in \mathbb{P}_n^G; \tag{3.11}$$

2. *compute suitable regularity indicators*

$$\mathrm{OSC}_0 = \mathrm{OSC}[P_{opt}], \quad \mathrm{OSC}_k = \mathrm{OSC}[P_k],\ k \geq 1; \tag{3.12}$$

3. *compute the non-linear coefficients $\{\omega_k\}_{k=0}^m$ or $\left\{\omega_k^Z\right\}_{k=0}^m$ as*

   (a) CWENO *operator: for $k = 0, ..., m$,*

$$\alpha_k = \frac{d_k}{(\mathrm{OSC}_k + \epsilon)^l}, \quad \omega_k^{CW} = \frac{\alpha_k}{\sum_{i=0}^m \alpha_i}, \tag{3.13}$$

   (b) CWENOZ *operator: for $k = 0, ..., m$,*

$$\alpha_k^Z = d_k\left(1 + \left(\frac{\tau}{\mathrm{OSC}_k + \epsilon}\right)^l\right), \quad \omega_k^{CWZ} = \frac{\alpha_k^Z}{\sum_{i=0}^m \alpha_i^Z}, \tag{3.14}$$

   where ϵ is a small positive quantity, $l \geq 1$, and, in the case of CWENOZ*, τ is a global smoothness indicator;*

4. *finally, define the reconstruction polynomial as*

$$P_{rec}^{CW}(x) = \sum_{k=0}^m \omega_k^{CW} P_k(x) \in \mathbb{P}_n^G, \tag{3.15}$$

$$P_{rec}^{CWZ}(x) = \sum_{k=0}^m \omega_k^{CWZ} P_k(x) \in \mathbb{P}_n^G. \tag{3.16}$$

Note that the reconstruction polynomial defined in (3.15) and in (3.16) can be evaluated at any point in the computational cell Ω at a very low computational cost and this happens because the reconstruction procedure, in both cases, starts with the definition of the linear coefficients $\{d_k\}_{k=0}^m$ that do not depend on the reconstruction point. Therefore, the non-linear coefficients given by (3.13) and (3.14) are computed once per cell and not once per reconstruction point, as in the standard WENO. We also remark that, since every polynomial considered in the reconstruction procedure is required to satisfy the interpolation constraint on $\mathcal{S}_\Omega$, then also P_{rec}^{CW} and P_{rec}^{CWZ} satisfy the same constraint on the vertices of Ω.

The accuracy and non-oscillatory properties of CWENO and CWENOZ schemes are guaranteed by the dependence of their non-linear weights (3.13) and (3.14) on the regularity indicators OSC_k. On smooth data, the non-linear weights are driven sufficiently close to the optimal ones, so that $P_{\text{rec}} \approx P_{\text{opt}}$ and the reconstruction reaches the optimal order of accuracy $G+1$. On the other hand, when a discontinuity is present in $\mathcal{S}_{\text{opt}}$, both $\text{OSC}_0 \asymp 1$ and at least one $\text{OSC}_{\hat{k}} \asymp 1$ for some $\hat{k} \in \{1, ..., m\}$. Then the formulas (3.13) and (3.14) for the non-linear weights will ensure that $\omega_0 \approx 0$ and $\omega_{\hat{k}} \approx 0$ for all $\hat{k}$ such that $P_{\hat{k}}$ would bring oscillations in the reconstruction. The reconstruction polynomial will then be a linear combination of all polynomials of degree g that are not affected by the discontinuity; the accuracy of the reconstruction thus reduces to $g+1$, but spurious oscillations would be tamed.

3.3.1 One spatial dimension

Let us describe the reconstruction for any point x in the cell $\Omega = [x_i, x_{i+1}]$, so that $\mathcal{S}_\Omega = \{i, i+1\}$. We consider $\mathcal{S}_{\text{opt}} = \{i-1, i, i+1, i+2\}$ and introduce the optimal cubic polynomial $P_{\text{opt}}(x) = Q(x) = \sum_{j=0}^3 z_j[(x-x_i)/\Delta x]^j$ that interpolates the data $u_{i-1}, u_i, u_{i+1}, u_{i+2}$ at the nodes $x_{i-1}, x_i, x_{i+1}, x_{i+2}$. Next, we consider two parabolas $P_L(x)$ and $P_R(x)$ that interpolate only the data u_{i-1}, u_i, u_{i+1} and, respectively, u_i, u_{i+1}, u_{i+2}.

The reconstruction operators thus compute

$$\begin{aligned} P_{\text{rec}}^{CW} &= \mathsf{CWENO}(Q, P_L, P_R) \in \mathbb{P}_1^3, \\ P_{\text{rec}}^{CWZ} &= \mathsf{CWENOZ}(Q, P_L, P_R) \in \mathbb{P}_1^3. \end{aligned} \tag{3.17}$$

For any polynomial, its oscillation indicator is defined as in (2.70), namely

$$\text{OSC}[P] = \sum_{\alpha \geq 2} \Delta x^{2\alpha-3} \int_{x_i}^{x_{i+1}} \left(\frac{d^{(\alpha)} P}{dx^\alpha} \right)^2 dx. \tag{3.18}$$

Let $P(x) = \sum_{j=0}^3 z_j[(x-x_i)/\Delta x]^j$ be a polynomial of degree up to 3. Its indicator (3.18) can be written as a quadratic form of its coefficients given by

$$\text{OSC}[P] = \frac{1}{\Delta x^2}(4z_2^2 + 12 z_2 z_3 + 48 z_3^2)$$

or equivalently, denoting with $\vec{z}$ the vector of coefficients,

$$\text{OSC}[P] = \frac{1}{\Delta x^2} \vec{z}^T M \vec{z}, \text{ with } \quad M = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 \\ 0 & 0 & 6 & 48 \end{pmatrix}. \tag{3.19}$$

Denoting by U the vector of data $(u_{i-1}, u_i, u_{i+1}, u_{i+2})^T$, since the coefficients of the polynomial linearly depends on U, we can express the regularity indicators also as a quadratic form on the data being interpolated. More precisely, the vector of coefficients $\vec{z}$ can be expressed as $\vec{z} = V^{-1}U$, where V^{-1} is the inverse of the Vandermonde matrix. Thus, in the one-dimensional case, one obtains that the matrices of the quadratic forms expressed in terms of U scale again globally as $1/\Delta x^2$:

$$\mathrm{OSC}[Q] = \frac{1}{\Delta x^2} U^T A_Q U, \quad \mathrm{OSC}[P_k] = \frac{1}{\Delta x^2} U^T A_k U \quad (k = L, R). \tag{3.20}$$

In our case, we get

$$A_Q = \begin{pmatrix} 4/3 & -7/2 & 3 & -5/6 \\ -7/2 & 10 & -19/2 & 3 \\ 3 & -19/2 & 10 & -7/2 \\ -5/6 & 3 & -7/2 & 4/3 \end{pmatrix},$$

$$A_L = \begin{pmatrix} 1 & -2 & 1 & 0 \\ -2 & 4 & -2 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad A_R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & -2 & 4 & -2 \\ 0 & 1 & -2 & 1 \end{pmatrix}.$$

The CWENO reconstruction applied in the numerical tests of this paper is then defined by choosing linear coefficients $d_L = d_R = 1/8$ and $d_0 = 3/4$, $l = 2$ and $\epsilon = \Delta x^2$ in the above construction.

For the optimal definition of the CWENOZ reconstruction, in order to exploit the results of [42], we need to study the Taylor expansions of the indicators centered at the point $x_0 = (x_i + x_{i+1})/2$, namely the center of the reconstruction cell Ω. They are given by

$$\begin{aligned} \mathrm{OSC}[P_L] &= B - u''(x_0)u'''(x_0)\Delta x^3 + \frac{1}{4}u'''(x_0)^2\Delta x^4 + \mathcal{O}(\Delta x^5), \\ \mathrm{OSC}[P_R] &= B + u''(x_0)u'''(x_0)\Delta x^3 + \frac{1}{4}u'''(x_0)^2\Delta x^4 + \mathcal{O}(\Delta x^5), \\ \mathrm{OSC}[P_L] &= B + \frac{13}{12}u'''(x_0)^2\Delta x^4 + \mathcal{O}(\Delta x^5), \end{aligned} \tag{3.21}$$

where $B = u''(x_0)^2\Delta x^2$. Thus, defining

$$\tau = \Big|2\mathrm{OSC}[Q] - \mathrm{OSC}[P_L] - \mathrm{OSC}[P_R]\Big|, \tag{3.22}$$

all terms up to $\mathcal{O}(\Delta x^3)$ cancel. Note that this is the optimal definition of τ since it is never possible to cancel all the $\mathcal{O}(\Delta x^4)$ terms in a convex combination of the three indicators. The hypotheses of Theorem 24 in [42] hold and we can guarantee the optimal order of accuracy of the CWENOZ reconstruction for $l = 2$.

Remark 3.1. *When a* WENO *interpolator is employed in the minimization algorithm, the oscillation indicators of the polynomials associated to each cell could be precomputed and stored, while the non-linear coefficients ω_k must be computed for each reconstruction point queried during the minimization. Using* CWENO*, instead, also the ω_k can be computed only once per cell, leading to saving in computational cost when multiple reconstructions per cell are needed during the minimization.*

3.3.2 Two spatial dimensions

In two space dimensions, we will not rely on dimensional splitting. Such an approach would in fact nullify the advantages of a fast evaluation of the reconstruction polynomial on a large number of arbitrary points in the reconstruction cell. Our reconstruction will instead generate a polynomial of two variables, like it is done in [103, 31] for Cartesian meshes or in general mesh settings [129, 47, 8].

Let Ω be the cell of the Cartesian grid containing the reconstruction point x. The setup of the stencils for the reconstruction is illustrated in Figure 3.2: the cell Ω, with corners (x_i, y_j) and (x_{i+1}, y_{j+1}), is hatched in red and the reconstruction stencil is shaded in gray.

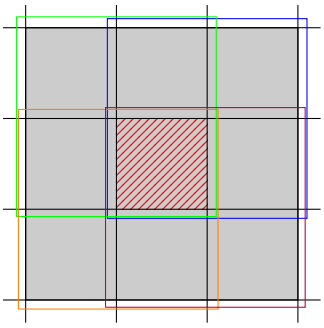

Figure 3.2: Stencils of the two-dimensional CWENO and CWENOZ reconstructions. The red hatched region represents the cell $\Omega_{i,j}$ in which we compute the reconstruction. The vertices of $\mathcal{S}_{\text{opt}}$ are enclosed in the grey shaded region, while the stencils for the low degree polynomials are enclosed in the coloured squares: blue, green, orange and purple, respectively for the north-east, north-west, south-west and south-east polynomial.

The reconstruction operators compute

$$\begin{aligned} P_{\text{rec}}^{CW} &= \mathsf{CWENO}(Q^{(3)}; Q_{\text{ne}}^{(2)}, Q_{\text{se}}^{(2)}, Q_{\text{sw}}^{(2)}, Q_{\text{nw}}^{(2)}) \in \mathbb{P}_1^3 \otimes \mathbb{P}_1^3, \\ P_{\text{rec}}^{CWZ} &= \mathsf{CWENOZ}(Q^{(3)}; Q_{\text{ne}}^{(2)}, Q_{\text{se}}^{(2)}, Q_{\text{sw}}^{(2)}, Q_{\text{nw}}^{(2)}) \in \mathbb{P}_1^3 \otimes \mathbb{P}_1^3, \end{aligned} \tag{3.23}$$

with the optimal polynomial set to the bicubic polynomial interpolating the 16 data points in the square with corners (x_{i-1}, y_{j-1}) and (x_{i+2}, y_{j+2}) enclosed in the grey shaded region of Fig 3.2. The other four polynomials are the biquadratic polynomials that interpolate the four 3×3 substencils in the north-east, south-east, south-west and north-west directions, as illustrated in Figure 3.2, named $\mathcal{S}_{\text{ne}}$, $\mathcal{S}_{\text{nw}}$, $\mathcal{S}_{\text{sw}}$ and $\mathcal{S}_{\text{se}}$, respectively. In particular, the blue box with vertices (x_i, y_j) and (x_{i+2}, y_{j+2}) encloses $\mathcal{S}_{\text{ne}}$, the green box with vertices (x_{i-1}, y_j) and (x_{i+1}, y_{j+2}) encloses $\mathcal{S}_{\text{nw}}$, the orange box with vertices (x_{i-1}, y_{j-1}) and (x_{i+1}, y_{j+1}) encloses $\mathcal{S}_{\text{sw}}$ and the purple box with vertices (x_i, y_{j-1}) and (x_{i+2}, y_{j+1}) encloses $\mathcal{S}_{\text{se}}$. The basis for bicubic and biquadratic polynomials are defined by tensorization and are thus respectively given by

$$\begin{aligned} \mathcal{B}_3 &= \{1, \hat{x}, \hat{y}, \hat{x}^2, \hat{x}\hat{y}, \hat{y}^2, \hat{x}^3, \hat{x}^2\hat{y}, \hat{x}\hat{y}^2, \hat{y}^3, \hat{x}^3\hat{y}, \hat{x}^2\hat{y}^2, \hat{x}\hat{y}^3, \hat{x}^3\hat{y}^2, \hat{x}^2\hat{y}^3, \hat{x}^3\hat{y}^3\}, \\ \mathcal{B}_2 &= \{1, \hat{x}, \hat{y}, \hat{x}^2, \hat{x}\hat{y}, \hat{y}^2\}, \end{aligned}$$

where $\hat{x} = (x - x_i)/\Delta x$ and $\hat{y} = (y - y_j)/\Delta y$.

The oscillation indicators are defined as

$$\mathrm{OSC}[P] = \sum_{|\vec{\alpha}|\geq 2} \Delta^{2|\vec{\alpha}|-4} \int_{\Omega_{i,j}} \left[\frac{\partial P}{\partial x^{\alpha_1}\partial y^{\alpha_2}}\right]^2 \Delta x \Delta y \tag{3.24}$$

where Δ is the diameter of the cell $\Omega_{i,j}$. These indicators are inspired to the classical ones of [72], but the powers of Δ appearing in (3.24) have been adjusted for the expected regularity of the solution to Hamilton-Jacobi equations. In particular the scaling $\Delta^{2|\vec{\alpha}|-4}$ is a generalization to two space dimensions of the choice of [75]; it is introduced so that $\mathrm{OSC}[P] \asymp 1$ for functions with discontinuous first derivative and $\mathrm{OSC}[P] = \mathcal{O}(\Delta^2)$ on regular solutions. A detailed discussion of this scaling can be found in [52].

Denoting with $\vec{z}$ the vector of coefficients along the basis $\mathcal{B}_3$ of a generic polynomial in two variables of degree up to 3, and considering a uniform grid of size Δx, its oscillation indicator can be expressed as

$$\mathrm{OSC}[P] = \frac{1}{\Delta x^2}\vec{z}^T M \vec{z} \tag{3.25}$$

for a 16×16 matrix M, which is reported in the Appendix. We point out that in the case of Cartesian but uneven grids, there would still be a global factor inversely proportional to the local grid size, but matrix M would also depend on the aspect ratios and on the ratios of sizes of nearby cells.

Consider now the stencil of the reconstruction given by $\{(x_{i-1+k}, y_{j-1+l})\}_{\substack{l=0,\dots,3\\k=0,\dots,3}}$ and denote by U the corresponding vector of values $\{u_{i-1+k,j-1+l}\}_{\substack{l=0,\dots,3\\k=0,\dots,3}}$ with a prescribed ordering, e.g., lexicographic. Thanks to the linear relation among the coefficients $\vec{z}$ and the data vector U, the regularity indicators can be expressed in the form

$$\mathrm{OSC}[Q] = \frac{1}{\Delta x^2}U^T A_Q U, \quad \mathrm{OSC}[Q_k] = \frac{1}{\Delta x^2}U^T A_k U \quad (k = \mathrm{ne}, \mathrm{se}, \mathrm{sw}, \mathrm{nw}). \tag{3.26}$$

The matrices A_Q and A_k will in general depend on the local aspect ratio of cells and on the neighbours size ratio. Their entries are reported in the Appendix for the case of a uniform Cartesian grid.

The CWENO reconstruction is then defined by choosing linear coefficients $d_k = 1/16$ and $d_0 = 1 - \sum_k d_k$ with $k \in \{\mathrm{ne}, \mathrm{se}, \mathrm{sw}, \mathrm{nw}\}$, $l = 2$ and $\epsilon = \Delta x^2$.

The CWENOZ reconstruction is computed choosing $l = 2$ and

$$\tau = \left|4\mathrm{OSC}[Q^{(3)}] - \mathrm{OSC}[Q^{(2)}_{\mathrm{ne}}] - \mathrm{OSC}[Q^{(2)}_{\mathrm{se}}] - \mathrm{OSC}[Q^{(2)}_{\mathrm{sw}}] - \mathrm{OSC}[Q^{(2)}_{\mathrm{nw}}]\right| \tag{3.27}$$

in order to guarantee the optimal order of accuracy of the CWENOZ reconstruction, see [42, Theorem 24]. In fact, computing the Taylor expansions of the indicators centered at the center of the cell Ω, one gets, for smooth enough data,

$$\begin{aligned}
\mathrm{OSC}[Q^{(2)}_{\mathrm{ne}}] &= B + 2u_{xx}u_{xxx}\Delta x^3 + 2u_{yy}u_{yyy}\Delta x^3 + \mathcal{O}(\Delta x^4),\\
\mathrm{OSC}[Q^{(2)}_{\mathrm{nw}}] &= B - 2u_{xx}u_{xxx}\Delta x^3 + 2u_{yy}u_{yyy}\Delta x^3 + \mathcal{O}(\Delta x^4),\\
\mathrm{OSC}[Q^{(2)}_{\mathrm{sw}}] &= B - 2u_{xx}u_{xxx}\Delta x^3 - 2u_{yy}u_{yyy}\Delta x^3 + \mathcal{O}(\Delta x^4),\\
\mathrm{OSC}[Q^{(2)}_{\mathrm{se}}] &= B + 2u_{xx}u_{xxx}\Delta x^3 - 2u_{yy}u_{yyy}\Delta x^3 + \mathcal{O}(\Delta x^4),\\
\mathrm{OSC}[Q^{(3)}] &= B + \mathcal{O}(\Delta x^4),
\end{aligned} \tag{3.28}$$

where $B = (u_{xx}^2 + u_{xy}^2 + u_{yy}^2)\Delta x^2$. Therefore, the coefficients chosen for the definition of τ in (3.27), cancel all terms up to $\mathcal{O}(\Delta x^3)$, thus ensuring that the hypotheses of [42, Theorem 24] hold.

Remark 3.2. *Using a* WENO*, dimensionally split, interpolator during the minimization process allows only the oscillation indicators for the x direction to be precomputed, while ω values for x-direction, low degree polynomials,* OSC *and ω values for the y-direction must be recomputed from scratch for each new interpolation point. Using the present* CWENO *scheme, instead the non-linear coefficients ω for any point in a given cell can be precomputed and stored. Despite the larger cost of computing the oscillation indicators in the 2d setting, this is expected to be faster than* WENO *when employed inside a minimization algorithm.*

Remark 3.3. *The present multi dimensional* CWENO *reconstruction, since it does not rely on dimensional splitting, can also be extended to non-uniform and non-Cartesian meshes. The ideas of [47] for stencil selection in the finite volume context may in fact be adapted to the present case. In fact, in order to precompute the oscillation indicators and the non-linear weghts ω, one has to fix the definition of the stencils used for interpolation. Compared to the Cartesian case, having no a-priori knowledge of the number of neighbours, leads in general to stencils with different cardinalities, for which a least-squared approach can be employed.*

3.4 Convergence

The convergence analysis will be carried out in one space dimension, and follow the guidelines of [56, 27], as already retraced in Section 2.2.3. Assume that equation (3.1) is posed on $\mathbb{R}$ in the simplified form

$$\begin{cases} v_t(t,x) + H(Dv(t,x)) = 0 & t, x \in (0,T) \times \mathbb{R}, \\ v(0,x) = v_0(x), & x \in \mathbb{R}. \end{cases} \tag{3.29}$$

Also assume that the Hamiltonian function H is a $W^{2,\infty}$ function and that

$$H''(p) \geq m_H > 0. \tag{3.30}$$

The convexity assumption (3.30) implies, by the Fenchel duality formula, that $H(\cdot)$ can always be written in the form

$$H(\nabla v(t,x)) = \sup_{a \in \mathbb{R}} \{a\, \nabla v(t,x) - H^*(a)\},$$

where H^* denotes the Legendre transform of H. Moreover, the supremum is a maximum, and its computation can be limited to a suitable compact set A, so that problem (3.29) can be recast in the case of a Hamiltonian of the form (3.3), by setting $f_D(t,x,a) = -a$ and $f_C(t,x,a) = H^*(a)$.

In order to prove convergence we need to verify that the key assumption (2.91) is satified by the CWENO interpolant. Recalling the requirements, we need to verify that for any Lipschitz continuous function $v(x)$, once defined the sequence $V =$

$\{v_j\}_j = \{v(x_i)\}_i$, the interpolation operator $I_r[V]$ satisfies $I_r[V](x_k) = v(x_k)$ and, for some constant $C_r < 1$:

$$|I_r[V](x) - I_1[V](x)| \leq C_r \max_{x_k \in B(x)} |v_{k+1} - 2v_k + v_{k-1}|, \tag{3.31}$$

where $I_1[V]$ and $B(x)$ are defined as in Section 2.2.3, the first one being the piecewise linear interpolator on the sequence V, and the second one being the interval $B(x) = (x - h_-\Delta x, x + h_+\Delta x)$ containing the stencil of the reconstruction $I_r[V](x)$. For example, in our case the third-order CWENO reconstruction is performed taking two nodes on the left and two nodes on the right of the point x, so that $h_- = h_+ = 2$.

Since CWENO interpolation, similarly to WENO is performed by taking a convex combination of polynomials which satisfy (2.91) up the degree 5 for the partial polynomials (and therefore, up to the degree 9 for the global interpolant), we just need to prove that the all the functions appearing in the convex combination defining the CWENO interpolant satisfy (2.91).

In the case of CWENO, once set $d_L = d_R = d$ and recast P_0 as

$$P_0(x) = \frac{1}{1-2d}(Q(x) - dP_L(x) - dP_R(x)),$$

we estimate the left-hand side of (3.31) as

$$\begin{aligned} |I_r - I_1| &= |\omega_0 P_0 + \omega_L P_L + \omega_R P_R - I_1| \\ &= |\omega_0(P_0 - I_1) + \omega_L(P_L - I_1) + \omega_R(P_R - I_1)| \\ &\leq \max\{|P_0 - I_1|, |P_L - I_1|, |P_R - I_1|\}, \end{aligned}$$

where the identity $\omega_0 + \omega_L + \omega_R = 1$ has been used twice. As it has been proved in [56], both P_L and P_R satisfy (3.31). It remains then to check that the same property holds for P_0. We have:

$$\begin{aligned} |P_0 - I_1| &= \left|\frac{1}{1-2d}(Q - dP_L - dP_R) - I_1\right| \\ &= \left|\frac{1}{1-2d}(Q - dP_L - dP_R + (2d-1)I_1)\right| \\ &= \left|\frac{1}{1-2d}\Big((Q - I_1) - d(P_L - I_1) - d(P_R - I_1)\Big)\right| \\ &\leq \frac{1}{1-2d}|Q - I_1| + \frac{d}{1-2d}|P_L - I_1| + \frac{d}{1-2d}|P_R - I_1|. \end{aligned}$$

Denoting now by C_r the constant appearing in (3.31) for the specific case of an interpolation of degree r, we finally obtain an estimate in the form

$$|P_0 - I_1| \leq \left(\frac{1}{1-2d}C_3 + \frac{2d}{1-2d}C_2\right) \max_{x_k \in U(x)} |v_{k+1} - 2v_k + v_{k-1}|.$$

As proved in [56], $C_2 = 1/8$ and $C_3 \approx 0.2533$; then, it turns out that (3.31) is satisfied for $d \lesssim 0.332$, and therefore, with this additional condition, all the assumptions of Theorem 2.8 are satisfied, and the scheme (3.7) converges to the viscosity solution of (3.29).

We point out that our choice of $d_0 = 3/4$, i.e. $d = d_L = d_R = 1/8$ in the CWENO and CWENOZ reconstructions employed in this paper fully satisfies the above requirement.

3.5 Numerical tests

Here, we present some numerical tests in order to assess the performance of the schemes proposed in the previous sections. In particular, we focus on the expected accuracy of the schemes, considering the L^1-norm of the error, and the computational times, reported in seconds. In particular we will compare the SL schemes using the CWENO and CWENOZ reconstructions of Section 3.3 and those employing the dimensionally-split WENO approach of [27].

In order to account for bounded domains, we consider either periodic boundary conditions or extrapolation technique, depending on the test case at hand. In particular, with extrapolation we mean that, for instance in 1d, we resort to a linear interpolation using the first two internal values adjacent to the boundary to compute the discretized value u_i in a ghost point.

It is noteworthy that in all the numerical tests where extrapolation is applied, the solution has a compact support, or the boundary of the numerical domain is entirely of outflow type. In both cases, these conditions ensure no loss in accuracy in the numerical treatment of the boundary. In all tests, unless otherwise stated, we employ an extrapolation technique. For more general boundary conditions, we refer to [20] for approximations of Neumann-type boundary conditions and to [17] for a second-order accurate treatment of Dirichlet boundary conditions.

Although the convergence analysis requires a relation $\Delta x = \mathcal{O}(\Delta t^2)$ between the time and space steps, in practice, smaller time steps are also allowed. Heuristically, assuming exact minimization, and supposing a third-order time discretization is used, then the consistency error is given by

$$\mathcal{T}(\Delta t, \Delta x) = \mathcal{O}\left(\Delta t^3 + \frac{\Delta x^q}{\Delta t}\right),$$

where q is the order of accuracy of the interpolation in space. For smooth solutions, the choice $\Delta t = \mathcal{O}\left(\Delta x^{q/4}\right)$ optimizes this error and leads to a numerical convergence rate of $3q/4$.

In particular, choosing $q = 4$ (i.e., a cubic reconstruction), we obtain

$$\mathcal{T}(\Delta t, \Delta x) = \mathcal{O}\left(\Delta t^3 + \frac{\Delta x^4}{\Delta t}\right),$$

which results in an overall order 3 for the choice $\Delta t = \mathcal{O}(\Delta x)$. In the particular case where the Hamiltonian is independent of (t, x), the characteristics solving problem the (3.2) are affine, and even a first-order time discretization computes them exactly, reducing the consistency error to

$$\mathcal{T}(\Delta t, \Delta x) = \mathcal{O}\left(\frac{\Delta x^q}{\Delta t}\right).$$

In this case, the larger the time step, the smaller the consistency errors.

In what follows N indicates the number of grid points per direction, so that $\Delta x \sim 1/N$. The algorithms have been implemented in C++, relying on the PETSc libraries

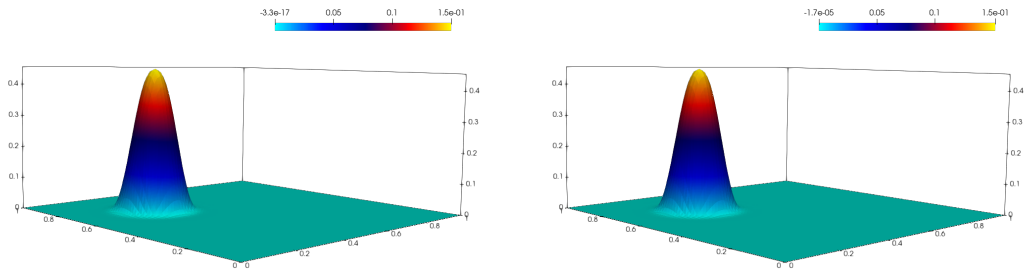

Figure 3.3: Initial condition (left) and numerical solution at $T = 1$ (right) for Test 1 computed with CWENOZ reconstruction and $N = 81$.

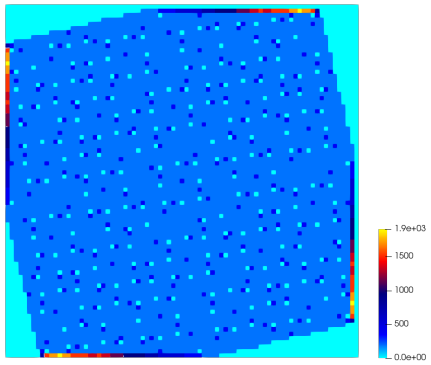

Figure 3.4: Reconstructions count for Test 1 in the final time step on a grid of 81×81 points.

[7, 6] for grid management. The tests have been performed on the cluster Galileo 100 hosted at CINECA[1], exploiting the resources assigned to ISCRA-C Projects[2].

3.5.1 Passive advection

As a first benchmark problem, we consider the uniform rotation of a scalar field in two space dimensions, expressed by the linear transport equation

$$v_t - f_D \cdot \nabla v = 0,$$

[1] https://www.hpc.cineca.it/systems/hardware/galileo100/

[2] *Surface Reconstruction with Level Set Method* (HP10CPQ93M)
Parallel Scalability of Surface Reconstruction with Level Set Method (HP10COSEJL)
Adaptive Mesh Refinement in Level Set Methods (HP10C7HWOL)

	WENO		CWENO		CWENOZ	
N	L^1-err	L^1-ord	L^1-err	L^1-ord	L^1-err	L^1-ord
21×21	$2.97e-03$		$2.98e-03$		$2.98e-03$	
41×41	$5.46e-04$	2.44	$4.88e-04$	2.61	$4.84e-04$	2.62
81×81	$1.24e-04$	2.13	$8.18e-05$	2.57	$7.85e-05$	2.62
161×161	$2.34e-05$	2.41	$1.22e-05$	2.75	$1.12e-05$	2.80
321×321	$3.92e-06$	2.58	$1.72e-06$	2.82	$1.49e-06$	2.92
641×641	$6.11e-07$	2.68	$2.27e-07$	2.92	$1.79e-07$	3.05
1281×1281	$8.87e-08$	2.78	$2.92e-08$	2.96	$2.06e-08$	3.12

Table 3.1: Errors at time $T = 1$ for Test 1.

	WENO	CWENO		CWENOZ	
grid	CPU time	CPU time	% gain	CPU time	% gain
21×21	$1.05e+00$	$7.22e-01$	31.21	$1.01e-01$	90.36
41×41	$1.10e+00$	$1.06e+00$	3.39	$7.20e-01$	34.25
81×81	$7.84e+00$	$5.42e+00$	30.87	$5.46e+00$	30.39
161×161	$6.19e+01$	$4.36e+01$	29.51	$4.37e+01$	29.35
321×321	$4.86e+02$	$3.38e+02$	30.47	$3.36e+02$	30.95
641×641	$3.87e+03$	$2.70e+03$	30.31	$2.67e+03$	30.99
1281×1281	$3.09e+04$	$2.12e+04$	31.25	$2.12e+04$	31.32

Table 3.2: CPU times for Test 1.

	WENO		CWENO		CWENOZ	
N	L^1-err	L^1-ord	L^1-err	L^1-ord	L^1-err	L^1-ord
81	$3.56e-06$		$2.24e-06$		$1.78e-06$	
161	$2.83e-07$	3.65	$1.80e-07$	3.64	$1.44e-07$	3.63
321	$2.45e-08$	3.53	$1.59e-08$	3.50	$1.30e-08$	3.47
641	$2.61e-09$	3.23	$1.96e-09$	3.02	$1.75e-09$	2.90

Table 3.3: Errors at time $T = 1$ for Test 2.

which is included in our setting by choosing in (3.3) $A = \emptyset$ and $f_C = 0$. The speed of propagation is given by the vector field

$$f_D = (-2\pi(x_2 - 0.5), 2\pi(x_1 - 0.5))$$

in the square domain $[0, 1] \times [0, 1]$. We choose as initial condition the globally $\mathcal{C}^2$ function

$$v(0, x) = v_0(r) = M\left(1 + \frac{r^3}{R^3}\left(-1 + 3\,\frac{r - R}{R}\left(1 - 2\,\frac{r - R}{R}\right)\right)\right)$$

with $r = |x - (0.3, 0.7)|$ and parameters $M = 0.15$, $R = 0.15$. Numerical solutions are computed at the final time $T = 1$, using a relationship of $\Delta t = 3\Delta x$ among the discretization steps. Since the advection is rigid, the exact solution at final time coincides with the initial condition. We point out that the characteristic lines are not affine, therefore we use the third-order RK scheme and a third-order reconstruction scheme as described in Section 2.3.1 and Section 2.3.3, respectively. Plots of the initial data and of the numerical solution are shown in Figure 3.3.

As expected, since the transport is rigid and the initial data is globally $\mathcal{C}^2$, all the schemes (WENO, CWENO and CWENOZ) yield solutions converging with the same order, see Table 3.1. However, central reconstructions are more accurate with respect to the traditional WENO one; in particular, CWENOZ achieves the best results: its errors are about half the errors of WENO. The fundamental difference among the schemes is in their computational costs, which are shown in Table 3.2. The high number of reconstructions performed in each cell (see Figure 3.4) allows the CWENO and CWENOZ to save about the 30% of time. This happens because Central WENO reconstructions compute their non-linear coefficients only once per cell instead of once per reconstruction point (see Remark 3.2). Looking at Figure 3.4, note in particular that the high number of reconstructions performed near the boundary is related to the boundedness of the domain: in fact, except for the case of periodic boundary conditions, each time a foot of a characteristic falls outside the domain, it is projected back in the first cell near the boundary in order to compute the reconstruction.

3.5.2 One-dimensional semi-concave data

This test deals with the HJ equation

$$\begin{cases} v_t(t, x) + \frac{1}{2}|\partial_x v(t, x)|^2 = 0 \\ v(0, x) = v_0(x) = \min(-\cos(\pi x/2), 0), \end{cases} \tag{3.32}$$

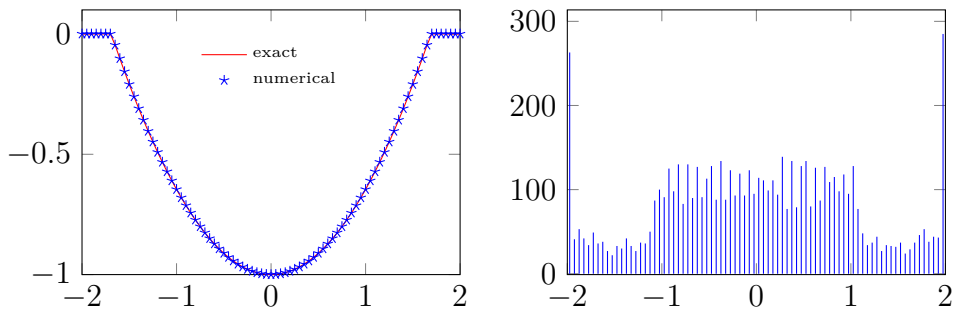

Figure 3.5: Left: the exact solution of Test 2 in red and the numerical one in blue computed with a third-order CWENOZ reconstruction and $N = 81$. Right: reconstructions count in the last time step.

