UC Santa Barbara

UC Santa Barbara Electronic Theses and Dissertations

Title

From single-phase to two-phase sharp-interface incompressible viscous flow simulation on distributed adaptive Quadtree/Octree grids

Permalink

https://escholarship.org/uc/item/6v54t604

Author

Egan, Raphael Michael

Publication Date

2021

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library University of California

University of California
Santa Barbara

From single-phase to two-phase sharp-interface incompressible viscous flow simulation on distributed adaptive Quadtree/Octree grids

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Mechanical Engineering

by

Raphael M. Egan

Committee in charge:

Professor Frédéric Gibou, Chair
Professor Paolo Luzzatto-Fegiz
Professor Igor Mezić
Professor John Gilbert

September 2021

The Dissertation of Raphael M. Egan is approved.

Professor Paolo Luzzatto-Fegiz

Professor Igor Mezić

Professor John Gilbert

Professor Frédéric Gibou, Committee Chair

June 2021

From single-phase to two-phase sharp-interface incompressible viscous flow simulation on distributed adaptive Quadtree/Octree grids

Copyright © 2021

by

Raphael M. Egan

Curriculum Vitæ
Raphael M. Egan

Education

2021	Ph.D. in Mechanical Engineering (Expected), University of California, Santa Barbara.
2014	M.Sc. in Engineering Physics, University of Liège, Belgium.
2012	B.Sc. in Engineering Sciences, University of Liège, Belgium.

Publications

- **Raphael Egan**, Arthur Guittet, Fernando Temprano-Coleto, Tobin Isaac, François J. Peaudecerf, Julien R. Landel, Paolo Luzzatto-Fegiz, Carsten Burstedde, and Frédéric Gibou: Direct numerical simulation of incompressible flows on parallel Octree grids. Journal of Computational Physics 428 (2021): 110084.
- **Raphael Egan** and Frédéric Gibou: xGFM: Recovering convergence of fluxes in the ghost fluid method. Journal of Computational Physics 409 (2020): 109351.
- **Raphael Egan** and Frédéric Gibou: Fast and scalable algorithms for constructing Solvent-Excluded Surfaces of large biomolecules. Journal of Computational Physics 374 (2018): 91-120.
- **Raphael Egan** and Frédéric Gibou: Geometric discretization of the multidimensional Dirac delta distribution Application to the Poisson equation with singular source terms. Journal of Computational Physics 346 (2017): 71-90.
- **Raphael Egan**, Matthieu Philippe, Laurent Wera, Jean-François Fagnard, Benoît Vanderheyden, Anthony Dennis, Yunhua Shi, David A. Cardwell, and Philippe Vanderbemden: A flux extraction device to measure the magnetic moment of large samples; application to bulk superconductors. Review of Scientific Instruments 86, no. 2 (2015): 025107.

Abstract

From single-phase to two-phase sharp-interface incompressible viscous flow simulation on distributed adaptive Quadtree/Octree grids

by

Raphael M. Egan

With the rise of computational power and the democratized access to supercomputers, numerical simulations have emerged as a new standard tool of scientific investigation over the last century, often complementing experimental and theoretical approaches. Among the most challenging problems to be tackled numerically, one finds multi-scale and free-boundary phenomena. Multi-scale problems typically prevent the use of uniform meshes; free boundaries may develop complex geometric features making body-fitted mesh generation challenging. The present dissertation covers the development of numerical tools and methods for implicitly-captured interfaces on distributed Quadtree/Octree grids, which are thus well-suited for such challenging problems.

The first part of this work focuses on the extension of numerical methods for the simulation of incompressible, viscous flows from single-phase to two-phase problems, on distributed Quadtree/Octree grids. In order to reconcile the numerical description of the phenomenon with its continuum-mechanics description, this extension requires the development of several sharp-interface numerical techniques capable of capturing discontinuities in material parameters as well as in the primary unknowns. In addition, the description of the first-principle conservation laws across a sharp interface underlines that interface discontinuities in primary unknowns depend on the solution itself, highlighting the need for novel numerical methods. The development of such methods is presented, and their combination into a simulation engine for incompressible, viscous two-phase flows is illustrated.

In the second part of this dissertation, two separate and independent works are presented. First, a numerical discretization for the point-located Dirac distribution is presented. Beyond its theoretical interest, the Dirac distribution is sometimes used for modeling extreme multiscale events that cannot be fully resolved even with adaptive grid capabilities: that project presents a simple, geometric discretization that is shown to correctly reproduce the expected behavior over finite length scales. Second, a series of algorithms implementing a parallel, load-balanced divide-and-conquer strategy for constructing levelset representations of the Solvent-Excluded Surfaces of large biomolecular compounds are presented. This set of algorithms is shown to successfully accelerate the construction of such highly convoluted interfaces and to satisfy strong scaling, opening the door to more efficient calculations of interface-resolved electrostatics phenomena around such complex molecular structures.

Contents

Part I

From single-phase to two-phase incompressible viscous flow simulation

Chapter 1

Introduction

Applications and problems involving the dynamics between two immiscible fluids are omnipresent in our everyday lives. To cite only a few examples, droplets, bubbles, sprays, waves and problems involving a phase change (boiling, evaporation or cavitation) are all characterized by the existence of two drastically different fluids coupled across the interface that separates them. As a natural consequence, a plethora of engineering and life science applications fall into this extremely broad category: for instance, design of ink-jet printer, oil extraction strategies, micro-fluidic devices, fuel atomization in engines, breakup of waves, spray optimization and bubbly flows are motivating intense industrial as well as fundamental research and development. In this work, we consider only incompressible flows.

Besides the typical non-linearity inherent to any fluid dynamics phenomenon, two-phase flow problems are typically characterized by the existence of discontinuities in material properties across the interface as well as an arbitrary interface geometry along which interface-defined (singular) phenomena take place. This severely restricts the range of problems for which theoretical descriptions and analyses are available. Quite naturally, these difficulties have thus pushed and motivated the development of numerical simulation tools over the last three decades.

Mathematically, the existence of interface-defined discontinuities and singular source terms can be represented by means of Heaviside functions and Dirac distributions on the interface of co-dimension one. Building upon this mathematical equivalence, original steps toward the numerical simulation of two-phase flows considered numerical discretizations smearing Heaviside functions and Dirac distributions over a few grid cells. This enabled an (almost) immediate use of single-phase simulation techniques with unconventional source terms and material parameters.

The *Continuum Surface Force* (CSF) model introduced by Brackbill [1] is the first such numerical method, and proposed to capture surface tension effects by means of a numerical approximation smearing a singular (volumetric) force over the computational grid. Though the latter is usually referred to as the pioneering work, the same principles were actually used the front-tracking method introduced by Unverdi and Tryggvason [2] wherein the numerical discretization of interface singularities follows the work of Peskin [3, 4]. Over subsequent years, the simplicity of the approach contributed to its popularity and helped it make its way into other computational frameworks including levelset simulations [5, 6, 7], Volume-Of-Fluid (VOF) approaches [8, 9] and Discontinuous Galerkin finite element method [10].

However, spreading interface-defined source terms over a finite volume region does not correctly translate continuum mechanics in its limit sense since it produces a continuous solution. While there may exist a continuous variation of relevant quantities across the interface at the molecular level, such variations appear as sharp discontinuities at the continuum length scales. Regarding the smearing of material properties, similar conclusions hold and, from a numerical standpoint, such an approach produces artifacts and erroneous results that prevent convergence of numerical methods in infinity norm, e.g. strong parasitic currents in simulation of two-phase flows [11, 12]. Noticeably, VOF techniques have the advantage of minimizing the smearing effect to cells crossed by the interface only, by making mass density and viscosity linear function of the local volume fraction. This produces a pixelized representation of material properties which is not inconsistent for the mass density but lacks justification for viscosity. Nevertheless, it was shown in [13] that parasitic currents could be alleviated entirely when using a CSF model within a VOF approach, provided special care in the discretizations and accurate evaluations of interface curvature. We also note that another approach successfully alleviating parasitic currents was presented in [14] in two dimensions.

The first major contribution toward a numerical treatment of the problem that is fully aligned and consistent with continuum mechanics in its limit sense was introduced in [15]. Building upon the numerical methods developed in [16], inspired from the *Ghost Fluid Method* (GFM) [17], an entirely sharp treatment of viscous fluxes and surface tension was presented with a levelset representation of the interface. The approach was shown to significantly reduce parasitic currents compared to CSF models. This seminal work considered problems without phase change (continuous velocity field) and presented an explicit treatment of viscous terms as their sharp treatment close to the interface proved complex and challenging to make implicit. The success of this approach in capturing a sharp, discontinuous pressure field made it particularly appealing and the corresponding treatment of surface tension terms soon made its way into other works [18, 19], often at the sacrifice of treating viscous terms in a smeared fashion though.

In subsequent years, this approach gained significant interest and efforts were undertaken to treat viscous terms in an implicit (or semi-implicit) fashion. However, strict equivalence with the explicit treatment from [15] was never truly recovered. In [20, 21], an implicit scheme inspired from [15] was considered for computer graphics applications: the scheme borrows the sharp numerical representation of shear viscosity but assumes perfectly balanced viscous fluxes across the interface, i.e. $[\mu \nabla \boldsymbol{u}] = \underline{\boldsymbol{0}}$ where $[q]$ represents the discontinuity in q across the interface. A similar assumption is made in the major work of Sussman [22] wherein levelset and VOF methods are coupled to gain the best from either approach, building upon earlier developments [7, 23, 5, 24]. In that work, a semi-implicit discretization of viscous terms (coupling all velocity components) is presented with a sharp numerical representation of shear viscosity similar to [16, 15]. It is shown to be consistent with a former one-phase formulation [25] in presence of a dynamically negligible phase. Though the material viscosity is treated sharply, $[\mu\nabla\boldsymbol{u}] = \underline{\boldsymbol{0}}$ is assumed as well. In [26], clear answers were provided regarding the need to account or not for viscous terms in pressure discontinuities when considering either [15] or [22] and an alternative scheme to [22] that effectively decouples velocity components was presented, though assuming $[\mu\nabla\boldsymbol{u}] = \underline{\boldsymbol{0}}$ again. While this assumption is (artificially) consistent with the balance of tangential viscous stress across the interface in the absence of interface-defined forces, it is not an appropriate assumption as $[\mu\nabla\boldsymbol{u}] \neq \underline{\boldsymbol{0}}$ in general. As presented in [27], when considering interface-defined forces (e.g. Marangoni force) in such an approach, their straightforward consideration requires to assimilate $\left[\mu\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{\mathrm{T}}\right)\cdot\boldsymbol{n}\right]$ to $[\mu\nabla\boldsymbol{u}\cdot\boldsymbol{n}]$ where $\boldsymbol{n}$ is the unit vector normal to the interface, although the equivalence holds true only if $[\mu] = 0$.

Besides the conceptual drawbacks described here above, these numerical methods all rely on the discontinuity-capturing method from [16] to determine the pressure field. However, that method is known to lack convergence in gradients unless special care is taken (see [28], i.e., chapter 4 of this document). Therefore, some alternative efforts have been undertaken to apply more accurate numerical discretization techniques (usually less robust with respect to under-resolved interfaces, though) and/or to address the implicit treatment of sharp interface without assuming $[\mu\nabla\boldsymbol{u}] = \underline{\boldsymbol{0}}$. For instance, the virtual node strategy from [29] has been extended to two-phase Stokes flows in [30] and two-phase Navier-Stokes problems in [31]. In either case, a saddle-point discretization was used to account for all primary unknowns simultaneously, accounting for the coupling between (implicit) viscous and pressure terms across the interface within the discretization for instance. In [32], second-order accurate interface-capturing schemes have been used for all (decoupled) degrees of freedom on quadtree/octree grids in association with an iterative method updating interface jump conditions to account for stress balance across the interface. More recently, [33] introduced a saddle-point discretization building upon the numerical methods from [16] embedded within an iterative strategy similar to [28] (chapter 4 herein).

All the developments mentioned here above were restricted to problems with continuous velocity across the interface. When considering problems involving mass transfer between the immiscible phases (e.g. phase change), discontinuities are expected to take place not only in the pressure fields but also in velocity fields and this naturally adds more constraints and complexity to the implementation of sharp simulation tools. The development of numerical methods for handling this kind of problems followed a very similar trajectory. After early two-dimensional attempts involving front-tracking strategies [34, 35, 36], levelset and VOF approaches became more popular (see [37, 38, 39] for recent developments involving VOF approaches).

Originally, numerical smearing of material parameters and interface-defined source terms along with explicit treatment of viscous terms attracted researchers' interest [40, 41, 42]. In [43], an extension of the strategy from [15] for inviscid two-phase flows with phase change was presented: the absence of viscosity simplifies the problem significantly but this work paved the way toward sharp numerical treatments for phase-change problems. It was extended to viscous fluids in [44] with smeared viscosity coefficients and explicit viscous terms, in two dimensions.

In [45, 46, 47], this approach was used with a levelset method and sharp representation of viscosity was used along with the definition of ghost fluid velocities. A similar idea was introduced in [48, 49] where the velocity from either phase is extrapolated (and made divergence-free) for enabling straightforward, sharp (explicit) treatment of advection and viscous terms. This strategy was further developed in [50] in combination with the numerical methods from [26] to enable implicit time-stepping as required for simulating extremely thin films in Leidenfrost effects. A similar approach can be found in [51]. Although effectively alleviating smearing across the interface and providing an effective strategy for advection terms, such a strategy also prevent the (viscous) coupling of velocity components from either phase and relaxes the stress balance requirement across the interface.

In this work, we consider the development of numerical methods targeted to enable the sharp treatment of interface conditions for incompressible, viscous two-phase flows for problems that may involve mass transfer across the interface (i.e., phase change). Since we build upon the distributed computing, adaptive framework from [52] previously developed for single-phase problems, an adaptive- and parallel-friendly approach needs to be considered, motivating efforts to capture viscous effects implicitly while decoupling all primary unknowns. Whereas such a decoupling of unknowns does not pose major issues in single-phase problems, it does not comply well with the governing interface conditions for momentum balance across the interface in two-phase problems, underlining the need for iterative methods.

The rest of (the first part of) this document is structured as follows: in chapter 2, we present the foundational computational framework developed for the direct numerical simulation of incompressible single-phase flows on distributed, adaptive quadtree/octree grids (partial reproduction of [52]). In chapter 3, the extension of these numerical methods to two-phase flows are presented and discussed in order to underline the required developments as well as how to integrate them in a flowchart consistently with the relevant governing equations and interface conditions. This discussion highlights the need for the development of sharp numerical methods for vector fields capable of handling interface jump conditions that depend on the solution itself (unknown a priori). This challenging problem motivated a simplified approach for scalar fields which is presented in chapter 4 (published in [28]). Its extension to the relevant vector-field problem of interest is presented and discussed in chapter 5. In chapter 6, the development of a finite-volume numerical method for solving (cell-sampled) scalar elliptic interface problems is presented: this emerged as another requirement from the discussion in chapter 3 for the stability of projection steps. Finally, the assembly of all these computational tools as a two-phase flow simulation engine is presented in chapter 7, assessed on relevant test problems and illustrated with applications in two and three dimensions.

1.1 Permissions and Attributions

1. The content of chapter 2 is a partial reproduction of the result of a collaboration with Arthur Guittet, Fernando Temprano-Coleto, Tobin Isaac, François J. Peaudecerf, Julien R. Landel, Paolo Luzzatto-Fegiz, Carsten Burstedde and Frédéric Gibou, which has previously appeared in the Journal of Computational Physics, as "Direct numerical simulation of incompressible flows on parallel Octree grids" [52].

2. The content of chapter 4 is the result of a collaboration with Frédéric Gibou, and has previously appeared in the Journal of Computational Physics, as "xGFM: Recovering convergence of fluxes in the ghost fluid method" [28].

3. The content of chapter 8 is the result of a collaboration with Frédéric Gibou, and has previously appeared in the Journal of Computational Physics, as "Geometric discretization of the multidimensional Dirac delta distribution – Application to the Poisson equation with singular source terms" [53].

4. The content of chapter 9 is the result of a collaboration with Frédéric Gibou, and has previously appeared in the Journal of Computational Physics, as “Fast and scalable algorithms for constructing Solvent-Excluded Surfaces of large biomolecules” [54].

Chapter 2

Simulating single-phase flows on parallel Quadtree/Octree grids

2.1 Introduction

In the last decade, the democratization of the access to supercomputers has prompted the development of massively parallel simulation techniques. The previously existing serial codes are progressively being adapted to exploit the hundreds of thousands of cores available through the main computing clusters. We propose a parallel implementation of the solver for the incompressible Navier-Stokes equations introduced in [55], based on the parallel level-set framework presented in [56]. Additional novel algorithms, necessary to solving the Navier-Stokes equations in a forest of Quad-/Oc-trees, are presented.

Numerical simulations at the continuum scale are generally divided into two categories characterized by their meshing techniques. On the one hand, the finite elements community relies on body-fitted unstructured meshes to represent irregular domains. Given a high quality mesh, the resulting solvers are fast and very accurate. This approach has been successfully applied to the simulation of incompressible viscous flows [57, 58, 59].

However, the mesh generation is very costly and impractical when tracking moving interfaces and fluid features requiring high spatial resolution. On the other hand, methods based on structured Cartesian grids render the mesh geometry mainly trivial, but lead to a higher complexity for the implicit representation of irregular interfaces. We focus here on the latter class of methods.

A common approach to represent an irregular interface in a implicit framework is to use Peskin's immersed boundary method [60, 61, 62] or its level-set counterpart [5]. However, these methods introduce a smoothing of the interface through a delta formulation and therefore restrict the accuracy of the solution with $O(1)$ errors near fluid-fluid interfaces[1]. We therefore opt for the sharp interface representation provided by the level-set function [63]. We use the finite-volume/cut-cell approach of Ng *et al.* [64] to impose the boundary condition at the solid-fluid interface for its demonstrated convergence in the L^∞-norm.

Fluid flows are by nature multiscale, thus limiting the scope of uniform Cartesian grids. A range of strategies have been proposed to leverage the spatial locality of the fluid information such as stretched grids [65, 66], nested grids [67, 68, 7, 69, 70], chimera grids [71, 72] or unstructured meshes [73, 74, 75, 76, 77]. Another approach is to use a Quadtree [78] (in two spatial dimensions) or Octree [79] (in three spatial dimensions) data structure to store the mesh information [80, 81]. Popinet applied this idea combined with a non-compact finite volume discretization on the Marker-And-Cell (MAC) configuration [82] to the simulation of incompressible fluid flows [83]. Losasso *et al.* also proposed a compact finite volume solver on Octree for inviscid free surface flows [84], while Min *et al.* presented a node-based second-order accurate viscous solver [85]. The present work is based on the approach presented in Guittet *et al.* [55], which solves the viscous Navier-

[1]Although we do not consider fluid-fluid interfaces in this chapter, it stands as a stepping stone toward that case.

Stokes equations implicitly on the MAC configuration using a Voronoi partition and where the advection part of the momentum equation is discretized along the characteristic curves with a Backward Differentiation Formula (BDF), semi-Lagrangian scheme [86, 87]. The projection step is solved with the second-order discretization of Losasso *et al.* [88] for the Poisson equation.

The extension of [55] to parallel architectures relies on the existence of an efficient parallel Quad-/Oc-tree structure. Possible ways to implement parallel tree structures include the replication of the entire grid on each process. This approach, however, is not feasible when the grid size exceeds the memory of a single compute node, which must be considered a common scenario nowadays. Using graph partitioners such as `parMETIS` [89] on a tree structure would discard the mathematical relations between neighbor and child elements that are implicit in the tree, and thus result in additional overhead. Another option, which we find preferable, is to exploit the tree's logical structure using space-filling curves [90]. This approach has been shown to lead to load balanced configurations with good information locality for a selection of space-filling curves including the Morton (or Z-ordering) curve and the Hilbert curve [91].

Space-filling curves have been used in several ways, for example augmented by hashing [92], tailored to PDE solvers [93], or focusing on optimized traversals [94]. `Octor` [95] and `Dendro` [96] are two examples of parallel Octree libraries making use of this strategy that have been scaled to 62,000 [97] and 32,000 [98] cores, operating on parent-child pointers and a linearized octant storage, respectively. Extending the linearized storage strategy to a forest of interconnected Octrees [99, 100], the `p4est` library [101] provides a publicly available implementation of the parallel algorithms required to handle the parallel mesh, including an efficient 2:1 balancing algorithm [102]. `p4est` has been shown to scale up to over 458,000 cores [103], with applications using it successfully on 1.57M cores [104] and 3.14M cores [105].

The algorithms pertaining to the second-order accurate level-set method on Quad-/Oc-tree presented in Min and Gibou [106] have been extended in Mirzadeh *et al.* [56] parallel architecture by leveraging the `p4est` library. Starting from this existing basis for the level-set function procedures, we present the implementation of the algorithms pertaining to the simulation of incompressible fluid flows detailed in [55]. The Voronoi tessellation that we construct over the adaptive tree mesh requires (at least) two layers of ghost cells, whose efficient parallel construction we describe in detail. We report on the scalability of the algorithms presented before illustrating the full capabilities of the resulting solver.

2.2 The computational method

In this section, we present mathematical and computational components pertaining to our numerical method for solving the incompressible Navier-Stokes equations on a forest of Octree grids. The first five subsections are mainly dedicated to the mathematical description of the discretization procedures (the interested reader may find more details in [55]). The implementation of these building bricks in a distributed computing framework reveals two grid-related requirements: access to second-degree (or third-degree) cell neighbors and unambiguous indexing of grid faces. The last two subsections present the computational strategies developed to address challenges related to these requirements. Throughout this section, schematics and illustrations are presented in two dimensions for the sake of clarity. Their extension to three dimensions follows the exact same principles without any loss of generality.

2.2.1 Representation of the spatial information

The level-set method

A central desired feature of the proposed solver is to be able to handle complex, possibly moving interfaces in a sharp fashion[2]. The level-set framework, first introduced by [63] and extended to Quad-/Oc-trees in [106] is a highly suited tool for such a goal. The level-set representation of an arbitrary contour Γ, separating a domain Ω into two subdomains Ω^- and Ω^+, is achieved by defining a function ϕ, called the level-set function, such that $\Gamma = \{\boldsymbol{x} \in \mathbb{R}^n | \phi(\boldsymbol{x}) = 0\}$, $\Omega^- = \{\boldsymbol{x} \in \mathbb{R}^n | \phi(\boldsymbol{x}) < 0\}$ and $\Omega^+ = \{\boldsymbol{x} \in \mathbb{R}^n | \phi(\boldsymbol{x}) > 0\}$.

Among all the possible candidates that satisfy these criteria, a signed distance function (i.e., $|\nabla\phi| = 1$) is the most convenient one. In order to transform any function $\varphi(\boldsymbol{x})$ into a signed distance function $\phi(\boldsymbol{x})$ that shares the same zero contour, one can solve the reinitialization problem

$$\frac{\partial \phi}{\partial \tau} + \text{sign}(\varphi)\left(|\nabla\phi| - 1\right) = 0, \quad \phi(\boldsymbol{x})|_{\tau=0} = \varphi(\boldsymbol{x})$$

until a steady state in the fictitious time τ is found. The finite difference discretization and its corresponding parallel implementation employed to solve this equation are presented respectively in [106] and [56].

Forests of Quad-/Oc-trees and the `p4est` library

When dealing with physical problems that exhibit a wide range of length scales, uniform Cartesian meshes become impractical since capturing the smallest length scales requires a very high resolution. This is the case for high Reynolds number flows, for

[2]We consider irregular solid objects in this work but the methodology is intended to be extended to multiphase interfaces.

which the boundary layers and any wake vortices have a length scale significantly smaller than that of the far-field flow. This observation naturally leads to the use of adaptive Cartesian grids, including Octrees grids.

The `p4est` library [101] is a collection of parallel algorithms that handles a linearized tree data structure and its manipulation methods, which were shown to collectively scale up to 458,752 cores [103], as noted in the previous section. In `p4est` the domain is first divided by a coarse grid, which we will refer to as the "macromesh", common to all the processes. For our purpose we will consider solely uniform Cartesian macromeshes in a brick layout, although a general macromesh is not limited to such a configuration in `p4est`. This layout can be constructed at no cost using predefined and self-contained functions. A collection of trees rooted in each cell of the macromesh is then constructed and partitioned, and their associated (expanded) ghost layers are generated. The refinement and coarsening criteria necessary for the construction of the trees are provided to `p4est` by defining callback functions. We propose to use four criteria based on the physical characteristics at hand. Different combinations of these criteria are used depending on the specific problem considered.

The first criterion, presented in [106] and [56], captures the location of the interface: coarse cells are allowed locally, provided they are (at least) K cell diagonal(s) away from the interface, where $K \geq 1$ is defined by the user. Specifically, a cell $\mathcal{C}$ is marked for refinement if

$$\min_{v \in V(\mathcal{C})} |\phi(v)| \leq K \operatorname{Lip}(\phi) \operatorname{diag}(\mathcal{C}), \tag{2.1}$$

where $V(\mathcal{C})$ is the set of all the vertices of cell $\mathcal{C}$, $\operatorname{Lip}(\phi)$ is the Lipschitz constant of the level-set function ϕ, and $\operatorname{diag}(\mathcal{C})$ is the length of the diagonal of cell $\mathcal{C}$. Similarly, a cell is marked for coarsening if

$$\min_{v \in V(\mathcal{C})} |\phi(v)| > 2K \,\mathrm{Lip}(\phi)\mathrm{diag}(\mathcal{C}).$$

The second criterion, introduced for Quad-/Oc-trees in [83] and used in [85] and [55], is based on the vorticity of the fluid. High vorticity corresponds to small length scales and therefore necessitates a high mesh resolution. We mark a cell $\mathcal{C}$ for refinement if

$$h_{\max} \frac{\max_{v \in V(\mathcal{C})} \|\nabla \times \boldsymbol{u}(v)\|_2}{\max_{\Omega} \|\boldsymbol{u}\|_2} \geq \gamma, \tag{2.2}$$

where $h_{\max}$ is the largest edge length of cell $\mathcal{C}$ and γ is a parameter controlling the level of refinement. Analogously, a cell $\mathcal{C}$ is marked for coarsening if

$$2\, h_{\max} \frac{\max_{v \in V(\mathcal{C})} \|\nabla \times \boldsymbol{u}(v)\|_2}{\max_{\Omega} \|\boldsymbol{u}\|_2} < \gamma.$$

Another criterion enforces a band of b grid cells of highest desired resolution around the irregular interface. We mark for refinement every cell such that

$$\min_{v \in V(\mathcal{C})} \mathrm{dist}\,(v, \Gamma) < b \,\max\left(\Delta x_{\text{finest}}, \Delta y_{\text{finest}}, \Delta z_{\text{finest}}\right),$$

where Δx_{finest}, Δy_{finest} and Δz_{finest} are the cell sizes along cartesian directions for the finest cells to be found in the domain.

The Marker-And-Cell layout

The standard data layout used to simulate incompressible viscous flows on uniform grids is the Marker-And-Cell (MAC)[82] layout. The analogous layout for Quadtrees is presented in figure 2.1 and leads to complications in the discretizations compared to

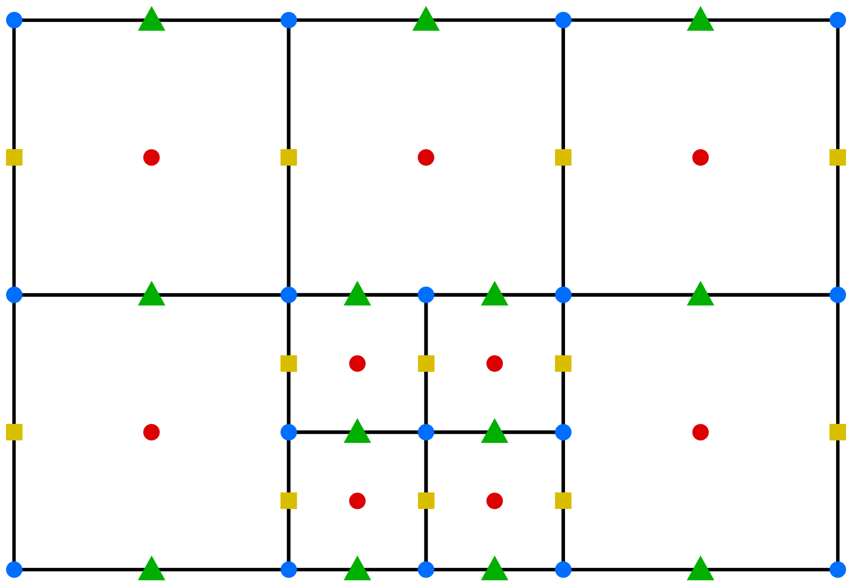


Figure 2.1: Representation of the Marker-And-Cell (MAC) data layout on a Quadtree structure with the location of the x-velocity (■), the y-velocity (▲), the Hodge variable (●) and the level-set values (●).

uniform grids. However, second order accuracy is achievable for the elliptic and advection-diffusion problems that appear in our discretization of the Navier-Stokes equations. Two possible corresponding discretizations are presented for the data located at the center of the cells (the leaves of the trees) and at their faces in sections 2.2.4 and 2.2.3 respectively.

2.2.2 The projection method

Consider the incompressible Navier-Stokes equations for a fluid with velocity $\boldsymbol{u}$, pressure p, density ρ and dynamic viscosity μ, with a force per unit mass $\boldsymbol{f}$

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \rho\boldsymbol{f} + \mu\nabla^2\boldsymbol{u}, \tag{2.3}$$

$$\nabla\cdot\boldsymbol{u} = 0. \tag{2.4}$$

The standard approach to solve this system is the projection method introduced by Chorin [107]. We refer the reader to [108] for a review of the variations of the projection method. The system is decomposed into two distinct steps, identified as the viscosity step and the projection step. The first step consists in solving the momentum equation (2.3) without the pressure term,

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = \rho\boldsymbol{f} + \mu\nabla^2\boldsymbol{u}, \tag{2.5}$$

to find an intermediate velocity field $\boldsymbol{u}^*$. Since this field does not satisfy the incompressibility condition (2.4), it is then projected on the divergence-free subspace to obtain $\boldsymbol{u}^{n+1}$, the solution at time t^{n+1}, via

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \nabla\Phi \tag{2.6}$$

where Φ is referred to as the Hodge variable and satisfies

$$\nabla^2\Phi = \nabla\cdot\boldsymbol{u}^*. \tag{2.7}$$

The two following sections describe the discretization applied to solve steps (2.5) and (2.7) respectively.

2.2.3 Implicit discretization of the viscosity step

The viscosity step (2.5) contains two distinct terms besides the possible bulk force: the advection term on the left-hand side and the viscous term on the right-hand side. In order to prevent stringent time step restrictions due to the latter, we opt for a second order backward differentiation method to advance (2.5) in time. This integration scheme can address stiff problems without theoretical stability-related constraints on the time step.

Discretization of the advection term with a semi-Lagrangian approach

We discretize the advection part of the viscosity step using a semi-Lagrangian approach [109, 86]. This method relies on the fact that the solution $\boldsymbol{u}\left(\boldsymbol{x}, t\right)$ of the advection equation

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = 0 \tag{2.8}$$

is constant along the characteristics of the equation, i.e., along material trajectories $(\boldsymbol{x}(s), t(s))$ such that $\dfrac{\mathrm{d}t}{\mathrm{d}s} = 1$ and $\dfrac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} = \boldsymbol{u}\left(\boldsymbol{x}, t\right)$. Using this parameterization, equation (2.8) is equivalent to

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}s} = 0,$$

which we integrate with respect to s using a second-order BDF [87].

Given the location $\boldsymbol{x}^*$ where the solution $\boldsymbol{u}^*$ is sought at time t_{n+1}, the local material trajectory passing through $\boldsymbol{x}^*$ at time t^{n+1} is traced back in time to find the points $\boldsymbol{x}_d^n$ and $\boldsymbol{x}_d^{n-1}$ through which it passed at times t^n and t^{n-1} respectively. The values $\boldsymbol{u}_d^n = \boldsymbol{u}\left(\boldsymbol{x}_d^n, t_n\right)$ and $\boldsymbol{u}_d^{n-1} = \boldsymbol{u}\left(\boldsymbol{x}_d^{n-1}, t_{n-1}\right)$ are then calculated using quadratic interpolation and the application of the second order BDF (note that $\Delta s = \Delta t$) leads to

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \approx \frac{\alpha}{\Delta t_n}\boldsymbol{u}^* + \left(\frac{\beta}{\Delta t_{n-1}} - \frac{\alpha}{\Delta t_n}\right)\boldsymbol{u}_d^n - \frac{\beta}{\Delta t_{n-1}}\boldsymbol{u}_d^{n-1},$$

where

$$\Delta t_j = t_{j+1} - t_j, \qquad \alpha = \frac{2\Delta t_n + \Delta t_{n-1}}{\Delta t_n + \Delta t_{n-1}} \qquad \text{and} \qquad \beta = -\frac{\Delta t_n}{\Delta t_n + \Delta t_{n-1}}.$$

We refer the reader to [55] for further details.

Discretization of face-centered Laplace operators

Since we consider constant-viscosity incompressible flows of Newtonian fluids, the velocity components are effectively decoupled in the viscous terms. This allows us to solve for the individual components of $\boldsymbol{u}^*$ separately when advancing (2.5). In that context, we require appropriate discretizations for the Laplace operators associated with degrees of freedom sampled at faces of similar orientations, i.e., at faces where similar velocity components are sampled. We obtain these discretized operators by applying a finite volume approach to Voronoi tessellations. We present a summary of the approach and refer the reader to [55] for further details.

Given a set of points in space, called seeds, we define the Voronoi cell of a seed as the region of space that is closer to that seed than to any other seed. The union of all the Voronoi cells forms a tessellation of the domain, i.e., a non-overlapping gap-free tiling of the domain. By placing these seeds at the centers of faces of the computational mesh sharing the same cartesian orientation, one obtains a new computational grid for the corresponding velocity component that is sampled at those faces. Two-dimensional examples of Voronoi tessellation are presented in figure 2.2 for Quadtree grids.

Considering a diffusion equation for the unknown u with constant diffusion coefficient μ

$$\mu \nabla^2 u = r,$$

it is discretized on the Voronoi tessellation with a finite volume approach where the control volume for each degree of freedom i is its Voronoi cell $\mathcal{C}_i$. This leads to

$$\int_{\mathcal{C}_i} \mu \nabla^2 u = \int_{\partial \mathcal{C}_i} \mu \; \nabla u \cdot \boldsymbol{n} \approx \sum_{j \in \text{ngbd}(i)} \mu \; s_{ij} \frac{u_j - u_i}{d_{ij}},$$

where ngbd(i) is the set of neighbors for the degree of freedom i, $\boldsymbol{n}$ is the vector normal to $\partial\mathcal{C}_i$ (pointing outwards), d_{ij} is the length between degrees of freedom i and j, and s_{ij} is the area —or the length, in 2D— of the face between them, as illustrated in figure 2.2. This discretization provides a second-order accurate solution [110].

General discretization for the viscosity step

Combining the discretizations presented in the two previous sections, we obtain the general discretization formula. Considering the x-component of the velocity field u, for the ith face with normal $\boldsymbol{e}_x$ and associated Voronoi cell $\mathcal{C}_i$, we have

$$\mathrm{Vol}(\mathcal{C}_i)\rho\frac{\alpha}{\Delta t_n}u_i^* + \mu\sum_{j\in\mathrm{ngbd}(i)} s_{ij}\frac{u_i^* - u_j^*}{d_{ij}} = \mathrm{Vol}(\mathcal{C}_i)\rho\left[\left(\frac{\alpha}{\Delta t_n} - \frac{\beta}{\Delta t_{n-1}}\right)u_{i,d}^n + \frac{\beta}{\Delta t_{n-1}}u_{i,d}^{n-1} + \boldsymbol{e}_x\cdot\boldsymbol{f}_i\right],$$

where Vol($\mathcal{C}_i$) is the volume of $\mathcal{C}_i$. The very same approach is then used for the y- and z-components of the velocity, i.e., v and w.

This produces a symmetric positive definite linear system that we solve using the BiConjugate Gradient stabilized iterative solver and the successive over-relaxation preconditioner provided by the PETSc library [111, 112, 113].

A note on the boundary conditions The boundary conditions to consider when solving the viscosity step are to be imposed on $\boldsymbol{u}^*$. As a consequence, the type of boundary condition that is desired for $\boldsymbol{u}$ is used for $\boldsymbol{u}^*$ as well, but the enforced boundary value is corrected in order to take into account the correction from the projection step (2.6), given the current best estimate of $\nabla\Phi$.

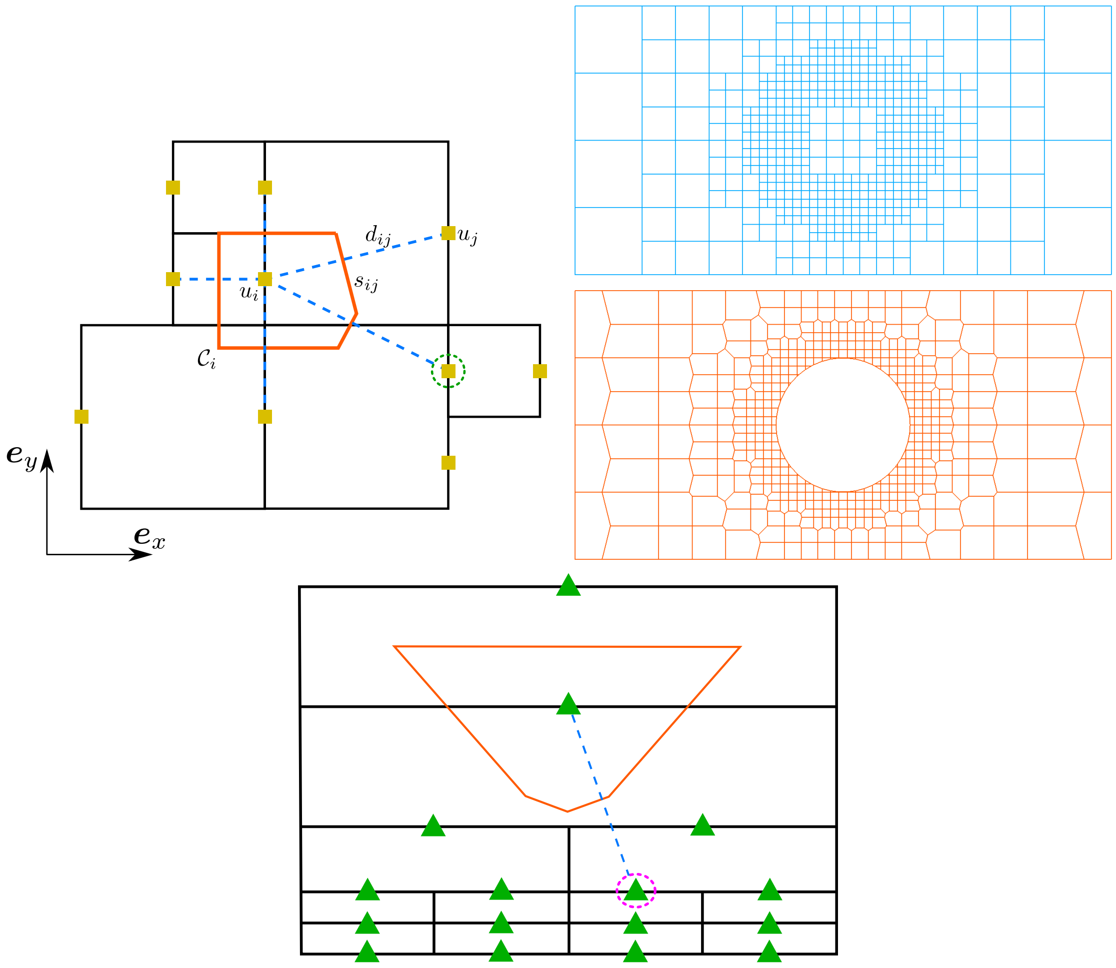


Figure 2.2: Top left: nomenclature for the discretization of the Laplace operator on a Voronoi diagram (illustrated for a vertical face). The degree of freedom circled in green can potentially belong to a second-degree neighbor cell, i.e., a cell that is not adjacent to the current cell but adjacent to one of the current cell's immediate neighbor. Top right: example of a Quadtree mesh (top) and its Voronoi tessellation for the vertical faces (bottom). Bottom: illustration of a two-dimensional Voronoi cell for a horizontal face which may require knowledge of a face associated with a third-degree neighbor cell, in case of stretched computational grids (aspect ratio much different from 1). The face circled in pink is indexed by a third-degree neighbor quadrant of the top quadrant indexing the center seed.

2.2.4 A stable projection

The projection step consists in solving the Poisson equation (2.7) with the data located at the center of the leaves of the tree. Stability and accuracy constraints result in the discretization presented in [88]. The method relies on a finite volume approach with a leaf being the control volume for the degree of freedom located at its center. Using the notations defined in figure 2.3, we now explain the discretization of the flux of the Hodge variable Φ on the right face of $\mathcal{C}_2$. For the sake of clarity, we assume that all other neighbor cells of $\mathcal{C}_2$ in Cartesian directions are of the same size as $\mathcal{C}_2$; if not, the reasoning presented here below needs to be applied for all variables sampled on faces shared between cells of different sizes.

The first step is to define the weighted average distance Δ between Φ_0 and its neighboring small leaves on the left side,

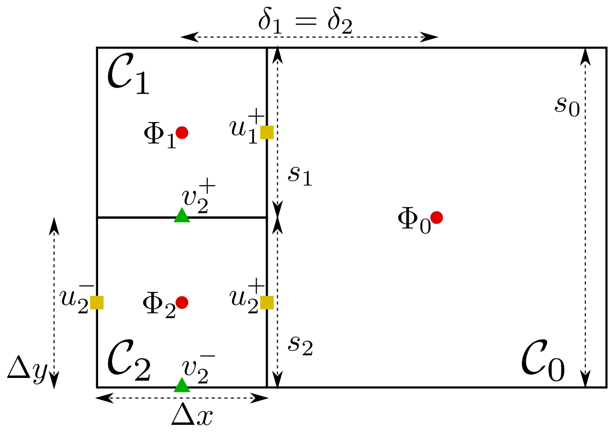


Figure 2.3: Nomenclature for the discretization of the flux of the Hodge variable at cell faces and the discretization of the divergence of the velocity field.

$$\Delta = \sum_{i \in \mathcal{N}} \frac{s_i}{s_0} \delta_i,$$

where $\mathcal{N}$ is the set of leaves whose right neighbor leaf is $\mathcal{C}_0$. We then define the partial derivative of Φ with respect to x on the right face of $\mathcal{C}_2$ as

$$\frac{\partial \Phi}{\partial x} = \sum_{i \in \mathcal{N}} \frac{s_i}{s_0} \frac{\Phi_0 - \Phi_i}{\Delta}.$$

This discretization collapses to the standard central finite difference discretization in case of (locally) uniform grids. The other components of $\nabla\Phi$ are defined analogously and stored at the corresponding faces. We then define the divergence of $\boldsymbol{u}$ at the center of the leaf containing Φ_2 as

$$\nabla \cdot \boldsymbol{u} = \frac{1}{\Delta x}\left(\sum_{i\in\mathcal{N}} \frac{s_i}{s_0} u_i^+ - u_2^-\right) + \frac{1}{\Delta y}\left(v_2^+ - v_2^-\right).$$

Both the divergence and the gradient operators involve all small leaves having $\mathcal{C}_0$ as a right neighbor. The cell-centered Laplace operator in eq. (2.7) is obtained by chaining the above divergence and gradient operator, i.e., $\nabla^2\Phi = \nabla \cdot (\nabla\Phi)$. This produces a second-order accurate discretization for cell-centered Poisson equations. The correspondence between the two operators defined here above ensures that the gradient is the negative adjoint of the divergence in a well-defined face-weighted norm, ensuring the stability of the projection step [55]. The linear system resulting from this approach is symmetric positive definite, and it is solved using a (possibly preconditioned) conjugate gradient method.

2.2.5 Typical flowchart of the solver

As detailed in subsection 2.2.3, boundary conditions to be enforced on the intermediate velocity field $\boldsymbol{u}^*$ require the knowledge of $\nabla\Phi$. Yet, Φ itself is defined as the solution of an elliptic problem that requires $\nabla \cdot \boldsymbol{u}^*$ (see (2.7)).

In order to best enforce the desired boundary conditions on $\boldsymbol{u}$, the solver addresses this circular dependency between $\boldsymbol{u}^*$ and $\nabla\Phi$ through a fixed-point iteration. The solver determines a sequence $\boldsymbol{u}^{*,k}$ and Φ^k, with $k \geq 1$: the intermediate velocity field $\boldsymbol{u}^{*,k}$ is the solution of the viscosity step (2.5) with boundary condition values defined using the known field Φ^{k-1} (see subsection 2.2.3 - note that Φ^0 is defined as the scalar field Φ

obtained at the end of the previous time step, or as 0 for the very first time step). The scalar field Φ^k is determined in turn as the solution of the projection equation (2.7) using $\nabla \cdot \boldsymbol{u}^{*,k}$ as the right-hand side.

This process is repeated for increasing k until convergence is reached or until a user-defined maximum number of iterations $k_{\max}$ is reached. Note that the standard, approximate projection method corresponds to $k_{\max} = 1$ (the computational cost and the relevance of additional inner-loop iterations are estimated and discussed for some relevant applications within section 2.4). Two different convergence criteria may be used: the user may choose to enforce

- either $\left\| \Phi^k - \Phi^{k-1} \right\|_\infty < \varepsilon_\Phi$, where ε_Φ is a user-defined threshold (most relevant if the pressure is a primary variable of interest and is well-defined everywhere);
- or $\left\| \dfrac{\partial \Phi^k}{\partial \zeta} - \dfrac{\partial \Phi^{k-1}}{\partial \zeta} \right\|_\infty < \varepsilon_{\nabla\Phi}$, where $\varepsilon_{\nabla\Phi}$ is a user-defined threshold and ζ is any (or all) of x, y, z (most relevant to ensure strict wall and/or interface no-slip boundary boundary conditions).

The structure and internal logic of the solver is designed so as to minimize the cost of such extra iterations when $1 < k \leq k_{\max}$: relevant computation-intensive data pertaining to the construction of the discretized linear systems is kept in memory (to avoid re-computing), as well as discretization matrices, possible preconditioners, etc.

2.2.6 Expansion of the ghost layer

Several building bricks of the solver require second-degree (or even third-degree) neighbor cells to ensure robust behavior and properly defined operators. For instance, the construction of Voronoi cells based on face-collocated seeds requires to connect neighboring face-sampled degrees of freedom. As illustrated in figure 2.2, such neighboring

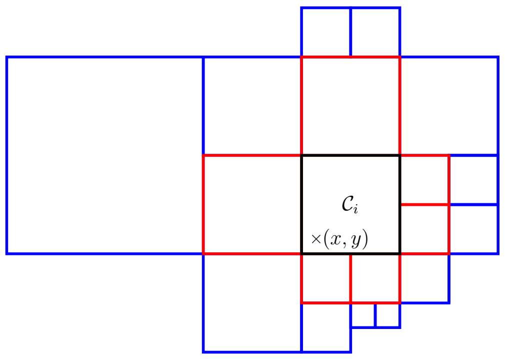


Figure 2.4: The stencil used to interpolate the velocity at (x, y) in cell $\mathcal{C}_i$ does not only require the data in $\text{ngbd}(\mathcal{C}_i)$ (red), but also in $\text{ngbd}^2(\mathcal{C}_i)$ (blue), a set of cells including second-degree (indirect) neighbors.

seeds may lie on a face that is shared between a (large) first-degree neighbor cell and a (small) second-degree neighbor cell. In such a case, only the (small) second-degree neighbor cell indexes the queried face. Therefore, second-degree neighbors need to be accessible from every locally owned face degree of freedom. Besides, when using stretched grids, more remote neighbors may be involved in the construction of a local Voronoi cell (see figure 2.2). Similarly, the cell-centered operators defined in section 2.2.4 require second-degree neighbors in case of non-graded grids. As depicted in figure 2.4, the ability to access second-degree neighbor cells is also desirable regarding the accuracy and the inter-processor smoothness of the moving least-square interpolation procedure used to define the node-sampled velocity fields based on the face-sampled components [55]. The ability to construct deep ghost layers is a recent extension to the `p4est` interface, which we briefly describe here.

The algorithm used by `p4est` to construct a single layer of ghosts ([101, Algorithm 19]) is able to maximize the overlap of computation and communication because each process can determine for itself which other processes are adjacent to it. This is because the "shape" of each process's subdomain (determined by the interval of the space-filling curve assigned to it) is known to every other process. As a consequence, the communication

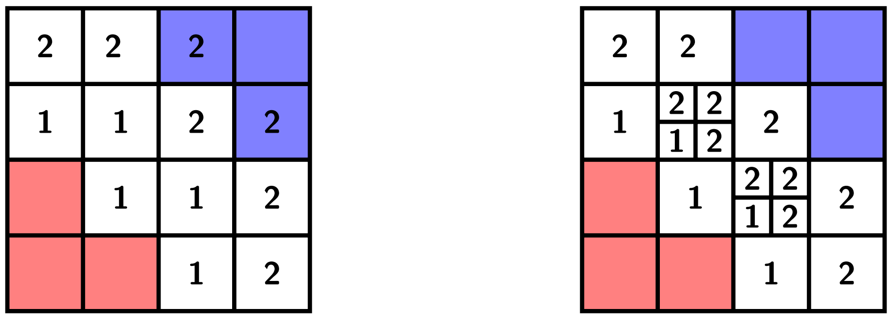


Figure 2.5: Two meshes with the same partition shapes, but with different two-deep ghost layers. For each mesh we show the first and second layers of the ghost layer of process p (red). In the first mesh, the second layer includes cells from process q (blue), but in the second it does not.

pattern is symmetric and no sender-receiver handshake is required.

As a first extension, when creating the send buffers we remember their entries, since they identify the subset of local cells that are ghosts to one or more remote processes. We store these pre-image cells or "mirrors" in ascending order with respect to the space filling curve, and create one separate index list per remote processor into this array. This data is accommodated inside the ghost layer data structure and proves useful for many purposes, the most common being the local processor needing to iterate through the pre-image to define and fill send buffers with application-dependent numerical data.

The communication pattern of a deeper ghost layer, on the other hand, depends not just on the shapes of the subdomains, but the leaves within them, as illustrated in figure 2.5. Rather than complicating the existing ghost layer construction algorithm to accommodate deep ghost layers, a function that adds an additional layer to an existing ghost layer has been added to `p4est`. This function is called `p4est_ghost_expand`() and adds to both the ghosts and the pre-images. Thus, as a second extension to the data structure, we also identify those local leaves that are on the inward-facing front of each preimage, in other words the most recently added mirrors. This is illustrated in figure 2.6.

When process p expands its portion of process q, it loops over the leaves in the front

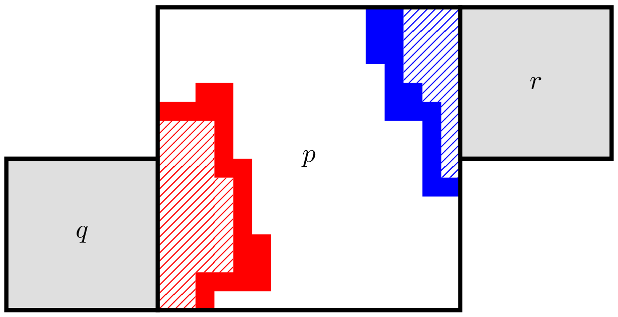


Figure 2.6: We show the preimages of process q's and process r's ghost layers in the leaves of process p. The solid red area represent the cells at the "front" of the preimage for q (`preimage_front`[q] in algorithm 1), while the solid and dashed together form the whole preimage (`preimage`[q]).

of the pre-image for process q and adds any neighbors that are not already in the ghost layer. Sometimes this will include a leaf from a third process r: process p will also send such leaves to process q, because it may be that r is not yet represented in q's ghost layer, and so communication between q and r is not yet expected. The basic structure of this algorithm is outlined in algorithm 1.

2.2.7 Indexing the faces

Although the `p4est` library provides a global numbering for the faces of the leaves, its numbering differs from our needs because it does not number the small faces on a coarse-fine interface, where we have degrees of freedom in our MAC scheme. Therefore, we implement a procedure to distribute the faces of the leaves across the processes and to generate a unique global index for each face. Since some faces are shared between two processes, we chose to attribute a shared face to the process with the smaller index. With this rule, each face belongs to a unique process and after broadcasting the local number of faces a global index can be generated for all the local faces. The second step is to update the remote index of the faces located in the ghost layer so that their global index can be constructed easily by simply adding the offset of the process each face belongs to.

Algorithm 1 Process p's algorithm for expanding other processes' ghost layers, and receiving expansions to its own ghost layer. Note that finding a neighbor of a leaf l entails a fixed number of binary searches through the `local_leaves`, which are sorted by the space-filling curve induced total ordering.

```
 1: for $q \in$ ghost_neighbors do                ▷ processes that contribute to ghost layer
 2:     initialize empty sets send_forward[$q$], send_back[$q$], and new_front[$q$]
 3: end for
 4: for $q \in$ ghost_neighbors do
 5:     for $l \in$ preimage_front[$q$] do
 6:         for each neighbor $n$ of $l$ in local_leaves do          ▷ $n$ found by search
 7:             if $n \notin$ preimage[$q$] then
 8:                 add $n$ to send_forward[$q$], preimage[$q$], and new_front[$q$]
 9:             end if
10:         end for
11:         for each neighbor $n$ of $l$ in ghost_layer do           ▷ $n$ found by search
12:             if $n$ belongs to process $r \neq q$ then
13:                 add $n$ to send_forward[$q$] and $(n, q)$ to send_back[$r$]
14:             end if
15:         end for
16:     end for
17:     replace preimage_front[$q$] with new_front[$q$]
18: end for
19: for $q \in$ ghost_neighbors do
20:     send send_forward[$q$] and send_back[$q$] and receive recv_forward[$q$] and
    recv_back[$q$]
21:     add all of recv_forward[$q$] to ghost_layer
22:     for $(l, r) \in$ recv_back[$q$] do
23:         if $r \notin$ ghost_neighbors or $l \notin$ preimage[$r$] then
24:             add $l$ to preimage[$r$] and preimage_front[r]          ▷ new lists if
    $r \notin$ ghost_neighbors
25:         end if
26:     end for
27: end for
28: recompute ghost_neighbors from leaves in ghost_layer
```

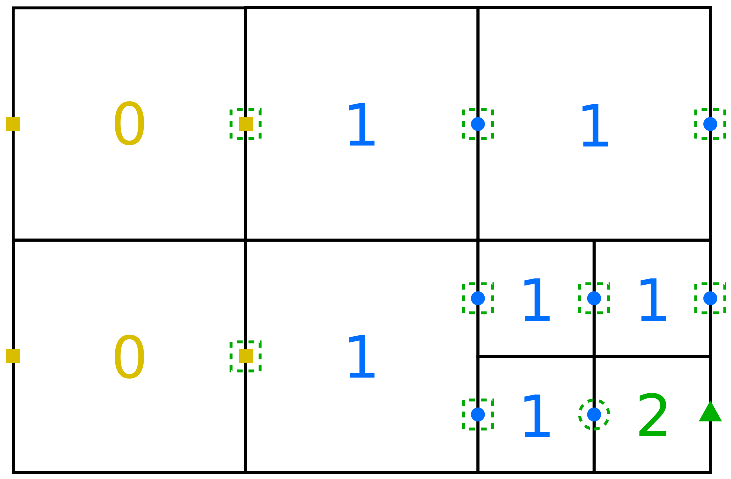


Figure 2.7: Illustration of the ghost layer of x-faces of depth 2 and of the global indexing procedure for process 2. The numbers in the leaves correspond to the indices of the processes owning them. After the first step, the remote index for the circled face is known to process 2, and after the second step the remote indices for the faces in a square are known to process 2. Note that a single step would not be sufficient for process 2 to gain knowledge of the remote index of the two faces belonging to process 0.

We do so in two steps, represented in figure 2.7. First, the indices of the ghost faces of the local leaves are synchronized, then the indices of the faces of the ghost layer of leaves are updated. This has some similarities to the two-pass node numbering from [114], here extended to two layers of ghosts. Algorithm 2 details the steps of our implementation and makes use of the `Notify` collective algorithm described in [102] to reverse the asymmetric communication pattern.

2.3 Scalability

In this section, we present an analysis of the scaling performance of our implementation. We define the parallel efficiency as $e = s\,(P_0/P)^\sigma$ where $s = t_0/t_p$ is the speed-up, σ is the optimal parallel scaling coefficient ($\sigma = 1$ for linear scaling), P_0 is the smallest considered number of processes with its associated runtime t_0 and P is the number of

Algorithm 2 Communication algorithm to generate a global indexing of the faces. The `Notify` collective algorithm is used to reverse the communication pattern, described in more detail in [102].

```
1: for l ∈ (local|ghost) leaves do
2:     for f ∈ remote_faces(l) do
3:         add proc(f) to receivers
4:         add f to buffer[proc(f)]
5:     end for
6: end for
7: Notify(receivers,senders)                    ▷ reverse communication pattern
8: for p ∈ receivers do                         ▷ send requests
9:     MPI_Isend(buffer[p])                     ▷ send request to process p
10: end for
11: for p ∈ senders do                          ▷ process remote requests
12:     MPI_Recv(req)                           ▷ receive request from process p
13:     assemble answer with local indices requested
14:     MPI_Isend(ans)                          ▷ send answer to process p
15: end for
16: for p ∈ receivers do                        ▷ process answers
17:     MPI_Recv(p)                             ▷ receive answer from process p
18:     update faces information
19: end for
```

processes with its associated runtime t_p. All the results were obtained on the “Knights Landing” Intel Xeon Phi 7250 (KNL) compute nodes of the Stampede2 supercomputer at the Texas Advanced Computing Center (TACC), at The University of Texas at Austin, and on the Comet supercomputer at the San Diego Supercomputer Center, at the University of California at San Diego. Those resources are available through the Extreme Science and Engineering Discovery Environment (XSEDE) [115]. The strong scaling performance was analyzed up to 32,768 cores on Stampede2.

2.3.1 Expansion of the ghost layer

We present both weak and strong scaling results for the algorithm used to expand the ghost layer of cells for each process in figure 2.8. The associated efficiency is presented in table 2.1. The strong scaling consists in choosing a problem and solving it with increasing number of processes. Ideally, for an algorithm with a workload increasing linearly with the problem size (i.e., with parallel scaling coefficient $\sigma = 1$), doubling the amount of resources spent on solving a problem should half the runtime. However, in the case of the ghost layer expansion, the amount of work depends on the size of the ghost layers, as explained in [103]. For a well behaved partition, we expect $O(N^{\frac{(d-1)}{d}})$ of the leaves to be in the ghost layer, where d is the number of spatial dimensions. We therefore consider a parallel scaling coefficient $\sigma = 2/3$ to be optimal for a three dimensional problem, i.e., $O((N/P)^{2/3})$ is the ideal scaling, with P the number of processes and N the problem size. The results presented in figure 2.8 were obtained on Stampede2 for a mesh of level 9/13, corresponding to 588,548,472 leaves, and on Comet for a mesh of level 10/13, corresponding to 1,595,058,088 leaves. The computed parallel efficiency between the smallest and the largest run is 66% for Stampede2 and 59% for Comet.

The idea behind the weak scaling is to keep the problem size constant for each process while increasing the number of processes. The right graph of figure 2.8 presents the results obtained on Stampede2 for two problems of sizes 30,248 leaves per process and 473,768 leaves per process, and for a number of processes ranging from 27 to 4,096. The runtime increases by 16% between the smallest and the largest run for the small problem and by 6% for the large problem.

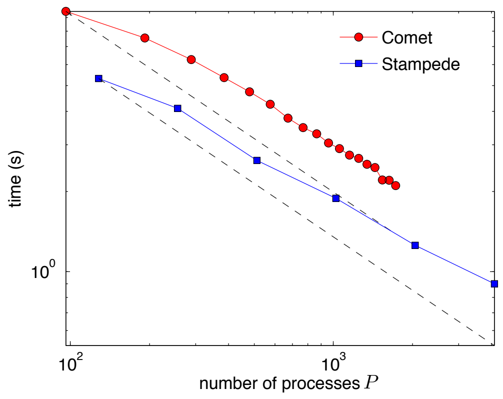
Stampede2

Number of processes P	128	256	512	1024	2048	4096
Efficiency e	100%	79%	70%	69%	66%	66%

Comet

Number of processes P	96	192	384	672	1152	1728
Efficiency e	100%	82%	81%	71%	67%	59%

Table 2.1: Efficiency of the procedure for expanding the ghost layer of leaves.



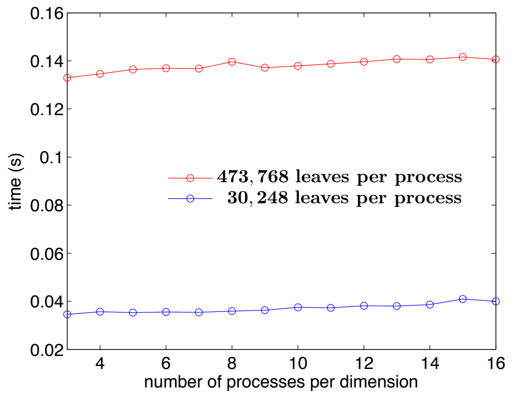


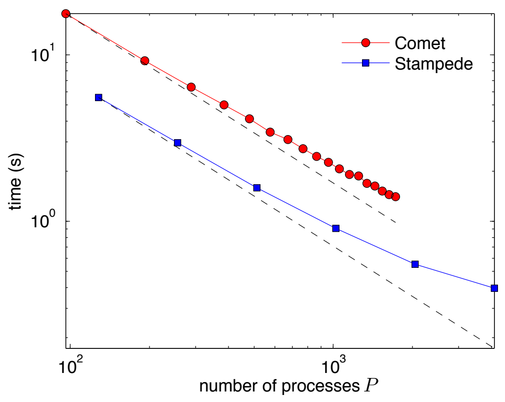
Figure 2.8: Scaling results for the expansion of the layer of ghost cells (see section 2.3.1). The strong scaling results are presented in the left figure together with the optimal reference scaling for a parallel scaling coefficient $\sigma = 2/3$ (dashed lines) while the weak scaling results are shown on the right figure. The increases in runtime observed for the weak scaling are of 16% for the small problem and 6% for the large problem.

2.3.2 Indexing the faces

The scaling procedure presented in the previous section is repeated for Algorithm 2 and the results are presented in figure 2.9. Even though the workload for this procedure increases slightly as the number of processes increases and the number of leaves in the ghost layers increases, we compare our results to an ideal linear scaling $\sigma = 1$. The corresponding efficiency is computed in table 2.2. The parallel efficiency e computed between the smallest and the largest run from the strong scaling results is 44% for Stampede2 and 70% for Comet. The weak scaling results show an increase in runtime of 71% for the small problem and of 14% for the large problem.

2.3.3 Scalability of the full solver

We now analyze the scaling performance of the full incompressible Navier-Stokes solver, breaking its execution time down into its four main fundamental components:



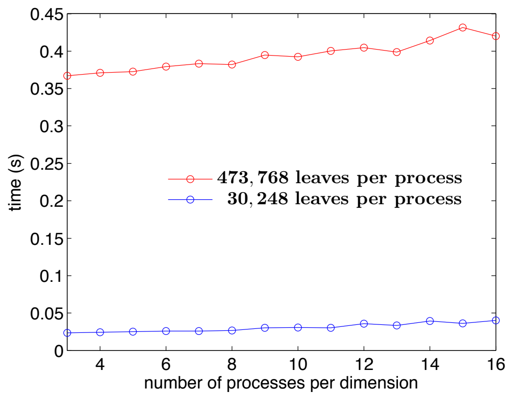


Figure 2.9: Scaling results for the indexing of the faces with Algorithm 2. The strong scaling results are presented in the left figure together with the reference ideal linear scaling (dash lines) while the weak scaling results are shown on the right figure. The strong scaling problem shown for Comet is three times larger than the one for Stampede2. The increases in runtime observed for the weak scaling are of 71% for the small problem and 14% for the large problem.

Stampede2

Number of processes P	128	256	512	1024	2048	4096
Efficiency e	100%	94%	87%	76%	63%	44%

Comet

Number of processes P	96	192	384	672	1152	1728
Efficiency e	100%	96%	88%	82%	77%	70%

Table 2.2: Efficiency of Algorithm 2 producing a global index for the faces.

the viscosity step (see subsection 2.2.3), the projection step (see subsection 2.2.4), the moving least-square interpolation of the velocity components from cell faces to the grid nodes and the re-meshing step (denoted as grid update). We intend to show satisfactory strong scaling on large numbers of processors, so this scaling analysis was conducted on Stampede2 only since we do not have access to the same resources on other supercomputers.

For this purpose, the solver is restarted from a physically relevant and computationally challenging simulation state, defined as the inception of vortex shedding for the flow past a sphere at $Re = 500$, as illustrated in figure 2.10. A macromesh of size $8 \times 4 \times 4$ is used with two different refinement criteria. In the first case, the Octrees are refined with a minimum level 6 and a maximum level 11 with a vorticity threshold $\gamma = 0.02$ (see (2.2)), leading to a total of about 270×10^6 grid computational cells. In the second case, the Octrees are refined with a minimum level 7 and a maximum level 11 with a vorticity threshold $\gamma = 0.015$, leading to a total of about 610×10^6 grid cells. The grids for the initial states of the two scenarios are illustrated in figure 2.11. The three successive linear systems of the viscosity steps are solved using a BiConjugate Gradient Stabilized solver, while a Conjugate Gradient solver is used for the (symmetric positive definite) projection step.

In the first case, the wall-clock execution time is measured and averaged over 10 full time steps, while only 5 time steps are considered for the second larger case (to limit the cost of these runs). A minimum of 64 (resp. 90) KNL nodes were required for the problem to fit in memory in the first (resp. second) case. Therefore, the first two data points in the left (resp. right) graph from figure 2.12 used less than 68 cores per node (maximum available). In either case, the solver performs two subiterations of the inner loop per time step: for each time step, $\left\|\Phi^k - \Phi^{k-1}\right\|_\infty$ drops by 4 orders of magnitude between $k = 1$ and $k = 2$.

The results are presented in figure 2.12 and table 2.3. As expected, the projection step is the most challenging part, thus determining the strong scalability limits of the solver: no significant speed-up is observed when the number of cells per process falls under $10,000$[3]. In comparison, the scaling behavior of the grid update procedure is not impeded yet around that limit, which shows the extremely good performance of all `p4est`'s grid management operations that are in play: grid refinement and/or grid coarsening, grid

[3]This is consistent with PETSc scaling performance reported in their documentation.

partitioning, ghost layer creation and ghost layer expansion.

Given that the global solver makes use of various separate routines having different (theoretical) ideal scaling coefficients σ, it is expected that the strong scaling performance of the solver is less than ideal. In fact, some of the operations at play cannot even be attributed such a theoretical scaling coefficients: when considering a very large number of processes, several of them will be associated with regions of the computational domain that are (very) far away from the interface and will end up stalling during the geometric extrapolation tasks of primary face- and cell-sampled fields, for instance. Though these extrapolations represent a small portion of the overall workload when using a small number of processes, they do contribute to less than ideal scaling on large numbers of processes in such an application. Therefore, regarding several aspects, this analysis may be considered a "worst case" scenario, which aims to produce insightful information when it comes to estimating a lower bound for the (effective) scaling coefficient σ to consider when estimating the computational cost of future large-scale simulations and/or when assessing the limits of accessible simulations for the solver. As illustrated in figure 2.12, the solver's scaling behavior seems to follow an asymptotic law of $\sigma \simeq 0.78$, in such a symptomatic case; Table 2.3 also indicates an efficiency above 80% (in the relevant range of P) when considering $\sigma \simeq 0.85$ (almost all calculated efficiencies are 100% or higher when considering $\sigma = 0.78$).

2.4 Numerical validation and illustrations

In this section, we present a series of numerical examples to validate the implementation as well as to demonstrate the potential of the approach.

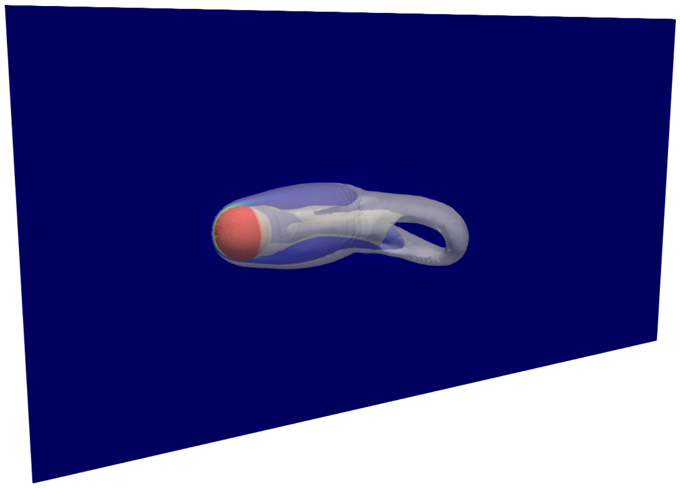


Figure 2.10: Physically relevant initial state considered for the scaling analysis on large number of cores. This figure illustrates the inception of vortex shedding for the flow past a sphere at $Re = 500$ (the full description of the computational set-up can be found in section 2.4.2). The vertical slice has equation $z = -0.5$ and it is colored by vorticity. The static sphere is colored in red and the translucent white surface represents the isocontour of vorticity $\|\nabla \times \boldsymbol{u}\| = 0.6\, u_0/r$.

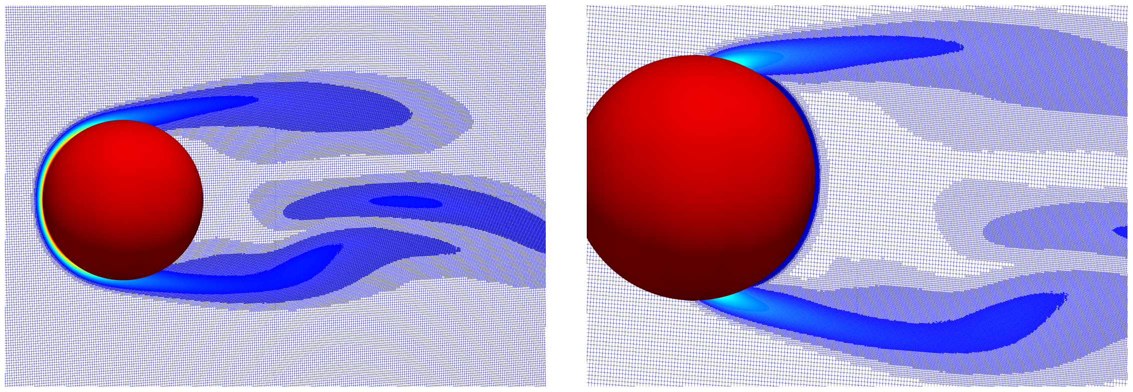
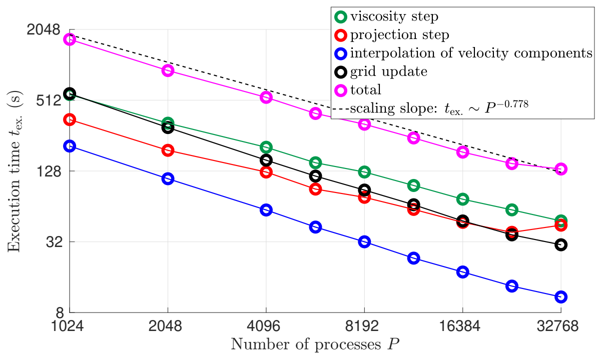


Figure 2.11: Grid illustrations for the scaling analyses from section 2.3.3. A grid slice in the computational domain is illustrated and its edges are colored by vorticity intensity. The Octrees are refined with a minimum level 7 and a maximum level 11 with a vorticity threshold $\gamma = 0.015$ (total of about 610×10^6 grid cells).



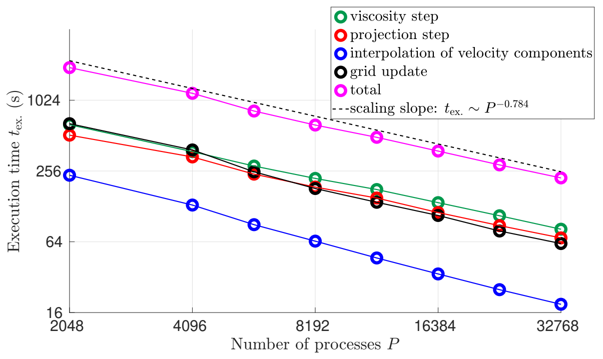


Figure 2.12: Scaling results on large number of cores on Stampede2 for the two grids considered. Left: scaling results for Octrees refined with a minimum level 6 and a maximum level 11 with a vorticity threshold $\gamma = 0.02$ (total of about 270×10^6 grid cells). Right: scaling results for Octrees refined with a minimum level 7 and a maximum level 11 with a vorticity threshold $\gamma = 0.015$ (total of about 610×10^6 grid cells).

Stampede2 (270×10^6 grid cells)

# of processes P	1,024	2,048	4,096	5,800	8,192	11,590	16,384	23,170	32,768
e ($\sigma = 1$)	100%	92.1%	77.9%	75.4%	65.9%	60.9%	56.9%	50.3%	39.6%
e ($\sigma = 0.85$)	100%	102.2%	95.8%	97.7%	90.0%	87.7%	86.2%	80.36%	66.6%

Stampede2 (610×10^6 grid cells)

# of processes P	2,048	4,096	5,900	8,192	11,590	16,384	23,170	32,768
e ($\sigma = 1$)	100%	82.7 %	82.7 %	77 %	69.2 %	64.1 %	59.4 %	54.2 %
e ($\sigma = 0.85$)	100%	91.8 %	96.6 %	94.8 %	89.8 %	87.6 %	85.5 %	82.2 %

Table 2.3: Efficiencies e of the full solver proposed for the incompressible Navier-Stokes equations on Stampede2, when considering an ideal linear scaling (i.e., $\sigma = 1$) or a scaling coefficient of 85%.

2.4.1 Validation with an analytical solution

The first application aims at validating the implementation by monitoring the convergence of the solver using the analytical solution presented in [64]. Consider the irregular domain $\Omega = \{(x, y, z)| -\cos(x)\cos(y)\cos(z) \geq 0.4 \text{ and } \frac{\pi}{2} \leq x, y, z \leq \frac{3\pi}{2}\}$ and the exact solution

$$\begin{aligned} u(x,y,z) &= \cos(x)\sin(y)\sin(z)\cos(t), \\ v(x,y,z) &= \sin(x)\cos(y)\sin(z)\cos(t), \\ w(x,y,z) &= -2\sin(x)\sin(y)\cos(z)\cos(t), \\ p(x,y,z) &= 0. \end{aligned}$$

The exact velocity is prescribed at the domain's boundary and homogeneous Neumann boundary conditions are enforced on the Hodge variable. The corresponding forcing term is applied to the viscosity step. We take a final time of $\frac{\pi}{3}$ and monitor the error on the velocity field and on the Hodge variable as the mesh resolution increases. The computational grid is *not* dynamically adapted for this accuracy analysis, so the grid-parameter γ (see (2.2)) is irrelevant in this case. The first computational grid is built to satisfy the distance-based criterion from section 2.2.1 using $\phi(x,y,z) = \cos(x)\cos(y)\cos(z) + .4$, $K = 1.2$ and $b = 5$. The successive resolutions are then obtained by splitting every cell from the previous resolution. The results are presented in table 2.4 and indicate first-order accuracy for the velocity field and second order accuracy for the Hodge variable in the L^∞ norm.

	u, v		w		Hodge variable	
level (min/max)	L^∞ error	order	L^∞ error	order	L^∞ error	order
4/6	$4.65 \cdot 10^{-3}$	-	$3.50 \cdot 10^{-3}$	-	$8.96 \cdot 10^{-4}$	-
5/7	$3.27 \cdot 10^{-3}$	0.50	$2.11 \cdot 10^{-3}$	0.73	$2.85 \cdot 10^{-4}$	1.65
6/8	$1.67 \cdot 10^{-3}$	0.97	$1.16 \cdot 10^{-3}$	0.86	$8.07 \cdot 10^{-5}$	1.82
7/9	$8.42 \cdot 10^{-4}$	0.99	$5.52 \cdot 10^{-4}$	1.07	$2.27 \cdot 10^{-5}$	1.83

Table 2.4: Convergence of the solver for the analytical solution presented in section 2.4.1. First-order accuracy is observed for the velocity field and second order accuracy for the Hodge variable.

2.4.2 Vortex shedding of a flow past a sphere

In order to further assess the performance of the solver, we also address the flow past a sphere. Related properties like drag and lift forces as well as vortex shedding frequency (if applicable) are calculated and compared to available data for this canonical problem.

We consider a static sphere of radius $r = 1$, located at $(8, 0, 0)$ in the domain $\Omega = [0, 32] \times [-8, 8] \times [-8, 8]$. An inflow velocity $\boldsymbol{u}_0 = u_0 \boldsymbol{e}_x$ is imposed on the $x = 0$ face of the domain as well as on the the side walls, homogeneous Neumann boundary conditions are imposed on the velocity field at the outlet $x = 32$ and no-slip conditions are imposed on the sphere. The pressure is set to zero at the outlet and is subject to homogeneous Neumann boundary conditions on the other walls as well as on the sphere. We set $u_0 = 1$, the density of the fluid to $\rho = 1$ and vary the viscosity μ to match the desired Reynolds number $Re = \dfrac{2\,\rho u_0 r}{\mu}$, set by the user. Eight Reynolds numbers ranging from 50 to 500 are considered.

The Octree mesh is refined around the sphere and according to the vorticity criterion (2.2) of section 2.2.1: we use $\phi\left(\boldsymbol{x}\right) = r - \sqrt{\left(x-8\right)^2 + y^2 + z^2}$, $K = 1.2$, $b = 16$, with a vorticity-based threshold of $\gamma = 0.01$. All the results were obtained with a macromesh $8 \times 4 \times 4$ and with trees of levels $4/7$, leading to approximately 6 million leaves and corresponding to an equivalent uniform grid resolution of 268,435,456 cells. The time step Δt is set such that $\dfrac{\max_\Omega \|\boldsymbol{u}\|\, \Delta t}{\Delta x_{\min}} \leq 1$ at all times. Figure 2.13 shows a snapshot of the unsteady flow for $Re = 300$ at $t = 221\, r/u_0$.

The nondimensional force $\boldsymbol{F}$ applied onto the static sphere is monitored over time. The force is evaluated by geometric integration [116] of the stress vector (including viscous and pressure contributions) on the surface of the sphere Γ, i.e.,

$$\boldsymbol{F} = \frac{1}{\frac{1}{2}\rho u_0^2 \pi r^2} \int_\Gamma \left(-p\underline{\underline{I}} + 2\mu \underline{\underline{D}}\right) \cdot \boldsymbol{n}\, \mathrm{d}\Gamma, \tag{2.9}$$

where $\underline{\underline{I}}$ is the identity tensor, $\underline{\underline{D}}$ is the symmetric strain-rate tensor and $\boldsymbol{n}$ is the outward normal to the sphere. Figure 2.15 shows the evolution of the streamwise force component. Figure 2.14 shows the evolution of the pressure and azimuthal vorticity on the sphere surface, at steady state for $Re = 100$. Figure 2.16 illustrates the transverse force components with respect to time for $Re \geq 250$. Time-averaged drag and lift coefficients C_D and C_L are then calculated as

$$C_D = \frac{1}{(t_{\text{end}} - t_{\text{start}})} \int_{t_{\text{start}}}^{t_{\text{end}}} \boldsymbol{F} \cdot \boldsymbol{e}_x \, \mathrm{d}t, \qquad C_L = \frac{1}{(t_{\text{end}} - t_{\text{start}})} \int_{t_{\text{start}}}^{t_{\text{end}}} \left((\boldsymbol{F} \cdot \boldsymbol{e}_y)^2 + (\boldsymbol{F} \cdot \boldsymbol{e}_z)^2 \right)^{1/2} \mathrm{d}t, \tag{2.10}$$

where t_{start} is chosen to disregard the initial transient due to the chosen (uniform) initial condition and t_{end} is the final simulation time.

For $Re \geq 275 \pm 5$, the flow is unsteady and vortices shed from the static sphere [117, 118]. Similarly to [118], we evaluate the vortex shedding frequency f and the corresponding Strouhal number $St = \frac{2rf}{u_0}$ by calculating an averaged period between successive peak values in the transverse force components[4]. A main vortex shedding frequency cannot be reliably defined using this methodology for $Re = 500$: as it can be seen from figure 2.16, the time variations in the transverse force components do not reveal a well-defined periodic pattern for $Re = 500$. In fact, a Fourier decomposition of (pseudoperiodic portions of) the signals actually reveals a broad frequency spectrum with significant contributions up to $St \simeq 0.17$ in that case.

Our results are summarized and presented in tables 2.5 and 2.6 along with available data from various publications from the literature.

For all the simulations from this section, we have used the inner-loop convergence criterion $\left\| \nabla \Phi^k - \nabla \Phi^{k-1} \right\| < 10^{-4} \, u_0$ (see section 2.2.5) in order to ensure a proper and

[4]Note that the analysis from [117] is different in that it is based on time variations of the pressure in the wake.

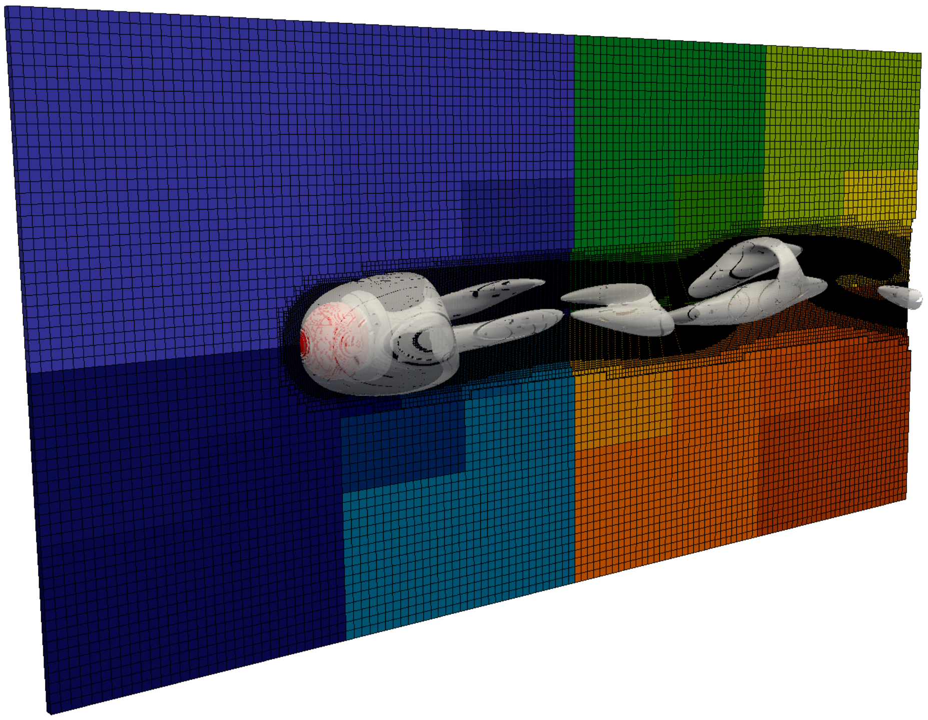


Figure 2.13: Visualization of the unsteady flow past a sphere for $Re = 300$. The trees are level 4/7 rooted in a $8 \times 4 \times 4$ macromesh, leading to approximately 6 million leaves. The snapshot is taken at time $t = 221\, r/u_0$. The colors correspond to the process ranks and the surface is an isocontour of the Q-criterion [125] for $Q = 0.006$. This simulation was run on the Stampede2 supercomputer with 1024 processes.

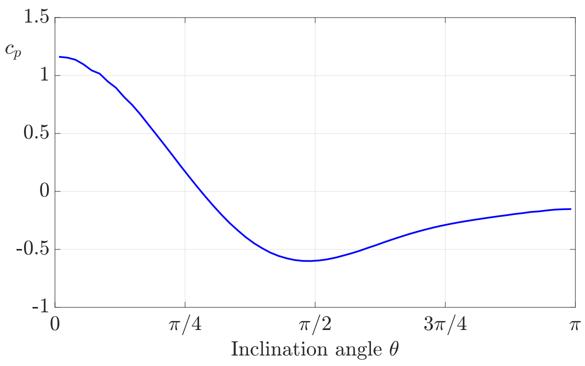
	$Re = 50$	$Re = 100$	$Re = 150$	$Re = 215$	$Re = 250$	
	C_D	C_D	C_D	C_D	C_D	C_L
Kim *et al.* [119]	-	1.09	-	-	0.70	0.059
Johnson *et al.* [120]	1.57	1.08	0.90	-	0.70	0.062
Constantinescu *et al.* [121]	-	-	-	-	0.70	0.062
Choi *et al.* [122]	-	1.09	-	-	0.70	0.052
Bagchi *et al.* [118]	1.57	1.09	-	-	0.70	-
Marella *et al.* [123]	1.56	1.06	0.85	0.70	-	-
Guittet *et al.* [55]	-	1.11	-	-	-	-
Present	1.61	1.11	0.91	0.76	0.72	0.062

Table 2.5: Drag coefficient (and lift coefficient, if relevant) for the steady flow past a sphere. The time averages were obtained with $t_{\text{start}} = 50\, r/u_0$ and $t_{\text{end}} = 200\, r/u_0$ for $Re \leq 215$ and with $t_{\text{start}} = 275\, r/u_0$ and $t_{\text{end}} = 400\, r/u_0$ for $Re = 250$ (see (2.10)).

	$Re = 300$			$Re = 350$		$Re = 500$
	C_D	C_L	St	C_D	St	C_D
Kim *et al.* [124]	0.657	0.067	0.134	-	-	-
Johnson *et al.* [120]	0.656	0.069	0.137	-	-	-
Constantinescu *et al.* [121]	0.655	0.065	0.136	-	-	-
Choi *et al.* [122]	0.658	0.068	0.134	-	-	-
Marella *et al.* [123]	0.621	-	0.133	-	-	-
Bagchi *et al.* [118]	-	-	-	0.62	0.135	0.555
Mittal *et al.* [117]	0.64	-	0.135	0.625	0.142	-
Guittet *et al.* [55]	0.659	-	0.137	0.627	0.141	-
Present	0.673	0.068	0.134	0.633	0.132±0.002	0.558

Table 2.6: Drag coefficients for the unsteady flow past a sphere. If relevant, the lift coefficient and the Strouhal number are presented as well. The time averages were obtained with $t_{\text{start}} = 200\, r/u_0$ and $t_{\text{end}} = 400\, r/u_0$ (see (2.10)).

accurate enforcement of the no-slip boundary condition on the surface of the sphere and to investigate the relevance of such a procedure in this specific case. Figure 2.17 shows relevant information pertaining to that analysis. As illustrated in that figure, the solver converges most of the time in two iterations and the correction brought by the second iteration is 3 to 4 orders of magnitude smaller than for the approximate projection step, in this case. Although the added computational cost is limited (around 30%), the accuracy of the results would most likely not have suffered from using an approximate projection



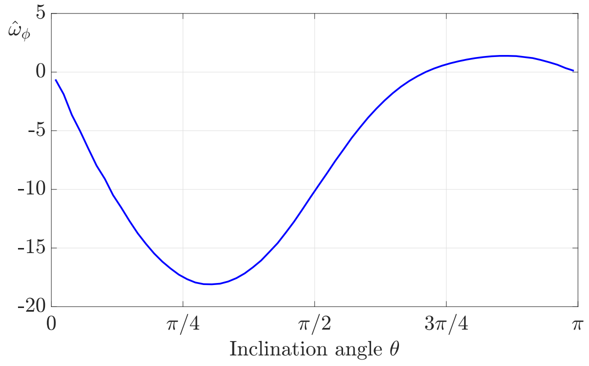
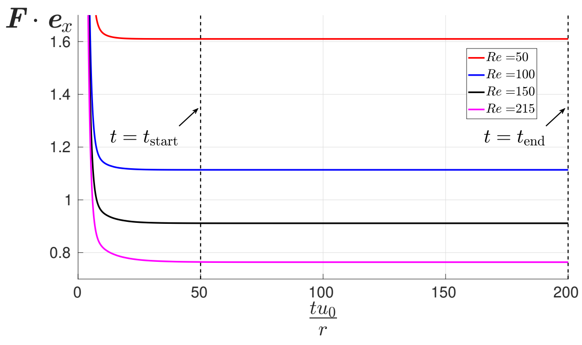


Figure 2.14: Evolution of relevant surface quantities on the surface of the static sphere for $Re = 100$, as a function of the inclination angle θ measured from the front point $(-r, 0, 0)$. Left: non-dimensional local pressure $c_p = p/\left(\frac{1}{2}\rho u_0^2\right)$ as a function of the inclination angle. Right: non-dimensional azimuthal vorticity $\hat{\omega}_\phi = \frac{-2r}{u_0}\boldsymbol{e}_\phi \cdot (\nabla \times \boldsymbol{u})$, where $\boldsymbol{e}_\phi = \boldsymbol{e}_r \times \boldsymbol{e}_\theta$, using spherical coordinates centered at the sphere's center along with the defintion of θ given here above. These results are in good agreement with [124].



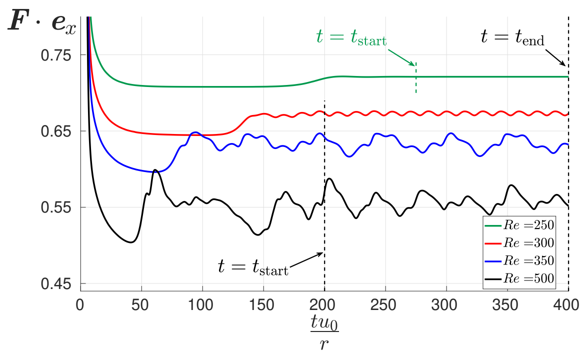


Figure 2.15: Drag coefficient on a sphere for axisymmetric steady flows (left) and for non-axisymmetric or unsteady flows (right), corresponding to Reynolds numbers ranging from 50 to 500.

in this context.

2.4.3 Oscillating sphere in a viscous fluid

The solver presented in this article is able to handle moving geometries. We illustrate this capacity by computing the drag force developed by the flow of a viscous fluid due to

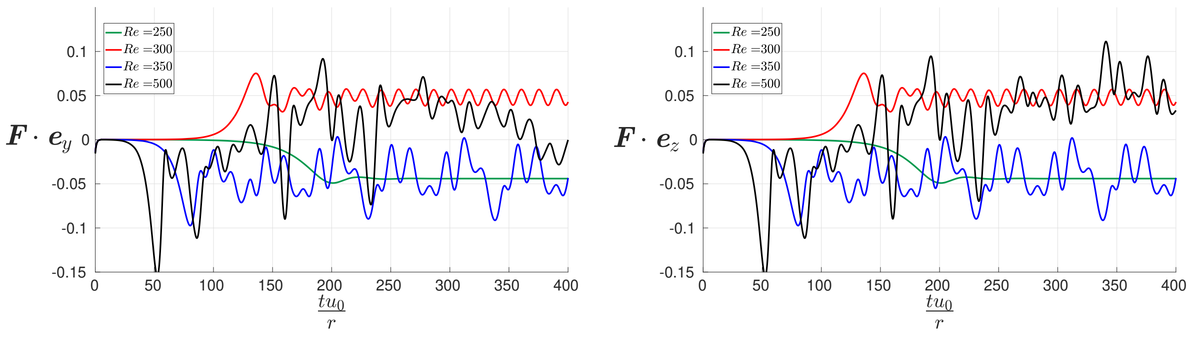


Figure 2.16: Nondimensional transverse force components for non-axisymmetric and/or unsteady flows.

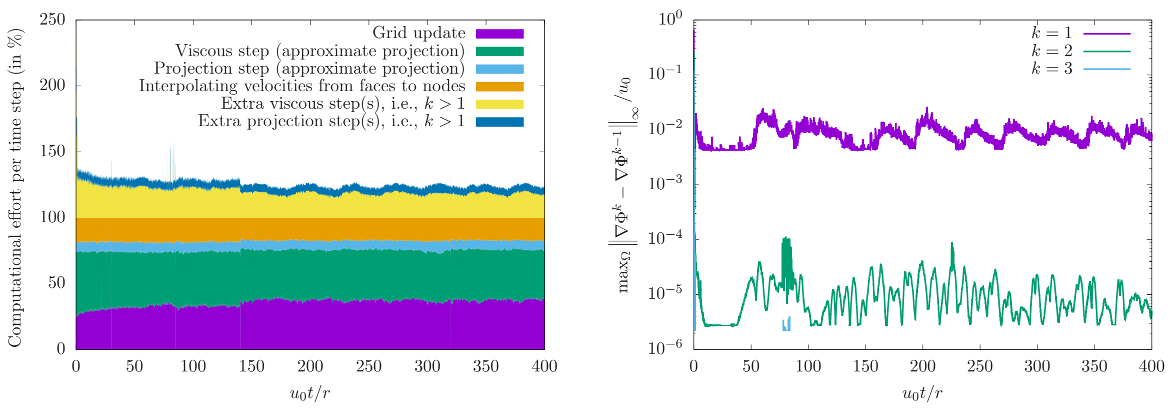


Figure 2.17: Illustration of the computational cost and convergence rate associated with the fixed point iteration from section 2.2.5, when ensuring a stringent user-defined control on $\boldsymbol{u}$ at all time steps, for the simulation of the flow past a sphere with $Re = 500$. Left: ratio(s) of the computational costs per time step for the main tasks at play, normalized to the raw computational cost of an approximate projection method (which would correspond to $k_{\text{max}} = 1$). Right: measures of interest considered by the inner loop criterion (only a few time steps required 3 inner iterations hence the partial blue curve).

the oscillatory motion of a rigid sphere in a closed box.

We consider a sphere of radius $r = 0.1$ in a domain $\Omega = [-1, 1]^3$. The kinematics of the center of the sphere $\boldsymbol{c}(t)$ is dictated by

$$\boldsymbol{c}(t) = -X_0 \cos(2\pi f_0 t)\, \boldsymbol{e}_x, \tag{2.11}$$

and we use the (time-varying) levelset function $\phi\left(\boldsymbol{x}, t\right) = r - \left\|\boldsymbol{x} - \boldsymbol{c}\left(t\right)\right\|$ with the corresponding grid-construction parameters $K = 1.2$, $b = 4$ and $\gamma = 0.1$. The motion of the sphere is set to be purely translational (no rotation) so that its kinematics is fully described by (2.11). The dynamics of the surrounding fluid is dictated by no-slip boundary conditions enforced onto the surface of the oscillatory sphere, i.e.,

$$\boldsymbol{u}\left(\boldsymbol{x}, t\right) = 2\pi f_0 X_0 \sin\left(2\pi f_0 t\right) \boldsymbol{e}_x, \quad \forall \boldsymbol{x} \in \Omega : \left\|\boldsymbol{x} - \boldsymbol{c}\left(t\right)\right\| = r. \tag{2.12}$$

No-slip boundary conditions are also enforced on the (static) borders of the computational domain.

This setup naturally defines a characteristic velocity scale $u_0 = 2\pi f_0 X_0$, a characteristic frequency f_0 and a characteristic length scale r leading to two nondimensional numbers

$$\frac{r}{X_0} \quad \text{and} \quad Re = \frac{\rho\, 2\pi f_0 X_0\, 2r}{\mu}. \tag{2.13}$$

We set the first nondimensional number to 4 by assigning $X_0 = r/4$. The density is set to 1 and the dynamic viscosity μ is determined to match the desired Reynolds number set by the user based on (2.13).

For this example, we choose the fixed time step $\Delta t = \dfrac{1}{200 f_0}$ and the simulations are run for three full oscillation cycles for 7 different Reynolds numbers ranging from 10 to 300. The iterative procedure explained in section 2.2.5 is necessary in this case of moving boundary in order to correctly enforce the desired no-slip condition (2.12). Since we are interested in the overall forces applied onto the sphere as a result of its motion, the inner iterative technique is carried on until $\left\|\Phi^k - \Phi^{k-1}\right\|_\infty < \varepsilon_\Phi = 2\,\varepsilon\left(\pi f_0 X_0\right)^2 \Delta t$ where $\varepsilon = 0.1$ for all time steps. This corresponds to limiting the difference in pressure between successive iterates to 10% at most of the maximum dynamic pressure resulting from the motion of the sphere. The fixed point method seems more relevant in this context compared to what was observed in section 2.4.2: the number of iterations required to ensure this convergence condition grows with Re: for every time step, the solver performs 3 to 4 inner iterations to enforce the desired boundary conditions correctly, as illustrated in figure 2.20 for $Re = 300$. A mesh of resolution 5/10 rooted in a single macromesh cell is used, resulting in about 500,000 leaves. A uniform grid with similar finest resolution would have $2^{30} \simeq 10^9$ computational cells. Every simulation completed in about 8 hours using 40 MPI tasks on a local workstation running a Dual Intel Xeon Gold 6148 processor with 64 GB of RAM.

As for the flow past a sphere, the nondimensional force applied onto the oscillating sphere is monitored over time, according to equation (2.9). The results are presented in figure 2.18. As expected, we observe that the amplitude of the drag coefficient increases as the Reynolds number decreases. Furthermore, we observe a lag in the response as the Reynolds number decreases. Indeed, as the viscous forces become more important, the information takes longer to propagate in the fluid. In contrast, the forces in a system dominated by inertia come mainly from the pressure term and the incompressibility condition enforces instantaneous propagation of the information. Figure 2.19 shows some visual representations of the computational grid for $Re = 50$ at $t = 2.4/f_0$.

2.4.4 Turbulent superhydrophobic channel

As detailed and illustrated above, the solver is designed to allow dynamic grid adaptation over time. This feature is especially relevant and appealing when the complex flow dynamics to be captured are bounded to a (small) evolving portion of the computational domain, as it dramatically reduces the global number of computational cells compared to a regular uniform grid. Indeed, in such cases, the computational overhead associated

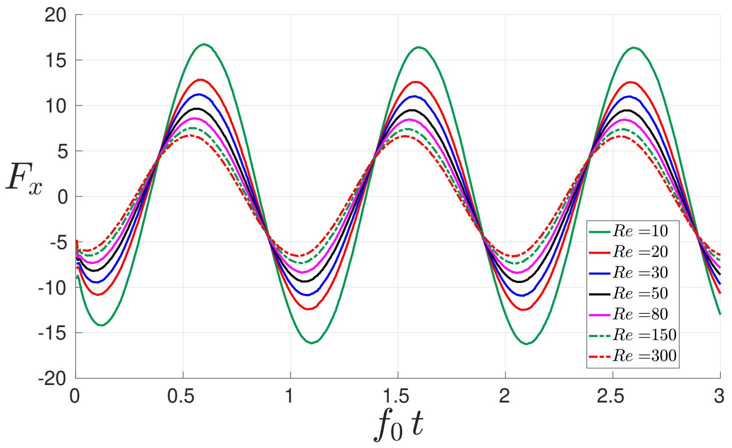


Figure 2.18: Evolution of the $x-$component of the nondimensional force applied onto the periodically oscillating sphere in a closed box for a range of Reynolds numbers. The magnitude of the force increases and the peaks appear at later times as the Reynolds number decreases and the viscous forces become dominant.

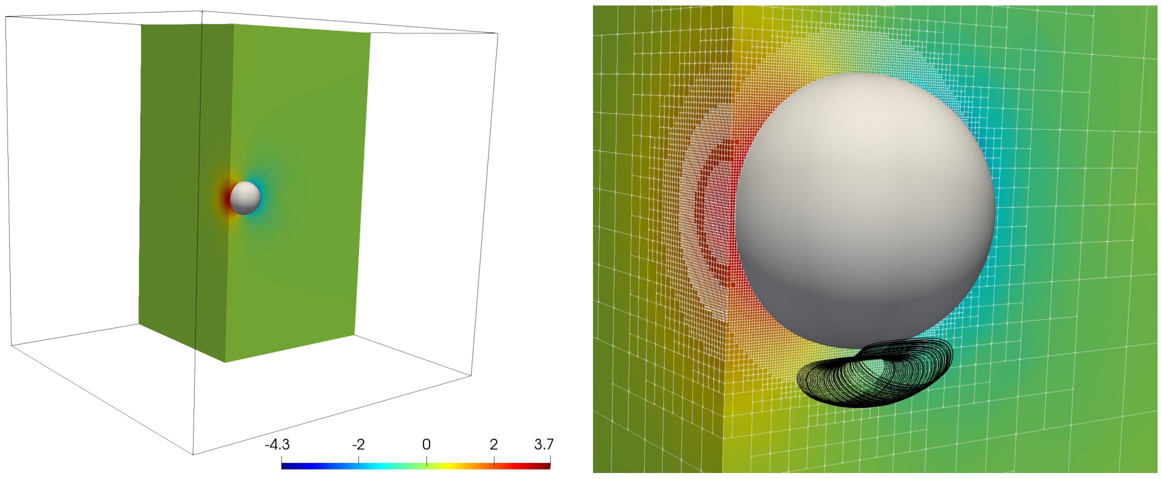


Figure 2.19: Left: illustration of one fourth of the computational domain colored with the pressure (nondimensionalized by $\frac{1}{2}\rho\pi r^2 f_0^2$), along with the solid sphere ($Re = 50$, $t = 2.4/f_0$). Right: zoom-in on the sphere, illustration of the local adaptivity of the computational grid and representation of some streamlines near the moving interface.

with dynamically adapting the computational grid, with setting new operators and new linear solvers is small compared to the prohibitive computational cost of using a uniform grid of equivalent finest resolution throughout the domain.

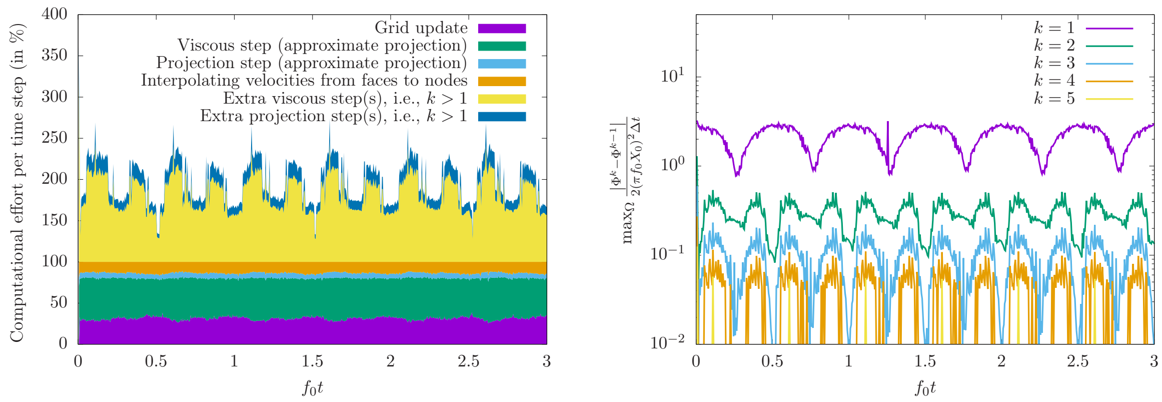


Figure 2.20: Illustration of the computational cost and convergence rate associated with the fixed point iteration from section 2.2.5, when ensuring point-wise convergence on the Hodge variable within bounds corresponding to a pressure threshold of 10% of the maximum dynamic pressure resulting from the kinematics of the oscillating sphere, as considered in section 2.4.3, for $Re = 300$. Left: ratio(s) of the computational costs per time step for the main tasks at play, normalized to the raw computational cost of an approximate projection method (which would correspond to $k_{\max} = 1$). Right: measures of interest considered by the inner loop criterion.

However, while local mesh refinement remains valuable, dynamic re-meshing may not be most desirable in applications requiring (almost) static, dense regions of fine grid cells, as extra operations associated with dynamic grid adaptation would significantly increase the overall computational cost, without any significant reduction of the overall number of computational cells to consider. We have alleviated this issue by allowing the solver to store all possible data structures[5] (linear solvers, possible preconditioners, interpolation operators, etc.) in memory and use them as long as they are valid, i.e., as long as the grid is not modified.

Statistically steady physical problems align perfectly with the above solver features, since they require fixed regions of specified spatial resolution by nature, and the results need to be accumulated over a (very) large number of time steps to ensure their statistical convergence. In order to illustrate the capability of the solver to address such

[5]For the linear solvers associated with the viscosity step, only diagonal terms are affected by a new value of Δt (see subsection 2.2.3). Therefore, only diagonal terms are updated when the computational grid is not modified.

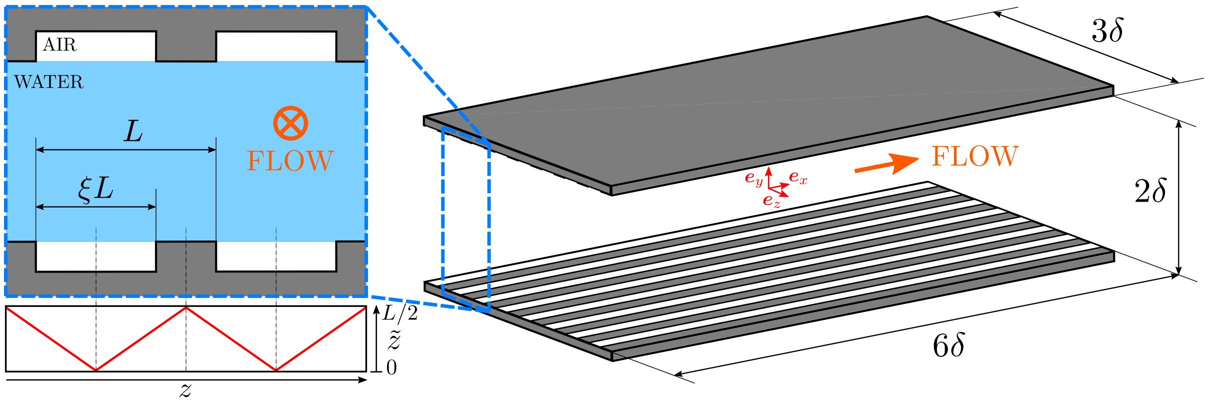


Figure 2.21: Schematic of the superhydrophobic surface and corresponding notations considered in section 2.4.4: we denote the pitch by L and the gas fraction by ξ. The variable $\tilde{z}$ is used to average over corresponding spanwise locations.

problems, we consider the fully developed turbulent flow in a superhydrophobic channel as previously simulated in [126, 127]. We use a computational domain of dimensions $6\delta \times 2\delta \times 3\delta$, where δ is half of the channel height, with periodic boundary conditions along the streamwise and spanwise directions, as illustrated in figure 2.21. The coordinates are chosen such that x points downstream, y is normal to the walls and z is in the spanwise direction. The flow is driven in the positive streamwise direction by a spatially uniform and constant force per unit mass $\boldsymbol{f} = f_x \boldsymbol{e}_x$. By analogy with canonical channel flows, we define the friction velocity $u_\tau = \sqrt{f_x \delta}$.

We consider gratings oriented parallel to the flow on both walls $y = \pm\delta$. The superhydrophobic nature of these surfaces enables them to entrap pockets of air, such that parts of he walls are replaced by a liquid-air interface, which is assumed to be shear-free. We assume the liquid-air interface to be and remain flat at all times (deflections of the interface are neglected). We therefore model the air-liquid interface regions using no-penetration, free-slip boundary conditions[6]

$$v\big|_{y=\pm\delta} = 0, \qquad \left.\frac{\partial u}{\partial y}\right|_{y=\pm\delta} = 0 \quad \text{and} \quad \left.\frac{\partial w}{\partial y}\right|_{y=\pm\delta} = 0, \tag{2.14}$$

while the rest of wall surfaces use no-slip boundary conditions,

$$\boldsymbol{u}\big|_{y=\pm\delta} = \mathbf{0}.$$

Eight longitudinal grates giving a gas fraction of 50% are used, i.e., the pitch length L is set to $3\delta/8$ and the gas fraction ξ is set to 0.5 (see figure 2.21). The fluid properties and other control parameters are set such that the canonical friction Reynolds number $Re_\tau = \dfrac{\rho u_\tau \delta}{\mu}$ is equal to 143. The computational domain is meshed with one single Octree of minimum level 7 and maximum level 9. This choice of macromesh results in computational cells with an aspect ratio so different from 1 that third-degree neighbor cells are required for the reliable construction of face-seeded Voronoi cells (see figure 2.2 for a two-dimensional illustration). Therefore, this simulation setup makes an extensive use of the capability to fetch *third*-degree ghost neighbor cells, which is enabled by the algorithms from section 2.2.6. The capability of the solver to address such problems even with stretched computational cells was investigated and validated using a known analytical solution in the laminar case.

In order to ensure sufficient grid resolution for regions close to the no-slip parts of the walls, we define the level-set function[7]

$$\phi\left(\boldsymbol{x}\right) = -\text{dist}\left(\boldsymbol{x}, \text{no-slip region of the wall}\right)$$

[6]Note that $\left.\dfrac{\partial v}{\partial x}\right|_{y=\pm\delta} = \left.\dfrac{\partial v}{\partial z}\right|_{y=\pm\delta} = 0$ since $v\big|_{y=\pm\delta} = 0$ on the entire wall surfaces.

[7]Note that ϕ is negative everywhere so no interface is defined within the domain.

to be used with the refinement criterion (2.1) setting $K = 10$. This choice of K and maximum refinement level ensures that the walls are entirely covered by the finest computational cells over a thickness of 0.1δ. Except for the thickness of four grid cells layering the walls, the local grid resolution is equivalent to, or finer than, the resolution from [126, 127] everywhere. In [126, 127], a stretched grid was used with constant mesh size in the streamwise and spanwise directions while the cell thickness was distributed using a hyperbolic tangent profile in the wall-normal direction. The main difference between such a stretched grid and our Octree approach lies in the fact that the aspect ratio of our computational cells is constant. As cells get thinner when approaching the wall regions, they also get shorter and narrower: the spatial resolution close to the walls for the Octree grid is four times finer than for the stretched grid in the spanwise and streamwise directions, hence producing more accurate results in those directions than stretched grids do. This however significantly increases the total number of computational cells to be used in our approach, since we need more than 21.8×10^6 cells, as opposed to about 2.1×10^6 in the case of a stretched grid. Dynamic grid adaptation based on local vorticity is less useful in this example, since the background grid already captures enough details (γ is thus irrelevant in this case), which enables us to reduce time execution. The conjugate gradient method is used for solving the projection step along with an algebraic multigrid preconditioner (from the HYPRE distribution).

