UNIVERSITY of CALIFORNIA
Santa Barbara

Computational Methods for Solidification of Multicomponent Materials and Moving Boundary Problems in Physics of Inhomogeneous Polymers

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

Doctor of Philosophy

in

Mechanical Engineering

by

Daniil Bochkov

Committee in charge:

Professor Frederic Gibou, Chair

Professor Sumita Pennathur

Professor Paolo Luzzatto-Fegiz

Professor Glenn H. Fredrickson

September 2020

The dissertation of Daniil Bochkov is approved.

Professor Glenn H. Fredrickson

Professor Paolo Luzzatto-Fegiz

Professor Sumita Pennathur

Professor Frederic Gibou, Committee Chair

August 2020

To my family and friends

Acknowledgements

First of all, I would like thank my PhD advisor Professor Frederic Gibou. While giving me immense freedom in my research he also provided plenty of guidance throughout my PhD program. Working under his supervision was always extremely easy and pleasant. His care for students' career successes as well as their well-being outside of professional setting is greatly appreciated.

I would also like to thank Professor Sumita Pennathur, Professor Paolo Luzzatto-Fegiz, and Professor Glenn Fredrickson for serving on my doctoral committee. They provided highly valuable feedback from different perspectives.

I would like to thank all teachers and professors I have met throughout my educational journey and from whom I have had a privilege to learn. Especially, I would like to thank my high school teachers of mathematics and physics Natalia Vladimirovna Melnichenko, Lyubov Yurievna Scherbinina, and Enikeev Dmitry Vladilenovich from Licey 153. I am certain that the fundamental problem solving skills that they had developed in me have been absolutely crucial for my successes at all stages of my research career.

I would like to thank all members and visitors of the Computational Applied Science Laboratory with whom I had a pleasure and honor to work: Arthur, Elyce, Fernando, Gaddiel, Gina, Ivana, Maxime, Miles, Mohammad, Pouria, Raphael, Rochishnu, Samira, Victoria. Our discussions about science and research have been extremely enriching and our social gatherings have been a highlight of my stay in Santa Barbara.

I would like to say a special thank you to my family. I owe absolutely all of my accomplishments to them. I greatly appreciate their support, encouragement, and advice.

A special thank you to Carol for her unceasing love, care, and support.

Last but not least, I would like to thank all of my friends and acquaintances, the ones I have known for a long time and the ones I have met during my stay in Santa Barbara: Adam, Aditya, Alexey, Amir, Andrew, Anirban, Aviral, Axel, Aydar, Brandon, Dan,

David, Emile, Eugene, Igor, Jacob, Jess, Jin, John, Justin, Katharine, Keran, Krishnan, Kristen, Lanhe, Leslie, Luke, Manik, Michael, Neeraj, Nhan, Nicholas, Nikita S., Nikita R., Nikolai, Sarah, Savya, Sergey, Shannon, Shriniwas, Stefany, Tanya, Thomas, Tuan, Vasily, Vu, William, Yongchao, Zack, and others. I feel very lucky to know them all and value very much their influence on my life.

Curriculum Vitæ

Daniil Bochkov

Education

2020	Ph.D. in Mechanical Engineering, University of California, Santa Barbara, California, United States of America
2014	M.S. in Applied Mathematics and Physics, Moscow Institute of Physics and Technology, Moscow, Russia
2012	B.S. in Applied Mathematics and Physics, Moscow Institute of Physics and Technology, Moscow, Russia

Publications

Gaddiel Ouaknin, Nabil Laachi, **Daniil Bochkov**, Kris Delaney, Glenn H. Fredrickson, Frederic Gibou, "Functional Level-Set Derivative for a Polymer Self Consistent Field Theory Hamiltonian," *Journal of Computational Physics*, Volume 345, Pages 207-223, 2017.

Pouria Mistani, Arthur Guittet, **Daniil Bochkov**, Joshua Schneider, Dionisios Margetis, Christian Ratsch, Frederic Gibou, "The Island Dynamics Model on Parallel Quadtree Grids," *Journal of Computational Physics*, Volume 361, Pages 150-166, 2018.

Victoria Arias, **Daniil Bochkov**, Frederic Gibou, "Poisson Equations in Irregular Domains with Robin Boundary Conditions - Solver with Second-Order Accurate Gradients," *Journal of Computational Physics*, Volume 365, Pages 1-6, 2018.

Daniil Bochkov, Frederic Gibou, "Solving Poisson-type equations with Robin boundary conditions on piecewise smooth interfaces," *Journal of Computational Physics*, Volume 376, Pages 1156-1198, 2019.

Daniil Bochkov, Frederic Gibou, "Solving Elliptic Interface Problems with Jump Conditions on Cartesian Grids," *Journal of Computational Physics*, Volume 407, Article 109269, 2020.

Daniil Bochkov, Frederic Gibou, "PDE-Based Multidimensional Extrapolation of Scalar Fields Over Interfaces With Kinks and High Curvatures," *SIAM Journal of Scientific Computing*, Volume 42, Number 4, Pages A2344-A2359.

Daniil Bochkov, Tresa Pollock, Frederic Gibou, "Sharp-Interface Simulations of Multialloy Solidification," *In preparation.*

Daniil Bochkov, Gaddiel Ouaknin, Frederic Gibou, "Computational Framework for Simulation of Free-Surface Block Copolymers," *In preparation.*

Daniil Bochkov, Frederic Gibou, "An Adjoint State Method for Inverse Design Problem in Directed Self-Assembly of Block Copolymers," *In preparation.*

Daniil Bochkov, Ivana Bagaric, Frederic Gibou, "A Versatile Sharp-Interface Particle-Field Computational Method for Co-assembly of Block Copolymer Nanocomposites," *In preparation.*

Rochishnu Chowdhury, Raphael Egan, **Daniil Bochkov**, Frederic Gibou, "Efficient Calculation of Fully Resolved Electrostatics around Large Biomolecules," *In preparation.*

Asdis Helgadottir, **Daniil Bochkov**, Frederic Gibou, "A Hybrid Finite Difference/Finite Volume Approach on Adaptive Octree Grids for Biofilm Growth," *In preparation.*

Abstract

Computational Methods for Solidification of Multicomponent Materials and Moving Boundary Problems in Physics of Inhomogeneous Polymers

by

Daniil Bochkov

This dissertation focuses on the development of numerical methods for the solidification of multicomponent alloys and a number of moving boundary problems in physics of block copolymer (BCP) materials: the self-assembly of free surface BCP melts, the co-assembly of BCP nanocomposites, and the inverse design problem for the Directed Self-Assembly (DSA). While these processes have different physical nature, they share several computational challenges: the nonlinearity of the governing equations, the diffusion-dominated character, and the presence of moving boundaries/interfaces that are themselves part of the solution. Accurate and efficient simulations of these phenomena require advanced fundamental numerical capabilities as well as solving conceptual challenges specific to each application.

The first part of this work describes the general numerical methods for solving partial differential equations (PDEs) that are necessary for the simulation of the considered processes and which, at the same time, have significance beyond these applications. Specifically, we present novel second-order accurate finite-volume discretizations on Cartesian

grids for Poisson-type equations in irregular domains subject to Robin boundary conditions and/or with discontinuities across immersed interfaces. In addition, we provide a PDE-based approach for smooth extensions of scalar fields across piecewise smooth interfaces.

In the second part, we present a computational approach for the simulation of multialloy solidification in the sharp-interface limit. The main challenge – solving a non-linearly coupled system of PDEs at each time step – is solved by a novel Newton-type approach. Further, a combination of adaptive Cartesian quadtree grids and the Level-Set Method is used to address the highly complex evolution of the solidification front and the multiscale nature of the process. The proposed method is validated on cases with known analytical solutions and applied to study the segregation behavior of a Co-Al-W ternary alloy.

Finally, the third part considers the moving boundary problems related to the self-assembly of BCPs. First, we augment the Self-Consistent Field Theory with a consistent approach for imposing surface energies, which is crucial for accurate modeling of polymer-air and polymer-wall interactions. Second, the machinery of PDE-constrained shape sensitivity analysis is applied to derive the equations governing the free surface shape of BCP melts and the placement of nanoparticles as well as the shape derivative of the cost functional associated with the inverse design problem for DSA. The obtained numerical methods are then used to study the self-assembling behavior of substrate supported BCP droplets, to investigate the co-assembly of BCP and nanoparticles of complex shapes, and to design confining masks for nanolithography applications.

Contents

Permissions and Attributions

1. The content of chapter 1 is the result of a collaboration with Frederic Gibou, and has previously appeared in the Journal of Computational Physics as "Solving Poisson-type equations with Robin boundary conditions on piecewise smooth interfaces" [18]. It is reproduced here in accordance with the publisher's copyright policy allowing authors to include their articles in a thesis or dissertation, provided that this is not to be published commercially: https://www.elsevier.com/about/policies/copyright#Author-rights.

2. The content of chapter 2 is the result of a collaboration with Frederic Gibou, and has previously appeared in the Journal of Computational Physics as "Solving elliptic interface problems with jump conditions on Cartesian grids" [19]. It is reproduced here in accordance with the publisher's copyright policy allowing authors to include their articles in a thesis or dissertation, provided that this is not to be published commercially: https://www.elsevier.com/about/policies/copyright#Author-rights.

3. The content of chapter 3 is the result of a collaboration with Frederic Gibou, and has

previously appeared in the SIAM Journal on Scientific Computing as "PDE-Based Multidimensional Extrapolation of Scalar Fields over Interfaces with Kinks and High Curvatures" [16]. It is reproduced here with the permission of the publisher.

Chapter 1

Solving Elliptic PDEs with Robin Boundary Conditions in Domains with Piecewise Boundaries

1.1 Introduction

Poisson-type equations in irregular domains are one of the core fundamental models in science and engineering, whether as standalones or as part of more complex models. Since most of the models of modern science and engineering involve an irregular boundary that evolves in time, finding a numerical solution in the context of the level-set method is particularly desirable because it provides a framework that automatically handles changes in topology. Numerical methods for applying boundary conditions such as Dirichlet

[58, 56, 29, 147], Robin [128] or jump conditions [95, 66] in this 'capturing interface' context have been developed in the level-set community in the case of smooth boundaries (see also the reviews [61, 59]) and have been applied to a wide variety of problems [129, 123, 124, 125, 28, 44, 55, 57, 68, 90, 96, 104, 116, 136, 158, 159, 38]

However, the modeling of certain important physical phenomena require that different Robin (mixed) boundary conditions be imposed on different parts of the boundary of irregular domains; i.e. an irregular boundary that may present kinks. Examples of models with significant scientific and industrial applications are (i) the Robin boundary conditions that model the repulsive/attractive air-polymer interfaces in the context of the self-consistent field theory for block-copolymers [47], (ii) the Robin boundary conditions that model the solute-rejection equations that is at the center of dendritic solidification of multicomponent metal alloys [159], and (iii) the Robin boundary conditions that model the Ehrlich-Schwoebel step-edge energy barrier used in the simulation of epitaxial growth [170, 128, 127]. In those problems, the presence of confining solid boundaries add to the complexity in that surface tension forces lead to sharp contact angles between different phases (e.g. the air-polymer-substrate in the context of polymer droplets on a substrate or the liquid-solid-wall in the context of confined dendritic growth). To simulate these physical processes, one must consider the solution of a diffusion-dominated equation in domains with piecewise smooth boundaries.

We thus focus on the following problem: consider a domain Ω with a piecewise smooth (C^0) boundary $\partial\Omega$ that consists of n_ϕ smooth (C^2) components $\Gamma_1, \ldots, \Gamma_{n_\phi}$, that is,

$\partial\Omega = \Gamma_1 \cup \ldots \cup \Gamma_{n_\phi}$. We are interested in solving a Poisson-type equation of the form:

$$-\nabla(\mu(\boldsymbol{r})\nabla u(\boldsymbol{r})) + k(\boldsymbol{r})u(\boldsymbol{r}) = f(\boldsymbol{r}) \quad \text{in } \Omega, \tag{1.1}$$

where $\mu(\boldsymbol{r})$ is the diffusion coefficient bounded below by some positive constant, $k(\boldsymbol{r})$ and $f(\boldsymbol{r})$ are given functions, and where the function $u(\boldsymbol{r})$ satisfies Robin boundary conditions on each of the smooth parts $\Gamma_1, \ldots, \Gamma_{n_\phi}$ of the domain boundary:

$$\mu(\boldsymbol{r})\frac{\partial u(\boldsymbol{r})}{\partial \boldsymbol{n}_p} + \alpha_p(\boldsymbol{r})u(\boldsymbol{r}) = g_p(\boldsymbol{r}) \quad \text{on } \Gamma_p, \quad p = 1, \ldots, n_\phi. \tag{1.2}$$

We present two finite volume numerical schemes. The first method produces a symmetric linear system, solutions that are second-order accurate in the L^∞-norm and gradients that are first-order accurate in the L^∞-norm. The second method leads to a nonsymmetric linear system but produces solutions and gradients that are both second-order accurate in the L^∞-norm. We note, however, that these optimal convergence rates are achieved for solutions that are sufficiently smooth. It is well-known that sharp features in the domain's boundary may lead to solutions with a low regularity. Throughout the derivation of the numerical methods and their analysis it is assumed that smoothness requirements for the solution (not the domain, which may have kinks) are satisfied and the effect of violating such requirements is studied numerically (see section 1.5.5).

The rest of this chapter is organized as follows. In Section 1.2, we review relevant prior works. In Section 1.3, we describe the class of domains we consider in this work. In Section 1.4 we present the two numerical schemes for solving (1.1)-(1.2). In Section 1.5, we present numerical results that illustrate the accuracy of the two schemes. Section 1.6

concludes the chapter. Finally, in 1.A, we provide a detailed description of the integration method that we use.

1.2 Relevant Literature

We consider a Eulerian approach to avoid the problems related to mesh generation that requires conforming the elements near evolving boundaries [137, 140, 143, 126, 138, 182, 12] or explicit representations that rely on complex schemes to handle changes in topology [130, 63, 15, 62, 83, 166, 164, 70]. By far the most successful and popular method for describing irregular domains on fixed grids is the level-set method [122, 145, 121, 59]. In this approach the boundary of an irregular domain is implicitly defined by an iso-contour of a Lipschitz-continuous function and the motion of the domain is described by a simple (possibly nonlinear) advection equation. Due to its implicit character, the level-set approach naturally handles such challenging (especially in three dimensions) problems as merging and breaking up of domains. It has been successfully applied to a wide variety of computational problems [129, 123, 124, 125, 28, 44, 55, 57, 68, 90, 96, 104, 116, 136, 158, 159, 38]. Moreover, the method can be also applied to describe domains with sharp features using the idea that certain domains with piecewise smooth boundaries can be represented as intersections/unions of domains with smooth boundaries [113, 78, 151, 152, 153, 90].

The convenience of Eulerian methods comes with the need to develop special methods for imposing boundary conditions on irregular domains. Few studies have addressed

Robin boundary conditions in this context. In Greenspan [64], two finite difference methods, based on linear or quadratic approximations of normal derivatives and the Shortley-Weller operator, are proposed for solving elliptic equations in irregular domains with mixed boundary conditions in two spatial dimensions. In the case of the linear approximation, the resulting linear system is non-singular and numerical solutions converge to exact solutions with first-order accuracy in the L^∞ norm. The scheme based on quadratic approximation is also derived using geometric considerations in Jomaa and Macaskill [82] and an extension to three spatial dimensions is presented by the same authors in [81]. In both [82] and [81], the convergence of those schemes is analyzed theoretically in one spatial dimension and studied numerically in higher dimensions. It is shown that linear and quadratic interpolations of the solution leads to first- and second-order accuracy in the L^∞ norm, respectively. Accuracy of gradients are not studied.

In Bramble and Hubbard [22], a finite difference method is proposed for solving the Poisson equation with mixed boundary conditions. This work focuses on the development of a scheme that results in a "positive type" linear system. The authors proved that the numerical solution converges to its exact solution with second-order accuracy in the L^∞ norm if the exact solution is sufficiently smooth. Although the authors proved that it is always possible to choose three internal points for discretization of boundary conditions such that the resulting linear system is of "positive type", no simple and clear algorithm is presented on how to select such points. Another difficulty of this scheme is that it requires the computation of the tangential derivatives of boundary data, a task that may

be challenging. Numerical examples are not provided and accuracy of gradients are not studied.

In Van Linde [168], higher-order finite difference schemes are proposed to solve the Poisson equation with Dirichlet, Neumann and Robin boundary conditions. These schemes use four internal points to discretize boundary conditions, a special third-order accurate discretization of the Laplace operator near the irregular boundary that depends on the particular configuration of the boundary and first and second tangential derivatives of the boundary conditions data. The author proved that the schemes are of "positive type" and produce third-order-accurate numerical solutions in the L^∞ norm for the Neumann and Robin problems and fourth-order-accurate solutions in the L^∞ norm for the Dirichlet problem.

An alternative second-order accurate finite difference scheme for Neumann and Robin problems that does not use tangential derivatives is proposed in Bouchon and Peichl [21]. This is achieved by using four internal points for discretizing the boundary conditions, which are selected so that the resulting linear system is represented by an M-matrix. However, such points can be as far as 11 grid-spacings away from a point of consideration, limiting the applicability of such an approach. Second-order accuracy in the L^∞ norm is proved analytically and demonstrated numerically.

In Papac *et al.* [128], a simple second-order accurate finite volume scheme is presented for the Robin problem. This method uses only direct neighbors in the Cartesian directions and results in a symmetric positive definite linear system. The Robin boundary

condition requires an integration over irregular domains and their boundaries, which is performed with the geometric integration of Min and Gibou [99]. Later, the method of [128] was extended to three spatial dimensions and to adaptive Octree grids and applied to Stefan-type problems [129]. In Arias *et al.* [5], a simple numerical example on a two-dimensional irregular smooth domain illustrated that second-order accurate solutions and second-order accurate gradients can be obtained if the method of [128] is modified in two important ways: (1) the fluxes between cells are discretized using ideas of Johansen and Colella [80], and (2) a Taylor expansion of the unknown function is used to approximate the boundary condition *at* the boundary.

Another important work in the development of finite volume methods is the work of [41] that presents a fourth-order accurate finite volume method for the Poisson equation with Neumann and Dirichlet boundary conditions (Robin boundary conditions are not considered). The higher accuracy is achieved by constructing rather wide stencils for each grid node independently using a least-squares approach. Piece-wise smooth interfaces are considered, however such cases are dealt with by some regularization/smoothing of the problem's geometry. It is not clear whether some modification of boundary conditions is needed for such a treatment. It is shown that the choice of the smoothing length affects the convergence behavior.

Coco and Russo [33] presented a finite difference ghost-point approach for discretizing Dirichlet, Neumann and Robin boundary conditions in two spatial dimensions, based on a quadratic interpolation of the unknown function near the domain's boundary. This

approach produces second-order accurate numerical solutions and second-order accurate gradients and is applicable to domains with sharp features as well.

In Gallinato and Poignard [53, 54], second-order accurate finite difference superconvergent schemes, i.e. schemes producing derivatives of the numerical solution of the same accuracy as the numerical solutions themselves, are presented for Dirichlet, Neumann and Robin boundary conditions in two spatial dimensions. These schemes are based on the ghost-fluid method of Gibou *et al.* [58], i.e. the standard 5-point stencil is used everywhere and values of the unknown function at points that fall outside of the solution's domain are extrapolated from internal points and the given Dirichlet boundary condition. To produce different levels of superconvergence, linear, quadratic and cubic extrapolations are considered, as in [56]. In cases of Neumann and Robin boundary conditions, where the value of the unknown function is not immediately available at the domain's boundary, the boundary value is obtained from bi-linear, bi-quadratic or bi-cubic (depending on the desired level of superconvergence) interpolation from internal points. It is demonstrated that to obtain second-order accurate gradients (first level of superconvergence), quadratic extrapolations and interpolations are required, while cubic extrapolations/interpolation produce second-order accurate second order derivatives as well (second level of superconvergence).

Obtaining accurate numerical derivatives is important in problems where the solution process is driven by the accuracy of the solution's gradient. This is the case for example of the Stefan problem, which is the building block for simulating processes like the

solidification of multicomponent alloys or the islands growth in the context of molecular beam epitaxy. Among the aforementioned works only [33] demonstrated an immediate applicability to domains with piecewise smooth boundaries. An extension to three spatial dimensions seems to be rather straightforward, but could be limited by the fact that the stencil near the domain's boundary needs to be adjusted in region of high curvature in such a way that it contains only points that are internal or ghost points. Although possible, this requirement makes the implementation less straightforward and the requirement that far-neighbor grid points are involved in the stencil may lower the efficiency of parallelism. The other finite difference methods [64, 82, 81, 22, 168, 21, 53, 54] have similar drawbacks, since all of them require a certain number of strictly internal grid points to approximate boundary conditions. It should also be noted that none of the methods [64, 82, 81, 22, 168, 21, 53, 54] results in a symmetric linear system.

Contrary to finite difference methods, finite volume methods [128, 5] use only immediate neighboring points to approximate the equation and the structure of the stencil stays the same even in the regions of high curvature. In these frameworks, the complexity is shifted from constructing stencils to that of integrating over implicitly defined domains and their boundaries. Clearly, the task of developing a finite volume method for geometries with piecewise smooth boundaries requires the development of an integration method over such geometries as well.

In the level-set community, there exists several integration methods, which can be divided into the following categories: (1) methods based on the discretization of the

Heaviside step-function and Dirac delta-function [150, 43, 161, 180, 163, 162, 101]; (2) methods based on the reduction of the integral dimensionality using the divergence theorem [114, 142]; (3) dimension-by-dimension integration approaches [139, 173, 172, 171]; and (4) geometric integration approaches [99, 50, 51]. Although some of the aforementioned methods have been extended to high-order accuracy, most of them are designed only for smooth geometries. It appears that only the geometric reconstruction methods admit an immediate extension to domains with piecewise smooth boundaries. Such an extension was used in [113] in conjunction with linear geometric reconstruction and discussed in [51] for higher-order reconstructions. In [113] the proposed integration method was used within the XFEM framework, however the accuracy of the integration method itself was not investigated. Higher order methods of [50, 51] has been mainly used as a basis for conformal grid generation [48, 52, 119].

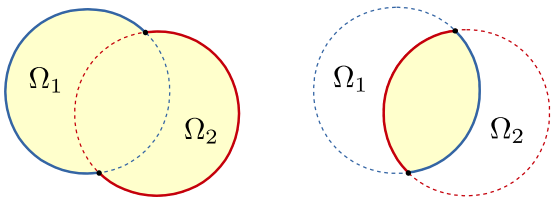
The two schemes we present build and improve on results of [128] and [5]. In particular, the symmetric scheme presented in this work is an improvement of the scheme of [128] in two aspects: (1) Robin boundary conditions are discretized more accurately for smooth boundaries; (2) it considers domains with sharp features - a special treatment is proposed for cells with sharp features and an integration method specifically developed for piecewise smooth domains is used. The more accurate (nonsymmetric) scheme improves on the approach of [5] in that (1) it considers domains with piecewise smooth boundaries and (2) discretization in three spatial dimensions are developed. As it is the case of [5], the nonsymmetric discretization uses ideas from [80, 141] for discretizing the

fluxes across cells boundaries. We note however that [80, 141] considered neither Robin boundary conditions nor domains with sharp features, while in the present work consider both.

As it was mentioned above, integration plays a crucial parts of finite volume methods. We employ a geometric reconstruction approach and propose a hierarchical algorithm for its implementation. This approach is in large part based on the methods and results of [99, 50, 51]. We implement the algorithm for linear and quadratic geometric reconstructions and demonstrate second- and third-order accuracy for the integration procedure, respectively.

1.3 Compound Domains

We focus on irregular domains with piecewise smooth boundaries. Specifically, we consider domains that can be described by a combination of domains with smooth boundaries, which we shall call *generating domains*. We consider boolean operations of *intersection* and *union*[1] that can be used to combine generating domains (see Figure 1.1).



(a) Union. (b) Intersection.

Figure 1.1: Elementary operations to construct compound domains.

[1]We do not consider separately the boolean operation of *difference*, since it is equivalent to the operation of *intersection* with the complement of a domain.

We call *compound domain* a domain that is obtained through a sequential combination of generating domains with smooth boundaries using the two aforementioned boolean operations. An example of such a domain is pictured in Figure 1.2. This domain is described by three generating domains $\{\Omega_i\}_{i=1}^3$ and the construction of the compound domain proceeds as follows: Ω_1 is the initial shape (combined with empty space using the *union* operation). Ω_2 is added using the *union* operation. Finally, Ω_3, defined as the exterior of a circle, is combined using the *intersection* operation. Clearly, such an approach can be used to generate complicated geometries, with possibly kinks in their boundaries, in both two and three spatial dimensions.

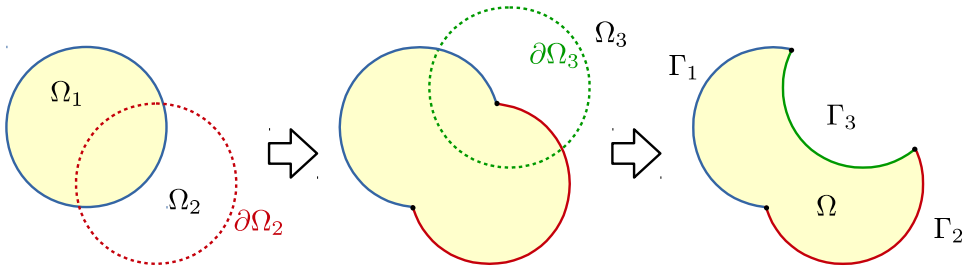


Figure 1.2: example of the construction of a compound domain.

We denote by Ω a compound domain and by $\partial\Omega$ its piecewise smooth boundary. We denote by Ω_1, Ω_2, ..., the generating domains and by $\partial\Omega_1$, $\partial\Omega_2$, ..., their boundaries (also called *interfaces*). Finally, we denote by Γ_1, Γ_2, ..., the corresponding smooth components of $\partial\Omega$ so that $\partial\Omega = (\Gamma_1 \cup \Gamma_2 \cup \ldots)$.

We employ the level-set framework to represent each generating domain. Briefly, in the level-set method the boundary $\partial\Omega$ of a domain Ω is described by the zero-isocontour of a Lipschitz-continuous function, called the level-set function, $\phi(\boldsymbol{r})$, such that the value

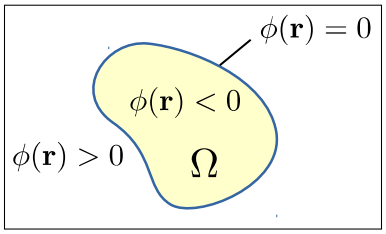


Figure 1.3: Level-set representation of a two-dimensional irregular domain Ω.

of the function is negative inside the domain Ω and positive outside, i.e.

$$\begin{aligned} \phi(\boldsymbol{r}) &< 0, \quad \forall \boldsymbol{r} \in \Omega, \\ \phi(\boldsymbol{r}) &= 0, \quad \forall \boldsymbol{r} \in \partial\Omega, \\ \phi(\boldsymbol{r}) &> 0, \quad \forall \boldsymbol{r} \notin \Omega. \end{aligned}$$

We consider locally C^2-smooth level-set functions. We refer the interested reader to [122, 145, 121] for more details on the level-set method and to [59] for a recent review.

To summarize, we consider compound domains that are described by collections of locally C^2-smooth level-set functions $\{\phi_i\}_{i=1}^{n_\phi}$, each of which represents a generating domain, with associated sets of operations $\{\sigma_i\}_{i=1}^{n_\phi}$, where σ_i = *intersection* or *union* and n_ϕ is the total number of level-set functions describing a compound domain.

1.4 Numerical Method

Let us consider solving equation (1.1) with boundary conditions (1.2), where Ω is a compound domain. For the sake of clarity we present numerical schemes in the two dimensional case; extension to three spatial dimensions is straightforward.

We enclose the domain Ω into a rectangular computational area $[x_{\min}, x_{\max}] \times$

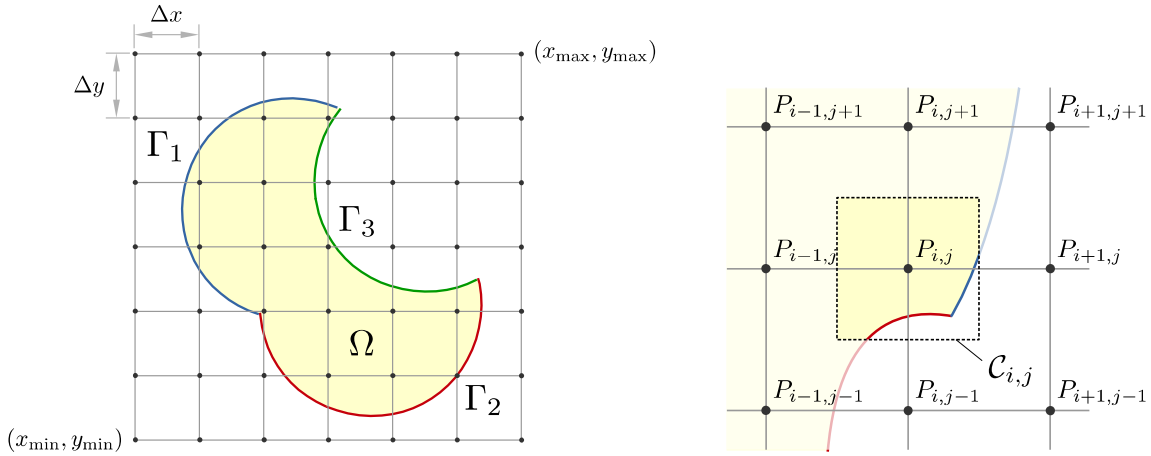
$[y_{\min}, y_{\max}]$ and discretize the area into a uniform rectangular grid of n_x grid points in the x-direction and n_y grid points in the y-direction (Figure 1.4a). Spatial steps of such a grid are given by

$$\Delta x = \frac{x_{\max} - x_{\min}}{n_x - 1} \quad \text{and} \quad \Delta y = \frac{y_{\max} - y_{\min}}{n_y - 1}.$$

We enumerate grid points in a standard manner, i.e. $P_{i,j}$ stands for a point located at

$$\boldsymbol{r}_{i,j} = \begin{pmatrix} x_{i,j} \\ y_{i,j} \end{pmatrix} = \begin{pmatrix} x_{\min} + (i-1)\Delta x \\ y_{\min} + (j-1)\Delta y \end{pmatrix}, \quad \forall (i,j) \in [1, n_x] \times [1, n_y].$$

We also denote $u_{i,j} = u(\boldsymbol{r}_{i,j})$.



(a) (b)

Figure 1.4: (a) Discretization of space and (b) definition of the control volume for $P_{i,j}$.

With each grid point $P_{i,j}$, we associate a rectangular finite volume $\mathcal{C}_{i,j}$ of dimensions $\Delta x \times \Delta y$ such that $P_{i,j}$ is the center of $\mathcal{C}_{i,j}$ (see Figure 1.4b) and employ a finite volume approach to discretize equation (1.1). Let us consider a cell $\mathcal{C}_{i,j}$, which may be crossed by the domain boundary $\partial\Omega$ (Figure 1.4b). Integration of equation (1.1) over the intersection of Ω with $\mathcal{C}_{i,j}$, and application of the divergence theorem yields

$$- \underbrace{\int_{\mathcal{C}_{i,j}\cap\partial\Omega} \mu(\boldsymbol{r})\frac{\partial u(\boldsymbol{r})}{\partial \boldsymbol{n}}\, d\Gamma}_{\text{Flux through the domain's boundary in } \mathcal{C}_{i,j}} - \underbrace{\int_{\partial\mathcal{C}_{i,j}\cap\Omega} \mu(\boldsymbol{r})\frac{\partial u(\boldsymbol{r})}{\partial \boldsymbol{n}}\, d\Gamma}_{\text{Fluxes between adjacent cells}} + \int_{\mathcal{C}_{i,j}\cap\Omega} k(\boldsymbol{r})u(\mathbf{r})\, d\Omega = \int_{\mathcal{C}_{i,j}\cap\Omega} f(\boldsymbol{r})\, d\Omega,$$

where we split the boundary integral over $\partial(\mathcal{C}_{i,j}\cap\Omega)$ into two separate integrals: over the part of the domain boundary contained in $\mathcal{C}_{i,j}$, i.e. $\mathcal{C}_{i,j}\cap\partial\Omega$, and over the cell's boundary included in the domain, i.e. $\partial\mathcal{C}_{i,j}\cap\Omega$. Recalling that the domain's boundary consists of C^2-smooth parts $\{\Gamma_p\}_{p=1}^{n_\phi}$, on which u satisfies Robin boundary conditions (1.2), the equation above can be transformed into:

$$\underbrace{\sum_{p=1}^{n_\phi}\int_{\mathcal{C}_{i,j}\cap\Gamma_p} \alpha_p(\boldsymbol{r})u(\boldsymbol{r})\, d\Gamma}_{\text{Robin b.c. term}} - \underbrace{\int_{\partial\mathcal{C}_{i,j}\cap\Omega} \mu(\boldsymbol{r})\frac{\partial u(\boldsymbol{r})}{\partial \boldsymbol{n}}\, d\Gamma}_{\text{Fluxes between adjacent cells}} + \int_{\mathcal{C}_{i,j}\cap\Omega} k(\boldsymbol{r})u(\mathbf{r})\, d\Omega = \int_{\mathcal{C}_{i,j}\cap\Omega} f(\boldsymbol{r})\, d\Omega + \sum_{p=1}^{n_\phi}\int_{\mathcal{C}_{i,j}\cap\Gamma_p} g_p(\boldsymbol{r})\, d\Gamma. \quad (1.3)$$

We first note that the domain integrals in (1.3), i.e. the last term in the left-hand side and the first term in the right-hand side, can be approximated by the value of the integrand multiplied by the volume of $\mathcal{C}_{i,j}\cap\Omega$, which we shall denote as $V_{i,j}$, i.e., we have:

$$\int_{\mathcal{C}_{i,j}\cap\Omega} k(\boldsymbol{r})u(\mathbf{r})\, d\Omega = k(\boldsymbol{r}_{i,j})u_{i,j}V_{i,j} + \mathcal{O}\left(h^{\mathcal{D}+1}\right), \quad (1.4)$$

and

$$\int_{\mathcal{C}_{i,j}\cap\Omega} f(\boldsymbol{r})\, d\Omega = f(\boldsymbol{r}_{i,j})V_{i,j} + \mathcal{O}\left(h^{\mathcal{D}+1}\right), \quad (1.5)$$

where $h = \max(\Delta x, \Delta y)$ and $\mathcal{D}$ is the problem dimensionality. The second term on the right-hand side of (1.3) represents integrals of known functions over C^2-smooth components of a C^0 domain boundary. Calculation of such integrals is discussed in Section 1.4.7.

In what follows we discuss the discretization of fluxes between cells (second term in (1.3)) followed by a discussion of the Robin boundary term discretization (first term in (1.3)). For each, we first present a simple symmetric discretization before describing a more accurate, albeit nonsymmetric, discretization.

1.4.1 Discretization of fluxes between cells

Let us denote by $F_{i\pm\frac{1}{2},j}$ (resp. $F_{i,j\pm\frac{1}{2}}$) the face between cells $\mathcal{C}_{i,j}$ and $\mathcal{C}_{i\pm1,j}$ (resp. $\mathcal{C}_{i,j}$ and $\mathcal{C}_{i,j\pm1}$). Using the fact that on each of the faces the normal derivative of u has a simple form, one can express the second term in the left-hand side of (1.3) as:

$$\begin{aligned}\int_{\partial\mathcal{C}_{i,j}\cap\Omega} \mu\frac{\partial u}{\partial \boldsymbol{n}}\, d\Gamma = &\int_{F_{i+\frac{1}{2},j}\cap\Omega} \mu\frac{\partial u}{\partial x}\, d\Gamma - \int_{F_{i-\frac{1}{2},j}\cap\Omega} \mu\frac{\partial u}{\partial x}\, d\Gamma \\ &+ \int_{F_{i,j+\frac{1}{2}}\cap\Omega} \mu\frac{\partial u}{\partial y}\, d\Gamma - \int_{F_{i,j-\frac{1}{2}}\cap\Omega} \mu\frac{\partial u}{\partial y}\, d\Gamma. \end{aligned} \tag{1.6}$$

Symmetric discretization

A simple discretization of (1.6) that leads to a symmetric linear system can be obtained following the approach of [128, 116]. For example, to approximate the total flux through the cut face $F_{i+\frac{1}{2},j}\cap\Omega$ one can estimate the x-derivative of u at the center of $F_{i+\frac{1}{2},j}$ using values $u_{i,j}$ and $u_{i+1,j}$ and multiply it by the value of $\mu(\boldsymbol{r})$ at the same point and the

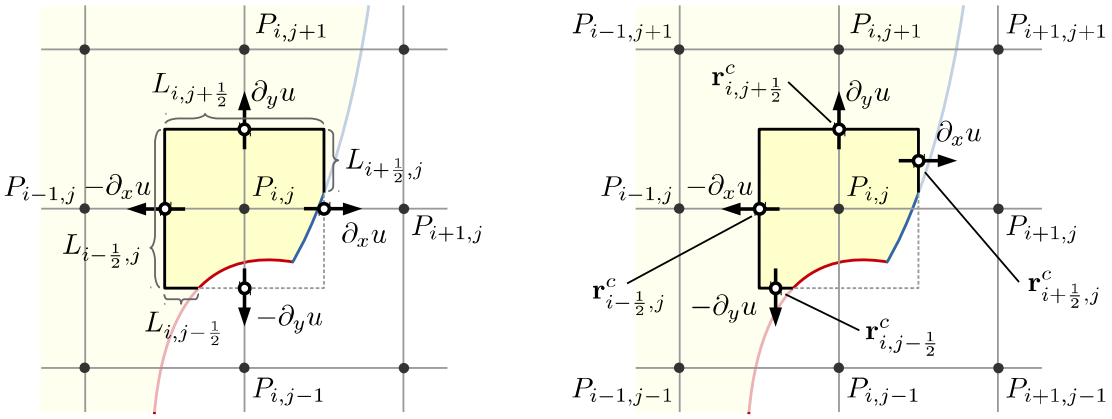


Figure 1.5: Nomenclature for the symmetric (a) and nonsymmetric (b) discretizations of the fluxes between cells.

length $L_{i+\frac{1}{2},j}$ of the cut face $F_{i+\frac{1}{2},j} \cap \Omega$, i.e.

$$\begin{aligned}\int_{F_{i+\frac{1}{2},j}\cap\Omega} \mu\frac{\partial u}{\partial x}\, d\Gamma &= \int_{F_{i+\frac{1}{2},j}\cap\Omega} \left(\mu\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j} + \mathcal{O}\left(h\right)\, d\Gamma \\ &= \left(\mu\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j} \int_{F_{i+\frac{1}{2},j}\cap\Omega} d\Gamma + \mathcal{O}\left(h\right) \times \mathcal{O}\left(h^{\mathcal{D}-1}\right) \\ &= \mu_{i+\frac{1}{2},j}\frac{u_{i+1,j}-u_{i,j}}{\Delta x} L_{i+\frac{1}{2},j} + \mathcal{O}\left(h^{\mathcal{D}}\right).\end{aligned}$$

The application of the same procedure to the rest of the faces leads to:

$$\begin{aligned}\int_{\partial\mathcal{C}_{i,j}\cap\Omega} \mu\frac{\partial u}{\partial \mathbf{n}}\, d\Gamma = \mu_{i+\frac{1}{2},j}\frac{u_{i+1,j}-u_{i,j}}{\Delta x} L_{i+\frac{1}{2},j} + \mu_{i-\frac{1}{2},j}\frac{u_{i-1,j}-u_{i,j}}{\Delta x} L_{i-\frac{1}{2},j} \\ + \mu_{i,j+\frac{1}{2}}\frac{u_{i,j+1}-u_{i,j}}{\Delta y} L_{i,j+\frac{1}{2}} + \mu_{i,j-\frac{1}{2}}\frac{u_{i,j-1}-u_{i,j}}{\Delta y} L_{i,j-\frac{1}{2}} + \mathcal{O}\left(h^{\mathcal{D}}\right), \quad (1.7)\end{aligned}$$

where $L_{i\pm\frac{1}{2},j}$ (resp. $L_{i,j\pm\frac{1}{2}}$) is the length of the cut edge $F_{i\pm\frac{1}{2},j} \cap \Omega$ (resp. $F_{i,j\pm\frac{1}{2}} \cap \Omega$) and $\mu_{i\pm\frac{1}{2},j}$ (resp. $\mu_{i,j\pm\frac{1}{2}}$) is the value of $\mu(\boldsymbol{r})$ at the center of $F_{i\pm\frac{1}{2},j}$ (resp. $F_{i,j\pm\frac{1}{2}}$) (see Figure 1.5a). It is straightforward to show that such a discretization is symmetric [128].

Higher-order discretization

A more accurate discretization of the fluxes through cell faces can be obtained by estimating the normal derivatives and the diffusion coefficient at the centers of the cut faces, rather than at the centers of full faces (see Figure 1.5b), as it is done in [80, 141]. Consider the flux through the face $F_{i+\frac{1}{2},j}$. Using a Taylor expansion of the integrand at the center of the cut face, we have:

$$\int_{F_{i+\frac{1}{2},j}\cap\Omega} \mu\frac{\partial u}{\partial x}\,d\Gamma = \int_{F_{i+\frac{1}{2},j}\cap\Omega} \left(\left(\mu\frac{\partial u}{\partial x}\right)_{\boldsymbol{r}^c_{i+\frac{1}{2},j}} + (y-y^c_{i+\frac{1}{2},j})\frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial x}\right)_{\boldsymbol{r}^c_{i+\frac{1}{2},j}} + \mathcal{O}\left(h^2\right)\right) d\Gamma = \mu(\boldsymbol{r}^c_{i+\frac{1}{2},j})\left(\frac{\partial u}{\partial x}\right)_{\boldsymbol{r}^c_{i+\frac{1}{2},j}} L_{i+\frac{1}{2},j} + \underbrace{\mathcal{O}\left(h^2\right)\times\mathcal{O}\left(h^{\mathcal{D}-1}\right)}_{\mathcal{O}(h^{\mathcal{D}+1})}, \quad (1.8)$$

where $\boldsymbol{r}^c_{i+\frac{1}{2},j} = (x_{i+\frac{1}{2},j}, y^c_{i+\frac{1}{2},j})$ denotes the centroid of the cut face $F_{i+\frac{1}{2},j}\cap\Omega$ (see Figure 1.5b). Note that $\displaystyle\int_{F_{i+\frac{1}{2},j}\cap\Omega}(y-y^c_{i+\frac{1}{2},j})\,d\Gamma = 0$, hence the $\mathcal{O}\left(h^{\mathcal{D}+1}\right)$ term.

The normal derivatives at the centroid of the cut faces are linearly interpolated from the normal derivatives at the centers of the full face under consideration and its neighboring faces. For example, when considering the face $F_{i+\frac{1}{2},j}$, the normal derivative is discretized as:

$$\left(\frac{\partial u}{\partial x}\right)_{\boldsymbol{r}^c_{i+\frac{1}{2},j}} = \left(1-\left|\theta_{i+\frac{1}{2},j}\right|\right)\left(\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j} + \left|\theta_{i+\frac{1}{2},j}\right| \times \begin{cases} \left(\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j+1}, & \theta_{i+\frac{1}{2},j} < 0, \\ \left(\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j-1}, & \theta_{i+\frac{1}{2},j} > 0. \end{cases} + \mathcal{O}\left(h^2\right),$$

where $\theta_{i+\frac{1}{2},j} = \dfrac{y^c_{i+\frac{1}{2},j} - y_{i,j}}{\Delta y}$.

Similar discretizations hold for the other faces of a cell.

1.4.2 Discretization of the Robin b.c. term

We focus on the following term in equation (1.3):

$$\sum_{p=1}^{n_\phi} \int_{\mathcal{C}_{i,j} \cap \Gamma_p} \alpha_p(\boldsymbol{r}) u(\boldsymbol{r})\, d\Gamma,$$

and describe a discretization that leads to a symmetric linear system (with first-order accurate gradients) and a discretization that produces second-order accurate gradients (with a nonsymmetric linear system).

Symmetric discretization

Let us first assume that the part of the domain's boundary $\partial\Omega$ contained in the cell $\mathcal{C}_{i,j}$ is entirely smooth, that is, $\mathcal{C}_{i,j}$ contains only one smooth component of $\partial\Omega$, say, $\Gamma_{\hat{p}}$. Then,

$$\sum_{p=1}^{n_\phi} \int_{\mathcal{C}_{i,j} \cap \Gamma_p} \alpha_p(\boldsymbol{r}) u(\boldsymbol{r})\, d\Gamma = \int_{\mathcal{C}_{i,j} \cap \Gamma_{\hat{p}}} \alpha_{\hat{p}}(\boldsymbol{r}) u(\boldsymbol{r})\, d\Gamma. \tag{1.9}$$

We approximate the integral in (1.9) by the length of $\mathcal{C}_{i,j} \cap \Gamma_{\hat{p}}$ multiplied by the integrand value at the point $\boldsymbol{r}^{\hat{p}}_{i,j}$ on $\Gamma_{\hat{p}}$, which also lies on the line collinear to the normal to $\Gamma_{\hat{p}}$ and passing through the cell center $\boldsymbol{r}_{i,j}$ (see Figure 1.6), i.e.

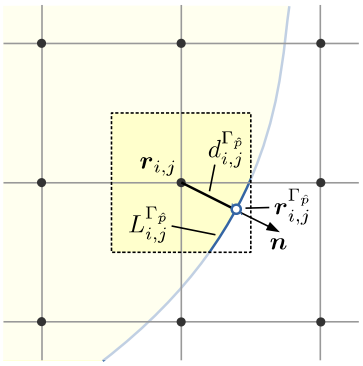


Figure 1.6: Symmetric discretization of the Robin b.c. term in case of a smooth boundary.

$$\begin{aligned}\int_{\mathcal{C}_{i,j}\cap\Gamma_{\hat{p}}} \alpha_{\hat{p}}(\boldsymbol{r})u(\boldsymbol{r})\,d\Gamma &= \int_{\mathcal{C}_{i,j}\cap\Gamma_{\hat{p}}} \left(\alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})u(\boldsymbol{r}_{i,j}^{\hat{p}}) + \mathcal{O}\left(h\right)\right)\,d\Gamma \\ &= \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})u(\boldsymbol{r}_{i,j}^{\hat{p}})L_{i,j}^{\hat{p}} + \underbrace{\mathcal{O}\left(h\right)\times\mathcal{O}\left(h^{\mathcal{D}-1}\right)}_{\mathcal{O}(h^{\mathcal{D}})},\end{aligned} \tag{1.10}$$

where $L_{i,j}^{\hat{p}} = \int_{\mathcal{C}_{i,j}\cap\Gamma_{\hat{p}}} d\Gamma$ is the length of $\mathcal{C}_{i,j}\cap\Gamma_{\hat{p}}$.

The coordinates of $\boldsymbol{r}_{i,j}^{\hat{p}}$ can be calculated using the level-set function $\phi_{\hat{p}}$ as (see, e.g., [145, 121]):

$$\boldsymbol{r}_{i,j}^{\hat{p}} = \boldsymbol{r}_{i,j} - d_{i,j}^{\hat{p}}\frac{\nabla\phi_{\hat{p}}(\boldsymbol{r}_{i,j})}{|\nabla\phi_{\hat{p}}(\boldsymbol{r}_{i,j})|} + \mathcal{O}\left(h^2\right),$$

where $d_{i,j}^{\hat{p}}$ is the signed distance between $\boldsymbol{r}_{i,j}$ and the sub-boundary $\Gamma_{\hat{p}}$, which can be estimated as:

$$d_{i,j}^{\hat{p}} = \frac{\phi_{\hat{p}}(\boldsymbol{r}_{i,j})}{|\nabla\phi_{\hat{p}}(\boldsymbol{r}_{i,j})|} + \mathcal{O}\left(h^2\right).$$

Using the Taylor expansion of u at $\boldsymbol{r}_{i,j}^{\hat{p}}$, one can obtain the following relation between $u_{i,j}$ and $u(\boldsymbol{r}_{i,j}^{\hat{p}})$:

$$u_{i,j} = u(\boldsymbol{r}_{i,j}^{\hat{p}}) + d_{i,j}^{\hat{p}} \frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{r}_{i,j}^{\hat{p}}) + \mathcal{O}\left(h^2\right). \tag{1.11}$$

At the same time the Robin boundary condition (1.2) on $\Gamma_{\hat{p}}$ gives a relation between $\frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{r}_{i,j}^{\hat{p}})$ and $u(\boldsymbol{r}_{i,j}^{\hat{p}})$:

$$\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) \frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{r}_{i,j}^{\hat{p}}) + \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}}) u(\boldsymbol{r}_{i,j}^{\hat{p}}) = g_{\hat{p}}(\boldsymbol{r}_{i,j}^{\Gamma_p}). \tag{1.12}$$

Solving equations (1.11) and (1.12) for $u(\boldsymbol{r}_{i,j}^{\hat{p}})$ one obtains

$$u(\boldsymbol{r}_{i,j}^{\hat{p}}) = u_{i,j} \frac{\mu(\boldsymbol{r}_{i,j}^{\hat{p}})}{\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) - \alpha_p(\boldsymbol{r}_{i,j}^{\hat{p}}) d_{i,j}^{\hat{p}}} - \frac{g(\boldsymbol{r}_{i,j}^{\hat{p}}) d_{i,j}^{\hat{p}}}{\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) - \alpha_p(\boldsymbol{r}_{i,j}^{\hat{p}}) d_{i,j}^{\hat{p}}} + \mathcal{O}\left(h^2\right) \tag{1.13}$$

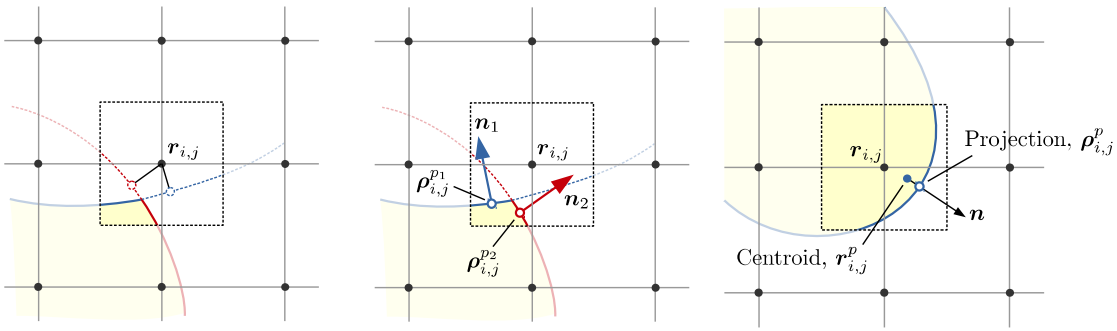
Note that for sufficiently refined computational grids in the sense that $h < C \min_{\alpha>0}(\mu/\alpha)$, where C is a $O(1)$ constant accounting for the problem dimensionality, the denominators in the above expression cannot turn zero. Substituting of (1.13) into (1.10) produces the following discretization for the Robin b.c. term:

$$\int_{\mathcal{C}_{i,j} \cap \Gamma_{\hat{p}}} \alpha_{\hat{p}}(\boldsymbol{r}) u(\boldsymbol{r})\, d\Gamma = u_{i,j} \left(\frac{\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}}) L_{i,j}^{\hat{p}}}{\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) - \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}}) d_{i,j}^{\hat{p}}} \right) + \left(\frac{g(\boldsymbol{r}_{i,j}^{\hat{p}}) d_{i,j}^{\hat{p}} \alpha_p(\boldsymbol{r}_{i,j}^{\hat{p}}) L_{i,j}^{\hat{p}}}{\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) - \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}}) d_{i,j}^{\hat{p}}} \right) + \mathcal{O}\left(h^{\mathcal{D}}\right). \tag{1.14}$$

Since the discretization above does not involve any value of the unknown function at any of neighboring points, it preserves the symmetry of the resulting linear system.

Now let us consider a cell that contains an intersection of two interfaces, say Γ_{p_1} and Γ_{p_2}. In this case the Robin b.c. term reduces to:

$$\sum_{p=1}^{n_\phi} \int_{\mathcal{C}_{i,j} \cap \Gamma_p} \alpha_p(\boldsymbol{r}) u(\boldsymbol{r})\, d\Gamma = \sum_{p=p_1,p_2} \int_{\mathcal{C}_{i,j} \cap \Gamma_p} \alpha_p(\boldsymbol{r}) u(\boldsymbol{r})\, d\Gamma. \tag{1.15}$$



(a) (b) (c)

Figure 1.7: An example when projection points are outside of the domain due to the presence of a kink (a), projections of centroids at which boundary conditions are used to approximate u (b) and an illustration of the relation between an interface's centroid and its projections (c).

In the case where the domain's boundary has a kink, the accuracy of the method presented above will drop since projection points to Γ_{p_1} and Γ_{p_2} may fall outside of the domain Ω (see Figure 1.7a). Therefore we propose a special treatment for cells near kinks. The idea of this approach is to use a multidimensional Taylor expansion of u at (i,j):

$$u(\boldsymbol{r}) = u_{i,j} + \nabla u(\boldsymbol{r}_{i,j}) \cdot (\boldsymbol{r} - \boldsymbol{r}_{i,j}) + \mathcal{O}\left(h^2\right) \tag{1.16}$$

in equation (1.15), where the gradient $\nabla u(\boldsymbol{r}_{i,j})$ is approximated using the boundary conditions on the two intersecting interfaces. Such an approximation of the gradient is possible since the boundary conditions on two interfaces provide values of the u derivatives

in two non-collinear directions.

Specifically, we enforce that the above approximation of u satisfy the boundary condition at one point of Γ_{p_1} and at one point of Γ_{p_2}. We choose those points to be the projections of the interfaces' centroid onto the interfaces themselves (see Figure 1.7b). Centroids $\boldsymbol{r}_{i,j}^{p_1}$ and $\boldsymbol{r}_{i,j}^{p_2}$ of interfaces $\Gamma_{p_1} \cap \mathcal{C}_{i,j}$ and $\Gamma_{p_2} \cap \mathcal{C}_{i,j}$ can be calculated as:

$$\boldsymbol{r}_{i,j}^{p_1} = \frac{\int_{\Gamma_{p_1} \cap \mathcal{C}_{i,j}} \boldsymbol{r}\, d\boldsymbol{r}}{\int_{\Gamma_{p_1} \cap \mathcal{C}_{i,j}} d\boldsymbol{r}} \quad \text{and} \quad \boldsymbol{r}_{i,j}^{p_2} = \frac{\int_{\Gamma_{p_2} \cap \mathcal{C}_{i,j}} \boldsymbol{r}\, d\boldsymbol{r}}{\int_{\Gamma_{p_2} \cap \mathcal{C}_{i,j}} d\boldsymbol{r}}.$$

Then their projections $\boldsymbol{\rho}_{i,j}^{p_1}$ and $\boldsymbol{\rho}_{i,j}^{p_1}$ can be easily found using level-set functions ϕ_{p_1} and ϕ_{p_2} (see, e.g., [145, 121]):

$$\begin{aligned}
\boldsymbol{\rho}_{i,j}^{p_1} &= \boldsymbol{r}_{i,j}^{p_1} - \left(\frac{\phi_{p_1}}{|\nabla \phi_{p_1}|} \boldsymbol{n}_1 \right)_{\boldsymbol{r}=\boldsymbol{r}_{i,j}^{p_1}} + \mathcal{O}\left(h^2\right), \\
\boldsymbol{\rho}_{i,j}^{p_2} &= \boldsymbol{r}_{i,j}^{p_2} - \left(\frac{\phi_{p_2}}{|\nabla \phi_{p_2}|} \boldsymbol{n}_2 \right)_{\boldsymbol{r}=\boldsymbol{r}_{i,j}^{p_2}} + \mathcal{O}\left(h^2\right),
\end{aligned}$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are the normals to the two interfaces at their respective centroid:

$$\boldsymbol{n}_1 = \left(\frac{\nabla \phi_{p_1}}{|\nabla \phi_{p_1}|} \right)_{\boldsymbol{r}=\boldsymbol{r}_{i,j}^{p_1}} \quad \text{and} \quad \boldsymbol{n}_2 = \left(\frac{\nabla \phi_{p_2}}{|\nabla \phi_{p_2}|} \right)_{\boldsymbol{r}=\boldsymbol{r}_{i,j}^{p_2}},$$

and which are estimated by a multi-linear interpolation from the grid nodes. A typical placement of an interface's centroid and its projection are illustrated in Figure 1.7c.

Boundary conditions at the projection points $\boldsymbol{\rho}_{i,j}^{p_1}$ and $\boldsymbol{\rho}_{i,j}^{p_2}$ have the form:

$$\begin{aligned}
\mu(\boldsymbol{\rho}_{i,j}^{p_1}) \frac{\partial u}{\partial \boldsymbol{n}_1} + \alpha_{p_1}(\boldsymbol{\rho}_{i,j}^{p_1}) u(\boldsymbol{\rho}_{i,j}^{p_1}) &= g_{p_1}(\boldsymbol{\rho}_{i,j}^{p_1}) + \mathcal{O}\left(h\right), \\
\mu(\boldsymbol{\rho}_{i,j}^{p_2}) \frac{\partial u}{\partial \boldsymbol{n}_2} + \alpha_{p_2}(\boldsymbol{\rho}_{i,j}^{p_2}) u(\boldsymbol{\rho}_{i,j}^{p_2}) &= g_{p_2}(\boldsymbol{\rho}_{i,j}^{p_2}) + \mathcal{O}\left(h\right),
\end{aligned}$$

where the $\mathcal{O}(h)$ errors are due to using estimations of the normals at the interfaces' centroid. Substituting approximation (1.16) for u into the above two equations we obtain the following 2×2 linear system:

$$\underline{\underline{\boldsymbol{N}}} \cdot \nabla u(\boldsymbol{r}_{i,j}) = \begin{pmatrix} g_{p_1}(\boldsymbol{\rho}_{i,j}^{p_1}) \\ g_{p_2}(\boldsymbol{\rho}_{i,j}^{p_2}) \end{pmatrix} - u_{i,j} \begin{pmatrix} \alpha_{p_1}(\boldsymbol{\rho}_{i,j}^{p_1}) \\ \alpha_{p_2}(\boldsymbol{\rho}_{i,j}^{p_2}) \end{pmatrix}, \tag{1.17}$$

where the matrix $\underline{\underline{\boldsymbol{N}}}$ is defined as:

$$\underline{\underline{\boldsymbol{N}}} = \begin{pmatrix} \mu(\boldsymbol{\rho}_{i,j}^{p_1}) n_x^1 + \alpha_{p_1}(\boldsymbol{\rho}_{i,j}^{p_1})(\zeta_{i,j}^{p_1} - x_{i,j}) & \mu(\boldsymbol{\rho}_{i,j}^{p_1}) n_y^1 + \alpha_{p_1}(\boldsymbol{\rho}_{i,j}^{p_1})(\eta_{i,j}^{p_1} - y_{i,j}) \\ \mu(\boldsymbol{\rho}_{i,j}^{p_2}) n_x^2 + \alpha_{p_2}(\boldsymbol{\rho}_{i,j}^{p_2})(\zeta_{i,j}^{p_2} - x_{i,j}) & \mu(\boldsymbol{\rho}_{i,j}^{p_2}) n_y^2 + \alpha_{p_2}(\boldsymbol{\rho}_{i,j}^{p_2})(\eta_{i,j}^{p_2} - y_{i,j}) \end{pmatrix}.$$

Thus, the gradient $\nabla u(\boldsymbol{r}_{i,j})$ can be approximated as:

$$\nabla u(\boldsymbol{r}_{i,j}) = \boldsymbol{b}_{i,j} - \boldsymbol{a}_{i,j} u_{i,j} + (\mathcal{O}(h), \mathcal{O}(h))^T, \tag{1.18}$$

where the vectors $\boldsymbol{a}_{i,j}$ and $\boldsymbol{b}_{i,j}$ are given by:

$$\boldsymbol{a}_{i,j} = \underline{\underline{\boldsymbol{N}}}^{-1} \cdot \begin{pmatrix} \alpha_{p_1}(\boldsymbol{\rho}_{i,j}^{p_1}) \\ \alpha_{p_2}(\boldsymbol{\rho}_{i,j}^{p_2}) \end{pmatrix} \quad \text{and} \quad \boldsymbol{b}_{i,j} = \underline{\underline{\boldsymbol{N}}}^{-1} \cdot \begin{pmatrix} g_{p_1}(\boldsymbol{\rho}_{i,j}^{p_1}) \\ g_{p_2}(\boldsymbol{\rho}_{i,j}^{p_2}) \end{pmatrix}.$$

As a result, using (1.16) with the gradient estimate (1.18), we have:

$$\begin{aligned} \sum_{p=1}^{n_\phi} \int_{\mathcal{C}_{i,j} \cap \Gamma_p} \alpha_p(\boldsymbol{r}) u(\boldsymbol{r})\, d\Gamma = \\ u_{i,j} \sum_{p=p_1,p_2} \left(\int_{\mathcal{C}_{i,j} \cap \Gamma_p} \alpha_p(\boldsymbol{r}) \left(1 - \boldsymbol{a}_{i,j} \cdot (\boldsymbol{r} - \boldsymbol{r}_{i,j})\right) d\Gamma \right) \\ + \sum_{p=p_1,p_2} \left(\int_{\mathcal{C}_{i,j} \cap \Gamma_p} \alpha_p(\boldsymbol{r}) \boldsymbol{b}_{i,j} \cdot (\boldsymbol{r} - \boldsymbol{r}_{i,j})\, d\Gamma \right) + \mathcal{O}\left(h^{\mathcal{D}+1}\right). \end{aligned} \tag{1.19}$$

Again, this discretization does not involve the unknown function at any neighboring

points, therefore preserving the symmetry of the linear system.

Remarks:

1. It is interesting to note that since additional information is given by the boundary conditions for computational cells with kinks (i.e. imposing boundary conditions at two points rather than at one), we are able, rather fortuitously, to obtain an approximation of the total flux through the domain's boundary $\partial\Omega \cap \mathcal{C}_{i,j}$ that is of a higher, $(\mathcal{D}+1)$, order of accuracy than for cells without kinks. However, since this approximation is combined with $\mathcal{O}\left(h^{\mathcal{D}}\right)$ accurate discretization of fluxes between cells (section 1.4.1), the overall error is still $\mathcal{O}\left(h^{\mathcal{D}}\right)$. Moreover, since this discretization is applied only in $\mathcal{O}\left(N^{\mathcal{D}-2}\right)$ cells containing kinks (compared to the total $\mathcal{O}\left(N^{\mathcal{D}-1}\right)$ number of cells containing a domain's boundary), where N is a characteristic number of grid cells in one Cartesian direction, it is not expected to affect significantly the overall accuracy of the numerical method.

2. We do not consider cases where a cell contains more than 2 interfaces in two dimensions, since such cases are essentially under-resolved. Still, such cases pose no problem: a linear system analogous to (1.17) would be just over-determined and should be solved using a least-squares approach.

3. In the case of two intersecting interfaces in three spatial dimensions, a system analogous to (1.17) would be under-determined. We found that closing the system by approximating u as a constant in the third direction, i.e. $\dfrac{\partial u}{\partial \boldsymbol{n}_3} = 0$, where $\boldsymbol{n}_3$ is a

unit-vector perpendicular to the normals of the two intersecting interfaces, produces good results: designed convergence rates are recovered. In fact, such an approach can be viewed as a slight improvement over the approach described above for cells containing only one smooth component of a domain's boundary, which could be interpreted as the present approach with setting $\dfrac{\partial u}{\partial \boldsymbol{n}_2} = \dfrac{\partial u}{\partial \boldsymbol{n}_3} = 0$.

4. Unlike the two dimensional case, there may be well-resolved situation in three spatial dimensions where more than three interfaces are intersecting in the same cell (e.g. the top of a square pyramid at which four planes meet). In such cases a system analogous to (1.17) would be over-determined and should be solved using a least-squares approach.

5. As intersecting interfaces get almost parallel matrix $\underline{\underline{\boldsymbol{N}}}$ is expected to become singular. Thus described above special method for kinks should not be used in such cases, instead the approach for cells not containing sharp features should be used for each of the intersecting interfaces independently.

Higher-order discretization

To obtain a $(\mathcal{D}+1)$-order accurate discretization of the Robin b.c. term in all cells, we apply an approach similar to the one for cells with kinks described in the previous section.

Denote by $\bar{\boldsymbol{r}}^p_{i,j} = \begin{pmatrix} \bar{x}^p_{i,j} \\ \bar{y}^p_{i,j} \end{pmatrix}$ the centroid of the p-th interface in the cell $\mathcal{C}_{i,j}$, that is:

$$\bar{x}^p_{i,j} = \frac{\int_{\mathcal{C}_{i,j}\cap\Gamma_p} x\, d\Gamma}{\int_{\mathcal{C}_{i,j}\cap\Gamma_p} d\Gamma} \quad \text{and} \quad \bar{y}^p_{i,j} = \frac{\int_{\mathcal{C}_{i,j}\cap\Gamma_p} y\, d\Gamma}{\int_{\mathcal{C}_{i,j}\cap\Gamma_p} d\Gamma}.$$

Expanding the integrands in the first term of (1.3) around the interfaces' centroid one obtains:

$$\begin{aligned}\sum_{p=1}^{n_\phi} \int_{\mathcal{C}_{i,j}\cap\Gamma_p} \alpha_p(\boldsymbol{r})u(\boldsymbol{r})\, d\Gamma = \sum_{p=1}^{n_\phi} \int_{\mathcal{C}_{i,j}\cap\Gamma_p} \Big(&\alpha_p(\bar{\boldsymbol{r}}^p_{i,j})u(\bar{\boldsymbol{r}}^p_{i,j}) \\ &+ \frac{\partial(\alpha_p u)}{\partial x}(\bar{\boldsymbol{r}}^p_{i,j})(x-\bar{x}^p_{i,j}) + \frac{\partial(\alpha_p u)}{\partial y}(\bar{\boldsymbol{r}}^p_{i,j})(y-\bar{y}^p_{i,j}) + \mathcal{O}\left(h^2\right)\Big)\, d\Gamma \\ &= \sum_{p=1}^{n_\phi} \alpha_p(\bar{\boldsymbol{r}}^p_{i,j})u(\bar{\boldsymbol{r}}^p_{i,j}) \int_{\mathcal{C}_{i,j}\cap\Gamma_p} d\Gamma + \mathcal{O}\left(h^{\mathcal{D}+1}\right), \quad (1.20)\end{aligned}$$

where we use the same symbol $\alpha_p(\boldsymbol{r})$ for a smooth extension of $\alpha_p(\boldsymbol{r})$, $\boldsymbol{r} \in \Gamma_p$, to a neighborhood of Γ_p (specific extension will be discussed further). Thus, to obtain a third-order approximation of the Robin b.c. term, one needs to estimate the values of the unknown function u at the centroids of the interfaces $\bar{\boldsymbol{r}}^p_{i,j}$ and multiply them by the length of the interfaces contained in $\mathcal{C}_{i,j}$. We perform this task by approximating u by a linear interpolant:

$$u(\boldsymbol{r}) = u_0 + u_x x + u_y y + \mathcal{O}\left(h^2\right),$$

where the coefficients u_0, u_x and u_y are found by the least square approach described below. Note that a second-order accurate estimate of u is sufficient to preserve the third-order truncation error in equation (1.20) since such estimate is multiplied by the length of the interface, which itself is $\mathcal{O}(h)$.