	WENO	CWENO		CWENOZ	
N	CPU time	CPU time	% gain	CPU time	% gain
81	$6.19e-03$	$4.83e-03$	21.95	$4.86e-03$	21.48
161	$9.79e-03$	$8.88e-03$	9.31	$8.64e-03$	11.76
321	$2.74e-02$	$2.39e-02$	13.03	$2.39e-02$	12.84
641	$9.42e-02$	$7.65e-02$	18.81	$7.68e-02$	18.41
1281	$3.48e-01$	$2.83e-01$	18.65	$2.90e-01$	16.72
2561	$1.33e+00$	$1.10e+00$	17.15	$1.10e+00$	16.83
5121	$5.28e+00$	$4.30e+00$	18.51	$4.29e+00$	18.77
10241	$2.09e+01$	$1.68e+01$	19.42	$1.68e+01$	19.46

Table 3.4: CPU times for Test 2.

in the domain $[-2, 2]$, with homogeneous boundary conditions, which are treated using the extrapolation technique. The approximate solution is computed, at $T = 1$, with $\Delta t = 10\Delta x$, while the exact solution $v(t, x)$ is defined as follows:

$$v(t, x) = \min(0, \frac{1}{2}[a^*(t, x)]^2 + v_0(x - ta^*(t, x))), \tag{3.33}$$

where, for a given (t, x), $a^*(t, x)$ represents the optimal control, which is constant along the characteristics, and it can be computed as the solution of

$$a^*(t, x) = g(t, x, a^*(t, x)) \tag{3.34}$$

with the function g defined by

$$g(t, x, a^*(t, x)) = \begin{cases} -\frac{2}{\pi t} \arcsin \frac{2a^*(t,x)}{\pi} + \frac{x}{t} & \text{if } |a^*(t, x)| \leq \frac{\pi}{2}, \\ \frac{\pi}{2} & \text{if } a^*(t, x) > \frac{\pi}{2}, \\ -\frac{\pi}{2} & \text{if } a^*(t, x) < -\frac{\pi}{2}. \end{cases} \tag{3.35}$$

For our purposes, since the exact solution of (3.34) is not available, we compute an approximate solution by using a fixed point algorithm with initial guess $a_0^*(t, x) = \frac{1}{2}x^2 t$.

Since the characteristics are affine, in this test we have combined the third-order spatial reconstructions with first order RK time-stepping, which computes these curves exactly. Plots of the numerical and exact solutions are shown in the left panel of Figure 3.5 and L^1 errors are listed in Table 3.3. In this case too, all schemes compute approximations affected by errors of the same order but the central reconstructions turn out to be more accurate (errors are about 2/3 than the WENO case). In fact, the feet of the characteristics fall in regular regions of v at each time step, and therefore all the reconstructions are close to the optimal cubic one. Table 3.4 reports the computational costs of each scheme, showing an average gain of $16 - 19\%$ for both the Central WENO reconstructions. This is again the effect of the multiple reconstructions performed in each cell (see the right panel of Figure 3.5, in which the number of reconstructions performed in each cell during the last time step is depicted). We point out that, as in Test 1, the high number of reconstructions computed in the first and in the last cell is related to the boundary conditions. The saving is less pronounced than in the 2d setting, as expected from the comparison between remark 3.1 and remark 3.2.

3.5.3 One-dimensional eikonal equation

In this test, we consider the HJ equation

$$\begin{cases} v_t(t, x) + \frac{1}{2}|\partial_x v(t, x)|^2 - f(t, x) = 0 \\ v(0, x) = v_0(x), \end{cases} \tag{3.36}$$

with $f(t, x) = -\sin(x) + (\frac{9}{8} + \frac{t^2-3t}{2})\cos^2(x)$ and $v_0(x) = \frac{3}{2}\sin(x)$. We consider the problem in $[0, 2\pi]$ with periodic boundary conditions and compute the numerical

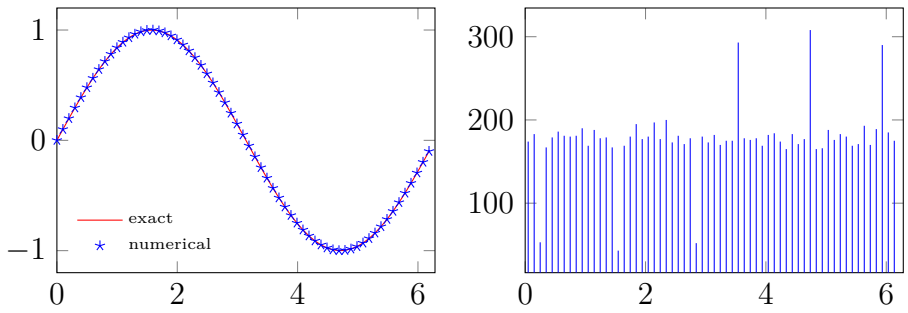

Figure 3.6: Left: the exact solution of Test 3 in red and the numerical one in blue computed with a third-order CWENOZ scheme and $N = 63$. Right: reconstructions count in the last time step.

	WENO		CWENO		CWENOZ	
N	L^1-err	L^1-ord	L^1-err	L^1-ord	L^1-err	L^1-ord
126	$1.28e-05$		$1.38e-05$		$1.43e-05$	
252	$1.54e-06$	3.05	$1.65e-06$	3.07	$1.70e-06$	3.07
503	$1.93e-07$	3.00	$2.03e-07$	3.02	$2.10e-07$	3.02
1006	$2.56e-08$	2.91	$2.63e-08$	2.95	$2.71e-08$	2.95

Table 3.5: Errors at time $T = 1$ for Test 3.

	WENO	CWENO		CWENOZ	
grid	CPU time	CPU time	% gain	CPU time	% gain
126	$5.92e-02$	$5.42e-02$	8.43	$5.64e-02$	4.81
252	$2.11e-01$	$1.94e-01$	8.04	$1.94e-01$	8.05
503	$7.97e-01$	$7.37e-01$	7.53	$7.48e-01$	6.05
1006	$3.13e+00$	$2.83e+00$	9.75	$2.87e+00$	8.45
2011	$1.19e+01$	$1.11e+01$	6.67	$1.11e+01$	6.79
4022	$4.73e+01$	$4.27e+01$	9.58	$4.21e+01$	11.03
8043	$1.80e+02$	$1.65e+02$	8.63	$1.62e+02$	10.02
16085	$6.96e+02$	$6.41e+02$	7.96	$6.38e+02$	8.35

Table 3.6: CPU times for Test 3.

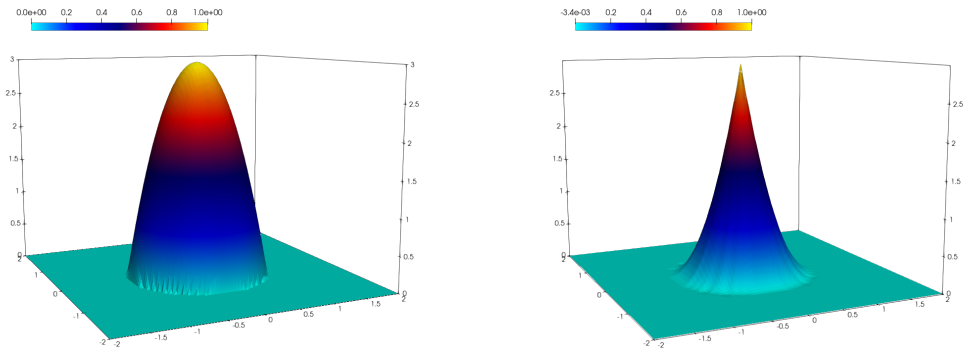

Figure 3.7: Test 4. Initial condition (left) and numerical solution computed with CWENO at $T = 1$ (right) with $N = 81$.

	WENO		CWENO		CWENOZ	
N	L^1-err	L^1-ord	L^1-err	L^1-ord	L^1-err	L^1-ord
41×41	$4.02e-02$		$3.38e-02$		$3.38e-02$	
81×81	$2.25e-02$	0.84	$1.82e-02$	0.90	$1.81e-02$	0.90
161×161	$1.15e-02$	0.96	$9.01e-03$	1.01	$8.99e-03$	1.01
321×321	$5.56e-03$	1.05	$4.10e-03$	1.14	$4.09e-03$	1.14
641×641	$2.85e-03$	0.97	$2.03e-03$	1.02	$2.02e-03$	1.02

Table 3.7: Errors at time $T = 1$ for Test 4.

solution at time $T = 0.5$, with $\Delta t = \Delta x$. This problem has a known exact solution given by $v(t, x) = (\frac{3}{2} - t)\sin(x)$.

In this case, characteristics are not affine and both the dynamic and the cost function explicitly depend on x and t. Therefore this is a specific test for the third-order accuracy of our scheme, which depends on the use of the third-order RK scheme (2.45) and on the approximation of the integral of the cost function.

Plots of the exact and of the numerical solution are shown in the left panel of Fig.3.6, while in the right panel the reconstruction count in the last time step is depicted. We point out that in this case, with periodic boundary conditions, there is no artificial increase in the first and in the last cell near the boundary.

Errors and CPU times are listed in Tables 3.5 and 3.6. As in the previous tests, due to the regularity of the exact solution, all schemes perform almost equally when compared in terms of the error. Looking at the computational costs, the Central WENO reconstructions allow for a saving of about $6 - 10\%$.

3.5.4 Two-dimensional semi-convex data

In this test, the HJ equation

$$\begin{cases} v_t(t, x) + \frac{1}{2}|\nabla v(t, x)|^2 = 0 \\ v(0, x) = v_0(x) = \max(1 - |x|^2, 0), \end{cases} \tag{3.37}$$

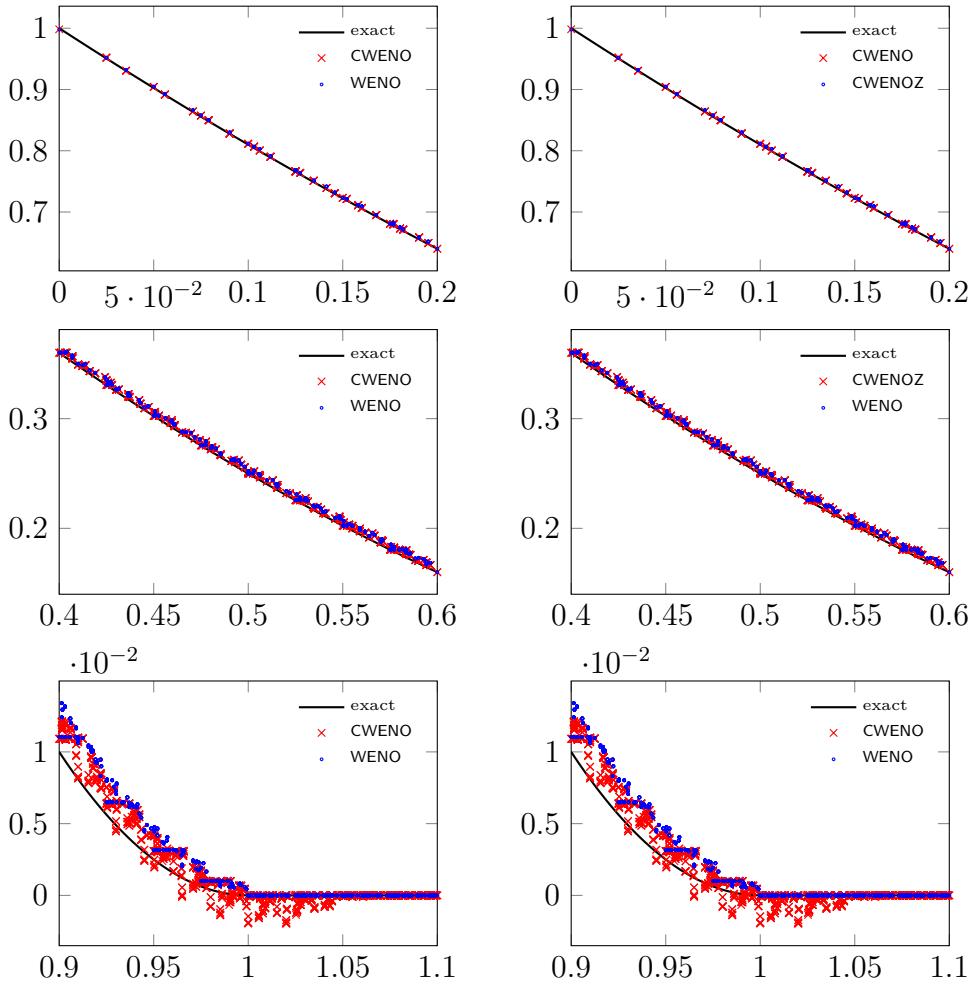

Figure 3.8: Test 4. Comparison of overshoots and undershoots of the numerical solutions computed on a 161×161 grid. In the left panels WENO and CWENO are compared, while in the right panels similar plots are shown for WENO and CWENOZ. The exact solution is always represented with a thick black line. First row: scatter plots of all data points with $r \in [0, 0.2]$ corresponding to the spike of the solution. Second row: scatter plots for $r \in [0.4, 0.6]$ corresponding to regular region of the solution. Third row: scatter plots for $r \in [0.9, 1.1]$ corresponding to the singular region of the initial data.

	WENO	CWENO		CWENOZ	
grid	CPU time	CPU time	% gain	CPU time	% gain
41×41	$2.88e+00$	$1.98e+00$	31.37	$1.96e+00$	32.07
81×81	$2.19e+01$	$1.54e+01$	29.70	$1.53e+01$	30.07
161×161	$1.72e+02$	$1.19e+02$	31.10	$1.18e+02$	31.35
321×321	$1.37e+03$	$9.38e+02$	31.56	$9.34e+02$	31.86
641×641	$1.09e+04$	$7.44e+03$	31.59	$7.51e+03$	30.93

Table 3.8: CPU times for Test 4.

	WENO		CWENO		CWENOZ	
N	min	max	min	max	min	max
21×21	$-5.67e-05$	$9.76e-01$	$-6.08e-03$	$9.80e-01$	$-6.08e-03$	$9.80e-01$
41×41	$-4.62e-05$	$9.89e-01$	$-5.19e-03$	$9.92e-01$	$-5.19e-03$	$9.92e-01$
81×81	$-1.21e-05$	$9.95e-01$	$-3.39e-03$	$9.96e-01$	$-3.39e-03$	$9.96e-01$
161×161	$-3.45e-06$	$9.98e-01$	$-1.97e-03$	$9.98e-01$	$-1.97e-03$	$9.98e-01$

Table 3.9: Overshoots and undershoots observed in Test 4. The exact solution has range $[0, 1]$.

	WENO	CWENO		CWENOZ	
grid	CPU time	CPU time	% gain	CPU time	% gain
126×101	$6.28e+00$	$5.34e+00$	14.94	$5.30e+00$	15.66
251×201	$4.56e+01$	$3.96e+01$	13.04	$3.96e+01$	13.06
501×401	$3.61e+02$	$3.11e+02$	13.87	$3.20e+02$	11.50
1001×801	$2.81e+03$	$2.43e+03$	13.64	$2.40e+03$	14.59
2001×1601	$2.20e+04$	$1.92e+04$	13.01	$1.89e+04$	14.00

Table 3.10: CPU times for Test 5.

is considered in $[-2, 2]^2$, using extrapolation techniques to treat the homogeneous boundary condition. The exact solution of this problem is known and is given, for $t \geq 1/2$, by

$$v(t,x) = \begin{cases} \frac{(|x|-1)^2}{2t} & \text{if } |x| \leq 1, \\ 0 & \text{if } |x| > 1. \end{cases} \tag{3.38}$$

We set final time $T = 0.5$ and compute the approximate solution with $\Delta t = \frac{5}{4}\Delta x$. Initial condition and final numerical solution are shown in Figure 3.7, while L^1 errors and CPU times are reported respectively in Tables 3.7 and 3.8.

Since in this case the feet of characteristics always fall in the singular region of the solution, we observe, as expected, that all numerical schemes are degraded to first-order accuracy. From Table 3.7 one can see that the L^1-norm of the error is about 30% lower for the central reconstructions, as in the previous cases.

From (3.38) we observe that the exact solution attaints its maximum value 1 at $x = 0$ and is everywhere non-negative. The maximum values of the numerical solutions reported in Table 3.9 show a slightly lower numerical diffusion for the central schemes as opposed to the WENO one. Furthermore, from the minimum values reported in Table 3.9 and from Figure 3.8 we can see that near the kink in the solution, the central reconstructions produce larger undershoots in the flat region, while being closer to the exact solution in the non-zero region. Finally, from Table 3.8 one can observe that central reconstructions schemes are about 30% faster.

3.5.5 Front propagation with obstacles

Last, we consider a state constrained Zermelo problem, proposed in [15], for a swimmer that has to reach a circular island at $(1, 0)$, facing a non-constant current flowing in the horizontal direction. Two obstacles are placed in the domain, near the target.

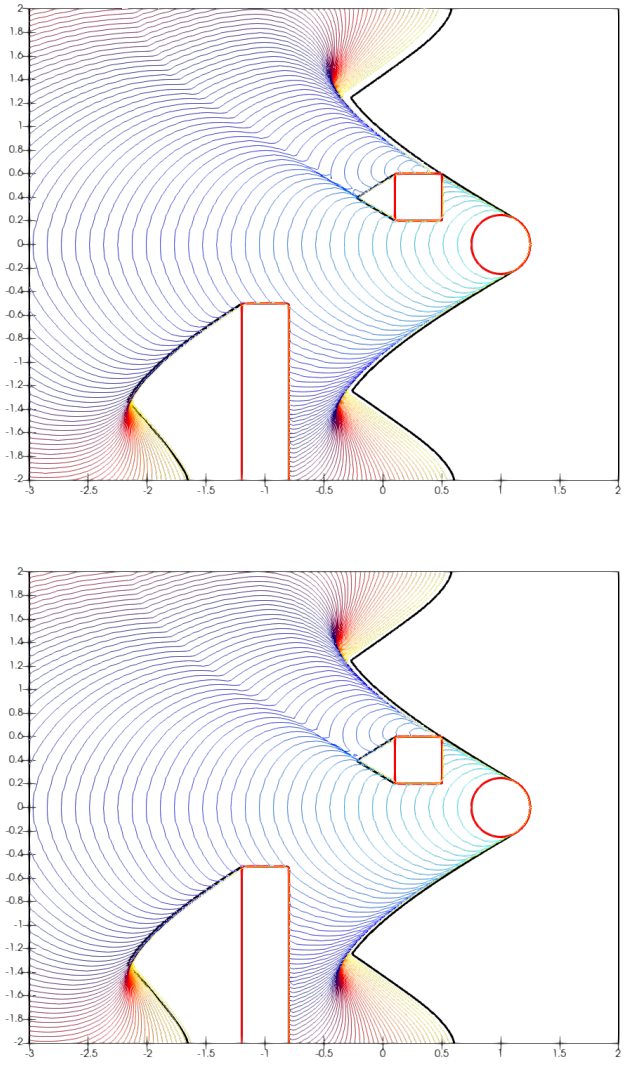

Figure 3.9: Reachable sets (3.40) computed at each time step for Test 5 on a grid 101×81 and $T = 3$, using WENO (above) and CWENO (below) reconstructions. In both panel the black line represents a reference solution computed on a grid 1001×801, the red circle represents the target and the red rectangles represent the obstacles.

More precisely, to consider the minimum time problem the running cost $f_C(t,x,a)$ is set to zero, and the non-linear dynamics is given by $f_D(t,x,a) = (2-0.5x_2^2+a_1, a_2)$, with $A := \{a \in \mathbb{R}^2 \,|\, \|a\| = (a_1^2+a_2^2)^{1/2} = 1\}$. The swimmer has unit speed and can move in any direction $a \in A$; the current has speed $2-0.5x_2^2$ and flows in the direction $(1,0)$. The target is the set $\mathcal{X} = \{x \mid v_0(x) \le 0\}$, where $v_0(x) = C\min\Big(\|(x_1-1,x_2)\|-r, r\Big)$ with $r = 0.25$ and $C = 20$. In order to take into account the presence of the two obstacles, we follow the level set approach, as proposed in [15], and represent the set of constrained states as $\mathcal{K} = \{x \in \mathbb{R}^2 \mid g(x) \le 0\}$ where

$$g(x) = \max\Big(-\gamma, g_1(x), g_2(x)\Big),$$

setting

$$g_1(x) = C\Big(\gamma - \max(|x_1-0.3|, |x_2-0.4|)\Big),$$
$$g_2(x) = C\Big(\gamma - \max(|x_1+1|, |x_2+1.5|)\Big),$$

with $C = 20$, $\gamma = 0.2$.

For every $t \in [0,T]$, the set of points from which the swimmer can reach the target *before time t* (Backward Reachable set) is represented by the level set $\mathcal{R}[0,t] = \{x \in \mathbb{R}^2 \,|\, \exists\, s \in [0,t],\, v(x,s) \le 0\}$ where v is the solution to a constrained version of (3.1)

$$\min(v_t + H(t,x,\nabla v), v - g(x)) = 0 \tag{3.39}$$

with initial condition $v(x,0) = \max(v_0(x), g(x))$, (see [15, Remark 2]). The space domain is defined as $[-3,2]\times[-2,2]$ and the final time to $T = 3$. By using scheme (3.7), we discretize (3.39) as

$$\min(u_i^{n+1} - \min_{\underline{a}\in A^\nu}\{I[u^n](y_i^n(\underline{a}))\}, u^{n+1} - g(x_i)) = 0,$$

resulting in the following time-marching scheme:

$$u_i^{n+1} = \max\left(\min_{\underline{a}\in A^\nu}\{I[u^n](y_i^n(\underline{a}))\}, g(x_i)\right),$$

from which we define the numerical approximation of the seachable set $\mathcal{R}[0,t_n]$ as

$$\mathcal{R}^n = \{x_i \,|\, \exists\, k \in \{0,\dots,n\}, u_i^k \le 0\}. \tag{3.40}$$

The solution is computed with RK3 and $\Delta t = \Delta x$. In Figure 3.9, we show that, even in this very complex situation, no relevant difference in the accuracy of the computed solution can be observed between the WENO and the Central WENO approaches. On the other hand, Central reconstruction schemes provide a reduction of about $13-14\%$ of the computational time.

Chapter 4

Surface reconstruction from point cloud

In this chapter we will propose the application of the LSM, introduced in the first chapter, to the problem of reconstructing unknown surfaces from unorganized sets of points, namely the point clouds, which are usually the only information one gets from the scanning of real objects [99, 98]. A point cloud is a dataset that is just constituted by a list of points approximately laying on the surface of the object and that carries no information about the connectivity between them.

This work actually constituted the starting point of our research, but comes as almost the last topic in this thesis since it collects most of the ingredients presented before. In fact, our surface reconstruction method will be based on a variational formulation that leads to solve an advection-diffusion equation that governs the evolution towards the point cloud of an initial surface described implicitly by a level set function. The numerical method for the approximation of the solution resorts to a SL scheme coupled with a local interpolator, in which, as we will see, the diffusive term originated by some curvature regularization is treated properly in order to overcome well-known time constraints related to the CFL condition.

Besides the numerical aspects related to the main PDE, this work aims to design a complete workflow to get a final implicit representation of the surface approximating the point cloud, focusing also on the computational aspects of such a procedure. It is well-known indeed that in real-world applications, where one could be interested in processing very large and detailed datasets, surface reconstruction could be very complex and time consuming. Hence it is crucial to propose an efficient and flexible tool, hopefully suitable for dynamic representations and not only for static ones.

We point out that the level set functions computed by (an early version of) the methods presented here have been employed in test cases for a more complete workflow in the setting of forecast of stone degradation by chemical pollutants in the field of management of cultural heritage [37].

The presentation in this chapter follows closely the one of [96]. After introducing the mathematical model, the SL scheme is presented, focusing in particular on the treatment of the parabolic term; since, as we have seen in Chapter 1, the LSM requires a lot of procedures to be put together, the complete strategy will be resumed, and finally, a collection of numerical tests in two and three dimensions will be presented.

4.1 Mathematical model

Let us assume that we are given a set of points $\mathcal{P} = \{Q_1, \dots, Q_N\}$ in a bounded region of $\mathbb{R}^d$ and let us define $d(\vec{x}) = \min_{Q_j \in \mathcal{P}} |\vec{x} - Q_j|$ to be the distance function to $\mathcal{P}$. Our model for surface reconstruction from point clouds is based on the minimization of the surface energy functional

$$E_p(\Gamma) = \Big(\int_\Gamma d^p(\vec{x}) \mathrm{d}s \Big)^{1/p}, \quad 1 \le p \le \infty \tag{4.1}$$

derived in [124], where Γ is an arbitrary closed surface of co-dimension one in $\mathbb{R}^d$ and $\mathrm{d}s$ is the surface area. The energy functional (4.1) is independent of parametrization and invariant under rigid transformation and, if we think of the distance function as a potential function for $\mathcal{P}$, our energy is the L^p norm of the potential on Γ. In particular, when $p = \infty$, $E_\infty(\Gamma)$ is the value of the distance from $\mathcal{P}$ of the most remote point on Γ.

Along the lines of [124], we will try to find a local minimizer of (4.1) that behaves like a minimal surface or an elastic membrane attached to the data set. The way to achieve this minimum and get the final shape is based on a continuous deformation of an initial surface Γ_0 following the gradient descent of the energy functional (4.1).

Retracing the basics of the model, we can easily calculate the first variation of the energy with respect to small perturbation of Γ

$$\frac{\delta E_p(\Gamma)}{\delta \Gamma} = \frac{1}{p} \Big[\int_\Gamma d^p(\vec{x}) \mathrm{d}s \Big]^{\frac{1}{p}-1} [p d^{p-1} \nabla d \cdot \vec{n} + d^p \kappa], \tag{4.2}$$

where $\vec{n}$ is the outward unit normal and κ is the mean curvature of Γ. Note that in the last square brackets of (4.2) appears the sum of the variation in the potential and the variation in the surface area, respectively.

Looking for a minimum of (4.1), from (4.2) one derives the Euler-Lagrange equation

$$d^{p-1}(\vec{x}) \Big[\nabla d(\vec{x}) \cdot \vec{n} + \frac{1}{p} d(\vec{x}) \kappa \Big] = 0, \tag{4.3}$$

which suggests that a stationary point of (4.1) is reached when we see a balance between the potential force $\nabla d(\vec{x}) \cdot \vec{n}$ and the surface tension $d(\vec{x})\kappa$. Furthermore, in the balance of the two terms, note the desirable role of the scaling $d(\vec{x})$ of the surface tension regularization: it allows more flexibility in regions where $d(\vec{x})$ is small, namely when we are approaching the data and where the sampling density is higher, while, on the other hand, further away from $\mathcal{P}$ or in cases of a lower sampling density, Γ will be more rigid.

Now, choosing an initial guess that encompasses all the data in $\mathcal{P}$, equation (4.3) suggests to continuously deform this Γ until steady-state following the gradient descent of the energy functional (4.1), considering the equation

$$\frac{d\Gamma}{dt} = - \Big[\int_\Gamma d^p(\vec{x}) \mathrm{d}s \Big]^{\frac{1}{p}-1} d^{p-1}(\vec{x}) \Big[\nabla d(\vec{x}) \cdot \vec{n} + \frac{1}{p} d(\vec{x}) \kappa \Big] \vec{n}. \tag{4.4}$$

If we start the evolution further away from $\mathcal{P}$, the effect of both the terms (the potential force and the surface tension) in (4.4) is to make the surface shrink towards

$\mathcal{P}$. When getting closer to $\mathcal{P}$, the two forces start to interplay trying to balance each other. The choice of starting the evolution with an initial Γ enclosing the dataset is just for robustness reasons: unless we are very close to the dataset, we might be exposed to the risk of a degeneration of the surface to the empty set.

It is also worth saying a few words about the existence of the minimum surface. In the continuous limit, when $d(\vec{x})$ is the distance function to a smooth surface Γ^*, we have two global minima, i.e., $\Gamma = \Gamma^*$ and $\Gamma = \emptyset$. When considering the distance function to a scattered set of points, as the case of $\mathcal{P}$, things become more complicated and, while in two dimensions the polygon connecting the points in $\mathcal{P}$ constitutes a trivial local minumum, in three or higher dimensions, the only trivial minimum is $\Gamma = \emptyset$. Thus, in three or higher dimensions, things become more interesting and the problem turns out to be ill-posed, i.e., there is no unique solution, since infinite surfaces may pass through or near $\mathcal{P}$ locally minimizing the energy functional (4.1). Therefore we will be only interested in finding a proper local minimum that solves the Euler-Lagrange equation (4.3) considering it as the desired final reconstruction.

So far, we have derived a model for the reconstruction problem which is based on an evolution process of an interface Γ. As stated in Chapter 1 this type of tracking problems might be very difficult to be handled explicitly since, in general, one doesn't have any a-priori knowledge about the topology of the shape to be reconstructed. Topological changes may occur during the continuous deformation process strongly discouraging any use of parametrization techniques, which would be almost impossible to implement. Thus again, this is where the implicit approach via level set functions comes into play, together with the LSM.

4.1.1 Derivation of the level set equation

Changing our perspective, the evolving surface $\Gamma(t)$ can be represented implicitly using a level set function to capture the moving interface. Just to recall the notation, considering $\Omega(t)$ the region enclosed by $\Gamma(t)$, a level set function $\phi(\vec{x}, t)$, $\phi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$, is associated to $\Omega(t)$ if it assumes negative values inside $\Omega(t)$, positive outside and returns zero on the boundary $\Gamma(t)$, according to (1.9).

In terms of the level set function ϕ, the energy functional (4.1) can be rewritten as

$$E_p(\phi) = \left(\int_{\mathbb{R}^d} |d(\vec{x})|^p \delta(\phi) |\nabla \phi| \mathrm{d}\vec{x} \right)^{1/p} \tag{4.5}$$

where δ denotes the Dirac-delta function and $\delta(\phi)\,|\nabla\phi|\,\mathrm{d}\vec{x}$ is the surface area element at the zero level set of ϕ.

Using (4.4), we can derive the evolutionary equation for the zero level set of ϕ, which reads

$$\frac{\partial \phi}{\partial t} = \delta(\phi) \frac{1}{p} \Big[\int d^p(\vec{x}) \delta(\phi)\,|\nabla\phi|\,\mathrm{d}\vec{x} \Big]^{\frac{1}{p}-1} \nabla \cdot \Big[d^p(\vec{x}) \frac{\nabla \phi}{|\nabla \phi|} \Big]. \tag{4.6}$$

Recalling that we aim to solve a level set equation of type

$$\phi_t + \frac{\mathrm{d}\Gamma(t)}{\mathrm{d}t} \cdot \nabla\phi = 0 \quad \text{or} \quad \phi_t + v_n \left|\nabla\phi\right| = 0, \tag{4.7}$$

on the whole domain $\mathbb{R}^d \times \mathbb{R}^+$, as presented in Section 1.4, we need to extend the geometric motion at $\phi = 0$ defined by (4.6) to all the other level sets. This is done by replacing $\delta(\phi)$ with $|\nabla\phi|$ and leads to the level set formulation

$$\begin{cases} \phi_t(\vec{x},t) & = \left[\frac{d(\vec{x})}{E_p(\phi)}\right]^{p-1} \left(\nabla d(\vec{x}) \cdot \nabla\phi(\vec{x},t) + \frac{d(\vec{x})}{p} \nabla \cdot \left(\frac{\nabla\phi(\vec{x},t)}{|\nabla\phi(\vec{x},t)|}\right) |\nabla\phi(\vec{x},t)|\right), \\ \phi(\vec{x},0) & = \phi_0(\vec{x}), \end{cases} \tag{4.8}$$

where $\phi_0(\vec{x})$ is a suitable initial data such that $\Gamma_0 = \{\vec{x} \in \mathbb{R}^d : \phi_0(\vec{x}) = 0\}$ and $\frac{\nabla\phi}{|\nabla\phi|}$ and $\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$ are the usual level set representations of the outward unit normal and of the mean curvature, respectively.

In the above equation, the term $\nabla d(\vec{x}) \cdot \nabla\phi(\vec{x},t)$ drives the level sets, thus the surface itself, towards the dataset $\mathcal{P}$, while the second term tempers the maximal curvature. Changing the balance between them can lead to surfaces closer to the point cloud but with sharper edges or to more rounded surfaces that are a little further away from $\mathcal{P}$.

The parameter p controls the relative influence of the curvature term and of the global scaling factor $\left[\frac{d(\vec{x})}{E_p(\phi)}\right]^{p-1}$ which makes the the most remote points to move quicker than the ones closer to $\mathcal{P}$. We point out that the $E_p(\phi)$ denominator slows down the entire evolution when the functional is large; this can be an issue when the initial surface is very far from the cloud.

In [79], a model similar to (4.8) is employed, wherein the factor $d(\vec{x})$ in front of the diffusion term is replaced by a constant $\eta \in [0,1]$ to control the balance between the two terms. While we prefer to keep the $d(\vec{x})$ factor in the evolution equation which will slow down the evolution in the vicinity of the point cloud, we also introduce an additional parameter $\eta \geq 0$ multiplying $d(\vec{x})$ in the curvature term. Our final formulation then reads

$$\begin{cases} \phi_t(\vec{x},t) & = \left[\frac{d(\vec{x})}{E_p(\phi)}\right]^{p-1} \left(\nabla d(\vec{x}) \cdot \nabla\phi(\vec{x},t) + \eta\frac{d(\vec{x})}{p} \nabla \cdot \left(\frac{\nabla\phi(\vec{x},t)}{|\nabla\phi(\vec{x},t)|}\right) |\nabla\phi(\vec{x},t)|\right), \\ \phi(\vec{x},0) & = \phi_0(\vec{x}). \end{cases} \tag{4.9}$$

In this way our numerical algorithm can be used in both regimes. If we set $\eta = 0$, the model becomes purely convective and leads quickly to a reconstructed surface close to a piecewise linear approximation. In contrast, the case with $\eta = 1$ corresponds to the original formulation and contains a weighted curvature regularization effect, is more computationally expensive, but leads to a smoother reconstructed surface. Higher values of μ are also useful in the case of noisy data.