A thorough analysis of the analytical solution known in the laminar cases shows that the viscous stress is singular at the edges of the walls transitioning between free-slip and no-slip boundary conditions (see [128, 129, 130, 131] for more details). Early numerical tests revealed that setting the inner loop convergence criterion (see section 2.2.5) to ensure $\left\|\Phi^k - \Phi^{k-1}\right\|_\infty < \varepsilon_\Phi$ would fail because the (floating-value) Hodge variable Φ would grow unbounded in the cells layering these transition edges. However, the velocity field must be bounded everywhere, and therefore, so must be $\left\|\nabla\Phi\right\|_\infty$. As a matter of fact, setting the inner loop convergence criterion (see section 2.2.5) to ensure $\max_\Omega \left\|\nabla\Phi^k - \nabla\Phi^{k-1}\right\|_\infty < 10^{-6}U_\mathrm{b}$, wherein U_b is the mean, bulk velocity in the streamwise direction through the channel, resulted in fully controlled simulations. For most time steps, the solver required three inner iterations to converge (the value of the convergence measure for the first iterate, i.e., $\max_\Omega \|\nabla\Phi^1 - \nabla\Phi^0\|_\infty$, was observed to be of the order of $10^{-4}U_\mathrm{b}$).

The simulation is initialized to the known laminar solution and executes until flow instabilities amplify and a fully-developed turbulent state is eventually reached. The bulk Reynolds number

$$Re_\mathrm{b} = \frac{\rho U_\mathrm{b}\delta}{\mu},\ \text{where } U_\mathrm{b} = \frac{1}{6\delta^2}\int_{-\delta}^{\delta}\int_{-1.5\delta}^{1.5\delta} \boldsymbol{u}\cdot\boldsymbol{e}_x \,\mathrm{d}z\,\mathrm{d}y$$

and the nondimensional viscous forces from the no-slip regions of the walls

$$\boldsymbol{F}_\text{wall, visc.} = \frac{1}{\rho f_x\, 36\delta^3}\sum_{k_y=\{-1,1\}}\sum_{k_z=-4}^{3}\int_{-3\delta}^{3\delta}\int_{(k_z+\xi)L}^{(k_z+1)L} -k_y\left[\mu\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^\mathrm{T}\right)\cdot\boldsymbol{e}_y\right]\Big|_{y=k_y\delta}\,\mathrm{d}z\,\mathrm{d}x$$

are monitored over time. Their evolution is illustrated in figure 2.22.

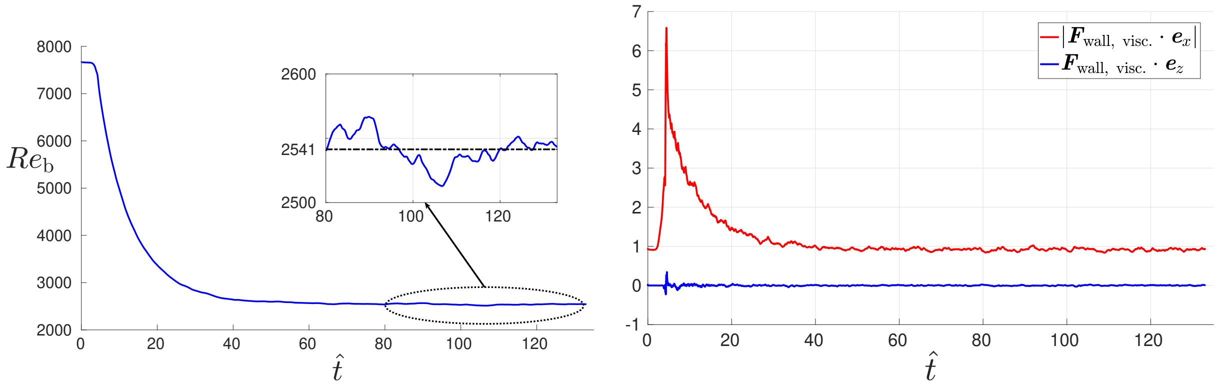


Figure 2.22: Macroscopic variables monitored over the course of the simulation of a turbulent flow through a superhydrophobic channel. Left: evolution of $Re_\mathrm{b} = \rho U_\mathrm{b}\delta/\mu$; right: evolution of the nondimensional viscous forces from the no-slip regions of the walls. In these graphs, the nondimensional time $\hat{t}$ is defined as $\hat{t} = \dfrac{u_\tau t}{\delta}$.

From the evolution of the monitored macroscopic quantities of interest illustrated in figure 2.22, we consider the time window from $\hat{t}_{\text{start}} = \dfrac{u_\tau t}{\delta} = 80$ until the end of the simulation, $\hat{t}_{\text{end}} = 133.1$, which we use for time-averaging results associated with the fully-developed regime (which corresponds to about 300,000 time steps). An illustrative snapshot of the simulation in this time window is presented in figure 2.24.

We obtain an average bulk Reynolds number of 2541; we also consider the time-and-slice-averaged velocity profile, i.e.,

$$\left\langle \frac{u}{u_\tau} \right\rangle_{x,z,t} = \frac{1}{18\delta^2 \left(\hat{t}_{\text{end}} - \hat{t}_{\text{start}}\right)} \int_{\hat{t}_{\text{start}}}^{\hat{t}_{\text{end}}} \int_{-3\delta}^{3\delta} \int_{-1.5\delta}^{1.5\delta} \frac{u}{u_\tau} \,\mathrm{d}z\,\mathrm{d}x\,\mathrm{d}\hat{t}$$

as a function of y/δ, as well as the time-and-line-averaged velocity profile, i.e.,

$$\left\langle \frac{u}{u_\tau} \right\rangle_{x,t} = \frac{1}{6\delta \left(\hat{t}_{\text{end}} - \hat{t}_{\text{start}}\right)} \int_{\hat{t}_{\text{start}}}^{\hat{t}_{\text{end}}} \int_{-3\delta}^{3\delta} \frac{u}{u_\tau} \,\mathrm{d}x\,\mathrm{d}\hat{t},$$

which is built by also averaging corresponding locations over the air-interface and over the ridges in the spanwise direction. Formally, this results in a function of y/δ and of the *grate-normalized* spanwise coordinate $\tilde{z}$ defined as

$$\tilde{z} = \begin{cases} \left|\left(z \pmod{L}\right) - \dfrac{L\xi}{2}\right| & \text{if } z \pmod{L} \leq L\xi, \\ \dfrac{L}{2} - \left|\left(z \pmod{L}\right) - \dfrac{L\left(1+\xi\right)}{2}\right| & \text{otherwise,} \end{cases} \tag{2.15}$$

(see illustration in figure 2.21). These time-averaged velocity profiles are illustrated in figure 2.23.

As illustrated in figure 2.23, the mean fluid velocity at the air interface can reach up to 50% of the maximum mean velocity (found at the center of the channel). This figure also illustrates how sharp the change is: 74% of the variation from the no-slip ridge to the maximum air interface velocity (found above the center line of the interface) occurs over

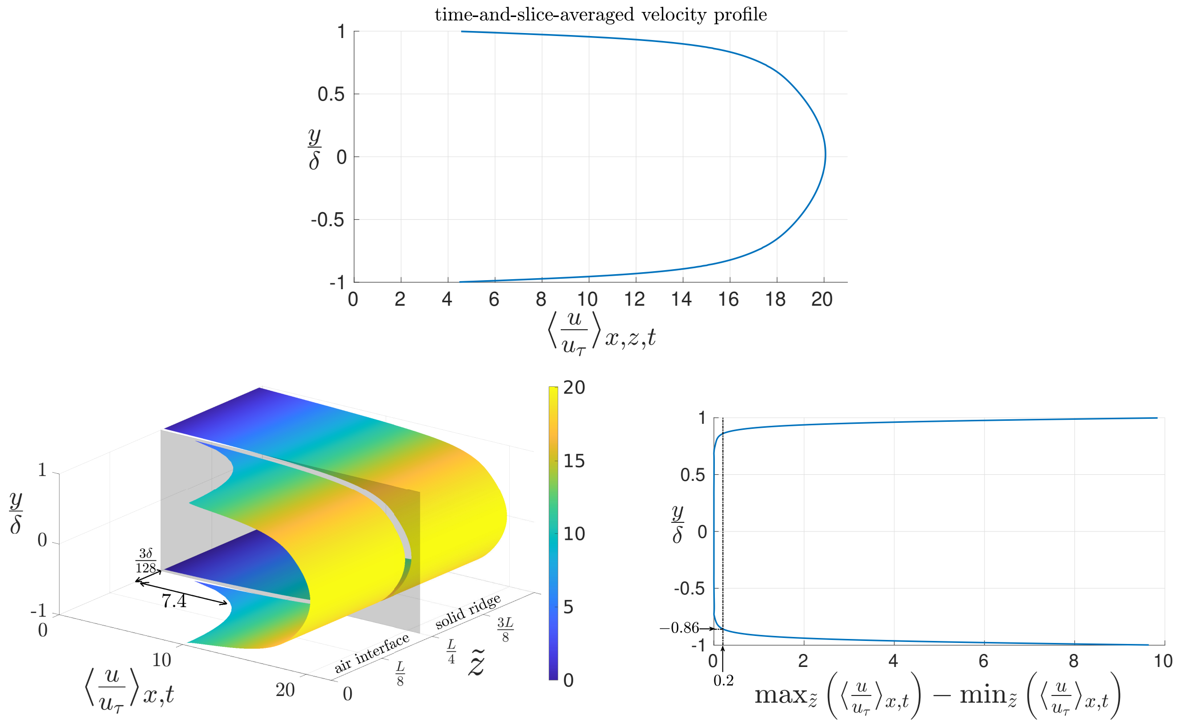


Figure 2.23: Top: time-and-slice-averaged velocity profile. Bottom left: time-and–line-averaged velocity profiles. Notice the sharpness of the transition between no-slip and free-slip line-averaged profiles. A significant portion of the transition (74%) takes place over a distance of $\dfrac{3\delta}{128}$ in the spanwise direction. While this distance corresponds to the width of one single computational cell in the stretched grid approach of [127], our octree grid uses 4 narrower computational cells over that region. Bottom right: illustration of how the time-and-line-averaged profiles become $\tilde{z}$-independent far enough from the walls; in this case, time-and-line-averaged velocity profiles are essentially all equivalent (within 1% of the mean velocity to be found in the center of the channel) farther than 0.14δ from the walls.

a distance of $3\delta/128$ in the spanwise direction, which corresponds to 4 computational cells in our setup (versus 1 cell in [127]).

When averaging velocity profiles across entire planar sections of the channel, the existence of such free-slip regions results in a nonzero slip velocity U_s at the wall. This slip velocity is a quantity of primary relevance in the context of Navier's slip model, along with the slip length b which relates the slip velocity to the mean wall shear via $U_s = \pm b \left.\dfrac{\partial}{\partial y} \langle u \rangle_{x,z,t}\right|_{y=\mp\delta}$. The slip parameters U_s and b were evaluated by least-square fitting the linear profiles $\frac{U_\mathrm{s}}{u_\tau}\left(1+\frac{\delta \pm y}{b}\right)$ (+ and − correspond to bottom and top walls, respectively) to our results for $\left\langle \frac{u}{u_\tau} \right\rangle_{x,z,t}$ over the 5 finest grid cells layering the walls, which corresponds to a thickness of about $2.8\delta_\tau$, where $\delta_\tau = \frac{\mu}{\rho u_\tau}$ is the viscous lengthscale. The two sets of fitting parameters yield $b = (0.0302 \pm 0.0002)\,\delta = (4.320 \pm 0.025)\,\delta_\tau$ and $U_s = (4.255 \pm 0.040)\,u_\tau$.

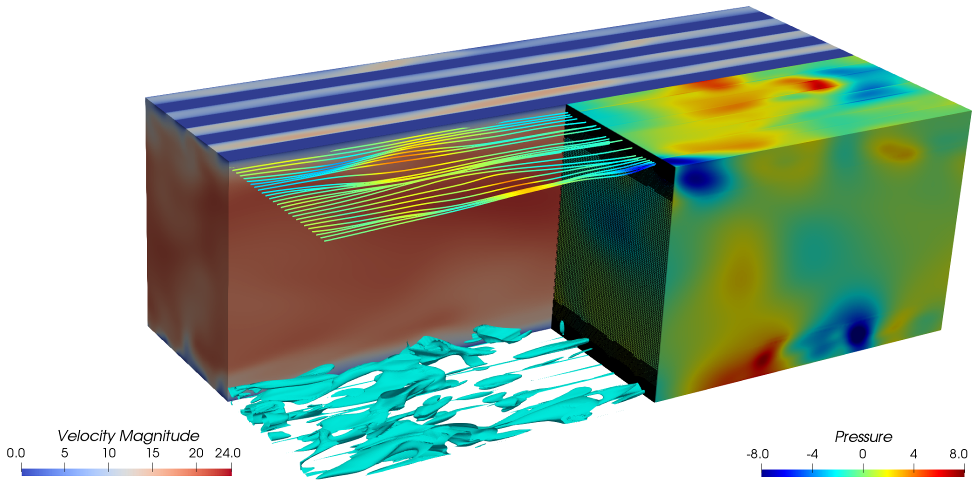


Figure 2.24: Visualization of a snapshot for the simulation of the turbulent superhydrophobic channel flow. The half of the domain corresponding to $z < 0$ is colored by $\frac{\|\boldsymbol{u}\|}{u_\tau}$; the quarter of the domain corresponding to $z > 0$ and $x > 0$ is colored by $\frac{p}{\rho u_\tau^2}$. A slice of the computational grid, streamlines and isocontours of $\lambda_2 = -0.3\left(\frac{U_\mathrm{b}}{\delta}\right)^2$ (using the λ_2-criterion from [132]) are also shown.

Our setup value of Re_τ was chosen for comparison purposes with one of the simulations from [127, 126], which reports a (mean) value of $Re_\tau = 143$ using a simulation setup enforcing a (constant) mass flow corresponding to $Re_\mathrm{b} = 2,800$ for the same channel geometry. Under these conditions, [127, 126] reports $b = 0.0366\,\delta = 5.17\,\delta_\tau$ and $U_s = 5.26\,u_\tau$. Therefore, when compared to those results, our simulation leads to a reduced flow rate for a comparable driving force (about 10% less) and to a smaller slip velocity as well as a smaller slip length. Besides the difference in simulation setup (constant driving force as opposed to constant mass flow rate), such deviations may also originate from numerical and/or modeling differences to be found between the two approaches. To assess this, we also compare our results to the simulations of [133], who used a lattice-Boltzmann method. We select their simulation whose value of L/δ_τ is closest to ours, since [127] established that L/δ_τ is the single most important parameter that determines slip length (at fixed gas fraction). We therefore compare our simulation, which has $L/\delta_\tau = 53.6$, to the $L/\delta_\tau = 56.2$ case of [133], who found $b/\delta_\tau = 4.23$. This value is appreciably smaller than the result of 5.17 of [127], but matches closely our $b/\delta_\tau = 4.32$.

In terms of modeling, [127, 126] opted for a simplified wall-treatment for the spanwise velocity component w, by setting $w|_{y=\pm\delta} = 0$ instead of the stress-free condition (2.14) above the free-slip wall regions. Enforcing $w|_{y=\pm\delta} = 0$ above the air pockets does not rely on physical grounds, and may result in simplified near-wall flow structures that artificially promote streamwise velocity. Indeed, this simplified condition results in $\dfrac{\partial w}{\partial z} = 0$, which in turn simplifies the incompressibility condition, at the walls, into $\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} = 0$. At the wall-located air interfaces, this latter equation stands as an artificial constraint, within z-orthogonal planes, enabling transfer of kinetic energy only between wall-normal and streamwise velocity components. This constraint could lead, in turn, to an overestimation of the slip velocity and/or of the total mass flow across the channel, which could help explain the difference between the results of [127] and those of subsequent simulations.

In terms of numerical methods, we emphasize that the use of a semi-Lagrangian scheme for the advection terms comes with a significant amount of numerical dissipation, which may in turn lead to overestimated viscous dissipation. While this cannot be excluded, we also want to point out that our simulation setup makes use of a spatial resolution that is 4 times finer than the resolution from [127], in both the streamwise and spanwise directions. Firstly, such a fine resolution in the streamwise direction is expected to alleviate the numerical dissipation associated with our advection scheme.

Secondly, such a resolution in the spanwise direction may actually stand as a *requirement* in order to capture the sharp variation in velocity profiles between free-slip and no-slip wall regions. Indeed, figure 2.23 illustrates that $\left.\frac{\partial}{\partial z}\left\langle u\right\rangle_{x,t}\right|_{y=\pm\delta}$ is the largest near the boundaries transitioning from free-slip to no-slip regions; therefore, using a coarse spanwise resolution in that area may lead to an underestimation of the overall viscous dissipation.

In order to quantify the relative importance of this term, we estimate its contribution to the viscous dissipation taking place in near-wall layers and compare it to the contribution of the mean, slice-averaged streamwise shear term (i.e., as if we were dealing with a regular channel). Assuming that $\left.\frac{\partial}{\partial z}\left\langle u\right\rangle_{x,t}\right|_{y=\pm\delta}$ is not negligible within bands of $3\delta/128$ around solid ridges only (in our case, we have 16 such bands on either wall), our comparative estimate is

$$\frac{6\delta\,\frac{16\times 3\delta}{128}\mu\left(\left.\frac{\partial}{\partial z}\left\langle u\right\rangle_{x,t}\right|_{y=-\delta}\right)^2}{6\delta\,3\delta\,\mu\left(\left.\frac{\partial}{\partial y}\left\langle u\right\rangle_{x,z,t}\right|_{y=-\delta}\right)^2}\approx\frac{16}{128}\frac{\left(\frac{7.4u_\tau}{3\delta/128}\right)^2}{\left(\frac{U_s}{b}\right)^2}=62.8\%,$$

where we estimated $\left.\frac{\partial}{\partial z}\left\langle u\right\rangle_{x,t}\right|_{y=-\delta}=\frac{7.4u_\tau}{3\delta/128}$ based on the results illustrated in figure 2.23 to produce a fair measure in comparison with the grid resolution from [127], although $\left.\frac{\partial}{\partial z}\left\langle u\right\rangle_{x,t}\right|_{y=-\delta}$ becomes almost twice as large as we approach the edge in our computational setting. Although wall viscous shear dissipation may be smaller than the overall (bulk) turbulent dissipation, we expect it to be non-negligible nonetheless, in particular when considering a relatively low friction Reynolds number as it is the case here[8]; in this context, the above comparative estimation indicates that spanwise wall shear, though not evenly distributed on the walls, is not a negligible factor to the overall viscous dissipation.

[8]In fact, if we consider an equivalent canonical channel with a mean, streamwise wall shear of U_s/b

2.5 Summary

We have described a Navier-Stokes solver for simulating incompressible flows in irregular domains on a forest of Octrees in a distributed computing framework. The parallel implementation of the solver requires the ability to access second- (or third-)degree cell neighbors, which led to the need for an expanded ghost layer of cells. We have introduced an algorithm to address that computational challenge on distributed forest of octree grids. We also have introduced parallel algorithms for the unambiguous definition and synchronization of global faces indices as required in a standard MAC arrangement. The performance of these individual algorithms has been assessed in terms of strong and weak scaling analyses. The strong scaling behavior of the entire solver has been verified up to more than 32,000 cores using a problem on a grid of 6.1×10^8 grid cells. The performance of the solver has been assessed on several large-scale three-dimensional problems: accurate results for the flow past sphere at various Reynolds numbers have been shown, several illustrations of the capabilities of our adaptive refinement approach have been provided and a simulation of the turbulent flow through a superhydrophobic channel has been performed with unparalleled spanwise (and streamwise) spatial resolution over regions of interest. When flow structures have a limited lifetime and/or are bounded to relatively small regions in the computational domain, our adaptive grid refinement approach was shown to successfully simulate the problems of interest using only a few percent of the number of grid cells that a uniform grid of equivalent finest resolution would require. The encapsulation of such a feature in a distributed computing

that we assume constant over layers of at least $3\dfrac{\mu}{\rho u_\tau}$, the viscous dissipation in these layers amounts for

$$2 \times 3\frac{\mu}{\rho u_\tau} \times 6\delta \times 3\delta \times \mu \left(\frac{U_s}{b}\right)^2 \simeq 105\, \rho u_\tau^3 \delta^2$$

which represents about 1/6 of the energy injection rate $36\delta^3 \rho f_x U_\mathrm{b}$ in our simulation setup.

framework allows for very-large scale simulations to be considered tractable from a computational standpoint and, therefore, to address increasingly complex multiscale and/or multiphysics problems.

Chapter 3

Transitioning to two-phase flows

3.1 Introduction

In this chapter, we turn our attention to the equations governing the dynamics of two-phase incompressible viscous flows. While they are piecewise equivalent to the single-phase flow problem, conditions coupling the flow dynamics across the interface separating two immiscible fluids must be taken into account. After introducing those interface conditions, we discuss their consequences on the extension of the numerical methodology presented for single-phase to two-phase problems. Finally, we outline the description of a flowchart that intends to account for these interface conditions while decoupling all unknowns.

3.2 Governing equations and interface conditions

We consider the incompressible flow dynamics of two immiscible fluids in a domain Ω. The fluids are separated by an interface Γ, which is represented as the zero-contour of a levelset function ϕ, i.e., $\Gamma = \{\boldsymbol{x} : \phi\left(\boldsymbol{x}\right) = 0\}$. The sign of ϕ naturally distinguishes

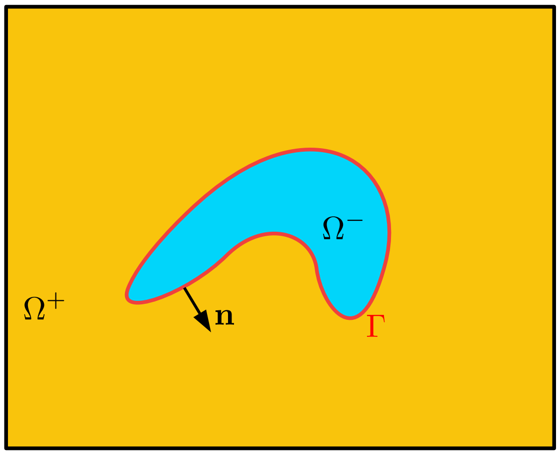


Figure 3.1: illustrations of the two subdomains Ω^+ and Ω^-, separated by the interface Γ, and its corresponding normal vector in two dimensions.

the two subdomains containing the distinct fluids $\Omega^+ = \{\boldsymbol{x} \in \Omega : \phi(\boldsymbol{x}) > 0\}$ and $\Omega^- = \{\boldsymbol{x} \in \Omega : \phi(\boldsymbol{x}) < 0\}$. We denote the unit vector normal to the interface pointing toward Ω^+ by $\boldsymbol{n}$ (see Figure 3.1). We make use of the superscript $+$ (resp. $-$) for properties related to the fluid in Ω^+ (resp. Ω^-). We denote by $\boldsymbol{w}$ the velocity of the interface, whose kinematics is thus described by the following advection equation for the levelset function

$$\frac{\partial \phi}{\partial t} + \boldsymbol{w} \cdot \nabla \phi = 0. \tag{3.1}$$

Far away from the interface, the flow dynamics is governed by the incompressible Navier-Stokes equations

$$\nabla \cdot \boldsymbol{u} = 0 \tag{3.2}$$

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \rho \boldsymbol{f} + \mu \nabla^2 \boldsymbol{u} \tag{3.3}$$

where ρ and μ are respectively the mass density and shear viscosity of the appropriate fluid.

Across the interface Γ, the conservation of mass reads

$$[\rho\,(\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n}]=0, \tag{3.4}$$

wherein $[q]=q^{+}-q^{-}$ represents the jump in the quantity q across Γ. Thus, (3.4) expresses that the mass flux through the interface $\dot{M}=\rho\,(\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n}$ must be continuous (no loss nor creation of mass on the interface). In presence of two fluids of different nature, no phase change may occur and therefore $\dot{M}=0$, but liquid-gas phase transition may be considered when allowing $\dot{M}\neq 0$ in general.

The conservation of momentum across the interface requires

$$\left[\left(\frac{\dot{M}^2}{\rho}+p\right)\boldsymbol{n}-2\mu\underline{\boldsymbol{E}}\cdot\boldsymbol{n}\right]=-\kappa\gamma\boldsymbol{n}+(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n})\cdot\nabla\gamma+\boldsymbol{G}, \tag{3.5}$$

where $\underline{\boldsymbol{E}}=\dfrac{1}{2}\left(\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^{\mathrm{T}}\right)$ is the strain rate tensor, $\kappa=\nabla\cdot\boldsymbol{n}$ is the interface curvature, $\underline{\boldsymbol{\delta}}$ is the second-order identity tensor, γ is the surface tension coefficient and $\boldsymbol{G}$ is an interface-defined, i.e., singular, force that may account for other or additional interface physics (e.g., hyper-elasticity of a membrane). In (3.5) and hereafter, $(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n})\cdot\boldsymbol{a}$ represents the projection of vector $\boldsymbol{a}$ orthogonal to $\boldsymbol{n}$, that is $(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n})\cdot\boldsymbol{a}=\boldsymbol{a}-(\boldsymbol{a}\cdot\boldsymbol{n})\,\boldsymbol{n}$. Hence, $(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n})\cdot\nabla\gamma$ is the surface gradient of the surface tension coefficient (Marangoni force).

In addition to the conservation of mass and momentum, one may need to account for the conservation of energy if required (e.g. for problems involving heat transfer and/or phase change). Far away from the interface, the conservation of energy reads

$$\rho\left(\frac{\partial e}{\partial t}+\boldsymbol{u}\cdot\nabla e\right)=2\mu\underline{\boldsymbol{E}}\colon\underline{\boldsymbol{E}}+k\nabla^2T+\dot{\Theta}. \tag{3.6}$$

where e is the local internal energy (per unit mass), k is the thermal conductivity and $\dot{\Theta}$ represents a possible external heat source.

The corresponding conservation principle across the interface requires

$$\left[\dot{M}\left(h+\frac{\dot{M}^2}{2\rho^2}\right)-2\dot{M}\nu\boldsymbol{n}\cdot\underline{\boldsymbol{E}}\cdot\boldsymbol{n}-k\nabla T\cdot\boldsymbol{n}\right]=T_{\mathrm{I}}\left(\frac{\mathrm{D}}{\mathrm{D}t}\left(\frac{\mathrm{d}\gamma}{\mathrm{d}T_{\mathrm{I}}}\right)+\frac{\mathrm{d}\gamma}{\mathrm{d}T_{\mathrm{I}}}\nabla_{\mathrm{s}}\cdot\boldsymbol{w}\right). \quad (3.7)$$

where $h=e+p/\rho$ is the local enthalpy (per unit mass), $\nu=\mu/\rho$ is the kinematic viscosity and T_{I} is the interface temperature. The interested reader will find a detailed derivation of (3.2)-(3.7) in appendix A.

3.2.1 Interface conditions on velocity components

Since we build upon the framework developed for the numerical simulation of single-phase flows which decouples all velocity components, its extension to two-phase problems would require expressions for $[\boldsymbol{u}]$ and either $[\mu\nabla\boldsymbol{u}\cdot\boldsymbol{n}]$ or $[\mu\nabla\boldsymbol{u}]$ (if using a Cartesian, dimension-by-dimension approach for interface discontinuities).

The conservation of mass across the interface (3.4) may be reformulated as a jump condition on the normal velocity across the interface. Indeed, since $\dot{M}=\rho\left(\boldsymbol{u}-\boldsymbol{w}\right)\cdot\boldsymbol{n}$, we have $\boldsymbol{u}\cdot\boldsymbol{n}=\dfrac{\dot{M}}{\rho}+\boldsymbol{w}\cdot\boldsymbol{n}$ on (either side of) Γ and therefore

$$[\boldsymbol{u}\cdot\boldsymbol{n}]=\dot{M}\left[\frac{1}{\rho}\right] \quad (3.8)$$

since $\left[\dot{M}\right]=0$ and $[\boldsymbol{w}]=\boldsymbol{0}$ (interface-defined quantities). Additionally, it has been assumed in the derivation of the governing equations presented here above that the tangential velocity components are continuous across the interface[1]. Mathematically,

[1]For viscous fluids, this is required to ensure finite shear rates everywhere (see appendix A.2.2).

this translates into $[(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot \boldsymbol{u}] = \mathbf{0}$, which may be combined with (3.8) to obtain

$$[\boldsymbol{u}] = \dot{M} \left[\frac{1}{\rho}\right] \boldsymbol{n}. \tag{3.9}$$

We now consider $[\mu \nabla \boldsymbol{u}]$. Reproducing the developments by Kang, Fedkiw and Liu [15, 16], one has, using index notations and Einstein Summation Convention[2]

$$\begin{aligned}
[\mu \partial_j u_i] &= [\mu \partial_k u_i \left(\delta_{kj} - n_k n_j\right)] + [\mu \partial_k u_i \; n_k n_j] \\
&= [\mu \partial_k u_i \left(\delta_{kj} - n_k n_j\right)] + [\mu \left(\delta_{ir} - n_i n_r\right) \partial_k u_r \; n_k n_j] + [\mu n_i n_r \partial_k u_r \; n_k n_j] \\
&= [\mu \partial_k u_i \left(\delta_{kj} - n_k n_j\right)] + [\mu \left(\delta_{ir} - n_i n_r\right) \left(2E_{rk} - \partial_r u_k\right) n_k n_j] + [\mu n_i n_r \partial_k u_r \; n_k n_j]
\end{aligned}$$

for the $(i,\, j)$ component of $[\mu \nabla \boldsymbol{u}]$. Now note that the projection of (3.5) on the hyperplane normal to $\boldsymbol{n}$ reads

$$[2\mu \left(\delta_{ir} - n_i n_r\right) E_{rk} n_k] = - \left(\delta_{ir} - n_i n_r\right) \left(\partial_r \gamma + G_r\right). \tag{3.10}$$

By making use of the equality $[AB] = A^{\pm}[B] + B^{\mp}[A]$, one may also exploit

$$[\left(\delta_{jk} - n_j n_k\right) \partial_k u_i] = \left(\delta_{jk} - n_j n_k\right) \partial_k \left(\dot{M} \left[\frac{1}{\rho}\right] n_i\right). \tag{3.11}$$

which holds true by differentiating (3.9) side by side along directions that are tangential to the interface. In particular, a side result of (3.11) is

$$[\boldsymbol{n} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{n}] = [\boldsymbol{n} \cdot \underline{\boldsymbol{E}} \cdot \boldsymbol{n}] = -\kappa \dot{M} \left[\frac{1}{\rho}\right]. \tag{3.12}$$

[2] In index notations, a_i represents the Cartesian component of vector $\boldsymbol{a}$ along direction i. Einstein Summation Convention means that repeated indices implicitly represent a sum over those indices. For instance, $\boldsymbol{a} = a_i \boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is the unit vector oriented along Cartesian direction i. Similarly, $\boldsymbol{a} \cdot \boldsymbol{b} = a_i b_i$. The reader is referred to the list of symbolic operations and notations on page 217 for more details.

which results from the contraction over (i, j) of (3.11) and the incompressibility condition $\nabla \cdot \boldsymbol{u} = 0$.

When incorporating (3.10), (3.11) and (3.12) into the above expressions for $[\mu \partial_j u_i]$, one obtains[3]

$$\begin{aligned}
[\mu \partial_j u_i] \quad = \quad & \mu^{\pm} \left(\delta_{jk} - n_j n_k\right) \partial_k \left(\dot{M} \left[\frac{1}{\rho}\right] n_i \right) + [\mu] \left(\delta_{jk} - n_j n_k\right) \partial_k u_i^{\mp} \\
& - \left(\delta_{ir} - n_i n_r\right) \left(\partial_r \gamma + G_r\right) \ n_j \\
& -\mu^{\pm} \ n_j \left(\delta_{ir} - n_i n_r\right) \partial_r \left(\dot{M} \left[\frac{1}{\rho}\right] \right) - [\mu] \ n_j \left(\delta_{ir} - n_i n_r\right) \partial_r u_k^{\mp} \ n_k \\
& -\mu^{\pm} \ n_i n_j \left(\kappa \dot{M} \left[\frac{1}{\rho}\right] \right) + [\mu] \ n_i n_j n_r \partial_k u_r^{\mp} n_k.
\end{aligned} \tag{3.13}$$

If required instead of the latter[4], the corresponding expression for $[\mu \nabla \boldsymbol{u} \cdot \boldsymbol{n}]$ may be obtained, for the ith velocity component, by multiplying (3.13) with n_j, side by side, which yields

$$\begin{aligned}
[\mu \partial_j u_i n_j] \quad = \quad & - \left(\delta_{ir} - n_i n_r\right) \left(\partial_r \gamma + G_r\right) \\
& -\mu^{\pm} \left(\delta_{ir} - n_i n_r\right) \partial_r \left(\dot{M} \left[\frac{1}{\rho}\right] \right) - [\mu] \ \left(\delta_{ir} - n_i n_r\right) \partial_r u_k^{\mp} \ n_k \\
& -\mu^{\pm} \ n_i \left(\kappa \dot{M} \left[\frac{1}{\rho}\right] \right) + [\mu] \ n_i n_r \partial_k u_r^{\mp} n_k.
\end{aligned}$$

The jump conditions (3.13) express the discontinuities in viscous flux generally, for velocity component i along Cartesian direction j, as required when using a dimension-by-dimension treatment of interface discontinuities (similarly to [16, 15]). As it can be seen, solution-dependent terms are involved in the expansion (the same conclusion holds

[3]To the best of our knowledge, this expression stands as an original contribution which must be taken into account if one intends to account for stress balance across the interface, in a sharp fashion. A similar expression is derived in [134], by means of curvilinear axes local to the interface mapped back to the computational reference frame

[4]If using jump solvers for the velocity components that rely on the jump in normal flux only, instead of the jumps in all Cartesian flux components.

for $[\mu \boldsymbol{n} \cdot \nabla \boldsymbol{u}]$) which underlines the need for an iterative strategy, if one wants to capture the discontinuities implicitly.

A note about under-resolved interfaces As highlighted by (3.13), the discontinuities in viscous fluxes require an accurate and reliable local description of the interface. Indeed,

- several tangential ($\left(\delta_{ij} - n_i n_j\right) \partial_j \left(\cdot\right)$) and normal derivatives ($n_i \partial_i \left(\cdot\right)$) are involved;
- derivatives of local normal vectors may also be required in presence of a mass flux across the interface (if using a dimension-by-dimension jump solver).

Therefore, the consistent treatment of (3.13) in presence of locally under-resolved interfaces stands as a challenge that is left for future work. In presence of locally under-resolved interface, the standard evaluation of the interface normal vector by differentiation of the levelset function, i.e., $\boldsymbol{n} = \dfrac{\nabla \phi}{\left\| \nabla \phi \right\|}$, will usually lead to an inaccurate normal vector ($\mathcal{O}\left(1\right)$ error, pointing in an arbitrary direction), and inaccurate local evaluations of curvature as well.

3.2.2 Interface conditions on pressure

The projection of (3.5) onto $\boldsymbol{n}$ yields the following discontinuity in pressure

$$[p] = -\kappa\gamma - \dot{M}^2 \left[\frac{1}{\rho}\right] + 2\left[\mu \boldsymbol{n} \cdot \underline{\boldsymbol{E}} \cdot \boldsymbol{n}\right] + \boldsymbol{G} \cdot \boldsymbol{n}. \tag{3.14}$$

Alongside the discontinuity in pressure (3.14), the value of $\left[\dfrac{1}{\rho}\boldsymbol{n} \cdot \nabla p\right]$ needs to be prescribed as well when decoupling velocity and pressure in a numerical simulation engine. Though a formal derivation for $\left[\dfrac{1}{\rho}\boldsymbol{n} \cdot \nabla p\right]$ may be found in case of Stokes flows (see [135], for instance), it cannot be extended to general flow dynamics. However, when considering the conservation of momentum (3.3), one has

$$\left[\frac{\nabla p}{\rho}\right] = -\left[\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t}\right] + \left[\frac{\mu}{\rho}\nabla^2\boldsymbol{u}\right] \tag{3.15}$$

which reduces to

$$\left[\frac{\nabla p}{\rho}\right] = \left[\frac{\mu}{\rho}\nabla^2\boldsymbol{u}\right] \tag{3.16}$$

in absence of a mass transfer across the interface, since the acceleration of fluid material on either side of the interface must be continuous in that case. When allowing differentiation across the interface for the evaluation of the divergence of the intermediary velocity field within the projection step, one may show (see [26, 27]) that it introduces a singular contribution that is equivalent to the the right hand side of (3.16) (which is therefore naturally accounted for).

In the context of an entirely sharp approach, enforcing the general condition (3.15) stands as yet another difficulty numerically which would require an order of accuracy on the velocity components that is not matched even in single-phase problems. Therefore, the current work considers the simplified condition

$$\left[\frac{\boldsymbol{n}\cdot\nabla p}{\rho}\right] = 0 \tag{3.17}$$

instead.

3.3 Decoupling degrees of freedom in simulation of two-phase flows

Though pressure and velocity components are coupled across the interface as equations (3.14) shows, handling that inter-dependency between $[p]$ and $[\mu \boldsymbol{n} \cdot \underline{\boldsymbol{E}} \cdot \boldsymbol{n}]$ directly would require a discretization as a saddle-point system for all unknowns. In other words, at every time step, one would need to solve a system of the kind

$$\begin{pmatrix} \mathtt{A} & -\mathtt{G} \\ \mathtt{D} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ p \end{pmatrix} = \begin{pmatrix} \mathtt{r}\left(\boldsymbol{u}^{n}, \boldsymbol{u}^{n-1}, \boldsymbol{f}\right) \\ 0 \end{pmatrix} + \begin{pmatrix} \mathtt{j}\left(\boldsymbol{u}^{n+1}, p\right) \\ \mathtt{m}\left(\dot{M}\right) \end{pmatrix} \tag{3.18}$$

wherein $\mathtt{A}$ accounts for the (sharp) discretization of advection and implicit viscous terms, $\mathtt{G}$ and $\mathtt{D}$ are (sharp) gradient and divergence discrete operators, $\mathtt{r}$ stands for known, bulk and/or jump-related, right-hand side contributions to the (discretized) momentum equation whereas $\mathtt{m}\left(\dot{M}\right)$ accounts for interface-located mass-flux contributions to $\nabla \cdot \boldsymbol{u}$ and $\mathtt{j}\left(\boldsymbol{u}^{n+1}, p\right)$ accounts for the solution-dependent terms resulting from the incorporation of (3.13) and (3.14) into the discretization ([33] considers such an approach in two-dimensions). Beyond the need for an iterative procedure that would not be alleviated in such a case (because of $\mathtt{j}\left(\boldsymbol{u}^{n+1}, p\right)$), the implementation and resolution of such a large saddle-point systems may prove complex and prohibitive for applications in large-scale computing.

We therefore opt for a projection method approach, instead. Introduced by Chorin [107] for the calculation of single-phase incompressible flows (see [108] for a review), the projection method relies on the Helmholtz-Hodge decomposition, which states that vector fields may be split into curl-free and divergence-free components. One may thus advance the momentum equations (3.3) in time while relaxing the divergence-free constraint (3.2) and find a curl-free (correction to the) pressure gradient thereafter. It can be shown that such a strategy, also called fractional-step method, is actually equivalent to an approximation of the block LU discretization of (3.18) (see [136, 137]).

Though extensively used in numerical simulations of two-phase flows as well, its application ensuring strict consistency with (3.14) (and (3.16), in fact) requires careful consideration about whether or not $[(2)\,\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}\cdot\boldsymbol{n}]$ should be accounted for in the discontinuity of the projection variable (pressure in such a case), in light of the discretization scheme used for viscous terms and the differentiation across the interface used when evaluating $\nabla\cdot\boldsymbol{u}$ (as shown in [26]).

However, we mean to avoid any discretization across the interface (particularly undesirable in case of discontinuous velocity components). We therefore developed the following numerical strategy that intends to approach (3.18) via a fix-point iteration, if desired by the user, while decoupling velocity and pressure. The pressure gradient is taken into account as the momentum equations are advanced in time. Since the subsequent projection is numerically equivalent to an *a posteriori* pressure correction, the user may want to account for it and start the procedure over again.

3.3.1 Semi-discrete workflow per time step

At time step n, given ϕ^{n+1} and a possible estimate[5] of $\boldsymbol{u}^{\star,0}$:

0. Set p^0 as the solution to the following sharp "pressure guess" problem

$$\begin{cases} -\nabla\cdot\left(\dfrac{1}{\rho}\nabla p^0\right) = 0, \\ \left[p^0\right] = -\gamma\kappa - \dot{M}^2\left[\dfrac{1}{\rho}\right] + 2\left[\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}^{\star,0}\cdot\boldsymbol{n}\right] + \boldsymbol{G}\cdot\boldsymbol{n}, \\ \left[\dfrac{1}{\rho}\nabla p^0\cdot\boldsymbol{n}\right] = 0, \end{cases} \tag{3.19}$$

   and extrapolate the sharp solution from either side of the interface to the rest of the domain;

1. Set an iteration counter $k \leftarrow 0$ and iterate until convergence or maximum number of iterations $k_{\max}$ reached:

   (a) solve the "viscosity step": find the solution $\boldsymbol{u}^{\star,k+1}$ to

$$\begin{cases} \rho\left(\alpha\dfrac{\boldsymbol{u}^{\star,k+1}-\boldsymbol{u}_d^n}{\Delta t_n} + \beta\dfrac{\boldsymbol{u}_d^n-\boldsymbol{u}_d^{n-1}}{\Delta t_{n-1}}\right) = \mu\nabla^2\boldsymbol{u}^{\star,k+1} - \nabla p^k + \rho\boldsymbol{f} \\ \left[\boldsymbol{u}^{\star,k+1}\right] = \dot{M}\left[\dfrac{1}{\rho}\right]\boldsymbol{n} \\ \left[\mu\partial_j u_i^\star\right] = \mu^\pm\left(\delta_{jk}-n_jn_k\right)\partial_k\left(\dot{M}\left[\dfrac{1}{\rho}\right]n_i\right) + [\mu]\left(\delta_{jk}-n_jn_k\right)\partial_k u_i^{\star,\mp} \\ \qquad -\left(\delta_{ir}-n_in_r\right)\left(\partial_r\gamma + G_r\right)\ n_j \\ \qquad -\mu^\pm\ n_j\left(\delta_{ir}-n_in_r\right)\partial_r\left(\dot{M}\left[\dfrac{1}{\rho}\right]\right) - [\mu]\ n_j\left(\delta_{ir}-n_in_r\right)\partial_r u_k^{\star,\mp}\ n_k \\ \qquad -\mu^\pm\ n_in_j\left(\kappa\dot{M}\left[\dfrac{1}{\rho}\right]\right) + [\mu]\ n_in_jn_r\partial_k u_r^{\star,\mp}n_k \end{cases} \tag{3.20}$$

[5]Though not required conceptually (one could use $\boldsymbol{u}^{\star,0} = \boldsymbol{0}$), this estimate $\boldsymbol{u}^{\star,0}$ was introduced to improve the overall approach, in particular when using few iterations. Over the course of the simulation, the solver defines this estimate as a time-extrapolation of previous sharp velocity fields. By default, a linear time extrapolation of the two latest velocity fields is used.

wherein $\alpha = \dfrac{2\Delta t_n + \Delta t_{n-1}}{\Delta t_n + \Delta t_{n-1}}$ and $\beta = \dfrac{-\Delta t_n}{\Delta t_n + \Delta t_{n-1}}$ are second-order backward differentiation formula parameters; $\boldsymbol{u}_d^n$ and $\boldsymbol{u}_d^{n-1}$ are velocities at semi-lagrangian backtraced points (more details in subsection 7.2.1).

Extrapolate the sharp vector field $\boldsymbol{u}^{\star,k+1}$ from either side of the interface to the rest of the domain;

(b) find a pressure correction $\delta p^{k+1} = \left(p^{k+1} - p^k\right)$ such that $\boldsymbol{u}^{\star,k+1} - \dfrac{\Delta t_n}{\alpha\rho}\nabla\delta p^{k+1}$ is (piecewise) divergence-free. Solve the two-phase "projection step"

$$\begin{cases} -\nabla\cdot\left(\dfrac{1}{\rho}\nabla\Phi^{k+1}\right) = -\nabla\cdot\boldsymbol{u}^{\star,k+1} \\ \left[\Phi^{k+1}\right] = \dfrac{2\Delta t_n}{\alpha}\left(\left[\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}^{\star,k+1}\cdot\boldsymbol{n}\right] - \left[\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}^{\star,k}\cdot\boldsymbol{n}\right]\right), \\ \left[\dfrac{1}{\rho}\nabla\Phi^{k+1}\cdot\boldsymbol{n}\right] = 0, \end{cases} \tag{3.21}$$

wherein $\Phi^{k+1} = \dfrac{\Delta t_n}{\alpha}\delta p^{k+1}$ (Φ will be referred to as the Hodge variable). Extrapolate Φ^{k+1} from either side of the interface to the rest of the domain and define $p^{\pm,k+1} = p^{\pm,k} + \dfrac{\alpha}{\Delta t_n}\Phi^{\pm,k+1}$.

Set $k \leftarrow k+1$ and repeat from step 1a with the updated pressure, if desired.

2. Make (extrapolated) velocity fields divergence-free: $\boldsymbol{u}^{\pm,n+1} = \boldsymbol{u}^{\pm,\star,k} - \dfrac{1}{\rho^{\pm}}\nabla\Phi^{\pm,k}$ and interpolate from faces to nodes.

3. Advance time: determine the new time step Δt_{n+1}, compute the interface velocity $\boldsymbol{w}^{n+1}$ and update the levelset function by advancing (3.1) from t_{n+1} to $t_{n+2} = t_{n+1} + \Delta t_{n+1}$. Build the new grid that captures the zero-contour of ϕ^{n+2}, and satisfies the other refinement criteria. Proceed to next time step.

3.3.2 Some notes about the workflow per time step

About the termination of the fix-point iteration. By choosing $k_{\max} = 1$, the numerical approach corresponds to a fractional step method, in principle. If desired by the user (by setting $k_{\max} > 1$), the described flowchart enables fix-point iterations in order to account for pressure corrections resulting from projection steps to be accounted for when advancing the momentum equations. Enabling $k_{\max} > 1$ was found necessary in some test problems, in particular when the time step is significantly larger than with an explicit treatment of viscous terms. If $k_{\max} > 1$, the solver monitors the ratios

$$\frac{\max_{\Omega^{\pm}} \left\| \frac{1}{\rho^{\pm}} \nabla \Phi^{k+1} \right\|_{\infty}}{\max_{\Omega^{\pm}} \left\| \boldsymbol{u}^{\star, k+1} \right\|_{\infty}} \tag{3.22}$$

in either subdomain, at every iteration k. If the measure (3.22) is found below a user-defined threshold ξ (default is $\xi = 1\%$) in either subdomain, the fix-point iteration terminates even if $k < k_{\max}$. Though usually monotonically decreasing in both subdomains, this measure was often found to decrease slowly or plateau in the lighter-fluid (gas-like) subdomain on adaptive grids, with the value of the numerator in (3.22) being usually found at T-junctions, underlining a possible shortcoming of local grid resolution.

About the pressure guess. The use of a pressure guess was introduced in [32]. It allows for Laplace pressure and other interface-defined forces and momentum discontinuities to be (roughly) accounted for when advancing the momentum equations. It was found to be necessary to ensure vanishing parasitic currents in benchmark test cases. It also has the advantage of improving stability of the subsequent two-phase projection step(s).

About the stability of the projection step. The numerical stability of the projection step is a key requirement to ensure overall robustness in numerical simulations. The specific discretization introduced for single-phase flows (see [55]) ensures stability by constructing a mimetic equivalence with continuum operators for the discrete gradient of cell-sampled fields and the discrete divergence of face-sampled vector fields. This mimetic equivalence ensures that the projection step is not responsible for creating spurious (kinetic) energy. While extending the reasoning for such discrete operators to strictly sharp, two-phase problems is an open question, one may observe that their construction basically relies on a cell-based, finite-volume approach. Therefore, it seems preferable (regarding stability) to use a finite-volume approach as well for solving the two-phase projection step problems, as well.

About the interface conditions in the projection step. The interface condition $\left[\frac{1}{\rho}\nabla\Phi^{k+1}\cdot\boldsymbol{n}\right] = 0$ is required to ensure $[\boldsymbol{u}^{n+1}\cdot\boldsymbol{n}] = \dot{M}\,[1/\rho]$. Indeed, since the viscosity step enforces $\left[\boldsymbol{u}^{\star,k+1}\cdot\boldsymbol{n}\right] = \dot{M}\,[1/\rho]$, one has

$$\left[\boldsymbol{u}^{n+1}\cdot\boldsymbol{n}\right] = \left[\boldsymbol{u}^{\star,k+1}\cdot\boldsymbol{n}\right] - \left[\frac{1}{\rho}\nabla\Phi^{k+1}\cdot\boldsymbol{n}\right] = \dot{M}\,[1/\rho]\,,$$

as desired.

The interface condition $\left[\Phi^{k+1}\right] = \frac{2\Delta t_n}{\alpha}\left(\left[\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}^{\star,k+1}\cdot\boldsymbol{n}\right] - \left[\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}^{\star,k}\cdot\boldsymbol{n}\right]\right)$ is designed to account for the latest known velocity field in the pressure discontinuity. Indeed, after K fir-point iterations, the pressure discontinuity reads

$$\begin{aligned}\left[p^{K}\right] &= \left[p^{0}\right] + \sum_{j=1}^{K}\left[\delta p^{j}\right] = \left[p^{0}\right] + \frac{\alpha}{\Delta t_n}\sum_{j=1}^{K}\left[\Phi^{j}\right] \\ &= -\gamma\kappa - \dot{M}^{2}\left[\frac{1}{\rho}\right] + 2\left[\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}^{\star,K}\cdot\boldsymbol{n}\right] + \boldsymbol{G}\cdot\boldsymbol{n}. \end{aligned} \tag{3.23}$$

About the difference between $\boldsymbol{u}^\star$ and $\boldsymbol{u}^{n+1}$. Though the procedure described in subsection 3.3.1 intends to reach minimal differences between $\boldsymbol{u}^\star$ and $\boldsymbol{u}^{n+1}$ eventually, it may be truncated after a finite number of $k_{\max}$ iterations in practice. The final divergence-free projection $\boldsymbol{u}^{n+1} = \boldsymbol{u}^{\star,k_{\max}} - \frac{1}{\rho}\nabla\Phi^{k_{\max}}$ may thus alter the following interface conditions enforced during the viscosity step:

- the continuity of tangential fluid velocities may be altered: $(\underline{\boldsymbol{\delta}} - \boldsymbol{nn})\cdot\left[\frac{1}{\rho}\nabla\Phi^{k_{\max}}\right] = \boldsymbol{0}$ is not enforced[6], this may affect the continuity of fluid velocity, tangentially to the interface;
- a similar argument holds for the balance of stress across the interface. In that case, the projection-related terms polluting the interface conditions are second-order derivatives of Φ (thus they cannot be evaluated and accounted for accurately in our framework).
- by substituting $\boldsymbol{u}^{\star,k_{\max}} = \boldsymbol{u}^{n+1} + \frac{1}{\rho}\nabla\Phi^{k_{\max}}$ into the discretization of the viscous step, one finds that the finalized pressure field should read $p^{k_{\max}} - \mu\nabla\cdot\boldsymbol{u}^{\star,k_{\max}}$. This term alters all boundary and interface conditions enforced on the pressure.

In order to alleviate these sources of uncertainty (in particular related the continuity of tangential fluid velocities), an alternative definition of the measure (3.22) may be considered. For instance, one may evaluate how much the projection step alters interface conditions by monitoring

$$\max_{\Omega_{\Gamma\pm\varepsilon}} \max\left(\left|\frac{\partial_x\Phi^{k+1}}{\rho^\pm\boldsymbol{e}_x\cdot\boldsymbol{u}^{\star,k+1}}\right|, \left|\frac{\partial_y\Phi^{k+1}}{\rho^\pm\boldsymbol{e}_y\cdot\boldsymbol{u}^{\star,k+1}}\right|, \left|\frac{\partial_z\Phi^{k+1}}{\rho^\pm\boldsymbol{e}_z\cdot\boldsymbol{u}^{\star,k+1}}\right|\right) \tag{3.24}$$

instead of (3.22), and where $\Omega_{\Gamma\pm\varepsilon}$ represents the part of the computational domain layering the interface Γ with thickness $\varepsilon > \Delta x$. By enforcing (3.24) to fall below a user-

[6]and *cannot* be enforced numerically in a single pass, except if using the method from [16].

defined threshold, the continuity of tangential velocities across Γ is ensured within the same (relatively) tolerance. The consideration of such alternative convergence measures is left for future work.

3.4 Summary

In this chapter, we have outlined and discussed the governing equations for two-phase incompressible viscous flows. In particular, the interface conditions coupling the flow dynamics across the interface have been discussed and analyzed in light of the elaboration of an appropriate numerical strategy decoupling velocity components and pressure. The balance of stress across the interface reveals that, though decoupled far away from the interface by virtue of the incompressibility condition, the velocity components are coupled across the interface via solution-dependent jump conditions on viscous fluxes across the interface. Their consideration in the context of an implicit treatment of viscous terms thus requires an iterative strategy for elliptic interface problems involving face-sampled vector fields.

In chapter 4, we introduce such a strategy for elliptic interface problems involving scalar fields, and show that it is capable of recovering absolute convergence in gradients when using a dimension-by-dimension numerical approach [16]. This strategy is then extended to problems of interest when treating viscous terms implicitly with face-sampled velocity components, in chapter 5. In chapter 6, we present the numerical tools used to solve elliptic interface problems for cell-sampled scalar fields (i.e., for the pressure guess and projection step) using a finite-volume approach. Finally, in chapter 7, we assess the performance and present results for the two-phase flow solver outlined in this chapter, assembling the computational tools described in the subsequent chapters.

Chapter 4

A strategy for scalar elliptic interface problems with solution-dependent flux discontinuities

4.1 Introduction

We consider a computational domain Ω divided into two distinct regions by a smooth interface Γ that we define as the zero-level set of a differentiable function ϕ. We have two distinct regions

$$\Omega^{-} = \{\boldsymbol{x} \in \Omega \mid \phi(\boldsymbol{x}) \leq 0\}, \tag{4.1}$$

$$\Omega^{+} = \{\boldsymbol{x} \in \Omega \mid \phi(\boldsymbol{x}) > 0\}. \tag{4.2}$$

Throughout the rest of this chapter, we will consider $\phi\left(\boldsymbol{x}\right)$ to be the exact (signed) distance from $\boldsymbol{x}$ to Γ, which gives a straightforward expression for the unit vector $\boldsymbol{n}$ normal to Γ, as $\boldsymbol{n} = \nabla\phi$ (pointing toward Ω^{+}). We are interested in problems of the category ($\beta > 0$)

$$\begin{cases} -\nabla\cdot\left(\beta\nabla u\right) &= f, \\ [u] &= a, \\ [\beta\boldsymbol{n}\cdot\nabla u] &= b, \end{cases} \tag{4.3}$$

where $[q]$ represents a discontinuity at the interface Γ in the quantity q, defined as $[q] = q^{+} - q^{-}$, where $q^{\pm}\left(\boldsymbol{x}\right) = \lim_{\varepsilon\to 0,\,\varepsilon>0} q\left(\boldsymbol{x}\pm\varepsilon\boldsymbol{n}\right),\ \forall\boldsymbol{x}\in\Gamma$. The reader should understand a and b as smooth extensions of the relevant fields defined on the co-dimension one interface Γ, whenever implicitly considered as scalar fields defined over the whole domain Ω.

The problems that fall into the above category are ubiquitous in engineering and applied physics. A non-exhaustive range of examples includes: electrostatics problems where the potential must be continuous across the interface as well as the normal electric displacement field (in the absence of interface electric charge); problems involving heat conduction across different materials where the temperature must be continuous across the interface as well as the normal heat flux (in the absence of a surface heat source); the Young-Laplace equation for immiscible fluids and several others. As a consequence, a plethora of methods have been developed over the last few decades to address problems like (4.3) numerically.

The technical difficulties associated with the interface conditions $[u] = a$ and $[\beta\boldsymbol{n}\cdot\nabla u] = b$ in (4.3) can be significantly alleviated by using an interface-conforming layout for the discrete unknowns, hence an unstructured grid. It is thus natural to find early successes exploiting Finite Elements Methods (FEM), see for instance [138, 139]. More recently, this very same idea has been exploited successfully using a Finite Volume (FV) approach with interface-tracking Voronoi diagrams (the VIM for Voronoi Interface

Method), leading to second-order accurate results[140]. Given their ability to naturally capture the relevant interface conditions in (4.3), these strategies may be considered first-choice methods when it comes to scalar problems with static interfaces.

However, the construction of such a conforming mesh adds complexity and significant workload to the numerical procedures, especially in case of moving interfaces and/or in case of several equations like (4.3) coupling unknowns that are not collocated, *e.g.*, the velocity components in a Marker-And-Cell layout. *Capturing* the effects of an embedded interface using a non-conforming mesh is preferable in such cases.

Still in the FEM community, methods like the eXtended Finite Element Method (XFEM) and Discontinuous Galerkin (DG) Methods have proved successful in capturing interface conditions by enriching the approximation function space locally for the elements crossed by the interface (see [141, 142, 143, 144, 145] and the references therein).

Regarding methods on Cartesian grids, the Immersed Interface Method (IIM) by LeVeque and Li [146, 147, 148] was the first successful approach to capture such jump conditions using a finite difference approach. The IIM builds upon second-degree Taylor expansions on both sides of the interface, which are matched through the given jump conditions and their local differentiation along the interface. The IIM achieves second-order accuracy in infinity norm, at the expense of solving a non-symmetric linear system of equations, with (rather arbitrarily chosen) extra discretization point(s) in the stencil, and a fairly complex and challenging implementation, especially in a three-dimensional setting[149].

In cases where $[\beta] = 0$, and $[u] = 0$, simplified approaches have been suggested. For instance, such problems can be reformulated as regular Poisson equations in the entire computational domain $\Omega = \Omega^- \bigcup \Omega^+$ with singular source terms distributed on the co-dimension one interface Γ, see [150, 151, 152].

In order to address more general problems on Cartesian grids, the idea of constructing a sharp numerical method based on well-defined ghost values across the interface soon became the mainstream approach. This strategy overcomes most numerical challenges associated with the intrinsic sharp discontinuities of the solution by virtually expanding the subdomains where the solution can be considered locally smooth and differentiable at the grid scale. For problems where $[\beta] = 0$ for instance, the approach becomes particularly straightforward: well-defined ghost values can be defined as direct algebaric expressions involving the value of b, as developed in [153] for solidification problems for instance. A generalization to problems with $[\beta] \neq 0$ using a finite volume approach was developed in [154], building upon techniques and analyses from [155]. This generalization leverages local, one-sided, least-square field reconstructions to determine appropriate interpolation weights for the solution-dependent terms that come into play. These weights are then incorporated within the discretization scheme so that appropriate and accurate ghost values are implicitly taken into account at all required grid nodes. This methodology achieves second-order accuracy for the solution and second-order accurate gradients in infinity norm but sacrifices the symmetry, and possibly the positive definiteness, of the linear system to be solved. Nevertheless, [154] showed that the condition number is bounded for any ratios of β^+/β^- and that the linear system can be solved efficiently.

Along the same lines, the Correction Function Method (CFM) [156] was shown to achieve up to 4th order accurate results for problems with constant β throughout Ω (and thus $[\beta] = 0$). The idea consists of defining a *correction function* $D = u^+ - u^-$ where $u^\pm$ is an extrapolation of u from $\Omega^\pm$ to Ω, *i.e.* across the interface Γ. At grid nodes where the standard discretization stencil crosses the interface, the node values of D are then used to determine well-defined ghost-values $u^\pm = u \pm D$. The function D is shown to be governed by an elliptic Cauchy-problem for D in Ω, that is solved *locally* by finding the *local* minimizer of a functional in a least-square sense. The CFM has been further developed to address problems with piecewise constant β but $[\beta] \neq 0$ in [157] by combining it with a Boundary Integral Method based on the early work by Mayo [158]. Although claimed to straightforwardly extend in a three-dimensional framework, no three-dimensional examples are shown in [156, 157].

Similarly, the Matched Interface and Boundary (MIB) method by Zhou and al. [159] uses an extended number of ghost values across the interface in order to achieve higher orders of accuracy. These ghost values are extended recursively so that interpolants of increasing degree match the jump conditions with increasing accuracy. The linear system is no longer symmetric nor diagonally dominant. Up to sixth-order accurate results are reported in cases where the interface allows the use of such extended stencils. However, second-order accuracy seems to be the barrier otherwise, *i.e.*, for general irregular interfaces whose details are captured by a only few grid nodes. Alternatively, one could add extra unknowns at intersections between grid lines and the interface Γ. This has been used in [160] and naturally prevents the coupling of grid nodes *across* Γ. However, the additional equations for the unknowns located on the interface use extended stencils (wherever allowed by the interface) that break the symmetric positive definiteness of the linear system.

The idea of using ghost values across an interface can be traced back to the Ghost Fluid Method introduced in [17] as a way to capture Ranking-Hugoniot relations across a strong shock. Building upon this idea, Liu, Fedkiw and Kang developed an interface-capturing method for problems like (4.3) on Cartesian grids in [16]. We will refer to this method as the GFM hereafter. Contrary to the other jump-capturing schemes mentioned here above, the GFM captures sharp discontinuities on a regular Cartesian grid through the resolution of a well-behaved symmetric positive-definite (SPD) linear system of equations. Moreover, it builds on the smallest possible discretization stencil, enhancing its robustness with respect to (close to) under-resolved interfaces. These features make the GFM very appealing and highly suited for large-scale simulations because of its ease of implementation in a distributed computing framework, for stability reasons and/or for applications requiring robustness with respect to the interface geometry. As a consequence, the GFM has been the focus of considerable academic attention over the last two decades, especially within the community of sharp interface methods for computational fluid dynamics and multiphase flows, see [50, 161, 15, 162, 163, 44, 164, 43, 48]. Its convergence has been studied in [165] by means of a weak formulation of (4.3), which was exploited in turn in [166] along with a triangulation of the Cartesian grid to develop a second-order accurate version.

Nevertheless, the GFM suffers from the lack of convergence for the gradient of the solution or, more precisely, for the flux $\beta\nabla u$. The latter may pose serious accuracy issues, especially in the context of projection methods in multiphase incompressible flows, because the accuracy of the updated incompressible fluid velocity is directly related to the flux $\beta\nabla u$.

In this chapter, we introduce the xGFM: a union between e$\underline{\text{x}}$tension techniques and the $\underline{\text{GFM}}$ that intends to recover absolutely convergent gradients in the computational domain. The structure of the linear system is *not* modified; only the right-hand side is iteratively updated. First, we introduce an interpretation of the GFM as two distinct Poisson problems coupled numerically through interface values, in a one-dimensional setting. Then, we underline an inconsistency in the GFM for multi-dimensional problems and we introduce an iterative approach to solve that problem. Illustrations are provided in two and three spatial dimensions to illustrate the applicability and the robustness of the method.

4.2 An interpretation of the GFM

When it comes to solving problems like (4.3), the robustness of the numerical method is essential if strong discontinuities across Γ are expected (*e.g.*, very large or very small values of β^+/β^-). In such a case, an efficient technique consists of decoupling the problems in Ω^+ and in Ω^- as much as possible. For instance, assuming that the function u_Γ^- such that $u_\Gamma^-(\boldsymbol{x}) = \lim_{\varepsilon\to 0,\, \varepsilon>0} u(\boldsymbol{x}-\varepsilon\boldsymbol{n})$, $\forall \boldsymbol{x} \in \Gamma$ is known beforehand, then (4.3) is equivalent to solving two decoupled Poisson equations in irregular domains: a first one for u in Ω^- with Dirichlet boundary condition u_Γ^- on Γ, a second one for u in Ω^+ with Dirichlet boundary condition $u_\Gamma^+ = \left(u_\Gamma^- + a\right)$ on Γ. Several well-known methods can be found in the literature for solving such Poisson equations with Dirichlet boundary conditions on an irregular domain using Cartesian grids, see e.g. [167, 168, 169, 170].

For instance, this strategy was recently exploited and developed in [171] where the superconvergent Shortley-Weller [167] was used in the separate subdomains with matching conditions through the interface. This was shown to yield second-order accurate results for the solution *and* for its gradient, as expected. However, such quality results come with a complexity of implementation and a computational cost that would very likely prevent large-scale scientific computing applications. Indeed, the original nonsymmetry of Shortley-Weller's linear system is exacerbated by the addition of at least one (resp. three) extra discretization node(s) in the stencil in two (resp. three) dimensions in the best-case scenario; under-resolved geometries require to include up to (at least) third-degree neighbors in the under-resolved Cartesian direction. On top of the costly resolution of nonsymmetric and non positive-definite linear systems, this would also set a requirement onto the layers of ghost grid nodes to cover at least three computational cells (possibly more depending on the resolution of the geometry) in a distributed computing framework, which would significantly increase the inter-process communications and limit its strong scalability. Besides, the behavior of the method is not clear regarding grid node that would fall too close to the interface and the corresponding divisions by very small numbers, if not zero[1].

Following the same idea, we show that the GFM is conceptually an equivalent strategy, making use of the second-order accurate symmetric scheme introduced by Gibou [170] in the distinct regions Ω^+ and Ω^- instead of the Shortley-Weller method[167]. For standard Poisson problems, this alternative building brick alleviates the above problems (one obtains an SPD linear system, optimal compactness, no division by arbitrarily small number) but sacrifices the second-order accuracy for the gradient of the solution to first-order. However, this indicates that first-order accurate gradients are theoretically reachable by slightly modifying the GFM.

For the sake of clarity, we show the reasoning in one spatial dimension. The extension to multi-dimensional problems is treated in a dimension-by-dimension fashion: it is discussed thereafter and unravels the origin of the issue preventing convergence of the gradients in the GFM. We consider the problem

$$\begin{cases} -\dfrac{\mathrm{d}}{\mathrm{d}x}\left(\beta\dfrac{\mathrm{d}u}{\mathrm{d}x}\right) & = f, \\ [u] & = a, \\ \left[\beta\dfrac{\mathrm{d}u}{\mathrm{d}x}\right] & = b, \end{cases} \tag{4.4}$$

for $x \in (x_{\min}, x_{\max})$, the boundary values $u(x_{\min})$ and $u(x_{\max})$ are known. We consider

[1]While the corresponding fix for standard Poisson problems is fairly straightforward (simply set the local solution value to the known Dirichlet-boundary value), it cannot be transposed to problems like (4.3). This issue is not discussed in [171].

the values of u, ϕ, f, a and b to be sampled at points[2]

$$x_i = x_{\min} + ih, \quad h = \frac{x_{\max} - x_{\min}}{N}, \quad i = \{0, 1, \ldots, N-1, N\}. \tag{4.5}$$

We use the notation $\zeta_j = \zeta(x_j), \forall j \in \{0, 1, \ldots, N-1, N\}$ for any field ζ. Since the scheme uses a three-point stencil, we call a grid node $0 < i < N$ *regular*, when ϕ_{i-1}, ϕ_i and ϕ_{i+1} are either all non-positive, or all strictly positive; otherwise, the grid node i is called a *jump* node. Figure 4.1 illustrates the general case of a jump node i.

For a regular grid node j, the finite difference discretization is the standard 3-point discretization

$$\frac{\beta^{\pm}(u_j - u_{j-1}) - \beta^{\pm}(u_{j+1} - u_j)}{h^2} = f_j, \tag{4.6}$$

where the appropriate superscript for β is chosen based on the local sign of ϕ_j.

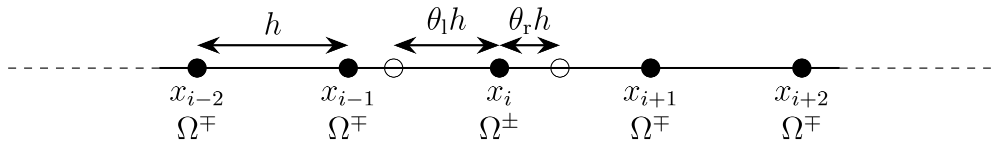


Figure 4.1: general illustration for a *jump* node i in a one-dimensional framework. The interface, defined by the set of points where $\phi = 0$, crosses the stencil centered in x_i at a distance $\theta_{\mathrm{l}}h$ (resp. $\theta_{\mathrm{r}}h$) to the left (resp. right) of grid node i (θ_{l} and θ_{r} are in [0, 1]).

Let us now consider a jump node i, and let us assume first that the value of u^- is known (and thus $u^+ = u^- + a$ is known as well) at points x_{I} such $\phi(x_{\mathrm{I}}) = 0$. In order to consider all possible scenarii, we consider the most general case such that both neighbor nodes of x_i are jump nodes as well, as illustrated in Figure 4.1 (the case where only one neighbor is in the other domain is a straightforward simpler configuration). In the following, all upper superscripts are associated with $\phi_{i-1} \leq 0$, $\phi_i > 0$ and $\phi_{i+1} \leq 0$, while

[2]For clarity purposes, the diffusion coefficients $\beta^{\pm}$ are assumed piecewise constant in this chapter so that corresponding indices can be omitted (this is not a restriction of the method, though).

lower superscripts are associated with $\phi_{i-1} > 0$, $\phi_i \leq 0$, and $\phi_{i+1} > 0$.

In such a case, assuming momentarily that $0 < \theta_\mathrm{l} < 1$ and $0 < \theta_\mathrm{r} < 1$, the application of the symmetric scheme [170] for nodes $i-1$, i and $i+1$ yields

$$\frac{\beta^{\mp}\dfrac{u_{i-1}-u_{i-2}}{h} - \beta^{\mp}\dfrac{u^{\mp}_{i-\theta_\mathrm{l}}-u_{i-1}}{(1-\theta_\mathrm{l})\,h}}{h} = f_{i-1}, \tag{4.7a}$$

$$\frac{\beta^{\pm}\dfrac{u_{i}-u^{\pm}_{i-\theta_\mathrm{l}}}{\theta_\mathrm{l} h} - \beta^{\pm}\dfrac{u^{\pm}_{i+\theta_\mathrm{r}}-u_{i}}{\theta_\mathrm{r} h}}{h} = f_{i}, \tag{4.7b}$$

$$\frac{\beta^{\mp}\dfrac{u_{i+1}-u^{\mp}_{i+\theta_\mathrm{r}}}{(1-\theta_\mathrm{r})\,h} - \beta^{\mp}\dfrac{u_{i+2}-u_{i+1}}{h}}{h} = f_{i+1}. \tag{4.7c}$$

In (4.7), the values of $u^{\pm}_{i-\theta_\mathrm{l}}$ and $u^{\pm}_{i+\theta_\mathrm{r}}$, which have been assumed to be known beforehand, are not arbitrary: they must be consistent with the interface condition $\left[\beta\dfrac{\mathrm{d}u}{\mathrm{d}x}\right] = b$. Therefore, one can write

$$\beta^{\pm}\frac{u_{i}-u^{\pm}_{i-\theta_\mathrm{l}}}{\theta_\mathrm{l} h} - \beta^{\mp}\frac{u^{\mp}_{i-\theta_\mathrm{l}}-u_{i-1}}{(1-\theta_\mathrm{l})\,h} = \pm b_{i-\theta_\mathrm{l}}, \tag{4.8a}$$

$$\beta^{\mp}\frac{u_{i+1}-u^{\mp}_{i+\theta_\mathrm{r}}}{(1-\theta_\mathrm{r})\,h} - \beta^{\pm}\frac{u^{\pm}_{i+\theta_\mathrm{r}}-u_{i}}{\theta_\mathrm{r} h} = \mp b_{i+\theta_\mathrm{r}}, \tag{4.8b}$$

which yields

$$u^{\mp}_{i-\theta_\mathrm{l}} = \frac{(1-\theta_\mathrm{l})\,\beta^{\pm}}{\tilde{\beta}_{i-\theta_\mathrm{l}}}u_i + \frac{\theta_\mathrm{l}\beta^{\mp}}{\tilde{\beta}_{i-\theta_\mathrm{l}}}u_{i-1} \mp \frac{\theta_\mathrm{l}(1-\theta_\mathrm{l})\,h}{\tilde{\beta}_{i-\theta_\mathrm{l}}}b_{i-\theta_\mathrm{l}} \mp \frac{\beta^{\pm}(1-\theta_\mathrm{l})}{\tilde{\beta}_{i-\theta_\mathrm{l}}}a_{i-\theta_\mathrm{l}}, \tag{4.9a}$$

$$u^{\mp}_{i+\theta_\mathrm{r}} = \frac{(1-\theta_\mathrm{r})\,\beta^{\pm}}{\tilde{\beta}_{i+\theta_\mathrm{r}}}u_i + \frac{\theta_\mathrm{r}\beta^{\mp}}{\tilde{\beta}_{i+\theta_\mathrm{r}}}u_{i+1} \pm \frac{\theta_\mathrm{r}(1-\theta_\mathrm{r})\,h}{\tilde{\beta}_{i+\theta_\mathrm{r}}}b_{i+\theta_\mathrm{r}} \mp \frac{\beta^{\pm}(1-\theta_\mathrm{r})}{\tilde{\beta}_{i+\theta_\mathrm{r}}}a_{i+\theta_\mathrm{r}}, \tag{4.9b}$$

where

$$\tilde{\beta}_{i-\theta_\mathrm{l}} = (1-\theta_\mathrm{l})\,\beta^{\pm} + \theta_\mathrm{l}\beta^{\mp} \qquad \text{and} \qquad \tilde{\beta}_{i+\theta_\mathrm{r}} = (1-\theta_\mathrm{r})\,\beta^{\pm} + \theta_\mathrm{r}\beta^{\mp}. \tag{4.10}$$

Substituting (4.9) into (4.7), we have

$$\beta^{\mp}\frac{u_{i-1}-u_{i-2}}{h^2}-\hat{\beta}_{i-1/2}\frac{u_i-u_{i-1}}{h^2}=f_{i-1}\mp\hat{\beta}_{i-1/2}\left(\frac{\theta_{\mathrm{l}}b_{i-\theta_{\mathrm{l}}}}{\beta^{\pm}h}+\frac{a_{i-\theta_{\mathrm{l}}}}{h^2}\right), \tag{4.11a}$$

$$\begin{aligned}\hat{\beta}_{i-1/2}\frac{u_i-u_{i-1}}{h^2}-\hat{\beta}_{i+1/2}\frac{u_{i+1}-u_i}{h^2}&=f_i\pm\hat{\beta}_{i-1/2}\left(-\frac{(1-\theta_{\mathrm{l}})\,b_{i-\theta_{\mathrm{l}}}}{\beta^{\mp}h}+\frac{a_{i-\theta_{\mathrm{l}}}}{h^2}\right)\\&\quad\pm\hat{\beta}_{i+1/2}\left(\frac{(1-\theta_{\mathrm{r}})\,b_{i+\theta_{\mathrm{r}}}}{\beta^{\mp}h}+\frac{a_{i+\theta_{\mathrm{r}}}}{h^2}\right),\end{aligned} \tag{4.11b}$$

$$\hat{\beta}_{i+1/2}\frac{u_{i+1}-u_i}{h^2}-\beta^{\mp}\frac{u_{i+2}-u_{i+1}}{h^2}=f_{i+1}\mp\hat{\beta}_{i+1/2}\left(-\frac{\theta_{\mathrm{r}}b_{i+\theta_{\mathrm{r}}}}{\beta^{\pm}h}+\frac{a_{i+\theta_{\mathrm{r}}}}{h^2}\right), \tag{4.11c}$$

where

$$\hat{\beta}_{i-1/2}=\frac{\beta^{\mp}\beta^{\pm}}{\tilde{\beta}_{i-\theta_{\mathrm{l}}}}\qquad\text{and}\qquad\hat{\beta}_{i+1/2}=\frac{\beta^{\pm}\beta^{\mp}}{\tilde{\beta}_{i+\theta_{\mathrm{r}}}}, \tag{4.12}$$

leading to a closed system of linear equations.

The approach leading to the above scheme (4.11) builds upon a sharp treatment of the interface Γ and the use of an appropriate method in the distinct regions that Γ separates in Ω (see equations (4.6) and (4.7)). The resulting scheme (4.11) is exactly the GFM and the corresponding linear system to be solved is *symmetric and positive definite.*

Remark. *As can be seen from (4.7), the cases such that $\theta_{\mathrm{l}}=0$, $\theta_{\mathrm{l}}=1$, $\theta_{\mathrm{r}}=0$ or $\theta_{\mathrm{r}}=1$ lead to ill-defined terms. In the context of solving Poisson problems on irregular domains, a common fix consists of disregarding unknowns associated with grid nodes that are too close to the interface and imposing the relevant Dirichlet condition instead. However, no such ill-defined terms appear in (4.11) as a result of the algebraic calculations from (4.8) and (4.9), which makes the method intrinsically robust with respect to the location of the interface.*

It is worth pointing out that an approximation of the derivative of u at grid nodes x_i can be found, consistently with the underlying framework from equations (4.7). Indeed, one can use

$$\begin{aligned}\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} &\simeq \frac{\theta_\mathrm{l}\dfrac{u^\pm_{i+\theta_\mathrm{r}}-u_i}{\theta_\mathrm{r}h}+\theta_\mathrm{r}\dfrac{u_i-u^\pm_{i-\theta_\mathrm{l}}}{\theta_\mathrm{l}h}}{\theta_\mathrm{l}+\theta_\mathrm{r}} \\ &\simeq \frac{\theta_\mathrm{l}\left(\dfrac{\beta^\mp\left(u_{i+1}\pm a_{i+\theta_\mathrm{r}}-u_i\right)\pm\left(1-\theta_\mathrm{r}\right)hb_{i+\theta_\mathrm{r}}}{h\tilde{\beta}_{i+\theta_\mathrm{r}}}\right)}{\theta_\mathrm{l}+\theta_\mathrm{r}} \\ &\qquad +\frac{\theta_\mathrm{r}\left(\dfrac{\beta^\mp\left(u_i-u_{i-1}\mp a_{i-\theta_\mathrm{l}}\right)\pm\left(1-\theta_\mathrm{l}\right)hb_{i-\theta_\mathrm{l}}}{h\tilde{\beta}_{i-\theta_\mathrm{l}}}\right)}{\theta_\mathrm{l}+\theta_\mathrm{r}},\end{aligned} \tag{4.13}$$

if $\theta_\mathrm{l}\neq 0$ or $\theta_\mathrm{r}\neq 0$. The case where both $\theta_\mathrm{l}=0$ and $\theta_\mathrm{r}=0$ would correspond to $\phi_i=0$, $\phi_{i-1}>0$ and $\phi_{i+1}>0$, *i.e.*, either an under-resolved case or $\Gamma=\{x_i\}$ in one dimension. In a higher dimension setting however, such a configuration can be found when considering the evaluation of the partial derivative of u along a direction locally tangent to Γ, at the point of tangency. In such a case, we modify (4.13) slightly as

$$\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} \simeq \frac{1}{\beta^\pm}\left(\pm b_i+\beta^\mp\frac{u_{i+1}-u_{i-1}}{2h}\right). \tag{4.14}$$

Note that expressions (4.13) and (4.14) make use of points *on either side of the interface* Γ, and of the values of a and b. In particular, it is noteworthy from the two expressions that an error $\mathcal{O}\left(1\right)$ in b would prevent convergence for the approximation of the derivative of u.

4.3 Recovering consistency for higher dimension problems

The results from section 4.2 can be extended to multi-dimensional problems in a straightforward dimension-by-dimension fashion. However, this strategy actually leads to extra degrees of freedom, since all (Cartesian) components of $[\beta \nabla u]$ can be imposed independently of each other. Conceptually, the resulting numerical method can be thought of as a computational tool to solve problems of the category

$$\begin{cases} -\nabla \cdot (\beta \nabla u) & = f, \\ [u] & = a, \\ [\beta \nabla u] & = \boldsymbol{c}, \end{cases} \tag{4.15}$$

where $\boldsymbol{c}$ is a vector field defined on Γ.

Noticeably, problems like (4.15) are ill-posed unless $\boldsymbol{c}$ satisfies some problem-dependent consistency conditions. Nevertheless, the above strategy is actually consistent with (4.15) and does not encounter any obstacle at the discrete level: the linear system to be solved is independent of the components of $\boldsymbol{c}$ (which assign the values of the right-hand side only, see (4.11)), and therefore it is always SPD.

In fact, the GFM [16] reconciles (4.15) and (4.3) by setting

$$\boldsymbol{c} = b\boldsymbol{n}, \tag{4.16}$$

where $\boldsymbol{n}$ is the unit vector normal to Γ (pointing toward Ω^+). Although the latter has been shown adequate to ensure the convergence of u, it does not satisfy elementary consistency conditions at the interface since it inherently assumes that $(\underline{\boldsymbol{\delta}} - \boldsymbol{nn})\cdot[\beta\nabla u] = \boldsymbol{0}$, where $(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot \boldsymbol{v} = \boldsymbol{v} - (\boldsymbol{v} \cdot \boldsymbol{n})\, \boldsymbol{n}$ represents the projection of $\boldsymbol{v}$ onto the hyperplane of unit normal vector $\boldsymbol{n}$, *i.e.*, onto the hyperplane tangent to Γ. In general, neglecting $(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot [\beta \nabla u]$ would lead to $\mathcal{O}(1)$ errors in the components of $[\beta \nabla u]$, thus impeding the convergence of the gradient of u. The fact that the GFM neglects the tangential component(s) of $[\beta \nabla u]$ was pointed out by the authors themselves in [16]. However, they focused their attention on the accuracy of the solution u only. In that context, they showed that (4.16) was sufficient to ensure accurate results for u, which marked the birth of the first (and only) SPD interface-capturing accurate solver on cartesian grids. In fact, (4.16) can even be shown to hold true in some specific applications[3], making the GFM the most appropriate tool to use in such cases.

The present work presents a more general reconciliation relation than (4.16), which in turn ensures the convergence of the flux $\beta \nabla u$. First note that for any scalar fields p and q, one has

$$[pq] = \left(\alpha p^- + (1-\alpha)\, p^+\right) [q] + \left(\alpha q^+ + (1-\alpha)\, q^-\right) [p] \tag{4.17}$$

for any scalar α, as developed in [43, 15]. Using such a decomposition with $\alpha = 0$ and $\alpha = 1$ respectively, and making use of $[u] = a$, we obtain

$$\begin{aligned} (\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot [\beta \nabla u] &= \beta^+ (\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot \nabla a + [\beta] \left((\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot \nabla u\right)^- && (4.18) \\ &= \beta^- (\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot \nabla a + [\beta] \left((\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot \nabla u\right)^+ && (4.19) \end{aligned}$$

[3]For instance, when considering the sharp numerical simulation of incompressible, inviscid and immiscible two-phase flows without mass transfer, the continuity of material acceleration of fluid particles across the interface implies that $\left[\frac{1}{\rho}\nabla p\right] = \boldsymbol{0}$ can be used in the projection step, instead of $\left[\frac{1}{\rho}\boldsymbol{n} \cdot \nabla p\right] = 0$ (where p and ρ are the pressure and mass density, respectively). This does not hold true in case of viscous flows and/or in presence of mass transfer across the interface (as in phase change problems).

which is non-zero in general. Therefore, the crux of the present method is to use either

$$\boldsymbol{c} = b\boldsymbol{n} + \beta^{+}\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla a + [\beta]\left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla u\right)^{-} \tag{4.20}$$

or

$$\boldsymbol{c} = b\boldsymbol{n} + \beta^{-}\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla a + [\beta]\left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla u\right)^{+} \tag{4.21}$$

instead of (4.16) in order to ensure consistency between the formulations (4.15), which is compliant with the numerical discretization, and the mathematically well-posed problem (4.3). Numerical tests seem to indicate that (4.20) is more well-behaved than (4.21) with respect to accuracy and to the convergence rate of the iterative method when $\beta^{-} > \beta^{+}$, and vice versa. In the rest of this section, we introduce the procedure when using (4.20). The procedure for (4.21) is a straightforward equivalent and, in fact, the implemented solver switches from one to another depending on the values of β^{-} and β^{+}.

The main bottleneck with such a strategy is the dependency of $\boldsymbol{c}$ on the solution itself through the term $[\beta]\left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla u\right)^{-}$. Since that dependency is linear in ∇u, an appropriate discretization of this term could be included within the linear system to be solved, conceptually. However, that maneuver would require an undesirable extension of the discretization stencil close to the interface and, most importantly, it would very likely break its symmetric positive-definiteness.

We present an alternative approach that consists of finding the appropriate correction $[\beta]\left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla u\right)^{-}$ iteratively. Considering the discrete point of view so that problems like (4.15) for arbitrary $\boldsymbol{c}$ can be considered, we define the desired u as the solution of

$$\left\{\begin{array}{ll} -\nabla\cdot\left(\beta\nabla u\right) & = f, \\ {[u]} & = a, \\ {[\beta\nabla u]} & = b\boldsymbol{n} + \beta^{+}\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla a + [\beta]\left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla u\right)^{-}, \end{array}\right. \tag{4.22}$$

and the desired solution u is approached iteratively by a sequence u_k, $k \geq 0$.

This approach is motivated by the fact that the GFM is reported and shown convergent for the solution field itself [165], supporting the idea that the information that was omitted when using (4.16) could be somehow recovered *a posteriori* in order to apply consistent jump conditions. In the two following subsections, we first present our approach to estimate $[\beta]\left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}\right)\cdot\nabla u\right)^-$ at relevant grid points, then the iterative method is developed.

4.3.1 Evaluation of the correction jump terms

The evaluation of $[\beta]\left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}\right)\cdot\nabla u_k\right)^-$ at jump nodes stands as a building brick of the iterative method, since it is required to calculate the appropriate corrections in order to determine u_{k+1}. For the sake of generality, let q be the field, sampled at all grid nodes in Ω, such that

$$\left\{\begin{array}{ll} -\nabla\cdot(\beta\nabla q) & = f, \\ [q] & = a, \\ [\beta\nabla q] & = \boldsymbol{c}. \end{array}\right. \tag{4.23}$$

Hereafter, we introduce the *extension from the interface* that allows a definition of an extended node-sampled field q^{ext} over a layer of a few grid nodes across Γ such that q^{ext} is consistent with q^- on the interface Γ. Such an extension allows $(\nabla q)^-$ to be estimated as ∇q^{ext} at the relevant jump nodes (even in Ω^+) using standard central finite differences. This has the advantage of bypassing the difficulties originating from one-sided differentiation in locally under-resolved scenarii, thus making the overall approach more systematic.

The method is inspired from the *extrapolation over the interface* introduced by Aslam [172]. In fact, it can be viewed as a 'two-sided constant extrapolation with subcell resolution' for a field defined on the co-dimension one interface Γ, by the terminology from [172]. Assuming that the field to extend has well-defined values q_Γ^- on Γ, one can define q^{ext} as the solution of

$$\begin{cases} \dfrac{\partial q^{\text{ext}}}{\partial \tau} + \text{sign}\left(\phi\right) \boldsymbol{n} \cdot \nabla q^{\text{ext}} = 0, \\ q^{\text{ext}}\left(\boldsymbol{x}\right) = q_\Gamma^-\left(\boldsymbol{x}\right), \ \forall \boldsymbol{x} \in \Gamma, \end{cases} \tag{4.24}$$

at steady-state in pseudo-time τ. In the case of a field q satisfying (4.23), well-defined values of q_Γ^- can be evaluated at intersections between Γ and grid lines by means of (4.9), which carries over to multidimensional cases by replacing b by the appropriate Cartesian component of $\boldsymbol{c}$.

This method builds upon those locally-defined interface values and solves (4.24) to extend them to relevant surrounding grid nodes in Ω, with sub-cell resolution at jump nodes. Consider for instance a jump node $(i,\, j)$ such that $\phi_{i,\,j} > 0$ in a two-dimensional case, as illustrated in Figure 4.2, the corresponding node value for the extended field $q_{i,\,j}^{\text{ext}}$ is defined as the steady state value of the pseudo-time problem

$$\frac{\mathrm{d}q_{i,\,j}^{\text{ext}}}{\mathrm{d}\tau} = -n_x \frac{q_{i,\,j}^{\text{ext}} - q_\Gamma^-\left(\boldsymbol{x}_{i-\theta_\text{l},\,j}\right)}{\theta_\text{l} h_x} - n_y \frac{q_{i,\,j}^{\text{ext}} - q_\Gamma^-\left(\boldsymbol{x}_{i,\,j-\theta_\text{b}}\right)}{\theta_\text{b} h_y} \tag{4.25}$$

where $\boldsymbol{x}_{i-\theta_\text{l},\,j}$ (resp. $\boldsymbol{x}_{i,\,j-\theta_\text{b}}$) represents the location of the intersection between Γ and the grid line joining nodes $(i-1,\, j)$ (resp. $(i,\, j-1)$) and node (i, j). We solve the latter forward in pseudo-time τ until steady-state. Any time-integration technique can be used for solving this pseudo-time problem. For simplicity and as a proof of concept, we choose to advance (4.25) forward in time using the explicit Euler method. The pseudo-time step $\Delta\tau$ is set to satisfy $\Delta\tau \leq \dfrac{\theta_i h_i}{\mathcal{D}}$, $\forall i \in \{x, y\}$ and $\mathcal{D} = 2$ (or $\forall i \in \{x, y, z\}$ and $\mathcal{D} = 3$, in three dimensions), which naturally ensures that the Courant number is everywhere less than 1 since all normal components n_i are bounded above by 1. The most critical extended values are to be obtained for the grid nodes close the interface and for their own neighborhoods; we have used a fixed number of 20 pseudo-time steps for extending interface values. Finally, the node value $q_{i,j}^{\text{ext}}$ is set equal to $q_\Gamma^-\,(\boldsymbol{x}_{i-\theta_\text{l},\,j})$ (resp. $q_\Gamma^-\,(\boldsymbol{x}_{i,\,j-\theta_\text{b}})$) when $\theta_\text{l} \leq \varepsilon_\theta$ (resp. $\theta_\text{b} \leq \varepsilon_\theta$), where $\varepsilon_\theta = \min\left(\dfrac{h_x}{x_\text{max} - x_\text{min}},\, \dfrac{h_y}{y_\text{max} - y_\text{min}}\right)$.

The above subcell resolution establishes a difference with the original *constant* extrapolation from [172]. It is meant to fully exploit the interface values q_Γ^- that are well-defined through (4.9) and to better ensure consistency between the extension q^{ext} and q_Γ^-. Moreover, the above extension procedure benefits from linearity, contrary to the original *linear* and *quadratic* extrapolation technique from [172]: q^{ext} depends linearly on the values of q_Γ^- and thus, linearly on the values of q. As discussed next, linearity is essential in order to ensure the stability of the iterative procedure. Finally, it does not require a minimal number of well-defined inner nodes with well-defined neighbors since only the value of q_Γ^- are required. Therefore, this ensures a robust behavior even in case of severely under-resolved interfaces (*e.g.*, a sphere of radius of the order of the grid spacing).

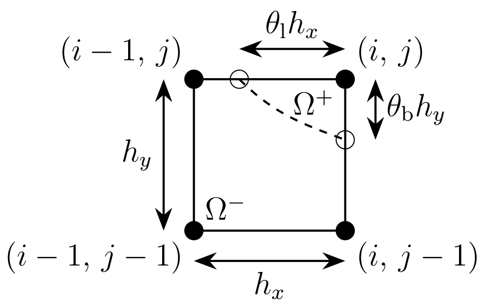


Figure 4.2: illustration of a jump node $(i,\,j)$ in two dimensions.

4.3.2 Iterative method

In this section, we present the iterative method used to solve (4.22). Let m be the total number of unknowns. We use the notation $\mathsf{q} \in \mathbb{R}^{m\times 1}$ for the column vector listing all the node values of a field q. We denote by $\mathtt{A} \in \mathbb{R}^{m\times m}$ the SPD matrix resulting of the discretization detailed in section 4.2, applied in a dimension-by-dimension fashion. Note that $\mathtt{A}$ is entirely defined by the interface Γ (and the node values of β in case of variable β).

The vector $\mathtt{u}$ of the node values of the desired solution u is governed by

$$\mathtt{Au} = \mathtt{f} + \mathtt{j}\left(\mathtt{a}, \mathtt{b}, \mathtt{u}^{\mathrm{ext}}\right) \tag{4.26}$$

where $\mathtt{j}\left(\mathtt{a}, \mathtt{b}, \mathtt{u}^{\mathrm{ext}}\right)$ represents the contributions to the right-hand side, at jump nodes, due to the jump conditions $[u] = a$ and $[\beta\nabla u] = b\boldsymbol{n} + \beta^{+}\left(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}\right)\cdot\nabla a + [\beta]\left(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}\right)\cdot\nabla u^{\mathrm{ext}}$, where u^{ext} is a consistent extension of u_{Γ}^{-}. Note that $\mathtt{j}$ is a multilinear map.

The iterative procedure approximates $\mathtt{u}$ by a converging iterative sequence $\mathtt{u_k}$, $k \geq 0$. Since no approximation of u (and thus u^{ext}) is known beforehand, the sequence is initialized with the solution $\mathtt{u_0}$ of

$$\mathtt{Au_0} = \mathtt{f} + \mathtt{j}\left(\mathtt{a}, \mathtt{b}, 0\right) \tag{4.27}$$

and a consistent extension $\mathtt{u}_0^{\mathrm{ext}}$ can be evaluated, with the technique from subsection 4.3.1. A fix-point update $\tilde{\mathtt{u}}_{\mathtt{k+1}}$ is then calculated as the solution of

$$\mathtt{A}\tilde{\mathtt{u}}_{\mathtt{k+1}} = \mathtt{f} + \mathtt{j}\left(\mathtt{a}, \mathtt{b}, \mathtt{u}_{\mathtt{k}}^{\mathrm{ext}}\right), \tag{4.28}$$

as well as its extension $\tilde{\mathtt{u}}_{\mathtt{k+1}}^{\mathrm{ext}}$, using the technique from subsection 4.3.1. The new element of the sequence $\mathtt{u}_{\mathtt{k+1}}$ and its extension $\mathtt{u}^{\mathrm{ext}}_{\mathtt{k+1}}$ are then defined as[4]

$$\begin{aligned} \mathtt{u}_{\mathtt{k+1}} &= \mathtt{u}_{\mathtt{k}} + \eta_k \left(\tilde{\mathtt{u}}_{\mathtt{k+1}} - \mathtt{u}_{\mathtt{k}}\right) && (4.29) \\ \mathtt{u}^{\mathrm{ext}}_{\mathtt{k+1}} &= \mathtt{u}^{\mathrm{ext}}_{\mathtt{k}} + \eta_k \left(\tilde{\mathtt{u}}^{\mathrm{ext}}_{\mathtt{k+1}} - \mathtt{u}^{\mathrm{ext}}_{\mathtt{k}}\right) && (4.30) \end{aligned}$$

where η_k is chosen to minimize the ℓ_2-norm of the residual

$$\mathtt{r}_{\mathtt{k+1}} = \mathtt{A}\mathtt{u}_{\mathtt{k+1}} - \mathtt{f} - \mathtt{j}\left(\mathtt{a}, \mathtt{b}, \mathtt{u}^{\mathrm{ext}}_{\mathtt{k+1}}\right). \tag{4.31}$$

Using the multilinearity of $\mathtt{j}$, we have

$$\begin{aligned} \mathtt{r}_{\mathtt{k+1}} &= (1-\eta_k)\left(\mathtt{A}\mathtt{u}_{\mathtt{k}} - \mathtt{f} - \mathtt{j}\left(\mathtt{a}, \mathtt{b}, \mathtt{u}^{\mathrm{ext}}_{\mathtt{k}}\right)\right) + \eta_k\left(\mathtt{A}\tilde{\mathtt{u}}_{\mathtt{k+1}} - \mathtt{f} - \mathtt{j}\left(\mathtt{a}, \mathtt{b}, \tilde{\mathtt{u}}^{\mathrm{ext}}_{\mathtt{k+1}}\right)\right) \\ &= (1-\eta_k)\,\mathtt{r}_{\mathtt{k}} + \eta_k \tilde{\mathtt{r}}_{\mathtt{k+1}} && (4.32) \end{aligned}$$

so that the value of η_k minimizing $\left\|\mathtt{r}_{\mathtt{k+1}}\right\|_2$ is

$$\eta_k = \frac{\mathtt{r}_{\mathtt{k}}{}^{\mathrm{T}}\left(\mathtt{r}_{\mathtt{k}} - \tilde{\mathtt{r}}_{\mathtt{k+1}}\right)}{\left\|\tilde{\mathtt{r}}_{\mathtt{k+1}} - \mathtt{r}_{\mathtt{k}}\right\|_2^2}. \tag{4.33}$$

The iterative procedure repeats steps (4.28), (4.29) and (4.30) with (4.33) until convergence, *i.e.*, until either the $\ell_\infty-$norm of $\eta_k\left(\tilde{\mathtt{u}}_{\mathtt{k+1}} - \mathtt{u}_{\mathtt{k}}\right)$ or the ℓ_2-norm of $\mathtt{r}_{\mathtt{k+1}}$ falls below a user-defined threshold.

Remark. *A standard fix-point iterative method would follow the same procedure as above but with a constant value of $\eta_k = 1$, $\forall k \geq 0$. This was observed to be well-behaved very often, but a standard fix-point method sometimes suffers from a slow convergence rate and/or oscillatory behavior around the desired solution or even divergence, especially in*

[4] $\mathtt{u}^{\mathrm{ext}}_{\mathtt{k+1}}$ can indeed be defined with such a linear combination, by virtue of the linearity of the method from subsection 4.3.1.

the case of large ratios of the β coefficients across the interface. The above residue-minimization method prevents such drawbacks.

Remark. *Only the fields $\mathtt{u}_0$ and $\tilde{\mathtt{u}}_k$, $k \geq 1$, are extended using the method detailed in subsection 4.3.1. The extended field u^{ext}_{k+1} is defined as a linear combination of the (known) extended fields u^{ext}_k and $\tilde{u}^{ext}_{k+1}$, see (4.30). Only one field extension is required per iterative step.*

Remark. *The left-hand sides in linear systems (4.27) and (4.28) are $k-$independent, only the right-hand sides differ between two iterations. Therefore, the additional cost associated with one step of the iterative procedure is mitigated, since 1) the same (preconditioned) linear solver can be used for all iterative steps, 2) the only non-zero modifications to the right-hand side terms are associated with jump nodes, 3) the use of $\mathtt{u}_\mathtt{k}$ as an initial guess when solving (4.28) significantly accelerates the resolution of the successive linear systems, as the procedure advances. In this work, we take advantage of the symmetric positive-definiteness of the linear system by using the Conjugate Gradient solver with a multigrid preconditioner (or a Modified Incomplete Cholesky preconditioner for serial runs). Section 4.4 provides data on the additional computational cost of xGFM.*

Remark. *The xGFM recovers consistency immediately, i.e., $\tilde{\mathtt{u}}_1 = \mathtt{u}_0$, in cases where* $[\beta] = 0$.

4.4 Illustrations and examples in two spatial dimensions

In this section, we show results obtained in two spatial dimensions using the strategy described in section 4.3 with uniform grids of $N \times N$ grid cells. We analyze the numerical accuracy of the components of ∇u (or $\beta\nabla u$ in case of large ratios of coefficients β). The derivatives are evaluated using (4.13) at jump nodes, hence including every node of the computational domain (even under-resolved ones) in the analysis.

4.4.1 Example 1

We consider the problem (4.3) in $\Omega = [-1,\, 1] \times [-1,\, 1]$ with $\phi\,(x,\, y) = \sqrt{x^2 + y^2} - \dfrac{1}{2}$, $f = 0$, $a = 0$, $b = 2$, $\beta^+ = \beta^- = 1$ and Dirichlet boundary condition $u\,(x,\, y) = 1 + \log\left(2\sqrt{x^2 + y^2}\right)$ on $\partial\Omega$. The exact solution is

$$\begin{cases} u\,(x,\, y) = 1, & \forall\,(x,\, y) \in \Omega^-, \\ u\,(x,\, y) = 1 + \log\left(2\sqrt{x^2 + y^2}\right), & \forall\,(x,\, y) \in \Omega^+. \end{cases}$$

This is the fifth example from [16], which reports slightly better convergence for this problem compared to the other examples they consider. This relates to our interpretation of the method since $\boldsymbol{c} = b\boldsymbol{n}$ actually satisfies the consistency requirement given that $(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot \nabla a = \boldsymbol{0}$ and $[\beta] = 0$. As illustrated in Figure 4.3, both the solution *and* its gradient converge in infinity norm, when the derivatives are evaluated with (4.13) for jump nodes.

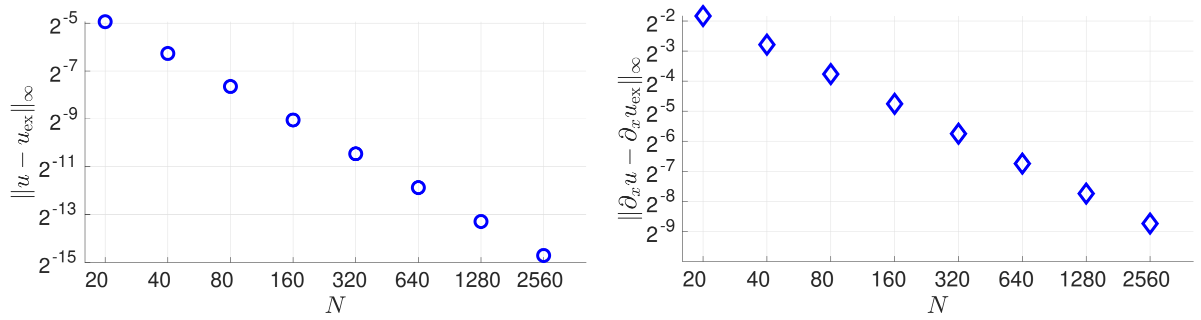


Figure 4.3: convergence analysis for the example of subsection 4.4.1, using the GFM. Left: errors in infinity norm for the solution field; right: errors in infinity norm for the partial derivative with respect to x (the partial derivative with respect to y is totally equivalent), as evaluated with equation (4.13) for the jump nodes. As it can be seen in these graphs, the solution *and* its gradient do converge with the GFM for such a problem. The respective orders of convergence are approximately 1.4 and 0.99.

4.4.2 Example 2

In this example, we consider $\Omega = [-1,\, 1] \times [-1,\, 1]$ with $\phi\,(x,\, y) = \sqrt{x^2 + y^2} - \dfrac{1}{2}$, $\beta^- = \beta^+ = 1$. The exact solution is chosen to be

$$\begin{cases} u\,(x,\, y) = x^2 - y^2, & \forall\,(x,\, y) \in \Omega^-, \\ u\,(x,\, y) = 0, & \forall\,(x,\, y) \in \Omega^+. \end{cases}$$

The right-hand side terms are chosen accordingly, as

$$\begin{aligned} f\,(x,\, y) &= 0, \\ a\,(x,\, y) &= y^2 - x^2, \\ b\,(x,\, y) &= 4\left(y^2 - x^2\right). \end{aligned}$$

This is the seventh example from [16]. Note that $\boldsymbol{c} = b\boldsymbol{n}$ does not satisfy the consistency requirement, but that consistency can be recovered through $\boldsymbol{c} = b\boldsymbol{n} + \beta^+\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right) \cdot \nabla a$, since $[\beta] = 0$, once again. No iterative procedure is required to recover the consistent jump conditions in this case once again.

Figure 4.4 shows that the xGFM outperforms the GFM: the rate of convergence for the solution is improved from approximately 0.90 to 1.35. This example illustrates the recovery of convergent gradients by means of consistent jump conditions, as well. The differences in the Cartesian components of $[\beta\nabla u]$ between the values determined by the xGFM and the GFM are illustrated in Figure 4.5. Figure 4.6 compares the magnitude of the flux $\|\beta\nabla u\|$, as evaluated by the xGFM and the GFM for $N = 640$.

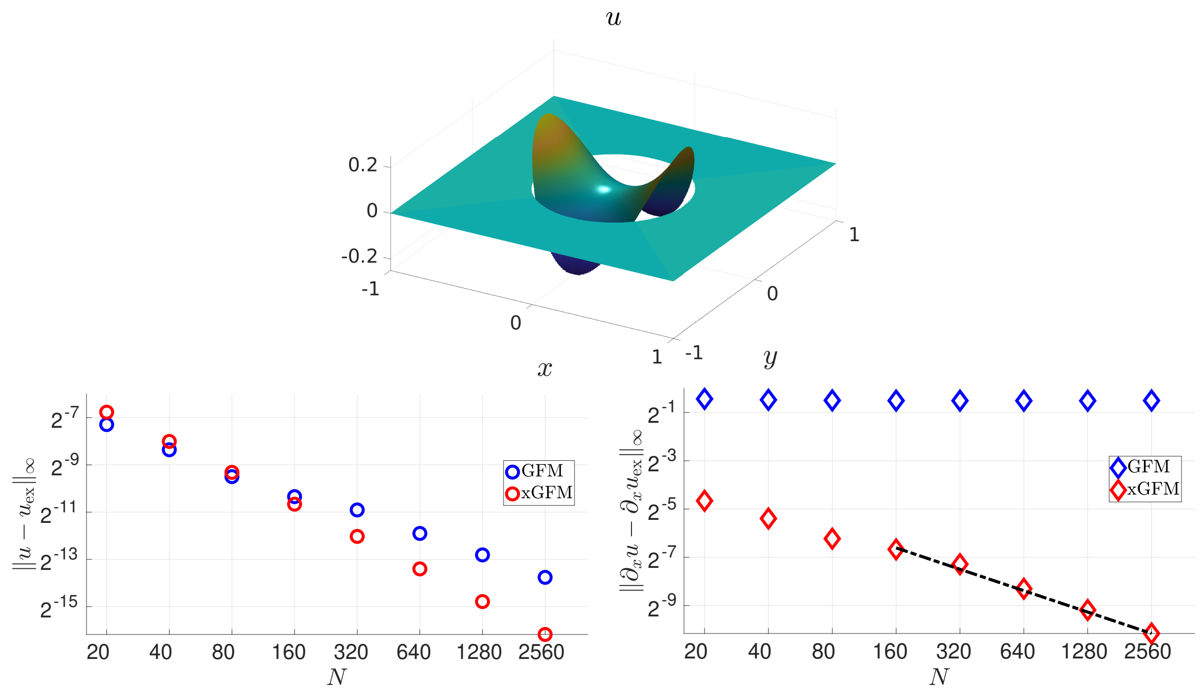


Figure 4.4: convergence analysis for the example of subsection 4.4.2. Top: illustration of the solution for the example of subsection 4.4.2, with $N = 640$. Bottom left: errors in infinity norm for the solution field. Bottom right: right: errors in infinity norm for the partial derivative with respect to x (the partial derivative with respect to y is totally equivalent), the slope of the dashed line is -0.89.

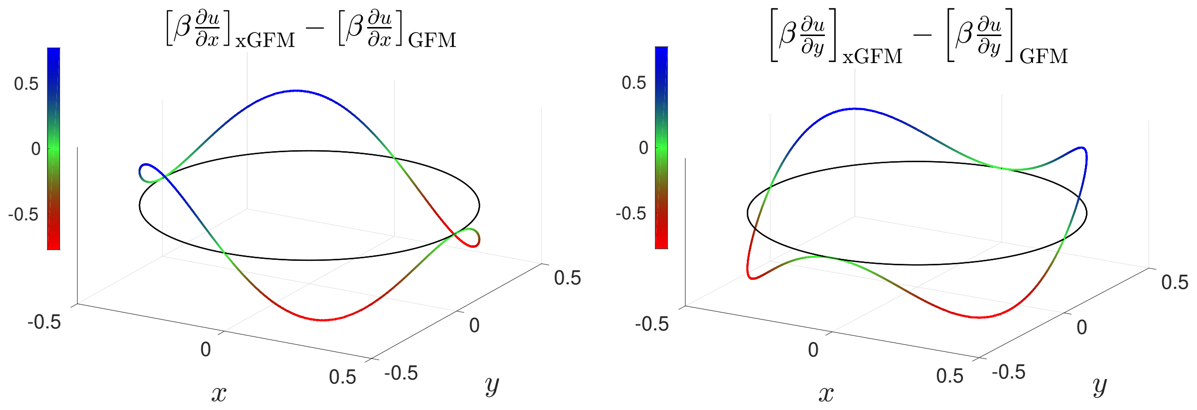


Figure 4.5: differences, along the interface Γ, in the jumps $\left[\beta\dfrac{\partial u}{\partial x}\right]$ (left) and $\left[\beta\dfrac{\partial u}{\partial y}\right]$ (right), as enforced by the xGFM and the GFM, for the problem of subsection 4.4.2.

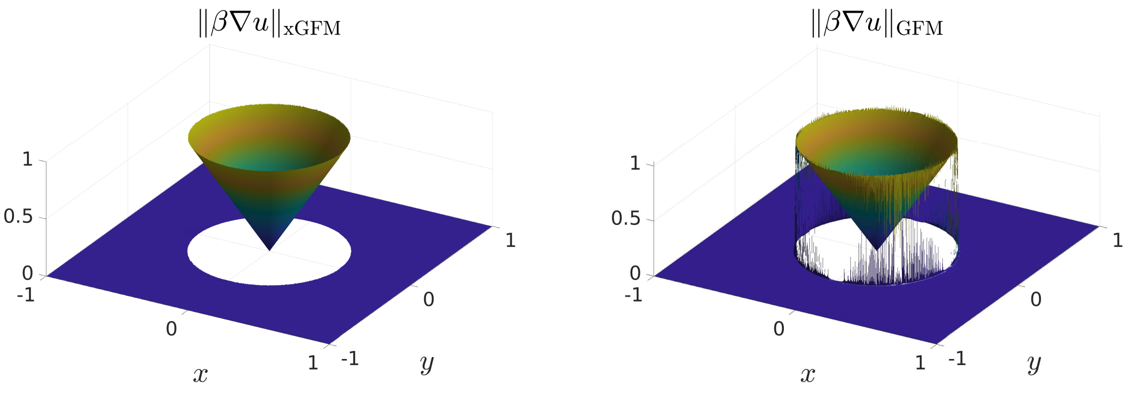


Figure 4.6: magnitude of the flux $\|\beta\nabla u\|$, as evaluated by the xGFM (left) and the GFM (right), for the problem of subsection 4.4.2.

4.4.3 Example 3

In this example, we consider $\Omega = [0,\, 1]\times[0, 1]$ with $\phi\left(x,\, y\right) = \sqrt{\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2} - \frac{1}{4}$, $\beta^- = 2$, $\beta^+ = 1$. The exact solution is chosen to be

$$\begin{cases} u\left(x,\, y\right) = \exp\left(-x^2-y^2\right), & \forall\left(x,\, y\right)\in\Omega^-, \\ u\left(x,\, y\right) = 0, & \forall\left(x,\, y\right)\in\Omega^+. \end{cases}$$

The right-hand side terms are chosen accordingly, as

$$\begin{aligned} f\left(x,\, y\right) &= -8\left(x^2+y^2-1\right)\exp\left(-x^2-y^2\right), \quad \forall\left(x,\, y\right)\in\Omega^- \\ a\left(x,\, y\right) &= -\exp\left(-x^2-y^2\right), \\ b\left(x,\, y\right) &= 8\left(2x^2+2y^2-x-y\right)\exp\left(-x^2-y^2\right), \end{aligned}$$

and $f = 0$ in Ω^+. This is the third example from [16]. Note that the consistency requirement at the interface is more complicated in this case since $[\beta]\neq 0$. As a consequence, the xGFM procedure is required in order to recover consistency, in this case.