The coefficients u_0, u_x and u_y are obtained based on the values of u at $\mathcal{C}_{i,j}$ and its neighboring cells and by taking into account the boundary conditions. Thus, we have 9 constraints of the form:

$$u_0 + u_x x_{i+I,j+J} + u_y y_{i+I,j+J} = u_{i+I,j+J}, \quad I = -1, 0, 1, \quad J = -1, 0, 1,$$

and n_ϕ constraints of the form:

$$\mu(\boldsymbol{\rho}_{i,j}^p) \underbrace{(u_x n_x^p + u_y n_y^p)}_{\approx \frac{\partial u}{\partial \boldsymbol{n}_p}(\boldsymbol{\rho}_{i,j}^p)} + \alpha_p(\boldsymbol{\rho}_{i,j}^p) \underbrace{(u_0 + u_x \zeta_{i,j}^p + u_y \eta_{i,j}^p)}_{\approx u(\boldsymbol{\rho}_{i,j}^p)} = g_p(\boldsymbol{\rho}_{i,j}^p),\ p = 1, \ldots, n_\phi,$$

where by $\boldsymbol{\rho}_{i,j}^p = \begin{pmatrix} \zeta_{i,j}^p \\ \eta_{i,j}^p \end{pmatrix}$, we denote the projection of the centroid, $\bar{\boldsymbol{r}}_{i,j}^p$, of the p-th interface onto the interface itself (see Figure 1.7c). Such projection point can be approximately calculated using the level-set function ϕ_p as:

$$\boldsymbol{\rho}_{i,j}^p = \bar{\boldsymbol{r}}_{i,j}^p - \left(\frac{\phi_p}{|\nabla \phi_p|} \boldsymbol{n}_p \right)_{\boldsymbol{r} = \bar{\boldsymbol{r}}_{i,j}^p}.$$

Not all neighboring cells may be part of the discretization as well as not all of n_ϕ interfaces are present in a given finite volume cell. To ignore such invalid constraints we introduce $9 + n_\phi$ weights $w_k, k = 1, \ldots, 9 + n_\phi$, such that $w_k = 0$ if the k^{th} constraint is invalid. Moreover, to increase the influence of nearby constraints, we assign weights for valid constraints based on the distance between the point where u needs to be estimated and the point where a constraint is imposed. Specifically, we compute weights from a Gaussian function:

$$w_k(\boldsymbol{R}) = \exp(-\sigma(\boldsymbol{R} - \boldsymbol{r}_k)^2/h^2),$$

where $\boldsymbol{R}$ and $\boldsymbol{r}_k$ denote points where the approximation of u is needed and where the k^{th} constraint is imposed, respectively. Parameter σ is chosen such that $w_k = 10^{-3}$ if $|\boldsymbol{R} - \boldsymbol{r}_k| = h$, i.e $\sigma = 3\ln(10)$. Such a steep decrease of weight functions $w_k(\boldsymbol{R})$ with distance allows the least-squares approach automatically pick the nearest points for interpolation (as done in [41]).

Thus, at the centroid of the p-th interface $\bar{\boldsymbol{r}}^p_{i,j}$, u is approximated by:

$$u(\bar{\boldsymbol{r}}^p_{i,j}) = u_0 + u_x\bar{x}^p_{i,j} + u_y\bar{y}^p_{i,j} + \mathcal{O}\left(h^2\right),$$

where the coefficients u_0, u_x and u_y are found as the least-squares solution to the following linear system:

$$\underline{\underline{\boldsymbol{X}}}\,\underline{\underline{\boldsymbol{W}}} \begin{pmatrix} u_0 \\ u_x \\ u_y \end{pmatrix} = \underline{\underline{\boldsymbol{W}}} \begin{pmatrix} u_{i-1,j-1} \\ u_{i,j-1} \\ \cdots \\ u_{i+1,j+1} \\ g_1(\bar{\boldsymbol{\rho}}^1_{i,j}) \\ \cdots \\ g_{n_\phi}(\bar{\boldsymbol{\rho}}^{n_\phi}_{i,j}) \end{pmatrix},$$

where matrices $\underline{\underline{\boldsymbol{X}}}$ and $\underline{\underline{\boldsymbol{W}}}$ are defined as:

$$\underline{\underline{\boldsymbol{X}}} = \begin{pmatrix} 1 & x_{i-1,j-1} & y_{i-1,j-1} \\ 1 & x_{i,j-1} & y_{i,j-1} \\ \vdots & \vdots & \vdots \\ 1 & x_{i+1,j+1} & y_{i+1,j+1} \\ \alpha^1(\bar{\boldsymbol{\rho}}_{i,j}^1) & \mu_{i,j}(\bar{\boldsymbol{\rho}}_{i,j}^1)n_x^1 + \alpha^1(\bar{\boldsymbol{\rho}}_{i,j}^1)\zeta_{i,j}^1 & \mu(\bar{\boldsymbol{\rho}}_{i,j}^1)n_y^1 + \alpha^1(\bar{\boldsymbol{\rho}}_{i,j}^1)\eta_{i,j}^1 \\ \vdots & \vdots & \vdots \\ \alpha^{n_\phi}(\bar{\boldsymbol{\rho}}_{i,j}^{n_\phi}) & \mu_{i,j}(\bar{\boldsymbol{\rho}}_{i,j}^{n_\phi})n_x^{n_\phi} + \alpha^{n_\phi}(\bar{\boldsymbol{\rho}}_{i,j}^{n_\phi})\zeta_{i,j}^{n_\phi} & \mu(\bar{\boldsymbol{\rho}}_{i,j}^{n_\phi})n_y^{n_\phi} + \alpha^{n_\phi}(\bar{\boldsymbol{\rho}}_{i,j}^{n_\phi})\eta_{i,j}^{n_\phi} \end{pmatrix}$$

and

$$\underline{\underline{\boldsymbol{W}}} = \begin{pmatrix} w_1 & & \\ & \ddots & \\ & & w_{9+n_\phi} \end{pmatrix}.$$

It is well-known that the solution to such a least-squares problem is given by:

$$\begin{pmatrix} u_0 \\ u_x \\ u_y \end{pmatrix} = \left(\underline{\underline{\boldsymbol{X}}}^T \underline{\underline{\boldsymbol{W}}}\, \underline{\underline{\boldsymbol{X}}}\right)^{-1} \left(\underline{\underline{\boldsymbol{W}}}\, \underline{\underline{\boldsymbol{X}}}\right)^T \begin{pmatrix} u_{i-1,j-1} \\ u_{i,j-1} \\ \cdots \\ u_{i+1,j+1} \\ g_1(\bar{\boldsymbol{\rho}}_{i,j}^1) \\ \cdots \\ g_{n_\phi}(\bar{\boldsymbol{\rho}}_{i,j}^{n_\phi}) \end{pmatrix}.$$

The existence of the inverse in the above equation is insured by checking that at least three/four points not lying on the same line/plane are used for interpolation in two/three spatial dimensions. The approximation given in (1.20) also requires the values of the Robin b.c. coefficients α_p at the centroids, $\boldsymbol{r}_{i,j}^p$, of the interfaces. However, strictly speaking, these coefficients are not defined outside of their respective interfaces Γ_p, while the centroids of $\Gamma_p \cap \mathcal{C}_{i,j}$ do not necessarily lie on the interfaces themselves. We choose constant-in-normal-direction extensions of the coefficients α_p to obtain well-defined values of $\alpha_p(\boldsymbol{r}_{i,j}^p)$, that is, we use $\alpha_p(\boldsymbol{r}_{i,j}^p) = \alpha_p(\boldsymbol{\rho}_{i,j}^p)$.

The overall procedure to impose Robin boundary conditions is thus completely automated and only requires the inversion of the 3-by-3 matrix $\underline{\underline{\boldsymbol{X}}}^T \underline{\underline{\boldsymbol{W}}}\, \underline{\underline{\boldsymbol{X}}}$ for finite volume cells adjacent to the domain's boundary. The approach described above is similar to the one from [5] in that it approximates u by a linear interpolant. However, in [5], the value u_0 is always set to $u_{i,j}$ and the derivatives u_x and u_y are computed using standard finite difference formulas based on available neighboring grid nodes. The approach presented here is more flexible in that it allows u_0 to be computed as part of a least-squares procedure. Another important difference is that it also takes into account the boundary conditions while constructing a linear interpolant for u. We have found that such a least-squares approach produces significantly better results than the simpler approach from [5] in the presence of sharp kinks and corners.

1.4.3 Matrix structure

One of important advantages of the presented methods is that they use a small number of neighboring cells to obtain a discretization of the Poisson-type equation (1.1). Namely, the symmetric scheme uses only the nearest neighbors in the Cartesian directions (that is 4 and 6 neighbors in two and three spatial dimensions, resp.), while the superconvergent scheme additionally uses the nearest neighbors in diagonal directions (that is, 8 and 26 neighbors in two and three spatial dimensions, resp.). Also, such localized numerical stencils result in linear systems of a very simple structure. For example, the non-zero pattern for rows in the resulting matrix in two spatial dimensions using the natural

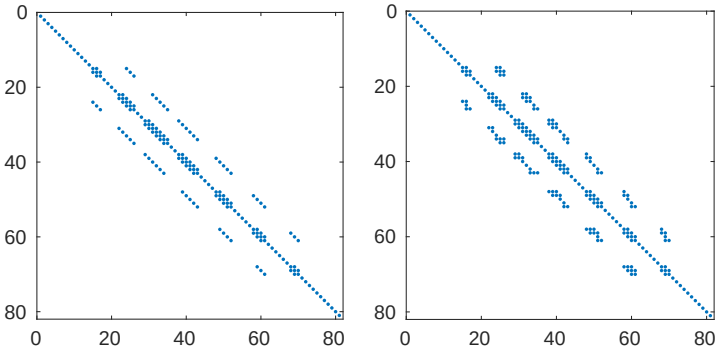
ordering of cells has the following form for the symmetric scheme:

$$\begin{pmatrix} \cdots & a_{n-n_x} & \cdots & a_{n-1} & a_n & a_{n+1} & \cdots & a_{n+n_x} & \cdots \end{pmatrix}$$

i.e. non-zero entries are located only on the main diagonal, the 1-st and the n_x-th subdiagonals as well as the 1-st and the n_x-th superdiagonals. Compared to the case of symmetric scheme, the matrix in case of the superconvergent scheme additionally has non-zero entries on $(n_x - 1)$-th and $(n_x + 1)$-th sub- and super-diagonals:

$$\begin{pmatrix} \cdots & a_{n-n_x-1} & a_{n-n_x} & a_{n-n_x+1} & \cdots & a_{n-1} & a_n & a_{n+1} & \cdots & a_{n+n_x-1} & a_{n+n_x} & a_{n+n_x+1} & \cdots \end{pmatrix}$$

As an example, Figure 1.8 shows the visualization of matrix structure in one of the tests considered later in this work.



(a) Symmetric (b) Superconvergent

Figure 1.8: Visualization of the linear system's structure for the symmetric and superconvergent schemes for example 1.5.2 on a 9^2 grid.

1.4.4 Truncation errors

In this section we analyze the truncation errors of the two schemes presented above. First, for all *interior* nodes, i.e. nodes for which their associated finite volume is completely

inside of a compound domain, both schemes reduce to the standard 5-point scheme for the Poisson equation with second-order truncation error. Let us now consider *boundary* nodes, i.e. nodes for which their control volume is crossed by the irregular domain. Assuming that all the integrals in (1.3) are computed with the required accuracy, it is easy to see that the truncation error for the symmetric scheme is second order (see equations (1.4), (1.5), (1.7), (1.14) and (1.19)), while the truncation error of the nonsymmetric scheme is third order (see equations (1.4), (1.5), (1.8) and (1.20)). However, these are truncation errors for the integrated Poisson-type equation (1.3), i.e. they need to be scaled by the area (volume in 3D) of a finite cell, which is $\mathcal{O}\left(h^{\mathcal{D}}\right)$, before they can be compared with the conventionally defined truncation error. This leads to the following estimates for truncation errors:

$$\begin{aligned} \varepsilon_{\text{tr}}^{\text{sym}} &= \mathcal{O}\left(1\right), \\ \varepsilon_{\text{tr}}^{\text{non-sym}} &= \mathcal{O}\left(h\right), \end{aligned}$$

that is, the symmetric scheme is formally inconsistent at the boundary nodes, while the truncation error of the nonsymmetric scheme is first-order accurate. Taking into consideration results of similar works [80, 58, 141, 27, 115, 128, 53, 54], we expect the symmetric scheme to produce second-order accurate numerical solutions with first-order-accurate gradients and the nonsymmetric scheme to produce second-order accurate solutions and gradients. This will be confirmed in the numerical study in section 1.5.

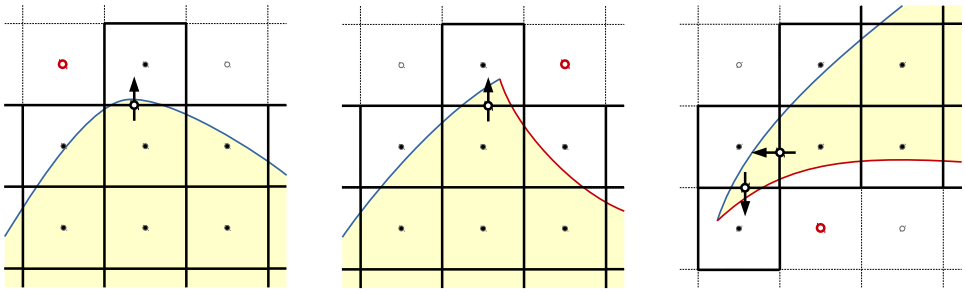


Figure 1.9: examples when the superconvergent discretization is not possible. Arrows indicate fluxes between cells that cannot be estimated with second order of accuracy. Red points indicate missing grid nodes that are required by the superconvergent discretization.

1.4.5 Fall-back strategies for the superconvergent scheme

The superconvergent scheme (equations (1.8) and (1.20)) requires a certain number of valid neighboring nodes, i.e. nodes whose finite volume overlaps with the compound domain Ω, for calculating fluxes between cells. However, such a requirement may not be satisfied in some cases. Regions near sharp edges and corners (Figure 1.9 (center and right)) are prone to fail this requirement, but it can also happen in cases of smooth interfaces as well (Figure 1.9 (left)). The natural strategy in such situations is to "fall-back" to the symmetric scheme (equations (1.7), (1.14) and (1.19)), which always produces a valid discretization. While this provides a robust strategy for solving Poisson-type equations, one has to expect an increase in the numerical errors, especially the potential loss of the superconvergent properties near cells where the symmetric scheme is applied.

A simple procedure that helps to avoid such an undesirable loss in accuracy consists of merging "hanging" cells, i.e. cells that do not have enough of valid neighbors, to neighboring cells for which the requirement on the number of valid neighboring cells is satisfied. Specifically, when considering the discretization at a current cell, we first check

whether there exists a subset of its neighboring cells that are connected only to the present cell. If such a subset exists, cells substituting the subset are merged with the present cell. The described procedure eliminates "hanging" nodes as long as sharp corners of an irregular domain are not extremely acute (see Figure 1.10). Once all "hanging" nodes are merged to proper nodes the supercovergent scheme (equations (1.8) and (1.20)) is successfully applied in all boundary cells.

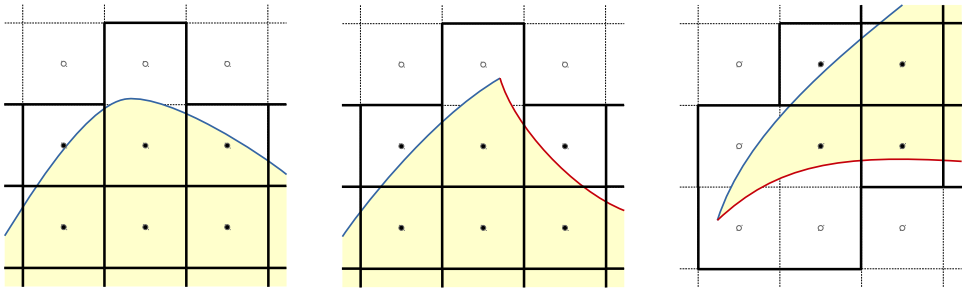


Figure 1.10: examples of successful elimination of "hanging" cells using procedure described in Section 1.4.5

This procedure can also be applied in the three-dimensional case. However, due to an increased complexity of relative placements of irregular domains with respect to computational grids, the procedure is less successful in eliminating "hanging" cells. However, from numerical experiments we have found that in practice superconvergent numerical solutions can still be obtained by ignoring values of numerical solutions at cells in which the symmetric scheme is used. It appears that the accuracy reduction due to fall-backs to the symmetric scheme is localized to cells where such fall-backs occur and does not affect the global accuracy due to the fact that the total amount of such cells is two orders of magnitude lower than the total number of cells. Accurate and superconvergent values at such cells can be obtained by extrapolating numerical solutions from surrounding

valid cells in a post-processing step. Such an extrapolation must be at least third-order accurate to retain the superconvergent properties of the numerical solutions. In sections 1.5.2 and 1.5.3, we present numerical results on this issue in two- and three-dimensional cases, respectively.

1.4.6 Non-singularity of the symmetric scheme

The resulting approximation of equation (1.3) in case of the symmetric discretization can be written in the form

$$\begin{aligned} u_{i,j}\Bigg[k(\boldsymbol{r}_{i,j})V_{i,j} + \frac{\mu_{i-\frac{1}{2},j}L_{i-\frac{1}{2},j} + \mu_{i+\frac{1}{2},j}L_{i+\frac{1}{2},j}}{\Delta x} + \frac{\mu_{i,j-\frac{1}{2}}L_{i,j-\frac{1}{2}} + \mu_{i,j+\frac{1}{2}}L_{i,j+\frac{1}{2}}}{\Delta y} \\ + (\text{add to diag})\Bigg] - u_{i-1,j}\frac{\mu_{i-\frac{1}{2},j}L_{i-\frac{1}{2},j}}{\Delta x} - u_{i+1,j}\frac{\mu_{i+\frac{1}{2},j}L_{i+\frac{1}{2},j}}{\Delta x} \\ - u_{i,j-1}\frac{\mu_{i,j-\frac{1}{2}}L_{i,j-\frac{1}{2}}}{\Delta y} - u_{i,j+1}\frac{\mu_{i,j+\frac{1}{2}}L_{i,j+\frac{1}{2}}}{\Delta y} \\ = f_{i,j}V_{i,j} + (\text{add to RHS}) + \mathcal{O}\left(h^{\mathcal{D}}\right), \end{aligned}$$

where (add to diag) and (add to RHS) denote terms that come from imposing boundary conditions. For cells that are not crossed by the domain's boundary

$$\begin{aligned} (\text{add to diag}) &= 0, \\ (\text{add to RHS}) &= 0. \end{aligned}$$

For cells crossed by the domain's boundary but without kinks and corners

$$\text{(add to diag)} = \frac{\mu(\boldsymbol{r}_{i,j}^{\hat{p}})\alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})L_{i,j}^{\hat{p}}}{\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) - \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})d_{i,j}^{\hat{p}}},$$
$$\text{(add to RHS)} = \int_{\mathcal{C}_{i,j}\cap\Gamma_{\hat{p}}} g_{\hat{p}}(\boldsymbol{r})\, d\Gamma + \frac{g(\boldsymbol{r}_{i,j}^{\hat{p}})d_{i,j}^{\hat{p}}\alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})L_{i,j}^{\hat{p}}}{\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) - \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})d_{i,j}^{\hat{p}}}.$$

Finally, for cells with sharp kinks

$$\text{(add to diag)} = \sum_{p=p_1,p_2} \int_{\mathcal{C}_{i,j}\cap\Gamma_p} \alpha_p(\boldsymbol{r})\left(1 - \boldsymbol{a}_{i,j}\cdot(\boldsymbol{r}-\boldsymbol{r}_{i,j})\right)\, d\Gamma,$$
$$\text{(add to RHS)} = \int_{\mathcal{C}_{i,j}\cap\Gamma_{\hat{p}}} g_{\hat{p}}(\boldsymbol{r})\, d\Gamma + \frac{g(\boldsymbol{r}_{i,j}^{\hat{p}})d_{i,j}^{\hat{p}}\alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})L_{i,j}^{\hat{p}}}{\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) - \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})d_{i,j}^{\hat{p}}}.$$

In case of non-negative $k(\boldsymbol{r})$ and $\{\alpha_p(\boldsymbol{r})\}_{p=0}^{n_\phi}$, and for sufficiently small grid spacings satisfying:

$$\mu(\boldsymbol{r}_{i,j}^{\hat{p}}) - \alpha_{\hat{p}}(\boldsymbol{r}_{i,j}^{\hat{p}})d_{i,j}^{\hat{p}} > 0 \quad \text{and} \quad (1 - \boldsymbol{a}_{i,j}\cdot(\boldsymbol{r}-\boldsymbol{r}_{i,j})) > 0, \tag{1.21}$$

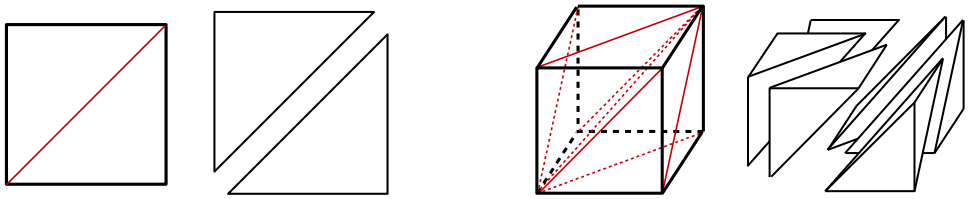
the sum $(k(\boldsymbol{r}_{i,j})V_{i,j} + \text{(add to diag)})$ is non-negative for all cells, which makes the matrix associated with the linear system diagonally dominant. Moreover, by construction the directed graph of the matrix is strongly connected[2], hence the matrix is irreducible. Finally, assuming that at least for one cell $(k(\boldsymbol{r}_{i,j})V_{i,j} + \text{(add to diag)})$ is positive (otherwise, the equation being solved is the Poisson equation with Neumann boundary conditions, which has infinitely many solutions), we conclude that the linear system is non-singular [169, Theorem 1.21]. We have found that in practice, even when the grid resolution does not satisfy the condition (1.21), the linear system is still invertible.

[2]We assume that the solution domain does not contain disjoint sets, otherwise the solution is independent on disjoint sets and the same reasoning should be applied to each isolated area.

The question of whether the linear system remains non-singular in the case of arbitrary $k(\boldsymbol{r})$ and $\{\alpha_p(\boldsymbol{r})\}_{p=0}^{n_\phi}$ (excluding identically zero case) is still open, however in our numerical experiments we never encountered situations when the linear system was not invertible. Likewise, we do not prove the non-singularity of the linear system in the nonsymmetric case; however a large number of numerical tests indicate that resulting linear systems are invertible. In our implementation we use the biconjugate gradient stabilized method (BiCGSTAB) provided by PETSc library [9] and the HYPRE multigrid preconditioner [45] for solving the linear systems. We choose to use the BiCGSTAB method even for the symmetric discretization because we are interested, first, in cases when functions $k(\boldsymbol{r})$ and $\{\alpha_p(\boldsymbol{r})\}_{p=1}^{n_\phi}$ may be negative (which result in a non-positive definite matrix) and, second, in applying this discretization in the context of adaptive Quad-/Oc-tree grids along the lines of [129] (which produce non-symmetric matrices). However, we note that in case of uniform grids and non-negative coefficients $k(\boldsymbol{r})$ and $\{\alpha_p(\boldsymbol{r})\}_{p=1}^{n_\phi}$, the preconditioned conjugate gradient method would be used to efficiently solve the linear system produced by the symmetric discretization.

1.4.7 Computation of integrals

The numerical schemes presented to solve (1.1)-(1.2) require the calculation of integrals over compound domains that may have boundaries with kinks. Moreover, one needs to calculate integrals with at least second-order accuracy in the case of the symmetric scheme and with at least third-order accuracy in the case of the superconvergent scheme.



(a) Two spatial dimensions. (b) Three spatial dimensions.

Figure 1.11: The Kuhn triangulation of a rectangular cell.

As it was mentioned in the introduction, only geometric reconstruction methods are readily extendable to integration over geometries with sharp features. Geometric reconstruction methods that admits a relatively simple extension to the case of domains described by multiple level-set functions fall under the class of simplex-based methods (see [99] for linear and [50, 51] for higher-order reconstructions). In such approaches, a cell in which integration has to be performed is first decomposed into simplices (triangles in 2D or tetrahedra in 3D - see Figure 1.11 for an example of the Kuhn triangulation of a rectangular cell) and then the zero-isocontour of a level-set function is reconstructed inside each of the simplices. Provided that an interface does not cross the edges or the faces of a simplex multiple times, one has to take care of just a few (two in 2D and three in 3D) non-trivial topologically different cases, as illustrated in Figures 1.12 and 1.13. Once an interface is reconstructed, the integration can be performed by mapping Gauss quadrature points from reference elements to reconstructed, possibly curved, geometric elements.

What makes simplex-based approaches easy to extend to the case of crossing interfaces, is the fact that the reconstruction of an interface within a simplex leads to a splitting of the initial simplex into a set of smaller simplices (see Figures 1.12 and

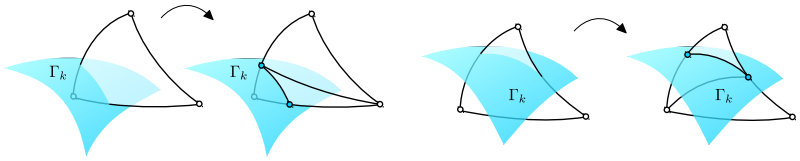


Figure 1.12: The two topologically different possible cases of a simplex crossed by an interface in 2D.

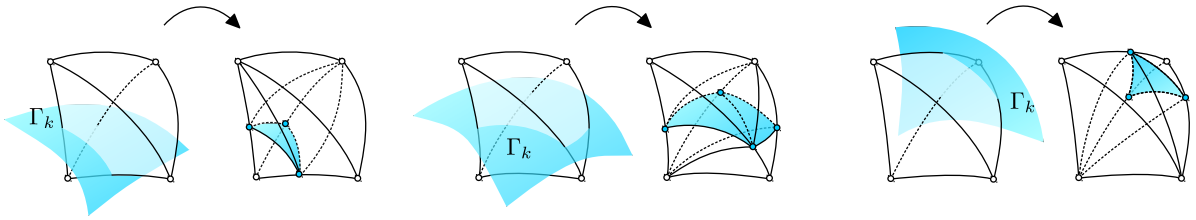


Figure 1.13: The three topologically different possible cases of crossing a simplex by an interface in 3D.

1.13). Repeating the same reconstruction procedure on each of those smaller simplices for another interface results in reconstructing an interface described by a combination of the two intersecting interfaces. Figure 1.14 illustrates the idea in 2D. Since each of the smaller simplices are split into sets of even smaller simplices during the reconstruction of the second interface, it is clear that such an approach allows for the reconstruction of any number of intersecting interfaces.

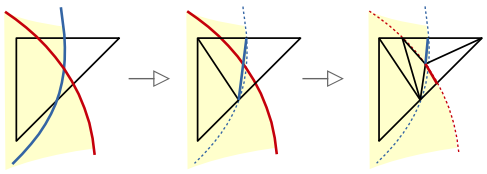


Figure 1.14: Geometric integration applied to compound domains.

Such an idea of reconstructing domains produced by several intersecting interfaces was used in [113] for linear geometric reconstructions. The same idea was discussed in [51] in case of high-order elements. However, neither [113] or [51] have studied the

accuracy of integration procedures over piecewise smooth geometries. We present a hierarchical approach to reconstruct multiple interfaces within a simplex that is based on representing a simplex as a collection of geometric elements such as vertices, edges, triangles and tetrahedra (which can be regarded as 0-simplex, 1-simplex, 2-simplex and 3-simplex) and performing the reconstruction of an interface on each of the geometric type separately in the order of increasing dimensionality, i.e. vertices $\rightarrow$ edges $\rightarrow$ triangles $\rightarrow$ tetrahedra. Such an approach simplifies the task of reconstruction rather significantly, since when a geometric element is considered, all lower-dimensional elements have already been processed. Another beneficial feature of this approach is that it avoids the potential duplication of geometric elements, which not only saves computational resources, but also enables integrations over intersections of interfaces. We also note that this approach is suitable for reconstructing interfaces with geometric elements of any order. In the present work, we report results for linear and quadratic geometric elements. Due to a rather technical character of the integration algorithm, we provide a detailed description in 1.A.

It is worth emphasizing that such an integration algorithm does not require any explicit information about the location of kinks and corners; such features are automatically reconstructed with needed accuracy from implicit representation of domains by multiple level-set functions. It is also capable of performing integration over distinct parts of piece-wise smooth interfaces.

When solving the Poisson equation we use integration based on reconstruction with

quadratic geometric elements for both the symmetric and the superconvergent schemes, although we note that for the symmetric scheme it would be sufficient to use an integration scheme based on linear reconstruction.

1.5 Numerical Results

This section is devoted to numerical examples. In Section 1.5.1, we provide numerical examples for the integration algorithm in both in two and three spatial dimensions. In sections 1.5.2 and 1.5.3, we numerically study the convergence of the numerical schemes for Poisson-type equations (1.1) in two and three spatial dimensions, respectively. In Section 1.5.4, we investigate the requirement of the methods on the accuracy of the level-set representation. Finally, in Section 1.5.5 we test the presented methods on singular solutions that have limited number of bounded derivatives.

To numerically analyze the accuracy of the numerical schemes for solving Poisson-type equations, we compute the errors in the L^∞-norm for the solutions and their gradients. Gradients of numerical solutions are computed using standard second-order accurate central finite differences. In addition, we also plot the error distributions of both methods for sake of comparison.

1.5.1 Integration

It is standard knowledge that a circular domain in two spatial dimensions can be defined by the level-set function:

$$\phi(x, y) = \sqrt{(x - x_c)^2 + (y - y_c)^2} - r,$$

while the level-set function of a spherical domain can be defined as:

$$\phi(x, y, z) = \sqrt{(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2} - r,$$

where r is the radius of the disk/sphere and (x_c, y_c) and (x_c, y_c, z_c) are the centers' coordinates. We use those level-set functions to build compound domains (see Table 1.1) that are then used to test the integration methods.

example	**Geometry**	Radius 1	Center 1	Radius 2	Center 2
Union 2D	Two disks	$r_1 = 0.77$	$(0.13, 0.21)$	$r_2 = 0.49$	$(-0.33, -0.37)$
Difference 2D	Two disks	$r_1 = 0.84$	$(0.03, 0.04)$	$r_2 = 0.63$	$(-0.42, -0.37)$
Union 3D	Two spheres	$r_1 = 0.71$	$(0.22, 0.17, 0.21)$	$r_2 = 0.63$	$(-0.19, -0.19, -0.23)$
Difference 3D	Two spheres	$r_1 = 0.86$	$(0.08, 0.11, 0.03)$	$r_2 = 0.83$	$(-0.51, -0.46, -0.63)$

Table 1.1: Geometries used in the numerical tests for the integration procedures.

For each of the compound domains, we compute the following three types of integrals in two ($\mathcal{D} = 2$) and three ($\mathcal{D} = 3$) spatial dimensions:

- The integral over the compound domain:

$$I_\Omega[f] = \int_\Omega f \, d\boldsymbol{x}^{\mathcal{D}}$$

- The integrals over the smooth components of the compound domain boundary:

$$I_{\Gamma_i}[f] = \int_{\Gamma_i} f \, d\boldsymbol{x}^{\mathcal{D}-1}, \quad i = 1, 2$$

- The integral over the intersection of the smooth components of the boundary (case of kinks):

$$I_{X_1}[f] = \int_{\Gamma_1 \cap \Gamma_2} f \, d\boldsymbol{x}^{\mathcal{D}-2}$$

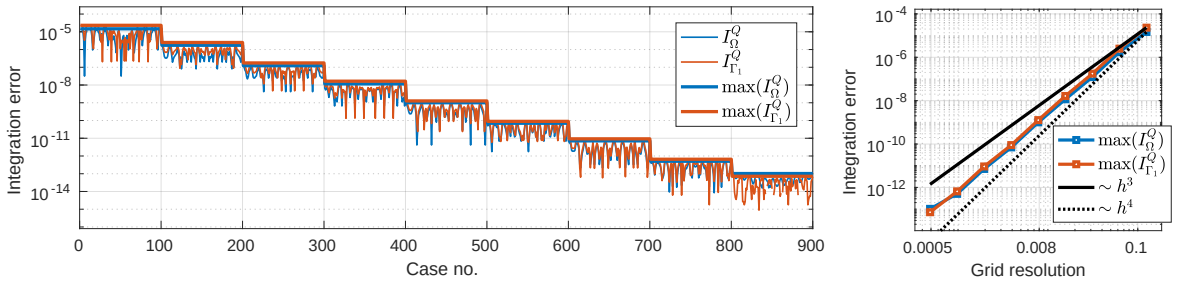


Figure 1.15: Left: dependence of the error in computing the area and perimeter of a circle (with radius $R = 0.75$) on the relative placement of the irregular domain and the computational grids (cases 1-100, 101-200, ..., 801-900 are obtained by shifting boundaries of computational domain for grid resolutions 16^2, 32^2, ..., 4096^2, respectively). Right: convergence of the error in the area and perimeter of the circle measured as the maximum error for a given grid resolution.

During the numerical approximation of domain and boundary integrals, certain relative placements of irregular domains and computational grids may cause cancellations of errors in the numerical values of such integrals. For example, Figure 1.15 (left) depicts the errors in the area and the perimeter length of a circle with radius $R = 0.75$ and center $(0, 0)$, computed using quadratic geometric reconstruction for grid resolutions 16^2, ..., 4098^2, where for each of the grid resolutions, boundaries of the $[-1; 1]^2$ computational domain are slightly shifted $n_x = 10$ and $n_y = 10$ times in the x- and y-directions by amounts $\Delta x/n_x$ and $\Delta y/n_y$, respectively. This procedure generates $n_x \times n_y$ different relative placements of the irregular domain with respect to the computational grid for each of the grid resolutions, where in each of the cases the computational domain is described

by:

$$\Omega_{I,J} = [-1+\varepsilon_x; 1+\varepsilon_x] \times [-1+\varepsilon_y; 1+\varepsilon_y], \quad \varepsilon_x = \frac{I-1}{n_x}\Delta x, \quad \varepsilon_y = \frac{J-1}{n_y}\Delta y,$$
$$I = 1, \ldots, n_x, \quad J = 1, \ldots, n_y. \quad (1.22)$$

As seen from Figure 1.15 (left) the accuracy of the approximate numerical values of the area and of the perimeter oscillates quite significantly for any given resolution depending on the relative placement of the irregular domain with respect to the computational grid. Thus, to avoid untrue apparent experimental convergence rates and obtain more reliable results, we measure the integration error for a given grid resolution as the *maximum* error among all relative placements of the computational grid and an irregular domain obtained by shifting boundaries of the computational domain 10×10 times in two spatial dimensions and $5 \times 5 \times 5$ times in three spatial dimensions. Convergence results obtained in this way for the present example of a circular domain are given in Figure 1.15 (right). Interestingly, although integral quantities are expected to converge with third order of accuracy for quadratic geometric reconstructions, apparently due to cancellations of errors in the integral sums for entire shapes, the experimental convergence rates are almost one order higher and close to fourth. In [50], such effect was observed for even orders of geometric reconstruction higher than quadratic in case of smooth geometries.

Numerical values of the integrals for the examples listed in Table 1.1 are computed twice: first, when the interfaces are reconstructed using linear (L) geometric elements and, second, when the interfaces are reconstructed using quadratic (Q) geometric elements. Convergence results for all test cases along with the visualization of the geomet-

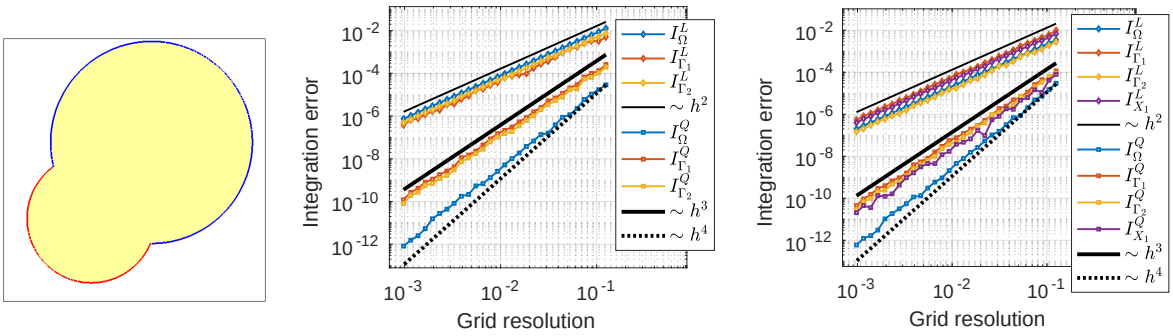


Figure 1.16: Compound domain (left) and convergence in the L^∞-norm of the computed integrals in the cases $f = 1$ (center) and $f = |\boldsymbol{x}|^2$ (right) for the union of two disks. The superscripts L and Q denote quantities computed using linear and quadratic geometric reconstructions, respectively.

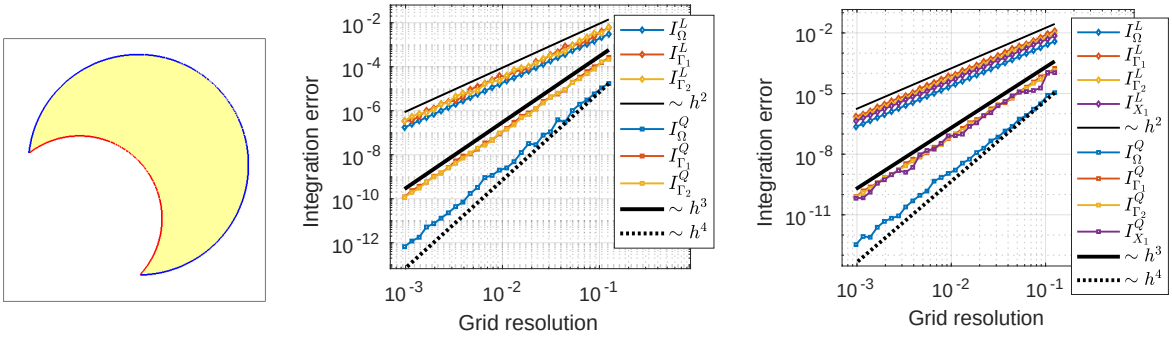


Figure 1.17: Compound domain (left) and convergence in the L^∞-norm of the computed integrals in the cases $f = 1$ (center) and $f = |\boldsymbol{x}|^2$ (right) for the difference of two disks. The superscripts L and Q denote quantities computed using linear and quadratic geometric reconstructions, respectively.

ric reconstruction of the compound domains produced by the integration algorithm are provided in Figures 1.16, 1.17 1.18 and 1.19.

Numerical results indicate that the integration procedure based on reconstructions of implicit interfaces with linear geometric elements produces second-order accurate solutions for all types of integrals both in two and three spatial dimensions, as expected. In case of quadratic geometric reconstructions we again observe a higher than third, expected, order of convergence. However, unlike the case of entirely smooth shapes,

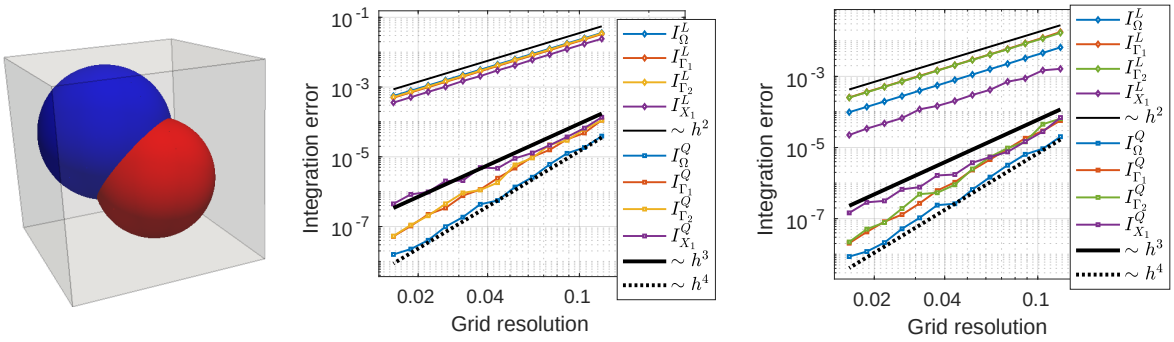


Figure 1.18: Compound domain (left) and convergence in the L^∞-norm of the computed integrals in the cases $f = 1$ (center) and $f = |\boldsymbol{x}|^2$ (right) for the union of two spheres. The superscripts L and Q denote quantities computed using linear and quadratic geometric reconstructions, respectively.

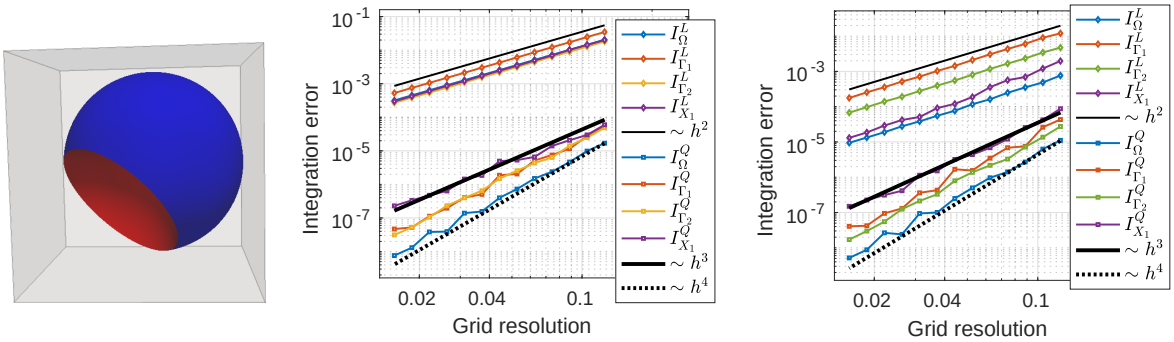


Figure 1.19: Compound domain (left) and convergence in the L^∞-norm of the computed integrals in the cases $f = 1$ (center) and $f = |\boldsymbol{x}|^2$ (right) for the difference of two spheres. The superscripts L and Q denote quantities computed using linear and quadratic geometric reconstructions, respectively.

experimental convergence rates for piecewise smooth shapes are not exactly one order higher than expected and vary between third and fourth depending on the type of integral and dimensionality. For purposes of presented discretizations of Poisson-type equations it is not of big importance, first, because integration methods are required to be just at least third order accurate and, second, because accuracy of integration within independent cells, not for entire shapes, is what matters most for discretization in finite volume methods.

1.5.2 Two spatial dimensions

In all the numerical examples, we consider a computational grid with size $[-1, 1]^{\mathcal{D}}$ in $\mathcal{D}$ spatial dimensions.

Triangular domain

Consider a triangular domain generated by the intersection of three half-spaces such that the corners of the domain are located at $(0.74, -0.86)$, $(-0.83, -0.11)$ and $(0.37, 0.87)$ (see Figure 1.20 (left)). A "half-space" domain can be describe by a level-set function

$$\phi(x, y) = n_x(x - x_0) + n_y(y - y_0),$$

where the vector (n_x, n_y) defines the normal to the domain's boundary and (x_0, y_0) is a point lying on the domain boundary. We take the exact solution to be $u = \sin(x)\cos(y)$, the diffusion coefficient $\mu = 1$, the function $k = 0$ (in equation 1.1) and the Robin coefficients $\alpha_1 = 1$, $\alpha_2 = 0$ and $\alpha_3 = x - y + (x + y)^2$ for the right, bottom and left boundaries, respectively. The compound domain and the numerical solution on a 256^2 grid are shown in Figures 1.20a and 1.20b.

As for the integration tests, we analyze the sensitivity of the proposed numerical schemes on the relative placement of the irregular domains and the computational grids by performing shifts of boundaries of the computational domain (see Section 1.5.1): for each grid resolution we consider $n_x \times n_y$ perturbed computational domains given by equation (1.22), where $n_x = n_y = 10$. Figures 1.21 (left) and 1.22 (left) give the L^∞-norm of the numerical error of the solutions and its gradients, respectively, obtained for each con-

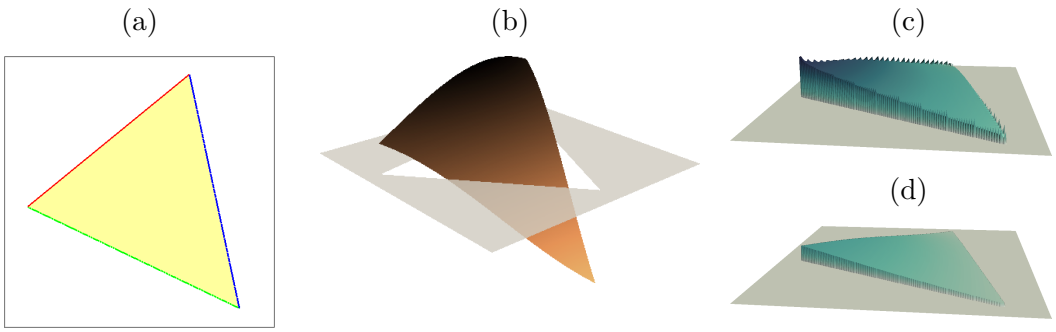


Figure 1.20: Compound domain (a), numerical solution (b) and localization of the error in the case of the symmetric (c) and the superconvergent ("SC 2") (d) schemes on a 256^2 grid for example 1.5.2.

sidered relative placements for grid resolutions 16^2, 32^2, ..., 1024^2. Figures 1.21 (right) and 1.22 (right) demonstrate the convergence of the numerical solution and its gradient, where the error for each grid resolution is measured as the *maximum* error among all relative placements of the irregular domain and the computational grid considered.

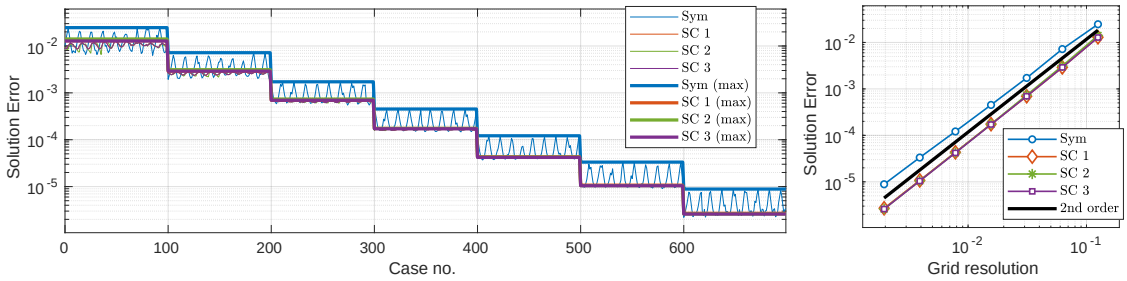


Figure 1.21: Left: Dependence of the L^∞-norm of the solution error on the relative placement of an irregular domain and a computational grid for grid resolutions 16^2, 32^2, ..., 1024^2. Right: Convergence of numerical solution in the L^∞ where the error is measured as the maximum error among all considered relative placements.

The numerical results indicate that the symmetric numerical scheme produces second-order accurate numerical solutions and first-order accurate gradients. Figure 1.20c depicts the distribution of the error in the numerical solution for the symmetric scheme on a 256^2 grid. Clearly, the behavior of the error near the domain boundary is discontinuous with

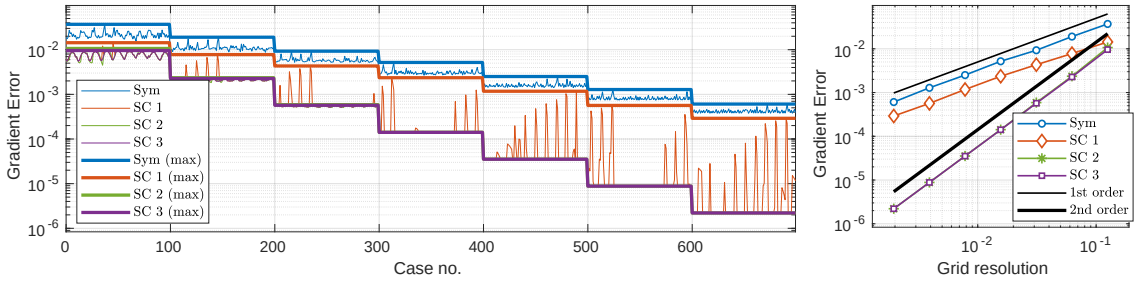


Figure 1.22: Left: Dependence of the L^∞-norm of the error of the gradients on the relative placement of an irregular domain and the computational grid for grid resolutions 16^2, 32^2, ..., 1024^2. Right: Convergence of the gradients in the L^∞-norm where the error is measured as the maximum error among all considered relative placements.

many spikes, which results in the loss of one order of accuracy during differentiation when computing the gradients.

For the nonsymmetric superconvergent scheme, we consider three different strategies for dealing with cases where the superconvergent discretization is not possible for some computational cells (see Section 1.4.5):

1. In cells where the superconvergent discretization is not possible, the symmetric scheme is applied (denoted as "SC 1");

2. Cells where the superconvergent discretization is not possible are attached to valid neighboring cells (denoted as "SC 2");

3. Same as strategy no. 1, but values of numerical solutions at cells in which symmetric scheme is applied are discarded after the linear system has been solved (denoted as "SC 3").

As expected, when the superconvergent treatment is not possible at a particular grid point, the simple switch to the symmetric scheme (strategy 1) may result in the

loss of the second-order accuracy in gradients. In this case, the error of the gradients near such a grid point spikes up to the level of the error produced by fully symmetric scheme. We have found that strategies 2 and 3 produce very similar results in two spatial dimensions; for both approaches the gradients of numerical solutions are second-order accurate. Figure 1.20c depicts the distributions of the error in the numerical solution for the superconvergent scheme (strategy 2). Compared to the symmetric scheme, the error is smooth everywhere including near the boundary of the irregular domain (in the sense that the numerical differentiation of such an error field does not produce $\mathcal{O}(1/h)$ terms), which leads to the preservation of the order of accuracy of the gradients.

Figure 1.23 shows the dependence of the condition number of the resulting linear systems for the presented methods on the relative placement of the solution domain and computational grid and the grid resolution. As one can see the condition numbers for all considered methods are very close to each other and show little sensitivity to the solution domain placement. The magnitude of the condition numbers demonstrates the inverse proportionality to the second power of the mesh size h.

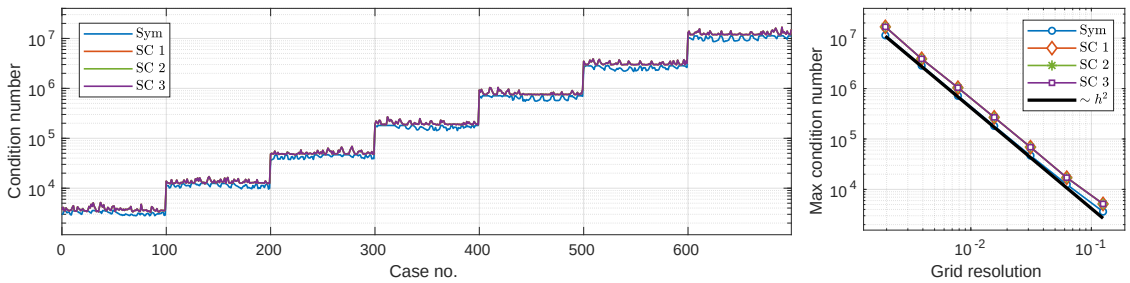


Figure 1.23: Left: Dependence of the condition number of the linear system on the relative placement of an irregular domain and the computational grid for grid resolutions 16^2, ..., 1024^2. Right: Dependence of the maximum condition number among all considered relative placements on the grid resolution.

In the following two-dimensional examples the same perturbation procedure of the computational domains is performed. However, only the maximum errors and the maximum condition number for a given grid resolution are reported. Also, we report only the results produced by strategy no. 2 for the superconvergent scheme, since it is a more comprehensive approach that does not require any additional post-processing steps.

Union of two disks

We consider the compound domain to be the union of two disks with radii $r_1 = 0.77$ and $r_2 = 0.49$ that are located at $(0.13, 0.21)$ and $(-0.33, -0.37)$ (see Figure 1.24a). The test function is $u = 2\log\left((x + 0.8y)^2 + x - 0.7y + 4\right) - 3$. The parameters in equation (1.1) are $\mu = 1$ and $k = 0$ (Poisson equation). The Robin coefficients are $\alpha_1 = 1$ and $\alpha_2 = \sin(x + y)\cos(x - y)$ for the larger and smaller disks, respectively.

The convergence results are presented in Figure 1.25. The numerical solution and the localization of the error computed on a 256^2 grid are plotted in Figures 1.24b, 1.24c and 1.24d. Concerning the order of accuracy, behavior of the condition number and the location of errors in the case of the symmetric and the superconvergent schemes, one can draw the same conclusions as in the previous example, with the exception for convergence of gradients. In this particular example, the convergence rate for the solution's gradient is found to be 1.75, i.e. slightly worse than 2. This may be related to the fact that acute inner kinks are more prone to cause singular behavior of solutions.

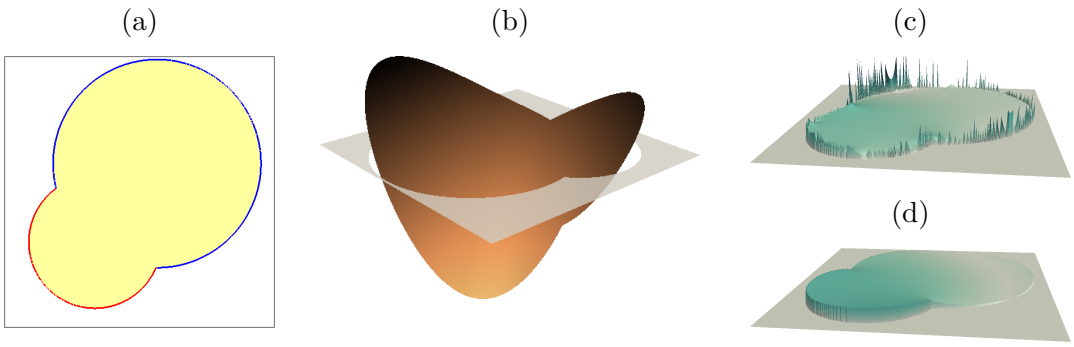


Figure 1.24: Compound domain (a), numerical solution (b) and localization of the error in the case of the symmetric (c) and the superconvergent (d) discretizations on a 256^2 grid for example 1.5.2.

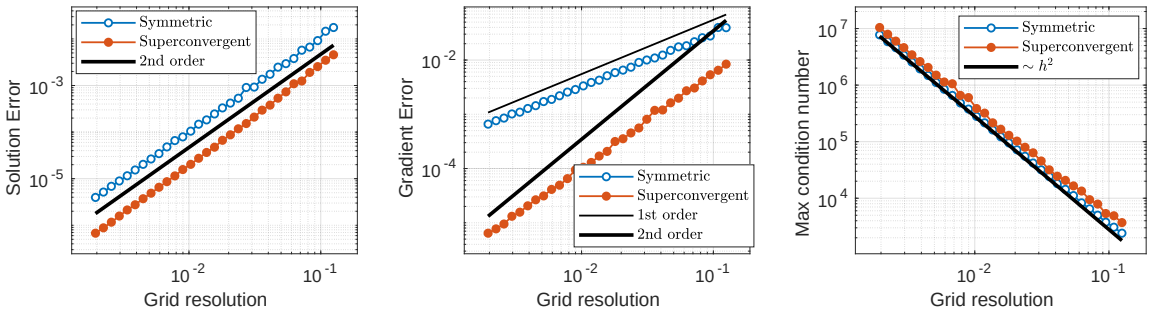


Figure 1.25: Convergence of the numerical solution (left) and its gradient (center) in the L^∞-norm and dependence of the condition number on grid resolution (right) for example 1.5.2.

Difference of two disks

We consider the compound domain to be the intersection of a disk of radius $r_1 = 0.84$ located at $(0.03, 0.04)$ and the exterior of a disk of radius $r_2 = 0.63$ located at $(-0.42, -0.37)$ (see Figure 1.26a). The exact solution is taken to be $u = 4\log\left(\dfrac{0.7x+3}{y+3}\right)\sin(x+0.5y)$. The diffusion coefficient in this example is variable $\mu = 1 + 0.2\sin(x) + 0.3\cos(y)$ and $k = 1$. The Robin coefficients are $\alpha_1 = \sin(x-y)\cos(x+y)$ and $\alpha_2 = \sin(x+y)\cos(x-y)$ for the convex and concave parts of the boundary, respectively.

The convergence results are presented in Figure 1.27. The numerical solution and

the localization of the error computed on a 256^2 grid are plotted in Figures 1.26b, 1.26c and 1.26d. Concerning the order of accuracy, the behavior of the condition number and the location of errors in the case of the symmetric and the superconvergent schemes, the conclusions of the previous examples hold.

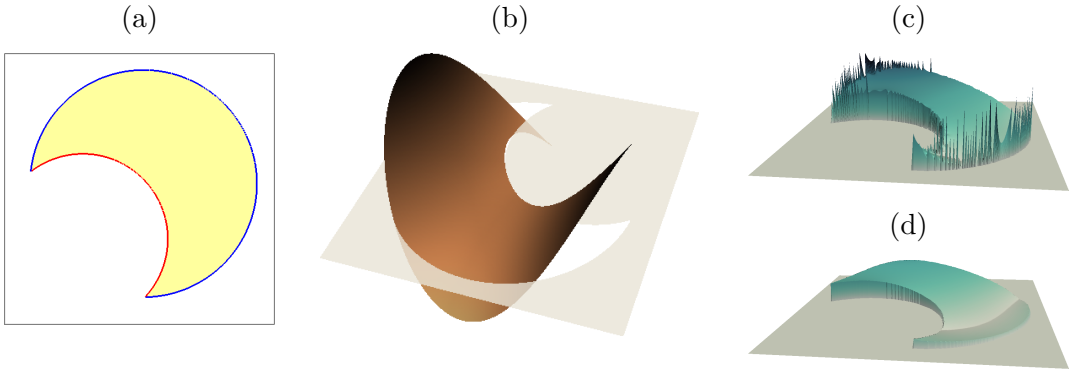


Figure 1.26: Compound domain (a), numerical solution (b) and localization of the error in the case of the symmetric (c) and the nonsymmetric (d) discretizations on a 256^2 grid for example 1.5.2.

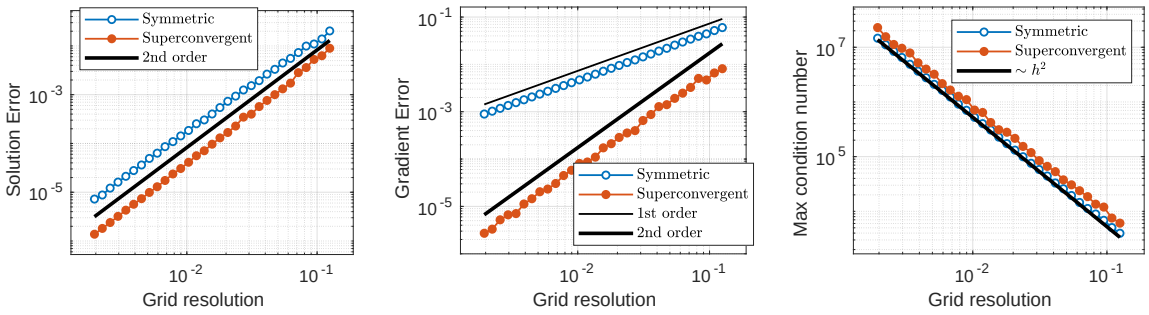


Figure 1.27: Convergence of the numerical solution (left) and its gradient (center) in the L^∞-norm and dependence of the condition number on grid resolution (right) for example 1.5.2.

Three flower-shaped domains

We consider a compound domain generated by three flower-shaped subdomains that are defined by the level-set functions:

$$\phi_p = s_p \left(\sqrt{X_p^2 + Y_p^2} - r_p + \beta_p \frac{X_p^2 + 5X_p^4 Y_p - 10X_p^2 Y_p^3}{(X_p^2 + Y_p^2)^{\frac{5}{2}}} \right), \quad p = 1, 2, 3,$$

where

$$\begin{pmatrix} X_p \\ Y_p \end{pmatrix} = \begin{pmatrix} \cos\theta_p & -\sin\theta_p \\ \sin\theta_p & \cos\theta_p \end{pmatrix} \begin{pmatrix} x - x_p \\ y - y_p \end{pmatrix}.$$

The parameters x_p and y_p define the center of mass of the p^{th} shape. The parameter θ_p defines the angle of rotation of the p^{th} subdomain around its center. The parameter $s_p = \pm 1$ is a toggle switch controlling whether we consider the interior or the exterior of a flower-shaped domain. Finally, the parameter β_p determines the degree of non-sphericity of a shape. The resulting compound domain is shown in Figure 1.28a. All the parameters defining the subdomains are listed in Table 1.2. Table 1.2 also contains the set operators that are used to construct the compound domain as well as the Robin coefficients imposed on the different parts of its boundary. We consider the test function $u = \sin(x + 0.3y)\cos(x - 0.7y) + 3\log(\sqrt{x^2 + y^2 + 0.5})$, the diffusion coefficient $\mu = 1 + 0.2\sin(x) + 0.3\cos(y)$ and $k = \sin(x)\exp(y)$.

p	r_p	x_p	y_p	β_p	s_p	Rotation θ_p	Set operator	Robin coeff. α_p
1	0.73	0.13	0.16	0.08	1	0.1π	-	1
2	0.66	-0.14	-0.21	-0.08	1	-0.2π	`union`	$1 + \sin(x)\cos(y)$
3	0.59	0.45	-0.53	-0.08	-1	0.2π	`intersection`	$\exp(x + y)$

Table 1.2: Shape parameters for example 1.5.2.

The convergence results are presented in Figure 1.29. The numerical solution and the localization of the error for a 256^2 grid are plotted in Figures 1.28b, 1.28c and 1.28d.

Concerning the order of accuracy, the behavior of the condition number and the location of errors in the case of the symmetric and the superconvergent schemes, one can draw the same conclusions as in the previous examples.

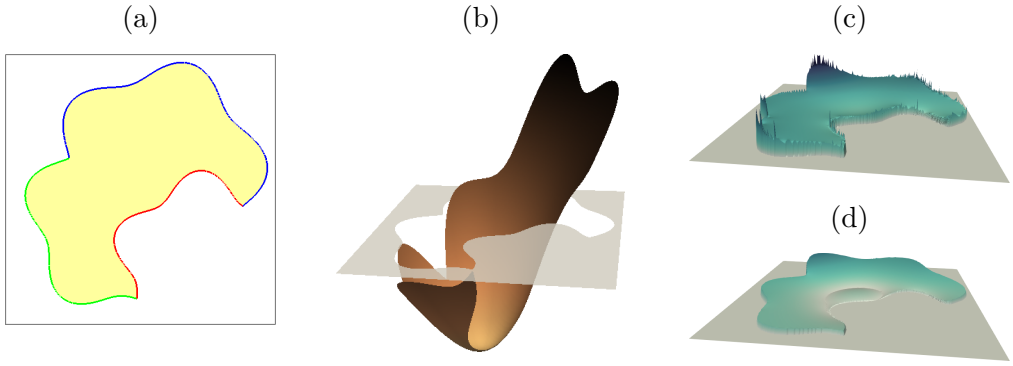


Figure 1.28: Compound domain (a), numerical solution (b) and localization of the error in the case of the symmetric (c) and the nonsymmetric (d) discretizations on a 256^2 grid for example 1.5.2.

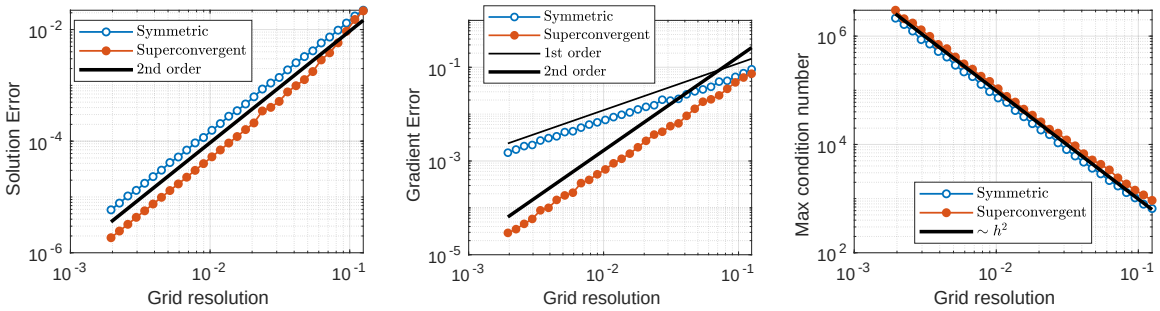


Figure 1.29: Convergence of the numerical solution (left) and its gradient (center) in the L^∞-norm and dependence of the condition number on grid resolution (right) for example 1.5.2.

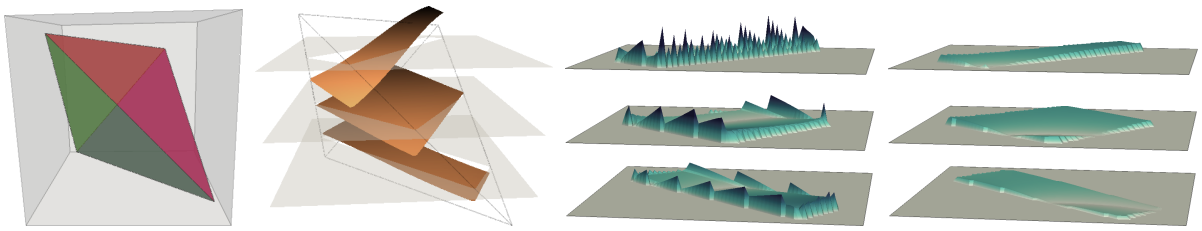
1.5.3 Three spatial dimensions

Tetrahedral domain

As in the two dimensional case, we start with a tetrahedral domain produced by intersecting four half-planes such that the corners of the tetrahedron are located at $\boldsymbol{r}_1 = (-0.86, -0.87, -0.83)$, $\boldsymbol{r}_2 = (0.88, -0.52, 0.63)$, $\boldsymbol{r}_3 = (0.67, 0.82, -0.87)$ and $\boldsymbol{r}_4 = (-0.78, 0.73, 0.85)$. We take the diffusion coefficient $\mu = 1$, and $k = 0$. The Robin coefficients are:

$$\begin{aligned}
\alpha_1 &= 1, \\
\alpha_2 &= 0, \\
\alpha_3 &= x - y + (x + y)^2, \\
\alpha_4 &= x + y,
\end{aligned}$$

and are imposed on the faces of the tetrahedron with vertices $(\boldsymbol{r}_0, \boldsymbol{r}_1, \boldsymbol{r}_2)$, $(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$, $(\boldsymbol{r}_0, \boldsymbol{r}_2, \boldsymbol{r}_3)$ and $(\boldsymbol{r}_0, \boldsymbol{r}_1, \boldsymbol{r}_3)$, respectively. We take $u = \sin(x)\cos(y)\exp(z)$. A visualization of the compound domain and of the numerical solution is given in Figures 1.30a and 1.30b.



(a) (b) (c) (d)

Figure 1.30: Compound domain (a), numerical solution (scaled by .3) (b) and localization of the error in case of the symmetric (c) and the superconvergent ("SC 3") (d) schemes in planes $z = 0$ and ± 0.5 on a 64^3 grid for example 1.5.3.