Finally, we observe that for visualization purposes it is often sufficient to compute a function ϕ such that its zero level set is close to the data set $\mathcal{P}$. Equation (4.9) is

apt for this purpose, but its transport term tends to produce high gradients in the computed function ϕ. This behaviour can constitute an issue when one is interested in computing the normals of the reconstructed surface or in using the level set function as a mathematical description of a domain into which a PDE solver has to be applied; it is indeed more robust to compute a level set function with moderate gradients, as in [37]. To this aim, our algorithm will be designed to ensure that the computed solution is a SDF at least in the vicinity of the point cloud.

4.2 SL schemes for mean curvature motion

To numerically solve the IVP (4.9) we have to take into account the nature of this equation, which contains both a linear advective term and a parabolic one, represented by the curvature regularization. This last term, when resorting to an explicit scheme, imposes a time restriction of type $\Delta t = \mathcal{O}(\Delta x^2)$ which might constitute a real obstacle, especially when the spatial grid is refined. Having in mind our application to surface reconstruction, it is absolutely natural to think of using a finer resolution at least around the zero level set of ϕ, for instance via AMR, as will be shown in the fifth chapter, thus it is crucial to design a scheme which is able to overcome this CFL limitation. Also, as pointed out in Section 1.3.1, when dealing with LSM, and in particular with the evolution described by MCM, the implicit time-stepping approach is hardly discouraged due to the complexity of the problem.

Therefore, the SL approach seems to be apt to this aim in the sense that merges the advantages on an explicit scheme with the stability properties of an implicit one, allowing to overcome very harsh timestep restrictions. In the second chapter, we have seen how to derive a SL discretization for different general types of hyperbolic PDEs but still we haven't explored the treatment of second-order HJ equations. Since we are interested in the particular case of the second-order evolutive equation arising in the level set formulation of MCM, we prefer to treat in this chapter this specific case.

Thus, let us consider the simplified problem

$$\begin{cases} \phi_t(\vec{x},t) = |\nabla\phi(\vec{x},t)|\,\nabla\cdot\left(\frac{\nabla\phi(\vec{x},t)}{|\nabla\phi(\vec{x},t)|}\right) \\ \phi(\vec{x},0) = \phi_0(\vec{x}), \end{cases} \tag{4.10}$$

where ϕ_0 is a level set function representing the front Γ_0 at time $t = 0$. The scheme we use here has been first proposed in [50] and is based on the representation formula proved by Soner and Touzi in [108] for the solution of a large class of geometric second-order HJ equations which includes (4.10). This scheme has then gone through a number of improvements and applications; the interested reader can refer to [23] for a comprehensive convergence analysis and to [21, 24] for two applications to image processing.

The representation formula introduced in [108] has the form

$$\phi(\vec{x},t) = \mathbb{E}\{\phi_0(\vec{y}(\vec{x},t;t))\} \tag{4.11}$$

where $\mathbb{E}(\cdot)$ is the probabilistic expectation and $\vec{y}(\vec{x},t;s)$ is the position at time s of the solution trajectory of the Stochastic Initial Value Problem (SIVP) originating

at time t from $\vec{x}$; more precisely we have to solve the system

$$\begin{cases} \dot{\vec{y}}(\vec{x},t;s) = \sqrt{2}P(\nabla\phi(\vec{y}(\vec{x},t;s),t-s))dW(s), \\ \vec{y}(\vec{x},t;0) = \vec{x}, \end{cases} \tag{4.12}$$

where W is a d-dimensional standard Brownian motion and P is a $d \times d$ projection matrix defined by

$$P(p) = I - \frac{pp^T}{|p|^2}. \tag{4.13}$$

In practice, the diffusion process in (4.10) will be replaced by a Brownian motion, restricted to the tangent plane of the level sets, and the probabilistic expectation will be discretized by a weighted average.

In fact, let us observe that the projection matrix in (4.12) is a matrix of rank $d-1$ with $d-1$ eigenvectors corresponding to the eigenvalue $\lambda = 1$, thus we can rewrite

$$P(\nabla\phi) = \sigma(\nabla\phi)\sigma(\nabla\phi)^T, \tag{4.14}$$

where σ is a $d \times (d-1)$ matrix whose columns are the eigenvectors of P.

Using the decomposition (4.14) we can rewrite (4.12) as

$$\begin{cases} \dot{\vec{y}}(\vec{x},t;s) = \sqrt{2}\sigma(\nabla\phi(\vec{y}(\vec{x},t;s),t-s))d\widehat{W}(s), \\ \vec{y}(\vec{x},t;0) = \vec{x}. \end{cases} \tag{4.15}$$

where $d\widehat{W} = \sigma^T dW$ is the differential of a standard $(d-1)$-dimensional Brownian motion which is one dimension lower than W.

As usual, to derive the SL discretization we consider a node x_i, with $i \in \mathbb{Z}^d$, and a single time step $[t_n, t_{n+1}]$ and mimic the representation formula

$$\phi(x_i, t_{n+1}) = \mathbb{E}\{\phi(\vec{y}(x_i, t_{n+1}; \Delta t), t_n)\}, \tag{4.16}$$

with $\vec{y}(x_i, t_{n+1}; \Delta t)$ given by the stochastic ODE (4.15) written on a single time step.

Along the lines of [50, 23], to discretize (4.16), we resort to the theory of weak convergence for numerical schemes for Stochastic Differential Equations (SDEs); see [78] for full details.

Applying a Stochastic Euler scheme to solve (4.15) we get

$$y_i^n = x_i + \sqrt{2}\sigma(\nabla\phi(x_i, t_{n+1}))\Delta\widehat{W}, \tag{4.17}$$

where $\Delta\hat{W}$ should represent a Gaussian variable with mean 0 and variance Δt. However, to obtain first-order convergence it suffices to have a variable with 2-point discrete probability density,

$$P(\Delta\widehat{W} = \pm\sqrt{\Delta t}) = \frac{1}{2}, \tag{4.18}$$

for the one dimensional Brownian, and with 4-point discrete probability density

$$P((\Delta\widehat{W}_1, \Delta\hat{W}_2) = (\pm\sqrt{\Delta t}, \pm\sqrt{\Delta t})) = \frac{1}{4}, \tag{4.19}$$

for the two dimensional case.

Note that in (4.17) the matrix σ is evaluated in the gradient of the level set function in the node x_i at time t_{n+1}, but the only information available for us will be at time t_n. However, since the level set function ϕ evolves in the normal direction, namely the direction of $\nabla\phi$, we can use the gradient at time t_n in (4.17) and get the final explicit scheme

$$\phi_i^{n+1} = \frac{1}{2^{d-1}} \sum_{j=1}^{2^{d-1}} I[\phi^n](x_i + \sqrt{2\Delta t}\sigma_i^n \xi_j) \tag{4.20}$$

where d is the dimension of the problem, $\sigma_i^n = \sigma(\nabla\phi(x_i, t_n))$ and ξ_j ranges over $\{+1, -1\}$ and $\{(\pm 1, \pm 1)\}$ for the 2d and 3d case, respectively.

The explicit expression for the matrix σ will be given in the next section where the discretization of the parabolic term, as stated here, will be considered along with the discretization of the advective part. The two and three dimensional case will be treated separately in order to have a clear design of the complete scheme.

4.3 Numerical scheme

The scheme we propose here mixes the SL approach both for the hyperbolic and the parabolic term in (4.9) and mimics the scheme presented in [16, 26] with some differences: the parameters p and η, and the use of the cut-off function for localization, will lead to a slightly modified scheme, but the main difference will be related to the interpolation operator. In particular, in [26] the authors resort to a global Radial Basis Function (RBF) interpolator that requires to solve a large linear system at each timestep which is something that we would like to avoid and is the reason why we decide instead to use a local interpolator, which is more efficient, especially in view of a parallel implementation.

We will follow the notation of the previous sections distinguishing between the two and three dimensional case and indicating with x_i a generic node of the computational grid, with $i \in \mathbb{Z}^d$. Also, instead of specifying the set in which the integer index i of the discretized space is varying, we will refer to the whole computational grid as $\mathcal{G}$ such that

$$\mathcal{G} = \{x_i : x_i \in \Omega', i \in \mathbb{Z}^d\}, \tag{4.21}$$

where Ω' is a large enough computational domain in which we are embedding $\Omega(t)$; see Section 1.1.2. Specific subgrids to locate the LSM, as introduced in Section 1.6 will be detailed later.

The SL scheme will be detailed in this section for the two and three dimensional case and the interpolation operator will be described in Section 4.3.1. The next section will describe the other auxiliary procedures used in order to design the complete algorithm: the computation of the distance function $d(\vec{x})$ from the point cloud, the evaluation of the energy functional $E_p(\phi)$, the computation of the initial data ϕ_i^0, the reinitialization procedure and the localization technique to reduce the computational effort. The reader will be also referred to previous sections were some of these topics have been already treated. The complete algorithm is summarized in Algorithm 1. Unless otherwise specified, all numerical gradients needed in the algorithms are computed by centered finite differences.

Algorithm 1 Computing the signed distance function ϕ from a point cloud $\mathcal{P}$

1. Create a Cartesian grid $\mathcal{G}$ encompassing all of $\mathcal{P}$
2. Compute the distance function $d(x_i)$ at all grid points, as in Section 4.4.2
3. Set initial data $\{\phi_i^0\}_{x_i \in \mathcal{G}}$, as in Section 4.4.4
4. Loop:
   (a) From $\{\phi_i^n\}_{x_i \in \mathcal{G}}$ set the computational subgrid $\mathcal{B}^n \subset \mathcal{G}$, as in Section 1.6
   (b) Compute the energy functional (4.5) with one of the methods of Section 4.4.3
   (c) Compute $\{\phi_i^{n+1}\}_{x_i \in \mathcal{B}^n}$ using (4.22) or (4.24) from Section 4.3 and the interpolator (4.28) or (2.69) from Section 4.3.1 and Section 2.3.4
   (d) From $\{\phi_i^{n+1}\}_{x_i \in \mathcal{G}}$ set the computational subgrid for reinitialiaztion $\overline{\mathcal{B}}^{n+1} \subset \mathcal{G}$, as in Section 1.6
   (e) Reinitialize $\{\phi_i^{n+1}\}_{x_i \in \overline{\mathcal{B}}^{n+1}}$, as in Section 1.5 and cut with (1.49)

2d Case

We compute the update of ϕ_i^n as

$$\begin{cases} \phi_i^{n+1} \;=\; \frac{1}{2} \sum_{j=1}^{2} I[\Phi^n]\left(x_{i,j}^*\right), \\ x_{i,j}^* = x_i + C_i^n \Delta t \nabla d(x_i) + \sqrt{\frac{2C_i^n \, \eta \, d(x_i) \Delta t}{p}} \, \sigma_i^n \xi_j, \end{cases} \tag{4.22}$$

where ξ_i ranges over $\{-1, +1\}$ and C_i^n is the scale factor $\left[\frac{d(x_i)}{E_p(\phi^n)}\right]^{p-1}$. The operator $I[\Phi^n](\vec{x})$ denotes an interpolation at point $\vec{x}$ of the data $\{\phi_k^n : x_k \in \mathcal{G}\}$, which will be specified later. In (4.22) σ_i^n denotes the unit vector tangent to the level sets of ϕ: it is thus orthogonal to the gradient of the level set function and is given by

$$\sigma_i^n = \frac{1}{|\nabla \phi_i^n|} \begin{pmatrix} \partial_y \phi_i^n \\ - \partial_x \phi_i^n \end{pmatrix}. \tag{4.23}$$

In equation (4.22), the interpolation points are obtained by adding two terms: the first is the foot of the characteristic pertaining to the advection term in (4.9), while the second is the further displacement discretizing the curvature term in (4.9). A sketch of the 2d scheme is shown in Figure 4.1.

3d Case

Similarly to the previous case, the update of ϕ_i^n is computed as

$$\begin{cases} \phi_i^{n+1} \;=\; \frac{1}{4} \sum_{j=1}^{4} I[\Phi^n]\left(x_{i,j}^*\right), \\ x_{i,j}^* = x_i + C_i^n \Delta t \nabla d(x_i) + \sqrt{\frac{2C_i^n \, \eta \, d(x_i) \Delta t}{p}} \, \sigma_i^n \xi_j, \end{cases} \tag{4.24}$$

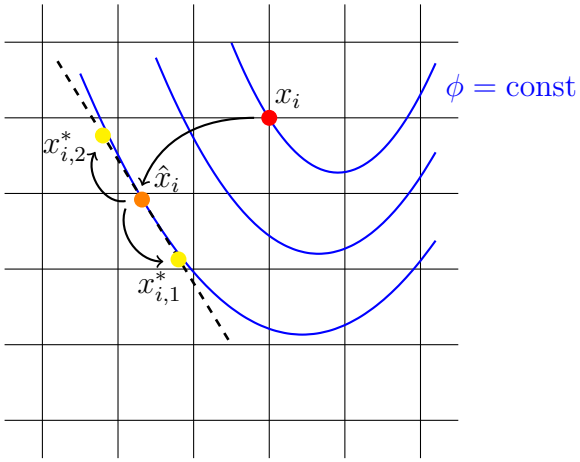

Figure 4.1: A sketch of the procedure for updating the level set function ϕ in a grid node in two dimensions. From a grid node x_i we trace backward the corresponding characteristic line for a time Δt finding the point $\hat{x}_i$ in orange. The curvature regularization thus requires to compute the locations of the two yellow points $x^*_{i,1}$, $x^*_{i,2}$ on the plane tangent to the level sets of ϕ. Once the two yellow points are located, we interpolate using the value of the function ϕ at time t_n and then perform an average to get the final update ϕ_i^{n+1}.

where again we treat the scale factor C_i^n as a constant. In the above equation, $\sigma_i^n = [\nu_1(\nabla\phi_i^n), \nu_2(\nabla\phi_i^n)]$ is a 3×2 matrix whose columns span the $2d$ space orthogonal to $\nabla\phi^n$ and the column vector ξ_j ranges over $\{(\pm 1, \pm 1)^T\}$, so that $\sigma_i^n \xi_j$ represents four points in the local tangent plane.

In order to avoid numerical singularities, one regularizes the two orthonormal eigenvectors of the projection onto the plane tangent to the level sets of ϕ as

$$\nu_1 = \begin{cases} \tilde{\nu}_1 & \text{if } \sqrt{(\partial_x\phi)^2 + (\partial_z\phi)^2} \neq 0 \\ (1,0,0)^T & \text{otherwise} \end{cases}$$
$$\nu_2 = \begin{cases} \tilde{\nu}_2 & \text{if } \sqrt{(\partial_x\phi)^2 + (\partial_z\phi)^2} \neq 0 \\ (0,0,1)^T & \text{otherwise} \end{cases}$$

with $\tilde{\nu}_k$ denoting the exact eigenvectors, which are given by

$$\tilde{\nu}_1 = \begin{pmatrix} \frac{-\partial_z\phi}{\sqrt{(\partial_x\phi)^2+(\partial_z\phi)^2}} \\ 0 \\ \frac{\partial_x\phi}{\sqrt{(\partial_x\phi)^2+(\partial_z\phi)^2}} \end{pmatrix}, \quad \tilde{\nu}_2 = \frac{1}{|\nabla\phi|} \begin{pmatrix} \frac{-\partial_x\phi\,\partial_y\phi}{\sqrt{(\partial_x\phi)^2+(\partial_z\phi)^2}} \\ \sqrt{(\partial_x\phi)^2 + (\partial_z\phi)^2} \\ \frac{-\partial_y\phi\,\partial_z\phi}{\sqrt{(\partial_x\phi)^2+(\partial_z\phi)^2}} \end{pmatrix}. \tag{4.25}$$

Because of the factor in front of σ_i^n in (4.23), respectively in front of $\tilde{\nu}_2$ in (4.25), the numerical schemes (4.22) and (4.24) would be singular in cases where $|\nabla\phi_i^n|$ is close to zero. Thus, when $|\nabla\phi_i^n| < D\Delta t^\alpha$, the schemes are replaced by

$$\phi_i^{n+1} = \frac{1}{|\mathcal{N}_i|} \sum_{x_j \in \mathcal{N}_i} I[\Phi^n](x_j) \tag{4.26}$$

where $\mathcal{N}_i$ is the set of the first neighbours of x_i in the Cartesian grid $\mathcal{G}$ and $|\mathcal{N}_i|$ represents its cardinality. In all computations presented in this work, we set $D = 10^{-3}$ and $\alpha = 1$.

4.3.1 Interpolation

An important issue for the accuracy and the efficiency of the numerical scheme is clearly represented by the choice of the interpolation operator $I[\Phi^n]$. In [26], the authors employ a global RBF interpolation of the data on $\mathcal{G}$ defined as

$$I_{\mathsf{RBF}}[\Phi^n](\vec{x}) = c_0(\Phi^n) + \vec{c}(\Phi^n) \cdot \vec{x} + \sum_{x_j \in \mathcal{G}} \mu_j(\Phi^n)\psi(|\vec{x} - x_j|).$$

The coefficients $c_0 \in \mathbb{R}, \vec{c} \in \mathbb{R}^d$ and $\mu_j \in \mathbb{R}$ of the RBF interpolator are computed by imposing interpolation conditions at all (or a subset of) points of the grid ($I[\Phi](x_k) = \phi_k$ for all $x_k \in \mathcal{G}$) and the additional conditions

$$\sum_{x_j \in \mathcal{G}} \mu_j = 0, \quad \sum_{x_j \in \mathcal{G}} P_x x_j \mu_j = 0, \quad \sum_{x_j \in \mathcal{G}} P_y x_j \mu_j = 0, \quad \sum_{x_j \in \mathcal{G}} P_z x_j \mu_j = 0, \tag{4.27}$$

where $P_x x_j$, $P_y x_j$ and $P_z x_j$ denote the spatial coordinates of the node x_j. The interpolation is thus computed by solving, at each time step, a linear system with matrix M of the form

$$M = \begin{bmatrix} B & P \\ P^T & 0 \end{bmatrix},$$

where B is an $N \times N$ block ($N = |\mathcal{G}|$) and P is $N \times 4$.

However, the computation of the RBF interpolator then becomes a bottleneck of the algorithm due to the solution of the linear system involved. Furthermore, the global linear term $c_0(\phi^n) + \vec{c}(\phi^n) \cdot \vec{x}$, i.e. the blocks P and P^T in the system, forms a strong coupling of all the equations which is difficult to handle for parallel runs as it requires a lot of inter-processor communications.

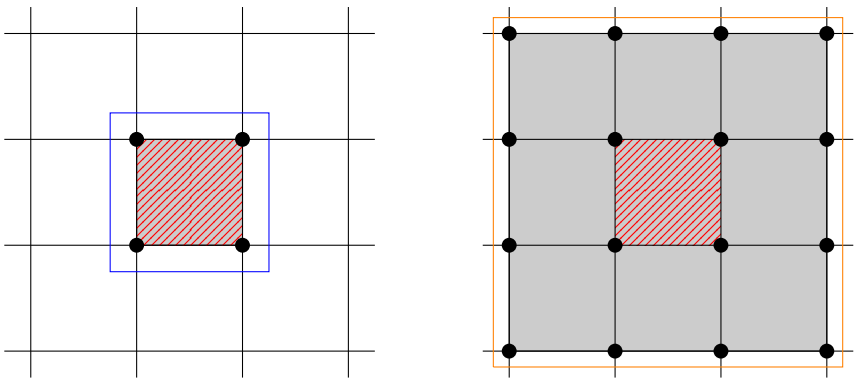

Figure 4.2: Stencils of the two-dimensional Q1 and WENO reconstructions. The red hatched region represents the cell Ω in which we compute the reconstruction. The multi-linear interpolator only requires the vertices of the cell Ω, enclosed by the blue square on the left. On the right, the WENO interpolator involves the cell vertices and their first neighbours, enclosed in the orange square.

Here instead we resort to two different types of local interpolator with the aim of making the algorithm faster and practical for surface reconstructions on a finer grid, allowing also the efficient use of parallel computing. The first one is a multilinear interpolator, while the second is a WENO interpolator.

The multilinear interpolator is simply the Q1 Finite Element interpolation on the grid $\mathcal{G}$. Let $\{\varphi_i\}_{x_i \in \mathcal{G}}$ be the shape functions that are, in each cell, a tensor product of degree 1 polynomials in each space direction and such that $\varphi_i(x_k) = \delta_{i,k}$. Then, for any function $v(\vec{x})$, we consider the interpolator

$$I_{\mathsf{Q1}}[v](\vec{x}) = \sum_{x_i \in \mathcal{G}} v_i \varphi_i(\vec{x}) \tag{4.28}$$

where v_i denotes the point value of the function v at the point x_i.

We remark that, once the voxel of the grid containing $\vec{x}$ is located, the summation above in the 2d case (respectively in the 3d case) involves only four (resp. eight) terms and can be locally computed.

We expect that this kind of interpolator will be computationally very cheap, but that it might give rise to artifacts in the reconstructed surfaces since it is piecewise linear and cannot faithfully represent the curvature of the surface.

The WENO interpolator is exactly the one introduced in Section 2.3.4 and we recall that in two or higher dimensions, a splitting dimension procedure is applied, thus guaranteeing the continuity con the global reconstruction, as for the case of the Q1 interpolant.

In this case, once the feet of the characteristics are located, the stencil for the interpolation involves one more layer around the cell, is more accurate and of course more expensive, but still not as expensive as the RBF one, since it is local, except for special cases related to parallel implementation.

As an example, the stencils involved in the reconstruction procedures for the 2d case are shown in Figure 4.2. In this way, we can guarantee that the only communications in the semi-Lagrangian scheme occur when the two (resp. four) interpolation points needed for the update (4.22) (respectively (4.24)) belong to a different processor than the one owning the point x_i (see Section 4.4.1) and not during the reconstruction procedure itself.

4.4 Auxiliary procedures

As showed in the first Chapter, and summarized in Section 1.7, designing a complete algorithm for the level set equation (4.9) requires different procedures to be put together. In this section we present all the necessary computations and auxiliary techniques we use in our method.

In particular, since especially three-dimensional computations suggest the use of parallel computing for reducing the execution time, a special focus will be put on the organization of the computational grid, the parallelization of each procedure and the main communications between workers that might occur during the execution.

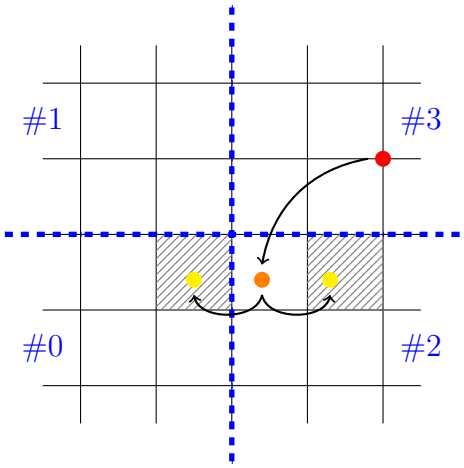

Figure 4.3: An example of communication that can occur between processes in a parallel implementation of the SL scheme. To compute the update of ϕ in local node x_i (red point), rank 3 should ask to ranks 0 and 1 the interpolated values at remote points $x^*_{i,1}$ and $x^*_{i,2}$ (yellow points).

4.4.1 Domain decomposition for parallel runs

As usual for Cartesian grids, each rank m owns a rectangular portion G_m of the computational grid $\mathcal{G}$ such that $\mathcal{G} = \cup_m G_m$ and is responsible to update the solution on it. Since the computation of the reconstruction and of the time update might require knowledge of the solution at the previous time step in a neighbourhood of G_m, each rank also keeps a local copy of the data in a halo region $\widetilde{G}_m$ of at least one ghost point per direction around G_m. The width of this halo region depends on which information are needed to perform all the computations in the algorithm. The halo data in $\widetilde{G}_m$ have to be synchronized between workers during the computations through inter-process communications.

In the case of a semi-Lagrangian scheme, the main sources of communication arise from: first, the interpolation of ϕ^n at feet of the characteristics emanating backwards from the nodes x_i and specified in equations (4.22) and (4.24); second, the stencil width required for the computation of the interpolant. This latter is mitigated by our choice of local interpolation techniques with very small stencils, like those of Section 4.3.1: in both cases we are able to run with one ghost-cell per direction for the Q1 reconstruction and two ghosts per direction in the WENO case.

Because the SL scheme can require the reconstruction points far away from the node to be updated, it is not practical to cover the stencil of the SL scheme with the halo region. However, the task of interpolating at the feet of characteristics is facilitated by the Cartesian structure of the grid and of its partitioning scheme. Consider any of the points $x^*_{i,j}$ appearing in (4.22) or (4.24) associated to a node x_i owned by the processor m. First $x^*_{i,j}$ is located on the grid partition and let m^* be the processor owning the cell into which it falls. If $m^* = m$, the interpolation $I[\Phi^n](x^*_{i,j})$ is computed without the need of any communication. Otherwise, a request to the processor m^* is sent, communicating the point $x^*_{i,j}$ and receiving in response the value of $I[\Phi^n](x^*_{i,j})$. Of course all these communications are gathered in a single push. In Figure 4.3 an example of situation in which we need to communicate

between processors to perform the update of ϕ is depicted.

In our implementation also the point cloud $\mathcal{P}$ is distributed among the workers. Some more communications are needed in the auxiliary routines described in the next sections and will be highlighted therein.

4.4.2 Distance function

In the semi-Lagrangian schemes (4.22) and (4.24) one needs to evaluate the distance $d(\vec{x}) = \min_{Q\in\mathcal{P}} |\vec{x} - Q|$ from the dataset $\mathcal{P}$ at every grid point. We remark that a direct computation would have computational complexity proportional to $N \times |\mathcal{P}|$, where N is the number of points in the computational grid and $|\mathcal{P}|$ the size of the point cloud. It would also require an overwhelming amount of communications in parallel runs, when both the grid and the point cloud are distributed among different workers.

However, one acknowledges that accurate values of the distance function are needed only close to the object surface to which $\mathcal{P}$ belongs. In practice we initially set $d(x_i)$ to the exact distance from $\mathcal{P}$ on all nodes of $\mathcal{G}$ that are in a box of 4×4 or $4\times4\times4$ grid points around each point $Q \in \mathcal{P}$. Then, we apply a sequence of the fast sweeping method of [121] and halo region interprocess communications to compute approximate values for $d(\vec{x})$ at all other grid points $x_i \in \mathcal{G}$. These approximate values are still good enough to drive the evolution of the surface Γ towards $\mathcal{P}$, while the final stages of the evolution will be guided by the exact values of the distance set in the neighbourhood of the points in $\mathcal{P}$.

4.4.3 Energy functional

Applying the schemes (4.22) and (4.24) with $p > 1$ also requires to approximate, at each time step, the value of the energy functional $E_p(\phi)$ defined in (4.5). We approximate the Dirac δ function by restricting the integration domain to the subset $\mathcal{G}_0$ composed by the cells with vertexes in $\mathcal{G}$ where, at a specific time step, the front is located. Further, we assume that the function ϕ is, at least locally around $\mathcal{G}_0$, a signed distance so that $|\nabla\phi| = 1$. This latter point is ensured by the reinitialization procedure described in Section 1.5. We thus compute the energy as

$$E_p(\phi) \approx \left(\sum_{\mathcal{C}\in\mathcal{G}_0} \int_{\mathcal{C}} |d(\vec{x})|^p \mathrm{d}\vec{x} \right)^{1/p} . \tag{4.29}$$

To detect the cells in $\mathcal{G}_0$, we check the values of ϕ on their vertexes and consider only the cells across which the function ϕ changes sign. Once the front is located in a cell $\mathcal{C}$, different strategies can be considered to compute (4.29).

- In two space dimensions, we identify two points intercepted by the front of ϕ on the cell boundary, by the linear interpolation of ϕ on the edges. Then the trapezoidal rule is used to approximate the integral over the segment connecting them. This approach would be much more involved in the three-dimensional case since one would need to identify the (two-dimensional) intersection of the zero level set surface with the cube $\mathcal{C}$ and this could be placed in a general position.

- In three space dimensions, to avoid the aforementioned complications, we further approximate the Dirac δ function within the volume Ω by considering a local refinement of the grid element with $\Delta x' = \Delta x/R$, $R > 1$, to detect the smaller subcells containing the front, and consider the approximation

$$E_p(\phi) \approx \left(\sum_{\mathcal{C}' \in \mathcal{G}'_0} |d(x_{i'})|^p (\Delta x')^{n-1} \right)^{1/p}, \tag{4.30}$$

  where $d(x_{i'})$ is the interpolated value of the distance function at the center of the subcell $\mathcal{C}'$.

  The subgrid $\mathcal{G}'_0$ is composed by the subcells $\mathcal{C}'$ such that the reconstruction of ϕ at the center of the cell satisfies

$$|I[\Phi](x_{i'})| < \frac{\sqrt{3}}{2}\Delta x'.$$

  We employ this approach with $R = 5$.

4.4.4 Initial data

The initial data should be chosen as an approximation of the signed distance function from the data set $\mathcal{P}$. Obviously, the better the initial data, the more efficient the method will be, but one has to take into account also the computational effort spent in the computation of the initial datum itself. In practice, we compute ϕ^0 from the distance function, similarly to [124], to obtain a signed distance function whose zero level surface encompasses the point cloud and is as close as possible to it.

In particular we start by the approximate distance function computed as in Section 4.4.2. First, the cells on the outer boundary of $\mathcal{G}$ are marked as external. Then, moving inwards from all boundaries in all Cartesian directions, we propagate the external point marking to nearby cells until we find a point for which $d(x_i) < \gamma_{\mathcal{P}}$, where $\gamma_{\mathcal{P}}$ is a suitable threshold related to the resolution of the point cloud.

After external regions have been so identified, ϕ^0 is provisionally set to $d(x_i) - \gamma_{\mathcal{P}}$ on external points and to an artificially high value otherwise. Next, the fast sweeping method of [121] is applied to ϕ^0, recomputing its values at internal points; finally, the sign of ϕ^0 is reversed on internal points.

We point out that in parallel runs the procedure for initial marking of external points is similar to the serial run, but is iterated more times and interleaved with communications of point marking in the halo regions. This is illustrated in Figure 4.4. The top-left processor, in the first run (middle panel), expands the marking in the right and bottom direction from the outer border of the computational grid and is thus unable to mark the small area under the hook. In the second sweep (right panel), the external marking is propagated from the marking done by the top-right processor. A number of iterations of at least half the number of processors per direction in the decomposition ensures that the initial marking obtained is the same that would have been computed in a serial run.

The one described above, is a good strategy to find an initial data and start the evolution having only the information carried by the point cloud. As a matter of fact,

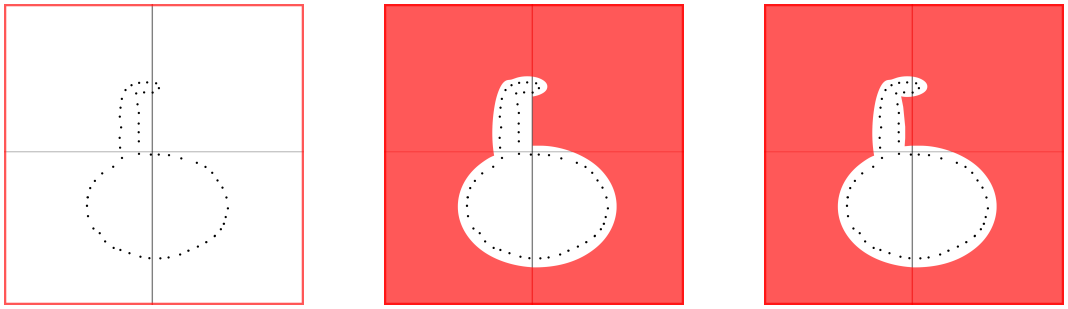

Figure 4.4: Illustration of marking external points in parallel run with a 2×2 domain decomposition (grey lines). Left: initial state. Center: after the first sweep. Right: after the second sweep.

one could also choose to evolve a less accurate initial guess (for example by choosing the initial data Γ^0 to be approximately the boundary of the computational domain Ω') until steady-state on a coarse grid and then use this final level set function as the initial data for a finer evolution. This initialization procedure, however, is not very costly and saves for waiting long times while the surface $\Gamma(t)$ moves form $\partial\Omega'$ to being close to $\mathcal{P}$.

4.4.5 Reinitialization

As already stated in Chapter 1 one of the nice feature of using a signed distance function to capture interfaces is that some geometric quantities, such as normals and curvatures, are easier to compute in terms of ϕ. In particular this is of paramount importance when the level set is used in ghost-fluid algorithms for the discretization of PDEs. The evolution determined by (4.9) will push the solution towards the data set, but will create sharp gradients in the level set function ϕ around it.

To keep $|\nabla\phi|$ close to 1, we employ a reinitialization procedure as described in Section 1.5; in particular the constrained version is preferred since it shows a better accuracy in preserving the location the zero set of ϕ during the reinitialization procedure.

Note that in parallel runs, each step of this algorithm also requires an update of the ϕ data in the halo region.

4.4.6 Localizing the computational effort

To localize our methods and reduce the computational effort, especially for the three-dimensional case, we involve the same narrow bands introduced in Section 1.6. In particular, in order to define the computational narrow bands, we set $\beta = 2\Delta x$ and $\gamma = 4\Delta x$, which, according to [94], are apt for the spatial reconstructions employed (multilinear and third-order WENO) and for the value of the CFL we will use in the numerical tests.

Moreover, if, in order to prevent from oscillations, we introduce the cut-off func-

tion $c(\phi)$ described by (1.55), the model equation (4.9) becomes

$$\phi_t(\vec{x},t) = c(\phi)\left[\frac{d(\vec{x})}{E_p(\phi)}\right]^{p-1}\left(\nabla d(\vec{x})\cdot\nabla\phi(\vec{x},t) + \frac{\eta}{p}d(\vec{x})\nabla\cdot\left(\frac{\nabla\phi(\vec{x},t)}{|\nabla\phi(\vec{x},t)|}\right)|\nabla\phi(\vec{x},t)|\right) \quad (4.31)$$

which states that outside the computational band $\mathcal{B}$ it is not useful at all to perform the computation, since the cut-off function would then overwrite the evolved values.