As illustrated in Figure 4.7, the xGFM recovers the consistent jump conditions and outperforms the GFM with respect to accuracy for the solution and ensures convergence for its gradient. The rate of convergence for the solution is improved from approximately 0.88 to 1.45 and the gradient of the solution does converge, as well. Table 4.1 compares the cumulative number of iterations of the conjugate gradient solver between the xGFM and the GFM. As shown in this table the xGFM requires two to three times as many iterations of the conjugate gradient solver as the GFM. Noticeably, this ratio decreases with increasing N. This table also reports the computational overhead, measured in terms of execution time, due to extending the solution (possibly several times) and, separately, the overhead due to solving the additional linear system(s) of equations. It can be seen that the solution extension steps add a significant overhead for coarse grids but this part becomes less and less significant, as the grid is refined. On the other hand, the extra cost associated with the additional linear systems to be solved always compares with the rough execution time of the GFM[5]. We also note that there is no difference in the accuracy of the gradient whether one uses a linear or a quadratic interpolation to locate the interface position, unlike the case of Dirichlet boundary conditions (see the analysis of [173]). The differences in the Cartesian components of $[\beta \nabla u]$ between the values determined by the xGFM and the GFM are illustrated in Figure 4.8. Figure 4.9 compares the magnitude of the flux $\|\beta \nabla u\|$, as evaluated by the xGFM and the GFM for $N = 640$.

[5]Throughout this chapter, we use a fixed user-defined threshold value for convergence of the iterative procedure, namely $\|\mathtt{r}_{\mathtt{k+1}}\|_2 \leq 10^{-8} \left\|\mathtt{f} + \mathtt{j}\left(\mathtt{a}, \mathtt{b}, \mathtt{u}^{\text{ext}}_{\mathtt{k+1}}\right)\right\|_2$ (see subsection 4.3.2) independently of the level of refinement of the grid. This means that the used convergence criterion is actually more stringent for coarser than finer grids, effectively leading to more iterations of our iterative procedure overall hence the significant overheads due to extension for coarse grids.

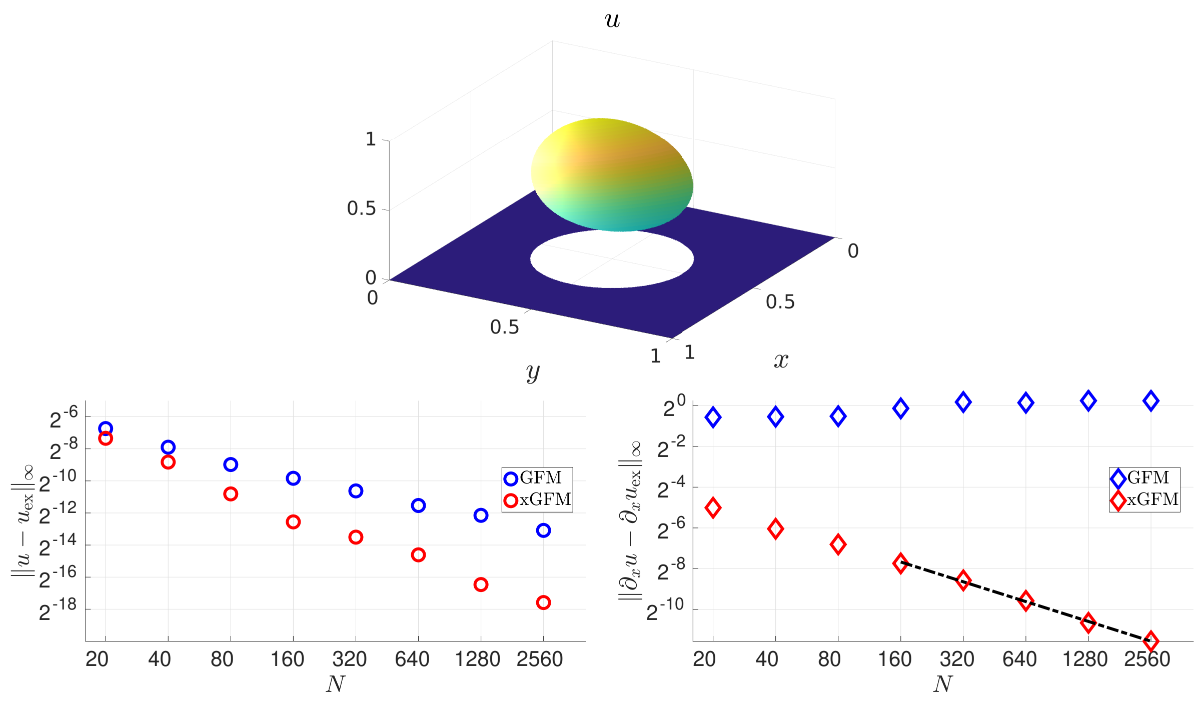


Figure 4.7: convergence analysis for the example of subsection 4.4.3. Top: illustration of the solution for the example of subsection 4.4.3, with $N = 640$. Bototm left: errors in infinity norm for the solution field. Bottom right: errors in infinity norm for the partial derivative with respect to x (the partial derivative with respect to y is totally equivalent), the slope of the dashed line is -0.97

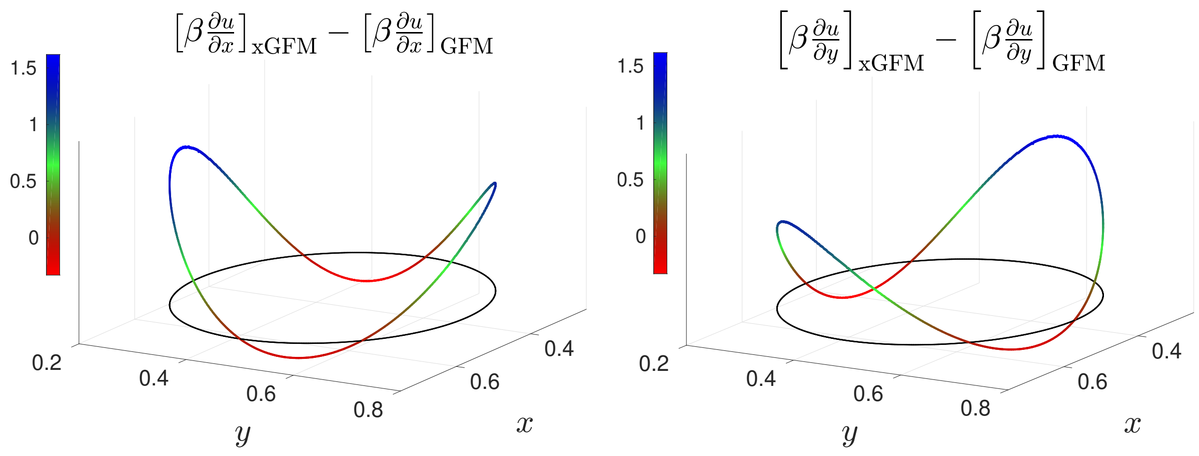


Figure 4.8: differences, along the interface Γ, in the jumps $\left[\beta \dfrac{\partial u}{\partial x}\right]$ (left) and $\left[\beta \dfrac{\partial u}{\partial y}\right]$ (right), as enforced by the xGFM and the GFM, for the problem of subsection 4.4.3.

	Linear interpolation of ϕ		Quadratic interpolation of ϕ	
N	GFM	xGFM	GFM	xGFM
20	17	52	17	52
40	26	74	26	75
80	39	108	39	108
160	58	148	58	150
320	86	202	86	202
640	128	280	128	280
1280	191	384	190	383
2560	282	525	281	524

	Execution time				
		xGFM: extension(s) and update(s) of RHS		xGFM: additional linear solve(s)	
N	GFM				
20	0.077 s	0.112 s	+145.3%	0.015 s	+18.9%
40	0.018 s	0.074 s	+408.7%	0.008 s	+47.0%
80	0.051 s	0.147 s	+286.1%	0.036 s	+69.4%
160	0.185 s	0.250 s	+135.2%	0.158 s	+85.3%
320	0.815 s	0.479 s	+58.6%	0.684 s	+83.9%
640	3.821 s	1.046 s	+27.4%	2.997 s	+78.5%
1280	19.23 s	4.969 s	+25.8%	14.65 s	+76.2%
2560	129.6 s	13.89 s	+10.7%	94.21 s	+72.7%

Table 4.1: performance and execution times. Top: cumulative number of iterations of the preconditioned conjugate gradient solver for the problem of subsection 4.4.3. Linear (resp. quadratic) interpolation of ϕ means that the values of the θ parameters (as defined in section 4.2) are evaluated as the roots of a local linear (resp. quadratic) interpolant of ϕ. Bottom: execution times obtained with a Dual Intel Xeon Gold 6148 processor with 64 GB of RAM (single core execution of a non-optimized Matlab implementation).

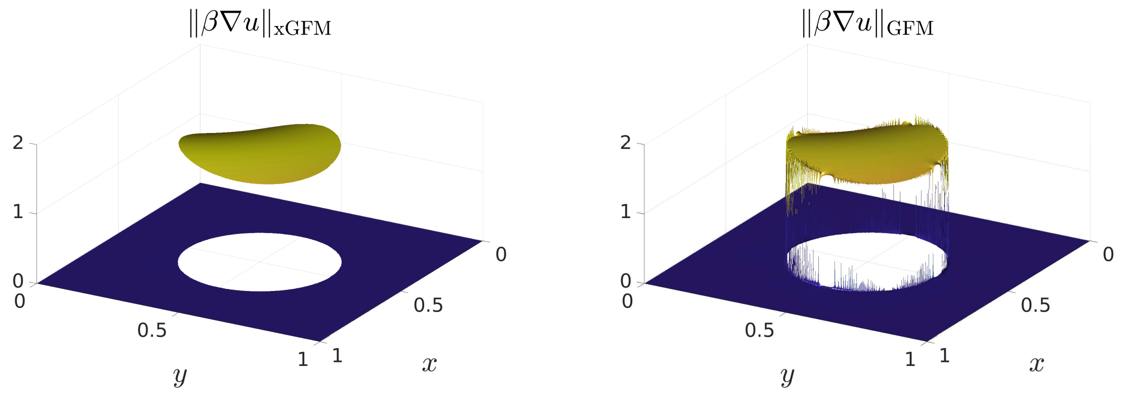


Figure 4.9: magnitude of the flux $\|\beta\nabla u\|$, as evaluated by the xGFM (left) and the GFM (right), for the problem of subsection 4.4.3.

4.4.4 Example 4

In this example, we consider $\Omega = [-1,\,1]\times[-1,\,1]$. The interface is defined as the parameterized curve $\Gamma \equiv (x(\vartheta)\,,\,y(\vartheta))\,,\;\vartheta\in[0,\,2\pi[$ where

$$\begin{cases} x(\vartheta) = 0.02\sqrt{5} + (0.5 + 0.2\sin(5\vartheta))\cos(\vartheta)\,, \\ y(\vartheta) = 0.02\sqrt{5} + (0.5 + 0.2\sin(5\vartheta))\sin(\vartheta)\,. \end{cases} \tag{4.34}$$

The corresponding levelset function is defined as the exact signed distance to Γ, negative in the interior region. The coefficients are $\beta^- = 1$ and $\beta^+ = 10$, hence (4.21) is used instead of (4.20). The exact solution is chosen to be

$$\begin{cases} u(x,\,y) = x^2 + y^2, & \forall (x,\,y)\in\Omega^-, \\ u(x,\,y) = 0.1\left(x^2+y^2\right)^2 - 0.01\log\left(2\sqrt{x^2+y^2}\right), & \forall (x,\,y)\in\Omega^+, \end{cases}$$

the right-hand side f and the jump terms a and b are defined accordingly. This is the eighth example from [16]. The consistency requirement at the interface is not trivial for this test case once more, since $[\beta]\neq 0$ and because of the convoluted interface. As a consequence, the xGFM procedure is required in order to recover consistency.

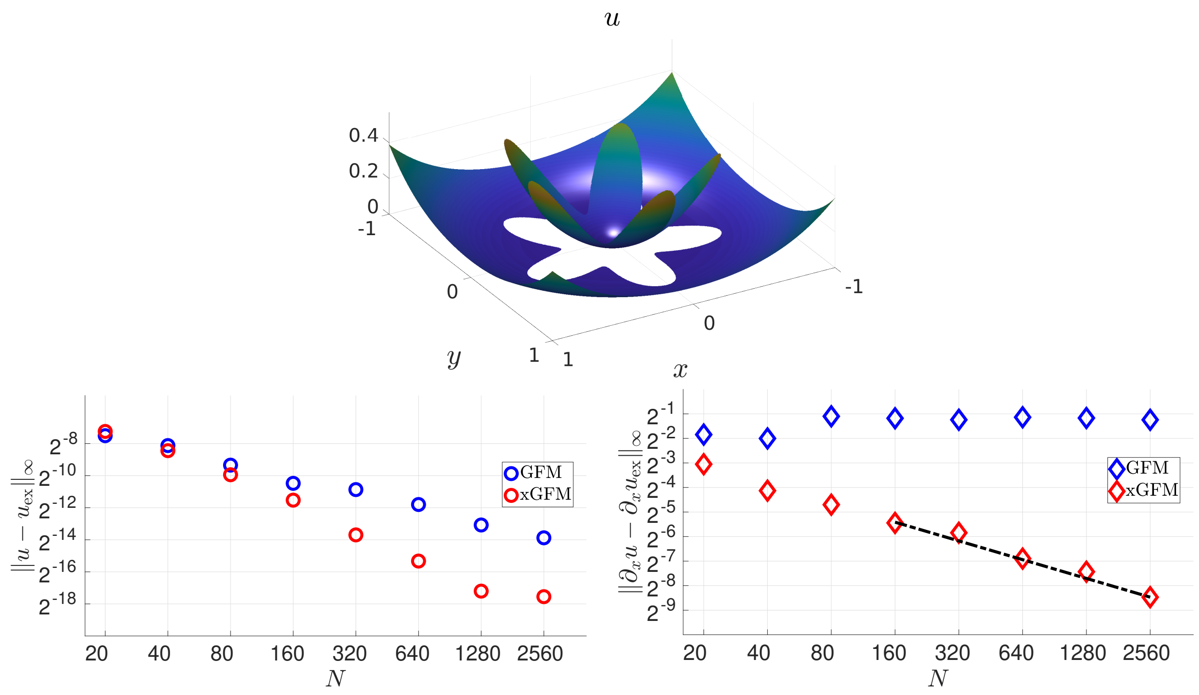


Figure 4.10: convergence analysis for the example of subsection 4.4.4. Top: illustration of the solution for the example of subsection 4.4.4, with $N = 640$. Bottom left: errors in infinity norm for the solution field. Bottom right: errors in infinity norm for the partial derivative with respect to x (the partial derivative with respect to y behaves very similarly), the slope of the dashed line is -0.76.

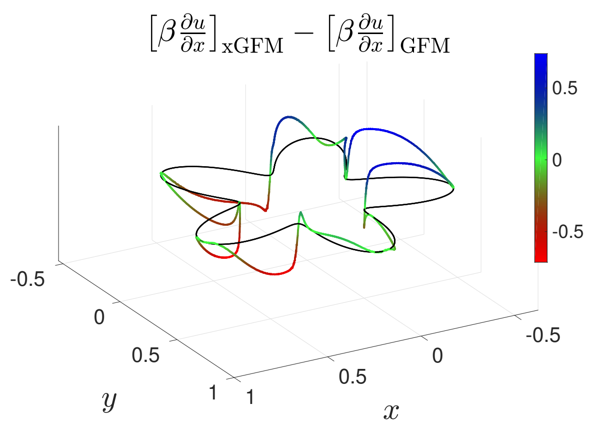
As illustrated in Figure 4.10, the xGFM recovers the consistent jump conditions. It outperforms the GFM in terms of accuracy for the solution (rate of convergence improved from 0.92 to 1.60) but, more importantly, it also ensures convergence for its gradient. Table 4.2 shows that the xGFM requires four to two and half times as many iterations of the conjugate gradient solver as the GFM, this ratio decreasing with N. The observations regarding the computational overheads associated with extensions and with the additional systems of equations to be solved are comparable to the conclusion drawn in subsection 4.4.3.

The differences in the Cartesian components of $[\beta\nabla u]$ between the values determined by the xGFM and the GFM are illustrated in Figure 4.11.

	Linear interpolation of ϕ		Quadratic interpolation of ϕ	
N	GFM	xGFM	GFM	xGFM
20	19	85	18	84
40	28	105	28	105
80	40	140	40	141
160	58	204	58	202
320	85	260	85	258
640	125	343	125	343
1280	187	467	187	467
2560	280	640	281	642

	Execution time				
		xGFM: extension(s) and update(s) of RHS		xGFM: additional linear solve(s)	
N	GFM				
20	0.078 s	0.176 s	+226.9%	0.020 s	+26.3%
40	0.017 s	0.178 s	+1042.1%	0.009 s	+53.4%
80	0.049 s	0.322 s	+656.0%	0.058 s	+118.0%
160	0.198 s	0.774 s	+390.2%	0.294 s	+148.3%
320	0.822 s	1.392 s	+169.3%	1.044 s	+127.1%
640	3.891 s	2.990 s	+76.8%	4.491 s	+115.4%
1280	19.43 s	5.530 s	+79.9%	21.41 s	+110.2%
2560	127.9 s	30.67 s	+24.0%	137.9 s	+107.8%

Table 4.2: performance and execution times. Top: cumulative number of iterations of the preconditioned conjugate gradient solver for the problem of subsection 4.4.4. Linear (resp. quadratic) interpolation of ϕ means that the values of the θ parameters (as defined in section 4.2) are evaluated as the roots of a local linear (resp. quadratic) interpolant of ϕ. Bottom: execution times obtained with a Dual Intel Xeon Gold 6148 processor with 64 GB of RAM (single core execution of a non-optimized Matlab implementation).



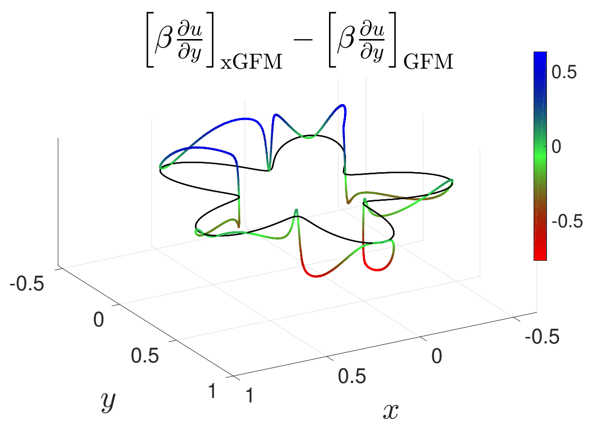


Figure 4.11: differences, along the interface Γ, in the jumps $\left[\beta\dfrac{\partial u}{\partial x}\right]$ (left) and $\left[\beta\dfrac{\partial u}{\partial y}\right]$ (right), as enforced by the xGFM and the GFM, for the problem of subsection 4.4.4.

4.4.5 Example 5

In this example, we consider $\Omega = [-1,\, 1] \times [0,\, 3]$. The interface is defined as the parameterized curve $\Gamma \equiv (x(\vartheta),\, y(\vartheta))\,,\ \vartheta \in [0,\, 2\pi[$ where

$$\begin{cases} x(\vartheta) = 0.6\cos(\vartheta) - 0.3\cos(3\vartheta)\,, \\ y(\vartheta) = 1.5 + 0.7\sin(\vartheta) - 0.07\sin(3\vartheta) + 0.2\sin(7\vartheta)\,. \end{cases} \tag{4.35}$$

The corresponding levelset function is defined as the exact signed distance to Γ, negative in the interior region. The coefficients are $\beta^- = 1$ and $\beta^+ = 10$, hence (4.21) is used instead of (4.20) again. The exact solution is chosen to be

$$\begin{cases} u(x,\, y) = \exp(x)\left(x^2\sin(y) + y^2\right), & \forall (x,\, y) \in \Omega^-, \\ u(x,\, y) = -\left(x^2 + y^2\right), & \forall (x,\, y) \in \Omega^+, \end{cases}$$

the right-hand side f and the jump terms a and b are defined accordingly. This is the ninth example from [16]. The consistency requirement at the interface is not trivial for this test case, once more, since $[\beta] \neq 0$ and because of the convoluted interface. As a consequence, the xGFM procedure is required in order to recover consistency.

As illustrated in Figure 4.12, the xGFM recovers the consistent jump conditions. It outperforms the GFM regarding the accuracy for the solution u (rate of convergence improved from 0.88 to 1.58) *and* it enables the accurate calculation of the gradient. Table 4.3 shows that the xGFM requires five to three times as many iterations of the conjugate gradient solver as the GFM, this ratio decreasing with N. The observations regarding the computational overheads associated with extensions and with the additional systems of equations to be solved are comparable to the conclusions drawn here above for other examples.

The differences in the Cartesian components of $[\beta \nabla u]$ between the values determined by the xGFM and the GFM are illustrated in Figure 4.13. Figure 4.14 compares the magnitude of the flux $\|\beta \nabla u\|$, as evaluated by the xGFM and the GFM for $N = 640$.

4.4.6 Example 6: large contrast of β coefficients

In this example, we consider $\Omega = [-1,\, 1] \times [-1,\, 1]$. The interface is defined as the parameterized curve $\Gamma \equiv \left(x\left(\vartheta\right),\, y\left(\vartheta\right)\right),\ \vartheta \in [0,\, 2\pi[$ where

$$\begin{cases} x\left(\vartheta\right) = \left(0.5 + 0.1\sin\left(5\vartheta\right)\right)\cos\left(\vartheta\right), \\ y\left(\vartheta\right) = \left(0.5 + 0.1\sin\left(5\vartheta\right)\right)\sin\left(\vartheta\right). \end{cases} \tag{4.36}$$

The corresponding levelset function is defined as the exact signed distance to Γ, negative in the interior region. The coefficients are $\beta^- = 10^4$ and $\beta^+ = 1$. The exact solution is chosen to be

$$\begin{cases} u\left(x,\, y\right) = \dfrac{\exp\left(x\right)\left(x^2\sin\left(y\right) + y^2\right)}{\beta^-}, & \forall\left(x,\, y\right) \in \Omega^-, \\ u\left(x,\, y\right) = 0.5 + \cos\left(x\right)\left(y^4 + \sin\left(y^2 - x^2\right)\right), & \forall\left(x,\, y\right) \in \Omega^+, \end{cases}$$

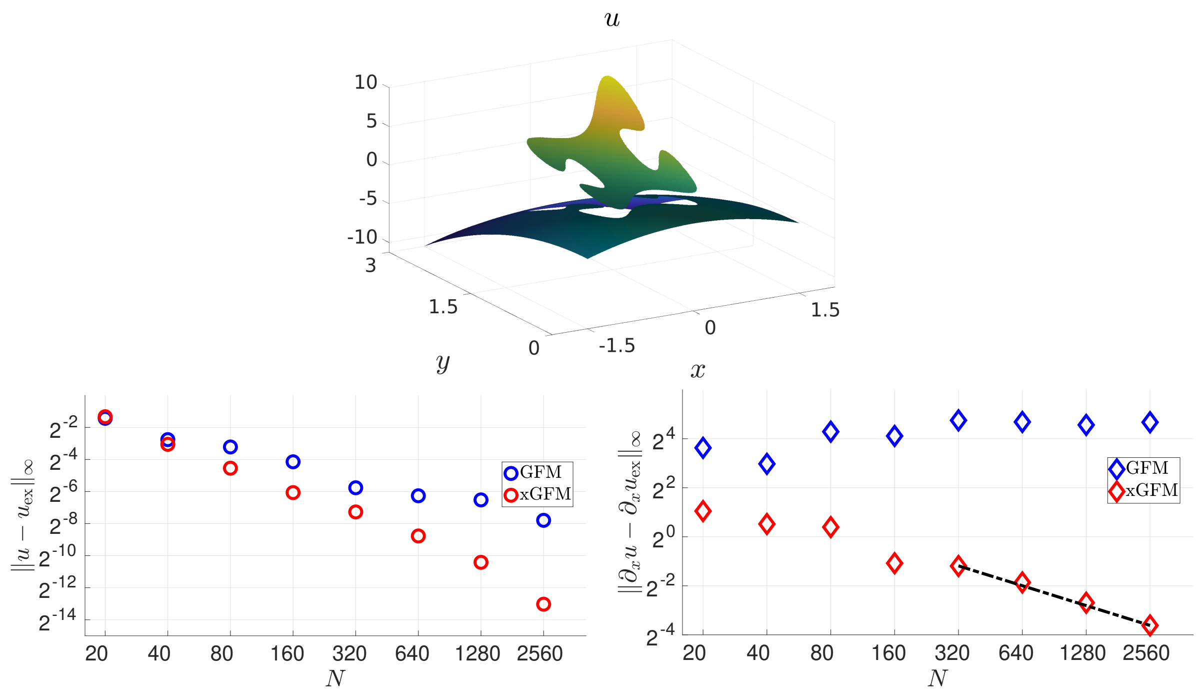


Figure 4.12: convergence analysis for the example of subsection 4.4.5. Top: illustration of the solution for the example of subsection 4.4.5, with $N = 640$. Bottom left: errors in infinity norm for the solution field. Bottom right: errors in infinity norm for the partial derivative with respect to x (the partial derivative with respect to y behaves very similarly), the slope of the dashed line is -0.81.

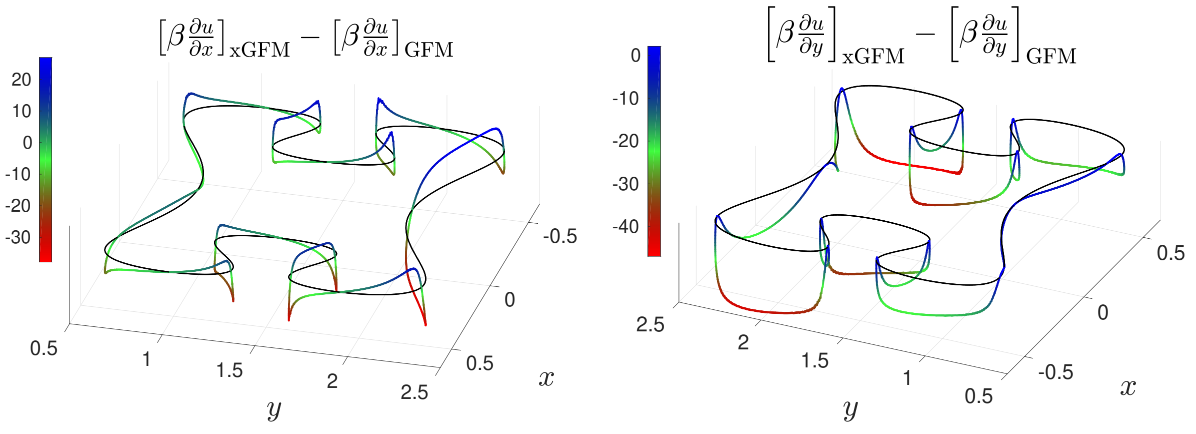
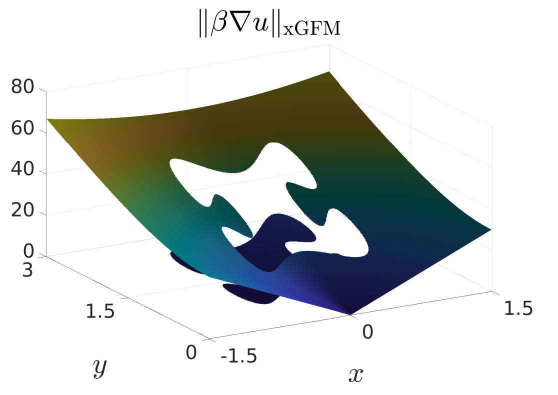


Figure 4.13: differences, along the interface Γ, in the jumps $\left[\beta\dfrac{\partial u}{\partial x}\right]$ (left) and $\left[\beta\dfrac{\partial u}{\partial y}\right]$ (right), as enforced by the xGFM and the GFM, for the problem of subsection 4.4.5. Note that different viewing angles are used between both sub-figures, for the sake of clarity.

	Linear interpolation of ϕ		Quadratic interpolation of ϕ	
N	GFM	xGFM	GFM	xGFM
20	19	84	19	83
40	28	122	28	120
80	41	166	41	167
160	60	217	60	217
320	88	298	87	299
640	128	399	128	399
1280	188	537	187	537
2560	274	719	273	719

	Execution time				
N	GFM	xGFM: extension(s) and update(s) of RHS		xGFM: additional linear solve(s)	
20	0.217 s	0.184 s	+84.8%	0.020 s	+9.3%
40	0.018 s	0.205 s	+1170.3%	0.012 s	+66.1%
80	0.121 s	0.373 s	+307.1%	0.048 s	+39.5%
160	0.178 s	0.715 s	+403.1%	0.224 s	+126.0%
320	0.871 s	1.572 s	+180.3%	1.294 s	+148.5%
640	4.356 s	8.634 s	+198.2%	5.590 s	+128.3%
1280	19.54 s	17.23 s	+88.2%	25.77 s	+131.9%
2560	124.9 s	42.68 s	+34.2%	166.8 s	+133.6%

Table 4.3: performance and execution times. Top: cumulative number of iterations of the preconditioned conjugate gradient solver for the problem of subsection 4.4.5. Linear (resp. quadratic) interpolation of ϕ means that the values of the θ parameters (as defined in section 4.2) are evaluated as the roots of a local linear (resp. quadratic) interpolant of ϕ. Bottom: execution times obtained with a Dual Intel Xeon Gold 6148 processor with 64 GB of RAM (single core execution of a non-optimized Matlab implementation).



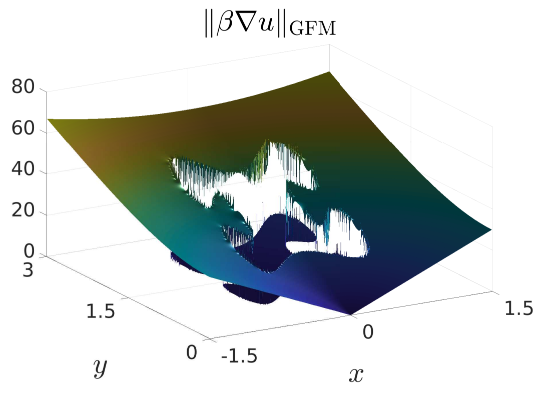


Figure 4.14: magnitude of the flux $\|\beta\nabla u\|$, as evaluated by the xGFM (left) and the GFM (right), for the problem of subsection 4.4.5.

the right-hand side f and the jump terms a and b are defined accordingly.

This example is representative of cases that can be encountered in two-phase flows problems, where large viscosity and/or mass density ratios can be encountered. Such large ratios increase the condition number of the discretization matrix, hence the increased number of iterations of the preconditioned gradient solver compared to other cases, see Table 4.4. Nevertheless, Table 4.4 also shows that the xGFM requires two to four times as many iterations of the conjugate gradient solver as the GFM, this ratio decreasing with N. The observations regarding the computational overheads associated with extensions and with the additional systems of equations to be solved are comparable to the conclusions drawn here above for other examples. In particular, it is remarkable to notice that the overheads associated with extensions become insignificant for fine grids comparatively to the cost of solving the linear system(s).

This test case is designed to have comparable fluxes $\|\beta\nabla u\|$ in both Ω^- and Ω^+ and therefore a finite value for b (as encountered in applications). As a consequence, the variations of u are much less pronounced in Ω^- than in Ω^+, to such an extent that misleading conclusions can be drawn regarding the convergence of the GFM for ∇u (or $\beta\nabla u$). In order to illustrate this assertion, we introduce a twofold expansion of the convergence analysis. First, we analyze not only the convergence of ∇u but also that of $\beta\nabla u$. Second, we consider an alternative evaluation of the partial derivatives: for jump nodes, we consider the non-centered differentiation rule involving only the forward or backward neighbor that lies on the same side of Γ as the considered jump node; if the jump node is under-resolved, *i.e.* if no such neighbor can be found, the point is disregarded from the analysis. We refer to this evaluation strategy as *intrinsically one-sided* evaluation.

The results and an illustration are shown in Figure 4.15. For such a symptomatic test case, the accuracy of u and the orders of convergence are very similar between the xGFM and the GFM(observed rate of about 1.66). When using intrinsically one-sided evaluations, ∇u *seems* to converge even when using the GFM although some points might be disregarded from the analysis. However, this no longer holds above some level of grid refinement[6] when using (4.13), hence including all grid nodes. Finally, when evaluating the flux $\beta\nabla u$, *i.e.*, the quantity of interest in the context of a projection method in incompressible multiphase fluid flow simulations for instance, the xGFM clearly outperforms the GFM once again, no matter how the derivatives are evaluated.

The differences in the Cartesian components of $[\beta\nabla u]$ between the values determined by the xGFM and the GFM are illustrated in Figure 4.16.

[6]Given that the solution's variations are scaled by β, analyzing ∇u instead of $\beta\nabla u$ is effectively equivalent to simply disregarding Ω^- from the analysis unless the maximum error observed in Ω^- is of the same magnitude as the maximum variations of the solutions in Ω^+.

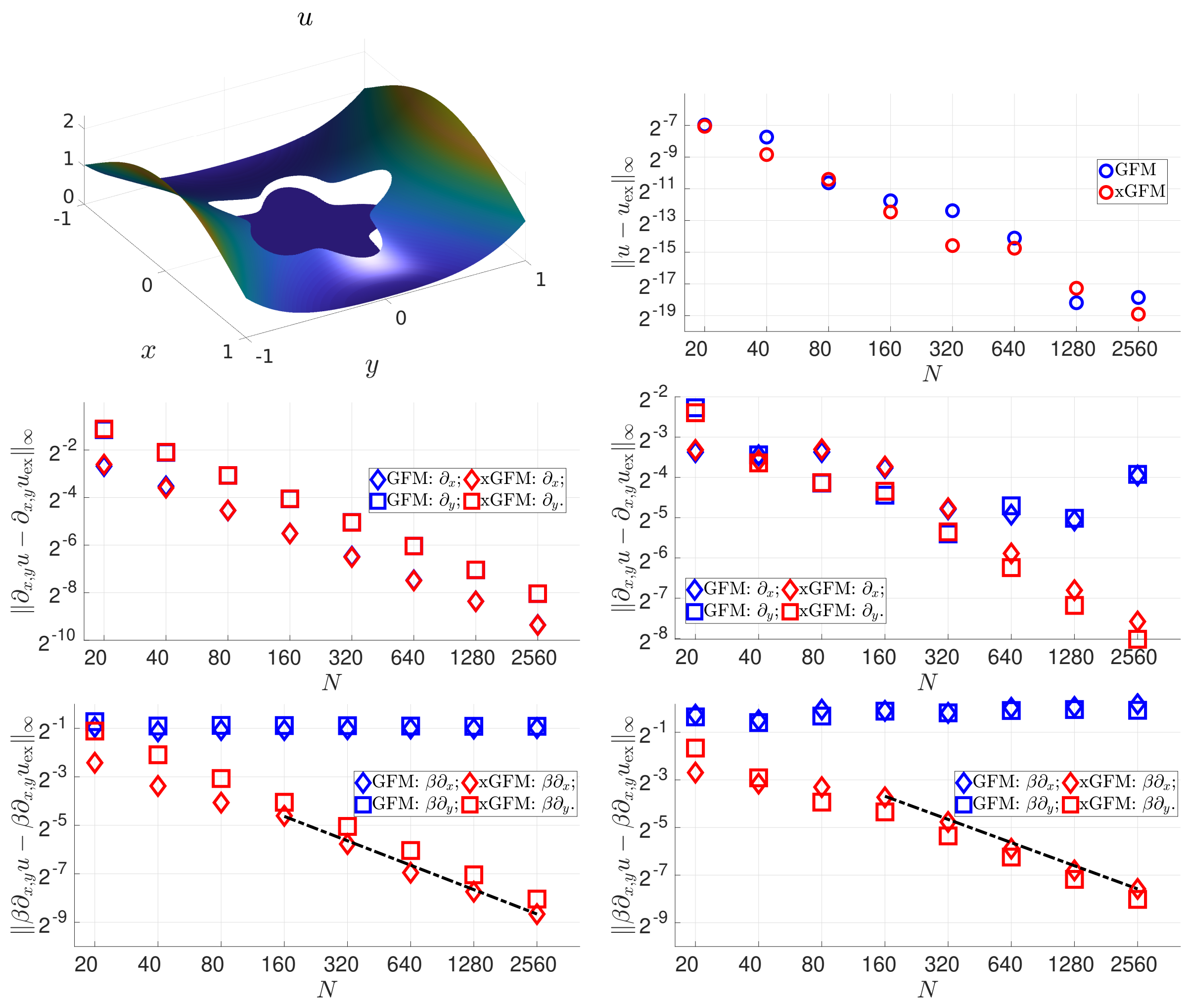


Figure 4.15: convergence analysis for the example of subsection 4.4.6. Top left: illustration of the solution for the example of subsection 4.4.6, with $N = 640$. Top right: errors in infinity norm for the solution field. Center left: errors in infinity norm for the intrinsically one-sided partial derivatives (no significant difference between blue and red markers in this case). Center right: errors in infinity norm for the partial derivatives evaluated with (4.13). Bottom left: errors in infinity norm for the intrinsically one-sided flux components, the slope of the dashed line is -1.01. Bottom right: errors in infinity norm for the flux components evaluated with (4.13), the slope of the dashed line is -0.97.

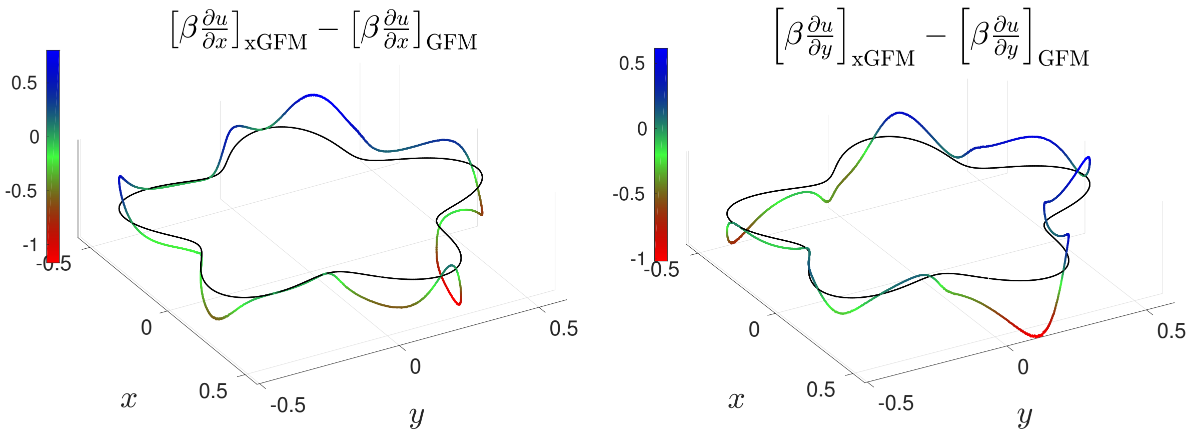


Figure 4.16: differences, along the interface Γ, in the jumps $\left[\beta\dfrac{\partial u}{\partial x}\right]$ (left) and $\left[\beta\dfrac{\partial u}{\partial y}\right]$ (right), as enforced by the xGFM and the GFM, for the problem of subsection 4.4.6.

N	Linear interpolation of ϕ		Quadratic interpolation of ϕ	
	GFM	xGFM	GFM	xGFM
20	50	324	48	313
40	83	334	82	327
80	188	731	188	737
160	430	1521	436	1547
320	1088	3586	1090	3590
640	2577	7012	2603	7034
1280	5713	13627	5709	13593
2560	11277	24044	11255	23970

20	0.076 s	0.182 s	+239.9%	0.024 s	+31.9%
40	0.020 s	0.137 s	+685.2%	0.020 s	+97.4%
80	0.105 s	0.295 s	+279.8%	0.219 s	+208.0%
160	0.727 s	0.549 s	+75.5%	1.481 s	+203.7%
320	6.256 s	1.165 s	+18.6%	13.56 s	+216.7%
640	51.68 s	2.482 s	+4.8%	86.96 s	+168.2%
1280	421.2 s	12.32 s	+2.9%	581.2 s	+138.0%
2560	4306 s	24.94 s	+0.6%	4843 s	+112.5%

Table 4.4: cumulative number of iterations of the preconditioned conjugate gradient solver for the problem of subsection 4.4.6. Linear (resp. quadratic) interpolation of ϕ means that the values of the θ parameters (as defined in section 4.2) are evaluated as the roots of a local linear (resp. quadratic) interpolant of ϕ.

4.5 Adaptive grids and three-dimensional illustrations

A significant advantage of our suggested approach is the optimally small stencil used in both the discretization of the jump problem (see section 4.2) and the extension technique (see section 4.3). Since every elementary operation involves only the direct Cartesian neighbor, the method does not require prohibitively thick layers of ghost cells when using a distributed memory paradigm. Moreover, the small stencil size and the focus of the technique onto the jump nodes makes it fairly straightforward to couple with a solver exploiting an adaptive grid far away from the interface.

In this section, we illustrate the above advantages with two- and three-dimensional problems addressed by coupling the suggested technique with the method introduced in [55] for stable projection in simulations of incompressible flows on quadtree or octree grids. The full solver was parallelized using distributed memory, exploiting the `p4est` library for grid management [174] and the `PetSc` library for handling the linear algebra aspects [111, 112].

4.5.1 A multiscale two-dimensional problem

In this example, we consider $\Omega = [-1, 1] \times [-1, 1]$. The interface is defined as 15 non-overlapping small inclusions randomly located in the domain. These inclusions are individually defined as circles deformed under a fictitious parabolic flow field of random orientation; the radius of those circles (before deformations) is randomly chosen in $[0.005;\ 0.02]$. The corresponding levelset function is defined as negative within the inclusions. The coefficients are $\beta^- = 10^3$ and $\beta^+ = 1$. The exact solution is chosen to be

$$\begin{cases} u\left(x,\, y\right) = \dfrac{\cos\left(200\pi\left(x+3y\right)\right) - \sin\left(100\pi\left(y-2x\right)\right)}{\beta^-}, & \forall \left(x,\, y\right) \in \Omega^-, \\ u\left(x,\, y\right) = \cos\left(x+y\right)\exp\left(-x^2\cos^2\left(y\right)\right), & \forall \left(x,\, y\right) \in \Omega^+, \end{cases}$$

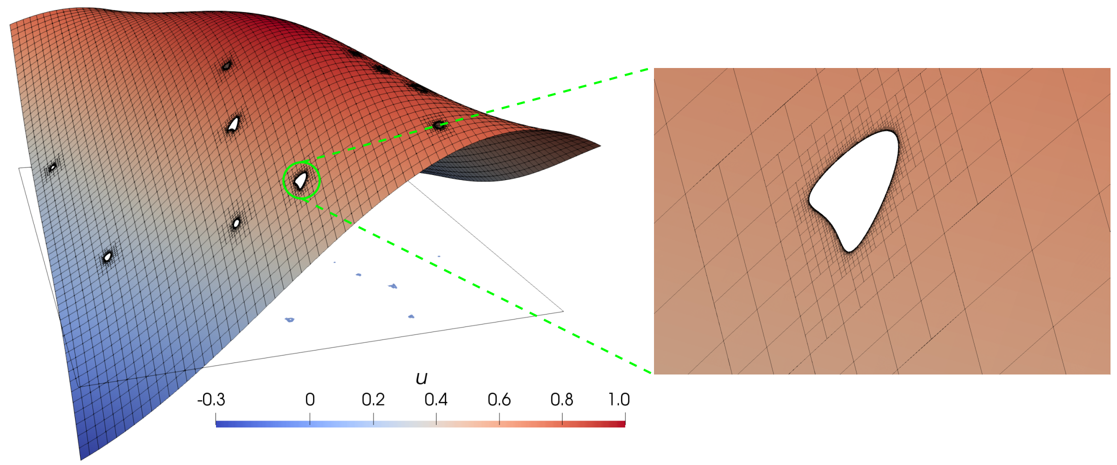
the right-hand side f and the jump terms a and b are defined accordingly. The interfaces, an adaptive grid and the analytical solutions are illustrated in Figure 4.17.

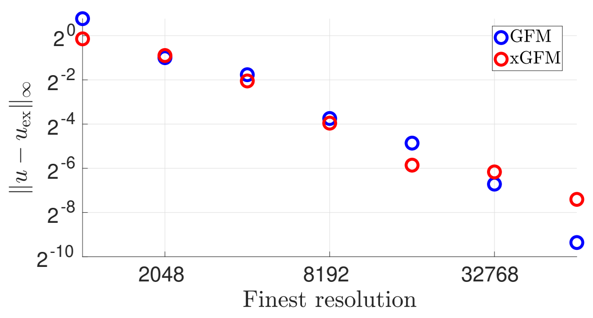
In order to capture the details of the interfaces, a grid resolution of $h \sim 0.002$ (at least) is required. Using uniform grids, this would lead to $\mathcal{O}\left(10^6\right)$ unknowns (at least) although most of the computational domain does not require such a resolution: the inclusions represent a small portion of the overall computational domain and the variations of the solution in Ω^+ occur over (much) larger length scales. Therefore, a significant amount of computational resources can be saved by using an adaptive grid with equivalent finest level of refinement. In this example, the grid cells are made smaller as they approach Γ in Ω^+ and are set to their smallest size in Ω^-. The coarsest cells are 256 times bigger (along every Cartesian direction) than the smallest cells. The convergence results are illustrated in Figure 4.17 and show that the xGFM recovers convergence for $\beta\nabla u$ once again. Note that the maximum finest resolution that we consider with such an adaptive approach would lead to $\mathcal{O}\left(10^9\right)$ unknowns with a uniform grid.

4.5.2 A three-dimensional problem with a convoluted interface

In this example, we consider the three-dimensional domain $\Omega = [-2,\, 2] \times [-2,\, 2] \times [-2,\, 2]$. The interface is the surface parameterized by

$$r\left(\vartheta, \varphi\right) = \frac{3}{4} + \frac{5 - 3\cos\left(6\varphi\right)\left(1-\cos\left(6\vartheta\right)\right)}{25}, \quad \vartheta \in \left[0,\, \pi\right], \varphi \in \left[0,\, 2\pi\right[\tag{4.37}$$





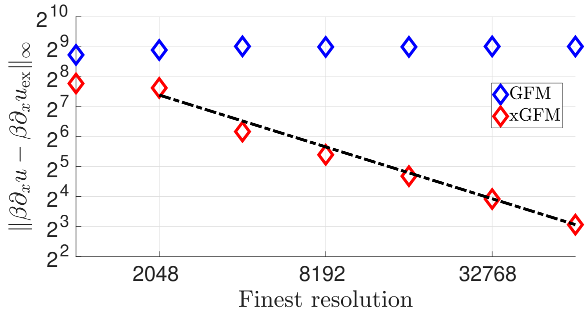


Figure 4.17: illustrations and results for the problem addressed in section 4.5.1. Top left: computational domain and grid, elevated and colored by the analytical solution. Top right: zoom-in on one of the inclusions. Bottom left: errors in infinity norm for the solution field. Bottom right: errors in infinity norm for the x-component of the flux vector (the y-component behaves similarly), the slope of the dashed line is -0.87.

in standard spherical coordinates; the corresponding levelset function is built as negative inside and positive outside. The coefficients are $\beta^- = 1$ and $\beta^+ = 1250$. The exact solution is chosen to be

$$\begin{cases} u\left(x,\,y\right) = \exp\left(\dfrac{x-z}{2}\right)\left(x\sin\left(y\right) - \cos\left(x+y\right)\arctan\left(z\right)\right), & \forall\left(x,\,y\right)\in\Omega^-, \\ u\left(x,\,y\right) = -1 + \dfrac{5\arctan\left(\dfrac{x^3y}{10} + 2z\cos\left(y\right) - y\sin\left(x+z\right)\right)}{2\beta^+}, & \forall\left(x,\,y\right)\in\Omega^+, \end{cases}$$

the right-hand side f and the jump terms a and b are defined accordingly. The interface, the grid and its partition are illustrated in Figure 4.18. The coarsest cells are 8 times bigger (along every Cartesian direction) than the smallest cells. This figure also shows that xGFM recovers convergence for $\beta\nabla u$. These results were obtained using either 40 cores on a local workstation running a Dual Intel Xeon Gold 6148 processor with 64 GB of RAM or Stampede2 supercomputer for the finest grids.

4.5.3 A three-dimensional problem with a not radially convex interface

For this final example, we consider the three-dimensional domain $\Omega = [-1.5,\,1.5]\times[-1.5,\,1.5]\times[-1.5,\,1.5]$. The interface is defined as the revolution of the bone-shaped interface from section 4.4.5 around one of its axis of symmetry. Periodic boundary conditions are considered and the interface is translated so that it lies crosses all computational boundaries, as illustrated in Figure 4.19. The coefficients are $\beta^- = 1$ and $\beta^+ = 80$. The exact solution is chosen to be

$$\begin{cases} u\left(x,\,y\right) = \arctan\left(\sin\left(\dfrac{2\pi}{3}\left(2x-y\right)\right)\right)\log\left(\dfrac{3}{2} + \cos\left(\dfrac{2\pi}{3}\left(2y-z\right)\right)\right), & \forall\left(x,\,y\right)\in\Omega^-, \\ u\left(x,\,y\right) = \tanh\left(\cos\left(\dfrac{2\pi}{3}\left(2x+y\right)\right)\right)\arccos\left(\dfrac{1}{2}\sin\left(\dfrac{2\pi}{3}\left(2z-x\right)\right)\right), & \forall\left(x,\,y\right)\in\Omega^+, \end{cases}$$

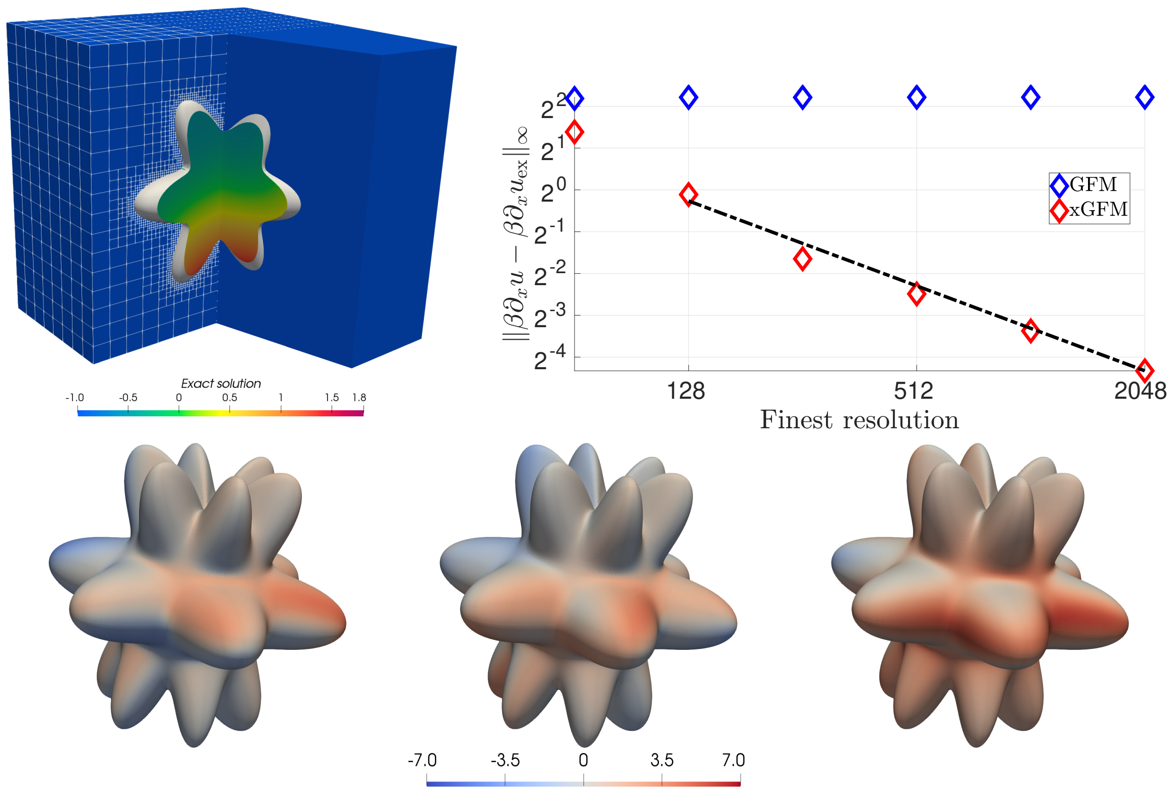
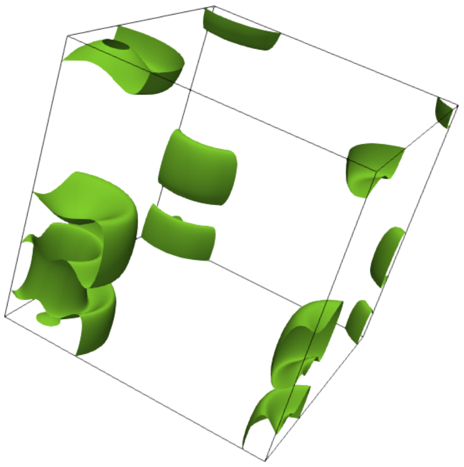


Figure 4.18: top left: illustration of the star-shaped interface (truncated, in white), the grid and the exact solution for the problem addressed in section 4.5.2; top right: convergence analysis in infinity norm for the flux component $\beta\partial_x u$ (the other flux components $\beta\partial_y u$ and $\beta\partial_z u$ behave very similarly), the slope of the dashed line is -1.02. Bottom: illustration of $[\beta\nabla u]_{\mathrm{xGFM}} - [\beta\nabla u]_{\mathrm{GFM}} = \beta^- \left(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}\right) \cdot \nabla a + [\beta] \left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}\right) \cdot \nabla u\right)^+$ onto the interface, as calculated by the solver (x, y and z components from left to right, respectively).



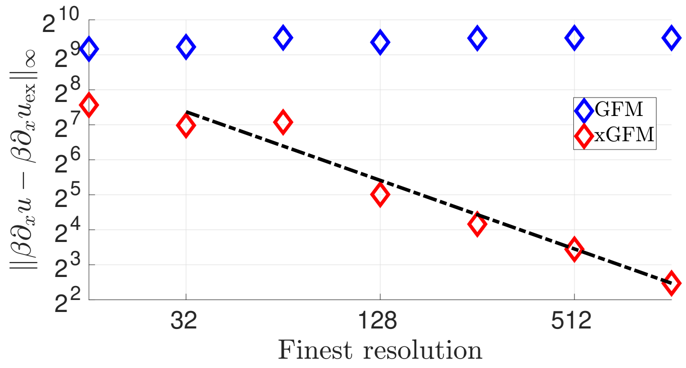


Figure 4.19: left: illustration of the interface for the problem addressed in section 4.5.3; right: convergence analysis in infinity norm for the flux component $\beta\partial_x u$ (the other flux components $\beta\partial_y u$ and $\beta\partial_z u$ behave very similarly), the slope of the dashed line is -0.98.

the right-hand side f and the jump terms a and b are defined accordingly. As illustrated in Figure 4.19, convergent $\beta\nabla u$ are recovered everywhere in the domain here again.

4.6 Summary

We introduced the xGFM, a robust numerical method that recovers consistency in jump conditions when using the 'boundary condition capturing method' by Liu, Fedkiw and Kang [16] (the Ghost Fluid Method) for capturing sharp interface conditions when solving Poisson equations with discontinuities on a Cartesian grid. The strategy favors the symmetric positive definiteness of the linear system of equations to be solved and adopts an iterative technique to update correcting terms. The original inconsistency was shown to be responsible for the lack of convergence for the gradient of the solution (and thus of the flux). Point-wise convergence of the solution and its partial derivatives was shown and illustrated on several two- and three-dimensional problems, including cases with large ratios of β coefficients. The approach was successfully implemented within a parallel framework for adaptive quadtree/octree grids, in a distributed memory framework. Future work should investigate the possibility to formulate the suggested method into a single (possible not semi-positive definite) system of linear equations and/or theoretically investigate the formal rates of convergence (for the solution and its derivatives) and numerical properties of the suggested method.

Chapter 5

Sharp and implicit treatment of viscous terms

5.1 Introduction

In this chapter, we present an extension of the strategy presented in Chapter 4 for the resolution of elliptic interface problems of the kind (using index notations and Einstein Summation Convention)

$$\begin{cases} K\boldsymbol{u} - \mu\nabla^2\boldsymbol{u} = \boldsymbol{r} \\ [\boldsymbol{u}] = m\boldsymbol{n} \\ [\mu\partial_j u_i] = \mu^{\pm}\left(\delta_{jk} - n_j n_k\right)\partial_k\left(m n_i\right) + [\mu]\left(\delta_{jk} - n_j n_k\right)\partial_k u_i^{\mp} \\ \qquad -\left(\delta_{ir} - n_i n_r\right)T_r\ n_j \\ \qquad -\mu^{\pm}\ n_j\left(\delta_{ir} - n_i n_r\right)\partial_r\left(m\right) - [\mu]\ \ n_j\left(\delta_{ir} - n_i n_r\right)\partial_r u_k^{\mp}\ n_k \\ \qquad -\mu^{\pm}\ n_i n_j\left(\kappa m\right) + [\mu]\ \ n_i n_j n_r \partial_k u_r^{\mp} n_k \end{cases} \tag{5.1}$$

for the vector field $\boldsymbol{u} = u_1\boldsymbol{e}_1 + u_2\boldsymbol{e}_2 + u_3\boldsymbol{e}_3$, with components u_i sampled at the faces of normal $\boldsymbol{e}_i$. In (5.1), K and μ are (piecewise) constant, positive scalars, $\boldsymbol{r}$ and $\boldsymbol{T}$ are known vector fields, and m is a known scalar field.

As outlined in Chapter 3, this problem is relevant for the "viscosity step", when treating viscous terms implicitly in simulations of incompressible viscous two-phase flows, while accounting for stress balance across the interface. In that context, μ is the local shear viscosity, $K = \dfrac{\rho\,(2\Delta t_n + \Delta t_{n-1})}{(\Delta t_n + \Delta t_{n-1})\,\Delta t_n}$ at time step n, $\boldsymbol{r}$ accounts for discretized advection terms as well as pressure gradients, $m = \dot{M}\,[1/\rho]$ and $\boldsymbol{T} = \nabla\gamma + \boldsymbol{G}$. As illustrated in Figure 3.1, we denote by Ω the computational domain, and by Γ the interface which separates Ω into Ω^+ and Ω^- (based on the sign of the levelset of zero-contour Γ). The unit vector normal to Γ, pointing toward Ω^+ is $\boldsymbol{n}$.

Building upon the strategy developed in single-phase flows, we make use of a finite-volume discretization leveraging Voronoi tessellations of the computational domain, away from the interface. This strategy effectively decouples the velocity components and creates three (two in 2D) independent linear systems of equations for the separate components of the vector field $\boldsymbol{u}$. Although this approach provides an elegant solution for handling non-uniform regions of the computational grid, it brings its share of complications when extended to tackle interface problems like (5.1). Indeed, the interface conditions on $[\mu\partial_j u_i]$ show that interface discontinuities are solution-dependent *and* that the velocity components are actually coupled across the interface (as a direct consequence of the symmetry of the viscous stress tensor). Therefore, however the single-phase approach is extended to capture interface discontinuities, this latter observation highlights the need for an iterative strategy: a solution that is self-consistent with its interface conditions cannot be determined in a single pass through 3 (2 in 2D) decoupled, independent solvers.

In this chapter, we present an extension of the strategy presented in Chapter 4 to tackle vector problems like (5.1). We first present a discretization that couples a finite-volume discretization on Voronoi tessellations away from the interface to the dimension-by-dimension, finite difference discretization from [16] across the interface, in order to capture interface conditions in a sharp fashion. This tool is thus suited for (over-determined) problems like

$$\begin{cases} K\boldsymbol{u} - \mu\nabla^2\boldsymbol{u} = \boldsymbol{r} \\ [\boldsymbol{u}] = m\boldsymbol{n} \\ [\mu\partial_j u_i] = S_{ij} \end{cases} \tag{5.2}$$

for a known tensor field $\underline{\boldsymbol{S}}$. The equivalence between (5.1) and (5.2) is then recovered by incorporating the dependence of $\underline{\boldsymbol{S}}$ on the solution $\boldsymbol{u}$ itself, via an iterative method, similarly to [28] (Chapter 4). This iterative strategy relies on extensions of interface-defined values of the solution (as well as extrapolations, in this particular vector case): a robust extension method for face-sampled fields is required as well. We thus introduce PDE-based extension and extrapolation techniques inspired from [172], and adapted to face-sampled fields on adaptive Quadtree/Octree grids. These two main elements (i.e., a numerical method for (5.2) along with the extension and extrapolation tools) may then be combined to tackle problems like (5.1). After presenting that iterative strategy, its performance (and the correctness of the implementations) is assessed by analyzing the accuracy of the solver on a three-dimensional Stokes flow problem that involves a gradient of surface tension (i.e., a Marangoni force).

About the problem formulation. In this work, we have chosen to use the dimension-by-dimension approach from [16] for handling interface discontinuities on velocity components. This choice was motivated by the simplicity of implementation, the robustness of the method and its low computational cost (compared to other methods which would produce nonsymmetric matrices) in light of the need for embedding it within an iterative procedure. Therefore, every component of $[\mu\nabla\boldsymbol{u}]$ needs to be prescribed, hence the problem formulation (5.1) considered herein. If another interface-capturing scheme requiring the discontinuity in normal flux only was used for the separate velocity components, the problem formulation would require $[\mu \nabla \boldsymbol{u} \cdot \boldsymbol{n}]$ to be prescribed instead, as discussed in Chapter 3. However, this does not alleviate the dependence of the interface conditions on the solution itself: hence, this approach was not preferred in light of the increased complexity and computation cost associated with such methods.

5.2 Numerical discretization

In this section, we present the numerical discretization of (5.2), which is a cornerstone of the iterative method designed for solving (5.1). We distinguish cases away from the interface (similar to the single-phase approach) and faces involving neighbors across the interface.

5.2.1 Discretization away from the interface

Away from the interface, i.e., for faces of the computational grids that have all their relevant neighbor faces in the same subdomain, we use a finite-volume approach on Voronoi cells, seeded at the face centers. Given a set of seeds (center points of Voronoi cells), the Voronoi cell associated with one such seed is the volume (area in two dimensions) containing all points that are closer to that seed than to any other. As illustrated in Figure 5.1, the construction procedure can be done locally, provided that all closest direct neighbors are known. In the context of interest, we consider three (resp. two) Voronoi tessellations, constructed by placing the seeds at the centers of the faces of the computational grid of similar orientation. One such cell is illustrated in Figure 5.1 for faces of normal $\boldsymbol{e}_1$. It can be shown that the set of faces associated with all second-degree neighbor cells of the cell owning the seed of interest contains all relevant neighbor seeds, for graded Quadtree/Octree grids of moderate aspect ratio, as we consider in applications (see [52] for considerations of large aspect ratio). We use the `voro++` library [175]

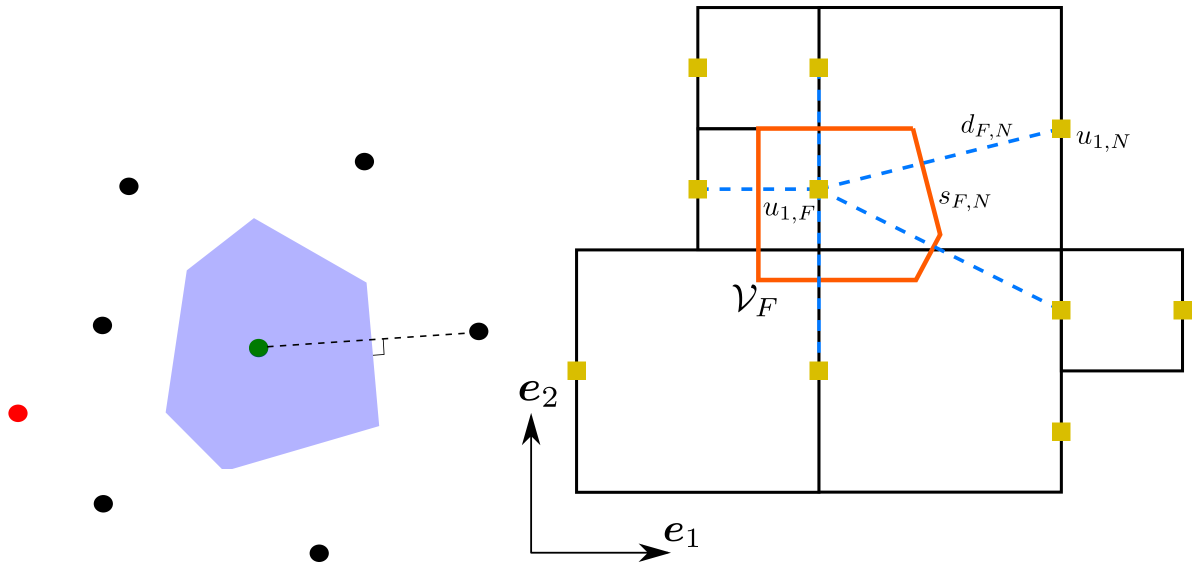


Figure 5.1: two-dimensional illustration of Voronoi cell construction and notations used for the finite-volume discretization based on face-seeded Voronoi tessellations. Left: illustration of the Voronoi cell associated with the green seed, constructed based on an unstructured set of neighbor seeds in black. Provided that all the relevant closest neighbors are found, this construction may be done locally: if provided, the red neighbor is discarded as it does not contribute to the cell's construction. Right: illustration of the discretization notations for a voronoi cell $\mathcal{V}_F$ associated with face F of normal $\boldsymbol{e}_1$.

to build such Voronoi cells in three dimensions.

Let us consider a face F of normal $\boldsymbol{e}_i$. Let $\mathcal{V}_F$ be the Voronoi cell associated with that face and $\mathrm{ngbd}\,(F)$ the set of neighbor faces of F, of normal $\boldsymbol{e}_i$, that are involved in the construction of $\mathcal{V}_F$. For every neighbor face N involved in $\mathcal{V}_F$'s construction, we denote by $d_{F,N}$ the distance between the centers of F and N and by $s_{F,N}$ the area (length in 2D) of the surface of $\mathcal{V}_F$ orthogonal to the segment joining F and N, as illustrated in Figure 5.1.

Integrating $Ku_i - \mu\nabla^2 u_i = r_i$ side-by-side over $\mathcal{V}_F$, one has

$$K\int_{\mathcal{V}_F} u_i \,\mathrm{d}\mathcal{V}_F - \int_{\partial\mathcal{V}_f} \mu\nabla u_i \cdot \boldsymbol{\eta}\,\mathrm{d}\partial\mathcal{V}_F = \int_{\mathcal{V}_F} r_i \,\mathrm{d}\mathcal{V}_F, \tag{5.3}$$

wherein $\boldsymbol{\eta}$ is the unit vector, normal to $\partial\mathcal{V}_F$ pointing outward $\mathcal{V}_F$. Since the faces of $\mathcal{V}_F$

are orthogonal to segments joining F and the relevant face neighbors $N \in \operatorname{ngbd}(F)$, one may use a finite-difference discretization for the net fluxes out of $\mathcal{V}_F$, that is

$$\int_{\partial \mathcal{V}_f} \mu \nabla u_i \cdot \boldsymbol{\eta} \, \mathrm{d}\partial \mathcal{V}_F \approx \sum_{N \in \operatorname{ngbd}(F)} \mu \frac{u_{i,N} - u_{i,F}}{d_{F,N}} s_{F,N},$$

where $u_{i,F}$ is the value of the i^{th} velocity component, sampled at face F (of normal $\boldsymbol{e}_i$). Approximating volume integrals to their leading term, we obtain the final discretization form

$$K \operatorname{vol}(\mathcal{V}_F)\, u_{i,F} + \sum_{N \in \operatorname{ngbd}(F)} \mu \frac{u_{i,F} - u_{i,N}}{d_{F,N}} s_{F,N} = \operatorname{vol}(\mathcal{V}_F)\, r_{i,F}. \tag{5.4}$$

For faces nearby walls of the computational domain, the Voronoi cells are clipped to the limits of the computational domain. Corresponding wall boundary condition are then invoked for the discretization of the relevant flux on the faces of the Voronoi cells matching domain boundaries. For Neumann boundary conditions, the flux value is prescribed and used right away. For Dirichlet boundary conditions, the flux value is estimated by linear interpolation between the cell's seed and the value prescribed at the wall, or the value at the cell's seed is prescribed right away if the face is a wall itself.

5.2.2 Discretization capturing interface discontinuities

For faces of uniform neighborhood, Voronoi cells are parallelepipeds and the discretization (5.4) is equivalent to the standard 7-point (5-point, in 2D) stencil discretization, multiplied by the volume of the cell. This enables the method to be naturally coupled to a finite-difference approach close to the interface, as we enforce a band of uniform cells across the interface.

Let F be a face of normal $\boldsymbol{e}_i$ in a locally uniform patch of the computational grid. We denote by $F_{\pm \boldsymbol{e}_j}$ the neighbor face of F (of same orientation) found in direction $\pm \boldsymbol{e}_j$.

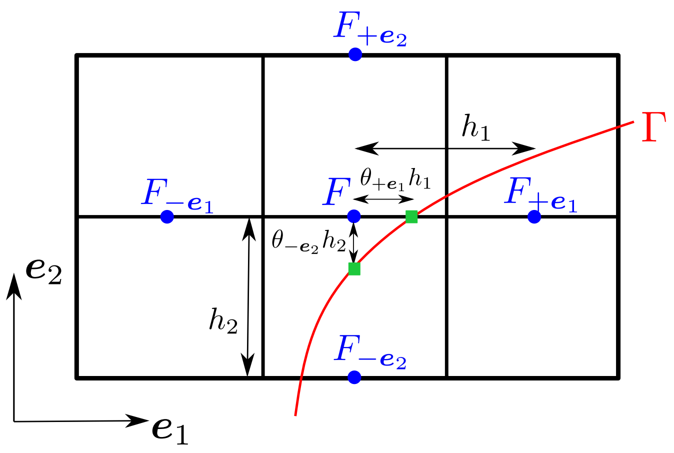


Figure 5.2: illustration of the notations used in subsection 5.2.2 for faces in a locally uniform patch of the computational grid, possibly crossed by the interface. The neighbor face of F (of same orientation) found in direction $\pm\boldsymbol{e}_j$ is denoted by $F_{\pm\boldsymbol{e}_j}$.

We denote by h_j the size of the computational cells along direction $\boldsymbol{e}_j$. If F and $F_{\pm\boldsymbol{e}_j}$ lie across the interface, the distance between F and the intersection point between the interface Γ and the line joining F and $F_{\pm\boldsymbol{e}_j}$ is denoted by $\theta_{\pm\boldsymbol{e}_j}h_j$ ($\theta_{\pm\boldsymbol{e}_j} \in \,]0; 1[$). The notations are illustrated in Figure 5.2, in two dimensions for a face of normal $\boldsymbol{e}_2$. A first-order approximation of $\theta_{\pm\boldsymbol{e}_j}$ can be obtained by linear interpolation of the levelset values between F and $F_{\pm\boldsymbol{e}_j}$, i.e., $\theta_{\pm\boldsymbol{e}_j} = \dfrac{|\phi_F|}{|\phi_F| + \left|\phi_{F_{\pm\boldsymbol{e}_j}}\right|}$. A second-order evaluation may be derived similarly and requires second derivatives of the levelset function.

We follow a sharp, dimension-by-dimension approach (see [16, 15, 28]) for capturing interface discontinuities on every component of $\boldsymbol{u}$, separately. Let $s = +$ (resp. $s = -$) if the center of F is in Ω^+ (resp. Ω^-); we denote by $-s$ the sign opposite to s. Considering the i^{th} component of $\boldsymbol{u}$ in (5.2), we have the following discretization at face F

$$Ku_{i,F} + \sum_{j=1}^{3}\left(\hat{\mu}_{+\boldsymbol{e}_j}\frac{u_{i,F} - u_{i,F_{+\boldsymbol{e}_j}}}{h_j^2} + \hat{\mu}_{-\boldsymbol{e}_j}\frac{u_{i,F} - u_{i,F_{-\boldsymbol{e}_j}}}{h_j^2}\right) = r_{i,F} + \sum_{j=1}^{3}\left(\hat{\mu}_{+\boldsymbol{e}_j}J_{+\boldsymbol{e}_j} + \hat{\mu}_{-\boldsymbol{e}_j}J_{-\boldsymbol{e}_j}\right) \tag{5.5}$$

where

$$\hat{\mu}_{\pm\boldsymbol{e}_j} = \begin{cases} \mu^s & \text{if } F \text{ and } F_{\pm\boldsymbol{e}_j} \text{ are both in } \Omega^s, \\ \dfrac{\mu^s \mu^{-s}}{\left(1-\theta_{\pm\boldsymbol{e}_j}\right)\mu^s + \theta_{\pm\boldsymbol{e}_j}\mu^{-s}} & \text{if } F \in \Omega^s \text{ and } F_{\pm\boldsymbol{e}_j} \in \Omega^{-s}, \end{cases} \tag{5.6}$$

and the contribution J of the interface discontinuities to the right-hand side of (5.5) are

$$J_{\pm\boldsymbol{e}_j} = \begin{cases} 0 & \text{if } F \text{ and } F_{\pm\boldsymbol{e}_j} \text{ are both in } \Omega^s, \\ s\left(\pm\dfrac{\left(1-\theta_{\pm\boldsymbol{e}_j}\right)S_{ij}}{\mu^{-s}h_j} + \dfrac{mn_i}{h_j^2}\right)\Bigg|_{F_{\pm\theta\boldsymbol{e}_j}} & \text{if } F \in \Omega^s \text{ and } F_{\pm\boldsymbol{e}_j} \in \Omega^{-s}. \end{cases} \tag{5.7}$$

In (5.7), $(\cdot)|_{F_{\pm\theta\boldsymbol{e}_j}}$ indicates that $(\cdot)$ is evaluated at the intersection point between the interface and the line joining F and $F_{\pm\boldsymbol{e}_j}$ (see green squares in Figure 5.2, for instance).

As presented in [28], this discretization is consistent with the definition of interface values at intersection points between the interface and the line joining face centers across the interface (green squares in Figure 5.2). If $F_{\pm\boldsymbol{e}_j}$ is a neighbor face across the interface, one may define $u^s_{i,F_{\pm\theta\boldsymbol{e}_j}}$ as the value of u^s_i at the intersection point between the interface and the line joining F and $F_{\pm\boldsymbol{e}_j}$. After calculations, this value is

$$\begin{aligned} u^s_{i,F_{\pm\theta\boldsymbol{e}_j}} = {} & \frac{\left(1-\theta_{\pm\boldsymbol{e}_j}\right)\mu^s}{\tilde{\mu}_{\pm\boldsymbol{e}_j}}u_{i,F} + \frac{\theta_{\pm\boldsymbol{e}_j}\mu^{-s}}{\tilde{\mu}_{\pm\boldsymbol{e}_j}}u_{i,F_{\pm\boldsymbol{e}_j}} \\ & \pm\frac{s\left(1-\theta_{\pm\boldsymbol{e}_j}\right)\theta_{\pm\boldsymbol{e}_j}h_j}{\tilde{\mu}_{\pm\boldsymbol{e}_j}}\, S_{ij}|_{F_{\pm\theta\boldsymbol{e}_j}} + s\frac{\theta_{\pm\boldsymbol{e}_j}\mu^{-s}}{\tilde{\mu}_{\pm\boldsymbol{e}_j}}\,(mn_i)|_{F_{\pm\theta\boldsymbol{e}_j}} \end{aligned} \tag{5.8}$$

where $\tilde{\mu}_{\pm\boldsymbol{e}_j} = \left(1-\theta_{\pm\boldsymbol{e}_j}\right)\mu^s + \theta_{\pm\boldsymbol{e}_j}\mu^{-s}$. Similarly, one may define

$$u^{-s}_{i,F_{\pm\theta\boldsymbol{e}_j}} = u^s_{i,F_{\pm\theta\boldsymbol{e}_j}} - s\,(mn_i)|_{F_{\pm\theta\boldsymbol{e}_j}}$$

as the value of u_i^{-s} at the intersection point between the interface and the line joining F

and $F_{\pm\boldsymbol{e}_j}$. We obtain

$$\begin{aligned} u^{-s}_{i,F_{\pm\theta\boldsymbol{e}_j}} = & \frac{\left(1-\theta_{\pm\boldsymbol{e}_j}\right)\mu^{s}}{\tilde{\mu}_{\pm\boldsymbol{e}_j}}u_{i,F} + \frac{\theta_{\pm\boldsymbol{e}_j}\mu^{-s}}{\tilde{\mu}_{\pm\boldsymbol{e}_j}}u_{i,F_{\pm\boldsymbol{e}_j}} \\ & \pm\frac{s\left(1-\theta_{\pm\boldsymbol{e}_j}\right)\theta_{\pm\boldsymbol{e}_j}h_j}{\tilde{\mu}_{\pm\boldsymbol{e}_j}}\left.S_{ij}\right|_{F_{\pm\theta\boldsymbol{e}_j}} - s\frac{\left(1-\theta_{\pm\boldsymbol{e}_j}\right)\mu^{s}}{\tilde{\mu}_{\pm\boldsymbol{e}_j}}\left.(mn_i)\right|_{F_{\pm\theta\boldsymbol{e}_j}}. \end{aligned} \tag{5.9}$$

The interface-defined values (5.8) and (5.9) are unambiguously defined and identical when evaluated from either side of the interface, so long as

- interface discontinuities S_{ij} and (mn_i) are evaluated at interface intersection points (green squares in Figure 5.2);
- the values of θ are consistent as seen from either face across the interface, i.e. they sum up to 1.

5.3 PDE-based extensions and extrapolations for face-sampled fields

In (5.1), $[\mu\partial_j u_i]$ are (implicitly meant) to be evaluated at the interface. Therefore, $(\delta_{jk} - n_j n_k)\,\partial_k u_i^{\mp}$ and $(\delta_{ir} - n_i n_r)\,\partial_r u_k^{\mp}$ actually represent tangential derivatives of interface-defined values. On the other hand, $n_k\partial_k u_r^{\mp}$ represents the normal derivatives of $u_r^{\mp}$ at the interface.

The one-sided evaluation of such derivatives for points close to the interface comes with its lot of complications (e.g. locally underdetermined and/or nonsymmetric differentiation stencil). In order to circumvent those difficulties, we rely on the creation of two alternative vector fields (sampled at faces, as well)

1. $\boldsymbol{u}^{\Gamma,\pm}$ which extends the relevant interface-defined values ((5.8) or (5.9)) normally to the interface;

2. $\boldsymbol{u}^{\pm}$ which extrapolates the value of $\boldsymbol{u}$ from $\Omega^{\pm}$.

By construction, those vector fields are well-defined in the neighborhood of any point close to the interface, which alleviates the difficulty of evaluating one-sided tangential or normal derivatives. We use $\boldsymbol{u}^{\Gamma,\pm}$ for the evaluation of tangential derivatives while $\boldsymbol{u}^{\pm}$ is used for the evaluation of normal derivatives.

In this section, we present an adaptation of the PDE-based extension and extrapolation strategies from [172] for face-sampled (components of) vector fields on Quadtree/Octree grids. Besides its robustness, this method presents one significant advantage compared to geometric methods[1]: it can be associated with a subcell-resolution strategy to capture intersections between the interface and grid lines, which allows for an extension of the interface-defined values (5.8) and (5.9), as required.

5.3.1 Extension of interface-defined values

Without loss of generality, let us consider that $\boldsymbol{u}^{\Gamma,-}$ is required, i.e., an extension of $\boldsymbol{u}$ from Γ^{-}. Mathematically, the ith component of $\boldsymbol{u}^{\Gamma,-}$ may be defined as the steady-state solution of

$$\left\{\begin{array}{l} \dfrac{\partial u_i^{\Gamma,-}}{\partial \tau} + \mathrm{sign}\left(\phi\right)\boldsymbol{n}\cdot\nabla u_i^{\Gamma,-} = 0 \\ u_i^{\Gamma,-}\left(\boldsymbol{x}\right) = u_i^{-}\left(\boldsymbol{x}\right), \quad \forall \boldsymbol{x}\in\Gamma \end{array}\right. 1 \tag{5.10}$$

in pseudo-time τ.

Let us consider first a face F in a locally uniform patch of the computational grid, which may have face neighbors across the interface (see Figure 5.2 for an illustration in 2D). Using first-order upwind finite differences with subcell resolution, the semi-discrete

[1]Geometric extrapolation relies on the construction of an interpolation polynomial of arbitrary degree along the line that follows the local normal vector. Points from the desired subdomain are fetched and sampled along this line, an interpolation polynomial is constructed to match those sampled values and evaluated at the point of interest thereafter. This technique may introduce undesirable local maxima and requires a good resolution of the queried subdomain.

form of (5.10) at face F reads

$$\frac{\mathrm{d}u_{i,F}^{\Gamma,-}}{\mathrm{d}\tau} = -\sum_{j=1}^{3}\left(\max\left(0, \operatorname{sign}\left(\phi_F\right) n_j\right)\frac{\delta_j^- u_{i,F}^{\Gamma,-}}{\delta_j^- h} + \min\left(0, \operatorname{sign}\left(\phi_F\right) n_j\right)\frac{\delta_j^+ u_{i,F}^{\Gamma,-}}{\delta_j^+ h}\right) \tag{5.11}$$

where the upwind finite-differences are defined as

$$\delta_j^\pm u_{i,F}^{\Gamma,-} = \begin{cases} \pm\left(u_{i,F_{\pm e_j}}^{\Gamma,-} - u_{i,F}^{\Gamma,-}\right) & \text{if } F \text{ and } F_{\pm \boldsymbol{e}_j} \text{ are both on the same side of } \Gamma,\\ \pm\left(u_{i,F_{\pm\theta e_j}}^{-} - u_{i,F}^{\Gamma,-}\right) & \text{if } F \text{ and } F_{\pm \boldsymbol{e}_j} \text{ are across } \Gamma \text{ (see (5.8) and (5.9))}, \end{cases} \tag{5.12}$$

and

$$\delta_j^\pm h = \begin{cases} h_j & \text{if } F \text{ and } F_{\pm \boldsymbol{e}_j} \text{ are both on the same side of } \Gamma,\\ \theta_{\pm \boldsymbol{e}_j} h_j & \text{if } F \text{ and } F_{\pm \boldsymbol{e}_j} \text{ are across } \Gamma. \end{cases} \tag{5.13}$$

We define the minimal discretization distance $\Delta^{\text{ext.}}$ associated with (5.11) as the minimal value of $\delta_j^\pm h$ involved in (5.11). If any relevant $\delta_j^\pm h$ is found too small, it means that F is too close to an interface-defined point and $u_{i,F}^{\Gamma,-}$ is set to that interface-defined value in such a case.

Second, let us consider nonuniform patches of the computational grid, i.e., away from the interface. Though the extension procedure must be specified in nonuniform regions as well for completeness, the lack of a structured neighborhood makes the definition of upwind differentiation operators more challenging locally. In such a case, $\operatorname{sign}(\phi)\,\boldsymbol{n}\cdot\nabla u_i^{\Gamma,-}$ is discretized using a least-square discretization for $\nabla u_i^{\Gamma,-}$ (see appendix B) constructed with the face neighbors N of F that satisfy the upwind condition $\operatorname{sign}(\phi_F)\,\boldsymbol{n}\cdot(\boldsymbol{x}_F - \boldsymbol{x}_N) \geq 0$ ($\boldsymbol{x}_F$ and $\boldsymbol{x}_N$ are the coordinates of the centers of faces F and N). Second-degree face neighbors are fetched if too few elements satisfy the upwind condition within the first-degree neighbors for constructing a least-square gradient operator. In this case, we define the minimal discretization distance $\Delta^{\text{ext.}}$ as the minimum value of $\|\boldsymbol{x}_F - \boldsymbol{x}_N\|$ for all neighbor face N involved in the construction of the discretized, least-square $\nabla u_i^{\Gamma,-}$.

Finally, for either of its semi-discrete form (i.e., using (5.11) or least-square upwind gradient operators), (5.10) is integrated 20 pseudo-time steps $\Delta\tau$ forward using an explicit forward Euler method. The pseudo-time step is locally adapted to the discretization distance via $\Delta\tau = \Delta^{\text{ext.}}/3$ (resp. $\Delta\tau = \Delta^{\text{ext.}}/2$ in 2D).

Note: for faces nearby boundaries of the computational domain, homogeneous Neumann boundary conditions are used on the walls of the domain for $u_i^{\Gamma,-}$.

5.3.2 Extrapolation of face-sampled vector fields

Without loss of generality, let us consider that $\boldsymbol{u}^-$ is required, i.e., an extrapolation of $\boldsymbol{u}$ from Ω^-, and let us consider the i^{th} component of $\boldsymbol{u}^-$ (the extrapolation of $\boldsymbol{u}^+$ follows the same principles, with a reverted normal vector). The extrapolation of u_i^- is a two-step procedure: first its normal derivative $u_{\boldsymbol{n},i}^-$ is extrapolated (constant extrapolation), then u_i^- itself is extrapolated (linear extrapolation).

Let us consider a face F of normal $\boldsymbol{e}_i$. We define two indicator functions

$$H_{i,F} = \begin{cases} 0 & \text{if } F \in \Omega^-, \\ 1 & \text{otherwise,} \end{cases} \tag{5.14}$$

and

$$H_{i,\boldsymbol{n},F} = \begin{cases} 0 & \text{if } F \text{ and all its neighbors} \in \Omega^-, \\ 1 & \text{otherwise,} \end{cases} \tag{5.15}$$

sampled at faces of normal $\boldsymbol{e}_i$.

At every face F such that $H_{i,\boldsymbol{n},F} = 0$, the normal derivative $u_{\boldsymbol{n},i}^-$ may be evaluated either as

$$u_{\boldsymbol{n},i,F}^- = \sum_{j=1}^{3} n_j \frac{u_{i,F_{+\boldsymbol{e}_j}}^- - u_{i,F_{-\boldsymbol{e}_j}}^-}{2h_j} \tag{5.16}$$

if F is in a locally uniform patch, or by means of least-square derivatives constructed with all its direct neighbors otherwise (see appendix B). Then, the values of $u^-_{\boldsymbol{n},i}$ are set in the rest of the domain by solving

$$\partial_\tau u^-_{\boldsymbol{n},i} + H_{i,\boldsymbol{n}} \boldsymbol{n} \cdot \nabla u^-_{\boldsymbol{n},i} = 0, \tag{5.17}$$

until steady-state in pseudo-time τ. Afterwards, the values of u^-_i are set in the rest of the domain by solving

$$\partial_\tau u^-_i + H_i \left(\boldsymbol{n} \cdot \nabla u^-_i - u^-_{\boldsymbol{n},i} \right) = 0, \tag{5.18}$$

until steady-state in pseudo-time τ.

In practice, (5.17) and (5.18) are solved for a finite number of pseudo-time steps using a forward Euler integration scheme in pseudo-time: 20 pseudo-time steps are used at every stage of the iterative strategy described hereafter[2]. Upwind discretization is used for the spatial derivatives in (5.17) and (5.18). For instance, $\boldsymbol{n} \cdot \nabla u^-_i$ is discretized as

$$\sum_{j=1}^{3} \max\left(n_j, 0\right) \frac{u^-_{i,F} - u^-_{i,F_{-\boldsymbol{e}_j}}}{h_j} + \min\left(n_j, 0\right) \frac{u^-_{i,F_{+\boldsymbol{e}_j}} - u^-_{i,F}}{h_j} \tag{5.19}$$

at face F if it belongs to a uniform patch of the computational grid. Upwind least-square derivatives are used otherwise, as described in subsection 5.3.1. The pseudo-time step $\Delta\tau$ is set to 1/3 (resp. 1/2 in 2D) of the minimal discretization distance involved in the calculation of upwind derivatives (adaptive pseudo-time stepping).

About the interface-defined values. The interface-defined values (5.8) and (5.9) are not used in the extrapolation procedure. Original attempts including them (with subcell resolution) in the initialization of $u^-_{\boldsymbol{n},i}$ resulted in numerically unstable simulations, in

[2]30 iterations are used at termination of the iterative strategy to ensure convergence of the extrapolations within a band of 3 smallest computational cells across the interface.

two-phase flow applications.

5.4 xGFM approach for vector fields

Without loss of generality, let us assume that we use the values from the negative subdomain[3] to recover solution-dependent terms in (5.1). On may isolate the solution-dependent terms in interface discontinuity conditions (5.1) by defining $\underline{\boldsymbol{F}}\left(\boldsymbol{a},\boldsymbol{b}\right)$ as (using Einstein Summation Convention)

$$\begin{aligned} F_{ij}\left(\boldsymbol{a},\boldsymbol{b}\right) \;=\; & \mu^{+}\left(\delta_{jk}-n_j n_k\right)\partial_k\left(m n_i\right)+\left[\mu\right]\left(\delta_{jk}-n_j n_k\right)\partial_k a_i \\ & -\left(\delta_{ir}-n_i n_r\right)T_r\ n_j \\ & -\mu^{+}\ n_j\left(\delta_{ir}-n_i n_r\right)\partial_r\left(m\right)-\left[\mu\right]\ n_j\left(\delta_{ir}-n_i n_r\right)\partial_r a_k\ n_k \\ & -\mu^{+}\ n_i n_j\left(\kappa m\right)+\left[\mu\right]\ n_i n_j n_r \partial_k b_r n_k, \end{aligned} \tag{5.20}$$

and recast the problem of interest (5.1) into

$$\begin{cases} K\boldsymbol{u}-\mu\nabla^2\boldsymbol{u}=\boldsymbol{r} \\ \left[\boldsymbol{u}\right]=m\boldsymbol{n} \\ \left[\mu\partial_j u_i\right]=F_{ij}\left(\boldsymbol{u}^{\Gamma,-},\boldsymbol{u}^{-}\right) \end{cases} \tag{5.21}$$

where $\boldsymbol{u}^{\Gamma,-}$ and $\boldsymbol{u}^{-}$ are respectively extension of interface-defined values (5.8) and (5.9) (which themselves must be self-consistent with the corresponding F_{ij}) and extrapolation of $\boldsymbol{u}$ from Ω^{-}.

Let $\mathtt{v_i}$ be the (column) array of the discretized i^{th} component of a vector field $\boldsymbol{v}$, sampled at faces of normal $\boldsymbol{e}_i$. We also denote by $\mathtt{A_i}$ the discretization matrix for component

[3]We follow the same convention as for the scalar-field problem [28] regarding which side of the interface is chosen for evaluating the solution-dependent terms: the upper (resp. lower) superscript is considered for $[\mu\nabla\boldsymbol{u}]$ in (5.1) when $\mu^{-}>\mu^{+}$ (resp. $\mu^{+}>\mu^{-}$).

u_i in (5.21) and by $\mathtt{j_i}\left(\mathtt{a},\mathtt{b}\right)$ the contribution of jump terms to the discretized right-hand side for component i when considering $[\mu\partial_j u_i] = F_{ij}\left(\boldsymbol{a},\boldsymbol{b}\right)$ (note that $\mathtt{j_i}$ is a multilinear map in $\mathtt{a}$ and $\mathtt{b}$). For any candidate solution $\boldsymbol{w}$, with extension $\boldsymbol{w}^{\Gamma,-}$ and extrapolation $\boldsymbol{w}^{-}$, one may measure its residual for component i via

$$\mathtt{res_i}\left(\boldsymbol{w},\boldsymbol{w}^{\Gamma,-},\boldsymbol{w}^{-}\right) = \mathtt{A_i}\mathtt{w_i} - \mathtt{r} - \mathtt{j_i}\left(\mathtt{w}^{\Gamma,-},\mathtt{w}^{-}\right). \tag{5.22}$$

Using this measure, the optimal (discrete) solution to (5.21) may be defined as the one minimizing all $\mathtt{res_i}\left(\boldsymbol{u},\boldsymbol{u}^{\Gamma,-},\boldsymbol{u}^{-}\right)$, in a given norm.

We present now the iterative method implemented to approach the solution $\boldsymbol{u}$ of (5.21) as a sequence of solutions $\boldsymbol{u}^k$ $(k = 0, 1, \ldots)$. We denote by $\boldsymbol{u}^{\Gamma,k,-}$ the extension of interface-defined values associated with iterate $\boldsymbol{u}^k$, and by $\boldsymbol{u}^{k,-}$ the extrapolation of $\boldsymbol{u}^k$ from Ω^-.

The method is initialized by defining $\boldsymbol{u}^0$ as the solution to

$$\begin{cases} K\boldsymbol{u}^0 - \mu\nabla^2\boldsymbol{u}^0 = \boldsymbol{r} \\ [\boldsymbol{u}^0] = m\boldsymbol{n} \\ [\mu\partial_j u_i^0] = F_{ij}\left(\boldsymbol{0},\boldsymbol{0}\right). \end{cases} \tag{5.23}$$

After resolution of (5.23), $\boldsymbol{u}^{\Gamma,0,-}$ and $\boldsymbol{u}^{0,-}$ are determined, as described in section 5.3 ($S_{ij} = F_{ij}\left(\boldsymbol{0},\boldsymbol{0}\right)$ being used for the evaluation of interface-defined values, consistently with (5.23)).

Thereafter, given an iterate $\boldsymbol{u}^k$ as well as it its extension $\boldsymbol{u}^{\Gamma,k,-}$ and extrapolation

$\boldsymbol{u}^{k,-}$, a subsequent fix-point update $\tilde{\boldsymbol{u}}^{k+1}$ may be defined as the solution to

$$\begin{cases} K\tilde{\boldsymbol{u}}^{k+1} - \mu\nabla^2\tilde{\boldsymbol{u}}^{k+1} = \boldsymbol{r} \\ \left[\tilde{\boldsymbol{u}}^{k+1}\right] = m\boldsymbol{n} \\ \left[\mu\partial_j\tilde{u}_i^{k+1}\right] = F_{ij}\left(\boldsymbol{u}^{\Gamma,k,-}, \boldsymbol{u}^{k,-}\right). \end{cases} \tag{5.24}$$

After resolution of (5.24), $\tilde{\boldsymbol{u}}^{\Gamma,k+1,-}$ and $\tilde{\boldsymbol{u}}^{k+1,-}$ are determined, as described in section 5.3 ($F_{ij}\left(\boldsymbol{u}^{\Gamma,k,-}, \boldsymbol{u}^{k,-}\right)$ being used for the evaluation of interface-defined values, consistently with (5.24)).

The next iterate $\boldsymbol{u}^{k+1}$, its extension $\boldsymbol{u}^{\Gamma,k+1,-}$ and extrapolation $\boldsymbol{u}^{k+1,-}$ are then defined as[4]

$$\begin{cases} \boldsymbol{u}^{k+1} &= \boldsymbol{u}^{k} + \eta_k\left(\tilde{\boldsymbol{u}}^{k+1} - \boldsymbol{u}^{k}\right) \\ \boldsymbol{u}^{\Gamma,k+1,-} &= \boldsymbol{u}^{\Gamma,k,-} + \eta_k\left(\tilde{\boldsymbol{u}}^{\Gamma,k+1,-} - \boldsymbol{u}^{\Gamma,k,-}\right) \\ \boldsymbol{u}^{k+1,-} &= \boldsymbol{u}^{k,-} + \eta_k\left(\tilde{\boldsymbol{u}}^{k+1,-} - \boldsymbol{u}^{k,-}\right) \end{cases} \tag{5.25}$$

where η_k is chosen to minimize the global residual measure

$$c_k = \sum_{\mathtt{i}} \left\|\mathtt{res}_{\mathtt{i}}\left(\boldsymbol{u}^{k+1}, \boldsymbol{u}^{\Gamma,k+1,-}, \boldsymbol{u}^{k+1,-}\right)\right\|_2^2. \tag{5.26}$$

By defining the global residual measure as (5.26), we indirectly recover coupling between velocity components and consider convergence on all components of the vector field of interest.

We define the residuals $\mathtt{e}_{\mathtt{i}}^{k+1} = \mathtt{res}_{\mathtt{i}}\left(\boldsymbol{u}^{k+1}, \boldsymbol{u}^{\Gamma,k+1,-}, \boldsymbol{u}^{k+1,-}\right)$, $\mathtt{e}_{\mathtt{i}}^{k} = \mathtt{res}_{\mathtt{i}}\left(\boldsymbol{u}^{k}, \boldsymbol{u}^{\Gamma,k,-}, \boldsymbol{u}^{k,-}\right)$

[4]Since extrapolated values do not depend linearly on the face-sampled data, defining $\boldsymbol{u}^{k+1,-}$ as $\boldsymbol{u}^{k,-} + \eta_k\left(\tilde{\boldsymbol{u}}^{k+1,-} - \boldsymbol{u}^{k,-}\right)$ is an approximation strictly speaking. However, only its normal derivatives are relevant within the solution-dependent interface conditions and the extrapolated normal derivatives *do* depend linearly on face-sampled data by construction. We refer the reader to the alternative approach described in subsection 5.4.1 for alleviating the need for extrapolations and strictly complying to linear dependence on discretized data, if desired.

and $\tilde{\mathtt{e}}_{\mathtt{i}}^{k+1} = \mathtt{res}_{\mathtt{i}}\left(\tilde{\boldsymbol{u}}^{k+1}, \tilde{\boldsymbol{u}}^{\Gamma,k+1,-}, \tilde{\boldsymbol{u}}^{k+1,-}\right)$. Since $\mathtt{j}_{\mathtt{i}}$ are multilinear maps, we have

$$\mathtt{e}_{\mathtt{i}}^{k+1} = \left(1 - \eta_k\right) \mathtt{e}_{\mathtt{i}}^{k} + \eta_k \tilde{\mathtt{e}}_{\mathtt{i}}^{k+1} \tag{5.27}$$

by construction, and the value of η_k minimizing c_k is

$$\eta_k = \frac{\sum_{\mathtt{i}} {\mathtt{e}_{\mathtt{i}}^{k}}^{\mathrm{T}} \left(\tilde{\mathtt{e}}_{\mathtt{i}}^{k+1} - \mathtt{e}_{\mathtt{i}}^{k}\right)}{\sum_{\mathtt{i}} \left\|\tilde{\mathtt{e}}_{\mathtt{i}}^{k+1} - \mathtt{e}_{\mathtt{i}}^{k}\right\|_2^2}. \tag{5.28}$$

The procedure repeats (5.24)-(5.28) until it declares termination either because it has reached a user-defined maximum number of iterations or because $\left\|\eta_k \left(\tilde{\boldsymbol{u}}^{k+1} - \boldsymbol{u}^k\right)\right\|_\infty$ falls below an absolute threshold (default value is 10^{-8}) and $\left(\frac{\sum_{\mathtt{i}} \left\|\mathtt{e}_{\mathtt{i}}^{k}\right\|_2^2}{\sum_{\mathtt{i}} \left\|\mathtt{r}_{\mathtt{i}}\right\|_2^2}\right)^{1/2}$ either falls below a relative threshold (default is 10^{-12}) or does not decrease by more than 10^{-6} relatively to its value in the previous iteration.

As for the scalar case, the discretization matrices $\mathtt{A}_{\mathtt{i}}$ are unchanged by the procedure, only the discretized right-hand sides are modified at each iteration: this allows for efficient reuse of (preconditioned) linear solvers as the iterative procedure progresses. Similarly to the scalar problem as well, no iteration is required when $[\mu] = 0$.

5.4.1 A note about using extensions only

As explained at the beginning of section 5.3, recovering consistency with (5.1) requires the evaluation of normal derivatives of the solution from one side of the interface. Since such normal derivatives are null by construction of $\boldsymbol{u}^{\Gamma,\pm}$, an extrapolation of $\boldsymbol{u}^{\pm}$ is used instead for their evaluation. This is a significant difference with the strategy for scalar-field problems [28].

However, this could be alleviated by exploiting the incompressibility condition on $\boldsymbol{u}^{\pm}$.

Indeed, by invoking (using Einstein Summation Convention)

$$0 = \partial_i u_i^{\pm} = \left(\left(\delta_{ij} - n_i n_j \right) + n_i n_j \right) \partial_j u_i^{\pm},$$

we have

$$n_i n_j \partial_j u_i^{\pm} = - \left(\delta_{ij} - n_i n_j \right) \partial_j u_i^{\pm},$$

where the right-hand side involves only tangential derivatives of velocity components and could be evaluated consistently as $-\nabla \cdot \boldsymbol{u}^{\Gamma,\pm}$. We note that this had been presented and used in [33], in two dimensions. It is not used in our implementation but could be considered as a possible optimization improvement to gain execution time.

5.5 A relevant test case

In order to test and validate the implementation of the techniques presented here before, we suggest to assess their performance by analyzing the accuracy obtained for a velocity field on a relevant test problem. Since $[\nabla \cdot \boldsymbol{u}] = 0$ is exploited in the derivation of the interface conditions on the binormal component of $[\mu \nabla \boldsymbol{u}]$ in (5.1), this prevents the use of an arbitrary manufactured solution to verify convergence of numerical results with grid refinement.

We therefore consider the analytical solution for the creeping flow of a viscous liquid sphere of radius R in another viscous liquid, due to a uniform background gradient of surface tension, i.e., $\gamma(\boldsymbol{x}) = \gamma_0 \left(1 + \beta \dfrac{\boldsymbol{x} \cdot \boldsymbol{e}_3}{R} \right)$. The theoretical analytisis of such a problem was first considered by Hadamard [176] and Rybczynski [177], under the assumption that the drop is and remains spherical. In [178], their derivations were extended to account for possible discontinuities in (tangential) stress across the interface (i.e., Marangoni forces). For the problem of interest, using spherical coordinates centered

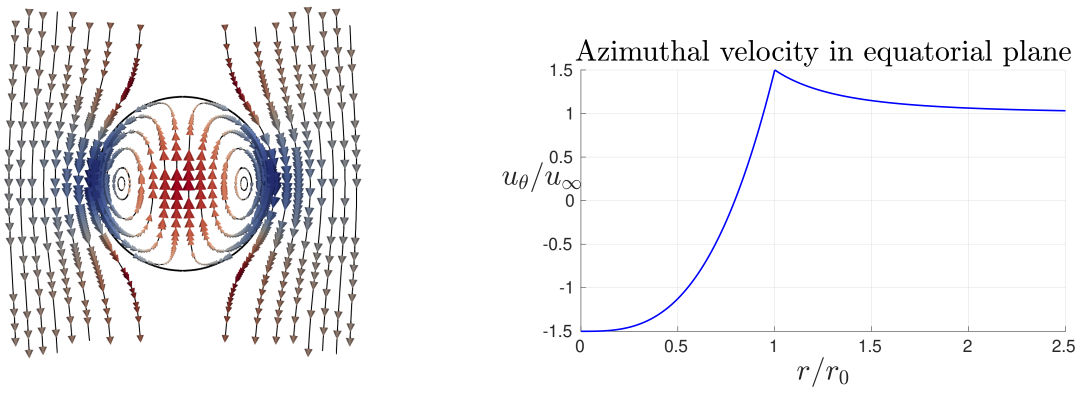


Figure 5.3: illustration of the flow field considered in section 5.5. Left: streamlines in a transverse plane parallel to $\boldsymbol{e}_3$ (zoom-in). Right: azimuthal velocity u_θ in the equatorial plane $\boldsymbol{x}\cdot\boldsymbol{e}_3 = 0$ (note the discontinuity in derivatives across the interface due to the existence of a background gradient in surface tension).

with the sphere, the analytical flow field is given by

$$\begin{cases} u_r^+(r,\theta) = u_\infty\left(1-\left(\frac{R}{r}\right)^3\right)\cos(\theta) \\ u_\theta^+(r,\theta) = -u_\infty\left(1+\frac{1}{2}\left(\frac{R}{r}\right)^3\right)\sin(\theta) \\ u_\phi^+ = 0 \\ p^+ = 0 \end{cases} \qquad \begin{cases} u_r^-(r,\theta) = \frac{3}{2}u_\infty\left(\left(\frac{r}{R}\right)^2-1\right)\cos(\theta) \\ u_\theta^-(r,\theta) = -\frac{3}{2}u_\infty\left(2\left(\frac{r}{R}\right)^2-1\right)\sin(\theta) \\ u_\phi^- = 0 \\ p^-(r,\theta) = 2\frac{\gamma_0}{R}+15\frac{\mu^- u_\infty r\cos(\theta)}{R^2} \end{cases}$$

where $u_\infty = 2\gamma_0\beta/\left(3\left(3\mu^-+2\mu^+\right)\right)$ and Ω^- is the interior of the sphere. This analytical solution is illustrated in Figure 5.3.

This reference solution has been used in benchmark testing for numerical simulations of two-phase flows in [9, 27]. Full flow dynamics was considered in such tests (simulating from rest until steady-state) but with $[\mu] = 0$. This choice simplifies dramatically the problem as it annihilates the effects of solution-dependent terms in the stress balance across the interface. Indeed, Table 5.1 shows that absolute convergence on the velocity components is obtained, without any iteration as described in section 5.4, i.e. after solving (5.23) only.

$\ell_{\min}/\ell_{\max}$		3/6	4/7	5/8	6/9	7/10	8/11
0 iteration	u_1, u_2	$2.8e-2$	$1.3e-2$	$6.7e-3$	$4.1e-3$	$1.8e-3$	$1e-3$
	u_3	$3.0e-2$	$2.6e-2$	$8.1e-3$	$7.1e-3$	$2.7e-3$	$1e-3$

Table 5.1: errors in infinity norm (normalized by u_∞) on the velocity components when solving (5.23), with a cubic computational domain $[-5,\,5]^3$ and a sphere of radius 0.5 centered in that domain, $\mu^- = \mu^+ = 0.1$, $\gamma_0 = 0.075$ and $\beta = -\dfrac{2}{15}$. When $\mu^- = \mu^+$, no solution-dependent term is involved in interface conditions which alleviates the need for iterations, as described in section 5.4.

We analyze the accuracy of the solution obtained when solving the corresponding Stokes flow problem (using Einstein Summation Convention)

$$\begin{cases} -\mu\nabla^2\boldsymbol{u} = -\nabla p \\ [\boldsymbol{u}] = \boldsymbol{0} \\ [\mu\partial_j u_i] = -\left(\delta_{ir} - n_i n_r\right)\partial_r\gamma\ n_j + [\mu]\left(\delta_{jk} - n_j n_k\right)\partial_k u_i^{\mp} \\ \qquad\qquad - [\mu]\ \ n_j\left(\delta_{ir} - n_i n_r\right)\partial_r u_k^{\mp}\ n_k + [\mu]\ \ n_i n_j n_r \partial_k u_r^{\mp} n_k \end{cases} \tag{5.29}$$

for $[\mu] \neq 0$ with the methodology previously described. We feed the solver with $-\nabla p$ (calculated from the exact solution) and $\nabla\gamma$; we consider a cubic computational domain $[-5,\,5]^3$ and a sphere of radius 0.5 centered in that domain. We consider[5] $\mu^- = 0.1$, $\mu^+ = 0.5$, $\gamma_0 = 0.075$ and $\beta = -\dfrac{2}{15}$. The exact velocity field is prescribed as a Dirichlet boundary condition on the boundaries of the computational domain. We consider adaptive octree grids with a layer of fine cells covering 15% of the bubble radius on either side of the interface.

The results summarizing the errors in infinity norm on the velocity components are presented in Table 5.2, Figure 5.4 and Figure 5.5, for various maximum number of iterations of the strategy presented in section 5.4. As illustrated, the errors on velocity

[5]For this particular test case (Stokes flow), numerical experiments indicate that the required grid refinement for observing convergence grows with μ^+/μ^-. Since we conduct the analysis with full three-dimensional considerations, we restrict ourselves to a moderate value of μ^+/μ^-. Axisymmetric computational tools could be considered to push the analysis further at a limited computational cost.

$\ell_{\min}/\ell_{\max}$		3/6	4/7	5/8	6/9	7/10	8/11
0 iteration	u_1, u_2	$1.5e-1$	$1.7e-1$	$1.7e-1$	$1.8e-1$	$1.8e-1$	$1.8e-1$
	u_3	1.5	1.5	1.5	1.6	1.6	1.6
5 iterations	u_1, u_2	$2.2e-1$	$1.6e-1$	$9.8e-2$	$5.1e-2$	$3.1e-2$	$2.3e-2$
	u_3	1.1	$7.2e-1$	$4.1e-1$	$2.3e-1$	$1.6e-1$	$1.3e-1$
10 iterations	u_1, u_2	$2.2e-1$	$1.6e-1$	$9.6e-2$	$5.5e-2$	$2.7e-2$	$1.4e-2$
	u_3	1.1	$7.0e-1$	$4.10e-1$	$2.3e-1$	$1.2e-1$	$6.4e-2$
unlimited	u_1, u_2	$2.2e-1$	$1.6e-1$	$9.6e-2$	$5.5e-2$	$2.7e-2$	$1.4e-2$
	u_3	1.1	$7.0e-1$	$4.1e-1$	$2.3e-1$	$1.2e-1$	$6.2e-2$

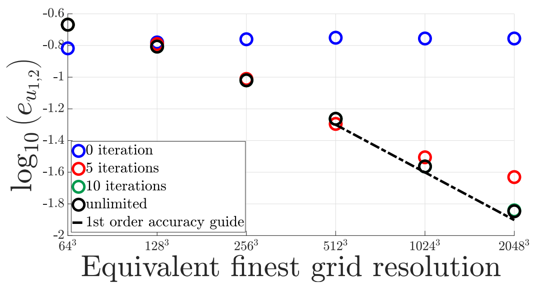
Table 5.2: errors in infinity norm (normalized by u_∞) on the velocity components when solving (5.29), with the considered parameters. $\ell_{\min}/\ell_{\max}$ denote the minimum and maximum levels of refinement (the equivalent finest grid resolution is $\left(2^{\ell_{\max}}\right)^3$). The leftmost column refers to the maximum number of iteration(s) set for the strategy described in section 5.4 ("0 iteration" corresponding to the solution of (5.23), i.e., discarding solution-dependent terms).

components do not decrease with grid refinement when the solution-dependent terms are simply discarded (0 iteration), whereas the suggested method successfully recovers convergence on velocity components, even when restricting the number of iterations to 10.

This observation represents a fundamental difference with the problem for scalar fields. Indeed, as shown in [16], disregarding solution-dependent terms when solving elliptic interface problems for *scalar* fields with a dimension-by-dimension approach does not affect the convergence of the solution itself (though its gradient is inaccurate [28]). However, when solving for separate components of vector fields, as we consider here, this no longer holds and the velocity components do not converge.

5.6 Summary

In this chapter, we have introduced and presented a numerical approach targeted for the implicit and sharp treatment of viscous terms in simulations of incompressible, viscous two-phase flows. A finite volume approach on Voronoi tessellations, first intro-



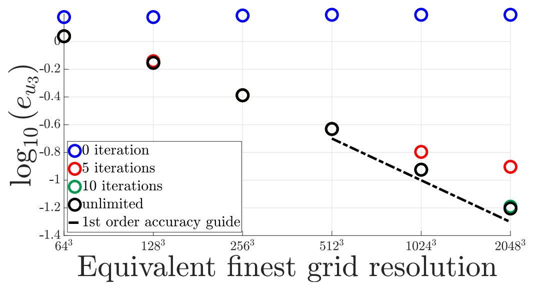


Figure 5.4: graph representation of the data from Table 5.2. e_{u_i} is the max error on u_i, normalized by u_∞. The black and green circles overlap.