The sensitivity of the numerical schemes to the relative placement of the irregular domain with respect to the computational grid is studied in the same fashion as in two dimensional case by perturbing the boundaries of the computational domain $n_x \times n_y \times n_z$ times, where $n_x = n_y = n_z = 5$. Figures 1.31 (left) and 1.32 (left) report the L^∞-norms of the errors of the numerical solution and its gradients for all of the perturbed domains considered. Figures 1.31 (right) and 1.32 (right) depict the convergence of the numerical solution and its gradient field. Figure 1.33 depicts the dependence of the condition number on the domain placement and grid resolution. All three strategies described in Section 1.5.2 for dealing with exceptional cells in case of the superconvergent scheme are considered. Convergence results are almost entirely identical to the analogous example in two spatial dimensions, with the exception of strategy no. 2 for the superconvergent scheme. Due to an increased complexity of intersections of irregular domains with finite volume cells in three dimensions compared to two-dimensional case, the algorithm proposed in Section 1.4.5 does not always successfully eliminate "hanging" nodes in which case the less accurate symmetric discretization has to be used. As a result, only strategy no. 3 produces superconvergent results. The error of the numerical solution for the symmetric and the superconvergent (strategy 3) schemes are plotted in Figures 1.30c and 1.30d.

In the following examples, the same perturbation procedure is performed but the maximum errors and condition numbers are only reported for a given grid resolution. In addition, we only report the results produced by strategy no. 3 for the superconvergent

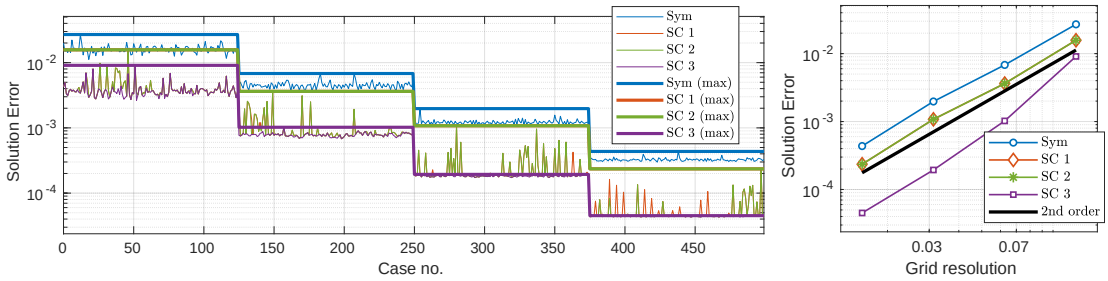


Figure 1.31: Left: dependence of the L^∞-norm of the solution error on the relative placement of an irregular domain with respect to the computational grid for grid resolutions 16^3, ..., 128^3. Right: convergence of numerical solution in the L^∞-norm where the error is measured as the maximum error among all considered relative placements.

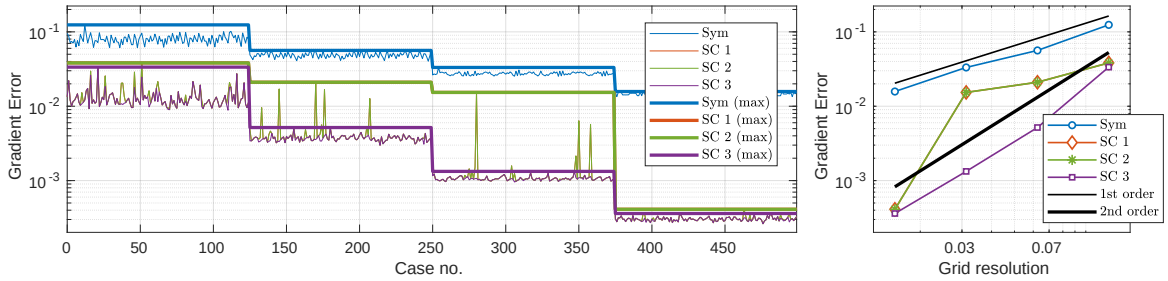


Figure 1.32: Left: dependence of the L^∞-norm of the error in gradients on the relative placement of an irregular domain with respect to the computational grid for grid resolutions 16^3, ..., 128^3. Right: convergence of gradients in the L^∞-norm where the error is measured as the maximum error among all considered relative placements.

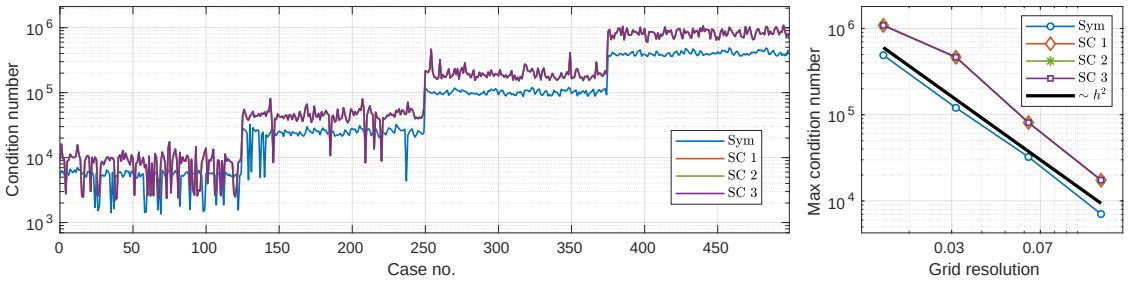


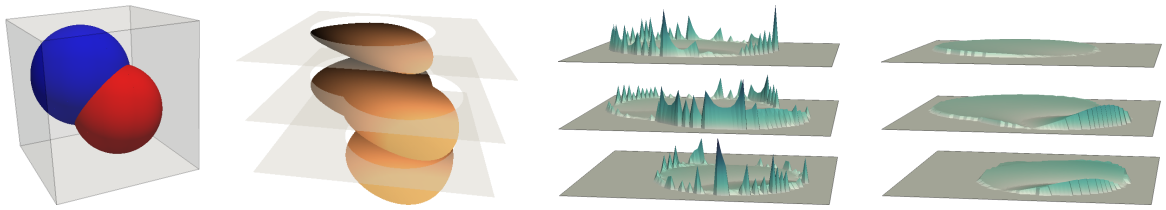
Figure 1.33: Left: Dependence of the condition number of the linear system on the relative placement of an irregular domain and the computational grid for grid resolutions 16^2, ..., 128^2. Right: Dependence of the maximum condition number among all considered relative placements on the grid resolution.

scheme.

Union of two spheres

We consider a compound domain obtained by the union of two spheres with radii $r_1 = 0.71$ and $r_2 = 0.63$, and centers $(0.22, 0.17, 0.21)$ and $(-0.19, -0.19, -0.23)$, respectively. The parameters of the Poisson equation are $\mu = 1$ and $k = 0$. The Robin coefficients are $\alpha_1 = 1$ and $\alpha_2 = \sin(x+y)\cos(x-y)\log(z+4)$ for the larger and the smaller spheres, respectively. The exact solution is $u = 2\log(x + 0.5y - 0.3z + 3 + (x - 0.7y - 0.9z)^2) - 3$.

Visualizations of the compound domain, numerical solution and error distributions on a 64^3 grid are plotted in Figure 1.34. Figure 1.35 shows the results of the convergence analysis. Again, one can draw the same conclusions about the rates of convergence as in the previous examples.



(a) (b) (c) (d)

Figure 1.34: Compound domain (a), numerical solution (scaled by .3) (b) and localization of the error in case of the symmetric (c) and the superconvergent ("SC 3") (d) schemes in planes $z = 0$ and ± 0.5 on a 64^3 grid for example 1.5.3.

Difference of two spheres

The compound domain is the intersection of a sphere of radius $r_1 = 0.86$ with center located at $(0.08, 0.11, 0.03)$ with the exterior of a sphere of radius $r_2 = 0.83$ with center located at $(-0.51, -0.46, -0.63)$. We choose the variable diffusion coefficient as $\mu = 1 + (0.2\sin(x) + 0.3\cos(y))\cos(z)$ and set $k = 1$. The Robin coefficients are $\alpha_1 = \cos(x +$

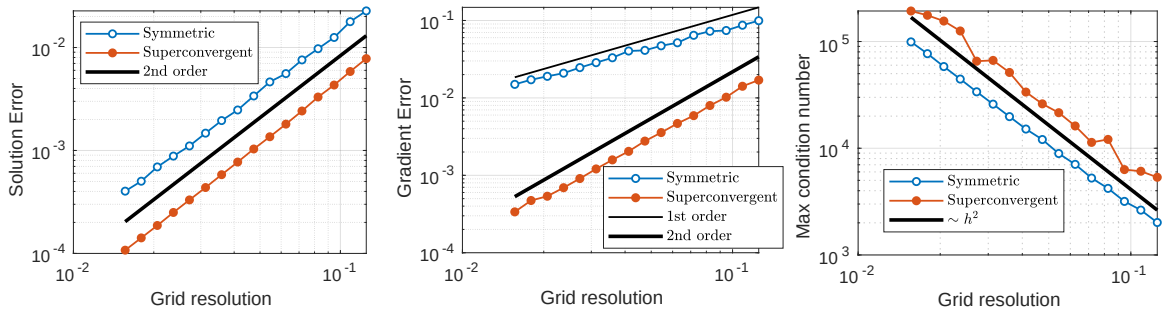
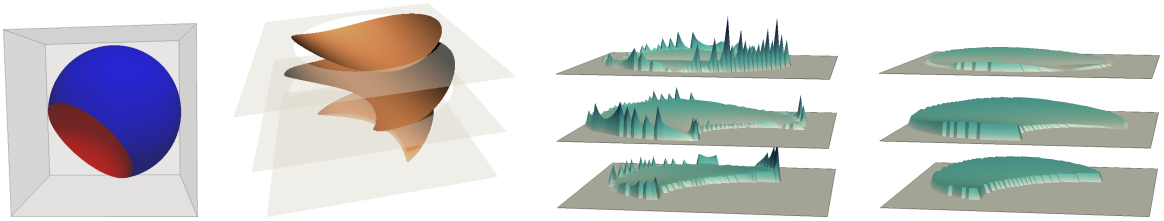


Figure 1.35: Convergence of the numerical solution (left) and its gradient (center) in the L^∞-norm and dependence of the condition number on grid resolution (right) for example 1.5.3.

$y)\sin(x-y)\exp(z)$ and $\alpha_2 = \sin(x+y)\cos(x-y)\log(z+4)$ for the convex and the concave parts of the boundary, respectively. The test function is $u = 4\log\left(\dfrac{x+y+3}{y+z+3}\right)\sin(x + 0.5y + 0.7z)$.

Visualizations of the compound domain, numerical solution and error distributions on a 64^3 grid are plotted in Figure 1.36. Figure 1.37 shows the results of the convergence analysis. One can draw the same conclusions about the rates of convergence as in the previous examples.



(a) (b) (c) (d)

Figure 1.36: Compound domain (a), numerical solution (scaled by .3) (b) and localization of the error in case of the symmetric (c) and the superconvergent ("SC 3") (d) schemes in planes $z = 0$ and ± 0.5 on a 64^3 grid for example 1.5.3.

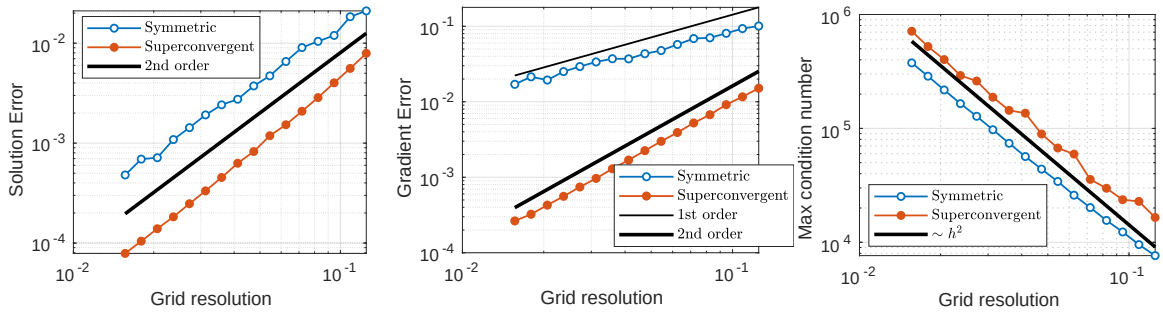


Figure 1.37: Convergence of the numerical solution (left) and its gradient (center) in the L^∞-norm and dependence of the condition number on grid resolution (right) for example 1.5.3.

Three star-shaped domains

We construct the compound domain with the help of three star-shaped domains defined by the level-set functions:

$$\phi_p = s_p \left(\sqrt{X_p^2 + Y_p^2 + Z_p^2} - r_p + \beta_p \frac{X_p^2 + 5X_p^4 Y_p - 10X_p^2 Y_p^3}{(X_p^2 + Y_p^2)^{\frac{5}{2}}} \cos\left(\frac{\pi}{2}\frac{Z_p}{r_1}\right) \right), \qquad p = 1, 2, 3,$$

where

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \end{pmatrix} = \underline{\underline{\boldsymbol{R}}}(\boldsymbol{n}_p, \theta_p) \begin{pmatrix} x - x_p \\ y - y_p \\ z - z_p \end{pmatrix}$$

with $\underline{\underline{\boldsymbol{R}}}(\boldsymbol{n}_p, \theta_p)$ being the rotational matrix about the vector $\boldsymbol{n}_p$ by an angle θ_p:

$$\underline{\underline{\boldsymbol{R}}}(\boldsymbol{n}_p, \theta_p) = \begin{pmatrix} \cos\theta_p + n_x^2(1-\cos\theta_p) & n_x n_y(1-\cos\theta_p) - n_z\sin\theta_p & n_x n_z(1-\cos\theta_p) + n_y\sin\theta_p \\ n_y n_x(1-\cos\theta_p) + n_z\sin\theta_p & \cos\theta_p + n_y^2(1-\cos\theta_p) & n_y n_z(1-\cos\theta_p) - n_x\sin\theta_p \\ n_z n_x(1-\cos\theta_p) - n_y\sin\theta_p & n_z n_y(1-\cos\theta_p) + n_x\sin\theta_p & \cos\theta_p + n_z^2(1-\cos\theta_p) \end{pmatrix}.$$

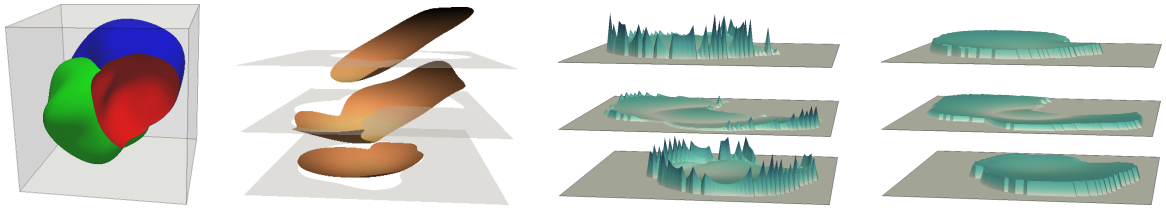
The parameters of the three shapes, the set operators used and Robin coefficients for each part of the boundary are summarized in Table 1.3. The parameters of the Poisson-

type equation are $\mu = 1 + (0.2\sin(x) + 0.3\cos(y))\cos(z)$ and $k = \cos(x+z)\exp(y)$. The test function is $u = \sin(x + 0.3y)\cos(x - 0.7y)\exp(z) + 3\log(\sqrt{x^2+y^2+z^2+0.5})$

p	r_p	x_p	y_p	z_p	β_p	s_p	$\boldsymbol{n}_p$	θ_p	Set operator	Robin coeff. α_p
1	0.73	0.13	0.16	0.19	0.08	1	$(1,1,1)$	0.3π	-	1
2	0.66	-0.21	-0.23	-0.17	-0.08	1	$(1,1,1)$	-0.3π	`union`	$1+\sin(x)\cos(y)\exp(z)$
3	0.59	0.45	-0.53	0.03	-0.08	-1	$(-1,1,0)$	-0.2π	`intersection`	$\exp(x+y+z)$

Table 1.3: Shape parameters for example 1.5.3.

Visualizations of the compound domain, numerical solution and error distributions on a 64^3 grid are plotted in Figure 1.38. Figure 1.39 shows the results of the convergence analysis. One can draw the same conclusions about the rates of convergence as in the previous examples.



(a) (b) (c) (d)

Figure 1.38: Compound domain (a), numerical solution (scaled by .3) (b) and localization of the error in case of the symmetric (c) and the superconvergent ("SC 3") (d) schemes in planes $z = 0$ and ± 0.5 on a 64^3 grid for example 1.5.3.

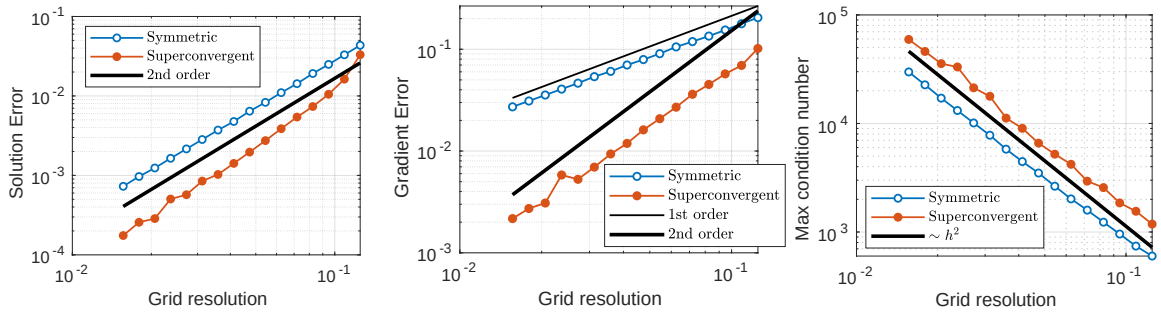


Figure 1.39: Convergence of the numerical solution (left) and its gradient (center) in the L^∞-norm and dependence of the condition number on grid resolution (right) for example 1.5.3.

1.5.4 Accuracy of the level-set representation

In the all above examples, the exact values of the level-set functions describing compound domains were used. However, in real applications, especially involving moving free boundaries, only approximate values of level-set functions are available. Inaccurate descriptions of domains may lead to a reduction in the accuracy of numerical solutions and, more importantly, the loss of the superconvergent property for the non-symmetric scheme. In this section we investigate the requirement of the presented numerical methods on the accuracy of level-set representation of irregular domains. We demonstrate that second-order of accuracy is sufficient for both considered methods; however, for the superconvergent scheme to produce second-order gradients such a second-order accurate level-set representation must be superconvergent itself in the sense that derivatives of the level-set data are second-order accurate as well.

Specifically, we repeat the example from Section 1.5.2 with artificially perturbed values of level-set functions by a specific error function. We consider three different choices of such perturbations:

1. Second-order discontinuous:

$$\phi^{(1)}(\boldsymbol{r}) = \phi_{\text{exact}}(\boldsymbol{r}) + 0.1\zeta_d(\boldsymbol{r})h^2$$

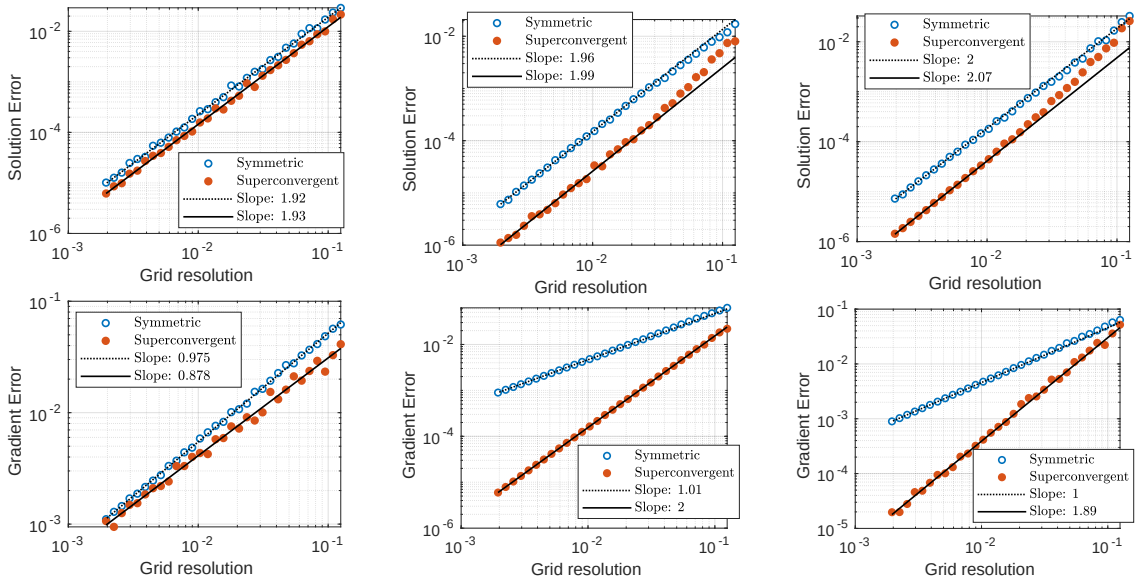
2. Second-order continuous:

$$\phi^{(2)}(\boldsymbol{r}) = \phi_{\text{exact}}(\boldsymbol{r}) + 0.1\zeta_c(\boldsymbol{r})h^2$$

3. Third-order discontinuous:

$$\phi^{(3)}(\boldsymbol{r}) = \phi_{\text{exact}}(\boldsymbol{r}) + \zeta_d(\boldsymbol{r})h^3$$

where $\phi_{\text{exact}}(\boldsymbol{r})$ denotes the exact level-set function, $\zeta_d(\boldsymbol{r})$ is a uniformly distributed random variable with values in $[-1; 1]$ and $\zeta_c(\boldsymbol{r})$ is a continuous function. Specificaly, we use $\zeta_c(\boldsymbol{r}) = \sin(10x)\cos(10y)$. Such perturbations produce second-, second- and third-order accurate level-set data with first-, second- and second-order accurate derivatives, respectively. The convergence results are presented in Figure 1.40. For all types of considered perturbations both schemes produce second-order accurate numerical solutions, however, in case of noisy error of second order, these numerical solutions are not superconvergent anymore for the non-symmetric scheme.

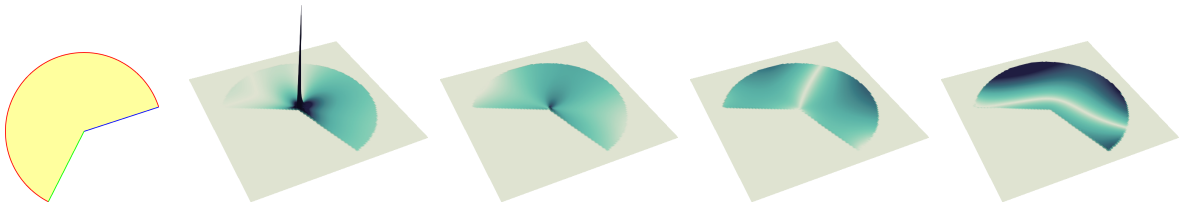


(a) Second-order noisy error (b) Second-order smooth error (c) Third-order noisy error

Figure 1.40: Convergence of the numerical solution (top row) and its gradient (bottom row) in the L^∞-norm in cases of various level-set data perturbations for example 1.5.4.

1.5.5 Singular solutions

Throughout this work it has been implicitly assumed that the sought-for solutions are sufficiently smooth, specifically, they need to have two and three bounded derivatives in cases of the symmetric scheme and the superconvergent one, respectively, for the presented error analysis to be valid. However, it is well-known that sharp features of domain boundaries might lead to solutions that are not smooth [94]. The results of this work are still of great importance, first, because the presented methods can be applied to problems in which the required smoothness conditions are satisfied and, second, because the presented methods can serve as a basis in the development of methods for singular solutions, e.g. along the lines of [175].



(a) Domain (b) $p = 1$ (c) $p = 2$ (d) $p = 3$ (e) $p = 4$

Figure 1.41: Visualization of the irregular domain (a) and error distributions in case of the superconvergent scheme (b)-(e) on a 256^2 grid for example 1.5.5.

To validate the above smoothness requirement and study the behavior of the presented discretizations in cases when such a requirement is not satisfied, we consider the following example. The irregular domain is taken as a circular sector of angle $\Theta = \gamma\pi$, with $\gamma = 1.25$, radius $R_0 = 0.85$, located at $(x_0, y_0) = (0.03, -0.04)$ and rotated by $\theta_0 = 0.1\pi$ such that straight edges of the sector make angles θ_0 and $\theta_0 + \Theta$ with the positive x-direction as illustrated in Figure 1.41a. We solve problem (1.1) with $\mu = 1$, $k = 0$, $f = 0$

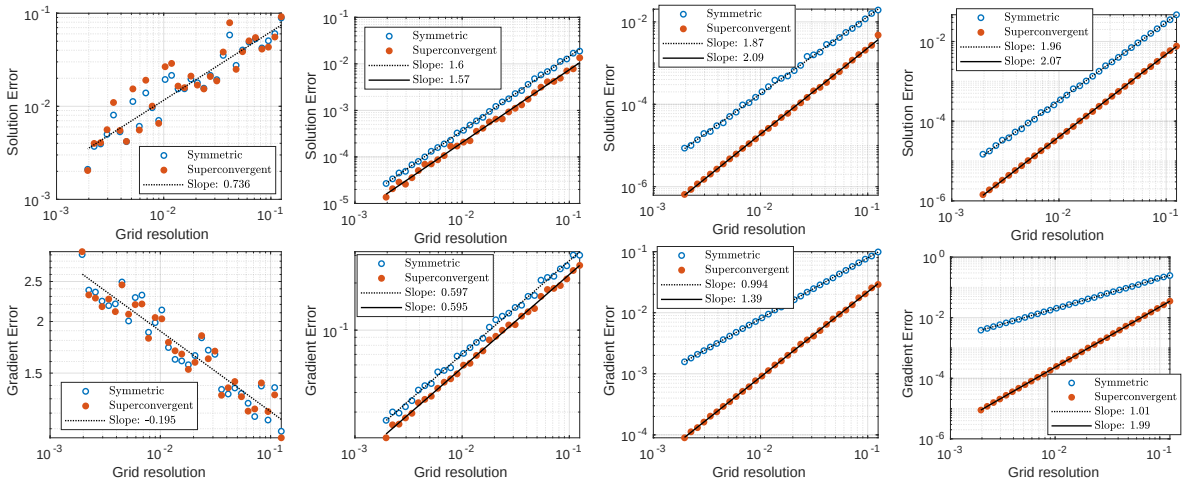
subject to boundary conditions (1.2) with $\alpha_0 = \alpha_1 = \alpha_2 = 1$ and

$$g_0 = g_1 = u_p \quad \text{and} \quad g_2 = \frac{\partial u_p}{\partial \boldsymbol{n}} + u_p,$$

where

$$u_p = r^{\frac{p}{\gamma}} \cos\left(\frac{p}{\gamma}(\theta - \theta_0)\right), \quad p = 1, 2, \ldots. \tag{1.23}$$

The analytical solution to such a problem for every $p = 1, 2, \ldots$ is given by (1.23) and has $p - 1$ bounded derivatives.

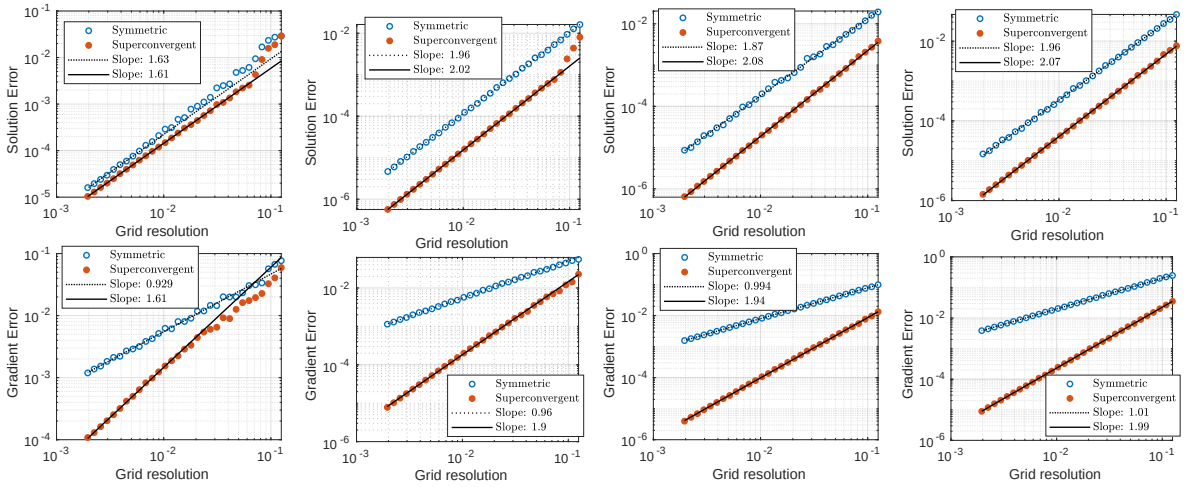


(a) $p = 1$ (b) $p = 2$ (c) $p = 3$ (d) $p = 4$

Figure 1.42: Convergence of numerical solutions (top row) and their gradients (bottom row) in the L^∞-norm measured across the entire irregular domain for example 1.5.5.

The convergence results of numerical solution of this problem with the presented methods for $p = 1, 2, 3$ and 4 are plotted in Figure 1.42. As expected, the symmetric scheme achieves designed convergence rates in cases $p = 3$ and 4 (solution has at least two bounded derivatives) while the superconvergent scheme produces second-order gradients only for $p = 4$ (solution has at least three bounded derivatives). In case of $p = 3$ the

superconvergent scheme does produce significantly better results than the symmetric one, however, the solution gradient converges only at rate ~ 1.4. For $p = 2$ (one bounded derivative) the numerical solutions and their gradients convergence with rates ~ 1.6 and ~ 0.6, respectively, for both schemes. For $p = 1$ (zero bounded derivatives) both schemes produce only first-order accurate numerical solutions with non-convergent gradients.



(a) $p = 1$ (b) $p = 2$ (c) $p = 3$ (d) $p = 4$

Figure 1.43: Convergence of numerical solutions (top row) and their gradients (bottom row) in the L^∞-norm measured across a reduced domain $\tilde{\Omega}$ that excludes the point of singularity for example 1.5.5.

It is also important to study the convergence of numerical solutions in a domain that excludes singular points. Specifically, we consider a domain $\tilde{\Omega}$ obtained from the original circular sector by subtracting a small circle of radius 0.1 located at (x_0, y_0). The convergence results computed across this new domain are shown in Figure 1.43. Interestingly, both methods achieve their designed rates of convergence in $\tilde{\Omega}$ for both the numerical solutions and their gradients with exception for case $p = 1$ (zero bounded derivative) in which numerical solutions converge at rate of 1.6 for both schemes. High accuracy

of numerical solutions and their gradients away from the singularity are consistent with results for the error distribution, examples of which on a 256^2 grid are plotted in Figure 1.41 (b)-(e), that show that the largest errors are localized in the vicinity of the singularity. These results can be exploited in the further extension of the presented methods to problems with singular solutions along the lines of [175].

1.6 Conclusion

We have presented two finite volume discretizations of the Poisson equation subject to Robin boundary conditions in irregular domains with piecewise smooth boundaries. In the case of the symmetric discretization, fluxes between cells are evaluated at the centers of the cell faces and boundary conditions are approximated either with the help of a Taylor expansion in the normal to boundary in the case of smooth interfaces or by a special treatment in the case where kinks occur. Our analysis suggests that such a discretization has a $\mathcal{O}(1)$ truncation error near the domain's boundary. A more-accurate, albeit nonsymmetric, discretization with first order truncation error is obtained by evaluating fluxes at the centers of cut cell faces and by using a linear least-squares interpolant of the unknown to approximate the boundary conditions. An important part of both schemes is a hierarchical integration algorithm that enables the computation of volume and surface integrals over irregular geometries that may have sharp features.

On a number of numerical examples in two and three spatial dimensions, we have illustrated that both schemes produce second-order accurate solutions. The difference

between the two methods is that (i) the magnitude of error for the nonsymmetric discretization is smaller (in some examples by an order of magnitude) and more importantly (ii) numerical gradients are second-order accurate in the case of the nonsymmetric discretization, compared to first-order accurate in the case of the symmetric discretization. Consequently, we conclude that in applications in which gradients play an important role, it is desirable to use the nonsymmetric superconvergent discretization. However, in applications in which only the numerical solution is of interest, the symmetric discretization offers the advantage of resulting in a symmetric linear system that can be efficiently solved using fast iterative methods such as the preconditioned conjugate gradient.

The requirement on the level-set representation's accuracy is numerically investigated. It is shown that a second-order accurate description of irregular domain is sufficient for both the symmetric and superconvergent schemes. For the superconvergent method to produce second-order accurate gradients the level-set representation must be "superconvergent" itself in the sense that derivatives of level-set data is second-order accurate as well.

The methods are also investigated on examples with singular solutions that have limited number of bounded derivatives. It is demonstrated that the methods should be used with care for such problems, since the accuracy of numerical solutions may severely deteriorate in such cases. However, it is also shown that even for such problems, the methods produce accurate solution away from singularities.

We also note that both discretizations use only immediate neighboring points, thus

making them excellent candidates for parallelization and/or extension to adaptive grids. Besides these extensions, future work will include exploring such aspects as an improved fall-back strategy for the superconvergent scheme in the three-dimensional case and special methods for extrapolating functions across piecewise smooth interfaces.

1.A Hierarchical geometry reconstruction algorithm

The integration method we use is in large part based on the methods and results of Min and Gibou [99, 101] and Fries *et al.* [50, 51]. We illustrate our hierarchical algorithm using second order geometric elements. With modifying or removing certain steps, the algorithm can either be extended to higher order elements or reduced to linear geometric ones. Since an extension from two to three spatial dimensions is not straightforward, we consider the three dimensional case here.

Let us consider the reconstruction of a compound domain described by n_ϕ level-set functions $\{\phi_i\}_{i=1}^{n_\phi}$ with associated set operations $\{\alpha_i\}_{i=1}^{n_\phi}$ in a simplex $\mathcal{S}$, which we shall call the *parent simplex*. To describe the reconstruction of a compound domain in a simplex, we introduce four types of basic geometric element: *vertex*, *edge*, *triangle* and *tetrahedron*, defined as follows:

- *vertex*: defined by its coordinates (x, y, z).
- *edge*: defined by three vertices (v_0, v_1, v_2).
- *triangle*: defined by three vertices (v_0, v_1, v_2) and three opposing edges (e_0, e_1, e_2).

- *tetrahedron*: defined by four vertices (v_0, v_1, v_2, v_3) and four opposing triangles (f_0, f_1, f_2, f_3).

Each type of geometric element is illustrated in Figure 1.44 along with its corresponding reference element.

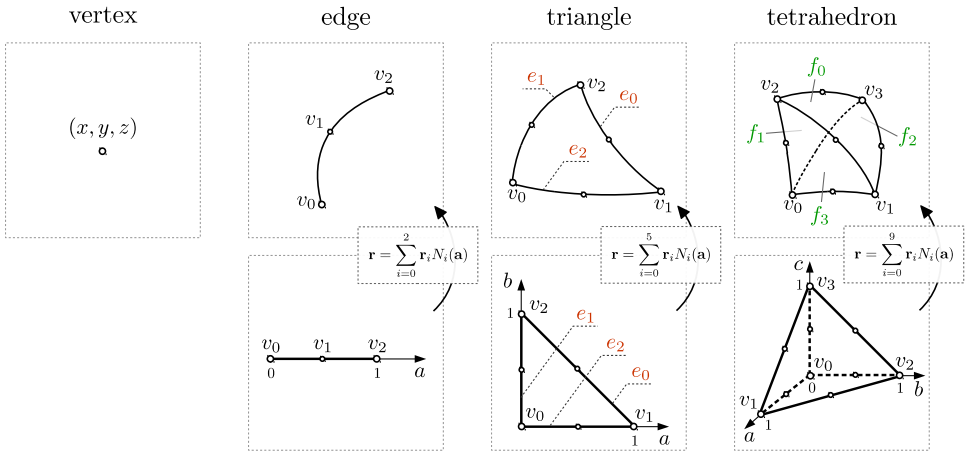


Figure 1.44: Basic geometric types (top row) and corresponding reference elements (bottom row).

For every type of geometric element, we associate an iso-parametric mapping from its reference element:

$$\boldsymbol{r} = \sum_{i=0}^{n_{\text{nodes}}-1} \boldsymbol{r}_i N_i(\boldsymbol{a}), \tag{1.24}$$

where $\boldsymbol{r}$ is the coordinate in physical space, $\boldsymbol{a}$ is the coordinate in reference element space, $N_i(\boldsymbol{a})$ are Lagrangian shape functions, $\boldsymbol{r}_i$ are coordinates of the vertices composing an element and $n_{\texttt{nodes}}$ is their cardinal (3, 6 and 10 for edges, triangles and tetrahedra, respectively). In addition, for every reference element we define the interpolation function as:

$$\phi(\boldsymbol{a}) = \sum_{i=0}^{n_{\text{nodes}}-1} \phi_i N_i(\boldsymbol{a}), \tag{1.25}$$

where the ϕ_i's are the values of a level-set function at vertices of a reference element (directly transferred from a physical element).

The reconstruction starts with creating empty sets of vertices V, edges E, triangles F and tetrahedra T, and filling them with the initial structure of the simplex $\mathcal{S}$, so that V, E, F and T contain 10, 6, 4 and 1 elements, respectively. After the structure of the parent simplex $\mathcal{S}$ is decomposed into basic geometric elements, our algorithm reconstructs zero-isocontours of all level-set functions one after another.

Let us assume that $k-1$ interfaces have been reconstructed, and consider the reconstruction of the zero-isocontour of the k^{th} level-set function in more details. The first step is to evaluate the values of the k^{th} level-set function at all active vertices from the set V. This can be done, for example, by interpolating the level-set function from the parent simplex, assuming that values of the function are provided at vertices of the parent simplex. Then, we ensure that the level-set values form a "valid" data set [50] in the sense that the zero-isocontour does not cross any edges or triangles multiple times as discussed in 1.A.5. Once a valid set of data is obtained, the algorithm reconstructs an interface on each geometric type separately. Specifically, (1) the algorithm loops through all the vertices of V, determining whether a vertex is inside or outside of the domain described by ϕ_k and records this information; (2) the algorithm loops through all active edges from E, finding an intersection of an edge with the interface and splitting edges

that are crossed; (3) the algorithm loops through all active triangles from F, splitting all crossed triangles (note that when a crossed triangle is considered, then its edges have already been split during the previous step); (4) the algorithm loops through all active tetrahedra from T, splitting all crossed tetrahedra (note that when a crossed tetrahedron is considered its faces and edges have already been split during previous steps). In the case of intersecting elements (1.A.6) or elements with under-resolved curvature (1.A.7), necessary geometric elements are refined and the reconstruction is attempted again. For clarity the algorithm is summarized in Algorithm 1.

In what follows, we describe in details the splitting procedures for each basic geometric types. We exploit the ideas of Min [99] to determine the topological relationship between a geometric element and an interface. Specifically, we enumerate the *corner* vertices of an element in the ascending order with respect to the values of the level-set function ϕ_k and count the total number of *corner* vertices $n_{(-)}$ with negative values of ϕ_k[3]. In such a way, the number $n_{(-)}$ unequivocally corresponds to a certain splitting scheme of a geometric element.

There exist two trivial cases: the case where $n_{(-)} = 0$ corresponds to the situation when an element is completely outside of the domain described by the k^{th} level-set function and no splitting is required. Similarly, the case when $n_{(-)}$ is equal to the total number of *corner* vertices of an element (i.e. $n_{(-)} = 1$ for vertices, $n_{(-)} = 2$ for edges, $n_{(-)} = 3$ for triangles, $n_{(-)} = 4$ for tetrahedra) corresponds to the situation when the

[3]As in [99], we exclude cases where $\phi_k = 0$ at a vertex by perturbing the value of ϕ_k at such a vertex by a very small amount $\varepsilon = 10^{-20}$

```
1: Create empty sets of vertices V, edges E, triangles F and tetrahedra T
2: Fill sets V, E, F and T with the initial structure of parent simplex S
3: for every level-set function φ_j from j = 1 to n_φ do
4:     Set valid_reconstruction = false
5:     while valid_reconstruction = false do
6:         V_tmp ← V, E_tmp ← E, F_tmp ← F, T_tmp ← T
7:         Set valid_data = false
8:         while valid_data = false do
9:             Sample values of φ_j at all active vertices from V_tmp
10:            Set valid_data = true
11:            Check validity of level-set data(may set valid_data = false)
12:            if valid_data = false then
13:                Refine necessary elements in E_tmp, F_tmp and T_tmp (1.A.5)
14:            end if
15:        end while
16:        Set valid_reconstruction = true
17:        for every vertex v from V_tmp do
18:            Apply procedure from 1.A.1 to v
19:        end for
20:        for every edge e from E_tmp do
21:            Apply procedure from 1.A.2 to e
22:        end for
23:        for every triangle f from F_tmp do
24:            Apply procedure from 1.A.3 to f
25:            (may set valid_reconstruction = false (1.A.6 and 1.A.7))
26:        end for
27:        for every tetrahedron t from T_tmp do
28:            Apply procedure from 1.A.4 to t
29:            (may set valid_reconstruction = false (1.A.6 and 1.A.7))
30:        end for
31:        if valid_reconstruction = false then
32:            Refine necessary elements in E, F and T (1.A.6 and 1.A.7)
33:            Discard V_tmp, E_tmp, F_tmp and T_tmp
34:        else
35:            V ← V_tmp, E ← E_tmp, F ← F_tmp, T ← T_tmp
36:        end if
37:    end while
38: end for
```

Algorithm 1: Reconstruction of a compound domain in a 3D simplex $\mathcal{S}$.

element is entirely inside the k^{th} generating domain and no splitting is required as well.

In what follows, we describe the non-trivial cases for each geometric type.

1.A.1 Vertex (0-simplex)

In the case of vertices, it is only necessary to determine whether a vertex is located inside or outside of the k^{th} generating domain by checking whether $\phi_k < 0$ or $\phi_k > 0$ at this vertex and store this information. Such information will be later used during the post-processing step to determine the role of a vertex in the geometric reconstruction such as *inside vertex*, *outside vertex*, *vertex on an interface*, *vertex on an intersection of interfaces*, etc.

1.A.2 Edge (1-simplex)

Let us denote the vertices of an edge as v_0, v_1, and v_2, such that $\phi_k(\boldsymbol{r}_0) \leq \phi_k(\boldsymbol{r}_1) \leq \phi_k(\boldsymbol{r}_2)$, where $\boldsymbol{r}_0$, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are coordinates of the vertices v_0, v_1 and v_2, respectively. Possible values of $n_{(-)}$ for the edges are 0, 1 and 2. Cases $n_{(-)} = 0$ and $n_{(-)} = 2$ are trivial and only in the case where $n_{(-)} = 1$ is an edge crossed by the k^{th} interface.

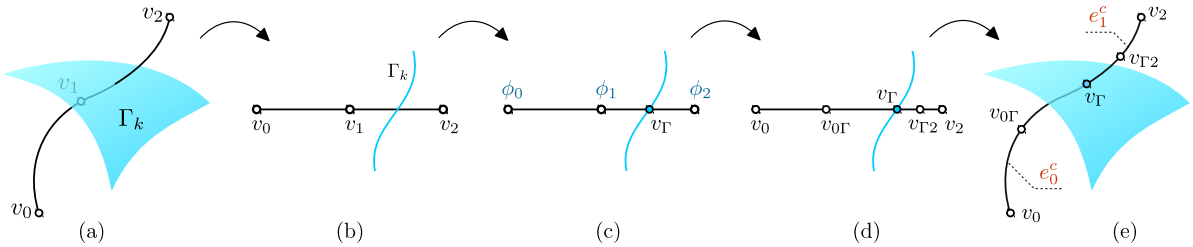


Figure 1.45: Finding an intersection of an edge with the zero-isocountor of a level-set function.

To find an intersection between a possibly curved edge and an interface, a corresponding reference element is considered (Figure 1.45 (b)). Values of a level-set function are directly transferred from a physical element and the location of the zero-level-set, denoted as v_Γ, is found as a root of a quadratic interpolating polynomial (1.25) (Figure 1.45 (c)). Next, midpoints between pairs of vertices (v_0, v_Γ) and (v_Γ, v_2), denoted as $v_{0\Gamma}$ and $v_{\Gamma 2}$, respectively, are found (Figure 1.45 (d)). The three newly created vertices v_Γ, $v_{0\Gamma}$ and $v_{\Gamma 2}$ are mapped to the physical space using (1.24) and two new edges $e_0^c = (v_0, v_{0\Gamma}, v_\Gamma)$ and $e_1^c = (v_\Gamma, v_{\Gamma 2}, v_2)$ are created. Finally, the crossed edge and its vertex v_1 are marked as *inactive*.

1.A.3 Triangle (2-simplex)

Let us denote the corner vertices of a triangle as v_0, v_1, v_2 and their opposite edges as e_0, e_1, e_2, such that $\phi_k(\boldsymbol{r}_0) \leq \phi_k(\boldsymbol{r}_1) \leq \phi_k(\boldsymbol{r}_2)$, where $\boldsymbol{r}_0$, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are the locations of vertices v_0, v_1 and v_2, respectively. Possible values of $n_{(-)}$ for triangles are 0, 1, 2 and 3. Cases $n_{(-)} = 0$ and $n_{(-)} = 3$ are trivial and not considered here.

Case $n_{(-)} = 1$: A triangle is crossed by the k^{th} interface, i.e. the interface intersects the edges e_1 and e_2 (Figure 1.46 (a)). We first note that these points of intersection have already been found and edges e_1 and e_2 have been replaced with new elements (Figure 1.46 (b)). Let us denote the points of intersection as v_{01}^c and v_{02}^c, the edge children of e_1 as $e_1 \to e_0^c$ and $e_1 \to e_1^c$ and the edge children of e_2 as $e_2 \to e_0^c$ and $e_2 \to e_1^c$. Two more edges need to be created, one corresponding to the zero-level-set, denoted as e_0^c, and the

other one being an auxiliary edge that splits a quadrilateral into two triangles, denoted as e_1^c. This is done on a corresponding reference triangle (Figures 1.46 (c)-(f)).

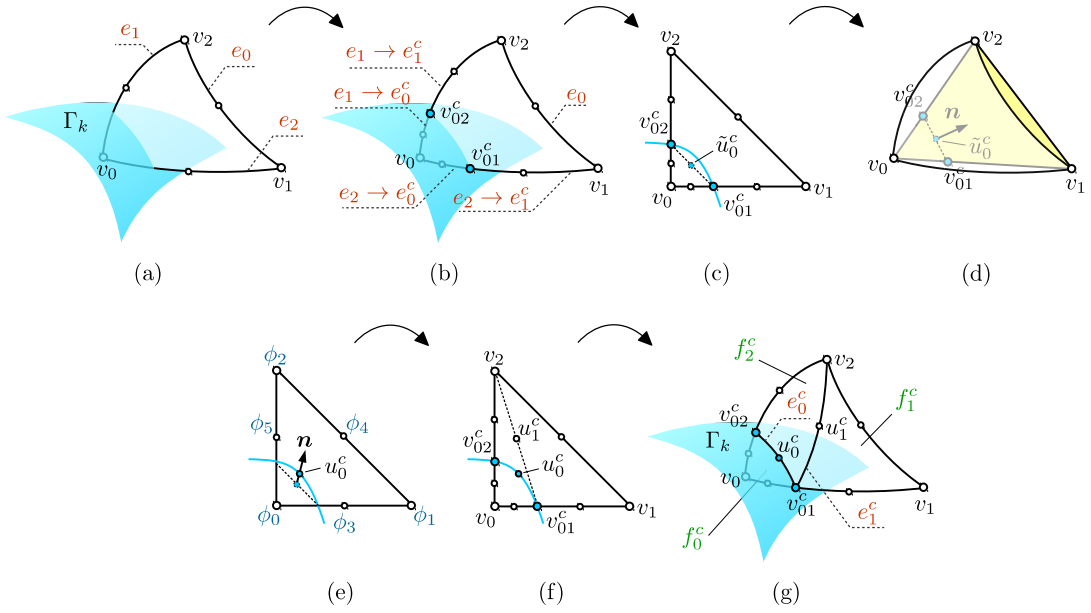


Figure 1.46: Splitting of a triangle by the zero-isocountor of a level-set function in the case $n_{(-)} = 1$.

The procedure for creating the midpoint for e_0^c is as follows: (1) a midpoint between the vertices v_0^c and v_1^c is found and denoted as $\tilde{u}_0^c$ (Figure 1.46 (c)). This point serves as the starting point for finding the midpoint for e_0^c; (2) To select the search direction we consider a linear approximation of the original curved triangle in the physical space (Figure 1.46 (d)), find the direction $\boldsymbol{n}$ normal to the (straight) line segment (v_{01}^c, v_{02}^c) and map it back onto the reference triangle. Chosen in such a way, the direction approximates the normal direction to the (curved) line segment (v_{01}^c, v_{02}^c) in the original curved triangle. Note that such a direction may differ significantly from the normal direction to the line segment (v_{01}^c, v_{02}^c) in the reference triangle; (3) the values of the k^{th} level-set function, its first and second derivatives in the chosen search direction are estimated based on

interpolating formula (1.25) (see Figure 1.46 (e)). These values are used to approximate the level-set function in the normal direction by a quadratic polynomial, whose root corresponds to the location of the interface, denoted as u_0^c. The midpoint for e_1^c, denoted as u_1^c, is defined simply as the midpoint between v_{01}^c and v_2 (Figure 1.46 (f)).

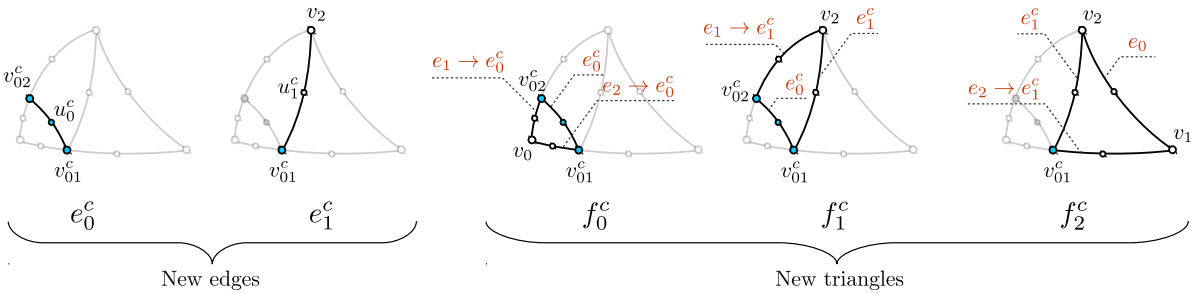


Figure 1.47: New elements created during the splitting of a triangle in the case $n_{(-)} = 1$.

The two newly created vertices u_0^c and u_1^c are mapped onto the physical space and two new edges $e_0^c = (v_{01}^c, u_0^c, v_{02}^c)$ and $e_1^c = (v_{01}^c, u_1^c, v_2)$ are created. Finally, the initial triangle is replaced with three smaller triangles f_0, f_1 and f_2, as illustrated in Figure 1.47.

Case $n_{(-)} = 2$: in this case the interface crosses the edges e_0 and e_1. We note again that the points of intersections of the interface with edges have already been found and the edges e_0 and e_1 have already been split. Let us denote the points of intersection as v_{12}^c and v_{02}^c, the edge children of e_0 as $e_0 \to e_0^c$ and $e_0 \to e_1^c$ and the edge children of e_1 as $e_1 \to e_0^c$ and $e_1 \to e_1^c$.

The procedure for this case is very similar to the procedure used in the case $n_{(-)} = 1$. Two new vertices u_0^c and u_1^c, representing the midpoints of two new edges, are found on a reference triangle and mapped onto the physical space. Two new edges $e_0^c = (v_0, u_0^c, v_{12}^c)$ and $e_1^c = (v_{02}^c, u_1^c, v_{12}^c)$ are created and the initial triangle is replaced with three new

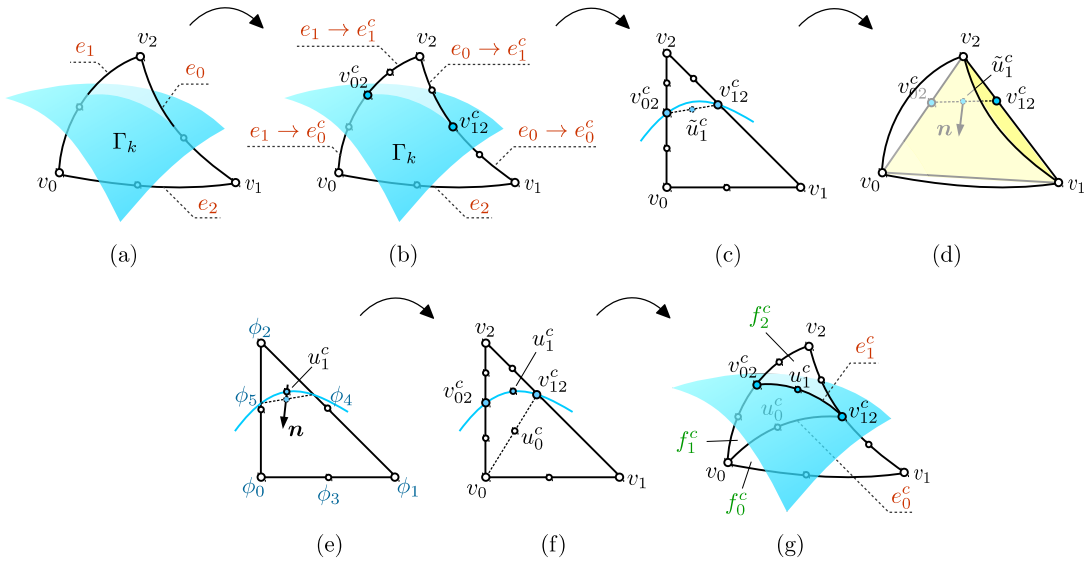


Figure 1.48: Splitting of a triangle by the zero-isocountor of a level-set function in the case $n_{(-)} = 2$.

triangle children f_0^c, f_1^c and f_2^c, as illustrated in Figure 1.49.

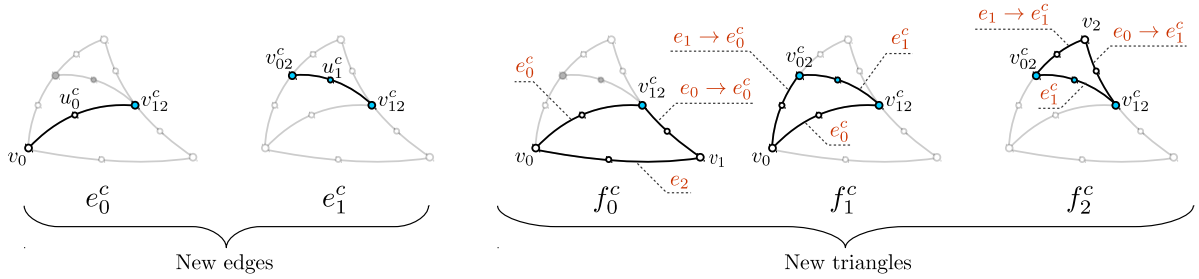


Figure 1.49: New elements created during the splitting of a triangle in the case $n_{(-)} = 2$.

1.A.4 Tetrahedron (3-simplex)

Let us denote the vertices of a tetrahedron as v_0, v_1, v_2, v_3 and the opposite triangles as f_0, f_1, f_2, f_3, such that $\phi_k(\boldsymbol{r}_0) \leq \phi_k(\boldsymbol{r}_1) \leq \phi_k(\boldsymbol{r}_2) \leq \phi_k(\boldsymbol{r}_3)$, where $\boldsymbol{r}_0$, $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$ are the locations of the vertices v_0, v_1, v_2 and v_3, respectively. Possible values of $n_{(-)}$ for triangles are 0, 1, 2, 3 and 4. Cases $n_{(-)} = 0$ and $n_{(-)} = 4$ are trivial and not considered

here.

Case $n_{(-)} = 1$: the k^{th} interface crosses the edges corresponding to the pairs of vertices (v_0, v_1), (v_0, v_2) and (v_0, v_3) and the triangles f_1, f_2 and f_3. These elements have already been split during the previous steps. Let us denote the points of intersection of the boundary with the tetrahedron's edges as v_{01}^c, v_{02}^c and v_{03}^c. To complete the reconstruction of the k^{th} interface in the tetrahedron, one only needs to create three new triangles f_0^c, f_1^c, f_2^c and four new tetrahedra t_0^c, t_1^c, t_2^c, t_3^c (Figure 1.50 (a)), as illustrated in Figures 1.50 (b) and 1.50 (c), respectively.

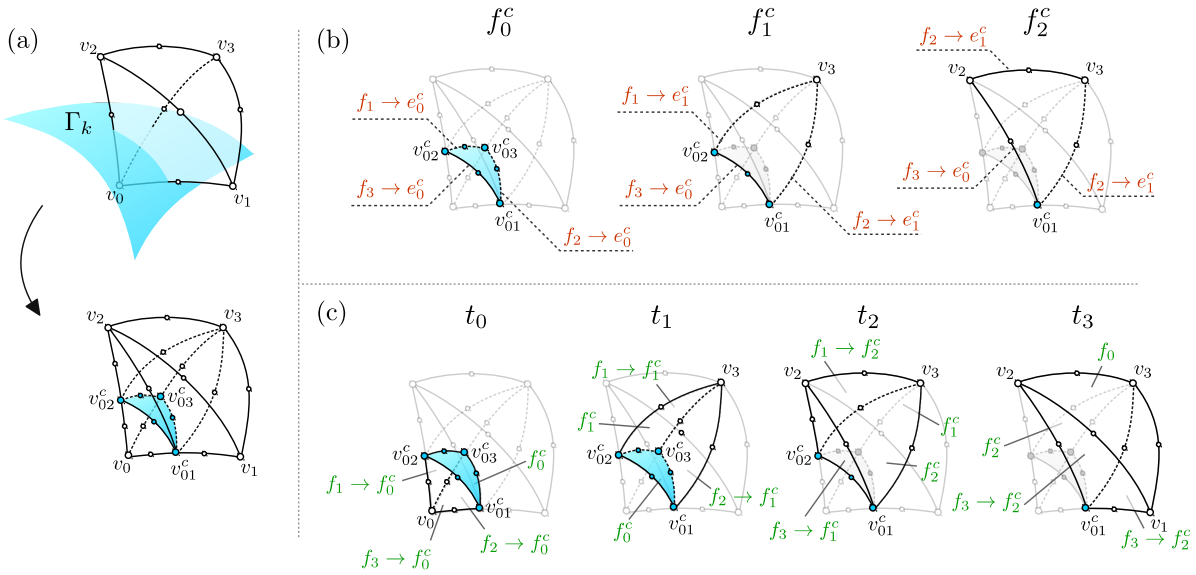


Figure 1.50: Splitting of a tetrahedron by the zero-isocountor of a level-set function in the case $n_{(-)} = 1$.

Case $n_{(-)} = 2$: this case is more complicated than the cases $n_{(-)} = 1$ and $n_{(-)} = 3$. The interface crosses the edges corresponding to pairs of vertices (v_0, v_2), (v_0, v_3), (v_1, v_2) and (v_1, v_3) and all the triangles f_0, f_1, f_2 and f_3 (Figure 1.51 (a)). Let us denote the points of intersection of the boundary with the tetrahedron's edges as v_{02}^c, v_{03}^c, v_{12}^c and

v_{13}^c (Figure 1.51 (b)).

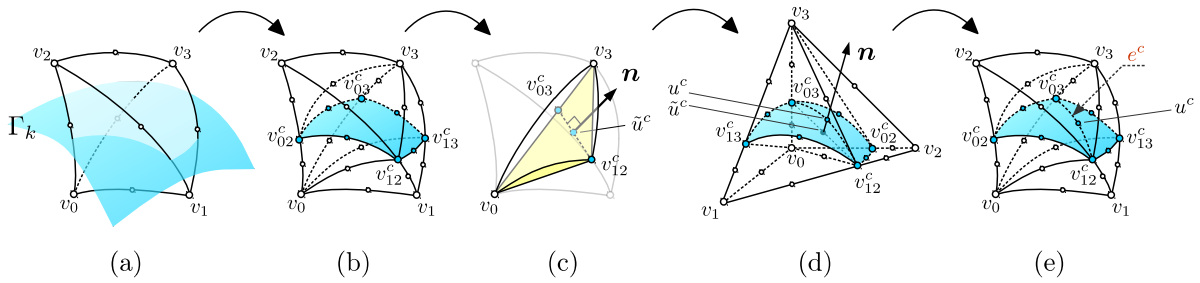


Figure 1.51: Splitting of a tetrahedron by the zero-isocountor of a level-set function in the case $n_{(-)} = 2$.

To complete the interface reconstruction in a tetrahedron, it is necessary to first create one more edge, e^c, connecting the vertices v_{12}^c and v_{03}^c (or v_{13}^c and v_{02}^c, alternatively). The midpoint of this new edge, u^c, is found using a corresponding reference tetrahedral element (Figure 1.51 (e)):

1. the starting point for search of u^c, denoted as $\tilde{u}^c$, is found as the midpoint of the line segment (v_{12}^c, v_{03}^c);

2. the search direction is defined by considering a linear approximation of the curved triangle (v_0, v_{12}^c, v_3) in the physical space and taking the direction normal to line segment (v_{12}^c, v_{03}^c) (Figure 1.51 (c));

3. the values of the level-set functions and its first and second derivatives along the chosen search direction are estimated using interpolating formulas (1.25);

4. the location of u^c is found as a root of the quadratic polynomial approximating the level-set function in the chosen direction (Figure 1.51 (d));

5. finally, u^c is mapped onto the physical space using (1.24) (Figure 1.51 (e)).

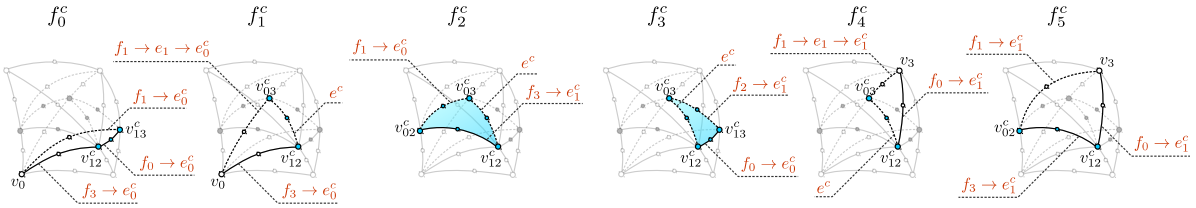


Figure 1.52: New triangles created during the splitting of a tetrahedron in the case $n_{(-)} = 2$.

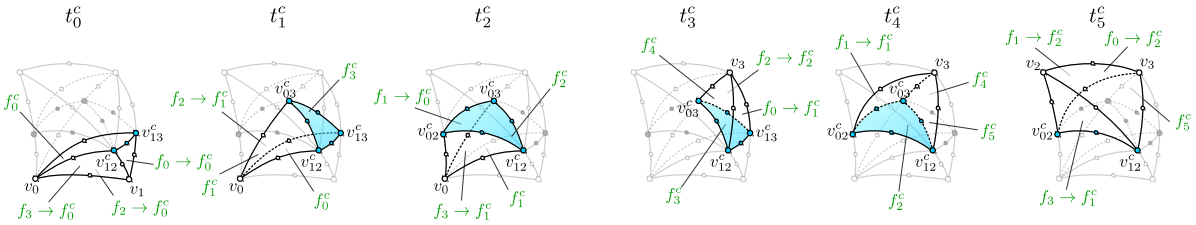


Figure 1.53: New tetrahedra created during the splitting of a tetrahedron in the case $n_{(-)} = 2$.

The remaining steps in the interface reconstruction are to create a new edge $e^c = (v_{03}^c, u^c, v_{12}^c)$, six new triangles $f_0^c, \ldots, f_5^c$ and six new tetrahedra $t_0^c, \ldots, t_5^c$, as illustrated in Figures 1.52 and 1.53.

Case $n_{(-)} = 3$: this case is very similar to the case where $n_{(-)} = 1$. The interface crosses the edges corresponding to the pairs of vertices (v_0, v_3), (v_1, v_3) and (v_2, v_3) and the triangles f_0, f_1 and f_2. As usual, we denote the points of intersection of the interface with the tetrahedron's edges as v_{03}^c, v_{13}^c and v_{23}^c. To complete the reconstruction of the k^{th} interface in the tetrahedron, one only needs to create three new triangles f_0^c, f_1^c, f_2^c and four new tetrahedra t_0^c, t_1^c, t_2^c, t_3^c (Figures 1.54 (a)), as illustrated in Figures 1.54 (b) and 1.54 (c).

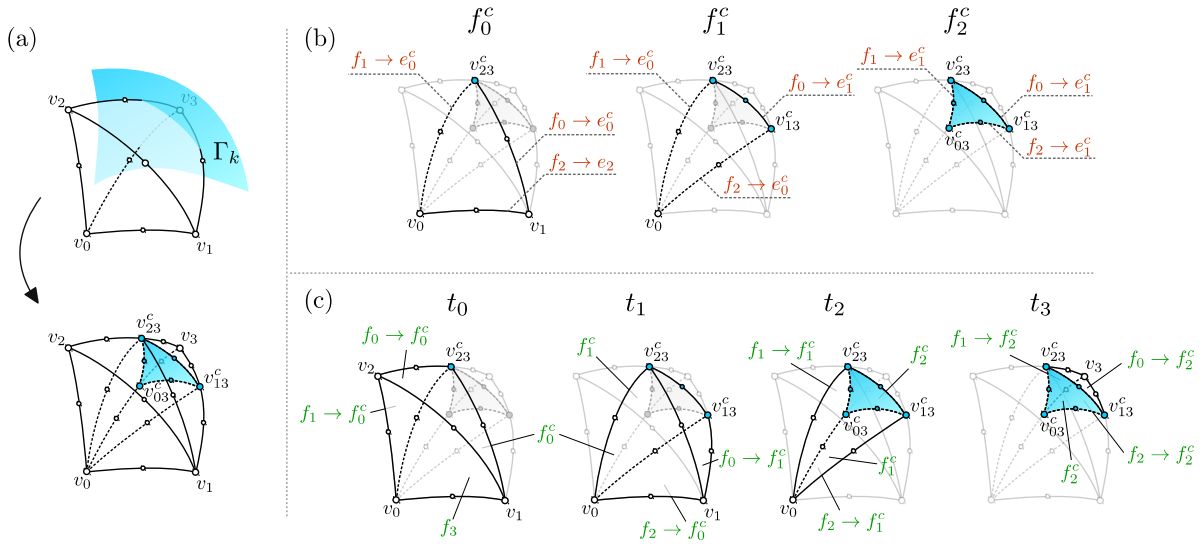


Figure 1.54: Splitting of a tetrahedron by the zero-isocountor of a level-set function in the case $n_{(-)} = 3$.

1.A.5 Valid level-set data

The algorithm described above assumes that the values of level-set functions that are used form a "valid" data, i.e., that the implicit interface being reconstructed does not cross any edge element multiple times and does not cross any triangular face without crossing its edges (in 3D). In [50], such a condition was checked by evaluating the level-set functions on sample grids of vertices distributed evenly in a geometric element. If it is found that the condition is not satisfied, the geometric element is recursively refined until "valid" level-set data is obtained.