Therefore, the semi-Lagrangian schemes (4.22) and (4.24) are modified in order to incorporate the additional factor $c(\phi)$, such that the scale factor C_i^n in (4.22), (4.24) becomes

$$C_i^n = c(\phi_i^n)\Big[\frac{d(x_i)}{E_p(\phi^n)}\Big]^{p-1} \quad (4.32)$$

and the computations are restricted to the narrow bands described in Section 1.6.

As an example, the bands involved in the different parts of the scheme are depicted in a 2d case in Figure 4.5. Following the notation of Section 1.6 and referring to the left panel of Figure 4.5, the computational band for the update $\mathcal{B}^n$ is constituted by the points depicted in green. Moving to the right panel, one can observe that the reinitialization band $\overline{\mathcal{B}}^{n+1}$ is composed by all the points depicted in green, light blue and red. In particular, these red points are the ones detected as the closest points to the interface Γ^{n+1}, constituting the set $\mathcal{X}$ in the reinitialization scheme presented in Section 1.5.

Moreover, we point out that the addition of the first neighbours of active points, namely the light blue points, in the computational band for the reinitialization, is crucial to let the computational bands $\mathcal{B}^{n+1}$ move contextually with the zero level set of ϕ. Otherwise, the evolution will remain confined to the first computational band. In our surface reconstruction application, this would mean that if $\mathcal{P} \not\subset \mathcal{B}^0$ we could never hope to reach the point cloud.

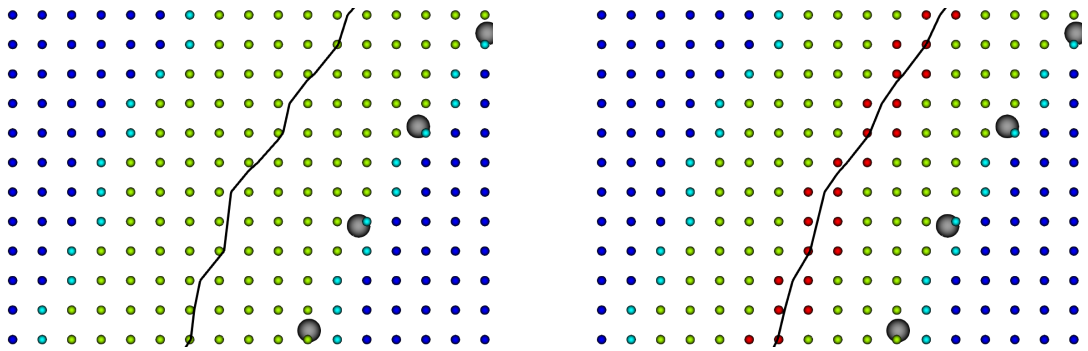

Figure 4.5: On the left, the mask that detects the computational subgrid $\mathcal{B}^n$, base of the level set function ϕ at time n. Active nodes are depicted in green, their first neighbours are depicted in light blue (they are inactive during the update, while they are active during the reinitialization step) and remaining inactive nodes are depicted in blue. On the right, the mask associated to the computational subgrid $\overline{\mathcal{B}}^{n+1}$ for the reinitialization of $\tilde{\phi}^{n+1}$. Colors are used in the same way, with in addition red nodes representing the nodes immediately close to the interface on which the signed distance function is computed explicitly with the one step procedure.

4.5 Numerical tests

In this section we present some representative numerical results obtained with the algorithm described in the previous sections. In order to compare easily the different $\mathcal{P}$, we have rescaled all the point clouds in the box $[-1,1]^d$.

As we are looking for a minimum of the energy functional (4.1), we will consider the same stopping criterion used in [68]: at time n, the algorithm stops if

$$\Delta_E^n = \frac{|\bar{e}_{n-1}^k - \bar{e}_n^k|}{\bar{e}_n^k} < 10^{-4}, \quad \text{where} \quad \bar{e}_n^k = \frac{1}{k} \sum_{i=n-k+1}^{n} E_2(\phi^i) \tag{4.33}$$

or after a maximum of 100 iterations. Note that in (4.33), the energy functional (4.1) is computed with $p = 2$. This condition is in practice a way to detect stationary points or flat areas of the energy functional. In all the tests we set $k = \min(n, 10)$ and forced the algorithm to do at least 10 iterations.

We point out that some other stopping criteria can be considered. In [124] the authors propose to stop the evolution either all data points are close enough to the zero level set, i.e. $|\phi(Q)| < tol$ for any $Q \in \mathcal{P}$ or $|\phi_t|$ is small enough, meaning that we are close to an equilibrium state. Alternatively, the mean of squared differences of two subsequent time steps is tested in [62] is order to get a right choice of the number of time steps for the model creation.

4.5.1 Quantitative evaluation

In order to get a quantitative insight into the quality of the reconstruction achieved with our scheme and compare it with already existent methods, different quantities are considered. In all the tests we compute, alongside (4.33), the normalized L^1-norm of the update between two successive iterates, namely

$$\epsilon_1^n = \frac{\sum_{\vec{x}_j \in \mathcal{G}} |\phi_j^{n+1} - \phi_j^n|}{\sum_{\vec{x}_j \in \mathcal{G}} |\phi_j^n|}, \tag{4.34}$$

as in [26], and and the L^1-norm of the error, when the exact signed distance function ϕ^* is given.

Furthermore, we compute the average of the error on the points of the cloud

$$Err_{\mathcal{P}}^n = \frac{\sum_{Q \in \mathcal{P}} |I[\Phi^n](Q)|}{|\mathcal{P}|}, \tag{4.35}$$

to make some considerations on the role of the curvature regularization and to evaluate how much the final reconstruction is attached to the data set.

4.5.2 Choice of parameters

It only remains now to detail the choices for the parameters p and η and for the spatial and temporal discretization.

r	p	η
1	1	0.05
2	2	0.05
≥ 3	2	1

Table 4.1: Standard values for p and η in each run r of Algorithm 1.

In [26], the authors consider the case with $p = 1$, which grants for a faster evolution of the initial data towards the cloud and also does not require to compute the factor $[d(\vec{x})/E_p]^{p-1}$, while in [124] the authors suggest to set $p = 2$, which is more effective in reaching a steady state for the evolution. Also, setting $\eta = 0$ simplifies the update formulas (4.22) and (4.24) since all $x^*_{j,i}$ coincide. Furthermore, disregarding the curvature effect, it prevents from loosing too much details, especially on coarse grids. On the other hand, increasing values of $\eta \geq 0$ will smoothen the solution and will help handling changes of topology, at the price of having a zero level set slightly off from the cloud $\mathcal{P}$.

Our goal is to compute a levelset with $p = 2$ on a fine grid. In order to save computational time, we combine different choices of the parameters in more than one run of Algorithm 1, gradually increasing the resolution of the grid.

Let r represent the number of the run of Algorithm 1, Table 4.1 summarizes the usual approach. We point out that $\eta = 0.05$ is chosen accordingly to the analysis made in [62] and also with the aim of better handling possible changes in topology, while $\eta = 1$ is chosen in order to reproduce the original model. On the other hand, we will see that the choice $\eta > 1$ might be useful when the dataset is affected by noise in order to further smoothen the solution.

Regarding the spatial and temporal discretization we set

$$\Delta x^{(r)} = \frac{C_{\Delta x}}{2^{r-1}} h_{\mathcal{P}}, \quad \Delta t^{(r)} = \Delta x^{(r)}, \tag{4.36}$$

where r is the number of the current run, $h_{\mathcal{P}}$ is specified below, and, unless otherwise specified, $C_{\Delta x} = 1$. The above spatio temporal discretization clearly states the advantages of using the SL approach since in (4.36) we set $\Delta t = \mathcal{O}(\Delta x)$, not being prohibitively constrained by the parabolic term. In the usual procedure three runs of Algorithm 1 are performed.

In (4.36) $h_{\mathcal{P}}$ represents an estimate of the resolution of the cloud $\mathcal{P}$; its value is approximated by randomly choosing a sample made up of the 10% of the points in $\mathcal{P}$ and then computing the average of the distances between each of these points and their nearest neighbour in $\mathcal{P}$.

In the first and coarsest run ($r = 1$), we compute the initial data as described in subsection 4.4.4. The required threshold is set as $\gamma_{\mathcal{P}} = K_{\mathcal{P}}\, h_{\mathcal{P}}$ with $K_{\mathcal{P}}$ set to 2.0, unless otherwise specified. The role of $K_{\mathcal{P}}$ is relevant when the points in $\mathcal{P}$ are not evenly distributed on the sampled surface. It is in fact common, especially in 3d, due to the supports used for the object during laser scanning, to find data sets that have piggy bank-like shapes, which have fake holes that could distort the result. In such cases, the good practice consists in choosing a larger $K_{\mathcal{P}}$, to start with a level set that is further away from the data, but encloses the point cloud without

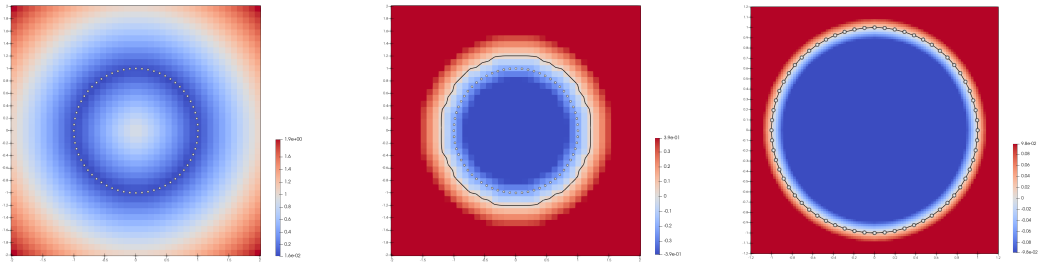

Figure 4.6: Steps of the algorithm for the 2d circle case: on the left, the distance function is represented; the central panel shows the initial data and its zero level set (black line); the final data and its contour (black line) are represented on the right. Data represented here has been obtained with WENO reconstruction.

entering in the fake cavities. For the later runs ($r > 1$), we use as initial data the final solution computed by the previous run and interpolated on the current grid.

All the codes have been written in C++ language with the support of the MPI and of the PETSc library [7, 6]. In particular, the communications in the SL step 4c of Algorithm 1 have been performed with the help of a DMSWARM object. The tests have been performed on the Galileo100 cluster hosted at CINECA[1], exploiting the resources assigned to ISCRA-C Projects[2].

4.5.3 2d data sets

Circle test

We first consider a data set $\mathcal{P}$ made up of 64 points uniformly chosen on a circle of radius 1, thus having a cloud size approximately equal to 9.81×10^{-2}.

The main steps of the reconstruction of the circle are depicted in Fig. 4.6, while Fig. 4.7 collects significant graphs to evaluate the accuracy of the method. The three runs of the algorithm required respectively 18, 20 and 12 iterations using the Q1 reconstruction, while 20, 13 and 33 iterations are required in the WENO case, with an amount of computational time equal to $9.40e-02$ and $1.31e-01$ seconds, respectively.

Looking at Fig. 4.7 note how the zero level set approaches the cloud: as expected, the normalized L^1-norm of the update does not show a monotone profile and the error on the cloud $Err^n_{\mathcal{P}}$ is not vanishing due to the finite grid size and to the curvature term. During the first run, the interface quickly moves towards the data,

[1] https://www.hpc.cineca.it/systems/hardware/galileo100/

[2] *Surface Reconstruction with Level Set Method* (HP10CPQ93M)
Parallel Scalability of Surface Reconstruction with Level Set Method (HP10COSEJL)
Adaptive Mesh Refinement in Level Set Methods (HP10C7HWOL)

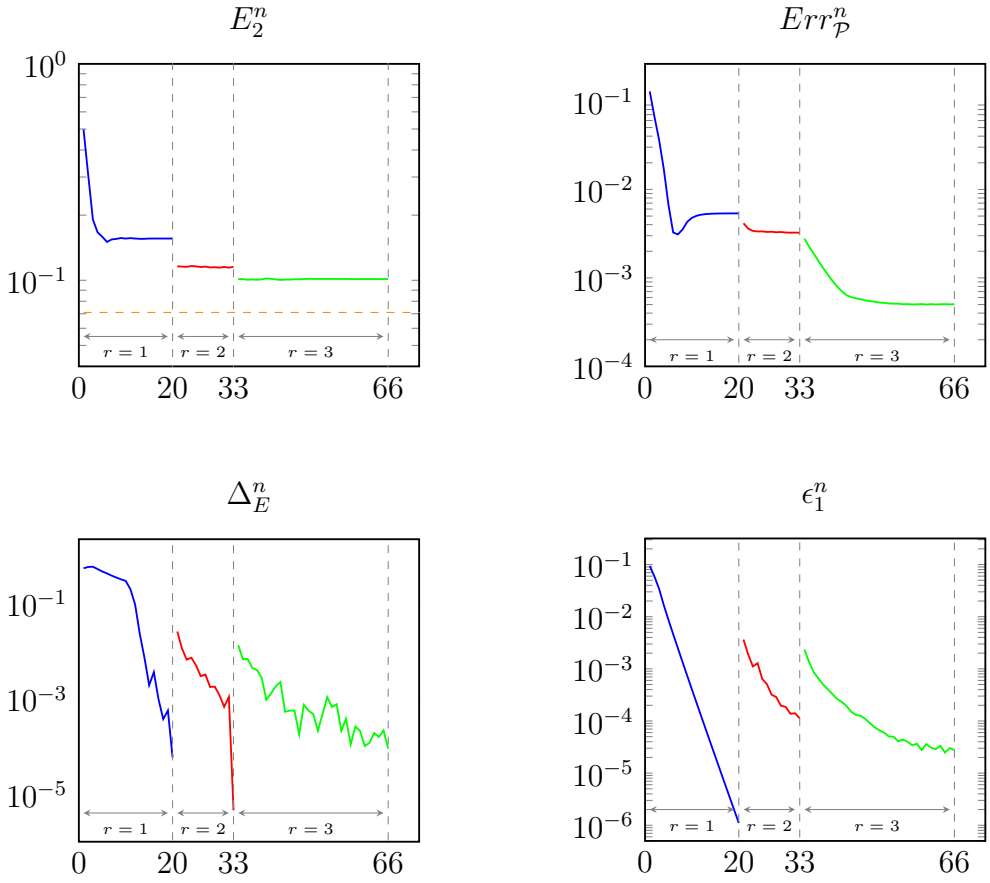

Figure 4.7: Significant quantities computed for the circle test using WENO interpolator. In the top-left panel the energy functional is depicted: the drop between two successive runs is due to the change of the grid size and the consequent new computation of the distance function. In the top-right panel the error computed on the cloud is shown: as expected, its profile is not monotone since during the evolution the interface can even pass the cloud and then come back. The bottom-left panel represents the running average of the energy functional: note that in the first 10 iterations of each run the moving average is still forming up. The bottom-right panel represents the normalized L^1-norm of the update between two successive iterates.

		Q1			WENO		
r	Grid	L^1-err	Energy	Error on $\mathcal{P}$	L^1-err	Energy	Error on $\mathcal{P}$
1	42×42	$8.01e-02$	$1.76e-01$	$1.29e-02$	$7.49e-02$	$1.57e-01$	$1.20e-02$
2	58×58	$9.74e-03$	$1.19e-01$	$3.10e-03$	$1.01e-02$	$1.15e-01$	$3.86e-03$
3	99×99	$1.71e-03$	$1.04e-01$	$6.43e-04$	$1.48e-03$	$1.01e-01$	$5.05e-04$
4	181×181	$6.80e-04$	$1.04e-01$	$5.06e-04$	$6.29e-04$	$1.03e-01$	$4.61e-04$
5	344×344	$2.73e-04$	$1.04e-01$	$3.77e-04$	$2.48e-04$	$1.04e-01$	$3.19e-04$

Table 4.2: Errors and energy computed for the circle test at the end of each run. To better appreciate the results Algorithm 1 has been performed five times, fixing the parameters according to Table 4.1. Note that, the better results on WENO, especially in terms of the error computed on the cloud $\mathcal{P}$, are linked to a greater ability to be faithful to the data set and to capture details.

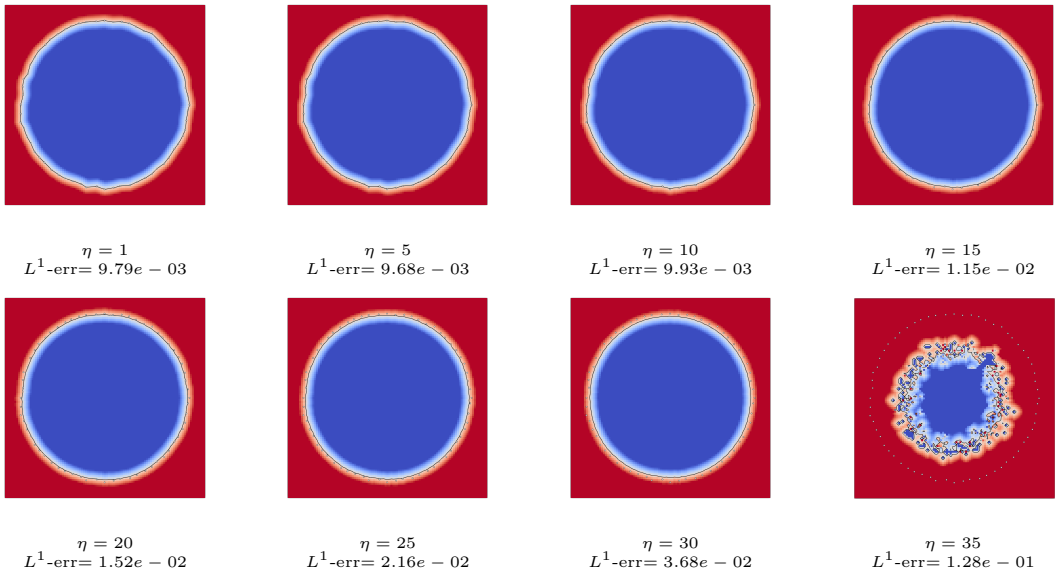

Figure 4.8: Reconstructions form noisy data. A 10% of normally distributed noise is added to $\mathcal{P}$ and increasing values of η in the final run are considered. The solution keeps smoothing, until it degenerates ($\eta = 35$).

		Q1			WENO		
r	Grid	L^1-err	Energy	Error on $\mathcal{P}$	L^1-err	Energy	Error on $\mathcal{P}$
1	30×30	$1.01e+00$	$3.76e-01$	$9.44e-02$	$9.09e-01$	$3.33e-01$	$8.07e-02$
2	34×34	$1.18e-01$	$2.65e-01$	$3.08e-02$	$7.71e-02$	$2.96e-01$	$1.69e-02$
3	51×51	$1.27e-02$	$2.39e-01$	$4.71e-03$	$9.69e-03$	$2.32e-01$	$3.43e-03$
4	85×85	$4.30e-03$	$2.38e-01$	$7.10e-03$	$3.01e-03$	$2.37e-01$	$4.37e-02$
5	153×153	$1.45e-03$	$2.42e-01$	$5.95e-02$	$1.04e-03$	$2.42e-01$	$4.16e-02$

Table 4.3: Errors and energy computed for the square test at the end of each run. To better appreciate the results, Algorithm 1 has been performed five times.

it can even pass through the cloud, and then stabilizes. During the successive runs the reconstruction becomes more accurate: the energy gets closer to the exact value of the energy computed for a circle, represented by the dashed orange line ($\approx 7.10 \times 10^{-2}$), and the error made on the cloud decreases. In this case, the exact signed distance function from the circle is known and we can compute the L^1-norm of the error (see Table 4.2).

Finally, we perform the circle test adding a normally distributed noise to the data (10%). Fig. 4.8 shows the WENO reconstructions corresponding to increasing values of η in the last run: for $\eta = 5$ we have the best result in term of L^1 error, while for larger values of η we get a smoother solution, but a larger L^1 error due to the shrinking of the level sets, until, for $\eta = 35$, the zero level set completely degenerates.

Square test

In the second test we consider a cloud made up of 24 points representing a square rotated by 45 degrees with respect to the Cartesian axis. The reconstruction takes

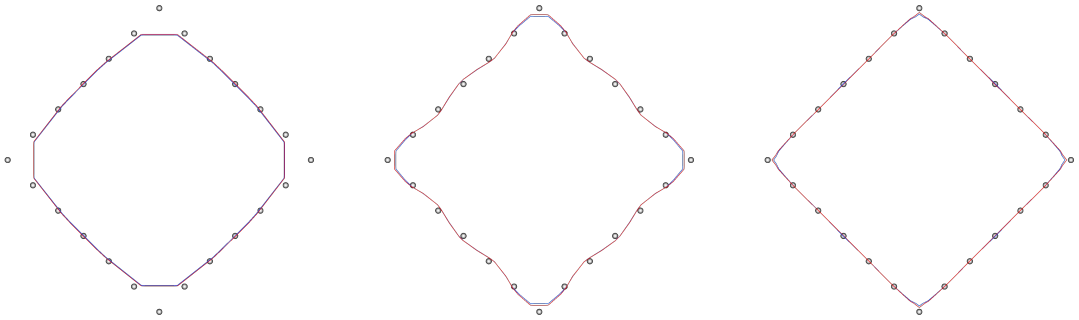

Figure 4.9: From left to right: a comparison between the curves reconstructed with Q1 (blue line) and WENO (red line) interpolators at each run.

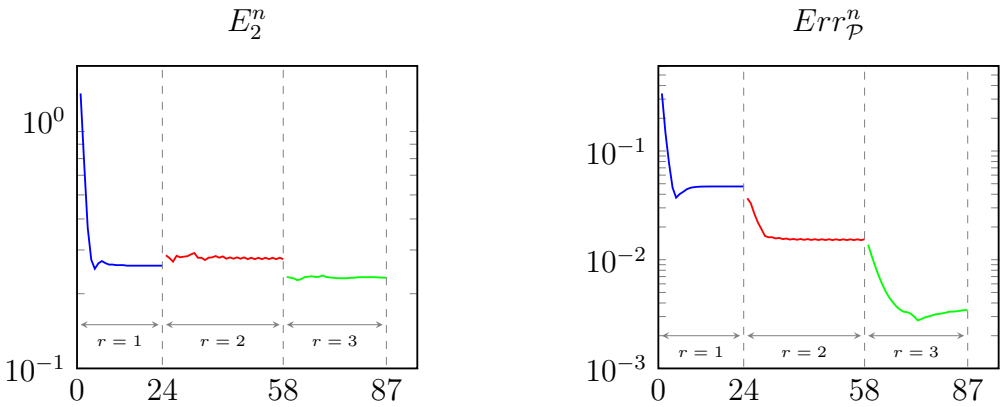

Figure 4.10: Energy and error on cloud computed for the square test using WENO interpolator.

	Q1			WENO		
r	L^1-err	Energy	Error on $\mathcal{P}$	L^1-err	Energy	Error on $\mathcal{P}$
1	$5.56e-02$	$2.12e-01$	$4.60e-03$	$4.97e-02$	$1.90e-01$	$4.05e-03$
2	$9.02e-03$	$1.74e-01$	$1.81e-03$	$8.73e-03$	$1.71e-01$	$1.93e-03$
3	$3.65e-03$	$1.61e-01$	$1.56e-03$	$3.27e-03$	$1.57e-01$	$1.37e-03$

Table 4.4: Errors and energy computed for the sphere at the end of each run.

respectively 23, 26, 14 iterations and $1.08e-01$ seconds in the Q1 case, and 24, 34, 29 iterations and $1.50e-01$ seconds in the WENO case. The final curves of each run are depicted in Fig. 4.9. Graphs of the energy functional and of the error on the cloud are reported in Fig. 4.10, exclusively for the WENO case. It is worth highlighting the role of μ: the curvature regularization in the last run penalizes the sharp corners, thus providing a final contour with a controlled maximum curvature, which, although the higher resolution of the grid, is not passing exactly through the corresponding points of the cloud.

Table 4.3 shows the errors computed with respect to the exact data. We point out that in this test the WENO based algorithm leads to significantly lower errors, due to the differences near the corners.

4.5.4 Synthetic 3d data sets

In this subsection we present numerical tests for the reconstruction of 3d shapes. Here the data sets are synthetic and are made up by sampling points on simple geometrical forms for which the exact signed distance function is known.

Sphere

The first 3d test has been performed on a point cloud made up of 2562 points chosen on a sphere of radius 1. The reconstruction procedure is illustrated in Fig. 4.11 for the WENO case with the original data set (first row) and a perturbed one (second row, adding 10% of normally distributed noise). In this simple case it is worth emphasizing that the role of the curvature regularization is to provide a smoother final surface than the ones provided by the previous two runs. In fact, setting $\mu = 10$ in the last run of the noisy case, we aimed at smoothing the final surface that would have been sensibly rough with the usual setting of the parameter. Looking a Table 4.4 one can also compare the results obtained with different interpolators and observe the better performance of the WENO interpolator.

Cube&Spheres

The second test concerns a three-dimensional domain composed by a cube joined with three spheres, sampled by 2346 points. The cube edge length was 0.8, the first sphere has radius 0.25 and centre at the middle of an edge of the cube, while the other two had radius 0.15 and were centred onto the two vertices of the opposite edge of the cube. The geometrical object was rotated in such a way that no face nor edge were aligned with the background Cartesian grid.

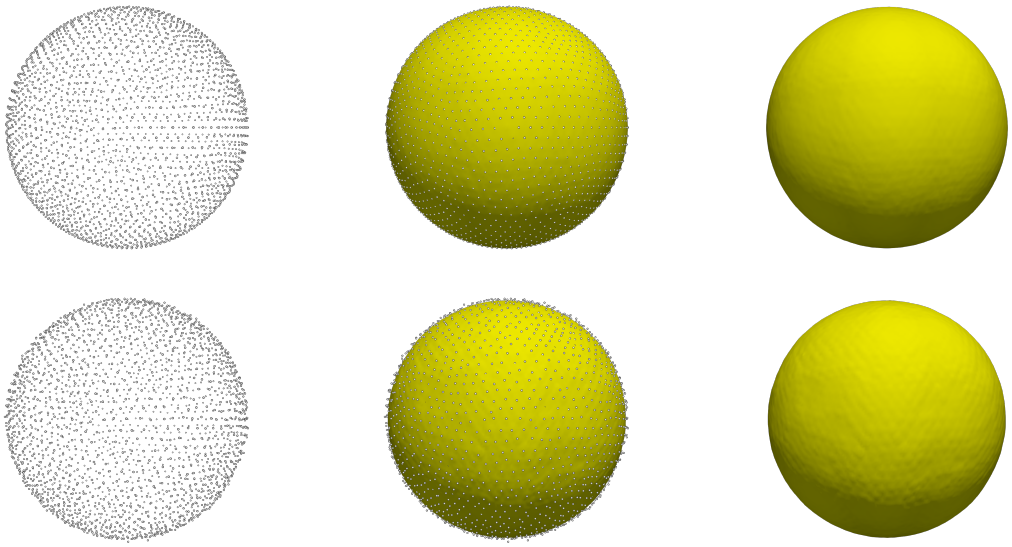

Figure 4.11: Reconstruction steps of a sphere for the WENO case. In the first row the point cloud and the final zero surface with and without the point cloud are depicted for the unperturbed case. In the second row the same figures are produced adding a 10% of noise to the dataset and setting $\mu = 10$ in the last run in order to smoothen the solution.

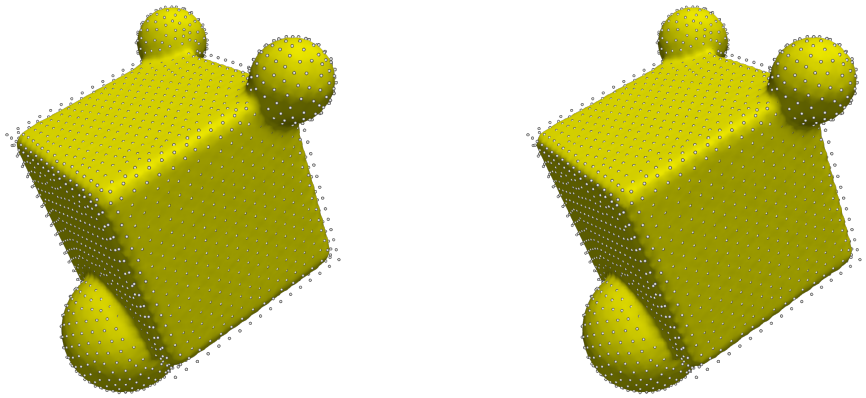

Figure 4.12: The reconstructed surfaces of the "Cube&Spheres". On the left, the result obtained with the Q1 interpolant, while on the right, the result obtained using the WENO interpolant.

	Q1			WENO		
r	L^1-err	Energy	Error on $\mathcal{P}$	L^1-err	Energy	Error on $\mathcal{P}$
1	$9.86e-02$	$1.82e-01$	$1.05e-02$	$9.34e-02$	$1.66e-01$	$8.92e-03$
2	$2.90e-02$	$1.55e-01$	$4.64e-03$	$2.85e-02$	$1.53e-01$	$4.18e-03$
3	$1.34e-03$	$1.46e-01$	$4.91e-03$	$1.30e-02$	$1.42e-01$	$3.95e-03$

Table 4.5: Errors and energy computed for the Cube&Spheres at the end of each run.

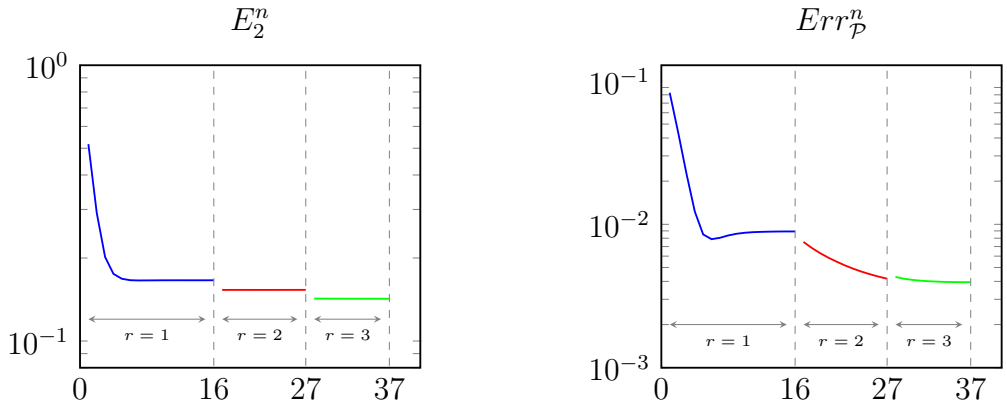

Figure 4.13: Energy and error on cloud computed for the "Cube&Spheres" using WENO interpolator.

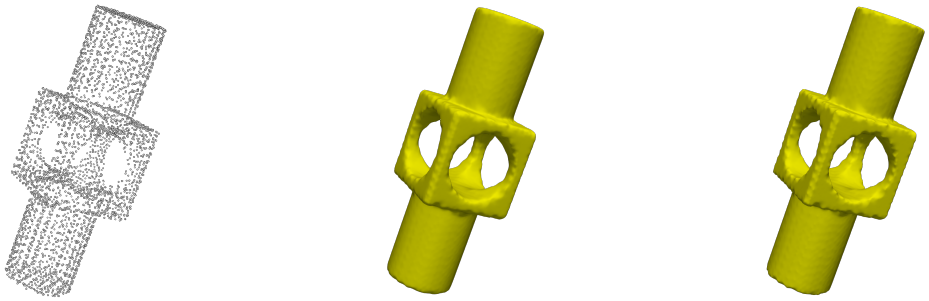

Figure 4.14: Mechanical part: the dataset, and the zero level set obtained with Q1 (center) and WENO (right) interpolator.

Results are shown in Fig. 4.12 and in Table 4.5. As for the 2d case of the square, the curvature regularization provides rounded edges and corners in the final reconstruction, especially in the Q1 case. Here too the WENO reconstruction shows better performances according to all three measures and especially to the error on the point cloud. Plot of the energy functional and of the error on the cloud are shown in Fig. 4.13 for the WENO case. Note in particular that the minimum of the error on $\mathcal{P}$ is reached at the end of the second run and is about $4.64e-03$, while at the end of the third run the error is about $4.91e-03$, which is an effect of the curvature regularization.

Mechanical part and knot

In the following tests we consider two point clouds that has been tested also in reference works as [71, 124].

The first one is made up of 4102 points representing a mechanical part which presents both rounded and sharp features. The results are depicted in Fig. 4.14 where one can notice how the WENO reconstruction fits more faithfully the dataset; in fact, the final errors computed on $\mathcal{P}$ are equal to $7.59e-03$ and $3.74e-03$, respectively for the Q1 interpolator and the WENO one.

The second dataset consists of 10000 points and represents a knot. Motivated

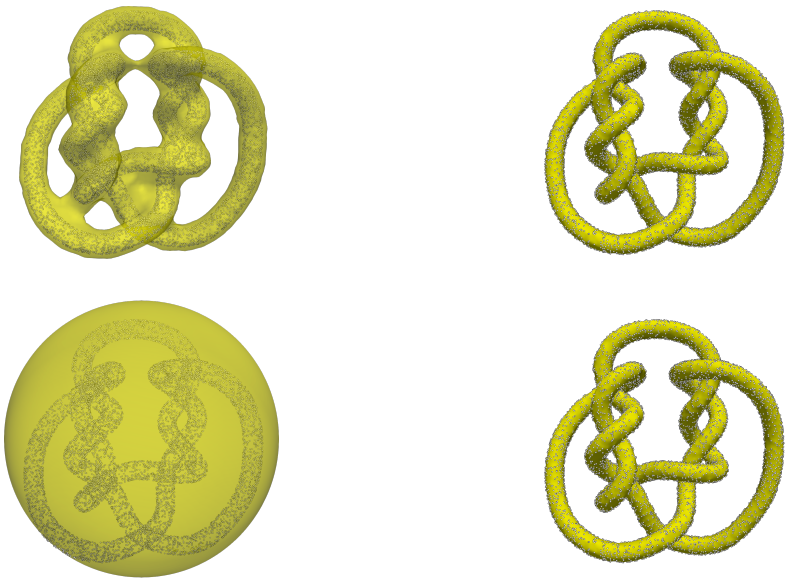

Figure 4.15: The reconstruction procedure for the knot point cloud, using WENO interpolator. First row: usual initial guess obtained as a proper isocontour of the distance function from $\mathcal{P}$. Second row: ellipsoidal initial guess.

by its intriguing topological features, we explored the capability of our method to accurately reconstruct its shape, even when we start from an initial data with a completely different topology. If Fig. 4.15 we show the initial and final data of the reconstruction procedure using the WENO interpolator setting $C_{\Delta x} = 2$: in the first row the initial data is computed as usual, choosing $K_{\mathcal{P}} = 10$, while in the second row the initial data is chosen to be an ellipsoid encompassing the dataset (the maximum number of iterations in this latter case has been set to 200). Of course, we observe a difference in the number of iterations: the first case requires 61, 23 and 10 iterations, while the second one requires 120, 24 and 14. The computational times are reported in Tab. 4.9.