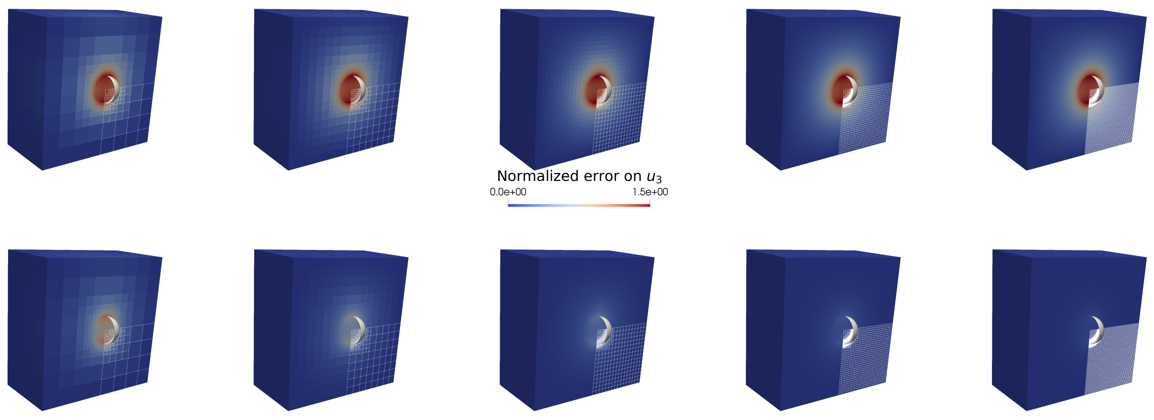


Figure 5.5: visualization of the error on u_3 when solving (5.29), with the considered parameters. Top: discarding solution-dependent terms in interface conditions (0 iteration). Bottom: enabling the iterative method described in section 5.4 to run until termination. From left to right, computational grids of increasing levels of refinement (a portion of a slice in the grid is illustrated). The same color bar is used for all images.

duced in [55], is used away from the interface. On the other hand, the relevant interface discontinuities are treated using a dimension-by-dimension approach (see [16, 15]). The velocity components are effectively decoupled from one another in this approach, except for the interface conditions which involve solution-dependent terms. In order to account for those terms (unknown a priori), extension and extrapolation techniques from [172] have been developed and adapted for face-sampled fields on adaptive quadtree/octree grids. Their use within an iterative strategy, extending the idea from [28] to problems involving vector fields, intends to recover such solution-dependent terms and include them in the relevant interface conditions. The relevance of these developments has been assessed and illustrated on a Stokes flow problem, highlighting the need to account for such terms when $[\mu] \neq 0$.

Chapter 6

Finite volume approach for elliptic interface problems with cell-sampled scalar fields

6.1 Introduction

In this chapter, we present the finite-volume approach implemented for solving elliptic interface problems of the kind

$$\begin{cases} -\nabla\cdot(\beta\nabla\psi) &= f, \\ [\psi] &= a, \\ [\beta\boldsymbol{n}\cdot\nabla\psi] &= b, \end{cases} \tag{6.1}$$

for a cell-sampled scalar field ψ and piecewise constant diffusion coefficient β, as relevant for the "pressure guess" and "projection step" problems outlined in chapter 3. As illustrated in Figure 3.1, we denote by Ω the computational domain, and by Γ the interface which separates Ω into Ω^+ and Ω^- (based on the sign of the levelset of zero-contour Γ).

The unit vector normal to Γ, pointing toward Ω^+ is $\boldsymbol{n}$.

While this might seem redundant with the content of chapter 4, the techniques presented therein were found inappropriate to use within a two-phase flow simulation engine, as outlined in chapter 3. First, that iterative strategy requires accurate (tangential) derivatives of a, which encompasses curvature-related terms as well as derivatives in velocity components in the targeted applications (see (3.19)). Therefore, the evaluation of $(\underline{\boldsymbol{\delta}} - \boldsymbol{nn}) \cdot \nabla a$ would not lead to accurate, reliable results in our framework. Second, the strategy from chapter 4 builds upon a finite-difference approach for capturing interface discontinuities. While it can be coupled to the stability-guaranteeing finite-volume approach from [55] away from the interface (see section 4.5), an entirely finite-volume approach may be preferable regarding stability.

We therefore opt for the method introduced in [154] which alleviates the drawbacks discussed here above. Originally implemented for node-sampled scalar fields and coupled to the superconvergent discretization from [179] away from the interface, the targeted application requires an implementation for cell-sampled scalar fields, coupled to the discretization presented in [55] away from the interface.

We also introduce a novel robust strategy for extrapolating the cell-sampled solution of (6.1) from either subdomain to the rest of the domain. This extrapolation relies on the PDE-based approach introduced in [172], which is adapted to cell-sampled fields on quadtree/octree grids.

6.2 Numerical discretization

The numerical discretization of (6.1) relies on the duplication of unknowns in cells that are crossed by the interface: in cell j which has its center in say Ω^+, one defines the primary unknown $\psi_j = \psi_j^+$ and an intermediary (ghost) unknown ψ_j^- which approximates the solution at the center of cell j as if it was in Ω^-. When the superscript is omitted, we implicitly refer to the primary unknown in a cell (that means $\psi_{\mathcal{C}} = \psi_{\mathcal{C}}^+$ if $\mathcal{C}$ has its center in Ω^+).

This duplication of unknowns allows one to safely assume that cells that are *not* crossed by the interface have access to values corresponding to the same subdomain for all their neighbors, even if any such neighbor has its center across the interface. However, it also leads to an underdetermined system of equations, which is closed by constraining the duplicates by means of truncated Taylor series, built consistently with the prescribed interface conditions.

6.2.1 Discretization away from the interface

Away from the interface, we use the discretization introduced in [55]. We summarize it here below and show that it is consistent with a finite-volume strategy, over the computational cells. For a clear and succinct presentation, we omit the $\pm$ superscripts and we consider (piecewise) constant diffusion coefficient β, as relevant for the targeted application.

Consider a computational cell $\mathcal{C}$ that is not crossed by the interface, by integrating $-\nabla \cdot (\beta \nabla \psi) = f$ side-by-side over $\mathcal{C}$ and using the divergence theorem (see appendix A.3), one has

$$\int_{\partial \mathcal{C}} -\beta \nabla \psi \cdot \boldsymbol{\eta} \, \mathrm{d}\partial \mathcal{C} = \int_{\mathcal{C}} f \, \mathrm{d}\mathcal{C} \tag{6.2}$$

wherein $\boldsymbol{\eta}$ is the unit vector normal to $\partial \mathcal{C}$ pointing outward $\mathcal{C}$. For rectangular grid cells, the left-hand side of (6.2) can be approximated by

$$\int_{\partial \mathcal{C}} -\beta \nabla \psi \cdot \boldsymbol{\eta} \, \mathrm{d}\partial \mathcal{C} \approx \sum_{i = x,\, y,\, z} -\beta \left(\partial_i \psi|_{\partial \mathcal{C}_{i+1/2}} - \partial_i \psi|_{\partial \mathcal{C}_{i-1/2}} \right) s_{\perp i}^{\mathcal{C}} \tag{6.3}$$

where $\partial\mathcal{C}_{i+1/2}$ (resp. $\partial\mathcal{C}_{i-1/2}$) is the face of $\mathcal{C}$ found in the positive (resp. negative) cartesian direction i and $s^{\mathcal{C}}_{\perp i}$ is the area of the cross section of $\mathcal{C}$, orthogonal to cartesian direction i. We denote by $h^{\mathcal{C}}_i$ the length of cell $\mathcal{C}$ along Cartesian direction i (see Figure 6.1).

Now the discretization used for $\partial_i\psi|_{\partial\mathcal{C}_{i\pm1/2}}$ requires special care as it may involve nonuniform cell neighbors (consider $\partial_x\psi|_{\partial D_{x+1/2}}$ or $\partial_y\psi|_{\partial A_{y+1/2}}$ in Figure 6.1 for instance). The discretization from [55] is used and may be generally described as follows. For $\partial_i\psi|_{\mathbb{F}}$ where $\mathbb{F}$ is any (non-wall) face orthogonal to the cartesian direction i

- find the biggest computational cell having $\mathbb{F}$ as (part of) one of its faces, let that cell be $\mathbb{Q}$;
- find all neighbor cells of $\mathbb{Q}$ across its face encompassing $\mathbb{F}$ and let that set of cells be $\mathcal{S}$;
- define the discretization distance

$$\Delta_{\mathbb{F}} = \sum_{\mathbb{C}\in\mathcal{S}} \frac{s^{\mathbb{C}}_{\perp i}}{s^{\mathbb{Q}}_{\perp i}} \frac{h^{\mathbb{C}}_i + h^{\mathbb{Q}}_i}{2}; \tag{6.4}$$

- discretize $\partial_i\psi|_{\mathbb{F}}$ as

$$\partial_i\psi|_{\mathbb{F}} \approx m \sum_{\mathbb{C}\in\mathcal{S}} \frac{s^{\mathbb{C}}_{\perp i}}{s^{\mathbb{Q}}_{\perp i}} \frac{\psi_{\mathbb{Q}} - \psi_{\mathbb{C}}}{\Delta_{\mathbb{F}}} \tag{6.5}$$

  where $m = +1$ (resp. $m = -1$) if $\mathbb{Q}$ is found in the positive (resp. negative) cartesian direction i, with respect to $\mathbb{F}$.

It is shown in [55] that this discretization is second-order accurate, at the center of the face of $\mathbb{Q}$ that encompasses $\mathbb{F}$. Note that, by construction, the discretization described here above will produce identical values of $\partial_i\psi|_{\mathbb{F}}$ for all faces subdividing another, larger face.

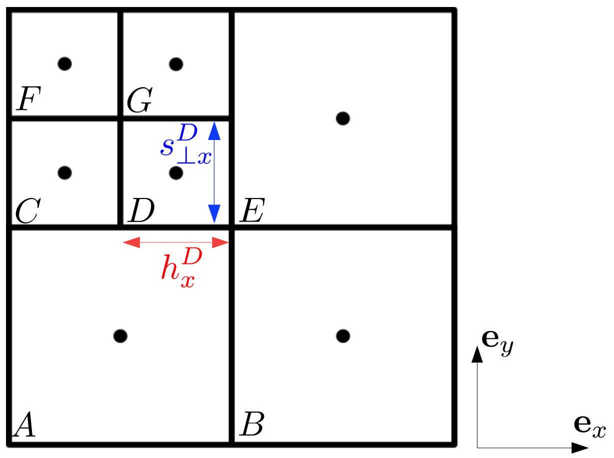


Figure 6.1: two-dimensional illustration of the notations used in subsection 6.2.1.

For example, $\partial_x \psi|_{\partial D_{x+1/2}}$ in Figure 6.1 would be approximated as

$$\partial_x \psi|_{\partial D_{x+1/2}} \approx \frac{(2\psi_E - \psi_D - \psi_G)/2}{(2h_x^E + h_x^D + h_x^G)/4},$$

and, by construction, the discretization produces $\partial_x \psi|_{\partial D_{x+1/2}} = \partial_x \psi|_{\partial G_{x+1/2}} = \partial_x \psi|_{\partial E_{x-1/2}}$.

If the face $\mathbb{F}$ is a wall face, i.e., if $\mathbb{F}$ lies on a (non-periodic) border of the computational domain, it corresponds to a boundary condition for ψ. Let $\mathcal{C}$ be the cell having $\mathbb{F}$ as a face. In case of Dirichlet boundary conditions, $\psi_{\mathbb{F}}$ is prescribed, and one may use

$$\partial_i \psi|_{\mathbb{F}} \approx m \frac{\psi^{\mathcal{C}} - \psi_{\mathbb{F}}}{h_i^{\mathcal{C}}/2}$$

where $m = +1$ (resp. $m = -1$) if $\mathcal{C}$ is found in the positive (resp. negative) cartesian direction i, with respect to $\mathbb{F}$. In case of Neumann boundary condition, the value of $\partial_i \psi|_{\mathbb{F}}$ is prescribed and thus used right away.

The right-hand side of (6.3) may be discretized as $f\mathrm{vol}(\mathcal{C})$ for arbitrary scalar field f provided as a source term. However, in the context of simulations of incompressible flows, the projection step (3.21) feeds $f = -\nabla \cdot \boldsymbol{u}^\star$ as a source term and the right-hand side of (6.3) then is discretized as

$$\int_{\mathcal{C}} -\nabla \cdot \boldsymbol{u}^\star \, \mathrm{d}\mathcal{C} \approx \sum_{i=x,\,y,\,z} -\left(\langle u_i^\star \rangle_{\partial\mathcal{C}_{i+1/2}} - \langle u_i^\star \rangle_{\partial\mathcal{C}_{i-1/2}} \right) s_{\perp i}^{\mathcal{C}} \tag{6.6}$$

wherein $\langle u_i^\star \rangle_{\mathbb{F}}$ is evaluated as follows for a face $\mathbb{F}$ orthogonal to the cartesian direction i:

- find the biggest computational cell having $\mathbb{F}$ as (part of) one of its faces, let that cell be $\mathbb{Q}$;
- find all neighbor cells of $\mathbb{Q}$ across its face encompassing $\mathbb{F}$ and let that set of cells be $\mathcal{S}$;
- let $s = +$ (resp. $s = -$) if $\mathbb{Q}$ is found in the positive (resp. negative) cartesian direction i, with respect to $\mathbb{F}$, then

$$\langle u_i^\star \rangle_{\mathbb{F}} = \sum_{\mathbb{C} \in \mathcal{S}} \frac{s_{\perp i}^{\mathbb{C}}}{s_{\perp i}^{\mathbb{Q}}} \, u_i^\star |_{\partial\mathbb{C},\, i\, s\, 1/2} \,. \tag{6.7}$$

Note that, by construction, this averaging procedure produces identical values for $\langle u_i^\star \rangle_{\mathbb{F}}$ on all faces subdividing another, larger face.

For example, $\int_{\mathcal{C}} -\nabla \cdot \boldsymbol{u}^\star \, \mathrm{d}\mathcal{C}$ in Figure 6.1 is discretized as

$$-s_{\perp x}^{D} \left(\frac{1}{2} \left(u^\star_{\partial D_{x+1/2}} + u^\star_{\partial G_{x+1/2}} \right) - u^\star_{\partial D_{x-1/2}} \right) - s_{\perp y}^{D} \left(v^\star_{\partial D_{y+1/2}} - \frac{1}{2} \left(v^\star_{\partial C_{y-1/2}} + v^\star_{\partial D_{y-1/2}} \right) \right),$$

in cell D.

As shown in [55], the discrete divergence operator $\mathtt{D}$ resulting from (6.6) and (6.7) is linked to the discrete gradient operator $\mathtt{G}$ defined by (6.5) via $\mathtt{L}_F \mathtt{G} = -(\mathtt{L}_C \mathtt{D})^{\mathrm{T}}$ wherein $\mathtt{L}_F$ and $\mathtt{L}_C$ are diagonal, positive-definite face- and cell-based operators, respectively. This latter relation stands as a discrete mimetic property of the corresponding continuum operators and guarantees that the resulting projection step does not inject spurious energy, as measured in the $\mathtt{L}_F$-norm.

6.2.2 Discretization for cells crossed by the interface

The discretization used for capturing the desired jump conditions from (6.1) was introduced in [154]; it is briefly summarized here. Consider a computational cell $\mathcal{C}$ that is crossed by the interface, let $\mathcal{C}^+ = \mathcal{C} \cap \Omega^+$ and $\mathcal{C}^- = \mathcal{C} \cap \Omega^-$. Integrating (6.1) over $\mathcal{C}^-$, one has

$$-\int_{\partial\mathcal{C}\cap\Omega^-} \beta^- \boldsymbol{\eta} \cdot \nabla\psi^- \,\mathrm{d}\partial\mathcal{C} - \int_{\Gamma\cap\mathcal{C}} \beta^- \boldsymbol{n} \cdot \nabla\psi^- \,\mathrm{d}\Gamma = \int_{\mathcal{C}^-} f^- \mathrm{d}\mathcal{C} \tag{6.8}$$

where $\boldsymbol{\eta}$ is the unit vector normal to $\partial\mathcal{C}$ pointing outward $\mathcal{C}$ and $\boldsymbol{n}$ is the unit vector normal to Γ pointing toward Ω^+ (see Figure 6.2). Similarly, the integration over $\mathcal{C}^+$ gives

$$-\int_{\partial\mathcal{C}\cap\Omega^+} \beta^+ \boldsymbol{\eta} \cdot \nabla\psi^+ \,\mathrm{d}\partial\mathcal{C} + \int_{\Gamma\cap\mathcal{C}} \beta^+ \boldsymbol{n} \cdot \nabla\psi^+ \,\mathrm{d}\Gamma = \int_{\mathcal{C}^+} f^+ \mathrm{d}\mathcal{C}. \tag{6.9}$$

Adding (6.8) and (6.9) side by side, the interface condition $[\beta\boldsymbol{n} \cdot \nabla\psi] = b$ may be used and one obtains

$$-\int_{\partial\mathcal{C}\cap\Omega^+} \beta^+ \boldsymbol{\eta} \cdot \nabla\psi^+ \,\mathrm{d}\partial\mathcal{C} - \int_{\partial\mathcal{C}\cap\Omega^-} \beta^- \boldsymbol{\eta} \cdot \nabla\psi^- \,\mathrm{d}\partial\mathcal{C} = \int_{\mathcal{C}^+} f^+ \mathrm{d}\mathcal{C} + \int_{\mathcal{C}^-} f^- \mathrm{d}\mathcal{C} - \int_{\Gamma\cap\mathcal{C}} b \,\mathrm{d}\Gamma. \tag{6.10}$$

The integrals in the left-hand side of (6.10) are discretized as

$$-\int_{\partial\mathcal{C}\cap\Omega^\pm} \beta^\pm \boldsymbol{\eta} \cdot \nabla\psi^\pm \,\mathrm{d}\partial\mathcal{C} \approx \sum_{i=x,y,z} -\beta^\pm \left(s_{\partial\mathcal{C}^\pm{}_{i+1/2}} \,\partial_i\psi^\pm\big|_{\partial\mathcal{C}_{i+1/2}} - s_{\partial\mathcal{C}^\pm{}_{i-1/2}} \,\partial_i\psi^\pm\big|_{\partial\mathcal{C}_{i-1/2}} \right) \tag{6.11}$$

where $s_{\partial\mathcal{C}^\pm{}_{i+1/2}}$ is the area of the portion of the face $\partial\mathcal{C}_{i+1/2}$ that belongs to $\Omega^\pm$ (note that it might be 0 if the face is not intersected). The notations are illustrated in Figure 6.2. We use the toolbox from [116, 180] for the surface reconstruction within cells, the

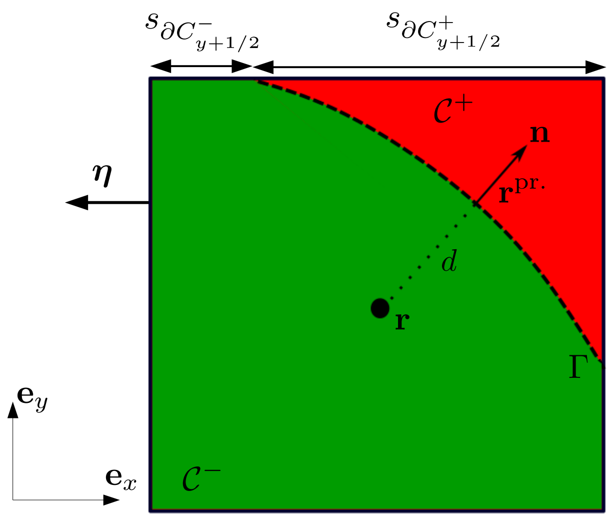


Figure 6.2: two-dimensional illustration of the notations used in subsection 6.2.2. Since $\boldsymbol{r}$ lies in Ω^-, the signed distance d is negative in this particular example.

evaluation of surface integrals and the calculation of surface areas.

In (6.11), the derivatives $\partial_i \psi^\pm|_{\mathbb{F}}$ are discretized using standard central finite difference between the values of $\psi^\pm$ associated with the two cells sharing the face $\mathbb{F}$ (the mesh is uniform in the neighborhood of the interface). Those derivatives are required only if they correspond to a nonzero value of $s_{\partial \mathcal{C}^\pm{}_{i\pm 1/2}}$, which ensures by itself that the required value of ψ^+ or ψ^- will be accessible in either cell.

Additional constraints on the duplicated unknowns in crossed cells must be provided in order to fully determine the discretized system of equations. Consider the center $\boldsymbol{r}$ of a cell $\mathcal{C}$ crossed by the interface, the normal vector $\boldsymbol{n} = \dfrac{\nabla \phi}{\|\nabla \phi\|}$ may be evaluated from the levelset function and the (signed) distance between $\boldsymbol{r}$ and Γ is $d = \dfrac{\phi(\boldsymbol{r})}{\|\nabla \phi\|}$. One may thus define the projection of $\boldsymbol{r}$ onto the interface as $\boldsymbol{r}^{\text{pr.}} = \boldsymbol{r} - d\boldsymbol{n}$ (see Figure 6.2). A Taylor expansion (truncated to linear terms) for $\psi^+ - \psi^-$ around $\boldsymbol{r}^{\text{pr.}}$, along the direction of $\boldsymbol{n}$ allows to write

$$\begin{aligned}
\psi^{+}(\boldsymbol{r}) - \psi^{-}(\boldsymbol{r}) &= [\psi]\left(\boldsymbol{r}^{\mathrm{pr.}}\right) + d\left[\partial_{\boldsymbol{n}}\psi\right] \\
&= a\left(\boldsymbol{r}^{\mathrm{pr.}}\right) + d\frac{b\left(\boldsymbol{r}^{\mathrm{pr.}}\right) - [\beta]\,\partial_{\boldsymbol{n}}\psi^{-}}{\beta^{+}} \qquad (6.12) \\
&= a\left(\boldsymbol{r}^{\mathrm{pr.}}\right) + d\frac{b\left(\boldsymbol{r}^{\mathrm{pr.}}\right) - [\beta]\,\partial_{\boldsymbol{n}}\psi^{+}}{\beta^{-}} \qquad (6.13)
\end{aligned}$$

where we have used $[AB] = A^{\pm}[B] + [A]\,B^{\mp}$ to exploit the (sharp) solution from one side of the interface only.

For the sake of clarity, let us consider that (6.12) is used to constraint the duplicated unknowns in $\mathcal{C}$. Let $\mathcal{N}_{\mathcal{C}}^{-}$ be the set of neighbor cells of $\mathcal{C}$ in a $3 \times 3 \times 3$ neighborhood that have their centers in Ω^{-}. If $\mathcal{N}_{\mathcal{C}}^{-}$ allows it[1], a least-square linear interpolant of ψ^{-} may be built around $\mathcal{C}$, minimizing the sampling errors on $\mathcal{C} \bigcup \mathcal{N}_{\mathcal{C}}^{-}$ and its coefficients are independent of the cell-sampled values of ψ^{-} (see appendix B). By approximating $\partial_{\boldsymbol{n}}\psi^{-}$ in (6.12) by the derivative of such a least-square interpolant, one finds the appropriate coefficients c's such that

$$\partial_{\boldsymbol{n}}\psi^{-} \approx c_{\mathcal{C}}\psi_{\mathcal{C}}^{-} + \sum_{\mathcal{D}\in\mathcal{N}_{\mathcal{C}}^{-}} c_{\mathcal{D}}\psi_{\mathcal{D}} \qquad (6.14)$$

(note that $\psi_{\mathcal{D}} = \psi_{\mathcal{D}}^{-}$ since $\mathcal{D} \in \mathcal{N}_{\mathcal{C}}^{-}$). Substituting (6.14) into (6.12), one obtains

$$\begin{cases}
\psi_{\mathcal{C}}^{+} &= \psi_{\mathcal{C}} + a + \dfrac{d}{\beta^{+}}b - \dfrac{d\,[\beta]}{\beta^{+}}\left(c_{\mathcal{C}}\psi_{\mathcal{C}} + \sum_{\mathcal{D}\in\mathcal{N}_{\mathcal{C}}^{-}} c_{\mathcal{D}}\psi_{\mathcal{D}}\right) \\
\psi_{\mathcal{C}}^{-} &= \psi_{\mathcal{C}}
\end{cases} \qquad (6.15)$$

[1]If not, that could be because $\mathcal{N}_{\mathcal{C}}^{-}$ does not contain enough elements or because these elements are geometrically degenerate (e.g. they all lie in a same plane, for instance).

if the center of cell $\mathcal{C}$ lies in Ω^-. Otherwise, if the center of cell $\mathcal{C}$ lies in Ω^+, one has

$$\begin{cases} \psi_{\mathcal{C}}^{+} &= \psi_{\mathcal{C}} \\ \psi_{\mathcal{C}}^{-} &= \psi_{\mathcal{C}} - a - \dfrac{d}{\beta^+} b + \dfrac{d\,[\beta]}{\beta^+} \dfrac{\left(c_{\mathcal{C}} \left(\psi_{\mathcal{C}} - a - \dfrac{d}{\beta^+} b \right) + \sum_{\mathcal{D} \in \mathcal{N}_{\mathcal{C}}^{-}} c_{\mathcal{D}} \psi_{\mathcal{D}} \right)}{\left(1 - c_{\mathcal{C}} d \dfrac{[\beta]}{\beta^+} \right)}. \end{cases} \tag{6.16}$$

Relations similar to (6.15) and (6.16) can be obtained in a totally similar way if (6.13) is used instead of (6.12). Note that (6.15) and (6.16) express the intermediary (ghost) unknowns in crossed cells entirely as linear combinations of surrounding *primary* unknowns. Therefore, the problem is entirely determined for the primary unknowns after substituting (6.15) and (6.16) into the discretized form of (6.10) and (6.11).

The volume integrals in the right-hand side of (6.10) may be discretized as $f^+ \mathrm{vol}\left(\mathcal{C}^+\right) + f^- \mathrm{vol}\left(\mathcal{C}^-\right)$ for an arbitrary scalar field f provided as a source term. However, in the context of simulations of incompressible flows, the projection step (3.21) feeds $f = -\nabla \cdot \boldsymbol{u}^\star$ as a source term and, in such a case, one has

$$-\int_{\mathcal{C}^+} \nabla \cdot \boldsymbol{u}^{\star,+} \mathrm{d}\mathcal{C} - \int_{\mathcal{C}^-} \nabla \cdot \boldsymbol{u}^{\star,-} \mathrm{d}\mathcal{C} = -\int_{\partial\mathcal{C} \bigcap \Omega^+} \boldsymbol{u}^{\star,+} \cdot \boldsymbol{\eta} \, \mathrm{d}\partial\mathcal{C} - \int_{\partial\mathcal{C} \bigcap \Omega^-} \boldsymbol{u}^{\star,-} \cdot \boldsymbol{\eta} \, \mathrm{d}\partial\mathcal{C} + \int_{\Gamma \bigcap \mathcal{C}} \left[\boldsymbol{u}^\star \cdot \boldsymbol{n} \right] \mathrm{d}\Gamma \tag{6.17}$$

which naturally handles a possible discontinuity in normal velocity components (as it is the case in presence of phase change). In (6.17), the discretization

$$\int_{\partial\mathcal{C} \bigcap \Omega^{\pm}} \boldsymbol{u}^{\star,\pm} \cdot \boldsymbol{\eta} \, \mathrm{d}\partial\mathcal{C} \approx \sum_{i = x,\, y,\, z} \left(s_{\partial\mathcal{C}_{i+1/2}^{\pm}} \, u_i^{\star,\pm} \big|_{i+1/2} - s_{\partial\mathcal{C}_{i-1/2}^{\pm}} \, u_i^{\star,\pm} \big|_{i-1/2} \right) \tag{6.18}$$

is used (note that we need velocity components extrapolated from either side, at faces of $\mathcal{C}$ intersected by the interface).

The system of linear equations resulting from this discretization is non-symmetric and the discretized equations for crossed cells involve primary unknowns associated with second-degree neighbors. We favor (6.12) (resp. (6.13)) when $\beta^+ > \beta^-$ (resp. when $\beta^- > \beta^+$) as that choice was shown to keep the condition number of the overall discretization matrix bounded in numerical experiments (see [154]).

6.3 Resolution of the linear system

The resolution of the linear system resulting from the above discretizations is tackled with a BiCGStab algorithm provided by PetSc [111]. By default, the solver uses Hypre [181] as a *right* preconditioner. This allows BiCGStab to monitor the true residual to determine convergence and termination.

Indeed, the simulation of three-dimensional buoyant bubbles in viscous fluids revealed technical challenges with the resolution of this system of equations. When used as a left preconditioner (hence monitoring convergence of the preconditioned residual within BiCGStab), Hypre was found to produce spurious effects leading to a very large residual in a few cells close to the interface in some cases. This in turn could either impede the successful termination of BiCGStab (and crash simulations) or produce highly inaccurate results on termination because the preconditioned residual was found below the relative threshold as compared to its (very large) initial value, whereas the true residual had not decreased much.

The use of HYPRE as a *right* preconditioner does not alleviate the production of such numerical artifacts intrinsically, but it makes results more reliable upon successful termination of BiCGStab, since the true residual is monitored in such a case. If BiCGStab fails to converge, the following fall-back strategy is used for successive further attempts:

1. the “strong threshold” parameter of the Hypre preconditioner is increased to 0.9. A larger value translates into an algebraic condition that is more likely to discard

loosely connected degrees of freedom during the coarsening steps of the multigrid method. This means that coarsening steps are less likely to involve degrees of freedom that are loosely connected across the interface [182];

2. BiCGStab is used without preconditioner if the latter fails to converge;
3. GMRES is used without preconditioner if the latter fails;
4. an enhanced BiCGStab(L) algorithm is used without preconditioner if the latter fails (reported most stable in [51]).

6.4 PDE-based extrapolation of cell-sampled fields

In order to ensure sharpness in our numerical approach, primary (sharp) fields of interest need to be extrapolated from either subdomain. This is required to prevent differentiation across interface discontinuities, for instance. Similarly, the points traced back in time in the context of the semi-Lagrangian treatment of advection terms may land across the interface and one must use extrapolated values to ensure a sharp discretization in such a case.

Once the system of discretized equations is solved, its sharp solution is extrapolated from either subdomain to the rest of the domain. We present here an adaptation of the PDE-based, linear extrapolation method from [172] for cell-sampled scalar fields on adaptive Quadtree/Octree grids. For the sake of clarity, we consider the extrapolation of ψ^- from Ω^- hereafter (the extrapolation of ψ^+ from Ω^+ follows the same principles with a reverted normal vector).

The methodology starts by defining the normal derivatives $\psi^-_{\boldsymbol{n}} = \boldsymbol{n} \cdot \nabla \psi^-$ at cells of Ω^- where it can be defined, i.e., where all the required well-defined neighbor values are accessible and valid (see subsection 6.4.1). In such a cell $\mathcal{C}$, we discretize $\psi^-_{\boldsymbol{n}}$ as

$$\sum_{i=x,y,z} n_i \frac{\left(\partial_i \psi^-|_{\partial\mathcal{C}_{i+1/2}} + \partial_i \psi^-|_{\partial\mathcal{C}_{i-1/2}}\right)}{2} \tag{6.19}$$

using the notations from subsection 6.2.1 (this is equivalent to the standard central finite difference formula in locally uniform regions).

Let $H^-_{\boldsymbol{n}} = 0$ in cells where $\psi^-_{\boldsymbol{n}}$ could be defined, and $H^-_{\boldsymbol{n}} = 1$ where it could not. These normal derivatives are then extrapolated using constant extrapolation along $\boldsymbol{n}$, by solving

$$\partial_\tau \psi^-_{\boldsymbol{n}} + H^-_{\boldsymbol{n}} \boldsymbol{n} \cdot \nabla \psi^-_{\boldsymbol{n}} = 0 \tag{6.20}$$

until steady-state in pseudo-time τ.

The procedure then extrapolates the values of ψ^-. In cells having their centers in Ω^-, one has $\psi^- = \psi$ and ψ^- may also be defined in cells crossed by the interface, by exploiting relations (6.15) and (6.16) (this is further discussed in subsection 6.4.1). Let $H^- = 0$ in cells where ψ^- could be defined, and $H^- = 1$ where it could not. The value of ψ^- is then extrapolated along $\boldsymbol{n}$ by solving

$$\partial_\tau \psi^- + H^- \left(\boldsymbol{n} \cdot \nabla \psi^- - \psi^-_{\boldsymbol{n}}\right) = 0 \tag{6.21}$$

until steady-state in pseudo-time τ.

In practice, (6.20) and (6.21) are solved for a finite number of pseudo-time steps using a forward Euler time discretization (we use 30 time steps to ensure convergence within a band of 3 smallest computational cells across the interface) and upwind discretization for the spatial derivatives. For instance, $\boldsymbol{n} \cdot \nabla \psi^-$ is discretized as

$$\sum_{i=x,\,y,\,z} \max\left(n_i, 0\right) \partial_i \psi^- \big|_{\partial \mathcal{C}_{i-1/2}} + \min\left(n_i, 0\right) \partial_i \psi^- \big|_{\partial \mathcal{C}_{i+1/2}} \tag{6.22}$$

in (6.21). The pseudo-time step $\Delta\tau$ is set to 1/3 in 3D (resp. 1/2 in 2D) of the minimal discretization distance (6.4) associated with the relevant derivatives in (6.22). Note that adaptive pseudo time stepping is used (the pseudo-time step is larger in large computational cells).

6.4.1 About the use of ghost values for extrapolation purposes

As explained in subsection 6.2.2, cells that are crossed by the interface are augmented with secondary (ghost) unknowns, corresponding to the subdomain across the interface. For instance, ψ^+ is defined via relation (6.15) in cell from figure 6.2 although its center lies in Ω^-. Including such ghost values in the initialization of the extrapolation methods from section 6.4 (within calculations of (6.19), for instance) naturally expands the regions of well-defined neighbors, which is desirable in case of locally under-resolved interfaces.

However, those ghost values were found to be inaccurate in two particular cases:

1. as stated in subsection 6.2.2, (6.12) is favored over (6.13) when $\beta^+ > \beta^-$ (and vice versa). Let us assume $\beta^+ > \beta^-$ without loss of generality. For a given cell crossed by the interface, one may not find enough cells in a $3 \times 3 \times 3$ neighborhood with their centers in Ω^- and therefore the local construction of the required least-square operator for $\partial_{\boldsymbol{n}} \psi^-$ (see (6.14)) cannot be done. In such a case, the discretization resorts to using (6.13) locally instead, since the least-square operator for $\partial_{\boldsymbol{n}} \psi^+$ can be constructed with surrounding primary unknowns, in such a case. However, this produces a much more ill-conditioned (numerically) definition for the ghost values,

in particular in case of large ratios of diffusion coefficients, which may in turn lead to inaccurate ghost values for the considered cell. This is illustrated in Figure 6.3 for one of the considered test cases in two dimensions.

2. the ghost value in crossed cell was found to be inaccurate when the volume across the interface within the cell is very small. The smaller the volume, the less the computational weight associated with such a ghost value within the discretization scheme. Unless one uses a much more stringent (and less practical) termination criterion on the (relative) residual, problems of this sort may occur for cells having tiny portions of their volume across the interface.

In order to avoid ill-defined ghost values, the extrapolation procedure for cell-sampled scalar fields addresses the above issues by considering ghost values in crossed cells valid only if (6.12) (resp. (6.13)) could be used locally if $\beta^+ > \beta^-$ (resp. if $\beta^- > \beta^+$) and if the volume across the interface within the cell is at least 1% of the total volume of the cell[2].

6.5 Validation and illustrations

In order to validate the implementation of the solver as well as the extrapolation procedure, we analyze the accuracy of the solution it produces on manufactured test problems.

[2]This threshold value is arbitrary and was found to be large enough to stabilized projection steps in simulations. A less arbitrary value would depend on the tolerance of the relative residual, the ratio of diffusion coefficients and a very thorough analysis of the discretization scheme which goes beyond the scope of this work.

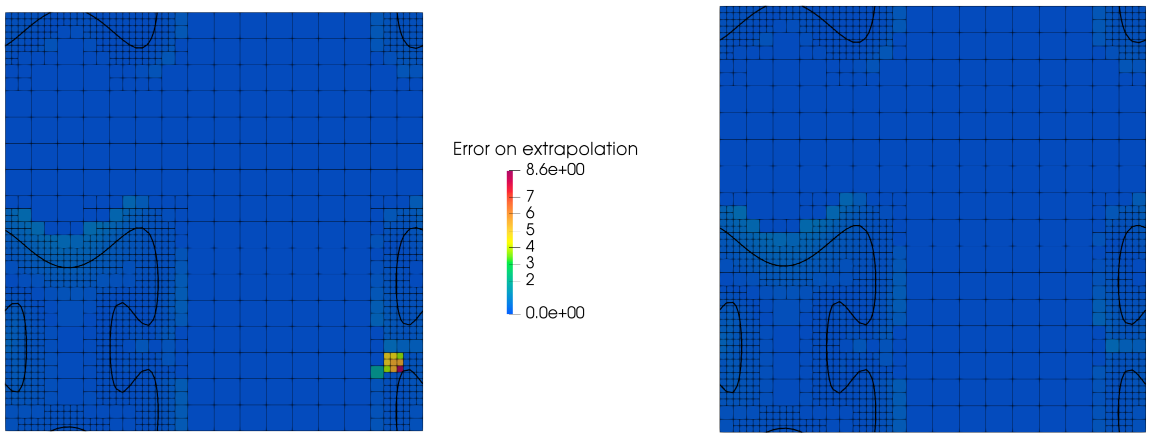


Figure 6.3: illustration of inaccurate ghost values obtained due to a lack of enough cell neighbors on the slow-diffusion side of the interface for a given crossed cell. For this fully periodic example, we have $\beta^- = 1$ and $\beta^+ = 100$, Ω^- is the subdomain interior to the interface. Left: error on the extrapolation of the solution from Ω^- when using all ghost values. A large error is found in one cell and propagates to its domain of influence; the symptomatic cell has only one cell neighbor in Ω^- within its 3×3 neighborhood and (6.13) is used locally instead of (6.12). Right: error on the extrapolation of the solution from Ω^- (using the same color scale) when using only well-defined ghost values, i.e., discarding the ill-conditioned ghost value(s) built with (6.13).

6.5.1 In two dimensions

Benchmark test 1

We consider $\Omega = [-1,\, 1] \times [0,\, 3]$. The interface is defined as the parameterized curve $\Gamma \equiv (x(\vartheta)\,,\, y(\vartheta))\,,\ \vartheta \in [0,\, 2\pi[$ where

$$\begin{cases} x(\vartheta) = 0.6\cos(\vartheta) - 0.3\cos(3\vartheta)\,, \\ y(\vartheta) = 1.5 + 0.7\sin(\vartheta) - 0.07\sin(3\vartheta) + 0.2\sin(7\vartheta)\,. \end{cases}$$

The corresponding levelset function is defined as the exact signed distance to Γ, negative in the interior region. The coefficients are $\beta^- = 1$ and $\beta^+ = 10$. The exact solution is chosen to be

$$\begin{cases} u(x,\, y) = \exp(x)\left(x^2\sin(y) + y^2\right), & \forall (x,\, y) \in \Omega^-, \\ u(x,\, y) = -\left(x^2 + y^2\right), & \forall (x,\, y) \in \Omega^+, \end{cases}$$

the right-hand side f and the jump terms a and b are defined accordingly. Dirichlet boundary conditions are used on the walls of the computational domain. This is the ninth example from [16].

We analyze the performance of the solver and the extrapolation on this test problem. The accuracy of the solution is analyzed on grids of increasing resolution: the finest computational cells are set to be 8 times finer than the coarsest and the performance of the solver is assessed on grids of increasing resolution. The solution is also extrapolated from either subdomain and the accuracy of that extrapolation is analyzed in a band of 3 finest computational cells across the interface. The accuracy of the partial derivatives, evaluated at faces of the computational grid is also analyzed.

As illustrated in Figure 6.4, the solver achieves close to second order accuracy on the

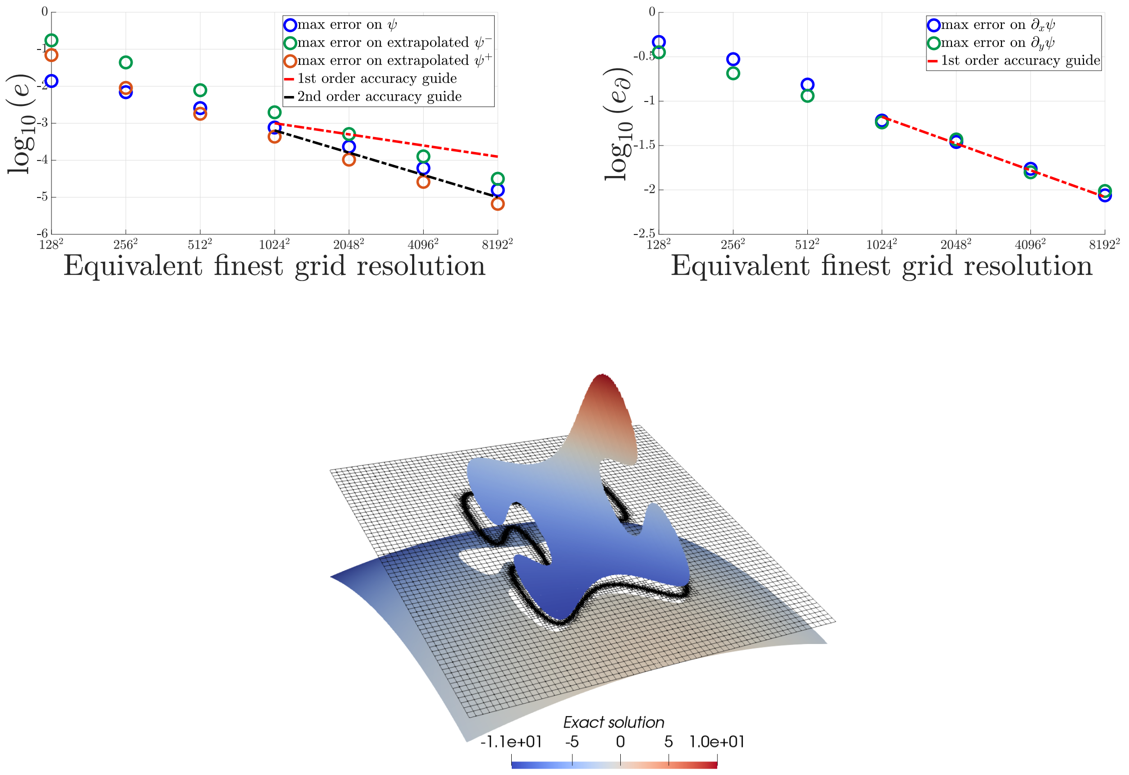


Figure 6.4: convergence analysis for the benchmark test 1 in subsection 6.5.1. Top left: maximum error on the solution for increasing grid resolutions. Top right: maximum errors on the partial derivatives of the solution (evaluated at faces) for increasing grid resolutions. Bottom: illustration of the solution and the computational grid for an equivalent finest resolution of 512^2.

solution and the extrapolations are also second order accurate. The derivatives of the solution are first order accurate (note that the evaluation of derivatives at faces close to the interface may involve extrapolated values).

Benchmark test 2: large ratio of diffusion coefficients

We consider $\Omega = [-1,\, 1] \times [-1,\, 1]$. The interface is defined as the parameterized curve $\Gamma \equiv (x\,(\vartheta)\,,\, y\,(\vartheta))\,,\ \vartheta \in [0,\, 2\pi[$ where

$$\begin{cases} x\,(\vartheta) = (0.5 + 0.1 \sin(5\vartheta)) \cos(\vartheta)\,, \\ y\,(\vartheta) = (0.5 + 0.1 \sin(5\vartheta)) \sin(\vartheta)\,. \end{cases}$$

The corresponding levelset function is defined as the exact signed distance to Γ, negative in the interior region. The coefficients are $\beta^- = 10^4$ and $\beta^+ = 1$. The exact solution is chosen to be

$$\begin{cases} u\,(x,\, y) = \dfrac{\exp(x)\,(x^2 \sin(y) + y^2)}{\beta^-}, & \forall\,(x,\, y) \in \Omega^-, \\ u\,(x,\, y) = 0.5 + \cos(x)\,(y^4 + \sin(y^2 - x^2))\,, & \forall\,(x,\, y) \in \Omega^+, \end{cases}$$

the right-hand side f and the jump terms a and b are defined accordingly. Dirichlet boundary conditions are used on the walls of the computational domain. This is the sixth example from [28], and it is designed to test the ability of the solver to deal with large ratios of diffusion coefficients, though flux magnitudes are comparable across the interface.

The accuracy of the solution is analyzed on grids of increasing resolution: the finest computational cells are set to be 8 times finer than the coarsest and the performance of the solver is assessed on grids of increasing resolution. The solution is also extrapolated from either subdomain and the accuracy of that extrapolation is analyzed in a band of 3 finest computational cells across the interface. The accuracy of the *components of the flux vector* $\beta\nabla\psi$, evaluated at faces of the computational grid is also analyzed.

For this challenging problem, the default setup of BiCGStab with Hypre as a right-preconditioner failed for the default value of the strong threshold (0.25 for two-dimensional

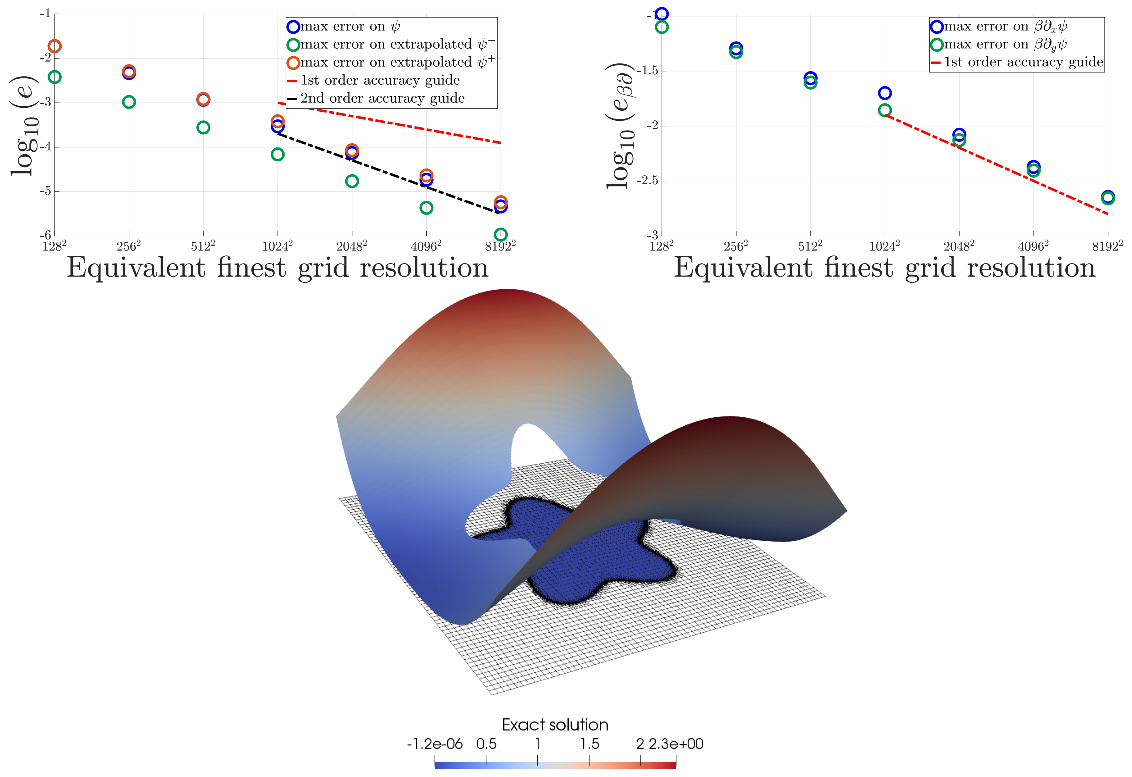


Figure 6.5: convergence analysis for the benchmark test 2 in subsection 6.5.1. Top left: maximum error on the solution for increasing grid resolutions. Top right: maximum errors on the flux components (evaluated at faces) for increasing grid resolution. Bottom: illustration of the solution and the computational grid of equivalent finest resolution 512^2.

problems), on the finest grid considered. Therefore, the solver resorted to using a larger value of the strong threshold of 0.9, which successfully enabled BiCGStab to solve the system (see section 6.3). As illustrated in Figure 6.5, the solver achieves second order accuracy on the solution and the extrapolations are also second order accurate. The components of the flux vector are first order accurate.

6.5.2 In three dimensions

We consider the three-dimensional domain $\Omega = [-2,\, 2] \times [-2,\, 2] \times [-2,\, 2]$. The interface is the surface parameterized by

$$r\left(\vartheta, \varphi\right) = \frac{3}{4} + \frac{5 - 3\cos\left(6\varphi\right)\left(1 - \cos\left(6\vartheta\right)\right)}{25}, \quad \vartheta \in \left[0,\, \pi\right], \varphi \in \left[0,\, 2\pi\right[$$

in standard spherical coordinates; the corresponding levelset function is built as negative inside and positive outside. The coefficients are $\beta^- = 1$ and $\beta^+ = 1250$. The exact solution is chosen to be

$$\begin{cases} u\left(x,\, y\right) = \exp\left(\dfrac{x-z}{2}\right)\left(x\sin\left(y\right) - \cos\left(x+y\right)\arctan\left(z\right)\right), & \forall\left(x,\, y\right) \in \Omega^-, \\ u\left(x,\, y\right) = -1 + \dfrac{5\arctan\left(\dfrac{x^3 y}{10} + 2z\cos\left(y\right) - y\sin\left(x+z\right)\right)}{2\beta^+}, & \forall\left(x,\, y\right) \in \Omega^+, \end{cases}$$

the right-hand side f and the jump terms a and b are defined accordingly. Dirichlet boundary conditions are used on the walls of the computational domain. This is the example from section 5.2 in [28], and it is designed to test the ability of the solver to deal with large ratios of diffusion coefficients, though flux magnitudes are comparable across the interface.

The accuracy of the solution is analyzed on grids of increasing resolution: the finest computational cells are set to be 8 times finer than the coarsest and the performance of the solver is assessed on grids of increasing resolution. The solution is also extrapolated from either subdomain and the accuracy of that extrapolation is analyzed in a band of 3 finest computational cells across the interface. The accuracy of the components of the flux vector $\beta\nabla\psi$, evaluated at faces of the computational grid is also analyzed. As illustrated in Figure 6.6, the solver achieves second order accuracy on the solution and the extrapolations are also second order accurate on sufficiently resolved grids. The

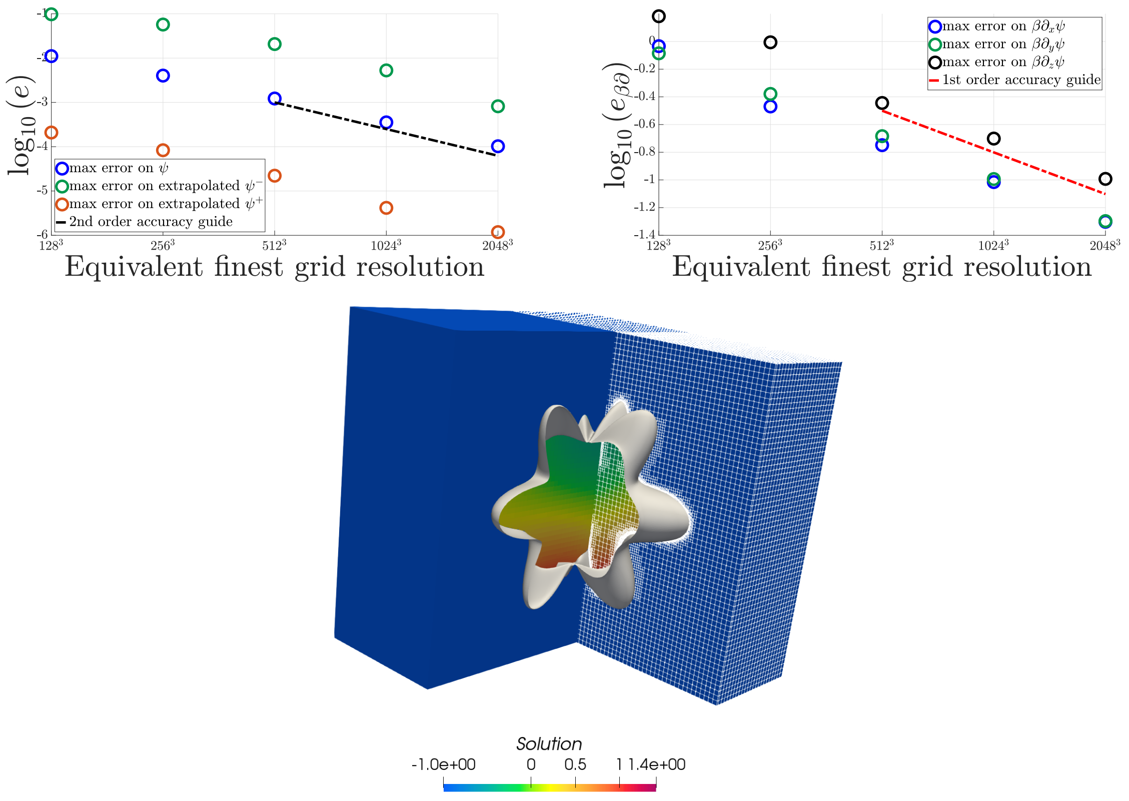


Figure 6.6: convergence analysis for the three-dimensional benchmark test subsection 6.5.2. Top left: maximum error on the solution for increasing grid resolutions. Top right: maximum errors on the flux components (evaluated at faces) for increasing grid resolution. Bottom: illustration of the solution, the (truncated) interface and the computational grid of equivalent finest resolution 512^3.

components of the flux vector are first order accurate. This analysis was conducted on 12 KNL nodes, using 768 MPI tasks, on Stampede2.

6.6 Summary

In this chapter we have introduced and presented a finite-volume strategy for the numerical resolution of scalar, elliptic interface problems. The discretization couples the method introduced in [154] for capturing interface discontinuity conditions to the method from [55] away from the interface, as it guarantees numerical stability for the projection step in simulation of single-phase flows. We also introduced an adaptation of the PDE-based extrapolation scheme from [172] for cell-sampled scalar fields on adaptive quadtree/octree grids. The performance of these methods and the quality of their implementations was assessed by the the methods of manufactured solutions which showed (close to) second-order accurate results for the solution and its extrapolations as well as first-order accurate results for its gradient (and/or flux vector).

Chapter 7

Simulation of incompressible, viscous two-phase flows

7.1 Introduction

In this chapter, we consider the numerical simulation of incompressible, viscous two-phase flows with a sharp levelset representation and implicit treatment of viscous terms. As outlined in chapter 3 (see subsection 3.3.1), the developed solver builds upon the novel numerical methods described in chapters 5 and 6. However, it also involves several other, peripheral or well-established numerical methods and features that have not been presented yet. After reviewing these omitted parts, we assess the performance of the solver (accuracy in infinity norm) on validation problems and benchmark tests. The simulation of actual two-phase flows is then considered and illustrated.

7.2 Complementary numerical methods

In this section, we provide complementary descriptions and explanations pertaining to additional peripheral or well-established numerical methods used within the solver.

7.2.1 Discretization of fluid advection terms

The implementation of simulation tools on adaptive grids comes with added complexity, but it is usually motivated by the perspective of using extreme levels of resolutions, with a limited computational cost. While fine spatial resolutions may be desirable, simulations harnessing them would still be intractable if they are associated with a stringent time-step restriction like $\Delta t \sim \mathcal{O}\left(\Delta x^2\right)$.

This motivates the treatment of viscous terms in a fully implicit fashion, which should alleviate such a constraint when associated with an unconditionally stable time-integration scheme. Borrowing from the approach used in single-phase flows, the material derivative $\dfrac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t}$ is discretized with a semi-Lagrangian approach, combined with a second-order backward differentiation formula [109, 87], in order to guarantee absolute stability.

Consider the fluid velocity $\boldsymbol{u}\left(\boldsymbol{x}, t\right)$. For any point $\boldsymbol{x}_0$ and time t_0, one may define the material particle trajectory $\boldsymbol{X}_{\left(\boldsymbol{x}_0, t_0\right)}\left(t\right)$ such that

$$\begin{cases} \boldsymbol{X}_{\left(\boldsymbol{x}_0, t_0\right)}\left(t_0\right) = \boldsymbol{x}_0, \\ \dfrac{\mathrm{d}\boldsymbol{X}_{\left(\boldsymbol{x}_0, t_0\right)}}{\mathrm{d}t} = \boldsymbol{u}\left(\boldsymbol{X}_{\left(\boldsymbol{x}_0, t_0\right)}\left(t\right), t\right). \end{cases}$$

Let $\boldsymbol{U}\left(t\right) = \boldsymbol{u}\left(\boldsymbol{X}_{\left(\boldsymbol{x}_0, t_0\right)}\left(t\right), t\right)$, then

$$\frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}t} = \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right)_{\left(\boldsymbol{X}_{\left(\boldsymbol{x}_0, t_0\right)}\left(t\right), t\right)}. \tag{7.1}$$

The semi-lagrangian approach builds upon (7.1) and uses a discretization for $\frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}t}$ as material advection terms. Using a second-order backward differentiation formula, we have

$$\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)_{(\boldsymbol{x},t_{n+1})} \simeq \alpha\frac{\boldsymbol{u}\left(\boldsymbol{x},t_{n+1}\right) - \boldsymbol{u}\left(\boldsymbol{X}_{(\boldsymbol{x},t_{n+1})}\left(t_n\right),t_n\right)}{\Delta t_n} + \beta\frac{\boldsymbol{u}\left(\boldsymbol{X}_{(\boldsymbol{x},t_{n+1})}\left(t_n\right),t_n\right) - \boldsymbol{u}\left(\boldsymbol{X}_{(\boldsymbol{x},t_{n+1})}\left(t_{n-1}\right),t_{n-1}\right)}{\Delta t_{n-1}} \tag{7.2}$$

where $\alpha = \frac{2\Delta t_n + \Delta t_{n-1}}{\Delta t_n + \Delta t_{n-1}}$ and $\beta = \frac{-\Delta t_n}{\Delta t_n + \Delta t_{n-1}}$.

Numerically, the backtraced points $\boldsymbol{X}_{(\boldsymbol{x},t_{n+1})}\left(t_{n+1}-T\right)$ (where T is either Δt_n or $\Delta t_n + \Delta t_{n-1}$) are evaluated with the following two-stage procedure. First, one approximates $\boldsymbol{X}_{(\boldsymbol{x},t_{n+1})}\left(t_{n+1}-T/2\right)$ by

$$\bar{\boldsymbol{X}} = \boldsymbol{x} - \frac{T}{2}\boldsymbol{u}\left(\boldsymbol{x},t_n\right) \tag{7.3}$$

then

$$\boldsymbol{X}_{(\boldsymbol{x},t_{n+1})}\left(t_{n+1}-T\right) \approx \boldsymbol{x} - T\boldsymbol{v}\left(\bar{\boldsymbol{X}},t_{n+1}-\frac{T}{2}\right) \tag{7.4}$$

where $\boldsymbol{v}\left(\bar{\boldsymbol{X}},t\right)$ is interpolating linearly in time between the velocity fields at times t_{n-1} and t_n, i.e.,

$$\boldsymbol{v}\left(\boldsymbol{X},t\right) = \boldsymbol{u}\left(\boldsymbol{X},t_n\right) + \frac{\boldsymbol{u}\left(\boldsymbol{X},t_n\right) - \boldsymbol{u}\left(\boldsymbol{X},t_{n-1}\right)}{\Delta t_{n-1}}\left(t-t_n\right). \tag{7.5}$$

The spatial interpolation of the velocity components at times t_n and t_{n-1} in (7.2) and (7.5) is done with quadratic interpolation of the appropriate node-sampled velocity fields, i.e., the node-sampled velocities extrapolated from the subdomain that contains $\boldsymbol{x}$ at time t_{n+1}.

7.2.2 Interface advection and levelset reinitialization

The final stage of the workflow per time step (see subsection 3.3.1) involves the advection of the interface from time t_{n+1} to time t_{n+2}. This is done by solving the advection equation on the levelset function (3.1) with an interface velocity $\boldsymbol{w}$. Theoretically, this interface velocity may be defined as either[1]

$$\boldsymbol{w} = \boldsymbol{u}^{+} - \frac{\dot{M}}{\rho^{+}}\boldsymbol{n}, \tag{7.6}$$

or

$$\boldsymbol{w} = \boldsymbol{u}^{-} - \frac{\dot{M}}{\rho^{-}}\boldsymbol{n}, \tag{7.7}$$

or any linear combination of the two. The solver allows the user to choose either (7.6), (7.7), an arithmetic average of the two, i.e.,

$$\boldsymbol{w} = \frac{1}{2}\left(\boldsymbol{u}^{+} - \frac{\dot{M}}{\rho^{+}}\boldsymbol{n}\right) + \frac{1}{2}\left(\boldsymbol{u}^{-} - \frac{\dot{M}}{\rho^{-}}\boldsymbol{n}\right) \tag{7.8}$$

or a mass-weighted averaged of the two, i.e.,

$$\boldsymbol{w} = \frac{\rho^{+}}{\rho^{+}+\rho^{-}}\left(\boldsymbol{u}^{+} - \frac{\dot{M}}{\rho^{+}}\boldsymbol{n}\right) + \frac{\rho^{-}}{\rho^{+}+\rho^{-}}\left(\boldsymbol{u}^{-} - \frac{\dot{M}}{\rho^{-}}\boldsymbol{n}\right). \tag{7.9}$$

By default, (7.9) is used. However $\boldsymbol{w}$ is defined, its interface-defined values are then extended normally to the interface in either subdomain, using the techniques from [172, 106].

Once the interface velocity $\boldsymbol{w}$ is defined, the levelset advection equation (3.1) is advanced from t_{n+1} to t_{n+2} using a second-order semi-Lagrangian method [106]: the value

[1]Since $\dot{M} = \rho\,(\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n}$, we have $\boldsymbol{w}\cdot\boldsymbol{n} = \boldsymbol{u}\cdot\boldsymbol{n} - \dot{M}/\rho$ on either side of the interface. Naturally, the tangential components of $\boldsymbol{w}$ are set equal to the tangential components of the fluid velocity on either side of the interface.

of $\phi\left(\boldsymbol{x}, t_{n+2}\right)$ is set to $\phi\left(\boldsymbol{x}^{\mathrm{d}}, t_{n+1}\right)$ where $\boldsymbol{x}^{\mathrm{d}}$ is defined as

$$\begin{aligned} \hat{\boldsymbol{x}} &= \boldsymbol{x} - \frac{\Delta t_{n+1}}{2}\boldsymbol{w}\left(\boldsymbol{x}, t_{n+1}\right), \\ \boldsymbol{x}^{\mathrm{d}} &= \boldsymbol{x} - \Delta t_{n+1}\tilde{\boldsymbol{w}}\left(\hat{\boldsymbol{x}}, t_{n+1} + \frac{\Delta t_{n+1}}{2}\right) \end{aligned} \tag{7.10}$$

where $\tilde{\boldsymbol{w}}\left(\boldsymbol{x}, t\right)$ interpolates the interface velocity linearly (in time) between t_n and t_{n+1}, i.e.,

$$\tilde{\boldsymbol{w}}\left(\boldsymbol{x}, t\right) = \boldsymbol{w}\left(\boldsymbol{x}, t_n\right) + \frac{\boldsymbol{w}\left(\boldsymbol{x}, t_{n+1}\right) - \boldsymbol{w}\left(\boldsymbol{x}, t_n\right)}{\Delta t_n}\left(t - t_n\right). \tag{7.11}$$

The velocity fields are interpolated in space with quadratic interpolation.

The interpolation in space of the levelset function is a user-defined choice. For a consistent interface description (e.g. the evaluation of θ values in chapter 5 and the in-cell interface reconstruction from chapter 6), the interpolation scheme must ensure a continuous representation of the levelset function ϕ across adjacent cells of same size. Linear interpolation or continuity-enforcing adaptations of the stabilized quadratic interpolation from [106] are used[2] (see appendix C).

Once advected, the levelset is systematically reinitialized to keep its signed-distance property. The reinitialization equation

$$\frac{\partial \phi}{\partial \tau} + \operatorname{sign}\left(\phi\right)\left(\left|\nabla \phi\right| - 1\right) = 0 \tag{7.12}$$

is solved in pseudo-time τ using a Total Variation Diminishing second-order Runge-Kutta method [183] with the subcell-resolution strategy from [184] (more details in [106]).

[2]Though satisfying the continuity requirement, Hermite quadratic interpolations may introduce spurious results due to discontinuous derivatives in signed distance functions.

7.2.3 Calculation of curvature

Though the numerical methods from subsection 7.2.2 are second order accurate, fluid velocities are only first-order accurate (in infinity norm) in our framework, as shown hereafter. As a consequence, the interface advection velocity $\boldsymbol{w}$ is only first-order accurate as well and so is the levelset representation of the interface after a finite simulation time.

In comparison with using a numerically smeared sign function when solving (7.12) (see [5]), the use of the second-order subcell, interface-pinning method from [184] alleviates mass loss during reinitialization. However, it is less prone to filtering noise out of a $\mathcal{O}\left(\Delta x\right)$ error signal at points close to the interface. This in turn may be responsible for spurious results when considering derivatives of the levelset function to extract interface-related properties, in particular if they depend on second derivatives (e.g. curvature). As illustrated in Figure 7.1 (left), the evaluation of the curvature $\kappa = \nabla \cdot \left(\frac{\nabla \phi}{\|\nabla \phi\|}\right)$ by means of standard differentiation was found to produce noisy, spurious results in regions of large curvatures, with significant variations over the length of one single grid cell.

In [10], comparable observations were made in the context of levelset simulations associated with Discontinuous Galerkin methods, leading to dramatically inaccurate results in such a context. The authors introduced the following alternative method for evaluating the curvature (and normal vectors) and showed that it produces significantly better (more accurate and less noise-corrupted) curvature results.

For a given point $(\bar{x}, \bar{y}, \bar{z})$ where the curvature is desired, a quadratic form

$$\phi_{\text{quad.}}\left(x, y, z\right) = \alpha_0 + \alpha_x x + \alpha_y y + \alpha_z z + \alpha_{xx}\frac{x^2}{2} + \alpha_{xy} xy + \alpha_{xz} xz + \alpha_{yy}\frac{y^2}{2} + \alpha_{yz} yz + \alpha_{zz}\frac{z^2}{2}$$

is built such that the error measure

$$\sum_{(x,y,z)\in\mathcal{N}^2(\bar{x},\bar{y},\bar{z})} \left|\phi_{\text{quad.}}\left(x-\bar{x}, y-\bar{y}, z-\bar{z}\right) - \phi\left(x,y,z\right)\right|^2$$

is minimized (see appendix B for more detail), where $\mathcal{N}^2\left(\bar{x},\bar{y},\bar{z}\right)$ is the set of second-degree node neighbors[3] of point $(\bar{x},\bar{y},\bar{z})$. The local curvature at $(\bar{x},\bar{y},\bar{z})$ can then be evaluated from the α coefficients as

$$\kappa_{\text{lsqr}} = \frac{\alpha_{xx}+\alpha_{yy}+\alpha_{zz}}{\alpha_x^2+\alpha_y^2+\alpha_z^2} - \frac{\alpha_{xx}\alpha_x^2+\alpha_{yy}\alpha_y^2+\alpha_{zz}\alpha_z^2+2\alpha_{xy}\alpha_x\alpha_y+2\alpha_{xz}\alpha_x\alpha_z+2\alpha_{yz}\alpha_y\alpha_z}{\left(\alpha_x^2+\alpha_y^2+\alpha_z^2\right)^{3/2}}. \tag{7.13}$$

As illustrated in Figure 7.1 (right), this approach significantly improves the evaluation of local curvature in our framework, as well. A similar method is used for the evaluation of $\nabla\boldsymbol{n}$ (gradient of normal vector), as required in presence of a nonzero mass flux across the interface.

7.2.4 Optional sub-refining grid for interface capturing

The sharp treatment of the interface requires the evaluation of the levelset function at relevant degrees of freedom in the numerical methods from chapters 5 and 6. The use of a MAC grid discretization naturally generates non-collocated degrees of freedom in the computational grid. Face- and cell-sampled degrees of freedom are the primary unknowns and the value of the levelset function at such points requires thus interpolation of the (node-sampled) levelset function. As a consequence, the actual description of the interface locally depends on the interpolation scheme for the levelset function, numerically.

The entire solver and the tools used therein have been extended with the possibility

[3]The set of second-degree node neighbors of a node $(\bar{x},\bar{y},\bar{z})$ is the set of vertices belonging to the cells surrounding $(\bar{x},\bar{y},\bar{z})$ and to their neighbor cells.

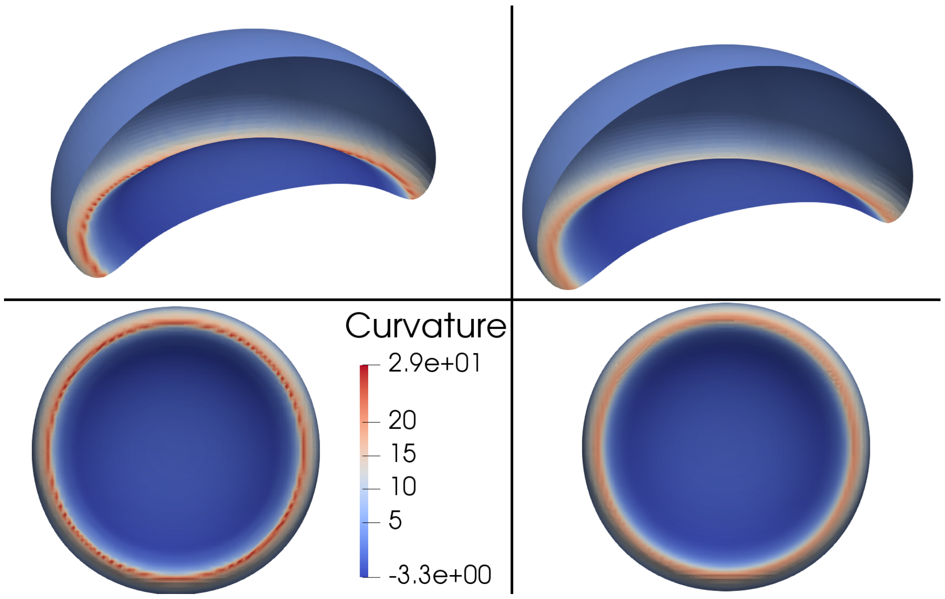


Figure 7.1: comparison between curvature evaluation by standard differentiation (left) and by the least-square local reconstruction (7.13) based on second-degree node neighborhood (right), for the steady state of a three-dimensional buoyant bubble ($\mu^+/\mu^- = \rho^+/\rho^- = 10^3,\ Eo = 116,\ Mo = 5.51$). Top: truncated three-dimensional view of the bubble. Bottom: views from the bottom of the bubble. The same color map is used in all figures.

of using a sub-refining, interface-capturing grid, that basically copies (local partitions of) the computational and exploits additional level(s) of refinement for cells crossed by the interface. As illustrated in Figure 7.2, this naturally produces a grid with nodes that are collocated with face and cell centers of the computational grid, close to the interface and therefore provides an unambiguous representation of the interface, numerically.

The motivation for this optional feature is to enable identification of under-resolved geometries and topology changes in future work, and the possible application of ad-hoc local strategies in such cases, regarding the calculation of curvature and/or normal vector for instance (see [185, 186, 187, 188]). Such a dual grid srategy has also been used in other works (see [51]). The use of a sub-refining, interface-capturing grid was also found to alleviate the inaccuracies in mass conservation inherent to levelset formulations, although it naturally adds to the computational and memory load of the simulation.

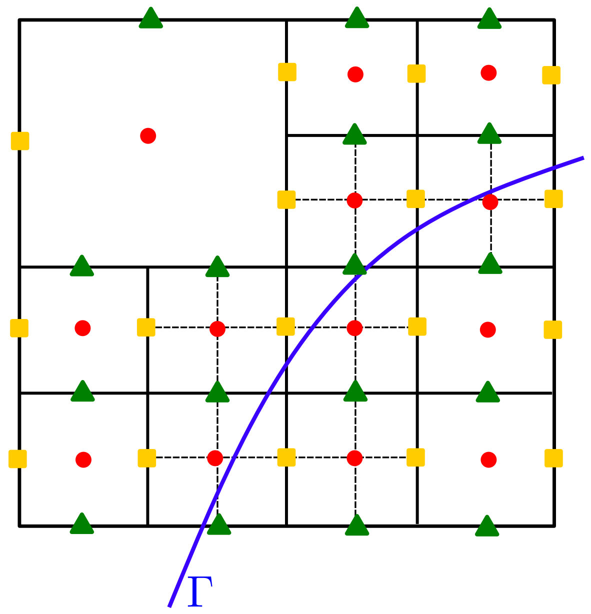


Figure 7.2: illustration of the use of a subrefining grid. The computational grid (solid lines) produces non-collocated degrees of freedom in a MAC grid discretization. The interface-capturing grid (solid *and* dashed lines) has its nodes collocated with the relevant degrees of freedom close to the interface.

7.2.5 Time step

By treating viscous terms implicitly, the stringent time step restriction

$$\Delta t_{\text{visc.}} \left(\max\left(\frac{\mu^+}{\rho^+}, \frac{\mu^-}{\rho^-} \right) \left(\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2} + \frac{2}{(\Delta z)^2} \right) \right) < 1 \tag{7.14}$$

(see [15]) is expected to be alleviated.

In presence of surface tension, guaranteeing numerical stability requires to ensure that the numerical propagation speed for capillary waves does not exceed the theoretical one. In [1], the time step constraint

$$\Delta t < \sqrt{\frac{(\rho^+ + \rho^-)(\Delta x)^3}{4\pi\gamma}} \tag{7.15}$$

was derived from these considerations. However, this time step restriction alone neglects viscous effects as it results from a phenomenon governed by inertia and surface tension only. At fine length scales, the balance between viscous and surface tension effects becomes more significant and a time scale of the order of $\mu\Delta x/\gamma$ may be better suited and less computational restrictive than (7.15) for small Δx.

In [32], the time step[4]

$$\Delta t = \min\left(c_0 \frac{\Delta x}{\max_\Omega \|\boldsymbol{u}\|}, \frac{c_1\mu_{\min}\Delta x}{\gamma} + \sqrt{\left(\frac{c_1\mu_{\min}\Delta x}{\gamma}\right)^2 + c_2\frac{(\rho^+ + \rho^-)(\Delta x)^3}{4\pi\gamma}}\right) \tag{7.16}$$

has been introduced, as an extension of the work from [189] for fluids of different viscosities and mass densities ($\mu_{\min} = \min(\mu^-, \mu^+)$). We found our approach to be stable under the same time step constraint (7.16): unless otherwise stated, we use $c_0 = 1$, $c_1 = 0.95$ and $c_2 = 0.95$.

7.2.6 Grid refinement

The refinement and coarsening criteria necessary for the construction of the trees are provided to `p4est`[101, 103] by means of cell-level callback functions. A combination of distance-based and vorticity-based refinement criteria are used in this work for building the computational grid. Specifically, consider a cell C and its vertices $V(C)$. If C is not already of maximum level of refinement, it may be marked for refinement in any of the

[4]Asymptotically, this time step definition satisfies $\Delta t \sim \mathcal{O}(\Delta x)$, as $\Delta x \to 0$. However, the second term in (7.16) can be rewritten as

$$\frac{c_1\mu_{\min}\Delta x}{\gamma}\left(1 + \sqrt{1 + \frac{c_2}{4\pi c_1^2}\left(\frac{\rho_{\min} + \rho_{\max}}{\rho_{\max}}\right)\left(\frac{\mu_{\max}}{\mu_{\min}}\right)^2 \frac{Re^2}{We}\frac{\Delta x}{L}}\right)$$

where the Reynolds and Weber numbers are defined as $Re = \dfrac{\rho_{\max}UL}{\mu_{\max}}$ and $We = \dfrac{\rho_{\max}U^2L}{\gamma}$ for a given velocity scale U and length scale L. Therefore, one may observe $\Delta t \sim \mathcal{O}(\Delta x)$ with reasonable grid resolutions only for moderate values of $\left(\dfrac{\mu_{\max}}{\mu_{\min}}\right)^2 \dfrac{Re^2}{We}$.

following cases:

1. C is within a user-defined band layering the interface. In Ω^- (resp. Ω^+), a band of b^- (resp. b^+) finest grid cells is enforced around the interface, by marking for refinement C if

$$\min_{\substack{v \in V(C)\\ v \in \Omega^s}} |\phi(v)| < b^s \max\left(\Delta x_{\text{finest}}, \Delta y_{\text{finest}}, \Delta z_{\text{finest}}\right) \tag{7.17}$$

where Δx_{finest}, Δy_{finest} and Δz_{finest} are the cell sizes along cartesian directions for the finest cells.

2. C is too coarse and too close to the interface. C is marked for refinement if its distance to the interface is found comparable to the length of its diagonal, i.e., if

$$\min_{v \in V(C)} |\phi(v)| \leq L \operatorname{diag}(\mathcal{C}). \tag{7.18}$$

Unless otherwise stated, we use $L = 1.2$.

3. C is in a region of high vorticity. Similarly to [83, 85, 52], a vorticity-based criterion is used: C is marked for refinement if

$$h_{\max} \frac{\max_{\substack{v \in V(C)\\ v \in \Omega^s}} \left\|\nabla \times \boldsymbol{u}(v)\right\|_2}{\max_{\Omega^s} \left\|\boldsymbol{u}\right\|_2} \geq P, \tag{7.19}$$

where $h_{\max}$ is the largest edge length of cell C and P is a user-defined parameter controlling the level of refinement (its value is the same for Ω^- and Ω^+ in the current implementation). By $P = \infty$, we indicate that vorticity-based refinement is considered.

Similar, consistent conditions with reverted inequality signs control whether or not computational cells may be marked for coarsening.

7.3 Numerical validation and benchmark testing

In this section we assess the performance of the solver and consider validation problems. In particular, we consider comparisons between our results and analytical (manufactured) solutions in order to assess and evaluate the rates of convergence for the primary unknowns of interest, i.e., fluid velocities and pressure. Actual two-phase flow benchmark problems are also considered thereafter.

7.3.1 Accuracy analyses by comparison with manufactured solutions

Manufactured problem 1

The first considered problem involves the manufactured velocity field (in non-dimensional form)

$$\begin{cases} u^- = (y-t+0.5)\left(r(x,y,t)^2-1\right)+1 \\ u^+ = 1 \end{cases} \qquad \begin{cases} v^- = -(x-t+0.5)\left(r(x,y,t)^2-1\right)+1 \\ v^+ = 1 \end{cases} \tag{7.20}$$

with the pressure field

$$\begin{cases} p^- = 2-r(x,y,t)^2 \\ p^+ = 0 \end{cases} \tag{7.21}$$

where $r(x,y,t)=\sqrt{(x-t+0.5)^2+(y-t+0.5)^2}$ and the moving interface is defined by the levelset function $\phi(x,y,t)=r(x,y,t)-1$. The solution is stationary in a reference frame moving with the interface.

The computational domain is $[-2,\ 2]^2$, the simulation time window is $[0,\ 1]$, the material parameters are set to $\mu^-=0.01$, $\mu^+=0.1$, $\rho^-=0.1$, $\rho^+=1$ and $\gamma=0$. On the walls of the computational domain, Dirichlet boundary conditions are considered for velocity components with homogeneous Neumann boundary conditions for the pressure. We feed the solver with the (sharp) source terms

$$\boldsymbol{f}^{\pm} = \rho^{\pm}\left(\frac{\partial \boldsymbol{u}^{\pm}}{\partial t} + \boldsymbol{u}^{\pm}\cdot\nabla\boldsymbol{u}^{\pm}\right) + \nabla p^{\pm} - \mu^{\pm}\nabla^{2}\boldsymbol{u}^{\pm}, \tag{7.22}$$

and the interface-defined force term

$$\boldsymbol{G} = \left[p\boldsymbol{n} - 2\mu\underline{\boldsymbol{E}}\cdot\boldsymbol{n}\right], \tag{7.23}$$

calculated from the manufactured solution (7.20)-(7.21) and the chosen parameters. This problem was introduced in [33], where a full-block discretization approach was undertaken (like (3.18)) combined with an iterative strategy similar to [28] (chapter 4) on all primary unknowns.

This test problem involves a nontrivial solution in the least dynamically active phase which makes it numerically more challenging. It also involves $[\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}\cdot\boldsymbol{n}] \neq 0$ (hence coupling velocity components and pressure through (3.14)) and a nonzero tangential component for the interface-defined force, i.e. $(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n})\cdot\boldsymbol{G} \neq \boldsymbol{0}$.

The errors (evaluated in max norm) are summarized in Table 7.1 for uniform grids of increasing resolution, at the final simulation time. Uniform grids are considered for comparison purposes with [33]. The ratio $\Delta t/\Delta t_{\text{visc.}}$ is also indicated to illustrate the gain in time step from an implicit treatment of viscous terms (see (7.14) and (7.16), where we have used $c_0 = 0.85$, $c_1 = 0.95$ and $c_2 = 0.95$). We used $k_{\max} = 5$ with $\xi = 0.1\%$ (see 3.3.1 and (3.22)) and limited the number of xGFM iterations to a maximum of 10 for the viscous steps

Compared to the results from [33], we obtain more accurate results on comparable grids. We observe first order accurate results for the velocity components. While the error on the interface location is observed to decrease with grid refinement, it indicates a less clear convergence rate, though (better than) first-order accuracy is obtained when comparing the coarsest and finest resolutions. The visualization of the error distributions shows that better than first order accuracy is observed for the levelset function between two successive grid resolutions (e.g. between the 32^2 and 64^2 grids) when the maximum error on the velocity field is reached away from the interface. In such a case, the error on the levelset due to advection, though first order, may be negligible for the considered grid resolution in comparison to the reinitialization error (second order). The convergence on the pressure seems more challenging which may be due to using $\left[\frac{1}{\rho}\partial_{\boldsymbol{n}}p\right] = 0$ instead of (3.16) but also to the nature of the problem involving a nontrivial solution in the least dynamically active phase, producing interface conditions that require fine resolution to be captured with accuracy. The maximum error on p is reached in Ω^+ on grids up to 256^2, then in computational cells crossed by the interface lying on either side for the 512^2 grid before being found in computational cells crossed by the interface, lying in Ω^- on the 1024^2 grid. We note that our results for p are comparable (in fact, slightly more accurate) to those from [33] on comparable grids.

Manufactured problem 2

The second considered problem involves the manufactured velocity field (in non-dimensional form)

$$u^- = u^+ = \sin(x)\cos(y)\sin(t) \qquad v^- = v^+ = -\cos(x)\sin(y)\sin(t) \tag{7.24}$$

Grid resolution	32^2	64^2	128^2	256^2	512^2	1024^2
Error on ϕ	$3.5e-2$	$7.9e-3$	$4.5e-3$	$1.3e-3$	$1.0e-3$	$5.2e-4$
order		2.15	0.81	1.79	0.38	0.94
Error on u	$4.0e-2$	$1.9e-2$	$8.9e-3$	$3.5e-3$	$2.1e-3$	$9.5e-4$
order		1.1	1.1	1.35	0.74	1.14
Error on v	$4.9e-2$	$1.9e-2$	$7.4e-3$	$3.3e-3$	$1.7e-3$	$9.0e-4$
order		1.4	1.4	1.2	0.97	0.92
Error on p	$6.5e-2$	$4.1e-2$	$2.3e-2$	$1.4e-2$	$1.1e-2$	$7.3e-3$
order		0.7	0.8	0.7	0.3	0.6
$\Delta t/\Delta t_{\text{visc.}}$	1.9	3.9	7.7	15.4	30.8	61.5

Table 7.1: maximum errors on all primary unknowns for the manufactured problem 1, at the final simulation time. The error on the levelset function is evaluated in a band of $3\Delta x$ across the interface.

with the pressure field $p^- = p^+ = 0$. The interface is defined as the zero-level set of

$$\phi(x, y) = \begin{cases} 0.1 - \sin(x)\sin(y) & \text{if } (x, y) \in [0,\ \pi]^2 \\ 0.1 & \text{otherwise.} \end{cases} \tag{7.25}$$

One may easily show that $\Gamma = \{(x, y) : \phi(x, y) = 0\}$ is a streamline of the flow field (7.24) and that it should thus remain static.

The computational domain is $[-\pi/3,\ 4\pi/3]^2$, the simulation time window is $[0,\ \pi]$, the material parameters are set to $\mu^- = 0.1$, $\mu^+ = 1$, $\rho^- = 0.1$, $\rho^+ = 1$ and $\gamma = 0.1$. We feed the solver with the (sharp) source terms

$$\boldsymbol{f}^{\pm} = \rho^{\pm}\left(\frac{\partial \boldsymbol{u}^{\pm}}{\partial t} + \boldsymbol{u}^{\pm}\cdot\nabla\boldsymbol{u}^{\pm}\right) + \nabla p^{\pm} - \mu^{\pm}\nabla^2\boldsymbol{u}^{\pm}, \tag{7.26}$$

and the interface-defined force term

$$\boldsymbol{G} = -\left[2\mu\underline{\boldsymbol{E}}\cdot\boldsymbol{n}\right] + \gamma\kappa\boldsymbol{n}, \tag{7.27}$$

calculated from the manufactured solution (7.24)-(7.25) and the chosen parameters. This problem is adapted from [32] such that the interface does not intersect the walls of the computational domain[5].

This test problem involves $[\mu \boldsymbol{n} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{n}] \neq 0$ (hence coupling velocity components and pressure through (3.14)) and a nonzero tangential component for the interface-defined force, i.e. $(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}) \cdot \boldsymbol{G} \neq \boldsymbol{0}$, and a non-uniform curvature. It is also particularly well-suited for testing the implementation of all numerical methods on adaptive grids in a fairly straightforward fashion: since the streamlines of the flow field 7.24 correspond to constant values of (7.25), the use of a refinement criterion based on the distance to Γ only does not produce major under-resolution issues. The solution at time $t = \pi/2$ is illustrated in Figure 7.3.

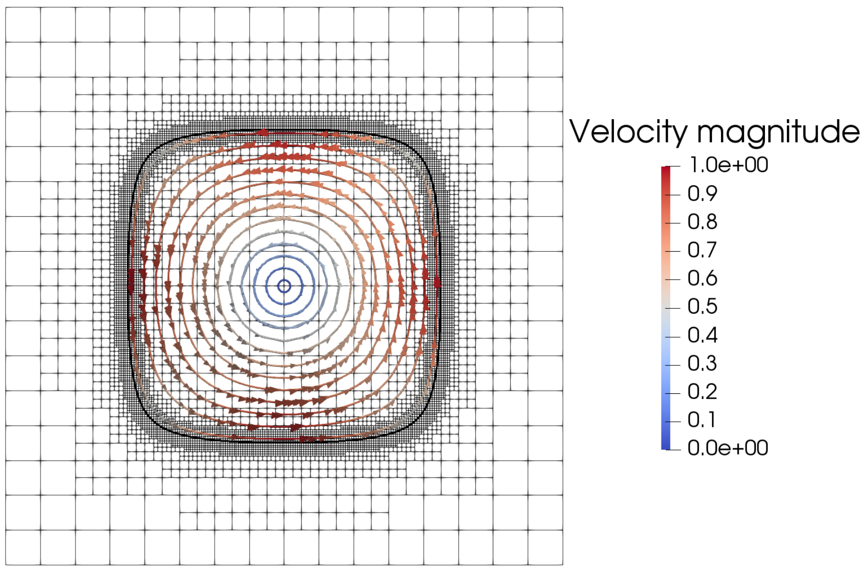


Figure 7.3: illustration of the computational grid ($\ell_{\text{min}} = 4$, $\ell_{\text{max}} = 8$), the interface (thick black line) and streamlines in Ω^- (colored by the velocity magnitude) for the manufactured problem (7.24) at time $t = \pi/2$.

We consider quadtree grids with increasing maximum level of refinement ℓ_{max} and

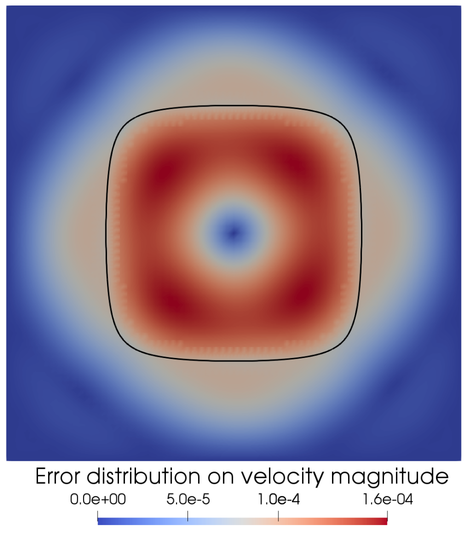
[5]Wettability problems are left for future work.

minimum level of refinement $\ell_{\min} = \ell_{\max} - 4$, with the following refinement parameters: $b^- = b^+ = 5$, $L = 1.2$, and $P = \infty$ (see subsection 7.2.6). The errors (evaluated in max norm) are summarized in Table 7.2 for grids of increasing resolution, at the final simulation time. Note that this analysis does not refine every single computational cell as the grid resolution is increased. The ratio $\Delta t/\Delta t_{\text{visc.}}$ is also indicated in Table 7.2 to illustrate the gain in time step from an implicit treatment of viscous terms (see (7.14) and (7.16), where we have used $c_0 = 0.85$, $c_1 = 0.95$ and $c_2 = 0.95$). We used $k_{\max} = 10$ with $\xi = 0.1\%$ (see 3.3.1 and (3.22)) and limited the number of xGFM iterations to a maximum of 10 for the viscous steps (a value of $k_{\max} > 5$ was necessary to ensure stability on the two finest grids).

Compared to the results from [32], we obtain more accurate results for velocities and interface location on comparable grids (the accuracy on p is not reported in [32]). Velocities and pressure seem to converge with (at least) first-order accuracy. The convergence on the levelset function (i.e. on the interface location) seems more challenging in this case. The visualization of the error distribution on the velocity field indicates that the maximum errors are found in Ω^- away from the interface on all considered grids but the last one, as illustrated in Figure 7.4. Therefore, the rates of convergence on the velocity components reported in Table 7.2 is not representative of the rates of convergence on the velocity components close to the interface for this particular test. This explains why one does not observe such a clear rate of convergence for the interface location in this particular case.

Manufactured problem 3, nonzero mass flux

Finally, we consider the capability of the solver to handle a nonzero mass flux across the interface (which represents a phase change phenomenon in practice). In actual applications, a nonzero mass flux would originate from additional physics being coupled to the



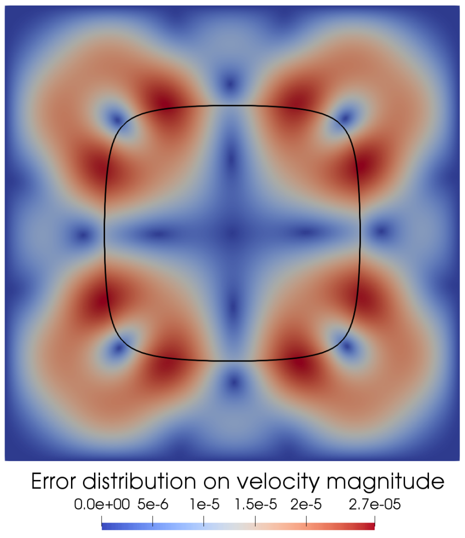


Figure 7.4: error distribution for the velocity field on computational grid $\ell_{\min} = 6$, $\ell_{\max} = 10$ (left) and $\ell_{\min} = 8$, $\ell_{\max} = 12$ (right), at final simulation time $t = \pi$. The interface is illustrated as the black line.

fluid flow (e.g. boiling or cavitation, for instance), which therefore adds another layer of complexity to the simulation tools. In this example, we are interested in assessing the capability of the solver to handle a nonzero mass flux, independently of its physical origin.

We therefore consider the growth of a gas bubble due to a constant and uniform mass flow rate $\dot{M}$, as considered first in [49]. In SI units, we consider a computational domain

$\ell_{\min}/\ell_{\max}$	4/8	5/9	6/10	7/11	8/12
Error on ϕ	$2.1e-3$	$1.6e-3$	$1.0e-3$	$7.9e-4$	$4.4e-4$
order		0.4	0.7	0.3	0.8
Error on u, v	$2.7e-3$	$9.3e-4$	$1.4e-4$	$4.5e-5$	$2.4e-5$
order		1.5	2.7	1.6	0.9
Error on p	$3.0e-3$	$1.0e-3$	$2.7e-4$	$1.2e-4$	$5.6e-5$
order		1.6	1.9	1.2	1.1
$\Delta t/\Delta t_{\text{visc.}}$	117.5	235.1	470.2	940.4	1880

Table 7.2: maximum errors on all primary unknowns for the manufactured problem 2, at the final simulation time. The error on the levelset function is evaluated in a band of $3\Delta x$ across the interface.

$\Omega = [-0.004,\ 0.004]^2$ with a circular bubble of radius $R_0 = 0.001$ initially (the negative domain is inside the gas bubble); the mass flux across the interface is set to $\dot{M} = -0.1$, constant in time and space. The material properties are set to $\mu^- = 1.78 \times 10^{-5}$, $\rho^- = 1$, $\mu^+ = 1 \times 10^{-3}$, $\rho^+ = 10^3$ and $\gamma = 0.07$ (all in Si units; this is similar to an air-water system).

Assuming that the fluid is and remains at rest inside the gas bubble, the interface velocity is $-\frac{\dot{M}}{\rho^-}\boldsymbol{e}_r$ where $\boldsymbol{e}_r$ is the unit radial vector, meaning that the bubble growths in an isotropic fashion and that its radius follows $R(t) = R_0 - \frac{\dot{M}}{\rho^-}t$, so the exact levelset function at time t is

$$\phi(x, y, t) = \sqrt{x^2 + y^2} - \left(R_0 - \frac{\dot{M}}{\rho^-}t\right). \tag{7.28}$$

Assuming rotational symmetry in an infinite (two-dimensional) medium and $\boldsymbol{u}^- = \boldsymbol{0}$, the velocity field in the liquid phase is entirely governed by the interface conditions and the incompressibility condition so that

$$\boldsymbol{u}^+ = \dot{M}\left[\frac{1}{\rho}\right]\left(\frac{R(t)}{\sqrt{x^2 + y^2}}\right)\boldsymbol{e}_r. \tag{7.29}$$

Finally, one may derive the corresponding pressure field[6] in the absence of external force,

$$\begin{cases} p^- = \frac{1}{2}\dot{M}^2\rho^+\left[\frac{1}{\rho^2}\right] + 2\mu^+\frac{\dot{M}}{R(t)}\left[\frac{1}{\rho}\right] + \frac{\gamma}{R(t)}, \\ p^+ = +\frac{\rho^+}{\rho^-}\left[\frac{1}{\rho}\right]\dot{M}^2\log\left(\frac{\sqrt{x^2 + y^2}}{R(t)}\right) - \frac{1}{2}\rho^+\frac{\left(\dot{M}\left[\frac{1}{\rho}\right]R(t)\right)^2}{x^2 + y^2}, \end{cases} \tag{7.30}$$

down to a constant (arbitrarily set to 0).

This test problem involves $[\mu\boldsymbol{n}\cdot\nabla\boldsymbol{u}\cdot\boldsymbol{n}] \neq 0$ (hence coupling velocity components

[6]Due to the two-dimensional nature of this problem, the far-field pressure is not bounded. Though this is a paradox from a physics viewpoint, this problem is numerically well-defined in a finite computational domain and appropriate to assess the targeted capability of the solver.

and pressure through (3.14)) and a nonzero mass flux across the interface $\dot{M} \neq 0$. We consider the numerical simulation of this test problem by feeding the solver with the appropriate initial condition, material parameters, and no bulk or interface-defined source terms, i.e., $\boldsymbol{f} = \mathbf{0}$ and $\boldsymbol{G} = \mathbf{0}$. The mass flux across the interface is externally defined (constant in space and time), we consider a simulation time window $t \in [0,\ 0.01]$ so that $R\left(t_{\text{end}}\right) = 2R_0$, and we mimic outflow conditions on the walls of the computational domain by using non-homogeneous Dirichlet (resp. Neumann) boundary conditions for p (resp. $\boldsymbol{u}$), consistent with (7.30) (resp. with (7.29)).

We consider uniform grids[7] of increasing resolution and analyze the error (in infinity norm), at the final simulation time. We have used $c_0 = 0.85$, $c_1 = 0.95$ and $c_2 = 0.95$) for the time step calculation (see (7.16)), and we have set $k_{\max} = 3$, while limiting the number of xGFM iterations to a maximum of 10 for the viscous steps. The results are presented in Table 7.3 and the solution at final simulation time is illustrated in Figure 7.5. We observe first-order of accuracy on velocity components and better than first-order accuracy on the interface location. As underlined in [49, 48], one does not observe convergence on interface location for such problems when using a smeared interface approach, which highlights the need for sharp treatment of interface conditions.

We note that the convergence rate for the pressure could be improved with a better curvature estimation and/or a less numerically polluted levelset function (i.e. more accurate velocity components and interface velocity) after a finite simulation time. Indeed, the error distribution on p was found to reach its maximum close to the interface and to be correlated with significant errors in the numerical estimation of the local curvature

[7]We restrict the analysis to uniform grids in this case: the investigations on adaptive grids revealed that a better-designed grid refinement criterion was required to ensure convergence in this particular case. The error on p was found to reach its maximum at T-junctions away form the interface in case of adaptive grids, and this maximum error failed to converge. However, the interface location was found to be predicted accurately as the error on the levelset function was found to converge, indicating that velocities near the interface (and interface jump conditions) are accurately captured.

Grid resolution	32^2	64^2	128^2	256^2	512^2
Error on ϕ	$5.9e-5$	$1.6e-5$	$4.9e-6$	$1.5e-6$	$4.6e-7$
order		1.9	1.7	1.71	1.7
Error on u, v	$4.2e-3$	$1.2e-3$	$7.5e-4$	$3.9e-4$	$1.8e-4$
order		1.8	0.7	0.9	1.1
Error on p	1.3	$3.3e-1$	$1.4e-1$	$7.3e-2$	$5.3e-2$
order		2.0	1.2	0.9	0.5

Table 7.3: maximum errors on all primary unknowns for the manufactured problem 3, at the final simulation time. The error on the levelset function is evaluated in a band of $3\Delta x$ across the interface. All values are in the corresponding SI unit.

(which should be constant for a circular interface). Unfortunately, such numerical errors in curvature estimation fail to converge with grid refinement (or indicate a convergence rate lower than 1). This test problem motivated the development of the alternative curvature evaluation from subsection 7.2.3, which improved former results (not shown in this document) by one order of magnitude on the finest computational grids.

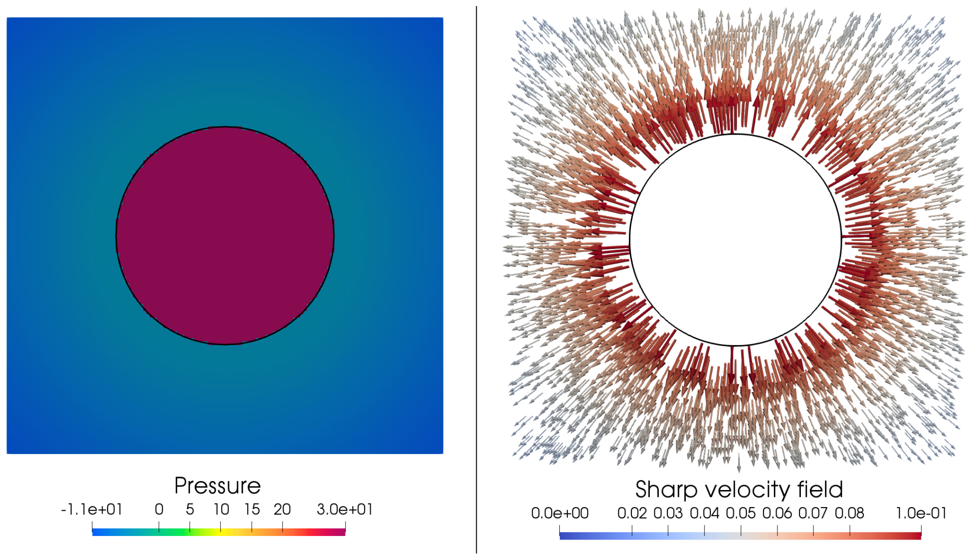


Figure 7.5: illustration of the final sharp pressure field (left) and final velocity field (right) for the manufactured problem (7.28)-(7.30), on a 128^2 computational grid. Note that the (numerical) velocity vector in the gas phase is displayed as well but too small to be observed.

7.3.2 Analysis of parasitic currents in two dimensions

One of the most infamous consequences of modeling surface tension effects with a continuum surface force model like [1] is the existence of strong parasitic currents. In practice, this spurious numerical artifacts imposes a restriction on the range of physical parameters that may be considered when using such modeling techniques. Reconciling the numerical approach with a sharp interface description of the corresponding physics dramatically reduces the magnitude of such parasitic currents.

In this analysis, we consider the canonical test of a circular, two-dimensional drop of diameter D immersed in a fluid of same viscosity and density (i.e. $\mu^- = \mu^+$ and $\rho^- = \rho^+$) with surface tension is γ (see [22, 14, 13, 19, 10, 32]). Of course, the exact solution of this problem is $\boldsymbol{u} = \boldsymbol{0}$ everywhere and a piecewise constant pressure field satisfying $[p] = -\gamma\kappa$. However, an imbalance between pressure gradient and surface tension exists numerically and it is responsible for nonzero fluid velocities (i.e., parasitic currents).

Physically, this problem is fully determined by the Laplace number $La = \dfrac{\gamma\rho D}{\mu^2}$. Based on dimensional grounds, the magnitude of the parasitic currents should be proportional to γ/μ, meaning that the capillary number $Ca = \dfrac{\|\boldsymbol{u}\|\,\mu}{\gamma}$ should be independent of La.

We consider the simulation of the problem described here above in two dimensions, with a drop of diameter D centered in a computational domain of side length $5D/2$. We set no slip boundary conditions on the walls of the computational domain with homogeneous Neumann boundary conditions on the pressure field. The simulations are conducted until $t = 75\mu D/\gamma$ and the maximum value of $\|\boldsymbol{u}\|$ is monitored over the course of the simulation. We consider quadtree grids with a minimum level of refinement $\ell_{\text{min}} = 2$ and maximum levels of refinement ℓ_{max} ranging from 7 to 10 ($\ell_{\text{max}} \geq 7$ to ensure a mass loss of less than 1%). The vorticity-based refinement criterion is deactivated ($P = \infty$ in (7.19)) and $b^- = b^+$ are chosen so that 15% of the bubble radius is covered with the

La	1.2	12	120	1,200	12,000
$\ell_{\max} = 7$	$8.1e-5$	$8.0e-5$	$6.8e-5$	$4.4e-5$	$2.0e-5$
$\ell_{\max} = 8$	$1.9e-5$	$1.8e-5$	$1.4e-5$	$1.1e-5$	$6.1e-6$
$\ell_{\max} = 9$	$6.0e-6$	$5.9e-6$	$6.0e-6$	$5.8e-6$	$4.6e-6$
$\ell_{\max} = 10$	$1.7e-6$	$1.7e-6$	$1.7e-6$	$1.6e-6$	$1.4e-6$

Table 7.4: values of $\|\boldsymbol{u}\|_{\max} \mu/\gamma$ for various grid resolution and Laplace numbers. $\|\boldsymbol{u}\|_{\max}$ is the maximum magnitude of parasitic currents observed over the course of the simulation.

finest computational cells on either side of the interface. We used $k_{\max} = 3$ for all runs, except for the finest computational grid with $La = 1.2$ which required more iterations for stability (we set $k_{\max} = 5$ in that case). The results are presented in Table 7.4. We observe close to second-order accurate results. Compared to previous works [22, 13], the current methodology requires finer grid resolution for comparable magnitudes of parasitic currents.

7.3.3 Shape oscillations of a three-dimensional droplet

At the microscopic scale, surface tension originates from the imbalance in cohesive forces between materials. Unless these cohesive forces are locally modified and made nonuniform (e.g., by the addition of surfactants), surface tension tends to minimize the area of the interface between two fluids, hence producing spherical droplet and bubble shapes at equilibrium in the absence of other forces.

Let us consider a drop of fluid of viscosity μ^- and mass density ρ^- surrounded by another fluid of viscosity μ^+ and mass density ρ^+; the surface tension between the two fluids is γ. Assuming a radially convex shape for the drop at all times and considering spherical coordinates centered with the drop, the interface can be represented as $\Gamma = \{\boldsymbol{x} \in \mathbb{R}^3 : \boldsymbol{x} = (R_0 + \delta R(\theta, \phi, t))\, \boldsymbol{e}_r\}$ where θ and ϕ are the inclination and azimuthal angles, and $\boldsymbol{e}_r$ is the unit radial vector.

The prediction of the interface kinematics for a given initial state was first considered theoretically for inviscid fluids by Sir Horace Lamb [190] who showed that, for an infinitesimal initial perturbation as a spherical harmonic of order n, i.e., $\delta R(\theta, \phi, 0) = \varepsilon P_n(\cos(\theta))$, the kinematics of the interface could be described as $R_0 + \varepsilon P_n(\cos(\theta))\cos(\omega_n t)$ where

$$\omega_n = \sqrt{\gamma \frac{n(n-1)(n+1)(n+2)}{R_0^3(\rho^-(n+1)+\rho^+ n)}}. \tag{7.31}$$

Though strictly valid for inviscid fluids only, Lamb used this result to estimate the rate of damping due to viscosity (neglecting viscous effects in Ω^+) and conjectured that, in such a case, the interface kinematics could be described as $R_0 + \varepsilon P_n(\cos(\theta))\cos(\omega_n t)\, e^{-t/\tau_n}$ where

$$\tau_n = \frac{R_0^2 \rho^-}{\mu^-(2n+1)(n-1)}. \tag{7.32}$$

This analysis was refined later on by Miller and Scriven [191] and Prosperetti [192]. The latter predicts an interface kinematics of the type $R_0 + \varepsilon P_n(\cos(\theta))\,\mathbb{R}\{e^{\sigma t}\}$ where σ is the complex root of a complex nonlinear characteristic equation.

In light of this strong theoretical basis, this problem has become a benchmark test for the validation of simulation methods for two-phase flows as well (see [22, 26, 193, 12]). We consider the numerical simulation of this phenomenon and compare our simulation results to the theoretical predictions of Lamb [190] and Prosperetti [192]. A satisfying match on the frequency of oscillation is obtained when inertia and surface tension are adequately captured while a correct prediction for the damping rate requires viscous effects to be accurately accounted for. Since no two-dimensional equivalent for the theoretical damping rate could be found, we consider three-dimensional simulation runs.

We consider the same material parameters as in [22], for a drop of radius R_0 centered in a domain $\Omega = [-4R_0,\ 4R_0]^3$ and[8] $\varepsilon/R_0 = 0.01$. No-slip boundary conditions are set

[8]We set ε/R_0 significantly lower than in other works to better comply with the underlying assumptions

on the walls of the computational domain. We consider a drop deformation as a second spherical harmonic, that is $n = 2$ and $P_2(\cos(\theta)) = \dfrac{3\cos^2(\theta) - 1}{2}$ (higher order modes of deformation are damped more aggressively). Besides the ratio $R_0/L = 1/8$ where L is the side length of the computational domain, the problem is entirely determined by the following nondimensional numbers

$$\frac{\varepsilon}{R_0} = 0.01, \quad \frac{\mu^-}{\mu^+} = 1000, \quad \frac{\rho^-}{\rho^+} = 1000, \quad \frac{\gamma \rho^- R_0}{(\mu^-)^2} = 1250. \tag{7.33}$$

The simulations are run for two full oscillation cycles on octree grids. The interface kinematics at time t is extracted by determining $a_2(t)$ such that the levelset is best represented by $\phi(r, \theta, \phi; t) = r - R_0(1 + a_2(t) P_2(\cos(\theta)))$, as in [26]. The maximum level of refinement $\ell_{\max}$ is increased from 7 to 9 ($\ell_{\max} \geq 7$ because the mean radius of the bubble shrinks by more than 1% otherwise) and the minimum level of refinement is set to $\ell_{\min} = \ell_{\max} - 3$.

The results are presented in Figure 7.6 where we considered two different numerical configurations: on one hand we used a fractional step approach ($k_{\max} = 1$) with a limit of 10 xGFM iterations for the viscous steps and, on the other hand, we enabled a maximum of $k_{\max} = 3$ fix-point iterations (repeating viscous and projection steps), limiting the number of internal xGFM iterations to 5 for the viscous steps. Except for the coarsest computational grid on which the second configuration estimates a larger damping rate, numerical results do not differ much in either case otherwise and agree well with the theoretical prediction as the grid is refined.

For the particular choice of material parameter (7.33), the capillary time step restriction is more restrictive than the (explicit) viscous time step restriction, which inhibits the benefit of an implicit treatment of viscous terms. The same analysis was conducted

of the theoretical works [190, 191, 192].

with the alternative set of parameters

$$\frac{\varepsilon}{R_0} = 0.01, \qquad \frac{\mu^-}{\mu^+} = 10, \qquad \frac{\rho^-}{\rho^+} = 100, \qquad \frac{\gamma\rho^- R_0}{\left(\mu^-\right)^2} = 900 \tag{7.34}$$

for which we could use a time step approximately 5 times larger than $\Delta t_{\text{visc.}}$ on the finest computational grid. The results for a fractional step approach ($k_{\max} = 1$) with a limit of 10 xGFM iterations are presented in Figure 7.7. Although the numerical results do approach the theoretical predictions as the grid is refined in this case as well, the damping rate seems to be significantly overestimated, compared to the set of parameters (7.33), which may be inherent to the implicit treatment of viscous terms in association with a large time step. The frequency of oscillation is also underestimated in comparison.

The simulations on the finest computational grids were done using 80 cores on four Intel 6148 CPUs on Pod cluster (center for scientific computing, UCSB). The run with $k_{\max} = 1$ and (7.33) required about 10 days of execution time.

7.4 Dynamics of buoyant bubbles in a viscous fluid

The rising motion of a gas bubble due to buoyancy in a surrounding viscous liquid is probably one of the most familiar real-life application of incompressible two-phase flows (in the absence of phase change). The direct numerical simulation of this phenomenon has also attracted interest over the past decades [22, 23, 26, 32, 10]. Of course, this phenomenon had also been extensively investigated experimentally in former works [194, 195, 196]: the ascension velocities they measured and the steady-state bubble shape they observed provide key comparison points for assessing the performance of a simulation tool on such a real-life application.