We employ a different approach to address this issue. First, since we restrict ourselves to geometric reconstructions with second order elements, on which level-set functions are approximated by second-order polynomials (1.25), we make use of analytical form of level-set functions' approximations to check for multiple intersections of implicit interfaces with

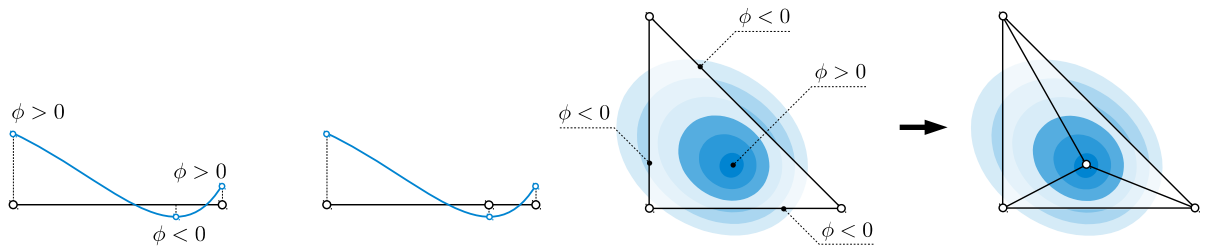


Figure 1.55: Refinement procedure to obtain valid level-set data for edges (a) and triangles (b).

edges and triangles. Specifically, in case when a level-set functions has the same sign on the boundary of an edge or a triangle, we find the point of extremum of the polynomial approximating a level-set function on that geometric element, check whether such point lies inside the element and whether the sign of the level-set at this point coincides with the sign at the end points of the element. In case of opposite signs, the element is refined by placing an additional vertex at the found point of extremum as illustrated in Figure 1.55.

To preserve consistency in the overall geometric reconstruction, higher-level elements contained in refined edges and triangles also need to be refined. Clearly, it is straightforward to perform such a task when a higher-level element contains only one refined sub-element as illustrated in Figure 1.56a for triangles and in Figure 1.57 for tetrahedra. However, even when a higher-level element contains several refined sub-elements, this task can easily be performed in an iterative fashion taking into account one refined sub-element at a time as demonstrated Figures 1.56b and 1.58.

Such a refinement strategy has great advantages over the simple recursive refinement used in [50]. First, it efficiently eliminates elements with invalid level-set data: only one

pass is needed for this geometry-aware refinement, while the simple recursive subdivision may require several passes and in fact does not guarantee that the invalidity of data is resolved in some limited number of refinements. Second, it generates much less geometric elements, thus saving computational time and resources: in [50], one simple recursive refinement of a tetrahedron generates 8 new tetrahedra, while our geometry-aware refinement generates only 2 (refined edge) or 3 (refined triangle) new tetrahedra.

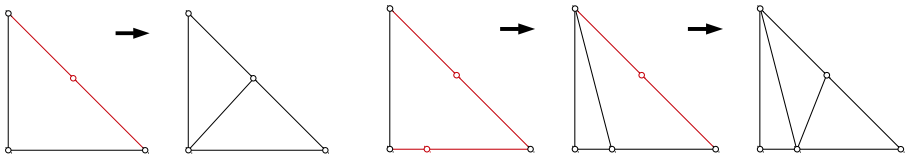


Figure 1.56: Refinement of a triangle containing one (a) and multiple (b) refined edges.

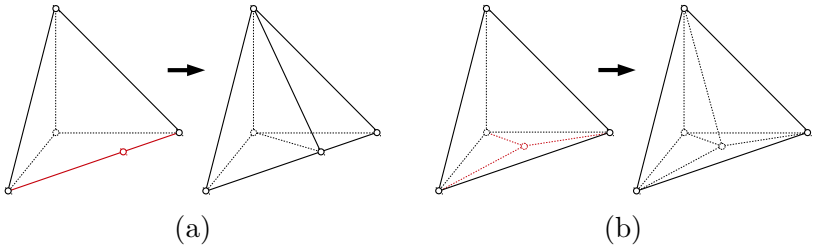


Figure 1.57: Refinement of a tetrahedron containing a refined edge (a) and a refined face (b).

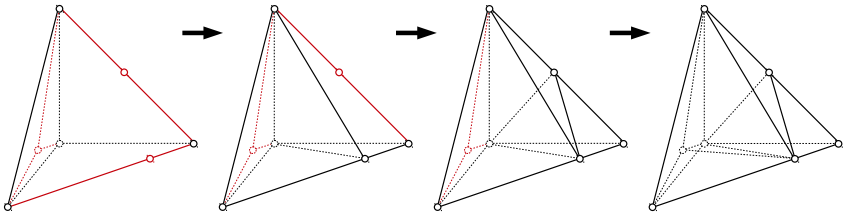


Figure 1.58: An example of iterative refinement of a tetrahedron containing multiple refined sub-elements.

1.A.6 Removing invalid geometric reconstruction

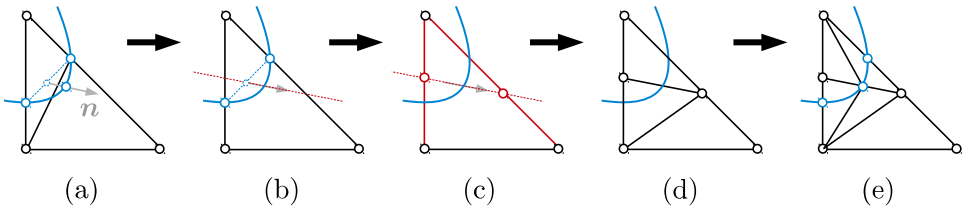


Figure 1.59: Illustration of refinement procedure to eliminate intersecting elements.

During the reconstruction of implicit interfaces with high order geometric element, much attention must be given to ensure that generated reconstructions do not contain overlapping or intersecting elements. Problems of this type occur during the search for midpoints of curved elements (cases $n_{(-)} = 1$ and 2 for triangles). In the case where intersecting edges are detected (see Figure 1.59 (a)), the edges of the underlying triangle are refined by a line passing through the starting point of the midpoint search and oriented along the search direction (see Figure 1.59 (b) and (c)). The intersection points of such line with the edges of the triangle can be found by defining a local level-set function describing the line:

$$\phi_{\text{line}}(\boldsymbol{a}) = \boldsymbol{\tau}_{\text{line}} \cdot (\boldsymbol{a} - \boldsymbol{a}_{\text{line}}),$$

where $\boldsymbol{\tau}_{\text{line}}$ is a unit vector perpendicular to the line and $\boldsymbol{a}_{\text{line}}$ is the coordinate vector of a point lying on the line (in this particular case, it is convenient to use the starting point). The line crosses an edge if the function ϕ_{line} has opposite signs at the endpoints of that edge. Moreover, the location of the intersection is easily found as the root of the restriction of the level-set function ϕ_{line} to the edge, which is itself a linear function. After

the required edges are refined, higher-level elements (triangles and tetrahedra) contained refined edges are also refined as described the same iterative described in the previous section (see Figure 1.59 (d)) and the reconstruction of an implicit interface is attempted again (see Figure 1.59 (e))

1.A.7 On curvature resolution in three spatial dimensions

There exists a fundamental difference between geometric reconstructions of co-dimension one surfaces in two and three dimensions on Cartesian grids: in the two-dimensional case, the increase in grid resolution always leads to a decrease in the ratio of the mesh size to the curvature radius of a surface being reconstructed, while in the three-dimensional case this does not necessarily occur. Consider, for example, the calculation of the surface of a sphere of radius R. Suppose the Cartesian grid used for the reconstruction is such that one of the grid cells is located at the distance $d = \sqrt{R^2 - \left(\frac{1}{2}h\right)^2}$ from the sphere's center and oriented as shown in Figure 1.60, where h is the linear dimension of the cell. In that case, the intersection of the cell and the sphere's surface is a quarter of a spherical cap, with base radius equal to $r = \sqrt{R^2 - d^2} = \frac{1}{2}h$. Thus, the ratio of the mesh size to the curvature radius for the cap's base curve is equal to $h/\frac{1}{2}h = 2$. Clearly, such a situation can occur no matter how fine the computational grid is.

Now let us consider how the presence of such highly curved regions affects the accuracy of the numerical integration. First, let us focus on the error in calculating the area bounded by the sphere's surface in the cell's faces in the negative y-direction (in fact, the

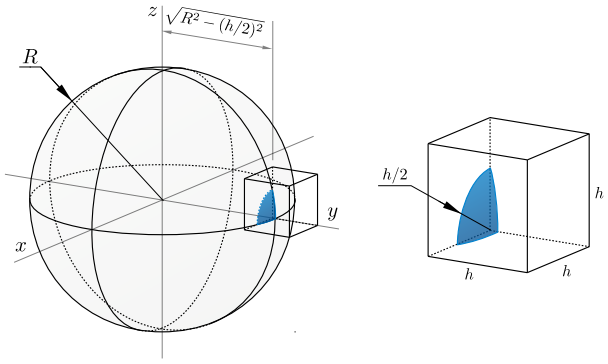


Figure 1.60: An example of a highly curved region during integration on Cartesian grids in three dimensions.

calculation of such areas is very important for computing the total fluxes between cells in the finite volume methods), which is a quarter of circle of radius $\frac{1}{2}h$ (Figure 1.61 (a)). It is convenient to scale the dimensions of the cell to $1 \times 1 \times 1$ (Figure 1.61 (b)). Suppose for simplicity that all necessary intersections of the zero-isocontour of a level-set function with the grid lines are found exactly and that the only error present is the error inherent to the integration method used. For example, in the case of the surface reconstruction with geometric elements, such an error is the difference in the areas bounded by the exact surface and by the approximating element (Figure 1.61 (b)). Let us denote such error as ε. Then, to obtain the error made in a cell with dimensions $h \times h \times h$, one just needs to multiply this error by h^2 (Figure 1.61 (c)). Thus, the presence of highly curved elements leads to $\varepsilon_h = \varepsilon h^2 = \mathcal{O}(h^2)$ errors in calculating areas bounded by implicit surfaces on cells' faces. A similar estimate holds for errors in calculating areas of surfaces themselves (Figure 1.60 (right)). The analysis for this type of integration is complicated by the fact that the radii of curvature of other curves bounding a piece of a surface depend on h. However, for instance in the example considered, as $h \to 0$ the area of the curved

surface's piece becomes closer and closer to the area cut by the sphere's surface in the face of the cell.

Note that in this argument, we assume neither a specific type of integration method nor a specific order of accuracy of the integration method. Therefore, the emergence of $\mathcal{O}(h^2)$ errors in the presence of regions of non-decreasing mesh-size-to-curvature-radius ratios may occur in other methods for numerical integration as well.

In [50] it was reported that high-order geometric reconstructions of implicit interfaces on three dimensional grids do not demonstrate expected optimal convergence rates for integration. It was concluded that such a decrease in accuracy is caused by the constraint that certain approximating elements must lie in (Cartesian) planes of grid cells' faces, which confirms the above argument. It was shown that if such a constraint is eliminated, which effectively eliminates the presence of non-decreasing mesh-size-to-curvature-radius ratio elements, then the optimal convergence rates are recovered.

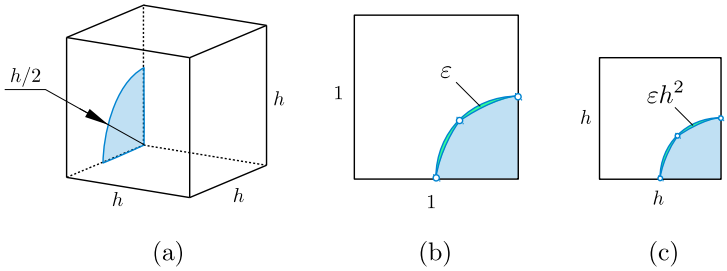


Figure 1.61: $\mathcal{O}(h^2)$ error emerging near highly curved regions

To confirm the above argument, we consider the geometric reconstruction (with quadratic elements) of a sphere with radius $R = 0.77$ placed at the distance $d = R - 2h + \frac{1}{2}h$ from computational domain boundary, where h is the mesh size. Thus,

for every grid resolution there is a cell containing a quarter of a spherical cap of radius $\frac{1}{2}h$ as depicted in Figure 1.60. We calculate the total sphere's area and the area of a quarter of a spherical cap located in a cell near the sphere's pole as shown in Figure 1.60. The convergence results are presented in Figure 1.62 (left). As expected, once the error associated with locating the intersections of the sphere's surface with the grid lines becomes negligible, the error in the area of the spherical cap's quarter contained in one cell scales as $\mathcal{O}(h^2)$. Interestingly, the error in the total sphere's area is often less than the error coming from one cell, which is probably due to the cancellations of errors from adjacent cells. However, for such applications as solving PDEs using finite volume methods, it is the error for a single cell that affects the accuracy of numerical solutions. One can also observe an oscillatory behavior of the error in the total sphere's area with peaks that decrease as $\mathcal{O}(h^2)$.

To avoid such a disastrous drop to second order of accuracy, we employ the following approach. Before reconstructing an irregular interface in a cell, we compute the absolute values of the surface's principal curvatures κ_1 and κ_2 and select the maximum one, which we denote as $\kappa_{3\mathrm{D}} = \max(|\kappa_1|, |\kappa_2|)$. Principal curvatures κ_1 and κ_2 of an implicit interface can be computed from its mean and Gaussian curvatures κ_M and κ_G as

$$\kappa_{1,2} = \frac{1}{2}\left(\kappa_M \pm \sqrt{\kappa_M^2 - 4\kappa_G}\right),$$

which in their turn can be computed from the level-set function ϕ using formulas

$$\kappa_M = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) \quad \text{and} \quad \kappa_G = \frac{\det \begin{pmatrix} \nabla\nabla\phi & \nabla\phi \\ \nabla\phi^T & 0 \end{pmatrix}}{|\nabla\phi|^4}$$

Then, during the reconstruction we ensure that for all geometric elements the following criterion is satisfied

$$\kappa l < C \kappa_{3\mathrm{D}} h, \tag{1.26}$$

where $\kappa_{3\mathrm{D}}$ is the curvature of a geometric element, l is its length and C is a $\mathcal{O}(1)$ constant (specifically, we choose $C = 2$). If this condition is violated, then the underlying element (triangle or tetrahedron) is refined in the same way as that described in 1.A.6.

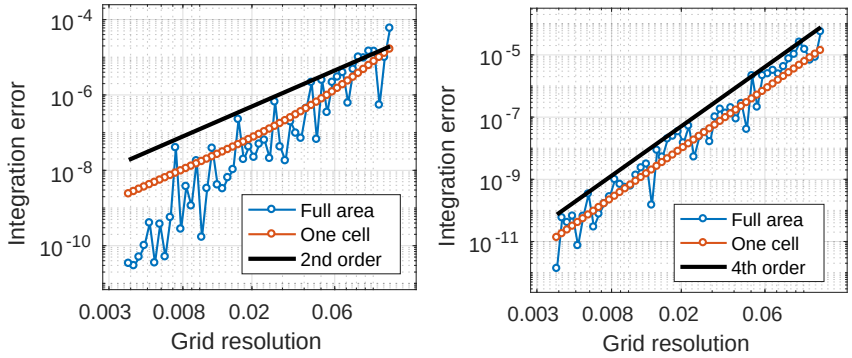


Figure 1.62: Convergence of the area of the total sphere and the area of a quarter of a spherical cap contained in one cell near the sphere's pole without (left) and with (right) enforcing condition 1.26.

The purpose of the procedure described above is to enforce that the mesh-size-to-curvature-radius ratio decrease as the computational grid is refined. Examples of reconstructions of a sphere with second-order geometric elements without and with this procedure are shown in Figure 1.63. In Figure 1.62 (right) convergence results for the

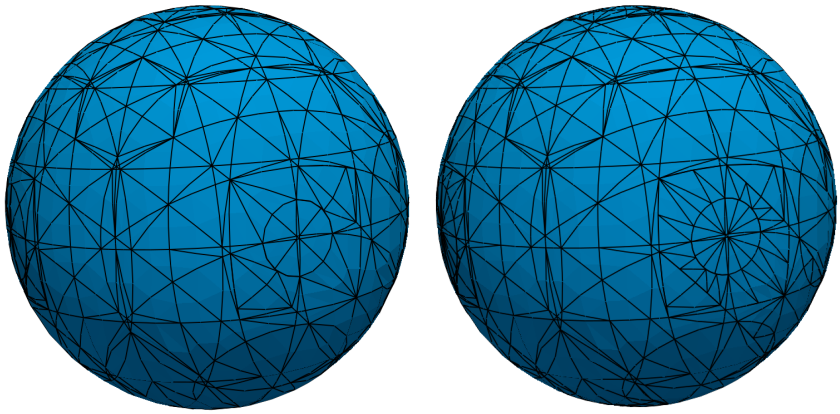


Figure 1.63: Comparison between the geometric reconstructions of a sphere produced without (left) and with (right) enforcing condition 1.26.

calculation of the total sphere's area and the area contained in the single cell are plotted. The expected convergence rates[4] for both areas are recovered.

1.A.8 Integration

Once all interfaces are reconstructed in a simplex, it is straightforward to perform integration. One just needs to select appropriate geometric elements and map quadrature points of any desired order onto these elements using (1.24), scaling the weights of integration points by $\sqrt{\det(\underline{\underline{\boldsymbol{J}}} \cdot \underline{\underline{\boldsymbol{J}}})}$, where $\underline{\underline{\boldsymbol{J}}}$ is the Jacobian matrix of the mapping.

In our implementation of the quadratic geometric reconstruction, we use the following quadrature rules, which is exact for polynomials of degrees ≤ 2

- *edges*: 3 points at $a_i = 0$, 0.5 and 1.0 with weights $w_i = 1/6$, $4/6$ and $1/6$ (Gauss-Lobatto quadrature rule).

[4] As noted in [50], for $(2k+1)^{\text{th}}$-order accurate methods, a $2k+2$ convergence rate is observed (for smooth surfaces).

- *triangles*: 3 points at $(a_i, b_i) = (0.5, 0.0)$, $(0.0, 0.5)$ and $(0.5, 0.5)$ with equal weights $w_i = 1/3$.

- *tetrahedra*: 4 points at $(a_i, b_i, c_i) = (\beta, \beta, \beta)$, (α, β, β), (β, α, β) and (β, β, α) with equal weights $w_i = 1/4$, where $\alpha = (5 + 3\sqrt{5})/20$ and $\beta = (5 - \sqrt{5})/20$.

Note that quadrature points for *edge* and *triangle* elements coincide with the nodes of the geometric elements. We have observed that such a choice of quadrature rules improves the overall accuracy of the integration procedure.

In order to reduce the computational time in the case of linear geometric reconstructions, we calculate the integral over a d-dimensional simplex $\mathcal{S}^d$ using the formula

$$\int_{\mathcal{S}^d} f(\boldsymbol{r})\, d\boldsymbol{r}^d \approx M(\mathcal{S}^d) \frac{f(\boldsymbol{r}_1) + ... + f(\boldsymbol{r}_{d+1})}{d+1},$$

where $M(\mathcal{S}^d)$ is the measure (length for $d = 1$, area for $d = 2$ and volume for $d = 3$) of S^d, and $\{\boldsymbol{r}_i\}_{i=1}^{d+1}$ are the vertices of S^d. That is, an integral over a simplex is approximated by the average of an integrand at the vertices of the simplex multiplied by its measure. Such an approach requires neither the mapping of quadrature points nor the calculation of Jacobians.

To select appropriate elements, i.e., determine which geometric elements represent a compound domain, determine the smooth components of the domain boundary, the intersections of different intersection, etc, we use the following simple procedure. During the reconstruction of the interfaces in a simplex, we record for each element the information about its relation (whether it is inside, outside or on the interface) to every generating

domain. At the end of the reconstruction, every geometric element owns a list containing the information about the element's relation to all generating domains. Post-processing of such lists allows to robustly determine the role of the elements in the reconstruction of a compound domain.

Chapter 2

Solving Elliptic PDEs with Discontinuities across Irregular Interfaces

2.1 Introduction

It is crucial, for simulating important processes in the physical and life sciences, to find the numerical solution of elliptic equations with discontinuities in the diffusion coefficient, the source term, the solution and its flux. In the case of interfacial flows for example, jump conditions describe the discontinuity in stress that is balanced by forces that exist between phases [23]. In the simulation of protein folding, it is the electrostatic potential that has a jump across the protein's Solvent-Excluded Surface [42, 106, 105, 176]. Other

examples include solidification of multicomponent alloys [159, 88, 17] or any diffusion dominated processes with different materials properties. At the macroscale, changes across the surface can only be represented by sharp jumps, hence the need to numerically represent them as such. Failure to do so introduces errors that change the characteristics of the problem.

Numerical approximations to solve such problems have been proposed and fall into two categories, depending on whether the interface is represented explicitly or implicitly. For example, finite element discretizations approximate the space in which the solution is defined and rely on a mesh that explicitly describes the surface [8]. It is straightforward to impose boundary conditions in that framework, which is ideally suited for cases where deformations are small. For large deformations, difficulties associated with the mesh generation process are severe. Consequently, in this case, implicit representations of the interface have proved to be a better choice; imposing jump conditions, however, is a difficulty task in that framework. One of the first attempts is the Immersed Interface Method (IIM), where the jump conditions are combined with Taylor expansions of the solution on each side of the interface in order to modify the stencils of grid points adjacent to the interface. The main difficulties are the need to evaluate high-order jump conditions and surface derivatives. Several authors have further developed numerical methods within the IIM framework, e.g. [30, 92, 93, 174, 14, 1, 2, 3, 159]. Another approach is the Ghost Fluid Method (GFM) [46], first developed to treat shocks and contact discontinuities in compressible flows. The idea is to define a ghost fluid in the regions across

the discontinuities by adding the interface jump to the true fluid. This simple treatment avoids the large error incurred by differentiating discontinuous solutions, and thus gives an elegant framework to manage jump conditions. The idea of the GFM was used for solving the Poisson equation with jump conditions in [95]. In this case, the jump in the normal derivative of the solution is projected onto the Cartesian directions in order to use a dimension-by-dimension approach. The authors showed that the normal jump is accurately captured, while the tangential jump is smeared; this, in turn, leads to a lack of convergence in the flux. The Voronoi Interface Method [66] solved that problem by first constructing a local Voronoi mesh adjacent to the interface and by then considering a GFM treatment. In that case, the solution is second-order accurate in the L^∞-norm with first-order accurate fluxes in the same norm. This method has been applied to electroporation problems [67, 108], where the unknown is the electric potential at each grid points. While this method produces symmetric positive definite linear systems and only requires the right-hand side of the linear system to be modified, it requires the generation of a local Voronoi mesh and interpolation of numerical solutions from such unstructured meshes back onto Cartesian grids, which may add some challenges, especially in three spatial dimensions. The literature on solving elliptic problems with jump conditions is quite vast and we refer the interested reader to the review [61] and to other approaches, such as cut-cell approaches [34, 118], discontinuous Galerkin and the eXtended Finite Element Method (XFEM) [91, 69, 110, 36, 13, 109, 79, 49, 65, 167], the Virtual Node Method [111, 11, 111, 149, 132, 72] or other fictitious domain approaches [32, 31, 54].

In this work, we propose a finite volume discretization for elliptic interface problems in a similar vein as in [116, 128, 18] for the treatment of Neuman and Robin boundary conditions. To take into account the jump conditions, we adopt the ideas of relating the values of discontinuous functions using Taylor expansions in the normal direction and employing one-sided local least-square interpolations. In these aspects, the present method is similar to the variant of the augmented immersed interface method used in [75, 177]. The differences are, however, substantial: first, a finite volume approach is used instead of finite differences; second, the resulting linear system contains no augmented variable, which makes it straightforward to invert with "black-box" linear solvers (BiCGSTAB is used in this work; in [75, 177] the system is solved iteratively using GMRES in conjunction with a fast Poisson solver); third, the method does not involve quadratic terms in Taylor expansions and local interpolations are linear (compared to cubic ones in [75, 177]), which keeps the discretization stencil quite compact while still resulting in second-order accurate solutions (thanks to the finite volume approach). The small stencil size and the simplicity of inverting the resulting linear system make the method a good candidate for parallelization and application in the context of adaptive grids, which will be demonstrated in this work as well. We consider a level-set representation of the interface so that the method can be used in free boundary problems [122, 121, 144, 59].

2.2 Numerical Discretization

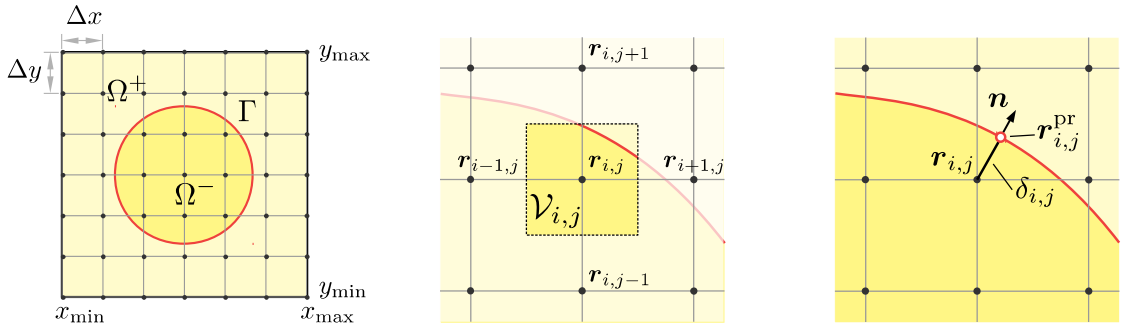
Consider a rectangular domain $\Omega = [x_{\min}; x_{\max}] \times [y_{\min}; y_{\max}]$ with an immersed irregular interface Γ that splits Ω into two sets Ω^- and Ω^+ as illustrated in Fig. 2.1a. We seek a numerical solution $u = u(\boldsymbol{r})$, with $\boldsymbol{r} = (x, y)$, to the following problem:

$$k^{\pm}u^{\pm} - \nabla \cdot \left(\mu^{\pm}\nabla u^{\pm}\right) = f^{\pm}, \qquad \text{in } \Omega^{\pm}, \tag{2.1}$$

$$[u] = \alpha, \qquad \text{on } \Gamma, \tag{2.2}$$

$$[\mu\partial_{\boldsymbol{n}} u] = \beta, \qquad \text{on } \Gamma, \tag{2.3}$$

where the functions $k^{\pm} = k^{\pm}(\boldsymbol{r})$, $\mu^{\pm} = \mu^{\pm}(\boldsymbol{r}) \geq \epsilon > 0$, $f^{\pm} = f^{\pm}(\boldsymbol{r})$, $\boldsymbol{r} \in \Omega^{\pm}$, and $\alpha = \alpha(\boldsymbol{r})$, $\beta = \beta(\boldsymbol{r})$, $\boldsymbol{r} \in \Gamma$, are given. We denote by $[q]$ the jump of a scalar quantity q across Γ, i.e. $[q] = q^+ - q^-$. For simplicity, we impose Dirichlet boundary conditions on the boundary of the computation domain, i.e. $u^+ = g$ on $\partial\Omega$, where $g = g(\boldsymbol{r})$ is given.



(a) (b) (c)

Figure 2.1: (a) Notation used in this work. (b) Illustration of a finite volume associated with a grid point (i, j). (c) Illustration of the projection of a grid point onto the interface Γ

We discretize the domain Ω into a uniform rectangular grid of $N_x \times N_y$ points with

spatial steps

$$\Delta x = \frac{x_{\max} - x_{\min}}{n_x - 1}, \quad \Delta y = \frac{y_{\max} - y_{\min}}{n_y - 1}$$

and associate with each point $\boldsymbol{r}_{i,j} = (x_i,\, y_j) = (x_{\min} + (i-1)\Delta x,\, y_{\min} + (j-1)\Delta y)$ a finite volume $\mathcal{C}_{i,j} = \left[x_i - \frac{1}{2}\Delta x; x_i + \frac{1}{2}\Delta x\right] \times \left[y_j - \frac{1}{2}\Delta y; y_j + \frac{1}{2}\Delta y\right]$, $i \in [2; N_x - 1]$, $j \in [2; N_y - 1]$ (see Fig. 2.1b). The Level-Set Method [122] is used to describe the irregular interface Γ. That is, we use a Lipschitz-continuous function $\phi(\boldsymbol{r})$ such that $\Omega^+ = \{\boldsymbol{r} :\, \phi(\boldsymbol{r}) > 0\}$, $\Omega^- = \{\boldsymbol{r} :\, \phi(\boldsymbol{r}) < 0\}$ and $\Gamma = \{\boldsymbol{r} :\, \phi(\boldsymbol{r}) = 0\}$.

At the grid points for which the interface Γ does not cross the finite volumes, equation (2.1) is discretized using the standard five-point stencil. Let us consider a point $\boldsymbol{r}_{i,j}$ with its finite volume $\mathcal{C}_{i,j}$ crossed by Γ. Integrating equations (2.1) over $\mathcal{C}_{i,j}$ and applying the divergence theorem, one gets the following expression:

$$\underbrace{\sum_{s=+,-} \int_{\Omega^s \cap \mathcal{C}_{i,j}} k^s u^s \, d\Omega}_{\text{Linear term}} - \underbrace{\sum_{s=+,-} \int_{\Omega^s \cap \partial\mathcal{C}_{i,j}} \mu^s \partial_{\boldsymbol{n}^s} u^s \, d\Gamma}_{\text{Flux between finite volumes}} = \underbrace{\sum_{s=+,-} \int_{\Omega^s \cap \mathcal{C}_{i,j}} f^s \, d\Omega}_{\text{Volumetric generation}} + \underbrace{\int_{\Gamma \cap \mathcal{C}_{i,j}} [\mu \partial_{\boldsymbol{n}} u] \, d\Gamma}_{\text{Surface generation}},$$

where the superscript ^{s} refers to the sign $^{\pm}$.

Following [116, 128], that is, approximating the domain integrals by the integrand value multiplied by the corresponding volumes, and estimating the fluxes between cells using values at nearest-neighbor grid points and central difference formulas, one obtains:

$$\sum_{s=+,-} k^s_{i,j} u^s_{i,j} |\mathcal{C}^s_{i,j}| - \sum_{s=+,-} \left(\mu^s_{i-\frac{1}{2},j} A^s_{i-\frac{1}{2},j} \frac{u^s_{i-1,j} - u^s_{i,j}}{\Delta x} + \mu^s_{i+\frac{1}{2},j} A^s_{i+\frac{1}{2},j} \frac{u^s_{i+1,j} - u^s_{i,j}}{\Delta x} + \right.$$
$$\left. \mu^s_{i,j-\frac{1}{2}} A^s_{i,j-\frac{1}{2}} \frac{u^s_{i,j-1} - u^s_{i,j}}{\Delta y} + \mu^s_{i,j+\frac{1}{2}} A^s_{i,j+\frac{1}{2}} \frac{u^s_{i,j+1} - u^s_{i,j}}{\Delta y} \right)$$
$$= \sum_{s=+,-} f^s_{i,j} |\mathcal{C}^s_{i,j}| + \int_{\Gamma \cap \mathcal{C}_{i,j}} \beta \, d\Gamma + \mathcal{O}\left(h^{\mathcal{D}}\right), \quad (2.4)$$

where $\mathcal{D}$ is the problem dimensionality, $h = \max(\Delta x, \Delta y)$, $|\mathcal{C}^{\pm}_{i,j}|$ denotes the volume of $\mathcal{C}_{i,j} \cap \Omega^{\pm}$, $u_{i,j} = u(\boldsymbol{r}_{i,j})$, $A^{\pm}_{i\pm\frac{1}{2},j}$ and $A^{\pm}_{i,j\pm\frac{1}{2}}$ are face areas of $\mathcal{C}^{\pm}_{i,j}$ in the x- and y-directions, respectively. To compute the boundary and domain integrals required by the proposed discretization, we use the geometric reconstruction approach from [99]. In case when an immersed interface is only piece-wise smooth the method from [18] can be used.

The discretization given by equation (2.4) requires that both values of u^- and u^+ be available at grid points with a control volume crossed by Γ, thus, one more equation is required at such grid points for the system of equations to be uniquely invertible. We derive the additional equation based on the jump conditions (2.2)-(2.3) and Taylor expansions of $u^{\pm}$ in the normal to the interface direction. Moreover, equations derived in this way can be used to express $u^{\pm}_{i,j}$ near the interface as a function of $u^{\mp}_{i,j}$ and the jump conditions (2.2)-(2.3). As a result, it enables us to eliminate the additional degrees of freedom, i.e., reduce the system's size back to $N_x \times N_y$, and make its structure more homogeneous (all of the equations in the linear system are of the same type). This is expected to make the system of equations less difficult to invert by black-box linear solvers. We select the $N_x \times N_y$ unknowns to solve for as:

$$u_{i,j} = \begin{cases} u^+_{i,j}, & \boldsymbol{r}_{i,j} \in \Omega^+, \\ u^-_{i,j}, & \boldsymbol{r}_{i,j} \in \Omega^-. \end{cases} \tag{2.5}$$

We then develop formulas to express $u^+_{i,j}$ for $\boldsymbol{r}_{i,j} \in \Omega^-$ and for $u^-_{i,j}$ for $\boldsymbol{r}_{i,j} \in \Omega^+$ as a function of the unknowns $u_{i,j}$. This is described next.

Consider a grid point $\boldsymbol{r}_{i,j}$ near the interface Γ and its projection, $\boldsymbol{r}^{\rm pr}_{i,j}$, onto the interface (see Fig. 2.1c). Taylor expansion relates the values of $u^\pm$ at $\boldsymbol{r}_{i,j}$ and $\boldsymbol{r}^{\rm pr}_{i,j}$ as:

$$u^\pm_{i,j} = u^\pm(\boldsymbol{r}^{\rm pr}_{i,j}) + \delta_{i,j}\partial_{\boldsymbol{n}} u^\pm(\boldsymbol{r}^{\rm pr}_{i,j}) + \mathcal{O}\left(h^2\right), \tag{2.6}$$

where $\delta_{i,j}$ is the signed distance from $\boldsymbol{r}_{i,j}$ to $\boldsymbol{r}^{\rm pr}_{i,j}$ ($\pm\delta_{i,j} > 0$ if $\boldsymbol{r}_{i,j} \in \Omega^\pm$). The geometrical quantities $\boldsymbol{n}(\boldsymbol{r}^{\rm pr}_{i,j})$, $\boldsymbol{r}^{\rm pr}_{i,j}$ and $\delta_{i,j}$ are estimated from the level-set function as:

$$\begin{aligned} \boldsymbol{n}(\boldsymbol{r}^{\rm pr}_{i,j}) &= \boldsymbol{n}_{i,j} + \mathcal{O}\left(h\right) \qquad \text{where} \qquad \boldsymbol{n}_{i,j} = \frac{\nabla\phi(\boldsymbol{r}_{i,j})}{|\nabla\phi(\boldsymbol{r}_{i,j})|}, \\ \boldsymbol{r}^{\rm pr}_{i,j} &= \boldsymbol{r}_{i,j} - \delta_{i,j}\boldsymbol{n}_{i,j} + \mathcal{O}\left(h^2\right), \\ \delta_{i,j} &= \frac{\phi(\boldsymbol{r}_{i,j})}{|\nabla\phi(\boldsymbol{r}_{i,j})|} + \mathcal{O}\left(h^2\right). \end{aligned}$$

Subtracting $u^-_{i,j}$ from $u^+_{i,j}$ given in (2.6) and taking into account the jump condition (2.2) one obtains:

$$u^+_{i,j} - u^-_{i,j} = \alpha(\boldsymbol{r}^{\rm pr}_{i,j}) + \delta_{i,j}\left(\partial_{\boldsymbol{n}} u^+(\boldsymbol{r}^{\rm pr}_{i,j}) - \partial_{\boldsymbol{n}} u^-(\boldsymbol{r}^{\rm pr}_{i,j})\right) + \mathcal{O}\left(h^2\right).$$

Furthermore, eliminating either $\partial_{\boldsymbol{n}} u^+(\boldsymbol{r}^{\rm pr}_{i,j})$ or $\partial_{\boldsymbol{n}} u^-(\boldsymbol{r}^{\rm pr}_{i,j})$ in the above expression using the jump condition (2.3) results in the following two equations:

$$u^+_{i,j} - u^-_{i,j} = \begin{cases} \alpha(\boldsymbol{r}^{\rm pr}_{i,j}) + \delta_{i,j}\dfrac{\beta(\boldsymbol{r}^{\rm pr}_{i,j})}{\mu^+(\boldsymbol{r}^{\rm pr}_{i,j})} - \delta_{i,j}\dfrac{\mu^+(\boldsymbol{r}^{\rm pr}_{i,j}) - \mu^-(\boldsymbol{r}^{\rm pr}_{i,j})}{\mu^+(\boldsymbol{r}^{\rm pr}_{i,j})}\partial_{\boldsymbol{n}} u^-(\boldsymbol{r}^{\rm pr}_{i,j}) \\ \alpha(\boldsymbol{r}^{\rm pr}_{i,j}) + \delta_{i,j}\dfrac{\beta(\boldsymbol{r}^{\rm pr}_{i,j})}{\mu^-(\boldsymbol{r}^{\rm pr}_{i,j})} - \delta_{i,j}\dfrac{\mu^+(\boldsymbol{r}^{\rm pr}_{i,j}) - \mu^-(\boldsymbol{r}^{\rm pr}_{i,j})}{\mu^-(\boldsymbol{r}^{\rm pr}_{i,j})}\partial_{\boldsymbol{n}} u^+(\boldsymbol{r}^{\rm pr}_{i,j}) \end{cases} + \mathcal{O}\left(h^2\right). \tag{2.7}$$

If one approximates either $\partial_{\boldsymbol{n}} u^+(\boldsymbol{r}^{\rm pr}_{i,j})$ or $\partial_{\boldsymbol{n}} u^-(\boldsymbol{r}^{\rm pr}_{i,j})$ using $u^{\pm}_{i,j}$ and $\{u_{p,q}, p \in [1, N_x], q \in [1, N_y]\}$, then these formulas can be used to eliminate additional degrees of freedom. A straightforward way to do that is to use a suitable local interpolants $u^{\pm}_I = u^{\pm}_I(\boldsymbol{r})$ of unknown functions $u^{\pm}$ as:

$$\partial_{\boldsymbol{n}} u^{\pm}(\boldsymbol{r}^{\rm pr}_{i,j}) = \boldsymbol{n}(\boldsymbol{r}^{\rm pr}_{i,j}) \nabla u^{\pm}_I(\boldsymbol{r}^{\rm pr}_{i,j}). \tag{2.8}$$

Specifically, in this work we use the linear interpolation:

$$u^{\pm}_I(\boldsymbol{r}) = u^{\pm}_{i,j} + (\boldsymbol{r} - \boldsymbol{r}_{i,j})^T \left(\nabla u^{\pm}\right)_{i,j} + \mathcal{O}\left(h^2\right), \tag{2.9}$$

where the gradient $\left(\nabla u^{\pm}\right)_{i,j}$ is found as the least-square solution satisfying the constraints:

$$u_{i+p,j+q} = u^{\pm}_{i,j} + (\boldsymbol{r}_{i+p,j+q} - \boldsymbol{r}_{i,j})^T \left(\nabla u^{\pm}\right)_{i,j}, \quad (p,q) \in N^{\pm}_{i,j}.$$

$N^{\pm}_{i,j}$ denotes the set of neighboring grid points of $\boldsymbol{r}_{i,j}$, lying in the region $\Omega^{\pm}$, that is:

$$N^{\pm}_{i,j} = \left\{(p,q): \quad p = -1, 0, 1, \quad q = -1, 0, 1, \quad (p,q) \neq (0,0), \quad \boldsymbol{r}_{i+p,j+q} \in \Omega^{\pm}\right\}.$$

In other words, the local linear interpolants are constructed using available values at the nearest-neighbor grid points of $\boldsymbol{r}_{i,j}$ (in Cartesian and diagonal directions). Note also that $u_{i+p,j+q} = u^{\pm}_{i+p,j+q}$ if $(p,q) \in N^{\pm}_{i,j}$.

Thus, the gradient $(\nabla u^{\pm})_{i,j}$ is the least-squares solution of the following linear system:

$$\underline{\underline{\boldsymbol{X}}}_{i,j}\underline{\underline{\boldsymbol{W}}}^{\pm}_{i,j}\left(\nabla u^{\pm}\right)_{i,j} = \underline{\underline{\boldsymbol{W}}}^{\pm}_{i,j}\begin{pmatrix} u_{i-1,j-1}-u^{\pm}_{i,j} \\ u_{i,j-1}-u^{\pm}_{i,j} \\ \cdots \\ u_{i+1,j+1}-u^{\pm}_{i,j}\end{pmatrix},$$

that is:

$$\left(\nabla u^{\pm}\right)_{i,j} = \underline{\underline{\boldsymbol{D}}}^{\pm}_{i,j}\begin{pmatrix} u_{i-1,j-1}-u^{\pm}_{i,j} \\ u_{i,j-1}-u^{\pm}_{i,j} \\ \cdots \\ u_{i+1,j+1}-u^{\pm}_{i,j}\end{pmatrix}, \quad \underline{\underline{\boldsymbol{D}}}^{\pm}_{i,j} = \left(\underline{\underline{\boldsymbol{X}}}^{T}_{i,j}\underline{\underline{\boldsymbol{W}}}^{\pm}_{i,j}\underline{\underline{\boldsymbol{X}}}_{i,j}\right)^{-1}\left(\underline{\underline{\boldsymbol{W}}}^{\pm}_{i,j}\underline{\underline{\boldsymbol{X}}}_{i,j}\right)^{T},$$

where the $3^{\mathcal{D}}\times\mathcal{D}$ and $3^{\mathcal{D}}\times 3^{\mathcal{D}}$ matrices $\underline{\underline{\boldsymbol{X}}}_{i,j}$ and $\underline{\underline{\boldsymbol{W}}}_{i,j}$ are given by:

$$\underline{\underline{\boldsymbol{X}}}_{i,j} = \begin{pmatrix} \left(\boldsymbol{r}_{i-1,j-1}-\boldsymbol{r}_{i,j}\right)^{T} \\ \left(\boldsymbol{r}_{i,j-1}-\boldsymbol{r}_{i,j}\right)^{T} \\ \cdots \\ \left(\boldsymbol{r}_{i+1,j+1}-\boldsymbol{r}_{i,j}\right)^{T}\end{pmatrix} \quad \text{and}$$

$$\underline{\underline{\boldsymbol{W}}}^{\pm}_{i,j} = \begin{pmatrix} \omega^{\pm}_{i,j}(-1,-1) & & & \\ & \omega^{\pm}_{i,j}(0,-1) & & \\ & & \ddots & \\ & & & \omega^{\pm}_{i,j}(1,1)\end{pmatrix},$$

$$\omega^{\pm}_{i,j}(p,q) = \begin{cases} 1,\ (p,q)\in N^{\pm}_{i,j} \\ 0,\ (p,q)\in N^{\mp}_{i,j}\end{cases}.$$

Substitution of $u^{\pm}_{I}(\boldsymbol{r})$ into (2.8) yields approximations of $\partial_{\boldsymbol{n}}u^{\pm}(\boldsymbol{r}^{\mathrm{pr}}_{i,j})$ as linear combinations of $\{u_{i+p,j+q}, p=-1,0,1, q=-1,0,1\}$. Specifically, let us write the $\mathcal{D}\times 3^{\mathcal{D}}$ matrix $\underline{\underline{\boldsymbol{D}}}^{\pm}_{i,j}$ as:

$$\underline{\underline{\boldsymbol{D}}}^{\pm}_{i,j} = \begin{pmatrix}\boldsymbol{d}^{\pm}_{i,j,-1,-1} & \boldsymbol{d}^{\pm}_{i,j,0,-1} & \cdots & \boldsymbol{d}^{\pm}_{i,j,1,1}\end{pmatrix},$$

where vectors $\boldsymbol{d}^{\pm}_{i,j,-1,-1}$, …, $\boldsymbol{d}^{\pm}_{i,j,1,1}$ represent the columns of the matrix $\underline{\underline{\boldsymbol{D}}}^{\pm}_{i,j}$. Then the

normal derivative can be expressed as:

$$\partial_{\boldsymbol{n}} u^{\pm}(\boldsymbol{r}_{i,j}^{\mathrm{pr}}) = c_{i,j}^{\pm} u_{i,j}^{\pm} + \sum_{(p,q)\in N_{i,j}^{\pm}} c_{i,j,p,q}^{\pm} u_{i+p,j+q} + \mathcal{O}(h), \tag{2.10}$$

where the coefficients are given by:

$$c_{i,j,p,q}^{\pm} = \boldsymbol{n}_{i,j}^{T} \boldsymbol{d}_{i,j,p,q}^{\pm}, \quad (p,q) \in N_{i,j}^{\pm}, \quad \text{and} \quad c_{i,j}^{\pm} = -\sum_{(p,q)\in N_{i,j}^{\pm}} c_{i,j,p,q}^{\pm}.$$

Substitution of (2.10) into (2.7) produces formulas expressing $u_{i,j}^{+}$ and $u_{i,j}^{-}$ in terms of the selected $N_x \times N_y$ unknowns $\{u_{p,q}, p \in [1, N_x], q \in [1, N_y]\}$. Combining them with the definition (2.5), we get the following two sets of rules (which are $\mathcal{O}(h^2)$ accurate in the value of u): one is based on approximating $\partial_{\boldsymbol{n}} u^{-}(\boldsymbol{r}_{i,j}^{\mathrm{pr}})$:

$$u_{i,j}^{-} = \begin{cases} u_{i,j}, & \boldsymbol{r}_{i,j} \in \Omega^{-}, \\ u_{i,j} - \alpha - \delta_{i,j}\frac{\beta}{\mu^{+}} + \delta_{i,j}\frac{[\mu]}{\mu^{+}} \dfrac{\left(c_{i,j}^{-}\left(u_{i,j} - \alpha - \delta_{i,j}\frac{\beta}{\mu^{+}}\right) + \sum_{(p,q)\in N_{i,j}^{-}} c_{i,j,p,q}^{-} u_{i+p,j+q}\right)}{\left(1 - \delta_{i,j}\frac{[\mu]}{\mu^{+}} c_{i,j}^{-}\right)}, & \boldsymbol{r}_{i,j} \in \Omega^{+}, \end{cases}$$

$$u_{i,j}^{+} = \begin{cases} u_{i,j} + \alpha + \delta_{i,j}\frac{\beta}{\mu^{+}} - \delta_{i,j}\frac{[\mu]}{\mu^{+}} \left(c_{i,j}^{-} u_{i,j} + \sum_{(p,q)\in N_{i,j}^{-}} c_{i,j,p,q}^{-} u_{i+p,j+q}\right), & \boldsymbol{r}_{i,j} \in \Omega^{-}, \\ u_{i,j}, & \boldsymbol{r}_{i,j} \in \Omega^{+}, \end{cases}$$

while the other one is based on approximating $\partial_{\boldsymbol{n}} u^{+}(\boldsymbol{r}_{i,j}^{\mathrm{pr}})$:

$$u_{i,j}^{-} = \begin{cases} u_{i,j}, & \boldsymbol{r}_{i,j} \in \Omega^{-}, \\ u_{i,j} - \alpha - \delta_{i,j}\frac{\beta}{\mu^{-}} + \delta_{i,j}\frac{[\mu]}{\mu^{-}} \left(c_{i,j}^{+} u_{i,j} + \sum_{(p,q)\in N_{i,j}^{+}} c_{i,j,p,q}^{+} u_{i+p,j+q}\right), & \boldsymbol{r}_{i,j} \in \Omega^{+}, \end{cases}$$

$$u_{i,j}^{+} = \begin{cases} u_{i,j} + \alpha + \delta_{i,j}\frac{\beta}{\mu^{-}} - \delta_{i,j}\frac{[\mu]}{\mu^{-}} \dfrac{\left(c_{i,j}^{+}\left(u_{i,j} + \alpha + \delta_{i,j}\frac{\beta}{\mu^{-}}\right) + \sum_{(p,q)\in N_{i,j}^{+}} c_{i,j,p,q}^{+} u_{i+p,j+q}\right)}{\left(1 + \delta_{i,j}\frac{[\mu]}{\mu^{-}} c_{i,j}^{+}\right)}, & \boldsymbol{r}_{i,j} \in \Omega^{-}, \\ u_{i,j}, & \boldsymbol{r}_{i,j} \in \Omega^{+}. \end{cases}$$

Thus, one has a certain flexibility in constructing the final discretization. For example, one could choose, for each $\boldsymbol{r}_{i,j}$, the formula based on approximating $\partial_{\boldsymbol{n}} u^{-}(\boldsymbol{r}_{i,j}^{\mathrm{pr}})$

or $\partial_{\boldsymbol{n}} u^+(\boldsymbol{r}_{i,j}^{\mathrm{pr}})$ depending on the largest number of neighboring points of $\boldsymbol{r}_{i,j}$ that are in Ω^- or in Ω^+ (let us denote this scheme as `Random`). However, this choice would ignore the magnitude of the diffusion constants μ^- and μ^+ and their influence on the condition number of the linear system. To investigate this issue, we consider two additional schemes: the first one (referred to as `Bias Fast`) uses interpolation in the fast-diffusion region (i.e., if $\mu^- > \mu^+$ then the formula based on $\partial_{\boldsymbol{n}} u^-(\boldsymbol{r}_{i,j}^{\mathrm{pr}})$ is used); the second scheme (referred to as `Bias Slow`) uses interpolation in the slow-diffusion region (i.e., if $\mu^- > \mu^+$ then we use the formula based on $\partial_{\boldsymbol{n}} u^+(\boldsymbol{r}_{i,j}^{\mathrm{pr}})$).

Remarks:

- In the limiting cases $\frac{\mu^-}{\mu^+} \to \infty$ or $\frac{\mu^-}{\mu^+} \to 0$, only the scheme `Bias Slow` remains well defined, thus, we expect it to perform the best and be well-conditioned for any ratio of diffusion coefficients. We will illustrate in section 2.3 that only the scheme `Bias Slow` produces a condition number that is bounded. This is consistent with the results reported in [178] describing a variant of the augmented immersed interface method.

- In the limiting case $\frac{\mu^-}{\mu^+} \equiv 1$, the three schemes coincide. Moreover, the matrix associated with the resulting linear system is the same as for the case when no interface is present (that is, as for the standard five-point stencil in 2D) and only the right-hand is changed to account for jump conditions.

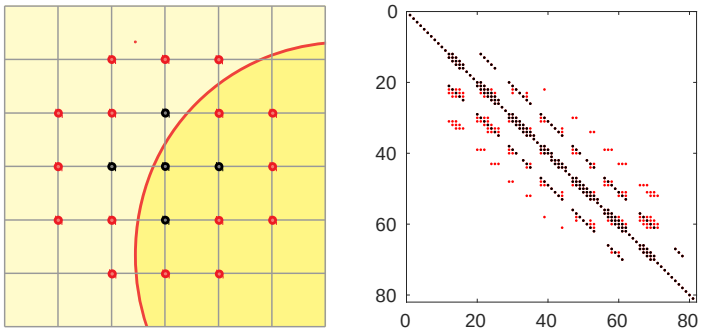
- The truncation error is the same for all three schemes. Therefore, we expect them

to have similar accuracies. Specifically, the truncation error[1] is $\mathcal{O}(h^2)$ for grid points away from the immersed interface and $\mathcal{O}(1)$ for cells crossed by the interface. Following the results of [80, 58, 141, 27, 115, 128, 53, 54, 18], we expect the schemes to produce second-order accurate numerical solutions with first-order accurate gradients.

- The truncation error can be improved to be $\mathcal{O}(h)$ for cells crossed by the interface by making the following changes in the discretization scheme: 1) Estimate fluxes between cells at the centroids of cell faces using linear interpolation as done, for example, in [80, 18]; 2) Retain the quadratic term in Taylor expansion (2.6); 3) Use a quadratic interpolant (instead of linear (2.9)) to approximate $\partial_{\boldsymbol{n}} u^{\pm}$ at projection points. This is expected to increase the accuracy of solution gradients to second order. We leave this investigation to future work.

- In general the case where $\mu^+ \neq \mu^-$, the resulting linear system is nonsymmetric. In the worst case scenario the computational stencil involves nearest neighbors (both in Cartesian and diagonal directions) of the standard five-point stencil as illustrated in Fig. 2.2a. An example of the matrix associated with the resulting linear system structure is shown in Fig. 2.2b.

[1] After scaling the resulting discretization by the cell volume $\mathcal{O}\left(h^{\mathcal{D}}\right)$ to account for the integration of the PDEs over a finite volume.



(a) (b)

Figure 2.2: (a) Computational stencil (the red color indicates additional grid points used in case $\mu^+ \neq \mu^-$). (b) Matrix structure of the resulting linear system in case of two-dimensional example from Sec. 2.3.1 on a 8^2 grid (the red color indicates additional elements in case $\mu^+ \neq \mu^-$).

2.3 Numerical tests

To numerically illustrate the properties of the proposed schemes, we study three characteristics: the order of accuracy of the numerical solution in the L^∞-norm, the order of accuracy of the numerical gradients in the L^∞-norm, and the condition number of the linear system, estimated by the MATLAB `condest` function. We consider two tests: the first one, the `convergence test`, studies the dependence of those three characteristics on the grid resolution. The second one, the `conditioning test`, focuses on the dependence of the three characteristics on the ratio, $\frac{\mu^-}{\mu^+}$, of the diffusion coefficients. We perform both tests in two and three spatial dimensions. In all the examples, we use the implementation of the BiCGStab algorithm provided by PETSc [10] with the Hypre preconditioner [45].

2.3.1 Two-dimensional case

Consider an annular region[2] with inner and outer radii $r_i = 0.151$ and $r_e = 0.911$, and an immersed star-shaped interface (see Fig. 2.3a), described by the following level-set function:

$$\phi(x,y) = \sqrt{x^2+y^2} - r_0\left(1 + \sum_{k=1}^{3} \beta_k \cos\left(n_k\left(\arctan\left(\frac{y}{x}\right) - \theta_k\right)\right)\right),$$

with parameters:

$$r_0 = 0.483,\ \begin{pmatrix} n_1 \\ \beta_1 \\ \theta_1 \end{pmatrix} = \begin{pmatrix} 3 \\ 0.1 \\ 0.5 \end{pmatrix},\ \begin{pmatrix} n_2 \\ \beta_2 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} 4 \\ -0.1 \\ 1.8 \end{pmatrix},\ \begin{pmatrix} n_3 \\ \beta_3 \\ \theta_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 0.15 \\ 0 \end{pmatrix}. \tag{2.11}$$

Using the method of manufactured solutions, we take the exact solution to be (see Fig. 2.3c):

$$\begin{aligned} u^- &= \sin(2x)\cos(2y), \\ u^+ &= \left(16\left(\frac{y-x}{3}\right)^5 - 20\left(\frac{y-x}{3}\right)^3 + 5\left(\frac{y-x}{3}\right)\right)\log(x+y+3). \end{aligned}$$

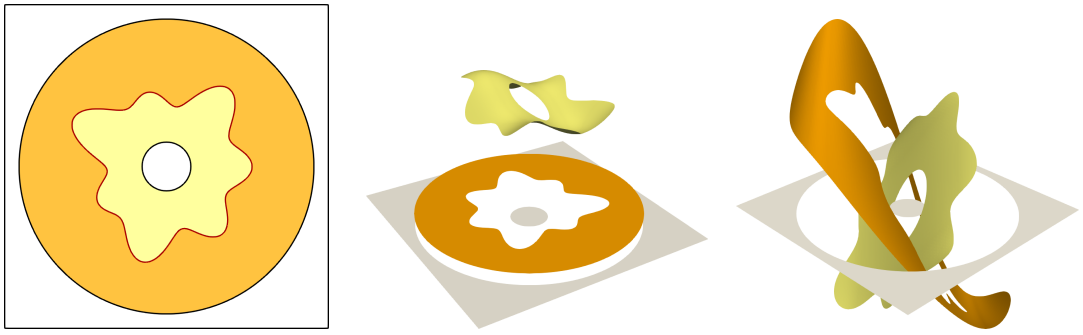
For the `convergence test`, we set the diffusion coefficients to

$$\mu^- = 10\left(1 + \frac{1}{5}\cos(2\pi(x+y))\sin(2\pi(x-y))\right) \quad \text{and} \quad \mu^+ = 1,$$

(see Fig. 2.3b), and we vary the grid resolution from 2^{-4} to 2^{-9}. For the `conditioning test`, we fix the grid resolution at 2^{-6} and $\mu^+ = 1$ and vary μ^- from 10^{-4} to 10^4. The

[2]We enclose an immersed interface inside another region in order to be able to obtain results for different placements of the immersed interface and the computational grid without changing the problem statement. On the boundaries of the enclosing region, Dirichlet boundary conditions can be imposed with any of the methods [58, 56, 147]

results are presented in Fig. 2.4 and 2.5, where each data point represents the maximum value among $10 \times 10 = 100$ different relative placements of the immersed interface on the computational grid (as done in [18]). The different placements thus account for cases where the interface defines a control volume that is arbitrarily small or large, relative to an elementary grid cell. Section 2.3.3 will draw some conclusions from these results.



(a) (b) (c)

Figure 2.3: (a) Problem geometry. (b) Diffusion coefficients (scaled by 0.1 for visualization). (c) Numerical solution on a 256^2 grid.

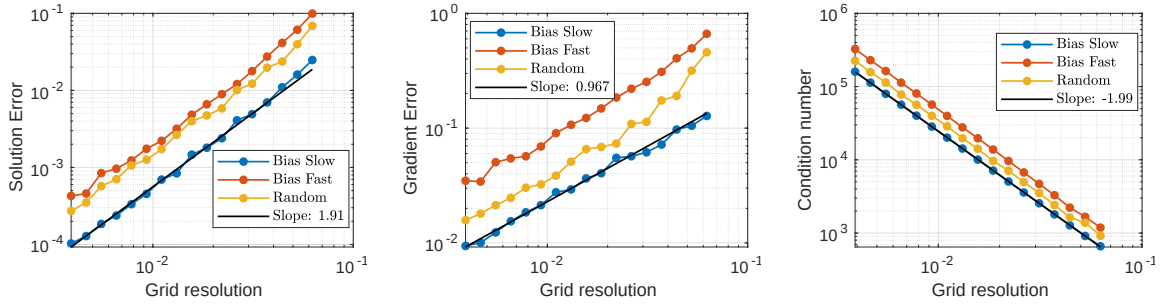


Figure 2.4: `convergence test` in two spatial dimensions (each data point represents the maximum value among $10 \times 10 = 100$ different relative placements of an immersed interface on the computational grid).

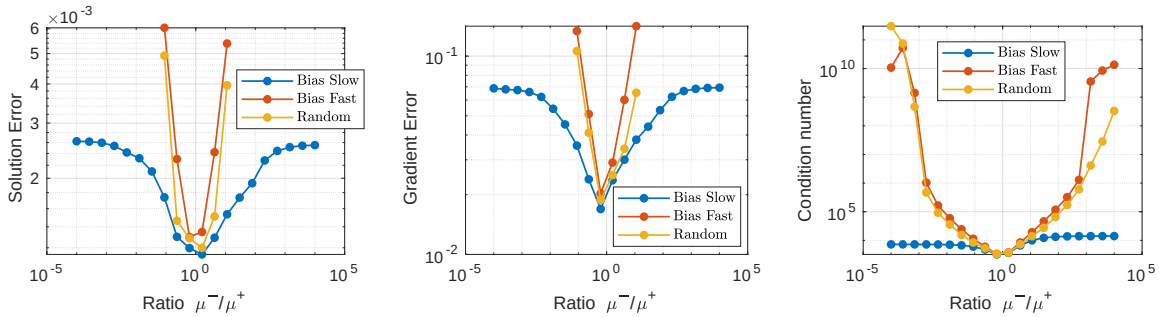


Figure 2.5: `conditioning test` in two spatial dimensions (each data point represents the maximum value among $10 \times 10 = 100$ different relative placements of an immersed interface on the computational grid).

2.3.2 Three-dimensional case

Consider a spherical shell[3] with inner and outer radii $r_i = 0.151$ and $r_e = 0.911$, and an immersed star-shaped interface described by the level-set function:

$$\phi(x,y,z) = \sqrt{x^2+y^2+z^2} - r_0\left(1 + \left(\frac{x^2+y^2}{x^2+y^2+z^2}\right)^2 \sum_{k=1}^{3} \beta_k \cos\left(n_k\left(\arctan\left(\frac{y}{x}\right) - \theta_k\right)\right)\right),$$

with the same parameters (2.11) as for the two-dimensional case. The problem geometry is illustrated in Fig. 2.6. The exact solutions are taken to be:

$$u^- = \sin(2x)\cos(2y)\exp(z),$$
$$u^+ = \left(16\left(\frac{y-x}{3}\right)^5 - 20\left(\frac{y-x}{3}\right)^3 + 5\left(\frac{y-x}{3}\right)\right)\log(x+y+3)\cos(z).$$

In the `convergence test`, the diffusion coefficients are set to:

[3] As in the two-dimensional case, Dirichlet boundary conditions are enforced on the boundaries of the enclosing region

$$\mu^- = 10\left(1 + \frac{1}{5}\cos(2\pi(x+y))\sin(2\pi(x-y))\cos(z)\right),$$
$$\mu^+ = 1.$$

In the `conditioning test`, the grid resolution is fixed at 2^{-4}, $\mu^+ = 1$ and μ^- is varied from 10^{-4} to 10^4. The test results are presented in Fig. 2.7 and 2.8 ,where each data point is obtained as the maximum (worse) value among $5 \times 5 \times 5 = 125$ different relative placements of the immersed interface on the computational grid. As for the two dimensional case, section 2.3.3 will draw some conclusions from these results.

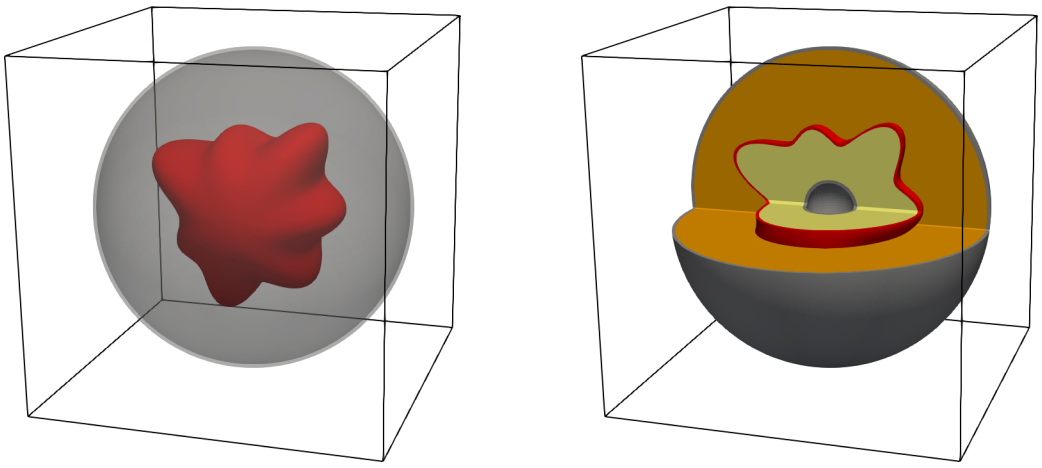


Figure 2.6: Illustration of problem geometry in the three-dimensional case.

2.3.3 Analysis

From the results presented in sections 2.3.1 and 2.3.2, it is clear that the numerical schemes have the same behavior in two and three spatial dimensions. The `convergence test` results (see Fig. 2.4 and 2.7) indicate that, for a moderate diffusion coefficient ratio, all three schemes have comparable convergence properties: the numerical solutions are

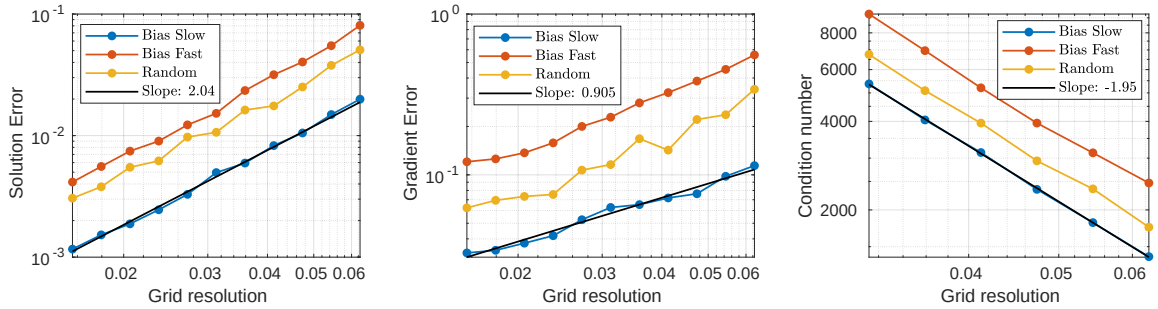


Figure 2.7: `convergence test` in three spatial dimensions (each data point represents maximum value among $5 \times 5 \times 5 = 125$ different relative placements of an immersed interface and the computational grid).

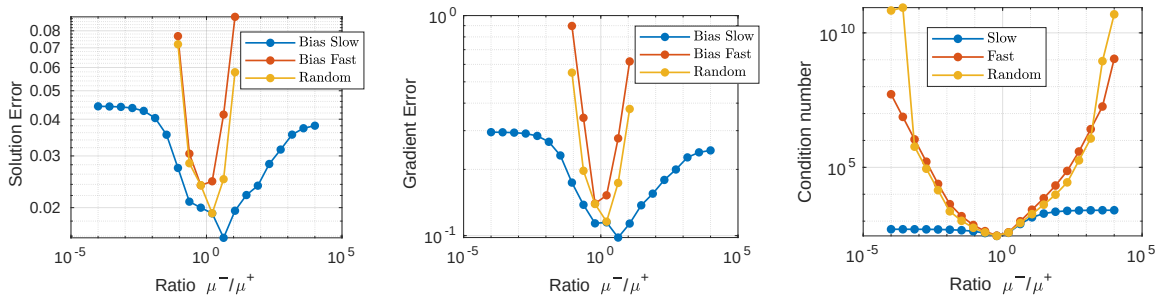


Figure 2.8: `conditioning test` in three spatial dimensions (each data point represents maximum value among $5 \times 5 \times 5 = 125$ different relative placements of an immersed interface and the computational grid).

second-order accurate with first-order accurate gradients in the L^∞-norm. The condition number scales with the grid resolution as h^{-2}, which is similar to the scaling of the condition number for the standard five-point (nine-point) stencil in 2D (3D). The only difference between the three schemes is the magnitude of the errors and the magnitude of the condition numbers, with the scheme `Bias Slow` giving the best results.

On the other hand, the `conditioning test` in two and three spatial dimensions demonstrate that the three schemes behaviors are drastically different when the ratio $\frac{\mu^-}{\mu^+}$ varies (Fig. 2.5 and 2.8). In particular, the condition numbers for the schemes `Random` and `Bias Fast` grow unboundedly as the ratio of the diffusion coefficients either decreases

or increases away from 1. As a result, the magnitude of the errors in the solution and its gradient grow significantly. We also note that, for approximately $\frac{\mu^-}{\mu^+} > 10$ and $\frac{\mu^-}{\mu^+} < 10^{-1}$, the linear solver is not able to invert the resulting linear system in a given number of iterations (we set that number to 50 in those numerical examples). In contrast, the condition number for the scheme `Bias Slow` converges to finite values as $\frac{\mu^-}{\mu^+} \to 0$ or $\frac{\mu^-}{\mu^+} \to \infty$. As a result, the linear solver is able to invert the resulting linear sytem for any values of $\frac{\mu^-}{\mu^+}$ (the number of iterations depends only on the grid resolution). Moreover, the errors of the numerical solutions and their gradients are only moderately affected by small or large ratios $\frac{\mu^-}{\mu^+}$.