4.5.5 Data sets from laser scans

Here we test our method using data sets coming from laser-scanning of real objects. These point clouds are made available in the Digital Shape WorkBench of the AIM@SHAPE project [2] or in the 3D Scanning Repository of the Stanford University [109].

The "Frog" and the "Bunny"

We first consider a point cloud named "Frog" [2] and one named "Bunny" [109], made up of 2512 and 35947 points, respectively. The results are shown in Fig. 4.16 and in Tables 4.6, 4.7 and we point out that for the "Bunny" we set $C_{\Delta x} = 2$. In both these two cases the data sets have holes in the bottom and that's why we had to increase the threshold $\gamma_{\mathcal{P}}$ for the initial data computation, thus we set $K_{\mathcal{P}} = 10$. This obviously produced an enlarged initial contour and therefore more iterations

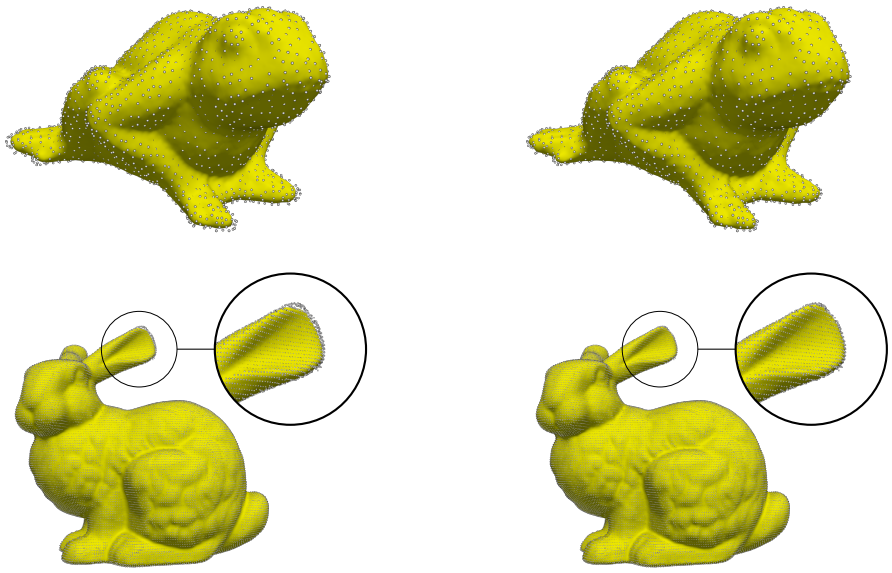

Figure 4.16: The reconstructed surfaces of the “Frog" and the “Bunny", respectively on the first and on the second row. On the left, the result obtained with the Q1 interpolant, while on the right, the result obtained using the WENO interpolant. One can appreciate how the WENO recostruction better recovers the details of the paws, in the “Frog" case, and of the ear, in the “Bunny" case.

	Q1		WENO	
r	Energy	Error on $\mathcal{P}$	Energy	Error on $\mathcal{P}$
1	$2.70e-01$	$3.97e-03$	$2.64e-01$	$3.54e-03$
2	$2.77e-01$	$3.17e-03$	$2.75e-01$	$2.55e-03$
3	$2.82e-01$	$3.33e-03$	$2.82e-01$	$2.34e-03$

Table 4.6: Errors and energy computed for the “Frog” at the end of each run.

	Q1		WENO	
r	Energy	Error on $\mathcal{P}$	Energy	Error on $\mathcal{P}$
1	$9.80e-02$	$3.63e-03$	$8.78e-02$	$3.16e-03$
2	$8.00e-02$	$1.38e-03$	$7.62e-02$	$1.18e-03$
3	$7.23e-02$	$1.09e-03$	$7.08e-02$	$8.09e-04$

Table 4.7: Errors and energy computed for the “Bunny” at the end of each run.

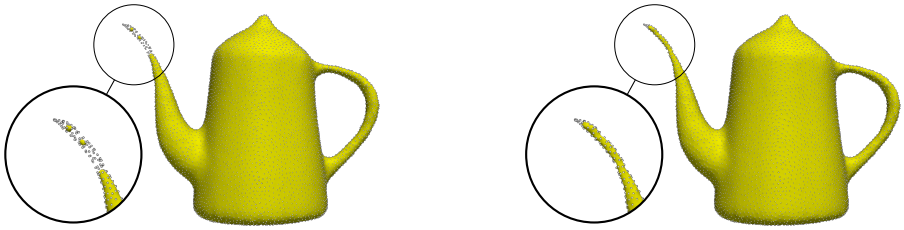

Figure 4.17: Final reconstructions of the "Teapot" with WENO interpolator. The left panel collects the result obtained with $C_{\Delta x} = 0.5$, while on the right $C_{\Delta x} = 0.25$. The first setting fails in recovering the spout, while the second one succeeds.

were needed during the first run, but this simple strategy secured us from getting distorted results deriving from a bad initial data. Of course, alternatively, one could also fill these holes in an ad-hoc preprocessing stage by adding some points to the cloud.

Note in particular, from Fig. 4.16, that the high order interpolator provides a more faithful reconstruction, which can be also visually appreciated from some details of the dataset where the cross section of the point cloud it's comparable with its resolution.

A teapot with tiny details

In the next test we focus our attention on the ability of our algorithm to capture tiny details of an object. A point cloud named "Teapot" [2] has been considered for this aim since it presents a more complex topology and parts whose cross section is more or less comparable with the size of the cloud. In Fig. 4.17, the reconstructions obtained with the WENO interpolator, choosing respectively $C_{\Delta x} = 0.5$ and $C_{\Delta x} = 0.25$, are depicted. We point out that in is this test we force $\mu = 0$ in the first two runs in order to retain as much details as possible. In fact, higher values of μ expose to the risk of loosing important details of the shape. The higher resolution of the grid of course requires a great computational effort: the finer reconstruction took approximately six times as long as the first (see Table 4.9).

Complex shapes and topology: the "Happy Buddha" and the "Dragon'

Finally we have done some tests on very complex point clouds present in the Stanford 3D Scanning Repository [109]. We considered two data sets, named "Happy Buddha" and "Dragon" respectively, because they present some nice features like free holes, small bridges due to the carving and many details to be recovered. The results of our reconstructions are depicted in Fig. 4.18 and has been obtained setting $K_{\mathcal{P}} = 10$ in the computation of the initial data, and $C_{\Delta x} = 4$ in order to limit the huge amount of grid points.

On the "Happy Buddha" test, we point out that the reconstructed surface recovered equally well both the very flat surfaces of the base and the tiny details on the sides, on the belly and on the drapery. Also all the holes of the highly nontrivial

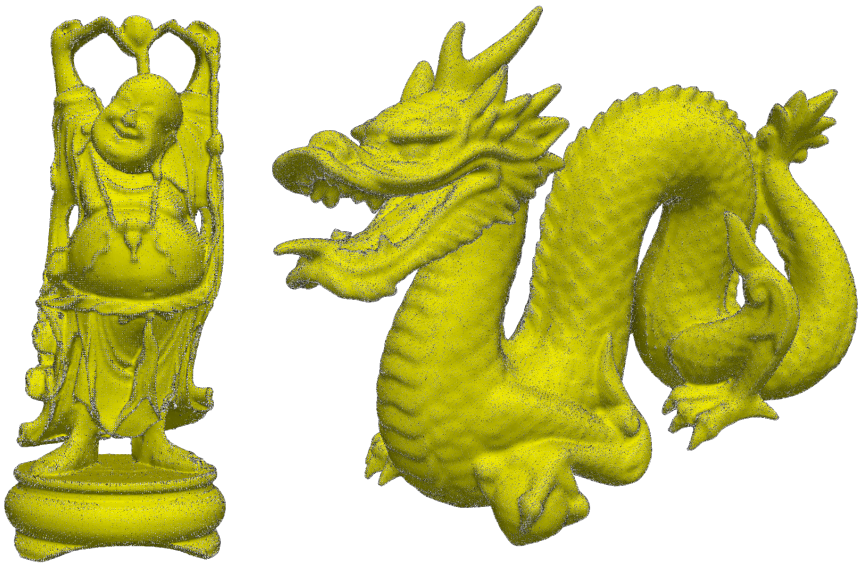

Figure 4.18: Final reconstructions of the "Happy Buddha" and the "Dragon" using WENO interpolator.

topology were correctly recovered, including the small ones on the sides: we point out that the initial data had detected the two big holes at the top but missed the two tiny ones at the sides which were recovered during the surface evolution, confirming the ability of the method to deal with topological changes of the surface.

On the "Dragon" test, we point out that the scales on the skin are still well approximated despite the curvature regularization. Also, we stress how the level set has been pushed inside the mouth during its evolution and how the sharp teeth shapes have been well approximated.

4.5.6 Detecting tunnels

In this Subsection, we would like to point out a drawback of the reconstruction method presented here, suggesting a possible way to overcome this issue.

More specifically, there exist some situations in which the evolution of the level set function ϕ stops and reaches a local minimum which is not even close to the desired one. This is for example the case of *tunnels*, and by tunnels we mean all the situations, like the one in Figure 4.19 in which there is a portion of the point cloud $\mathcal{P}$ which is distributed along two close parallel lines in the 2d case or two close parallel planes or even on the curved surface of a cylinder in the 3d case; see Figure 4.20. This is topologically equivalent to the case of holes in the "Teapot" and the "Happy Buddha" tests, but metrically different. By "tunnels" we mean "holes" that are quite long with respect to the size of their cross section.

In such situations, at some point, the gradient of the level set function ϕ and the gradient of the distance to $\mathcal{P}$, which leads the evolution, become orthogonal and the evolution stops, even if the distance function is still large. Although we known that changing the initial data and choosing an initial zero set which starts already inside

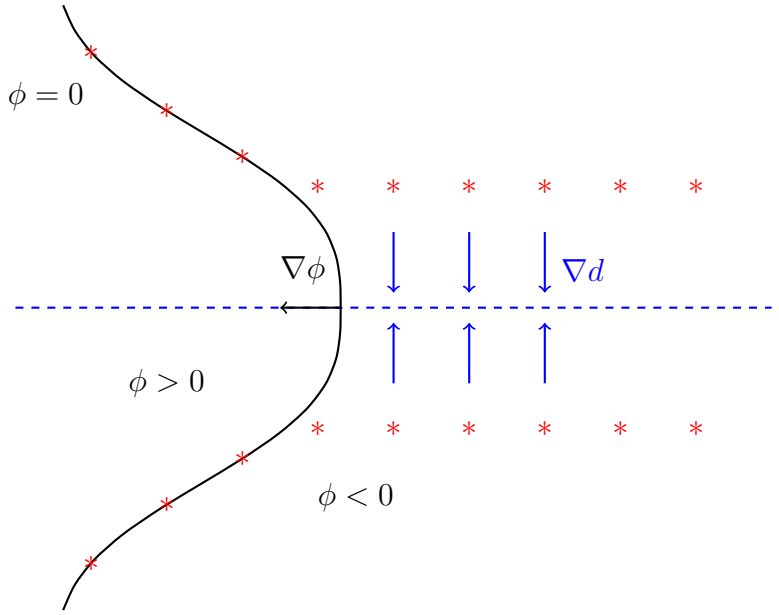

Figure 4.19: Configuration of a tunnel in a point cloud $\mathcal{P}$, represented by the red star points. The evolution of ϕ initially pushes its zero level set inside of the corridor, but then is slowed down until it stops due to the vanishing of the transport term. Approaching the tunnel the gradient of ϕ and the velocity field ∇d become orthogonal.

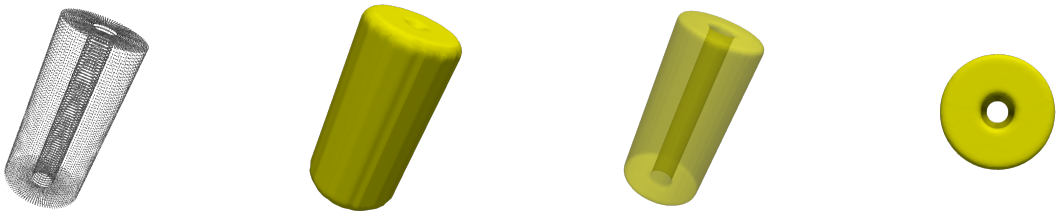

Figure 4.20: The pierced cylinder reconstruction procedure: given a point cloud $\mathcal{P}$ sampled from a cylinder presenting a long and narrow hole in the center (first panel), we recover its shape starting from an initial data which is topologically equivalent to a sphere (second panel). Substituting the velocity field ∇d with $\nabla\phi$ we can a final reconstruction that faithfully represents the shape of the cylinder, including the whole (third and fourth panel).

the cavity can solve the situation, from a mathematical point of view one would not like to have such a dependence on the initial guess. The point is how to distinguish these situations from other ones like for instance the ones of the “Frog” and the “Bunny” in which the plugging behaviour of the level set function is the desirable one.

The trick we propose is to check the following conditions: if $d(x_i) < 2\Delta x$ and $|\nabla d(x_i)| < 0.5$, we replace the velocity field ∇d with $\nabla \phi$ which in practice, supposing that the level set function is coming from the "outside" of the cavity, is a way of letting the level set function proceed in the same direction followed so far.

In fact, a condition on the scalar product $\nabla \phi \cdot \nabla d$ might be too strong and might expose to the risk of perturbing the evolution even in well resolved areas. On the other hand, we expect that in correspondence of a tunnel, the two computational bands built around the different sides of the tunnel merge with each other including the points located in the middle in which the distance function presents dangerous kinks. In a continuity framework this of course poses a serious issue, but in the meanwhile, in the discretized setting, it offers a criterion for the detection of such regions.

A specific investigation of this issue has been done on a synthetic object constituted by a cylinder with a hole connecting the two parallel faces and the result are showed in Figure 4.20; the tunnel in the cylinder has been detected even if the initial data exhibits a different topology and this wouldn't have happened applying the original scheme.

4.6 Scalability of the algorithm

Our algorithm, thanks to the localization of the computational effort on a band around the evolving zero level set, which is a codimension 1 variety of $\mathbb{R}^n$, has a cost that scales as $\mathcal{O}(1/\Delta x^{n-1})$ under grid refinement. This is confirmed by comparing the two “Teapot” experiments (Table 4.8 and 4.9). The memory footprint of our implementation scales instead as $\mathcal{O}(1/\Delta x^n)$, due to the data attached to the full grid $\mathcal{G}$. This has suggested to consider a distributed memory parallel implementation based on the MPI paradigm.

In this last subsection we have collected some scalability results to evaluate the efficiency of the algorithm. The test depicted in Fig. 4.21 has been performed using the “Dragon” point cloud (setting $C_{\Delta x} = 2$) progressively increasing the number of nodes, while keeping constant the number of processors per node.

Computing the parallel efficiency of a run with M cores as $\epsilon = \frac{40 \cdot T_{40}}{M \cdot T_M}$, where T_M is the computational time, we obtain 84%, 63%, 58%, respectively for 80, 120 and 160 cores. This can be ascribed to the interplay between the Cartesian grid partitioning for $\mathcal{G}$ and the localization techniques of subsection 4.4.6. In fact, when the number of cores is increased, also the chance of having cores with no (or small) intersection with the active computational band is also increased, with a negative impact on load balancing.

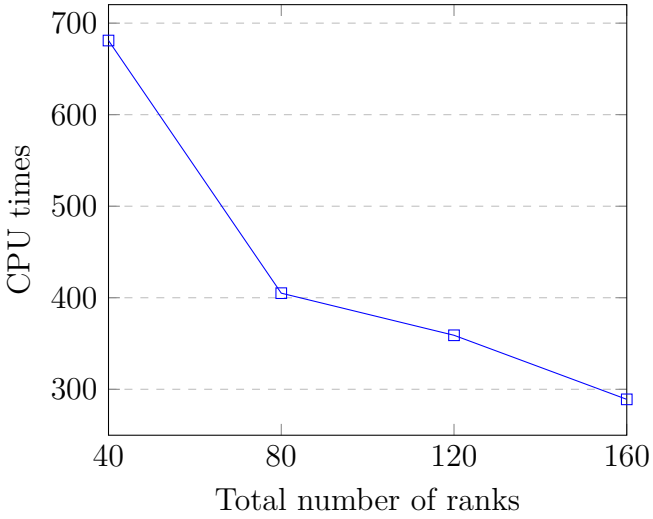

Figure 4.21: Scalability results for the "Dragon" reconstruction. The blue marks report the CPU times in seconds required using a fixed number of ranks per node (40) and gradually increasing the number of nodes, from 1 to 4.

Point cloud	Grid		
	$r = 1$	$r = 2$	$r = 3$
Sphere	$54 \times 54 \times 54$	$78 \times 78 \times 78$	$135 \times 135 \times 135$
Cube&Spheres	$55 \times 62 \times 55$	$81 \times 94 \times 81$	$141 \times 166 \times 141$
Frog	$89 \times 75 \times 87$	$116 \times 89 \times 113$	$211 \times 157 \times 204$
Bunny	$109 \times 108 \times 92$	$177 \times 175 \times 142$	$332 \times 329 \times 262$
Knot	$105 \times 106 \times 67$	$169 \times 171 \times 93$	$317 \times 321 \times 165$
Knot (ellipse)	$105 \times 106 \times 67$	$169 \times 171 \times 93$	$317 \times 321 \times 165$
Mechanical part	$77 \times 77 \times 148$	$92 \times 92 \times 234$	$163 \times 163 \times 446$
Teapot	$119 \times 81 \times 102$	$176 \times 101 \times 143$	$330 \times 181 \times 264$
Teapot (finer)	$184 \times 109 \times 151$	$330 \times 181 \times 264$	$638 \times 340 \times 506$
Happy Buddha	$152 \times 331 \times 152$	$272 \times 631 \times 272$	$522 \times 1241 \times 522$
Dragon	$263 \times 193 \times 132$	$494 \times 354 \times 233$	$966 \times 687 \times 444$

Table 4.8: Dimensions of the Cartesian grid involved for each 3d test at each progressive run.

Point cloud	Ranks	Total CPU time	
		Q1	WENO
Sphere	16	0.03	0.05
Cube&Spheres	16	0.03	0.05
Frog	16	0.08	0.11
Bunny	16	0.28	0.34
Knot	16	0.27	0.42
Knot (ellipse)	16	0.32	0.68
Mechanical part	16	0.11	0.36
Teapot	32	0.43	0.58
Teapot (finer)	32	2.15	3.58
Happy Buddha	32	1.35	2.00
Dragon	32	1.48	2.15

Table 4.9: Computational times (min) of the algorithm.

Chapter 5
Adaptive Mesh Refinement

So far, the main topics of this thesis have revolved around SL schemes, high-order local interpolators, the LSM and, in particular, the application of these techniques to the HJB equations and to the design of an algorithm for surface reconstruction from point clouds. Moreover, in what precedes, we have always been concerned about the computational efforts required by the techniques involved, emphasizing the features of different interpolators, the unconditional stability of the SL scheme, and the importance of localization when dealing with the LSM. As a matter of fact, focusing specifically on the case of surface reconstruction detailed in Chapter 4, we have also seen how the advantages carried by these techniques might be strongly limited by the use of a uniform Cartesian mesh. Instead, the considerations made above, tempt us to abandon this Cartesian uniform framework in favour of an adaptive one. Thus, to conclude our dissertation, we will present in this chapter an extension of the numerical techniques seen before, on Quadtree, in two spatial dimensions, and on Octree, in three spatial dimensions, focusing in particular to the application on surface reconstruction, as described in Chapter 4. We believe that this kind of meshes allow for a smaller memory footprint and computational cost since they can concentrate the efforts near the evolving front, but also retain the simplicity of point location typical of Cartesian grids.

In the level set context, the choice of extending the previous techniques to AMR is naturally prompted first of all by the advantages of working with level set functions to detect an interface. In fact, given a level set function ϕ associated to an interface Γ, constructing a tree-based grid refined around Γ is quite simple since the values of $|\phi|$ naturally give a proper criterion for refinement which causes to have a smaller grid size close to Γ and a larger one when we are far away from the interface. Thus, immediately, one deduces that the additional cost introduced by the implicit approach introduced in Chapter 1, and mitigated by localization, might be directly reduced by the adaptivity properties of the grid, focusing most of the grid cells close to the zero levet set of ϕ, namely the interface Γ.

Also, as already stated, using a SL approach, we can satisfy our need of refining the grid in the proximity of Γ, without being prohibitively limited by the CFL restriction, which would be instead the case of Eulerian schemes, especially in this modelling case where, due to the curvature regularzation, a parabolic term appears in the governing PDE. On the other side, SL algorithms, compared to Eulerian ones, might not be easy to parallelize, especially on adaptive grids, since, depending of the CFL number, the feet of the characteristics may end up outside the halo region, in remote ranks, and may also significantly cluster, consequently affecting the load balancing.

In what follows we will present the formulation of the surface reconstruction

method of Chapter 4 based on AMR. We will start by introducing the main features of the adapted grid and its management mainly following [19, 45, 88], and then we will retrace all the ingredients needed to design the surface reconstruction algorithm in this octree-based setting. The main aspects related to AMR and parallelization will be emphasized along the way and some preliminary results will be collected at the end of the chapter. For this adaptive version, the codes have been implemented in C++ using the P4EST library [19] for grid management and MPI parallelization.

5.1 Grid management with P4EST

In the AMR context, the terms "quadtree" and "octree" denote a recursive tree structure where each node is either a leaf or has respectively four and eight children. Given a cubic domain Ω'_k, we can associated to Ω'_k a quadtree or an octree, depending on whether we are in two or three dimensions. Here, the notation Ω'_k is chosen in order to be consistent with the embedding of the region $\Omega \subset \Omega'$ of Chapter 1.

The nodes of a quadtree are called quadrants and the nodes of an octree are called octants, while the root node corresponds to a cubic domain that is recursively subdivided according to the tree structure. In this context, a "forest" is a collection of such logical cubes Ω'_k that are connected conformingly through faces, edges, or corners, each cube containing an independent tree.

In P4EST, to discretze a physical domain Ω' we consider multiple trees, the forest, each one of them covering a subset Ω'_k of the domain, fitting its geometry. The trees are based on reference cubes $[0, 2^L]^d$, where L is the maximum level of refinement and d is the space dimension, such that a one-to-one transformation $\Psi_k : [0, 2^L]^d \to \Omega'_k$ is defined. In practice, with this approach one is allowed to define also complex geometries involving a macro-mesh composed by trees, representing the entire domain, and the inner tree micro-meshes composed by the refined quadrants or octants. While the cells of the macro-mesh have to be conforming, this is not required for the cells of the micro-mesh. In the parallel perspective, we point out that each quadrant or octant representing a leaf belongs to precisely one process and is stored only there.

For simplicity we will cover here the description of the grid just in the two dimensional case, thus we will refer to the elements of the adaptive grid as quadtrees and quadrants. When needed, we will specify the differences related to the three dimensional case. Also, we will not cover the case of a forest composed by multiple trees, but we will limit ourselves to the case of a single cubic domain Ω' represented by a single tree. The interested reader is referred to [19, 45] for more details about the management of a non-trivial forest.

5.1.1 Meshing and data storage

One of the main challenges in managing and parallelizing algorithms on adaptive grids is handling the grid itself. Earliest libraries worked by providing a copy of the entire grid to each process and employing serial ordering techniques [9, 77], with consequent limits to the scalability of the algorithms. On the other side, the modern idea behind storing data is based on the concept of space-filling curves (SFCs) which

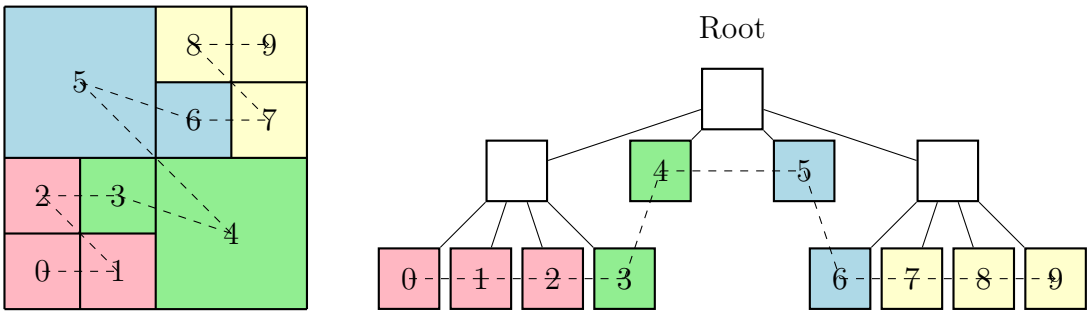

Figure 5.1: Representation of a refined 2d square domain. In the left panel the mesh and the z-order curve are depicted, with different colors representing different owning processes. In the right panel the corresponding subdivisions from the root are shown. The black dashed line represents the z-curve filling the domain.

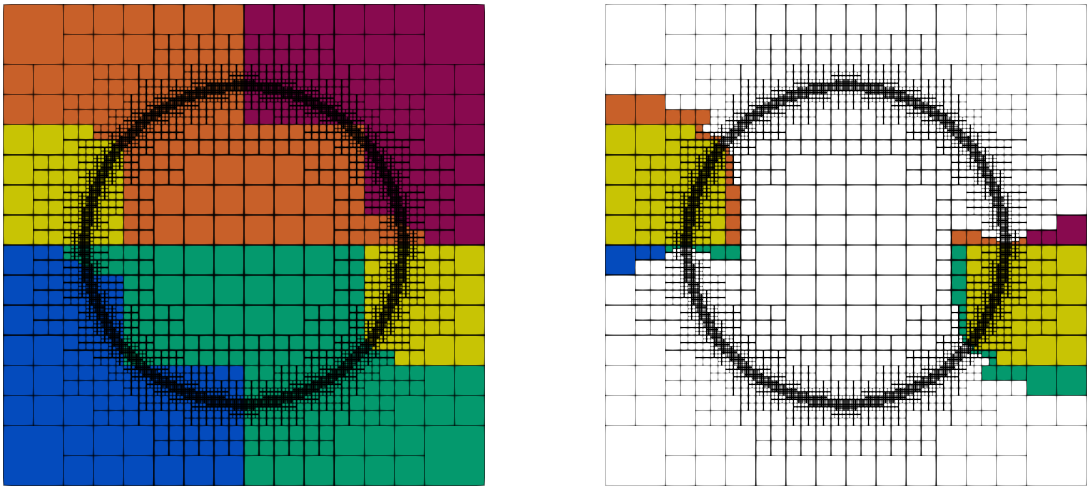

Figure 5.2: A Quadree refined around the zero level set of the signed distance function to a circle. Left: grid partitioning among five processes, each one represented by a different color. Right: detail of the portion of the domain owned by process number 2, together with its ghost layer. Even if the yellow area is not topologically connected, its components are contiguous in the z-order storage.

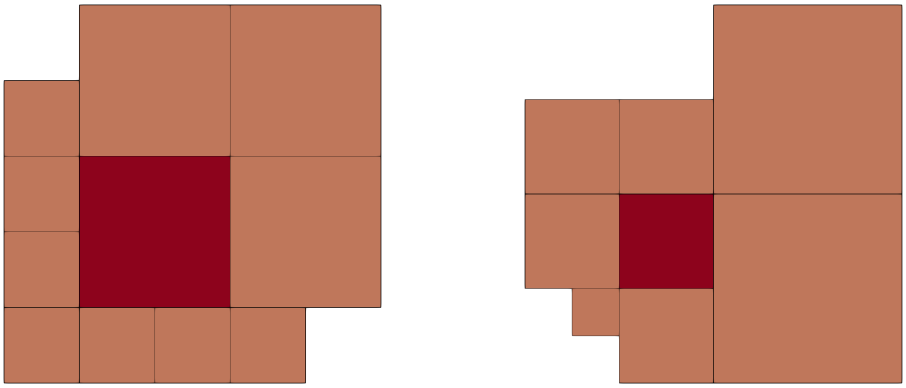

Figure 5.3: Two quadrants of the same tree with different neighbours. The main quadrants are depicted in red and may have themselves different size.

has been demostrated to be efficiently exploited for parallel load balancing, too [3, 61, 22]. Indeed, many SFCs have a nice property called compactness, which states that contiguity along the SFC index implies, to a certain extend, contiguity in the d-dimensional space of the Cartesian mesh. As a consequence, one can expect improved AMR performance due to a better cache memory usage resulting from a certain degree of preserved locality between the computational mesh and the data memory layout.

The P4EST library [19] is one recent example of a cell-based AMR implementation that uses linear storage given by a SFC. Each quadrant in the micro-mesh is then associated with its position in the reference cube $[0, 2^L]^2$ and is therefore uniquely tracked by its integer spatial coordinates, $(s, t) \in [\![0, 2^L]\!]^2$. Linear storage requires a one-to-one mapping from the spatial coordinates (s, t) to a linear index i, which in P4EST is provided by the Morton space filling curve, also called z-order curve. In Figure 5.1 one can see how the z-order curve, the black dashed line, covers a 2d mesh and how the leaves of the tree are distributed among four different processes.

5.1.2 Working with cell centers

As usual, when discretizing a domain, one can choose to work in a node-based framework, as in a finite differences context, or in a cell-center one, as it is done in finite volume schemes. It is crucial to point out that, compared to what we have done in the previous chapters, here we will involve a finite different approach based on cell centers, instead of vertexes. Thus, the centers of the quadrants will be the grid points of our spatial discretization, and our solution, namely the level set function ϕ, will be computed in these centers. In particular, when we use the notation ϕ_i^n we mean to denote the value of the function ϕ in the node i, corresponding to the center x_i of the quadrant i, at time n. This choice is mainly suggested because it leads to a simpler implementation, as data can be directly associated to the leaves of the tree in P4EST.

Moreover, for each quadrant i, we denote with Δx_i its edge length and, once the

maximum level of refinement L is fixed, we define as

$$\Delta x_{\min} = \frac{\Delta\Omega'}{2^L} \tag{5.1}$$

the minimum edge length of the quadrants composing the tree, where $\Delta\Omega'$ is the edge length of the cubic domain Ω'.

The most important consequence of working with quadrant centers is that, performing one of the interpolation techniques proposed in the previous sections among the nodal values, we cannot anymore guarantee the continuity of the reconstruction at the vertexes of the quadrants themselves, and even more so on their edges. This is true not only in the case of an adapted mesh, but also in the case of a uniform Cartesian one, which is for instance the situation that will locally occur around the zero level set of ϕ.

Moreover, we have to take into account the fact that, due to adaptivity, the number of neighbours of each quadrant is not fixed, as well as their dimensions. As a consequence, one should resort to proper strategies to compute, for instance, the derivatives in each grid point, namely the cell centers, and also, if one would like to perform an interpolation of ϕ of degree $r \geq 1$, the stencil of the interpolation operator cannot be fixed, too. Figure 5.3 shows an example of two possible situations: two different quadrants of the same grid may have a different number of neighbours, of different sizes, and these type of changes can be observed not only looking at different locations of the mesh, but also during the evolution of the mesh in a fixed location.

Putting together all these considerations, in what follows, we will resort to an interpolation between the values of ϕ following a least-squares approach. The interpolation will be necessary for the SL discretization, and also exploited for the computation of the derivatives and for the adaptivity procedure.

5.1.3 Adaptivity

P4EST creates the macro-mesh only once, initially, and then adapt it by modifying the micro-mesh. The steps below describe the typical adaptivity procedure.

- First of all, going through the linear array, the leaves are marked for refinement, coarsening or to be left unchanged, depending on the criteria given by the user.

- The refinement and coarsening is then applied to each marked leaf, if possible, and recursively, if required. With "possible", we mean that the refinement and coarsening procedures need to be consistent with the minimum and maximum level of refinement set by the user. Moreover, a very important feature of the adaptive algorithms is that the z-order is maintained while modifying the linear array.

- Once the grid has been roughly adapted, in order to guarantee that the level difference between a quadrant and each of its neighbours is at most 1, 2:1 balancing is performed. Trees with such a property are also denoted as "graded trees". 2:1 balancing comes of course at a price: since the grid is forced to

degrade progressively from finer regions to coarser ones, there will be a natural increase in the number of cells compared to non-graded trees. Nonetheless, this procedure will ensure an easier to handle structure of the mesh, due to the knowledge of the maximum number of neighbours per face.

- Finally, as far as parallelization is concerned, load distribution is operated between processes by an equal division of the new array of leaves.

An example of a refined grid is shown in Fig 5.2, where we have uses the signed distance function ϕ to a circle of radius r in order to refine a squared grid around the circle until the maximum allowed level of refinement $L = 8$. More specifically, a quadrant i of center x_i is refined if the inequality

$$||x_i| - r| \leq \sqrt{2}\Delta x_i \tag{5.2}$$

is satisfied, where Δx_i is the edge length of the quadrant i; otherwise, the grid is coarsened.

In the example of Figure 5.2 the signed distance function to the circle ϕ is analitically prescribed, thus refining and coarsening are easily performed by evaluating ϕ in the new quadrant centers. In a more general situation in which one only knows the point values of ϕ in the nodes x_i before the adaptivity step, an interpolator operator could be used, with proper strategies to handle the procedure. The specific example involving the `Q1` interpolator will be described in the next sections.

5.2 Application to surface reconstruction

As already stated in the introductory part, the aim of this chapter is to present the extension of the method for surface reconstruction from point clouds to the AMR framework. The derivation of the model is thus exactly the same as presented in Chapter 4 and also the SL scheme derived for the numerical approximation of the solution remains the same. Nonetheless, some other ingredients presented before need to be revised due to the unstructured grid and its partitioning scheme.