We consider a spherical bubble of diameter D and material properties μ^-, ρ^- (gas),

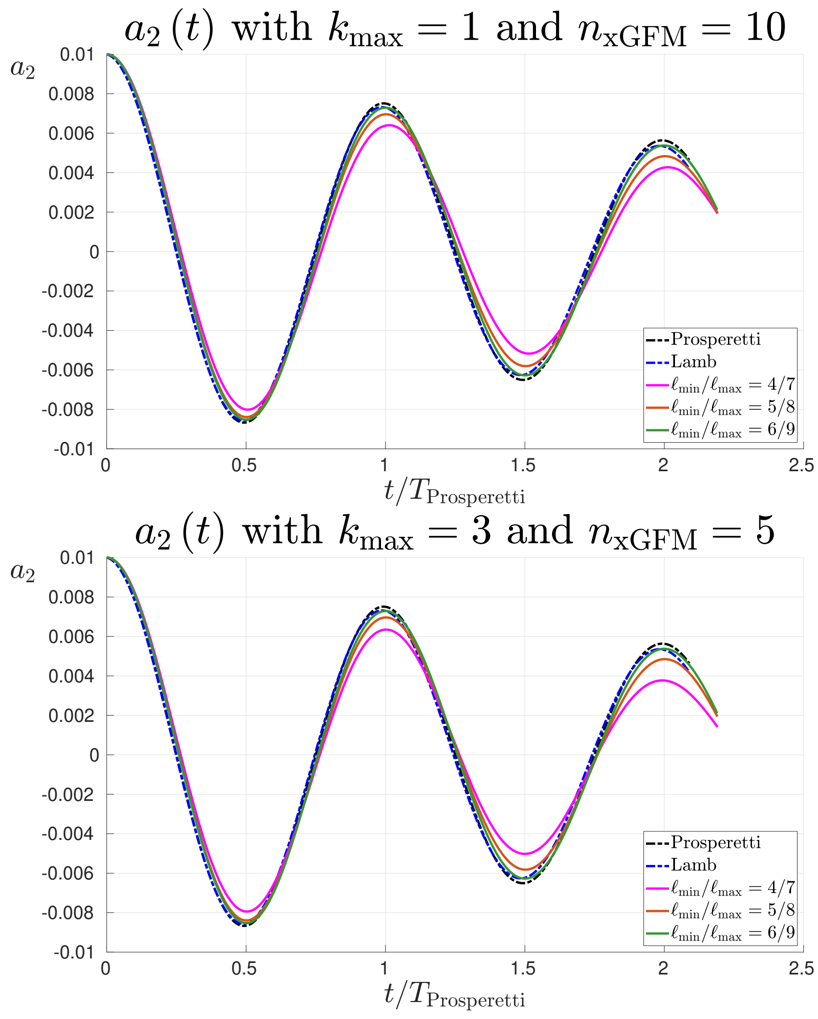


Figure 7.6: amplitude of the second harmonic deformation of the drop for the problem of subsection 7.3.3, with parameters (7.33). Top: $k_{\max} = 1$ with a limit of 10 xGFM iterations for the viscous steps; bottom: $k_{\max} = 3$ with a limit of 5 xGFM iterations for the viscous steps. The agreement with theoretical predictions gets better as the computational grid is refined.

surrounded by another fluid of properties μ^+, ρ^+ (liquid); the surface tension between the two fluids is γ. We are interested in the motion and deformation of the bubble resulting from buoyancy, as gravity acts as the only bulk force per unit mass, i.e. $\boldsymbol{f} = -g\boldsymbol{e}_y$ where g is the gravity acceleration. This physical set-up is entirely determined by the

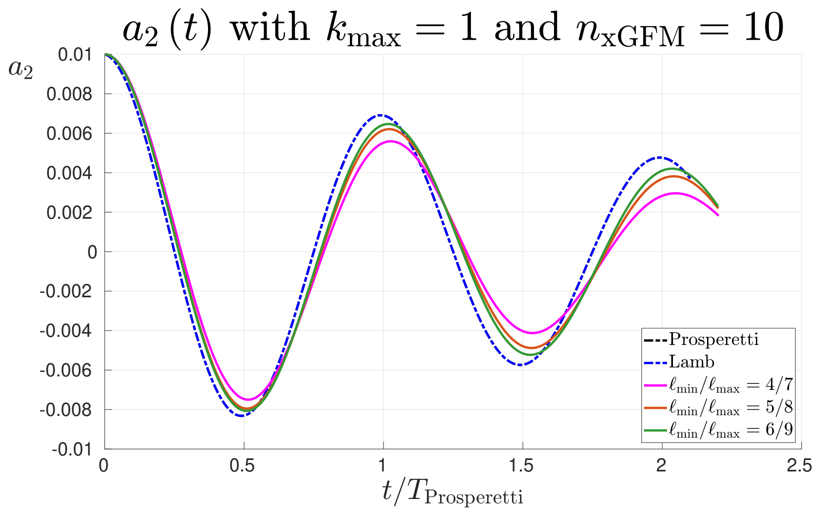


Figure 7.7: amplitude of the second harmonic deformation of the drop for the problem of subsection 7.3.3, with parameters (7.34). These results were obtained with $k_{\max} = 1$ and a limit of 10 xGFM iterations for the viscous steps. The theoretical predictions of Lamb and Prosperetti overlap in this case.

four nondimensional numbers

$$\Pi_\mu = \frac{\mu^+}{\mu^-}, \quad \Pi_\rho = \frac{\rho^+}{\rho^-}, \quad Eo = \frac{\rho^+ D^2 g}{\gamma} \quad \text{and} \quad Mo = \frac{g\left(\mu^+\right)^4}{\rho^+\gamma^3}. \tag{7.35}$$

When considering the numerical simulation of this phenomenon, a finite computational domain is considered with no-slip wall boundary conditions on all walls except the top wall on which a homogeneous Neumann boundary condition is set on velocity components along with a homogeneous Dirichlet boundary condition on pressure.

Over the course of the simulation, the ascension velocity of the bubble

$$\langle V \rangle = \frac{\int_{\Omega^-} \boldsymbol{u} \cdot \boldsymbol{e}_y \, \mathrm{d}\Omega}{\int_{\Omega^-} \mathrm{d}\Omega} \tag{7.36}$$

is evaluated and monitored as a Reynolds number $Re = \dfrac{\rho^+ \langle V \rangle D}{\mu^+}$. The corresponding Weber number is defined as $We = \dfrac{\rho^+ \langle V \rangle^2 D}{\gamma} = Re^2 \sqrt{\dfrac{Eo}{Mo}}$.

7.4.1 Small air cavity in water (two dimensions)

Although this problem is intrinsically three-dimensional, numerical simulations allow a two-dimensional equivalent to be considered. As an intermediary step toward full three-dimensional runs, we consider the numerical simulation of this problem in two dimensions and consider the same parameters as in [15, 33] for comparison purposes (air-water system). This choice of parameters corresponds to

$$\Pi_\mu = 63.9, \quad \Pi_\rho = 815.7, \quad Eo = 6.0 \quad \text{and} \quad Mo = 4.2 \times 10^{-11}.$$

The computational domain is $[-1.5D,\, 1.5D] \times [-1.5D,\, 3D]$ and the bubble is initially centered at the origin. We consider the simulation until the final (non-dimensional) time $\frac{\gamma t_{\text{end}}}{\mu^+ D} = 480$. Besides the ascension velocity, we also monitor the circularity measure

$$\text{circularity} = \frac{\pi D}{\int_\Gamma \mathrm{d}\Gamma} \tag{7.37}$$

for comparison purposes with [33]. We consider uniform and adaptive grids of increasing resolutions, with $k_{\max} = 5$ and a limit of 5 xGFM iterations per viscous step. The shape of the two-dimensional bubble at the final simulation time is illustrated in Figure 7.8 (left) for uniform grids of increasing resolutions: we obtain a good qualitative agreement with the results from [15, 33]. As this figure also illustrates, one does not sacrifice much accuracy on the interface kinematics when enabling adaptive grids: the predicted interface location is close to the predicted result obtained on uniform grid of equivalent finest resolution. The results for the ascension velocity and circularity are illustrated in Figure 7.9 and show good agreement with [33] as well. The mass loss at final simulation time on uniform grids of resolution 32×64, 64×128, 128×256, 256×512 were respectively 2%, 0.7%, 0.25% and 0.03%.

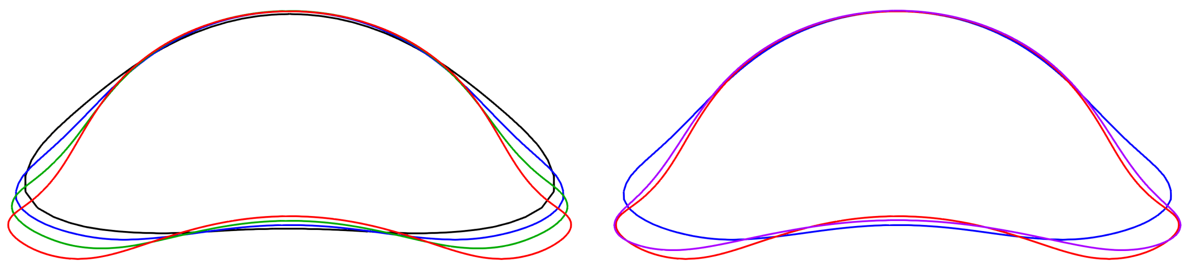


Figure 7.8: shapes of the two-dimensional bubble at final simulation time for the problem of subsection 7.4.1, with various grid resolution. Left: results on uniform grids of increasing resolution (the black, blue, green and red interfaces correspond respectively to 32×64, 64×128, 128×256 and 256×512 computational grids). Right: comparison of the results obtained on uniform grids of resolution 64×128 and 256×512 (blue and red interfaces) with the result obtain on an adaptive grid sharing the minimum and maximum levels of refinement of both (purple interface).

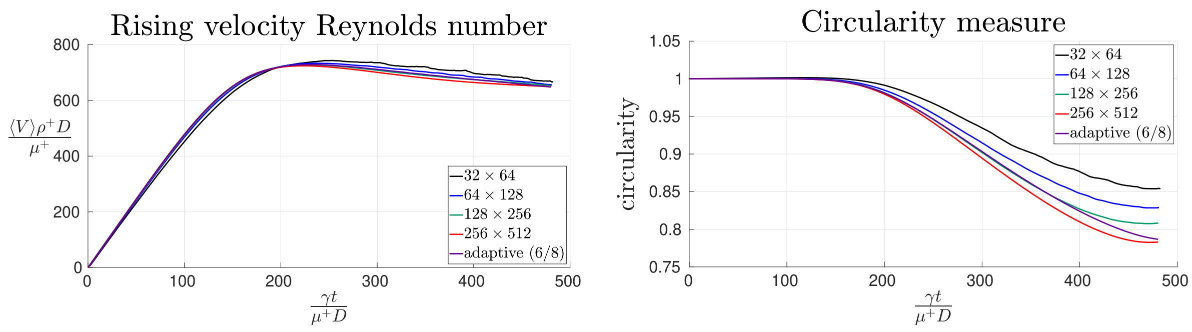


Figure 7.9: evolution of the (nondimensional) ascension velocity (left) and interface circularity (right) for the problem of subsection 7.4.1, with various grid resolution. The "adaptive (6/8)" grid has a minimum resolution equivalent to 64×128 and a maximum resolution equivalent to 256×512.

7.4.2 Dynamics of three-dimensional bubbles in viscous fluids

We now consider the three-dimensional dynamics of buoyant bubbles in viscous liquids. For this purpose, we consider 7 of the configurations experimentally observed and reported by Bhaga and Weber [195]. We consider the computational domain $[-5D, 5D] \times [-8D, 32D] \times [-5D, 5D]$ and the gas bubble is placed at the origin initially. We set the ratio of material properties to $\Pi_\mu = \Pi_\rho = 10^3$; the surface tension and gravity acceleration are determined accordingly to the desired values of[9] Eo and Mo, for each considered configuration.

The computational domain is meshed with 4 octrees (to ensure unit aspect ratio for the computational cells) of minimum level of refinement $\ell_{\text{min}} = 6$ and maximum level of refinement $\ell_{\text{max}} = 9$ or $\ell_{\text{max}} = 10$ for cases involving more challenging interface dynamics, and/or as to alleviate excessive mass loss. The number of (finest) grid cells per diameter D is $2^{\ell_{\text{max}}}/10$ (i.e., approximately 50 for $\ell_{\text{max}} = 9$ and 100 for $\ell_{\text{max}} = 10$). We consider $k_{\text{max}} = 3$ with a threshold value of 0.1% for (3.22), while limiting the number of xGFM iterations per viscous step to 10; the simulations are run until steady state. The vorticity-based refinement criterion is used with $P = 0.02$ in (7.19) and $b^- = b^+$ are chosen so that finest computational cells cover a layer of thickness $0.15D$ across the interface.

The evolution of the bubble for $Eo = 116$, $Mo = 0.103$ is illustrated with a slice in the computational grid in Figure 7.11, for snapshots corresponding to annotated times in Figure 7.10 showing the monitored ascension velocity of the bubble over the course of the simulation. In this particular example, the bubble undergoes large transient deformations that require a fine level of refinement to be correctly captured (see snapshot C in Figure 7.11, for instance). As illustrated in Figure 7.11, the vorticity-based refinement criterion produces regions of fine computational cells capturing the wakes generated by the ascending motion of the bubble. This improves the estimation of the pressure difference above and below the bubble which plays an important role in the evaluation of the ascension velocity. At steady state, the computational grid contains around 6×10^6 computational cells in our framework whereas a uniform grid of equivalent resolution would

[9]We note that this approach differs from the experimental procedure described in [195], in which the main tuning parameter is the viscosity of the external liquid μ^+ (aqueous sugar solution). In fact, the value of Π_μ is even larger than 10^5 in some cases of [195]. A strict equivalence of parameters would have led to very challenging (and probably prohibitively expensive) simulations. Setting $\Pi_\mu = 10^3$ may be responsible for discrepancies between our results and the expectations, although we expect little dependence on Π_μ so long as $\Pi_\mu \gg 1$.

require more than 4×10^9 grid cells. The final simulation state for $Eo = 116$, $Mo = 1.31$ is also illustrated in Figure 7.12 which shows that, in a reference frame moving with the buoyant bubble, the streamlines do not cross the interface, as expected.

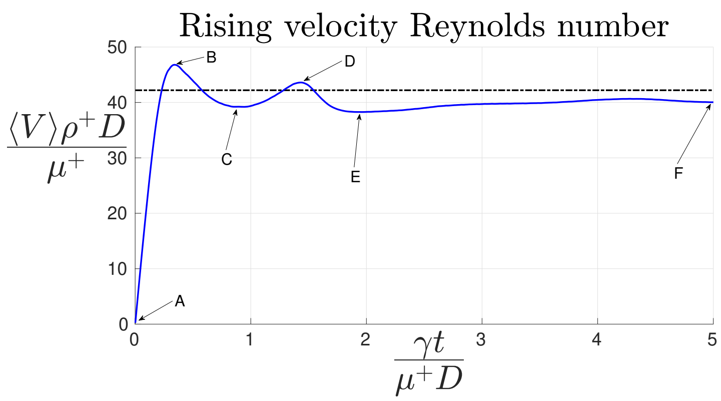


Figure 7.10: evolution of the ascension velocity (in its nondimensional form) for the simulation of the buoyant bubble with $Eo = 116$, $Mo = 0.103$. The dashed line represents the value reported in [195]. The bubble shape and (a slice of) the computational grid are illustrated n Figure 7.11 for the annotated times.

In Figure 7.13, we compare the simulation results for the final bubble shape with the experimental observations from [195]. As illustrated in this figure, the agreement with the expected bubble shape (observed experimentally) is qualitatively satisfactory for the considered parameters. In Table 7.5, the ascending Reynolds number is presented for each considered run: the simulation results are found to be within 10% of the experimental values. Since we use a levelset formulation for the interface representation, strict volume conservation is not guaranteed over the course of the simulation: we note that the volume of the bubble was found to increase in our framework for cases involving large bubble deformations.

The simulations were run on SKX nodes on Stampede 2 (Intel Xeon Platinum 8160, “Skylake”). 192 (resp. 384) MPI tasks were used for cases involving $\ell_{\max} = 9$ (resp. $\ell_{\max} = 10$) and the total execution time ranged from several hours to several days of computation (involving restarts).

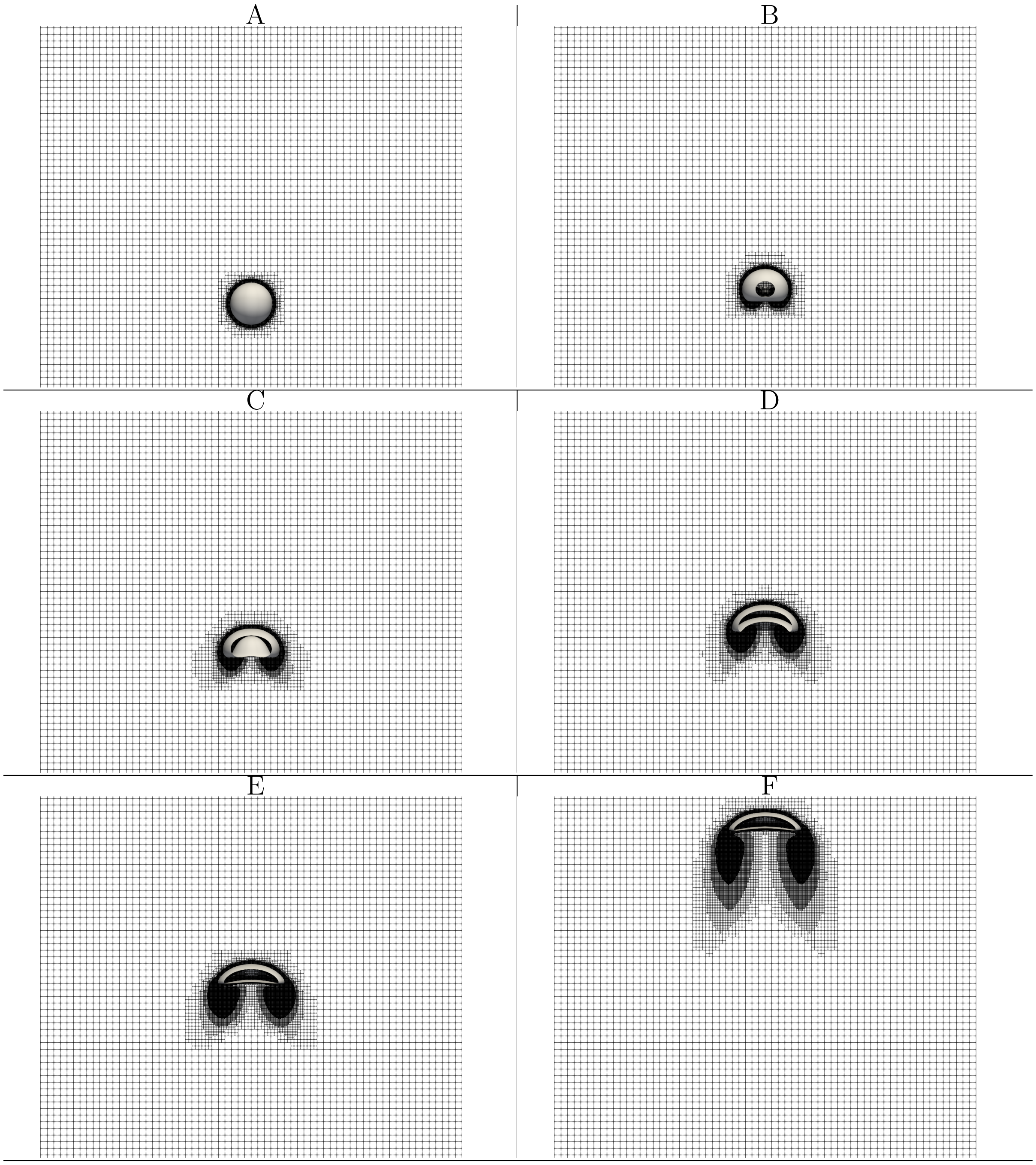


Figure 7.11: illustration of the (truncated) bubble shape along with a slice in the computational grid for $Eo = 116$, $Mo = 0.103$. See Figure 7.10 for references to the snapshot tags.

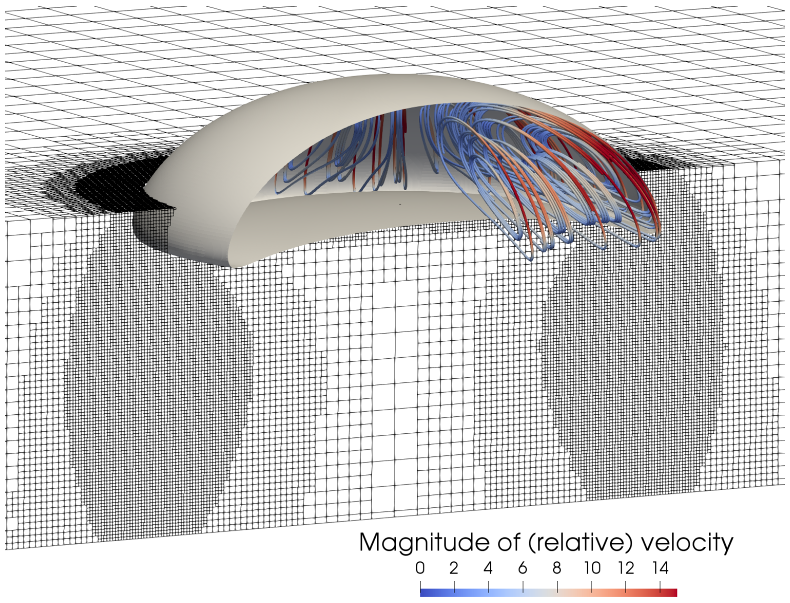


Figure 7.12: illustration of the final state for the simulation of a buoyant bubble with $Eo = 116$, $Mo = 0.103$. The truncated interface is represented along with two perpendicular cross sections of the computational grid. The illustrated streamlines correspond to the fluid velocity field inside the bubble, relatively to the bubble's ascension velocity.

7.5 Applications and illustrations with a nonzero mass flux

In this final section, we consider actual two-phase flow problems involving a phase change, i.e., problems that require $\dot{M} \neq 0$. For simplification purposes, we will consider problems involving phase transition of pure substances due to temperature, i.e. evaporation or boiling of a pure liquid substance into its (pure) vapor phase.

As underlined by equation (3.7), the strict consideration of all interface phenomena involved in the balance of energy across the interface leads to a nonlinear equation for $\dot{M}$ involving viscous effects, which are unknown at the required time step $n + 1$, prior to solving the considered time step. The exact consideration of such viscous terms when determining $\dot{M}$ would require yet another iterative approach. However, it was show in

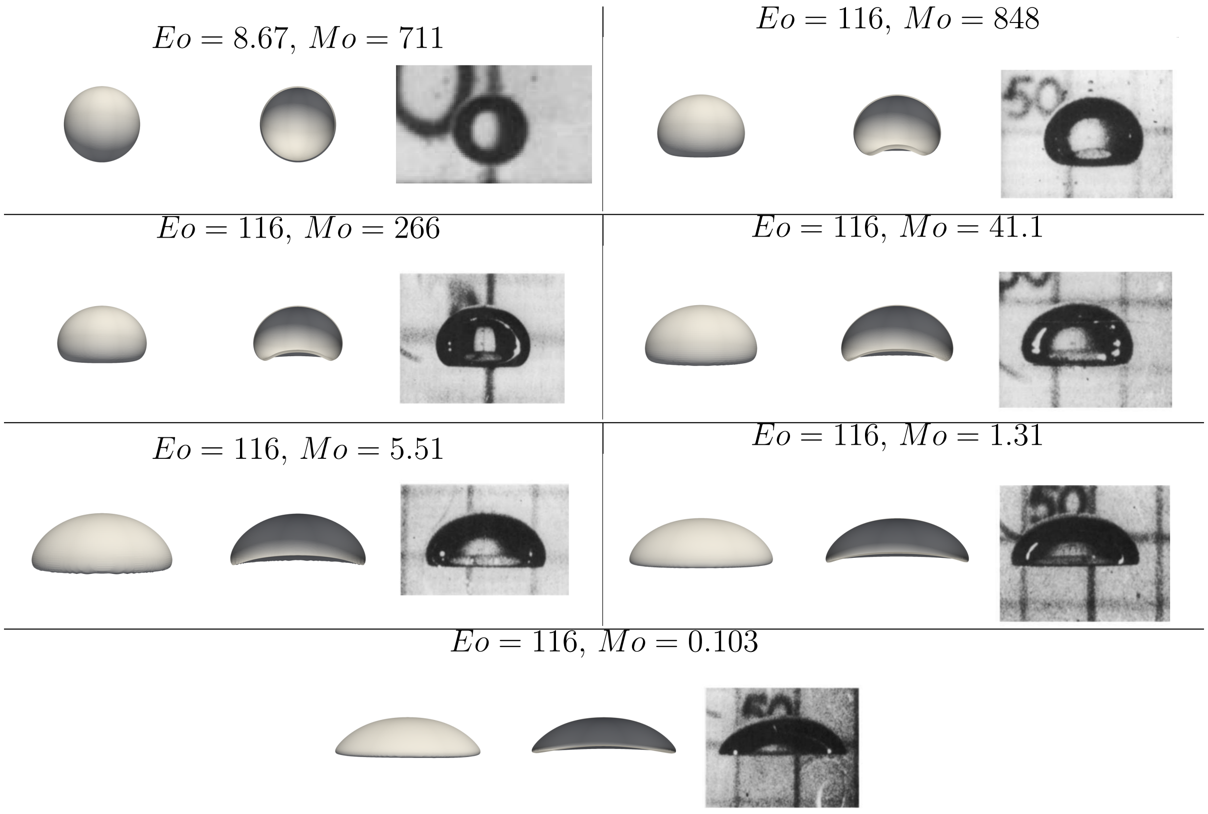


Figure 7.13: comparison between the steady-state bubble shapes, as numerically simulated, and the experimentaly observed shapes. For every run, the final shape is illustrated (left subfigures), along with a truncated version (center subfigures) and the experimental result (right subfigures) as reported in [195]. For every run, $\Pi_\mu = \Pi_\rho = 10^3$.

[36] that viscous effects are negligible in (3.7) for the substances considered therein (including water), in boiling flow applications. For simplification purposes, we also neglect a dependence of the surface tension on temperature. Finally, it is common to neglect the difference in kinetic energy across the interface (nonlinear terms in $\dot{M}$ in (3.7)) compared to the difference of enthalpy across the interface in boiling applications (see [51, 50, 49, 44, 39]).

Taking such simplifying assumptions into consideration, the conservation of energy

	$Re\|_{t_{\text{end}}}$ (sim.)	$Re\|_{t_{\text{end}}}$ (exp. [195])	$\ell_{\min}/\ell_{\max}$	$\Delta t/\Delta t_{\text{visc.}}$	$(V_{\text{end}} - V_0)/V_0$
$Eo = 8.67,\ Mo = 711$	0.073	0.078	6/9	38.3	−5%
$Eo = 116,\ Mo = 848$	2.43	2.47	6/9	20.2	−5%
$Eo = 116,\ Mo = 266$	3.7	3.57	6/9	15.0	−3.4%
$Eo = 116,\ Mo = 41.1$	7.4	7.16	6/9	9.3	+1%
$Eo = 116,\ Mo = 5.51$	14.4	13.3	6/9	5.6	+14.6%
$Eo = 116,\ Mo = 1.31$	20.4	20.4	6/10	5.5	+3.3%
$Eo = 116,\ Mo = 0.103$	40.0	42.2	6/10	2.9	+6.5%

Table 7.5: quantitative results associated with the considered numerical simulation of buoyant bubbles in viscous fluids. $(V_{\text{end}} - V_0)/V_0$ represents the relative difference in bubble volume at final simulation time, compared to its initial value.

across the interface reduces to

$$\dot{M}\,[h] - [k\nabla T \cdot \boldsymbol{n}] = 0, \tag{7.38}$$

where h is the enthalpy per unit mass. Considering the interface entropy and the second law of thermodynamics, the temperature must be continuous across the interface, i.e. $[T] = 0$ (see [197, 198]). Let T_{I} be the interface temperature and T_{sat} be the saturation temperature at the considered (ambient) pressure. Assuming linear dependence of h with respect to temperature, we have

$$[h] = L + [C_p]\,(T_{\text{I}} - T_{\text{sat}})\,, \tag{7.39}$$

where L is the latent heat associated with the phase change of the considered substance from Ω^- to Ω^+, at the saturation temperature.

In [36], models for $(T_{\text{I}} - T_{\text{sat}})$ have been presented, accounting for variation of saturation temperature with local pressure, thermodynamics considerations and other molecular phenomena. Such models naturally make $(T_{\text{I}} - T_{\text{sat}})$ another interface-defined unknown that depends on $\dot{M}$ (and other parameters). In this work, we do not account for such considerations and we consider $T_{\mathrm{I}} = T_{\mathrm{sat}}$, as commonly done in former works treating boiling or evaporating flows (see [51, 50, 49, 38, 44, 39]). As a consequence, we evaluate the mass flux across the interface as

$$\dot{M} = \frac{[k\nabla T \cdot \boldsymbol{n}]}{L}. \tag{7.40}$$

Phase-change problems require to account for the conservation of energy (3.6) in either phase. Neglecting the viscous dissipation (usually negligible at low Mach numbers) and assuming linear dependence between internal energy and temperature in the considered temperature range, (3.6) can be written as the following temperature equation

$$\rho C_p \frac{\mathrm{D}T}{\mathrm{D}t} = k\nabla^2 T + \dot{\Theta}. \tag{7.41}$$

In our computational framework, the temperature is sampled at the nodes of the computational grid. Prior to entering the flowchart from subsection 3.3.1 at time step n, in either subdomain we solve the following implicit discretization of (7.41)

$$\rho C_p \left.\frac{\mathrm{D}T}{\mathrm{D}t}\right|_{n+1} = k\nabla^2 T_{n+1} + \dot{\Theta} \tag{7.42}$$

where the material derivative of temperature is approximated with a second-order backward differentiation formula in association with a semi-Lagrangian method, that is

$$\begin{aligned} \left.\frac{\mathrm{D}T}{\mathrm{D}t}\right|_{n+1} = & \left(\alpha \frac{T\left(\boldsymbol{x}, t_{n+1}\right) - T\left(\boldsymbol{X}_{(\boldsymbol{x}, t_{n+1})}\left(t_n\right), t_n\right)}{\Delta t_n}\right. \\ & \left. + \beta \frac{T\left(\boldsymbol{X}_{(\boldsymbol{x}, t_{n+1})}\left(t_n\right), t_n\right) - T\left(\boldsymbol{X}_{(\boldsymbol{x}, t_{n+1})}\left(t_{n-1}\right), t_{n-1}\right)}{\Delta t_{n-1}}\right) \end{aligned} \tag{7.43}$$

at node located at $\boldsymbol{x}$, using the same notations as in subsection 7.2.1. The interpolation in space of temperature fields is done with a continuity-enforcing adaptation of the stabilized quadratic interpolation from [106] (see appendix C). The discrete equation (7.42) is solved in either subdomain with the interface Dirichlet boundary condition $T|_{\Gamma} = T_{\text{sat}}$, using the discretization from [179]. The temperature fields are then extrapolated from either subdomain to the rest of the computational domain using the methods from [172, 199], so that the mass flux can be evaluated as per (7.40) at every grid node. The interface-defined values of $\dot{M}$ are then extended from the interface using constant extrapolation, before the solver enters the flowchart described in subsection 3.3.1 accounting for this computed mass flux.

7.5.1 Two-dimensional vaporization of a drop

As a first example and in order to illustrated the sharp treatment of interface conditions, we consider the vaporization of a two-dimensional liquid drop into its vapor phase, as presented in [44]. We consider a drop of initial radius 0.02 centered in a computational domain $\Omega = [-0.035,\, 0.035]^2$. The liquid (inside the drop) properties are $\rho^- = 200$, $\mu^- = 0.1$, $k^- = 40$ and $C_p^- = 400$ while the vapor phase material parameters are $\rho^+ = 5$, $\mu^+ = 0.005$, $k^+ = 1$ and $C_p^+ = 200$; the surface tension between the two fluids is set to $\gamma = 0.1$ and the latent heat of vaporization is $L = [h] = 10^3$. (All quantities are expressed in SI units).

We consider the vaporization of such a droplet in a super-heated vapor environment: the walls of the computational domain are set to be kept at a prescribed temperature $T_{\text{wall}} = T_{\text{sat}} + 10$ generating a heat flux toward the droplet which, in turn, induces a nonzero evaporation rate $\dot{M} \neq 0$. A prescribed pressure value (Dirichlet boundary condition) is set on the walls of the computational domain along with a homogeneous Neumann boundary condition for the velocity components (mimicking a free inflow/outflow condition). We consider a quadtree grid of minimum level $\ell_{\min} = 5$ and maximum level $\ell_{\max} = 7$ with $k_{\max} = 3$, a threshold value of 0.1% for (3.22), while limiting the number of xGFM iterations per viscous step to 10; the simulations is run until $t = 1$. The vorticity-based refinement criterion is used with $P = 0.02$ in (7.19) and $b^{-} = b^{+} = 5$.

In several ways, this problem resembles the third benchmark test considered in subsection 7.3.1. Besides the inversion of fluid phases (we consider a drop not a bubble), the main difference stands in the boundary conditions that are not manufactured to enforce rotational symmetry, in this case. Nonetheless, the sharp treatment of all interface condition still produces negligible velocities everywhere in the liquid domain (as expected), as illustrated in Figure 7.14.

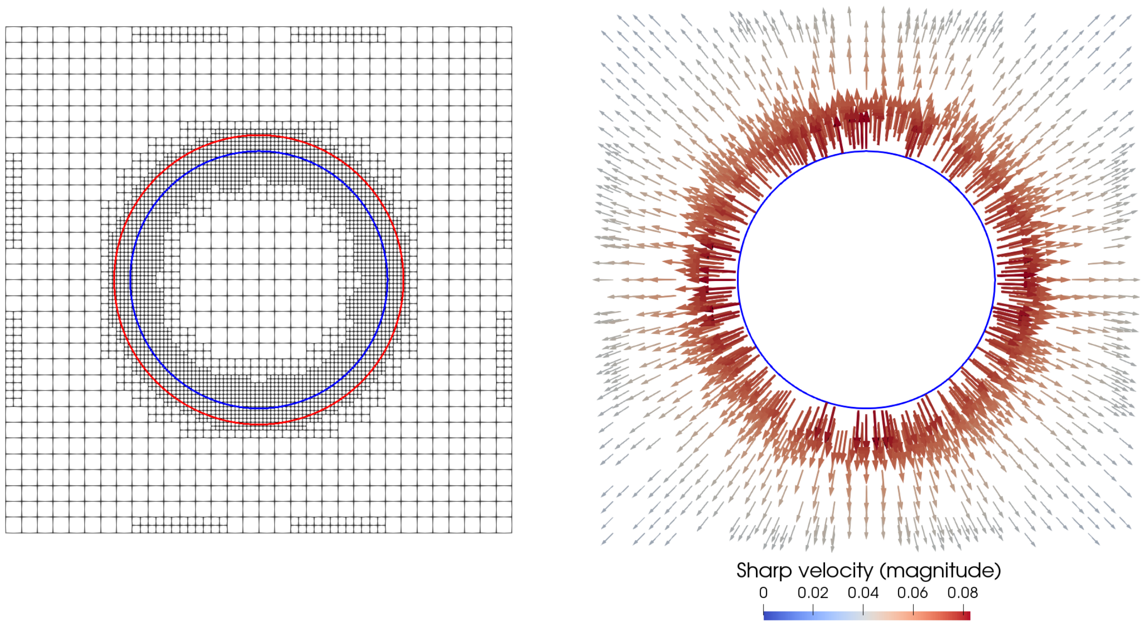


Figure 7.14: drop evaporation problem from subsection 7.5.1. Left: illustration of the computational grid at final simulation time along with the final (resp. initial) interface in blue (resp. in red). Right: computed sharp velocity field at final simulation time (displayed inside as well, though not distinguishable).

7.5.2 Thin-film boiling in two dimensions

We now consider the numerical simulation of thin film boiling, as presented in [40] and [44]. All quantities are expressed in SI units, hereafter. We consider a thin vapor layer of material properties $\rho^- = 5$, $\mu^- = 0.005$, $k^- = 1$, $C_p^- = 200$ created by a hot wall maintained at temperature $T_{\text{wall}} = T_{\text{sat}} + 5$, submersed in a bulk of liquid of material properties $\rho^+ = 200$, $\mu^+ = 0.1$, $k^+ = 40$ and $C_p^+ = 400$; the surface tension between the two fluids is $\gamma = 0.1$ and the latent heat of vaporization is $L = -10^4$ ($L < 0$ in this case since Ω^+ corresponds to the liquid phase). Gravity is the only bulk force acting on the system, i.e., $\boldsymbol{f} = -g\boldsymbol{e}_y$ where $g = 9.81$.

In the absence of a hot wall, the vapor layer would progressively make its way upwards by virtue of buoyancy, triggered by a Rayleigh-Taylor instability. However, the hot bottom wall is responsible for a heat flux toward the liquid domain, which itself generates an evaporation rate. The closer to the hot wall the interface is the larger the heat flux and the local evaporation rate: as the Rayleigh-Taylor instability develops, a non-uniform evaporation rate takes place across the interface and is responsible for nontrivial interface kinematics as well as for maintaining a vapor film.

We consider the inception of a single bubble in two dimensions: in a x-periodic computational domain $\Omega = [-\lambda_{2\text{D}}/2,\, \lambda_{2\text{D}}/2] \times [0,\, \lambda_{2\text{D}}]$ where $\lambda_{2\text{D}} = 2\pi\sqrt{3\gamma/\left(g\left(\rho^+ - \rho^-\right)\right)}$ is the most unstable wavelength (see [200]), we initialize the level function as

$$\phi\left(x, y, t = 0\right) = y - \left(4 + \cos\left(\frac{2\pi x}{\lambda_{2\text{D}}}\right)\right)\frac{\lambda_{2\text{D}}}{128}. \tag{7.44}$$

The temperature field in the vapor phase is initialized as a linear variation from T_{wall} from the wall to T_{sat} at the interface. The boundary conditions are $T = T_{\text{wall}}$, with no-slip boundary conditions on the bottom wall while a prescribed pressure is set at the top wall along with homogeneous Neumann boundary conditions on temperature and fluid velocity (free outflow). The simulation is run until $t = 0.5$ with $k_{\mathrm{max}} = 3$, a threshold value of 1% for (3.22), while limiting the number of xGFM iterations per viscous step to 10. We consider quadtree grids with $\ell_{\mathrm{max}} - \ell_{\mathrm{min}} = 2$, the vorticity-based refinement criterion is used with $P = 0.02$ in (7.19) and $b^{-} = b^{+} = 5$.

The results at time $t = 0.425\,\mathrm{s}$ are illustrated in Figure 7.15 for a quadtree of resolution $\ell_{\mathrm{min}}/\ell_{\mathrm{max}} = 6/8$. As expected, a nearly constant pressure is observed inside the bubble being formed while a quasi hydro-static pressure profile is found in the liquid domain. For interpretation and explanation purposes, let us assume that the difference in pressure across the interface is mostly balanced by surface tension effects. Away from the bottom wall (in the bubble being formed), the local interface curvature adjusts for balancing the nearly constant pressure inside with the quasi hydro-static pressure outside the bubble. However, in thin film regions away from the bubble being formed, the interface is almost flat leading to a less significant difference in pressure across the interface. Therefore, the pressure is found larger in the thin film, resulting in a significant pressure gradient in the gas phase pushing the vapor being produced in those regions towards the bubble being formed.

In Figure 7.16, the interfaces at time $t = 0.425\,\mathrm{s}$ as predicted on four quadtree grids of increasing resolution are superimposed. As illustrated in this figure, we observe convergence for the interface location under grid refinement: noticeably, the required time for the bubble to separate from the thin film is delayed as the grid is refined. In this case (Figure 7.16), the bubble has already detached on the coarsest grid while this stage has not been reached yet on finer grids. The differences in interface location observed between the finer computational grids are of the order of the size of the computational cells. For this particular problem, we have $\Delta t = 5.4\Delta t_{\mathrm{visc}}$ for the finest considered grid ($\ell_{\mathrm{min}}/\ell_{\mathrm{max}} = 7/9$, i.e., equivalent finest uniform resolution of 512^2).

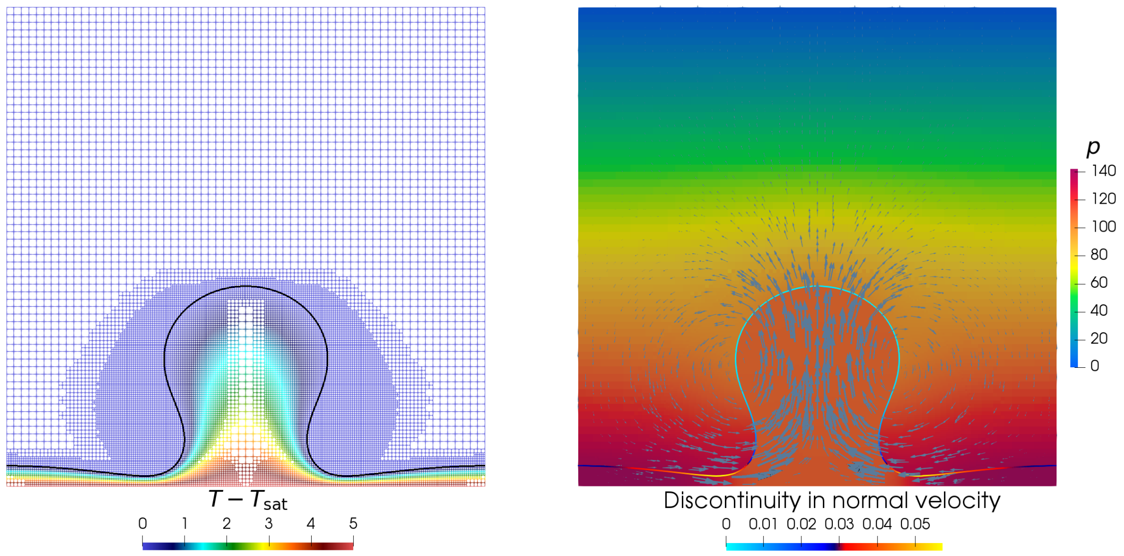


Figure 7.15: illustration of the computational results obtained on a 6/8 quadtree grid at time $t = 0.425\,\text{s}$. Left: illustration of the interface along with the computational grid, colored by $T - T_{\text{sat}}$. Right: illustration of the pressure and velocity field, the interface is colored with respect to $[\boldsymbol{u} \cdot \boldsymbol{n}] = \dot{M}\,[1/\rho]$ (corresponding to the bottom color legend). One observes a sharp treatment for the pressure terms and $[\boldsymbol{u} \cdot \boldsymbol{n}]$ is larger for interface points that are close to the bottom wall (as expected).

7.5.3 Thin-film boiling in three dimensions

As a final illustration of the capabilities of the developed solver, we consider the simulation of of the same problem as in subsection 7.5.2 but in three dimensions. We consider the same material parameters for both fluid phases and we bring some modifications and adaptations to the simulation setup. The x- and z-periodic computational domain is $\Omega = [-\lambda_{3\text{D}}/2,\, \lambda_{3\text{D}}/2] \times [0,\, 3\lambda_{3\text{D}}] \times [-\lambda_{3\text{D}}/2,\, \lambda_{3\text{D}}/2]$ where $\lambda_{3\text{D}} = 2\pi\sqrt{6\gamma/\left(g\left(\rho^+ - \rho^-\right)\right)} = \sqrt{2}\lambda_{2\text{D}}$ (see [38, 201]) and we initialize the level function as

$$\phi\left(x, y, t = 0\right) = y - \left(5 + \frac{1}{4}\left(1 + \cos\left(\frac{2\pi x}{\lambda_{3\text{D}}}\right)\right)\left(1 + \cos\left(\frac{2\pi z}{\lambda_{3\text{D}}}\right)\right)\right)\frac{\lambda}{128}. \qquad (7.45)$$

The bottom wall is kept at temperature $T_{\text{wall}} = T_{\text{sat}} + 10$ and the initial temperature field in the vapor phase is set as a linear variation from T_{wall} from the wall to T_{sat} at the

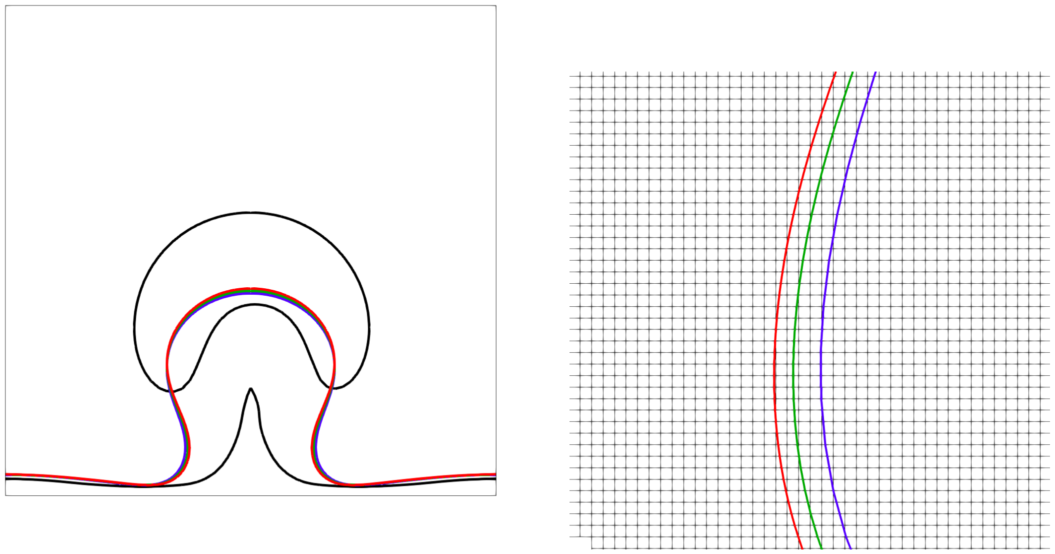


Figure 7.16: interface location at time $t = 0.425\,\mathrm{s}$, as simulated numerically with quadtree grids of resolution $\ell_{\min}/\ell_{\max} = 4/6$ (black), $\ell_{\min}/\ell_{\max} = 5/7$ (blue), $\ell_{\min}/\ell_{\max} = 6/8$ (green) and $\ell_{\min}/\ell_{\max} = 7/9$ (red). Right: zoom-in onto the lower right re-entrant portion of the bubble interface, with the finest computational grid in the background.

interface. The boundary conditions are $T = T_{\text{wall}}$, with no-slip boundary conditions on the bottom wall while a prescribed pressure is set at the top wall along with homogeneous Neumann boundary conditions on temperature and fluid velocity (free outflow). The simulation is run with $k_{\max} = 3$, a threshold value of 1% for (3.22), while limiting the number of xGFM iterations per viscous step to 10. We use linear interpolation instead of the continuity-enforcing stabilized quadratic interpolation for the levelset function in this three-dimensional case. The computational domain is meshed with three octrees (to ensure unit aspect ratio for the computational cells) of minimum level $\ell_{\min} = 5$ and maximum level $\ell_{\max} = 8$, the vorticity-based refinement criterion is used with $P = 0.02$ in (7.19) and $b^- = b^+ = 5$. The simulation was run until two bubbles were fully created and separated from the vapor film, corresponding a final simulation time of $t \simeq 0.83\,\mathrm{s}$ (corresponding to approximately 6250 time steps). The simulation was run on 160 cores on 8 Intel 6148 CPUs on Pod cluster (center for scientific computing, UCSB) for about

10 days. As the simulation evolved, the total number of computational cells grew from around 10^6 to 15×10^6 (a uniform grid of equivalent finest resolution would contain 50×10^6 computational cells).

We underline that this simulation created under-resolved interface configurations, at the moment of bubble separation from the thin film, which would theoretically require special, additional care for identification and robust numerical treatment, in particular regarding the calculation of curvature and normal vectors (see the note at the end of subsection 3.2.1). Nonetheless, in this particular case, the simulation engine could tackle such under-resolved geometries[10]. As for the two-dimensional case, we illustrate the results at time $t = 0.34\,\mathrm{s}$ in Figure 7.17: the same observations and interpretations as in the two-dimensional case hold. The dynamics is illustrated along with cross sections of the computational grid colored by $T - T_{\mathrm{sat}}$ in Figure 7.18.

[10]We also point out that the use of a linear interpolation for the levelset function defaults the interface reconstruction procedure to build linear elements, in cells crossed by the interface (as in chapter 6). This was found to be significantly more stable and robust than quadratic interface reconstructions.

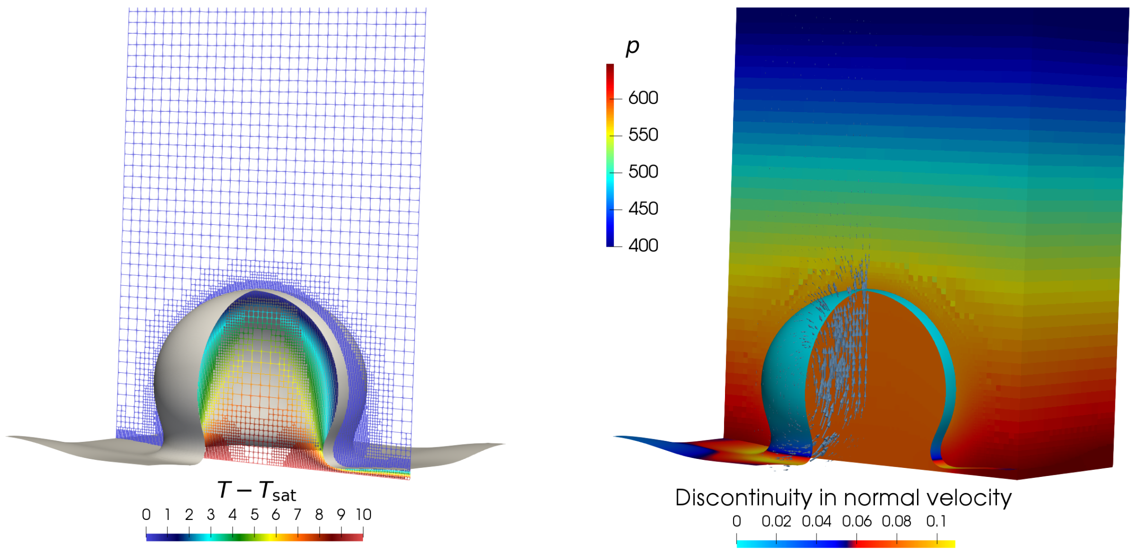


Figure 7.17: illustration of the computational results obtained on a forest of three 5/8 octree grids at time $t = 0.34\,\mathrm{s}$ (zoom-in close to the bottom wall and to the interface). Left: illustration of the (truncated) interface along with the computational grid, colored by $T - T_{\mathrm{sat}}$. Right: illustration of the pressure and velocity field, the interface is colored with respect to $[\boldsymbol{u} \cdot \boldsymbol{n}] = \dot{M}\,[1/\rho]$ (corresponding to the bottom color legend). One observes a sharp treatment for the pressure terms and $[\boldsymbol{u} \cdot \boldsymbol{n}]$ is larger for interface points that are close to the bottom wall (as expected).

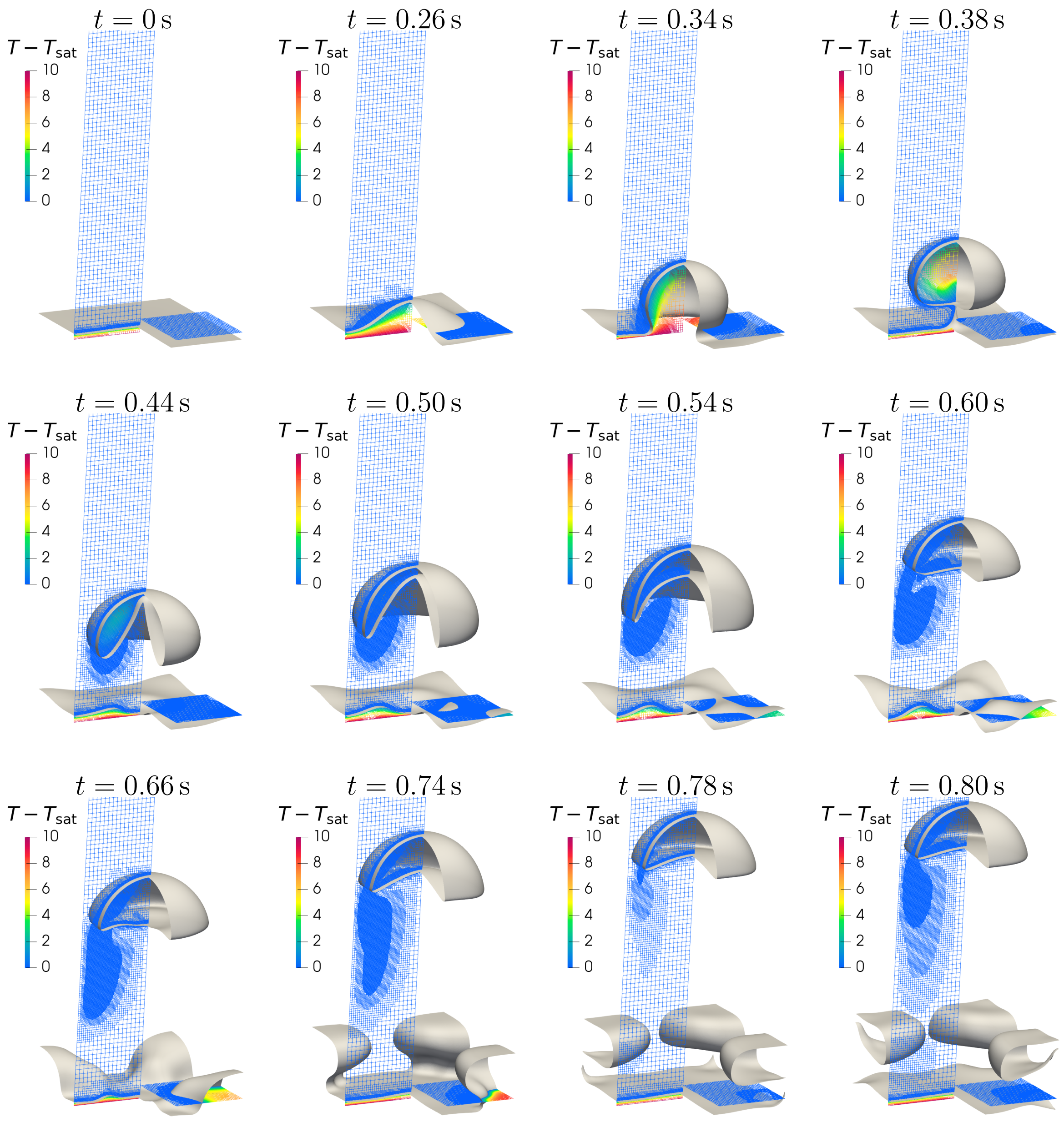


Figure 7.18: illustration of the (truncated) interface dynamics along with (slices of) the computational grid colored by $T - T_{\mathrm{sat}}$ for the problem from subsection 7.5.3.

Conclusion

We have presented an original numerical approach for simulating incompressible, viscous two-phase flows. The strategy relies on a fully sharp treatment of all material properties and primary unknowns, enabling discontinuities not only in pressure but also in the normal velocity component (as required for phase-change problems). The viscous terms are treated fully implicitly and in a sharp fashion, building upon a generalization of the treatment of viscous terms from Kang's work [15] to problems that may involve a mass transfer across the interface. Such an approach leverages on a numerical discretization that consistently translates the governing equations from continuum-mechanics in the limit sense of zero-thickness interfaces, which is the only way to observe absolute convergence for such problems. To the best of our knowledge, this numerical approach is innovative and stands as the first attempt in treating sharp interface conditions fully implicitly in presence of mass flux across the interface, in dynamical two-phase fluid simulations.

Building upon the computational framework originally developed for single-phase problems, all primary unknowns (pressure and velocity components) are decoupled: while this produces smaller and better-behaved linear systems of equations, it also prevents the strict consideration of all interface conditions in a single pass and one requires to iteratively approach it in this case. A nested iterative approach is presented to account for (i) the balance of (tangential) stress across the interface (ii) the interdependence between pressure and viscous terms in the balance of normal stress across the interface. This nested approach aims at approaching a saddle-point, block discretization theoretically.

However, its computational cost grows fairly quickly as every elementary subiteration comes with extension and extrapolation of face-sampled velocity fields as well as the inversion of corresponding linear systems. A formal termination criterion would need to be determined to keep the overall number of iterations minimal: in our tests we observed that truncating the iterative procedure to only a few iterations produced erroneous (numerically unstable) results only in case of very large time steps, compared to an explicit treatment of viscous terms. Nevertheless, such a nested iterative approach makes three-dimensional simulation runs very expensive computationally and an explicit treatment of viscous terms exploiting (3.13) might be preferable for practical applications where implicit time stepping comes with a limited (or no) gain in time step. An explicit treatment of viscous terms would also be less sensitive to under-resolved geometries compared to its implicit counterpart.

The lack of strict volume conservation of the current levelset-based method is another drawback that would need to be alleviated to enable long simulation runs at limited computational cost. Well-established numerical strategies may be considered to that purpose like coupling levelset and Volume-Of-Fluid approaches (as in [22]) although the exponentially growing interest in machine and deep learning techniques may bring an elegant solution to this issue in the near future.

List of symbolic operations and notations

ESC Einstein Summation Convention: when an index appears twice in an expression, it implies summation over all the values of the index. Example: $a_i b_i$ actually represents $\sum_i a_i b_i$.

$(\boldsymbol{e}_1,\, \boldsymbol{e}_2,\, \boldsymbol{e}_3)$ right-handed orthonormal basis.

$\boldsymbol{v}$ vector of components v_i $(i = 1, 2, 3)$, $\boldsymbol{v} = v_i \boldsymbol{e}_i$, using ESC.

$\boldsymbol{a} \cdot \boldsymbol{b}$ inner product between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\boldsymbol{a} \cdot \boldsymbol{b} = a_i b_i$, using ESC.

$\boldsymbol{a} \times \boldsymbol{b}$ cross product between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, the i^{th} component of $\boldsymbol{a} \times \boldsymbol{b}$ is given by $\varepsilon_{ijk} a_j b_k$, using ESC, where ε_{ijk} is the Levi-Civita symbol.

$\boldsymbol{ab}$ dyadic product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, also sometimes represented by $\boldsymbol{ab}^{\mathrm{T}}$ or $\boldsymbol{a} \otimes \boldsymbol{b}$. The result of this operation is the second-order tensor of rank one $a_i b_j \boldsymbol{e}_i \boldsymbol{e}_j$, using ESC.

ε_{ijk} Levi-Civita symbol: ε_{ijk} is 1 if (i, j, k) is an even permutation of $(1, 2, 3)$, -1 if (i, j, k) is an odd permutation of $(1, 2, 3)$, and 0 if any index is repeated.

$\nabla \boldsymbol{a}$ gradient of the vector $\boldsymbol{a}$: $\nabla \boldsymbol{a}$ is the second-order tensor $\dfrac{\partial a_i}{\partial x_j} \boldsymbol{e}_i \boldsymbol{e}_j$ using ESC.

$\underline{\boldsymbol{R}}$ second-order tensor of components R_{ij}, $\underline{\boldsymbol{R}} = R_{ij}\boldsymbol{e}_i\boldsymbol{e}_j$, using ESC.

$\underline{\boldsymbol{R}}^{\mathrm{T}}$ transpose of the second-order tensor $\underline{\boldsymbol{R}}$. If $\underline{\boldsymbol{R}} = R_{ij}\boldsymbol{e}_i\boldsymbol{e}_j$, then $\underline{\boldsymbol{R}}^{\mathrm{T}} = R_{ji}\boldsymbol{e}_i\boldsymbol{e}_j$, using ESC.

$\boldsymbol{a} \cdot \underline{\boldsymbol{R}}$ left inner product between a vector $\boldsymbol{a}$ and a second-order tensor $\underline{\boldsymbol{R}}$, the result is $a_k R_{ki}\boldsymbol{e}_i$, using ESC.

$\underline{\boldsymbol{R}} \cdot \boldsymbol{a}$ right inner product between a second-order tensor $\underline{\boldsymbol{R}}$ and a vector $\boldsymbol{a}$, the result is $R_{ij}a_j\boldsymbol{e}_i$, using ESC.

$\boldsymbol{a} \cdot \underline{\boldsymbol{R}} \cdot \boldsymbol{b}$ bilinear form of vector arguments $\boldsymbol{a}$ and $\boldsymbol{b}$ built on the second-order tensor $\underline{\boldsymbol{R}}$, the result is $a_i R_{ij} b_j$, using ESC. Note that $\boldsymbol{a} \cdot \underline{\boldsymbol{R}} \cdot \boldsymbol{b} = (\boldsymbol{a} \cdot \underline{\boldsymbol{R}}) \cdot \boldsymbol{b} = \boldsymbol{b} \cdot (\boldsymbol{a} \cdot \underline{\boldsymbol{R}}) = \boldsymbol{a} \cdot (\underline{\boldsymbol{R}} \cdot \boldsymbol{b}) = (\underline{\boldsymbol{R}} \cdot \boldsymbol{b}) \cdot \boldsymbol{a}$.

$\underline{\boldsymbol{R}} \cdot \underline{\boldsymbol{R}}$ simple inner product between two second-order tensors $\underline{\boldsymbol{R}}$ and $\underline{\boldsymbol{R}}$, the result is the second-order tensor $R_{ij}R_{jk}\boldsymbol{e}_i\boldsymbol{e}_k$, using ESC.

$\underline{\boldsymbol{R}} \colon \underline{\boldsymbol{R}}$ double inner product between two second-order tensors $\underline{\boldsymbol{R}}$ and $\underline{\boldsymbol{R}}$, we define the scalar result as $R_{ij}R_{ij}$, using ESC. Note that $\underline{\boldsymbol{R}} \colon \boldsymbol{ab} = \boldsymbol{a} \cdot \underline{\boldsymbol{R}} \cdot \boldsymbol{b}$.

$\nabla \cdot \underline{\boldsymbol{R}}$ divergence of the second-order tensors $\underline{\boldsymbol{R}}$, the i^{th} component of $\nabla \cdot \underline{\boldsymbol{R}}$ is $\dfrac{\partial R_{ik}}{\partial x_k}$, using ESC.

κ total curvature (twice the mean), defined as $\kappa = \nabla \cdot \boldsymbol{n}$, where $\boldsymbol{n}$ is the unit normal vector to the liquid-gas interface.

h enthalpy per unit mass, $h = e + \dfrac{p}{\rho}$ where e is the internal energy per unit mass, p is the thermodynamic pressure and ρ is the mass density.

λ expansion viscosity.

C_p heat capacity per unit mass.

Φ Hodge variable.

ψ interfacial energy per unit area.

e internal energy per unit mass.

ν kinematic viscosity, $\nu = \mu/\rho$ where μ is the shear viscosity and ρ is the mass density.

ϕ levelset function.

ρ mass density.

μ shear viscosity.

v specific volume, $v = \rho^{-1}$ where ρ is the mass density.

γ surface tension coefficient.

T temperature.

k thermal conductivity.

p thermodynamic pressure.

$\dot{\Theta}$ volumetric heat source.

$\mathcal{E}$ total energy per unit mass, $\mathcal{E} = e + \dfrac{\|\boldsymbol{u}\|^2}{2}$ where e is the internal energy per unit mass and $\boldsymbol{u}$ is the fluid velocity.

$\boldsymbol{f}$ body force per unit mass.

$\boldsymbol{u}$ fluid velocity.

$\boldsymbol{q}$ heat flux.

$\boldsymbol{n}$ unit normal vector to $\Sigma(t)$, where $\Sigma(t)$ is a portion of the liquid-gas interface.

$\boldsymbol{\tau}$ unit vector tangent to $\partial\Sigma(t)$, where $\Sigma(t)$ is a portion of the liquid-gas interface. The orientation of $\boldsymbol{\tau}$ is determined from $\boldsymbol{n}$ to ensure the consistency when applying Stokes' theorem.

$\boldsymbol{m}$ unit vector, outward normal to $\partial\Sigma(t)$, tangent to $\Sigma(t)$. We have $\boldsymbol{m} = \boldsymbol{\tau} \times \boldsymbol{n}$.

$\boldsymbol{w}$ velocity of the liquid-gas interface.

$\boldsymbol{\eta}$ outer unit normal vector to any considered material volume.

$\boldsymbol{G}$ interface-defined singular force.

$\underline{\boldsymbol{\delta}}$ identity second-order tensor, $\underline{\boldsymbol{\delta}} = \delta_{ij}\boldsymbol{e}_i\boldsymbol{e}_j$, using ESC, and where δ_{ij} is the Kronecker delta: $\delta_{ij} = 1$ if $i = j$, 0 otherwise.

$\underline{\boldsymbol{E}}$ strain rate tensor, $\underline{\boldsymbol{E}} = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{\mathrm{T}}\right) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\boldsymbol{e}_i\boldsymbol{e}_j$, using ESC.

$\underline{\boldsymbol{\sigma}}$ stress tensor. For Newtonian fluids,

$$\underline{\boldsymbol{\sigma}} = -p\underline{\boldsymbol{\delta}} + \left(\lambda - \frac{2\mu}{3}\right)(\nabla\cdot\boldsymbol{u})\,\underline{\boldsymbol{\delta}} + 2\mu\underline{\boldsymbol{E}},$$

where p is the thermodynamic pressure, $\underline{\boldsymbol{\delta}}$ is the identity tensor, λ is the expansion viscosity, μ is the shear viscosity, $\boldsymbol{u}$ is the fluid velocity and $\underline{\boldsymbol{E}}$ is the strain rate tensor.

$\Omega_{\Sigma(t),\,\epsilon}$ volume of thickness 2ϵ extruded symmetrically on both sides of the arbitrary portion of liquid-gas interface $\Sigma(t)$. Mathematically, $\Omega_{\Sigma(t),\,\epsilon} = \Omega_{\Sigma(t),\,\epsilon^-} \bigcup \Omega_{\Sigma(t),\,\epsilon^+}$.

$\Omega_{\Sigma(t),\,\epsilon^-}$ volume of thickness ϵ extruded on one side of the arbitrary portion of liquid-gas interface $\Sigma(t)$ (the side pointed by the negative interface normal vector). Mathematically,

$$\Omega_{\Sigma(t),\,\epsilon^-} = \{\boldsymbol{z} \,|\, \boldsymbol{z} = \boldsymbol{s} + \ell\boldsymbol{n},\ \boldsymbol{s} \in \Sigma(t)\,,\ -\epsilon \leq \ell \leq 0\}\,,$$

where $\boldsymbol{n}$ is a unit normal vector to the liquid-gas interface.

$\Omega_{\Sigma(t),\,\epsilon^+}$ volume of thickness ϵ extruded on one side of the arbitrary portion of liquid-gas interface $\Sigma(t)$ (the side pointed by the interface normal vector). Mathematically,

$$\Omega_{\Sigma(t),\,\epsilon^+} = \{\boldsymbol{z} \,|\, \boldsymbol{z} = \boldsymbol{s} + \ell\boldsymbol{n},\ \boldsymbol{s} \in \Sigma(t)\,,\ 0 \leq \ell \leq \epsilon\}\,,$$

where $\boldsymbol{n}$ is a unit normal vector to the liquid-gas interface.

$\dot{M}$ net mass flux through the interface, $\dot{M} = \rho(\boldsymbol{u} - \boldsymbol{w}) \cdot \boldsymbol{n}$ where $\boldsymbol{n}$ is the unit normal vector to the liquid-gas interface and $\boldsymbol{w}$ is the velocity of the liquid-gas interface.

$\Sigma(t)$ arbitrary portion of the liquid-gas interface at time t.

ϵ half the thickness of the material volume across the interface on both sides of the interface.

Part II

Other completed projects

Chapter 8

Geometric discretization of the multidimensional Dirac delta distribution - Application to the Poisson equation with singular source terms

8.1 Introduction

The Poisson equation appears in many problems ranging from diffusion-dominated phenomena to incompressible flows. Because of this ubiquitous prominence, substantial efforts have been devoted over the past decades to developing efficient and accurate numerical methods for its solution. As a result, several advances have been made in the context of irregular domains and free boundaries to impose Dirichlet boundary conditions [170, 202, 169], discontinuous coefficient and/or discontinuous solutions across an interface [146, 15, 17, 106, 5, 140, 203], and Robin boundary conditions [204, 205, 206, 64]. Concurrently, improvements have been made for the discretization of singular source terms like Dirac delta function[1] supported on a level set, as they are needed to evaluate surface tension forces for instance [146, 207, 5, 151, 208, 3].

However, very little has been proposed for the discretization of point-located Dirac delta source terms in equations like

$$-\nabla \cdot (\beta \nabla u) = \sum_i \delta \left(\boldsymbol{x} - \boldsymbol{y}_i \right), \tag{8.1}$$

in a (possibly irregular) domain Ω ($\boldsymbol{y}_i \in \Omega$), with given boundary conditions at its boundary $\partial\Omega$ (here β is bounded from below by a positive constant). Two typical examples are stresslet calculations using fundamental solutions of the creeping flow equations [209, 210, 211], and the calculation of the total electrostatic potential due to the nuclei and the electron charge distribution in the context of Kohn-Sham density functional theory [212, 213].

Given a function f defined in some domain $\Omega \subset \mathbb{R}^d$, the d-dimensional Dirac distribution δ is defined by

$$\int_\Omega f\left(\boldsymbol{x}\right) \delta \left(\boldsymbol{x} - \boldsymbol{y}\right) \mathrm{d}\boldsymbol{x} = f\left(\boldsymbol{y}\right), \quad \forall \boldsymbol{y} \in \Omega. \tag{8.2}$$

This mathematical entity has some profound theoretical importance since any function can be written as the convolution between itself and that distribution. Therefore, in the context of a linear PDE in $\mathbb{R}^d$, *i.e.* $\mathcal{L}\left[u\right]\left(\boldsymbol{x}\right) = g\left(\boldsymbol{x}\right)$, $\boldsymbol{x} \in \mathbb{R}^d$, u and g being functions from $\mathbb{R}^d$ to $\mathbb{R}$ and $\mathcal{L}$ being a linear (integro-)differential operator (with constant coefficients),

[1]Although the Dirac delta δ has to be understood in the sense of distribution or as a linear functional, *i.e.* if δ acts on a function f, δ belongs to the space that is dual to the function space of f, we may misuse the word "function" hereafter by referring to δ as the "Dirac delta function".

if one knows the solution to such a right-hand side as $\delta(\boldsymbol{x})$, say $h(\boldsymbol{x})$ (referred to as the *fundamental solution*), the solution to any general input $g(\boldsymbol{x})$ can be expressed as the convolution between g and h, *i.e.*

$$u(\boldsymbol{x}) = \int_{\mathbb{R}^d} g(\boldsymbol{\xi})\, h(\boldsymbol{x}-\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} \tag{8.3}$$

when that latter integral exists.

Motivated by the computation of the semiclassical limit for Schrödinger equations and high-frequency geometrical optics, several approaches have been suggested over the last decade for the calculation of integrals like or similar to (8.2) when the source location $\boldsymbol{y}$ is defined as the (non-degenerate) intersection of d codimension one surfaces in d dimensions [214, 215, 216, 217]. The most stringent challenge in those problems arises from the lack of consistency associated with naive approaches when the codimension one surfaces are not aligned with the grid axes or, in the worst case scenario, when the surfaces are close to degenerate (*e.g.* almost parallel). When the source location $\boldsymbol{y}$ is *known*, these techniques can be used very simply by defining d codimension one surfaces that are parallel to the grid axes and whose intersection is $\boldsymbol{y}$ to compute integrals like (8.2) accurately. However, the corresponding result and the discretized form of δ that one may obtain based on such an approach *do* depend on the choice of the d codimension one surfaces more generally, hence a lack of geometric robustness.

In this chapter, we present a geometric approach for calculating integrals like (8.2) by discretizing the domain into a disjoint union of simplices. The approach leads to second-order accurate results. An explicit discretization for the Dirac d-dimensional distribution ($d = 2$ or 3) is then derived by comparing this approach to the standard second-order approximate integral for scalar field on uniform grids. This discretization is consistent (in the discrete sense) with the method for Dirac distribution on codimension one surfaces that is presented in [152, 152]. Finally, we use this approximation of the Dirac distribution to solve a Poisson equation with point-distributed singular source terms. Results for such problems with possibly many such source terms are then presented in two and three dimensions. Superlinear convergence of the solution *and its gradient* is observed and illustrated; based on our numerical experiments, the order of accuracy of the solution and of its gradient is either 2 or slightly smaller. Supra-convergence for the gradient of the solution is desirable for the calculation of electric fields and electrostatic energy for instance.

8.2 Numerical integration

In this section we present a numerical approximation to $\int_\Omega f\left(\boldsymbol{x}\right)\delta\left(\boldsymbol{x}-\boldsymbol{y}\right)\mathrm{d}\boldsymbol{x}$. This approximation builds on second-order accurate linear interpolation in a simplex, which is first presented. The method is then developed for a general domain that we decompose into a disjoint union of simplices. The following applies equally in the context of uniform or Quadtree/Octree grids by virtue of the local nature of the approach. Since the approach is simplex-based, triangulated meshes could also use the formula in a straightforward way.

8.2.1 Second-order interpolation in a simplex

Consider a point $\boldsymbol{y} \in \Omega \subset \mathbb{R}^d$ contained in a d-simplex $\mathcal{S}$ defined by $(d+1)$ affinely independent points $\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_d$ (see Figure 8.1). Let f be a twice differentiable function in Ω and f_i the sampled value of f at point $\boldsymbol{z}_i$, $i = 0, \ldots, d$. Then, the Taylor expansions of f around $\boldsymbol{y}$ gives

$$f_i = f(\boldsymbol{y}) + \sum_{k=1}^{d} \left.\frac{\partial f}{\partial x_k}\right|_{\boldsymbol{y}} (\boldsymbol{z}_i - \boldsymbol{y}) \cdot \boldsymbol{e}_k + \mathcal{O}\left(\Delta^2\right) \tag{8.4}$$

where $\boldsymbol{e}_i$ represents the ith canonical unit basis vector and $\Delta = \max_i \|\boldsymbol{z}_i - \boldsymbol{y}\|$. Hence, the linear combination $\sum_{i=0}^{d} \alpha_i f_i$ is a second-order accurate approximation of $f(\boldsymbol{y})$ iff

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \boldsymbol{e}_1 \cdot (\boldsymbol{z}_0 - \boldsymbol{y}) & \boldsymbol{e}_1 \cdot (\boldsymbol{z}_1 - \boldsymbol{y}) & \cdots & \boldsymbol{e}_1 \cdot (\boldsymbol{z}_d - \boldsymbol{y}) \\ \boldsymbol{e}_2 \cdot (\boldsymbol{z}_0 - \boldsymbol{y}) & \boldsymbol{e}_2 \cdot (\boldsymbol{z}_1 - \boldsymbol{y}) & \cdots & \boldsymbol{e}_2 \cdot (\boldsymbol{z}_d - \boldsymbol{y}) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{e}_d \cdot (\boldsymbol{z}_0 - \boldsymbol{y}) & \boldsymbol{e}_d \cdot (\boldsymbol{z}_1 - \boldsymbol{y}) & \cdots & \boldsymbol{e}_d \cdot (\boldsymbol{z}_d - \boldsymbol{y}) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_d \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

from which we deduce

$$\alpha_i = \frac{\operatorname{vol}\left(\mathcal{S}_{\boldsymbol{z}_i \leftrightarrow \boldsymbol{y}}\right)}{\operatorname{vol}(\mathcal{S})}, \quad i = 0, \ldots, d \tag{8.5}$$

wherein $\operatorname{vol}(\mathcal{S})$ denotes the area (resp. the volume) of the simplex $\mathcal{S}$ in 2D (resp. 3D) and $\mathcal{S}_{\boldsymbol{p} \leftrightarrow \boldsymbol{q}}$ denotes the new simplex created from $\mathcal{S}$ by replacing its point $\boldsymbol{p}$ by a new point $\boldsymbol{q}$.

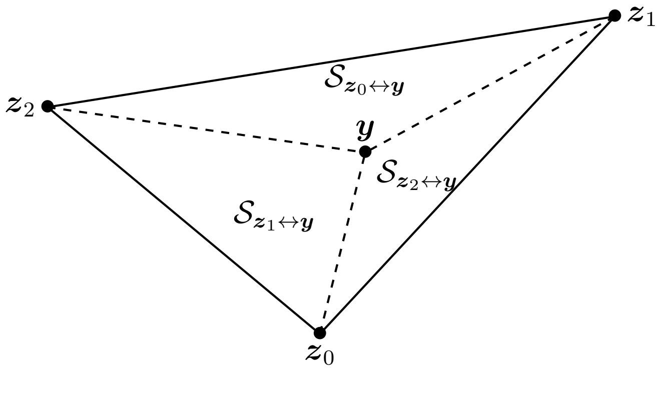

$\boldsymbol{z}_1$
$\mathcal{S}_{\boldsymbol{z}_0 \leftrightarrow \boldsymbol{y}}$
$\boldsymbol{z}_2$
$\boldsymbol{y}$
$\mathcal{S}_{\boldsymbol{z}_2 \leftrightarrow \boldsymbol{y}}$
$\mathcal{S}_{\boldsymbol{z}_1 \leftrightarrow \boldsymbol{y}}$
$\boldsymbol{z}_0$

Figure 8.1: a 2-simplex $\mathcal{S}$ containing $\boldsymbol{y}$ and its inner simplices $\mathcal{S}_{\boldsymbol{z}_i \leftrightarrow \boldsymbol{y}}$.

8.2.2 Systematic approach by triangulation

Based on the results from section 8.2.1 and the definition (8.2), a second-order accurate approximation of

$$\int_{\Omega} f\left(\boldsymbol{x}\right) \delta\left(\boldsymbol{x}-\boldsymbol{y}\right) \mathrm{d}\boldsymbol{x}$$

follows right away. This approach has the advantage of being local to each simplex: it does not require values of f from another simplex. In order to build a systematic approach, even in case of non-triangulated meshes, we choose to decompose into simplices the grid cell in which $\boldsymbol{y}$ is located (if needed) and consider only the (sub)-simplex that contains $\boldsymbol{y}$ in that cell.

In the rest of this chapter, we will consider uniform and Quad-/Oc-tree grids[2] with a node-based field sampling, *i.e.* the scalar or vector field of interest is sampled at the vertices of the grid cells. In any case, the computational domain is decomposed into elementary regular d-dimensional grid cells whose edges are aligned with the Cartesian axes. Therefore, any grid cell that contains the source of a Dirac-distributed term needs to be decomposed into elementary d-simplices and only the sub-simplex containing the source is further considered to apply the above reasoning. Since we restrict ourselves to $d = 2$ or 3, we prefer the middle cut triangulation to the Kuhn triangulation. The general middle-cut triangulation is illustrated in Figure 8.2.

Denoting by $\operatorname{conv}\left(P_1, \ldots, P_n\right)$ the simplex defined by the points $P_1, \ldots, P_n$, we define

[2]In a d-dimensional space with a Cartesian orthonormal basis $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$, a *uniform grid* of seed $\boldsymbol{w}$ and (real) grid sizes $h_1, h_2, \ldots, h_d$ (along the basis vectors and based on which one defines $\boldsymbol{h} = \sum_{k=1}^{d} h_k \boldsymbol{e}_k$) is a set of uniformly spaced nodes

$$\Omega_{\boldsymbol{h}}^{\boldsymbol{w}} = \left\{\boldsymbol{z} \in \mathbb{R}^d : \boldsymbol{z} = \boldsymbol{w} + i_1 h_1 \boldsymbol{e}_1 + \ldots + i_d h_d \boldsymbol{e}_d, \left\{i_1, i_2, \ldots, i_{d-1}, i_d\right\} \in \mathcal{I} \subseteq \mathbb{Z}^d\right\}.$$

A Quad-/Oc-tree grid is a *nonuniform* two-/three-dimensional Cartesian grid, it is formally defined in section 8.5.1.

the triangulation $T(C)$ of a cell C by

$$T(C) = \{\operatorname{conv}(P_{00}, P_{10}, P_{11}), \operatorname{conv}(P_{00}, P_{01}, P_{11})\}$$

in 2D and

$$\begin{aligned} T(C) \quad = \quad & \{\operatorname{conv}(P_{000}, P_{100}, P_{010}, P_{001}), \operatorname{conv}(P_{110}, P_{100}, P_{010}, P_{111}), \operatorname{conv}(P_{101}, P_{100}, P_{111}, P_{001}), \\ & \operatorname{conv}(P_{011}, P_{111}, P_{010}, P_{001}), \operatorname{conv}(P_{111}, P_{100}, P_{010}, P_{001})\} \end{aligned}$$

in 3D. Any grid cell C is then equivalent to $\bigcup_{\mathcal{S}\in T(C)} \mathcal{S}$. Consequently, we have

$$\int_\Omega f(\boldsymbol{x})\, \delta(\boldsymbol{x}-\boldsymbol{y})\,\mathrm{d}\boldsymbol{x} = \sum_{C:\text{grid cell}} \sum_{\mathcal{S}\in T(C)} \sum_{k\in\mathcal{N}(S)} \delta_k(\mathcal{S},\boldsymbol{y})\, f_k + \mathcal{O}\left(\Delta^2\right), \tag{8.6}$$

where δ_k is defined as

$$\delta_k(\mathcal{S},\boldsymbol{y}) = \begin{cases} 0 & \text{if } \boldsymbol{y}\notin\mathcal{S}, \\ \frac{\operatorname{vol}\left(\mathcal{S}_{\boldsymbol{z}_k\leftrightarrow\boldsymbol{y}}\right)}{\operatorname{vol}(\mathcal{S})} & \text{otherwise,} \end{cases} \tag{8.7}$$

and $\mathcal{N}(S)$ is the set of indices k such that the grid node $\boldsymbol{z}_k$ is a vertex of simplex $\mathcal{S}$.

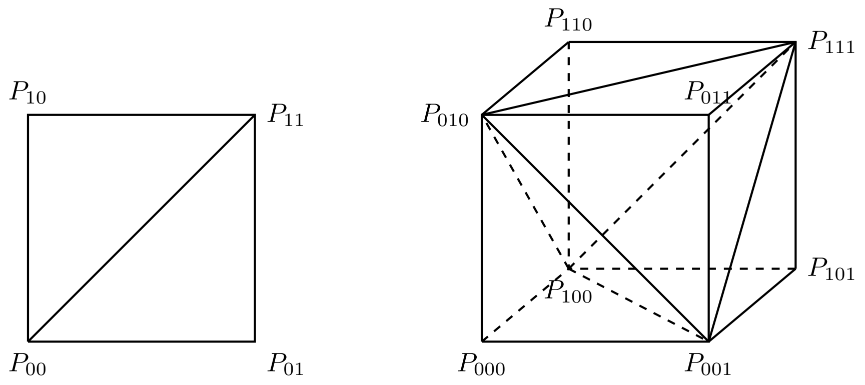


Figure 8.2: middle-cut triangulation of a two- (left) and three- (right) dimensional cell.

8.3 Integration examples

In this section, we show that the approach presented here above produces second-order accurate results to (8.2) in two and three spatial dimensions by comparing the right-hand side of (8.6) to $f(\boldsymbol{y})$.

8.3.1 Results in two spatial dimensions

We illustrate the order of accuracy by evaluating integrals of the type (8.2) where $\Omega = [-1, 1]^2$, $f(\boldsymbol{x}) = \cos(\boldsymbol{x} \cdot \boldsymbol{e}_1) \sin(\boldsymbol{x} \cdot \boldsymbol{e}_2)$ and for a point $\boldsymbol{y}$ randomly located in Ω. We use uniform grids of increasing refinement level[3]. The error is averaged over 50 trials for various random points $\boldsymbol{y}$ in Ω. The results are presented in Table 8.1 and show the second-order rate of convergence. Our results are compared with the approximations found by using a product of one-dimensional discretized delta functions from [3]. As it can be seen in Table 8.1, the error is comparable with both approaches.

Remark. *Although the approximations always converge, the rate of convergence may fluctuate around 2 in the case of one single trial. However, second-order accuracy is obtained in the mean-sense, as illustrated in Table 8.1. This table also indicate that the minimum, the maximum and the standard deviation of the error distribution converge, which is a compelling evidence that the approximation of the integral is convergent.*

8.3.2 Results in three spatial dimensions

We repeat a similar example as in the section 8.3.1 with $\Omega = [-1, 1]^3$ and $f(\boldsymbol{x}) = \cos(\boldsymbol{x} \cdot \boldsymbol{e}_1) \sin(\boldsymbol{x} \cdot \boldsymbol{e}_2) \exp(\boldsymbol{x} \cdot \boldsymbol{e}_3)$, for a point $\boldsymbol{y}$ randomly located in Ω. We use uniform grids of increasing refinement level, the error is averaged over 50 trials. The results

[3]A refinement level p corresponds to cells of size 2^{-p} for a domain $[0, 1]$.

Statistics of 50 trials for computing an integral like (8.2) in $\Omega \subset \mathbb{R}^2$

Results using (8.6)					
Grid level	Average error	Order	SD	Min	Max
10	3.05e-07		2.03e-07	2.31e-08	8.52e-07
11	7.37e-08	2.05	5.52e-08	3.40e-09	2.07e-07
12	1.82e-08	2.02	1.19e-08	2.85e-10	4.49e-08
13	4.38e-09	2.05	3.40e-09	1.46e-10	1.33e-08
14	1.09e-09	2.01	8.06e-10	6.58e-11	3.43e-09
15	2.71e-10	2.01	1.97e-10	2.11e-11	8.69e-10
Results using a product of one-dimensional delta functions from [3]					
10	7.74e-07		4.52e-07	1.72e-08	1.67e-06
11	1.93e-07	2.00	1.12e-07	4.38e-09	4.24e-07
12	4.81e-08	2.00	2.80e-08	1.08e-09	1.04e-07
13	1.20e-08	2.00	7.03e-09	2.73e-10	2.68e-08
14	3.00e-09	2.00	1.75e-09	6.71e-11	6.65e-09
15	7.57e-10	1.99	4.41e-10	1.66e-11	1.64e-09

Table 8.1: second-order accurate integration in 2D

are presented in Table 8.2 and show the second-order rate of convergence. Our results are compared with the approximations found by using a product of one-dimensional discretized delta functions from [3]; the error is comparable with both approaches, here again.

8.4 Discretization of the multidimensional Dirac distribution

Building upon the results presented here above, we can formulate a consistent discretization for the d-dimensional Dirac distribution. Indeed, using bi-/tri-linear interpolation for the scalar field f, we have the following second-order accurate approximation

$$\int_{\text{cell } C} f \, \mathrm{d}\boldsymbol{x} \approx \frac{1}{2^d} \sum_{i \in \mathcal{N}(C)} f_i \operatorname{vol}(C) . \tag{8.8}$$

Statistics of 50 trials for computing an integral like (8.2) in $\Omega \subset \mathbb{R}^3$

Grid level	Average error	Order	SD	Min	Max
Results using (8.6)					
10	4.49e-07		2.98e-07	1.27e-08	1.45e-06
11	1.14e-07	1.98	8.44e-08	2.42e-09	3.62e-07
12	2.86e-08	2.00	2.43e-08	6.90e-10	8.55e-08
13	6.82e-09	2.07	5.68e-09	6.72e-11	2.01e-08
14	1.61e-09	2.09	1.27e-09	6.54e-11	4.99e-09
15	3.40e-10	2.24	2.87e-10	9.82e-12	1.16e-09
Results using a product of one-dimensional delta functions from [3]					
10	5.01e-07		1.26e-11	4.48e-07	5.64e-07
11	1.26e-07	1.99	7.94e-13	1.14e-07	1.37e-07
12	3.13e-08	2.01	4.92e-14	2.79e-08	3.45e-08
13	7.77e-09	2.01	3.03e-15	6.90e-09	8.55e-09
14	1.94e-09	2.01	1.88e-16	1.72e-09	2.13e-09
15	4.81e-10	2.01	1.16e-17	4.37e-10	5.28e-10

Table 8.2: second-order accurate integration in 3D

Hence, the contribution of the kth grid node to $\int_\Omega f \, \mathrm{d}\boldsymbol{x}$ is found to be $f_k \operatorname{vol}(C)$ by summing this result over the grid cells of a uniform grid. Therefore, the contribution of node k (located at $\boldsymbol{z}_k$) to the integral $\int_\Omega f(\boldsymbol{x})\, \delta(\boldsymbol{x}-\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}$ can be written as

$$f_k \hat{\delta}(\boldsymbol{z}_k, \boldsymbol{y}) \operatorname{vol}(C) \tag{8.9}$$

wherein $\hat{\delta}(\boldsymbol{z}_k, \boldsymbol{y})$ would be some appropriate discretization of $\delta(\boldsymbol{z}_k - \boldsymbol{y})$. Yet, that contribution is also known to be

$$f_k \sum_{C \in \mathcal{V}_k} \sum_{\mathcal{S} \in T(C)} \delta_k(\mathcal{S}, \boldsymbol{y}) \tag{8.10}$$

where $\mathcal{V}_k$ is the set of grid cells surrounding the node k and δ_k is defined in (8.7). Hence, by comparing (8.9) and (8.10), we have the following discretization $\hat{\delta}\left(\boldsymbol{z}_k, \boldsymbol{y}\right)$ for $\delta\left(\boldsymbol{z}_k - \boldsymbol{y}\right)$

$$\hat{\delta}\left(\boldsymbol{z}_k, \boldsymbol{y}\right) = \frac{1}{\mathrm{vol}\left(C\right)} \sum_{C \in \mathcal{V}_k} \sum_{\mathcal{S} \in T(C)} \delta_k\left(\mathcal{S}, \boldsymbol{y}\right). \tag{8.11}$$

We recall that this approach makes sense *only if the grid is uniform in some neighborhood of* $\boldsymbol{y}$. This discretization (8.11) satisfies the equivalent discrete forms of $\int_\Omega \delta\left(\boldsymbol{y} - \boldsymbol{x}\right)\,\mathrm{d}\boldsymbol{x} = 1$ and $\int_\Omega \boldsymbol{x}\delta\left(\boldsymbol{y} - \boldsymbol{x}\right)\,\mathrm{d}\boldsymbol{x} = \boldsymbol{y}$, $\forall \boldsymbol{y} \in \Omega$.

8.4.1 Link with interface delta in two spatial dimensions

This discretization is consistent with what is suggested in [152] for the Dirac distribution over a co-dimension one interface. Indeed, consider a regular co-dimension one interface (in 2D)

$$\Gamma = \left\{\boldsymbol{x} \in \mathbb{R}^2 | \boldsymbol{x} = \boldsymbol{r}\left(s\right),\ s \in [a,b] \subset \mathbb{R}\right\}$$

wherein $\boldsymbol{r}$ is an invertible differentiable mapping between $[a,b]$ and Γ. Then, for a scalar field g defined over Ω, we have

$$\begin{aligned}\int_\Omega g\,\delta_\Gamma\,\mathrm{d}\Omega &= \int_\Gamma g\,\mathrm{d}\Gamma = \int_a^b g\left(\boldsymbol{r}\left(s\right)\right)\left|\frac{\partial \boldsymbol{r}}{\partial s}\right|\,\mathrm{d}s = \int_a^b \int_\Omega g\left(\boldsymbol{x}\right)\left[\left|\frac{\partial \boldsymbol{r}}{\partial s}\right|\ \delta\left(\boldsymbol{x} - \boldsymbol{r}\left(s\right)\right)\right]\mathrm{d}\boldsymbol{x}\,\mathrm{d}s \\ &= \int_\Omega g\left(\boldsymbol{x}\right)\int_a^b \left[\left|\frac{\partial \boldsymbol{r}}{\partial s}\right|\ \delta\left(\boldsymbol{x} - \boldsymbol{r}\left(s\right)\right)\right]\mathrm{d}s\,\mathrm{d}\boldsymbol{x}\end{aligned}$$

so that

$$\delta_\Gamma\left(\boldsymbol{x}\right) = \int_a^b \left|\frac{\partial \boldsymbol{r}}{\partial s}\right| \delta\left(\boldsymbol{x} - \boldsymbol{r}\left(s\right)\right)\,\mathrm{d}s. \tag{8.12}$$

Consider now the two possible cases (see [116]) of an interface crossing a 2-simplex, as illustrated in Figure 8.3. Let ϕ_k be the signed distance from $\boldsymbol{z}_k$ to the interface[4], and

[4]Without loss of generality, the simplex nodes are sorted in a decreasing order of ϕ.

assume $\phi_0 > 0$. Approximating the interface by a straight line in the simplex, one can use the curvilinear coordinate $\boldsymbol{r}(s) = \boldsymbol{z}_{02} + (\boldsymbol{z}_{\mathrm{e}} - \boldsymbol{z}_{02})\, s/L$ where $L = \|\boldsymbol{z}_{\mathrm{e}} - \boldsymbol{z}_{02}\|$, $\boldsymbol{z}_{\mathrm{e}}$ stands for either $\boldsymbol{z}_{12}$ or $\boldsymbol{z}_{01}$ and where $\boldsymbol{z}_{ik} = \boldsymbol{z}_i + (\boldsymbol{z}_k - \boldsymbol{z}_i)\, \phi_i / (\phi_i - \phi_k)$ when ϕ_i and ϕ_k are of different signs.

Then, building upon (8.11) and (8.12) a consistent approximation for the contribution of the considered simplex $\mathcal{S}$ (included in cell C) to $\delta_\Gamma(\boldsymbol{z}_0)$ would be

$$\frac{1}{\mathrm{vol}(C)} \int_0^L \delta_0(\mathcal{S}, \boldsymbol{r}(s))\, \mathrm{d}s \quad = \quad \frac{L}{\mathrm{vol}(C)} \frac{|(\boldsymbol{z}_2 - \boldsymbol{r}(L/2)) \times (\boldsymbol{z}_1 - \boldsymbol{r}(L/2))|}{|(\boldsymbol{z}_2 - \boldsymbol{z}_0) \times (\boldsymbol{z}_1 - \boldsymbol{z}_0)|}.$$

After calculations, the above expression yields to

$$\frac{\|\boldsymbol{z}_{12} - \boldsymbol{z}_{02}\|}{2\,\mathrm{vol}(C)} \frac{\phi_2}{\phi_2 - \phi_0} \qquad \text{or} \qquad \frac{\|\boldsymbol{z}_{01} - \boldsymbol{z}_{02}\|}{2\,\mathrm{vol}(C)} \left(\frac{\phi_1}{\phi_1 - \phi_0} + \frac{\phi_2}{\phi_2 - \phi_0}\right) \tag{8.13}$$

respectively when $\phi_1 > 0$ and $\phi_2 < 0$ (*i.e.* $\boldsymbol{z}_{\mathrm{e}} = \boldsymbol{z}_{12}$) or $\phi_1 < 0$ and $\phi_2 < 0$ (*i.e.* $\boldsymbol{z}_{\mathrm{e}} = \boldsymbol{z}_{01}$). Both expressions in (8.13) are exactly the approximations derived in [152] for the contribution of the simplex $\mathcal{S}$ to $\delta_\Gamma(\boldsymbol{z}_0)$.

8.4.2 Link with interface delta in three spatial dimensions

The same consistency is observed in the 3D case. Consider, a regular co-dimension one interface (in 3D)

$$\Gamma = \left\{\boldsymbol{x} \in \mathbb{R}^3 | \boldsymbol{x} = \boldsymbol{r}(s,t),\ (s,\, t) \in \Lambda \subset \mathbb{R}^2\right\}$$

wherein $\boldsymbol{r}$ is an invertible differentiable mapping between Λ and Γ. Then, for a scalar field g defined over Ω, we have

$$\begin{aligned}\int_\Omega g\,\delta_\Gamma\,\mathrm{d}\Omega &= \int_\Gamma g\,\mathrm{d}\Gamma = \int_{\Lambda(s,t)}\int_\Omega g\left(\boldsymbol{x}\right)\left[\left|\frac{\partial\boldsymbol{r}}{\partial s}\times\frac{\partial\boldsymbol{r}}{\partial t}\right|\,\delta\left(\boldsymbol{x}-\boldsymbol{r}\left(s,t\right)\right)\right]\mathrm{d}\boldsymbol{x}\,\mathrm{d}s\,\mathrm{d}t\\ &= \int_\Omega g\left(\boldsymbol{x}\right)\int_{\Lambda(s,t)}\left[\left|\frac{\partial\boldsymbol{r}}{\partial s}\times\frac{\partial\boldsymbol{r}}{\partial t}\right|\,\delta\left(\boldsymbol{x}-\boldsymbol{r}\left(s,t\right)\right)\right]\mathrm{d}s\,\mathrm{d}t\,\mathrm{d}\boldsymbol{x}\end{aligned}$$

so that

$$\delta_\Gamma\left(\boldsymbol{x}\right) = \int_{\Lambda(s,t)}\left|\frac{\partial\boldsymbol{r}}{\partial s}\times\frac{\partial\boldsymbol{r}}{\partial t}\right|\delta\left(\boldsymbol{x}-\boldsymbol{r}\left(s,t\right)\right)\,\mathrm{d}s\,\mathrm{d}t. \tag{8.14}$$

For the sake of illustration, consider now one of the three possible different cases for an interface crossing a 3-simplex (see [116]), as illustrated in Figure 8.3. For this illustration, we assume $\phi_0 > 0$, ϕ_1, ϕ_2 and ϕ_3 negative. Let the vectors $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ be

$$\boldsymbol{u} = \boldsymbol{z}_2 - \boldsymbol{z}_1, \quad \boldsymbol{v} = \boldsymbol{z}_3 - \boldsymbol{z}_1, \quad \boldsymbol{w} = \boldsymbol{z}_0 - \boldsymbol{z}_1,$$

then we have

$$\boldsymbol{z}_{01} = \boldsymbol{z}_1 + \frac{\phi_1}{\phi_1-\phi_0}\boldsymbol{w}, \quad \boldsymbol{z}_{02} = \boldsymbol{z}_1 + \frac{\phi_2}{\phi_2-\phi_0}\boldsymbol{w} - \frac{\phi_0}{\phi_2-\phi_0}\boldsymbol{u}, \text{ and } \boldsymbol{z}_{03} = \boldsymbol{z}_1 + \frac{\phi_3}{\phi_3-\phi_0}\boldsymbol{w} - \frac{\phi_0}{\phi_3-\phi_0}\boldsymbol{v},$$

and we can use the mapping $\boldsymbol{r}\left(s,t\right) = \boldsymbol{z}_{01} + s\left(\boldsymbol{z}_{02}-\boldsymbol{z}_{01}\right) + t\left(\boldsymbol{z}_{03}-\boldsymbol{z}_{01}\right)$, $s \in [0,\,1]\,, t \in [0,\,1-s]$, and $\delta_0\left(\mathcal{S},\boldsymbol{r}\left(s,t\right)\right)$ reads

$$\frac{\left(\boldsymbol{u}\times\boldsymbol{v}\right)\cdot\left(\boldsymbol{r}\left(s,t\right)-\boldsymbol{z}_1\right)}{\left(\boldsymbol{u}\times\boldsymbol{v}\right)\cdot\boldsymbol{w}} = \frac{\phi_1}{\phi_1-\phi_0} + s\left(\frac{\phi_2}{\phi_2-\phi_0} - \frac{\phi_1}{\phi_1-\phi_0}\right) + t\left(\frac{\phi_3}{\phi_3-\phi_0} - \frac{\phi_1}{\phi_1-\phi_0}\right).$$

Therefore, the discrete equivalent of (8.14) for the contribution of the current simplex $\mathcal{S}$

(included in cell C) to $\delta_\Gamma(\boldsymbol{z}_0)$ yields after calculations[5]

$$\frac{2\,A\left(\boldsymbol{z}_{01}\boldsymbol{z}_{02}\boldsymbol{z}_{03}\right)}{\mathrm{vol}\left(C\right)}\int_0^1\int_0^{1-s}\delta_0\left(\mathcal{S},\boldsymbol{r}\left(s,t\right)\right)\mathrm{d}t\,\mathrm{d}s=\frac{1}{3}\frac{A\left(\boldsymbol{z}_{01}\boldsymbol{z}_{02}\boldsymbol{z}_{03}\right)}{\mathrm{vol}\left(C\right)}\left(\frac{\phi_1}{\phi_1-\phi_0}+\frac{\phi_2}{\phi_2-\phi_0}+\frac{\phi_3}{\phi_3-\phi_0}\right),$$

which is exactly what is suggested in [152]. The same conclusion holds for the two other cases.

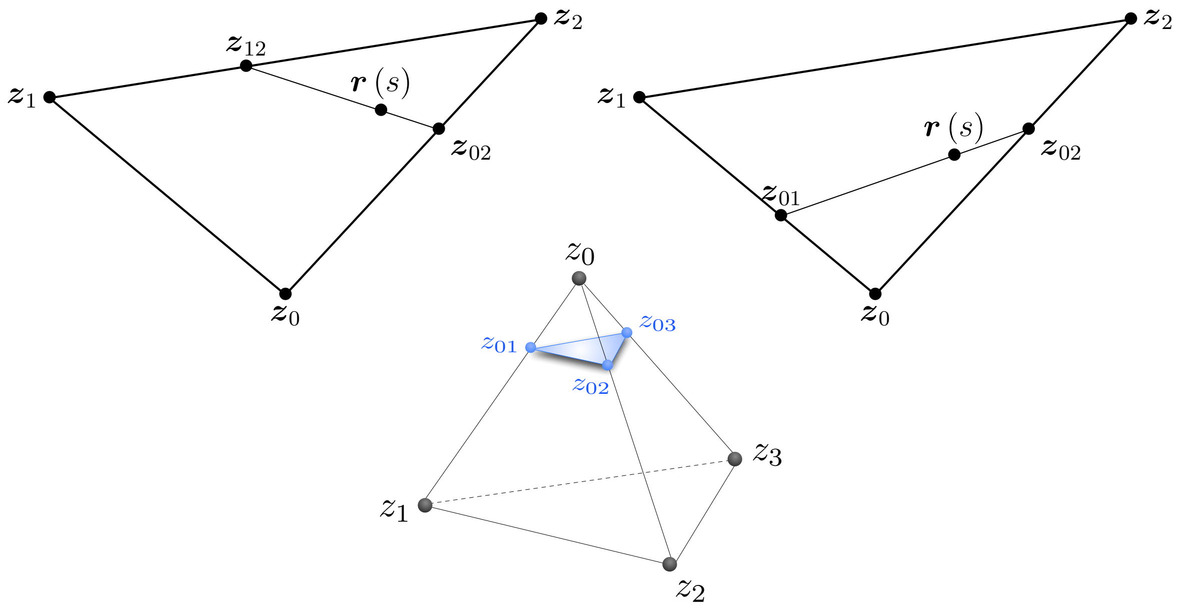


Figure 8.3: top: the two distinct cases for an interface crossing a 2-simplex, as described in [116]. Bottom: one of the three cases for an interface crossing a 3-simplex (see Fig. 4 in [116] for the two other cases).

8.5 Solving Poisson equation with singular source terms

In this section, we show that the discrete formula of δ can be used for solving Poisson equations with singular source terms. When dealing with constant coefficient Poisson

[5]Note that $\left|\frac{\partial\boldsymbol{r}}{\partial s}\times\frac{\partial\boldsymbol{r}}{\partial t}\right|=2\,A\left(\boldsymbol{z}_{01}\boldsymbol{z}_{02}\boldsymbol{z}_{03}\right)$ with the considered mapping $\boldsymbol{r}\left(s,t\right)$, where $A\left(\boldsymbol{z}_i\boldsymbol{z}_j\boldsymbol{z}_k\right)$ is the area of the triangle defined by points $\boldsymbol{z}_i,\boldsymbol{z}_j$ and $\boldsymbol{z}_k$.

equations, a very convenient and well-known way to circumvent any difficulty raised by such singular source terms is the following approach. Consider the problem

$$-\Delta u = f(\boldsymbol{x}) + \sum_{s=1}^{S} \alpha_s \delta(\boldsymbol{x} - \boldsymbol{x}_s) \text{ for } \boldsymbol{x} \in \Omega, \quad \text{and} \quad \mathcal{B}[u](\boldsymbol{x}) = g(\boldsymbol{x}) \text{ for } \boldsymbol{x} \in \partial\Omega,$$

where f and g are continuous functions, $S \geq 1$ is an integer, $\alpha_s \in \mathbb{R}$, $\boldsymbol{x}_s \in \Omega$ and $\mathcal{B}$ is a linear functional representing the boundary condition(s) on u. Since the latter problem is fully linear in u, we can write the solution as $u(\boldsymbol{x}) := v(\boldsymbol{x}) + \sum_{s=1}^{N} \alpha_s w(\boldsymbol{x} - \boldsymbol{x}_s)$ where w is a fundamental solution of $-\Delta w = \delta(\boldsymbol{x})$ in $\mathbb{R}^d$, *e.g.* $w(\boldsymbol{x}) = -\frac{1}{2\pi} \log |\boldsymbol{x}|$ if $d = 2$, and v satisfies

$$-\Delta v = f(\boldsymbol{x}) \text{ for } \boldsymbol{x} \in \Omega, \quad \text{and} \quad \mathcal{B}[v](\boldsymbol{x}) = g(\boldsymbol{x}) - \sum_{s=1}^{S} \alpha_s \mathcal{B}[w](\boldsymbol{x} - \boldsymbol{x}_s) \text{ for } \boldsymbol{x} \in \partial\Omega.$$

which has the advantage of being well posed in the entire domain Ω and free of singular source terms.

While this approach is preferable when it can be used, it does not easily extend to more general cases like nonlinear boundary conditions, periodic boundary conditions or variable diffusion coefficients for instance. In the following, we aim to show that the discretization of δ presented here above stands as a more general way to address such problems from a purely computational point of view.

First, we recall the finite difference discretization of the Poisson equation that is used throughout this section. Then, we show that the resulting solutions to the Poisson equation with Dirac-distributed source terms converge in the L^1- and L^∞-norms, in both two and three spatial dimensions. Supra-convergence of the gradients is observed as well: the convergence of the solutions *and* their gradients is superlinear. The order of convergence is either 2 or slightly smaller. Although we show only the results obtained with the above discretization here below (for the sake of conciseness), we have compared them to results found with products of one-dimensional discretized delta functions from [3]: the errors were of the same order of magnitude in all cases. However, for the applications illustrated in sections 8.5.2, 8.5.2, 8.5.3 and 8.5.3, the extended stencil associated with the approach from [3] made it impossible to use for the coarsest considered grids. Hence, the discretization (8.11) gains significant practical interest compared to the method from [3] for multi-scale applications.

8.5.1 Discretization of the Poisson equation

We restrict ourselves to two spatial dimensions for the presentation of the numerical scheme, the extension to three spatial dimensions being straightforward. More details about this scheme and formal proofs of its convergence can be found in [179]. We also refer the interested readers to [218, 219] for formal proofs of the second-order convergence, and [220] for the second-order convergence of the gradient in the L^2-norm. In all cases, such proofs of rate(s) of convergence (in the entire domain Ω) always assume well-defined, smooth, non-singular source terms.

Let us consider the variable coefficient Poisson equation $-\nabla\cdot(\beta\nabla u) = f$ on a Cartesian domain Ω with Dirichlet boundary conditions on its boundary $\partial\Omega$ (β is bounded from below by a positive constant). In the case of a uniform grid $(x_i, y_j) = (ih_x, jh_y)$, $i = 0, 1, \dots, I$, $j = 0, 1, \dots, J$, the standard finite difference discretization can be written as

$$\begin{aligned} -\frac{1}{h_x}\left(\frac{\beta_{i+1,j}+\beta_{i,j}}{2}\frac{u_{i+1,j}-u_{i,j}}{h_x} - \frac{\beta_{i,j}+\beta_{i-1,j}}{2}\frac{u_{i,j}-u_{i-1,j}}{h_x}\right) \\ -\frac{1}{h_y}\left(\frac{\beta_{i,j+1}+\beta_{i,j}}{2}\frac{u_{i,j+1}-u_{i,j}}{h_y} - \frac{\beta_{i,j}+\beta_{i,j-1}}{2}\frac{u_{i,j}-u_{i,j-1}}{h_y}\right) = f_{i,j}, \end{aligned} \tag{8.15}$$

$0 < i < I$, $0 < j < J$, where $u_{i,j}$ is the numerical approximation of the unknown function u at node (x_i, y_j) and $f_{i,j} = f(x_i, y_j)$. This scheme has been used extensively and produces second-order accurate results for u and its derivatives (assuming f is bounded in Ω).

Uniform grids restrict the range of tractable problems since the computational cost is proportional to the number of grid points, regardless of whether the grid needs to be refined only locally or not. Quadtree and Octree grids preform much more efficiently with such problems since they allow for local grid refinement. More precisely, consider the case depicted in Figure 8.4 in two spatial dimensions: the entire rectangular domain is originally associated with the root of the tree and then split into four cells of equal sizes, called the children of the root. The discretization proceeds recursively, i.e. each cell can be in turn split into four children and so forth. A cell with no children is called a leaf. By definition, the `level` corresponding to the root is zero and is incremented by one every time a new generation of children is added.

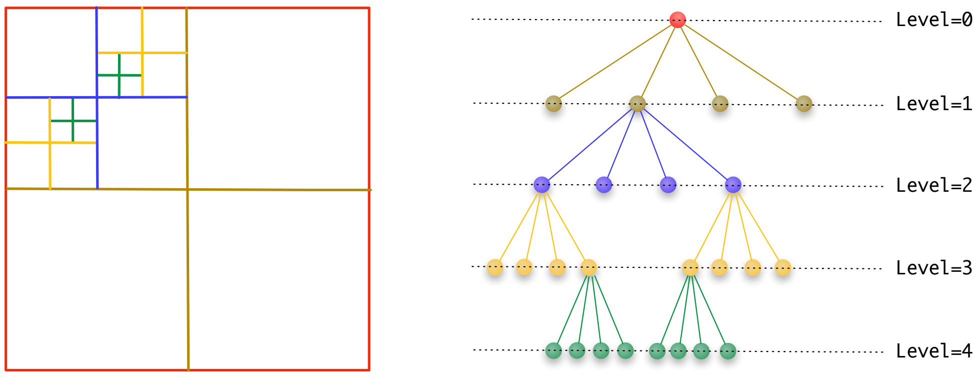


Figure 8.4: discretization of a two-dimensional domain (left) and its Quadtree representation (right). The entire domain corresponds to the root of the tree (level 0), and each cell subdivided further points to its four children. In this example, the tree is not graded since the difference of level between neighboring cells can exceed one.

In the following sections, we use the node-based discretization of the Poisson equation on Quad-/Oc-tree introduced in [179]. Consider first the case of a point having four neighboring nodes along the x and y axes, as illustrated in Figure 8.5. In that case, the discretization of the Poisson equation reads

$$
\begin{aligned}
-\left(\frac{\beta_1+\beta_0}{2}\frac{u_1-u_0}{s_1}-\frac{\beta_0+\beta_3}{2}\frac{u_0-u_3}{s_3}\right)\frac{2}{s_1+s_3}& \\
-\left(\frac{\beta_2+\beta_0}{2}\frac{u_2-u_0}{s_2}-\frac{\beta_0+\beta_4}{2}\frac{u_0-u_4}{s_4}\right)\frac{2}{s_2+s_4}&=f_0
\end{aligned}
\tag{8.16}
$$

where f_0 is the value of the function f at the node associated with u_0. Note that this scheme reduces to (8.16) in the case of a locally uniform grid (*i.e.* when $s_1 = s_3 = h_x$ and $s_2 = s_4 = h_y$).