2.3.4 Application to adaptive quadtree and octree grids

Thanks to its fairly small stencil, it is simple to apply the proposed scheme in the context of adaptive grids. In this section, we demonstrate perhaps the easiest way of doing so: we consider adaptive Cartesian quadtree (octree) grids that are locally uniform around immersed interfaces. In the regions where an adaptive grid is non-uniform (and which are away from immersed interfaces), we use the second-order accurate superconvergent finite difference scheme of [100], while in the regions close to immersed interfaces (and where the grid is locally uniform), we use the proposed scheme for imposing interface jump conditions. Note that the discretization of [100] and the one described in this work reduce to the standard 5-point (9-point in three spatial dimensions) stencil on uniform

grids and in the absence of immersed interfaces. This fact makes the combination of the two discretizations seamless.

In this example, we consider 10 clusters of small star-shaped uniformly charged dielectric particles in a 2m×2m vacuum domain and compute the electric field generated by this configuration. The number of particles in each cluster varies from 3 to 10, the average size of clusters is 1 cm, particles have between 2 and 6 bumps and sizes in the range $[0.1; 1]$ mm. Specifically, the total number of particles 67 each of which is described by a level-set function of the form:

$$\phi(r,\theta) = \sqrt{(r\cos(\theta-\theta_0)-x_c)^2 + (r\sin(\theta-\theta_0)-y_c)^2} - r_0\left(1+\delta\cos(m(\theta-\theta_0))\right).$$

Parameters x_c, y_c, r_0, δ, m and θ_0 for each particle as well as the sign of its charge are given in the following table:

No.	x_c	y_c	r_0	δ	m	θ_0	Charge
1	0.294320	-0.731980	0.000948	0.177428	5	3.779031	+
2	0.292603	-0.722570	0.000196	0.120466	2	0.575194	+
3	0.296621	-0.730048	0.000553	0.195375	3	5.983689	+
4	0.301392	-0.723932	0.000374	0.161709	3	1.178485	+
5	0.293268	-0.730192	0.000798	0.193548	3	6.112317	-
6	0.289541	-0.732616	0.000755	0.088156	4	2.266495	-
7	0.290527	-0.736414	0.000341	0.140005	3	6.140219	-
8	0.295213	-0.726113	0.000792	0.121836	5	0.884964	-

9	0.300884	-0.733236	0.000857	0.078646	4	1.588758	-
10	0.935243	0.156877	0.000462	0.026599	4	1.152778	+
11	0.936367	0.165333	0.000946	0.129853	3	2.585702	+
12	0.937713	0.161907	0.000114	0.111576	3	3.069164	+
13	0.927721	0.160662	0.000653	0.059185	4	4.986538	-
14	0.926407	0.164860	0.000243	0.009826	2	3.613796	-
15	0.939303	0.170991	0.000443	0.141560	3	2.988958	+
16	0.926755	0.162948	0.000482	0.025398	5	3.380732	+
17	0.936908	0.160965	0.000713	0.179751	3	0.252419	+
18	0.934454	0.162162	0.000448	0.011111	6	2.899990	+
19	0.160228	-0.743358	0.000329	0.052804	6	4.507911	+
20	0.164293	-0.747939	0.000792	0.015578	3	3.753404	+
21	0.173756	-0.744021	0.000264	0.186352	4	6.101247	+
22	0.168733	-0.761421	0.000999	0.001466	5	4.162976	-
23	0.170695	-0.747981	0.000389	0.025428	4	0.952773	-
24	0.160737	-0.760843	0.000209	0.199829	6	0.814897	-
25	0.725447	0.889409	0.000966	0.064583	3	5.719127	-
26	0.740474	0.893693	0.000453	0.004282	3	5.779404	-
27	0.736547	0.884784	0.000708	0.036598	4	0.871440	-
28	0.723810	0.875899	0.000367	0.044844	3	0.309913	-
29	0.742820	0.874970	0.000306	0.191154	4	1.616747	-

30	0.728138	0.878376	0.000857	0.026387	4	5.913726	-
31	0.728254	0.880038	0.000451	0.069948	6	2.499319	+
32	0.052907	-0.344586	0.000660	0.184134	4	2.412480	-
33	0.034801	-0.356601	0.000834	0.079140	4	4.794653	-
34	0.048034	-0.350902	0.000961	0.175580	4	1.000997	-
35	0.038434	-0.350375	0.000546	0.119999	5	5.330543	-
36	0.037001	-0.361473	0.000859	0.104759	6	2.104655	-
37	0.048676	-0.344655	0.000590	0.018488	3	3.979605	-
38	-0.182697	0.433937	0.000642	0.191264	5	0.196232	-
39	-0.180690	0.449481	0.000499	0.107538	4	3.958288	+
40	-0.171622	0.442325	0.000714	0.162207	3	4.889946	-
41	-0.177232	0.435290	0.000706	0.108419	6	2.765306	+
42	-0.182553	0.446933	0.000927	0.139933	5	0.682710	-
43	-0.169332	0.440621	0.000652	0.197255	6	0.524252	-
44	0.082096	0.665721	0.000309	0.018026	6	3.925875	+
45	0.070903	0.669204	0.000229	0.189876	3	1.973639	+
46	0.086667	0.652859	0.000256	0.009732	4	5.337312	-
47	0.089332	0.668903	0.000778	0.189929	4	0.506358	+
48	-0.416502	-0.873554	0.000168	0.038622	2	4.772787	+
49	-0.435866	-0.873537	0.000167	0.040327	6	4.357852	+
50	-0.428412	-0.875426	0.000329	0.067637	4	4.422504	-

51	0.367410	-0.528637	0.000239	0.004636	6	1.018504	-
52	0.364188	-0.533808	0.000271	0.072883	3	2.058938	-
53	0.357943	-0.522965	0.000682	0.097692	3	4.725618	+
54	0.354170	-0.530541	0.000809	0.035313	5	0.726723	-
55	0.355478	-0.527060	0.000107	0.166186	4	5.450653	+
56	0.356304	-0.525395	0.000985	0.004988	2	3.471888	+
57	0.368282	-0.530008	0.000107	0.133467	5	5.475823	-
58	0.370185	-0.524812	0.000830	0.145549	5	6.126808	-
59	0.352045	-0.519527	0.000198	0.011507	4	3.886958	+
60	-0.187080	0.207027	0.000928	0.121431	2	1.730282	+
61	-0.177875	0.213721	0.000126	0.001767	3	0.076936	+
62	-0.177947	0.216082	0.000647	0.130215	4	6.002824	-
63	-0.193546	0.211842	0.000749	0.088286	6	4.366932	-
64	-0.190933	0.205428	0.000170	0.122096	3	0.951230	-
65	-0.184008	0.210547	0.000752	0.030592	3	3.986605	-
66	-0.192313	0.214133	0.000925	0.029170	4	1.284015	-
67	-0.190761	0.223399	0.000728	0.195173	4	5.881409	-

The absolute permittivity of particles is $10\varepsilon_0$ and their charge densities are $\pm 10^5\varepsilon_0$ (the sign is assigned to each particle randomly), where ε_0 is the vacuum permittivity. The domain boundaries are assumed to be a good conductor. Thus, the electric potential $\varphi \equiv u$ satisfies the boundary value problem (2.1)-(2.3) with parameters $k^{\pm} = \alpha = \beta = 0$,

$\mu^- = 10$, $\mu^+ = 1$, $f^- = \pm 10^5$ (depending which particle is considered), $f^+ = 0$, $g = 0$, where we denote the particles as Ω^- and the vacuum as Ω^+.

Figure 2.9 depicts the electric potential ϕ and the electric field $\boldsymbol{E} = -\nabla\phi$ computed on an adaptive grid with the coarsest and finest mesh sizes corresponding to uniform resolutions of $2^{10} \times 2^{10}$ and $2^{20} \times 2^{20}$ grid points, respectively. Such a computational grid contains 2754021 points, which is approximately just 0.00025% of the total number of points in a uniform $2^{20} \times 2^{20}$ grid.

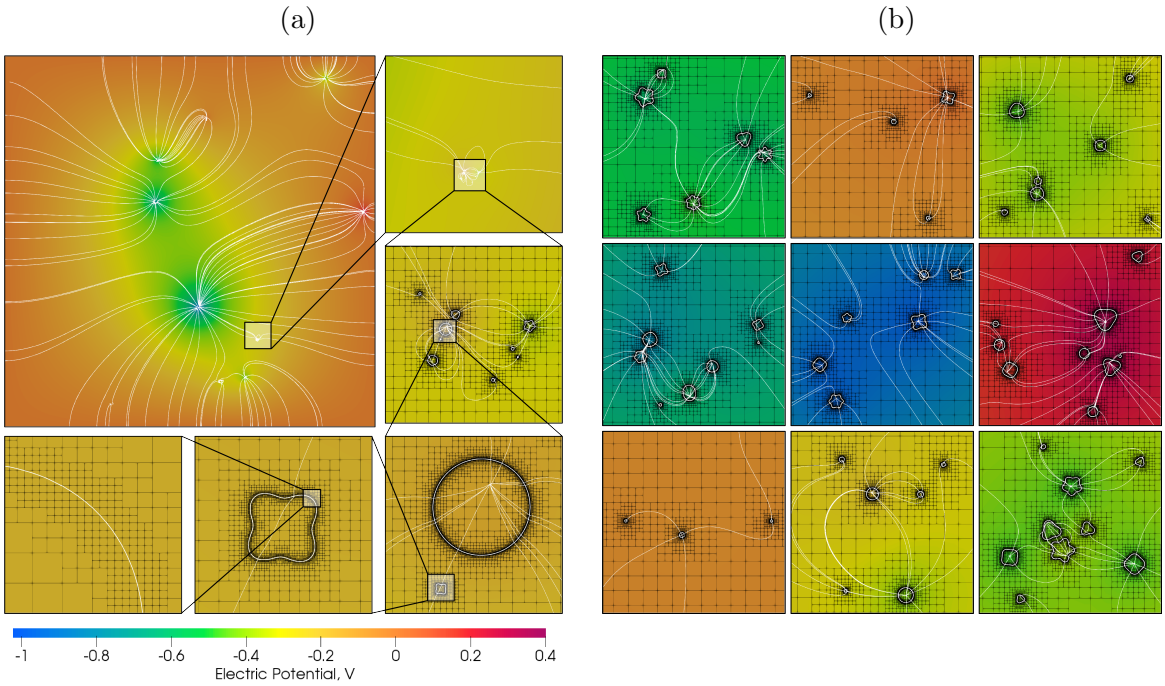


Figure 2.9: Application of the present method in the context of adaptive grids for computing electric field around clusters of charged dielectric particles: (a) The entire computational domain with successive zoom-ins for one of the clusters; (b) Zoom-ins for all other clusters. The color map indicates the electric potential and thin lines represent the electric field lines. The thicker white lines represent particles.

Remark: While the strategy of enforcing adaptive grids to be locally uniform around immersed interfaces is quite common and justified from the point of view of accuracy in

many situations, it may not be very efficient for certain applications; in particular those where parts of the interface has high curvature while other parts are rather flat. However, it seems straightforward to adapt the described method to fully-adaptive (non-uniform along immersed interfaces) and Voronoi grids (including Voronoi partitions generated based on adaptive quadtree and octree grids). This will be considered in future works.

2.4 Conclusions

We have presented a simple finite volume numerical method for solving Elliptic equations with jump conditions across irregular interfaces that are implicitly represented by a level-set function on Cartesian grids. Second-order accurate solutions and first-order accurate gradients are obtained in the L^∞-norm. The linear system is non-symmetric but the condition number is bounded, regardless of the ratio of the diffusion coefficients, so that the linear system can be inverted in a constant number of iterations that depends only on the grid resolution: the condition number scales as $\mathcal{O}\left(h^{-2}\right)$, similarly to the linear system obtained from the standard five-point stencil.

Future work will be focused on the analysis of the numerical scheme and its properties, the improvement to a superconvergent scheme (i.e, with second-order accurate gradients) and the extension to fully-adaptive (non-uniform along immersed interfaces) grids.

Chapter 3

PDE-Based Extrapolation of Scalar Fields over Piecewise Smooth Interfaces

3.1 Introduction

Extrapolation procedures are ubiquitous in scientific computing and generally allow one to estimate a valid value of a quantity at points where data is not given; either in space or in time. In the context of level-set methods [122], extrapolation procedures in space have been frequently used since the advent of the ghost-fluid method [46], where constant extrapolations were originally used. Generalized ghost-fluid methods were then designed, in part based on higher-order extrapolations for which Aslam introduced a partial dif-

ferential equation (PDE) approach to perform linear and quadratic extrapolation [6] and Gibou and Fedkiw introduced a cubic extrapolation in the same PDE framework [56]. It is natural in the level-set context to perform such extrapolations using PDE formulations for their solutions are based on Hamilton-Jacobi solvers that have been designed for other standard level-set equations, see e.g. [157]. A typical situation that needs extrapolation is that of an implicit treatment of a field in a free boundary problem. In this case, a valid value of the field at time t^n needs to be known when assembling the right-hand side of the linear system of equations at time t^{n+1}. Since the interface at the new time step has swept grid points that are outside the domain at the previous time step, valid values of the field at time t^n are needed in the domain at time t^{n+1}, which requires an extrapolation procedure.

Typical use of extrapolation methods can be found in a multitude of level-set applications including multiphase flow simulations [96, 55, 90, 117, 38, 60, 133], in the solution of Poisson-Boltzmann [106, 71] and Poisson-Nerntz-Planck equations [103] for studying transport in ionic solutions, in heat and diffusion flow problems [56, 58, 18, 128], in the study of epitaxial growth and diblock-copolymer self-assembly used in the semiconductor industry [129, 124], in shape optimization [4, 158], surface reconstruction of biomolecules [106, 42] and in Stefan-type problems [57, 28, 107, 136]. PDE-based extrapolation procedures have also been extended to adaptive Quad-/Oc-tree grids and parallel architectures [100, 104, 61]. In addition, fast methods have been introduced for computationally efficient extrapolation procedures using the Fast Marching method, including

parallel implementations [145, 146, 26] or the Fast Sweeping method [181, 7], including efficient parallel algorithms on adaptive grids [40, 39]. We also refer the interested reader to [112] for another implicit approach to extrapolation based on solving the biharmonic equation.

However, those methods behave poorly in the case where the free boundary presents high-curvature features or kinks. Typical examples of such situations are multimaterial flows with triple junction points, motion of sharp-edged bodies in fluids, contact line dynamics in wetting phenomena, phase-change front propagation in the presence of confining walls, etc. We introduce a method that solves that problem. We present the method in section 3.2 and numerical examples in sections 3.3 and 3.4 that illustrate its benefits and comment on its efficiency. Section 3.5 considers an example of solving a diffusion equation on evolving domains that demonstrates the importance of accurate extrapolation near sharp geometric features. Section 3.6 draws some conclusions.

3.2 Numerical Method

3.2.1 Level-set Representation

The level set representation [122] defines the interface of a domain by $\{\boldsymbol{x} : \phi(\boldsymbol{x}) = 0\}$, its interior and exterior by $\phi(\boldsymbol{x}) < 0$ and $\phi(\boldsymbol{x}) > 0$, respectively, where $\phi(\boldsymbol{x})$ is a Lipschitz continuous function called *the level-set function*. In this work, the only geometrical quantity that is needed is the outward normal to the interface, $\boldsymbol{n}$, which can be computed

as:

$$\boldsymbol{n} = \frac{\nabla \phi}{|\nabla \phi|}, \tag{3.1}$$

using central differencing for ϕ_x and ϕ_y. In typical level-set simulations, the level-set function is reinitialized as a signed distance function [157]. We refer the interested reader to [144, 120] for a thorough presentation of the level-set method and [59] for a recent review.

3.2.2 Normal-derivative based multidimensional PDE extrapolation of [6]

High order extrapolations in the normal direction are traditionally performed in a series of steps, as proposed by Aslam in [6] and referred to in the present manuscript as the *normal-derivative based partial differential equation (ND-PDE) extrapolation*. For example, suppose that we seek to extrapolate a scalar field q from the region where $\phi \leq 0$ to the region where $\phi > 0$. In the case of a quadratic extrapolation, we first compute $q_{\boldsymbol{nn}} = \nabla (\nabla q \cdot \boldsymbol{n}) \cdot \boldsymbol{n}$ in the region $\phi \leq 0$ and extrapolate it across the interface in a constant fashion, that is, such that its normal derivative is zero in the region $\phi > 0$, by solving the following partial differential equation:

$$\frac{\partial q_{\boldsymbol{nn}}}{\partial \tau} + H(\phi) (\boldsymbol{n} \cdot \nabla q_{\boldsymbol{nn}}) = 0, \tag{3.2}$$

where H is the Heaviside function. Then, the value of q across the interface is found by solving the following two partial differential equations:

$$\frac{\partial q_{\boldsymbol{n}}}{\partial \tau} + H(\phi)\left(\boldsymbol{n} \cdot \nabla q_{\boldsymbol{n}} - q_{\boldsymbol{nn}}\right) = 0, \tag{3.3}$$

$$\frac{\partial q}{\partial \tau} + H(\phi)\left(\boldsymbol{n} \cdot \nabla q - q_{\boldsymbol{n}}\right) = 0, \tag{3.4}$$

defining $q_{\boldsymbol{n}}$ in such a way that its normal derivative is equal to the previously extrapolated $q_{\boldsymbol{nn}}$ and then defining q in such a way that its normal derivative is equal to the previously extrapolated $q_{\boldsymbol{n}}$. These PDEs are solved in fictitious time τ for a few iterations (typically 15) since we only seek to extrapolate the values of q in a narrow band of a few grid cells around the interface.

This extrapolation procedure produces accurate results in the case where the interface is smooth, but generates large error in the case where sharp geometric features occur, e.g. thin elongated shapes or interfaces with kinks as illustrated in sections 3.3 and 3.4.

3.2.3 Weighted-Cartesian-derivative based multidimensional PDE extrapolation

Instead of calculating the normal derivatives in the negative region before extrapolating them, we instead compute the derivatives in the Cartesian directions, extrapolate them and then construct the normal derivatives. Specifically, consider the following quantities, that are computed in the negative level-set region:

$$\boldsymbol{q}_{\nabla} = \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix} \quad \text{and the symmetric matrix} \quad \underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla} = \begin{pmatrix} q_{xx} & q_{xy} & q_{xz} \\ q_{xy} & q_{yy} & q_{yz} \\ q_{xz} & q_{zy} & q_{zz} \end{pmatrix}.$$

Similar to the method described in the previous section, we extrapolate the elements of $\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}$ in a constant fashion:

$$\frac{\partial \underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}}{\partial \tau} + H(\phi)\left(\boldsymbol{n}\cdot\nabla\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}\right) = 0, \tag{3.5}$$

before successively solving the following equations:

$$\frac{\partial \boldsymbol{q}_{\nabla}}{\partial \tau} + H(\phi)\left(\boldsymbol{n}\cdot\left(\nabla\boldsymbol{q}_{\nabla} - \underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}\right)\right) = 0, \tag{3.6}$$

$$\frac{\partial q}{\partial \tau} + H(\phi)\left(\boldsymbol{n}\cdot\left(\nabla q - \boldsymbol{q}_{\nabla}\right)\right) = 0. \tag{3.7}$$

Note that now, the normal vector field $\boldsymbol{n}$ enters the equations merely as some sort of weighting factor. Thus, as long as field q is sufficiently smooth this approach to multidimensional extrapolation is expected to produce accurate results even when the normal vector field $\boldsymbol{n}$ is not smooth (as is the case of domains with sharp features). To distinguish the proposed approach from the one in [6], we refer to it as the *weighted-Cartesian-derivative based partial differential equation (WCD-PDE)* extrapolation.

Remark: It is possible to construct cubic and even higher-order extrapolations following this approach as well, however one needs to keep in mind the rapidly growing computational cost, because an m-th order method requires solving advection equations for tensor variables of order up to m ($3\times3\times3$ for cubic, $3\times3\times3\times3$ for quartic, etc).

3.2.4 Implementation details

In this work we demonstrate the proposed method on uniform Cartesian grids and our implementation follows very closely the one from [6] with just few differences. Consider a two dimensional computational grid with nodes defined as:

$$\boldsymbol{r}_{i,j} = \begin{pmatrix} x_{\min} + (i-1)\Delta x \\ y_{\min} + (j-1)\Delta y \end{pmatrix}, \quad i \in [1; N_x],\ j \in [1; N_y],$$
$$\Delta x = \frac{x_{\max} - x_{\min}}{N_x - 1}, \quad \Delta y = \frac{y_{\max} - y_{\min}}{N_y - 1},$$

where $[x_{\min}; x_{\max}] \times [y_{\min}; y_{\max}]$ denotes the computational domain, N_x and N_y are number of grid nodes in the Cartesian directions. Standard second-order accurate central difference formulas are used for calculating the normal vector field $\boldsymbol{n}(\boldsymbol{r})$ (in the entire domain) and derivatives (first and second) of q in the negative region. Normal derivatives of q are computed as:

$$q_{\boldsymbol{n}} = \nabla q \cdot \boldsymbol{n} \quad \text{and} \quad q_{\boldsymbol{nn}} = \boldsymbol{n} \cdot \nabla\nabla q \cdot \boldsymbol{n} + \boldsymbol{n} \cdot \nabla \boldsymbol{n} \cdot \nabla q.$$

Since the first and second order derivatives of q are not well-defined at all grid points where $\phi < 0$ we replace the Heaviside function $\mathcal{H}(\phi)$ in equations (3.3), (3.6) and in equations (3.2), (3.5) with discrete fields $\mathcal{H}^{\phi,\nabla}$ and $\mathcal{H}^{\phi,\nabla\nabla}$, respectively, where:

$$\mathcal{H}^{\phi,\nabla}_{i,j} = \begin{cases} 0, \text{ if } \phi \le 0 \text{ at } \boldsymbol{r}_{i\pm1,j},\ \boldsymbol{r}_{i,j\pm1}, \\ 1, \text{ otherwise,} \end{cases}$$
$$\mathcal{H}^{\phi,\nabla\nabla}_{i,j} = \begin{cases} 0, \text{ if } \phi \le 0 \text{ at } \boldsymbol{r}_{i\pm1,j},\ \boldsymbol{r}_{i,j\pm1},\ \boldsymbol{r}_{i\pm1,j\pm1},\ \boldsymbol{r}_{i\pm1,j\mp1} \\ 1, \text{ otherwise.} \end{cases}$$

Applying an explicit first-order accurate in time discretization to equations (3.2)-(3.7) one obtains the following updating formulas:

$$\begin{aligned}
[q_{\boldsymbol{nn}}]_{i,j}^{k+1} &= [q_{\boldsymbol{nn}}]_{i,j}^{k} - \Delta\tau \mathcal{H}_{i,j}^{\phi,\nabla\nabla}\left([\boldsymbol{n}\cdot\nabla q_{\boldsymbol{nn}}]_{i,j}^{k}\right), \\
[q_{\boldsymbol{n}}]_{i,j}^{k+1} &= [q_{\boldsymbol{n}}]_{i,j}^{k} - \Delta\tau \mathcal{H}_{i,j}^{\phi,\nabla}\left([\boldsymbol{n}\cdot\nabla q_{\boldsymbol{n}}]_{i,j}^{k} - [q_{\boldsymbol{nn}}]_{i,j}\right), \\
[q]_{i,j}^{k+1} &= [q]_{i,j}^{k} - \Delta\tau \mathcal{H}_{i,j}^{\phi}\left([\boldsymbol{n}\cdot\nabla q]_{i,j}^{k} - [q_{\boldsymbol{n}}]_{i,j}\right),
\end{aligned} \tag{3.8}$$

and

$$\begin{aligned}
\left[\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}\right]_{i,j}^{k+1} &= \left[\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}\right]_{i,j}^{k} - \Delta\tau \mathcal{H}_{i,j}^{\phi,\nabla\nabla}\left(\left[\boldsymbol{n}\cdot\nabla\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}\right]_{i,j}^{k}\right), \\
[\boldsymbol{q}_{\nabla}]_{i,j}^{k+1} &= [\boldsymbol{q}_{\nabla}]_{i,j}^{k} - \Delta\tau \mathcal{H}_{i,j}^{\phi,\nabla}\left([\boldsymbol{n}\cdot\nabla\boldsymbol{q}_{\nabla}]_{i,j}^{k} - \left[\boldsymbol{n}\cdot\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}\right]_{i,j}\right), \\
[q]_{i,j}^{k+1} &= [q]_{i,j}^{k} - \Delta\tau \mathcal{H}_{i,j}^{\phi}\left([\boldsymbol{n}\cdot\nabla q]_{i,j}^{k} - [\boldsymbol{n}\cdot\boldsymbol{q}_{\nabla}]_{i,j}\right),
\end{aligned} \tag{3.9}$$

When extrapolating first- and second-order derivatives (i.e. $q_{\boldsymbol{n}}$, $q_{\boldsymbol{nn}}$, $\boldsymbol{q}_{\nabla}$ and $\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}$), first-order spatial derivatives in the equations above are computed using first-order accurate upwind discretizations. For example, derivatives in the x-direction are approximated as:

$$[n_x\partial_x f]_{i,j}^{k} = \begin{cases} [n_x]_{i,j}\dfrac{[f]_{i,j}^{k} - [f]_{i-1,j}^{k}}{\Delta x} + \mathcal{O}(\Delta x), \text{ if } [n_x]_{i,j} > 0, \\ [n_x]_{i,j}\dfrac{[f]_{i+1,j}^{k} - [f]_{i,j}^{k}}{\Delta x} + \mathcal{O}(\Delta x), \text{ if } [n_x]_{i,j} < 0, \end{cases} \tag{3.10}$$

where f is $q_{\boldsymbol{n}}$, $q_{\boldsymbol{nn}}$, $\boldsymbol{q}_{\nabla}$ or $\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}$. This is sufficient to achieve second-order accuracy in the extended fields $q_{\boldsymbol{n}}$ and $\boldsymbol{q}_{\nabla}$. For extrapolation of the field q itself (last equations in (3.8) and (3.9)), however, second-order accurate upwind discretizations are used. For example, derivatives in the x-direction are approximated as:

$$[n_x \partial_x q]_{i,j} = \begin{cases} [n_x]_{i,j} \left(\dfrac{[q]_{i,j} - [q]_{i-1,j}}{\Delta x} + \dfrac{\Delta x}{2} \text{minmod} \left([q_{xx}]_{i,j}, [q_{xx}]_{i-1,j} \right) \right) \\ \qquad\qquad + \mathcal{O}\left(\Delta x^2\right), \text{ if } [n_x]_{i,j} > 0, \\ [n_x]_{i,j} \left(\dfrac{[q]_{i+1,j} - [q]_{i,j}}{\Delta x} - \dfrac{\Delta x}{2} \text{minmod} \left([q_{xx}]_{i,j}, [q_{xx}]_{i+1,j} \right) \right) \\ \qquad\qquad + \mathcal{O}\left(\Delta x^2\right), \text{ if } [n_x]_{i,j} < 0, \end{cases} \tag{3.11}$$

where

$$\text{minmod}(a,b) = \begin{cases} 0, \text{ if } ab \leq 0, \\ a, \text{ if } |a| \leq |b|, \\ b, \text{ if } |b| \leq |a|. \end{cases}$$

Derivatives in the y-directions are approximated in a similar fashion. We note that approximation of derivatives is done in the same way for both, ND-PDE and WCD-PDE, extrapolation methods. The difference between the approaches lies in which quantities are extended over interfaces.

Since in the new method the approximation of second-order derivatives in all Cartesian directions are already available during solving the PDE for q, the minmod corrections in (3.11) can be computed only once during first iteration and reused in subsequent iterations, reducing the cost of each iteration by approximately 2 times. Specifically, the total count of arithmetic operations to compute $[q]_{i,j}^{k+1}$ using the ND-PDE method is approximately 22 in two spatial dimensions and 32 in three spatial dimensions, while for the WCD-PDE method the total count is 10 and 14, correspondingly. Thus, if we denote as T the cost of solving a single advection equation using first-order accurate approximations of derivatives, then the total cost of performing quadratic extrapolation

using the ND-PDE method is approximately $(1+1+2)T = 4T$ in two and three spatial dimensions, while the total cost of performing quadratic extrapolation using the WCD-PDE method is $(3+2+1)T = 5T$ in two spatial dimensions and $(6+3+1)T = 10T$ in three spatial dimensions.

Equations (3.8) and (3.9) are iterated in the fictitious time τ until steady-state. Time step $\Delta\tau$ is chosen based on consideration of satisfying the CFL condition as:

$$\Delta\tau = \frac{\min(\Delta x, \Delta y)}{2} \quad \text{and} \quad \Delta\tau = \frac{\min(\Delta x, \Delta y, \Delta z)}{3}$$

in two and three spatial dimensions, respectively. Iterations are terminated when the maximum difference between two successive steps $\max_{i,j} \left| [f]_{i,j}^{k+1} - [f]_{i,j}^{k} \right|$ within the band of interest, that is, among all grid nodes within the distance of $2\sqrt{\Delta x^2 + \Delta y^2}$ (or $2\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ in three spatial dimensions) around the domain boundary, is less than a specified tolerance $\epsilon_{\text{tol}} = 10^{-12}$.

Remark. Since in the proposed approach there is no need to recalculate second derivatives and apply the nonlinear minmod operator at every iteration, it is possible to obtain the steady-state solution of the advection equations in an implicit fashion. This could be very beneficial in cases when a good guess for the extended field is available (for example, solutions from preceding time instants in time-dependent problems). Such an approach will be explored in future works.

Remark. In case of linear extrapolations, the first equations in (3.8) and (3.9) are not solved, in second equations $[q_{\boldsymbol{nn}}]_{i,j}$ and $\left[\underline{\underline{\boldsymbol{Q}}}_{\nabla\nabla}\right]_{i,j}$ are set to zero and first-order accurate

formulas (3.10) are used during the extrapolation of both field q and its derivatives for more efficient computations.

Extension to adaptive Quad-/Oc-tree grids

The methodology introduced in this work can be trivially extended to Quad-/Oc-tree data structures. Specifically, we sample data fields at nodes of Quad-/Oc-tree grids and use the second-order accurate discretizations of [100] for regions where grids are non-uniform. A band of uniform grids are usually imposed near the interface in practical free boundary applications (see Fig. 3.1). In this case the extrapolation within some neighborhood around the interface (where it is primarily required) is as accurate as for uniform grids, however, the extrapolation procedure is much faster on adaptive grids for their significant reduction in the total number of grid points.

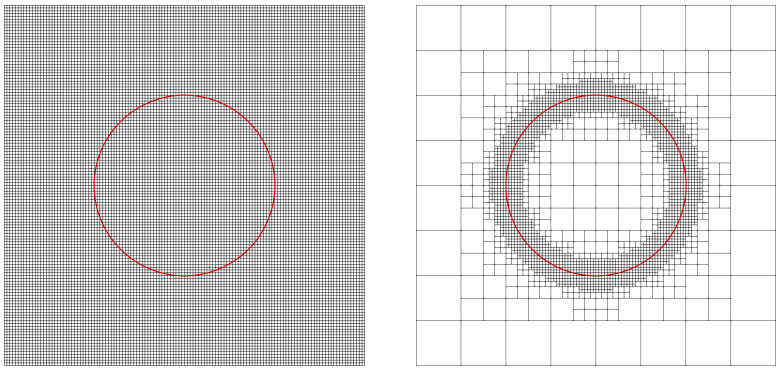
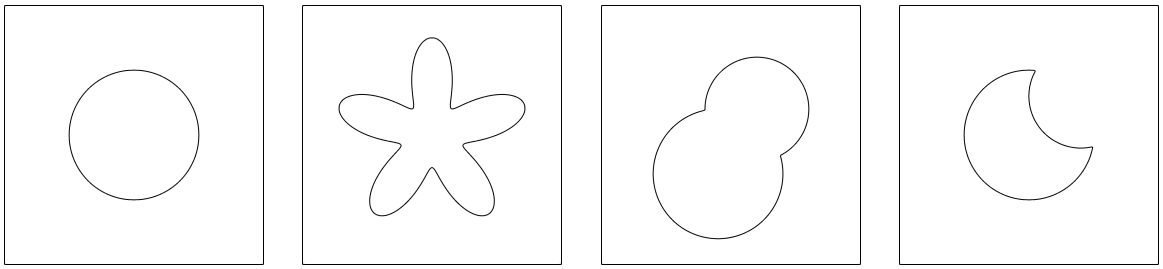


Figure 3.1: *Examples of uniform (left) and adaptive (right) Cartesian grids for a circular interface.*

3.3 Numerical Results in Two Spatial Dimensions

We consider four physical domains: a disk, a star shape, a union of two disks and an intersection of two disks (see figure 3.2). The disk is a smooth interface for which the approach of [6] performs well. The star-shape domain is an example where regions of high curvature are present (crest and trough of the wavy shape). The union/intersection of two disks are examples where kinks occur and illustrate the case of typical free boundary simulations where changes in topology occur. The definition of those domains are given by the level-set functions:

$$\begin{aligned}
\phi_0(x,y) &= \sqrt{x^2+y^2} - 0.501, \\
\phi_1(x,y) &= \sqrt{x^2+y^2} - 0.501 - 0.25\frac{y^5 + 5x^4y - 10x^2y^3}{(x^2+y^2)^{\frac{5}{2}}}, \\
\phi_2(x,y) &= \min\Big(\sqrt{(x+.1)^2+(y+.3)^2} - 0.501, \\
&\qquad\qquad \sqrt{(x-.2)^2+(y-.2)^2} - 0.401\Big), \\
\phi_3(x,y) &= \max\left(\sqrt{x^2+y^2} - 0.501, \sqrt{(x-.4)^2+(y-.3)^2} - 0.401\right),
\end{aligned}$$



(a) Disk, Ω_0 (b) Star, Ω_1 (c) Union, Ω_2 (d) Intersection, Ω_3

Figure 3.2: *Computational domains considered in section 3.3.*

In each case we consider a computational domain $\Omega = (-1,1)\times(-1,1)$. We define the function $q = \sin(\pi x)\cos(\pi y)$ inside every domain and extrapolate it in the outside region.

Then the maximum difference between the exact values of q and extrapolated ones, that is, the L^∞ norm of the error, is computed within a band of thickness $2\sqrt{\Delta x^2 + \Delta y^2}$ in the outside regions. Figures 3.3 and 3.4 summarize the convergence behavior of the ND-PDE approach of [6] and the proposed WCD-PDE approach. Figure 3.5 demonstrates the error distribution for both methods in the case of the quadratic extrapolation on a 128^2 grid.

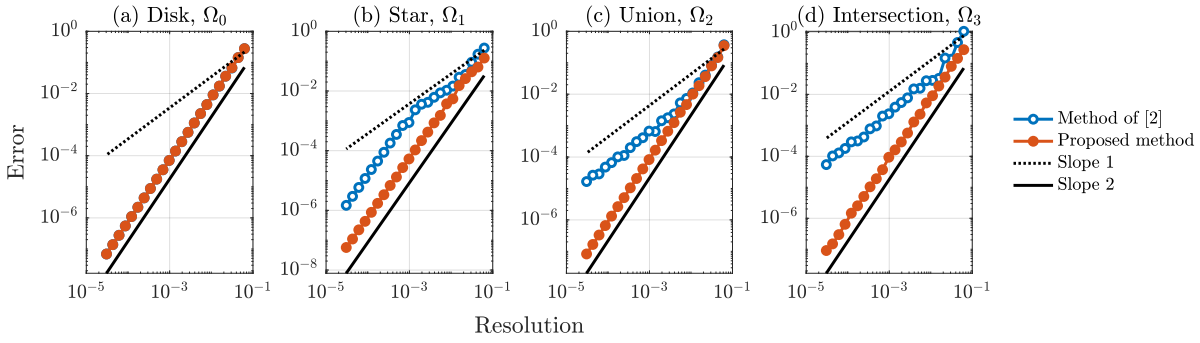


Figure 3.3: *Accuracy of the linear extrapolation (in the L^∞ norm) in two spatial dimensions measured in a narrow band of thickness $2\sqrt{\Delta x^2 + \Delta y^2}$ around an interface using the approach of [6] and the proposed approach.*

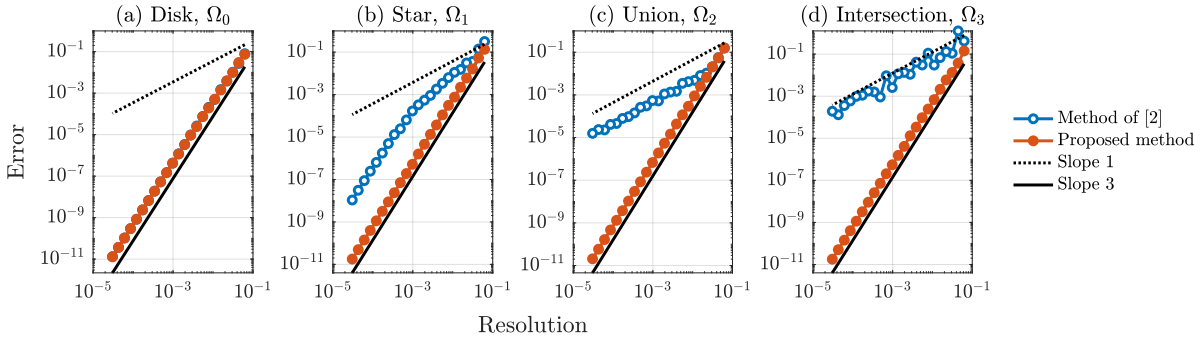
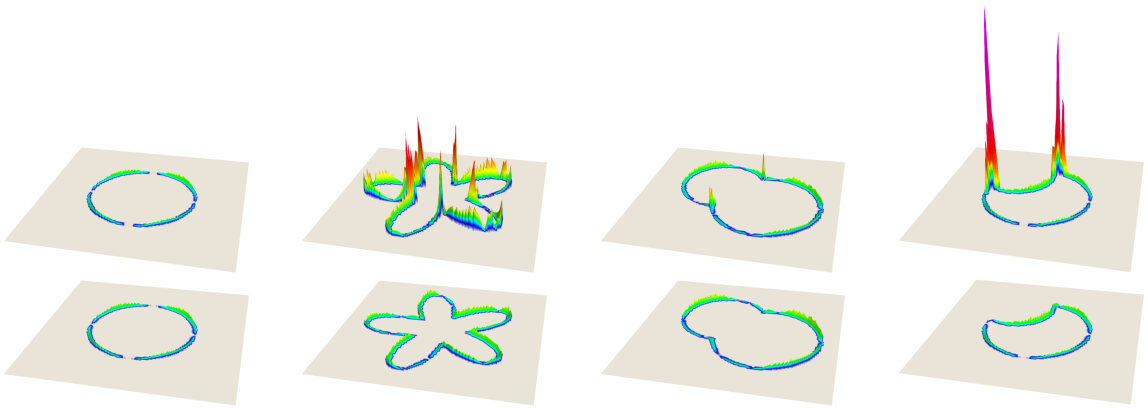


Figure 3.4: *Accuracy of the quadratic extrapolation (in the L^∞ norm) in two spatial dimensions measured in a narrow band of thickness $2\sqrt{\Delta x^2 + \Delta y^2}$ around an interface using the approach of [6] and the proposed approach.*

In case of the smooth domain Ω_0 (disk), both approaches produce almost indistin-



(a) Disk, Ω_0 (b) Star, Ω_1 (c) Union, Ω_2 (d) Intersection, Ω_3

Figure 3.5: *Comparison of error distributions in the case of the quadratic extrapolation on a* 128^2 *grid. Top row: the approach of [6]. Bottom row: the present approach. In each case the error is multiplied by a factor of 30 for visualization purpose.*

guishable results attaining second- and third-order rates of convergence for the linear and quadratic extrapolation, respectively.

For the high-curvature domain Ω_1 (star), both methods still reach optimal orders of convergence; however the ND-PDE approach demonstrates the optimal order of convergence only at relatively high grid resolutions when all geometric features are well-resolved. Moreover, for a given grid resolution the WCD-PDE approach produces results that are more than one order of magnitude more accurate in the case of the linear extrapolation and almost three orders of magnitude more accurate in the case of the quadratic extrapolation compared to the ND-PDE approach. Figure 3.5b shows that the ND-PDE approach produces very large errors near regions with the highest curvature, while the error in the case of the WCD-PDE approach is much smaller and exhibits very little variation throughout all regions around the interface.

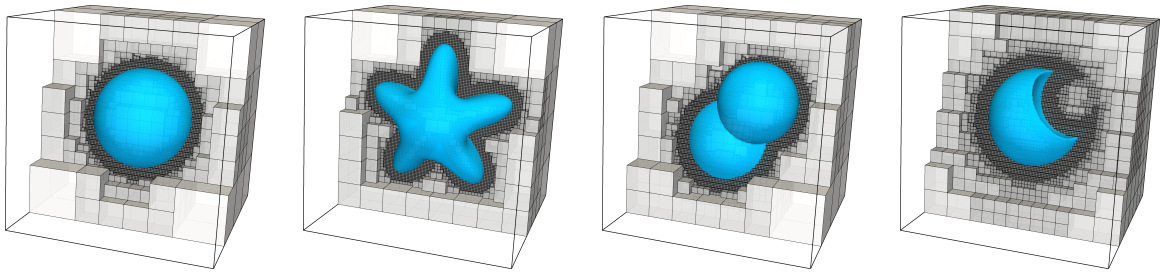
The results are even more significantly improved with the proposed approach in the case of interfaces with kinks Ω_2 (union) and Ω_3 (intersection). Figures 3.5c and 3.5d show that the ND-PDE method of [6] produces large errors near kinks; those errors are significantly reduced with the WCD-PDE approach. In particular, Figures 3.3c-d and 3.4c-d demonstrate that the second-order (third-order) accuracy of the linear (quadratic) extrapolations are recovered with the proposed approach; the rates of convergence for the approach of [6] are close to first order, which corresponds to the constant extrapolation, due to the fact that errors near kinks do not decrease despite grid refinement and the apparent first order of convergence is only because the neighborhood in which errors are computed is shrinking closer to the domain.

3.4 Numerical Results in Three Spatial Dimensions

We consider three different domains, $\tilde{\Omega}_1$, $\tilde{\Omega}_2$ and $\tilde{\Omega}_3$, that present high-curvature features or kinks in three spatial dimensions. In addition, we consider a smooth spherical domain $\tilde{\Omega}_0$ with center $(0, 0, 0)$ and radius 0.501. The definition of those domains are given by the level-set functions:

$$\tilde{\phi}_0(x,y,z) = \sqrt{x^2+y^2+z^2} - 0.501,$$
$$\tilde{\phi}_1(x,y,z) = \sqrt{x^2+y^2+z^2} - 0.501 - 0.15\frac{y^5+5x^4y-10x^2y^3}{(x^2+y^2+z^2)^{\frac{5}{2}}}\cos\left(\frac{\pi}{2}\frac{z}{0.501}\right),$$
$$\tilde{\phi}_2(x,y,z) = \min\Big(\sqrt{(x+.1)^2+(y+.3)^2+(z+.2)^2} - 0.501,$$
$$\sqrt{(x-.2)^2+(y-.2)^2+(z-.1)^2} - 0.401\Big),$$
$$\tilde{\phi}_3(x,y,z) = \max\Big(\sqrt{x^2+y^2+z^2} - 0.501,$$
$$\sqrt{(x-.4)^2+(y-.3)^2+(z-.2)^2} - 0.401\Big),$$

Figure 3.6 depicts those domains along with the octree grid refined near their boundaries.



(a) Sphere, $\tilde{\Omega}_0$ (b) Star, $\tilde{\Omega}_1$ (c) Union, $\tilde{\Omega}_2$ (d) Intersection, $\tilde{\Omega}_3$

Figure 3.6: *Irregular domains considered in section 3.4 along with the octree grids refined near their boundaries.*

Similar to the two-dimensional examples, we consider a computational domain $\Omega = (-1,1)^3$. We extrapolate the function $q = \sin(\pi x)\cos(\pi y)\exp(z)$ from the inside to the outside for every domain and compute the difference between the exact values of q and the extrapolated ones, that is, the L^∞ norm of the error, within a band of thickness $2\sqrt{\Delta x^2+\Delta y^2+\Delta z^2}$ in the outside region.

Conclusions similar to the two dimensional case can be drawn from the results in figures 3.7 and 3.8. Specifically, for a smooth and well-resolved domain $\tilde{\Omega}_0$ (sphere) both

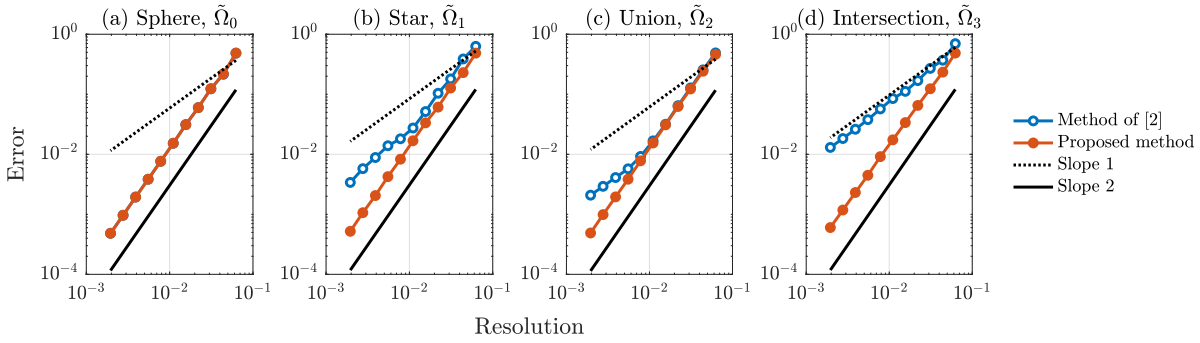


Figure 3.7: *Accuracy of the linear extrapolation (in the L^∞ norm) in three spatial dimensions measured in a narrow band of thickness $2\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ around an interface using the approach of [6] and the proposed approach.*

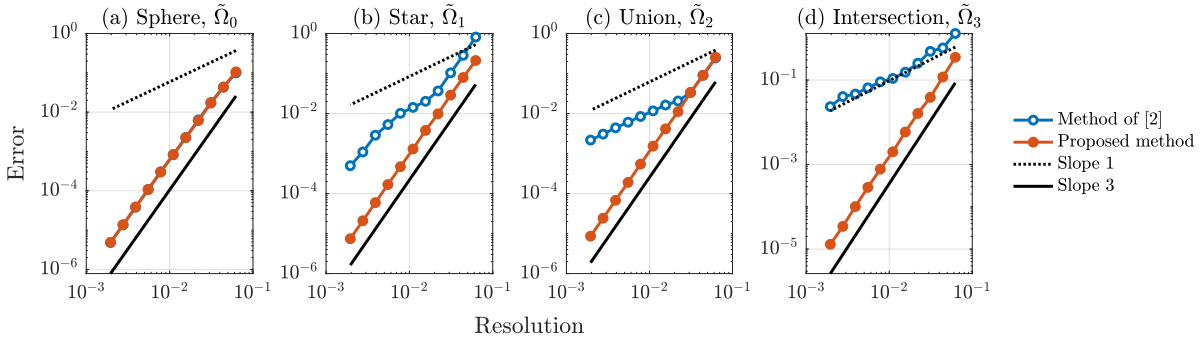


Figure 3.8: *Accuracy of the quadratic extrapolation (in the L^∞ norm) in three spatial dimensions measured in a narrow band of thickness $2\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ around an interface using the approach of [6] and the proposed approach.*

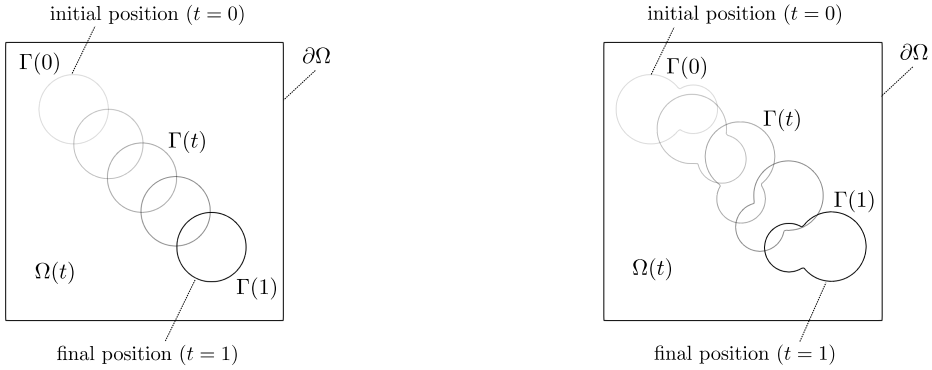
approaches produce almost indistinguishable results with optimal order of convergence (second and third for the linear and quadratic extrapolations, respectively). When the interface curvature is high ($\tilde{\Omega}_1$, star) the WCD-PDE approach produce extrapolated fields that are several orders of magnitude more accurate than for the ND-PDE approach. For geometries with sharp features $\tilde{\Omega}_2$ (union) and $\tilde{\Omega}_3$ (intersection) only the WCD-PDE approach demonstrates optimal orders of convergence, while for the ND-PDE approach the rate of convergence is stuck to 1.

3.5 Application to Solving the Diffusion Equation in Time-Dependent Domains

In order to illustrate the importance of accurate extrapolation near sharp corners in moving interface problems, we present a simple example of solving diffusion equation around a moving object that may have a non-smooth boundary. Specifically, we consider a two-dimensional rectangular region $[-1;1]\times[-1;1]$ and an object that moves diagonally from its starting position at $(x_s, y_s) = (-0.51, 0.52)$ at time $t = 0$ to the final position $(x_f, y_f) = (0.49, -0.48)$ at time $t = 1$ while making half a turn around its center as demonstrated in Figure 3.9. A diffusion equation subject to Neumann boundary conditions is solved in the rectangular box excluding the region occupied by the moving object. While the problem at hand does not correspond to any specific practical application, it represents a prototypical situation arising in simulation of more relevant (and more complex) processes and at the same time allows a precise analysis of numerical errors. Note that a non-deformable shape is considered for the sake of simplicity, we expect the main conclusions to hold true in more general cases, e.g. multiphase flow with triple junction points.

The Eulerian framework is employed, more precisely, the region $[-1;1] \times [-1;1]$ is discretized into a static uniform rectangular grid with N nodes in each Cartesian direction

while the object is implicitly described by a time-dependent level-set function.



(a) Smooth moving object (b) Moving object with corners

Figure 3.9: *Problem geometry in diffusion equation example from Section 3.5.*

Suppose the shape of the moving object in its local system of coordinate $\boldsymbol{\xi}$ is described by a level-set function $\phi_0(\boldsymbol{\xi})$ (such that $\phi_0(\boldsymbol{\xi}) > 0$ inside the object). Then the object's motion in the global system of coordinates $\boldsymbol{r}$ can be expressed by a time-dependent level-set function $\phi(t, \boldsymbol{r}) = \phi_0(\boldsymbol{\xi}(t, \boldsymbol{r}))$, where global-to-local coordinate transformation $\boldsymbol{\xi}(t, \boldsymbol{r})$ is given by:

$$\boldsymbol{\xi}(t, \boldsymbol{r}) = \begin{pmatrix} \xi(t, \boldsymbol{r}) \\ \eta(t, \boldsymbol{r}) \end{pmatrix} = \begin{pmatrix} x - (x_s + t(x_f - x_s)) \\ y - (y_s + t(y_f - y_s)) \end{pmatrix} \begin{pmatrix} \cos(\pi t) & \sin(\pi t) \\ -\sin(\pi t) & \cos(\pi t) \end{pmatrix}.$$

The solution domain $\Omega(t)$ can then be defined as:

$$\Omega(t) = \{\boldsymbol{r} \in [-1; 1] \times [-1; 1] \,:\, \phi(t, \boldsymbol{r}) < 0\}.$$

The boundary of the computational box is denoted as $\partial\Omega$ and the boundary of the moving object is denoted as $\Gamma(t)$.

In order to investigate the influence of non-smooth interface, we consider two choices

of moving object, one having a smooth boundary, a disk of radius 0.25, and another one having a non-smooth boundary, a union of two disk with radii 0.25 and 0.175, motivated by multimaterial compound bubbles. In the first case the level-set function of the object is given by (in the local system of coordinates):

$$\phi_0^{\text{smooth}}(\boldsymbol{\xi}) = (\boldsymbol{\xi}) = r_0 - \sqrt{\xi^2 + \eta^2},$$

while in the latter case:

$$\phi_0^{\text{non-smooth}}(\boldsymbol{\xi}) = \max\left(r_0 - \sqrt{(\xi + \xi_0)^2 + \eta^2}, r_0 q - \sqrt{(\xi - \xi_0)^2 + \eta^2}\right),$$

where $r_0 = 0.25$, $\xi_0 = \frac{1}{2} r_0 \sqrt{1 + q^2}$ and $q = 0.7$.

We choose the following test solution:

$$u(t, \boldsymbol{r}) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{i,j} \cos\left(\frac{i\pi}{2}(x+1)\right) \cos\left(\frac{j\pi}{2}(y+1)\right) \exp\left(-D\frac{\pi^2}{4}(i^2 + j^2)t\right),$$

with

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} = \begin{pmatrix} -0.5 & -0.1 & -0.5 & 0.6 \\ -0.6 & -0.5 & -0.1 & -0.1 \\ 0.2 & -0.2 & -0.2 & 0.4 \\ 0.1 & -0.9 & 0.8 & 0.4 \end{pmatrix},$$

which satisfies a homogeneous diffusion equation:

$$\partial_t u = D\nabla^2 u, \quad \text{for } t \in [0, 1],\ \boldsymbol{r} \in \Omega(t), \tag{3.12}$$

with initial conditions:

$$u(0,\boldsymbol{r}) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{i,j}\cos\left(\frac{i\pi}{2}(x+1)\right)\cos\left(\frac{j\pi}{2}(y+1)\right), \quad \text{for } \boldsymbol{r}\in\partial\Omega(0)$$

and boundary conditions:

$$\begin{aligned} &D\partial_{\boldsymbol{n}}u = 0, && \text{for } t\in[0,1],\ \boldsymbol{r}\in\partial\Omega,\\ &D\partial_{\boldsymbol{n}}u = g(t,\boldsymbol{r}), && \text{for } t\in[0,1],\ \boldsymbol{r}\in\Gamma(t), \end{aligned}$$

where function $g(t,\boldsymbol{r})$ is given by:

$$g(t,\boldsymbol{r}) = D\frac{\nabla\phi(t,\boldsymbol{r})\cdot\nabla u(t,\boldsymbol{r})}{|\nabla\phi(t,\boldsymbol{r})|}.$$

The time range $[0;1]$ is discretized into time layers t_n, $n=0,1,2\ldots$, where the time step between adjacent time layers $\Delta t_{n+1} = t_{n+1} - t_n$ is determined such that the maximum displacement of the moving object boundary during the given time step is expected not to exceed a user-defined fraction f of the grid cell diagonal $\sqrt{\Delta x^2+\Delta y^2}$, that is:

$$\Delta t_{n+1} = \frac{f\sqrt{\Delta x^2+\Delta y^2}}{\max\limits_{\Gamma(t_n)} v_{\boldsymbol{n}}(t_n,\boldsymbol{r})},$$

where $v_{\boldsymbol{n}}$ denotes the normal velocity of the object's boundary. Specifically, in this example $f=0.8$ is taken.

We use the second-order variable-step backward differentiation formula (BDF2) for discretizing the diffusion equation (3.12) in time, that is:

$$\left(\alpha_0 - \Delta t_n\nabla^2\right)u^n = -\alpha_1 u^{n-1} - \alpha_2 u^{n-2}, \tag{3.13}$$

where

$$\alpha_0 = \frac{1+2\rho}{1+\rho}, \quad \alpha_1 = -(1+\rho), \quad \alpha_2 = \frac{\rho^2}{1+\rho} \quad \text{and} \quad \rho = \frac{\Delta t_n}{\Delta t_{n-1}},$$

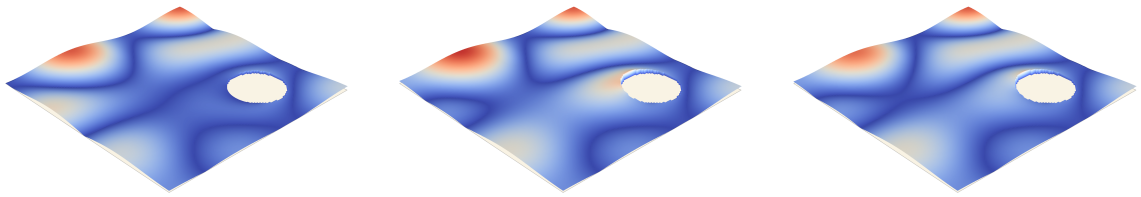
and use the superconvergent second-order accurate method of [18] (which is specifically designed to handle irregular domains with non-smooth boundaries) for solving the resulting Poisson-type equation (3.13).

Solution of equation (3.13) produces values of u^n at all grid nodes that belong to the current solution domain $\Omega(t_n)$. However, as the object moves in time some of grid nodes outside of $\Omega(t_n)$ may become part of $\Omega(t_{n+1})$ or $\Omega(t_{n+2})$ and solving at subsequent time layers t_{n+1} and t_{n+2} would require valid values of u^n at such grid nodes. This is typically addressed in free boundary value problems by smoothly extrapolating u^n into some neighborhood of $\Omega(t_n)$. In this example we quadratically extrapolate solutions using both the proposed in this work WCD-PDE approach and the ND-PDE approach of [6]. Also, since solving Poisson-type equations on irregular domains produces additional errors, we generate a reference solution where instead of performing extrapolation of numerical values we fill the grid nodes outside of the current solution domain $\Omega(t_n)$ with exact values given by the analytical solution.

We investigate the influence of extrapolation procedures on the accuracy of the numerical solution and its gradient. In order not to measure the error of extrapolation procedure itself but rather only its influence on solving the diffusion equation, the solution error at time layer t_n is calculated only for grid nodes in $\Omega(t_n)$ and the gradient is calculated only using values from $\Omega(t_n)$ as well.

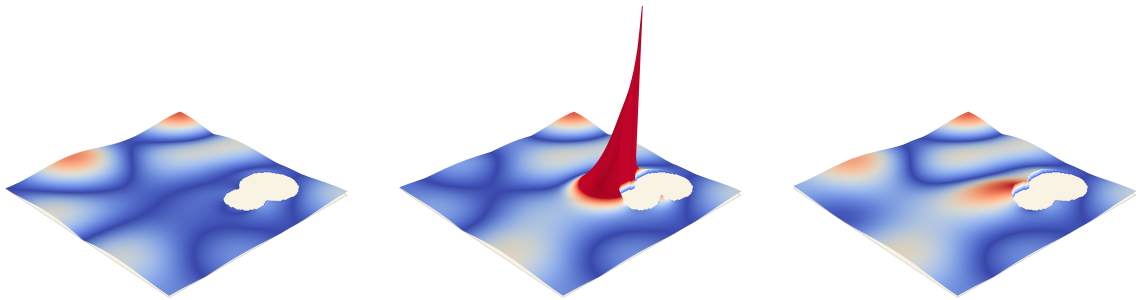
Obtained results are summarized in Figures 3.10, 3.11, 3.12 and 3.13. Figures 3.10 and 3.11 show error distributions at the final time $t = 1$ on a 128^2 grid. As one can see, for a smooth moving object (Fig. 3.10) using either the ND-PDE extension or the WCD-PDE one, results in errors that are very close to ones of the reference solution, while for a non-smooth moving object (Fig. 3.11) the WCD-PDE extension produces a much more accurate solution that is also very close to the reference one, whereas using the ND-PDE extension results in a significant accumulation of errors in the wake of the moving object. More quantitative conclusions can be drawn from the convergence studies shown in Figures 3.12 and 3.13, in which the dependence of the error in the L^∞ norm for both numerical solution and its gradient on the grid resolution is presented. In case of a smooth moving object (Fig. 3.12) both extension methods lead to second-order convergence both in the numerical solutions and its gradient (as what is expected from the superconvergent method of [18]) with magnitude of errors being very close to the ones of the reference solution. In case of a non-smooth moving object (Fig. 3.13) the accuracy of numerical solutions obtained using the ND-PDE extension degrades severely showing only first-order convergence in the solution itself and non-convergence in the gradient. At the same time the accuracy of computations based on the proposed WCD-PDE extension seems affected very slightly by the presence of sharp corners, retaining the second-order convergence in the numerical solution and its gradient with errors very close to the reference solution.

Remark: Note that the presented in this work extrapolation approach is designed



(a) Using exact values (b) Using method of [6] (c) Using proposed method

Figure 3.10: *Error distribution at the final time moment ($t = 1$) in case of a smooth moving object using different extrapolation approaches.*



(a) Using exact values (b) Using method of [6] (c) Using proposed method

Figure 3.11: *Error distribution at the final time moment ($t = 1$) in case of a non-smooth moving object using different extrapolation approaches.*

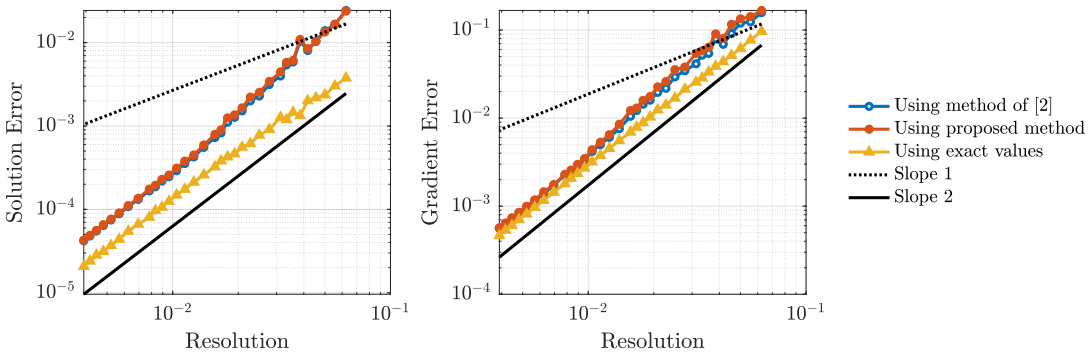


Figure 3.12: *Accuracy of solving diffusion equation (in the L^∞ norm) in case of a smooth moving object using different extrapolation approaches.*

for extending smooth scalar fields (as in the present example). However, in general, solutions to partial differential equations in domains with sharp features may contain

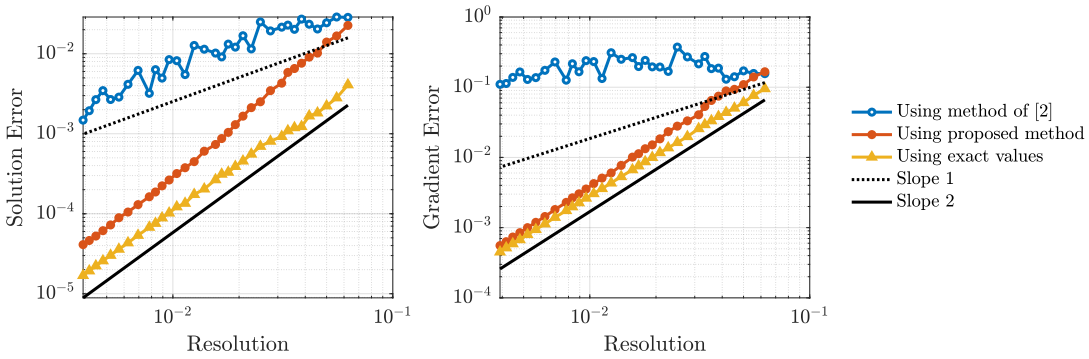


Figure 3.13: *Accuracy of solving diffusion equation (in the L^∞ norm) in case of a non-smooth moving object using different extrapolation approaches.*

singularities. In such cases for best results the proposed extrapolation procedure should only be applied to the regular part of the solution, the singular part must be dealt with separately using special methods, for example, as in [175].

3.6 Conclusion

We have presented a numerical method for extrapolating scalar quantities across the boundaries of irregular domains that may present high-curvature features or kinks. Linear and quadratic extrapolations procedures produce second- and third-order accurate results in the L^∞ norm, respectively and do so regardless of the irregularity of the boundaries, i.e. boundaries with kinks can readily be considered. These procedures are effective in both two and three spatial dimensions and can be implemented on quadtree and octree Cartesian grids. We have shown through numerical examples that errors associated with extrapolations can be reduced by several orders of magnitude in some cases, compared with the approach of [6] commonly used in level-set methods. We have also presented

an example of solving a diffusion equation on evolving domains in order to highlight the importance of accurate extrapolation near sharp geometric features for practical applications. The numerical method we introduced is based on solving PDEs in pseudo-time, but we note that static solutions based on an implicit approach like Fast Marching or Fast Sweeping could be obtained and we expect the results to follow the same general behavior as that presented in the current manuscript.

Chapter 4

Sharp-Interface Simulations of Multialloy Solidification

4.1 Introduction

Multicomponent alloys play a significant role in several areas of engineering such as energy, transportation, and propulsion due to their superior thermomechanical properties compared to pure materials. The exploration of novel multicomponent alloys with desired properties is hindered by the high dimensional nature of the parameter search space formed by all possible combinations of alloy components. As a consequence, accurate and reliable computational tools have a potential to significantly accelerate the discovery in this area. While a range of problems at multiple scales from atomistic to macroscopic require the numerical investigation, in this work we focus on the simulation of solidification

processes. The importance of this problem comes from the fact that the microstructure evolution and solutal segregation occurring during manufacturing processes have a strong influence on the resulting thermomechanical properties. There are several difficulties associated with the numerical simulation of such phenomena: the multiscale nature of the process, the highly irregular shape of the moving solidification front, the necessity of solving the system of non-linearly coupled PDEs.

Given the importance of the predictive modeling of solidification phenomena a great number of numerical approaches have been reported in the literature. These computational methods can be categorized in three categories: cellular automata methods [131], phase-field methods [84, 86, 156], and sharp-interface methods [179, 159]. Each of these frameworks has its own advantages and disadvantages. Cellular automata methods are computationally efficient however they are not based on physical equations of solidification but rather on special rules for interactions between automata. In phase field models the solid-liquid interface is described as a smooth transition of a "phase-field" variable, which allows efficient numerical implementations that do not require any specialized methods for dealing with moving interfaces. The phase field theory of solidification is mathematically well-justified and guarantees convergence to the sharp-interface equations as the transition width of the solid-liquid interface approaches zero, however, which is difficult to achieve in practice. The sharp-interface methods are expected to be the most accurate mathematically, however, are harder to develop. They require numerical capabilities for explicit handling of evolving interfaces and solving nonlinear systems of

PDEs in irregular domains. So far only cases of binary alloys have been successfully modeled [159, 179].

In this chapter we present a computational method for the simulation of the solidification of multicomponent alloys based on the sharp-interface formulation of the governing equations. Contrary to the case of binary alloys where a fixed point iteration is adequate [159], we propose a Newton-type approach to solve the nonlinear system of coupled PDEs arising from the time discretization of the governing equations. A combination of spatially adaptive quadtree grids, Level-Set Method, and sharp-interface numerical methods for imposing boundary conditions is used to accurately and efficiently resolve the complex behavior of the solidification front. We validate the proposed computational method on the case of axisymmetric radial solidification admitting an analytical solution and show that the overall method's accuracy is close to second order. Finally, we perform numerical experiments for the directional solidification of a Co-Al-W ternary alloy with a phase diagram obtained from the PANDAT$^{\text{TM}}$thermodynamic database and analyze the solutal segregation dependence on the processing conditions and alloy properties.