According to the change in the spatial discretization, also the notation should be properly updated. In what follows we will refer to the whole computational grid as $\mathcal{T}$, in order to refer to the tree structure, so that

$$\mathcal{T} = \{x_i : x_i \in \Omega', i \in [1, \dots, Q]\}, \tag{5.3}$$

where Ω' is the usual large enough computational domain in which we are embedding the evolving region $\Omega(t)$, the points x_i correspond to the centers of the quadrants of the tree, and Q is the total number of quadrants in the tree. We point out that, since for the sake of clarity we are not employing a multiple tree structure, the domain Ω' is required to be squared, cubic in three dimensions, in order to be represented by just one tree. Of course the general case of a forest can be treated by applying the techniques described here to each tree composing it.

5.2.1 SL scheme

For the sake of clarity, it is worth recalling the SL scheme governing the numerical evolution of the level set function ϕ, at least in the two dimensional case. Given an initial level set function, from which we get initial data $\{\phi_i^0\}_{x_i \in \mathcal{T}}$, we compute the update of ϕ_i^{n+1} as

$$\begin{cases} \phi_i^{n+1} \;=\; \frac{1}{2} \sum_{j=1}^{2} I[\Phi^n]\left(x_{i,j}^*\right), \\ x_{i,j}^* = x_i + C_i^n \Delta t \nabla d(x_i) + \sqrt{\frac{2C_i^n \, \eta \, d(x_i) \Delta t}{p}} \, \sigma_i^n \xi_j, \end{cases} \tag{5.4}$$

where η, ξ_i, σ_i^n are the same as in Section 4.3 and C_i^n is the scale factor that, according to (4.31), (4.32), is given by $c(\phi_i^n)\left[\frac{d(x_i)}{E_p(\phi^n)}\right]^{p-1}$, with the cut-off function c defined by (1.55).

Regarding specifically the scheme (5.4), the differences from the uniform Cartesian case are related to the localization of the feet $x_{i,j}^*$, the interpolation operator $I[\Phi^n]$ and the computation of the derivatives that, as already specified, are obtained from the interpolant defined for the quadrant i.

Localizing the feet While on Cartesian grids localizing the points $x_{i,j}^*$ is a trivial task, on Quadree and, moreover, involving parallel implementation, this is no longer an easy procedure due to the irregular shapes of the partitions and of the grid itself. At best we can only expect that the locations of the feet are bounded by a halo region of width proportional to $\Delta x_{\min}$, where the constant of proportionality depends on the value of the CFL number employed to define the time step. One remedy to avoid communications among processes could be then to increase the size of the ghost layer, but this approach clearly would limit our choice on the time step. Thus, similarly to what have been done in the previous chapter, also in this case we will handle local and remote points separately, localizing them in the partition and then making the processes communicating with each other, when needed.

The localization of a point x^* on Quadree can be based on the same principle used on a Cartesian grid. Once the level of refinement l, $1 \leq l \leq L$, is fixed, the quadrant of level l containing x^* can be easily detected as on a Cartesian grid of width $\Delta x_l = \Delta\Omega'/2^l$, by simply checking the relative location of x^* in the cube Ω'. Thus, starting from the root level, the idea is to progressively increase the searching level l and locate x^* in a Cartesian-like fashion until the quadrant detected corresponds to a leaf, namely having no children.

Actually, if one just need to find the owner process of x^*, the procedure above can stop before reaching a leaf, simply checking whether the quadrant of level l, containing x^*, is or is not split between multiple processors.

In our implementaton, this procedure is realized by making use of the P4EST search functions to detect the process that owns the region in which each foot of the characteristic falls and, locally, the specific quadrant in which to interpolate. The functions *p4est_search_partition* and *p4est_search_local* are apt to this aim: given an array of points, they proceed iteratively top-down respectively through the global partition and all the local quadrants in order to locate each one of the points in the array.

Once the feet are locally located, we can proceed by computing the values $I[\Phi^n](x^*_{i,j})$, sending back the interpolated values when needed and finally updating ϕ_i^{n+1}.

Interpolation Let us consider a space of polynomials $\mathbb{P}$ with a basis $\mathcal{B}$. The key point regarding interpolation on Quadtree is that, since each quadrant i may have a different number of neighbours $|\mathcal{N}_i|$, the system to solve to get the coefficients of the interpolant on the quadrant i may have a number of rows lower, equal or greater than $|\mathcal{B}|$, clearly taking into account also the central node x_i.

In our preliminary results we involved the multilinear interpolator, whose basis functions in multi-d are obtained by tensorization of the 1d basis functions defined for each Cartesian direction. Thus, in 2d we use the local basis $\mathcal{B}_i = \{\varphi_{i_k}\}_{k=1,\dots,4} = \{1, x'_1, x'_2, x'_1x'_2\}$ for the quadrant i, where $x' = (x - x_i)/\Delta x_i$, and we look for the vector of coefficients $\{c_{i_k}\}_{k=1,\dots,4}$ by imposing the interpolation condition

$$\sum_{k=1}^{4} c_{i_k}\varphi_{i_k}(x_j) = \phi_j^n \tag{5.5}$$

for each quadrant j in the set of neighbours $\mathcal{N}_i$ of the quadrant i, and for the quadrant i itself.

With this choice one can immediately note that, considering either edge and corner neighbours (see Figure 5.3), the case $|\mathcal{N}_i|+1 < |\mathcal{B}|$ never occurs, but, unless rare situations, we will be in the case in which the number of rows of the Vandermonde matrix is greater than its number of columns, causing the system for the c_{i_k} to be overdetermined.

Therefore, we use a least-squares approach to compute the coefficients c_{i_k} of the interpolant imposing a constraint on the node x_i, such that the interpolation condition always holds in the center of the quadrant i. As a consequence, we will always have $c_{i_0} = \phi_i^n$ and the least-squares approach, for the `Q1` case, is just applied to the reduced system with three unknowns.

5.2.2 Refinement and coarsening

In this subsection we detail the criteria used for adaptivity in the specific framework of surface reconstruction application, describing also the procedure used to initialize the new quadrants. Initializing the new incoming quadrants is of course needed in order to continue the evolution of the level set function ϕ. To this end, a reconstruction of the data at time t_n is computed. This will also be useful in the coarsening criterion.

Criteria Along the lines of [94], we have described in Chapter 1 the common procedure used to localize the LSM in a proper tube around the zero level set of ϕ. In order to follow this approach we need solid criteria for adaptivity that ensure to have a properly refined grid around Γ and a progressively decreasing resolution moving further away, in particular where our level set function ϕ is flattened to a prescribed value $\pm\gamma$, according to the sign of ϕ.

Thus, following the notation used in Section 1.6 and later on in Section 4.4.6, we proceed as follows:

- a quadrant i is refined if it is close enough to the zero level set of ϕ, i.e. if $|\phi_i^n| < \beta$;
- a quadrant i is coarsened if the reconstruction is flat, i.e. if $|\nabla\phi_i^n| \approx 0$.

In this adaptive version, the parameters β and γ, and more in general all the parameters proportional to the resolution of the grid, should be defined considering as Δx the minimum edge length in the grid $\Delta x_{\min}$, as defined by equation (5.1).

Refinement Since on each quadrant i we have defined a local polynomial p_i, the procedure for the refinement is trivial and is achieved as follows.

The quadrant i of center x_i originates 4 children in 2d, 8 in 3d, by dividing by two each of its edges. The new quadrants i_k have centers in x_{i_k} positioned at the vertexes of a square, or cube, of edge length $\Delta x_i/2$ and centered in x_i. The value $\phi_{i_k}^n$ is simply obtained by interpolation using the polynomial p_i associated to the parent cell i and this polynomial is inherited by the children. The inherited polynomial will be used when recursive refinement is allowed and in cases when 2:1 grid balancing will cause further refinements.

Coarsening The coarsening procedure consists in gluing together quadrants of the same level. To this aim, let us consider 4 outgoing quadrants in 2d (8 in 3d) of centers x_k, $k = 1, \ldots, 4$ ($k = 1, \ldots, 8$ in 3d), and one incoming quadrant of center x_i. For coarsening, either the center and the value of ϕ associated to the parent cell i are obtained performing an average of the centers x_k and of the values ϕ_k of the outgoing quadrants k.

In order to have a rough reconstruction on the incoming quadrant i, available for possible recursion, also an average between the coefficient of the polynomials defined of the quadrants k is performed. This coarse polynomial will be used to test again the condition for further coarsening.

Note that, due to balancing, a quadrant that has just been obtained by coarsening might be refined, hence the utility of having the rough reconstruction available.

Preparing the new time step Once the grid at time t_n is adapted and the new values are computed, a new reconstruction at time t_n, as described in Subsection 5.2.1, is computed, and this will be used to compute the values of the function ϕ at time t_{n+1}. Denoting with $\mathcal{T}^n$ and $\mathcal{T}^{n+1}$ the computational grids at time t_n and t_{n+1}, respectively, we will follow the steps

$$\{\phi_i^n\}_{x_i \in \mathcal{T}^n} \;\rightarrow\; \{\phi_i^n\}_{x_i \in \mathcal{T}^{n+1}} \;\rightarrow\; \{\phi_i^{n+1}\}_{x_i \in \mathcal{T}^{n+1}}. \tag{5.6}$$

We point out that, since in our implementation the adaptivity process is recursive, in addition to the values of ϕ, we might also need a temporary reconstruction defined on the incoming quadrants. As already stated, this rough reconstruction will be used in the coarsening step and could be completely avoided by not allowing recursion in the adaptivity process.

5.2.3 Distance function

As usual, to compute our reconstruction we need to define the velocity field given by the distance function d from the point cloud $\mathcal{P}$. Similarly to what we have seen in the previous chapter, one is not interested in computing the exact values of d on the whole grid, but just in the vicinity of the cloud.

Thus, the first step for the distance computation retraces exactly the case of the uniform Cartesian grid: each point in $\mathcal{P}$ is located in a quadrant i such that d_i^n is initialized as the minimum distance of the center x_i from the cloud points located in the quadrant i.

Now, an issue arises since the most popular algorithms for solving the Eikonal equation, i.e. the Fast Marching Method (FMM) or the Fast Sweeping Method (FSM), might suffers either from the adaptive discretization of the grid and the parallelization of the scheme, due to the causality across processes. Some of the earliest attempts in parallelizing the FMM was proposed in [69, 115], where one could notice how the number of iterations needed to get convergence greatly depends on the complexity of the interface and on the parallel partitioning and, in general, fewer iterations are required if the domains are aligned with the normals to the interface. In [120] a parallel FSM method was presented for the first time, while a hybrid FMM-FSM was presented in [33]. Finally, a parallel Fast Iterative Method (FIM) was proposed in [74].

In this chapter, we are still not focused on the parallel implementation of these family of methods to approximate the distance function in the whole domain. Instead, after doing the first initialization step, we propagate the information to the neighbours of each initialized quadrant, computing the exact values of the distance from the points in $\mathcal{P}$.

The details of the propagation procedure will be described below in Subsection 5.2.5. Here we just point out that, even if a quadrant has been initialized in the first step, it will be still considered in the propagation step since its value might be updated due to the information carried over from a neighbour.

5.2.4 Reinitialization

As pointed out in Chapter 1, during the evolution we need to keep the level set function ϕ well-behaved in the sense that we would like to approximately keep the signed distance property $|\nabla\phi| = 1$. In [88] the authors resort to the popular approach of solving the reinitialization equation

$$\phi_\tau + S(\tilde{\phi})(|\nabla\phi| - 1) = 0 \tag{5.7}$$

using explicit finite differences following the scheme described in [87], where a second-order accurate LSM on non-graded adaptive Cartesian grids is described.

Here instead we refer to the work of Saye [101] in which the author proposes an efficient method for calculating high-order approximations of closest points on implicit surfaces that can be applied also on a general unstructured grid and in any number of spatial dimensions.

Actually we will apply the procedure introduced in [101] just in the vicinity of the zero level set of ϕ and then, once we have reinitialized ϕ in these quadrants, we

propagate the information in the entire $\mathcal{T}$, or at least in a proper tube around the zero set of ϕ (see the next subsection).

Let us consider a fixed time n of the evolution of the level set ϕ for which we have obtained the updated level set function $\tilde{\phi}$ that need to be reinitialized.

Along the lines of [101], we start by detecting the quadrants that contain the zero level set of $\tilde{\phi}$, namely the set $\mathcal{T}_0 \subset \mathcal{T}$. In practice for each quadrant i, we say that $x_i \in \mathcal{T}_0$ if there exists a quadrant j in its set of neighbours $\mathcal{N}_i$ such that $\tilde{\phi}_i \tilde{\phi}_j \leq 0$.

Once the subset $\mathcal{T}_0$ is found, for each quadrant i such that $x_i \in \mathcal{T}_0$, a high-order closest point algorithm via Newton's method is applied as follows:

- for each quadrant i we create 4 points x_{i_k} in 2d (resp. 8 points in 3d) located at the centers of a subgrid 2×2 (resp $2 \times 2 \times 2$) of the quadrant i;
- each point x_{i_k} is thus projected onto the zero level set of the polynomial p_i given by the reconstruction defined on the quadrant i;
- considering the set X of all the points $\{x_{i_k}\}_{x_i \in \mathcal{T}_0, k=1,\dots,4}$, for each $x_i \in \mathcal{T}_0$ we find its closest point $x_i^* \in X$;
- the point $x_i^* \in X$ is then used as the initial guess to compute the minimum distance of the node x_i from the zero level set of $\tilde{\phi}$ via a Newton's method that relies on the reconstructed polynomial associated to the point x_i^*;
- finally, for each x_i, the reinitialized value ϕ_i is set equal to this minimum distance, multiplied by the sign of $\tilde{\phi}_i$, namely preserving the sign of the level set function before the reinitialization.

In practice, the method described in [101] is based on a two steps procedure to compute the closest point to the interface $\tilde{\phi} = 0$ for a point x_i located in its vicinity, relying on the reconstructions p_i defined on the quadrants, thus actually considering the zero level set of the polynomials p_i. The points $x_i^* \in X$ are referred to as *seed points* and one should note that the closest point in X to a quadrant center x_i might not have been originally generated by the same quadrant i.

We point out that, in opposition to the propagation step for the distance function, here, once the level set function ϕ has been reinitialized in a quadrant with the method described above, we fix this value such that it cannot be updated during the propagation step.

5.2.5 Propagation through the domain

Both for the distance computation and for the reinitialization procedure we need to propagate some initialized value to the entire grid $\mathcal{T}$, or at least part of it, having in mind to solve an Eikonal equation for the distance function or the signed distance one.

The problem can be stated as follows: let us consider a subset $\mathcal{D}^0 \subset \mathcal{T}$, on whose points a certain quantity q_k^0 is defined. In our specific case q_k^0 is the distance (resp. signed distance) of the quadrant center x_k from a point $\hat{x}$, which is a member of $\mathcal{P}$ (resp. a reinitialization seed). We also store this reference point as $\hat{x}_k$ attached

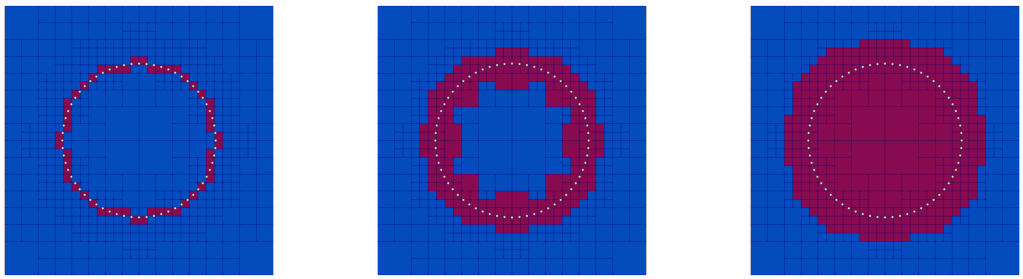

Figure 5.4: Propagation of the distance function. Left: the initialized quadrants before the propagation step are the one containing one or some of the points in $\mathcal{P}$. Center: quadrants with updated distance after one step of propagation. Right: quadrants with updated distance after two steps of propagation.

to the quadrant. At the start of the propagation, data of quadrants in $\mathcal{T} \setminus \mathcal{D}^0$ are initialized to a high enough value. Then, the iteration procedure for the propagation process is performed as follows.

During the iteration $m+1$, for each $x_k \in \mathcal{D}^m$ we consider its set of neighbours $\mathcal{N}_k^m$ and we define the set $\mathcal{N}^m := \bigcup_k \mathcal{N}_k^m$. Note that in general $\mathcal{N}^m \cap \mathcal{D}^m \neq \emptyset$, as we may need to update the values already computed in the previous step.

Now, for a point $x_j \in \mathcal{N}^m$ we compute $\tilde{q}_j^{m+1}$ as

$$\tilde{q}_j^{m+1} = \min_{x_k \in \mathcal{D}^m} |x_j - \hat{x}_k| \,, \tag{5.8}$$

where in each computation of the distance we are considering the point $\hat{x}_k$ associated to x_k.

The iteration process consists in testing if $\tilde{q}_j^{m+1} < q_j^m$ in order to verify if some information has propagated to the point x_j and the quantity q_j decreased. In this way we can define

$$\mathcal{D}^{m+1} = \{x_j \in \mathcal{N}^m : \tilde{q}_j^{m+1} < q_j^m\}. \tag{5.9}$$

To complete the iteration step we set $q_j^{m+1} = \tilde{q}_j^{m+1}$, for each $x_j \in \mathcal{D}^{m+1}$, while $q_j^{m+1} = q_j^m$, elsewhere. The propagation process is iterated until the set $\mathcal{D}$ is empty, namely when the minimum has been reached in all the quadrant centers of $\mathcal{T}$.

The procedure described above works exactly as it has been described, for the propagation of the distance function from the point cloud $\mathcal{P}$. The quadrants containing the cloud points are initialized and then their information is propagated, including the already initialized quadrants in the propagation procedure. In Figure 5.4 three different phases of the propagation are shown ($m = 0, 1, 2$). Red quadrants are the ones that has been initialized with the distance to a point $\hat{x}$, if $m = 0$, or updated lowering their value of the distance to $\mathcal{P}$, for $m = 1, 2$; the blue ones are the ones in which the information has not arrived yet.

Regarding the reinitialization, the procedure is done analogously, but in the definition of $\mathcal{N}^m$, $m \geq 0$, we exclude the quadrants in the set $\mathcal{D}^0$ since we don't want their value of the reinitialized ϕ to be changed by the propagation process. Also, in the update of q_j^{m+1} we should take into account the sign of q_j^0, such that $q_j^{m+1} = S(q_j^0)\tilde{q}_j^{m+1}$, where S is the sign function.

5.3 Numerical test

In this section we present some numerical results obtained with this new adaptive version of the algorithm, as described in the previous sections. The stopping criterion and the considerations for the choice of the parameters p and η are exactly the same as the ones described in Section 4.5.

Also, retracing the approach used in the uniform Cartesian version of the scheme, the adaptive algorithm can be iterated more than one time, increasing of one unit the maximum refinement level allowed and accordingly changing the parameters p and η as described in Table 4.1. The switch from one run to the finer next is performed automatically when the stopping criterion (4.33) is satisfied, just letting P4EST to refine one level more. No reloading nor interpolation of the steady-state of the previous run is needed. The maximum number of internal iteration for each run is again set to 100. More precisely, if we want to perform R total runs of the algorithm, this means that we perform a first run with a maximum refinement level $L = L_0$ and end up with a final maximum level given by $L = L_0 + R - 1$.

In order to take into account the resolution $h_{\mathcal{P}}$ of the point cloud from which we want to reconstruct the surface, the initial value of L is set as

$$L_0 = \left\lceil \log_2 \left(\frac{\Delta \mathcal{P}}{\Delta x^*_{\min}} \right) \right\rceil, \tag{5.10}$$

where $\Delta\mathcal{P}$ is the edge length of the minimal cube, aligned with the Cartesian axes, containing the point cloud $\mathcal{P}$, and $\Delta x^*_{\min}$ is approximately equal to the minimum grid size we would like to have in our discretization, namely $\Delta x_{\min}$. In the first run, according to (4.36), we set

$$\Delta x^*_{\min} = C_{\Delta x} h_{\mathcal{P}}, \tag{5.11}$$

with $C_{\Delta x}$ usually equal to 0.5.

Regarding the choice of the time step, we consider a CFL number equal to 1 with respect to the minimum length of the edge quadrant in the mesh, thus setting

$$\Delta t^{(r)} = \Delta x^{(r)}_{\min}, \tag{5.12}$$

where the superscript (r) indicates the corresponding run of the algorithm. We point out that this choice is coherent with the refinement criterion; for larger CFL numbers we would need more refined cells or a different strategy to guarantee the accuracy of the scheme. Note that the diffusion term would require $\Delta t = \mathcal{O}(\Delta x^2_{\min})$, while in (5.12), similarly to the relationship employed in Section 4.5, we set $\Delta t = \mathcal{O}(\Delta x_{\min})$.

In this first version of the adaptive reconstruction method the initial data is simply the signed distance function associated to a circle, or a sphere, encompassing the whole point cloud or, at least, most of the points.

5.3.1 2d data sets

Circle

The first benchmark test we propose is the reconstruction of a circle from a point cloud constituted by the same 64 points used in the test of Subsection 4.5.3. The

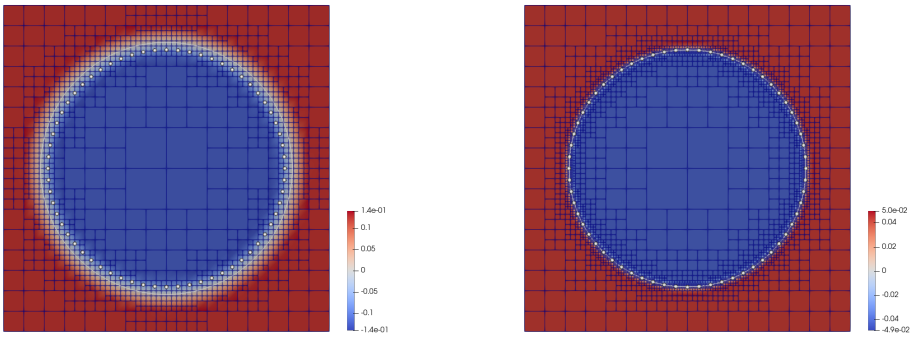

Figure 5.5: Left: the initial data for the circle reconstruction test, with the lower refined initial grid. Right: the final level set function on the finer grid.

initial signed distance function and the final one are depicted in Figure 5.5. The final reconstruction is obtained after 3 runs of the algorithm, progressively increasing the resolution of the grid, that took respectively 26, 12 and 15 internal iterations. We point out that the stopping criterion, as it is designed so far, forces to do at least 10 iterations.

The profiles of the energy functional and of the error on the cloud are shown in Figure 5.6 and can be compared to the ones shown in Figure 4.7 for the same test on the uniform Cartesian grid. In this adaptive case, at the end of the third run, the values of $E_2(\phi)$ and $Err_{\mathcal{P}}$, namely the error on the cloud (4.35), are $1.17e-01$ and $1.20e-03$, respectively. These values can be compared to the ones reported in Table 4.2 for the uniform Cartesian case.

Finally, note from Figure 5.7 the differences in the number of quadrants composing the adaptive grids with respect to the one that would have been required in order to perform the same computations on a uniform Cartesian mesh refined at the same maximum level L.

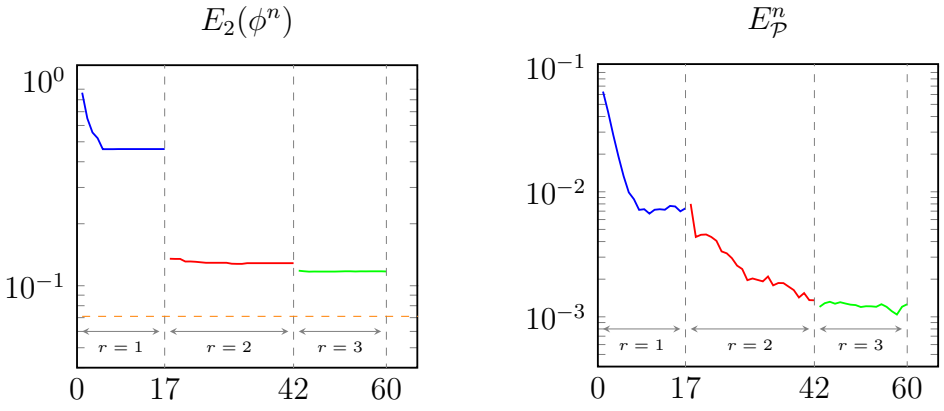

Figure 5.6: Energy and error on cloud computed for the circle reconstruction test with the adaptive scheme.

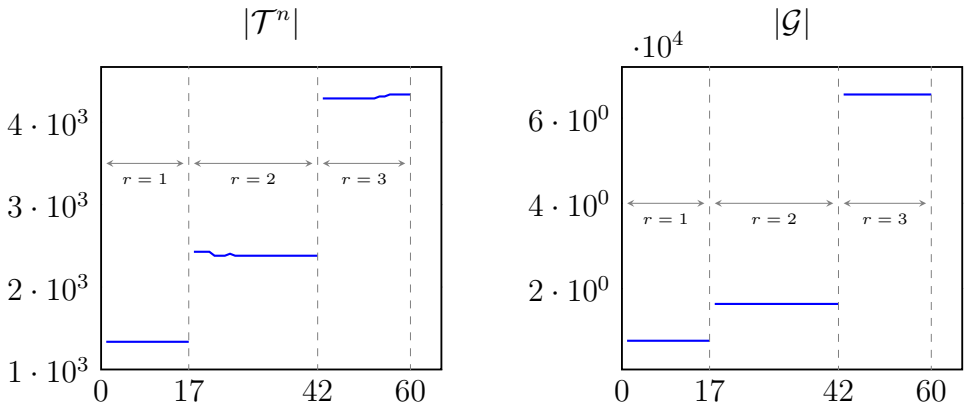

Figure 5.7: Circle test: comparison between the number of quadrants in the adaptive grid $\mathcal{T}^n$ at each iteration (left panel), and the number of quadrants required to perform the computations on a uniform Cartesian grid $\mathcal{G}$ refined at the maximum level L.

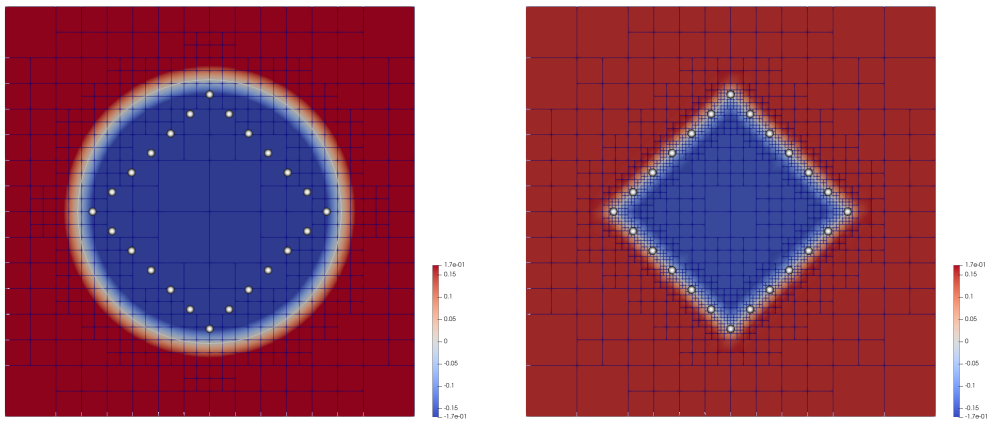

Figure 5.8: Left: the initial data for the square reconstruction test, with the lower refined initial grid. Right: the final level set function on the finer grid.

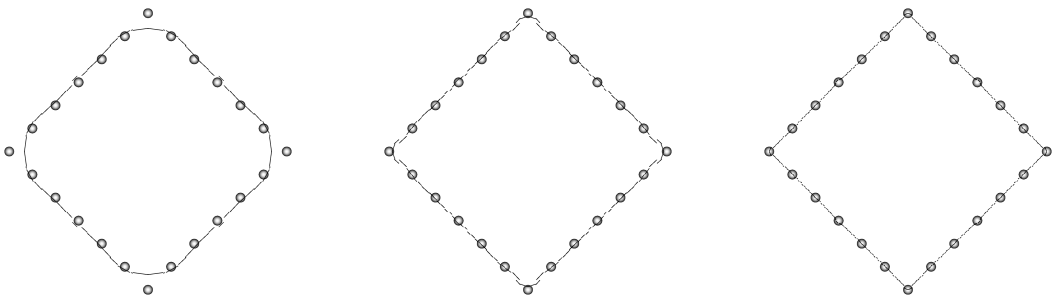

Figure 5.9: From left to right: a comparison between the zero level sets obtained for the square reconstruction test at each run.

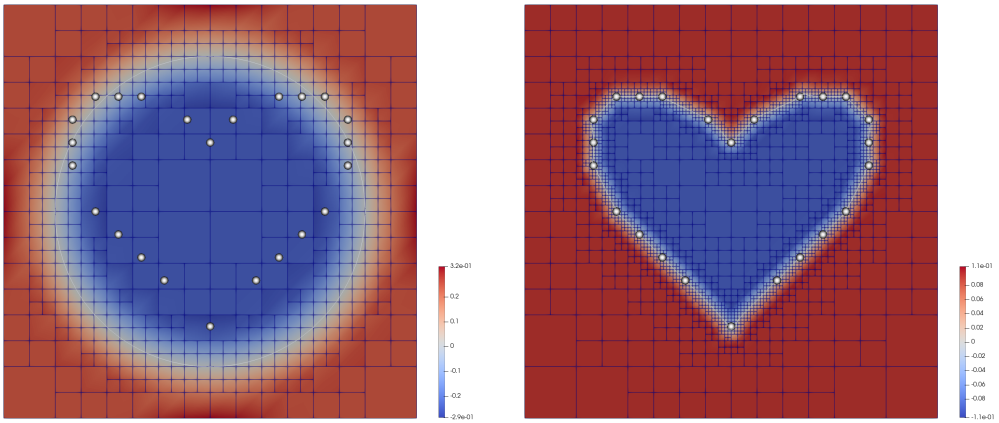

Figure 5.10: Left: the initial data for the heart reconstruction test, with the lower refined initial grid. Right: the final level set function on the finer grid.

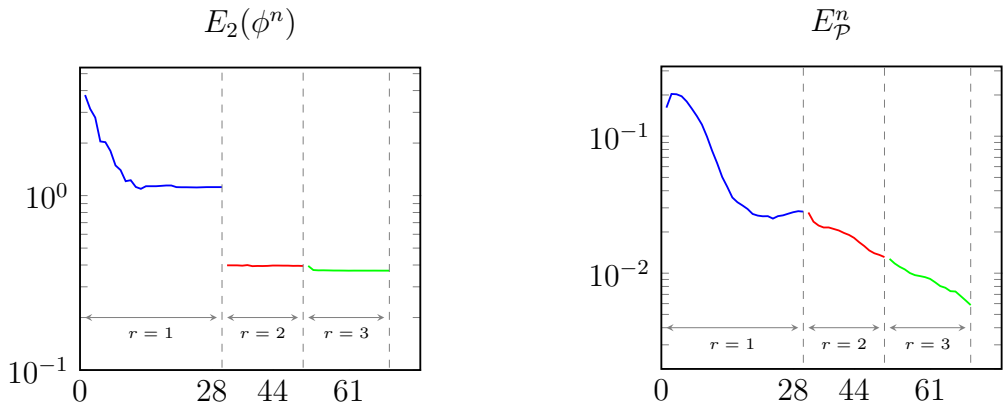

Figure 5.11: Energy and error on cloud computed for the heart reconstruction test with the adaptive scheme.

Square

The second two dimensional test we propose is again the reconstruction of a square sampled by 24 points, as in Subsection 4.5.3. The initial data and the final reconstruction are shown in Figure 5.8. The reconstruction procedure required 27, 16 and 21 internal iterations for each run, at the end of which the zero level sets are the ones depicted in Figure 5.9, which can be compared to the ones shown in Figure 4.9. Note that the zero level set appears noticeable disconnected due to the lack of continuity among quadrants.

Heart

We conclude the 2d tests session by considering a heart-shaped point cloud composed by 24 points, as in Subsection 4.5.3. This test shows how the method is able to recover either rounded, straight and sharp parts of the curve. The initial data and the final reconstruction are shown in Figure 5.10 and one could also notice

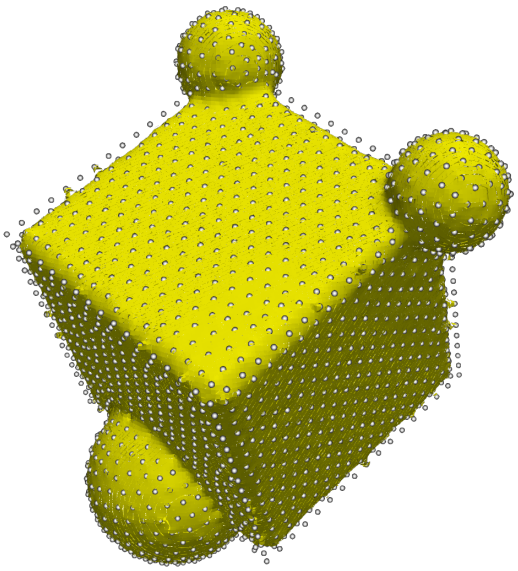

Figure 5.12: Left: the initial data for the heart reconstruction test, with the lower refined initial grid. Right: the final level set function on the finer grid.

that the initial zero level set is partially located inside the cloud, but, unless the curvature of the shape to recover is too high, this does not constitute a problem in the reconstruction procedure since the level set function is either able to shrink and to expand.

Graphs of the energy functional and of the error on the cloud are also shown in this case in Figure 5.11, where one could also notice that the entire run of the algorithm took 28, 16 and 17 internal iterations.

5.3.2 3d data sets

Cube&Spheres

We start the 3d numerical tests with the "Cube&Spheres" point cloud of Subsection 4.5.4 since it allows us to show how the level set function ϕ manages different features of a shape also in the adaptive case. The reconstruction is performed in 3 runs that took respectively 45, 33 and 23 internal iterations. In Figure 5.12 one can appreciate the final reconstruction obtained with our scheme.