In the more general case of a T-junction, one of the grid nodes in (8.16), say u_1 and β_1, might be missing and a linear interpolation based on available neighboring nodes is used instead (see Figure 8.5, nodes 5 and 6 would be used in that case). [179] observed that this produces a spurious term (to first order) that is proportional to $\partial_y(\beta\partial_y u)$. In the case of a node-based discretization, this spurious term can be canceled appropriately by weighting the approximation for $\partial_y(\beta\partial_y u)$ which does not require such an interpolation. The Poisson discretization at u_0 is thus:

$$
\begin{aligned}
-\left(\frac{\frac{\beta_5+\beta_0}{2}\frac{s_6}{s_5+s_6}(u_5-u_0)+\frac{\beta_6+\beta_0}{2}\frac{s_5}{s_5+s_6}(u_6-u_0)}{s_1}-\frac{\beta_0+\beta_3}{2}\frac{u_0-u_3}{s_3}\right)\frac{2}{s_1+s_3}& \\
-\left(\frac{\beta_2+\beta_0}{2}\frac{u_2-u_0}{s_2}-\frac{\beta_4+\beta_0}{2}\frac{u_0-u_4}{s_4}\right)\frac{2}{s_2+s_4}w&=f_0
\end{aligned}
\tag{8.17}
$$

where the weight $w = 1 - \dfrac{s_5 s_6}{s_1(s_1+s_3)}$ cancels the spurious term induced by the linear interpolation due to the missing grid node 1. This numerical scheme extends to three spatial dimensions in a very similar way (see [179] for the details).

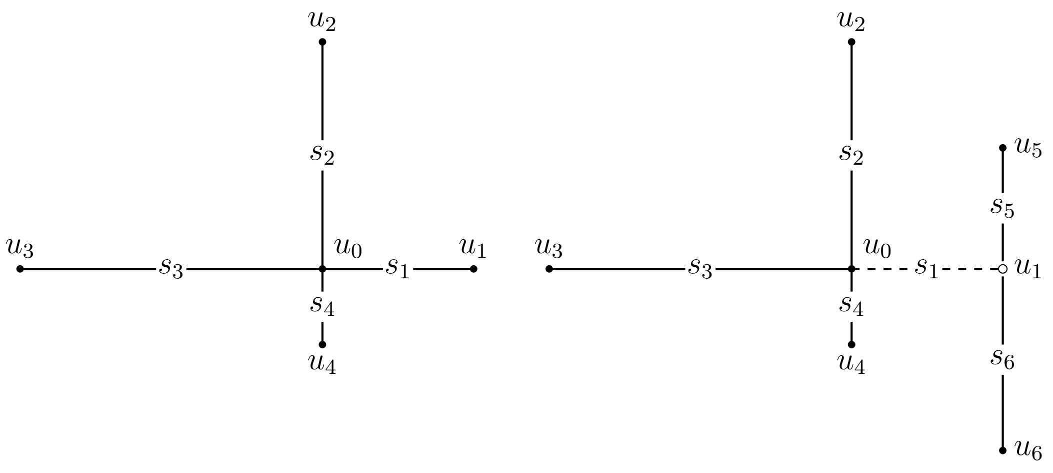


Figure 8.5: discretization of the Poisson equation. Left: case of a five-point stencil; right: general case of a T-junction.

8.5.2 Numerical examples in two spatial dimensions

In this section, we show that the discrete formula of δ can be applied to solving Poisson equations with point-located singular source terms, in two spatial dimensions. We show the superlinear convergence for the resulting solutions (and their gradients) in the L^1- and L^∞-norms.

One singular source term

Consider the Poisson equation

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= 2\pi\delta(\boldsymbol{x}-\overline{\boldsymbol{x}}) \quad && \in \Omega \\ u(\boldsymbol{x}) &= 1-\ln\left(\frac{|\boldsymbol{x}-\overline{\boldsymbol{x}}|}{\operatorname{diag}(\Omega)}\right) && \text{on } \partial\Omega \end{aligned}$$

where $\Omega = [-1,\,1]\times[-1,\,1]$ and $\operatorname{diag}(\Omega) = 2\sqrt{2}$ can be seen as an arbitrary length chosen to make the argument of the logarithm non-dimensional. The exact solution is $u(\boldsymbol{x}) = 1-\ln\left(\frac{|\boldsymbol{x}-\overline{\boldsymbol{x}}|}{\operatorname{diag}(\Omega)}\right)$ in $\Omega\setminus\{\overline{\boldsymbol{x}}\}$, thus singular at $\overline{\boldsymbol{x}}$.

This equation is solved using the above discretization for the multidimensional delta function. Since the analytic solution itself is singular at $\overline{\boldsymbol{x}}$, we define

$$\overline{\Omega}_\alpha = \{\boldsymbol{x} : \boldsymbol{x} \in \Omega, |\boldsymbol{x} - \overline{\boldsymbol{x}}| \geq \alpha\}$$

($\alpha \in]0, \operatorname{diag}(\Omega)]$) for the error analysis. Table 8.3 shows that our discretization produces results that are second-order accurate in the L^1-norm and in the L^∞-norm in $\overline{\Omega}_{0.05\,\operatorname{diag}(\Omega)}$. The convergence of the discrete fundamental solution to the corresponding fundamental continuous Green function on a uniform (infinite) grid away from the source location is well-known and studied (see [221] for instance); the current example shows that convergence on Quadtree grids holds as well.

A note on refinement: by integrating both sides of the Poisson equation over a ball $\mathcal{B}$ of radius ρ centered at $\overline{\boldsymbol{x}}$ where a source term $\alpha\delta(\boldsymbol{x} - \overline{\boldsymbol{x}})$ acts, the divergence theorem leads to

$$\overline{\nabla_{\boldsymbol{n}} u} \sim \frac{\alpha}{2\pi\rho}, \ \rho \to 0, \tag{8.18}$$

where $\boldsymbol{n} = \dfrac{\boldsymbol{y} - \overline{\boldsymbol{x}}}{|\boldsymbol{y} - \overline{\boldsymbol{x}}|}$, $\boldsymbol{y} \in \partial\mathcal{B}$ is the outer normal vector and $\overline{\nabla_{\boldsymbol{n}} u} = \dfrac{1}{2\pi\rho}\int_{\partial\mathcal{B}} \boldsymbol{n} \cdot \nabla u \, \mathrm{d}\boldsymbol{y}$ is the average of the normal gradient of u over $\partial\mathcal{B}$. The behavior (8.18) shows that the gradient of u diverges close to Dirac-distributed sources. Therefore, the closer to a Dirac source a region is, the more computationally expensive it is to capture the exact behavior of the solution in that region in the sense that it requires finer grids for a given level of accuracy. Nevertheless, the solution outside a small region away from the source can be computed accurately.

Building upon that observation, we propose the procedure 8.5.1 for constructing a grid of maximum level M and minimum level m (at least), which we refer to as an m/M Quadtree grid. This procedure is meant to balance the computational effort efficiently in the domain: for one single Dirac source, it decomposes the entire computational domain in circular annuli containing at least two grid cells and such that the local truncation error is comparable from one annulus to another, hoping for a balanced error in the whole domain.

If there exists a distance below which the numerical solution is irrelevant for the considered application, it makes sense to set d (see procedure 8.5.1) equal to that distance. For the results in Table 8.3, we chose $d = b = 0.05\,\mathrm{diag}\,(\Omega)$ consistently with the error analysis in $\overline{\Omega}_b$. Examples of such grids are illustrated in Figure 8.6.

Construction procedure 8.5.1.

Construction of an m/M Quadtree grid.

Given a minimal distance of interest d to the Dirac-distributed sources and the dimensions L_x, L_y of the computational domain Ω, define the threshold distances t_k, $k = 0, \dots, M-m-1$ such that $t_{M-m-1} = d\sqrt{2}$ and

$$t_{M-m-1-k} = \max\left(\sqrt{2}\,t_{M-m-k},\ t_{M-m-k} + \frac{\sqrt{L_x^2 + L_y^2}\,2^{k+1}}{2^{M-m}}\right),$$

$k = 1, 2, \dots, M-m-1.$

Starting from a root cell of dimensions L_x, L_y, split it m times.

Then, let an integer k run from 0 to $M-m-1$ and split any cell C if

$$\min_{v \in \mathrm{vertices}(C)} \phi\,(v) \le t_k$$

where v refers to a vertex (node) of the cell C and ϕ is the smallest distance to a Dirac-distributed source.

Note that the threshold distances t_k in procedure 8.5.1 depend only on $(M-m)$, for a given domain Ω and a distance d. Hence, the m/M Quadtree grid that it creates can be understood as the mth refinement of the corresponding $0/\,(M-m)$ Quadtree grid, which explains why the minimum level of the m/M grid may be greater than m. Besides, considering the $0/\left(M-m\right)$ Quadtree grid, it makes sense to require its finer grid size is to be smaller than d, hence

$$M-m>\log_2\left(\frac{\sqrt{L_x^2+L_y^2}}{d}\right).$$

We restrict ourselves to grids satisfying that condition when using Quadtree grids.

One singular source term with variable diffusion coefficient

The case of a variable diffusion coefficient is of significant interest both from a theoretical and a practical point of view. Indeed, in presence of a variable diffusion coefficient, the usual procedure to circumvent difficulties raised by the presence of Dirac-distributed source terms (as detailed in the beginning of section 8.5) cannot be applied in a straightforward manner since no general *fundamental* solution can be formulated easily in presence of a variable diffusion coefficient $\beta\left(\boldsymbol{x}\right)$. Moreover, a well-chosen diffusion coefficient β can help alleviate the unbounded growth of the solution gradient close to the source location and lead to a bounded solution in Ω, allowing for an error analysis in the entire domain (for any p-norm with p finite).

Consider for instance

$$\beta\left(\boldsymbol{x}\right)=2\sqrt{\frac{\operatorname{diag}\left(\Omega\right)}{\left|\boldsymbol{x}-\overline{\boldsymbol{x}}\right|}},\quad \boldsymbol{x}\in\Omega\backslash\left\{\overline{\boldsymbol{x}}\right\}$$

for some randomly located $\overline{\boldsymbol{x}}\in\Omega$ and the corresponding variable diffusion coefficient

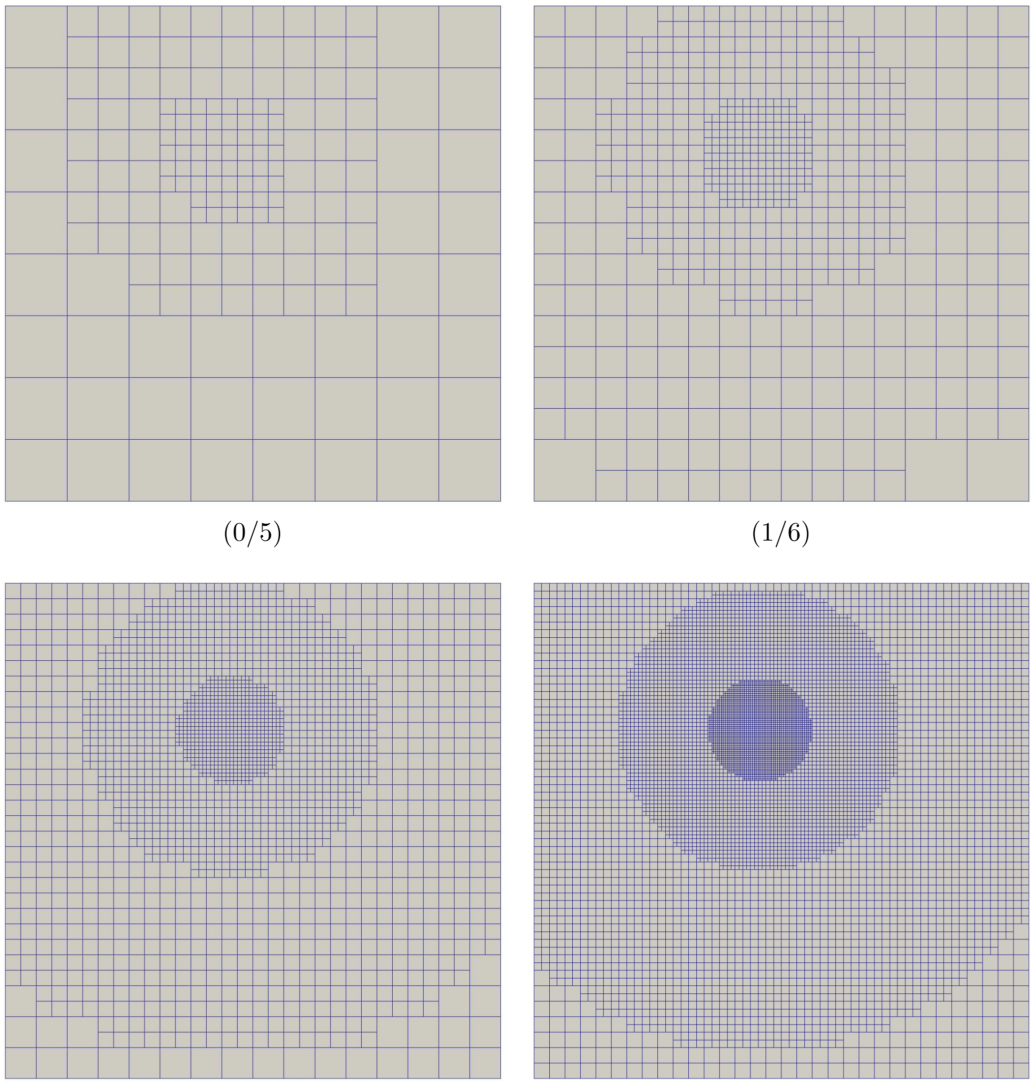


Figure 8.6: grids generated for problem 8.5.2. The distance d (see procedure 8.5.1) is chosen equal to $b = 0.05\,\mathrm{diag}\,(\Omega)$.

Error analysis for solving a problem like 8.5.2.

Grid level	$\frac{\lVert u-u_h\rVert_\infty}{\lVert u\rVert_\infty}$	rate	$\frac{\lVert u_x-u_{h,x}\rVert_\infty}{\lVert u_x\rVert_\infty}$	rate	$\frac{\lVert u_y-u_{h,y}\rVert_\infty}{\lVert u_y\rVert_\infty}$	rate
			QUADTREES			
0/5	4.796e-03		7.148e-02		6.434e-02	
1/6	1.810e-03	1.406	3.047e-02	1.230	2.731e-02	1.236
2/7	5.709e-04	1.665	6.665e-03	2.193	6.801e-03	2.006
3/8	1.357e-04	2.073	1.768e-03	1.914	1.216e-03	2.484
4/9	3.934e-05	1.786	5.175e-04	1.773	4.992e-04	1.284
5/10	9.146e-06	2.105	1.152e-04	2.168	1.233e-04	2.017
6/11	2.749e-06	1.734	3.267e-05	1.818	1.768e-05	2.802
7/12	5.097e-07	2.431	5.509e-06	2.568	6.766e-06	1.386
Mean order		1.886		1.952		1.888
			UNIFORM GRIDS			
5/5	4.297e-03		7.201e-02		6.498e-02	
6/6	2.025e-03	1.085	2.994e-02	1.266	2.696e-02	1.269
7/7	6.768e-04	1.581	6.700e-03	2.160	6.756e-03	1.996
8/8	1.027e-04	2.721	1.746e-03	1.940	1.205e-03	2.488
9/9	4.827e-05	1.089	5.145e-04	1.763	4.927e-04	1.290
10/10	1.106e-05	2.126	1.137e-04	2.178	1.226e-04	2.007
11/11	2.018e-06	2.454	3.183e-05	1.837	1.749e-05	2.809
12/12	3.314e-07	2.607	5.298e-06	2.587	6.635e-06	1.398
Mean order		1.952		1.962		1.894
Grid level	$\frac{\lVert u-u_h\rVert_1}{\lVert u\rVert_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\lVert u_x-u_{h,x}\rVert_1}{\lVert u_x\rVert_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\lVert u_y-u_{h,y}\rVert_1}{\lVert u_y\rVert_\infty \mathrm{vol}(\Omega)}$	rate
			QUADTREES			
0/5	3.286e-04		2.834e-03		2.704e-03	
1/6	1.069e-04	1.620	8.581e-04	1.724	8.250e-04	1.713
2/7	3.917e-05	1.449	2.434e-04	1.818	2.516e-04	1.713
3/8	1.113e-05	1.816	6.733e-05	1.854	5.907e-05	2.091
4/9	3.199e-06	1.798	1.784e-05	1.916	1.706e-05	1.792
5/10	8.269e-07	1.952	4.288e-06	2.057	4.495e-06	1.924
6/11	2.039e-07	2.020	1.183e-06	1.857	9.617e-07	2.225
7/12	5.217e-08	1.967	2.583e-07	2.196	2.739e-07	1.812
Mean order		1.803		1.917		1.890
			UNIFORM GRIDS			
5/5	1.468e-04		1.139e-03		1.068e-03	
6/6	4.637e-05	1.663	3.167e-04	1.846	2.975e-04	1.844
7/7	1.608e-05	1.528	8.544e-05	1.890	9.041e-05	1.718
8/8	2.470e-06	2.703	2.066e-05	2.048	1.452e-05	2.638
9/9	1.142e-06	1.113	6.161e-06	1.745	5.618e-06	1.370
10/10	2.530e-07	2.174	1.282e-06	2.265	1.472e-06	1.932
11/11	4.539e-08	2.479	3.700e-07	1.792	2.133e-07	2.787
12/12	8.687e-09	2.385	5.822e-08	2.668	7.110e-08	1.585
Mean order		2.006		2.037		1.982

Table 8.3: accuracy analysis of the Poisson problem 8.5.2: one trial for a random source location $\overline{\boldsymbol{x}}$. All norms are in $\overline{\Omega}_b$ where $b = 0.05\,\mathrm{diag}\,(\Omega)$. The Quadtree grids are constructed using procedure 8.5.1 with $d = b$. For this specific problem, one obtains comparable maximum errors between uniform grids and Quadtrees if $d = b$.

Poisson equation

$$
\begin{aligned}
-\nabla\cdot\left(\beta\nabla u\right) &= 2\pi\delta\left(\boldsymbol{x}-\overline{\boldsymbol{x}}\right) &&\in \Omega\backslash\left\{\overline{\boldsymbol{x}}\right\}\\
u\left(\boldsymbol{x}\right) &= 1-\sqrt{\frac{\left|\boldsymbol{x}-\overline{\boldsymbol{x}}\right|}{\operatorname{diag}\left(\Omega\right)}} &&\text{on } \partial\Omega
\end{aligned}
$$

where Ω and $\operatorname{diag}\left(\Omega\right)$ are as before[6]. The exact solution is $u\left(\boldsymbol{x}\right)=1-\sqrt{\left|\boldsymbol{x}-\overline{\boldsymbol{x}}\right|/\operatorname{diag}\left(\Omega\right)}$ (note that it is bounded in Ω).

This problem is solved using the above discretization for the multidimensional delta function. Since the exact solution has unbounded derivatives close to $\overline{\boldsymbol{x}}$, convergence in the L^{∞}-norm is not ensured in the entire domain Ω. However, Table 8.4 shows that the numerical approximation still converges in the L^{∞}-norm in $\overline{\Omega}_{0.01\operatorname{diag}(\Omega)}$ (as defined in section 8.5.2) *and* in the L^{1}-norm in the *entire* domain Ω. The order of convergence is about 1.6-1.7 for both the solution and its derivatives in any considered norm.

Multiple singular source terms

Consider now the Poisson equation with N Dirac source terms ($N\in\mathbb{N}^{*}=\mathbb{N}\backslash\left\{0\right\}$)

$$
\begin{aligned}
-\Delta u\left(\boldsymbol{x}\right) &= \sum_{k=1}^{N}\alpha_{k}\,\delta\left(\boldsymbol{x}-\overline{\boldsymbol{x}}_{k}\right) &&\in\Omega\\
u\left(\boldsymbol{x}\right) &= 1-\sum_{k=1}^{N}\frac{\alpha_{k}}{2\pi}\ln\left(\frac{\left|\boldsymbol{x}-\overline{\boldsymbol{x}}_{k}\right|}{\operatorname{diag}\left(\Omega\right)}\right) &&\text{on }\partial\Omega
\end{aligned}
$$

where Ω and $\operatorname{diag}\left(\Omega\right)$ are as before. In this problem, the points $\overline{\boldsymbol{x}}_{k}$ are randomly located in Ω and the intensities α_{k} are random variables with uniform probability density in

[6]The components of the random source location $\overline{\boldsymbol{x}}$ are defined as $\frac{\pi}{4}\frac{q}{Q}$ where $Q=1024$ and q is a random integer in $[-Q,Q]$, so that $\overline{\boldsymbol{x}}$ cannot coincide with a grid node and the grid values of β are well-defined at the grid nodes.

Error analysis for solving a problem like 8.5.2.

Grid level	$\frac{\|u-u_h\|_\infty}{\|u\|_\infty}$	rate	$\frac{\|u_x-u_{h,x}\|_\infty}{\|u_x\|_\infty}$	rate	$\frac{\|u_y-u_{h,y}\|_\infty}{\|u_y\|_\infty}$	rate
0/7	3.001e-03		5.499e-03		4.139e-03	
1/8	6.231e-04	2.268	1.885e-03	1.545	1.312e-03	1.657
2/9	2.603e-04	1.259	6.812e-04	1.468	4.543e-04	1.530
3/10	1.223e-04	1.089	1.635e-04	2.059	9.980e-05	2.187
4/11	2.771e-05	2.142	3.584e-05	2.190	2.558e-05	1.964
5/12	9.544e-06	1.538	1.414e-05	1.342	7.246e-06	1.820
6/13	2.551e-06	1.904	3.776e-06	1.905	1.943e-06	1.899
7/14	7.030e-07	1.859	1.023e-06	1.884	6.048e-07	1.683
Mean order		1.723		1.770		1.820
Grid level	$\frac{\|u-u_h\|_1}{\|u\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\|u_x-u_{h,x}\|_1}{\|u_x\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\|u_y-u_{h,y}\|_1}{\|u_y\|_\infty \mathrm{vol}(\Omega)}$	rate
0/7	2.179e-04		1.095e-04		7.824e-05	
1/8	1.514e-04	0.526	3.301e-05	1.731	2.344e-05	1.739
2/9	5.295e-05	1.516	1.085e-05	1.606	7.486e-06	1.647
3/10	1.808e-05	1.550	2.613e-06	2.054	2.409e-06	1.636
4/11	4.565e-06	1.986	7.164e-07	1.867	6.924e-07	1.799
5/12	1.053e-06	2.115	2.517e-07	1.509	2.161e-07	1.680
6/13	2.641e-07	1.996	7.013e-08	1.844	6.186e-08	1.804
7/14	8.194e-08	1.689	2.158e-08	1.700	1.915e-08	1.692
Mean order		1.625		1.759		1.714

Table 8.4: accuracy analysis of the variable diffusion coefficient Poisson problem 8.5.2: one trial for a random source location $\overline{\boldsymbol{x}}$. L^∞ norms are in $\overline{\Omega}_{0.01\,\mathrm{diag}(\Omega)}$ and L^1-norms are in the entire domain Ω. The grid construction follows the procedure 8.5.1 with $d = 0.01\mathrm{diag}\,(\Omega)$.

$[-4\,\pi, 4\,\pi]$. The exact solution is

$$u(\boldsymbol{x}) = 1 - \sum_{k=1}^{N} \frac{\alpha_k}{2\pi} \ln\left(\frac{|\boldsymbol{x} - \overline{\boldsymbol{x}}_k|}{\operatorname{diag}(\Omega)}\right)$$

in $\Omega \backslash \bigcup_{k=1}^{N} \{\overline{\boldsymbol{x}}_k\}$, hence singular at points $\overline{\boldsymbol{x}}_k$ $(k = 1, \ldots, N)$.

It is solved using the above discretization for the multidimensional delta function. Since the analytic solution itself is singular at $\overline{\boldsymbol{x}}_k$, we define

$$\overline{\Omega}_\alpha = \{\boldsymbol{x} : \boldsymbol{x} \in \Omega, |\boldsymbol{x} - \overline{\boldsymbol{x}}_k| \geq \alpha, \forall k = 1, \ldots, N\}$$

$(\alpha \in \,]0, \operatorname{diag}(\Omega)])$ for the error analysis. Table 8.5 shows that our discretization produces results that converge in the L^1-norm and in the L^∞-norm in $\overline{\Omega}_b$ where $b = 0.01 \operatorname{diag}(\Omega)$, the order of convergence being at least[7] 1.8. A typical example of the exact and numerical solutions is illustrated in Figure 8.7. The grid construction follows exactly the procedure 8.5.1, such grids are illustrated in Figure 8.8.

Multi-scale problem

In this section, we show that the proposed discretization also allows us to deal with multi-scale problems using highly graded quad-tree grids. In order to illustrate that feature, we solve the same problem as 8.5.2 except that the domain is now $\Omega = [-100,\, 100] \times [-100,\, 100]$, and the points $\overline{\boldsymbol{x}}_k$ are located randomly in

$$\Lambda = [\mu - 0.5,\, \mu + 0.5] \times [\nu - 0.5,\, \nu + 0.5]\,,$$

[7]Note that the order of convergence gets closer and closer to 2 as the grid is refined. Since the coarsest considered grids barely capture the solution's behavior, the solution enters the asymptotic convergence regime only after a few grid refinement, which accounts for this observation.

Error analysis for solving a problem like 8.5.2 with $N = 20$.

Grid level	$\frac{\|u-u_h\|_\infty}{\|u\|_\infty}$	rate	$\frac{\left\|u_x-u_{h,x}\right\|_\infty}{\|u_x\|_\infty}$	rate	$\frac{\left\|u_y-u_{h,y}\right\|_\infty}{\|u_y\|_\infty}$	rate
0/7	1.073e-02		9.585e-02		1.380e-01	
1/8	4.768e-03	1.170	4.389e-02	1.127	6.250e-02	1.143
2/9	1.171e-03	2.025	8.727e-03	2.330	1.182e-02	2.402
3/10	3.450e-04	1.764	2.587e-03	1.754	3.126e-03	1.919
4/11	7.614e-05	2.180	5.859e-04	2.143	6.355e-04	2.298
5/12	2.067e-05	1.881	1.459e-04	2.006	2.453e-04	1.373
6/13	5.563e-06	1.893	3.892e-05	1.906	6.926e-05	1.824
7/14	1.298e-06	2.100	1.062e-05	1.874	1.870e-05	1.889
Mean order		1.859		1.877		1.836

Grid level	$\frac{\|u-u_h\|_1}{\|u\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\left\|u_x-u_{h,x}\right\|_1}{\|u_x\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\left\|u_y-u_{h,y}\right\|_1}{\|u_y\|_\infty \mathrm{vol}(\Omega)}$	rate
0/7	1.010e-03		1.526e-03		1.973e-03	
1/8	4.781e-04	1.079	4.644e-04	1.717	5.651e-04	1.804
2/9	1.799e-04	1.411	1.339e-04	1.794	1.610e-04	1.811
3/10	4.884e-05	1.881	3.747e-05	1.838	4.334e-05	1.894
4/11	1.293e-05	1.918	1.013e-05	1.887	1.075e-05	2.012
5/12	3.380e-06	1.935	2.515e-06	2.010	2.804e-06	1.938
6/13	8.549e-07	1.983	6.499e-07	1.952	7.045e-07	1.993
7/14	2.151e-07	1.991	1.656e-07	1.972	1.790e-07	1.977
Mean order		1.742		1.881		1.918

Table 8.5: accuracy analysis of the Poisson problem 8.5.2: one trial for $N = 20$ random source locations $\overline{\boldsymbol{x}}_k$. All norms are in $\overline{\Omega}_b$ where $b = 0.01\,\mathrm{diag}\,(\Omega)$.

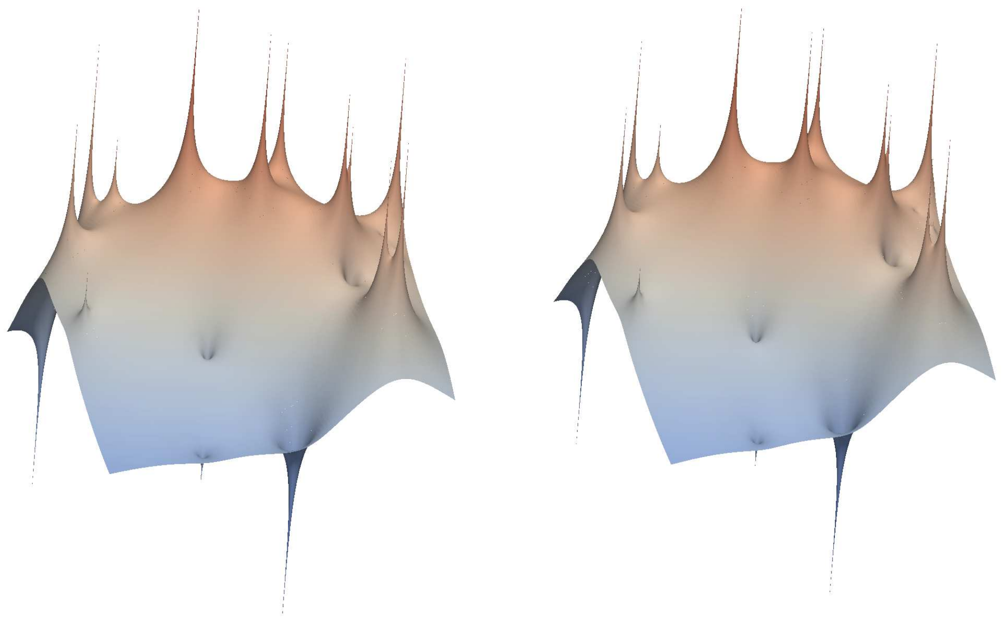


Figure 8.7: exact (left) and numerical (right) solutions for a problem of type 8.5.2.

μ and ν being random variables uniformly distributed in $[-99.5, 99.5]$. Assuming that we are interested only in the solution further than a distance of $\operatorname{diag}(\Omega)/30000$ from the sources, we need a local grid level of at least $\lceil \log_2(30000) \rceil = 15$ which would be intractable and prohibitively expensive in terms of computational resources with uniform grids since it would lead to $2^{30} \sim 10^9$ unknowns. On the other hand, the construction procedure 8.5.1 leads to a system of 3116 unknowns for the 0/15 grid in this case, yet producing results that are about 1% accurate as indicated in Table 8.6. The order of convergence for the solution and its gradient is at least 1.8 in this case too. The 2/17 grid is illustrated in Figure 8.9.

8.5.3 Numerical examples in three spatial dimensions

In this section, we show that the discrete formula of δ can be applied to solving Poisson equations with point-located singular source terms, in three spatial dimensions. Superlinear convergence of the resulting solutions *and* its gradient in the L^1- and L^∞-

Error analysis for solving a problem like 8.5.2 with $N = 10$ sources

Grid level	$\frac{\|u-u_h\|_\infty}{\|u\|_\infty}$	rate	$\frac{\left\|u_x-u_{h,x}\right\|_\infty}{\|u_x\|_\infty}$	rate	$\frac{\left\|u_y-u_{h,y}\right\|_\infty}{\|u_y\|_\infty}$	rate
0/15	1.059e-02		1.732e-01		1.537e-01	
1/16	5.290e-03	1.001	5.218e-02	1.731	5.795e-02	1.407
2/17	1.283e-03	2.044	1.676e-02	1.639	1.449e-02	1.999
3/18	3.317e-04	1.952	4.202e-03	1.996	3.619e-03	2.002
4/19	8.348e-05	1.991	8.031e-04	2.387	1.033e-03	1.809
5/20	2.261e-05	1.884	2.508e-04	1.679	2.606e-04	1.986
6/21	6.624e-06	1.771	7.074e-05	1.826	5.250e-05	2.312
7/22	1.524e-06	2.120	1.764e-05	2.003	1.768e-05	1.571
Mean order		1.823		1.894		1.869

Grid level	$\frac{\|u-u_h\|_1}{\|u\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\left\|u_x-u_{h,x}\right\|_1}{\|u_x\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\left\|u_y-u_{h,y}\right\|_1}{\|u_y\|_\infty \mathrm{vol}(\Omega)}$	rate
0/15	2.304e-04		5.116e-07		5.056e-07	
1/16	1.521e-04	0.599	2.495e-07	1.036	2.320e-07	1.124
2/17	3.177e-05	2.259	5.578e-08	2.161	5.117e-08	2.181
3/18	6.157e-06	2.367	1.270e-08	2.135	1.189e-08	2.105
4/19	2.122e-06	1.537	3.831e-09	1.729	3.542e-09	1.748
5/20	5.633e-07	1.913	1.004e-09	1.932	9.280e-10	1.932
6/21	1.491e-07	1.918	2.622e-10	1.936	2.414e-10	1.943
7/22	3.734e-08	1.997	6.579e-11	1.995	6.065e-11	1.993
Mean order		1.799		1.846		1.861

Table 8.6: accuracy analysis of the Poisson problem 8.5.2: one trial with $N = 10$ sources randomly located. All norms are in $\overline{\Omega}_b$ where $b = \mathrm{diag}\,(\Omega)\,/30000$.

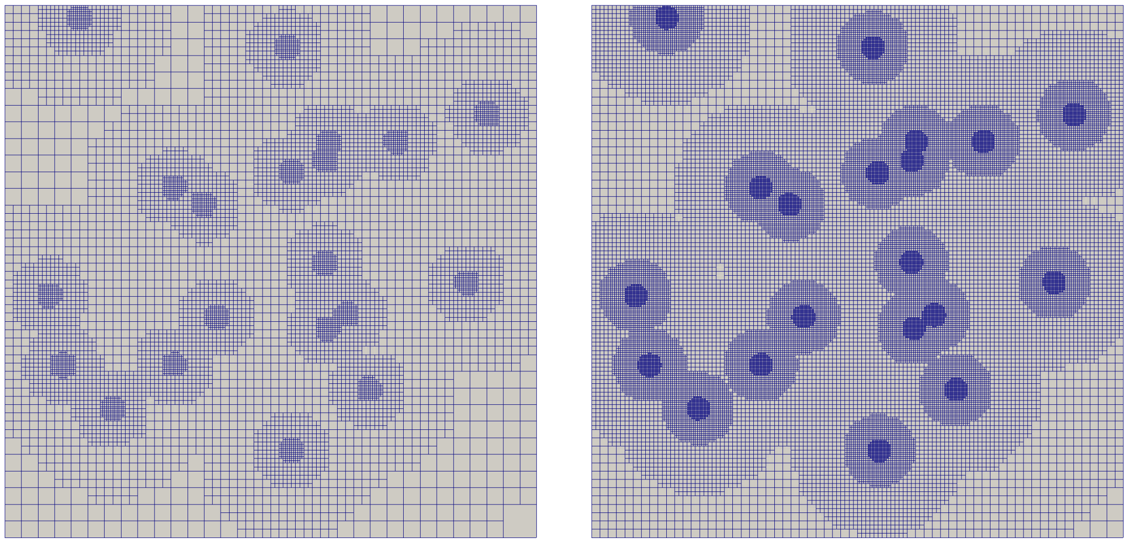


Figure 8.8: grids generated with the refinement criterion 8.5.1 for problem 8.5.2. Left: 1/8 grid; right: 2/9 grid.

norms is observed and discussed.

One singular source term

Consider the Poisson equation

$$
\begin{aligned}
-\Delta u(\boldsymbol{x}) &= 4\pi \operatorname{diag}(\Omega)\, \delta(\boldsymbol{x}-\overline{\boldsymbol{x}}) \quad \in \Omega \\
u(\boldsymbol{x}) &= 1 + \frac{\operatorname{diag}(\Omega)}{|\boldsymbol{x}-\overline{\boldsymbol{x}}|} \quad \text{on } \partial\Omega
\end{aligned}
$$

where $\Omega = [-1,\, 1]^3$ and $\operatorname{diag}(\Omega) = 2\sqrt{3}$ can be seen as an arbitrary length chosen to make u non-dimensional. The exact solution is $u(\boldsymbol{x}) = 1 + \dfrac{\operatorname{diag}(\Omega)}{|\boldsymbol{x}-\overline{\boldsymbol{x}}|}$ in $\Omega \backslash \{\overline{\boldsymbol{x}}\}$, thus singular at $\overline{\boldsymbol{x}}$.

This equation is solved using the above discretization for the multidimensional delta function. Since the analytic solution itself is singular at $\overline{\boldsymbol{x}}$, we define

$$
\overline{\Omega}_\alpha = \{\boldsymbol{x} : \boldsymbol{x} \in \Omega, |\boldsymbol{x}-\overline{\boldsymbol{x}}| \geq \alpha\}
$$

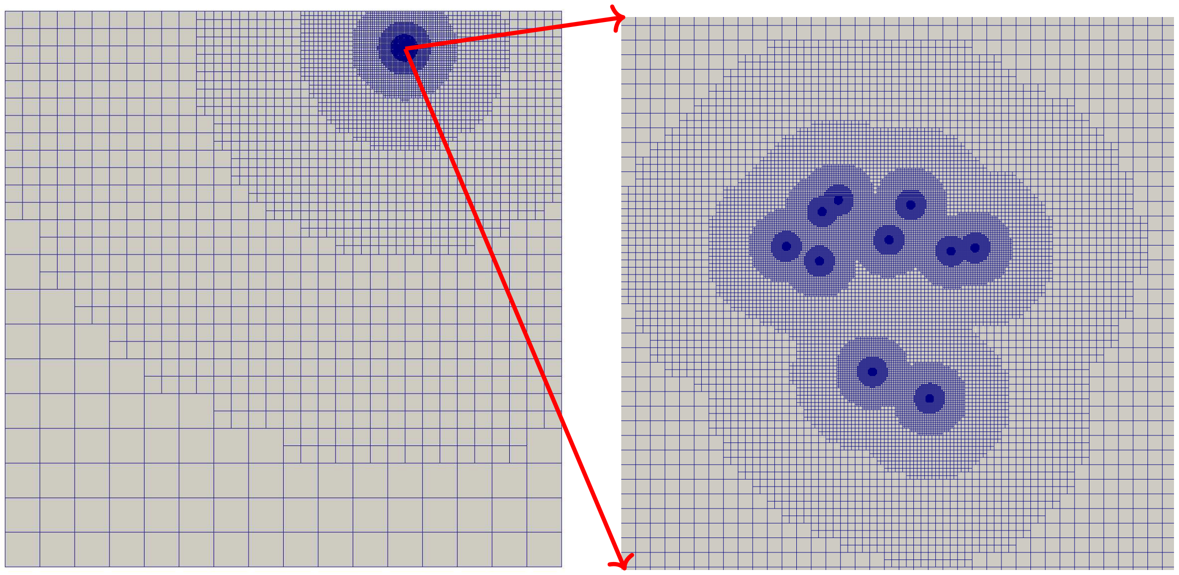


Figure 8.9: left: highly graded (2/17) grid constructed to solve problem 8.5.2. Right: enlarged image of the region including the sources points.

($\alpha \in \left]0, \operatorname{diag}(\Omega)\right]$) for the error analysis. Table 8.7 shows that our discretization produces results that converge in the L^1-norm and in the L^∞-norm in $\overline{\Omega}_{0.01\,\operatorname{diag}(\Omega)}$. Fine grid resolutions beyond level 11 cannot be investigated because of memory limitations but results from Table 8.7 seem to indicate that the order of convergence is at least 1.8.

The Octree grid is constructed in a way that is totally similar to the procedure 8.5.1. The only major difference is that the local truncation error $\tau\left(\Delta x, \Delta y, \Delta z\right)$ associated with a uniform grid of spacing $\left(\Delta x, \Delta y, \Delta z\right)$ behaves now as

$$\tau\left(\Delta x, \Delta y, \Delta z\right) \sim \mathcal{O}\left(\frac{\left(\Delta x\right)^2 + \left(\Delta y\right)^2 + \left(\Delta z\right)^2}{r^5}\right) \tag{8.19}$$

where r is the distance to the source location $\overline{\boldsymbol{x}}$. Therefore, we adapt the procedure 8.5.1 in the following way

Error analysis for a problem like 8.5.3. $b = 0.01\,\mathrm{diag}\,(\Omega)$

Grid level	$\frac{\\|u-u_h\\|_\infty}{\\|u\\|_\infty}$	rate	$\frac{\\|u_x-u_{h,x}\\|_\infty}{\\|u_x\\|_\infty}$	rate	$\frac{\\|u_y-u_{h,y}\\|_\infty}{\\|u_y\\|_\infty}$	rate	$\frac{\\|u_z-u_{h,z}\\|_\infty}{\\|u_z\\|_\infty}$	rate
0/7	5.74e-02		2.86e-01		2.02e-01		3.03e-01	
1/8	2.46e-02	1.22	8.62e-02	1.73	1.51e-01	0.42	7.29e-02	2.05
2/9	5.24e-03	2.23	2.45e-02	1.82	2.00e-02	2.92	2.23e-02	1.71
3/10	1.27e-03	2.04	4.88e-03	2.33	6.20e-03	1.69	6.56e-03	1.77
4/11	3.16e-04	2.01	1.37e-03	1.84	1.55e-03	2.00	1.43e-03	2.20
Mean order		1.88		1.93		1.76		1.93

Grid level	$\frac{\\|u-u_h\\|_1}{\\|u\\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\\|u_x-u_{h,x}\\|_1}{\\|u_x\\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\\|u_y-u_{h,y}\\|_1}{\\|u_y\\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\\|u_z-u_{h,z}\\|_1}{\\|u_z\\|_\infty \mathrm{vol}(\Omega)}$	rate
0/7	4.09e-05		1.89e-05		1.86e-05		2.04e-05	
1/8	2.15e-05	0.93	5.38e-06	1.81	5.67e-06	1.71	5.94e-06	1.78
2/9	6.80e-06	1.66	1.59e-06	1.76	1.48e-06	1.93	1.80e-06	1.72
3/10	2.02e-06	1.75	4.10e-07	1.96	4.23e-07	1.81	4.87e-07	1.88
4/11	5.59e-07	1.85	1.11e-07	1.88	1.13e-07	1.91	1.27e-07	1.94
Mean order		1.55		1.85		1.84		1.83

Table 8.7: accuracy analysis of the Poisson problem 8.5.3. One trial for a random source $\overline{\boldsymbol{x}}$ in Ω. All norms are in $\overline{\Omega}_b$ where $b = 0.01\,\mathrm{diag}\,(\Omega)$.

Construction procedure 8.5.2.

Construction of an m/M Octree grid.

Given a minimal distance of interest d to the Dirac-distributed sources and the dimensions L_x, L_y, L_z of the computational domain Ω, define the threshold distances t_k, $k = 0, \ldots, M-m-1$ such that $t_{M-m-1} = d\,2^{2/5}$ and

$$t_{M-m-1-k} = \max\left(2^{2/5}\,t_{M-m-k},\ t_{M-m-k} + \frac{\sqrt{L_x^2+L_y^2+L_z^2}\,2^{k+1}}{2^{M-m}}\right),$$

$k = 1, 2, \ldots, M-m-1$.

Starting from a root cell of dimensions L_x, L_y, L_z, split it m times.

Then, let an integer k run from 0 to $M-m-1$ and split any cell C if

$$\min_{v \in \mathrm{vertices}(C)} \phi(v) \le t_k$$

where v refers to a vertex (node) of the cell C and ϕ is the smallest distance to a Dirac-distributed source.

to produce a grid such that the local truncation error τ is comparable from one grid-level region to another. If there exists a distance below which the numerical solution is irrelevant for the considered application, it makes sense to set d equal to that distance. For the results in Table 8.7, we chose $d = b = 0.01\,\mathrm{diag}\,(\Omega)$ consistently with the error analysis in $\overline{\Omega}_b$.

One singular source term with variable diffusion coefficient

Three-dimensional problems involving variable diffusion coefficients can be addressed alike. Consider for instance the diffusion coefficient

$$\beta\left(\boldsymbol{x}\right) = 2\left(\frac{\mathrm{diag}\left(\Omega\right)}{\left|\boldsymbol{x}-\overline{\boldsymbol{x}}\right|}\right)^{\frac{3}{2}}, \quad \boldsymbol{x} \in \Omega\backslash\left\{\overline{\boldsymbol{x}}\right\}$$

for some randomly located $\overline{\boldsymbol{x}} \in \Omega$ and the corresponding variable diffusion coefficient Poisson equation

$$\begin{aligned} -\nabla\cdot\left(\beta\nabla u\right) &= 4\pi\,\mathrm{diag}\left(\Omega\right)\delta\left(\boldsymbol{x}-\overline{\boldsymbol{x}}\right) \quad &&\in \Omega\backslash\left\{\overline{\boldsymbol{x}}\right\} \\ u\left(\boldsymbol{x}\right) &= 1-\sqrt{\frac{\left|\boldsymbol{x}-\overline{\boldsymbol{x}}\right|}{\mathrm{diag}\left(\Omega\right)}} \quad &&\text{on } \partial\Omega \end{aligned}$$

where Ω and $\mathrm{diag}\left(\Omega\right)$ are as before[8]. The exact solution is $u\left(\boldsymbol{x}\right) = 1-\sqrt{\left|\boldsymbol{x}-\overline{\boldsymbol{x}}\right|/\mathrm{diag}\left(\Omega\right)}$ (note that it is bounded in Ω).

This problem is solved using the above discretization for the multidimensional delta function. Since the exact solution has unbounded derivatives close to $\overline{\boldsymbol{x}}$, convergence in the L^∞-norm is not ensured in the entire domain Ω. However, Table 8.8 shows that the numerical approximation still converges in the L^∞-norm in $\overline{\Omega}_{0.01\mathrm{diag}(\Omega)}$ (as defined in section 8.5.3) *and* in the L^1-norm in the *entire* domain Ω. Fine grid resolutions beyond

[8]The components of the random source location $\overline{\boldsymbol{x}}$ are defined as $\frac{\pi}{4}\frac{q}{Q}$ where $Q = 1024$ and q is a random integer in $[-Q, Q]$, so that $\overline{\boldsymbol{x}}$ cannot coincide with a grid node and the grid values of β are well-defined at the grid nodes.

Error analysis for solving a problem like 8.5.3.

Grid level	$\frac{\Vert u-u_h\Vert_\infty}{\Vert u\Vert_\infty}$	rate	$\frac{\Vert u_x-u_{h,x}\Vert_\infty}{\Vert u_x\Vert_\infty}$	rate	$\frac{\Vert u_y-u_{h,y}\Vert_\infty}{\Vert u_y\Vert_\infty}$	rate	$\frac{\Vert u_z-u_{h,z}\Vert_\infty}{\Vert u_z\Vert_\infty}$	rate
0/7	5.55e-03		2.48e-02		2.09e-02		2.25e-02	
1/8	1.11e-03	2.32	7.10e-03	1.80	6.21e-03	1.75	6.67e-03	1.76
2/9	3.99e-04	1.48	2.19e-03	1.70	1.99e-03	1.64	2.11e-03	1.66
3/10	9.10e-05	2.13	6.07e-04	1.85	5.47e-04	1.86	5.84e-04	1.86
4/11	1.73e-05	2.39	1.25e-04	2.28	1.11e-04	2.30	1.21e-04	2.27
Mean order		2.08		1.91		1.89		1.89

Grid level	$\frac{\Vert u-u_h\Vert_1}{\Vert u\Vert_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\Vert u_x-u_{h,x}\Vert_1}{\Vert u_x\Vert_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\Vert u_y-u_{h,y}\Vert_1}{\Vert u_y\Vert_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\Vert u_z-u_{h,z}\Vert_1}{\Vert u_z\Vert_\infty \mathrm{vol}(\Omega)}$	rate
0/7	2.07e-04		1.62e-04		1.59e-04		1.69e-04	
1/8	6.78e-05	1.61	5.24e-05	1.63	5.20e-05	1.61	5.55e-05	1.61
2/9	1.99e-05	1.77	1.57e-05	1.74	1.47e-05	1.82	1.57e-05	1.82
3/10	5.50e-06	1.85	4.28e-06	1.87	3.96e-06	1.89	4.28e-06	1.88
4/11	1.45e-06	1.93	1.13e-06	1.92	1.05e-06	1.93	1.13e-06	1.92
Mean order		1.79		1.79		1.81		1.81

Table 8.8: accuracy analysis of the variable diffusion coefficient Poisson problem 8.5.3: one trial for a random source location $\overline{\boldsymbol{x}}$. L^∞ norms are in $\overline{\Omega}_{0.01\,\mathrm{diag}(\Omega)}$ and L^1-norms are in the entire domain Ω. The grid construction follows the procedure 8.5.2 with $d = 0.01\mathrm{diag}\,(\Omega)$.

level 11 cannot be investigated because of memory limitations but results from Table 8.8 seem to indicate that the order of convergence is about 1.8-1.9.

Multiple singular source terms

Consider now the Poisson equation ($N \in \mathbb{N}_0$)

$$-\Delta u\,(\boldsymbol{x}) = \sum_{k=1}^{N} \alpha_k\,\delta\,(\boldsymbol{x} - \overline{\boldsymbol{x}}_k) \quad \in \Omega$$

$$u\,(\boldsymbol{x}) = 1 + \sum_{k=1}^{N} \frac{\alpha_k}{4\pi\,|\boldsymbol{x} - \overline{\boldsymbol{x}}_k|} \quad \text{on } \partial\Omega$$

where Ω and $\operatorname{diag}(\Omega)$ are as in section 8.5.3. In this problem, the points $\overline{\boldsymbol{x}}_k$ are randomly located in Ω and the intensities α_k are random variables with uniform probability density in $[-8\pi \operatorname{diag}(\Omega), 8\pi \operatorname{diag}(\Omega)]$. The exact solution is

$$u(\boldsymbol{x}) = 1 + \sum_{k=1}^{N} \frac{\alpha_k}{4\pi \, |\boldsymbol{x} - \overline{\boldsymbol{x}}_k|}$$

in $\Omega \backslash \bigcup_{k=1}^{N} \{\overline{\boldsymbol{x}}_k\}$, hence singular at points $\overline{\boldsymbol{x}}_k$ $(k = 1, \ldots, N)$.

It is solved using the above discretization for the multidimensional delta function. Since the analytic solution itself is singular at $\overline{\boldsymbol{x}}_k$, we define

$$\overline{\Omega}_\alpha = \{\boldsymbol{x} : \boldsymbol{x} \in \Omega, |\boldsymbol{x} - \overline{\boldsymbol{x}}_k| \geq \alpha, \, \forall k = 1, \ldots, N\}$$

($\alpha \in \,]0, \operatorname{diag}(\Omega)]$) for the error analysis. Table 8.9 shows that our discretization produces results that converge in the L^1-norm and in the L^∞-norm in $\overline{\Omega}_b$ where $b = 0.01 \operatorname{diag}(\Omega)$. Fine grid resolutions beyond level 10 cannot be investigated because of memory limitations but results from Table 8.8 seem to indicate that the order of convergence is at least 1.8. The Octree grid, the exact and numerical solutions are illustrated in Figure 8.10. The grid construction follows exactly the procedure 8.5.2, ϕ being now the distance to the closest point $\overline{\boldsymbol{x}}_k$.

Multi-scale problem

In this section, we show that the proposed discretization also allows us to deal with multi-scale problems using highly graded Octree grids. In order to illustrate that feature, we solve the same problem as 8.5.3 except that the domain is now $\Omega = [-100, \, 100]^3$, and

Error analysis for a problem like 8.5.3

Grid level	$\frac{\lVert u-u_h\rVert_\infty}{\lVert u\rVert_\infty}$	rate	$\frac{\lVert u_x-u_{h,x}\rVert_\infty}{\lVert u_x\rVert_\infty}$	rate	$\frac{\lVert u_y-u_{h,y}\rVert_\infty}{\lVert u_y\rVert_\infty}$	rate	$\frac{\lVert u_z-u_{h,z}\rVert_\infty}{\lVert u_z\rVert_\infty}$	rate
0/7	7.77e-02		4.94e-01		2.61e-01		4.26e-01	
1/8	1.97e-02	1.98	1.29e-01	1.94	1.31e-01	0.99	8.41e-02	2.34
2/9	4.68e-03	2.07	2.28e-02	2.50	2.48e-02	2.40	2.70e-02	1.64
3/10	1.34e-03	1.81	5.83e-03	1.97	6.63e-03	1.91	6.90e-03	1.97
Mean order		1.95		2.14		1.77		1.98

Grid level	$\frac{\lVert u-u_h\rVert_1}{\lVert u\rVert_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\lVert u_x-u_{h,x}\rVert_1}{\lVert u_x\rVert_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\lVert u_y-u_{h,y}\rVert_1}{\lVert u_y\rVert_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\lVert u_z-u_{h,z}\rVert_1}{\lVert u_z\rVert_\infty \mathrm{vol}(\Omega)}$	rate
0/7	1.69e-04		1.05e-04		1.05e-04		1.02e-04	
1/8	8.62e-05	0.98	3.24e-05	1.69	3.31e-05	1.66	3.25e-05	1.65
2/9	3.00e-05	1.52	9.43e-06	1.78	9.68e-06	1.77	9.66e-06	1.75
3/10	8.96e-06	1.74	2.66e-06	1.83	2.65e-06	1.87	2.64e-06	1.87
Mean order		1.41		1.77		1.77		1.76

Table 8.9: accuracy analysis of the Poisson problem 8.5.3. One test for $N = 20$ sources $\overline{\boldsymbol{x}}_k$ randomly located in Ω. All norms are in $\overline{\Omega}_b$ where $b = 0.01\,\mathrm{diag}\,(\Omega)$.

the points $\overline{\boldsymbol{x}}_k$ are located randomly in

$$\Lambda = [\xi - 0.5,\, \xi + 0.5] \times [\nu - 0.5,\, \nu + 0.5] \times [\mu - 0.5,\, \mu + 0.5]\,,$$

ξ, ν and μ being random variables uniformly distributed in $[-99.5,\, 99.5]$. Assuming that we are interested only in the solution further than a distance of $\mathrm{diag}\,(\Omega)\,/30000 \simeq 0.01$ from the sources, we need a local grid level of at least $\lceil \log_2 (30000) \rceil = 15$ which would be intractable and prohibitively expensive in terms of computational resources with uniform grids since it would lead to $2^{45} \sim 35\,10^{12}$ unknowns. In comparison, the construction procedure 8.5.2 leads to a system of about 72000 unknowns for the 0/15 grid in this case, yet producing results that are about 10% accurate as indicated in Table 8.10. Fine grid resolutions beyond level 18 cannot be investigated because of memory limitations but results from Table 8.8 seem to indicate superlinear convergence with an order of convergence that is at least 1.5.

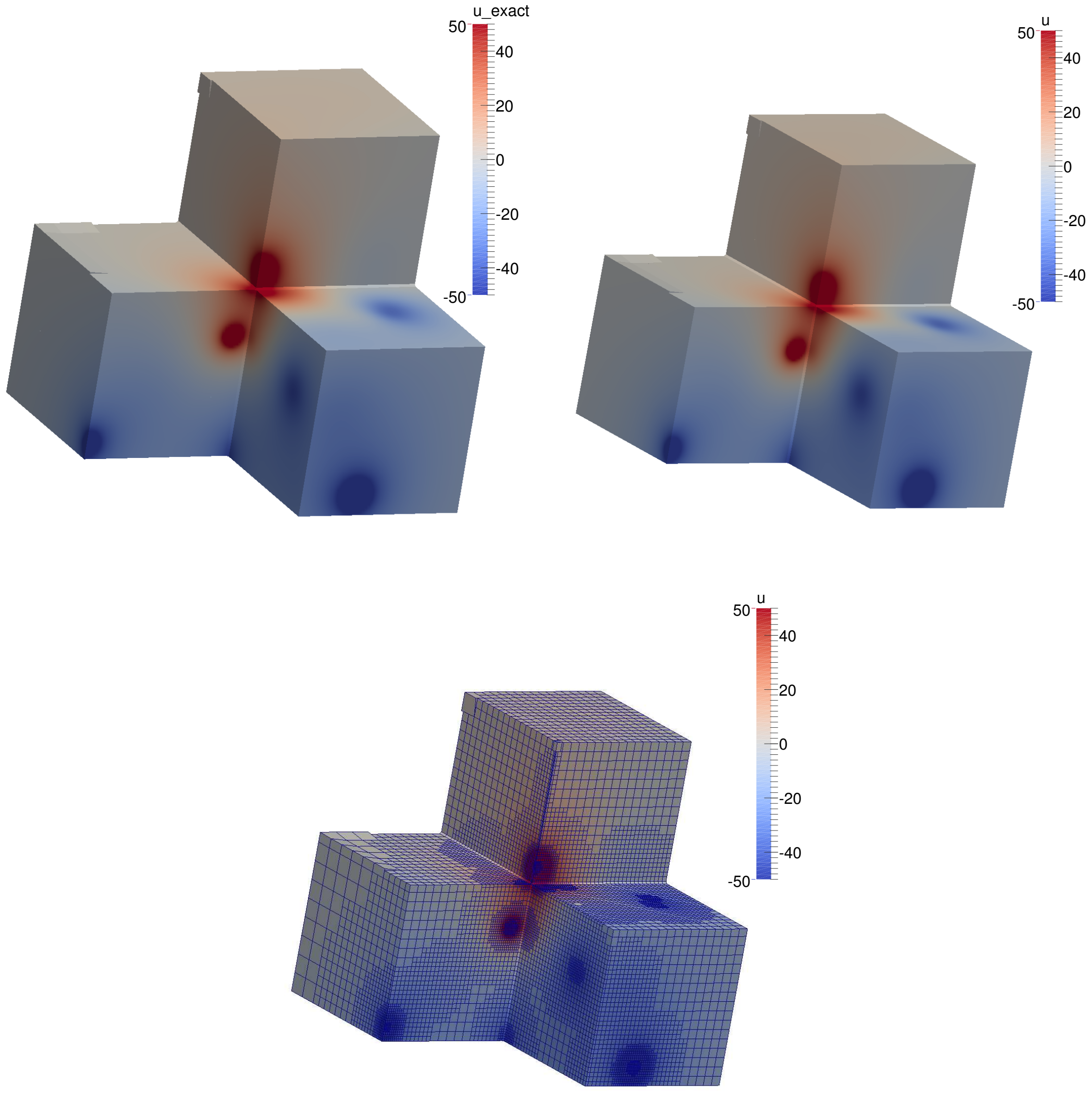


Figure 8.10: top: color map (restricted to $[-50, 50]$) of the exact (left) and numerical (right) solutions for a problem of type 8.5.3. Bottom: color map superimposed onto the Octree grid. The three regions $\Sigma_1 = \{\boldsymbol{x} \in \Omega | \, \boldsymbol{x} \cdot \boldsymbol{e}_1 < 0 \text{ and } \boldsymbol{x} \cdot \boldsymbol{e}_3 < 0\}$, $\Sigma_2 = \{\boldsymbol{x} \in \Omega | \, \boldsymbol{x} \cdot \boldsymbol{e}_2 > 0 \text{ and } \boldsymbol{x} \cdot \boldsymbol{e}_3 < 0\}$ and $\Sigma_3 = \{\boldsymbol{x} \in \Omega | \, \boldsymbol{x} \cdot \boldsymbol{e}_1 < 0 \text{ and } \boldsymbol{x} \cdot \boldsymbol{e}_2 > 0\}$ have been truncated for visualization purposes.

Error analysis for solving a problem like 8.5.3 with $N = 10$ sources.

Grid level	$\frac{\|u-u_h\|_\infty}{\|u\|_\infty}$	rate	$\frac{\|u_x-u_{h,x}\|_\infty}{\|u_x\|_\infty}$	rate	$\frac{\|u_y-u_{h,y}\|_\infty}{\|u_y\|_\infty}$	rate	$\frac{\|u_z-u_{h,z}\|_\infty}{\|u_z\|_\infty}$	rate
0/15	1.04e-01		5.15e-01		3.14e-01		5.36e-01	
1/16	2.92e-02	1.83	1.71e-01	1.59	1.30e-01	1.27	1.75e-01	1.61
2/17	7.45e-03	1.97	3.30e-02	2.37	3.63e-02	1.84	3.83e-02	2.19
3/18	1.95e-03	1.94	8.63e-03	1.93	8.66e-03	2.07	8.16e-03	2.23
Mean order		1.91		1.97		1.73		2.01
Grid level	$\frac{\|u-u_h\|_1}{\|u\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\|u_x-u_{h,x}\|_1}{\|u_x\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\|u_y-u_{h,y}\|_1}{\|u_y\|_\infty \mathrm{vol}(\Omega)}$	rate	$\frac{\|u_z-u_{h,z}\|_1}{\|u_z\|_\infty \mathrm{vol}(\Omega)}$	rate
0/15	1.35e-07		1.24e-10		1.57e-10		1.26e-10	
1/16	1.07e-07	0.34	5.14e-11	1.27	7.54e-11	1.05	5.25e-11	1.26
2/17	2.93e-08	1.86	1.44e-11	1.83	2.15e-11	1.81	1.46e-11	1.84
3/18	9.04e-09	1.70	4.33e-12	1.74	6.34e-12	1.76	4.36e-12	1.74
Mean order		1.30		1.61		1.54		1.62

Table 8.10: accuracy analysis of the Poisson problem 8.5.3. One test for $N = 10$ sources randomly located in a cube of side length 1 within a $[-100;\, 100]^3$ domain. All norms are in $\overline{\Omega}_b$ where $b = \mathrm{diag}\,(\Omega)\,/30000 \simeq 0.01$.

8.6 Summary

We have introduced a second-order accurate discretization of the multidimensional Dirac delta distribution in two and three spatial dimensions. We have shown its accuracy through several numerical examples, using benchmark test cases. We have shown that using the proposed discretization of the multidimensional Dirac delta distribution for solving the variable or constant coefficient Poisson equation with singular source terms leads to (close to) second-order accurate solutions in the L^1-norm and in the L^∞-norm away from some support of the singular sources, or even in the entire domain when the solution is bounded everywhere. This method is consistent, in the discrete sense, with the discretizations for codimension one interface delta functions that are presented in [116, 152]. We have also proposed appropriate procedures to construct non-uniform quadtree/octree grids for such applications in a way that does not impede convergence, yet enables the numerical solution of multi-scale problems.

Chapter 9

Fast and scalable algorithms for constructing Solvent-Excluded Surfaces of large biomolecules

9.1 Introduction

Modeling solute-solvent interactions stands as a cornerstone within the plethora of algorithmic and numerical challenges associated with computational biomolecular physics: these interactions play a fundamental role in the estimation of free energies related to chemical processes, see e.g. [222, 223]. Explicit solvent models, *i.e.*, treating solvent molecules explicitly and resolving their fundamental dynamics, are believed to give the most detailed description of such interactions. However, their computational cost quickly becomes intractable for large-scale systems. On the other hand, implicit solvent models treat the solvent as a continuum medium and capture the solvent-solute interactions through a classical electrostatic scalar field Ψ. Implicit solvent models have been used successfully and showed to be satisfactory in the last decades [224, 225, 226, 227].

Mathematically, the electrostatic potential Ψ is a scalar field that obeys an exponentially nonlinear Poisson-Boltzmann Equation (PBE), in the context of implicit models allowing mobile ions in the solvent. The permittivity ϵ is discontinuous across the solvent-solute interface, with a discontinuity of at least one order of magnitude. Dirac-distributed source terms model (partial) atom electric charges. Finally, specific interface *jump conditions* at the solute-solvent interface impose the continuity of Ψ and $\epsilon \boldsymbol{n} \cdot \nabla \Psi$ across the interface, where $\boldsymbol{n}$ is the vector normal to the interface.

Starting from the late 1980s, several advances have been made to address this problem numerically, either using Finite Element/Volume (FE/FV) methods like `APBS` [228, 229, 230], Finite Difference (FD) methods [231, 232, 233, 234, 235], or even, more recently, Boundary Integral (BI) methods [236, 237, 238]. The linearized version of PB, refered to as Debye-Huckel theory, has been considered in some of these works; however, the model becomes accurate only when the solution ionic strength approaches zero, a contradiction with typical biochemical applications that involve relatively high ionic strengths. We refer the interested reader to the reviews [239, 240] for more details about the PBE and related numerical aspects.

Because of their ability to enforce the continuity conditions at the interface and their natural capability for adaptive refinement, possibly with *a priori* error estimates, the FE/FV techniques soon outperformed the FD approach and thus became methods of choice. Indeed, original FD attempts were constrained to uniform grids; moreover, grid points were simply marked as *solvent-accessible* or not, and the same discretization stencil was applied throughout the computational domain, disregarding any conditions at the solute-solvent interface [241, 242]. From a numerical point of view, such a strategy leads to poor performances because of undesirable grid-size-dependent numerical smearing of the interface, although sometimes supported *qualitatively* in the literature by overlap of wavefunctions in a quantum mechanics sense [243]. However, more recent developments in FD techniques have addressed the issues of the continuity conditions at the solute-solvent interface [140, 244, 159] and/or adaptivity [245, 206, 246]. Moreover, typical finite element approaches rely on body-fitted meshes that are not trivial to construct in general, especially when considering the constant remeshing needed to study folding of molecules. Finite differences, on the other hand, treat such cases straightforwardly so that there is nowadays a clear advantage to use this strategy. The level-set method introduced by Osher and Sethian [63] stands as a linchpin in this context: relevant interfaces are implicitly represented by the zero-level set of a function ϕ. We refer the interested reader to [247] for a recent review of the capabilities of level-set methods.

When adaptivity comes into play, the ability to construct an adaptive mesh efficiently in a distributed computing framework is a key feature for efficient and/or large-scale numerical simulations. Such a mesh may be built upon a triangulation of the solvent-solute interface, which can be constructed efficiently by optimized tools like the `MSMS` [248] (although unable to process very large molecules, see [249]) or the `EDTSurf` [250] sequential softwares. In [251], the authors present another sequential strategy that struggles with molecules having more than 200 atoms and/or regions of high curvature. In [252], a mesh is built upon the triangulation of an alternative, approximated definition of the solvent-solute interface, defined as a level-set of a sum of Gaussian functions associated with all atoms: no equivalence with the actual molecular surface or accuracy analysis is showed to validate the technique. Finally, another meshing technique is used in `APBS` [230]. It was successfully implemented on parallel architectures [253, 254] although partitioning the grid leads to computational overheads: smaller problems, on a coarse mesh, need to be solved by every single process in order to determine the domain partitions, which slows down the procedure and impedes its scalability.

On the contrary, FD methods in a level-set framework allow grid lines and grid cells to be crossed arbitrarily by the interface, where special treatment and vigilant care in the implementation is required. Among all conceptual tools available in the level-set framework, the *reinitialization* [5, 184, 106, 255] plays an essential role, as it allows the calculation of a signed distance to the interface, while handling topological difficulties inherently, in its own mathematical formulation. Moreover, those techniques have proved scalable and capable of capturing highly convoluted surfaces [256, 247].

In this chapter, we present a fast and scalable algorithm to create an implicit representation of the *Solvent-Excluded Surface* Γ_{SES} (also dubbed *Molecular Surface* in the literature) of a biomolecule using the zero-level set of a function $\overline{\phi}$ and to build a corresponding adaptive Cartesian Octree grid, of aspect ratio 1. The latter is represented as a set of distributed adaptive octree, as implemented in the open-source `p4est` library [174, 103]. We note that the algorithms presented in this work have been optimized for the above purpose. When considering the (area of the) *Solvent-Accessible Surface only, other efficient and optimized approaches may be better suited [257, 258].*

Our refinement criterion is based on the local distance to Γ_{SES}: a grid cell C is refined if

$$\min_{\boldsymbol{v}\in\mathcal{V}(C)} \operatorname{dist}\left(\boldsymbol{v}, \Gamma_{\text{SES}}\right) \leq \frac{L \operatorname{diag}\left(C\right)}{2}, \tag{9.1}$$

where $\mathcal{V}\left(C\right)$ denotes the set of all vertices of cell C, the proportionality constant $L \geq 1$ is a free parameter determining the thickness of the computational grid, and $\operatorname{diag}\left(C\right)$ is the diagonal of cell C. If $L = 1$, this criterion ensures that cells of diagonal length d have all of their vertices farther than $d/2$ from the interface. The corresponding coarsening criterion is

$$\min_{\boldsymbol{v}\in\mathcal{V}(C)} \operatorname{dist}\left(\boldsymbol{v}, \Gamma_{\text{SES}}\right) \geq L \operatorname{diag}\left(C\right), \tag{9.2}$$

and a family of children cells of a same parent cell are merged together if they all satisfy the latter criterion.

We first describe the three relevant surfaces in biomolecular applications: the *van* *der Waals Surface*, the *Solvent-Accessible Surface* and the *Solvent-Excluded Surface*. A useful unifying theorem is introduced and proved. The algorithms are then presented and detailed. Results and illustrations are presented along with a discussion of accuracy and scalability analyses. Two-dimensional illustrations are used throughout the chapter but the reasoning, the implementation and the general structure of the algorithm are fully three-dimensional.

9.2 Three relevant surfaces

Implicit solvent models usually represent a molecule $\mathcal{M}$ as a list of its $n_{\mathcal{M}}$ constituent atoms[1] modeled as balls of center $\boldsymbol{c}_i \in \mathbb{R}^3$ and of van der Waals radius r_i ($i = 1, \ldots, n_{\mathcal{M}}$). Let[2] $B(\boldsymbol{c}, r) = \{\boldsymbol{x} \in \mathbb{R}^3 | \, \|\boldsymbol{x} - \boldsymbol{c}\| < r\}$ represent the open ball in $\mathbb{R}^3$ of center $\boldsymbol{c}$ and radius r. The simplest representation of the surface of molecule $\mathcal{M}$ is the *van der Waals Surface* Γ_{vdW} which can be defined as

$$\Gamma_{\text{vdW}} = \partial\Omega_{\text{vdW}} \text{ where } \Omega_{\text{vdW}} = \bigcup_{i=1}^{n_{\mathcal{M}}} B(\boldsymbol{c}_i, r_i). \tag{9.3}$$

Γ_{vdW} represents the most exhaustive set of points that can come in contact with an imaginary volumeless solvent molecule. Nevertheless, real solvent molecules are not volumeless and the spatial extension of one such molecule can be modeled by a solvent-sized spherical probe of radius $r_{\text{p}} > 0$. Therefore, when considering real solvents, not all points in $\Omega^{\text{C}}_{\text{vdW}} = \mathbb{R}^3 \backslash \Omega_{\text{vdW}}$ are accessible by solvent molecules, which leads to the definition of two additional surfaces.

Consider the set of points traced out by the center of one such probe sphere as it rolls over Γ_{vdW}. This describes the *Solvent-Accessible Surface* Γ_{SAS} [260], which can be

[1] We use the `pdb2pqr` tool [259] to obtain the list of atom centers and radii.

[2] In this chapter, we use $\|\boldsymbol{x}\|$ to represent the Euclidean norm $\sqrt{x_1^2 + x_2^2 + x_3^2}$ of $\boldsymbol{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$.

mathematically defined as

$$\Gamma_{\text{SAS}} = \partial\Omega^{\text{C}}_{\text{SAV}} \text{ where } \Omega^{\text{C}}_{\text{SAV}} = \bigcup_{i=1}^{n_{\mathcal{M}}} B\left(\boldsymbol{c}_i, r_i + r_{\text{p}}\right). \tag{9.4}$$

Note that the *Solvent-Accessible Volume* $\Omega_{\text{SAV}} = \mathbb{R}^3 \backslash \Omega^{\text{C}}_{\text{SAV}}$ is the set of points where the spherical probe of radius r_{p} can be centered without intersecting Ω_{vdW}.

Finally, the definition of Γ_{SAS} allows for a clear and straightforward identification of the set of points that cannot come in contact with solvent molecules, *i.e.*, the definition of the *Solvent-Excluded Volume* (SEV) [261, 262]

$$\Omega_{\text{SEV}} = \left\{\boldsymbol{x} \in \Omega^{\text{C}}_{\text{SAV}} \middle| \operatorname{dist}\left(\boldsymbol{x}, \Gamma_{\text{SAS}}\right) > r_{\text{p}}\right\}, \tag{9.5}$$

the surface of which is referred to as the *Solvent-Excluded Surface* $\Gamma_{\text{SES}} = \partial\Omega_{\text{SEV}}$, in this chapter. Those concepts are all illustrated (in 2D) in Figure 9.1 for a simplistic set of two atoms.

Remark. *The definition (9.5) of Ω_{SEV} does not prevent the existence of compact subsets of $\Omega^{C}_{SEV} = \mathbb{R}^3 \backslash \Omega_{SEV}$ being encapsulated by Ω_{SEV}. We refer to such encapsulated regions as internal cavities. Since those regions are isolated from the outer domain by the molecule itself, solvent molecules cannot fill those regions. As a consequence, it is essential to identify those regions in order to apply proper treatments (in the context of this chapter, those regions are added to Ω_{SEV}); a fast and robust identification method is presented in this chapter (see section 9.4.2).*

9.3 A germane theorem

Consider a set of atom centers $\boldsymbol{c}_i \in \mathbb{R}^3$ and radii $r_i \geq 0$, $(i = 1, \ldots, n_{\mathcal{M}})$, and consider the α-family $\mathcal{F}_\alpha$, $\alpha \geq 0$, of sets of $n_{\mathcal{M}}$ functions $\rho_{\alpha,i} : \mathbb{R}^3 \mapsto \mathbb{R}$ defined as

$$\rho_{\alpha,i}(\boldsymbol{x}) = \alpha + r_i - \|\boldsymbol{x} - \boldsymbol{c}_i\|, \quad i = 1, \ldots, n_{\mathcal{M}}, \tag{9.6}$$

and the corresponding α-family $\overline{\mathcal{F}}_\alpha$ of global functions $\overline{\rho}_\alpha : \mathbb{R}^3 \mapsto \mathbb{R}$ defined as

$$\overline{\rho}_\alpha(\boldsymbol{x}) = \max_{i \in \{1,\ldots,n_{\mathcal{M}}\}} \rho_{\alpha,i}(\boldsymbol{x}). \tag{9.7}$$

The family $\overline{\mathcal{F}}_\alpha$ plays an important role since the zero-level set Γ_α of $\overline{\rho}_\alpha$ is the definition of Γ_{vdW} (resp. of Γ_{SAS}) when $\alpha = 0$ (resp. $\alpha = r_{\text{p}}$). Note that Γ_α is a collection of portions of spherical surfaces since $\Gamma_\alpha \subseteq \bigcup_{i=1}^{n_{\mathcal{M}}} \partial B(\boldsymbol{c}_i, r_i + \alpha)$. Theorem 9.3.2 presented below links the local value of $\overline{\rho}_\alpha(\boldsymbol{x})$ to $\text{dist}(\boldsymbol{x}, \Gamma_\alpha)$.

First note that the functions $\rho_{\alpha,i}$ are signed distance functions, since $|\rho_{\alpha,i}(\boldsymbol{x})| = \text{dist}(\boldsymbol{x}, \partial B(\boldsymbol{c}_i, r_i + \alpha))$, $\rho_{\alpha,i}$ being positive inside the ball, and negative outside. Hence, from a geometrical point of view, it represents the (signed) distance from $\boldsymbol{x}$ to its projection $\boldsymbol{x}'_i$ on $\partial B(\boldsymbol{c}_i, r_i + \alpha)$,

$$\boldsymbol{x}'_i = \boldsymbol{x} + \frac{\boldsymbol{x} - \boldsymbol{c}_i}{\|\boldsymbol{x} - \boldsymbol{c}_i\|} \rho_{\alpha,i}(\boldsymbol{x}), \quad |\rho_{\alpha,i}(\boldsymbol{x})| = \|\boldsymbol{x} - \boldsymbol{x}'_i\|, \tag{9.8}$$

assuming $\boldsymbol{x} \neq \boldsymbol{c}_i$. Given a collection of $n_{\mathcal{M}}$ atoms, the three-dimensional space $\mathbb{R}^3$ can be divided into the regions

$$\mathcal{R}_i = \left\{\boldsymbol{x} \in \mathbb{R}^3 | \rho_{\alpha,i}(\boldsymbol{x}) > \rho_{\alpha,j}(\boldsymbol{x}) \ \forall j \in \{1, 2, \ldots, n_{\mathcal{M}}\} \setminus \{i\}\right\}, \tag{9.9}$$

for $i = 1, \ldots, n_{\mathcal{M}}$. The regions $\mathcal{R}_i$ are independent of α by definition.

Lemma 9.3.1. *The regions $\mathcal{R}_i$ are star-shaped with respect to $\boldsymbol{c}_i$ if non-empty: if $\boldsymbol{x} \in \mathcal{R}_i$, $\mathcal{R}_i$ contains all points $(1-t)\,\boldsymbol{x} + t\boldsymbol{c}_i$, where $t \in [0,\, 1]$ is a real parameter.*

Proof. For any point $\boldsymbol{x} \in \mathcal{R}_i$ and $j \in \{1, \ldots, n_{\mathcal{M}}\} \setminus \{i\}$, consider

$$\begin{aligned} f_{ij}(t) &= \rho_{\alpha,\,i}\left(\boldsymbol{x} + t\left(\boldsymbol{c}_i - \boldsymbol{x}\right)\right) - \rho_{\alpha,\,j}\left(\boldsymbol{x} + t\left(\boldsymbol{c}_i - \boldsymbol{x}\right)\right) \\ &= r_i - r_j + \left\|\boldsymbol{x} - \boldsymbol{c}_j + t\left(\boldsymbol{c}_i - \boldsymbol{x}\right)\right\| - |1-t|\left\|\boldsymbol{x} - \boldsymbol{c}_i\right\|, \end{aligned}$$

for $t \in [0,\, 1]$. This function is monotonically increasing since

$$\frac{\mathrm{d}f_{ij}}{\mathrm{d}t} = \left(\boldsymbol{c}_i - \boldsymbol{x}\right) \cdot \frac{\boldsymbol{x} - \boldsymbol{c}_j + t\left(\boldsymbol{c}_i - \boldsymbol{x}\right)}{\left\|\boldsymbol{x} - \boldsymbol{c}_j + t\left(\boldsymbol{c}_i - \boldsymbol{x}\right)\right\|} + \left\|\boldsymbol{c}_i - \boldsymbol{x}\right\| \geq 0, \quad t \in]0,\, 1[$$

by application of the Cauchy-Schwartz inequality. Therefore, for all $j \in \{1, \ldots, n_{\mathcal{M}}\} \setminus \{i\}$, $f_{ij}(t) \geq f_{ij}(0)$ and $f_{ij}(0) > 0$ given that $\boldsymbol{x} \in \mathcal{R}_i$; thus $(1-t)\,\boldsymbol{x} + t\boldsymbol{c}_i \in \mathcal{R}_i$ too. $\square$

Theorem 9.3.2. *If $\overline{\rho}_\alpha(\boldsymbol{x}) \leq 0$, we have $dist(\boldsymbol{x}, \Gamma_\alpha) = -\overline{\rho}_\alpha(\boldsymbol{x})$, i.e., $\overline{\rho}_\alpha(\boldsymbol{x})$ is already a (negative) distance function where it is negative. If $\overline{\rho}_\alpha(\boldsymbol{x}) > 0$, we have $dist(\boldsymbol{x}, \Gamma_\alpha) \geq \overline{\rho}_\alpha(\boldsymbol{x})$.*

Proof. For any point $\boldsymbol{x}$ in $\mathcal{R}_i$, consider the ball $B\left(\boldsymbol{x}, \left|\rho_{\alpha,\,i}(\boldsymbol{x})\right|\right)$. That ball is such that

$$B\left(\boldsymbol{x}, \left|\rho_{\alpha,\,i}(\boldsymbol{x})\right|\right) \cap \Gamma_\alpha = \emptyset.$$

Indeed,

- if $\rho_{\alpha,\,i}(\boldsymbol{x}) < 0$, $B\left(\boldsymbol{x}, \left|\rho_{\alpha,\,i}(\boldsymbol{x})\right|\right) \cap \partial B\left(\boldsymbol{c}_i, r_i + \alpha\right) = \emptyset$, by definition. Therefore, $B\left(\boldsymbol{x}, \left|\rho_{\alpha,\,i}(\boldsymbol{x})\right|\right) \cap \Gamma_\alpha \neq \emptyset$ would imply the existence of a point $\boldsymbol{y}$ on a sphere

$\partial B(\boldsymbol{c}_j, r_j + \alpha)$, $j \neq i$ such that $\|\boldsymbol{x} - \boldsymbol{y}\| \leq |\rho_{\alpha,i}(\boldsymbol{x})|$ and thus $\rho_{\alpha,j}(\boldsymbol{x}) \geq \rho_{\alpha,i}(\boldsymbol{x})$, which is a contradiction since $\boldsymbol{x} \in \mathcal{R}_i$;

- if $\rho_{\alpha,i}(\boldsymbol{x}) > 0$, $B(\boldsymbol{x}, |\rho_{\alpha,i}(\boldsymbol{x})|) \subset B(\boldsymbol{c}_i, r_i + \alpha)$ and $B(\boldsymbol{c}_i, |\rho_{\alpha,i}(\boldsymbol{x})|) \cap \Gamma_\alpha = \emptyset$ since $B(\boldsymbol{c}_i, r_i + \alpha) \cap \Gamma_\alpha = \emptyset$.

As a consequence, $\operatorname{dist}(\boldsymbol{x}, \Gamma_\alpha) \geq |\rho_{\alpha,i}(\boldsymbol{x})|$. Hence, $\overline{\rho}_\alpha(\boldsymbol{x})$, which is equal to $\rho_{\alpha,i}(\boldsymbol{x})$ in $\mathcal{R}_i$, is indeed a signed distance function only if $\boldsymbol{x}'_i \in \partial B(\boldsymbol{x}, |\rho_{\alpha,i}(\boldsymbol{x})|)$, as defined in (9.8), lies on Γ_α or, equivalently, if $\overline{\rho}_\alpha(\boldsymbol{x}'_i) = 0 = \rho_{\alpha,i}(\boldsymbol{x}'_i)$, *i.e.*, if $\boldsymbol{x}'_i \in \mathcal{R}_i$ itself too. Since $\mathcal{R}_i$ is star-shaped with respect to $\boldsymbol{c}_i$, the latter is true for any point $\boldsymbol{x} \in \mathcal{R}_i$ such that $\rho_{\alpha,i}(\boldsymbol{x}) < 0$ by virtue of Lemma 9.3.1. $\square$

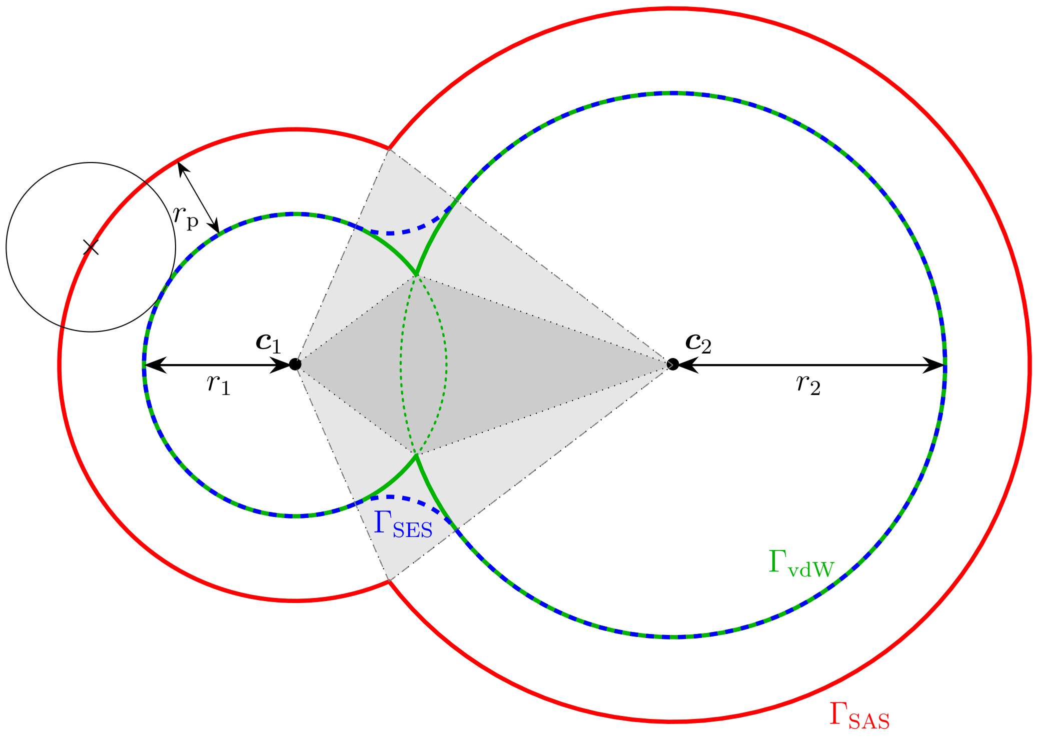


Figure 9.1: the three standard biomolecular surfaces for a simplistic system of two atoms in two spatial dimensions. Γ_{vdW} is drawn in solid green line; Γ_{SAS} is drawn in solid red line; Γ_{SES} is represented in dashed blue line. The light (resp. dark) shaded gray area is the only region of points $\boldsymbol{x}$ such that $|\overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{x})| \neq \operatorname{dist}(\boldsymbol{x}, \Gamma_{\mathrm{SAS}})$ (resp. $|\overline{\rho}_0(\boldsymbol{x})| \neq \operatorname{dist}(\boldsymbol{x}, \Gamma_{\mathrm{vdW}})$).

9.4 Capturing the interface and building an adaptive grid

In this section, we present two methods designed to build a computational grid satisfying criteria (9.1) and (9.2). An appropriate node-sampled function is built on-the-fly and its zero-level set represents Γ_{SES} implicitly. We refer to the first method as the *construction by reinitialization* while the second technique is called *construction by direct calculation*. We point out that we have developed the construction by direct calculation for the sole purpose of showing the first-order accuracy of the construction by reinitialization (see section 9.6.1), since it leads to the *exact* signed distance to Γ_{SES}. Optimization and acceleration of the construction by direct calculation *is* possible but is beyond the scope of this work.

The two algorithms ensure consistent values for the node-sampled functions in a layer of m finest grid cells on both sides of Γ_{SES}, where m is either 1 or 2. We use distributed computing over P processes of ranks $0, 1, \ldots, P-1$. The methods share the same following backbone:

1. construction of a grid and a node-sampled function φ_{SAS} whose zero-level set represents Γ_{SAS} implicitly;

2. correction of the values of φ_{SAS} at every grid node $\boldsymbol{z}$ such that $\varphi_{\mathrm{SAS}}(\boldsymbol{z}) > 0$ and $\mathrm{dist}(\boldsymbol{z}, \Gamma_{\mathrm{SAS}}) \leq r_{\mathrm{p}} + m\delta$, where δ is the length of the diagonal of the finest grid cells. For this operation,

    - the construction by reinitialization solves a nonlinear hyperbolic partial differential equation, with first order accuracy in the region of interest;

    - the construction by direct calculation calculates the exact distance from Γ_{SAS} at every node of interest.

A new node-sampled function $\overline{\phi}$ is then defined by subtracting r_{p} from ϕ_{SAS}, the updated φ_{SAS}: the zero-level of $\overline{\phi}$ represents Γ_{SES} implicitly;

3. identification of internal cavities and application of ad-hoc treatment (in this chapter, those internal cavities are removed by being added to Ω_{SEV});

4. coarsening of the computational grid based on criterion (9.2);

5. imposition of the desired minimum grid level.

Each of the steps above is explained in further details in the following sections: the computational workload is kept as low as possible at each step and load-balancing is addressed throughout the exposition to ensure the strong scalability of the algorithm. The step-by-step progression of the algorithm is illustrated with the construction of a 5/13 computational grid distributed over 8 processes, with $L = 1.2$, $m = 2$ and $r_{\mathrm{p}} = 1.4$ Å. For the sake of clarity, this step-by-step illustration is made two-dimensional by building the grid based on an imaginary planar molecule, constructed by projecting all atoms of the `3J6D` protein (131664 atoms) onto a plane. This example serves no other purpose than to illustrate intermediary steps of our algorithm. We emphasize that the algorithm and its derivation are intrinsically three-dimensional and happen to be translatable to two spatial dimensions: the actual three-dimensional representation of those illustrations were created with our method too and can be found in Figures 9.10, 9.11 and 9.12.

The implementation of the algorithm relies on the existence of basic parallel tools for handling adaptive meshes such as:

- the ability to locally (and possibly recursively) refine and/or coarsen cells based on a cell-level boolean criterion;

- the ability to partition the grid and cell-associated constant-size data (at least one integer per cell), over processes with respect to a cell-level weight function;

- unique global cell- and node-orderings, that are independent of the grid partitioning.

Our implementation builds upon the `p4est` library [174, 103], which provides all of the above features. Nevertheless, it is translatable to any other framework with corresponding tools. Every process $p \in \{0, 1, \dots, P-1\}$ has access to the full list of atom coordinates and radii.

9.4.1 Capturing the Solvent-Accessible Surface

As pointed out in [263, 245] and described in section 9.3, Γ_{SAS} is represented implicitly in a straightforward fashion by the zero level-set of

$$\overline{\rho}_{r_{\mathrm{p}}}\left(\boldsymbol{x}\right) = \max_{i=1,\dots,n_{\mathcal{M}}} r_{\mathrm{p}} + r_i - \left\|\boldsymbol{x} - \boldsymbol{c}_i\right\|. \tag{9.10}$$

Geometrically, the center of a solvent-sized spherical probe can be located at any point $\boldsymbol{x}$ where $\overline{\rho}_{r_{\mathrm{p}}}\left(\boldsymbol{x}\right) \leq 0$ without intersecting Γ_{vdW}, and the probe is in contact with Γ_{vdW} if and only if $\overline{\rho}_{r_{\mathrm{p}}}\left(\boldsymbol{x}\right) = 0$.

The first step of the algorithm consists of constructing an adaptive grid alongside a node-sampled function φ_{SAS}. Local differences between φ_{SAS} and $\overline{\rho}_{r_{\mathrm{p}}}$ may exist in $\Omega_{\mathrm{SAV}} = \mathbb{R}^3 \backslash \Omega_{\mathrm{SAV}}^{\mathrm{C}}$, but φ_{SAS} and $\overline{\rho}_{r_{\mathrm{p}}}$ share the same zero-level set Γ_{SAS} numerically, as shown in subsection 9.4.1. Algorithm 3 shows how this initial (adaptive) grid and the corresponding function φ_{SAS} are constructed. Figure 9.2 illustrates the resulting grid and Γ_{SAS} as the zero-level set of φ_{SAS} for our two-dimensional illustration, for the two methods.

Refinement criterion

Algorithm 3 Construction of the initial grid. This algorithm builds the initial grid of desired maximum level $l_{\max}$ from the root cell C_{root} representing the entire computational domain. `phi_sas` is the (distributed) vector of node-sampled values of φ_{SAS}.

```
 1: function CONSTRUCT_SAS_GRID_AND_LEVELSET(root cell $C_{\text{root}}$, $l_{\max}$)
 2:     phi_sas ← new vector of node-sampled values; ▷ 8 components for the root cell.
 3:     calculate the values of $\varphi_{\text{SAS}}$ at the vertices of $C_{\text{root}}$ and fill phi_sas
 4:     for ($i = 0$; $i < l_{\max}$; $i++$) do
 5:         phi_sas_coarse ← phi_sas;
 6:         loop through all cells in the grid and refine those marked true
                by Algorithm 4 for the construction by reinitialization or
                by Algorithm 5 for the construction by direct calculation;
 7:         phi_sas ← new vector of node-sampled values;
 8:         scatter the already-known values from phi_sas_coarse to phi_sas;
 9:         destroy phi_sas_coarse and release corresponding memory;
10:         partition the grid over the pool of processes $\{0, 1, \ldots, P-1\}$ for
                load balancing (subsection 9.4.1) and update phi_sas' layout accordingly;
11:         calculate the values of $\varphi_{\text{SAS}}$ at the newly added grid points and fill the
                corresponding (yet undetermined) components of phi_sas (Algorithm 7);
12:     end for
13:     if method_to_use is construction_by_reinitialization then
14:         partition the grid over the pool of processes $\{0, 1, \ldots, P-1\}$ for load balancing
                for next step (section 9.4.1) and update phi_sas' layout accordingly;
15:     end if
16: end function
```

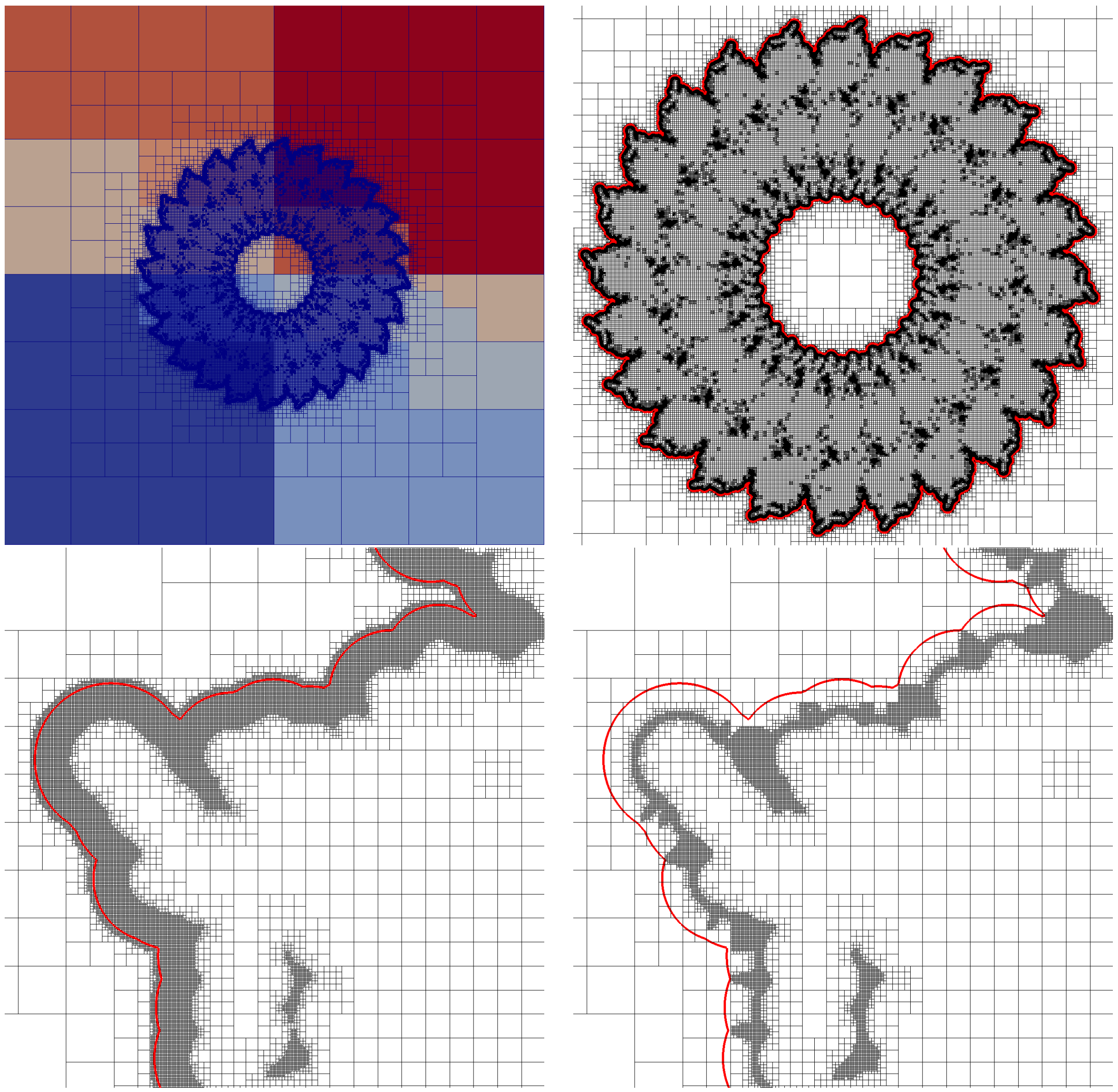


Figure 9.2: outcome of Algorithm 3 for the illustrative planar molecule, $r_{\mathrm{p}} = 1.4\,\text{Å}$, $L = 1.2$ and $m = 2$, the maximum level of refinement ($l_{\max} = 13$) is such that $r_{\mathrm{p}}/\delta \simeq 14.8$, where δ is the diagonal of the finest grid cells. Note that projecting the atoms of protein `3J6D` onto a plane packs them all in the torus-shaped region of the domain limited by the red contour lines, which explains the high density of fine cells in that region. Top left: computational grid created by Algorithm 3. Each color represents one grid partition. Top right: a zoom-in version of the top left subfigure; the red curve is the zero-level set of φ_{SAS}, *i.e.*, Γ_{SAS}. Bottom left: zoom-in close to Γ_{SAS}, grid created for the construction by reinitialization. Bottom right: zoom-in close to Γ_{SAS}, grid created for the construction by direct calculation.

For the construction by reinitialization, the adaptive refinement criterion from Algorithm 4 (see line 6 from Algorithm 3) fulfills several conditions that we intend to develop and explain here. As detailed in subsection 9.4.1, φ_{SAS} and $\overline{\rho}_{r_{\mathrm{p}}}$ might differ locally; these local differences are shown to be consistently irrelevant with the inequalities here below. Therefore, φ_{SAS} and $\overline{\rho}_{r_{\mathrm{p}}}$ can be invariably substituted in the (in)equalities in this context, which explains why φ_{SAS} is used instead of $\overline{\rho}_{r_{\mathrm{p}}}$ in Algorithm 4.

In the context of the construction by reinitialization, the initial grid and the function $\overline{\rho}_{r_{\mathrm{p}}}$ serve as the basis for subsequent calculations leading to Γ_{SES}, eventually. Let ϕ_{SAS} be the reinitialized version of $\overline{\rho}_{r_{\mathrm{p}}}$, *i.e.*, ϕ_{SAS} is a signed distance function that shares the same sign and the same zero-level set as $\overline{\rho}_{r_{\mathrm{p}}}$. A node-sampled version of this signed-distance function ϕ_{SAS} is obtained by solving a pseudo-time problem on the initial computational grid created by Algorithm 3 and its r_{p}-level set is critical for the definition of Γ_{SES} (see step number 2 on p. 271 and section 9.4.1).

As a consequence, the grid must obey the following requirements in order to ensure the accurate implicit representation of Γ_{SES}:

- it must be tessellated with the finest cells in the region $\Omega^{\mathrm{C}}_{\mathrm{SAV}} \backslash \Omega_{\mathrm{SEV}}$ in order to ensure the accurate computation of the r_{p}-level set of ϕ_{SAS};
- m layer(s) of finest cells must also cover Γ_{SAS} in Ω_{SAV} to ensure the accurate localization of Γ_{SAS}.

Those two latter constraints are required to ensure the *accuracy* of the calculations. They dictate the (maximum) level of refinement in parts of the computational domain. The grid nodes $\boldsymbol{z}$ that satisfy

$$-m\delta \leq \phi_{\mathrm{SAS}}(\boldsymbol{z}) \leq r_{\mathrm{p}} \tag{9.11}$$

fall under these two categories. Note that

$$\begin{aligned}\overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{z}) &= \phi_{\mathrm{SAS}}(\boldsymbol{z}) \quad \text{if } \overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{z}) \leq 0 \text{ and}\\ \overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{z}) &\leq \phi_{\mathrm{SAS}}(\boldsymbol{z}) \quad \text{if } \overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{z}) > 0,\end{aligned}$$

by virtue of Theorem 9.3.2. Therefore, the set of points $\boldsymbol{z}$ satisfying (9.11) is actually a subset of the set of points $\boldsymbol{z}'$ satisfying

$$-m\delta \leq \overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{z}') \leq r_{\mathrm{p}}. \tag{9.12}$$

Since $\overline{\rho}_{r_{\mathrm{p}}}$ is known beforehand while ϕ_{SAS} is not, the latter criterion is used to enforce those two accuracy constraints: line 4 in Algorithm 4 takes the above condition into account. The effect of that criterion is clearly illustrated in Figure 9.5.

Regarding the region of the domain that is not affected by these constraints, one can divide it two different cases:

- the points in Ω_{SAV}, *i.e.*, where $\overline{\rho}_{r_{\mathrm{p}}} < -m\delta$. In this case, the exact distance to Γ_{SES} is known to be $r_{\mathrm{p}} - \overline{\rho}_{r_{\mathrm{p}}}$. Therefore, the exact adaptive local grid level can be determined and imposed right away. Line 8 of Algorithm 4 imposes this condition;

- the points in $\Omega_{\mathrm{SAV}}^{\mathrm{C}}$, *i.e.*, where $\overline{\rho}_{r_{\mathrm{p}}} > r_{\mathrm{p}}$. Then, the distance to Γ_{SES} is known to be $\phi_{\mathrm{SAS}} - r_{\mathrm{p}}$, which is unknown beforehand. By using $\overline{\rho}_{r_{\mathrm{p}}} - r_{\mathrm{p}}$ instead, more fine cells than needed might be created since $\overline{\rho}_{r_{\mathrm{p}}} - r_{\mathrm{p}} \leq \phi_{\mathrm{SAS}} - r_{\mathrm{p}}$ in this case (see Theorem 9.3.2). Yet, this inequality also ensures that no further distance-induced refinement[3] is required afterwards. Line 6 of Algorithm 4 enforces this condition.

[3]We distinguish two reasons for refining a grid cell: a) for imposing the desired minimum level of refinement of the computational grid, b) because of criterion (9.1). We refer to the latter as *distance-induced refinement.*

For the construction by direct calculation, the adaptive refinement criterion from Algorithm 5 (see line 6 from Algorithm 3) differs from Algorithm 4 and is somehow optimal in order to reduce the workload for subsequent direct calculations to Γ_{SAS}. As pointed it out in the proof of Theorem 9.3.2, for a grid node $\boldsymbol{v}$ such that $\overline{\rho}_{r_\text{p}}(\boldsymbol{v}) > 0$, we have $\overline{\rho}_{r_\text{p}}(\boldsymbol{v}) \leq \text{dist}(\boldsymbol{v}, \Gamma_{\text{SAS}})$. However, it was also emphasized that $\overline{\rho}_{r_\text{p}}(\boldsymbol{v}) = \text{dist}(\boldsymbol{v}, \Gamma_{\text{SAS}})$ if $\overline{\rho}_{r_\text{p}}(\boldsymbol{v}_i') = 0$, where $\boldsymbol{v}_i'$ is the projection of $\boldsymbol{v}$ on $\partial B(\boldsymbol{c}_i, r_i + r_\text{p})$, as defined in (9.8). For every grid node $\boldsymbol{v}$ such that $0 < \overline{\rho}_{r_\text{p}}(\boldsymbol{v}) \leq r_\text{p} + m\delta$, this gives us a straightforward condition to check if $\overline{\rho}_{r_\text{p}}(\boldsymbol{v}) = \text{dist}(\boldsymbol{v}, \Gamma_{\text{SAS}})$. The evaluation of that condition is not prohibitively expensive since the number of atoms is optimally reduced (see subsection 9.4.1), and this strategy allows to tag all such nodes that $\overline{\rho}_{r_\text{p}}(\boldsymbol{v}) < \text{dist}(\boldsymbol{v}, \Gamma_{\text{SAS}})$.

In the context of the construction by direct calculation, the exact distance to Γ_{SAS} is calculated afterwards for all grid nodes $\boldsymbol{v}$ belonging to a grid cell of maximum refinement level. Therefore, by refining a grid cell C only if

$$\exists \boldsymbol{v} \in \mathcal{V}(C) : \left| \overline{\rho}_{r_\text{p}}(\boldsymbol{v}) - r_\text{p} \right| \leq \frac{L\text{diag}(C)}{2} \text{ or } \boldsymbol{v} \text{ has been tagged}, \tag{9.13}$$

the subsequent workload is minimized. Algorithm 5 summarizes the above condition, its effect is illustrated in Figure 9.5.

For both methods, note that subsequent coarsening steps intend to remove all unnecessary fine cells that have been eventually created (see section 9.4.2). Besides, refining a cell for the sole purpose of imposing a desired minimum level for the final computational grid is disregarded until the very last step of the global algorithm, as it would increase the computational workload unnecessarily until then.

Although a formal definition of $\overline{\rho}_{r_\text{p}}$ is available (see (9.10)), its brute force calculation,

Algorithm 4 Refinement criterion for the creation of the initial grid for the construction by reinitialization. This algorithm determines if a grid cell C needs to be refined based on the value of φ_{SAS} at its vertices, the desired maximum level of refinement $l_{\max}$, the desired proportionality constant L and the desired $m \in \{1, 2\}$.

```
1:  function REFINE_BASED_ON_SAS_FOR_REINITIALIZATION($C$, $l_{\max}$, $L$, $m$)
2:      for $\boldsymbol{v} \in \mathcal{V}(C)$ do
3:          read value of $\varphi_{\text{SAS}}(\boldsymbol{v})$;
4:          if $-m\delta \leq \varphi_{\text{SAS}}(\boldsymbol{v}) \leq r_{\text{p}}$ then
5:              return true;
6:          else if $0 \leq (\varphi_{\text{SAS}}(\boldsymbol{v}) - r_{\text{p}}) \leq L \operatorname{diag}(C)/2$ then
7:              return true;
8:          else if $(\varphi_{\text{SAS}}(\boldsymbol{v}) < -m\delta) \wedge (r_{\text{p}} - \varphi_{\text{SAS}}(\boldsymbol{v}) \leq L \operatorname{diag}(C)/2)$ then
9:              return true;
10:         end if
11:     end for
12:     return false;
13: end function
```

Algorithm 5 Refinement criterion for the creation of the initial grid for the construction by direct calculation. This algorithm determines if a grid cell C needs to be refined based on the value of φ_{SAS} at its vertices, the desired maximum level of refinement $l_{\max}$, the desired proportionality constant L and the desired $m \in \{1, 2\}$. A grid node $\boldsymbol{v}$ is *tagged* when $0 < \overline{\rho}_{r_{\text{p}}}(\boldsymbol{v}) < r_{\text{p}} + m\delta$ and $\overline{\rho}_{r_{\text{p}}}(\boldsymbol{v}) \neq \operatorname{dist}(\boldsymbol{v}, \Gamma_{\text{SAS}})$

```
1:  function REFINE_BASED_ON_SAS_FOR_DIRECT_CALCULATION($C$, $l_{\max}$, $L$, $m$)
2:      $l \leftarrow$ level of cell $C$;
3:      for $\boldsymbol{v} \in \mathcal{V}(C)$ do
4:          read value of $\varphi_{\text{SAS}}(\boldsymbol{v})$;
5:          if $|\varphi_{\text{SAS}}(\boldsymbol{v}) - r_{\text{p}}| \leq L \operatorname{diag}(C)/2 \vee \boldsymbol{v}$ is tagged then
6:              return true;
7:          end if
8:      end for
9:      return false;
10: end function
```

i.e., by looping through *all* $n_{\mathcal{M}}$ atoms in order to find the atom index i maximizing $r_{\mathrm{p}} + r_i - \|\boldsymbol{z} - \boldsymbol{x}_i\|$ for every single grid node $\boldsymbol{z}$, would be extremely inefficient especially for very large molecules (large $n_{\mathcal{M}}$) and fine computational grids. In this section we present a method that dramatically reduces the computational cost associated with these calculations. The method is based on a traditional *divide-and-conquer* approach and is thus very similar to [241], but uses an analytical function to represent the interface implicitly and accurately. Similar techniques have been implemented and shown to be successful in (fixed radius) nearest-neighbor-search problems [264, 265, 266]. Some minor modifications inherent to variable van der Waals radii $r_i \geq 0$ in the functions $\rho_{r_{\mathrm{p}},i}(\boldsymbol{x}) = r_{\mathrm{p}} + r_i - \|\boldsymbol{x} - \boldsymbol{c}_i\|$ $(i = 1, \ldots, n_{\mathcal{M}})$ and to the structure of the grid to be constructed are required, as detailed next.

The idea consists of reducing the list of atom indices to consider when evaluating a slightly modified version φ_{SAS} of $\overline{\rho}_{r_{\mathrm{p}}}$ as the grid is further and further refined. At the beginning of a given iteration of the `for` loop in Algorithm 3, the current finest cells of the grid under construction are all associated with *reduced lists* of atom indices. These lists are optimally reduced in the sense that they contain only the atom indices that are relevant either to determine the desired local refinement level or such that the corresponding atoms are closer than

- $m\delta$ away for the construction by reinitialization, to ensure the accurate localization of Γ_{SAS};

- $r_{\mathrm{p}} + m\delta$ away for the construction by direct calculation, to ensure the exactness of the direct calculation of the distance to Γ_{SAS}, within $r_{\mathrm{p}} + m\delta$.

The lists corresponding to cells that are eventually marked *not* to be refined by Algorithm 4 are deleted at the end of the current iteration of the `for` loop: since the final cell level has been determined, the lists are no longer needed and the corresponding memory can be freed. On the other hand, when one of those cells is further refined, one new reduced list of atom indices is built for each of the eight children cells that have been created. Each of these eight new reduced lists is built by reducing the list of the parent cell, which is itself eventually deleted afterwards. The values of φ_{SAS} at newly created grid nodes are initialized to $-D$, *i.e.*, a lower bound of $\overline{\rho}_{r_{\text{p}}}$ in the computational domain. Then, when evaluating the values of φ_{SAS} at those newly added points (line 11 of Algorithm 3), one loops through all newly created cells C_{new} and updates the values of $\varphi_{\text{SAS}}(\boldsymbol{v})$ at their vertices $\boldsymbol{v}$ by:

$$\varphi_{\text{SAS}}(\boldsymbol{v}) \leftarrow \max\left(\varphi_{\text{SAS}}(\boldsymbol{v}), \max_{j\in\mathcal{L}(C_{\text{new}})}\left(r_{\text{p}} + r_j - \|\boldsymbol{v} - \boldsymbol{c}_j\|\right)\right), \tag{9.14}$$

where $\mathcal{L}(C)$ is the reduced list of atom indices associated with cell C. At the beginning of Algorithm 3, the root cell is associated with the full list of $n_{\mathcal{M}}$ atoms; the procedure naturally creates lists of $\mathcal{O}(1)$ element(s) when the cells are small enough, accelerating the local calculations of the Γ_{SAS}-capturing level-set function drastically, as the grid is refined. Let D_{root} be the diagonal of the root cell(s) of the computational domain, the diagonal of the finest cells for a desired maximum level $l_{\max}$ is thus $\delta = D_{\text{root}}\,2^{-l_{\max}}$.