The rest of this chapter is organized in the following way. In section 4.2 we briefly introduce the basic mathematical model of the multialloy solidification. Section 4.3 describes the overall computational approach including the proposed Newton-type method for solving the nonlinearly coupled system of PDEs as well as numerical implementation aspects. Section 4.4 presents validation tests and application of the proposed method for investigation of solutal segregation of a Co-Al-W alloy. Finally, in 4.5 we draw conclusions

and discuss future work.

4.2 Physical Model

In this section we briefly present a mathematical model of the alloy solidification used in this work. For a detailed discussion on the theory of crystallization processes, we refer the interested reader to monograph [37].

Consider the solidification of an alloy containing $N+1$ different elements: a solvent that constitutes majority of the alloy and N solutes. Specifically, we assume that the process occurs in a rectangular domain Ω (possibly periodic in some directions). In this work we consider a mathematical model describing crystallization processes at the macroscopic level without resolving atomistic details. Thus, the transition between solid and liquid phases is assumed to be sharp. We denote this interphase boundary as Γ and regions of Ω occupied by solid and liquid phases as Ω_s and Ω_l, correspondingly, as demonstrated in Figure 4.1. The outward normal vectors to the boundaries of Ω_l and Ω_s are denoted as $\boldsymbol{n}_l$ and $\boldsymbol{n}_s$, respectively. The normal vector to interface Γ directed from the solid to the liquid is denoted as $\boldsymbol{n}$. Note that $\boldsymbol{n} = \boldsymbol{n}_s = -\boldsymbol{n}_l$ on Γ.

As time t proceeds, the solidification front, $\Gamma = \Gamma(t)$, evolves with a normal velocity $v_{\boldsymbol{n}} = v_{\boldsymbol{n}}(t, \boldsymbol{r})$, $\boldsymbol{r} \in \Gamma$, according to the crystallization kinetics. In the case of pure substances the process is mainly governed by the thermal transport: the phase transition occurs at the freezing temperature (that may depend on the curvature and normal velocity of the solid-liquid interface) and releases the latent heat which is transported

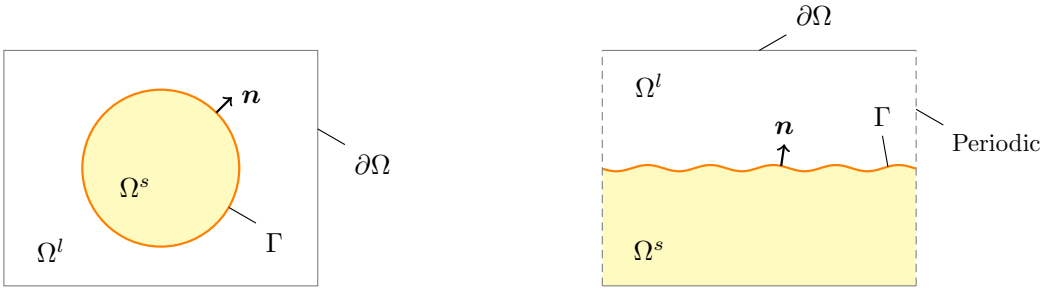


Figure 4.1: Notation used in this work demonstrated on examples of crystal growth from a seed (left) and directional solidification (right).

away by the thermal diffusion and, possibly, advection. The case of multicomponent substances, like metal alloys, is complicated by the transport of species in a two-way coupling: on the one hand, freezing temperatures depend on the local alloy composition, on the other hand, the advancing crystallization front affects concentration fields due to solute-rejection phenomena.

Thus, at any given moment of time t at every point $\boldsymbol{r} \in \Omega$ the alloy is characterized by the local temperature $T = T(t, \boldsymbol{r})$ and the composition $C_J = C_J(t, \boldsymbol{r})$, $J \in [1, N]$, where C_J denotes the Jth solute's concentration. For convenience, since temperature and concentration fields are not generally smooth and/or continuous across interphase boundaries, we use a separate set of fields for each of the two phases, that is:

$$T(t, \boldsymbol{r}) = \begin{cases} T_l(t, \boldsymbol{r}), & \boldsymbol{r} \in \Omega_l(t) \\ T_s(t, \boldsymbol{r}), & \boldsymbol{r} \in \Omega_s(t) \end{cases},$$
$$C_J(t, \boldsymbol{r}) = \begin{cases} C_{lJ}(t, \boldsymbol{r}), & \boldsymbol{r} \in \Omega_l(t) \\ C_{sJ}(t, \boldsymbol{r}), & \boldsymbol{r} \in \Omega_s(t) \end{cases}, \quad J \in [1, N],$$

where subscripts $_s$ and $_l$ denote quantities in solid and liquid phases, respectively.

Suppose, at some initial time $t = t_0$ the state of the system is described by the

following initial conditions:

$$\begin{aligned} \Gamma(t_0) &= \Gamma_0, \\ T_\nu(t_0, \boldsymbol{r}) &= T_{0\nu}(\boldsymbol{r}), \quad \boldsymbol{r} \in \Omega_\nu(t_0), \quad \nu = s, l, \\ C_{\nu J}(t_0, \boldsymbol{r}) &= C_{0\nu J}(\boldsymbol{r}), \quad \boldsymbol{r} \in \Omega_\nu(t_0), \quad \nu = s, l, \quad J \in [1, N], \end{aligned} \tag{4.1}$$

where Γ_0, $T_{0\nu}(\boldsymbol{r})$, and $C_{0\nu J}(\boldsymbol{r})$ describe the initial solid-liquid interface, temperature field and concentration fields. In the absence of convective effects the transport of heat and species is described by diffusion equations:

$$\rho_\nu c_{p_\nu} \partial_t T_\nu - \lambda_\nu \nabla^2 T_\nu = 0, \quad \boldsymbol{r} \in \Omega_\nu(t), \quad \nu = s, l, \tag{4.2}$$

$$\partial_t C_{\nu J} - D_{\nu J} \nabla^2 C_{\nu J} = 0, \quad \boldsymbol{r} \in \Omega_\nu(t), \quad \nu = s, l, \quad J \in [1, N], \tag{4.3}$$

where ρ_ν, c_{p_ν}, and λ_ν, $\nu = s, l$, are the density, the specific heat, and the heat conductivity of liquid and crystallized alloys; $D_{\nu J}$, $\nu = s, l$, are the Jth solute's diffusivity in liquid and solid phases, respectively. We assume alloy parameters ρ_ν, c_{p_ν}, λ_ν, $\{D_{\nu J}\}_{J=1}^N$, $\nu = s, l$, to be constant.

Since for typical metal alloys the diffusion of solutes in the solid phase is several orders of magnitude slower than in the liquid phase we neglect the species transport in the solid, that is, $D_{sJ} = 0$, $J \in [1, N]$. As a result, diffusion equations for the concentration fields (4.3) only need to be solved in the liquid.

Temperature and concentration fields must satisfy several conditions on the solidification front Γ. We assume that the phase transition occurs at the thermodynamic equilibrium, that is, the temperature is continuous across the solidification front:[1]

[1]Square brackets denote the jump in the value of a quantity across the solidification front, i.e. $[T] = T_s - T_l$.

$$[T] = 0, \quad \boldsymbol{r} \in \Gamma(t), \tag{4.4}$$

and satisfies the Gibbs-Thomson relation:

$$T_l = T_{liq}(C_{l1}, \ldots, C_{lN}) + \epsilon_v(\boldsymbol{n}) v_{\boldsymbol{n}} + \epsilon_c(\boldsymbol{n})\kappa, \quad \boldsymbol{r} \in \Gamma(t), \tag{4.5}$$

where $T_{liq} = T_{liq}(C_{l1}, \ldots, C_{lN})$ describes the liquidus surface of the alloy (i.e., melting temperature at a given composition), terms $\epsilon_c(\boldsymbol{n})$ and $\epsilon_v(\boldsymbol{n})$ account for the curvature and kinetic undercoolings, and κ is the front's mean curvature. Usually T_{liq} is assumed to be a linear function of solutal concentrations:

$$T_{liq}(C_{l1}, \ldots, C_{lN}) = T_m + m_{l1}C_{l1} + \ldots + m_{lN}C_{lN},$$

where T_m is the melting temperature of the pure solvent and m_{l1}, ..., m_{lN} are constants called the *liquidus slopes* corresponding to each of the solutes. However, this work is not restricted to such a case and considers $T_{liq}(C_{l1}, \ldots, C_{lN})$ an arbitrary function (hence, the liquidus slopes $m_{lJ} = \frac{\partial T_{liq}}{\partial C_{lJ}}$, $J \in [1, N]$, are no longer constants but functions of the local composition as well). Specifically, for the simulation results presented later in this work the data from the PANDAT$^{\text{TM}}$thermodynamic database was used.

Note that undercooling coefficients $\epsilon_c(\boldsymbol{n})$ and $\epsilon_v(\boldsymbol{n})$ may depend on the normal vector $\boldsymbol{n}$ to the solidification front accounting in such a way for specific crystalline structures of alloys. For example, a two-dimensional four-fold crystalline structure is commonly described as:

$$\epsilon_c(\boldsymbol{n}) = \varepsilon_c(1 - 15\varepsilon\cos(4\cos^{-1}(\boldsymbol{n}\cdot\boldsymbol{n}_0))),$$
$$\epsilon_v(\boldsymbol{n}) = \varepsilon_v(1 - 15\varepsilon\cos(4\cos^{-1}(\boldsymbol{n}\cdot\boldsymbol{n}_0))),$$

where ε_c and ε_v are curvature and kinetic undercooling magnitudes, ε is the degree of anisotropy and $\boldsymbol{n}_0$ is the preferred crystal growth direction.

The thermal balance at the interface leads to the following (Stefan) condition:

$$[\lambda\partial_{\boldsymbol{n}}T] = v_{\boldsymbol{n}}L_f, \quad \boldsymbol{r}\in\Gamma(t), \tag{4.6}$$

where L_f is the latent heat of fusion of the alloy. Note that, as commonly done, the change in surface energy due to stretching/contraction of the curved front's surface in the velocity field $v_{\boldsymbol{n}}$ is neglected in the above expression.

At the solidification front the compositions of liquid and solid phases are related to each other through chemical equilibrium. Typically such a relation is described by parameters called *partition coefficients* k_J, $J\in[1,N]$, which represent the ratios of component concentrations in solid and liquid phases, that is:

$$C_{sJ} = k_J C_{lJ}, \quad \boldsymbol{r}\in\Gamma(t), \quad J\in[1,N]\,.$$

Since in this work we do not restrict ourselves to linearized liquidus and solidus surfaces the partition coefficients are also assumed to depend on the local composition of the solidifying material, that is:

$$k_J = k_J(C_{l1},\ldots,C_{lN}), \quad J\in[1,N]\,.$$

The conservation of species at the solidification front lead to the following so-called *solute-rejection equations*

$$D_{lJ}\partial_{\boldsymbol{n}_l} C_{lJ} - (1-k_J)v_{\boldsymbol{n}} C_{lJ} = 0, \quad \boldsymbol{r} \in \Gamma(t), \quad J \in [1,N]\,. \tag{4.7}$$

The type of boundary conditions (Dirichlet, Neumann or Robin) on the boundary of the solidification region Ω, denoted as $\partial\Omega$, depends on particular physical circumstances. We assume that the total heat flux is specified and the boundary is impermeable to solutes:

$$\begin{aligned} \lambda_\nu \partial_{\boldsymbol{n}_\nu} T_\nu &= g_{T_\nu}, \quad && \boldsymbol{r} \in \Omega_\nu \cap \partial\Omega, \quad \nu = s,l, \\ D_{lJ}\partial_{\boldsymbol{n}_l} C_{lJ} &= 0, && \boldsymbol{r} \in \Omega_l \cap \partial\Omega, \quad J \in [1,N]\,, \end{aligned} \tag{4.8}$$

where $g_{T_\nu} = g_{T_\nu}(t,\boldsymbol{r})$, $\nu = s,l$, are prescribed heat fluxes for liquid and solid phases. We note, however, that switching to boundary conditions of another type (Dirichlet or Robin) has minimal consequences on the computational method presented in this work.

To summarize, in this work we present a computational method for solving a multialloy solidification model in which the crystallization process is described by the temporal evolution of temperature fields $T_\nu = T_\nu(t,\boldsymbol{r})$, $\nu = s,l$, solutes' concentration fields $C_{lJ} = C_{lJ}(t,\boldsymbol{r})$, $J \in [1,N]$, and a solidification front $\Gamma = \Gamma(t)$ according to partial differential equations (4.2)-(4.3) with interface conditions (4.4)-(4.7) on Γ and boundary conditions (4.8) on $\partial\Omega$.

4.3 Numerical Approach

In this section we present the overall numerical approach in detail, specifically:

1. We begin with discussing the temporal discretization of the system of governing equations and identify specific tasks needed to be performed during each time step (section 4.3.1).

2. Second, in section 4.3.2, we present a numerical method for solving the nonlinear system of elliptic PDEs resulted from the temporal discretization. The method is based on breaking down the system of nonlinearly coupled equations into a set of separate boundary value problems subject to classical boundary (Dirichlet, Neumann or Robin) and interface conditions.

3. Finally, section 4.3.3 provides a detailed description of specific methods we use for spatial discretization of the computational domain, evolving the solidification front in time, solving elliptic partial differential equations with different boundary and interface conditions on irregular interfaces.

4.3.1 Discretization in time

Consider a non-uniform discretization of time $\{t_j\}_{j\geq 0}$ with time steps $\{\Delta t_j = t_j - t_{j-1}\}_{i\geq 1}$ and denote the state of the system (i.e. temperature field, concentration fields, and location of the solidification front) at a time moment t_j as T_ν^j, C_{lJ}^j and Γ^j, $j \geq 0$. Given states of the system for t_j, $j < n$, the numerical solution at time instant t_n is computed

in the following fashion.

First, the new front's location Γ^n is obtained from Γ^{n-1} in an *explicit* way based on values of the normal velocity at previous time moments $v_{\boldsymbol{n}}^j$, $j < n$, as discussed in section 4.3.3.

Secondly, equations (4.1)-(4.8) are solved *implicitly* for T_s^n, T_l^n, $\{C_{lJ}^n\}_{J=1}^N$ and $v_{\boldsymbol{n}}^n$ in geometry defined by Γ^n. To this end, we use a second-order accurate implicit (BDF2) discretization in time. Let us write the approximation of the temporal derivative of a quantity A at a time instant $t = t_n$ as:

$$\partial_t A^n = \frac{1}{\Delta t_n} \sum_{j \geq 0} a_j A^{n-j} + \mathcal{O}\left(\Delta t_{\max}^q\right) \tag{4.9}$$

where $\Delta t_{\max} = \max\limits_{j \geq 0} \left(\Delta t_{n-j}\right)$ and coefficients $\{a_j\}_{j \geq 0}$ are given by:

$$a_0 = \frac{1+2r}{1+r}, \quad a_1 = -(1+r), \quad a_2 = \frac{r^2}{1+r}, \quad a_j = 0,\ j \geq 3,$$

where $r = \dfrac{\Delta t_n}{\Delta t_{n-1}}$ and $q = 2$. Using approximation (4.9) in diffusion equations (4.2) and (4.3) we get:

$$\left(\rho_\nu c_{p_\nu} \frac{1}{\Delta t_n} a_0 - \lambda_\nu \nabla^2\right) T_\nu^n = -\rho_\nu c_{p_\nu} \frac{1}{\Delta t_n} \sum_{j \geq 1} a_J T_\nu^{n-j}, \quad \boldsymbol{r} \in \Omega_\nu^n, \quad \nu = s, l, \tag{4.10}$$

$$\left(\frac{1}{\Delta t_n} a_0 - D_{lJ} \nabla^2\right) C_{lJ}^n = -\frac{1}{\Delta t_n} \sum_{j \geq 1} a_J C_{lJ}^{n-j}, \qquad \boldsymbol{r} \in \Omega_l^n, \quad J \in [1, N], \tag{4.11}$$

where known quantities are collected in the right-hand side. The above two expressions are simple linear Poisson-type equations, however they must be solved subject to the non-linear interface and boundary conditions (4.4)-(4.7) on Γ^n. The source of non-linearity

is in Robin-type boundary conditions (4.7) that contains the product of two unknowns – concentration C_{lJ}^n and velocity $v_{\boldsymbol{n}}^n$.[2] Once a method for solving (4.10)-(4.11) subject to (4.4)-(4.7) is available, then it is relatively easy to construct a time-stepping procedure for solving the entire dynamic problem . Thus, the solution of (4.10)-(4.11) subject to (4.4)-(4.7) is the cornerstone problem in simulating multialloy solidification.

4.3.2 Solving the non-linearly coupled system of Poisson-type equations

For clarity of presentation we write the system of the coupled Poisson-type equations (4.10)-(4.11) subject to (4.4)-(4.8) in a generic fashion and with abuse of notation as:

$$\textit{Heat transport:} \quad \left(s_\nu - \lambda_\nu \nabla^2\right) T_\nu = f_{T_\nu},\ \boldsymbol{r} \in \Omega_\nu,\ \nu = s, l, \tag{4.12}$$

$$\textit{Species transport:} \quad \left(a - D_J \nabla^2\right) C_J = f_{C_J},\ \boldsymbol{r} \in \Omega_l,\ J \in [1, N]\,, \tag{4.13}$$

Conditions on Γ:

$$\textit{Temperature continuity:} \quad [T] = h_T, \tag{4.14}$$

$$\textit{Stefan condition:} \quad [\lambda \partial_{\boldsymbol{n}} T] = h_S + v_{\boldsymbol{n}} L_f, \tag{4.15}$$

$$\textit{Gibbs-Thompson:} \quad T_l = h_G + T_{liq}(C_1, \ldots, C_N) + \epsilon_v v_{\boldsymbol{n}}, \tag{4.16}$$

$$\textit{Solute-rejection:} \quad D_J \partial_{\boldsymbol{n}_l} C_J = (1 - k_J) v_{\boldsymbol{n}} C_J + h_{C_J},\ J \in [1, N]\,, \tag{4.17}$$

Conditions on $\partial\Omega$:

$$\textit{Heat supply/withdrawal:} \quad \lambda_\nu \partial_{\boldsymbol{n}_\nu} T_\nu = g_{T_\nu}, \quad \nu = s, l, \tag{4.18}$$

$$\textit{Impermeable boundaries:} \quad D_J \partial_{\boldsymbol{n}_l} C_J = g_{C_J}, \quad J \in [1, N]\,, \tag{4.19}$$

The original system of equations related to the solidification process is recovered by the following substitutions:

[2] Note that even when the so-called Frozen Temperature Approximation is applied (i.e., the temperature field is not solved for but prescribed by an analytical expression) the system of equations is still non-linearly coupled for $N > 1$.

$$
\begin{array}{llll}
s_\nu \to \dfrac{a_0 \rho_\nu c_{p_\nu}}{\Delta t_n}, & \lambda_\nu \to \lambda_\nu, & T_\nu \to T_\nu^n, & f_{T_\nu} \to \dfrac{-\rho_\nu c_{p_\nu}}{\Delta t_n} \displaystyle\sum_{j\geq 1} a_j T_\nu^{n-j}, \\
a \to \dfrac{a_0}{\Delta t_n}, & D_J \to D_{lJ}, & C_J \to C_{lJ}^n, & f_{C_J} \to -\dfrac{1}{\Delta t_n} \displaystyle\sum_{j\geq 1} a_j C_{lJ}^{n-j}, \\
h_T \to 0, & h_S \to 0, & h_{C_J} \to 0, & h_G \to \epsilon_c(\boldsymbol{n}), \\
(1-k_J) \to (1-k_J), & g_{C_J} \to g_{C_{lJ}}, & g_{T_\nu} \to g_{T_\nu}, & L_f \to L_f.
\end{array}
$$

Fixed-point iteration

In [159] a simple fixed-point iteration algorithm was used for simulating the solidification of bi-alloys, that is, in the case $N = 1$. It is based on breaking system (4.12)-(4.19) down into separate boundary value problems (BVPs) for temperature and concentrations fields with simple boundary (Dirichlet, Neumann or Robin) and interface conditions and iterating . A direct extension of this algorithm to the case of arbitrary N has the following form:

1. Let us denote the value of concentration C_1 at the solidification front during the qth iteration as $C_1^{*(q)}(\boldsymbol{r})$, $\boldsymbol{r} \in \Gamma$. Set $q = 0$ and some initial guess $C_1^{*(0)}$ (e.g., using the previous time step).

2. Solve for $C_1^{(q)}$ imposing Dirichlet BC on Γ:

$$
\begin{cases}
\left(a - D_1 \nabla^2\right) C_1^{(q)} = f_{C_1}, & \boldsymbol{r} \in \Omega_l, \\
C_1^{(q)} = C_1^{*(q)}, & \boldsymbol{r} \in \Gamma, \\
D_1 \partial_{\boldsymbol{n}_l} C_1^{(q)} = g_{C_1}, & \boldsymbol{r} \in \partial\Omega \cap \overline{\Omega_l}.
\end{cases} \tag{4.20}
$$

3. Compute the front's velocity $v_{\boldsymbol{n}}^{(q)}$ using the solute-rejection equation (4.17):

$$v_{\boldsymbol{n}}^{(q)} = \frac{1}{(1-k_1)C_1}\left(D_1\partial_{\boldsymbol{n}_l}C_1^{(q)} - h_{C_1}\right), \quad \boldsymbol{r}\in\Gamma. \tag{4.21}$$

4. Solve for $C_J^{(q)}$, $J\in[2,N]$, imposing Robin BC on Γ:

$$\begin{cases} \left(a - D_J\nabla^2\right)C_J^{(q)} = f_{C_J}, & \boldsymbol{r}\in\Omega_l, \\ D_J\partial_{\boldsymbol{n}_l}C_J^{(q)} - (1-k_J)v_{\boldsymbol{n}}^{(q)}C_J^{(q)} = h_{C_J}, & \boldsymbol{r}\in\Gamma, \\ D_J\partial_{\boldsymbol{n}_l}C_J^{(q)} = g_{C_J}, & \boldsymbol{r}\in\partial\Omega\cap\overline{\Omega_l}. \end{cases} \tag{4.22}$$

5. Solve for $T_\nu^{(q)}$, $\nu = s,l$, imposing jump conditions on Γ:

$$\begin{cases} \left(s_\nu - \lambda_\nu\nabla^2\right)T_\nu^{(q)} = f_T, & \boldsymbol{r}\in\Omega_\nu, \quad \nu = s,l, \\ \left[T^{(q)}\right] = h_T, & \boldsymbol{r}\in\Gamma, \\ \left[\lambda\partial_{\boldsymbol{n}}T^{(q)}\right] = h_S + v_{\boldsymbol{n}}^{(q)}L_f, & \boldsymbol{r}\in\Gamma, \\ \lambda_\nu\partial_{\boldsymbol{n}_\nu}T_\nu^{(q)} = g_{T_\nu}(\boldsymbol{r}), & \boldsymbol{r}\in\partial\Omega\cap\overline{\Omega_\nu}, \quad \nu = s,l. \end{cases} \tag{4.23}$$

6. Compute error $\mathcal{E}^{(q)}(\boldsymbol{r})$ in satisfying the Gibbs-Thomson relation (4.16) on Γ:

$$\mathcal{E}^{(q)}(\boldsymbol{r}) = T_l^{(q)} - h_G - T_{liq}\left(C_1^{(q)},\ldots,C_N^{(q)}\right) - \epsilon_v v_{\boldsymbol{n}}^{(q)}, \quad \boldsymbol{r}\in\Gamma. \tag{4.24}$$

7. If the maximum error exceeds a user-defined tolerance ϵ_{tol}, i.e if

$$\max_{\boldsymbol{r}\in\Gamma}\left|\mathcal{E}^{(q)}(\boldsymbol{r})\right| > \epsilon_{\text{tol}},$$

then adjust $C_1^{*(q)}$ by inverting the Gibbs-Thomson relation (4.16):

$$C_1^{*(q+1)} = C_1^{*(q)} + \frac{T_l^{(q)} - h_G - T_{liq}\left(C_1^{(q)},\ldots,C_N^{(q)}\right) - \epsilon_v v_{\boldsymbol{n}}^{(q)}}{m_{l1}\left(C_1^{(q)},\ldots,C_N^{(q)}\right)} \tag{4.25}$$

set $q = q+1$ and go to step 2.

In this procedure C_1 should denote the slowest component, that is, $D_1 < D_J$, $J \geq 2$, since it is the slowest component that limits the velocity of front propagation. The case of arbitrary N differs from the case $N = 1$ by the presence of step 4, which is absent for $N = 1$.

Clearly, the above splitting scheme of coupled system (4.12)-(4.19) into separate simpler BVPs is not unique. In fact, perhaps a more "symmetric" way is to solve for temperature fields both in solid and liquid phases using Dirichlet boundary conditions on Γ, compute the front velocity using the Stefan condition (4.15), use the computed velocity to solve for concentrations $\{C_J\}_{J=1}^N$ imposing the solute-rejection equations (4.17) as Robin boundary conditions and, finally, correct the guessed value for the temperature on Γ using the Gibbs-Thompson relation (4.16). However, such an iterative procedure showed to be very unstable in numerical experiments for typical parameters of metal alloy, for which the thermal diffusivity is much less than the diffusivity of solutes, i.e. $\dfrac{\lambda_\nu}{s_\nu c_{p_\nu}} \ll D_J$. It appears to be crucial to compute the front's velocity based on values of the slowest components in the system in order to obtain a stable iterative scheme (as done in the scheme above).

The success, i.e., fast convergence, of the simple iterative scheme presented above in case of binary alloys also seems to ought to the fact that the diffusivity of the quantity which is used for velocity calculations (concentration C_1) is much less than the diffusivity of the quantity which is computed using the found velocity (temperatures T_l and T_s) . In case of multialloys ($N \geq 2$) this is no longer true because velocity $v_{\boldsymbol{n}}$ is also used

in Robin boundary conditions while solving for concentrations C_J, $J \geq 2$, which may diffuse at a rate very close to the rate of C_1, that is, $D_1 \approx D_J$, $i = 2, \ldots, N$. We have found from numerical experiments that in such a case the above iterative scheme exhibits a slow convergence and even an unstable behavior in many situations.

Remark. Note that specification of function $C_1^*(\boldsymbol{r})$, $\boldsymbol{r} \in \Gamma$, uniquely defines functions $v_{\boldsymbol{n}}$, $\{C_J\}_{J=1}^N$ and $\{T_\nu\}_{\nu=s,l}$ through equations (4.20)-(4.23). Thus, these functions can be considered as functionals of function C_1^*, that is, $v_{\boldsymbol{n}} = v_{\boldsymbol{n}}[C_1^*]$, $\{C_J = C_J[C_1^*]\}_{J=1}^N$ and $\{T_\nu = T_\nu[C_1^*]\}_{\nu=s,l}$ (here the square brackets denote the functional argument), and the above scheme as a fixed-point iteration for a non-linear functional equation:

$$C_1^* = \Phi[C_1^*],$$

where functional $\Phi[\varphi]$ is defined as:

$$\Phi[\varphi] = \varphi + \frac{T_l[\varphi] - h_G - T_{liq}(C_1[\varphi], \ldots, C_N[\varphi]) - \epsilon_v v_{\boldsymbol{n}}[\varphi]}{m_{l1}(C_1[\varphi], \ldots, C_N[\varphi])}.$$

Approximate Newton iteration

In order to obtain an improved method we use the above scheme as a basis and apply variational calculus to estimate how the error in satisfying the Gibbs-Thomson relation $\mathcal{E}(\boldsymbol{r})$ at any given point changes when the boundary concentration $C_1^*(\boldsymbol{r})$ is changed by some $\Delta C_1^*(\boldsymbol{r})$. This information is then used to obtain an alternative updating formula for C_1^* (instead of (4.25)) such that $\mathcal{E}(\boldsymbol{r})$ converges to zero as quickly as possible.

In general, the change in error $\mathcal{E}(\boldsymbol{r})$ due to a change in C_1^* up to linear order can be

expressed as:

$$\Delta\mathcal{E}(\boldsymbol{r}) = \int_{\Gamma} \frac{\delta\mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')} \Delta C_1^*(\boldsymbol{r}')\, d\Gamma, \quad \boldsymbol{r} \in \Gamma,$$

where functional derivative $\dfrac{\delta\mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')}$ represents the sensitivity of $\mathcal{E}$ at point $\boldsymbol{r}$ to the change in C_1^* at point $\boldsymbol{r}'$. We refer the interested reader to 4.A where it is shown how $\dfrac{\delta\mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')}$ can be expressed as the solution to an adjoint system of PDEs corresponding to (4.20)-(4.23). In principle, one could use the above expression to find the optimal $\Delta C_1^{*,\text{best}}(\boldsymbol{r})$ that is expected to reduce $\mathcal{E}$ to zero everywhere on Γ, that is, $\Delta\mathcal{E}(\boldsymbol{r}) = -\mathcal{E}(\boldsymbol{r})$ or:

$$\mathcal{E}(\boldsymbol{r}) + \int_{\Gamma} \frac{\delta\mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')} \Delta C_1^{*,\text{best}}(\boldsymbol{r}')\, d\Gamma = 0, \quad \boldsymbol{r} \in \Gamma. \tag{4.26}$$

However solution of the above boundary integral equation is not a simple task that requires, first, an efficient calculation of functional derivative $\dfrac{\delta\mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')}$ for all interface points $\boldsymbol{r} \in \Gamma$ and, second, inversion of the convolution term. We defer the further investigation of this avenue to future works. Instead, we propose to use a greatly simplified approach in which we approximate the boundary integral in (4.26) as:

$$\int_{\Gamma} \frac{\delta\mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')} \Delta C_1^{*,\text{best}}(\boldsymbol{r}')\, d\Gamma \approx \Delta C_1^{*,\text{best}}(\boldsymbol{r}) \int_{\Gamma} \frac{\delta\mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')}\, d\Gamma.$$

Such an approach is reasonable provided the sensitivity $\dfrac{\delta\mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')}$ decays fast as the distance between $\boldsymbol{r}$ and $\boldsymbol{r}'$ increases. Using such an approximation equation (4.26) is trivially solved to obtain:

$$\Delta C_1^{*,\text{best}}(\boldsymbol{r}) \approx -\frac{\mathcal{E}(\boldsymbol{r})}{\displaystyle\int_\Gamma \frac{\delta \mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')}\, d\Gamma}. \tag{4.27}$$

Thus, instead of using (4.25), we calculate $C_1^{*(q+1)}$ as:

$$C_1^{*(q+1)} = C_1^{*(q)} - \frac{\mathcal{E}^{(q)}(\boldsymbol{r})}{G^{(q)}(\boldsymbol{r})}, \tag{4.28}$$

where $G(\boldsymbol{r}) = \displaystyle\int_\Gamma \frac{\delta \mathcal{E}(\boldsymbol{r})}{\delta C_1^*(\boldsymbol{r}')}\, d\Gamma$. Note that quantity $G(\boldsymbol{r})$ has the meaning of the directional derivative of $\mathcal{E}(\boldsymbol{r})$ in the "direction" $\delta C_1^*(\boldsymbol{r}) \equiv 1$, $\boldsymbol{r} \in \Gamma$. Using results of 4.B, $G(\boldsymbol{r})$ can be efficiently computed as:

$$G^{(q)}(\boldsymbol{r}) = \Lambda_{T_l} - \sum_{J=1}^{N} m_{lJ}(C_1, \ldots, C_N)\Lambda_{C_J} - \varepsilon_v \Lambda_v$$

where Λ_{T_l}, $\{\Lambda_{C_J}\}_{J=1}^N$ and Λ_v is the solution to the following adjoint system of PDEs:

$$\begin{cases} \left(a - D_1\nabla^2\right)\Lambda_{C_1} = 0 & \text{in } \Omega_l \\ \Lambda_{C_1} = 1 & \text{on } \Gamma \\ D_1\partial_{\boldsymbol{n}_l}\Lambda_{C_1} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_l} \end{cases} \tag{4.29}$$

$$\Lambda_v = \frac{1}{(1-k_1)C_1}\left(D_1\partial_{\boldsymbol{n}_l}\Lambda_{C_1} - v_n(1-k_1)\Lambda_{C_1}\right) \quad \text{on } \Gamma \tag{4.30}$$

$$\begin{cases} \left(a - D_J\nabla^2\right)\Lambda_{C_J} = 0 & \text{in } \Omega_l \\ D_J\partial_{\boldsymbol{n}_l}\Lambda_{C_J} - (1-k_J)v_{\boldsymbol{n}}\Lambda_{C_J} = (1-k_J)\Lambda_v C_J & \text{on } \Gamma \\ D_J\partial_{\boldsymbol{n}_l}\Lambda_{C_J} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_l} \end{cases} \tag{4.31}$$

$$\begin{cases} \left(s_\nu - \lambda_\nu\nabla^2\right)\Lambda_{T_\nu} = 0 & \text{in } \Omega_\nu, \quad \nu = s, l \\ [\Lambda_T] = 0 & \text{on } \Gamma \\ [\lambda\partial_{\boldsymbol{n}}\Lambda_T] = L_f\Lambda_v & \text{on } \Gamma \\ \lambda_\nu\partial_{\boldsymbol{n}_\nu}\Lambda_{T_\nu} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_\nu}, \quad \nu = s, l \end{cases} \tag{4.32}$$

For clarity, Algorithm 2 summarizes the overall Newton-like iterative procedure for solving the system of nonlinearly coupled PDEs based on (4.28).

1: Provide an initial guess $C_1^{*(0)}$, tolerance ϵ_{tol}, and maximum iterations allowed $q_{\max}$
2: Set $q = 0$
3: Solve (4.20) for $C_1^{(q)}$
4: Compute velocity $v_{\boldsymbol{n}}$ using (4.21)
5: Solve (4.22) for $C_J^{(q)}$, $J = 2, N$
6: Solve (4.23) for $T_\nu^{(q)}$, $\nu = l, s$
7: Compute error $\mathcal{E}(\boldsymbol{r})^{(q)}$ on Γ (4.24)
8: **if** $\max_{\boldsymbol{r}\in\Gamma} \left|\mathcal{E}^{(q)}(\boldsymbol{r})\right| > \epsilon_{\text{tol}}$ and $q < q_{\max}$ **then**
9: Solve (4.29) for $\Lambda_{C_1}^{(q)}$
10: Compute $\Lambda_v^{(q)}$ using (4.30)
11: Solve (4.31) for $\Lambda_{C_J}^{(q)}$, $J = 2, N$
12: Solve (4.32) for $\Lambda_{T_\nu}^{(q)}$, $\nu = l, s$
13: Compute $C_1^{*(q+1)}$ using (4.28)
14: Set $q \leftarrow q + 1$
15: Go to step 3
16: **end if**

Algorithm 2: An approximate Newton iteration for solving nonlinear system of equations (4.12)-(4.19)

Remark. Note that the proposed Newton-type approach requires solving twice as many BVPs compared to the simple fixed-point iteration, however the above adjoint system of equations has the same structure as system of equations for physical quantities (4.20)-(4.23). It means that the discretization matrices obtained for (4.20)-(4.23) can be reused while solving (4.29)-(4.32) resulting in computational time savings. Also note that quantity Λ_{C_1} does not change form iteration to iteration, thus equation (4.29) needs to be solved only once.

Convergence of iterative schemes

In order to gain some insight into convergence properties of the presented above iterative schemes we analyze them from point of view of the linear stability analysis in a simple

setting of quasi one dimensional planar geometry. Specifically, we consider an infinity domain with interface located at $y = 0$ such that the $y > 0$ and $y < 0$ half-spaces are occupied by liquid and solid phases, respectively. For simplicity we assume the constitutional undercooling is linear, that is, $T_{liq} = T_m + \sum_{i=1}^{N} m_{lJ} C_J$, partition coefficients are constant, and the absence of kinetic and curvature undercoolings ($\epsilon_v = 0$ and $\epsilon_c = 0$).

Denote as $\tilde{C}_J = \tilde{C}_J(y)$, $J \in [1, N]$, $\tilde{T}_s = \tilde{T}_s(y)$ and $\tilde{T}_l = \tilde{T}_l(y)$ the solution to nonlinear system of equation of (4.12)-(4.19). Let us consider an infinitesimally perturbed boundary concentration with magnitude δ_C and spatial frequency ω_x, that is:

$$C_1^{*(0)} = \tilde{C}_J(0) + \delta_C e^{-i\omega_x x}$$

We seek solutions satisfying iterative equations (4.25) and (4.28) up to linear order in δ_C of the form:

$$\begin{aligned} C_1^{*(q)} &= \tilde{C}_J(0) + r_{\text{f.p.}}^q \delta_C \exp(-i\omega_x x) + \mathcal{O}\left(\delta_C^2\right) \quad \text{and} \\ C_1^{*(q)} &= \tilde{C}_J(0) + r_{\text{a.N.}}^q \delta_C \exp(-i\omega_x x) + \mathcal{O}\left(\delta_C^2\right), \end{aligned}$$

respectively, where $r_{\text{f.p.}} = r_{\text{f.p.}}(\omega_x)$ and $r_{\text{a.N.}} = r_{\text{a.N.}}(\omega_x)$ denote amplification factors for disturbances of frequency ω_x in cases of fixed-point and approximate Newton iterations, respectively. An iterative scheme is expected to be unstable if its amplification factor is greater than one and stable otherwise where a smaller amplification factor indicating a faster convergence. Substitution of the above expressions into (4.25) and (4.28) gives (see 4.C):

$$r_{\text{f.p.}}(\omega_x) = \left(\frac{L_f}{m_{l1}(1-k_1)\tilde{C}_1}\right)\left(\frac{D_1\Omega_1(\omega_x) - (1-k_1)\tilde{v_n}}{\lambda_s\Omega_s(\omega_x) + \lambda_l\Omega_l(\omega_x)}\right) - \sum_{i=2}^{N}\left(\frac{m_{lJ}}{m_{l1}}\right)\left(\frac{1-k_J}{1-k_1}\right)\left(\frac{\tilde{C}_J}{\tilde{C}_1}\right)\left(\frac{D_1\Omega_1(\omega_x) - (1-k_1)\tilde{v_n}}{D_{lJ}\Omega_J(\omega_x) - (1-k_J)\tilde{v_n}}\right), \quad (4.33)$$

and

$$r_{\text{a.N.}}(\omega_x) = 1 - \frac{1 - r_{\text{f.p.}}(\omega_x)}{1 - r_{\text{f.p.}}(0)}. \quad (4.34)$$

where

$$\begin{aligned} \Omega_J(\omega_x) &= \sqrt{\omega_x^2 + \frac{a}{\Delta t D_{lJ}}}, \quad J \in [1, N], \\ \Omega_\nu(\omega_x) &= \sqrt{\omega_x^2 + \frac{s_\nu}{\Delta t \lambda_\nu}}, \quad \nu = s, l. \end{aligned} \quad (4.35)$$

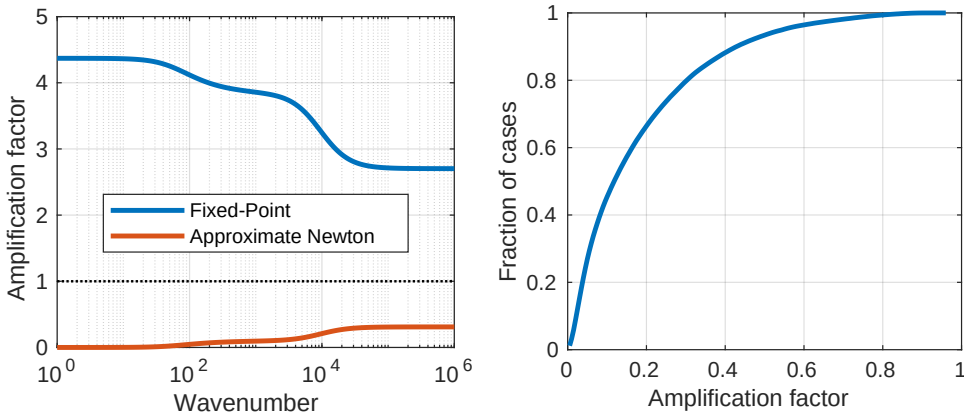
First, note that the second term in the expression for the amplification factor in the case of the fixed-point iteration is $\mathcal{O}(1)$, depending on problem parameters it can be greater or less than one. Thus, it follows immediately that the fixed-point iteration is not a robust approach for solving the nonlinear system of PDEs at hand. The expression for the amplification factor in the case of the proposed approximate Newton method is quite involved, however it appears that for typical physical parameters the value of $r_{\text{f.p.}}$ is always negative and reaches its maximum value at $\omega_x = 0$. It is straightforward to show that under this conditions $r_{\text{a.N.}} < 1$. Figure 4.2a shows the dependence of amplification factors in cases of the fixed-point iterative scheme and the approximate Newton one for a ternary alloy with parameters $C_1 = 10.7$ at%, $C_2 = 9.4$ at%, $T_m = 1900$ K, $m_{l1} = -5.43$ K/at%, $m_{l2} = -10.4$ K/at%, $k_1 = 0.94$, $k_2 = 0.83$, $D_1 = 10^{-5}$ cm^2/s, $D_2 = 2 \cdot 10^{-5}$

cm^2/s, $v_{\boldsymbol{n}} = 0.01$ cm/s, $L_f = 2600$, $\rho_l = \rho_s = 9.24 \cdot 10^{-3}$, $\lambda_l = \lambda_s = 1.3$, $c_{p_l} = c_{p_s} = 356$ (motivated by the Co-W-Al alloy simulated later in this work).

In order to further investigate the robustness of the proposed approximate Newton approach we perform a parameter sweep in ranges $C_1 \in [1, 20]$ at%, $C_2 \in [1, 20]$ at%, $m_{l1} \in [-20, -1]$ K/at%, $m_{l2} \in [-20, -1]$ K/at%, $k_1 \in [0.1, 0.9]$, $k_2 \in [0.1, 0.9]$, $D_1 \in [10^{-6}, 10^{-4}]$ cm^2/s, $D_2 \in [10^{-6}, 10^{-4}]$ cm^2/s and compute the maximum amplification factor in each case. The worst case is found to have an amplification factor of 0.9536, which still indicates convergence although a very slow one. However, the total number of cases with relatively high amplification factors is found to be small. As demonstrated in figure 4.2b the number of cases having the amplification factor 0.5 or better is more than 90%, thus we expect the proposed method to perform well for a wide range of alloys. A further investigation into developing a more accurate Newton-type method as outlined in section 4.3.2 in future works would likely result in a method performing well for alloys with any parameters.

4.3.3 Numerical Methods

Besides dealing with a system of nonlinearly coupled partial differential equations the simulation of the multialloy solidification also poses several big challenges from a more "technical" point of view. First of all, the solidification front evolves in time and undergo very large transformations (e.g., from a planar front into a forest of dendritic structures). Second, the solution of PDEs for the temperature and concentration fields



(a) (b)

Figure 4.2: Results of linear stability analysis of the fixed-point and approximate Newton iterations: (a) Dependence of the amplification factor on the perturbation's wavenumber; (b) Cumulative fraction of cases observed in the parameter sweep study with amplification factors equal or less than a given one.

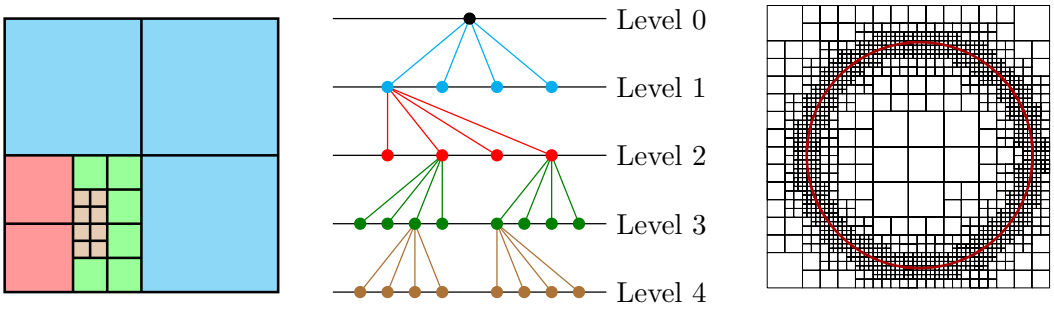
requires the imposition of Dirichlet and Robin boundary conditions and "jump" interface conditions on this very irregular interface between liquid and solid phases. Third, during the solidification of multicomponent alloys the solute-rejection phenomena leads to the development of a steep solutal boundary layer ahead of the crystallization front. It is crucial to accurately resolve such steep solutal boundary layers since they directly influence the dynamics of the process. Similarly to [159], we address these challenges by a combination of adaptive Cartesian quad-tree grids (to resolve steep gradients), Level-Set Method (to describe the front and its evolution) and sharp numerical methods for imposing boundary and interface conditions (to accurately solve BVPs). Specifically, the present work is based on a second-order accurate Level-Set framework on adaptive quadtree grids presented [100] and parallelized in [104] using scalable `p4est` grid management library [24].

Remark. Note that the proposed above Newton-type approach for solving the nonlinear system of governing equations can also be implemented in other numerical frameworks (e.g., interface-fitted finite element method).

Space discretization: Adaptive Quadtree grids

In order to efficiently address the multiscale nature of the solidification process without compromising the accuracy of numerical approximations we employ adaptive Cartesian quadtree grids to discretize the computational domain. Such grids are constructed using a selective recursive refinement of rectangular cells into four smaller equal cells starting from a root cell that represents the entire computational domain. Dimensions of cells in such a grid are equal to $(L_x \times L_y)/2^l$, where $(L_x \times L_y)$ is the root cell's size and l is an integer called *the level of refinement* and which is equal to the number of refinements necessary to make to obtain a given cell from the root cell. Usually the resolution of quad-tree grids is specified by the minimum and maximum levels of refinement, $l_{\min}$ and $l_{\max}$, that grid cells can have. Fig. 4.3 (a) illustrates the construction process and the concept of the refinement level for cells.

The spatially adaptive structure of a quad-tree grid allows to create high densities of computational nodes in areas where the finest resolution is needed while keeping the grid relatively coarse outside of such regions. Specifically, in this work we construct the computational grid such that in the band of width B around the solidification front every grid cell is refined to the highest level $l_{\max}$, while any cell outside of this band is refined if the Kth fraction of its diagonal is greater than the distance from that cell to the band.



(a) (b)

Figure 4.3: (a) illustration of the hierarchical structure of a quad-tree grid. (b) example of adaptive mesh refinement using Cartesian quad-tree grids and the refinement criterion used in this work in case of a circular interface for $l_{\min} = 3$, $l_{\max} = 6$, and $K = 1.4$.

Mathematically this criterion can be expressed as: a grid cells $\mathcal{C}$ is refined if

$$\min_{\boldsymbol{r} \in \mathcal{C}} |\mathrm{dist}(\boldsymbol{r})| < B + K \mathrm{diag}(\mathcal{C}) \tag{4.36}$$

This simple refinement criterion creates a computational grid with a band of the most finest grid cells with $l = l_{\max}$ around the solidification front that gradually coarsen to cells with $l = l_{\min}$ away front the front (see Fig. 4.3 (b)). Such a refinement strategy is adequate for simulating solidification processes considered in this work since the steepest gradients are expected in the vicinity of interface Γ. In this work we use parameters $B = 2$ and $K = 1$.

We choose to represent spatial fields by their values at corners of grid cells. Such a choice allows an easy calculation of first and second Cartesian derivatives as well as interpolation of spatial fields as described in [102, 100].

Description of solidification front: Level-Set Method

In the Level-Set Method [121] the boundary of an irregular domain is implicitly defined by the zero-isocontour of a Lipschitz-continuous function $\phi(\boldsymbol{r})$, called the level-set function, such that it has one sign inside the domain and the opposite one outside of it. In this work we set the level-set function $\phi(\boldsymbol{r})$ to be negative in the liquid phase and positive in the solid phase (see Fig. 4.4), i.e:

$$\begin{aligned} \phi(\boldsymbol{r}) &< 0 \quad \forall\ \boldsymbol{r} \in \Omega_l, \\ \phi(\boldsymbol{r}) &= 0 \quad \forall\ \boldsymbol{r} \in \Gamma, \\ \phi(\boldsymbol{r}) &> 0 \quad \forall\ \boldsymbol{r} \in \Omega_s. \end{aligned}$$

Following the common practice we choose the level-set function to be the signed distance to the interface Γ.

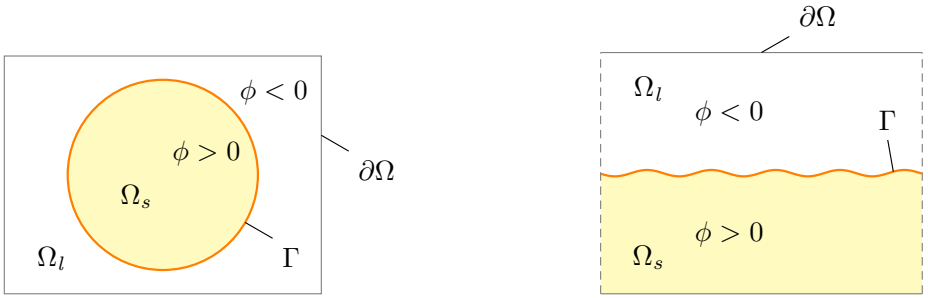


Figure 4.4: Illustration of representing irregular domains by the Level-Set Method on examples of crystal growth from a seed (left) and directional solidification (right).

Among advantages of the Level-Set Method is that it provides an easy way to compute such geometric quantities of an interface as the normal vector and its mean curvature:

$$\boldsymbol{n} = \boldsymbol{n}_s = -\boldsymbol{n}_l = -\frac{\nabla\phi}{|\nabla\phi|},$$
$$H = -\nabla\cdot\boldsymbol{n} = \nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}.$$

The evolution of a interface represented by a level-set function $\phi(\boldsymbol{r})$ in a velocity field $\boldsymbol{v}$ is described by a simple advection equation:

$$\partial_t\phi + \boldsymbol{v}\cdot\nabla\phi = 0. \tag{4.37}$$

We use a second-order accurate semi-Lagrangian method to solve the above equation. That is, the value of the level-set functions $\phi^n(\boldsymbol{r}_J)$ at a location $\boldsymbol{r}_J$ and time instant t_n is computed as the value of the level-set function at time instant t_{n-1} at the departing point $\boldsymbol{r}_d$ of the characteristic that passes through point $(\boldsymbol{r}_J, t_n)$:

$$\phi^n(\boldsymbol{r}_J) = \phi^{n-1}(\boldsymbol{r}_d),$$

where the departure point $\boldsymbol{r}_d$ is computed by tracing the characteristic backward in time using the midpoint rule as described, e.g., in [100].

To restore the signed distance property of the level-set function, which usually deteriorates during advection steps, we solve the reinitialization equation until steady state after each advection step:

$$\partial_\tau\phi^n(\tau,\boldsymbol{r}) + \mathtt{sgn}(\phi^n(0,\boldsymbol{r}))(|\nabla\phi^n(\tau,\boldsymbol{r})| - 1) = 0, \tag{4.38}$$

where τ is a fictitious time and `sgn` is the signum function. Specifically, we employ a

method based on using the Godunov Hamiltonian for discretization of $|\nabla\phi|$, TVD RK-2 time-stepping scheme and the sub-cell fix of [135] as described in [100].

During the course of simulation the solidification front may evolve in such a way that it leads to significantly under-resolved geometries. A typical situation is the slow solidification of narrow inter-dendritic gaps. The poor resolution by computational grid of such regions may cause an unstable behavior of the computational scheme. To avoid such issues after each motion of the front underresolved region (if any) are regularized as described in 4.D.

Solving BVPs

The advantages of using adaptive grids and the level-set method come at price of (1) discretizing PDEs on complex nonuniform grid structures and (2) imposing boundary conditions on implicitly defined interfaces that cut through grid lines arbitrary. We note, however, that the refinement criterion used in this work ensures that the computational grid is locally uniform near the solidification front, thus, nonuniform node arrangements (such as T-junctions and missing neighbors) are present only away from it and one needs to deal with the two tasks separately. Specifically, to for discretization of Poisson-type equations (4.12)-(4.19) on adaptive quadtree structures we use the second-order accurate approach of [102]. Dirichlet boundary conditions (for C_1) are imposed using the Shortley-Weller method [147]. Robin boundary conditions (for $\{C_J\}_{J=2}^{N}$) are imposed using the finite volume method described in [18]. Finally, to impose jump conditions (for $\{T_\nu\}_{\nu=s,l}$) we use a finite-volume approach of [19]. All the aforementioned numerical methods reduce

to the standard five-point discretization on uniform grids and away from the solidification front. This allows their seamless combination on adaptive Cartesian grids provided grids are locally uniform in the vicinity of boundaries and interfaces.

After solving each BVP, its numerical solution is smoothly extended across the interface Γ. This simplifies the calculation of the front's velocity (4.21) and also defines valid values at grid points near interface that will become part of solution domain during next time step. Specifically, we use the PDE-based quadratic extension presented in [16]. This approach consists in computing the first and second order Cartesian derivatives $\boldsymbol{q} = \nabla u$ and $\underline{\underline{\boldsymbol{Q}}} = \nabla\nabla u$ of a given spatial field u inside the domain where u is well-defined followed by the hierarchical extension of these quantities as well as the numerical values of u themselves to grid nodes outside of the domain by solving the following three advection-type equations until steady state one after another:

$$\begin{aligned}
&\partial_\tau \underline{\underline{\boldsymbol{Q}}} + \chi_{\nabla\nabla}\left(\nabla \underline{\underline{\boldsymbol{Q}}}\right)\cdot\boldsymbol{n} = 0,\\
&\partial_\tau \boldsymbol{q} + \chi_{\nabla}\left(\nabla\boldsymbol{q} - \underline{\underline{\boldsymbol{Q}}}\right)\cdot\boldsymbol{n} = 0,\\
&\partial_\tau u + \chi\left(\nabla u - \boldsymbol{q}\right)\cdot\boldsymbol{n} = 0,
\end{aligned}$$

where τ is a fictitious time and χ, χ_∇ and $\chi_{\nabla\nabla}$ denote characteristic functions representing grid nodes which did not contain valid values of u, ∇u and $\nabla\nabla u$, correspondingly. The above advection-type equations are solved using an explicit upwind scheme (see [16] for details).

Determining time-step

From the point of view of accuracy it is reasonable to select the time step such that the solidification front advances no more than a predefined fraction f_{CFL} of the smallest cell size. From the point of view of stability one has to choose the time step small enough to satisfy the stability constraint due to an explicit discretization of the curvature dependent evolution of the solidification front. In case of the solidification of pure materials such a criterion has the form (see [73]) $\Delta t_{\mathrm{crit}} = B\,(\Delta x)^{\frac{3}{2}}$, where constant B depends on parameters of the problem but not on the mesh size Δx. For the problem at hand we expect a similar dependence, however we do not attempt to analytically establish such a relation and simply find the critical time step Δt_{crit} by a trial and error approach for each run. The overall value of the time step is thus given by:

$$\Delta t^n = \min\left(f_{\mathrm{CFL}}\frac{\Delta x}{\max_{\boldsymbol{r}\in\Gamma}(v_{\boldsymbol{n}}^n)}, \Delta t_{\mathrm{crit}}\right) \tag{4.39}$$

In numerical experiments presented in this work we use $f_{\mathrm{CFL}} = 0.4$ unless stated otherwise.

Composition of the solid phase

The mathematical model of the solidification process presented in this work assumes infinitely slower transport of solutes within the solid phase compared to the liquid phase (which is a good approximation for typical metal alloys), as result the solid phase's composition has no influence on the evolution of the crystallization front. While it is

not necessary to solve for and keep track of concentration fields inside the solid phase $\{C_J^s(t,\boldsymbol{r})\}_{J=1}^N$ for simulating the solidification process, such information is of greatest importance for the analysis of resulting crystals.

Assuming negligible diffusion in the solid, the governing equations for solutes' concentrations can be formally written as:

$$\begin{cases} \partial_t C_J^s = 0, & \text{in } \Omega_s, \\ C_J^s = k_J C_{lJ}, & \text{on } \Gamma. \end{cases}, \quad J \in [1, N]$$

In other words, once the alloy at a point in space $\boldsymbol{r}_* \in \Omega$ has crystallized at some time $t = t_*$, that is, $\boldsymbol{r}_* \in \Gamma(t_*)$, its compositions at this point, given by $\{C_J^s(t,\boldsymbol{r}_*) = k_J C_{lJ}(t_*,\boldsymbol{r}_*)\}_{J=1}^N$, stays unchanged for $t > t_*$.

Our numerical method for determining the solid composition mimics this behavior. Specifically, after each motion of the solidification front, for each grid node $\boldsymbol{r}_p$ belonging to the solid phase, i.e, $\boldsymbol{r}_p \in \Omega_s$, it is checked whether the phase transition happened during the last front's movement, that is, whether $\phi^n(\boldsymbol{r}_p) > 0$ and $\phi^{n-1}(\boldsymbol{r}_p) < 0$. If so, the time moment of the phase transition at node $\boldsymbol{r}_p$ is approximated as:

$$t_*^{(p)} = t_{n-1} + \frac{|\phi_{n-1}|}{|\phi_{n-1}| + |\phi_n|}\Delta t_n,$$

which is used to estimate interface Γ composition when it travels across the grid node:

$$C_{sJ} = k_J C_{lJ}(t_*^{(p)}) = k_J \left(\frac{t_n - t_*^{(p)}}{\Delta t_n} C_{lJ}^{n-1} + \frac{t_*^{(p)} - t_{n-1}}{\Delta t_n} C_{lJ}^n + \mathcal{O}\left(\Delta t^2\right) \right),$$

where we used a linear interpolation between time instants t_{n-1} and t_n to compute

$C_{lJ}(t_*^{(p)})$. Once the solid's composition is obtained at a "just solidified" grid node it is stored unchanged for the remaining of a simulation run. In addition to the composition values we also compute and store such quantities of the front as the normal velocity, temperature, curvature, orientation at the moment of crystallization to obtain a comprehensive information about the process.

Since there is no species transport in the solid any frozen steep concentration gradients persist in time even when the solidification front has moved significantly far (in some sense concentration profiles in the solid phase remember the history of all front's locations). Thus the simple refinement strategy used for simulating the dynamics of the solidification process, namely, a very fine grid near interface Γ that coarsens away from it, is not adequate for representing composition fields in the solid. We address this issue by having a second grid which is refined to $l_{\max}$ everywhere in Ω_s for storing values of $\{C_J^s\}_{J=1}^N$. Because no other mathematical operations are performed on this grid such an approach has negligible effect on the overall computational time.

4.3.4 Overall simulation procedure

The overall computational procedure for simulating the multialloy solidification can be summarized in the following way:

1. Set $n = 0$ and $t_0 = 0$.

2. Provide initial conditions for the front's shape Γ^0 and velocity $v_{\boldsymbol{n}}^0$, temperature field $\{T_\nu^0\}_{\nu=l,s}$, and concentration fields $\{C_J^0\}_{J=1}^N$.

3. Set $n \leftarrow n+1$.

4. Compute time step Δt_n according to (4.39) and set $t_n = t_{n-1} + \Delta t_n$.

5. Advance solidification front $\Gamma^{n-1} \to \Gamma^n$ by solving (4.37).

6. Refine/coarsen computational grid according to (4.36).

7. Reinitialize the level-set function by solving (4.38) to restore the signed-distance property.

8. Compute properties of the just solidified material as described in section 4.3.3.

9. Solve nonlinear system of equations (4.12)-(4.19) for $\{T_\nu^n\}_{\nu=l,s}$, $\{C_J^n\}_{J=1}^N$ and $v_{\boldsymbol{n}}^n$ using Algorithm 2 and initial guess $C_1^{*(0)} = C_1^{n-1}$.

10. If $t_n < t_{\text{final}}$, go to step 3.

4.4 Results

In this section we present a number of simulation related to solidification of a Co-W-Al alloy with initial composition $C_{\text{W}}^\infty = 10.7$ at% and $C_{\text{Al}}^\infty = 9.4$ at% studied experimentally in [165]. First, we present tests for validation of the computational approach and then analyze the solutal segregation during the directional solidification. The parameters of the alloy are selected in the following way. The density, heat capacity, thermal conductivity and latent heat of fusion are estimated as weighted averages of those of pure elements: $\rho_l = \rho_s = 9.24 \cdot 10^{-3}$ kg$\cdot$cm^{-3}, $c_{p_l} = c_{p_s} = 356$ J$\cdot$kg$^{-1}\cdot$K^{-1}, $\lambda_l = \lambda_s = 1.3$ W$\cdot$cm$^{-1}\cdot$K^{-1},

$L_f = 2590\ \mathrm{J}\cdot\mathrm{cm}^{-3}$. Parameters of the kinetic and curvature undercoolings are taken as $\varepsilon = 0$, $\epsilon_c = 10^{-5}$ cm, and $\epsilon_v = 5\cdot 10^{-2}\ \mathrm{s}\cdot\mathrm{cm}^{-1}$. Solutes diffusivities are set to $D_{\mathrm{W}} = 10^{-5}$ $\mathrm{cm}^2\cdot\mathrm{s}^{-1}$, $D_{\mathrm{Al}} = 2\cdot 10^{-5}\ \mathrm{cm}^2\cdot\mathrm{s}^{-1}$. The dependence of the liquidus surface and partition

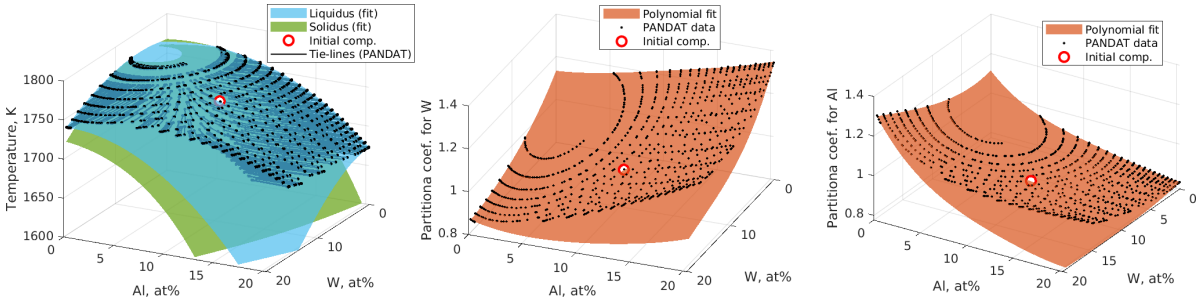


Figure 4.5: *Phase-diagram of Co-W-Al as predicted by the PANDAT^{TM}database and polynomial approximations used in this work.*

coefficients on the alloy composition are approximated by the fourth-order polynomials

$$\Delta T_C(C_{\mathrm{W}}, C_{\mathrm{Al}}) = \sum_{p=0}^{4}\sum_{q=0}^{p} a^{\Delta T}_{p-q,q} C_{\mathrm{W}}^{p-q} C_{\mathrm{Al}}^{q},$$

$$k_{\mathrm{W}}(C_{\mathrm{W}}, C_{\mathrm{Al}}) = \sum_{p=0}^{4}\sum_{q=0}^{p} a^{\mathrm{W}}_{p-q,q} C_{\mathrm{W}}^{p-q} C_{\mathrm{Al}}^{q},$$

$$k_{\mathrm{Al}}(C_{\mathrm{W}}, C_{\mathrm{Al}}) = \sum_{p=0}^{4}\sum_{q=0}^{p} a^{\mathrm{Al}}_{p-q,q} C_{\mathrm{W}}^{p-q} C_{\mathrm{Al}}^{q},$$

fitted to data obtained from the PANDAT$^{\mathrm{TM}}$database (see Figure 4.5).

4.4.1 Validation of the numerical approach: Axisymmetric stable solidification

To validate the proposed computational approach we consider the problem of an axisymmetric solidification of a ternary alloy in the infinite domain due to a line sink, which

has an analytical similarity solution (see 4.E) in the case of absence of kinetic and curvature undercooling effects, that is, $\epsilon_c = 0$ and $\epsilon_v = 0$. In order to avoid the necessity of simulating a singular heat source and an infinite domain, we confine the solidification process into an annular region with the internal radius $r_{\text{in}} = 0.1L$ and external radius $r_{\text{out}} = 0.45L$, impose on boundaries of this region Dirichlet boundary conditions for the heat and concentration fields based on the analytical solution, and set the starting position of the solidification front at $r_{\text{start}} = 0.2L$, where $L = 0.02$ cm denotes the simulation scale. We select the analytical solution that satisfies the following conditions: at the infinity concentration of solutes approaches the nominal alloy composition, that is, $C_{\text{W}}^{\infty} = 10.7$ at% and $C_{\text{Al}}^{\infty} = 9.4$ at%; the normal front's velocity at the initial moment is equal to $v_{\boldsymbol{n}}^{0} = 0.01$ cm/s; and the ratio of the compositional to thermal gradients at the solidification front is $M_0 = 0.75$ (see 4.E for details).

Figure 4.6 illustrates simulation results for the distribution of W in the solid and liquid phases at several moments of time.

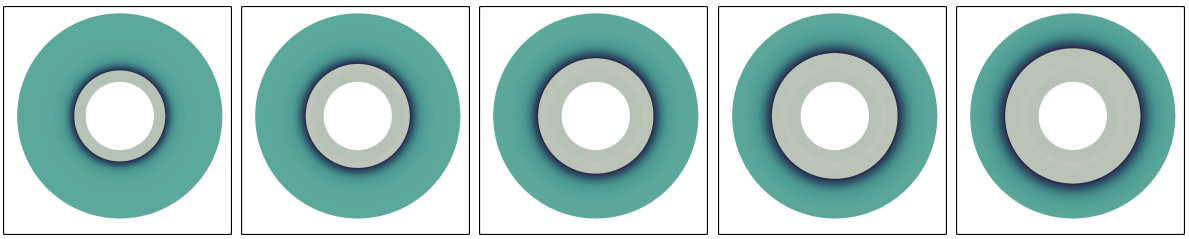


Figure 4.6: *Visualization of the W concentration field obtained in the case of axisymmetric solidification at several moments of time.*

Figure 4.7 shows the convergence of the Newton-type approach to the Gibbs-Thomson condition during a single time step and the deviation from the Gibbs-Thomson condition for all time steps on a 128×128 computational grid. As one can see the proposed

approach is able to maintain the error in satisfying interface conditions under 10^{-5} K throughout the entire course of simulation.

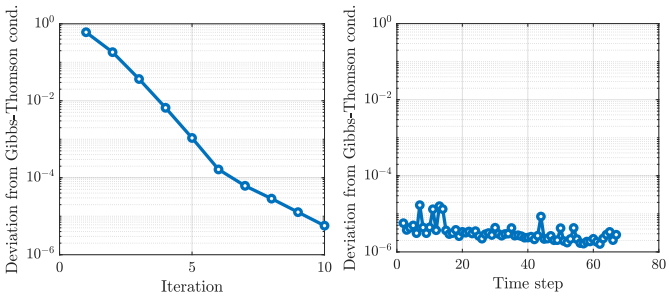


Figure 4.7: *Performance of the proposed approximate Newton method in the case of stable axisymmetric solidification. Left: convergence to the Gibbs-Thomson condition during one time step. Right: deviation from the Gibbs-Thomson condition at all time steps.*

The accuracy analysis for the temperature field, concentration fields, front's velocity and position is presented in Figure 4.8. The error of each quantity is defined as the maximum error in the L^{∞}-norm occurred throughout the entire course of simulation. The observed convergence rates are close to 2, which is consistent with the fact that second-order accurate numerical methods are used in all components of the overall computational approach.