In this test we also emphasize the partitioning of the three dimensional grid and how this partitioning evolves accordingly to the zero level set of ϕ, hence the refinement of the grid. In fact, in Figure 5.13 one can appreciate how the quadrants are distributed among difference processors, at initial and final time, each one depicted with a different color: adapting the grid following the evolution of the level set induces a repartitioning of the tree among the ranks.

Finally, as already done for the circle, we compare in Figure 5.14 the number of octants in the adaptive grid compared to the ones required in a uniform Cartesian setting refined at the same maximum level.

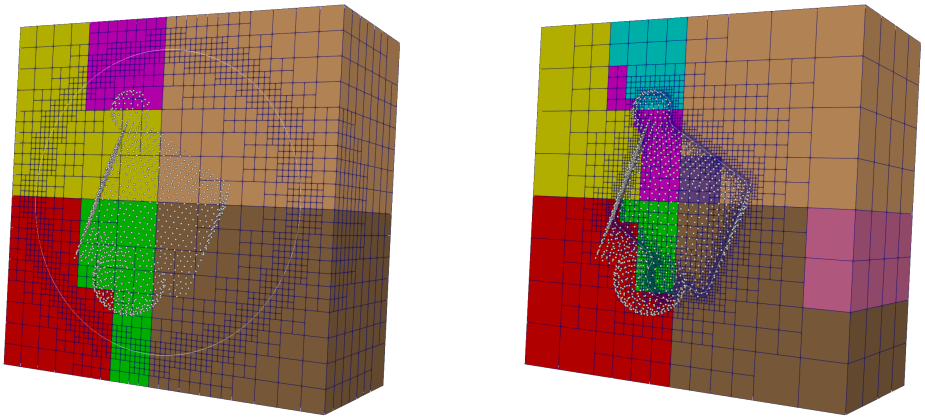

Figure 5.13: Grid partitioning for the "Cube&Spheres" reconstruction test. The initial and the final data are depicted in the left and in the right panel, respectively, together with the point cloud and the portions of the grid owned by different ranks are depicted with different corresponding colors. The zero level set of ϕ is also traced with a with line.

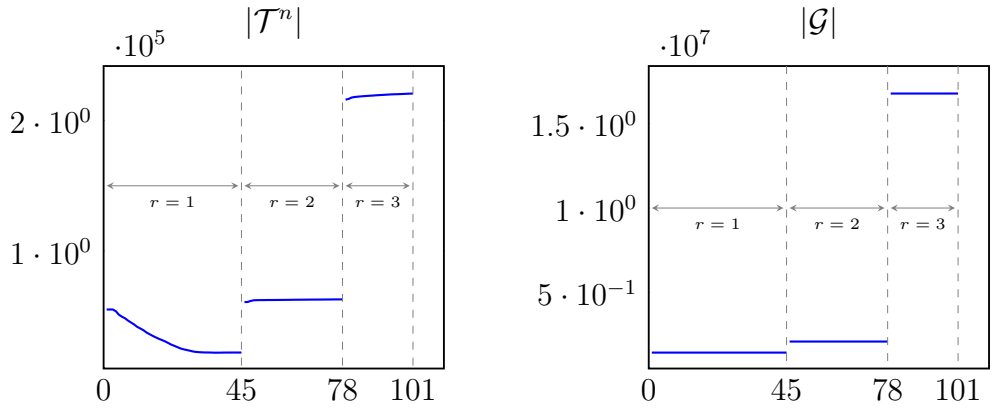

Figure 5.14: Circle test: comparison between the number of quadrants in the adaptive grid $\mathcal{T}^n$ at each iteration (left panel), and the number of quadrants required to perform the computations on a uniform Cartesian grid $\mathcal{G}$ refined at the maximum level L. Note that the significant decrease in the first run for the adaptive case is due to the shrinking of the zero surface.

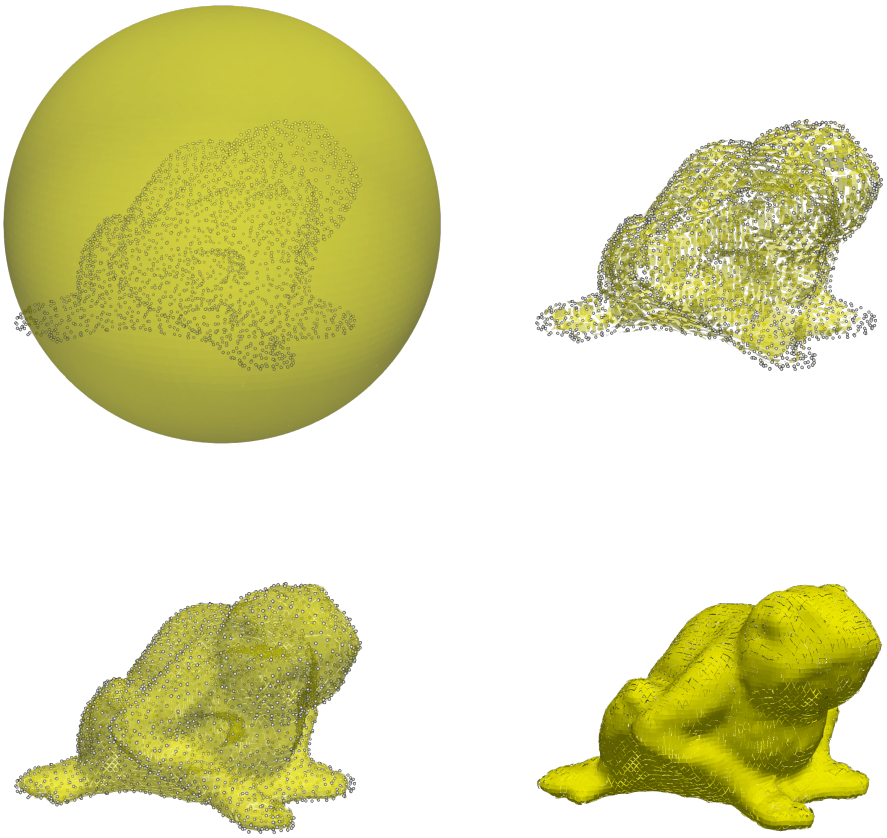

Figure 5.15: Reconstruction process for the "Frog" point cloud. Top left: the initial spherical data. Top right: the surface at the end of the first run. Bottom left: the final surface at the end of the second run. Bottom right: the final surface without the point cloud.

Frog

We conclude this chapter by considering as last numerical test the "Frog" reconstruction test of Subsection 4.5.5, in order to consider a data obtained via 3d laser scanning of a real object [2]. Only two runs are performed in this case and final reconstructions of each run are shown in Figure 5.15.

Also in this case, even if the initial sphere is not entirely encompassing all the points in $\mathcal{P}$, the zero level set does shrink or expand where appropriate, and all the details, including the right hind leg, are well recovered.

Finally, in Figure 5.16, an initial and a final slice of the computational grid are shown. The refined region follows the zero level set depicted with a white line and actives and non-active quadrants are indicated by different colors: red quadrants are the ones immediately closed to the interface in which the reinitialization procedured as described in Section 5.2.4 is performed; red and gray quadrants are the ones on which the SL scheme is applied; blue quadrants correspond to the inactive ones.

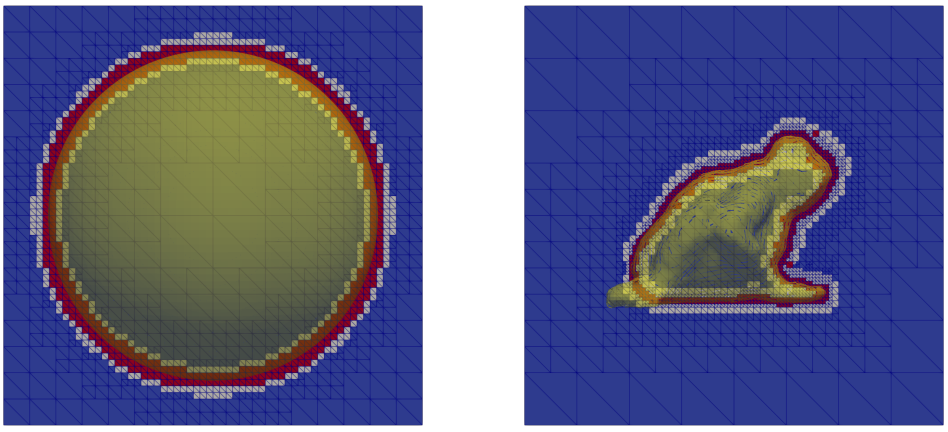

Figure 5.16: Slices of the initial and final data of the "Frog" reconstruction test. The slices emphasize the narrow bands used in the computation, finely adapted around the zero level set of ϕ. The red quadrants are the ones considering in the first step of the reinitialization, thus are the ones containing the front, or at least posed in its vicinity. Grey quadrants are active in the SL update. Blue quadrants remain inactive.

Conclusions and perspectives

In this thesis we have presented SL schemes coupled with Essentially Non-Oscillatory interpolation techniques in order to approximate the solution of first-order HJB equations and to desing a complete workflow for the reconstruction of surfaces starting from unorganized point cloud data. The schemes presented are aimed to high-order accuracy and are well-suited to be adapted to local mesh refinement. In particular, the potential of the LSM and of the SL approach employed in the surface reconstruction method, have been exploited in an AMR framework based on octrees.

Regarding HJB equations, the non-smooth nature of the solutions, which would lead to spurious oscillations when using unlimited high-order polynomial interpolation, is addressed by the use of a CWENO technique for the spatial reconstructions in the SL scheme. We also showed that this choice is apt to deal with the minimization procedure arising when applying the Dynamic Programming Principle to compute the solution.

Our study demonstrates that, in terms of errors, our new scheme maintains the favorable behavior of WENO schemes of [27], producing about 30% more accurate results in smooth regions at the price of some small extra over/undershoots. However, its computational cost is significantly lower, by a $10-30\%$ depending on the specific test. Additionally, we have established a convergence result in a simpler case, and provided several numerical simulations to further validate the effectiveness of our proposed scheme.

As a future perspective in this field of research, we wish to explore the inclusion of high-order treatment for boundary conditions, while also extending the convergence analysis to more general Hamiltonians. Investigating the scheme's performance in more complex scenarios would also contribute to a comprehensive understanding of its capabilities.

On the other hand, the SL approach has been also employed in the level set framework to approximate the solution of a transport-diffusion level set equation that moves an initial front towards the data in the point cloud. Resorting to the LSM, one evolves a level set function ϕ defined on a fixed domain, rather than the interface itself, thus capturing the moving interface implicitly as the zero level set of ϕ. The curvature regularization term in the governing PDE suggests the employment of a scheme which allows to overcome the very restrictive time-stepping constraints imposed by a parabolic-type CFL condition. The SL approach is apt to this aim and we presented a complete workflow for approximating the solution of the surface reconstruction problem.

The first application of this method to uniform Cartesian grids showed good results, especially when a third-order WENO interpolator is employed. From a computational point of view, the three-dimensionality of the problem can of course pose an important issue, hence the parallel implementation of the algorithms, coupled with suitable strategies to localize the computational effort around the front, have been explored. Nonetheless, the uniform Cartesian nature of the spatial discretiza-

tion and of the parallel partitioning showed some expected drawbacks: when the number of cores is increased, also the chance of having large portion of the grid with no active nodes is increased as well, with a negative impact on load balancing.

To overcome this issue, exploiting both the natural refinement criterion offered by the level set function and the unconditional stability of the SL scheme, we investigated the application of AMR to this surface reconstruction method. The use of quadree (resp. octree) which are locally refined around the zero level set of ϕ, naturally mitigates the additional computational cost and memory consumption introduced by the implicit approach, while also enhancing the performances of the algorithm when run in parallel. Moreover, the very high resolution one could achieve employing adaptive grids does not pose a real issue in the SL context, due to the unconditional stability of the method.

Numerical results in two and three dimensions showed in fact the promising performances of the surface reconstruction method proposed in the AMR context. However, the method, as presented in Chapter 5 is still in a preliminary state and need to be investigated more, first of all reintroducing the third-order WENO interpolator. Given the AMR setting, the CWENO variant presented in Chapter 3 seems also very suitable for this task.

Regarding in general the surface reconstruction method, interesting directions of research would be also the replacement of the current semi-Lagrangian scheme with its monotone version (see [25]) to achieve a monotone decrease of the error and of the energy functional. Moreover, this approach can be extended to other moving interface problems, which include segmentation via LSM (see [51]) or free boundary problems where the moving boundary can be captured implicitly.

Finally, the results of this thesis will allow to enhance the computational workflow from point cloud data of works of art to PDE simulations of damage models in the field of conservation of cultural heritage. In particular, they make it possible to introduce AMR in that workflow and to extend it to more complex models that can involve material ablation or loss of the surface, or the swelling of the material itself [36].

Bibliography

[1] R. Abgrall. "On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation". In: *J. Comput. Phys.* 114.6 (1994), pp. 45–48 (cit. on p. 4).

[2] *AIM@SHAPE Shape Repository*. Accessed March 2022. URL: http://visionair.ge.imati.cnr.it/ontologies/shapes/view.jsp?id=268-frog_-_merged (cit. on pp. 100, 102, 127).

[3] S. Aluru and F. E. Sevilgen. "Parallel Domain Decomposition and Load Balancing Using Space-Filling Curves". In: *Proceedings of the Fourth International Conference on High-Performance Computing*. HIPC '97. IEEE Computer Society, 1997, p. 230. ISBN: 0818680679 (cit. on p. 112).

[4] N. Amenta, M. Bern, and M. Kamvysselis. "A new voronoi-based surface reconstruction algorithm". In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998*. 1998, pp. 415–422. DOI: 10.1145/280814.280947 (cit. on p. 1).

[5] A. Baeza et al. "Central WENO Schemes Through a Global Average Weight". In: *J. Sci. Comput.* 78.1 (2019), pp. 499–530. DOI: 10.1007/s10915-018-0773-z (cit. on pp. 4, 19).

[6] S. Balay et al. "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries". In: *Modern Software Tools in Scientific Computing*. Ed. by E. Arge, A. M. Bruaset, and H. P. Langtangen. Birkhäuser Press, 1997, pp. 163–202 (cit. on pp. 6, 64, 93).

[7] S. Balay et al. *PETSc/TAO Users Manual*. Tech. rep. ANL-21/39 - Revision 3.19. Argonne National Laboratory, 2023 (cit. on pp. 6, 64, 93).

[8] D. S. Balsara et al. "An efficient class of WENO schemes with adaptive order for unstructured meshes". In: *J. Comput. Phys.* 404 (2020), p. 109062. DOI: 10.1016/j.jcp.2019.109062 (cit. on pp. 4, 59).

[9] W. Bangerth, R. Hartmann, and G. Kanschat. "deal.IIA general-purpose object-oriented finite element library". In: *ACM Trans. Math. Softw.* 33.4 (Aug. 2007), 24–es. DOI: 10.1145/1268776.1268779 (cit. on p. 110).

[10] M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Systems & Control: Foundations & Applications. With appendices by Maurizio Falcone and Pierpaolo Soravia. Birkhäuser Boston, Inc., Boston, MA, 1997, pp. xviii+570. ISBN: 0-8176-3640-4 (cit. on p. 52).

[11] M. Bardi and S. Osher. "The Nonconvex Multidimensional Riemann Problem for HamiltonJacobi Equations". In: *SIAM J. Math. Anal.* 22.2 (1991), pp. 344–351. DOI: 10.1137/0522022 (cit. on p. 19).

[12] S. Barles and P. E. Souganidis. “Convergence of approximation schemes for fully nonlinear second order equations”. In: *Asymptotic Analysis* 4 (Jan. 1991), pp. 271–283. DOI: 10.3233/ASY-1991-4305 (cit. on p. 41).

[13] M. Berger et al. “A Survey of Surface Reconstruction from Point Clouds”. In: *Comput. Graph. Forum* 36.1 (2017), pp. 301–329. DOI: 10.1111/cgf.12802 (cit. on p. 2).

[14] M. Berger et al. “State of the Art in Surface Reconstruction from Point Clouds”. In: *Eurographics 2014-State of the Art Reports* (Apr. 2014) (cit. on p. 2).

[15] O. Bokanowski, N. Forcadel, and H. Zidani. “Reachability and minimal times for state constrained nonlinear problems without any controllability assumption”. In: *SIAM J. Control Optim.* 48.7 (2010), pp. 4292–4316. DOI: 10.1137/090762075 (cit. on pp. 72, 74).

[16] L. Bonaventura and R. Ferretti. “Semi-Lagrangian Methods for Parabolic Problems in Divergence Form”. In: *SIAM J. Sci. Comput.* 36.5 (2014), A2458–A2477. DOI: 10.1137/140969713 (cit. on p. 81).

[17] L. Bonaventura et al. “Second order fully semi-Lagrangian discretizations of advection-diffusion-reaction systems”. In: *J. Sci. Comput.* 88.23 (2021) (cit. on p. 63).

[18] S. Bryson and D. Levy. “High-Order Schemes for Multi-Dimensional Hamilton-Jacobi Equations”. In: *Hyperbolic Problems: Theory, Numerics, Applications*. Ed. by T. Y. Hou and E. Tadmor. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 387–396 (cit. on p. 4).

[19] C. Burstedde, L. C. Wilcox, and O. Ghattas. “p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees”. In: *SIAM J. Sci. Comput.* 33.3 (2011), pp. 1103–1133. DOI: 10.1137/100791634 (cit. on pp. 6, 110, 112).

[20] E. Calzola et al. “A semi-Lagrangian scheme for Hamilton-Jacobi-Bellman equations with oblique derivatives boundary conditions”. In: *Numer. Math.* 153.1 (2023), pp. 49–84 (cit. on p. 63).

[21] F. Camilli and M. Falcone. “An approximation scheme for the optimal control of di usion processes”. In: *RAIRO. Modélisation Mathématique et Analyse Numérique* 29 (Jan. 1995). DOI: 10.1051/m2an/1995290100971 (cit. on p. 79).

[22] P. M. Campbell et al. *Dynamic Octree Load Balancing Using Space-Filling Curves*. Tech. rep. CS-03-01. Williams College Department of Computer Science, 2003 (cit. on p. 112).

[23] E. Carlini, M. Falcone, and R. Ferretti. “Convergence of a large time-step scheme for mean curvature motion”. In: *Interfaces and Free Boundaries* 12.4 (2010), pp. 409–411. DOI: 10.4171/IFB/240 (cit. on pp. 79, 80).

[24] E. Carlini and R. Ferretti. “A Semi-Lagrangian approximation for the AMSS model of image processing”. In: *Appl. Numer. Math.* 73 (2013), pp. 16–32. DOI: https://doi.org/10.1016/j.apnum.2012.07.003 (cit. on p. 79).

[25] E. Carlini and R. Ferretti. “A Semi-Lagrangian Approximation of MinMax Type for the Stationary Mean Curvature Equation”. In: *Numerical Mathematics and Advanced Applications*. Jan. 2008, pp. 679–686. ISBN: 978-3-540-69776-3. DOI: 10.1007/978-3-540-69777-0_81 (cit. on p. 130).

[26] E. Carlini and R. Ferretti. “A Semi-Lagrangian Scheme with Radial Basis Approximation for Surface Reconstruction”. In: *Comput. Vis. Sci.* 18.2-3 (2017), pp. 103–112. DOI: 10.1007/s00791-016-0274-2 (cit. on pp. 3, 81, 84, 91, 92).

[27] E. Carlini, R. Ferretti, and G. Russo. “A weighted essentially nonoscillatory, large time-step scheme for Hamilton-Jacobi equations”. In: *SIAM J. Sci. Comput.* 27.3 (2006), pp. 1071–1091. DOI: 10.1137/040608787 (cit. on pp. 4, 5, 41, 43, 44, 50, 51, 61, 63, 129).

[28] E. Carlini et al. “A CWENO large time-step scheme for Hamilton-Jacobi equations”. In: *Comm. Appl. Math.* (2024). In press. URL: https://arxiv.org/abs/2402.15367 (cit. on pp. 5, 51).

[29] J. C. Carr et al. “Reconstruction and representation of 3D objects with radial basis functions”. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’01. 2001, pp. 67–76. DOI: 10.1145/383259.383266 (cit. on p. 2).

[30] J. C. Carr et al. “Smooth surface reconstruction from noisy range data”. In: *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, GRAPHITE ’03*. 2003, pp. 119–126+297. DOI: 10.1145/604471.604495 (cit. on p. 2).

[31] M. J. Castro and M. Semplice. “Third- and fourth-order well-balanced schemes for the shallow water equations based on the CWENO reconstruction”. In: *Int. J. Numer. Meth. Fluid* 89.8 (2019), pp. 304–325. DOI: 10.1002/fld.4700 (cit. on pp. 4, 53, 59).

[32] F. Cazals and J. Giesen. “Delaunay Triangulation Based Surface Reconstruction”. In: *Effective Computational Geometry for Curves and Surfaces*. Springer Berlin Heidelberg, 2006, pp. 231–276. DOI: 10.1007/978-3-540-33259-6_6 (cit. on p. 1).

[33] A. Chacon and A. Vladimirsky. “A Parallel Two-Scale Method for Eikonal Equations”. In: *SIAM J. Sci. Comput.* 37.1 (2015), A156–A180. DOI: 10.1137/12088197X (cit. on p. 118).

[34] Z. Q. Cheng et al. “A survey of methods for moving least squares surfaces”. In: *Proceedings of the Fifth Eurographics / IEEE VGTC Conference on Point-Based Graphics*. SPBG’08. Los Angeles, CA: Eurographics Association, 2008, pp. 9–23. ISBN: 9783905674125 (cit. on p. 2).

[35] F. Clarelli, B. De Filippo, and R. Natalini. “Mathematical model of copper corrosion”. In: *Appl. Math. Model.* 38.19-20 (2014), pp. 4804–4816. DOI: 10.1016/j.apm.2014.03.040 (cit. on p. 1).

[36] F. Clarelli, A. Fasano, and R. Natalini. “Mathematics and monument conservation: Free boundary models of marble sulfation”. In: *SIAM J. Appl. Math.* 69.1 (2008), pp. 149–168. DOI: 10.1137/070695125 (cit. on pp. 1, 130).

[37] A. Coco, S. Preda, and M. Semplice. “From Point Clouds to 3D Simulations of Marble Sulfation”. In: *Mathematical Modeling in Cultural Heritage*. Springer Nature Singapore, 2023, pp. 153–174. ISBN: 978-981-99-3679-3 (cit. on pp. 1, 5, 75, 79).

[38] A. Coco and G. Russo. “Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface”. In: *J. Comput. Phys.* 361 (2018), pp. 299–330. DOI: https://doi.org/10.1016/j.jcp.2018.01.016 (cit. on pp. 3, 11, 12).

[39] A. Coco, M. Semplice, and S. Serra Capizzano. “A level-set multigrid technique for nonlinear diffusion in the numerical simulation of marble degradation under chemical pollutants”. In: *Appl. Math. & Comput.* 386 (2020), p. 125503. DOI: 10.1016/j.amc.2020.125503 (cit. on pp. 1, 11).

[40] A. Coco et al. “Numerical Simulations of Marble Sulfation”. In: *Mathematical Modeling in Cultural Heritage*. Ed. by E. Bonetti et al. Cham: Springer International Publishing, 2021, pp. 107–122. DOI: 10.1007/978-3-030-58077-3_7 (cit. on p. 11).

[41] R. Courant, E. Isaacson, and Mina Rees. “On the solution of nonlinear hyperbolic differential equations by finite differences”. In: *Commun. on Pure and Appl. Math.* 5.3 (1952), pp. 243–255. DOI: 10.1002/cpa.3160050303 (cit. on pp. 3, 27, 29).

[42] I. Cravero, M. Semplice, and G. Visconti. “Optimal definition of the nonlinear weights in multidimensional Central WENOZ reconstructions”. In: *SIAM J. Numer. Anal.* 57.5 (2019), pp. 2328–2358. DOI: 10.1007/s10915-015-0123-3 (cit. on pp. 4, 55, 56, 58, 60, 61).

[43] I. Cravero et al. “CWENO: uniformly accurate reconstructions for balance laws”. In: *Math. Comp.* 87.312 (2018), pp. 1689–1719. DOI: http://dx.doi.org/10.1090/mcom/3273 (cit. on pp. 4, 19, 53, 55).

[44] P. Daniel et al. “Reconstruction of Surfaces from Point Clouds Using a Lagrangian Surface Evolution Model”. In: *Scale Space and Variational Methods in Computer Vision*. Springer International Publishing, 2015, pp. 589–600. ISBN: 978-3-319-18461-6 (cit. on p. 1).

[45] F. Drui et al. “Experimenting with the p4est library for AMR simulations of two-phase flows”. In: *ESAIM: Proceedings and Surveys* 53 (Mar. 2016), pp. 232–247. DOI: 10.1051/proc/201653014 (cit. on p. 110).

[46] J. Duchon and R. Robert. “Évolution dune interface par capillarité et diffusion de volume I. Existence locale en temps”. In: *Annales de l’Institut Henri Poincaré C, Analyse non linéaire* 1.5 (1984), pp. 361–378. DOI: https://doi.org/10.1016/S0294-1449(16)30418-8 (cit. on p. 16).

[47] M. Dumbser et al. “Central Weighted ENO Schemes for Hyperbolic Conservation Laws on fixed and moving unstructured meshes”. In: *SIAM J. Sci. Comput.* 39.6 (2017), A2564–A2591 (cit. on pp. 4, 53, 59, 61).

[48] H. Edelsbrunner. “Shape reconstruction with Delaunay complex”. In: *LATIN'98: Theoretical Informatics*. Springer Berlin Heidelberg, 1998, pp. 119–132. ISBN: 978-3-540-69715-2 (cit. on p. 1).

[49] L. C. Evans. *Partial Differential Equations*. Graduate Series in Mathematics, vol. 19.R. Providence, R.I. : American Mathematical Society, 2010. ISBN: 978-0-8218-4974-3 (cit. on p. 46).

[50] M. Falcone and R. Ferretti. “Consistency of a large time-step scheme for mean curvature motion”. In: *Numerical Mathematics and Advanced Applications*. Ed. by F. Brezzi et al. Springer Milan, 2003, pp. 495–502. DOI: 10.1007/978-88-470-2089-4_46 (cit. on pp. 3, 79, 80).

[51] M. Falcone, G. Paolucci, and S. Tozza. “A High-Order Scheme for Image Segmentation via a Modified Level-Set Method”. In: *SIAM J. Imaging Sci.* 13.1 (2020), pp. 497–534. DOI: 10.1137/18M1231432 (cit. on p. 130).

[52] M. Falcone, G. Paolucci, and S. Tozza. “Convergence of adaptive filtered schemes for first order evolutionary Hamilton-Jacobi equations”. In: *Numer. Math.* 145.2 (2020), pp. 271–311 (cit. on pp. 42, 60).

[53] M. Falcone. and R. Ferretti. “Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations”. In: *Numer. Math.* 67.3 (1994), pp. 315–344 (cit. on pp. 4, 51).

[54] M. Falcone. and R. Ferretti. *Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2014, pp. xii+319 (cit. on pp. 3, 27, 31, 33, 46, 48, 49).

[55] M. Falcone. and R. Ferretti. “Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods”. In: *J. Comput. Phys.* 175.2 (2002), pp. 559–575 (cit. on p. 51).

[56] R. Ferretti. “Convergence of semi-Lagrangian approximations to convex Hamilton-Jacobi equations under (very) large Courant numbers”. In: *SIAM J. Numer. Anal.* 40.6 (2002), pp. 2240–2253. DOI: 10.1137/S0036142901388378 (cit. on pp. 3–5, 41, 44, 48–50, 61, 62).

[57] F. Gibou, F. Fedkiw, and S. Osher. “A review of level-set methods and some recent applications”. In: *J. Comput. Phys.* 353 (2018), pp. 82–109. DOI: https://doi.org/10.1016/j.jcp.2017.10.006 (cit. on p. 2).

[58] F. Gibou et al. “A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains”. In: *J. Comput. Phys.* 176.1 (2002), pp. 205–227. DOI: https://doi.org/10.1006/jcph.2001.6977 (cit. on p. 3).

[59] F. Gibou et al. “A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains”. In: *J. Comput. Phys.* 176.1 (2002), pp. 205–227. DOI: https://doi.org/10.1006/jcph.2001.6977 (cit. on pp. 11, 12).

[60] B. F. Gregorski, B. Hamann, and K. I. Joy. “Reconstruction of B-spline surfaces from scattered data points”. In: *Proceedings Computer Graphics International 2000*. 2000, pp. 163–170. DOI: 10.1109/CGI.2000.852331 (cit. on p. 1).

[61] M. Griebel and G. Zumbusch. “Parallel multigrid in an adaptive PDE solver based on hashing and space-filling curves”. In: *Parallel Computing* 25.7 (1999), pp. 827–843. ISSN: 0167-8191. DOI: https://doi.org/10.1016/S0167-8191(99)00020-4 (cit. on p. 112).

[62] J. Haliková and K. Mikula. “Level Set Method for Surface Reconstruction and Its Application in Surveying”. In: *J. Surv. Eng.* 142.3 (Feb. 2016), p. 04016007. DOI: 10.1061/(ASCE)SU.1943-5428.0000159 (cit. on pp. 91, 92).

[63] E. Harabetian and S. Osher. “Regularization of ill-posed problems via the level set approach”. In: *SIAM J. Appl. Math.* 58.6 (Oct. 1998), pp. 1689–1706. DOI: 10.1137/S0036139995290794 (cit. on p. 2).

[64] E. Harabetian, S. Osher, and C. W. Shu. “An Eulerian Approach for Vortex Motion Using a Level Set Regularization Procedure”. In: *J. Comput. Phys.* 127.1 (1996), pp. 15–26. DOI: https://doi.org/10.1006/jcph.1996.0155 (cit. on p. 2).

[65] A. Harten et al. “Uniformly high-order accurate essentially nonoscillatory schemes III.” In: *J. Comput. Phys.* 71.2 (1987), pp. 231–303. DOI: 10.1016/0021-9991(87)90031-3 (cit. on pp. 4, 19, 41).

[66] D. Hartmann, M. Meinke, and W. Schröder. “Differential equation based constrained reinitialization for level set methods”. In: *J. Comput. Phys.* 227.14 (2008), pp. 6821–6845. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2008.03.040 (cit. on pp. 3, 20, 22).

[67] D. Hartmann, M. Meinke, and W. Schröder. “The constrained reinitialization equation for level set methods”. In: *J. Comput. Phys.* 229.5 (2010), pp. 1514–1535. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2009.10.042 (cit. on pp. 3, 20, 22).

[68] Y. He et al. “Fast Algorithms for Surface Reconstruction from Point Cloud”. In: *Springer Proceedings in Mathematics and Statistics*. Vol. 360. 2021, pp. 61–80. DOI: 10.1007/978-981-16-2701-9_4 (cit. on p. 91).

[69] M. Herrmann. *A Domain Decomposition Parallelization of the Fast Marching Method*. Tech. rep. DTIC Document, 2003 (cit. on p. 118).

[70] E. Hopf. “Generalized Solutions of non-linear Equations of First Order”. In: *J. Math. Mech.* 14.6 (1965), pp. 951–973 (cit. on p. 46).

[71] H. Hoppe et al. “Surface reconstruction from unorganized point clouds”. In: *SIGGRAPH Comput. Graph.* 26.2 (July 1992), pp. 71–78. DOI: https://doi.org/10.1145/142920.134011 (cit. on pp. 2, 99).

[72] C. Hu and C. W. Shu. “Weighted Essentially Non-oscillatory Schemes on Triangular Meshes”. In: *J. Comput. Phys.* 150.1 (1999), pp. 97–127. DOI: https://doi.org/10.1006/jcph.1998.6165 (cit. on p. 60).

[73] Z. J. Huang et al. “Surface Reconstruction from Point Clouds: A Survey and a Benchmark”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), pp. 1–20. DOI: 10.1109/TPAMI.2024.3429209 (cit. on p. 2).

[74] W. K. Jeong and R. T. Whitaker. “A Fast Iterative Method for Eikonal Equations”. In: *SIAM J. Sci. Comput.* 30.5 (2008), pp. 2512–2534. DOI: 10.1137/060670298 (cit. on p. 118).

[75] G. S. Jiang and D. Peng. “Weighted ENO Schemes for Hamilton–Jacobi Equations”. In: *SIAM J. Sci. Comput.* 21.6 (2000), pp. 2126–2143. DOI: 10.1137/S106482759732455X (cit. on pp. 4, 17, 42, 60).

[76] G. S. Jiang and C. W. Shu. “Efficient Implementation of Weighted ENO Schemes”. In: *J. Comput. Phys.* 126 (1996), pp. 202–228 (cit. on pp. 4, 19, 41–43).

[77] G. Karypis and V. Kumar. *METIS – Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0*. Tech. rep. University of Minnesota, Department of Computer Science, Jan. 1995 (cit. on p. 110).

[78] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Stochastic Modelling and Applied Probability. Springer Berlin, Heidelberg, 1992, pp. XXXVI, 636. ISBN: 978-3-642-08107-1 (cit. on p. 80).

[79] B. Kósa, J. Haliková-Brehovská, and K. Mikula. “New efficient numerical method for 3D point cloud surface reconstruction by using level set methods”. In: *Proceedings of Equadiff 2017 Conference* (2017), pp. 387–396. URL: http://www.iam.fmph.uniba.sk/amuc/ojs/index.php/equadiff/article/view/798 (cit. on p. 78).

[80] D. Levy, G. Puppo, and G. Russo. “Compact central WENO schemes for multidimensional conservation laws”. In: *SIAM J. Sci. Comput.* 22.2 (2000), pp. 656–672. DOI: 10.1137/S1064827599359461 (cit. on p. 4).

[81] C. T. Lin and E. Tadmor. “High-resolution nonoscillatory central schemes for Hamilton-Jacobi equations”. In: *SIAM J. Sci. Comput.* 21.6 (2000), pp. 2163–2186 (cit. on p. 4).

[82] X. Liu. “Research on 3D Object Reconstruction Method based on Deep Learning”. In: *Highl. Sci. Eng. Technol.* 39 (Apr. 2023), pp. 1221–1227. DOI: 10.54097/hset.v39i.6732 (cit. on p. 2).