For the construction by reinitialization, as underlined in section 9.4.1, when the value of $\overline{\rho}_{r_{\text{p}}}$ is strictly smaller than $-m\delta$ ($m = 1$ or 2), its exact value becomes irrelevant regarding the accurate localization of Γ_{SAS}. Indeed, consider en edge $\mathcal{E} = [\boldsymbol{z}_{\mathcal{E}_1}, \boldsymbol{z}_{\mathcal{E}_2}]$ crossed by Γ_{SAS}, *i.e.*, such that $\overline{\rho}_{r_{\text{p}}}(\boldsymbol{z}_{\mathcal{E}_1})\,\overline{\rho}_{r_{\text{p}}}(\boldsymbol{z}_{\mathcal{E}_2}) \leq 0$, the point $\boldsymbol{\zeta} \in \mathcal{E}$ where $\overline{\rho}_{r_{\text{p}}}(\boldsymbol{\zeta}) = 0$ is approximated by the root of:

- the linear interpolant of $\overline{\rho}_{r_{\text{p}}}$ along $\mathcal{E}$ when $m = 1$;
- the quadratic interpolant of $\overline{\rho}_{r_{\text{p}}}$ along $\mathcal{E}$ when $m = 2$. The second-oder derivative of the interpolant is set to $\texttt{minmod}\left(\frac{\partial^2\overline{\rho}_{r_{\text{p}}}}{\partial s^2}(\boldsymbol{z}_{\mathcal{E}_1}), \frac{\partial^2\overline{\rho}_{r_{\text{p}}}}{\partial s^2}(\boldsymbol{z}_{\mathcal{E}_2})\right)$ where s is the direction of

the edge $\mathcal{E}$. Those second-order derivatives are approximated by standard centered finite differences, which extends the exactness requirement for $\overline{\rho}_{r_\mathrm{p}}$ by one layer of width δ.

Therefore the *exact* value of $\overline{\rho}_{r_\mathrm{p}}$ is required at a grid node $\boldsymbol{z}$ only when $\overline{\rho}_{r_\mathrm{p}}(\boldsymbol{z}) \geq -g$ where $g = m\delta$.

For the construction by direct calculation, we intend to calculate the exact distance to Γ_SAS within a layer of width $r_\mathrm{p} + m\delta$, eventually. Therefore, any atom of index j that is farther than $r_\mathrm{p} + m\delta$ away, *i.e.*, such that $\rho_{r_\mathrm{p},j} < -g$ where $g = r_\mathrm{p} + m\delta$, is irrelevant for that purpose.

For both methods, when the latter inequality is not satisfied, *i.e.*, $\overline{\rho}_{r_\mathrm{p}} < -g$, one only needs to know whether

$$r_\mathrm{p} - \overline{\rho}_{r_\mathrm{p}} \leq L\,\mathrm{diag}\,(C)\,/2 \tag{9.15}$$

is true or not for refinement purposes only (see line 8 from Algorithm 4). As a consequence, any substitution function φ_SAS, sampled at the nodes of the initial grid, and satisfying the two following equivalence conditions

$$\varphi_\mathrm{SAS}(\boldsymbol{v}) = \overline{\rho}_{r_\mathrm{p}}(\boldsymbol{v})\,,\ \text{where } \overline{\rho}_{r_\mathrm{p}}(\boldsymbol{v}) \geq -g \tag{9.16}$$

and, for any cell C of the grid,

$$\overline{\rho}_{r_\mathrm{p}}(\boldsymbol{v}) < r_\mathrm{p} - \frac{L\,\mathrm{diag}\,(C)}{2}\,\forall \boldsymbol{v} \in \mathcal{V}(C) \Leftrightarrow \varphi_\mathrm{SAS}(\boldsymbol{v}) < r_\mathrm{p} - \frac{L\,\mathrm{diag}\,(C)}{2}\,\forall \boldsymbol{v} \in \mathcal{V}(C) \tag{9.17}$$

satisfies all the requirements for accurately capturing Γ_SAS and determining the correct desired cell levels in Ω_SAV.

Algorithm 6 presents a method to create the reduced list of atom indices to associate with a newly created cell, based on the reduced list of its parent cell. As detailed below, the resulting node-sampled function φ_{SAS} satisfies the two equivalence conditions (9.16) and (9.17) by construction and is thus a valid substitution for $\overline{\rho}_{r_\text{p}}$. The rationale behind Algorithm 6 can be developed as follows: consider a newly created cell C_{child} and the reduced list of atom indices $\mathcal{L}\left(C_{\text{parent}}\right)$ associated with its parent cell C_{parent}. The list $\mathcal{L}\left(C_{\text{child}}\right)$ is a subset of $\mathcal{L}\left(C_{\text{parent}}\right)$ constructed by the following rules:

I. (lines 13 to 16) for all integer(s) j in $\mathcal{L}\left(C_{\text{parent}}\right)$, if there exists a point $\boldsymbol{z}$ in C_{child} such that

   I.i. either $\rho_{r_\text{p},j}\left(\boldsymbol{z}\right) \geq -g$,

   I.ii. or $\rho_{r_\text{p},j}\left(\boldsymbol{z}\right) > r_\text{p} - L\,\text{diag}\left(C_{\text{child}}\right)/4$,

   j must be added to $\mathcal{L}\left(C_{\text{child}}\right)$;

II. (lines 17 to 24 and 26 to 28) if no atom index has been added to $\mathcal{L}\left(C_{\text{child}}\right)$ by the above rule I, $\mathcal{L}\left(C_{\text{child}}\right)$ is set to contain the single atom index $k \in \mathcal{L}\left(C_{\text{parent}}\right)$ such that:

$$r_\text{p} + \max_{\boldsymbol{v}\in\mathcal{V}(C_{\text{child}})}\left(r_k - \|\boldsymbol{v} - \boldsymbol{c}_k\|\right) = r_\text{p} + \max_{i\in\mathcal{L}(C_{\text{parent}})}\max_{\boldsymbol{v}\in\mathcal{V}(C_{\text{child}})}\left(r_i - \|\boldsymbol{v} - \boldsymbol{c}_i\|\right). \quad (9.18)$$

Rule I.i ensures the equivalence (9.16) and also ensures that all atoms that are closer than g from C_{child} are kept in the list. On the other hand, rules I.ii and II intend to keep track of the smallest list of atom indices that might be responsible for adaptive grid refinement and thus, they ensure the equivalence (9.17). Indeed, let us consider a newly created cell C in Ω_{SAV} that is farther away than a distance g from Γ_{SAS}, at a given iteration of the `for` loop in Algorithm 3. Since C is farther than g away from Γ_{SAS}, C is irrelevant regarding the accurate localization of Γ_{SAS} (for the construction by reinitialization) or regarding the direct calculation of the distance to Γ_{SAS} (for the construction by direct calculation). Nevertheless, the cell may need further refinement at the next iteration of the `for` loop in Algorithm 3, depending on the distances from its vertices to Γ_{SAS}. As a consequence, an appropriate reduced list of atoms must be associated with the cell to ensure the correct cell sizes in Ω_{SAV}. In this context, we distinguish two cases:

a) C does need to be refined at the next stage and (some of) its eight children cells might need to be further refined too;

b) C might need to be refined or not at the next stage but, if it does, none of its children cells need to be further refined.

Any atom index $j \in \mathcal{L}(C_{\text{parent}})$ satisfying I.ii corresponds to case a. However, if no atom in $\mathcal{L}(C_{\text{parent}})$ satisfies I.ii, then C might need to be refined or not, depending on the maximum value of $\overline{\rho}_{r_{\text{p}}}$ at its vertices, for which one needs to keep track of the only atom index that may trigger the cell refinement. This is the purpose of rule II which corresponds to case b.

Reducing the list to that single atom index might lead to differences between $\overline{\rho}_{r_{\text{p}}}$ and φ_{SAS} at some grid nodes; however, those differences are irrelevant with the inequalities from subsection 9.4.1 and do not interfere with the final local grid levels since it is absolutely certain that none of the children cells of C need further refinement if C itself is refined. Figure 9.3 illustrates the regions of interest for the list reduction technique, in two spatial dimensions without loss of generality.

Algorithm 6 Build the reduced list of atom indices associated with cell C, based on the reduced list $\mathcal{L}(C_{\text{parent}})$ associated with its parent cell. $m \in \{1, 2\}$ is the desired order of accuracy, $l_{\max}$ is the maximum level of the grid cells, L is the proportionality constant used for grid construction, $\delta = D_{\text{root}}\, 2^{-l_{\max}}$.

```
 1: function REDUCE_LIST_OF_ATOMS($C$, $\mathcal{L}(C_{\text{parent}})$)
 2:     create an empty list $\mathcal{L}$;
 3:     $s \leftarrow -\infty$;                 ▷ $s = \max_{j \in \mathcal{L}(C_{\text{parent}})} \max_{\boldsymbol{v} \in \mathcal{V}(C)} \rho_{r_{\text{p}}, j}(\boldsymbol{v})$.
 4:     $j \leftarrow 0$;                       ▷ Index of the atom maximizing $s$.
 5:     $n \leftarrow 0$;                       ▷ Number of atom indices added to $\mathcal{L}$
 6:     if method_to_use is construction_by_reinitialization then
 7:         $g \leftarrow m\delta$
 8:     else
 9:         $g \leftarrow m r_{\text{p}} + \delta$
10:     end if
11:     for $i \in \mathcal{L}(C_{\text{parent}})$ do
12:         $d \leftarrow r_{\text{p}} + r_i - \min_{\boldsymbol{z} \in C} \|\boldsymbol{z} - \boldsymbol{c}_i\|$
13:         if $d \geq \min(-g, r_{\text{p}} - L\,\text{diag}(C)/4)$ then
14:             add $i$ to $\mathcal{L}$;
15:             $n \leftarrow n + 1$;
16:         end if
17:         if $n == 0$ then
18:             for $\boldsymbol{v} \in \mathcal{V}(C)$ do
19:                 if $r_{\text{p}} + r_i - \|\boldsymbol{v} - \boldsymbol{c}_i\| > s$ then
20:                     $s \leftarrow r_{\text{p}} + r_i - \|\boldsymbol{v} - \boldsymbol{c}_i\|$;
21:                     $j \leftarrow i$;
22:                 end if
23:             end for
24:         end if
25:     end for
26:     if $n == 0$ then
27:         add $j$ to $\mathcal{L}$;
28:     end if
29:     return $\mathcal{L}$;
30: end function
```

Load balancing

Load balancing is a key feature to avoiding stalling processes and thus ensuring the strong scalability of a parallel algorithm. When it comes to multi-dimensional problems involving (possibly adaptive) meshes, partitioning the mesh uniformly over the available processes might give satisfactory results (a) if elementary calculations need to be performed at *all* cells or nodes of the mesh and, (b) if those calculations have a roughly constant number of floating point operations. In the context of the list-reduction algorithm presented in this section, none of these conditions (a) and (b) are met. When

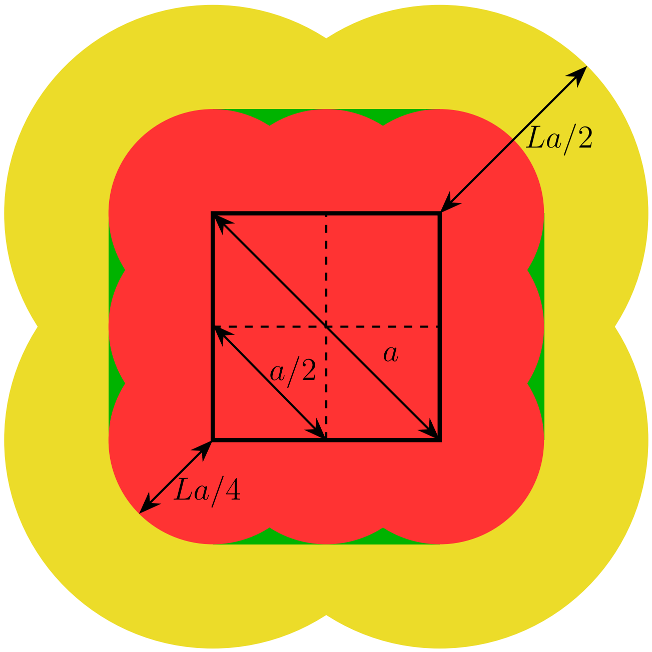


Figure 9.3: illustrations of relevant regions for recursive reduction of list of atom indices associated with cells in Ω_{SAV}, for $L = 1.3$ in two spatial dimensions. Let $\mathcal{R}$ be the red region; let $\mathcal{G}$ be the union of the green region and $\mathcal{R}$; let $\mathcal{Y}$ be the union of the yellow region and $\mathcal{G}$. For a cell C in Ω_{SAV} (solid black rectangle), if there exist indices j such that $B\left(\boldsymbol{c}_j, r_j\right) \cap \mathcal{R} \neq \emptyset$, C needs to be refined and one of its children needs to be refined once more. However, it is easier and computationally much less expensive to check if $B\left(\boldsymbol{c}_j, r_j\right) \cap \mathcal{G}$ is empty or not. If there exist indices j such that $B\left(\boldsymbol{c}_j, r_j\right) \cap \mathcal{G} \neq \emptyset$, C needs to be refined and one of its children *might* need further refinement, too. If $B\left(\boldsymbol{c}_j, r_j\right) \cap \mathcal{G} = \emptyset\, \forall j$, C needs to be refined only if there exists an atom index k such that $B\left(\boldsymbol{c}_k, r_k\right) \cap \mathcal{Y} \neq \emptyset$, but none of its children cells need further refinement if C itself is refined.

considering a brute-force version, *i.e.*, when calculating $\overline{\rho}_{r_\mathrm{p}}$ at all newly created grid nodes (without reduced lists) instead of φ_SAS, only condition (a) is not satisfied.

In order to ensure load balancing, each cell C is given a *computational weight* integer, $\mathcal{W}(C)$, before the grid is partitioned across the P available processes (line 10 of Algorithm 3). The grid-partitioning operation then ensures that the total computational workload per process, $\mathcal{W}_\mathrm{tot}$, is made as uniform as possible. We define $\mathcal{W}_\mathrm{tot}$ for process p as $\mathcal{W}_\mathrm{tot} = \sum_{C\in\mathcal{C}_p}\mathcal{W}(C)$, where $\mathcal{C}_p$ is the set of grid cells attributed to p when partitioning the grid [174, 103]. Since new values of φ_SAS are only calculated at newly created grid nodes by looping through reduced lists of newly created cells (see (9.14)), the computational weight[4]

$$\mathcal{W}_\mathrm{SAS}(C) = \begin{cases} 0 & \text{if the level of grid cell } C \text{ is not the current maximum level,} \\ \left|\mathcal{L}(C)\right| & \text{if the level of grid cell } C \text{ is the current maximum level,} \end{cases} \tag{9.19}$$

is a suitable local measure of the computational workload due to line 11 of Algorithm 3.

However, repartitioning the grid for load-balancing leads to a complication: when a cell C that was initially owned by process p (prior to grid partitioning) is moved to another process $p' \neq p$, its reduced list $\mathcal{L}(C)$ must be communicated from p to p' too. However, reduced lists do not have all the same number of elements; therefore, those cell-level data of non-constant size cannot be redistributed alongside the grid cells by the above partitioning process. In order to be able to find the reduced list associated with any cell C, an unsigned integer tag $T(C)$ is attached to C (and communicated alongside C, if C is). The tag $T(C)$ is defined as

$$T(C) = p2^{s-q} + J(C)\,, \tag{9.20}$$

[4]The notation $|\mathcal{S}|$ represents the cardinality of the set $\mathcal{S}$.

where s is the number of bits used to encode $T(C)$, p is the rank of the process that owns C before grid partitioning, $q = \lceil \log_2(P) \rceil$ is the minimal number of bits necessary to encode the process ranks $\{0, 1, \ldots, P-1\}$ and $J(C)$ is the index of the reduced list associated with C, locally to process p. Assuming that $J(C) < 2^{s-q}$, one can easily find p and $J(C)$ from $T(C)$ by the following bit-wise operations:

$$\begin{aligned} p &= \left(T(C) \,\&\, (2^q - 1)\, 2^{s-q}\right) >> (s-q)\,, && (9.21) \\ J(C) &= \left(T(C) \,\&\, \left(2^{s-q} - 1\right)\right), && (9.22) \end{aligned}$$

where $a \,\&\, b$ represents the bit-wise logical `and` between a and b and $a >> b$ represents the right bitwise shift of a by b bits (the b leftmost bits of a are filled with 0's). Using this technique the appropriate reduced list(s) can be queried by any process that is attributed a cell that belonged to another process prior to the grid partitioning. Algorithm 7 shows the procedure to update values of φ_{SAS} at newly created grid nodes (line 11 of Algorithm 3).

Remark. *The technique from (9.20), (9.21) and (9.22) breaks down if the assumption $J(C) < 2^{s-q}$ is violated. However, its violation would imply that a process owns 2^{s-q} reduced lists at least. Assuming a roughly constant number of reduced lists per process and the existence of at least 2^{q-1} processes, the total number of reduced lists would be $\mathcal{O}(2^{s-1})$, and the computational grid would contain even more cells. If $s = 32$ (resp. 64), $2^{s-1} \simeq 2 \times 10^9$ (resp. 9×10^{18}) which is much larger than any sensible number of grid cells of interest. A warning flag has been implemented in our version of the algorithm: none of our tests have ever activated it.*

When Algorithm 3 has terminated, an adaptive computational grid of maximum level $l_{\max}$ has been created and a function φ_{SAS} sampled at its nodes is available. However, the

Algorithm 7 Calculate the values of φ_{SAS} at newly created grid nodes. p is the rank of the executing process, i is the loop index from Algorithm 3, the current maximum level of refinement of the computational grid is thus $i+1$. `phi_sas` is the (distributed) vector of node-sampled values of φ_{SAS}.

```
1:  function UPDATE_PHI_SAS_AT_NEW_GRID_NODES
2:      initialize an empty query_map, mapping process rank $p'$ to a vector of
            indices of the reduced lists, queried by process $p$ but owned by process $p'$;
3:      initialize an empty map_of_cells_of_remote_lists, mapping process rank $p'$ to
            a vector of indices of cells that are local to process $p$
            and whose reduced lists are owned by process $p'$;
4:      initialize an empty vector locally_known_cell_indices;
5:      $n_{\mathrm{local}} \leftarrow 0$;
6:      for $C \in \mathcal{C}_p$ such that $\mathrm{level}\,(C) == i+1$ do
7:          read cell tag $T\,(C)$;
8:          determine the rank $p'$ of the owner process prior to grid partitioning, using
    (9.21);
9:          if $p' \neq p$ then
10:             determine the index $J\,(C)$ of the reduced list, local to process $p'$, using
    (9.22);
11:             add $J\,(C)$ to the vector query_map $(p')$;
12:             add the local cell index of $C$ to map_of_cells_of_remote_lists $(p')$;
13:         else
14:             add the local cell index of $C$ to locally_known_cells;
15:             $n_{\mathrm{local}} \leftarrow n_{\mathrm{local}} + 1$;
16:         end if
17:     end for
18:     $n_{\mathrm{replies}} \leftarrow 0$;                          ▷ Number of expected replies.
19:     $n_{\mathrm{queries}} \leftarrow 0$;                          ▷ Number of expected queries.
20:     initialize a vector queried_proc of $P$ integers, all equal to 0;
21:     for process rank $p' \neq p$ such that query_map $(p') \neq \emptyset$ do
22:         send the vector query_map $(p')$ to process $p'$;          ▷ Non-blocking.
23:         $n_{\mathrm{replies}} \leftarrow n_{\mathrm{replies}} + 1$;
24:         queried_proc $(p') \leftarrow 1$;
25:     end for
26:     reduce and scatter vectors queried_proc over all processes, to deduce $n_{\mathrm{queries}}$;
```

```
27:     $j \leftarrow 0$;
28:     done $= ((j == n_{\text{local}}) \wedge (n_{\text{replies}} == 0) \wedge (n_{\text{queries}} == 0))$;
29:     while $\neg$done do
30:         if $j < n_{\text{local}}$ then
31:             $C \leftarrow$ grid cell of local index locally_known_cells$(j)$;
32:             find the corresponding reduced list of atom indices $\mathcal{L}(C)$
                    indexed by $J(C)$, found using (9.22);
33:             for $\boldsymbol{v} \in \mathcal{V}(C)$ do
34:                 phi_sas$(\boldsymbol{v}) \leftarrow \max\left(\text{phi\_sas}(\boldsymbol{v}), \max_{j\in\mathcal{L}(C)}(r_{\text{p}} + r_j - \|\boldsymbol{v} - \boldsymbol{c}_j\|)\right)$;   ▷ (9.14)
35:             end for
36:             $j \leftarrow j + 1$;
37:         end if
38:         if $n_{\text{queries}} > 0$ then
39:             probe for any pending incoming query from another process $p' \neq p$;
40:             receive the vector of indices of queried reduced lists from $p'$;   ▷ Blocking.
41:             initialize an empty vector serialized_reply;
42:             for $j \in$ indices of queried lists do
43:                 $\mathcal{L} \leftarrow$ reduced list of local index $j$;
44:                 add $|\mathcal{L}|$ to serialized_reply;                ▷ serialization of the reply.
45:                 add $\mathcal{L}$ to serialized_reply;                  ▷ serialization of the reply.
46:             end for
47:             send serialized_reply to process $p'$;                       ▷ Non-blocking.
48:             $n_{\text{queries}} \leftarrow n_{\text{queries}} - 1$;
49:         end if
50:         if $n_{\text{replies}} > 0$ then
51:             probe for any pending incoming reply from another process $p' \neq p$;
52:             receive the vector serialized_reply from $p'$;                 ▷ Blocking.
53:             deserialize serialized_reply and associate the corresponding reduced lists
                    with the appropriate cells identified in map_of_cells_of_remote_lists;
54:             for cells $C$ identified by map_of_cells_of_remote_lists do
55:                 $\mathcal{L}(C) \leftarrow$ newly added reduced lists from the serialized reply;
56:                 for $\boldsymbol{v} \in \mathcal{V}(C)$ do
57:                     phi_sas$(\boldsymbol{v}) \leftarrow \max\left(\text{phi\_sas}(\boldsymbol{v}), \max_{j\in\mathcal{L}(C)}(r_{\text{p}} + r_j - \|\boldsymbol{v} - \boldsymbol{c}_j\|)\right)$;   ▷ (9.14)
58:                 end for
59:             end for
60:             $n_{\text{replies}} \leftarrow n_{\text{replies}} - 1$;
61:         end if
62:         done $= ((j == n_{\text{local}}) \wedge (n_{\text{replies}} == 0) \wedge (n_{\text{queries}} == 0))$;
63:     end while
64: end function
```

values of φ_{SAS} at grid nodes $\boldsymbol{v}$ belonging to a finest grid cell and such that $0 < \varphi_{\mathrm{SAS}}(\boldsymbol{v}) \leq r_{\mathrm{p}} + m\delta$ are *not* equal to $\operatorname{dist}(\boldsymbol{v}, \Gamma_{\mathrm{SAS}})$, in general, and need correction: in this section we show how to correct those values.

For the construction by reinitialization

The grid and φ_{SAS} are constructed in such a way that $\Omega_{\mathrm{SAV}}^{\mathrm{C}} \backslash \Omega_{\mathrm{SEV}}$ and m layers of width δ over Γ_{SAS} in Ω_{SAV} are tessellated with finest grid cells. Besides, the node values of φ_{SAS} are equal to $\overline{\rho}_{r_{\mathrm{p}}}$ in that region as shown in subsections 9.4.1 and 9.4.1. These conditions allow an accurate calculation of $\operatorname{dist}(\boldsymbol{v}, \Gamma_{\mathrm{SAS}})$ for all grid nodes $\boldsymbol{v} \in \Omega_{\mathrm{SAV}}^{\mathrm{C}} \backslash \Omega_{\mathrm{SEV}}$ by using an m^{th} order accurate reinitialization method, detailed hereafter. We note that this technique does not claim better than first-order accurate results because of the presence of numerous kinks in the definition of Γ_{SAS}. However, one obtains more accurate results when $m = 2$, although the results are still only first-order accurate. The computational efficiency of this technique compared to the construction by direct calculation makes it the method of choice if the user is not interested in higher-order quantities associated with Γ_{SES} like its curvature, for instance.

An accurate implicit representation of Γ_{SES} as the zero-level set of a function $\overline{\phi}$, can be found by subtracting r_{p} from the signed distance ϕ_{SAS} to Γ_{SAS}. This latter node-sampled scalar function can be calculated by reinitializing φ_{SAS} [5, 106], *i.e.*, by solving the following pseudo-time problem for $\varphi(\boldsymbol{x}, \tau)$:

$$\frac{\partial \varphi}{\partial \tau} + \operatorname{sgn}(\varphi_0)\left(\|\nabla \varphi\| - 1\right) = 0, \text{ with } \varphi(\boldsymbol{x}, 0) = \varphi_0 = \overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{x}), \tag{9.23}$$

for $\tau \geq 0$ and where the sgn function is defined as:

$$\operatorname{sgn}(a) = \begin{cases} -1 & \text{if } a < 0, \\ 0 & \text{if } a = 0, \\ 1 & \text{if } a > 0. \end{cases}$$

Usually, a smoothed version of the sgn function like $\operatorname{sgn}(a) = a/\sqrt{a^2 + h^2}$ (where h is the smallest cell size) is used instead, without affecting the correctness of the method. The steady-state (viscosity) solution $\lim_{\tau\to\infty} \varphi(\boldsymbol{x}, \tau)$ has the same sign as $\overline{\rho}_{r_\mathrm{p}}$ and has a normed gradient everywhere: it is thus the signed distance to Γ_SAS, *i.e.*, ϕ_SAS.

The problem (9.23) is solved using the technique from [106]. This technique approximates the solution to (9.23) by solving its corresponding semi-discrete version

$$\frac{\mathrm{d}\varphi}{\mathrm{d}\tau} + \operatorname{sgn}(\varphi_0)\left[H\left(\nabla^+\varphi, \nabla^-\varphi, \varphi\right) - 1\right] = 0, \text{ with } \varphi(\boldsymbol{v}, 0) = \varphi_0 = \varphi_\mathrm{SAS}(\boldsymbol{v}), \tag{9.24}$$

where $\nabla^+\varphi$ and $\nabla^-\varphi$ are m^th-order accurate gradients of φ (when φ is smooth) based on one-sided forward and backward derivatives, respectively, and where the Godunov Hamiltonian H is defined as

$$H(\boldsymbol{a}, \boldsymbol{b}, \beta) = \begin{cases} \sqrt{\sum_{k=1}^{3} \max\left((\max(a_k, 0))^2, (\min(b_k, 0))^2\right)} & \text{if } \beta \leq 0, \\ \sqrt{\sum_{k=1}^{3} \max\left((\min(a_k, 0))^2, (\max(b_k, 0))^2\right)} & \text{if } \beta > 0, \end{cases} \tag{9.25}$$

where a_k and b_k are the k^th components of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ respectively. The equation (9.24) is then integrated in time using a m^th-order TVD Runge-Kutta scheme with adaptive time-stepping [267]. Details related to the sub-cell fix method to prevent the motion of the interface and to non-uniform cartesian grids have been omitted since they are beyond the scope of this work [184, 255]. The interested reader is referred to [247] for a recent review of level-set methods.

We note that the Fast Sweeping Method (FSM) [268, 269, 270, 271] and the Fast Marching Method (FMM) [272, 273, 274, 275, 276] are known to have optimal complexity for solving Hamilton-Jacobi equations like (9.23) and should thus be preferred over the above method, in principle. However, although serial versions of those methods have been implemented successfully on adaptive Quad-/Oc-trees [277], the implementation of a strongly scalable parallel version, in the paradigm of distributed computing, is still an open problem.

Note that we have $\varphi_{\mathrm{SAS}} = \overline{\rho}_{r_{\mathrm{p}}}$ when $\overline{\rho}_{r_{\mathrm{p}}} \geq -m\delta$ (by construction) and that $\overline{\rho}_{r_{\mathrm{p}}}$ is already a signed distance function wherever it is negative, by Theorem 9.3.2. Moreover, the correct distance-based grid levels have already been determined in the region $\{\boldsymbol{z} \in \mathbb{R}^3 | \varphi_{\mathrm{SAS}}(\boldsymbol{z}) < -m\delta\}$ (by construction too). Therefore, the value of ϕ_{SAS} needs to be calculated only in the region $\mathcal{P}_{\varphi_{\mathrm{SAS}}}$ where the set of points $\mathcal{P}_f$ is defined as

$$\mathcal{P}_f = \left\{\boldsymbol{x} \in \mathbb{R}^3 | f(\boldsymbol{x}) > 0\right\}. \tag{9.26}$$

Hence, equation (9.24) is evolved in time only at grid nodes $\boldsymbol{z} \in \mathcal{P}_{\varphi_{\mathrm{SAS}}}$ in order to keep the computational workload minimal and load balancing is ensured by partitioning the grid according to the local computational weight $\mathcal{W}_{\mathcal{P}_{\varphi_{\mathrm{SAS}}}}(C)$ where $\mathcal{W}_{\mathcal{P}_f}(C)$ is defined as

$$\mathcal{W}_{\mathcal{P}_f}(C) = \begin{cases} 0 & \text{if } \mathcal{V}(C) \bigcap \mathcal{P}_f = \emptyset, \\ 1 & \text{else,} \end{cases} \tag{9.27}$$

before termination of Algorithm 3 (see line 14 of Algorithm 3).

Finally, a *finite* final resolution pseudo-time τ_{end} must be determined when solving (9.23). The analysis of (9.23) by the method of characteristics shows that the solution $\varphi(\boldsymbol{x}, \tau)$ propagates from $\varphi_0^{-1}(0)$, *i.e.*, the zero-level set of φ_0, along its normal with a pseudo-velocity of magnitude one. Therefore, after a pseudo-time $\tau > 0$, we have

$$|\varphi(\boldsymbol{x},\tau)| = \operatorname{dist}\left(\boldsymbol{x},\varphi_0^{-1}(0)\right), \quad \forall \boldsymbol{x} \text{ such that } \operatorname{dist}\left(\boldsymbol{x},\varphi_0^{-1}(0)\right) < \tau.$$

As a consequence, the final pseudo-time when solving (9.23) must be strictly greater than $r_{\mathrm{p}} + m\delta$ to ensure the accurate localization of the r_{p}-level set of ϕ_{SAS}. It has been found (through extensive tests including the cases illustrated below) that the final pseudo-time $\tau_{\mathrm{end}} = 3(r_{\mathrm{p}} + m\delta)$ ensures convergence of the calculations. The number of pseudo-time steps for solving (9.24) is thus set to the smallest integer greater than or equal to $\dfrac{3(r_{\mathrm{p}} + m\delta)}{\Delta\tau}$ where pseudo-time step $\Delta\tau$ satisfies the CFL condition $\Delta\tau = h/(m+1)$, $h = h_{\mathrm{root}} 2^{-l_{\max}}$ being the smallest grid size[5].

Once ϕ_{SAS} is calculated by the above procedure, an implicit representation of Γ_{SES} is given by its r_{p}-level set or, equivalently, the zero-level set of $\overline{\phi} = \phi_{\mathrm{SAS}} - r_{\mathrm{p}}$. Figure 9.5 illustrates the resulting representation of Γ_{SES} as the zero-level set of $\overline{\phi}$ for our two-dimensional illustration.

In this case, the grid is constructed in such a way that ϕ_{SAS}, *i.e.*, the exact distance to Γ_{SAS}, needs to be calculated at every grid node $\boldsymbol{v}$ that belongs to one of the finest cells. Our procedure uses calculations similar to [263, 278], that we develop and illustrate here after. Let $\boldsymbol{\xi} \in \Gamma_{\mathrm{SAS}}$ be the closest point to $\boldsymbol{v}$, the goal of this approach is to find $\boldsymbol{\xi}$ for all grid nodes $\boldsymbol{v}$ such that $\operatorname{dist}(\boldsymbol{v},\Gamma_{\mathrm{SAS}}) < r_{\mathrm{p}} + m\delta$, and set $\phi_{\mathrm{SAS}}(\boldsymbol{v}) = \|\boldsymbol{v} - \boldsymbol{\xi}\|$.

As showed in subsection 9.4.1, every finest cells C is associated with a reduced list containing only the indices of atoms that are closer than $r_{\mathrm{p}} + m\delta$ from C (see line 9 and lines 13 to 16 in Algorithm 6). Therefore, it contains all atom indices i that are potential candidates for satisfying $\boldsymbol{\xi} \in \partial B(\boldsymbol{c}_i, r_i + r_{\mathrm{p}})$. The following procedure intends to find $\boldsymbol{\xi}$ and is guaranteed to be exact for all grid nodes $\boldsymbol{v}$ such that $\overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{v}) \geq 0$ and

[5]This is true in that context despite the adaptive time stepping (*i.e.*, $\Delta\tau$ being proportional to the *local* smallest grid size) because of $\Omega_{\mathrm{SAV}}^{\mathrm{C}} \backslash \Omega_{\mathrm{SEV}}$ has been tessellated with the finest desired grid cells.

$\operatorname{dist}(\boldsymbol{v}, \Gamma_{\mathrm{SAS}}) < r_{\mathrm{p}} + m\delta$, by construction of the reduced lists. If the method fails to find $\boldsymbol{\xi}$, it means that $\operatorname{dist}(\boldsymbol{v}, \Gamma_{\mathrm{SAS}}) > r_{\mathrm{p}} + m\delta$ and $\boldsymbol{v}$ is thus irrelevant for the accurate localization of Γ_{SES}.

Given a finest cell C, its reduced list of atoms $\mathcal{L}(C)$, and a grid node $\boldsymbol{v} \in \mathcal{V}(C)$, consider any pair of intersecting spheres $\partial B(\boldsymbol{c}_i, r_i + r_{\mathrm{p}})$ and $\partial B(\boldsymbol{c}_j, r_j + r_{\mathrm{p}})$, where $\{i, j\} \in \mathcal{L}(C)$, as illustrated in Figure 9.4. The intersection of the two spheres is a circle of center

$$\boldsymbol{o} = \lambda \boldsymbol{c}_i + (1-\lambda)\, \boldsymbol{c}_j, \text{ where } \lambda = \frac{1}{2} + \frac{(r_{\mathrm{p}} + r_j)^2 - (r_{\mathrm{p}} + r_i)^2}{2\,\|\boldsymbol{c}_j - \boldsymbol{c}_i\|^2}, \tag{9.28}$$

of radius

$$\zeta = \left(\frac{(r_j + r_{\mathrm{p}})^2 + (r_i + r_{\mathrm{p}})^2}{2} - \frac{\|\boldsymbol{c}_j - \boldsymbol{c}_i\|^2}{4} - \frac{\left((r_j + r_{\mathrm{p}})^2 - (r_i + r_{\mathrm{p}})^2\right)^2}{4\,\|\boldsymbol{c}_j - \boldsymbol{c}_i\|^2} \right)^{1/2}, \tag{9.29}$$

and contained in a plane of normal vector $\boldsymbol{n} = \dfrac{\boldsymbol{c}_j - \boldsymbol{c}_i}{\|\boldsymbol{c}_j - \boldsymbol{c}_i\|}$. Let

$$\boldsymbol{\mu} = \frac{(\boldsymbol{v} - \boldsymbol{o}) - ((\boldsymbol{v} - \boldsymbol{o}) \cdot \boldsymbol{n})\, \boldsymbol{n}}{\|(\boldsymbol{v} - \boldsymbol{o}) - ((\boldsymbol{v} - \boldsymbol{o}) \cdot \boldsymbol{n})\, \boldsymbol{n}\|} \text{ and } \boldsymbol{\nu} = \boldsymbol{n} \times \boldsymbol{\mu} \tag{9.30}$$

be two vectors that are orthogonal to $\boldsymbol{n}$, all points on $\partial B(\boldsymbol{c}_i, r_i + r_{\mathrm{p}}) \bigcap \partial B(\boldsymbol{c}_j, r_j + r_{\mathrm{p}})$ can be parameterized with $\vartheta \in [-\pi, \pi[$ by

$$\boldsymbol{s}(\vartheta) = \boldsymbol{o} + \zeta \cos(\vartheta)\, \boldsymbol{\mu} + \zeta \sin(\vartheta)\, \boldsymbol{\nu}. \tag{9.31}$$

Note that $\|\boldsymbol{v} - \boldsymbol{s}(\vartheta)\|$ is an even function of ϑ that is minimal for $\vartheta = 0$ and strictly increasing with $|\vartheta|$. Therefore, we intend to find the smallest value of $|\vartheta| \in [0, \pi]$ such that $\overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{s}(|\vartheta|)) = 0$ or $\overline{\rho}_{r_{\mathrm{p}}}(\boldsymbol{s}(-|\vartheta|)) = 0$.

A point $\boldsymbol{s}(\vartheta)$ is *not* valid if there exists at least one atom index $k \in \mathcal{L}(C)$ such that $\rho_{r_{\mathrm{p}},k}(\boldsymbol{s}(\vartheta)) > 0$. In such a case, $\boldsymbol{s}(\vartheta)$ belongs to the following set of points

$$\left(\partial B(\boldsymbol{c}_i, r_i + r_{\mathrm{p}}) \bigcap \partial B(\boldsymbol{c}_j, r_j + r_{\mathrm{p}})\right) \bigcap B(\boldsymbol{c}_k, r_k + r_{\mathrm{p}}) \tag{9.32}$$

which should all be rejected as potential candidates for the closest point to $\boldsymbol{v}$ on Γ_{SAS}. Let

$$\boldsymbol{c}'_k = \boldsymbol{c}_k - ((\boldsymbol{c}_k - \boldsymbol{o}) \cdot \boldsymbol{n})\,\boldsymbol{n} \tag{9.33}$$

be the projection of $\boldsymbol{c}_k$ onto the plane of normal $\boldsymbol{n}$ through $\boldsymbol{o}$ and

$$r'_k = \sqrt{(r_k + r_{\mathrm{p}})^2 - \|\boldsymbol{c}'_k - \boldsymbol{c}_k\|^2} \tag{9.34}$$

be the radius of the intersection of $\partial B(\boldsymbol{c}_k, r_k + r_{\mathrm{p}})$ in that plane. All points in the set (9.32) can be parameterized by $\boldsymbol{s}(\vartheta)$ with $\theta \in \,]\beta - \alpha, \beta + \alpha[$, the angles β and α being

$$\beta = \begin{cases} \arccos\left(\dfrac{(\boldsymbol{c}'_k - \boldsymbol{o}) \cdot \boldsymbol{\mu}}{\|\boldsymbol{c}'_k - \boldsymbol{o}\|}\right) & \text{if } (\boldsymbol{c}'_k - \boldsymbol{o}) \cdot \boldsymbol{\nu} \geq 0, \\ -\arccos\left(\dfrac{(\boldsymbol{c}'_k - \boldsymbol{o}) \cdot \boldsymbol{\mu}}{\|\boldsymbol{c}'_k - \boldsymbol{o}\|}\right) & \text{if } (\boldsymbol{c}'_k - \boldsymbol{o}) \cdot \boldsymbol{\nu} < 0, \end{cases} \tag{9.35}$$

$$\alpha = \arccos\left(\frac{\|\boldsymbol{c}'_k - \boldsymbol{o}\|^2 + \zeta^2 - (r'_k)^2}{2\,\|\boldsymbol{c}'_k - \boldsymbol{o}\|\,\zeta}\right), \tag{9.36}$$

as illustrated in Figure 9.4.

Algorithm 13 in the appendix 9.7 uses the above results to find the closest point from $\boldsymbol{v}$ on Γ_{SAS} from the reduced list of atoms indices. We note that the complexity of Algorithm 13 is $\mathcal{O}\left(|\mathcal{L}(C)|^3\right)$. However, we point out that

- $|\mathcal{L}(C)| = \mathcal{O}(1)$ when the grid cells are fine enough;
- most loops in Algorithm 13 can be shortcut if a good, valid candidate point $\hat{\boldsymbol{\xi}} \in \Gamma_{\mathrm{SAS}}$

is already known and close to the actual $\boldsymbol{\xi} \in \Gamma_{\mathrm{SAS}}$ that is sought: $\boldsymbol{\xi}$ cannot belong to a ball that is farther away than $\|\hat{\boldsymbol{\xi}} - \boldsymbol{v}\|$ from $\boldsymbol{v}$. Since nodes and quadrants are locally and globally ordered using a Z-order code (Morton code) [279, 174], locality of successive nodes is ensured and the point $\boldsymbol{\xi}$ found for a grid node $\boldsymbol{v}$ is a good guess $\hat{\boldsymbol{\xi}}$ for the next investigated grid node, which helps reduce the workload.

We reiterate that this method is *not* claimed to be optimized, but it gives us an exact comparison basis to validate the construction by reinitialization and show its accuracy.

9.4.2 Identification of internal cavities

As pointed out in Remark 9.2, the representation of Γ_{SES} by the zero-level set of $\overline{\phi}$ does not prevent the existence of internal cavities. In other words, one may find compact

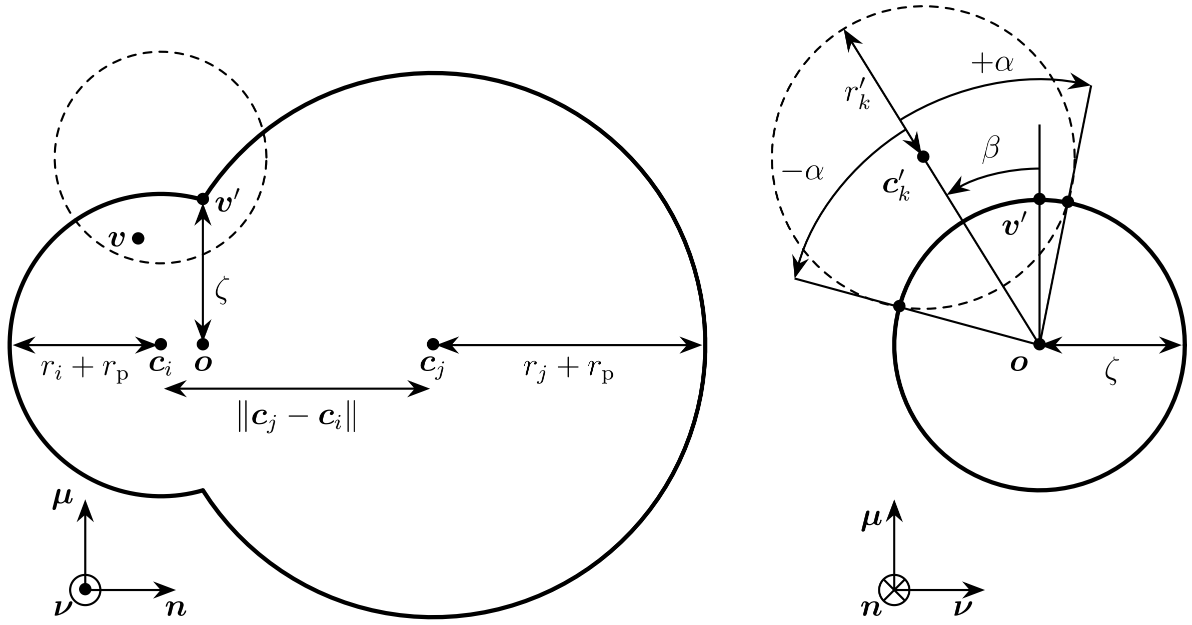


Figure 9.4: illustrations for the strategy to find the closest point on Γ_{SAS} from a grid node $\boldsymbol{v}$. Left: image of the intersection between two spheres $\partial B(\boldsymbol{c}_i, r_i + r_{\mathrm{p}})$ and $\partial B(\boldsymbol{c}_j, r_j + r_{\mathrm{p}})$ in the plane containing $\boldsymbol{c}_i$, $\boldsymbol{c}_j$ and $\boldsymbol{v}$; Right: image of $\partial B(\boldsymbol{c}_i, r_i + r_{\mathrm{p}}) \bigcap \partial B(\boldsymbol{c}_j, r_j + r_{\mathrm{p}})$ in its own plane. The dashed circles represent the intersection of another candidate sphere $\partial B(\boldsymbol{c}_k, r_k + r_{\mathrm{p}})$ in the respective planes. Note that we use $\boldsymbol{v}' = \boldsymbol{s}(0)$.

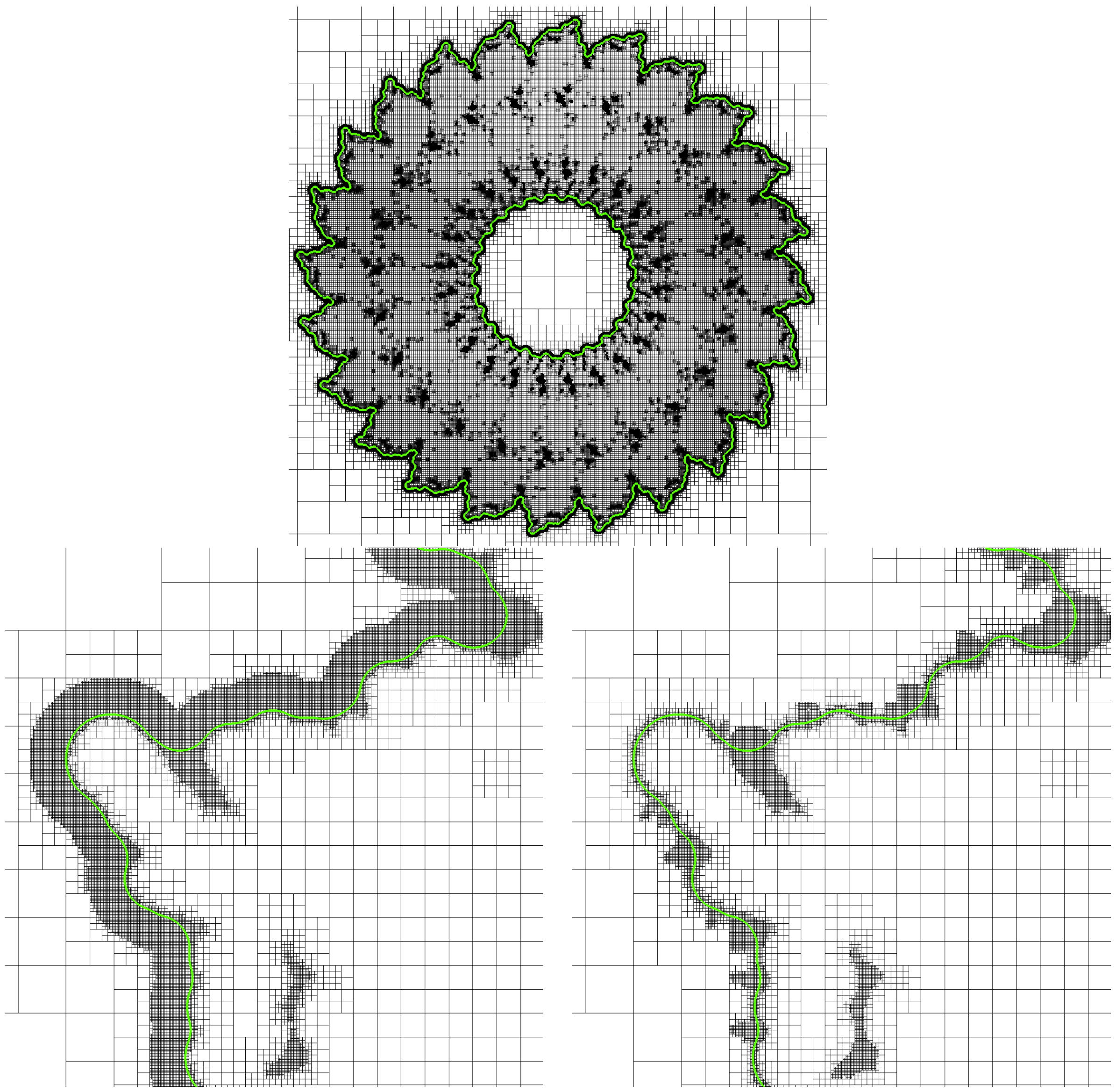


Figure 9.5: illustration of the representation of Γ_{SES} as the zero-level set of $\overline{\phi}$, as defined in section 9.4.1. Note that the kinks from Γ_{SAS} (as in Figure 9.2) have given birth to smooth portions of circular interfaces. Top: the implicit representation of Γ_{SES} as defined in section 9.4.1. Bottom left: zoom-in close to Γ_{SES} for the construction by reinitialization. Bottom right: zoom-in close to Γ_{SES} for the construction by direct calculation. (The reader is invited to compare these figures to Figure 9.2)

sets of points $\boldsymbol{z}$ where $\overline{\phi}\left(\boldsymbol{z}\right) \leq 0$ that are encapsulated in $\Omega_{\text{SEV}} = \left\{\boldsymbol{x} \in \mathbb{R}^3 | \overline{\phi}\left(\boldsymbol{x}\right) > 0\right\}$. Such sets of points require a special treatment as they are internal cavities: those regions are not connected to the outer domain and no solvent molecule can reach them. In the current context, our treatment consists of simply removing all such cavities: let $\mathcal{Q}$ be the set of grid nodes identified as belonging to a cavity, the sign of the node values of $\overline{\phi}\left(\boldsymbol{v}\right)$ is reversed for all grid nodes $\boldsymbol{v} \in \mathcal{Q}$.

We note that the method presented in Algorithm 8 identifies $\mathcal{Q}$ without distinction of all its simply connected subsets in a robust and efficient way, as this is sufficient for the above purpose. If one needs to decompose $\mathcal{Q}$ in all its simply connected subsets, additional operations are required. However, this goes beyond the scope of this work and we refer the interested reader to the island numbering strategy from [280] in the context of the Island Dynamics model of [281, 282, 283]. Let $\mathcal{N}$ be the subset of grid nodes from $\Omega_{\text{SEV}}^{\text{C}} = \mathbb{R}^3 \backslash \Omega_{\text{SEV}}$ that are simply connected to the boundary of the computational domain, then $\mathcal{Q} = \Omega_{\text{SEV}}^{\text{C}} \backslash \mathcal{N}$. Our method starts by marking all grid nodes in $\mathcal{N}$ using a region-growing technique from Algorithm 9, explained here below. Then, the sign of the node-sampled function $\overline{\phi}$ is reversed at all *unmarked* grid nodes $\boldsymbol{v}$ such that $\overline{\phi}\left(\boldsymbol{v}\right) < 0$.

As the lists of atom centers and radii are read and stored in memory, the centroids $\boldsymbol{C}_{\mathcal{M}}$ of every molecule $\mathcal{M}$, defined as:

$$\boldsymbol{C}_{\mathcal{M}} = \frac{1}{n_{\mathcal{M}}} \sum_{i=1}^{n_{\mathcal{M}}} \boldsymbol{c}_i, \tag{9.37}$$

are calculated on-the-fly. Then a second pass through the lists of atoms allows the calculation of the minimum side lengths of virtual cubes $\Xi_{\mathcal{M}}$ of centers $\boldsymbol{C}_{\mathcal{M}}$, and such that $\mathcal{M}$ is entirely contained in $\Xi_{\mathcal{M}}$. Each cube $\Xi_{\mathcal{M}}$ is virtually attached to molecule $\mathcal{M}$: it moves and scales with $\mathcal{M}$ when $\mathcal{M}$ is translated, rotated or scaled. Those cubes allow for a straightforward and robust identification of a rather big subset of $\mathcal{N}$.

Algorithm 9 starts by tagging all grid nodes that do not belong to any cube $\Xi_{\mathcal{M}}$ as members of $\mathcal{N}$ (line 2). Then, every process p loops through every grid node $\boldsymbol{v}$ in its domain partition that is not tagged yet. If $\boldsymbol{v}$ has a neighbor grid node in $\mathcal{N}$ and $\overline{\phi}(\boldsymbol{v}) < 0$, then $\boldsymbol{v}$ is tagged as member of $\mathcal{N}$ (see Algorithm 10). The tags associated with layering grid nodes are updated between two iterations using non-blocking communications: the nodes owned by process p that are close to the border of its domain partition are ghost nodes for (an)other process(es) p', their tags must be updated on process(es) p' (see lines 9 and 13). The method terminates when no new grid node is tagged. Figure 9.6 illustrates the identification of cavities.

Algorithm 8 Removal of internal cavities identified by complement of outer domain.

```
1: function REMOVAL_OF_INTERNAL_CAVITIES($\overline{\phi}$)
2:     tag all grid nodes $\boldsymbol{v} \in \mathcal{N}$;                    ▷ See Algorithm 9.
3:     for local grid nodes $\boldsymbol{v}$ that are not tagged and such that $\overline{\phi}(\boldsymbol{v}) < 0$ do
4:         $\overline{\phi}(\boldsymbol{v}) \leftarrow -\overline{\phi}(\boldsymbol{v})$;
5:     end for
6: end function
```

After step 3 of the global grid construction algorithm, the zero-level set of $\overline{\phi}$ is an implicit representation of a cavity-free Γ_{SES}. However, as pointed out in section 9.4.1, the computational grid contains too many fine cells, *i.e.*, cells violating the coarsening criterion (9.2). As such fine grid cells would dramatically and unnecessarily increase the corresponding number of unknowns in further calculations, we intend to remove them by a succession of coarsening steps.

Eight sibling cells are grouped together and replaced by their parent cells if criterion (9.2) is satisfied for all eight sibling cells together. For a given cell C, determining whether (9.2) is true or not generally requires to know the greatest lower bound of the distance to Γ_{SES} for all vertices $\boldsymbol{v} \in \mathcal{V}(C)$. In the context of the construction by reinitialization, we showed in subsection 9.4.1 that the correct distance-based grid levels are already

Algorithm 9 Identification of $\mathcal{N}$ by region-growing. See Algorithm 10 for lines 7 and 11.

```
 1: function TAG_ALL_GRID_NODES_IN_OUTER_DOMAIN($\overline{\phi}$)
 2:     tag all grid nodes $\boldsymbol{v}$ such that $\boldsymbol{v} \notin \Xi_{\mathcal{M}}, \forall \mathcal{M}$;
 3:     not_converged ← true;
 4:     while not_converged do
 5:         not_converged ← false;
 6:         for untagged $\boldsymbol{v} \in$ layer nodes do
 7:             not_converged ← is_point_in_outer_domain ($\boldsymbol{v}$);
 8:         end for
 9:         begin to update tags of layer grid nodes;        ▷ Collective and non-blocking.
10:         for untagged $\boldsymbol{v} \in$ local inner nodes do
11:             not_converged ← is_point_in_outer_domain ($\boldsymbol{v}$);
12:         end for
13:         finish updating tags of layer grid nodes;        ▷ Collective and non-blocking.
14:         collective reduction of not_converged
                (logical or over all process' not_converged variables);
15:     end while
16: end function
```

Algorithm 10 Determine if grid node $\boldsymbol{v} \in \mathcal{N}$.

```
 1: function IS_POINT_IN_OUTER_DOMAIN($\boldsymbol{v}$)
 2:     if $\overline{\phi}(\boldsymbol{v}) \geq 0$ then
 3:         return false;
 4:     end if
 5:     get neighbor grid nodes of $\boldsymbol{v}$;
 6:     for vertices $\boldsymbol{v}'$ in neighbor grid nodes of $\boldsymbol{v}$ do
 7:         if $\boldsymbol{v}'$ is tagged then
 8:             tag $\boldsymbol{v}$;
 9:             return true;
10:         end if
11:     end for
12:     return false;
13: end function
```

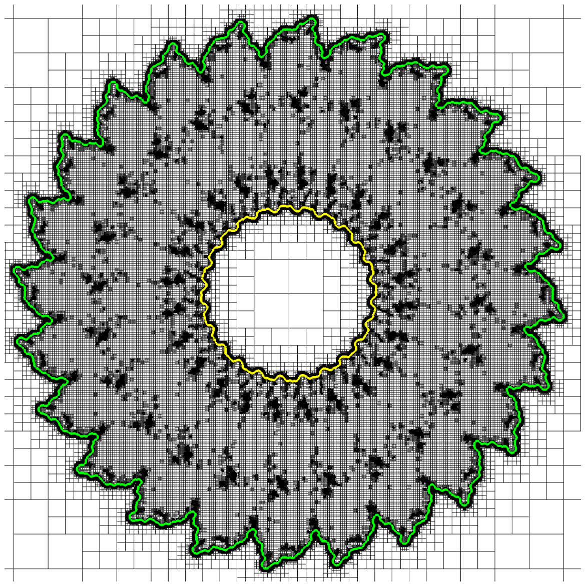


Figure 9.6: illustration of the cavity-identification method for the illustrative planar molecule: the zero-level set of $\overline{\phi}$ (after cavity removal) is the green contour line while the yellow contour line circles all grid nodes that have been marked as cavity nodes. We emphasize that this illustrative result cannot be extrapolated back to the actual three-dimensional `3J6D` protein: this two-dimensional cavity corresponds to the planar projection of a hole in the three-dimensional toroidal topology of protein `3J6D`. The cavities of the actual three-dimensional protein are illustrated in Figure 9.11. The reader is invited to compare this figure to Figure 9.5 (top).

determined for all cells in the outer domain that are farther than $m\delta$ away from Γ_{SAS} or, equivalently, farther than $m\delta + r_{\mathrm{p}}$ away from Γ_{SES}. Furthermore, the calculation of $\overline{\phi}$ by reinitialization of φ_{SAS} (see section 9.4.1) ensures that $\overline{\phi}$ is an accurate signed distance function to Γ_{SES} at all grid nodes $\boldsymbol{v}$ such that $-(m\delta + r_{\mathrm{p}}) \leq \overline{\phi}(\boldsymbol{v}) \leq m\delta$. The same conclusion holds for the construction by direct calculation. As a consequence, one only needs to estimate the local distance to Γ_{SES} for grid nodes in $\mathcal{P}_{\overline{\phi}-m\delta}$, *i.e.*, the grid nodes $\boldsymbol{z}$ where $\overline{\phi}(\boldsymbol{z}) - m\delta > 0$.

The method from Algorithm 11 uses a *recursive layer-by-layer* approach to coarsen the grid. Starting from a stage index $k = 0$, at each stage k the method intends to estimate the distance to Γ_{SES} for one new layer of grid nodes in $\mathcal{P}_{\overline{\phi}}$. Since accuracy is irrelevant for that purpose, we choose to apply the less resource-intensive first-order accurate reinitialization method for solving (9.23) for all grid nodes $\boldsymbol{z} \in \mathcal{P}_{\overline{\phi}-m\delta}$. The layer of the new width is $L\,\delta 2^k$, as dictated by (9.2), so that the number of iterations at stage k for solving (9.23) until pseudo-time $\tau_{\mathrm{end},k} = L\,\delta 2^k$ must be greater than $\dfrac{\tau_{\mathrm{end},k}}{\Delta\tau_k}$, where $\Delta\tau_k$ is the pseudo-time step at stage k. Since the method works recursively layer by layer, one can consider $\Delta\tau_k$ as the minimum of the adaptive pseudo-time steps in $\mathcal{P}_{\overline{\phi}-L\,\delta 2^k}$ where the current coarsening method ensures that the finest grid cells are of level $l_{\max} - k$. Hence, $\Delta\tau_k = \dfrac{h_{\mathrm{root}} 2^{-(l_{\max}-k)}}{2} = h2^{k-1}$, so that the theoretical minimum number of iterations becomes:

$$\frac{L\,\delta 2^k}{h2^{k-1}} = 2L\,\frac{\delta}{h}, \tag{9.38}$$

which is independent of k. In practice, we use the smallest integer greater than or equal to $3L\,\delta/h$. Note that $\delta/h = \sqrt{3}$ for a cubic computational domain.

As such, each stage k of the above procedure ensures that $\overline{\phi}(\boldsymbol{z})$ is a reliable estimate of the distance to Γ_{SES} at all grid nodes $\boldsymbol{z}$ such that $\text{dist}\left(\boldsymbol{z}, \Gamma_{\text{SES}}\right) \leq L\,\delta 2^k$. However, let

$$\mathcal{I}_k = \left\{ \boldsymbol{x} \in \mathcal{P}_{\overline{\phi}} \mid \text{dist}\left(\boldsymbol{x}, \Gamma_{\text{SES}}\right) \geq L\,\delta 2^k \right\}, \tag{9.39}$$

the values of $\overline{\phi}\left(\boldsymbol{z}\right)$ at grid nodes $\boldsymbol{z} \in \mathcal{I}_k$ are still undetermined: supposedly, $\overline{\phi}\left(\boldsymbol{z}\right) \geq L\,\delta 2^k$ but this is not ensured by the above procedure. As a consequence, $\overline{\phi}$ is not reliable in $\mathcal{I}_k$ with respect to the implementation of (9.2) from Algorithm 12. Nevertheless, all grid nodes $\boldsymbol{z} \in \mathcal{I}_k$ such that $\overline{\phi}\left(\boldsymbol{z}\right) < L\,\delta 2^k$, *i.e.*, grid nodes that would be false negatives with respect to Algorithm 12, can be easily identified as cavities if one subtracts $L\,\delta 2^k$ from $\overline{\phi}$, *temporarily*. Once false negatives are identified, the local sign of the temporary $\overline{\phi}$ is reversed[6]. Then, $L\,\delta 2^k$ is added back to the temporary $\overline{\phi}$: all the false negative grid nodes have been removed and the implicit representation of Γ_{SES} is unaffected. This latter maneuver (lines 6 to 8 in Algorithm 11) can be viewed as an accelerator of the reinitialization procedure. Figure 9.7 and Figure 9.8 illustrate some of the coarsening steps for our two-dimensional illustration.

9.4.3 Imposing the desired minimum level of refinement

The final operation of our general algorithm consists of imposing the desired minimum level $l_{\min}$ of refinement in the computational grid. Imposing a minimum level of refinement might be desired for the purpose of ensuring accuracy of further calculations that would be performed on the computational grid, for instance when solving the Poisson-Boltzmann equation in order to determine the electrostatic potential Ψ in the domain.

Except for Algorithm 12, we have purposefully omitted to ensure that no computa-

[6]If another ad-hoc treatment than simply removing all cavities is required (see section 9.4.2), then one can assume that relevant cavities to be kept were uniquely tagged in the previous step of the global algorithm (see [280] for a tag-numbering technique, using level-set methods and distributed computing). Then, one would reverse the sign of $\overline{\phi}$ only for false negative grid nodes that were *not* tagged as cavities.

Algorithm 11 Coarsen the grid based on criterion (9.2). `phi` represents the (distributed) vector of node-sampled values of $\overline{\phi}$.

```
 1: function COARSENING_PROCEDURE
 2:     grid_has_changed ← true;
 3:     $k \leftarrow 0$;
 4:     while grid_has_changed do
 5:         run $\lceil 3L\,\delta/h \rceil$ iterations of the first-order accurate
                reinitialization procedure for all grid nodes $\boldsymbol{z} \in \mathcal{P}_{\overline{\phi}-m\delta}$;
 6:         phi ← phi − $L\,\delta 2^k$;
 7:         call Algorithm 8;
 8:         phi ← phi + $L\,\delta 2^k$;
 9:         coarsen all local families of eight sibling cells that were
                all marked true by Algorithm 12 and update grid_has_changed;
10:         collective reduction of grid_has_changed
                (logical or over all process' grid_has_changed variables);
11:         if grid_has_changed then
12:             fine_phi ← phi;
13:             create a new node-sampled function phi;
14:             scatter appropriate values from fine_phi to phi;
15:             destroy fine_phi and release corresponding memory;
16:             partition the grid over the pool of processes using local computational
                    weight $\mathcal{W}_{\mathcal{P}_{\overline{\phi}}}(C)$, enforcing sibling cells of the same level to belong
                    to the same grid partition, and update phi's layout accordingly;
17:         end if
18:         $k \leftarrow k + 1$;
19:     end while
20: end function
```

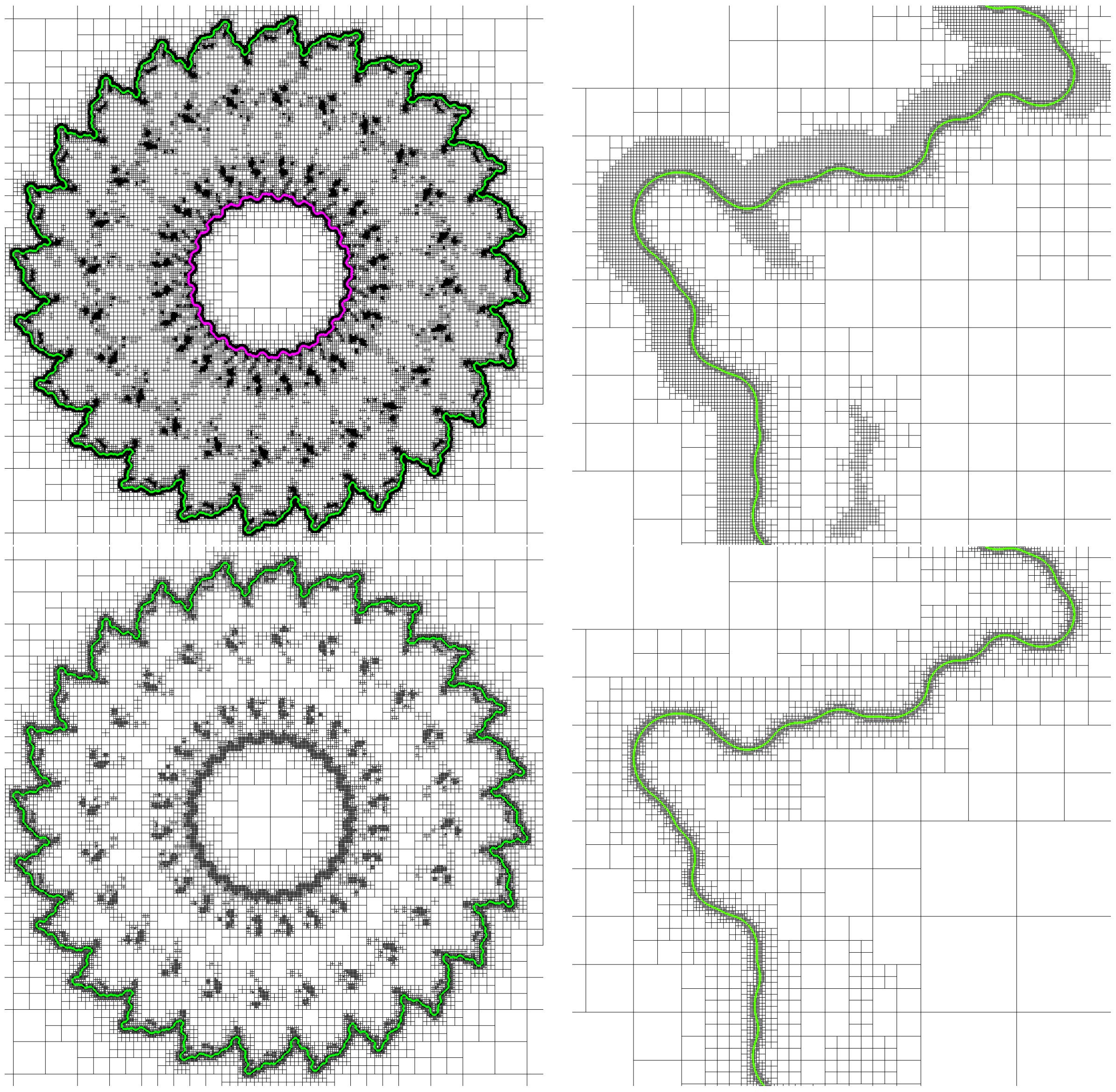


Figure 9.7: computational grid after the first and third coarsening stages for the illustrative planar molecule. The green contour line represents the cavity-free Γ_{SES}. Top left: computational grid after the first coarsening stage. Grid nodes affected by the acceleration maneuver are highlighted in pink. Top right: zoom-in close to the interface. Bottom left: computational grid after the third coarsening stage. Bottom right: zoom-in close to the interface.

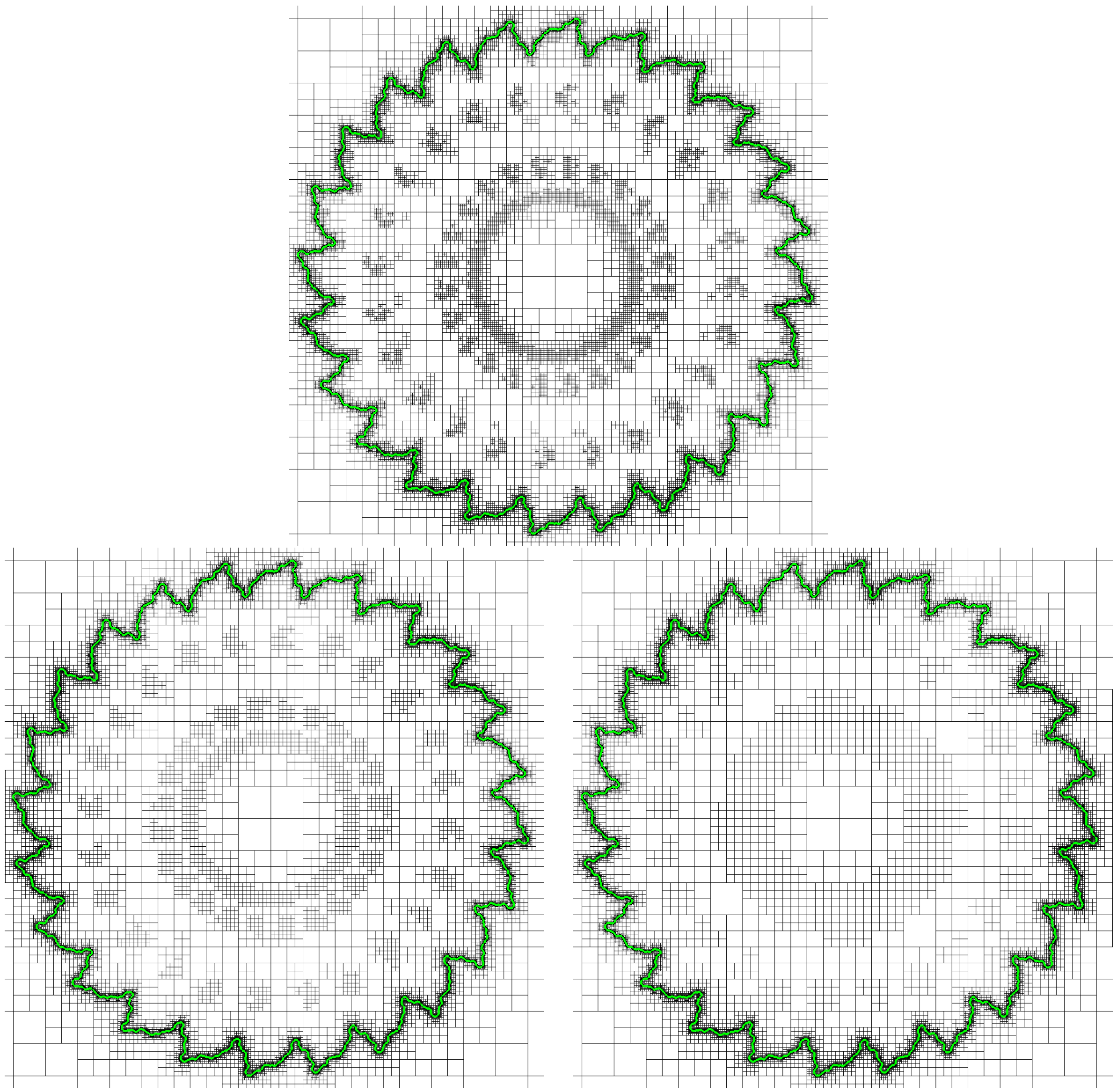


Figure 9.8: computational grid after the fourth, fifth and sixth coarsening stages for the illustrative planar molecule.

Algorithm 12 Coarsening criterion based on criterion (9.2) and a desired minimum refinement level $l_{\min}$. Note that the distance-based criterion (9.2) is shortcut by line 6 since it is irrelevant to coarsen cells that are meant to be refined again eventually.

```
1: function COARSENING_CRITERION(family of eight sibling cells of the same level $l$, $l_{\min}$, $L$)
2:     if $l \leq l_{\min}$ then
3:         return false;
4:     else
5:         for cell $C$ in the family of cells do
6:             for $\boldsymbol{v} \in \mathcal{V}(C)$ do
7:                 if $\overline{\phi}(\boldsymbol{v}) \leq L\, D_{\text{root}}\, 2^{-l}$ then
8:                     return false;
9:                 end if
10:            end for
11:        end for
12:    end if
13:    return true;
14: end function
```

tional cell has a refinement level smaller than $l_{\min}$ in order to keep the workload and the memory requirement at its bare minimum. Therefore, we eventually refine (recursively) every grid cell C whose refinement level would be smaller than $l_{\min}$. Every cell of level $l < l_{\min}$ is thus replaced by its $2^{3(l_{\min}-l)}$ descendant cells. The methodology presented in the previous subsections ensures that the maximum refinement level is imposed in $\mathcal{A} = \{\boldsymbol{x} \in \mathbb{R}^3 \mid \text{dist}(\boldsymbol{x}, \Gamma_{\text{SES}}) \leq m\delta\}$ and that the node values $\overline{\phi}(\boldsymbol{z})$ are accurate or exact signed distances to Γ_{SES} for all nodes $\boldsymbol{z} \in \mathcal{A}$. Therefore, the node values of $\overline{\phi}$ at the newly created grid nodes are irrelevant accuracy-wise and they are evaluated by simple linear interpolation using the vertices of the appropriate ancestor cell. The grid is partitioned using the local computational weight

$$\mathcal{W}(C) = 2^{\max(l_{\min}-l,0)}, \quad \text{where } l \text{ is the level of cell } C, \tag{9.40}$$

prior to this operation to ensure a quasi-uniform grid partition at the termination of the general algorithm. Figure 9.9 illustrates this final operation.

9.5 Illustrations for 3J6D in three spatial dimensions

In this section, we illustrate the above procedure for the same protein `3J6D` (131664 atoms), but in three spatial dimensions, *i.e.*, the actual representation of the protein. The minimum and maximum refinement levels are $l_{\text{min}} = 5$ and $l_{\text{max}} = 12$ respectively, the proportionality factor is $L = 1.2$. The grid was constructed with $m = 2$ using the construction by reinitialization with 2048 processors for a cubic computational domain of side length 1: the protein and all its atom radii are scaled to be contained in a cubic box of side length 0.5 centered in the domain, and so is the probe radius $r_{\text{p}} = 1.4\,\text{Å}$. The maximum level is such that the diagonal of the finest grid cells δ satisfies $r_{\text{p}}/\delta \simeq 6$.

Figure 9.10 illustrates Γ_{SAS} and a truncated version alongside a part of a slice in the computational grid as created by Algorithm 3 (and Algorithm 4). Figure 9.11 shows Γ_{SES} and illustrates the identification of cavities. Figure 9.12 shows Γ_{SES} alongside part of a slice in the final computational grid, after coarsening steps.

9.6 Performances of the construction by reinitialization

In this section, we analyze the accuracy of the construction by reinitialization and show the strong scalability of our implementation. For these purposes, we focus on 6 different molecules of various sizes:

- `1d65`, with 760 atoms;
- `2err`, with 1638 atoms;

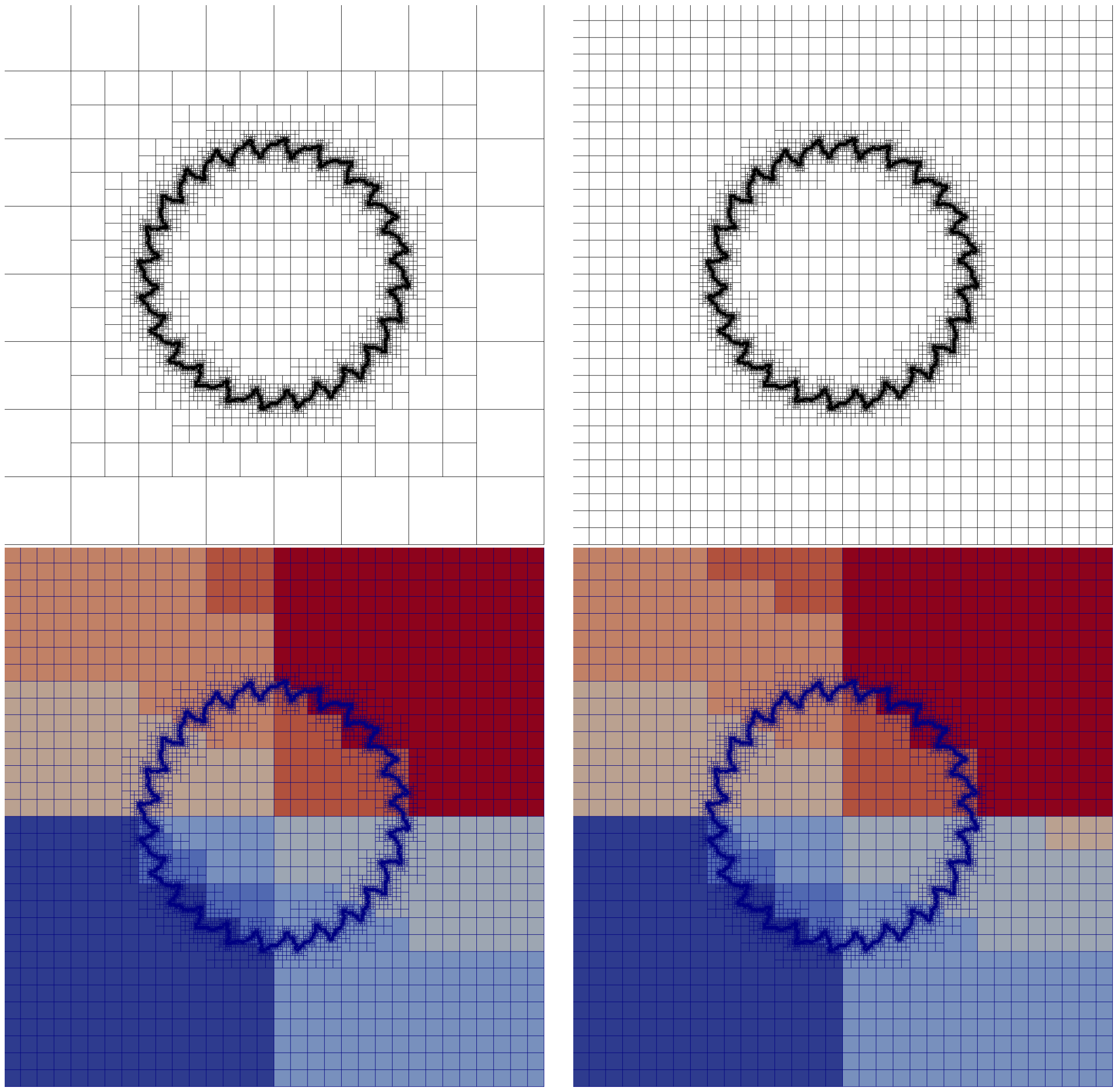


Figure 9.9: illustration of the final step of the grid construction, the desired minimum level of refinement (in this case, 5) is imposed everywhere. Note the very small difference in grid partitions between the bottom two figures, supporting the local computational weight (9.40). Top left: computational grid before imposition of the desired min level. Top right: computational grid after imposition of the desired min level. Bottom left: grid partitions as determined by computational weight (9.40). Each color represents one grid partition. Bottom right: exact uniform grid partitions (each cell being given an equal weight).

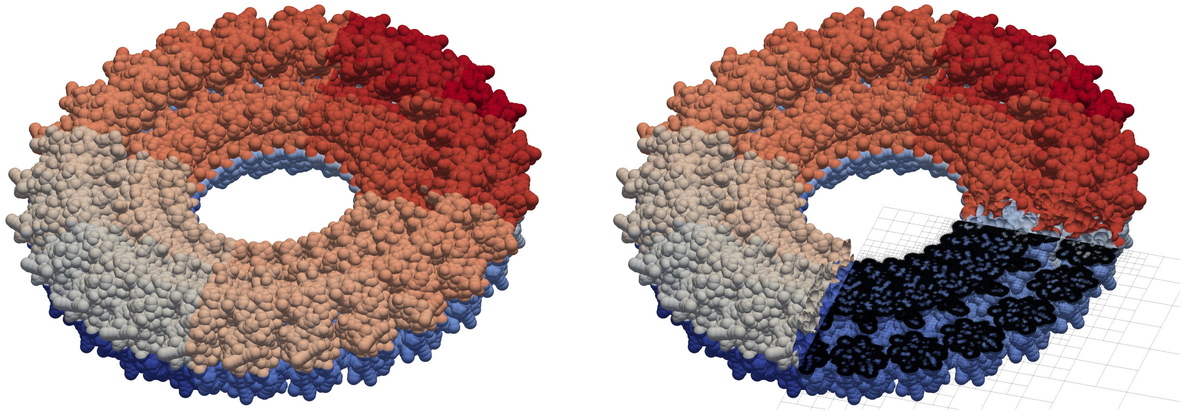


Figure 9.10: illustration of Γ_{SAS} for the protein 3J6D (131664 atoms) and the corresponding computational grid, as constructed by Algorithm 3 (and Algorithm 4). Left: representation of Γ_{SAS}. Right: truncated version with a slice in the computational grid.

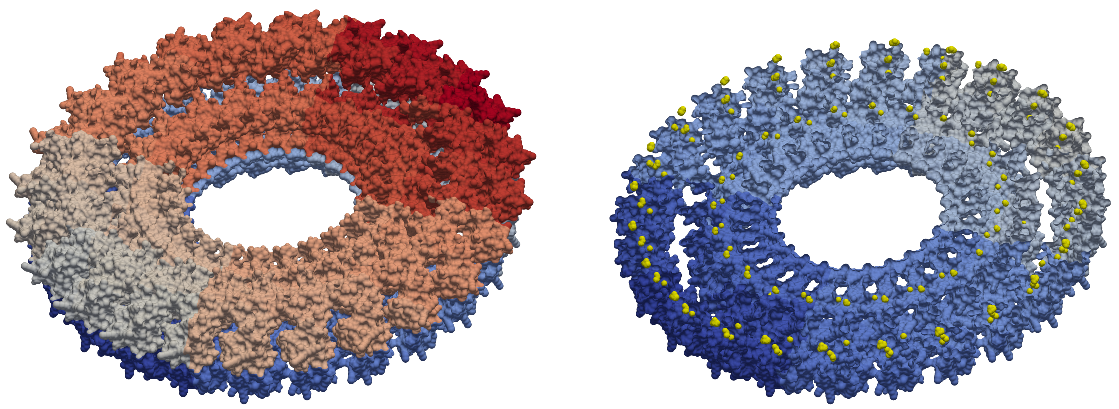


Figure 9.11: illustration of Γ_{SES} for the protein 3J6D (131664 atoms), as obtained by solving (9.23), and the cavities identified with Algorithm 8. Left: representation of Γ_{SES}. Right: clipped version of Γ_{SES} with a representation of the cavities, in yellow.

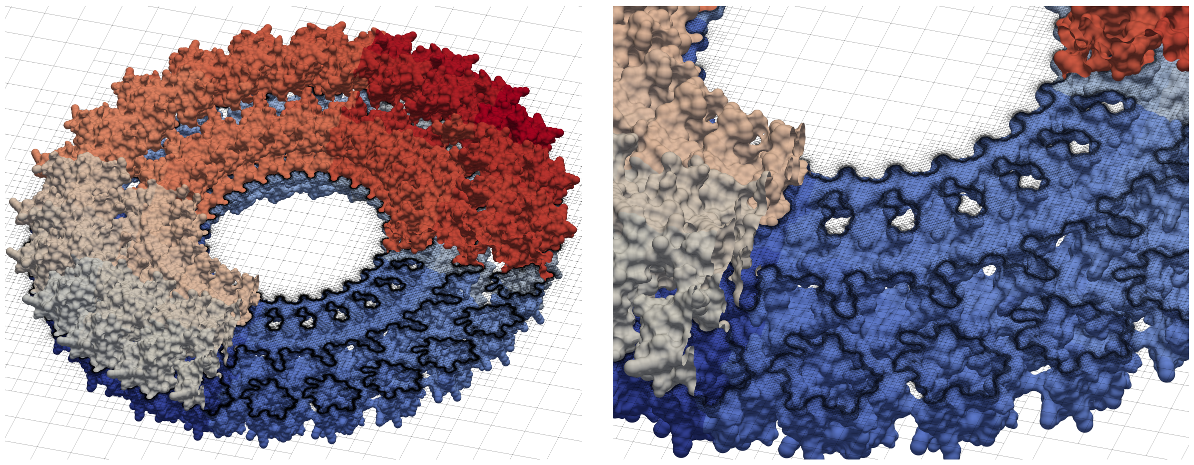


Figure 9.12: illustration of the final computational grid associated with Γ_{SES} for the protein `3J6D` (131664 atoms). The reader is invited to compare these figures to Figure 9.10. Left: truncated version of the left figure in Figure 9.11 with a slice in the final computational grid. Right: zoom-in close to part of Γ_{SES}, in the truncated region.

- `2aid`, with 3128 atoms;
- `1a2k`, with 13627 atoms;
- `3J6D`, with 131664 atoms;
- `1htq`, with 177240 atoms;

with a probe radius $r_{\text{p}} = 1.4\,\text{Å}$. Only the 4 first molecules are considered for the accuracy analysis as we intend to run the analyses up to levels of refinements such that $\delta < r_{\text{p}}/8$ at least. This would require $l_{\max} \geq 13$ for the last two molecules, which would lead to more than 1×10^9 grid nodes for the grid created by Algorithm 3 and Algorithm 4.

For every analysis, the molecule and all its atom radii are scaled to be contained in a cubic box of side length 0.5 centered in the domain of side length 1, and so is the probe radius $r_{\text{p}} = 1.4\,\text{Å}$. The Solvent-Excluded Surfaces of the 6 molecules here above, and slices of the final grids are illustrated in Figure 9.13.

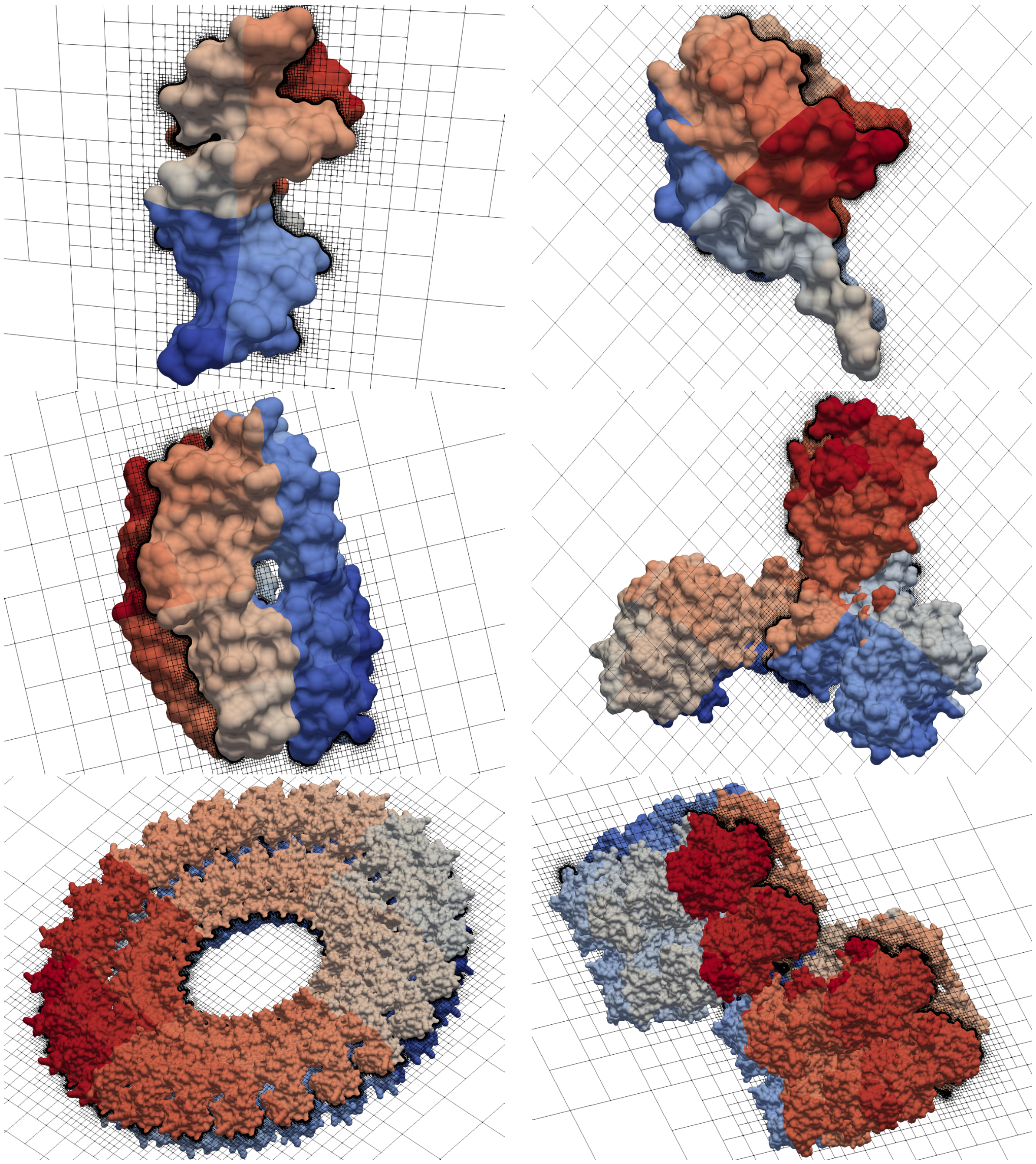


Figure 9.13: illustration of Γ_{SES} for the 6 molecules of interest for the performance analyses, and slices of the constructed computational grid. Top left: `1d65`. Top right: `2err`. Middle left: `2aid`. Middle right: `1a2k`. Bottom left: `3J6D`. Bottom right: `1htq`.

9.6.1 Accuracy

We show the accuracy of the construction by reinitialization by analyzing the interpolated values of the node-sampled level set function $\overline{\phi}$ at the actual definition of Γ_{SES}, found using the construction by direct calculation. Starting from the minimal value of $l_{\max}$ such that $\delta < r_{\text{p}}$, the analysis is performed for increasing $l_{\max} \leq 12$. For each level, we define two distinct error measures:

- the $1-$norm of the error, defined as

$$\int_{\Gamma_{\text{SES}}} \left|\overline{\phi}\left(\boldsymbol{x}\right)\right| \mathrm{d}\boldsymbol{x} \tag{9.41}$$

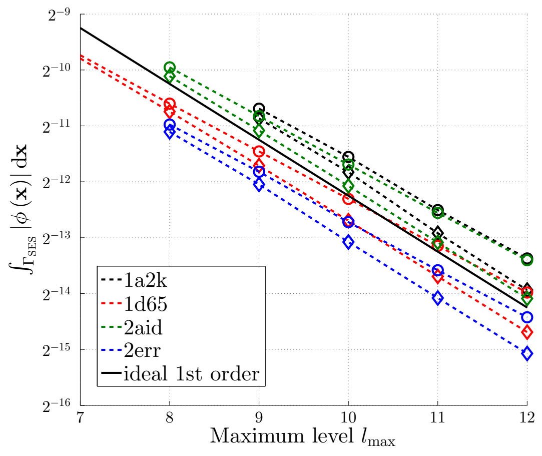
  where the integral is evaluated with the method from [116];

- the $\infty-$norm of the error, defined as

$$\max\left\{\left|\overline{\phi}\left(\boldsymbol{x}\right)\right| : \boldsymbol{x} \in \overline{\mathcal{E}} \bigcap \Gamma_{\text{SES}}\right\} \tag{9.42}$$

  where $\overline{\mathcal{E}}$ is the set of all grid edges.

The first error (9.41) averages the localization error over Γ_{SES}, while the second error (9.42) considers only the worst case scenario within all relevant localization errors. The results are illustrated in Figure 9.14, for the $1-$norm and the $\infty-$norm. As illustrated in Figure 9.14 (left), the construction by reinitialization is clearly first-order accurate in average and one obtains slightly more accurate results with $m = 2$. As it can be observed from Figure 9.14 (right), the $\infty-$norm converges too provided $l_{\max}$ is such that $\delta < r_{\text{p}}/4$. We relate this observation to the natural condition of resolving all kinks in Γ_{SAS}.



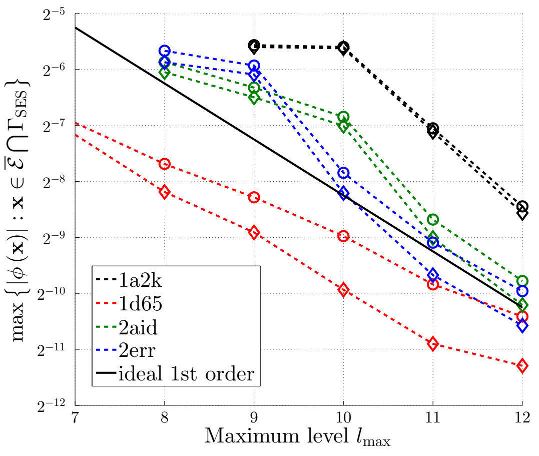


Figure 9.14: accuracy analyses for the four molecules `1d65`, `2err`, `2aid` and `1a2k`. Circles: $m = 1$; diamonds: $m = 2$. Left: error analysis in $1-$norm for the construction by reinitialization. Right: error analysis in $\infty-$norm for the construction by reinitialization.

9.6.2 Strong scalability

Our implementation of the construction by reinitialization benefits from strong scalability as illustrated in Figure 9.15. These scalability tests were performed for the construction of a given computational grid, with an increasing number of processors. Two different maximum level of refinement were chosen: $l_{\max}$ is set to the smallest integer such that either a) $\delta < r_{\rm p}$, or b) $\delta < r_{\rm p}/4$, where δ is the diagonal of the finest grid cells. The minimum level of refinement $l_{\min}$ is set to $l_{\max} - 5$. The tests were run on the Stampede 2 cluster at the Texas Advanced Computing Center (TACC). As illustrated in those graphs, the construction by reinitialization with $m = 2$ is more than two times slower than with $m = 1$.

9.7 Summary

We have presented a construction by reinitialization to build an implicit representation of the Solvent-Excluded Surface of a biomolecule using level-set strategies and to

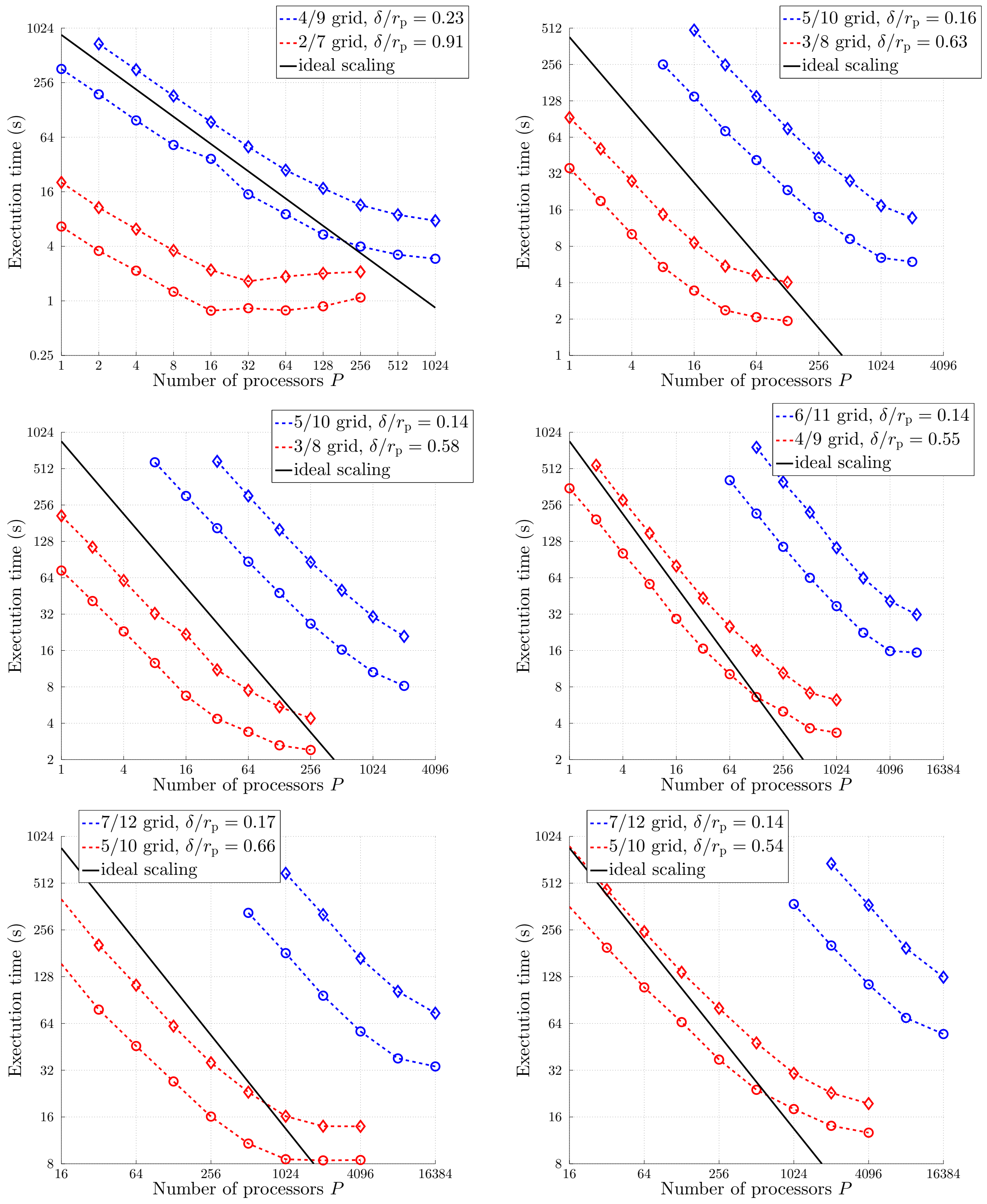


Figure 9.15: scaling analysis of the construction by reinitialization for the six molecules under consideration. Circles: $m = 1$; diamonds: $m = 2$. Top left: `1d65`. Top right: `2err`. Middle left: `2aid`. Middle right: `1a2k`. Bottom left: `3J6D`. Bottom right: `1htq`.

construct an adaptive Octree grid satisfying criterion (9.1), in the paradigm of distributed computing. Numerous illustrations have showed the capability of the construction by reinitialization to capture features of the Solvent-Excluded Surface regardless of their topological complexity. The method is showed to be first-order accurate and strongly scalable.

The implementation of such a robust method in a distributed computing framework offers new perspectives. For instance, representations of very large molecular structures can be handled and the corresponding adaptive Octree grids can be constructed, whereas the memory requirement would be intractable for a single-CPU architecture. Figure 9.16 illustrates such capabilities. For moderately large molecules ($\mathcal{O}\left(10^3 - 10^4\right)$ atoms) on the other hand, constructing the adaptive computational grid and the Solvent-Excluded Surface takes a few seconds at most (see Figure 9.15), allowing more efficient and easier analyses, and providing a practical tool for coupling electrostatic computation with dynamical molecular simulations.

As pointed out in our scaling analyses, the reinitialization of the level-set function is the most resource-intensive operation of the construction by reinitialization. We point out that the method can be further accelerated by implementing a parallel version of the Fast Sweeping Algorithm on adaptive Quad-/Oc-tree grids, which would make it optimally fast. Similarly, we believe that the method can be made second-order accurate when using $m = 2$, by taking special care of grid nodes located near kinks in the Solvent-Accessible Surface, at low additional cost. Finally, although using the construction by reinitialization in order to remesh the computational domain starting from the root cell would not be prohibitively expensive for moderately large molecules, in a dynamical simulation setting, we believe that the reduced lists as described in section 9.4.1 can be dynamically updated in an optimal fashion in order to save computational resources, when atoms' location change. These improvements would make for interesting future

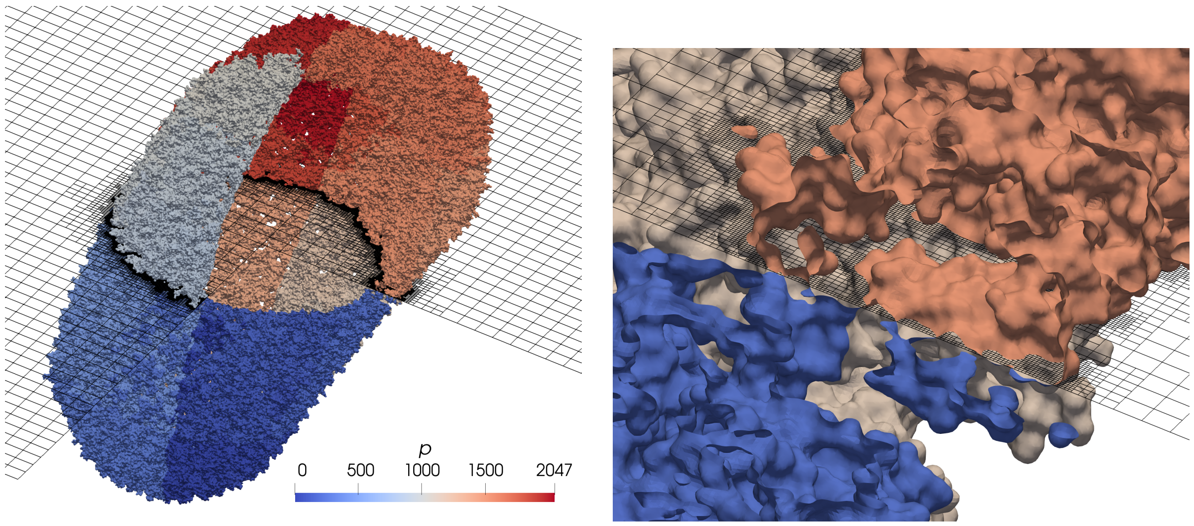


Figure 9.16: truncated view of Γ_{SES} of `3J3Q`, the HIV-1 capsid. The structure contains $4,884,312$ atoms, the probe radius is $1.4\,\text{Å}$. The computational grid has a minimum refinement level $l_{\min} = 6$ and a maximum refinement level $l_{\max} = 12$, the diagonal δ of the finest grid cells is $0.78 r_{\text{p}}$. The computational grid was built with $2,048$ processors in $194\,\text{s}$ and has $249,484,789$ grid nodes. Right: zoom-in close to the surface.

work.

Algorithm for calculating the construction by direct calculation

Algorithm 13 Direct calculation of $\mathrm{dist}\left(\boldsymbol{v},\Gamma_{\mathrm{SAS}}\right)$, based on the reduced list $\mathcal{L}\left(C\right)$ of a finest cell having $\boldsymbol{v}$ as a vertex. $\hat{\boldsymbol{\xi}}$ is an initial guess in Γ_{SAS} for $\boldsymbol{\xi}$ (in/out variable).

```
function GET_EXACT_DISTANCE_TO_SAS($\boldsymbol{v}$, $\mathcal{L}(C)$, $\hat{\boldsymbol{\xi}}$)
    $\boldsymbol{\xi} \leftarrow$ any point out of the domain;
    for $(ii = 0;\ ii < |\mathcal{L}(C)|;\ ii{+}{+})$ do
        sort the list of atom indices $n \in \mathcal{L}(C)$ by decreasing $\rho_{r_\mathrm{p},\,n}(\boldsymbol{v})$;
        $i \leftarrow ii^{\mathrm{th}}$ atom index in the sorted list;
        if $\left|\rho_{r_\mathrm{p},\,i}(\boldsymbol{v})\right| > \min\left(\|\hat{\boldsymbol{\xi}}-\boldsymbol{v}\|, \|\boldsymbol{\xi}-\boldsymbol{v}\|\right)$ then
            continue;
        end if
        $\boldsymbol{z} \leftarrow$ projection of $\boldsymbol{v}$ on $\partial B(\boldsymbol{c}_i, r_i + r_\mathrm{p})$, see (9.8);
        point_is_valid $\leftarrow \left(i = \mathrm{argmax}_{n\in\mathcal{L}(C)}\, \rho_{r_\mathrm{p},\,n}(\boldsymbol{z})\right)$;
        if $\neg$point_is_valid then
            for $(jj = 0;\ jj < |\mathcal{L}(C)|;\ jj{+}{+})$ do
                sort the list of atom indices $n \in \mathcal{L}(C)$ by decreasing $\rho_{r_\mathrm{p},\,n}(\boldsymbol{z})$;
                $j \leftarrow jj^{\mathrm{th}}$ atom index in the sorted list;
                if $\left|\rho_{r_\mathrm{p},\,j}(\boldsymbol{v})\right| > \min\left(\|\boldsymbol{\xi}-\boldsymbol{v}\|, \|\hat{\boldsymbol{\xi}}-\boldsymbol{v}\|\right)$ then
                    continue;
                end if
                calculate $\boldsymbol{o}$, $\boldsymbol{n}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, $\lambda$ and $\zeta$          ▷ see (9.28), (9.29) and (9.30).
                set_is_empty $\leftarrow$ false;
                initialize $\vartheta^- \leftarrow 0$, $\vartheta^+ \leftarrow 0$;
                while $\neg$point_is_valid $\wedge$ $\neg$set_is_empty do
                    set_is_empty $\leftarrow \left(\vartheta^+ - \vartheta^- = 2\pi\right)$;
                    $\vartheta \leftarrow \vartheta^+$ if $\left|\vartheta^+\right| < \left|\vartheta^-\right|$, $\vartheta \leftarrow \vartheta^-$ otherwise;
                    $\boldsymbol{z} \leftarrow \boldsymbol{s}(\vartheta)$;          ▷ see (9.31).
                    if $\|\boldsymbol{z}-\boldsymbol{v}\| > \min\left(\|\hat{\boldsymbol{\xi}}-\boldsymbol{v}\|, \|\boldsymbol{\xi}-\boldsymbol{v}\|\right)$ then
                        break;
                    end if
                    $k = \mathrm{argmax}_{n\in\mathcal{L}(C)}\, \rho_{r_\mathrm{p},\,n}(\boldsymbol{z})$
                    point_is_valid $\leftarrow \left(\rho_{r_\mathrm{p},\,k}(\boldsymbol{z}) = 0\right)$
                    if $\neg$point_is_valid then
                        calculate $\boldsymbol{c}'_k$ and $r'_k$;          ▷ see (9.33) and (9.34)
                        if $r'_k > \left\|\boldsymbol{c}'_k - \boldsymbol{o}\right\| + \zeta$ then
                            $\vartheta^+ \leftarrow \pi$, $\vartheta^- \leftarrow -\pi$, set_is_empty $\leftarrow$ true
                        else
                            calculate $\alpha$ and $\beta$;          ▷ see (9.35) and (9.36)
                            $\vartheta^+ \leftarrow \max\left(\vartheta^+, \min\left(\beta+\alpha, 2\pi-\vartheta^-\right)\right)$;
                            $\vartheta^- \leftarrow \min\left(\vartheta^-, \max\left(\beta-\alpha, -\left(2\pi-\vartheta^+\right)\right)\right)$;
                        end if
                    end if
                end while
                if point_is_valid $\wedge$ $\|\boldsymbol{z}-\boldsymbol{v}\| < \|\boldsymbol{\xi}-\boldsymbol{v}\|$ then
                    $\boldsymbol{\xi} \leftarrow \boldsymbol{z}$;
                end if
                point_is_valid $\leftarrow$ false;
                $\boldsymbol{z} \leftarrow$ projection of $\boldsymbol{v}$ on $\partial B(\boldsymbol{c}_i, r_i + r_\mathrm{p})$, see (9.8);
            end for
        else if $\|\boldsymbol{z}-\boldsymbol{v}\| < \|\boldsymbol{\xi}-\boldsymbol{v}\|$ then
            $\boldsymbol{\xi} \leftarrow \boldsymbol{z}$;
        end if
    end for
    if $\|\boldsymbol{\xi}-\boldsymbol{v}\| < \left\|\hat{\boldsymbol{\xi}}-\boldsymbol{v}\right\|$ then
        $\hat{\boldsymbol{\xi}} \leftarrow \boldsymbol{\xi}$;
    end if
    return $\min\left(\|\hat{\boldsymbol{\xi}}-\boldsymbol{v}\|, \|\boldsymbol{\xi}-\boldsymbol{v}\|\right)$;
end function
```

Appendix A

Detailed derivation of the governing equations for incompressible viscous two-phase flows

In this appendix, the equations governing the two-phase flow of immiscible, linear, isotropic fluids are derived. By linearity and isotropy of the fluids, it is meant that

- the fluids are Newtonian: the components of the deviatoric stress tensor are isotropic linear functions of the components of the velocity gradient,
- heat conduction is accurately dictated by Fourier's law,

for small departures from thermodynamic equilibrium, namely small rate of strain and small temperature gradients. As discussed by Batchelor ([284], pages 141-156), the Newtonian fluid assumption holds surprisingly well over a large range of strain rates for fluids that have isotropic structures, like most simple fluids. Mathematically, linearity and isotropy translate into the two following constitutive laws[1]

$$\underline{\boldsymbol{\sigma}} = -p\underline{\boldsymbol{\delta}} + \left(\lambda - \frac{2\mu}{3}\right)(\nabla \cdot \boldsymbol{u})\,\underline{\boldsymbol{\delta}} + 2\mu\underline{\boldsymbol{E}}, \tag{A.1}$$

(see [284] and [287]) and

$$\boldsymbol{q} = -k\nabla T, \tag{A.2}$$

where $\underline{\boldsymbol{\sigma}}$ is the local stress tensor, $\boldsymbol{q}$ is the local heat flux, $\boldsymbol{u}$ is the local fluid velocity, p is the local thermodynamic pressure, $\underline{\boldsymbol{E}} = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{\mathrm{T}}\right)$ is the local strain-rate tensor, T is the local temperature, λ is the expansion viscosity (also called second coefficient of viscosity), μ is the shear viscosity, k is the thermal conductivity and $\underline{\boldsymbol{\delta}}$ is the identity second-order tensor.

Furthermore, the present work focuses on incompressible flows, i.e., flows such that the mass density of a material element may be assumed constant (unless the material element undergoes a phase change). As shown hereunder, this flow property, along with the conservation of mass, has the mathematical consequence of constraining the velocity

[1]There is a certain lack of consensus in the literature about the exact definition of the expansion viscosity (also known as "bulk" or "second" viscosity). In this document, we follow the lines by Batchelor [284], supported by the fact that shear stresses do not contribute to a net isotropic stress (in other words, shear stresses do not contribute to *mechanical* pressure). However, in the literature, it is not uncommon to find the stress constitutive law for isotropic linear fluids written as

$$\underline{\boldsymbol{\sigma}} = -p\underline{\boldsymbol{\delta}} + \zeta\,(\nabla \cdot \boldsymbol{u})\,\underline{\boldsymbol{\delta}} + 2\mu\underline{\boldsymbol{E}}$$

where ζ is dubbed "bulk viscosity" (or "second coefficient of viscosity", or "dilatational viscosity", ...) in such a context. This differs from our formulation in (A.1), that follows Batchelor's lines [284], by $\zeta = \lambda - \frac{2\mu}{3}$.

This remark might lead to serious practical implications when matching experimentally measured coefficients for instance. Similarly, Stokes suggested in 1880 [285] that the departure of the *mechanical* pressure $\frac{-\sigma_{ii}}{3}$ from the *thermodynamic* pressure p is negligible, leading to $\lambda = 0$ in our formulation (A.1), but $\zeta = -\frac{2\mu}{3}$ in the above. This is known as *Stokes' hypothesis*, which lacks formal physical and experimental support but is used quite often in the literature. The expansion viscosity λ can be shown to be exactly 0 (or $\zeta = -2\mu/3$) for ideal gas; however there is no correlation between λ and μ in liquids in general, and the value of λ can be of the same order of magnitude as μ or even much larger [286]. For water at approximately 20 °C for instance, $\mu \simeq 1 \times 10^{-3}\,\mathrm{kg\,m^{-1}\,s^{-2}}$ while $\lambda \simeq 3.09 \times 10^{-3}\,\mathrm{kg\,m^{-1}\,s^{-2}}$

field to be solenoidal, i.e., $\nabla \cdot \boldsymbol{u} = 0$ (away from the interface). Therefore,

$$\underline{\boldsymbol{\sigma}} = -p\underline{\boldsymbol{\delta}} + 2\mu\underline{\boldsymbol{E}} \tag{A.3}$$

will be used in place of (A.1) when incorporating the constitutive stress-strain relation into the final forms of the governing equations.