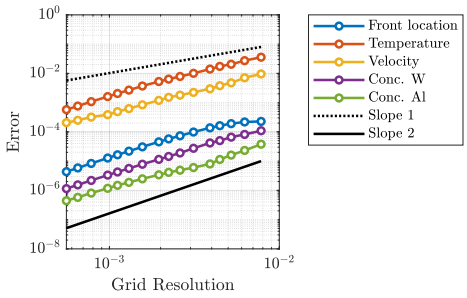


Figure 4.8: *Overall accuracy of the computational method in the case of stable axisymmetric solidification*

4.4.2 Directional solidification of a Co-W-Al ternary alloy

In this example we demonstrate the robustness of the proposed approach under different solidification regimes. In order to simulate the directional solidification of the Co-W-Al alloy, we consider a rectangular computational domain periodic in the x direction and having dimensions $L \times 10L$, where L is the scale of the simulation domain. The solidification front travels along the positive y direction. The processing conditions are modeled by imposing thermal fluxes at the top and bottom boundaries of the computational domain, such that

$$\begin{aligned}
&\lambda_l \partial_{\boldsymbol{n}} T_l = \lambda_l G_T, \quad y = 10L, \\
&\lambda_s \partial_{\boldsymbol{n}} T_s = -\lambda_l G_T + V\left(L_f + \rho_l c_{p_l} G_T\left(10L\right)\right), \quad y = 0.
\end{aligned}$$

where G_T is desired temperature gradient and V has the meaning of the approximate front velocity if the solidification were to occur in the planar regime. As the initial conditions we take a planar solidification front traveling upwards with the velocity V with temperature and concentration fields satisfying:

$$\begin{aligned}
C_{\mathrm{W}} &= C_{\mathrm{W}}^{\infty} + \frac{1-k_{\mathrm{W}}}{k_{\mathrm{W}}} \exp\left(-\frac{V}{D_{\mathrm{W}}}\left(y-y_0\right)\right), \\
C_{\mathrm{Al}} &= C_{\mathrm{Al}}^{\infty} + \frac{1-k_{\mathrm{Al}}}{k_{\mathrm{Al}}} \exp\left(-\frac{V}{D_{\mathrm{Al}}}\left(y-y_0\right)\right), \\
T_l &= T_{liq}\left(\frac{C_{\mathrm{W}}^{\infty}}{k_{\mathrm{W}}}, \frac{C_{\mathrm{Al}}^{\infty}}{k_{\mathrm{Al}}}\right) + \frac{\alpha_l}{V} G_T \left(1 - \exp\left(-\frac{V}{\alpha_l}\left(y-y_0\right)\right)\right), \\
T_s &= T_{liq}\left(\frac{C_{\mathrm{W}}^{\infty}}{k_{\mathrm{W}}}, \frac{C_{\mathrm{Al}}^{\infty}}{k_{\mathrm{Al}}}\right) + \frac{\alpha_l}{V} \left(\frac{\lambda_l}{\lambda_s} G_T + \frac{V L_f}{\lambda_s}\right) \left(1 - \exp\left(-\frac{V}{\alpha_l}\left(y-y_0\right)\right)\right),
\end{aligned}$$

where $y_0 = 0.1L$ is the initial front's location.

Figure 4.9 shows the obtained crystallized microstructures in the case $L = 0.04$ cm, $V = 0.01$ cm/s, and G_T in the range from 100 K/cm to 5000 K/cm while Figure 4.10 demonstrates corresponding numerical deviations from the Gibbs-Thomson condition for each time step. Despite a very high complexity of the front evolution the proposed computational method is able to maintain the maximum deviation at the level around 10^{-2} K and the average deviation at the level of 10^{-4}-10^{-3} K.

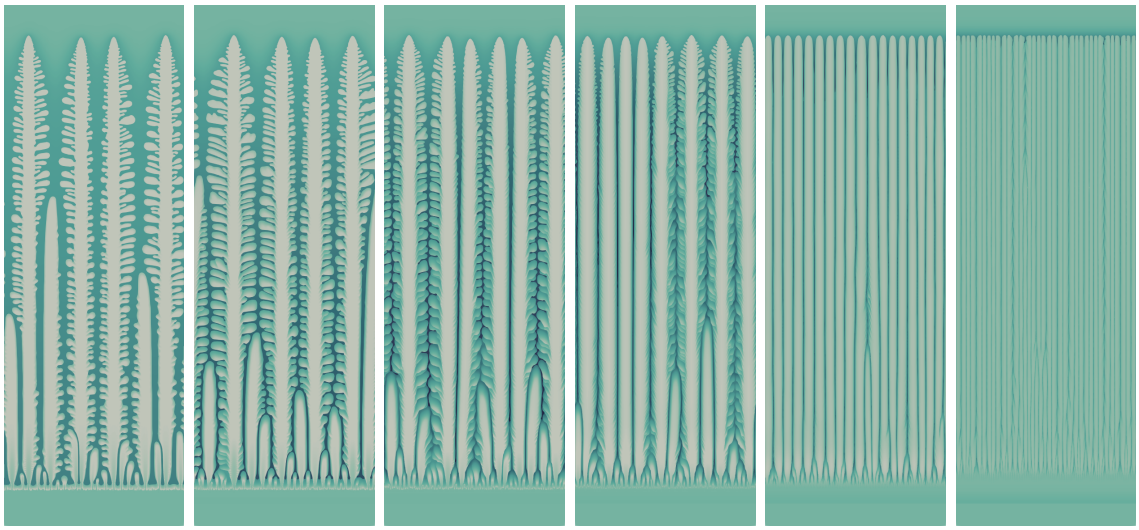


Figure 4.9: *Solidification microstructures obtained for $L = 0.04$ cm, $V = 0.01$ cm/s, and G_T =100 K/cm, 250 K/cm, 500 K/cm, 1000 K/cm, 2500 K/cm, and 5000 K/cm (left to right). Displayed in color is the concentration field of W.*

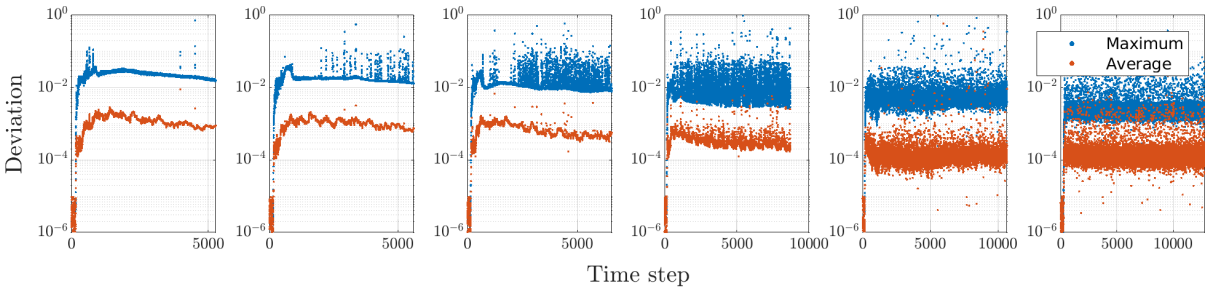


Figure 4.10: *Maximum and average deviation (in K) from the Gibbs-Thomson conditions for all time steps during simulation runs corresponding to shown in Figure 4.9*

4.4.3 Analysis of solutal segregation

Now we turn our attention to analyzing the dependence of solutal segregation on processing conditions and alloy parameters. In order to characterize concentration distributions inside the solidified material we apply a similar procedure to the one used in experimental studies: we sample concentration fields in the region form $1.8L$ to $2.0L$ and apply the sorting procedure based on the concentration of Co. Each sampling point is assigned a value of the solid fraction from 0 to 1 based on its ranking in the sorted array. To account for the fact that in numerical simulations the interdendritic region remains not completely solidified due to a finite computational time we directly compute the solid fraction in the region from $1.8L$ to $2.0L$ at the end of simulations, which corresponds to the maximum solid fraction achieved, and the originally assigned values of the solid fraction are then scaled by the computed quantity. Figure 4.11 demonstrates the concentration profiles for both solutes W and Al in cases of different solidification regimes corresponding to the ones presented in 4.9. First, one can note that up until the value of solid fraction of approximately 0.3, which corresponds to the cores of dendrites or cells, the obtained concentration profiles rise very slowly at an nearly constant concentration level. This indicates that the cores of dendrites/cells have a nearly uniform concentration distribution with the value that depends on the growth regime. From figure 4.11 it can be seen this concentration value varies rapidly from the planar regime to the cellular one, however, with a further transition into the dendritic regime this value very quickly saturates and remains the same and independent of the imposed thermal gradient. This

can be explained by the fact that dendritic tips become more isolated from each and their growth becomes almost independent. Sections of concentration profiles beyond the value of solid fraction of 0.3 corresponds to the growth of the side branches. As one can see the rate of concentration growth with respect to solid fraction is much higher than in the preceding section and strongly depends on the value of imposed thermal gradient. The more unstable is the growth regime (the lower temperature gradient) the higher is the tip velocity of side branches, which allows them to penetrate into regions with lower concentration levels and, thus, the observed rate of concentration rise is lower. Finally, at a certain value of solid fraction (different for each processing conditions) the concentration in the interdendritic region starts to rise rapidly causing the further solidification in this region to become very slow, compared to the tip velocity of primary dendrites. A further investigation of this stage requires much longer computational times and will be performed in future works.

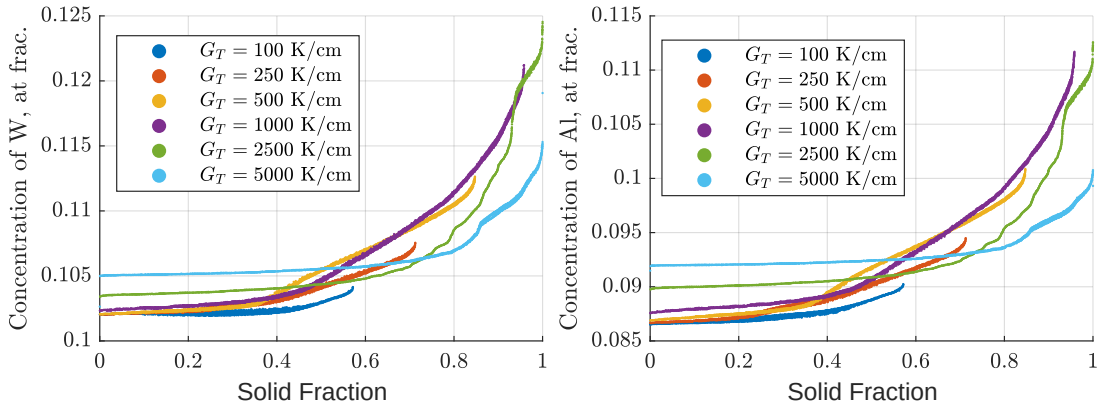
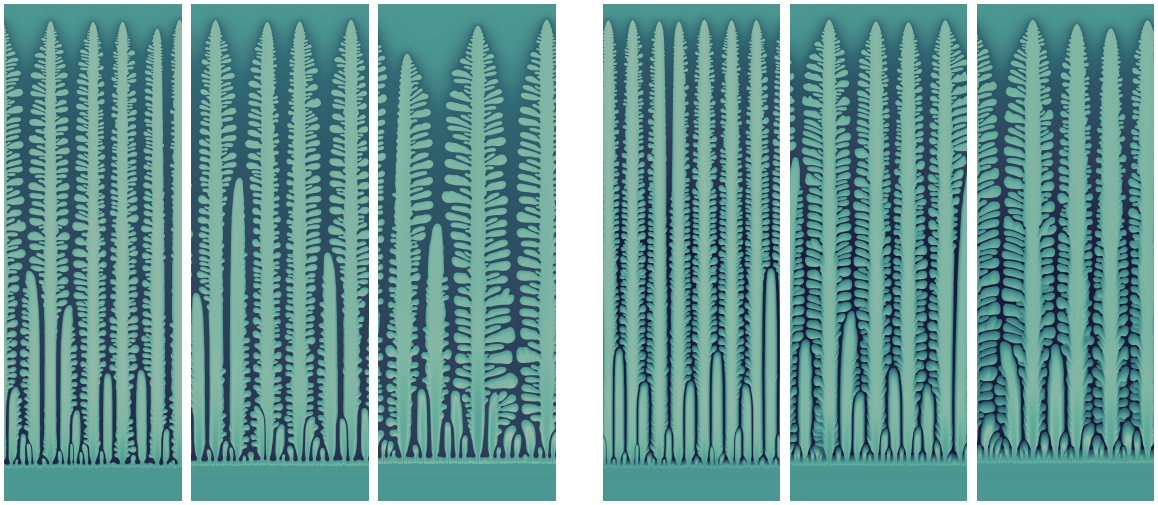


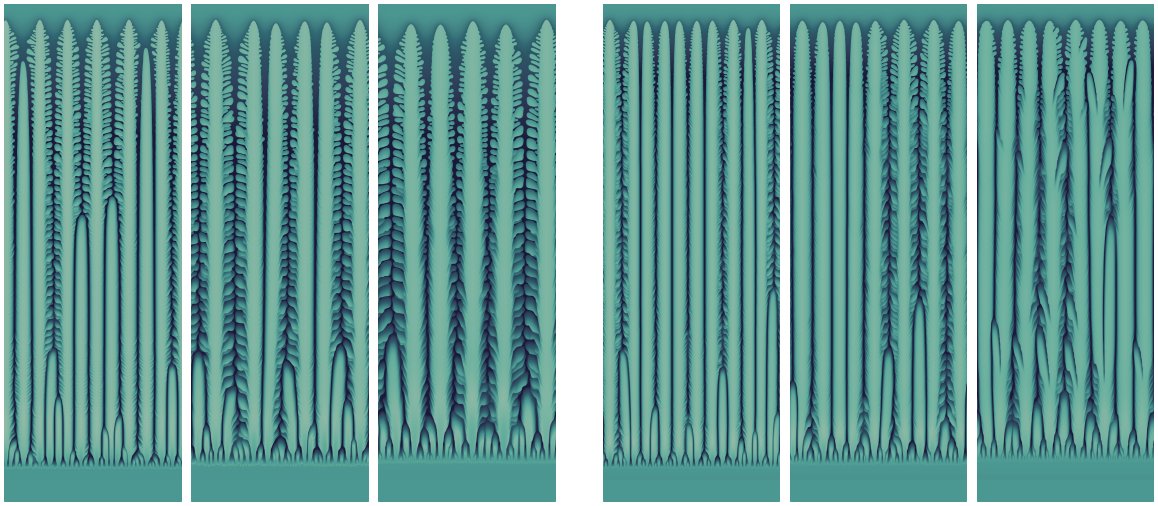
Figure 4.11: *Segregation profiles for W (left) and Al (right) for different processing conditions*

Next we investigate the influence of solutes' diffusivity on the segregation behavior.



(a) $G_T = 100$ K/cm

(b) $G_T = 250$ K/cm



(c) $G_T = 500$ K/cm

(d) $G_T = 1000$ K/cm

Figure 4.12: Visualization of simulated microstructures and concentration distributions of W for diffusivity ratios $D_{Al}/D_W = 1$, $D_{Al}/D_W = 2$, and $D_{Al}/D_W = 4$ (left to right in each subfigure).

Specifically, we keep the diffusivity of W unchanged $D_W = 10^{-5}$ while varying the diffusivity of Al from $D_{Al} = 1$ to $D_{Al} = 2$ to $D_{Al} = 4$. Figure 4.12 illustrates the obtained microstructures and concentration distributions for each of the values of diffusivity ratio

in different growth regimes. Comparing these results with figure 4.9, one can conclude that increasing the diffusivity of the third alloy component has an effect similar to increasing the imposed thermal gradient. Specifically, the interdendritic region solidifies faster and side branches are coarser. This can be explained by the fact that the higher diffusivity of *Al* allows this component to escape the interdendritic region faster, thus lowering the effective crystallization temperature. Figures 4.13-4.16 show the dependence of concentration profiles for each considered ratio of solutal diffusivities. As one can see the concentration profiles are changed only very slightly and likely due to the aforementioned changes in solidified microstructures.

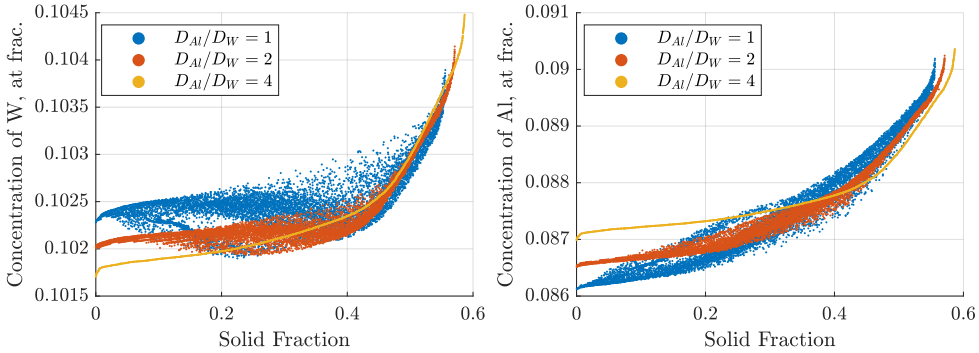


Figure 4.13: *Dependence of segregation profiles for W (left) and Al (right) on the ratio of solutal diffusivities in the case* $G_T = 100$ *K/cm.*

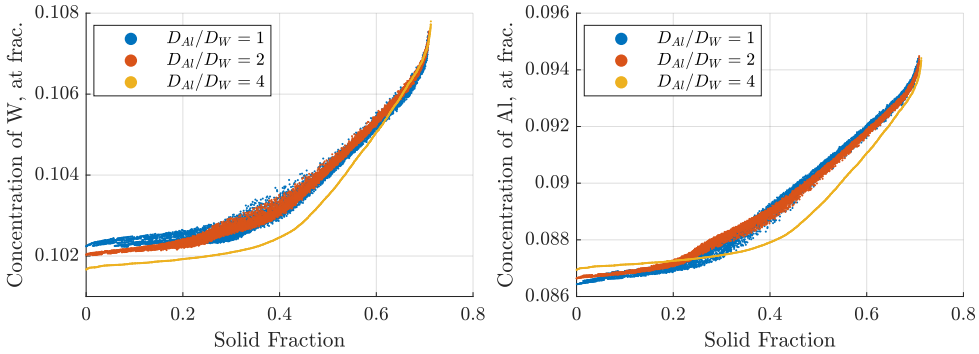


Figure 4.14: *Dependence of segregation profiles for W (left) and Al (right) on the ratio of solutal diffusivities in the case* $G_T = 250$ *K/cm.*

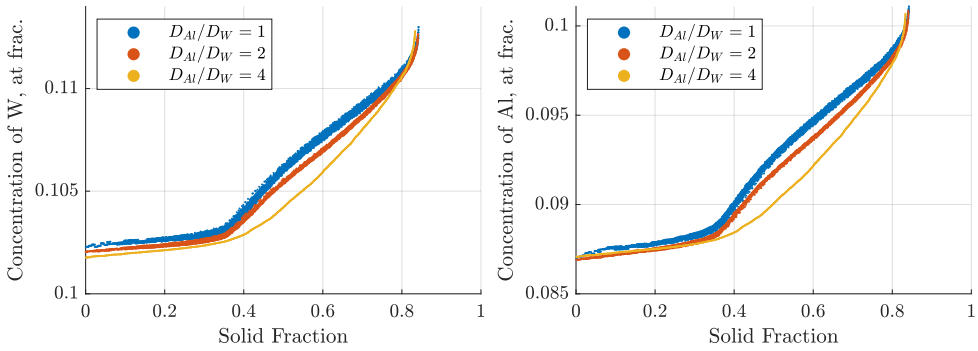


Figure 4.15: *Dependence of segregation profiles for W (left) and Al (right) on the ratio of solutal diffusivities in the case* $G_T = 500$ *K/cm.*

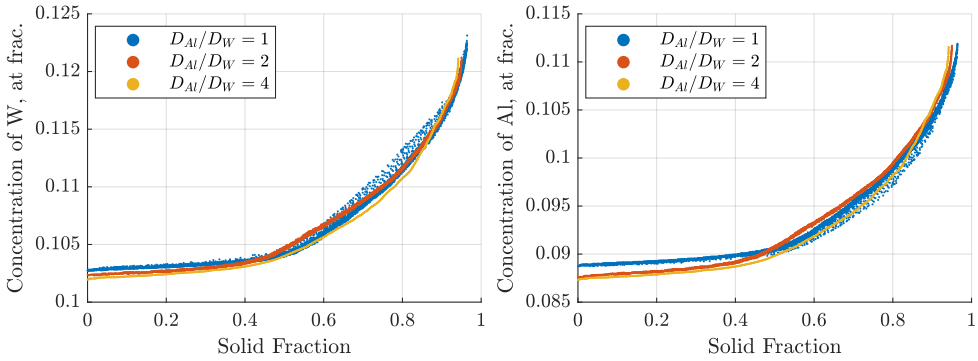


Figure 4.16: *Dependence of segregation profiles for W (left) and Al (right) on the ratio of solutal diffusivities in the case* $G_T = 1000$ *K/cm.*

4.5 Conclusion

In this chapter we have presented a computational method for the simulation of multialloy solidification. This method is based on a novel Newton-type approach for solving the nonlinearly coupled system of PDEs describing the solidification processes. The numerical implementation employs adaptive Cartesian quadtree grids, the Level Set method, and a combination of finite-volume and finite-difference schemes for solving PDEs in irregular domains. In validation tests involving stable axisymmetric solidification the presented method demonstrated second order of accuracy. Finally, the directional solidification of a Co-Al-W alloy was considered and the dependence of the solute segregation on processing

conditions and alloy parameters was analyzed.

The further development of the presented numerical method will include addition of convective effects and remelting processes. Additionally, alternative Newton-type procedures based on solving (4.26) more accurately, e.g. using the Fourier transform, will be investigated.

4.A Functional derivative with respect to δC_1^*

Consider a generic functional $\mathcal{F}$ defined as an integral of some function $\zeta(\boldsymbol{r})$ over interface Γ:

$$\mathcal{F} = \int_{\Gamma} \zeta(\boldsymbol{r})\, d\Gamma, \tag{4.40}$$

where function $\zeta(\boldsymbol{r})$ is a combination of solutions $\{C_J\}_{J=1}^{N}$, $\{T_\nu\}_{\nu=s,l}$, and $v_{\boldsymbol{n}}$ to BVPs (4.20)-(4.23). In the case considered in this work:

$$\begin{aligned}\zeta(\boldsymbol{r}) &= \delta(\boldsymbol{r}-\boldsymbol{r}_0)\mathcal{E}(\boldsymbol{r}) \\ &= \delta(\boldsymbol{r}-\boldsymbol{r}_0)(T_l(\boldsymbol{r}) - h_G(\boldsymbol{r}) - T_{liq}(C_1(\boldsymbol{r}),\ldots,C_N(\boldsymbol{r})) - \epsilon_v v_{\boldsymbol{n}}(\boldsymbol{r})).\end{aligned}$$

The deferential of $\mathcal{F}$ with respect to C_1^* can be obtained using a Lagrange multiplier approach. "Constraints" (4.20)-(4.23) can be incorporated into a Lagrangian in different ways. One approach is to treat every equations from (4.20)-(4.23) as a separate constraint. Another approach is to first obtain variational formulations of BVPs (4.20)-(4.23) and then use them in the Lagrangian definition. For sufficiently regular functions, the two approaches are equivalent and, roughly speaking, merely define the order of mathematical operations and number of Lagrange multipliers (the latter approach introduces less multipliers since for each BVP several equations, namely, a PDE and BCs, are combined into a single constraint, a variational form of BVP). For its conceptual simplicity, we employ the former approach. That is, we formally define a Lagrangian as:

$$\begin{aligned}\mathcal{L} = \mathcal{F} + \sum_{\nu=s,l} \big\{ \omega_{T_\nu} \cdot (\text{PDE for } T_\nu) + \beta_{T_\nu} \cdot (\text{BC for } T_\nu \text{ on } \partial\Omega) \big\} \\ + \gamma_T \cdot (\text{Cond for } [T] \text{ on } \Gamma) + \gamma_S \cdot (\text{Cond for } [\lambda \partial_{\boldsymbol{n}} T] \text{ on } \Gamma) \\ + \sum_{J=1}^{N} \big\{ \omega_{C_J} \cdot (\text{PDE for } C_J) + \gamma_{C_J} \cdot (\text{BC for } C_J \text{ on } \Gamma) + \beta_{C_J} \cdot (\text{BC for } C_J \text{ on } \partial\Omega) \big\} \\ + \gamma_v \cdot (\text{Equation for } v_{\boldsymbol{n}}) \end{aligned} \quad (4.41)$$

where $\{\omega_{T_\nu}, \beta_{T_\nu}\}_{\nu=s,l}$, γ_T, γ_S, $\{\omega_{C_J}, \gamma_{C_J}, \beta_{C_J}\}_{J=1}^{N}$, and γ_v are Lagrange multipliers. Note that these multipliers are functions of spatial variables with dimensionalities corresponding to constraints they enforce, that is, ω_{T_l} and $\{\omega_{C_J}\}_{J=1}^{N}$ are defined in Ω_l; ω_{T_s} is defined in Ω_s; γ_T, γ_S, $\{\gamma_{C_J}\}_{J=1}^{N}$, and γ_v are defined on Γ; β_{T_l} and $\{\beta_{C_J}\}_{J=1}^{N}$ are defined on $\partial\Omega \cap \overline{\Omega_l}$; finally, β_{T_s} is defined on $\partial\Omega \cap \overline{\Omega_s}$. Then the differential of $\mathcal{F}$ is equal to a variation of $\mathcal{L}$ with respect to C_1^*:

$$d\mathcal{F} = \delta_{C_1^*}\mathcal{L} = \int_\Gamma \frac{\delta \mathcal{L}}{\delta C_1^*}(\boldsymbol{r}) \delta C_1^*(\boldsymbol{r})\, d\Gamma,$$

provided the following conditions are satisfied:

$$\begin{array}{llll} \delta_{\omega_{T_\nu}}\mathcal{L} = 0 & \forall\, \delta\omega_{T_\nu}, \quad \nu = s,l, & \delta_{T_\nu}\mathcal{L} = 0 & \forall\, \delta T_\nu, \quad \nu = s,l, \\ \delta_{\omega_{C_J}}\mathcal{L} = 0 & \forall\, \delta\omega_{C_J}, \quad J \in [1,N], & \delta_{C_J}\mathcal{L} = 0 & \forall\, \delta C_J, \quad J \in [1,N], \\ \delta_{\gamma_v}\mathcal{L} = 0 & \forall\, \delta\gamma_v. & \delta_{v_{\boldsymbol{n}}}\mathcal{L} = 0 & \forall\, \delta v_{\boldsymbol{n}}, \\ \delta_{\gamma_{C_J}}\mathcal{L} = 0 & \forall\, \delta\gamma_{C_J}, \quad J \in [1,N], & & \\ \delta_{\gamma_T}\mathcal{L} = 0 & \forall\, \delta\gamma_T, & & \\ \delta_{\gamma_S}\mathcal{L} = 0 & \forall\, \delta\gamma_S, & & \\ \delta_{\beta_{T_\nu}}\mathcal{L} = 0 & \forall\, \delta\beta_{T_\nu}, \quad \nu = s,l, & & \\ \delta_{\beta_{C_J}}\mathcal{L} = 0 & \forall\, \delta\beta_{C_J}, \quad J \in [1,N], & & \end{array} \quad (4.42)$$

Conditions from the left column above ensure that "constraints" for $\{C_J\}_{J=1}^N$, $\{T_\nu\}_{\nu=s,l}$ and $v_{\boldsymbol{n}}$ are satisfied, while conditions from the right column give equations for Lagrange multipliers.

The explicit expression for the Lagrangian can be written as:

$$
\begin{aligned}
\mathcal{L} = &\int_\Gamma \zeta(\boldsymbol{r})\, d\Gamma \\
&+ \sum_{\nu=s,l} \Bigg\{ \int_{\Omega_\nu} \overbrace{\big(s_\nu T_\nu - \lambda_\nu \nabla^2 T_\nu - f_{T_\nu}\big)}^{\text{PDE for } T_\nu} \omega_{T_\nu}\, d\Omega + \int_{\partial\Omega \cap \overline{\Omega_l}} \overbrace{(\lambda_\nu \partial_{\boldsymbol{n}_\nu} T_\nu - g_{T_\nu})}^{\text{BC for } T_\nu} \beta_{T_\nu}\, d\Gamma \Bigg\} \\
&\quad + \int_\Gamma \underbrace{(T_s - T_l - h_T)}_{\text{Cond for } [T]} \gamma_T\, d\Gamma + \int_\Gamma \underbrace{(\lambda_s \partial_{\boldsymbol{n}_s} T_s + \lambda_l \partial_{\boldsymbol{n}_l} T_l - L v_{\boldsymbol{n}} - h_S)}_{\text{Cond for } [\lambda \partial_{\boldsymbol{n}} T]} \gamma_S\, d\Gamma \\
&+ \sum_{J=1}^N \Bigg\{ \int_{\Omega_l} \underbrace{\big(aC_J - D_J \nabla^2 C_J - f_{C_J}\big)}_{\text{PDE for } C_J} \omega_{C_J}\, d\Omega + \int_{\partial\Omega \cap \overline{\Omega_l}} \underbrace{(D_J \partial_{\boldsymbol{n}_l} C_J - g_{C_J})}_{\text{BC for } C_J \text{ on } \partial\Omega} \beta_{C_J}\, d\Gamma \Bigg\} \\
&\quad + \int_\Gamma \underbrace{(C_1 - C_1^*)}_{\text{BC for } C_1 \text{ on } \Gamma} \gamma_{C_1}\, d\Gamma + \sum_{J=2}^N \int_\Gamma \underbrace{(D_J \partial_{\boldsymbol{n}_l} C_J - (1-k_J) v_{\boldsymbol{n}} C_J - h_{C_J})}_{\text{BC for } C_J \text{ on } \Gamma} \gamma_{C_J}\, d\Gamma \\
&\qquad\qquad + \int_\Gamma \underbrace{(D_1 \partial_{\boldsymbol{n}_l} C_1 - (1-k_1) v_{\boldsymbol{n}} C_1 - h_{C_1})}_{\text{Equation for } v_n} \gamma_v\, d\Gamma
\end{aligned}
$$

Taking variations of the Lagrangian with respect to multipliers $\{\omega_{T_\nu}, \beta_{T_\nu}\}_{\nu=s,l}$, γ_T, γ_S, $\{\omega_{C_J}, \gamma_{C_J}, \beta_{C_J}\}_{J=1}^N$, and γ_v, it is trivial to confirm that conditions from the left column of (4.42) lead to equations (4.20)-(4.23).

Variations with respect to $\{C_J\}_{J=1}^N$, $\{T_\nu\}_{\nu=s,l}$ and $v_{\boldsymbol{n}}$ are equal to:

$$
\begin{aligned}
\delta_{C_1}\mathcal{L} &= \int_\Gamma \zeta'_{C_1}\delta C_1\,d\Gamma \\
&\quad + \int_{\Omega_l}\left(a\delta C_1 - D_1\nabla^2\delta C_1\right)\omega_{C_1}\,d\Omega + \int_{\partial\Omega\cap\overline{\Omega_l}}\beta_{C_1}D_1\partial_{\boldsymbol{n}_l}\delta C_1\,d\Gamma \\
&\quad + \int_\Gamma\left(\gamma_{C_1} - (1-k_1)v_{\boldsymbol{n}}\gamma_v\right)\delta C_1\,d\Gamma + \int_\Gamma \gamma_v D_1\partial_{\boldsymbol{n}_l}\delta C_1\,d\Gamma, \\
\delta_{C_J}\mathcal{L} &= \int_\Gamma \zeta'_{C_J}\delta C_J\,d\Gamma \\
&\quad + \int_{\Omega_l}\left(a\delta C_J - D_J\nabla^2\delta C_J\right)\omega_{C_J}\,d\Omega + \int_{\partial\Omega\cap\overline{\Omega_l}}\beta_{C_J}D_J\partial_{\boldsymbol{n}_l}\delta C_J\,d\Gamma \\
&\quad - \int_\Gamma \gamma_{C_J}(1-k_J)v_{\boldsymbol{n}}\delta C_J\,d\Gamma + \int_\Gamma \gamma_{C_J}D_J\partial_{\boldsymbol{n}_l}\delta C_J\,d\Gamma, \quad i=2,\ldots,N, \\
\delta_{T_s}\mathcal{L} &= \int_\Gamma \zeta'_{T_s}\delta T_s\,d\Gamma \\
&\quad + \int_{\Omega_s}\left(s_s\delta T_s - \lambda_s\nabla^2\delta T_s\right)\omega_{T_s}\,d\Omega + \int_{\partial\Omega\cap\overline{\Omega_s}}\beta_{T_s}\lambda_s\partial_{\boldsymbol{n}_s}\delta T_s\,d\Gamma \\
&\quad + \int_\Gamma \gamma_T\delta T_s\,d\Gamma + \int_\Gamma \gamma_S\lambda_s\partial_{\boldsymbol{n}_s}T_s\,d\Gamma, \\
\delta_{T_l}\mathcal{L} &= \int_\Gamma \zeta'_{T_s}\delta T_s\,d\Gamma + \int_{\Omega_l}\left(s_l\delta T_l - \lambda_l\nabla^2\delta T_l\right)\omega_{T_l}\,d\Omega + \int_{\partial\Omega\cap\overline{\Omega_l}}\beta_{T_l}\lambda_l\partial_{\boldsymbol{n}_l}\delta T_l\,d\Gamma \\
&\quad - \int_\Gamma \gamma_T\delta T_l\,d\Gamma + \int_\Gamma \gamma_S\lambda_l\partial_{\boldsymbol{n}_l}T_l\,d\Gamma, \\
\delta_{v_{\boldsymbol{n}}}\mathcal{L} &= \int_\Gamma\left(\zeta'_{v_{\boldsymbol{n}}} - (1-k_1)C_1\gamma_v - \sum_{J=2}^{N}(1-k_J)C_J\gamma_{C_J} - L\gamma_S\right)\delta v_{\boldsymbol{n}}\,d\Gamma,
\end{aligned}
$$

where ζ'_{T_ν}, ζ'_{C_J} and $\zeta'_{v_{\boldsymbol{n}}}$ denote classical partial derivatives of function ζ with respect to T_ν, C_J and $v_{\boldsymbol{n}}$, correspondingly. Using the Green's second identity the first four expressions can be transformed into:

$$
\begin{aligned}
\delta_{C_1}\mathcal{L} = &\int_{\Omega_l} \left(a\omega_{C_1} - D_1\nabla^2\omega_{C_1}\right)\delta C_1\, d\Omega \\
&+ \int_{\partial\Omega\cap\overline{\Omega_l}} \left(\beta_{C_1} - \omega_{C_1}\right) D_1\partial_{\boldsymbol{n}_l}\delta C_1\, d\Gamma + \int_{\partial\Omega\cap\overline{\Omega_l}} D_1\partial_{\boldsymbol{n}_l}\omega_{C_1}\delta C_1\, d\Gamma \\
&+ \int_{\Gamma} \left(\gamma_{C_1} - (1-k_1)v_{\boldsymbol{n}}\gamma_v + D_1\partial_{\boldsymbol{n}_l}\omega_{C_1} + \zeta'_{C_1}\right)\delta C_1\, d\Gamma \\
&+ \int_{\Gamma} \left(\gamma_v - \omega_{C_1}\right) D_1\partial_{\boldsymbol{n}_l}\delta C_1\, d\Gamma, \\
\delta_{C_J}\mathcal{L} = &\int_{\Omega_l} \left(a\omega_{C_J} - D_J\nabla^2\omega_{C_J}\right)\delta C_J\, d\Omega \\
&+ \int_{\partial\Omega\cap\overline{\Omega_l}} \left(\beta_{C_J} - \omega_{C_J}\right) D_J\partial_{\boldsymbol{n}_l}\delta C_J\, d\Gamma + \int_{\partial\Omega\cap\overline{\Omega_l}} D_J\partial_{\boldsymbol{n}_l}\omega_{C_J}\delta C_J\, d\Gamma \\
&+ \int_{\Gamma} \left(D_J\partial_{\boldsymbol{n}_l}\omega_{C_J} - (1-k_J)v_{\boldsymbol{n}}\gamma_{C_J} + \zeta'_{C_J}\right)\delta C_J\, d\Gamma \\
&+ \int_{\Gamma} \left(\gamma_{C_J} - \omega_{C_J}\right) D_J\partial_{\boldsymbol{n}_l}\delta C_J\, d\Gamma, \quad i = 2, \ldots, N, \\
\delta_{T_s}\mathcal{L} = &\int_{\Omega_s} \left(s_s\omega_{T_s} - \lambda_s\nabla^2\omega_{T_s}\right)\delta T_s\, d\Omega \\
&+ \int_{\partial\Omega\cap\overline{\Omega_s}} \left(\beta_{T_s} - \omega_{T_s}\right)\lambda_s\partial_{\boldsymbol{n}_s}\delta T_s\, d\Gamma + \int_{\partial\Omega\cap\overline{\Omega_s}} \lambda_s\partial_{\boldsymbol{n}_s}\omega_{T_s}\delta T_s\, d\Gamma \\
&+ \int_{\Gamma} \left(\gamma_T + \lambda_s\partial_{\boldsymbol{n}_s}\omega_{T_s} + \zeta'_{T_s}\right)\delta T_s\, d\Gamma + \int_{\Gamma} \left(\gamma_S - \omega_{T_s}\right)\lambda_s\partial_{\boldsymbol{n}_s}T_s\, d\Gamma, \\
\delta_{T_l}\mathcal{L} = &\int_{\Omega_l} \left(s_l\omega_{T_l} - \lambda_l\nabla^2\omega_{T_l}\right)\delta T_l\, d\Omega \\
&+ \int_{\partial\Omega\cap\overline{\Omega_l}} \left(\beta_{T_l} - \omega_{T_l}\right)\lambda_l\partial_{\boldsymbol{n}_l}\delta T_l\, d\Gamma + \int_{\partial\Omega\cap\overline{\Omega_l}} \lambda_l\partial_{\boldsymbol{n}_l}\omega_{T_l}\delta T_l\, d\Gamma \\
&+ \int_{\Gamma} \left(-\gamma_T + \lambda_l\partial_{\boldsymbol{n}_l}\omega_{T_l} + \zeta'_{T_l}\right)\delta T_l\, d\Gamma + \int_{\Gamma} \left(\gamma_S - \omega_{T_l}\right)\lambda_l\partial_{\boldsymbol{n}_l}T_l\, d\Gamma,
\end{aligned}
$$

It is easy to see now that conditions from the right column of (4.42) lead to the following equations for Lagrange multipliers:

$$
\begin{cases}
\left(a - D_1\nabla^2\right)\omega_{C_1} = 0 & \text{in } \Omega_l \\
\omega_{C_1} = \gamma_v & \text{on } \Gamma \\
D_1\partial_{\boldsymbol{n}_l}\omega_{C_1} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_l} \\
\gamma_{C_1} = (1-k_1)v_{\boldsymbol{n}}\gamma_v - \zeta'_{C_1} - D_1\partial_{\boldsymbol{n}_l}\omega_{C_1} & \text{on } \Gamma \\
\beta_{C_1} = \omega_{C_1} & \text{on } \partial\Omega \cap \overline{\Omega_l}
\end{cases}
$$

$$
\begin{cases}
\left(a - D_J\nabla^2\right)\omega_{C_J} = 0 & \text{in } \Omega_l \\
D_J\partial_{\boldsymbol{n}_l}\omega_{C_J} - (1-k_J)v_{\boldsymbol{n}}\gamma_{C_J} = -\zeta'_{C_J} & \text{on } \Gamma \\
D_J\partial_{\boldsymbol{n}_l}\omega_{C_J} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_l} \\
\gamma_{C_J} = \omega_{C_J} & \text{on } \Gamma \\
\beta_{C_J} = \omega_{C_J} & \text{on } \partial\Omega \cap \overline{\Omega_l}
\end{cases}
$$

$$
\begin{cases}
\left(s_s - \lambda_s\nabla^2\right)\omega_{T_s} = 0 & \text{in } \Omega_s \\
\omega_{T_s} = \gamma_S & \text{on } \Gamma \\
\lambda_s\partial_{\boldsymbol{n}_s}\omega_{T_s} = -\gamma_T - \zeta'_{T_s} & \text{on } \Gamma \\
\lambda_s\partial_{\boldsymbol{n}_s}\omega_{T_s} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_l} \\
\beta_{T_s} = \omega_{T_s} & \text{on } \partial\Omega \cap \overline{\Omega_s}
\end{cases}
$$

$$
\begin{cases}
\left(s_l - \lambda_l\nabla^2\right)\omega_{T_l} = 0 & \text{in } \Omega_l \\
\omega_{T_l} = \gamma_S & \text{on } \Gamma \\
\lambda_l\partial_{\boldsymbol{n}_l}\omega_{T_l} = \gamma_T - \zeta'_{T_l} & \text{on } \Gamma \\
\lambda_l\partial_{\boldsymbol{n}_l}\omega_{T_l} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_l} \\
\beta_{T_l} = \omega_{T_l} & \text{on } \partial\Omega \cap \overline{\Omega_l}
\end{cases}
\tag{4.43}
$$

$$
\gamma_v = \frac{1}{(1-k_1)C_1}\left(\zeta'_{v_{\boldsymbol{n}}} - \sum_{J=2}^{N}(1-k_J)C_J\gamma_{C_J} - L\gamma_S\right) \quad \text{on } \Gamma
$$

which after several eliminations and rearrangements can be expressed as:

$$\begin{cases} \left(s_\nu - \lambda_\nu \nabla^2\right) \omega_{T_\nu} = 0 & \text{in } \Omega_\nu, \quad \nu = s, l \\ [\omega_T] = 0 & \text{on } \Gamma \\ [\lambda \partial_{\boldsymbol{n}} \omega_T] = -\left(\zeta'_{T_l} + \zeta'_{T_s}\right) & \text{on } \Gamma \\ \lambda_\nu \partial_{\boldsymbol{n}_\nu} \omega_{T_\nu} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_\nu}, \quad \nu = s, l \end{cases}$$

$$\begin{cases} \left(a - D_J \nabla^2\right) \omega_{C_J} = 0 & \text{in } \Omega_l \\ D_J \partial_{\boldsymbol{n}_l} \omega_{C_J} - (1 - k_J) v_{\boldsymbol{n}} \omega_{C_J} = -\zeta'_{C_J} & \text{on } \Gamma \\ D_J \partial_{\boldsymbol{n}_l} \omega_{C_J} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_l} \end{cases}$$

$$\gamma_v = \frac{1}{(1 - k_1) C_1^*} \left(\zeta'_{v_{\boldsymbol{n}}} - \sum_{J=2}^{N} (1 - k_J) \omega_{C_J} C_J - L \omega_{T_l} \right) \quad \text{on } \Gamma$$

$$\begin{cases} \left(a - D_1 \nabla^2\right) \omega_{C_1} = 0 & \text{in } \Omega_l \\ \omega_{C_1} = \gamma_v & \text{on } \Gamma \\ D_1 \partial_{\boldsymbol{n}_l} \omega_{C_1} = 0 & \text{on } \partial\Omega \cap \overline{\Omega_l} \end{cases}$$

Finally, by taking variation of the Lagrangian with respect to C_1^* we obtain the full differential of the cost functional:

$$d\mathcal{F} = \delta_{C_1^*} \mathcal{L} = -\int_\Gamma \gamma_{C_1} \delta C_1^* \, d\Gamma,$$

or, after taking into account equations (4.43):

$$d\mathcal{F} = \delta_{C_1^*} \mathcal{L} = \int_\Gamma \left(\zeta'_{C_1} + D_1 \partial_{\boldsymbol{n}_l} \omega_{C_1} - (1 - k_1) v_{\boldsymbol{n}} \omega_{C_1} \right) \delta C_1^* \, d\Gamma,$$

thus, the functional derivative of $\mathcal{F}$ with respect to C_1^* is

$$\frac{\delta \mathcal{F}}{\delta C_1^*} = \zeta'_{C_1} + D_1 \partial_{\boldsymbol{n}_l} \omega_{C_1} - (1 - k_1) v_{\boldsymbol{n}} \omega_{C_1}.$$

Note that for the functional considered in 4.3.2:

$$
\begin{aligned}
\zeta'_{T_l} &= \delta(\boldsymbol{r} - \boldsymbol{r}_0),\\
\zeta'_{T_s} &= 0,\\
\zeta'_{C_J} &= -\frac{\partial \Delta T_C}{\partial C_J}\delta(\boldsymbol{r} - \boldsymbol{r}_0),\\
\zeta'_{v_{\boldsymbol{n}}} &= -\epsilon_v \delta(\boldsymbol{r} - \boldsymbol{r}_0).
\end{aligned}
$$

4.B Directional derivative with respect to δC_1^*

The expressions derived in 4.A predict the change in a functional in response to any perturbation of C_1^*, however, they require solution of an adjoint system of PDEs for every point on the boundary, which is hardly achievable in practice. Instead, sometimes it is necessary to only know the derivative of a functional along a given perturbation in C_1^*, e.g. as in the approximate Newton described in 4.3.2.

Let us again consider a generic functional $\mathcal{F}$ defined in (4.40). Let us consider system of equations (4.20)-(4.23) for C_1^* and $C_1^* + \varepsilon\delta C_1^*$, where $\varepsilon \ll 1$. It is easy to show that solutions of (4.20)-(4.23) in these two cases are related to each other as:

$$
\begin{aligned}
C_J|_{C_1^*+\varepsilon\delta C_1^*} &= C_J|_{C_1^*} + \varepsilon\Lambda_{C_J} + \mathcal{O}\left(\varepsilon^2\right), \quad J \in [1, N]\\
T_\nu|_{C_1^*+\varepsilon\delta C_1^*} &= T_\nu|_{C_1^*} + \varepsilon\Lambda_{T_\nu} + \mathcal{O}\left(\varepsilon^2\right), \quad \nu = s, l\\
v_{\boldsymbol{n}}|_{C_1^*+\varepsilon\delta C_1^*} &= v_{\boldsymbol{n}}|_{C_1^*} + \varepsilon\Lambda_{v_{\boldsymbol{n}}} + \mathcal{O}\left(\varepsilon^2\right),
\end{aligned}
$$

where $\{\Lambda_{C_J}\}_{J=1}^N$, $\{\Lambda_{T_\nu}\}_{\nu=s,l}$, and $\Lambda_{v_{\boldsymbol{n}}}$ satisfy the following adjoint system of equations:

$$
\begin{cases}
\left(a - D_1\nabla^2\right)\Lambda_{C_1} = 0, & \text{in } \Omega_l,\\
\Lambda_{C_1} = \delta C_1^*, & \text{on } \Gamma,\\
D_1\partial_{\boldsymbol{n}_l}\Lambda_{C_1} = 0, & \text{on } \partial\Omega \cap \Omega_l.
\end{cases}
$$

$$\Lambda_{v_{\boldsymbol{n}}} = \frac{1}{(1-k_1)C_1}\left(D_1\partial_{\boldsymbol{n}_l}\Lambda_{C_1} - v_{\boldsymbol{n}}(1-k_1)\Lambda_{C_1}\right)$$

$$\begin{cases} \left(a - D_J\nabla^2\right)\Lambda_{C_J} = 0, & \text{in } \Omega_l, \\ D_J\partial_{\boldsymbol{n}_l}\Lambda_{C_J} - (1-k_J)v_{\boldsymbol{n}}\Lambda_{C_J} = (1-k_J)\Lambda_{v_{\boldsymbol{n}}}C_J, & \text{on } \Gamma, \\ D_J\partial_{\boldsymbol{n}_l}\Lambda_{C_J} = 0, & \text{on } \partial\Omega\cap\Omega_l. \end{cases}$$

$$\begin{cases} \left(s_\nu - \lambda_\nu\nabla^2\right)\Lambda_{T_\nu} = 0 & \text{in } \Omega_\nu, \quad \nu = s, l \\ [\Lambda_T] = 0 & \text{on } \Gamma \\ [\lambda\partial_{\boldsymbol{n}}\Lambda_T] = L_f\Lambda_{v_{\boldsymbol{n}}} & \text{on } \Gamma \\ \lambda_\nu\partial_{\boldsymbol{n}_\nu}\Lambda_{T_\nu} = 0 & \text{on } \partial\Omega\cap\overline{\Omega_\nu}, \quad \nu = s, l \end{cases}$$

Using this result the derivative of $\mathcal{F}$ in the direction δC_1^* can be computed as:

$$\begin{aligned} \int_\Gamma \frac{\delta\mathcal{F}}{\delta C_1^*}\delta C_1^*\, d\Gamma &= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left(\mathcal{F}|_{C_1^*+\varepsilon\delta C_1^*} - \mathcal{F}|_{C_1^*}\right) \\ &= \int_\Gamma\left(\sum_{\nu=s,l}\zeta'_{T_\nu}\Lambda_{T_\nu} + \sum_{J=1}^N\zeta'_{C_J}\Lambda_{C_J} + \zeta'_{v_{\boldsymbol{n}}}\Lambda_{v_{\boldsymbol{n}}}\right)d\Gamma. \end{aligned}$$

4.C Details of linear stability analysis of iterative schemes for solving nonlinear system of PDEs

In the simple case of a planar geometry considered in section 4.3.2 the fixed-point and approximate Newton iterations can be explicitly written as:

$$C_1^{*(q+1)}(x) = C_1^{*(q)}(x) - \frac{T_l^{(q)}(x,0) - \sum_{J=1}^N m_{lJ}C_J^{(q)}(x,0) - h_G(x)}{m_{l1}}, \tag{4.44}$$

$$C_1^{*(q+1)}(x) = C_1^{*(q)}(x) - \frac{T_l^{(q)}(x,0) - \sum_{J=1}^N m_{lJ}C_J^{(q)}(x,0) - h_G(x)}{\Lambda_{T_l}(x,0) - \sum_{J=1}^N m_{lJ}\Lambda_{C_J}(x,0)}. \tag{4.45}$$

It is straightforward to show that the solution of (4.20)-(4.23) corresponding to a perturbed boundary concentration of the form

$$C_1^{*(q)} = \tilde{C}_1^* + r^q \delta_C \exp(-i\omega_x x)$$

can be found in the form:

$$\begin{aligned}
C_J^{(q)}(x,y) &= \tilde{C}_J(y) + A_J r^\nu \delta_0 \exp(-i\omega_x x)\exp(-\Omega_J y) + \mathcal{O}\left(\delta_C^2\right), \quad J \in [1,N], \\
T_\nu^{(q)}(x,y) &= \tilde{T}_\nu(y) + B_\nu r^\nu \delta_0 \exp(-i\omega_x x)\exp(-\Omega_\nu y) + \mathcal{O}\left(\delta_C^2\right), \quad \nu = s,l \\
v_{\boldsymbol{n}}^{(q)}(x) &= \tilde{v_{\boldsymbol{n}}} + E r^\nu \delta_0 \exp(-i\omega_x x) + \mathcal{O}\left(\delta_C^2\right),
\end{aligned}$$

where $\Omega_J = \Omega_J(\omega_x)$, $J \in [1,N]$, and $\Omega_\nu = \Omega_\nu(\omega_x)$, $\nu = l,s$, satisfy (4.35) and

$$\begin{aligned}
E &= \frac{D_1\Omega_1 - \tilde{v_{\boldsymbol{n}}}(1-k_1)}{(1-k_1)\tilde{C}_1(0)}, \\
A_J &= \left(\frac{\tilde{C}_J(0)}{\tilde{C}_1(0)}\right)\left(\frac{1-k_J}{1-k_1}\right)\left(\frac{D_1\Omega_1 - \tilde{v_{\boldsymbol{n}}}(1-k_1)}{D_J\Omega_J - \tilde{v_{\boldsymbol{n}}}(1-k_J)}\right), \quad J = 1,\ldots,N, \\
B_s = B_l &= \frac{L_f}{(1-k_1)\tilde{C}_1(0)}\frac{D_1\Omega_1 - \tilde{v_{\boldsymbol{n}}}(1-k_1)}{\lambda_s\Omega_s + \lambda_l\Omega_l}.
\end{aligned}$$

The adjoint system of equation (4.29)-(4.32) in this case has the solution:

$$
\begin{aligned}
\Lambda_{C_1}^{(q)}(x,y) &= \exp(-\Omega_1(0)y) + \mathcal{O}\left(\delta_C\right), \\
\Lambda_{v}^{(q)}(x) &= \frac{D_1\Omega_1(0) - (1-k_1)\tilde{v_{\boldsymbol{n}}}}{(1-k_1)\tilde{C}_1(0)} + \mathcal{O}\left(\delta_C\right), \\
\Lambda_{C_J}^{(q)}(x,y) &= \left(\frac{1-k_J}{1-k_1}\right)\left(\frac{\tilde{C}_J(0)}{\tilde{C}_1(0)}\right)\left(\frac{D_1\Omega_1(0) - (1-k_1)\tilde{v_{\boldsymbol{n}}}}{D_{lJ}\Omega_J(0) - (1-k_J)\tilde{v_{\boldsymbol{n}}}}\right)\exp(-\Omega_J(0)y) \\
&\quad + \mathcal{O}\left(\delta_C\right), \quad J = 2,\ldots,N \\
\Lambda_{T_l} &= \frac{L_f}{(1-k_1)\tilde{C}_1(0)}\frac{D_1\Omega_1(0) - \tilde{v_{\boldsymbol{n}}}(1-k_1)}{\lambda_s\Omega_s(0) + \lambda_l\Omega_l(0)}\exp(-\Omega_l(0)y) + \mathcal{O}\left(\delta_C\right), \\
\Lambda_{T_s} &= \frac{L_f}{(1-k_1)\tilde{C}_1(0)}\frac{D_1\Omega_1(0) - \tilde{v_{\boldsymbol{n}}}(1-k_1)}{\lambda_s\Omega_s(0) + \lambda_l\Omega_l(0)}\exp(\Omega_s(0)y) + \mathcal{O}\left(\delta_C\right).
\end{aligned}
$$

Note that it is not necessary to obtain linear correction while solving the adjoint system of equations because

$$
T_l^{(q)}(x,0) - \sum_{J=1}^{N} m_{lJ}C_J^{(q)}(x,0) - h_G(x) = \mathcal{O}\left(\delta_C\right)
$$

Substitution of the above expressions into (4.44) produces amplification factors (4.33) and (4.34)

4.D Removing extremely underresolved regions

In order to ensure the robustness of numerical simulations we use the following two-pass strategy that regularize underresolved geometries.

During the first pass, narrow gaps of liquid material of the size less than two grid spacings are "bridged". Specifically, we compute an auxiliary level-set function ϕ_{aux} as the signed distance to the $\phi_n = -\Delta x$ isocontour of the original level-set function ϕ_n and

shift it back by Δx. In the case when the geometry is sufficiently resolved, ϕ_{aux} and ϕ_n have coinciding signs on all grid nodes and very close in values. In the case when narrow regions of liquid material of width less than $2\Delta x$ are present ϕ_{aux} and ϕ_n will have different signs on grid nodes in such regions (more precisely, $\phi_{\text{aux}} > 0$ and $\phi_n < 0$). Substituting values of ϕ_n with values of ϕ_{aux} effectively eliminates such too narrow regions. Note that ϕ_n remains unchanged whenever the front's geometry is sufficiently resolved.

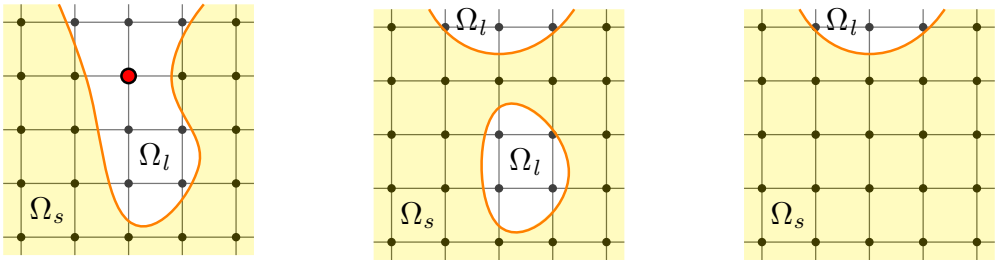


Figure 4.17: Illustration of the procedure used for removing extremely under-resolved regions of liquid: (a) identification of grid nodes at which locally distinct parts of Ω_s are separated just by a single node; (b) "bridging" narrow liquid gaps at such grid nodes; (c) identification and "solidification" of liquid pools left behind by the previous step.

During the second pass, isolated pools of liquid created as a result of such "bridging" procedure are identified and "solidified" as well (if any). The identification of isolated pools on distributed computational grids is done using the parallel "island counting" algorithm described in [107].

We have observed that such a procedure ensures an excellent robustness of the computational scheme across all crystal growth regimes. It is especially useful for simulating the cellular regime and its transition to the planar one.

4.E Similarity solution for the solidifying infinite cylinder due to a heat sink

Let us consider the axisymmetric solidification of an infinite cylinder from a line heat sink of strength Q located at the cylinder's center into an infinite liquid alloy of composition $\{C_J^\infty\}_{J=1}^N$ and temperature T^∞. In this case spatial distributions of temperature $T_\nu = T_\nu(t,r)$, $\nu = s,l$, and concentrations $C_{lJ} = C_{lJ}(t,r)$, $J \in [1,N]$, are only functions of time t and distance from the cylinder's center r, which without loss of generality can be assumed at $r = 0$. In addition to assumption made in section 4.2, further assume that the constitutional undercooling has a linear dependence on the composition ($T_{liq} = T_m + \sum_{J=1}^N m_{lJ} C_{lJ}$), the kinetic and curvature undercoolings are negligible ($\epsilon_c = \epsilon_v = 0$) and partition coefficients $\{k_i\}_{J=1}^N$ are constant. Denote the cylinder's radius as $R(t)$. Mathematically such a problem can be formulated as:

Governing equations:

$$
\begin{aligned}
&\textit{Heat transport:} && \rho_s c_{p_s} \partial_t T_s - \lambda_s \nabla^2 T_s = 0 \quad \text{for } 0 < r < R(t), \\
& && \rho_l c_{p_l} \partial_t T_l - \lambda_l \nabla^2 T_l = 0 \quad \text{for } R(t) < r < \infty, \\
&\textit{Species transport:} && \partial_t C_{lJ} - D_{lJ} \nabla^2 C_{lJ} = 0 \quad \text{for } R(t) < r < \infty, \\
& && \qquad J \in [1, N] \\
&\textbf{Interface conditions}: \\
&\textit{Temperature continuity:} && [T] = 0, \\
&\textit{Stefan condition:} && [\lambda \partial_r T] = v_{\boldsymbol{n}} L, \\
&\textit{Gibbs-Thomson:} && T_l = T_m + \sum_{J=1}^{N} m_{lJ} C_J, \\
&\textit{Solute-rejection:} && D_{lJ} \partial_r C_J = -(1 - k_J) v_{\boldsymbol{n}} C_J, \quad J \in [1, N], \\
&\textbf{Boundary conditions}: \\
&\textit{Line source:} && \lim_{r \to 0} (2\pi r \lambda_s \partial_r T_s) = Q, \\
&\textit{Temperature:} && \lim_{r \to \infty} T_l = T^{\infty}, \\
&\textit{Composition:} && \lim_{r \to \infty} C_{lJ} = C_J^{\infty}, \quad J \in [1, N], \\
&\textbf{Initial conditions}: \\
&\textit{Front location:} && R|_{t=0} = 0, \\
&\textit{Temperature:} && T_l|_{t=0} = T^{\infty}, \\
&\textit{Composition:} && C_{lJ}|_{t=0} = C_J^{\infty}, \quad J \in [1, N],
\end{aligned}
\tag{4.46}
$$

where the same notation as in section 4.2 is used. Note that in the axisymmetric case the Laplace operator has the form:

$$\nabla^2 = \frac{1}{r} \partial_r \left(r \partial_r \right).$$

It can be shown that similarly to other Stefan-type problems the considered problem admits a similarity solution of the form:

$$
\begin{aligned}
R(t) &= 2\sqrt{\theta t}, \\
v_{\boldsymbol{n}}(t) &= \sqrt{\frac{\theta}{t}}, \\
T_s(t,r) &= A^s + B^s E_1\left(\frac{r^2}{4\alpha_s t}\right), \\
T_l(t,r) &= A^l + B^l E_1\left(\frac{r^2}{4\alpha_l t}\right), \\
C_{lJ}(t,r) &= A_J + B_J E_1\left(\frac{r^2}{4D_{lJ} t}\right), \quad J \in [1,N],
\end{aligned}
\tag{4.47}
$$

where $\alpha_s = \frac{\lambda_s}{\rho_s c_{p_s}}$, $\alpha_l = \frac{\lambda_l}{\rho_l c_{p_l}}$ are thermal diffusivities and E_1 denotes the exponential integral:

$$
E_1(z) = \int_x^\infty \frac{e^{-s}}{s} ds, \quad z > 0.
$$

Indeed, using direct substitution one can show that (4.47) is the solution to (4.46) provided the values of constants A^s, B^s, A^l, B^l, A_J, B_J, $J \in [1,N]$, are given by:

$$
\begin{aligned}
A^s &= T^*(\theta) + \frac{Q}{4\pi\lambda_s} E_1\left(\frac{\theta}{\alpha_s}\right), \quad & B^s &= -\frac{Q}{4\pi\lambda_s}, \\
A^l &= T^\infty, & B^l &= \frac{T^*(\theta) - T^\infty}{E_1\left(\frac{\theta}{\alpha_l}\right)}, \\
A_J &= C_J^\infty, \quad J \in [1,N], & B_J &= \frac{C_J^*(\theta) - C_J^\infty}{E_1\left(\frac{\theta}{D_{lJ}}\right)}, \quad J \in [1,N],
\end{aligned}
$$

and quantity θ, called the growth constant, satisfies the nonlinear algebraic equation:

$$
T^*(\theta) = T_m + \sum_{J=1}^{N} m_{lJ} C_J^*(\theta)
$$

where

$$C_J^*(\theta) = \frac{C_J^\infty}{1-(1-k_J)\frac{\theta}{D_{lJ}}\exp\left(\frac{\theta}{D_{lJ}}\right)E_1\left(\frac{\theta}{D_{lJ}}\right)}, \quad J \in [1,N]$$

$$T^*(\theta) = T^\infty + \frac{\theta}{\alpha_l}\exp\left(\frac{\theta}{\alpha_l}\right)E_1\left(\frac{\theta}{\alpha_l}\right)\left(\frac{L}{\rho_l c_{p_l}} - \frac{Q}{4\pi\lambda_s}\frac{\rho_s c_{p_s}}{\rho_l c_{p_l}}\frac{1}{\frac{\theta}{\alpha_s}\exp\left(\frac{\theta}{\alpha_s}\right)}\right).$$

Note that:

$$\lim_{\theta\to 0} C_J^* = C_J^\infty, \quad \lim_{\theta\to\infty} C_J^* = \frac{C_J^\infty}{k_J}, \quad J \in [1,N],$$
$$\lim_{\theta\to 0} T^* = -\infty, \quad \lim_{\theta\to\infty} T^* = T^\infty + \frac{L}{\rho_l c_{p_l}}.$$

Thus, for $Q > 0$ and $0 < k_J < 1$ such a solution exists as long as:

$$T^\infty > T_m + \sum_{i=1}^{N} m_{lJ}\frac{C_J^\infty}{k_J} - \frac{L}{\rho_l c_{p_l}}.$$

Existing of such an analytical solution allows creating of a non-trivial benchmark test for a multidimensional solidification code. Specifically, in this work we consider an annular region with internal and external radii R_{in} and R_{out}. We start with initial conditions given by the analytical solution at some initial time t_0, such that $R_{\text{in}} < R(t_0) < R_{\text{out}}$, and impose time-dependent boundary conditions (Dirichlet or Neumann) on the inner and outer boundaries of the region based on the analytical solution.

Interesting features of this similarity solution are that the values of temperature and concentration at the solidification front are constant throughout the entire solidification process, that is:

$$T_l = T_s(t, R(t)) = T^* \quad \text{and} \quad C_J^l(t, R(t)) = C_J^*,\ J \in [1,N],\ \forall\, t > 0,$$

and that the ratio of compositional and thermal gradients at the solidification front is constant as well

$$M = \left.\frac{\sum_{J=1}^{N} m_{lJ}\partial_r C_J^l}{\partial_r T_l}\right|_{r=R(t)} = \sum_{J=1}^{N} m_{lJ}\frac{C_J^\infty - C_J^*}{T^\infty - T^\star}\frac{\exp\left(\frac{\theta}{\alpha_l}\right)E_1\left(\frac{\theta}{\alpha_l}\right)}{\exp\left(\frac{\theta}{D_{lJ}}\right)E_1\left(\frac{\theta}{D_{lJ}}\right)}.$$

Recall that the solidification front is expected to be stable for $M < 1$ and unstable for $M > 1$ (compositional undercooling ahead of the front). For purposes of verification of multidimensional solidification codes it is desired to consider stable processes, otherwise due to unavoidable numerical errors the numerical solution would quickly diverge from the symmetric configuration predicted by the analytical solution and a comparison would not be possible. For this reason it is more convenient to select an analytical solution based on the value of the gradients' ratio $M = M_0$ instead of imposing the strength of heat sink Q and the temperature value at infinity T^∞. In addition, to more easily and independently select the characteristic front velocity we impose a specific value of the front velocity $v_{\boldsymbol{n}}(t_0) = v_0$ at the beginning of simulation t_0 when the seed radius is equal to a given $R(t_0) = R_0$. Thus, alternatively to (4.46), boundary conditions can be formulated as:

$$\begin{aligned} M &= M_0, \\ v_{\boldsymbol{n}}(t_0) &= v_0, \text{ where } t_0 \text{ is such that } R(t_0) = R_0 \\ \lim_{r\to\infty} C_J(t,r) &= C_J^\infty, \quad J \in [1,N]\,. \end{aligned}$$

In this case the integration constants in the analytical solution are given by:

$$\theta = \frac{1}{2} v_0 R_0$$

$$A_J = C_J^{\infty}, \quad J \in [1, N], \qquad B_J = \frac{C_J^*(\theta) - C_J^{\infty}}{E_1\left(\frac{\theta}{D_{lJ}}\right)}, \quad J \in [1, N],$$

$$A^l = T^*(\theta) - B^l E_1\left(\frac{\theta}{\alpha_l}\right), \quad B^l = \frac{1}{M_0} \sum_{J=1}^{N} m_{lJ} B_J \frac{\exp\left(\frac{\theta}{\alpha_l}\right)}{\exp\left(\frac{\theta}{D_{lJ}}\right)},$$

$$A^s = T^*(\theta) - B^s E_1\left(\frac{\theta}{\alpha_s}\right), \quad B^s = B^l \frac{\rho_l c_{p_l}}{\rho_s c_{p_s}} \frac{\frac{\theta}{\alpha_s} \exp\left(\frac{\theta}{\alpha_s}\right)}{\frac{\theta}{\alpha_l} \exp\left(\frac{\theta}{\alpha_l}\right)} - \frac{L}{\rho_s c_{p_s}} \frac{\theta}{\alpha_s} \exp\left(\frac{\theta}{\alpha_s}\right),$$

where

$$C_J^*(\theta) = \frac{C_J^{\infty}}{1 - (1 - k_J) \frac{\theta}{D_{lJ}} \exp\left(\frac{\theta}{D_{lJ}}\right) E_1\left(\frac{\theta}{D_{lJ}}\right)}, \quad J \in [1, N],$$

$$T^*(\theta) = T_m + \sum_{J=1}^{N} m_{lJ} C_J^*(\theta).$$

Note that the initial moment of time for setting up numerical simulations is given by:

$$t_0 = \frac{1}{2} \frac{R_0}{v_0}.$$

It is also easy to extend the above similarity solution to the case of a nonlinear liquidus surface $T_{liq} = T_{liq}(C_{l1}^*, \ldots, C_{lN}^*)$ and nonconstant partition coefficients $\{k_J = k_J(C_{l1}^*, \ldots, C_{lN}^*)\}_{J=1}^N$. In such a case the interfacial concentration are found by solving the nonlinear algebraic system of equations (we found that the fixed-point iteration suffices):

$$C_J^*(\theta) = \frac{C_J^{\infty}}{1 - (1 - k_J(C_{l1}^*, \ldots, C_{lN}^*)) \frac{\theta}{D_{lJ}} \exp\left(\frac{\theta}{D_{lJ}}\right) E_1\left(\frac{\theta}{D_{lJ}}\right)}, \quad J \in [1, N],$$

and the interfacial temperature is simply given by:

$$T^*(\theta) = T_{liq}(C^*_{l1}, \ldots, C^*_{lN}).$$

Chapter 5

Moving Boundary Problems in Physics of Block Copolymer Materials

5.1 Introduction

Block copolymers are materials made-up of polymer chains that consist of blocks of dissimilar chemical species. For example, figure 5.1 schematically illustrates the structure of a diblock copolymer melt (e.g. polystyrene-b-poly(methyl methacrylate)). The inhomogeneity of polymer chains leads to a rich self-assembling behavior of such materials at the nanoscale. Even in the simplest case of symmetric diblock copolymers, a number of morphologies can form: lamellar, hexagonal cylindrical, cubic spherical, bicontinuous

gyroid. The formation and stability of a particular phase depend on the architecture of polymer chain and interaction properties between chemical species and are governed by an intricate balance between entropic and enthalpic contributions to the free energy of the system that promote disodered and ordered configurations, correspondingly. Given such a large variety of self-assembling nanomorphologies BCP materials have attracted a great attention as a promising avenue for a number of applications such as the fabrication of nanoelectronics [74], the nanophotonics [155], the targeted drug delivery [134], and others. From the theoretical point of view, probably the most common approach to study the self-assembly of BCPs is the Self-Consistent Field Theory (SCFT) [98, 47]. This framework is based on applying the Hubbard-Stratonovich transformation to convert the particle-based representation of the polymer material into a field-based one and has been successfully applied to qualitatively and quantitatively predict the self-assembly of BCPs in a countless number of applications [47].