[83] X. D. Liu, S. Osher, and T. Chan. “Weighted essentially non-oscillatory schemes”. In: *J. Comput. Phys.* 115.1 (1994), pp. 200–212 (cit. on pp. 4, 19).

[84] F. Losasso, F. Gibou, and R. Fedkiw. “Simulating water and smoke with an octree data structure”. In: *ACM SIGGRAPH 2004 Papers*. SIGGRAPH ’04. New York, NY, USA: Association for Computing Machinery, 2004, pp. 457–462. ISBN: 9781450378239. DOI: 10.1145/1186562.1015745 (cit. on p. 5).

[85] M. Marcon et al. “Fast point-cloud wrapping through level-set evolution”. In: *1st European Conference on Visual Media Production (CVMP) 2004*. 2004, pp. 119–125 (cit. on p. 2).

[86] B. Merriman, J. K. Bence, and S. J. Osher. “Motion of Multiple Junctions: A Level Set Approach”. In: *J. Comput. Phys.* 112.2 (1994), pp. 334–363. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.1994.1105 (cit. on p. 18).

[87] C. Min and F. Gibou. “A second order accurate level set method on non-graded adaptive cartesian grids”. In: *J. Comput. Phys.* 225.1 (2007), pp. 300–321. DOI: https://doi.org/10.1016/j.jcp.2006.11.034 (cit. on pp. 5, 118).

[88] M. Mirzadeh et al. “Parallel level-set methods on adaptive tree-based grids”. In: *J. Comput. Phys.* 322 (2016), pp. 345–364. DOI: https://doi.org/10.1016/j.jcp.2016.06.017 (cit. on pp. 110, 118).

[89] W. W. Mullins and R. F. Sekerka. “Morphological Stability of a Particle Growing by Diffusion or Heat Flow”. In: *J. Appl. Phys.* 34.2 (Feb. 1963), pp. 323–329. DOI: 10.1063/1.1702607 (cit. on p. 16).

[90] S. Osher and R. Fedkiw. “Level Set Methods and Dynamic Implicit Surfaces”. In: vol. 153. Appl. Math. Sci. Springer New York, NY, 2003, pp. XIII, 273. DOI: 10.1007/978-0-387-22746-7 (cit. on pp. 2, 6, 7, 11, 12).

[91] S. Osher and J. A. Sethian. “Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations”. In: *J. Comput. Phys.* 79.1 (1988), pp. 12–49. DOI: 10.1016/0021-9991(88)90002-2 (cit. on pp. 2, 6, 7, 12, 16, 19).

[92] S. Osher and C. W. Shu. “High-Order Essentially Nonoscillatory Schemes for HamiltonJacobi Equations”. In: *SIAM J. Numer. Anal.* 28.4 (1991), pp. 907–922. DOI: 10.1137/0728049 (cit. on pp. 4, 17, 19).

[93] I. K. Park, I. D. Yun, and S. U. Lee. “Constructing NURBS surface model from scattered and unorganized range data”. In: *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*. 1999, pp. 312–320. DOI: 10.1109/IM.1999.805361 (cit. on p. 1).

[94] D. Peng et al. “A PDE-Based Fast Local Level Set Method”. In: *J. Comput. Phys.* 155.2 (1999), pp. 410–438. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.1999.6345 (cit. on pp. 3, 7, 18, 22–24, 89, 116).

[95] S. Popinet. “Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries”. In: *J. Comput. Phys.* 190.2 (2003), pp. 572–600. ISSN: 0021-9991. DOI: https://doi.org/10.1016/S0021-9991(03)00298-5 (cit. on p. 5).

[96] S. Preda and M. Semplice. *Surface reconstruction from point cloud using a semi-Lagrangian scheme with local interpolator*. submitted. 2024. URL: https://arxiv.org/abs/2410.22205 (cit. on pp. 5, 75).

[97] J. Qiu and C. W. Shu. “Hermite WENO schemes for HamiltonJacobi equations”. In: *J. Comput. Phys.* 204.1 (2005), pp. 82–99. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2004.10.003 (cit. on p. 41).

[98] F. Remondino. “Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning”. In: *Remote Sensing* 3 (Dec. 2011), pp. 1104–1138. DOI: 10.3390/rs3061104 (cit. on pp. 1, 75).

[99] F. Remondino and S. El-Hakim. “Imagebased 3D Modelling: A Review”. In: *The Photogrammetric Record* 21 (Sept. 2006), pp. 269–291. DOI: 10.1111/j.1477-9730.2006.00383.x (cit. on pp. 1, 75).

[100] G. Russo and P. Smereka. “A Remark on Computing Distance Functions”. In: *J. Comput. Phys.* 163.1 (2000), pp. 51–67. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.2000.6553 (cit. on p. 20).

[101] R. Saye. “High-order methods for computing distances to implicitly defined surfaces”. In: *Commun. Appl. Math. Comput. Sci.* 9 (May 2014), pp. 107–141. DOI: 10.2140/camcos.2014.9.107 (cit. on pp. 118, 119).

[102] R. I. Saye and J. A. Sethian. “A review of level set methods to model interfaces moving under complex physics: Recent challenges and advances”. In: *Geometric Partial Differential Equations - Part I*. Vol. 21. Handbook of Numerical Analysis. Elsevier, 2020, pp. 509–554. DOI: https://doi.org/10.1016/bs.hna.2019.07.003 (cit. on p. 2).

[103] M. Semplice, A. Coco, and G. Russo. “Adaptive Mesh Refinement for Hyperbolic Systems based on Third-Order Compact WENO Reconstruction”. In: *J. Sci. Comput.* 66 (2016), pp. 692–724. DOI: 10.1007/s10915-015-0038-z (cit. on pp. 4, 53, 59).

[104] M. Semplice and G. Visconti. “Efficient Implementation of Adaptive Order Reconstructions”. In: *J. Sci. Comput.* 83.1 (2020). DOI: 10.1007/s10915-020-01156-6 (cit. on p. 4).

[105] J. Sethian. “An Analysis of Flame Propagation”. PhD thesis. University of California at Berkeley, 1982 (cit. on p. 16).

[106] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1999. ISBN: 9780521645577 (cit. on p. 2).

[107] R. Sharma et al. “Point Cloud Upsampling and Normal Estimation using Deep Learning for Robust Surface Reconstruction”. In: *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Jan. 2021, pp. 70–79. DOI: 10.5220/0010211600700079 (cit. on p. 2).

[108] M. H. Soner and N. Touzi. “A stochastic representation for mean curvature type geometric flows”. In: *The Annals of Probability* 31.3 (2003), pp. 1145–1165. DOI: 10.1214/aop/1055425773 (cit. on p. 79).

[109] *The Stanford 3D Scanning Repository*. Accessed March 2022. URL: http://graphics.stanford.edu/data/3Dscanrep/ (cit. on pp. 100, 102).

[110] J. Strain. "Semi-Lagrangian Methods for Level Set Equations". In: *J. Comput. Phys.* 151.2 (1999), pp. 498–533. DOI: https://doi.org/10.1006/jcph.1999.6194 (cit. on pp. 3, 7).

[111] J. Strain. "Tree Methods for Moving Interfaces". In: *J. Comput. Phys.* 151.2 (1999), pp. 616–648. DOI: https://doi.org/10.1006/jcph.1999.6205 (cit. on p. 5).

[112] M. Sussman, P. Smereka, and S. Osher. "A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow". In: *J. Comput. Phys.* 114.1 (1994), pp. 146–159. DOI: 10.1006/jcph.1994.1155 (cit. on pp. 2, 18).

[113] M. Sussman et al. "An Adaptive Level Set Approach for Incompressible Two-Phase Flows". In: *J. Comput. Phys.* 148.1 (1999), pp. 81–124. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.1998.6106 (cit. on p. 20).

[114] G. Tryggvason et al. "A Front-Tracking Method for the Computations of Multiphase Flow". In: *J. Comput. Phys.* 169.2 (2001), pp. 708–759. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.2001.6726 (cit. on p. 13).

[115] M. C. Tugurlan. "Fast Marching Methods - Parallel Implementation and Analysis". PhD thesis. 2008. ISBN: 9798802787359 (cit. on p. 118).

[116] S. O. Unverdi and G. Tryggvason. "A front-tracking method for viscous, incompressible, multi-fluid flows". In: *J. Comput. Phys.* 100.1 (1992), pp. 25–37. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(92)90307-K (cit. on p. 13).

[117] Q. Wang, Y. Tan, and Z. Mei. "Computational Methods of Acquisition and Processing of 3D Point Cloud Data for Construction Applications". In: *Archives of Computational Methods in Engineering* 27 (Apr. 2020), pp. 479–499. DOI: 10.1007/s11831-019-09320-4 (cit. on p. 1).

[118] Y. Zeng and Y. Zhu. "Implicit surface reconstruction based on a new interpolation / approximation radial basis function". In: *Comput. Aided Geom. Des.* 92 (Dec. 2021), p. 102062. DOI: 10.1016/j.cagd.2021.102062 (cit. on p. 2).

[119] M. Zennaro. "Natural continuous extensions of Runge-Kutta methods". In: *Math. Comp.* 46.173 (1986), pp. 119–133. DOI: 10.1090/S0025-5718-1986-0815835-1 (cit. on pp. 39, 40).

[120] H. Zhao. "Parallel implementations of the fast sweeping method". In: *Journal of Computational Mathematics* 25.4 (2007), pp. 421–429 (cit. on p. 118).

[121] H. K. Zhao. "A fast sweeping method for Eikonal equations". In: *Math. Comput.* 74 (2005), pp. 603–627. DOI: 10.1090/S0025-5718-04-01678-3 (cit. on pp. 87, 88).

[122] H. K. Zhao et al. "A Variational Level Set Approach to Multiphase Motion". In: *J. Comput. Phys.* 127.1 (1996), pp. 179–195. DOI: https://doi.org/10.1006/jcph.1996.0167 (cit. on p. 2).

[123] H. K. Zhao et al. "Capturing the Behavior of Bubbles and Drops Using the Variational Level Set Approach". In: *J. Comput. Phys.* 143.2 (1998), pp. 495–518. DOI: https://doi.org/10.1006/jcph.1997.5810 (cit. on p. 2).

[124] H. K. Zhao et al. “Implicit and Nonparametric Shape Reconstruction from Unorganized Data Using a Variational Level Set Method”. In: *Comput. Vis. Image Underst.* 80.3 (2000), pp. 295–314. DOI: 10.1006/cviu.2000.0875 (cit. on pp. 2, 6, 76, 88, 91, 92, 99).

[125] F. Zheng, C. W. Shu, and J. Qiu. “High order finite difference Hermite WENO schemes for the HamiltonJacobi equations on unstructured meshes”. In: *Comp. & Fluids* 183 (2019), pp. 53–65. DOI: 10.1016/j.compfluid.2019.02.010 (cit. on pp. 4, 19, 41).

[126] J. Zhou, L. Cai, and F.-Q. Zhou. “New high-resolution scheme for three-dimensional nonlinear hyperbolic conservation laws”. In: *Appl. Math. Comp.* 198.2 (2008), pp. 770–786. DOI: 10.1016/j.amc.2007.09.017 (cit. on p. 4).

[127] J. Zhu and J. Qiu. “A new fifth order finite difference WENO scheme for Hamilton-Jacobi equations”. In: *Numer. Meth. for PDEs* 33.4 (2017), pp. 1095–1113. DOI: 10.1002/num.22133 (cit. on pp. 4, 19, 41).

[128] J. Zhu and J. Qiu. “A new fifth order finite difference WENO scheme for solving hyperbolic conservation laws”. In: *J. Comput. Phys.* 318 (2016), pp. 110–121. DOI: 10.1016/j.jcp.2016.05.010 (cit. on pp. 4, 19).

[129] J. Zhu and J. Qiu. “A New Type of High-Order WENO Schemes for Hamilton-Jacobi Equations on Triangular Meshes”. In: *Comm. Computat. Phys.* 27.3 (2020), pp. 897–920. DOI: 10.4208/cicp.OA-2018-0156 (cit. on pp. 4, 41, 59).

Appendix

We report here the matrix involved in the definition (3.25).

$$
M = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 0 & 0 & 6 & 2 & 0 & 0 & 3 & \frac{4}{3} & 0 & 1 & \frac{3}{2} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 2 & 6 & 0 & \frac{4}{3} & 3 & 1 & \frac{3}{2} \\
0 & 0 & 0 & 6 & 0 & 0 & 0 & 84 & 3 & 0 & 0 & 42 & 2 & 0 & 28 & 21 \\
0 & 0 & 0 & 2 & 1 & 0 & 3 & 0 & \frac{32}{3} & 1 & 0 & \frac{31}{2} & \frac{31}{3} & 1 & 15 & \frac{147}{10} \\
0 & 0 & 0 & 0 & 1 & 2 & 0 & 1 & \frac{32}{3} & 3 & 84 & 0 & 2 & 42 & \frac{3}{2} & \frac{152}{15} & 15 & 28 & \frac{152}{15} & 15 & \frac{147}{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 3 & 1 & 0 & \frac{989}{5} & 15 & 1 & \frac{954}{5} & \frac{1918}{15} & \frac{147}{10} & \frac{954}{5} & \frac{147}{10} & 21 \\
0 & 0 & 0 & 3 & 1 & 0 & 42 & \frac{31}{2} & 1 & \frac{31}{3} & 2 & 15 & \frac{3992}{45} & 15 & \frac{1918}{15} & \frac{147}{10} & \frac{56076}{35} & \frac{933}{5} & \frac{737}{4} & \frac{933}{5} & \frac{737}{4} \\
0 & 0 & 0 & \frac{4}{3} & 1 & \frac{4}{3} & 2 & \frac{31}{3} & \frac{31}{3} & \frac{31}{2} & 42 & 1 & 15 & \frac{989}{5} & \frac{147}{10} & \frac{954}{5} & \frac{737}{4} & \frac{161501}{70} & \frac{1918}{15} & \frac{954}{5} & \frac{161501}{70} \\
0 & 0 & 0 & 0 & 0 & 1 & 3 & 0 & 1 & \frac{31}{2} & \frac{3}{2} & 28 & \frac{1918}{15} & \frac{1918}{15} & \frac{147}{10} & \frac{737}{4} & \frac{161501}{70} & \frac{721401}{25} & \frac{161501}{70} & \frac{737}{4} & \frac{933}{5} \\
0 & 0 & 0 & 2 & 1 & 1 & 28 & 15 & \frac{152}{15} & \frac{31}{3} & \frac{3}{2} & 21 & \frac{954}{5} & \frac{954}{5} & \frac{933}{5} & \frac{161501}{70} & \frac{737}{4} & \frac{933}{5} & \frac{161501}{70} & \frac{147}{10} & \frac{147}{10} \\
0 & 0 & 0 & 1 & 1 & 2 & \frac{3}{2} & \frac{152}{15} & 15 & \frac{147}{10} & 21 & \frac{737}{4} & \frac{737}{4} & \frac{933}{5} & \frac{161501}{70} & \frac{721401}{25} & \frac{161501}{70} & \frac{147}{10} & \frac{737}{4} & \frac{147}{10} & \frac{933}{5} \\
0 & 0 & 0 & \frac{3}{2} & 1 & \frac{3}{2} & 21 & \frac{147}{10} & \frac{147}{10} & \frac{933}{5} & \frac{737}{4} & \frac{161501}{70} & \frac{161501}{70} & \frac{147}{10} & \frac{737}{4} & \frac{933}{5} & \frac{147}{10} & \frac{933}{5} & \frac{147}{10} & \frac{933}{5} & \frac{737}{4}
\end{pmatrix}
$$

Also, we report the matrices involved in the definitions (3.26). We assume that the vector U of data in the 4×4 stencil is ordered lexicographically, i.e. x direction faster and y direction slower.

$$A_{\text{ne}} = \begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{3497}{720}&\frac{-623}{90}&\frac{1787}{720}&0&\frac{-623}{90}&\frac{1667}{180}&\frac{-571}{180}&0&\frac{1787}{720}&\frac{-571}{180}&\frac{797}{720}\\
0&0&0&0&0&\frac{-623}{90}&\frac{1141}{90}&\frac{-229}{45}&0&\frac{1667}{180}&\frac{-1547}{90}&\frac{1187}{180}&0&\frac{-571}{180}&\frac{278}{45}&\frac{-421}{180}\\
0&0&0&0&0&\frac{1787}{720}&\frac{-229}{45}&\frac{1817}{720}&0&\frac{-571}{180}&\frac{1187}{180}&\frac{-293}{90}&0&\frac{797}{720}&\frac{-421}{180}&\frac{827}{720}\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{-623}{90}&\frac{1667}{180}&\frac{-571}{180}&0&\frac{1141}{90}&\frac{-1547}{90}&\frac{278}{45}&0&\frac{-229}{45}&\frac{1187}{180}&\frac{-421}{180}\\
0&0&0&0&0&\frac{1667}{180}&\frac{-1547}{90}&\frac{1187}{180}&0&\frac{-1547}{90}&\frac{1472}{45}&\frac{-1157}{90}&0&\frac{1187}{180}&\frac{-1157}{90}&\frac{887}{180}\\
0&0&0&0&0&\frac{-571}{180}&\frac{1187}{180}&\frac{-293}{90}&0&\frac{278}{45}&\frac{-1157}{90}&\frac{571}{90}&0&\frac{-421}{180}&\frac{887}{180}&\frac{-109}{45}\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{1787}{720}&\frac{-571}{180}&\frac{797}{720}&0&\frac{-229}{45}&\frac{1187}{180}&\frac{-421}{180}&0&\frac{1817}{720}&\frac{-293}{90}&\frac{827}{720}\\
0&0&0&0&0&\frac{-571}{180}&\frac{278}{45}&\frac{-421}{180}&0&\frac{1187}{180}&\frac{-1157}{90}&\frac{887}{180}&0&\frac{-293}{90}&\frac{571}{90}&\frac{-109}{45}\\
0&0&0&0&0&\frac{797}{720}&\frac{-421}{180}&\frac{827}{720}&0&\frac{-421}{180}&\frac{887}{180}&\frac{-109}{45}&0&\frac{827}{720}&\frac{-109}{45}&\frac{857}{720}
\end{pmatrix}$$

$$A_{\mathrm{nw}} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1817}{720} & \frac{-229}{45} & \frac{1787}{720} & 0 & \frac{-293}{90} & \frac{1187}{180} & \frac{-571}{180} & 0 & \frac{827}{720} & \frac{-421}{180} & \frac{797}{720} & 0 \\
0 & 0 & 0 & 0 & \frac{-229}{45} & \frac{1141}{90} & \frac{-623}{90} & 0 & \frac{1187}{180} & \frac{-1547}{90} & \frac{1667}{180} & 0 & \frac{-421}{180} & \frac{278}{45} & \frac{-571}{180} & 0 \\
0 & 0 & 0 & 0 & \frac{1787}{720} & \frac{-623}{90} & \frac{3497}{720} & 0 & \frac{-571}{180} & \frac{1667}{180} & \frac{-623}{90} & 0 & \frac{797}{720} & \frac{-571}{180} & \frac{1787}{720} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{-293}{90} & \frac{1187}{180} & \frac{-571}{180} & 0 & \frac{571}{90} & \frac{-1157}{90} & \frac{278}{45} & 0 & \frac{-109}{45} & \frac{887}{180} & \frac{-421}{180} & 0 \\
0 & 0 & 0 & 0 & \frac{1187}{180} & \frac{-1547}{90} & \frac{1667}{180} & 0 & \frac{-1157}{90} & \frac{1472}{45} & \frac{-1547}{90} & 0 & \frac{887}{180} & \frac{-1157}{90} & \frac{1187}{180} & 0 \\
0 & 0 & 0 & 0 & \frac{-571}{180} & \frac{1667}{180} & \frac{-623}{90} & 0 & \frac{278}{45} & \frac{-1547}{90} & \frac{1141}{90} & 0 & \frac{-421}{180} & \frac{1187}{180} & \frac{-229}{45} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{827}{720} & \frac{-421}{180} & \frac{797}{720} & 0 & \frac{-109}{45} & \frac{887}{180} & \frac{-421}{180} & 0 & \frac{857}{720} & \frac{-109}{45} & \frac{827}{720} & 0 \\
0 & 0 & 0 & 0 & \frac{-421}{180} & \frac{278}{45} & \frac{-571}{180} & 0 & \frac{887}{180} & \frac{-1157}{90} & \frac{1187}{180} & 0 & \frac{-109}{45} & \frac{571}{90} & \frac{-293}{90} & 0 \\
0 & 0 & 0 & 0 & \frac{797}{720} & \frac{-571}{180} & \frac{1787}{720} & 0 & \frac{-421}{180} & \frac{1187}{180} & \frac{-229}{45} & 0 & \frac{827}{720} & \frac{-293}{90} & \frac{1817}{720} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix},$$

$$A_{\mathrm{sw}} = \begin{pmatrix}
\frac{857}{720} & \frac{-109}{45} & \frac{827}{720} & 0 & \frac{-109}{45} & \frac{887}{180} & \frac{-421}{180} & 0 & \frac{827}{720} & \frac{-421}{180} & \frac{797}{720} & 0 & 0 & 0 & 0 & 0 \\
\frac{-109}{45} & \frac{571}{90} & \frac{-293}{90} & 0 & \frac{887}{180} & \frac{-1157}{90} & \frac{1187}{180} & 0 & \frac{-421}{180} & \frac{278}{45} & \frac{-571}{180} & 0 & 0 & 0 & 0 & 0 \\
\frac{827}{720} & \frac{-293}{90} & \frac{1817}{720} & 0 & \frac{-421}{180} & \frac{1187}{180} & \frac{-229}{45} & 0 & \frac{797}{720} & \frac{-571}{180} & \frac{1787}{720} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{-109}{45} & \frac{887}{180} & \frac{-421}{180} & 0 & \frac{571}{90} & \frac{-1157}{90} & \frac{278}{45} & 0 & \frac{-293}{90} & \frac{1187}{180} & \frac{-571}{180} & 0 & 0 & 0 & 0 & 0 \\
\frac{887}{180} & \frac{-1157}{90} & \frac{1187}{180} & 0 & \frac{-1157}{90} & \frac{1472}{45} & \frac{-1547}{90} & 0 & \frac{1187}{180} & \frac{-1547}{90} & \frac{1667}{180} & 0 & 0 & 0 & 0 & 0 \\
\frac{-421}{180} & \frac{1187}{180} & \frac{-229}{45} & 0 & \frac{278}{45} & \frac{-1547}{90} & \frac{1141}{90} & 0 & \frac{-571}{180} & \frac{1667}{180} & \frac{-623}{90} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{827}{720} & \frac{-421}{180} & \frac{797}{720} & 0 & \frac{-293}{90} & \frac{1187}{180} & \frac{-571}{180} & 0 & \frac{1817}{720} & \frac{-229}{45} & \frac{1787}{720} & 0 & 0 & 0 & 0 & 0 \\
\frac{-421}{180} & \frac{278}{45} & \frac{-571}{180} & 0 & \frac{1187}{180} & \frac{-1547}{90} & \frac{1667}{180} & 0 & \frac{-229}{45} & \frac{1141}{90} & \frac{-623}{90} & 0 & 0 & 0 & 0 & 0 \\
\frac{797}{720} & \frac{-571}{180} & \frac{1787}{720} & 0 & \frac{-571}{180} & \frac{1667}{180} & \frac{-623}{90} & 0 & \frac{1787}{720} & \frac{-623}{90} & \frac{3497}{720} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix},$$

.

$$
A_{\mathrm{se}} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1817}{720} & \frac{-293}{90} & \frac{827}{720} & 0 & \frac{-229}{45} & \frac{1187}{180} & \frac{-421}{180} & 0 & \frac{1787}{720} & \frac{-571}{180} & \frac{797}{720} & 0 & 0 & 0 & 0 \\
0 & \frac{-293}{90} & \frac{571}{90} & \frac{-109}{45} & 0 & \frac{1187}{180} & \frac{-1157}{90} & \frac{887}{180} & 0 & \frac{-571}{180} & \frac{278}{45} & \frac{-421}{180} & 0 & 0 & 0 & 0 \\
0 & \frac{827}{720} & \frac{-109}{45} & \frac{857}{720} & 0 & \frac{-421}{180} & \frac{887}{180} & \frac{-109}{45} & 0 & \frac{797}{720} & \frac{-421}{180} & \frac{827}{720} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{-229}{45} & \frac{1187}{180} & \frac{-421}{180} & 0 & \frac{1141}{90} & \frac{-1547}{90} & \frac{278}{45} & 0 & \frac{-623}{90} & \frac{1667}{180} & \frac{-571}{180} & 0 & 0 & 0 & 0 \\
0 & \frac{1187}{180} & \frac{-1157}{90} & \frac{887}{180} & 0 & \frac{-1547}{90} & \frac{1472}{45} & \frac{-1157}{90} & 0 & \frac{1667}{180} & \frac{-1547}{90} & \frac{1187}{180} & 0 & 0 & 0 & 0 \\
0 & \frac{-421}{180} & \frac{887}{180} & \frac{-109}{45} & 0 & \frac{278}{45} & \frac{-1157}{90} & \frac{571}{90} & 0 & \frac{-571}{180} & \frac{1187}{180} & \frac{-293}{90} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1787}{720} & \frac{-571}{180} & \frac{797}{720} & 0 & \frac{-623}{90} & \frac{1667}{180} & \frac{-571}{180} & 0 & \frac{3497}{720} & \frac{-623}{90} & \frac{1787}{720} & 0 & 0 & 0 & 0 \\
0 & \frac{-571}{180} & \frac{278}{45} & \frac{-421}{180} & 0 & \frac{1667}{180} & \frac{-1547}{90} & \frac{1187}{180} & 0 & \frac{-623}{90} & \frac{1141}{90} & \frac{-229}{45} & 0 & 0 & 0 & 0 \\
0 & \frac{797}{720} & \frac{-421}{180} & \frac{827}{720} & 0 & \frac{-571}{180} & \frac{1187}{180} & \frac{-293}{90} & 0 & \frac{1787}{720} & \frac{-229}{45} & \frac{1817}{720} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

The matrix A_{opt} appearing in (3.26) is splitted into two matrices $A_{\mathrm{opt}}^{(1)}$ and $A_{\mathrm{opt}}^{(2)}$ reporting the first eight columns and the second eight columns of A_{opt}.

$$A^{(1)}_{\mathrm{opt}} = \begin{pmatrix}
\frac{2903}{1575} & \frac{-182131}{37800} & \frac{10859}{2700} & \frac{-13889}{12600} & \frac{-182131}{37800} & \frac{60463}{4800} & \frac{-58881}{5600} & \frac{871553}{302400} \\
\frac{-182131}{37800} & \frac{48799}{3150} & \frac{-58687}{4200} & \frac{10859}{2700} & \frac{60463}{4800} & \frac{-340087}{8400} & \frac{1226693}{33600} & \frac{-58881}{5600} \\
\frac{10859}{2700} & \frac{-58687}{4200} & \frac{48799}{3150} & \frac{-182131}{37800} & \frac{-58881}{5600} & \frac{1226693}{33600} & \frac{-340087}{8400} & \frac{60463}{4800} \\
\frac{-13889}{12600} & \frac{10859}{2700} & \frac{-182131}{37800} & \frac{2903}{1575} & \frac{871553}{302400} & \frac{-58881}{5600} & \frac{60463}{4800} & \frac{-182131}{37800} \\
\frac{-182131}{37800} & \frac{60463}{4800} & \frac{-58881}{5600} & \frac{871553}{302400} & \frac{48799}{3150} & \frac{-340087}{8400} & \frac{426623}{12600} & \frac{-11137}{1200} \\
\frac{60463}{4800} & \frac{-340087}{8400} & \frac{1226693}{33600} & \frac{-58881}{5600} & \frac{-340087}{8400} & \frac{815527}{6300} & \frac{-2958593}{25200} & \frac{426623}{12600} \\
\frac{-58881}{5600} & \frac{1226693}{33600} & \frac{-340087}{8400} & \frac{60463}{4800} & \frac{426623}{12600} & \frac{-2958593}{25200} & \frac{815527}{6300} & \frac{-340087}{8400} \\
\frac{871553}{302400} & \frac{-58881}{5600} & \frac{60463}{4800} & \frac{-182131}{37800} & \frac{-11137}{1200} & \frac{426623}{12600} & \frac{-340087}{8400} & \frac{48799}{3150} \\
\frac{10859}{2700} & \frac{-58881}{5600} & \frac{221047}{25200} & \frac{-363499}{151200} & \frac{-58687}{4200} & \frac{1226693}{33600} & \frac{-1536757}{50400} & \frac{841823}{100800} \\
\frac{-58881}{5600} & \frac{426623}{12600} & \frac{-1536757}{50400} & \frac{221047}{25200} & \frac{1226693}{33600} & \frac{-2958593}{25200} & \frac{10709107}{100800} & \frac{-1536757}{50400} \\
\frac{221047}{25200} & \frac{-1536757}{50400} & \frac{426623}{12600} & \frac{-58881}{5600} & \frac{-1536757}{50400} & \frac{10709107}{100800} & \frac{-2958593}{25200} & \frac{1226693}{33600} \\
\frac{-363499}{151200} & \frac{221047}{25200} & \frac{-58881}{5600} & \frac{10859}{2700} & \frac{841823}{100800} & \frac{-1536757}{50400} & \frac{1226693}{33600} & \frac{-58687}{4200} \\
\frac{-13889}{12600} & \frac{871553}{302400} & \frac{-363499}{151200} & \frac{66427}{100800} & \frac{10859}{2700} & \frac{-58881}{5600} & \frac{221047}{25200} & \frac{-363499}{151200} \\
\frac{871553}{302400} & \frac{-11137}{1200} & \frac{841823}{100800} & \frac{-363499}{151200} & \frac{-58881}{5600} & \frac{426623}{12600} & \frac{-1536757}{50400} & \frac{221047}{25200} \\
\frac{-363499}{151200} & \frac{841823}{100800} & \frac{-11137}{1200} & \frac{871553}{302400} & \frac{221047}{25200} & \frac{-1536757}{50400} & \frac{426623}{12600} & \frac{-58881}{5600} \\
\frac{66427}{100800} & \frac{-363499}{151200} & \frac{871553}{302400} & \frac{-13889}{12600} & \frac{-363499}{151200} & \frac{221047}{25200} & \frac{-58881}{5600} & \frac{10859}{2700}
\end{pmatrix},$$

$$A_{\text{opt}}^{(2)} = \begin{pmatrix} \frac{10859}{2700} & \frac{-58881}{5600} & \frac{221047}{25200} & \frac{-363499}{151200} & \frac{-13889}{12600} & \frac{871553}{302400} & \frac{-363499}{151200} & \frac{66427}{100800} \\ \frac{-58881}{5600} & \frac{426623}{12600} & \frac{-1536757}{50400} & \frac{221047}{25200} & \frac{871553}{302400} & \frac{-11137}{1200} & \frac{841823}{100800} & \frac{-363499}{151200} \\ \frac{221047}{25200} & \frac{-1536757}{50400} & \frac{426623}{12600} & \frac{-58881}{5600} & \frac{-363499}{151200} & \frac{841823}{100800} & \frac{-11137}{1200} & \frac{871553}{302400} \\ \frac{-363499}{151200} & \frac{221047}{25200} & \frac{-58881}{5600} & \frac{10859}{2700} & \frac{66427}{100800} & \frac{-363499}{151200} & \frac{871553}{302400} & \frac{-13889}{12600} \\ \frac{-58687}{4200} & \frac{1226693}{33600} & \frac{-1536757}{50400} & \frac{841823}{100800} & \frac{10859}{2700} & \frac{-58881}{5600} & \frac{221047}{25200} & \frac{-363499}{151200} \\ \frac{1226693}{33600} & \frac{-2958593}{25200} & \frac{10709107}{100800} & \frac{-1536757}{50400} & \frac{-58881}{5600} & \frac{426623}{12600} & \frac{-1536757}{50400} & \frac{221047}{25200} \\ \frac{-1536757}{50400} & \frac{10709107}{100800} & \frac{-2958593}{25200} & \frac{1226693}{33600} & \frac{221047}{25200} & \frac{-1536757}{50400} & \frac{426623}{12600} & \frac{-58881}{5600} \\ \frac{841823}{100800} & \frac{-1536757}{50400} & \frac{1226693}{33600} & \frac{-58687}{4200} & \frac{-363499}{151200} & \frac{221047}{25200} & \frac{-58881}{5600} & \frac{10859}{2700} \\ \frac{48799}{3150} & \frac{-340087}{8400} & \frac{426623}{12600} & \frac{-11137}{1200} & \frac{-182131}{37800} & \frac{60463}{4800} & \frac{-58881}{5600} & \frac{871553}{302400} \\ \frac{-340087}{8400} & \frac{815527}{6300} & \frac{-2958593}{25200} & \frac{426623}{12600} & \frac{60463}{4800} & \frac{-340087}{8400} & \frac{1226693}{33600} & \frac{-58881}{5600} \\ \frac{426623}{12600} & \frac{-2958593}{25200} & \frac{815527}{6300} & \frac{-340087}{8400} & \frac{-58881}{5600} & \frac{1226693}{33600} & \frac{-340087}{8400} & \frac{60463}{4800} \\ \frac{-11137}{1200} & \frac{426623}{12600} & \frac{-340087}{8400} & \frac{48799}{3150} & \frac{871553}{302400} & \frac{-58881}{5600} & \frac{60463}{4800} & \frac{-182131}{37800} \\ \frac{-182131}{37800} & \frac{60463}{4800} & \frac{-58881}{5600} & \frac{871553}{302400} & \frac{2903}{1575} & \frac{-182131}{37800} & \frac{10859}{2700} & \frac{-13889}{12600} \\ \frac{60463}{4800} & \frac{-340087}{8400} & \frac{1226693}{33600} & \frac{-58881}{5600} & \frac{-182131}{37800} & \frac{48799}{3150} & \frac{-58687}{4200} & \frac{10859}{2700} \\ \frac{-58881}{5600} & \frac{1226693}{33600} & \frac{-340087}{8400} & \frac{60463}{4800} & \frac{10859}{2700} & \frac{-58687}{4200} & \frac{48799}{3150} & \frac{-182131}{37800} \\ \frac{871553}{302400} & \frac{-58881}{5600} & \frac{60463}{4800} & \frac{-182131}{37800} & \frac{-13889}{12600} & \frac{10859}{2700} & \frac{-182131}{37800} & \frac{2903}{1575} \end{pmatrix}.$$