In subsection A.1, the partial differential equations describing the flow physics far from the interface are determined from first principles. As expected, the Navier-Stokes equations are recovered as a result, along with the familiar balance equation for internal energy. In subsection A.2, the first principles are then applied to a control volume encapsulating the two fluids under consideration in order to determine, without ambiguity, the mathematical conditions coupling the flow of the two distinct fluids, across the sharp interface that separates them. Since no field of interest may be assumed continuous *a priori* (unless rigorously proven so and/or physically justified), the mathematical toolkit needs to be extended in order to handle such discontinuities when applying macroscopic first principles.

The set of partial differential equations derived in subsection A.1 augmented with the interface conditions determined thereafter in subsection A.2 provides so a complete mathematical translation of the fundamental conservation principles, which is fully consistent with continuum mechanics in the limit of infinitesimally thin and massless interfaces.

A.1 Balance equations in each separate phase

In order to derive the fundamental conservation equations, we consider an arbitrary *material* volume $\mathbb{V}(t)$, i.e., a volume whose boundary is moving *with the fluid velocity* $\boldsymbol{u}(\boldsymbol{x}, t)$ so that it encapsulates the same material element *at all time.* The surface of that volume is denoted by $\mathbb{S}(t)$, its outer unit normal vector is $\boldsymbol{\eta}$. For a given (scalar, vector or tensor) field $q(\boldsymbol{x},t)$, the association of Leibniz Integral Rule and the divergence theorem (see app. A.3) yields

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{V}(t)} q(\boldsymbol{x},t)\,\mathrm{d}\mathbb{V} &= \int_{\mathbb{V}(t)} \frac{\partial q(\boldsymbol{x},t)}{\partial t}\,\mathrm{d}\mathbb{V} + \int_{\mathbb{S}(t)} q(\boldsymbol{x},t)\,\boldsymbol{u}(\boldsymbol{x},t)\cdot\boldsymbol{\eta}\,\mathrm{d}\mathbb{S} \\ &= \int_{\mathbb{V}(t)} \left(\frac{\partial q(\boldsymbol{x},t)}{\partial t} + \nabla\cdot\left(q(\boldsymbol{x},t)\,\boldsymbol{u}(\boldsymbol{x},t)\right)\right)\mathrm{d}\mathbb{V}. \end{aligned} \tag{A.4}$$

Here after, the dependence of the fields of interest on $\boldsymbol{x}$ and t will be omitted for the sake of clarity in the exposition. In this subsection, we assume that $\mathbb{V}(t)$ contains only one fluid phase and is free of phase transition around time t. As a consequence, every thermodynamic variable may be assumed to be *continuous* and *twice differentiable* function of any two other independent thermodynamic variables over $\mathbb{V}(t)$ and/or $\mathbb{S}(t)$.

A.1.1 Balance of mass

The conservation of mass for the material volume $\mathbb{V}(t)$ reads

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{V}(t)} \rho\,\mathrm{d}\mathbb{V} = \int_{\mathbb{V}(t)} \left(\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u})\right)\mathrm{d}\mathbb{V} = 0, \tag{A.5}$$

using (A.4), and holds for *any* material volume $\mathbb{V}(t)$. Therefore, we have

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}) = 0. \tag{A.6}$$

For incompressible flows

For incompressible flows, the mass density of material elements is assumed to remain constant so long as it does not undergo a phase change. This common flow property translates mathematically into $\frac{\mathrm{D}\rho}{\mathrm{D}t} = \frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho = 0$ and is known to be a suitable assumption, even for gas flows, so long as the local Mach number remains sufficiently low everywhere.

A direct consequence on the above conservation of mass (A.6) follows: the velocity fields must be solenoidal in both phases, i.e.

$$\boxed{\nabla \cdot \boldsymbol{u} = 0,} \tag{A.7}$$

which mathematically prevents material volumes to expand or contract (so long as they do not go across the interface).

A.1.2 Balance of momentum

The conservation of momentum for the material volume $\mathbb{V}(t)$, i.e., Newton's second law reads

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\mathbb{V}(t)} \rho \boldsymbol{u} \, \mathrm{d}\mathbb{V} = \int_{\mathbb{V}(t)} \rho \boldsymbol{f} \, \mathrm{d}\mathbb{V} + \int_{\mathbb{S}(t)} \underline{\boldsymbol{\sigma}} \cdot \boldsymbol{\eta} \, \mathrm{d}\mathbb{S} \tag{A.8}$$

where $\boldsymbol{f}$ is the body force per unit mass, $\underline{\boldsymbol{\sigma}}$ is the stress tensor. Using (A.4) for the LHS and the divergence theorem for the second term of the RHS, this equation becomes

$$\int_{\mathbb{V}(t)} \left(\frac{\partial}{\partial t} (\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u} \boldsymbol{u}) - \rho \boldsymbol{f} - \nabla \cdot \underline{\boldsymbol{\sigma}} \right) \mathrm{d}\mathbb{V} = \boldsymbol{0} \tag{A.9}$$

which holds for *any* material volume $\mathbb{V}(t)$. Therefore,

$$\frac{\partial}{\partial t} (\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u} \boldsymbol{u}) = \rho \boldsymbol{f} + \nabla \cdot \underline{\boldsymbol{\sigma}}. \tag{A.10}$$

For incompressible flows of linear, isotropic fluids

Taking into account (A.6) and using (A.3), the balance of momentum reads

$$\boxed{\rho\left(\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)=\rho\boldsymbol{f}-\nabla p+\mu\nabla^2\boldsymbol{u}.} \tag{A.11}$$

for incompressible flows of linear, isotropic fluids (μ is assumed constant away from the interface).

A.1.3 Balance of internal energy

The conservation of energy for the material volume $\mathbb{V}(t)$, i.e., the first law of thermodynamics reads

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{V}(t)}\rho\mathcal{E}\,\mathrm{d}\mathbb{V}=\int_{\mathbb{V}(t)}\rho\boldsymbol{f}\cdot\boldsymbol{u}\,\mathrm{d}\mathbb{V}+\int_{\mathbb{S}(t)}\boldsymbol{u}\cdot\left(\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{\eta}\right)\,\mathrm{d}\mathbb{S}-\int_{\mathbb{S}(t)}\boldsymbol{q}\cdot\boldsymbol{\eta}\,\mathrm{d}\mathbb{S}+\int_{\mathbb{V}(t)}\dot{\Theta}\,\mathrm{d}\mathbb{V} \tag{A.12}$$

where $\mathcal{E}$ is the total energy per unit mass, that is $\mathcal{E}=e+\dfrac{\|\boldsymbol{u}\|^2}{2}$ where e is the internal energy per unit mass, $\boldsymbol{q}$ is the heat flux and $\dot{\Theta}$ is a volumetric heat source. Using (A.4) for the LHS and the divergence theorem for the two last terms of the RHS, we have

$$\int_{\mathbb{V}(t)}\left(\frac{\partial}{\partial t}(\rho\mathcal{E})+\nabla\cdot(\rho\mathcal{E}\boldsymbol{u})-\rho\boldsymbol{f}\cdot\boldsymbol{u}-\nabla\cdot\left(\boldsymbol{u}\cdot\underline{\boldsymbol{\sigma}}\right)+\nabla\cdot\boldsymbol{q}-\dot{\Theta}\right)\mathrm{d}\mathbb{V}=0 \tag{A.13}$$

which holds for *any* material volume $\mathbb{V}(t)$. Therefore,

$$\frac{\partial}{\partial t}(\rho\mathcal{E})+\nabla\cdot(\rho\mathcal{E}\boldsymbol{u})=\rho\boldsymbol{f}\cdot\boldsymbol{u}+\nabla\cdot\left(\boldsymbol{u}\cdot\underline{\boldsymbol{\sigma}}\right)-\nabla\cdot\boldsymbol{q}+\dot{\Theta}. \tag{A.14}$$

On the other hand, by taking the dot product side-by-side of (A.11) with $\boldsymbol{u}$ and adding (see (A.7))

$$\frac{\|\boldsymbol{u}\|^2}{2}\left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u})\right) = 0$$

side-by-side, one obtains the balance of kinetic energy

$$\frac{\partial}{\partial t}\left(\rho \frac{\|\boldsymbol{u}\|^2}{2}\right) + \nabla \cdot \left(\rho \boldsymbol{u} \frac{\|\boldsymbol{u}\|^2}{2}\right) = \rho \boldsymbol{f} \cdot \boldsymbol{u} + \boldsymbol{u} \cdot (\nabla \cdot \underline{\boldsymbol{\sigma}}) \, . \tag{A.15}$$

The balance of internal energy results from (A.15) and (A.14). Indeed, subtracting the former from the latter side-by-side, one obtains

$$\frac{\partial}{\partial t}(\rho e) + \nabla \cdot (\rho e \boldsymbol{u}) = \underline{\boldsymbol{\sigma}} : \underline{\boldsymbol{E}} - \nabla \cdot \boldsymbol{q} + \dot{\Theta}, \tag{A.16}$$

wherein $\underline{\boldsymbol{E}} = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathrm{T}}\right)$ is the strain rate tensor [2].

For incompressible flows of linear, isotropic fluids

Taking into account (A.6), using (A.3) and (A.2), the balance of internal energy reads

$$\boxed{\rho \left(\frac{\partial e}{\partial t} + \boldsymbol{u} \cdot \nabla e\right) = 2\mu \underline{\boldsymbol{E}} : \underline{\boldsymbol{E}} + k \nabla^2 T + \dot{\Theta}.} \tag{A.17}$$

for incompressible flows of linear, isotropic fluids (k is assumed constant away from the interface).

[2]One may show that $\underline{\boldsymbol{\sigma}}$ is symmetric from the conservation of angular momentum.

A.2 Balance equations across the interface

The mathematical description of conservation laws across the interface requires special care, not only because of discontinuities in material properties but also because interface phenomena may take place. In the context of this work, we take into account surface tension effects as well as the possible existence of an interface-defined, i.e., singular, force $\boldsymbol{G}$ that may account for other or additional interface physics (e.g., hyper-elasticity of a membrane, interface-distributed stress due to coupled electromagnetic effects, etc.). Although conceptually equivalent to take into consideration from a mathematical viewpoint, an interface-defined heat source will not be considered in the following, owing to its limited practical interest[3].

Let $\Sigma(t)$ be a portion of the interface, we denote by $\boldsymbol{\tau}$ a unit vector tangent to $\partial\Sigma(t)$. Let $\boldsymbol{n}$ be the unit vector normal to $\Sigma(t)$ with positive orientation: at every point on $\partial\Sigma(t)$, $\boldsymbol{m} = \boldsymbol{\tau} \times \boldsymbol{n}$ is a unit vector, outward normal to $\partial\Sigma(t)$, and tangent to $\Sigma(t)$; this is illustrated in the left sketch of Figure A.1. The local velocity of the interface[4] is denoted by $\boldsymbol{w}$.

We define the extruded volumes $\Omega_{\Sigma(t),\,\epsilon^-}$ and $\Omega_{\Sigma(t),\,\epsilon^+}$ on either side of the interface by

$$\Omega_{\Sigma(t),\,\epsilon^\pm} = \{\boldsymbol{z} \,|\, \boldsymbol{z} = \boldsymbol{s} \pm \ell\boldsymbol{n},\ \boldsymbol{s} \in \Sigma(t)\,,\ 0 \leq \ell \leq \epsilon\}\,. \tag{A.18}$$

Note that $\Omega_{\Sigma(t),\,\epsilon^-}$ and $\Omega_{\Sigma(t),\,\epsilon^+}$ are both included in distinct fluid phases for ϵ small enough. Interface conditions coupling the dynamics of the two fluids across the interface result from the application of conservation principles to the control volume $\Omega_{\Sigma(t),\,\epsilon} = \Omega_{\Sigma(t),\,\epsilon^-} \bigcup \Omega_{\Sigma(t),\,\epsilon^+}$, in the limit of $\epsilon \to 0$. We denote by $\boldsymbol{\eta}$ the unit vector, normal to

[3]If an electrical current flows across the interface, the Peltier effect could be an example of such an interface-defined heat source, as pointed out in [288].

[4]While the interface kinematics is entirely determined by the normal component of the interface velocity $\boldsymbol{w} \cdot \boldsymbol{n}$ only, taking into consideration a tangential motion of interface points may be required to account for local stretching of the interface.

$\partial\Omega_{\Sigma(t),\,\epsilon}$ pointing outward $\Omega_{\Sigma(t),\,\epsilon}$, as illustrated in the right sketch of Figure A.1, in two dimensions.

When considering a field q that is continuous and differentiable in either phase but may be discontinuous across the interface, Leibniz Integral Rule applies only on either side of the interface strictly speaking[5], i.e., only in $\Omega_{\Sigma(t),\,\epsilon^-}$ and $\Omega_{\Sigma(t),\,\epsilon^+}$ but not in their

[5]Unless one resorts to distribution theory and exploits Dirac delta functions, which is suitable theoretically but known to be inappropriate when translated as such in a numerically framework.

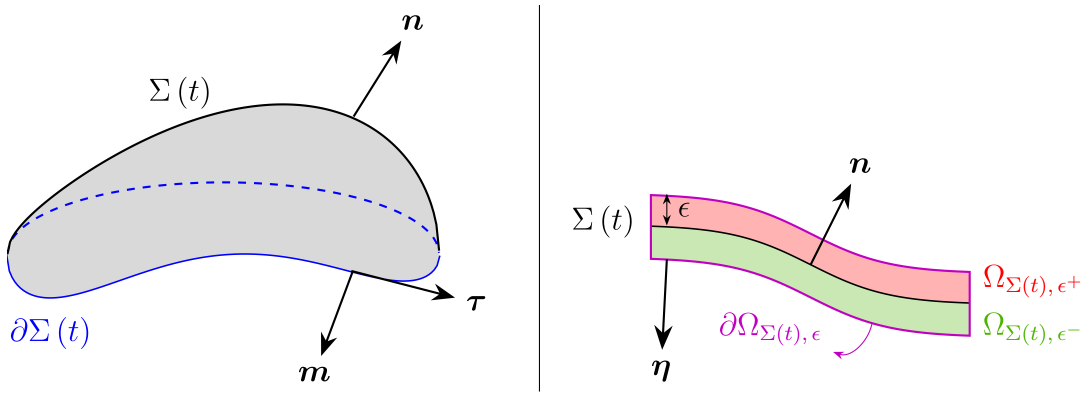


Figure A.1: Left: portion of an arbitrary three-dimensional interface $\Sigma\,(t)$ (shaded region); $\boldsymbol{\tau}$ is the unit tangent vector to $\partial\Sigma\,(t)$, $\boldsymbol{n}$ is the unit normal vector to $\Sigma\,(t)$ with positive orientation: at every point on $\partial\Sigma\,(t)$, $\boldsymbol{m} = \boldsymbol{\tau} \times \boldsymbol{n}$ is a unit vector, outward normal to $\partial\Sigma\,(t)$, tangent to $\Sigma\,(t)$. Right: illustration of the control volume under consideration for the derivation of balance relations across the interface, in two dimensions.

union. Therefore, one has (along the lines of [288], chapter 2)

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{\Sigma(t),\,\epsilon}} q\,\mathrm{d}\Omega &= \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{\Sigma(t),\,\epsilon^+}} q\,\mathrm{d}\Omega + \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{\Sigma(t),\,\epsilon^-}} q\,\mathrm{d}\Omega \\ &= \int_{\Omega_{\Sigma(t),\,\epsilon^+}} \frac{\partial q}{\partial t}\,\mathrm{d}\Omega + \int_{\Omega_{\Sigma(t),\,\epsilon^-}} \frac{\partial q}{\partial t}\,\mathrm{d}\Omega \\ &\quad + \int_{\partial\Omega_{\Sigma(t),\,\epsilon^+}\setminus\Sigma(t)} q\boldsymbol{w}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon^-}\setminus\Sigma(t)} q\boldsymbol{w}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega \\ &\quad + \int_{\Sigma(t)} q^+\boldsymbol{w}\cdot(-\boldsymbol{n})\,\mathrm{d}\Sigma + \int_{\Sigma(t)} q^-\boldsymbol{w}\cdot\boldsymbol{n}\,\mathrm{d}\Sigma \\ &= \int_{\Omega_{\Sigma(t),\,\epsilon}} \frac{\partial q}{\partial t}\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} q\boldsymbol{w}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega - \int_{\Sigma(t)} [q]\,\boldsymbol{w}\cdot\boldsymbol{n}\,\mathrm{d}\Sigma \end{aligned} \quad (A.19)$$

where the interface quantities $q^\pm(\boldsymbol{s})$ are defined as

$$q^\pm(\boldsymbol{s}) = \lim_{\ell>0,\ell\to 0} q(\boldsymbol{s}\pm\ell\boldsymbol{n}), \quad (A.20)$$

for $\boldsymbol{s}\in\Sigma(t)$ and the jump in q across the interface is defined as $[q] = q^+ - q^-$.

Furthermore, though the mass of the interface is negligible, one may need to take into consideration thermodynamic material fields per unit area on $\Sigma(t)$ (see [197, 198]). Let ζ be such a quantity, the corresponding form of (A.4) for such interface-defined fields is the *surface transport theorem* which states

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Sigma(t)} \zeta\,\mathrm{d}\Sigma &= \int_{\Sigma(t)} \frac{\partial \zeta}{\partial t} + \boldsymbol{w}\cdot\nabla\zeta + \zeta\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right):\nabla\boldsymbol{w} \\ &= \int_{\Sigma(t)} \frac{\mathrm{D}\zeta}{\mathrm{D}t} + \zeta\nabla_{\mathrm{s}}\cdot\boldsymbol{w}\,\mathrm{d}\Sigma, \end{aligned} \quad (A.21)$$

wherein $\nabla_{\mathrm{s}}\cdot\boldsymbol{w} = (\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}):\nabla\boldsymbol{w}$ is the surface divergence of the interface velocity. A formal proof of (A.21) requires the use of advanced differential geometry ([198] refers the interested reader to [289]).

Finally, while $\Omega_{\Sigma(t),\,\epsilon}$ is a time-dependent control volume, it is not a *material* volume. Therefore, net transfers of mass, momentum and total energy advected through $\partial\Omega_{\Sigma(t),\,\epsilon}$ must be taken into account when applying the conservation of mass, Newton's second law and the first law of thermodynamics, respectively. If q is a *material* quantity (i.e., a quantity associated, and therefore advected, with material elements), the rate of variation of $\int_{\Omega_{\Sigma(t),\,\epsilon}} q \,\mathrm{d}\Omega$ due to advection through $\partial\Omega_{\Sigma(t),\,\epsilon}$ is

$$\int_{\partial\Omega_{\Sigma(t),\,\epsilon}} q\,(\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega. \tag{A.22}$$

Remark. *In the derivation of* (A.19) *and* (A.22)*, the velocity of every point on* $\partial\Omega_{\Sigma(t),\,\epsilon}$ *was considered to be* $\boldsymbol{w}$*. While it should be* $\boldsymbol{w}+\ell\dot{\boldsymbol{n}}$*, with* $0\le|\ell|\le\epsilon$*, this latter expression differs from* $\boldsymbol{w}$ *by a term proportional to* ϵ*, at most. Since one intends to consider the limit* $\epsilon\to 0$ *eventually, it is appropriate to consider that the velocity of every point on* $\partial\Omega_{\Sigma(t),\,\epsilon}$ *is* $\boldsymbol{w}$*, without loss of generality in the conclusions.*

A.2.1 Balance of mass across the interface

The conservation of mass applied to the control volume $\Omega_{\Sigma(t),\,\epsilon}$ reads (the interface itself is massless)

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{\Sigma(t),\,\epsilon}} \rho\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \rho\,(\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega = 0. \tag{A.23}$$

Using (A.19), we get

$$\int_{\Omega_{\Sigma(t),\,\epsilon}} \frac{\partial\rho}{\partial t}\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \rho\boldsymbol{u}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega - \int_{\Sigma(t)} [\rho]\,\boldsymbol{w}\cdot\boldsymbol{n}\,\mathrm{d}\Sigma = 0. \tag{A.24}$$

When $\epsilon \to 0$, the first term in (A.24) becomes negligible as it scales as ϵ while the other terms do not. Moreover,

$$\lim_{\epsilon \to 0} \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \rho \boldsymbol{u} \cdot \boldsymbol{\eta} \, \mathrm{d}\partial\Omega = \int_{\Sigma(t)} \left[\rho \boldsymbol{u} \cdot \boldsymbol{n}\right] \, \mathrm{d}\Sigma.$$

Therefore, in the limit $\epsilon \to 0$, (A.24) dictates

$$\int_{\Sigma(t)} \left[\rho \left(\boldsymbol{u} - \boldsymbol{w}\right) \cdot \boldsymbol{n}\right] \, \mathrm{d}\Sigma = 0 \tag{A.25}$$

which must hold for any portion of interface $\Sigma(t)$ so that

$$\boxed{\left[\rho \left(\boldsymbol{u} - \boldsymbol{w}\right) \cdot \boldsymbol{n}\right] = 0.} \tag{A.26}$$

This last equation translates the continuity of the local mass flux $\dot{M} = \rho \left(\boldsymbol{u} - \boldsymbol{w}\right) \cdot \boldsymbol{n}$ through the interface.

A.2.2 Balance of momentum across the interface

The application of Newton's second law to the control volume $\Omega_{\Sigma(t),\,\epsilon}$ gives

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega_{\Sigma(t),\,\epsilon}} \rho \boldsymbol{u} \, \mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \rho \boldsymbol{u} \left(\boldsymbol{u} - \boldsymbol{w}\right) \cdot \boldsymbol{\eta} \, \mathrm{d}\partial\Omega &= \int_{\Omega_{\Sigma(t),\,\epsilon}} \rho \boldsymbol{f} \, \mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \underline{\boldsymbol{\sigma}} \cdot \boldsymbol{\eta} \, \mathrm{d}\partial\Omega \\ &+ \int_{\partial\Sigma(t)} \gamma \boldsymbol{m} \, \mathrm{d}\partial\Sigma + \int_{\Sigma(t)} \boldsymbol{G} \, \mathrm{d}\Sigma \end{aligned} \tag{A.27}$$

where γ is the surface tension coefficient. Since the interface itself is massless, its momentum is discarded from (A.27). Using (A.19), one has

$$\begin{aligned}\int_{\Omega_{\Sigma(t),\,\epsilon}} \frac{\partial}{\partial t}(\rho \boldsymbol{u})\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \rho \boldsymbol{u}\boldsymbol{u}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega - \int_{\Sigma(t)} [\rho\boldsymbol{u}]\,\boldsymbol{w}\cdot\boldsymbol{n}\,\mathrm{d}\Sigma \\ = \int_{\Omega_{\Sigma(t),\,\epsilon}} \rho\boldsymbol{f}\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \underline{\boldsymbol{\sigma}}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega + \int_{\partial\Sigma(t)} \gamma\boldsymbol{m}\,\mathrm{d}\partial\Sigma + \int_{\Sigma(t)} \boldsymbol{G}\,\mathrm{d}\Sigma.\end{aligned} \tag{A.28}$$

In (A.28), the integrals over $\Omega_{\Sigma(t),\,\epsilon}$ scale as ϵ while the other terms do not; therefore those terms becomes negligible when $\epsilon \to 0$. Moreover, note that

$$\lim_{\epsilon\to 0} \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \underline{\boldsymbol{R}}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega = \int_{\Sigma(t)} [\underline{\boldsymbol{R}}\cdot\boldsymbol{n}]\,\mathrm{d}\Sigma$$

for any second-order tensor $\underline{\boldsymbol{R}}$. Therefore, in the limit $\epsilon \to 0$, (A.28) dictates

$$\int_{\Sigma(t)} [\rho\boldsymbol{u}\,(\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n}]\,\mathrm{d}\Sigma = \int_{\Sigma(t)} [\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}]\,\mathrm{d}\Sigma + \int_{\partial\Sigma(t)} \gamma\boldsymbol{m}\,\mathrm{d}\partial\Sigma + \int_{\Sigma(t)} \boldsymbol{G}\,\mathrm{d}\Sigma. \tag{A.29}$$

As shown in appendix A.5.1, we also have

$$\int_{\partial\Sigma(t)} \gamma\boldsymbol{m}\,\mathrm{d}\partial\Sigma = \int_{\Sigma(t)} \left((\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n})\cdot\nabla\gamma - \kappa\gamma\boldsymbol{n}\right)\mathrm{d}\Sigma \tag{A.30}$$

where $\kappa = \nabla\cdot\boldsymbol{n}$ is the curvature of the interface. Therefore, (A.29) is equivalent to

$$\int_{\Sigma(t)} \left[\dot{M}\boldsymbol{u} - \underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right]\mathrm{d}\Sigma = \int_{\Sigma(t)} \left((\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n})\cdot\nabla\gamma - \kappa\gamma\boldsymbol{n} + \boldsymbol{G}\right)\mathrm{d}\Sigma \tag{A.31}$$

which must hold for any portion of interface $\Sigma(t)$, so that

$$\left[\dot{M}\boldsymbol{u} - \underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right] = (\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n})\cdot\nabla\gamma - \kappa\gamma\boldsymbol{n} + \boldsymbol{G}. \tag{A.32}$$

This last relation expresses the balance of momentum across the interface in an inertial reference frame. It is usually preferred (and more natural) to express it in a reference frame moving with the interface. By subtracting $\left[\dot{M}\boldsymbol{w}\right] = \mathbf{0}$ side-by-side from (A.32), one has

$$\left[\dot{M}\left(\boldsymbol{u}-\boldsymbol{w}\right)-\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right] = \left(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n}\right)\cdot\nabla\gamma-\kappa\gamma\boldsymbol{n}+\boldsymbol{G}, \tag{A.33}$$

wherein the contributions of the usual surface tension effects (Laplace pressure and Marangoni forces) appear in the RHS as well as the interface-defined, singular, force term (if any).

This vector balance relation (A.33) may naturally be separated into normal and tangential parts. When considering the normal component in (A.33), one has

$$\left[\dot{M}\left(\boldsymbol{u}-\boldsymbol{w}\right)\cdot\boldsymbol{n}-\boldsymbol{n}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right] = -\kappa\gamma+\boldsymbol{n}\cdot\boldsymbol{G}, \tag{A.34}$$

while the tangential part reads

$$\left[\dot{M}\left(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n}\right)\cdot\left(\boldsymbol{u}-\boldsymbol{w}\right)-\left(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n}\right)\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right] = \left(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n}\right)\cdot\left(\nabla\gamma+\boldsymbol{G}\right). \tag{A.35}$$

Although not strictly enforced by the latter equation, tangential components of the velocity fields are usually assumed continuous across interfaces in presence of viscous forces. Indeed, the existence of discontinuous tangential velocities across an interface would imply an unbounded shear rate at the interface that viscous effects would tend to eliminate immediately. As discussed in [284] (page 149):

> The condition of continuity of the velocity is thus not an exact law, but a statement of what may be expected to happen, approximately, in normal circumstances.

On the other hand, if at least one of the two fluid phases is considered inviscid, there is no physical argument to enforce continuous tangential velocity components across the interface (as in [290], for instance): this work does *not* allow nor consider such limit cases.

The continuity of tangential velocities across the interface translates into $[(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}) \cdot \boldsymbol{u}] = \boldsymbol{0}$ so that $\left[\dot{M}\,(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}) \cdot (\boldsymbol{u} - \boldsymbol{w})\right] = \boldsymbol{0}$ as well, given that $\left[\dot{M}\right] = 0$ (see (A.26)) and $[\boldsymbol{w}] = \boldsymbol{0}$ by definition. Therefore, (A.33) becomes

$$\boxed{\left[\left(\frac{\dot{M}^2}{\rho} + p\right)\boldsymbol{n} - 2\mu\underline{\boldsymbol{E}} \cdot \boldsymbol{n}\right] = -\kappa\gamma\boldsymbol{n} + (\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}) \cdot \nabla\gamma + \boldsymbol{G},} \tag{A.36}$$

for incompressible two-phase flows of linear isotropic fluids.

A.2.3 Balance of internal energy across the interface

The application of the first law of thermodynamics to the control volume $\Omega_{\Sigma(t),\,\epsilon}$ gives

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{\Sigma(t),\,\epsilon}} \rho\mathcal{E}\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \rho\mathcal{E}\,(\boldsymbol{u} - \boldsymbol{w}) \cdot \boldsymbol{\eta}\,\mathrm{d}\partial\Omega + \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Sigma(t)} \psi\,\mathrm{d}\Sigma \\ = \int_{\Omega_{\Sigma(t),\,\epsilon}} \rho\boldsymbol{f} \cdot \boldsymbol{u}\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \boldsymbol{u} \cdot \underline{\boldsymbol{\sigma}} \cdot \boldsymbol{\eta}\,\mathrm{d}\partial\Omega - \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \boldsymbol{q} \cdot \boldsymbol{\eta}\,\mathrm{d}\partial\Omega + \int_{\Omega_{\Sigma(t),\,\epsilon}} \dot{\Theta}\,\mathrm{d}\Omega \\ + \int_{\partial\Sigma(t)} \gamma\boldsymbol{m} \cdot \boldsymbol{w}\,\mathrm{d}\partial\Sigma + \int_{\Sigma(t)} \boldsymbol{G} \cdot \boldsymbol{w}\,\mathrm{d}\Sigma\end{aligned} \tag{A.37}$$

where ψ is the interfacial energy (see [197, 198]). Using (A.19) and (A.21), this equation can be written as

$$\begin{aligned}\int_{\Omega_{\Sigma(t),\,\epsilon}} \frac{\partial}{\partial t}(\rho\mathcal{E})\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \rho\mathcal{E}\boldsymbol{u}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega - \int_{\Sigma(t)} [\rho\mathcal{E}]\,\boldsymbol{w}\cdot\boldsymbol{n}\,\mathrm{d}\Sigma + \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Sigma(t)} \psi\,\mathrm{d}\Sigma \\ = \int_{\Omega_{\Sigma(t),\,\epsilon}} \rho\boldsymbol{f}\cdot\boldsymbol{u}\,\mathrm{d}\Omega + \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \boldsymbol{u}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega - \int_{\partial\Omega_{\Sigma(t),\,\epsilon}} \boldsymbol{q}\cdot\boldsymbol{\eta}\,\mathrm{d}\partial\Omega + \int_{\Omega_{\Sigma(t),\,\epsilon}} \dot{\Theta}\,\mathrm{d}\Omega \\ + \int_{\partial\Sigma(t)} \gamma\boldsymbol{m}\cdot\boldsymbol{w}\,\mathrm{d}\partial\Sigma + \int_{\Sigma(t)} \boldsymbol{G}\cdot\boldsymbol{w}\,\mathrm{d}\Sigma.\end{aligned} \tag{A.38}$$

Here again, in (A.38), the integrals over $\Omega_{\Sigma(t),\,\epsilon}$ scale as ϵ while the other terms do not; therefore those former terms becomes negligible when $\epsilon \to 0$. In the limit $\epsilon \to 0$, (A.38) thus dictates

$$\begin{aligned}\int_{\Sigma(t)} [\rho\mathcal{E}\,(\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n}]\,\mathrm{d}\Sigma + \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Sigma(t)} \psi\,\mathrm{d}\Sigma \;=\; \int_{\Sigma(t)} [\boldsymbol{u}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}]\,\mathrm{d}\Sigma - \int_{\Sigma(t)} [\boldsymbol{q}\cdot\boldsymbol{n}]\,\mathrm{d}\Sigma \\ + \int_{\partial\Sigma(t)} \gamma\boldsymbol{m}\cdot\boldsymbol{w}\,\mathrm{d}\partial\Sigma + \int_{\Sigma(t)} \boldsymbol{G}\cdot\boldsymbol{w}\,\mathrm{d}\Sigma.\end{aligned} \tag{A.39}$$

As shown in appendix A.5.2, we also have

$$\int_{\partial\Sigma(t)} \gamma\boldsymbol{m}\cdot\boldsymbol{w}\,\mathrm{d}\partial\Sigma = \int_{\Sigma(t)} \nabla_{\mathrm{s}}\cdot\left(\gamma\left(\underline{\boldsymbol{\delta}}-\boldsymbol{nn}\right)\cdot\boldsymbol{w}\right)\,\mathrm{d}\Sigma. \tag{A.40}$$

Therefore, (A.39) is equivalent to

$$\int_{\Sigma(t)} \left[\dot{M}\mathcal{E} - \boldsymbol{u}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n} + \boldsymbol{q}\cdot\boldsymbol{n}\right] + \frac{\mathrm{D}\psi}{\mathrm{D}t} + \psi\nabla_{\mathrm{s}}\cdot\boldsymbol{w}\,\mathrm{d}\Sigma = \int_{\Sigma(t)} \left(\nabla_{\mathrm{s}}\cdot\left(\gamma\left(\underline{\boldsymbol{\delta}}-\boldsymbol{nn}\right)\cdot\boldsymbol{w}\right) + \boldsymbol{G}\cdot\boldsymbol{w}\right)\,\mathrm{d}\Sigma$$

which must hold for any portion of interface $\Sigma(t)$, so that

$$\left[\dot{M}\mathcal{E} - \boldsymbol{u}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n} + \boldsymbol{q}\cdot\boldsymbol{n}\right] + \frac{\mathrm{D}\psi}{\mathrm{D}t} + \psi\nabla_{\mathrm{s}}\cdot\boldsymbol{w} = \nabla_{\mathrm{s}}\cdot\left(\gamma\left(\underline{\boldsymbol{\delta}}-\boldsymbol{nn}\right)\cdot\boldsymbol{w}\right) + \boldsymbol{G}\cdot\boldsymbol{w}. \tag{A.41}$$

This last relation expresses the balance of total energy through the interface in an inertial reference frame.

In order to express this balance in a reference frame moving with the interface (which is more convenient), consider the inner product of $\boldsymbol{w}$ with both sides of equation (A.32). This leads to

$$\left[\dot{M}\boldsymbol{u}\cdot\boldsymbol{w}-\boldsymbol{w}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right]=\boldsymbol{w}\cdot(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n})\cdot\nabla\gamma-\kappa\gamma\boldsymbol{w}\cdot\boldsymbol{n}+\boldsymbol{G}\cdot\boldsymbol{w}. \tag{A.42}$$

Similarly, multiplying side by side (A.26) by $\dfrac{\|\boldsymbol{w}\|^2}{2}$ leads to

$$\left[\dot{M}\frac{\|\boldsymbol{w}\|^2}{2}\right]=0. \tag{A.43}$$

Adding side-by-side (A.43) to (A.41) and subtracting side-by-side (A.42) from the result, one obtains

$$\left[\dot{M}\left(e+\frac{\|\boldsymbol{u}-\boldsymbol{w}\|^2}{2}\right)-(\boldsymbol{u}-\boldsymbol{w})\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}+\boldsymbol{q}\cdot\boldsymbol{n}\right]=\gamma\nabla_{\mathrm{s}}\cdot\boldsymbol{w}. \tag{A.44}$$

which is the desired balance in a reference frame moving with the interface.

Furthermore, since tangential velocities are continuous, note that

$$\left[\frac{\|\boldsymbol{u}-\boldsymbol{w}\|^2}{2}\right]=\left[\frac{\|(\underline{\boldsymbol{\delta}}-\boldsymbol{n}\boldsymbol{n})\cdot(\boldsymbol{u}-\boldsymbol{w})\|^2+((\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n})^2}{2}\right]=\left[\frac{((\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n})^2}{2}\right]=\left[\frac{\dot{M}^2}{2\rho^2}\right],$$

and

$$\begin{aligned}
\left[(\boldsymbol{u}-\boldsymbol{w})\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right] &= \left[\left(\left((\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n}\right)\boldsymbol{n}+\left(\underline{\boldsymbol{\delta}}-\boldsymbol{nn}\right)\cdot(\boldsymbol{u}-\boldsymbol{w})\right)\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right] \\
&= \left[\left((\boldsymbol{u}-\boldsymbol{w})\cdot\boldsymbol{n}\right)\boldsymbol{n}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right]+\left(\underline{\boldsymbol{\delta}}-\boldsymbol{nn}\right)\cdot(\boldsymbol{u}-\boldsymbol{w})\cdot\left[\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right] \\
&= \left[\frac{\dot{M}}{\rho}\boldsymbol{n}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}\right],
\end{aligned}$$

since $\left(\underline{\boldsymbol{\delta}}-\boldsymbol{nn}\right)\cdot(\boldsymbol{u}-\boldsymbol{w})\big|_{\Sigma(t)} = \boldsymbol{0}$. Indeed, since tangential velocities are continuous across $\Sigma(t)$, it is natural to define the tangential components of the interface velocity equal to either fluid's.

Therefore, the first law of thermodynamics translates into the following relation across the interface

$$\left[\dot{M}\left(e+\frac{\dot{M}^2}{2\rho^2}\right)-\frac{\dot{M}}{\rho}\boldsymbol{n}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}+\boldsymbol{q}\cdot\boldsymbol{n}\right] = -\frac{\mathrm{D}\psi}{\mathrm{D}t}+(\gamma-\psi)\,\nabla_{\mathrm{s}}\cdot\boldsymbol{w}. \tag{A.45}$$

Finally, it is shown in [198] (pages 32-34) that fundamental thermodynamics implies

$$\frac{\mathrm{D}\psi}{\mathrm{D}t} = -T_{\mathrm{I}}\frac{\mathrm{D}}{\mathrm{D}t}\left(\frac{\mathrm{d}\gamma}{\mathrm{d}T_{\mathrm{I}}}\right) \quad \text{and} \quad \gamma-\psi = T_{\mathrm{I}}\frac{\mathrm{d}\gamma}{\mathrm{d}T_{\mathrm{I}}} \tag{A.46}$$

for a massless interface of thermodynamic temperature T_{I}. Hence, conservation of energy across $\Sigma(t)$ translates into

$$\left[\dot{M}\left(e+\frac{\dot{M}^2}{2\rho^2}\right)-\frac{\dot{M}}{\rho}\boldsymbol{n}\cdot\underline{\boldsymbol{\sigma}}\cdot\boldsymbol{n}+\boldsymbol{q}\cdot\boldsymbol{n}\right] = T_{\mathrm{I}}\left(\frac{\mathrm{D}}{\mathrm{D}t}\left(\frac{\mathrm{d}\gamma}{\mathrm{d}T_{\mathrm{I}}}\right)+\frac{\mathrm{d}\gamma}{\mathrm{d}T_{\mathrm{I}}}\nabla_{\mathrm{s}}\cdot\boldsymbol{w}\right). \tag{A.47}$$

For incompressible two-phase flows of linear isotropic fluids, (A.47) becomes

$$\boxed{\left[\dot{M}\left(h+\frac{\dot{M}^2}{2\rho^2}\right)-2\dot{M}\nu\boldsymbol{n}\cdot\underline{\boldsymbol{E}}\cdot\boldsymbol{n}-k\nabla T\cdot\boldsymbol{n}\right]=T_{\mathrm{I}}\left(\frac{\mathrm{D}}{\mathrm{D}t}\left(\frac{\mathrm{d}\gamma}{\mathrm{d}T_{\mathrm{I}}}\right)+\frac{\mathrm{d}\gamma}{\mathrm{d}T_{\mathrm{I}}}\nabla_{\mathrm{s}}\cdot\boldsymbol{w}\right).} \tag{A.48}$$

wherein $h=e+\dfrac{p}{\rho}$ is the enthalpy per unit mass in (A.48), and $\dot{M}=\rho\left(\boldsymbol{u}-\boldsymbol{w}\right)\cdot\boldsymbol{n}$ is the mass flux across the interface (toward the positive domain).

A.3 Divergence theorem

Also known as Gauss's theorem or Ostrogradsky's theorem, the divergence theorem shows the equality between the integral of a flux across a closed surface Σ and the integral of the flux divergence over the volume Ω encapsulated by Σ. In other words,

$$\int_{\Sigma} Q_{ab\ldots}n_i\,\mathrm{d}\Sigma=\int_{\Omega}\frac{\partial}{\partial x_i}\left(Q_{ab\ldots}\right)\,\mathrm{d}\Omega \tag{A.49}$$

where $\boldsymbol{n}$ is the outer unit normal vector to Σ, and $Q_{ab\ldots}$ are the components of a general N^{th} order tensor.

A.4 Stokes' theorem

The Stokes' theorem relates the circulation along a closed curve Γ to a surface integral over any surface Σ bounded by Γ. In other words,

$$\int_{\Gamma} Q_{ab\ldots}\tau_i\,\mathrm{d}\Gamma=\int_{\Sigma}\varepsilon_{kli}\frac{\partial}{\partial x_l}\left(Q_{ab\ldots}\right)n_k\,\mathrm{d}\Sigma \tag{A.50}$$

where $\boldsymbol{\tau}$ is a unit tangent vector to Γ and $\boldsymbol{n}$ is the unit normal vector to Σ with positive orientation: at every point on Γ, $\boldsymbol{\tau} \times \boldsymbol{n}$ is a unit vector, outward normal to Γ, tangent to Σ.

A.5 Surface tension effects

A.5.1 Contribution to balance of momentum across the interface

We show here relation (A.30). Let $\boldsymbol{\mathcal{T}} = \int_{\partial\Sigma(t)} \gamma \boldsymbol{m} \,\mathrm{d}\Sigma = \int_{\partial\Sigma(t)} \gamma \boldsymbol{\tau} \times \boldsymbol{n} \,\mathrm{d}\Sigma$, consider the i^{th} component of $\boldsymbol{\mathcal{T}}$, we have (using Stokes' theorem, see app. A.4)

$$\begin{aligned}
\mathcal{T}_i &= \int_{\partial\Sigma(t)} \gamma m_i \,\mathrm{d}\partial\Sigma = \int_{\partial\Sigma(t)} \varepsilon_{ijk} \gamma \tau_j n_k \,\mathrm{d}\partial\Sigma \\
&= \int_{\Sigma(t)} \varepsilon_{pqj} \frac{\partial}{\partial x_q} \left(\varepsilon_{ijk} \gamma n_k\right) n_p \,\mathrm{d}\Sigma = \int_{\Sigma(t)} \varepsilon_{jpq} \varepsilon_{jki} \frac{\partial}{\partial x_q} \left(\gamma n_k\right) n_p \,\mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \left(\delta_{pk}\delta_{qi} - \delta_{pi}\delta_{qk}\right) \frac{\partial}{\partial x_q} \left(\gamma n_k\right) n_p \,\mathrm{d}\Sigma = \int_{\Sigma(t)} \left(\frac{\partial}{\partial x_i} \left(\gamma n_k\right) n_k - \frac{\partial}{\partial x_k} \left(\gamma n_k\right) n_i \right) \mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \left(\frac{\partial \gamma}{\partial x_i} - \frac{\partial \gamma}{\partial x_k} n_k n_i - \gamma \kappa n_i \right) \mathrm{d}\Sigma
\end{aligned}$$

which is the i^{th} component of

$$\int_{\Sigma(t)} \left(\left(\underline{\boldsymbol{\delta}} - \boldsymbol{n}\boldsymbol{n}\right) \cdot \nabla \gamma - \gamma \kappa \boldsymbol{n} \right) \mathrm{d}\Sigma.$$

Remark. *We define the curvature by* $\kappa = \nabla \cdot \boldsymbol{n} = \dfrac{\partial n_k}{\partial x_k}$.

A.5.2 Contribution to balance of energy across the interface

We show here relation (A.40). Using Stokes' theorem, see app. A.4, we have

$$\begin{aligned}
\int_{\partial\Sigma(t)} \gamma \boldsymbol{m}\cdot\boldsymbol{w}\,\mathrm{d}\partial\Sigma &= \int_{\partial\Sigma(t)} \gamma\varepsilon_{ijk}\tau_j n_k w_i\,\mathrm{d}\partial\Sigma = \int_{\Sigma(t)} \varepsilon_{pqj}\partial_{x_q}\left(\gamma\varepsilon_{ijk}n_k w_i\right) n_p\,\mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \varepsilon_{jki}\varepsilon_{jpq}\partial_{x_q}\left(\gamma n_k w_i\right) n_p\,\mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \left(\delta_{kp}\delta_{iq}-\delta_{kq}\delta_{ip}\right)\partial_{x_q}\left(\gamma n_k w_i\right) n_p\,\mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \left(\partial_{x_i}\left(\gamma n_k w_i\right) n_k - \partial_{x_k}\left(\gamma n_k w_i\right) n_i\right)\,\mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \left(\partial_{x_i}\left(\gamma w_i\right) - n_i n_k \partial_{x_k}\left(\gamma w_i\right) - \gamma w_i n_i \kappa\right)\,\mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \left(\left(\delta_{ik}-n_i n_k\right)\partial_{x_k}\left(\gamma w_i\right) - \gamma w_i n_i \kappa\right)\,\mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \left(\left(\delta_{ik}-n_i n_k\right)\partial_{x_k}\left(\left(\delta_{ij}-n_i n_j\right)\gamma w_j\right)\right)\,\mathrm{d}\Sigma \\
&= \int_{\Sigma(t)} \left(\underline{\boldsymbol{\delta}}-\boldsymbol{nn}\right) : \nabla\left(\gamma\left(\underline{\boldsymbol{\delta}}-\boldsymbol{nn}\right)\cdot\boldsymbol{w}\right)\,\mathrm{d}\Sigma.
\end{aligned}$$

Appendix B

Least-square interpolation and differential operators

In this appendix, we present the methodology used to build interpolation and differential operators based from a set of unstructured points. Consider a set of (unstructured) points $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and the corresponding values of a (scalar) field u sampled at those points $\{u_0, u_1, \ldots, u_N\}$. Let $\bar{\boldsymbol{x}}$ be the point where the value of u and/or its derivatives is/are sought; let $x = (\boldsymbol{x} - \bar{\boldsymbol{x}}) \cdot \boldsymbol{e}_x$, $y = (\boldsymbol{x} - \bar{\boldsymbol{x}}) \cdot \boldsymbol{e}_y$ and $z = (\boldsymbol{x} - \bar{\boldsymbol{x}}) \cdot \boldsymbol{e}_z$. We locally approximate u by an interpolation polynomial $p\,(x, y, z)$ such that (up to second-degree)

$$p\,(x,\, y,\, z) = \mathtt{x}\mathtt{a}^{\mathrm{T}} \tag{B.1}$$

wherein

$$\mathtt{a} = \begin{pmatrix} a_0 & a_x & a_y & a_z & a_{xx} & a_{xy} & a_{xz} & a_{yy} & a_{yz} & a_{zz} \end{pmatrix} \tag{B.2}$$

and

$$\mathtt{x} = \begin{pmatrix} 1 & x & y & z & x^2 & xy & xz & y^2 & yz & z^2 \end{pmatrix}. \tag{B.3}$$

If a lesser degree of interpolation is desired or required, one obtains a linear (resp. constant) interpolation polynomial by truncating (B.2) and (B.3) to their 4 (resp. 1) first elements.

Let $x_i = (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \cdot \boldsymbol{e}_x$, $y_i = (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \cdot \boldsymbol{e}_y$ and $z_i = (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \cdot \boldsymbol{e}_z$, $i = 0, \ldots, N$. We define the matrix

$$\mathtt{X} = \begin{pmatrix} 1 & x_0 & y_0 & z_0 & x_0^2 & x_0 y_0 & x_0 z_0 & y_0^2 & y_0 z_0 & z_0^2 \\ 1 & x_1 & y_1 & z_1 & x_1^2 & x_1 y_1 & x_1 z_1 & y_1^2 & y_1 z_0 & z_1^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & z_N & x_N^2 & x_N y_N & x_N z_N & y_N^2 & y_N z_N & z_N^2 \end{pmatrix}. \tag{B.4}$$

The sampling errors $e_i = p(x_i, y_i, z_i) - u_i$ are obtained via

$$\mathsf{e}^{\mathrm{T}} = \mathtt{X}\mathsf{a}^{\mathrm{T}} - \mathsf{u}^{\mathrm{T}} \tag{B.5}$$

where in $\mathsf{u} = (u_0 \ u_1 \ \ldots \ u_N)$ and $\mathsf{e} = (e_0 \ e_1 \ \ldots \ e_N)$. The least-square method aims to minimize the two-norm of the sampling errors e_i, which means minimizing

$$\|\mathsf{e}\|_2^2 = \mathsf{e}\mathsf{e}^{\mathrm{T}} = \mathsf{a}\mathtt{X}^{\mathrm{T}}\mathtt{X}\mathsf{a}^{\mathrm{T}} - 2\mathsf{a}\mathtt{X}^{\mathrm{T}}\mathsf{u}^{\mathrm{T}} + \mathsf{u}\mathsf{u}^{\mathrm{T}}. \tag{B.6}$$

The minimum of the positive-definite quadratic form (B.6) is obtained for the coefficients

$$\bar{\mathsf{a}} = \left(\mathtt{X}^{\mathrm{T}}\mathtt{X}\right)^{-1} \mathtt{X}^{\mathrm{T}}\mathsf{u}^{\mathrm{T}}, \tag{B.7}$$

wherein we evaluate $\left(\mathtt{X}^{\mathrm{T}}\mathtt{X}\right)^{-1}$ by Cholesky factorization of $\mathtt{X}^{\mathrm{T}}\mathtt{X}$.

This final result (B.7) shows that every coefficient $\bar{a}$ can be expressed as a linear combination of the values of u sampled on the surrounding data points. By extracting rows of $\left(\mathtt{X}^{\mathrm{T}}\mathtt{X}\right)^{-1} \mathtt{X}^{\mathrm{T}}$, one obtains the weights of that linear combination (which are independent of $\mathtt{u}$) producing the corresponding coefficient in $\mathtt{a}$, once applied to a particular set of values u.

In particular, let $\mathtt{r}_k$ be the k^{th} row of $\left(\mathtt{X}^{\mathrm{T}}\mathtt{X}\right)^{-1}\mathtt{X}^{\mathrm{T}}$, we have

- $a_0 = \mathtt{r}_0\mathtt{u}^{\mathrm{T}} = p(0,0,0)$ which interpolates u at $\bar{\boldsymbol{x}}$;
- $a_x = \mathtt{r}_1\mathtt{u}^{\mathrm{T}} = \partial_x p|_{\mathbf{0}}$ which one may use to approximate $\partial_x u|_{\bar{\boldsymbol{x}}}$;
- $a_y = \mathtt{r}_2\mathtt{u}^{\mathrm{T}} = \partial_y p|_{\mathbf{0}}$ which one may use to approximate $\partial_y u|_{\bar{\boldsymbol{x}}}$;
- $a_z = \mathtt{r}_3\mathtt{u}^{\mathrm{T}} = \partial_z p|_{\mathbf{0}}$ which one may use to approximate $\partial_z u|_{\bar{\boldsymbol{x}}}$;
- $a_{xx} = \mathtt{r}_4\mathtt{u}^{\mathrm{T}} = \left.\dfrac{1}{2}\dfrac{\partial^2 p}{\partial x\partial x}\right|_{\mathbf{0}}$ which one may use to approximate $\left.\dfrac{1}{2}\dfrac{\partial^2 u}{\partial x\partial x}\right|_{\bar{\boldsymbol{x}}}$;
- $a_{xy} = \mathtt{r}_5\mathtt{u}^{\mathrm{T}} = \left.\dfrac{\partial^2 p}{\partial x\partial y}\right|_{\mathbf{0}}$ which one may use to approximate $\left.\dfrac{\partial^2 u}{\partial x\partial y}\right|_{\bar{\boldsymbol{x}}}$;
- $a_{xz} = \mathtt{r}_6\mathtt{u}^{\mathrm{T}} = \left.\dfrac{\partial^2 p}{\partial x\partial z}\right|_{\mathbf{0}}$ which one may use to approximate $\left.\dfrac{\partial^2 u}{\partial x\partial z}\right|_{\bar{\boldsymbol{x}}}$;
- $a_{yy} = \mathtt{r}_7\mathtt{u}^{\mathrm{T}} = \left.\dfrac{1}{2}\dfrac{\partial^2 p}{\partial y\partial y}\right|_{\mathbf{0}}$ which one may use to approximate $\left.\dfrac{1}{2}\dfrac{\partial^2 u}{\partial y\partial y}\right|_{\bar{\boldsymbol{x}}}$;
- $a_{yz} = \mathtt{r}_8\mathtt{u}^{\mathrm{T}} = \left.\dfrac{\partial^2 p}{\partial y\partial z}\right|_{\mathbf{0}}$ which one may use to approximate $\left.\dfrac{\partial^2 u}{\partial y\partial z}\right|_{\bar{\boldsymbol{x}}}$;
- $a_{zz} = \mathtt{r}_9\mathtt{u}^{\mathrm{T}} = \left.\dfrac{1}{2}\dfrac{\partial^2 p}{\partial z\partial z}\right|_{\mathbf{0}}$ which one may use to approximate $\left.\dfrac{1}{2}\dfrac{\partial^2 u}{\partial z\partial z}\right|_{\bar{\boldsymbol{x}}}$;

Appendix C

Quadratic non-oscillatory interpolation schemes ensuring continuity in uniform regions

Let us consider a computational cell $C = [0;\, h_x] \times [0;\, h_y] \times [0;\, h_z]$, and a point (x, y, z) within that cell. Let q be a scalar field to be interpolated at (x, y, z) from its values (and second derivatives) sampled at the vertices of C. We denote by $q_{abc} = q\left(ah_x, bh_y, ch_z\right)$, where a, b and c are either 0 or 1.

Let

$$F_i\left(z\right) = \begin{cases} 1 - z & \text{if } i = 0, \\ z & \text{if } i = 1, \end{cases} \tag{C.1}$$

then the multilinear interpolation of q in C, at point (x, y, z) reads

$$q^{\text{lin.}}\left(x, y, z\right) = \sum_{i=0}^{1} \sum_{j=0}^{1} \sum_{k=0}^{1} q_{ijk} F_i\left(x\right) F_j\left(y\right) F_k\left(z\right) \tag{C.2}$$

and is continuous on faces shared between cells of same size, by construction.

One may build a quadratic interpolant by augmenting (C.2) with quadratic terms in the following fashion

$$q^{\text{quad.}}\left(x,y,z\right)=q^{\text{lin.}}\left(x,y,z\right)-q_{xx}\frac{x\left(1-x\right)}{2}-q_{yy}\frac{y\left(1-y\right)}{2}-q_{zz}\frac{z\left(1-z\right)}{2}. \tag{C.3}$$

In [106], a stabilized quadratic interpolation scheme like (C.3) is introduced with

$$q_{xx}=\text{minmod}_{(i,j,k)\in\{0,\,1\}^3}\left(\mathrm{D}_{xx}q_{i,j,k}\right), \tag{C.4}$$

and similarly for q_{yy} and q_{zz}. However, this interpolation scheme does not ensure continuity across cells in uniform regions because the value of q_{xx} (for instance) may differ across a face that is parallel to direction x.

Here below, we present two alternative interpolation schemes that alleviate this issue by splitting the minmod operation in (C.4) and combining it with linear interpolation in relevant cross-section. For the sake of clarity, we present only the calculation of q_{xx} hereafter but the reasoning is similar for q_{yy} and q_{zz}.

A first version ensuring continuity in uniform regions. Let e_{jk} be the edge of C parallel to direction x of coordinates $y=jh_y$ and $z=kh_z$. For every such edge, one may define

$$q_{xx,jk}=\text{minmod}_{i\in\{0,\,1\}}\left(\mathrm{D}_{xx}q_{i,j,k}\right), \tag{C.5}$$

and subsequently define

$$q_{xx}=\sum_{j=0}^{1}\sum_{k=0}^{1}q_{xx,jk}F_j\left(y\right)F_k\left(z\right). \tag{C.6}$$

By construction, this operation ensures that, at any point of a face parallel to x, q_{xx}

depends only on the values of $\mathrm{D}_{xx}q_{i,j,k}$ at the vertices of that face.

A second version ensuring continuity in uniform regions. Consider the faces of C perpendicular to direction x and interpolate the values of $\mathrm{D}_{xx}q$ on those faces. One obtains

$$q_{xx,0} = \sum_{j=0}^{1}\sum_{k=0}^{1} F_j\left(y\right) F_k\left(z\right) \mathrm{D}_{xx}q_{0,j,k}, \tag{C.7}$$

and

$$q_{xx,1} = \sum_{j=0}^{1}\sum_{k=0}^{1} F_j\left(y\right) F_k\left(z\right) \mathrm{D}_{xx}q_{1,j,k}. \tag{C.8}$$

One may then define

$$q_{xx} = \mathrm{minmod}\left(q_{xx,0}, q_{xx,1}\right). \tag{C.9}$$

By construction, this operation ensures that, at any point of a face parallel to x, q_{xx} depends only on the values of $\mathrm{D}_{xx}q_{i,j,k}$ at the vertices of that face, as well.

Bibliography

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

[2] S. O. Unverdi and G. Tryggvason, *A front-tracking method for viscous, incompressible, multifluid flows*, *J. Comput. Phys.* **100** (1992) 25–37.

[3] C. Peskin, *The immersed boundary method*, *Acta Numerica* **11** (2002) 479–517.

[4] C. Peskin, *Numerical analysis of blood flow in the heart*, *J. Comput. Phys.* **25** (1977) 220–252.

[5] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, *J. Comput. Phys.* **114** (1994) 146–159.

[6] Y.-C. Chang, T. Hou, B. Merriman, and S. Osher, *Eulerian capturing methods based on a level set formulation for incompressible fluid interfaces*, *J. Comput. Phys.* **124** (1996) 449–464.

[7] M. Sussman, A. S. Algrem, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, *An adaptive level set approach for incompressible two-phase flow*, *J. Comput. Phys* **148** (1999) 81–124.

[8] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, and S. Zaleski, *Volume of Fluid Interface Tracking with Smoothed Surface Stress Methods for Three Dimensional Flows*, *Journal of Computational Physics* **152** (1999) 423–456.

[9] M. Muradoglu and G. Tryggvason, *A front-tracking method for computation of interfacial flows with soluble surfactants*, *Journal of Computational Physics* **227** (2008), no. 4 2238 – 2262.

[10] E. Marchandise, P. Geuzaine, N. Chevaugeon, and J.-F. Remacle, *A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics*, *Journal of Computational Physics* **225** (2007), no. 1 949–974.

[11] J. Shaikh, A. Sharma, and R. Bhardwaj, *On comparison of the sharp-interface and diffuse-interface level set methods for 2d capillary or/and gravity induced flows*, *Chemical Engineering Science* **176** (2018) 77–95.

[12] S. Mirjalili, C. B. Ivey, and A. Mani, *Comparison between the diffuse interface and volume of fluid methods for simulating two-phase flows*, *International Journal of Multiphase Flow* **116** (2019) 221–238.

[13] S. Popinet, *An accurate adaptive solver for surface-tension-driven interfacial flows*, *Journal of Computational Physics* **228** (September, 2009) 5838–5866.

[14] S. Popinet and S. Zaleski, *A front-tracking algorithm for accurate representation of surface tension*, *International Journal for Numerical Methods in Fluids* **30** (1999), no. 6 775–793.

[15] M. Kang, R. Fedkiw, and X.-D. Liu, *A boundary condition capturing method for multiphase incompressible flow*, *J. Sci. Comput.* **15** (2000) 323–360.

[16] X.-D. Liu, R. P. Fedkiw, and M. Kang, *A boundary condition capturing method for poisson's equation on irregular domains*, *Journal of Computational Physics* **160** (2000), no. 1 151 – 178.

[17] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, *A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method)*, *Journal of Computational Physics* **152** (1999), no. 2 457 – 492.

[18] M. M. Francois, S. J. Cummins, E. D. Dendy, D. B. Kothe, J. M. Sicilian, and M. W. Williams, *A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework*, *Journal of Computational Physics* **213** (2006), no. 1 141–173.

[19] O. Desjardins, V. Moureau, and H. Pitsch, *An accurate conservative level set/ghost fluid method for simulating turbulent atomization*, *Journal of Computational Physics* **227** (2008), no. 18 8395–8416.

[20] J.-M. Hong, T. Shinar, M. Kang, and R. Fedkiw, *On boundary condition capturing for multiphase interfaces*, *Journal of scientific Computing* **31** (2007), no. 1 99–125.

[21] J.-M. Hong and C.-H. Kim, *Discontinuous fluids*, *ACM Transactions on Graphics (TOG)* **24** (2005), no. 3 915–920.

[22] M. Sussman, K. Smith, M. Hussaini, M. Ohta, and R. Zhiwei, *A sharp interface method for incompressible two-phase flows*, *Journal of Computational Physics* **221** (Feb., 2007) 469–505.

[23] M. Sussman and E. G. Puckett, *A coupled level let and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows*, *J. Comput. Phys.* **162** (2000) 301–337.

[24] M. Sussman, *A parallelized, adaptive algorithm for multiphase flows in general geometries*, *Computers & Structures* **83** (2005), no. 6 435–444. Frontier of Multi-Phase Flow Analysis and Fluid-Structure.

[25] M. Sussman, *A second-order coupled level-set and volume-of-fluid method for computing growth and collapse of vapor bubbles*, *J. Comp. Phys.* **187** (2003) 110 – 136.

[26] B. Lalanne, L. R. Villegas, S. Tanguy, and F. Risso, *On the computation of viscous terms for incompressible two-phase flows with Level Set/Ghost Fluid Method*, *Journal of Computational Physics* **301** (2015) 289 – 307.

[27] A. Dalmon, K. Kentheswaran, G. Mialhe, B. Lalanne, and S. Tanguy, *Fluids-membrane interaction with a full eulerian approach based on the level set method*, *Journal of Computational Physics* **406** (2020) 109171.

[28] R. Egan and F. Gibou, *xgfm: Recovering convergence of fluxes in the ghost fluid method*, *Journal of Computational Physics* **409** (2020) 109351.

[29] J. Bedrossian, J. H. von Brecht, S. Zhu, E. Sifakis, and J. M. Teran, *A second order virtual node method for elliptic problems with interfaces and irregular domains*, *Journal of Computational Physics* **229** (2010), no. 18 6405 – 6426.

[30] D. C. Assncio and J. M. Teran, *A second order virtual node algorithm for stokes flow problems with interfacial forces, discontinuous material properties and irregular domains*, *Journal of Computational Physics* **250** (2013) 77–105.

[31] C. Schroeder, A. Stomakhin, R. Howes, and J. M. Teran, *A second order virtual node algorithm for navierstokes flow problems with interfacial forces and discontinuous material properties*, *Journal of Computational Physics* **265** (2014) 221–245.

[32] M. Theillard, F. Gibou, and D. Saintillan, *Sharp numerical simulation of incompressible two-phase flows*, *Journal of Computational Physics* **391** (2019) 91–118.

[33] H. Cho and M. Kang, *Fully implicit and accurate treatment of jump conditions for two-phase incompressible navier-stokes equation*, *arXiv preprint arXiv:2005.13724* (2020).

[34] S.Welch, *Local simulation of two-phase flows including interface tracking with mass transfer*, *J. Comput. Phys.* **121** (1995) 142.

[35] G. Son and V. Dhir, *Numerical simulation of saturated film boiling on a horizontal surface*, *J. Heat. Transfer.* **119** (1997) 525.

[36] D. Juric and G. Tryggvason, *Computations of boiling flows*, *Int. J. Multiphase. Flow.* **24** (1998) 387–410.

[37] Y. Sato and B. Nieno, *A sharp-interface phase change model for a mass-conservative interface tracking method*, *Journal of Computational Physics* **249** (2013) 127–161.

[38] M. W. Akhtar and S. J. Kleis, *Boiling flow simulations on adaptive octree grids*, *International Journal of Multiphase Flow* **53** (2013) 88–99.

[39] J. Palmore and O. Desjardins, *A volume of fluid framework for interface-resolved simulations of vaporizing liquid-gas flows*, *Journal of Computational Physics* **399** (2019) 108954.

[40] S.Welch and J. Wilson, *A volume of fluid based method for fluid flows with phase change*, *J. Comput. Phys.* **160** (2000) 662.

[41] G. Son and V. Dhir, *Numerical simulation of film boiling near critical pressures with a level set method*, *J. Heat. Transfer.* **120** (1998) 183.

[42] G. Tomar, G. Biswas, A. Sharma, and A. Agrawal, *Numerical simulation of bubble growth in film boiling using a coupled level-set and volume-of-fluid method*, *Phys. of Fluid* **17** (2005) 112103.

[43] D. Q. Nguyen, R. P. Fedkiw, and M. Kang, *A boundary condition capturing method for incompressible flame discontinuities*, *Journal of Computational Physics* **172** (2001), no. 1 71 – 98.

[44] F. Gibou, L. Chen, D. Nguyen, and S. Banerjee, *A level set based sharp interface method for incompressible flows with phase change*, *J. Comput. Phys.* **222** (2007) 536–555.

[45] G. Son and V. Dhir, *A level set method for analysis of film boiling on an immersed solid surface*, *Numer. Heat Transf.* **B52** (2007) 153–177.

[46] G. Son and V. Dhir, *Numerical simulation of nucleate boiling on a horizontal surface at high heat fluxes*, *Int. J. Heat Mass Transf.* **51** (2008) 2566–2582.

[47] G. Son and V. K. Dhir, *Three-dimensional simulation of saturated film boiling on a horizontal cylinder*, *International Journal of Heat and Mass Transfer* **51** (2008), no. 5 1156–1167.

[48] S. Tanguy, T. Ménard, and A. Berlemont, *A Level Set Method for vaporizing two-phase flows*, *Journal of Computational Physics* **221** (2007), no. 2 837 – 853.

[49] S. Tanguy, M. Sagan, B. Lalanne, F. Couderc, and C. Colin, *Benchmarks and numerical methods for the simulation of boiling flows*, Journal of Computational Physics **264** (2014) 1 – 22.

[50] L. R. Villegas, R. Alis, M. Lepilliez, and S. Tanguy, *A Ghost Fluid/Level Set Method for boiling flows and liquid evaporation: Application to the Leidenfrost effect*, *Journal of Computational Physics* **316** (2016) 789 – 813.

[51] H. S. Raut, A. Bhattacharya, and A. Sharma, *Dual grid level set method based direct numerical simulations of nucleate boiling with oscillating base plate*, *International Journal of Thermal Sciences* **162** (2021) 106785.

[52] R. Egan, A. Guittet, F. Temprano-Coleto, T. Isaac, F. J. Peaudecerf, J. R. Landel, P. Luzzatto-Fegiz, C. Burstedde, and F. Gibou, *Direct numerical simulation of incompressible flows on parallel octree grids*, *Journal of Computational Physics* **428** (2021) 110084.

[53] R. Egan and F. Gibou, *Geometric discretization of the multidimensional dirac delta distribution application to the poisson equation with singular source terms*, *Journal of Computational Physics* **346** (2017) 71–90.

[54] R. Egan and F. Gibou, *Fast and scalable algorithms for constructing solvent-excluded surfaces of large biomolecules*, *Journal of Computational Physics* **374** (2018) 91–120.

[55] A. Guittet, M. Theillard, and F. Gibou, *A stable projection method for the incompressible navier-stokes equations on arbitrary geometries and adaptive quad/octrees*, *Journal of Computational Physics* (2015).

[56] M. Mirzadeh, A. Guittet, C. Burstedde, and F. Gibou, *Parallel level-set methods on adaptive tree-based grids*, *Journal of Computational Physics* **322** (2016) 345 – 364.

[57] J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics.* Springer-Verlag Berlin Heidelberg, 2002.

[58] K. Shahbazi, P. F. Fischer, and C. R. Ethier, *A high-order discontinuous Galerkin method for the unsteady incompressible Navier-Stokes equations*, *Journal of Computational Physics* **222** (2007), no. 1 391 – 407.

[59] V. Girault, B. Riviere, and M. F. Wheeler, *A discontinuous Galerkin method with nonoverlapping domain decomposition for the Stokes and Navier-Stokes problems*, *Mathematics of Computation* **74** (2005) 53–84.

[60] C. Peskin, *Flow patterns around heart valves: A numerical method*, *J. Comput. Phys.* **10** (1972) 252–271.

[61] M.-C. Lai and C. S. Peskin, *An immersed boundary method with formal second-order accuracy and reduced numerical viscosity*, *Journal of Computational Physics* **160** (2000), no. 2 705 – 719.

[62] E. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof, *Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations*, *Journal of Computational Physics* **161** (2000), no. 1 35 – 60.

[63] S. Osher and J. Sethian, *Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations*, *Journal of Computational Physics* **79** (1988) 12–49.

[64] Y. T. Ng, C. Min, and F. Gibou, *An efficient fluid–solid coupling algorithm for single-phase flows*, *Journal of Computational Physics* **228** (Dec., 2009) 8807–8829.

[65] M. Vinokur, *On one-dimensional stretching functions for finite difference calculations*, *Journal of Computational Physics* **50** (1983), no. 2 215–234.

[66] E. J. Avital, N. D. Sandham, and K. H. Luo, *Stretched cartesian grids for solution of the incompressible Navier-Stokes equations*, *International Journal for Numerical Methods in Fluids* **33** (2000), no. 6 897–918.

[67] M. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, *J. Comput. Phys.* **53** (1984) 484–512.

[68] L. H. Howell and J. B. Bell, *An adaptive mesh projection method for viscous incompressible flow*, *SIAM Journal on Scientific Computing* **18** (1997), no. 4 996–1013, [http://dx.doi.org/10.1137/S1064827594270555].

[69] M. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, *J. Comput. Phys.* **82** (1989) 64–84.

[70] F. Sousa, C. Lages, J. Ansoni, A. Castelo, and A. Simao, *A finite difference method with meshless interpolation for incompressible flows in non-graded tree-based grids*, *Journal of Computational Physics* **396** (2019) 848 – 866.

[71] J. A. Benek, J. Steger, and F. C. Dougherty, *A flexible grid embedding technique with applications to the Euler equations*, *6th Computational Fluid Dynamics Conference, AIAA, 373–382.* (1983).

[72] R. E. English, L. Qiu, Y. Yu, and R. Fedkiw, *An adaptive discretization of incompressible flow using a multitude of moving cartesian grids*, *Journal of Computational Physics* **254** (2013) 107 – 154.

[73] D. DeZeeuw and K. G. Powell, *An adaptively refined cartesian mesh solver for the Euler equations*, *Journal of Computational Physics* **104** (1993), no. 1 56 – 68.

[74] S. Karman, *SPLITFLOW: A 3D unstructured cartesian/prismatic grid CFD code for complex geometries*, in *33rd Aerospace Sciences Meeting and Exhibit*, 1995.

[75] J. Melton, *Automated three-dimensional cartesian grid generation and euler flow solutions for arbitrary geometries.* PhD thesis, Univerty of California, Davis, 1996.

[76] J. Melton, M. Berger, M. Aftosmis, and M. Wong, *3d applications of a cartesian grid Euler method*, in *33rd Aerospace Sciences Meeting and Exhibit*, 1995.

[77] J. Melton, F. Enomoto, and M. Berger, *3D automatic cartesian grid generation for Euler flows*, in *11th Computational Fluid Dynamics Conference, AIAA-93-3386-CP*, 1993.

[78] R. Finkel and J. Bentley, *Quad trees a data structure for retrieval on composite keys*, *Acta Informatica* **4** (1974), no. 1 1–9.

[79] D. Meagher, *Geometric modeling using octree encoding*, *Computer Graphics and Image Processing* **19** (1982), no. 2 129 – 147.

[80] W. J. Coirier, *An adaptively-refined, cartesian, cell-based scheme for the Euler and Navier-Stokes equations*, tech. rep., NASA, 1994.

[81] A. Khokhlov, *Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations*, *Journal of Computational Physics* **143** (1998), no. 2 519 – 543.

[82] F. Harlow and J. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, *Phys. Fluids* **8** (1965) 2182–2189.

[83] S. Popinet, *Gerris: A tree-based adaptive solver for the incompressible euler equations in complex geometries*, *J. Comput. Phys.* **190** (2003) 572–600.

[84] F. Losasso, F. Gibou, and R. Fedkiw, *Simulating water and smoke with an octree data structure*, *ACM Trans. Graph. (SIGGRAPH Proc.)* (2004) 457–462.

[85] C. Min and F. Gibou, *A second-order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids*, *J. Comput. Phys.* **219** (2006) 912–929.

[86] J. Strain, *Semi-Lagrangian methods for level set equations*, *Journal of Computational Physics* **151** (1999) 498–533.

[87] D. Xiu and G. Karniadakis, *A semi-Lagrangian high-order method for Navier-Stokes equations*, *J. Comput. Phys* **172** (2001) 658–684.

[88] F. Losasso, R. Fedkiw, and S. Osher, *Spatially Adaptive Techniques for Level Set Methods and Incompressible Flow*, *Computers and Fluids* **35** (2006) 995–1010.

[89] G. Karypis and V. Kumar, *A parallel algorithm for multilevel graph partitioning and sparse matrix ordering*, *Journal of Parallel Distributed Computing* **48** (Jan., 1998) 71–95.

[90] 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, (Washington, DC, USA), pp. 230–, IEEE Computer Society, 1997.

[91] P. M. Campbell, K. D. Devine, J. E. Flaherty, L. G. Gervasio, and J. D. Teresco, *Dynamic octree load balancing using space-filling curves*, Tech. Rep. CS-03-01, Williams College Department of Computer Science, 2003.

[92] M. Griebel and G. W. Zumbusch, *Parallel multigrid in an adaptive PDE solver based on hashing and space-filling curves*, *Parallel Computing* **25** (1999) 827–843.

[93] H.-J. Bungartz, M. Mehl, and T. Weinzierl, *A parallel adaptive Cartesian PDE solver using space–filling curves*, *Euro-Par 2006 Parallel Processing* (2006) 1064–1074.

[94] T. Weinzierl and M. Mehl, *Peano—a traversal and storage scheme for octree-like adaptive Cartesian multiscale grids*, *SIAM Journal on Scientific Computing* **33** (Oct., 2011) 2732–2760.

[95] T. Tu, D. R. O'Hallaron, and O. Ghattas, *Scalable parallel octree meshing for terascale applications*, in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, (Washington, DC, USA), pp. 4–, IEEE Computer Society, 2005.

[96] R. S. Sampath, S. S. Adavani, H. Sundar, I. Lashuk, and G. Biros, *Dendro: Parallel algorithms for multigrid and AMR methods on 2:1 balanced octrees*, in *International Conference for High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008.*, 2008.

[97] C. Burstedde, O. Ghattas, M. Gurnis, G. Stadler, E. Tan, T. Tu, L. C. Wilcox, and S. Zhong, *Scalable adaptive mantle convection simulation on petascale supercomputers*, in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08, (Piscataway, NJ, USA), pp. 62:1–62:15, IEEE Press, 2008.

[98] R. S. Sampath and G. Biros, *A parallel geometric multigrid method for finite elements on octree meshes, siam journal on scientific computing*, *SIAM J. of Scientific Comput.* **32** (2010), no. 3 1361–1392.

[99] J. R. Stewart and H. C. Edwards, *A framework approach for developing parallel adaptive multiphysics applications*, *Finite Elements in Analysis and Design* **40** (2004), no. 12 1599–1617.

[100] W. Bangerth, R. Hartmann, and G. Kanschat, *Deal.II - a general-purpose object-oriented finite element library*, *ACM Trans. Math. Software* **33** (2007).

[101] C. Burstedde, L. C. Wilcox, and O. Ghattas, *p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees*, *SIAM Journal on Scientific Computing* **33** (2011), no. 3 1103–1133.

[102] I. Tobin, C. Burstedde, and O. Ghattas, *Low-cost parallel algorithms for 2:1 octree balance*, in *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pp. 426–437, 2012.

[103] T. Isaac, C. Burstedde, L. C. Wilcox, and O. Ghattas, *Recursive algorithms for distributed forests of octrees*, *SIAM Journal on Scientific Computing* **37** (2015), no. 5 C497–C531.

[104] J. Rudi, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. Staar, Y. Ineichen, C. Bekas, A. Curioni, and O. Ghattas, *An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth's mantle*, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 5, ACM, 2015.

[105] A. Müller, M. A. Kopera, S. Marras, L. C. Wilcox, T. Isaac, and F. X. Giraldo, "Strong scaling for numerical weather prediction at petascale with the atmospheric model NUMA." http://arxiv.org/abs/1511.01561, 2015.

[106] C. Min and F. Gibou, *A second order accurate level set method on non-graded adaptive Cartesian grids*, *J. Comput. Phys.* **225** (2007) 300–321.

[107] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, *Mathematics of computation* **22** (1968), no. 104 745–762.

[108] D. Brown, R. Cortez, and M. Minion, *Accurate projection methods for the incompressible Navier-Stokes equations*, *J. Comput. Phys.* **168** (2001) 464–499.

[109] R. Courant, E. Isaacson, and M. Rees, *On the solution of nonlinear hyperbolic differential equations by finite differences*, *Comm. Pure Appl. Math.* **5** (1952) 243–255.

[110] I. D. Mishev, *Finite volume methods on voronoi meshes*, *Numerical Methods for Partial Differential Equations* **14** (1998), no. 2 193–212.

[111] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *Petsc web page*, 2012.

[112] S. Balay, J. Brown, , K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc Users Manual*. Argonne National Laboratory, 2012.

[113] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in *Modern Software Tools in Scientific Computing* (B. user Press, ed.), pp. 163—202, 2012.

[114] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler, *Algorithms and data structures for massively parallel generic adaptive finite element codes*, *ACM Transactions on Mathematical Software* **38** (2011), no. 2 14:1–14:28.

[115] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkens-Diehr, *Xsede: Accelerating scientific discovery*, *Computing in Science and Engineering* **16** (2014), no. 5 62–74.

[116] C. Min and F. Gibou, *Geometric integration over irregular domains with application to level set methods*, *J. Comput. Phys.* **226** (2007) 1432–1443.

[117] R. Mittal, H. Dong, M. Bozkurttas, F. Najjar, A. Vargas, and A. von Loebbecke, *A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries*, *Journal of Computational Physics* **227** (2008), no. 10 4825 – 4852.

[118] P. Bagchi, M. Y. Ha, and S. Balachandar, *Direct numerical simulation of flow and heat transfer from a sphere in a uniform cross-flow*, *Journal of Fluids Engineering* **123** (2001), no. 2 347–358.

[119] J. Kim, D. Kim, and H. Choi, *An immersed-boundary finite-volume method for simulations of flow in complex geometries*, *Journal of Computational Physics* **171** (2001), no. 1 132 – 150.

[120] T. A. Johnson and V. C. Patel, *Flow past a sphere up to a reynolds number of 300*, *Journal of Fluid Mechanics* **378** (1, 1999) 19–70.

[121] G. Constantinescu and K. Squires, *Les and des investigations of turbulent flow over a sphere at re = 10,000*, *Flow, Turbulence and Combustion* **70** (2003), no. 1-4 267–298.

[122] J.-I. Choi, R. C. Oberoi, J. R. Edwards, and J. A. Rosati, *An immersed boundary method for complex incompressible flows*, *Journal of Computational Physics* **224** (2007), no. 2 757 – 784.

[123] S. Marella, S. Krishnan, H. Liu, and H. Udaykumar, *Sharp interface Cartesian grid method I: An easily implemented technique for 3D moving boundary computations*, *Journal of Computational Physics* **210** (Nov., 2005) 1–31.

[124] J. Kim and H. Choi, *An immersed-boundary finite-volume method for simulation of heat transfer in complex geometries*, *KSME International Journal* **18** (2004), no. 6 1026–1035.

[125] J. C. Hunt, A. Wray, and P. Moin, *Eddies, stream, and convergence zones in turbulent flows.*, *Center for Turbulence Research Report* **CTR-S88** (1988).

[126] H. Park, *A numerical study of the effects of superhydrophobic surfaces on skin-friction drag reduction in wall-bounded shear flows.* PhD thesis, 2015.

[127] H. Park, H. Park, and J. Kim, *A numerical study of the effects of superhydrophobic surface on skin-friction drag in turbulent channel flow*, *Physics of Fluids* **25** (2013), no. 11 110815, [https://doi.org/10.1063/1.4819144].

[128] F. Temprano-Coleto, S. M. Smith, F. J. Peaudecerf, J. R. Landel, F. Gibou, and P. Luzzatto-Fegiz, *Slip on three-dimensional surfactant-contaminated superhydrophobic gratings*, *arXiv preprint arXiv:2103.16945* (2021).

[129] Z. Li and T. Lu, *Singularities and treatments of elliptic boundary value problems*, *Mathematical and Computer Modelling* **31** (2000), no. 8 97 – 145.

[130] J. R. Philip, *Flows satisfying mixed no-slip and no-shear conditions*, *Zeitschrift für angewandte Mathematik und Physik ZAMP* **23** (1972), no. 3 353–372.

[131] C. J. Teo and B. C. Khoo, *Analysis of Stokes flow in microchannels with superhydrophobic surfaces containing a periodic array of micro-grooves*, *Microfluidics and nanofluidics* **7** (2009), no. 3 353.

[132] J. Jeong and F. Hussain, *On the identification of a vortex*, *Journal of Fluid Mechanics* **285** (1995) 69–94.

[133] A. Rastegari and R. Akhavan, *On the mechanism of turbulent drag reduction with super-hydrophobic surfaces*, *Journal of Fluid Mechanics* **773** (2015) R4.

[134] A. Pathak and M. Raessi, *An implicit, sharp numerical treatment of viscous terms at arbitrarily shaped liquid-gas interfaces in evaporative flows*, *Journal of Computational Physics* **418** (2020) 109625.

[135] R. J. LeVeque and Z. Li, *Immersed interface methods for stokes flow with elastic boundaries or surface tension*, *SIAM Journal on Scientific Computing* **18** (1997), no. 3 709–735.

[136] J. Perot, *An analysis of the fractional step method*, *Journal of Computational Physics* **108** (1993), no. 1 51–58.

[137] M.-J. Ni, S. Komori, and N. Morley, *Projection methods for the calculation of incompressible unsteady flows*, *Numerical Heat Transfer: Part B: Fundamentals* **44** (2003), no. 6 533–551.

[138] Z. Chen and J. Zou, *Finite element methods and their convergence for elliptic and parabolic interface problems*, *Numerische Mathematik* **79** (Apr, 1998) 175–202.

[139] J. H. Bramble and J. T. King, *A finite element method for interface problems in domains with smooth boundaries and interfaces*, *Advances in Computational Mathematics* **6** (Dec, 1996) 109–138.

[140] A. Guittet, M. Lepilliez, S. Tanguy, and F. Gibou, *Solving elliptic problems with discontinuities on irregular domains – the voronoi interface method*, *J. Comp. Phys* **322** (2015) 345–364.

[141] A. Hansbo and P. Hansbo, *An unfitted finite element method, based on nitsches method, for elliptic interface problems*, *Computer Methods in Applied Mechanics and Engineering* **191** (2002), no. 47 5537 – 5552.

[142] D. John and H. Isaac, *An efficient finite element method for embedded interface problems*, *International Journal for Numerical Methods in Engineering* **78** no. 2 229–252, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2486].

[143] T.-P. Fries and T. Belytschko, *The extended/generalized finite element method: An overview of the method and its applications*, *International Journal for Numerical Methods in Engineering* **84** no. 3 253–304, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2914].

[144] M. Nicolas, D. John, and B. Ted, *A finite element method for crack growth without remeshing*, *International Journal for Numerical Methods in Engineering* **46** no. 1 131–150.

[145] R. Gracie, H. Wang, and T. Belytschko, *Blending in the extended finite element method by discontinuous Galerkin and assumed strain methods*, *International Journal for Numerical Methods in Engineering* **74** (2008), no. 11 1645–1669.

[146] R. J. LeVeque and Z. Li, *The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources*, *SIAM Journal on Numerical Analysis* **31** (1994), no. 4 1019–1044, [http://dx.doi.org/10.1137/0731054].

[147] Z. Li and K. Ito, *The Immersed Interface Method – Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*, vol. 33. SIAM Frontiers in Applied mathematics, 2006.

[148] Z. Li and M.-C. Lai, *The immersed interface method for Navier-Stokes equations with singular forces*, *J. Comput. Phys.* **171** (2001) 822–842.

[149] Z. Li, *A note on immersed interface method for three-dimensional elliptic equations*, *Computers & Mathematics with Applications* **31** (1996), no. 3 9 – 17.

[150] J. D. Towers, *Finite difference methods for discretizing singular source terms in a Poisson interface problem*, in *Nonlinear partial differential equations and hyperbolic wave phenomena*, vol. 526 of *Contemp. Math.*, pp. 359–389. Amer. Math. Soc., Providence, RI, 2010.

[151] B. Engquist, A. Tornberg, and R. Tsai, *Discretization of dirac delta functions in level set methods*, *J. Comput. Phys.* **207** (2005) 28–51.

[152] C. Min and F. Gibou, *Robust second order accurate discretizations of the multi-dimensional heaviside and dirac delta functions*, *J. Comput. Phys.* **227** (2008) 9686–9695.

[153] M. Theillard, F. Gibou, and T. Pollock, *A sharp computational method for the simulation of the solidification of binary alloys*, *J. Sci. Comput.* (2014).

[154] D. Bochkov and F. Gibou, *Solving Elliptic Interface Problems with Jump Conditions on Cartesian Grids*, *Journal of Computational Physics* **407** (2020) 109269.

[155] D. Bochkov and F. Gibou, *Solving Poisson-type equations with Robin boundary conditions on piecewise smooth interfaces*, *Journal of Computational Physics* **376** (2019) 1156 – 1198.

[156] A. N. Marques, J.-C. Nave, and R. R. Rosales, *A correction function method for poisson problems with interface jump conditions*, *Journal of Computational Physics* **230** (2011), no. 20 7567 – 7597.

[157] A. N. Marques, J.-C. Nave, and R. R. Rosales, *High order solution of Poisson problems with piecewise constant coefficients and interface jumps*, *Journal of Computational Physics* **335** (2017) 497 – 515.

[158] A. Mayo, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, *SIAM J. Numer. Anal.* **21** (1984) 285–299.

[159] Y. Zhou, S. Zhao, M. Feig, and G. Wei, *High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources*, *Journal of Computational Physics* **213** (2006), no. 1 1 – 30.

[160] M. Cisternino and L. Weynans, *A Parallel Second Order Cartesian Method for Elliptic Interface Problems*, *Communications in Computational Physics* **12** (2012), no. 5 15621587.

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

[162] T. Ménard, S. Tanguy, and A. Berlemont, *Coupling level set/VOF/ghost fluid methods: Validation and application to 3D simulation of the primary break-up of a liquid jet*, *International Journal of Multiphase Flow* **33** (2007), no. 5 510 – 524.

[163] R. Lebas, T. Ménard, P. Beau, A. Berlemont, and F. Demoulin, *Numerical simulation of primary break-up and atomization: DNS and modelling study*, *International Journal of Multiphase Flow* **35** (2009), no. 3 247 – 260.

[164] S. Tanguy and A. Berlemont, *Application of a level set method for simulation of droplet collisions*, *International Journal of Multiphase Flow* **31** (2005), no. 9 1015 – 1035.

[165] X.-D. Liu and T. Sideris, *Convergence of the ghost fluid method for elliptic equations with interfaces*, *Mathematics of computation* **72** (2003), no. 244 1731–1746.

[166] S. Hou and X.-D. Liu, *A numerical method for solving variable coefficient elliptic equation with interfaces*, *Journal of Computational Physics* **202** (2005), no. 2 411 – 445.

[167] G. H. Shortley and R. Weller, *Numerical solution of laplace's equation*, *J. Appl. Phys.* **9** (1938) 334–348.

[168] H. Chen, C. Min, and F. Gibou, *A supra-convergent finite difference scheme for the poisson and heat equations on irregular domains and non-graded adaptive cartesian grids*, *J. Sci. Comput.* **31** (2007) 19–60.

[169] F. Gibou and R. Fedkiw, *A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem*, *Journal of Computational Physics* **202** (2005), no. 2 577 – 601.

[170] F. Gibou, R. P. Fedkiw, L.-T. Cheng, and M. Kang, *A second-order-accurate symmetric discretization of the poisson equation on irregular domains*, *Journal of Computational Physics* **176** (2002), no. 1 205 – 227.

[171] H. Cho, H. Han, B. Lee, Y. Ha, and M. Kang, *A second-order boundary condition capturing method for solving the elliptic interface problems on irregular domains*, *Journal of Scientific Computing* (Jul, 2019).

[172] T. Aslam, *A partial differential equation approach to multidimensional extrapolation*, *J. Comput. Phys.* **193** (2004) 349–355.

[173] Y. T. Ng, H. Chen, C. Min, and F. Gibou, *Guidelines for Poisson Solvers on Irregular Domains with Dirichlet Boundary Conditions Using the Ghost Fluid Method*, *Journal of Scientific Computing* **41** (May, 2009) 300–320.

[174] C. Burstedde, L. C. Wilcox, and O. Ghattas, *p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees*, *SIAM Journal on Scientific Computing* **33** (2011), no. 3 1103–1133.

[175] C. H. Rycroft, *Voro++: A three-dimensional voronoi cell library in c++*, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **19** (2009), no. 4 041111, [https://doi.org/10.1063/1.3215722].

[176] J. S. Hadamard, *Mouvement permanent lent d'une sphère liquide et visqueuse dans un liquide visqueux*, *C. R. Acad. Sci.* **152** (1911) 1735 – 1752.

[177] W. Rybczynski, *Uber die fortschreitende bewegung einer flussigen kugel in einem zahen medium*, *Bull. Acad. Sci. Cracovie A* **1** (1911) 40–46.

[178] N. Young, J. S. Goldstein, and M. J. Block, *The motion of bubbles in a vertical temperature gradient*, *Journal of Fluid Mechanics* **6** (1959), no. 3 350–356.

[179] C. Min, F. Gibou, and H. Ceniceros, *A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids*, *J. Comput. Phys.* **218** (2006) 123–140.

[180] D. Bochkov and F. Gibou, *Solving poisson-type equations with robin boundary conditions on piecewise smooth interfaces*, *Journal of Computational Physics* **376** (2019) 1156–1198.

[181] R. D. Falgout and U. M. Yang, *hypre: A library of high performance preconditioners*, in *International Conference on Computational Science*, pp. 632–641, Springer, 2002.

[182] U. M. Yang, *Parallel algebraic multigrid methodshigh performance preconditioners*, in *Numerical solution of partial differential equations on parallel computers*, pp. 209–236. Springer, 2006.

[183] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, *J. Comput. Phys.* **77** (1988) 439–471.

[184] G. Russo and P. Smereka, *A remark on computing distance functions*, *J. Comput. Phys.* **163** (2000) 51–67.

[185] P. Smereka, *Semi-implicit level set methods for curvature and surface diffusion motion*, *J. Sci. Comput.* **19** (2003) 439 – 456.

[186] P. Macklin and J. Lowengrub, *Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth*, *Journal of Computational Physics* **203** (Feb., 2005) 191–220.

[187] D. Salac and W. Lu, *A local semi-implicit level-set method for interface motion*, *Journal of Scientific Computing* **35** (Jun, 2008) 330–349.

[188] smund Ervik, K. Y. Lervg, and S. T. Munkejord, *A robust method for calculating interface curvature and normal vectors using an extracted local level set*, *Journal of Computational Physics* **257** (2014) 259 – 277.

[189] C. Galusinski and P. Vigneaux, *On stability condition for bifluid flows with surface tension: Application to microfluidics*, *Journal of Computational Physics* **227** (2008), no. 12 6140–6164.

[190] H. Lamb, *Hydrodynamics.* Cambridge Univesity Press, London, 1932.

[191] C. Miller and L. Scriven, *The Oscillations of a Fluid Droplet Immersed in Another Fluid*, *Journal of Fluid Mechanics* **32** (1968) 417–435.

[192] A. Prosperetti *et. al.*, *Normal-mode analysis for the oscillations of a viscous liquid drop in an immiscible liquid*, .

[193] S. Shin, I. Yoon, and D. Juric, *The local front reconstruction method for direct simulation of two- and three-dimensional multiphase flows*, *Journal of Computational Physics* **230** (2011), no. 17 6605–6646.

[194] R. R. Clift, J. R. Grace, and M. E. Weber, *Bubbles, drops, and particles*, 1978.

[195] D. Bhaga and M. E. Weber, *Bubbles in viscous liquids: shapes, wakes and velocities*, *Journal of Fluid Mechanics* **105** (1981) 6185.

[196] J. G. Hnat and J. D. Buckmaster, *Spherical cap bubbles and skirt formation*, *The Physics of Fluids* **19** (1976), no. 2 182–194, [https://aip.scitation.org/doi/pdf/10.1063/1.861445].

[197] J. Delhaye, *Jump conditions and entropy sources in two-phase systems. local instant formulation*, *International Journal of Multiphase Flow* **1** (1974), no. 3 395–409.

[198] M. Ishii and T. Hibiki, *Thermo-fluid dynamics of two-phase flow.* Springer Science & Business Media, 2010.

[199] D. Bochkov and F. Gibou, *Pde-based multidimensional extrapolation of scalar fields over interfaces with kinks and high curvatures*, *SIAM Journal on Scientific Computing* **42** (2020), no. 4 A2344–A2359, [https://doi.org/10.1137/19M1307883].

[200] P. J. Berenson, *Film-Boiling Heat Transfer From a Horizontal Surface*, *Journal of Heat Transfer* **83** (08, 1961) 351–356, [https://asmedigitalcollection.asme.org/heattransfer/article-pdf/83/3/351/5623420/351_1.pdf].

[201] Y.-j. Lao, R. E. Barry, and R. E. Balzhiser, *A study of film boiling on a horizontal plate*, in *International Heat Transfer Conference 4*, vol. 24, Begel House Inc., 1970.

[202] P. McCorquodale, P. Colella, D. Grote, and J.-L. Vay, *A node-centered local refinement algorithm for Poisson's equation in complex geometries*, *J. Comput. Phys.* **201** (2004) 34–60.

[203] A. Guittet, C. Poignard, and F. Gibou, *A voronoi interface approach to cell aggregate electropermeabilization*, *Journal of Computational Physics* **332** (2017) 143 – 159.

[204] J. Papac, A. Helgadottir, C. Ratsch, and F. Gibou, *A level set approach for diffusion and stefan-type problems with robin boundary conditions on quadtree/octree adaptive cartesian grids*, *Journal of Computational Physics* (2012), no. 0 –.

[205] J. Papac, F. Gibou, and C. Ratsch, *Efficient symmetric discretization for the Poisson, heat and Stefan-type problems with Robin boundary conditions*, *Journal of Computational Physics* **229** (Feb., 2010) 875–889.

[206] A. Helgadóttir and F. Gibou, *A Poisson–Boltzmann solver on irregular domains with Neumann or Robin boundary conditions on non-graded adaptive grid*, *Journal of Computational Physics* **230** (May, 2011) 3830–3848.

[207] J. Towers, *Two methods for discretizing a delta function supported on a level set*, *J. Comput. Phys.* **220** (2007) 915–931.

[208] P. Smereka, *The numerical approximation of a delta function with application to level set methods*, *J. Comput. Phys.* **211** (2006) 77–90.

[209] L. G. Leal, *Advanced transport Phenomena - Fluid mechanics and convective transport processes.* Cambridge series in chemical engineering, 2007.

[210] G. Batchelor, *Slender-body theory for particles of arbitrary cross-section in Stokes flow*, *Journal of Fluid Mechanics* **44** (1970), no. 03 419–440.

[211] G. Batchelor, *The stress system in a suspension of force-free particles*, *Journal of fluid mechanics* **41** (1970), no. 03 545–570.

[212] P. Motamarri, M. Nowak, K. Leiter, J. Knap, and V. Gavini, *Higher-order adaptive finite-element methods for Kohn-Sham density functional theory*, *Journal of Computational Physics* **253** (2013) 308 – 343.

[213] P. Suryanarayana, V. Gavini, T. Blesgen, K. Bhattacharya, and M. Ortiz, *Non-periodic finite-element formulation of Kohn–Sham density functional theory*, *Journal of the Mechanics and Physics of Solids* **58** (2010), no. 2 256 – 280.

[214] H. Liu and Z. Wang, *Superposition of multi-valued solutions in high frequency wave dynamics*, *Journal of Scientific Computing* **35** (2008), no. 2 192–218.

[215] X. Wen, *High order numerical methods to three dimensional delta function integrals in level set methods*, *SIAM Journal on Scientific Computing* **32** (2010), no. 3 1288–1309, [http://dx.doi.org/10.1137/090758295].

[216] J. D. Towers, *Discretizing delta functions via finite differences and gradient normalization*, *Journal of Computational Physics* **228** (2009), no. 10 3816 – 3836.

[217] X. Wen, *High order numerical methods to a type of delta function integrals*, *Journal of Computational Physics* **226** (2007), no. 2 1952 – 1967.

[218] G. Yoon and C. Min, *A review of the supra-convergences of Shortley-Weller method for poisson equation*, *J. KSIAM* **18** (2014), no. 1 51–60.

[219] G. Yoon and C. Min, *Analyses on the finite difference method by Gibou et al. for Poisson equation*, *Journal of Computational Physics* **280** (2015) 184 – 194.

[220] G. Yoon and C. Min, *Convergence Analysis of the Standard Central Finite Difference Method for Poisson Equation*, *Journal of Scientific Computing* **67** (2016), no. 2 602–617.

[221] G. F. Lawler and V. Limic, *Random walk: a modern introduction*, vol. 123. Cambridge University Press, 2010.

[222] R. M. Levy, L. Y. Zhang, E. Gallicchio, and A. K. Felts, *On the nonpolar hydration free energy of proteins: surface area and continuum solvent models for the solute-solvent interaction energy*, *Journal of the American Chemical Society* **125** (2003), no. 31 9523–9530.

[223] F. Franks, D. Eagland, and R. Lumry, *The role of solvent interactions in protein conformatio*, *CRC critical reviews in biochemistry* **3** (1975), no. 2 165–219.

[224] B. Roux and T. Simonson, *Implicit solvent models*, *Biophysical Chemistry* **78** (1999), no. 1 1 – 20.

[225] K. A. Sharp and B. Honig, *Calculating total electrostatic energies with the nonlinear Poisson–Boltzmann equation*, *Journal of Physical Chemistry* **94** (1990), no. 19 7684–7692.

[226] M. K. Gilson and B. Honig, *Calculation of the total electrostatic energy of a macromolecular system: solvation energies, binding energies, and conformational analysis*, *Proteins: Structure, Function, and Bioinformatics* **4** (1988), no. 1 7–18.

[227] B. Honig, A. Nicholls, *et. al.*, *Classical electrostatics in biology and chemistry*, *SCIENCE-NEW YORK THEN WASHINGTON-* (1995) 1144–1144.

[228] M. J. Holst and F. Saied, *Numerical solution of the nonlinear Poisson–Boltzmann equation: developing more robust and efficient methods*, *Journal of computational chemistry* **16** (1995), no. 3 337–364.

[229] M. Holst, N. Baker, and F. Wang, *Adaptive multilevel finite element solution of the Poisson–Boltzmann equation I. Algorithms and examples*, *Journal of computational chemistry* **21** (2000), no. 15 1319–1342.

[230] N. Baker, M. Holst, and F. Wang, *Adaptive multilevel finite element solution of the Poisson–Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems*, *Journal of computational chemistry* **21** (2000), no. 15 1343–1352.

[231] L. Li, C. Li, S. Sarkar, J. Zhang, S. Witham, Z. Zhang, L. Wang, N. Smith, M. Petukh, and E. Alexov, *DelPhi: a comprehensive suite for DelPhi software and associated resources*, *BMC Biophysics* **5** (May, 2012) 9.

[232] A. Nicholls and B. Honig, *A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson–Boltzmann equation*, *Journal of computational chemistry* **12** (1991), no. 4 435–445.

[233] D. Bashford, *An object-oriented programming suite for electrostatic effects in biological molecules an experience report on the MEAD project*, in *Scientific Computing in Object-Oriented Parallel Environments* (Y. Ishikawa, R. R. Oldehoeft, J. V. W. Reynders, and M. Tholburn, eds.), (Berlin, Heidelberg), pp. 233–240, Springer Berlin Heidelberg, 1997.

[234] M. E. Davis and J. A. McCammon, *Solving the finite difference linearized Poisson–Boltzmann equation: A comparison of relaxation and conjugate gradient methods*, *Journal of Computational Chemistry* **10** (1989), no. 3 386–391.

[235] W. Im, D. Beglov, and B. Roux, *Continuum solvation model: Computation of electrostatic forces from numerical solutions to the Poisson–Boltzmann equation*, *Computer Physics Communications* **111** (1998), no. 1 59 – 75.

[236] Y. Zhong, K. Ren, and R. Tsai, *An implicit boundary integral method for computing electric potential of macromolecules in solvent*, *Journal of Computational Physics* **359** (2018) 199 – 215.

[237] A. J. Bordner and G. A. Huber, *Boundary element solution of the linear Poisson–Boltzmann equation and a multipole method for the rapid calculation of forces on macromolecules in solution*, *Journal of Computational Chemistry* **24** (2003), no. 3 353–367.

[238] A. H. Boschitsch and M. O. Fenley, *Hybrid boundary element and finite difference method for solving the nonlinear Poisson–Boltzmann equation*, *Journal of Computational Chemistry* **25** (2004), no. 7 935–955.

[239] N. A. Baker, D. Bashford, and D. A. Case, *Implicit Solvent Electrostatics in Biomolecular Simulation*, pp. 263–295. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[240] B. Lu, Y. Zhou, M. Holst, and J. McCammon, *Recent progress in numerical methods for the Poisson–Boltzmann equation in biophysical applications*, *Commun Comput Phys* **3** (2008), no. 5 973–1009.

[241] W. E. Lorensen and H. E. Cline, *Marching cubes: A high resolution 3D surface construction algorithm*, *SIGGRAPH Comput. Graph.* **21** (Aug., 1987) 163–169.

[242] W. Rocchia, S. Sridharan, A. Nicholls, E. Alexov, A. Chiabrera, and B. Honig, *Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: Applications to the molecular systems and geometric objects*, *Journal of computational chemistry* **23** (2002), no. 1 128–137.

[243] Z. Chen, N. A. Baker, and G. Wei, *Differential geometry based solvation model I: Eulerian formulation*, *Journal of Computational Physics* **229** (2010), no. 22 8231 – 8258.

[244] D. Chen, Z. Chen, C. Chen, W. Geng, and G.-W. Wei, *MIBPB: A software package for electrostatic analysis*, *Journal of Computational Chemistry* **32** (2011), no. 4 756–770.

[245] M. Mirzadeh, M. Theillard, A. Helgadottir, D. Boy, and F. Gibou, *An adaptive, finite difference solver for the nonlinear Poisson–Boltzmann equation with applications to biomolecular computations*, *Communications in Computational Physics* **13** (2012), no. 1 150–173.

[246] M. Mirzadeh, M. Theillard, and F. Gibou, *A Second-Order Discretization of the Nonlinear Poisson-Boltzmann Equation over Irregular Geometries using Non-Graded Adaptive Cartesian Grids*, *Journal of Computational Physics* **230** (Dec., 2010) 2125–2140.

[247] F. Gibou, R. Fedkiw, and S. Osher, *A review of level-set methods and some recent applications*, *Journal of Computational Physics* **353** (2018), no. Supplement C 82 – 109.

[248] M. F. Sanner, A. J. Olson, and J.-C. Spehner, *Reduced surface: an efficient way to compute molecular surfaces*, *Biopolymers* **38** (1996), no. 3 305 – 320.

[249] T. Can, C.-I. Chen, and Y.-F. Wang, *Efficient molecular surface generation using level-set methods*, *Journal of Molecular Graphics and Modelling* **25** (2006), no. 4 442 – 454.

[250] D. Xu and Y. Zhang, *Generating triangulated macromolecular surfaces by euclidean distance transform*, *PLOS ONE* **4** (12, 2009) 1–11.

[251] C. M. Cortis and R. A. Friesner, *An automatic three-dimensional finite element mesh generation system for the Poisson–Boltzmann equation*, *Journal of Computational Chemistry* **18** (1997), no. 13 1570–1590.

[252] Z. Yu, M. J. Holst, Y. Cheng, and J. McCammon, *Feature-preserving adaptive mesh generation for molecular shape modeling and simulation*, *Journal of Molecular Graphics and Modelling* **26** (2008), no. 8 1370 – 1380.

[253] R. E. Bank and M. Holst, *A new paradigm for parallel adaptive meshing algorithms*, *SIAM Journal on Scientific Computing* **22** (2000), no. 4 1411–1443.

[254] N. A. Baker, D. Sept, M. J. Holst, and J. A. McCammon, *The adaptive multilevel finite element solution of the Poisson–Boltzmann equation on massively parallel computers*, *IBM Journal of Research and Development* **45** (2001), no. 3.4 427–438.

[255] A. du Chéné, C. Min, and F. Gibou, *Second Order Accurate Computation of Interface Curvature in a Level Set Framework Using Novel High-Order Reinitialization Schemes*, *Journal of Scientific Computing* **35** (2008) 114–131.

[256] M. Mirzadeh, A. Guittet, C. Burstedde, and F. Gibou, *Parallel level-set methods on adaptive tree-based grids*, *J. Comp. Phys.* **322** (2016) 345–364.

[257] S. Mitternacht, *FreeSASA: An open source C library for solvent accessible surface area calculations*, *F1000Research* **5** (2016), no. 189.

[258] T. Goddard, "Solvent accessible surface area." `https://www.cgl.ucsf.edu/chimera/data/sasa-nov2013/sasa.html`, nov, 2013. Accessed: 2018-07-12.

[259] T. J. Dolinsky, J. E. Nielsen, J. A. McCammon, and N. A. Baker, *PDB2PQR: an automated pipeline for the setup of Poisson–Boltzmann electrostatics calculations*, *Nucleic acids research* **32** (2004), no. suppl_2 W665–W667.

[260] B. Lee and F. Richards, *The interpretation of protein structures: Estimation of static accessibility*, *Journal of Molecular Biology* **55** (1971), no. 3 379 – IN4.

[261] F. M. Richards, *Areas, volumes, packing, and protein structure*, *Annual review of biophysics and bioengineering* **6** (1977), no. 1 151–176.

[262] J. Greer and B. L. Bush, *Macromolecular shape and surface maps by solvent exclusion*, *Proceedings of the National Academy of Sciences* **75** (1978), no. 1 303–307.

[263] T. You and D. Bashford, *An analytical algorithm for the rapid determination of the solvent accessibility of points in a three-dimensional lattice around a solute molecule*, *Journal of Computational Chemistry* **16** (1995), no. 6 743–757.

[264] J. L. Bentley, D. F. Stanat, and E. Williams, *The complexity of finding fixed-radius near neighbors*, *Information Processing Letters* **6** (1977), no. 6 209 – 212.

[265] J. L. Bentley, *A survey of techniques for fixed radius near neighbor searching.*, tech. rep., Stanford, CA, USA, 1975.

[266] J. Barnes and P. Hut, *A hierarchical O(N log N) force-calculation algorithm*, *Nature* **324** (1986), no. 6096 446–449.

[267] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes II (two)*, *J. Comput. Phys.* **83** (1989) 32–78.

[268] H. Zhao, *Parallel implementations of the fast sweeping method*, *Journal of Computational Mathematics* **25** (2007) 421–429.

[269] M. Detrixhe, F. Gibou, and C. Min, *A Parallel Fast Sweeping Method for the Eikonal Equation*, *Journal of Computational Physics* **237** (Mar., 2013) 46–55.

[270] C.-Y. Kao, S. Osher, and Y.-H. Tsai, *Fast sweeping methods for static Hamilton–Jacobi equations*, *SIAM Journal on Numerical Analysis* **42** (2005), no. 6 2612–2632, [https://doi.org/10.1137/S0036142902419600].

[271] M. Detrixhe and F. Gibou, *Hybrid massively parallel fast sweeping method for static Hamilton-Jacobi equations*, *J. Comp. Phys.* **322** (2016) 199–223.

[272] J. Tsitsiklis, *Efficient Algorithms for Globally Optimal Trajectories*, *IEEE Trans. on Automatic Control* **40** (1995) 1528–1538.

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

[274] D. Chopp, *Some improvements of the fast marching method*, *SIAM J. Sci. Comput.* **223** (2001) 230–244.

[275] J. A. Sethian and A. Vladimirsky, *Ordered upwind methods for static hamilton–jacobi equations: Theory and algorithms*, *SIAM Journal on Numerical Analysis* **41** (2003), no. 1 325–363, [https://doi.org/10.1137/S0036142901392742].

[276] J. A. Sethian, *Level set methods and fast marching methods.* Cambridge University Press, 1999. Cambridge.

[277] M. Detrixhe, *Parallel fast sweeping : algorithms and applications.* PhD thesis, University of California, Santa Barbara, 2016.

[278] M. L. Connolly, *Analytical molecular surface calculation*, *Journal of Applied Crystallography* **16** (1983), no. 5 548–558.

[279] G. M. Morton, *A computer oriented geodetic data base and a new technique in file sequencing.* International Business Machines Company New York, 1966.

[280] P. Mistani, A. Guittet, D. Bochkov, J. Schneider, D. Margetis, C. Ratsch, and F. Gibou, *the island dynamics model on parallel quadtree grids*, *J. Comp. Phys. (in preparation)* (2017).

[281] R. Caflisch, M. Gyure, B. Merriman, S. Osher, C. Ratsch, D. Vvedensky, and J. Zinck, *Island dynamics and the level set method for epitaxial growth*, *Applied Mathematics Letters* **12** (1999) 13.

[282] S. Chen, B. Merriman, M. Kang, R. E. Caflisch, C. Ratsch, L.-T. Cheng, M. Gyure, R. P. Fedkiw, C. Anderson, and S. Osher, *A level set method for thin film epitaxial growth*, *Journal of Computational Physics* **167** (2001), no. 2 475 – 500.

[283] C. Ratsch, M. F. Gyure, R. E. Caflisch, F. Gibou, M. Petersen, M. Kang, J. Garcia, and D. D. Vvedensky, *Level-set method for island dynamics in epitaxial growth*, *Phys. Rev. B* **65** (2002) 195403.

[284] G. K. Batchelor, *An introduction to fluid dynamics.* Cambridge university press, 2000.

[285] G. G. Stokes, *On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids*, *Transactions of the Cambridge Philosophical Society* **8** (1880).

[286] L. N. Liebermann, *The Second Viscosity of Liquids*, *Phys. Rev.* **75** (May, 1949) 1415–1422.

[287] L.D. Landau and E.M. Lifshitz, *Chapter II - Viscous Fluids*, in *Fluid Mechanics*, pp. 44 – 94. Pergamon, second ed., 1987.

[288] W. M. Deen, *Analysis of transport phenomena.* Oxford University Press New York, 1998.

[289] C. Weatherburn, *Differential geometry, vol. i*, *I (Cambridge, 1927)* (1927).

[290] R. Caiden, R. P. Fedkiw, and C. Anderson, *A numerical method for two-phase flow consisting of separate compressible and incompressible regions*, *Journal of Computational Physics* **166** (2001), no. 1 1 – 27.