Figure 5.1: A schematic illustration of an AB diblock copolymer chain.

In its standard form, the SCFT is suitable for investigating periodic (bulk) morphologies of self-assembling BCPs or in static confinements. However, in certain situations, such as the presence of polymer-air interface or immersed nanoparticles, the confining boundaries are free to move and in fact must be considered as part of the solution.

The importance of simulating free surface block copolymer melts comes from the fact

that it is not uncommon in practical applications for the polymer material to be exposed to a gaseous or liquid phase. Besides the meniscus near confining walls, this can also lead to the formation of unique free surface features of block copolymers such as holes, islands, and terraces [35, 97]. The simulation of block copolymer free surfaces is challenging due to the two-way coupling between the free surface and the internal polymer morphology. Within the SCFT framework, free polymer surfaces are typically modeled by introducing into the system an additional chemical species that play the role of the air, see, for example, [85, 154, 25]. While being simple and straightforward to implement, this approach is not free of caveats. Among them is the fact that only relatively low interaction strengths between dissimilar species can be used in the SCFT for numerical calculations to remain stable. This leads to the inability to impose physically high incompatibility between the polymer material and the surrounding air as well as to a smeared-out polymer-air interface. In [123], a conceptually different framework has been proposed, where the polymer-air interface is assumed to be sharp and where its governing equation is derived based on the PDE-constrained shape sensitivity analysis. The approach presented in in [123] is, however, limited to neutral polymer-air interfaces due to the well-known inconsistency within the SCFT to model selectively attractive/repulsive boundaries. Moreover, [123] does not consider situations when polymer-air interfaces can terminate at solid walls, hence forming a triple junction line. In order to model such situations, an equation governing the value of contact angle is necessary.

BCP nanocomposites is a promising avenue for the creation of functional materi-

als and advanced polymer self-assembly applications [20]. Similar to the case of free polymer surfaces, the theoretical description of these systems is complicated by two-way couplings: on the one hand, the self-assembling polymer structures guide the placement of nanoparticles but, on the other hand, the presence of nanoparticles influence the resulting polymer morphologies. The problem of co-assembly of BCPs and nanoparticles in the SCFT framework has been approached from explicit and implicit perspectives. In the former case, each nanoparticle in the system is described explicitly while in the latter case, the distribution of nanoparticles is described in terms of density fields. Each framework has its own advantages: the first methodology allows for a straightforward incorporation of such features as excluded volume effects, arbitrary shapes of particles, complex particle-particle interactions; the second methodology is more efficient at obtaining statistical information about particles configurations. The explicit approach was employed in the hybrid particle-field method proposed in [148], in which nanoparticles were modeled by the so-called cavity function that represents a gradual transition between polymer and particle's material and the relaxation of nanoparticles towards the equilibrium state is accomplished using an approximate derivatives of the system's energy with respect to the particles' positions. Among implicit approaches are the hybrid SCFT-DFT method proposed in [160] and the field-theoretic method described in [87].

Another problem, which is of a different nature but can also be categorized as a moving boundary problem, is the inverse design problem for the Directed Self-Assembly (DSA) of BCPs used for lithography applications. Here, one is interested in finding con-

fining domains that drive the self-assembly of BCPs into desired patterns. Specifically, in this work we focus on the design of confining masks for placement of cylindrical domains. One of the approaches for this challenging inverse problem presented in [89] is based on parameterizing the confinement geometry with several degrees of freedom and approximating the derivatives of the cost functional with respect to these degrees of freedom either using a brute-force finite difference approach or a linearization of SCFT equations. In [125], a non-parametric description of the confining mask was employed leveraging the Level-Set Method, thus eliminating the constraint on the degrees of freedom needed. However, only approximate deformation velocities based on physical intuition were used in [125].

In this chapter, we present adjoint-state approaches for the solution of the three moving boundary problems described above by considering them in a unified PDE-constrained shape optimization framework: in the simulation of the self-assembly of free surface BCPs and the co-assembly of BCP nanocomposites, one needs to minimize the free energy of the system, while in solving the inverse design problem for the directed self-assembly, one needs to minimize the discrepancy between an actual polymer configuration and a desired one. Similar to [123], in all cases the moving boundaries are treated in a sharp fashion using the Level-Set Method and the full derivatives of the minimization quantity (the system's energy or the cost functional) with respect to these moving boundaries are analytically derived based on the theory of PDE-constrained shape sensitivity analysis. To address the singular behavior of the pressure field near selectively attractive/repulsive

boundaries a novel consistent approach for imposing surface energies is proposed.

The numerical implementation closely follows that of [124]. Specifically, adaptive Cartesian quadtree grids are used for spatial discretizations, the irregular interfaces are described using the Level Set Method, and the modified diffusion equations at the core of SCFT are solved using the second-order accurate finite-volume discretizations from [18].

The rest of this chapter is organized as follows. Section 5.2 presents a consistent approach for imposing arbitrary surface energies for incompressible BCP melts in the SCFT framework. In sections 5.3, 5.4, and 5.5, we provide detailed calculations of shape sensitivities for cases of an evolving free surface, a moving particle, and a confining mask, respectively. In section 5.7, we present numerical examples demonstrating the capabilities of the proposed methods. Finally, section 5.8 draws conclusions and discusses future research directions.

5.2 A consistent approach for imposing arbitrary surface energies in SCFT

Let us consider an incompressible melt of n continuous Gaussian AB diblock copolymer chains in a region Ω with a (possibly, piecewise) smooth boundary Γ (figure 5.2). Suppose that the total number of monomers in a polymer chain is N, the fraction of monomers A is f and the surface energy of blocks A and B on boundary Γ are γ_A and γ_B (possibly varying in space), correspondingly. Furthermore, as a common practice it is assumed

that A and B monomers have equal volumes v_0. The presented below derivation of a mean-field approximation for this system follows [47] very closely but differs in that it results in a system of equations that consistently models arbitrary surface energies.

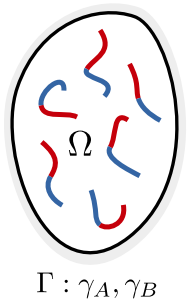


Figure 5.2: Problem geometry and notation used in section 5.2.

Polymer chains are parameterized using a contour variable $s \in [0, 1]$, such that the A-block spans from $s = 0$ to $s = f$ and the B-block spans from $s = f$ to $s = 1$. We denote the shape of the jth polymer chain as $\boldsymbol{r}_j = \boldsymbol{r}_j(s)$, $j \in [1, n]$.

The stretching energy of the n continuous Gaussian chains can be written as:

$$\beta U_0 = \sum_{j=1}^{n} \int_0^1 \frac{3}{2Nb^2} \left| \frac{d\boldsymbol{r}_j(s)}{ds} \right|^2 ds$$

where $b = b(s)$ is the statistical length of the polymer bead at location s and $\beta = \frac{1}{k_B T}$. Specifically, we assume that b is constant for each of the distinct blocks, that is:

$$b(s) = \begin{cases} b_A, & s \in [0, f] \\ b_B, & s \in [f, 1] \end{cases}$$

The interaction energy between dissimilar chemical blocks is taken into account through a Flory-Higgins effective interaction parameter χ_{AB}:

$$\beta U_1 = \frac{1}{v_0}\chi_{AB}\int_{\Omega}\hat{\rho}_A\hat{\rho}_B\,d\boldsymbol{r},$$

where the microscopic number concentrations are given by:

$$\hat{\rho}_A = v_0 N\sum_{j=1}^{n}\int_0^f \delta(\boldsymbol{r}-\boldsymbol{r}_j(s))\,ds$$

$$\hat{\rho}_B = v_0 N\sum_{j=1}^{n}\int_f^1 \delta(\boldsymbol{r}-\boldsymbol{r}_j(s))\,ds.$$

Finally, the total surface energy of the polymer melt can be expressed as:

$$\beta U_2 = \beta\int_{\Gamma}\left(\gamma_A\hat{\rho}_A+\gamma_B\hat{\rho}_B\right)\,d\boldsymbol{r} = \beta\int_{\Omega}\left(\gamma_A\hat{\rho}_A+\gamma_B\hat{\rho}_B\right)\delta_\Gamma\,d\boldsymbol{r},$$

where it is assumed that the local surface energy of the polymer material is equal to a weighted averaged of surface energies of A and B blocks, and where δ_Γ represents the surface delta function associated with Γ.

Taking into account the stretching, interaction and surface energies, the canonical partition function of the system under consideration can be written as:

$$\mathcal{Z}_C = \frac{1}{n!\left(\lambda_T^3\right)^{nN}}\prod_{j=1}^{n}\int\mathcal{D}\boldsymbol{r}_j e^{-\beta U_0-\beta U_1-\beta U_2}\delta[\hat{\rho}_A+\hat{\rho}_B-\rho_0]$$

where $\lambda_T = \dfrac{h}{\sqrt{2\pi m k_B T}}$ is the thermal length. In the above expression, the integral is taken among all possible configurations for each of the chains and the delta function enforces the incompressibility.

Using the Hubbard-Stratonovich transformation, the canonical partition function can

be expressed as:

$$\mathcal{Z}_C = \mathcal{Z}_0 \iint \exp\left(-\mathcal{H}[\mu_+, \mu_-]\right) \mathcal{D}\mu_+ \mathcal{D}\mu_-,$$

where $\mathcal{Z}_0$ denotes the ideal gas partition function, μ_+ and μ_- are fluctuating pressure-like and exchange-like potential fields, and the effective Hamiltonian $\mathcal{H}$ is given by:

$$\mathcal{H} = \frac{n}{V} \int_\Omega \left(\frac{\mu_-^2}{\chi_{AB} N} - \mu_+ \right) d\boldsymbol{r} - n \log(\mathcal{Q}).$$

The single-chain partition function $\mathcal{Q}$ can be computed from the chain propagator $q = q(s, \boldsymbol{r})$ or the complimentary chain propagator $q_c = q_c(s, \boldsymbol{r})$ as:

$$\mathcal{Q} = \frac{1}{V} \int_\Omega q(1, \boldsymbol{r})\, d\boldsymbol{r} = \frac{1}{V} \int_\Omega q_c(0, \boldsymbol{r})\, d\boldsymbol{r}, \tag{5.1}$$

which, in their turn, are solutions to the following modified diffusion equations:

$$\begin{cases} \partial_s q + \left(\mu(s) + \sigma\gamma(s)\delta_\Gamma\right) q = D(s)\nabla^2 q, & (s, \boldsymbol{r}) \in [0,1] \times \Omega, \\ \partial_{\boldsymbol{n}} q = 0, & (s, \boldsymbol{r}) \in [0,1] \times \Gamma, \\ q(0, \boldsymbol{r}) = 1, & \boldsymbol{r} \in \Omega, \end{cases} \tag{5.2}$$

and

$$\begin{cases} -\partial_s q_c + \left(\mu(s) + \sigma\gamma(s)\delta_\Gamma\right) q_c = D(s)\nabla^2 q_c, & (s, \boldsymbol{r}) \in [0,1] \times \Omega, \\ \partial_{\boldsymbol{n}} q_c = 0, & (s, \boldsymbol{r}) \in [0,1] \times \Gamma, \\ q_c(1, \boldsymbol{r}) = 1, & \boldsymbol{r} \in \Omega, \end{cases} \tag{5.3}$$

where $\sigma = \frac{N v_0}{R_g} \frac{1}{k_B T}$ and

$$D(s) = \begin{cases} D_A = b_A^2/6, & s \in [0, f], \\ D_B = b_B^2/6, & s \in [f, 1], \end{cases}$$
$$\mu(s) = \begin{cases} \mu_+ - \mu_-, & s \in [0, f], \\ \mu_+ + \mu_-, & s \in [f, 1], \end{cases}$$
$$\gamma(s) = \begin{cases} \gamma_A, & s \in [0, f], \\ \gamma_B, & s \in [f, 1]. \end{cases}$$

A mean-field approximation for this system is obtained by accounting only for instances of fields $\mu_+ = \mu_+^*$ and $\mu_- = \mu_-^*$ that have the largest contribution in $\mathcal{Z}_C$, that is:

$$\mathcal{Z}_C \approx \exp\left(-\mathcal{H}[\mu_+^*, \mu_-^*]\right),$$

where

$$\frac{\delta \mathcal{H}}{\delta \mu_+}[\mu_+^*, \mu_-^*] = 0 \quad \text{and} \quad \frac{\delta \mathcal{H}}{\delta \mu_-}[\mu_+^*, \mu_-^*] = 0. \tag{5.4}$$

This results in the following so-called SCFT equations:

$$\rho_A + \rho_B = 1, \tag{5.5}$$
$$\rho_A - \rho_B = \frac{2\mu_-}{\chi N}, \tag{5.6}$$

where concentrations of A and B blocks are given by:

$$\rho_A = \frac{1}{\mathcal{Q}} \int_0^f q q_c \, ds \quad \text{and} \quad \rho_B = \frac{1}{\mathcal{Q}} \int_f^1 q q_c \, ds. \tag{5.7}$$

Note that equations (5.2) and (5.3) involve delta-like potentials which may be diffi-

cult to deal with from the numerical point of view. Some of the previous works avoided this issue by approximating the delta-like potential by a smooth function. Such an approach, however, introduces an artificial scale over which the delta function is smoothed. Alternatively, it can be shown (see 5.B) that the diffusion equations (5.2) and (5.3) are equivalent to:

$$\begin{cases} \partial_s q + \mu(s) q = D(s) \nabla^2 q, & (s, \boldsymbol{r}) \in [0,1] \times \Omega, \\ D(s) \partial_{\boldsymbol{n}} q + \sigma \gamma(s) q = 0, & (s, \boldsymbol{r}) \in [0,1] \times \Gamma, \\ q(0, \boldsymbol{r}) = 1, & \boldsymbol{r} \in \Omega, \end{cases} \tag{5.8}$$

and

$$\begin{cases} -\partial_s q_c + \mu(s) q_c = D(s) \nabla^2 q_c, & (s, \boldsymbol{r}) \in [0,1] \times \Omega, \\ D(s) \partial_{\boldsymbol{n}} q_c + \sigma \gamma(s) q_c = 0, & (s, \boldsymbol{r}) \in [0,1] \times \Gamma, \\ q_c(1, \boldsymbol{r}) = 1, & \boldsymbol{r} \in \Omega, \end{cases} \tag{5.9}$$

where the surface energy terms now enter as a Robin (mixed) boundary conditions. This formulation contains no delta-like terms, thus, is immediately amenable to standard numerical methods. However, one can also clearly see that density fields obtained from solutions of these equations will not be able to satisfy equation (5.5) representing the incompressibility of the system. Indeed, taking the derivative of (5.7) on the domain boundary in the normal direction and using the boundary conditions from (5.8) and (5.9) one obtains:

$$\partial_{\boldsymbol{n}} \left(\rho_A + \rho_B \right) = -2\sigma \left(\gamma_A \rho_A + \gamma_B \rho_B \right),$$

while differentiating the incompressibility equation (5.5) leads to:

$$\partial_{\boldsymbol{n}} \left(\rho_A + \rho_B\right) = 0,$$

and, generally, $(\gamma_A \rho_A + \gamma_B \rho_B) \neq 0$. Attempting to solve the entire system of SCFT equations with chain propagators satisfying the above diffusion equations produces pressure fields with a singular behavior near domain boundaries. While for certain applications such a singular behavior does not have an adverse effect, it is crucial to have well-defined values of the pressure field for modeling polymer-air and polymer-particle interactions. We now show that the described inconsistency is only apparent and can be completely removed. Using again the results of 5.A, we can write the diffusion equations (5.8) and (5.9) in yet another equivalent form:

$$\begin{cases} \partial_s q + \left(\mu(s) - \sigma\tilde{\gamma}\delta_\Gamma\right) q = D(s)\nabla^2 q, & (s, \boldsymbol{r}) \in [0,1] \times \Omega, \\ D(s)\partial_{\boldsymbol{n}} q + \sigma\left(\gamma(s) - \tilde{\gamma}\right) q = 0, & (s, \boldsymbol{r}) \in [0,1] \times \Gamma, \\ q(0, \boldsymbol{r}) = 1, & \boldsymbol{r} \in \Omega, \end{cases} \tag{5.10}$$

and

$$\begin{cases} -\partial_s q_c + \left(\mu(s) - \sigma\tilde{\gamma}\delta_\Gamma\right) q_c = D(s)\nabla^2 q_c, & (s, \boldsymbol{r}) \in [0,1] \times \Omega, \\ D(s)\partial_{\boldsymbol{n}} q_c + \sigma\left(\gamma(s) - \tilde{\gamma}\right) q_c = 0, & (s, \boldsymbol{r}) \in [0,1] \times \Gamma, \\ q_c(1, \boldsymbol{r}) = 1, & \boldsymbol{r} \in \Omega, \end{cases} \tag{5.11}$$

where we added and subtracted an arbitrary function $\tilde{\gamma} = \tilde{\gamma}(\boldsymbol{r})$ defined on the domain's boundary Γ from the surface energy terms of the boundary conditions. Having this freedom in choosing the exact form of $\tilde{\gamma}$ allows one to ensure that the boundary conditions in equations (5.10) and (5.11) are consistent with the incompressibility equation (5.5). Indeed, taking the normal derivative of the total density on the domain boundary and

equating it to zero leads to the following equation for $\tilde{\gamma}$:

$$-2\frac{\gamma_A - \tilde{\gamma}}{D_A}\rho_A - 2\frac{\gamma_B - \tilde{\gamma}}{D_B}\rho_B = 0,$$

from which it follows that if $\tilde{\gamma}$ is chosen as:

$$\tilde{\gamma} = \frac{\gamma_A D_A^{-1}\rho_A + \gamma_B D_B^{-1}\rho_B}{D_A^{-1}\rho_A + D_B^{-1}\rho_B}.$$

then there is no inconsistency between the incompressibility equation (5.5) and the boundary conditions in (5.10) and (5.11). In the case where A and B blocks have equal statistical segment lengths, that is, $D_A = D_B$, the expression for $\tilde{\gamma}$ simplifies to:

$$\tilde{\gamma} = \gamma_A\rho_A + \gamma_B\rho_B$$

Finally, redefining the pressure field as $\tilde{\mu_+} = \mu_+ + \sigma\tilde{\gamma}\delta_\Gamma$, or in other words, explicitly subtracting from it the singular part, one arrives to the following equations for chain propagators:

$$\begin{cases} \partial_s q + \tilde{\mu}(s)q = D(s)\nabla^2 q, & (s, \boldsymbol{r}) \in [0,1] \times \Omega, \\ D(s)\partial_{\boldsymbol{n}} q + \sigma\left(\gamma(s) - \tilde{\gamma}\right) q = 0, & (s, \boldsymbol{r}) \in [0,1] \times \Gamma, \\ q(0, \boldsymbol{r}) = 1, & \boldsymbol{r} \in \Omega, \end{cases} \tag{5.12}$$

and

$$\begin{cases} -\partial_s q_c + \tilde{\mu}(s)q_c = D(s)\nabla^2 q_c, & (s, \boldsymbol{r}) \in [0,1] \times \Omega, \\ D(s)\partial_{\boldsymbol{n}} q_c + \sigma\left(\gamma(s) - \tilde{\gamma}\right) q_c = 0, & (s, \boldsymbol{r}) \in [0,1] \times \Gamma, \\ q_c(1, \boldsymbol{r}) = 1, & \boldsymbol{r} \in \Omega, \end{cases} \tag{5.13}$$

where

$$\tilde{\mu}(s) = \begin{cases} \tilde{\mu_+} - \mu_-, & s \in [0, f], \\ \tilde{\mu_+} + \mu_-, & s \in [f, 1], \end{cases}$$

and the system's energy:

$$\mathcal{H} = \frac{n}{V} \int\limits_{\Omega} \left(\frac{\mu_-^2}{\chi_{AB} N} - \tilde{\mu_+} \right) \, d\boldsymbol{r} - n \log(\mathcal{Q}) + \sigma \frac{n}{V} \int\limits_{\Gamma} \tilde{\gamma} \, d\Gamma, \tag{5.14}$$

which are completely free of inconsistencies.

It is interesting that such a procedure automatically results in an additional term in the energy functional of the form:

$$\sigma \frac{n}{V} \int\limits_{\Gamma} \tilde{\gamma} \, d\Gamma = \sigma \frac{n}{V} \int\limits_{\Gamma} \left(\gamma_A \rho_A + \gamma_B \rho_B \right) \, d\Gamma,$$

which has a straightforward meaning of the total surface energy of a mixture of A and B species.

For convenience, we drop the tilde symbol above $\tilde{\mu_+}$ in the rest of this manuscript.

In the numerical experiments, we characterize the strength of surface energy using the expression for interfacial tension of a symmetric polymer-polymer interface estimated in the case of strong segregation [47]:

$$\gamma = \frac{1}{\sigma} \sqrt{\chi N}.$$

Imposing surface energies using the above formula in terms of χN allows straightforward comparison with A-B interaction strength $\chi_{AB} N$.

5.2.1 Solving the SCFT equations for μ_+^* and μ_-^*

The common approach for obtaining a saddle point $(\mu_+, \mu_-) = \left(\mu_+^*, \mu_-^*\right)$ of the Hamiltonian $\mathcal{H}$ is to start from some initial guess (seed) $(\mu_+, \mu_-) = \left(\mu_+^{(0)}, \mu_-^{(0)}\right)$ and to iteratively evolve the fields (μ_+, μ_-) in the steepest descent/ascent directions until a saddle point is reached (within a specified tolerance ε_{tol}), that is, given values at the kth iteration fields at the $(k+1)$th iteration are updated as:

$$\begin{aligned}
\mu_+^{(k+1)} &= \mu_+^{(k)} + \lambda_+ \frac{\delta \mathcal{H}}{\delta \mu_+}[\mu_+^{(k)}, \mu_-^{(k)}] = \mu_+^{(k)} + \lambda_+ \left(\rho_A^{(k)} + \rho_B^{(k)} - 1\right), \\
\mu_-^{(k+1)} &= \mu_-^{(k)} - \lambda_- \frac{\delta \mathcal{H}}{\delta \mu_-}[\mu_+^{(k)}, \mu_-^{(k)}] = \mu_-^{(k)} - \lambda_- \left(\frac{2\mu_-^{(k)}}{\chi_{AB} N} - \rho_A^{(k)} + \rho_B^{(k)}\right),
\end{aligned} \tag{5.15}$$

where λ_+ and λ_- are step sizes in moving along steepest descent/ascent directions (typically, taken as $\lambda_+ = \lambda_- = 1$), and iterations are terminated once

$$\frac{1}{V}\sqrt{\int_\Omega \left(\frac{\delta \mathcal{H}}{\delta \mu_+}\right)^2 d\boldsymbol{r}} < \varepsilon_{\text{tol}} \quad \text{and} \quad \frac{1}{V}\sqrt{\int_\Omega \left(\frac{\delta \mathcal{H}}{\delta \mu_-}\right)^2 d\boldsymbol{r}} < \varepsilon_{\text{tol}}. \tag{5.16}$$

The solution of the proposed modified system of the SCFT equations accounting for surface energies is further complicated by the presence of terms depending on $\tilde{\gamma}$, which in turn is a function of the local densities ρ_A and ρ_B. One possible strategy could be simply to recalculate the values for $\tilde{\gamma}$ every few fields updates μ_- and μ_+. However, we observed that this approach still results in numerical artifacts in the pressure field μ_+. This seems to be linked to the fact that in such a case, the density fields ρ_A and ρ_B generally do not satisfy the condition $\partial_{\boldsymbol{n}} (\rho_A + \rho_B) = 0$ during such iterations. Once the pressure field is polluted with features of small enough scale they persist for extremely long times. It

showed to be a better strategy to iteratively find $\tilde{\gamma}$ that satisfies $\partial_{\boldsymbol{n}}(\rho_A + \rho_B) = 0$ (we observed that a small number of iterations n_γ = 2-4 is sufficient) before every update of fields μ_- and μ_+. The overall procedure for solving modified system of SCFT equations can be summarized as follows:

1. Set a desired seed for $\mu_-^{(0)}$ and $\mu_+^{(0)} = 0$.

2. Iterate in k until conditions (5.16) are satisfied:

   (a) Iterate n_γ times:

      i. Set $\tilde{\gamma} = \dfrac{\gamma_A \rho_A + \gamma_B \rho_B}{\rho_A + \rho_B}$

      ii. Solve diffusion equations (5.12) and (5.13)

      iii. Compute density fields according to (5.7)

   (b) Compute $\mu_+^{(k+1)}$ and $\mu_-^{(k+1)}$ using (5.15)

5.3 Sensitivity of free energy to shape of free surface

Let us now consider a polymer droplet on a (possibly curved) substrate as illustrated in figure 5.3. Denote the polymer-air and the polymer-wall interfaces by Γ_a and Γ_w, respectively, and by γ_{aA}, γ_{aB} and γ_{wA}, γ_{wB} the surface energies of the A and B blocks on these interfaces, correspondingly. Denote by γ_{aw} the surface energy of the air-wall interface. We assume that surface energies γ_{aA}, γ_{aB}, and γ_{aw} are constant while $\gamma_{wA} = \gamma_{wA}(\boldsymbol{r})$ and $\gamma_{wB} = \gamma_{wB}(\boldsymbol{r})$ may be non-uniform in space (as, for example, for chemically

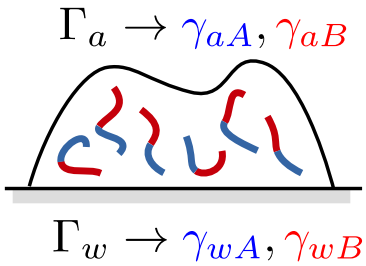


Figure 5.3: Problem geometry and notation used in section 5.3.

patterned substrates). Given the results of section 5.2, the energy of such a system can be expressed as:

$$\mathcal{H} \;=\; \frac{n}{V}\int_{\Omega}\left(\frac{\mu_-^2}{\chi_{AB}N}-\mu_+\right)d\boldsymbol{r} \;-\; n\log(\mathcal{Q}) \;+\; \sigma\frac{n}{V}\int_{\Gamma_a}\tilde{\gamma}_a\,d\Gamma \;+\; \sigma\frac{n}{V}\int_{\Gamma_w}(\tilde{\gamma}_w-\gamma_{aw})\,d\Gamma,$$

where the single chain partition function $\mathcal{Q}$ is given by (5.1) and where the chain propagator q satisfies the modified diffusion equation (5.12) provided $\Gamma = \Gamma_a \cup \Gamma_w$ and

$$\gamma_A = \begin{cases}\gamma_{aA}, & \boldsymbol{r}\in\Gamma_a,\\ \gamma_{wA}, & \boldsymbol{r}\in\Gamma_w,\end{cases} \qquad \gamma_B = \begin{cases}\gamma_{aB}, & \boldsymbol{r}\in\Gamma_a,\\ \gamma_{wB}, & \boldsymbol{r}\in\Gamma_w.\end{cases}$$

For the derivation of the system's energy sensitivity with respect to the shape of free surface Γ_a, it is convenient to consider a situation in which the free surface is being deformed with an arbitrary normal velocity $v_{\boldsymbol{n}} = v_{\boldsymbol{n}}(\tau, \boldsymbol{r})$ in fictitious time τ, i.e., $\Omega = \Omega(\tau)$, $\Gamma_a = \Gamma_a(\tau)$, and $\Gamma_w = \Gamma_w(\tau)$, as illustrated in figure 5.4. In such a case the

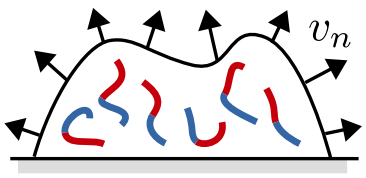


Figure 5.4: Deformation of free surface in normal velocity field $v_{\boldsymbol{n}}$.

shape sensitivity of the free energy can be expressed as the derivative of $\mathcal{H}$ with respect

to τ. This derivative must be obtained taking into account the fact that the fields μ_+ and μ_- and the chain propagator q are also functions of τ, satisfying the system of nonlinear SCFT equations (5.5)-(5.6) for any given domain configuration $\Omega(\tau)$. However, by applying the chain rule:

$$\frac{d\mathcal{H}}{d\tau} = \int_\Omega \underbrace{\frac{\delta\mathcal{H}}{\delta\mu_+}}_{=0} \frac{d\mu_+}{d\tau}\, d\boldsymbol{r} + \int_\Omega \underbrace{\frac{\delta\mathcal{H}}{\delta\mu_-}}_{=0} \frac{d\mu_-}{d\tau}\, d\boldsymbol{r} + \left.\frac{d\mathcal{H}}{d\tau}\right|_{\mu=\text{const}} = \left.\frac{d\mathcal{H}}{d\tau}\right|_{\mu=\text{const}}$$

one can see that since μ_+ and μ_- satisfy the saddle point conditions (5.4) they could be assumed constant during the derivation, and thus, only the equation for the chain propagator q (5.12) needs to be taken into account. Using the weak form of that equation (see 5.A), we define the following Lagrangian:

$$\begin{aligned}\mathcal{L} = \frac{n}{V}\int_\Omega \left(\frac{\mu_-^2}{\chi N} - \mu_+\right) d\boldsymbol{r} - n\log\left(\frac{1}{V}\int_\Omega q(1)\, d\boldsymbol{r}\right) \\ + \sigma\frac{n}{V}\int_{\Gamma_a} \tilde{\gamma}_a\, d\Gamma + \sigma\frac{n}{V}\int_{\Gamma_w} (\tilde{\gamma}_w - \gamma_{aw})\, d\Gamma \\ + \int_\Omega\int_0^1 (-q\partial_s\lambda_c + \mu(s)q\lambda_c + D(s)\nabla q\cdot\nabla\lambda_c)\, ds\, d\boldsymbol{r} + \int_\Omega (q(1)\lambda_c(1) - \lambda_c(0))\, d\boldsymbol{r} \\ + \int_{\Gamma_a}\int_0^1 \sigma(\gamma_a - \tilde{\gamma}_a)q\lambda_c\, ds\, d\boldsymbol{r} + \int_{\Gamma_w}\int_0^1 \sigma(\gamma_w - \tilde{\gamma}_w)q\lambda_c\, ds\, d\boldsymbol{r},\end{aligned}$$

where λ_c is a Lagrangian multiplier associated with the PDE-constraint. The full derivative of $\mathcal{H}$ with respect to τ is then equal to the partial derivative of $\mathcal{L}$, that is, $\frac{d\mathcal{H}}{d\tau} = \frac{\partial\mathcal{L}}{\partial\tau}$, provided that the following optimality conditions are satisfied:

$$\delta_{\lambda_c}\mathcal{L} = 0, \quad \forall \lambda_c,$$
$$\delta_q\mathcal{L} = 0, \quad \forall q.$$

It is trivial to show that the first expression recovers the weak form of (5.12). The second equation provides an equation for Lagrangian multiplier λ_c. After explicitly taking the variation of $\mathcal{L}$ with respect to q, one obtains the following expression:

$$\begin{aligned}\delta_q\mathcal{L} = &\int_\Omega \left(\lambda_c(1) - \frac{n}{\mathcal{Q}V}\right)\delta q(1)\, d\boldsymbol{r} \\ &+ \int_\Omega \int_0^1 \left(-\delta q \partial_s \lambda_c + \mu(s)\delta q \lambda_c + D(s)\nabla\delta q \cdot \nabla\lambda_c\right)\, ds\, d\boldsymbol{r} \\ &+ \int_{\Gamma_a} \int_0^1 \sigma(\gamma_a - \tilde{\gamma}_a)\delta q \lambda_c\, ds\, d\boldsymbol{r} + \int_{\Gamma_w} \int_0^1 \sigma(\gamma_w - \tilde{\gamma}_w)\delta q \lambda_c\, ds\, d\boldsymbol{r}.\end{aligned}$$

Using the integration by parts formula (5.25) leads to which is a weak of the following diffusion equation:

$$\begin{aligned}\delta_q\mathcal{L} = &\int_\Omega \left(\lambda_c(1) - \frac{n}{\mathcal{Q}V}\right)\delta q(1)\, d\boldsymbol{r} \\ &+ \int_\Omega \int_0^1 \left(-\partial_s\lambda_c + \mu(s)\lambda_c - D(s)\nabla^2\lambda_c\right)\delta q\, ds\, d\boldsymbol{r} \\ &+ \sum_{\nu=a,w} \int_{\Gamma_\nu} \int_0^1 \left(D(s)\partial_{\boldsymbol{n}}\lambda_c + \sigma(\gamma_\nu - \tilde{\gamma}_\nu)\lambda_c\right)\delta q\, ds\, d\boldsymbol{r}.\end{aligned}$$

As one can see $\delta_q\mathcal{L} = 0$, $\forall q$, leading to a weak formulation of the diffusion equation for the Lagrange multiplier λ_c that is completely analogous to the equation for the complimentary chain propagator q_c (5.13) with the exception that the initial conditions for λ_c

are $\lambda_c(1, \boldsymbol{r}) = \frac{n}{\mathcal{Q}V}$, $\boldsymbol{r} \in \Omega$. This means that the Lagrange multiplier λ_c is nothing but the complimentary chain propagator multiplied by a factor of $\frac{n}{\mathcal{Q}V}$, that is, $\lambda_c = \frac{n}{\mathcal{Q}V} q_c$.

We now turn our attention to computing the partial derivative of the Lagrangian $\mathcal{L}$ with respect to τ. It can be shown (see 5.B) that for integrals of the form:

$$\int\limits_{\Omega(\tau)} f \, d\boldsymbol{r}, \quad \int\limits_{\Gamma_a(\tau)} f \, d\boldsymbol{r}, \quad \text{and} \quad \int\limits_{\Gamma_w(\tau)} f \, d\boldsymbol{r}, \tag{5.17}$$

where $f(\tau, \boldsymbol{r})$ is some function, the following equations hold true:

$$\begin{aligned}
\frac{\partial}{\partial \tau} \int\limits_{\Omega} f \, d\boldsymbol{r} &= \int\limits_{\Omega} \frac{\partial f}{\partial \tau} \, d\boldsymbol{r} + \int\limits_{\Gamma_a} f v_{\boldsymbol{n}} \, d\boldsymbol{r}, \\
\frac{\partial}{\partial \tau} \int\limits_{\Gamma_w} f \, d\boldsymbol{r} &= \int\limits_{\Gamma_w} \frac{\partial f}{\partial \tau} \, d\boldsymbol{r} + \int\limits_{\Gamma_a \cap \Gamma_w} f \frac{v_{\boldsymbol{n}}}{\sin(\theta_c)} \, d\boldsymbol{r}, \\
\frac{\partial}{\partial \tau} \int\limits_{\Gamma_a} f \, d\boldsymbol{r} &= \int\limits_{\Gamma_a} \frac{\partial f}{\partial \tau} \, d\boldsymbol{r} + \int\limits_{\Gamma_a} (\kappa + \partial_{\boldsymbol{n}}) \, f v_{\boldsymbol{n}} \, d\boldsymbol{r} + \int\limits_{\Gamma_a \cap \Gamma_w} f \cot(\theta_c) v_{\boldsymbol{n}} \, d\boldsymbol{r},
\end{aligned} \tag{5.18}$$

where κ is the curvature of the polymer-air interface Γ_a and θ_c is the angle between Γ_a and Γ_w at their intersection. Applying these formulas while differentiating $\mathcal{L}$ with respect to τ, one arrives at (assuming the volume of the system V is constant):

$$\mathcal{L} = \frac{n}{V}\int\limits_{\Gamma_a}\left(\frac{\mu_-^2}{\chi N} - \mu_+\right) v_{\boldsymbol{n}}\, d\boldsymbol{r} - \frac{n}{\mathcal{Q}V}\int\limits_{\Gamma_a} q(1) v_{\boldsymbol{n}}\, d\boldsymbol{r}$$
$$+ \sigma\frac{n}{V}\int\limits_{\Gamma_a} v_{\boldsymbol{n}}\left(\kappa + \partial_{\boldsymbol{n}}\right)\tilde{\gamma}_a\, d\Gamma + \sigma\frac{n}{V}\int\limits_{\Gamma_a\cap\Gamma_w}\left(\tilde{\gamma}_a\cos(\theta_c) + \tilde{\gamma}_w - \gamma_{aw}\right)\frac{v_{\boldsymbol{n}}}{\sin(\theta_c)}\, d\Gamma$$
$$+ \int\limits_{\Gamma_a}\int\limits_0^1 \left(-q\partial_s\lambda_c + \mu(s) q\lambda_c + D(s)\nabla q\cdot\nabla\lambda_c\right) v_{\boldsymbol{n}}\, ds\, d\boldsymbol{r}$$
$$+ \int\limits_{\Gamma_a}\left(q(1)\lambda_c(1) - \lambda_c(0)\right) v_{\boldsymbol{n}}\, d\boldsymbol{r} + \sigma\int\limits_{\Gamma_a}\int\limits_0^1 v_{\boldsymbol{n}}\left(\kappa + \partial_{\boldsymbol{n}}\right)\left(\gamma_a - \tilde{\gamma}_a\right) q\lambda_c\, ds\, d\boldsymbol{r}$$
$$+ \sigma\int\limits_{\Gamma_a\cap\Gamma_w}\int\limits_0^1 \left(\left(\gamma_a - \tilde{\gamma}_a\right)\cos(\theta_c) + \gamma_w - \tilde{\gamma}_w\right)\frac{v_{\boldsymbol{n}}}{\sin(\theta_c)} q\lambda_c\, ds\, d\boldsymbol{r}.$$

Furthermore, using $\nabla q\cdot\nabla\lambda_c = \frac{1}{2}\nabla^2\left(q\lambda_c\right) - \frac{1}{2}q\nabla^2\lambda_c - \frac{1}{2}\lambda_c\nabla^2 q$, the diffusion equations (5.12) and (5.13) and the SCFT equations (5.5)-(5.6), the expression for the full derivative of the system's energy can be simplified to:

$$\frac{d\mathcal{H}}{d\tau} = \frac{n}{V}\int\limits_{\Gamma_a}\left(\frac{\mu_-^2}{\chi N} - \mu_+ - \frac{q(1)+q_c(0)}{2Q} + \sigma\left(\kappa + \frac{\partial}{\partial\boldsymbol{n}}\right)\gamma_a^{\text{eff}}\right) v_n\, d\Gamma$$
$$+ \sigma\frac{n}{V}\int\limits_{\Gamma_a\cap\Gamma_w}\left(\gamma_w^{\text{eff}} - \gamma_0 + \gamma_a^{\text{eff}}\cos(\theta_c)\right)\frac{v_n}{\sin(\theta_c)}\, d\Gamma, \quad (5.19)$$

where the effective surface energies γ_w^{eff} and γ_a^{eff} are defined as:

$$\gamma_w^{\text{eff}} = \rho_A\gamma_{wA} + \rho_B\gamma_{wB},$$
$$\gamma_a^{\text{eff}} = \rho_A\gamma_{aA} + \rho_B\gamma_{aB}.$$

The polymer material is in the equilibrium whenever the integrands in the above expressions are equal to zero. This produces the following two equilibrium conditions:

$$\frac{\mu_-^2}{\chi N} - \mu_+ - \frac{q(1)+q_c(0)}{2Q} + \sigma\left(\kappa + \frac{\partial}{\partial \boldsymbol{n}}\right)\gamma_a^{\text{eff}} = 0,$$
$$\gamma_w^{\text{eff}} - \gamma_{aw} + \gamma_a^{\text{eff}}\cos(\theta_c) = 0.$$

The first of these equations dictates the overall shape of free surface Γ_a, while the second one governs the value of the contact angle at the wall-polymer-air triple junction point and can be recognized as the extension of the Young equation for contact angles to block copolymer materials.

Numerically, the equilibrium shape can be achieved by starting from some initial free surface shape and deforming it according to:

$$v_{\boldsymbol{n}} = -\left(\frac{\mu_-^2}{\chi N} - \mu_+ - \frac{q(1)+q_c(0)}{2Q} + \sigma\left(\kappa + \frac{\partial}{\partial \boldsymbol{n}}\right)\gamma_a^{\text{eff}}\right),$$

while imposing the contact angle:

$$\theta_c = \cos^{-1}\left(\frac{\gamma_{aw} - \gamma_w^{\text{eff}}}{\gamma_w^{\text{eff}}}\right).$$

In such a way, the energy of the system is forced to decrease until it reaches a local minimum.

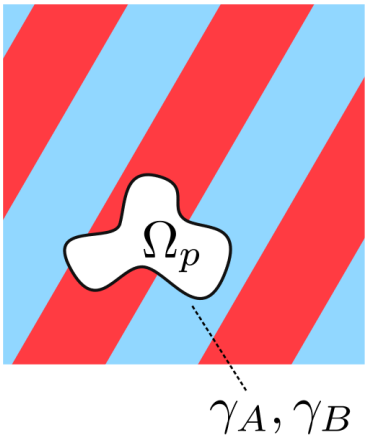


Figure 5.5: Problem geometry and notation used in section 5.4.

5.4 Sensitivity of free energy to position and orientation of a nanoparticle

We now consider a particle of arbitrary shape and arbitrary, possibly varying along the particle's boundary, surface energies $\gamma_{Ap} = \gamma_{Ap}(\boldsymbol{r})$ and $\gamma_{Bp} = \gamma_{Bp}(\boldsymbol{r})$ immersed in a block copolymer melt as illustrated in figure 5.5. The energy of the system in this case is given by expression (5.14) with the chain propagators satisfying (5.12). Similarly to the derivation in the case of a free surface, we assume that the particle moves with some translational and rotational velocities $\boldsymbol{v} = \boldsymbol{v}(\tau)$ and $\boldsymbol{\omega} = \boldsymbol{\omega}(\tau)$ in fictitious time τ (see figure 5.6).

Let us first consider the case of spatially uniform surface energies $\gamma_{Ap} = const$ and $\gamma_{Bp} = const$. For such a situation, the derivative of the system's energy can be immediately obtained from the expression derived in the previous section. Indeed, the normal velocity of the particle is equal to

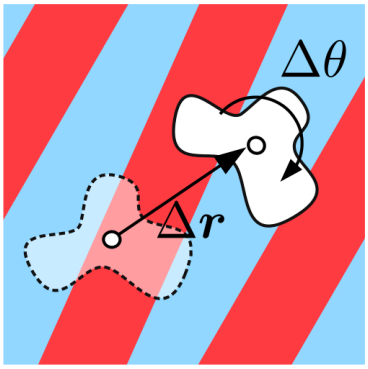


Figure 5.6: Motion of a nanoparticle in a block copolymer melt.

$$v_{\boldsymbol{n}} = \boldsymbol{n} \cdot (\boldsymbol{v} + \boldsymbol{\omega} \times (\boldsymbol{r} - \boldsymbol{r}_c)),$$

where $\boldsymbol{r}_c$ denotes the particle's center. Substituting the above expression into (5.19) leads to

$$\frac{d\mathcal{H}}{d\tau} = \int_\Gamma \boldsymbol{G} \cdot \boldsymbol{v}\, d\Gamma + \int_\Gamma ((\boldsymbol{r} - \boldsymbol{r}_c) \times \boldsymbol{G}) \cdot \boldsymbol{\omega}\, d\Gamma, \tag{5.20}$$

where

$$\boldsymbol{G} = \left(\frac{\mu_-^2}{\chi N} - \mu_+ - \frac{q(1) + q_c(0)}{2Q} + \left(\kappa + \frac{\partial}{\partial \boldsymbol{n}} \right) (\gamma_A \rho_A + \gamma_B \rho_B) \right) \boldsymbol{n}.$$

In order to obtain analogous expressions in the general case of non-uniform surface energies $\gamma_{Ap} = \gamma_{Ap}(\boldsymbol{r})$ and $\gamma_{Bp} = \gamma_{Bp}(\boldsymbol{r})$, one just needs to add the terms that result from their differentiation. Using

$$\begin{aligned} \frac{\partial \gamma_{Ap}}{\partial \tau} &= -(\boldsymbol{v} + \boldsymbol{\omega} \times (\boldsymbol{r} - \boldsymbol{r}_c)) \cdot \nabla \gamma_{Ap}, \\ \frac{\partial \gamma_{Bp}}{\partial \tau} &= -(\boldsymbol{v} + \boldsymbol{\omega} \times (\boldsymbol{r} - \boldsymbol{r}_c)) \cdot \nabla \gamma_{Bp}, \end{aligned}$$

it is straightforward to show that the energy derivative in this case is given by the same formula (5.20), but where $\boldsymbol{G}$ is now given by:

$$\boldsymbol{G} = \left(\frac{\mu_-^2}{\chi N} - \mu_+ - \frac{q(1) + q_c(0)}{2Q} + \left(\kappa + \frac{\partial}{\partial \boldsymbol{n}} \right) (\gamma_A \rho_A + \gamma_B \rho_B) \right) \boldsymbol{n} + \left(\frac{\partial \gamma_A}{\partial \boldsymbol{\tau}} \rho_A + \frac{\partial \gamma_B}{\partial \boldsymbol{\tau}} \rho_B \right) \boldsymbol{\tau},$$

where $\boldsymbol{\tau}$ denotes the tangential vector to the particle's boundary.

The equations

$$\boldsymbol{G} = 0 \quad \text{and} \quad (\boldsymbol{r} - \boldsymbol{r}_c) \times \boldsymbol{G} = 0$$

represent the equilibrium conditions for a particle in a block copolymer melt. Such a state can be numerically obtained by starting with some initial placement of the particle and iteratively adjusting its position and orientation according to

$$\boldsymbol{v} = -\boldsymbol{G} \quad \text{and} \quad \boldsymbol{\omega} = -(\boldsymbol{r} - \boldsymbol{r}_c) \times \boldsymbol{G}.$$

This procedure trivially extends to the case of many particles.

5.5 Sensitivity of polymer morphology to confining mask's geometry

The SCFT provides the means for obtaining BCP density fields for any confining geometry, that is, for solving the forward problem. In order to solve the inverse problem, that is, find the confining geometry that results in a specified density profile, we define a cost

functional that measures the deviation between an actual density field and a desired one and analytically derive its sensitivity to the confinement shape. Specifically, noting that

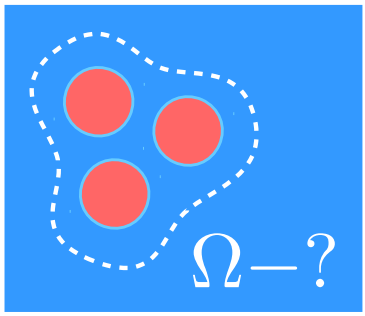


Figure 5.7: Problem geometry and notation used in section 5.5. The desired density field configuration is represented by the three disks.

the density fields ρ_A and ρ_B are directly related to the exchange field μ_- through the SCFT equation (5.6), we define the cost functional as:

$$\mathcal{F} = \int_{\Omega} \frac{1}{\chi_{AB}N} \left(\mu_- - \mu_t\right)^2 \, d\boldsymbol{r} + \alpha \int_{\Gamma} d\Gamma,$$

where Ω and Γ denotes the confining geometry and its boundary, μ_t is a desired density field configuration (see figure 5.7), and α is the curvature penalization parameter. The factor $\frac{1}{\chi_{AB}N}$ is introduced solely for the sake of convenience in further calculations. Since the efficiency and viability of the directed self-assembly is based on the ability to produce ordered nanostructures of characteristic dimensions smaller than those of guiding masks, it is important to have the means for controlling the maximum allowable curvature of the confining masks. To do so, we introduce a term penalizing for the total perimeter of the shape into the cost functional. However, we note that more intricate strategies can potentially be employed as well.

Following the approach used in section 5.3 for determining shape derivatives of the system's energy, we consider the situation in which the confining boundary is evolving with some normal velocity $v_{\boldsymbol{n}} = v_{\boldsymbol{n}}(\tau, \boldsymbol{r})$ in fictitious time τ and compute the derivative of the cost functional with respect to τ. Compared to the case of section 5.3 where the fields μ_- and μ_+ can be assumed to be constant in τ in the derivation, this assumption does not hold in the present case and the entire nonlinear system of SCFT equations needs to be taken into consideration while estimating the derivative of the cost functional. Thus, we define the Lagrangian as:

$$\begin{aligned}
\mathcal{L} = &\int_\Omega \frac{(\mu_- - \mu_t)^2}{\chi_{AB}N}\, d\boldsymbol{r} + \alpha \int_\Gamma d\Gamma + \int_\Omega \lambda_+ \left[V \int_0^1 q q_c \, ds - \int_\Omega \int_0^1 q q_c \, ds \, d\boldsymbol{r}' \right] d\boldsymbol{r} \\
&+ \int_\Omega \lambda_- \left[\frac{2}{\chi N} \mu_- \int_\Omega \int_0^1 q q_c \, ds \, d\boldsymbol{r}' + V \int_0^1 \operatorname{sign}(s-f) q q_c \, ds \right] d\boldsymbol{r} \\
&+ \int_\Omega \left(q(1)\lambda_c(1) - \lambda_c(0) \right) d\boldsymbol{r} + \int_\Omega \int_0^1 \left(-q\partial_s \lambda_c + \mu q \lambda_c + D(s) \nabla q \cdot \nabla \lambda_c \right) ds \, d\boldsymbol{r} \\
&+ \int_\Omega \left(q_c(0)\lambda(0) - \lambda(1) \right) d\boldsymbol{r} + \int_\Omega \int_0^1 \left(q_c \partial_s \lambda + \mu q_c \lambda + D(s) \nabla q_c \cdot \nabla \lambda \right) ds \, d\boldsymbol{r} \\
&+ \int_\Gamma \int_0^1 \sigma(\gamma(s) - \tilde{\gamma}) q \lambda_c \, ds \, d\boldsymbol{r} + \int_\Gamma \int_0^1 \sigma(\gamma(s) - \tilde{\gamma}) q_c \lambda \, ds \, d\boldsymbol{r},
\end{aligned}$$

where the diffusion equations (5.12) and (5.13) are incorporated using their weak from and λ, λ_c, λ_-, and λ_+ are the Lagrange multipliers. The full derivative of the cost functional with respect to τ is equal to the partial derivative of the above Lagrangian

provided the following optimality conditions are satisfied:

$$\delta_q \mathcal{L} = 0, \quad \delta_\lambda \mathcal{L} = 0,$$

$$\delta_{q_c} \mathcal{L} = 0, \quad \delta_{\lambda_c} \mathcal{L} = 0,$$

$$\delta_{\mu_+} \mathcal{L} = 0, \quad \delta_{\lambda_+} \mathcal{L} = 0,$$

$$\delta_{\mu_-} \mathcal{L} = 0, \quad \delta_{\lambda_-} \mathcal{L} = 0.$$

It is trivial to show that the conditions from the right column simply recover equations (5.12), (5.13), (5.5), and (5.5), respectively. The conditions from the left column provide the equations for the Lagrange multipliers. Taking the corresponding variations of the Lagrangian and equating them to zero, one obtains the following integral equations:

$$\delta_q \mathcal{L} = \int_\Omega \int_0^1 \left(-\delta q \partial_s \lambda_c + \mu \delta q \lambda_c + \mu_\lambda \delta q q_c + D(s) \nabla \delta q \cdot \nabla \lambda_c\right) \, ds \, d\boldsymbol{r} + \int_\Omega \delta q(1) \lambda_c(1) \, d\boldsymbol{r} + \int_\Gamma \int_0^1 \sigma(\gamma(s) - \tilde{\gamma}) \delta q \lambda_c \, ds \, d\boldsymbol{r} = 0,$$

$$\delta_{q_c} \mathcal{L} = \int_\Omega \int_0^1 \left(\delta q_c \partial_s \lambda + \mu \delta q_c \lambda + \mu_\lambda q \delta q_c + D(s) \nabla \delta q_c \cdot \nabla \lambda\right) \, ds \, d\boldsymbol{r} + \int_\Omega \delta q_c(0) \lambda(0) \, d\boldsymbol{r} + \int_\Gamma \int_0^1 \sigma(\gamma(s) - \tilde{\gamma}) \lambda \delta q_c \, ds \, d\boldsymbol{r} = 0,$$

$$\delta_{\mu_-}\mathcal{L} = \int_\Omega 2\frac{\mu_- - \mu_t}{\chi_{AB}N}\delta\mu_t\, d\boldsymbol{r} + \int_\Omega \frac{2}{\chi N}\lambda_- \int_\Omega \int_0^1 qq_c\, ds\, d\boldsymbol{r}'\delta\mu_-\, d\boldsymbol{r} + \int_\Omega \int_0^1 \operatorname{sign}(s-f)\left(q\lambda_c + q_c\lambda\right)\delta\mu_-\, ds\, d\boldsymbol{r} = 0,$$

$$\delta_{\mu_+}\mathcal{L} = \int_\Omega \int_0^1 \left(q\lambda_c + q_c\lambda\right)\delta\mu_-\, ds\, d\boldsymbol{r} = 0,$$

where

$$\mu_\lambda = \lambda_+ - \langle\lambda_+\rangle + \operatorname{sign}(s-f)\lambda_- + \left\langle \frac{2\mu_-\lambda_-}{\chi N} \right\rangle.$$

Using the divergence theorem, it can be shown that the first two of the above equations are weak forms of the following diffusion equations for λ and λ_c:

$$\begin{cases} \partial_s\lambda + \mu(s)\lambda + \mu_\lambda(s)q = D(s)\nabla^2\lambda, \\ D(s)\partial_{\boldsymbol{n}}\lambda + \sigma\left(\gamma(s) - \tilde{\gamma}\right)\lambda = 0, \\ \lambda(0,\boldsymbol{r}) = 0, \end{cases}$$

and

$$\begin{cases} -\partial_s\lambda_c + w\lambda_c + w_\lambda q_c = D(s)\nabla^2\lambda_c, \\ D(s)\partial_{\boldsymbol{n}}\lambda_c + \sigma\left(\gamma(s) - \tilde{\gamma}\right)\lambda_c = 0, \\ \lambda_c(1,\boldsymbol{r}) = 0. \end{cases}$$

The latter two integral equations are weak formulations of

$$\begin{aligned} &\rho_A^\lambda(\boldsymbol{r}) + \rho_B^\lambda(\boldsymbol{r}) = 0, \\ &\rho_A^\lambda(\boldsymbol{r}) - \rho_B^\lambda(\boldsymbol{r}) = 2\frac{\lambda_-(\boldsymbol{r}) + \mu_-(\boldsymbol{r}) - \mu_t(\boldsymbol{r})}{\chi_{AB}N}, \end{aligned}$$

where ρ_A^λ and ρ_B^λ are defined as

$$\rho_A^\lambda(\boldsymbol{r}) = \frac{1}{\mathcal{Q}} \int_0^f (q\lambda_c + q_c\lambda)\ ds,$$
$$\rho_B^\lambda(\boldsymbol{r}) = \frac{1}{\mathcal{Q}} \int_f^1 (q\lambda_c + q_c\lambda)\ ds.$$

Thus, the Lagrange multipliers λ, λ_c, λ_-, and λ_+ satisfy a nonlinear system with a similar structure as the original system of the SCFT equations.

Finally, taking the partial derivative of $\mathcal{L}$ with respect to τ produces:

$$\frac{d\mathcal{F}}{d\tau} = \frac{\partial \mathcal{L}}{\partial \tau} = \int_\Gamma \left\{ \frac{\left(\mu_-(\boldsymbol{r}) - \mu_t(\boldsymbol{r})\right)^2}{\chi N} - \frac{\lambda(1) + \lambda_c(0)}{2Q} + \frac{2\mu_-\lambda_-}{\chi N} - \left\langle \frac{2\mu_-\lambda_-}{\chi N} \right\rangle \right.$$
$$\left. - \lambda_+ + \langle \lambda_+ \rangle + \alpha\kappa \right\} v_{\boldsymbol{n}}\, d\Gamma, \quad (5.21)$$

where κ is the mean curvature of the confining geometry. The derived expression can be used for finding the confinement shapes that results in the closet match between the desired configuration μ_t and the actual one μ_- by iteratively evolving the shapes under the velocity field:

$$v_{\boldsymbol{n}} = -\left\{ \frac{\left(\mu_-(\boldsymbol{r}) - \mu_t(\boldsymbol{r})\right)^2}{\chi N} - \frac{\lambda(1) + \lambda_c(0)}{2Q} + \frac{2\mu_-\lambda_-}{\chi N} - \left\langle \frac{2\mu_-\lambda_-}{\chi N} \right\rangle \right.$$
$$\left. - \lambda_+ + \langle \lambda_+ \rangle + \alpha\kappa \right\}.$$

5.6 Numerical aspects

The application of the presented framework for investigating non-trivial situation requires quite advanced numerical capabilities. Specifically, one needs tools for the description and evolution of irregular interfaces as well as capabilities to solve modified diffusion equations (5.12) and (5.13) subject to Robin boundary conditions on such irregular interfaces. In this work, we use a combination of the Level-Set Method, adaptive Cartesian grids and sharp-interface finite-volume methods for solving PDE along the lines of [124], with the exception that the diffusion equations are solved with a more accurate schemes presented in [18]. The evolution of the free surface in the normal direction is performed by the semi-implicit advection scheme described [100].

5.7 Results

In this section, we present a number of numerical examples to validate the proposed framework and demonstrate its capabilities.

5.7.1 Imposing surface energies

We demonstrate the importance of the consistent approach for imposing surface energies presented in section 5.2 on the following three examples.

In the first example, we consider a flower-shaped domain defined by the zero-isocontour of the following function:

$$\phi_1(x,y) = \sqrt{x^2+y^2} - 5\left(1 + 0.3\cos(5\theta)\right),$$

where $\theta = \arctan\left(\frac{x}{y}\right)$ is the polar angle. In this example the surface energies are uniform and equal to $\gamma_A = \frac{1}{2\sigma}$ and $\gamma_B = -\frac{1}{2\sigma}$, that is, the domain's boundary preferentially attracts the polymer component B.

In the second example, we consider a domain formed by the intersection of a disk of radius $r_2 = 6.5R_g$ and a horizontal stripe of width $w_2 = 8R_g$. The straight segments of the domain's boundary preferentially attract the polymer component B, such that the surface energies there are equal to $\gamma_{As} = \frac{1}{2\sigma}$ and $\gamma_{Bs} = -\frac{1}{2\sigma}$. The curved segments of the domain's boundary preferentially attract the polymer component A, such that the surface energies there are equal to $\gamma_{Ac} = -\frac{1}{2\sigma}$ and $\gamma_{Bc} = \frac{1}{2\sigma}$.

Finally, in the third example we consider a circular disk of radius $r_3 = 6.5R_g$ located at $(x_3, y_3) = (0.1R_g, 2.1R_g)$ that is cut by a corrugated horizontal line defined as the zero-isocontour of function:

$$\phi_3(x,y) = -y + 0.01 - 0.25\cos\Big(2\pi\frac{x}{\lambda}\Big),$$

where $\lambda = 4\sqrt[6]{\frac{8\chi_{AB}N}{3\pi^4}}$. The circular section of the domain's boundary has uniform surface energies $\gamma_{Ac} = \frac{1}{2\sigma}$ and $\gamma_{Bc} = -\frac{1}{2\sigma}$ (attraction of the component B), while the bottom segment has nonuniform surface energies given by expressions:

$$\gamma_{Ab} = -2.5\cos\Big(2\pi\frac{x}{\lambda}\Big),$$
$$\gamma_{Bb} = 2.5\cos\Big(2\pi\frac{x}{\lambda}\Big).$$

As a result, the grooves attract the component B while the hills attract the component A.

In all the examples, the parameters of the polymer material are set to $\chi_{AB}N = 30$, $f = 0.33$. Polymer chains are discretized using $n_s = 60$ beads and the grid resolution is $20/256R_g$. The simulations begin with the field μ_- initialized with random values.

Figures 5.8 and 5.9 depict the confining geometries considered and the resulting density fields obtained using the original system of SCFT equations and the modified system in (5.12)-(5.13), while figures 5.10 and 5.11 demonstrate the resulting pressure fields obtained using the two approaches. As one can see, the solution of the two systems of equations leads to practically indistinguishable density fields, however, the standard approach leads to a singular behavior of μ_+ near domains boundaries, while the proposed approach results in pressure fields with well-defined values everywhere.

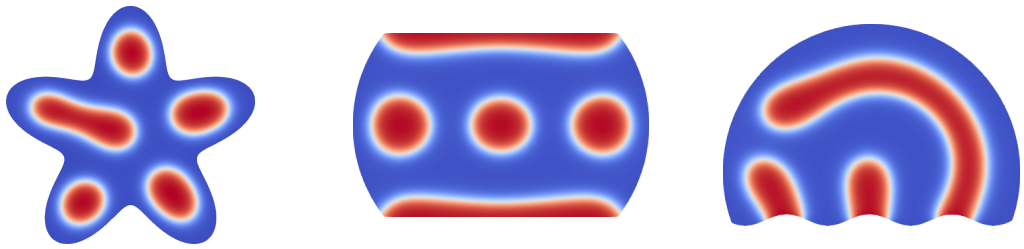


Figure 5.8: Resulting polymer morphologies obtained using the standard SCFT approach for imposing surface energies.

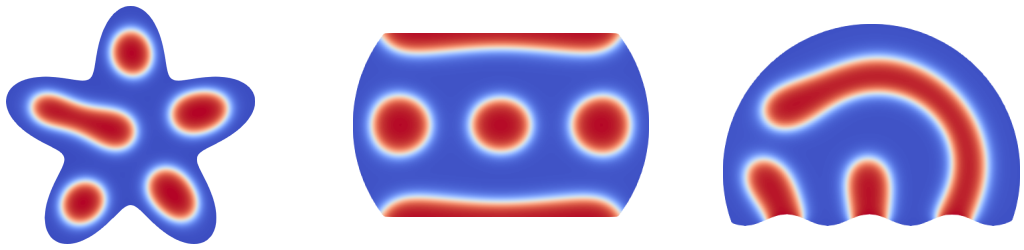


Figure 5.9: Resulting polymer morphologies obtained using the proposed approach for imposing surface energies.

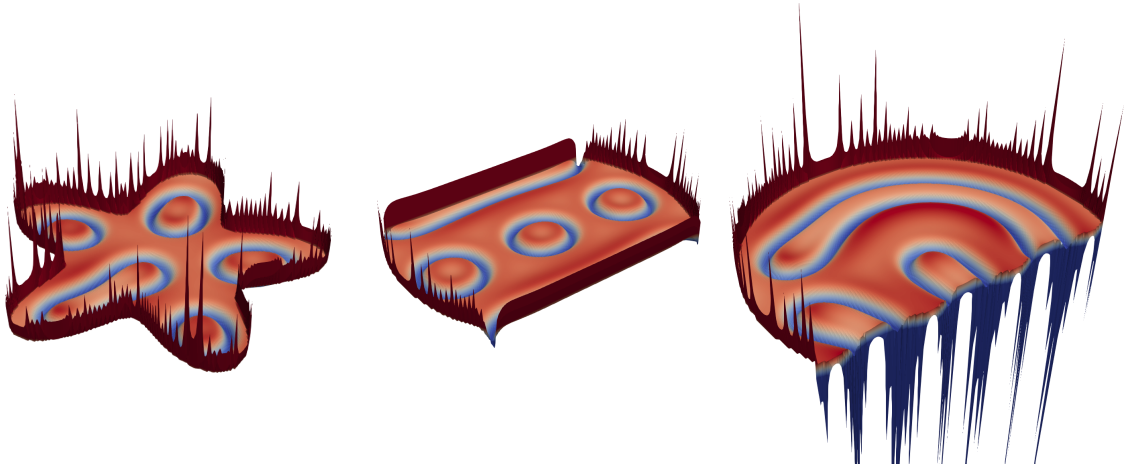


Figure 5.10: Resulting pressure field μ_+ obtained using the standard SCFT approach for imposing surface energies.

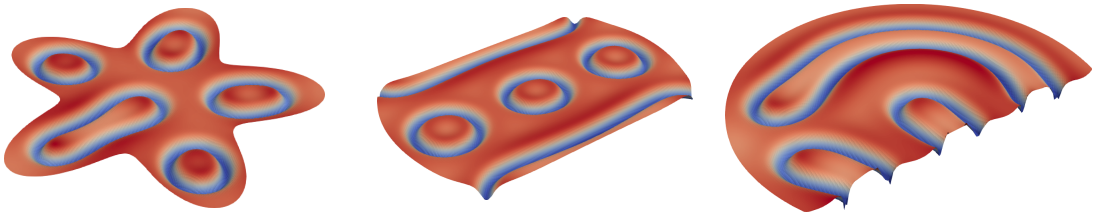


Figure 5.11: Resulting pressure field μ_+ obtained using the proposed approach for imposing surface energies.

5.7.2 Free surface block copolymers

Validation test

To validate the obtained expressions for the system energy sensitivity to the free surface shape, we consider a synthetic test in which we externally impose a velocity field for

the free surface, we track the change in energy at each iteration and we compare the numerical results to the prediction given in (5.19). Specifically, we consider a polymer droplet of radius $r = 0.611R_g$ and deform it in the velocity field (v_x, v_y):

$$\begin{aligned} v_x &= x\cos(5\tau), \\ v_y &= -y\cos(5\tau). \end{aligned} \tag{5.22}$$

Figure 5.12 visualize deformations that the polymer droplet undergoes and figure 5.13

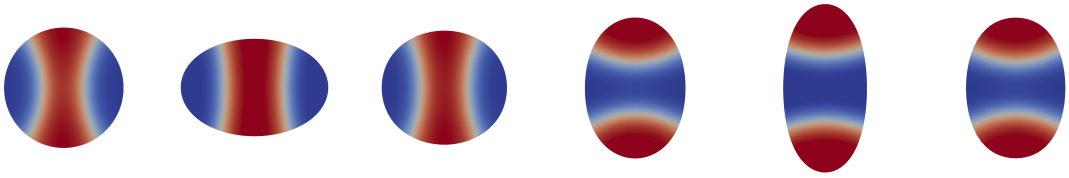


Figure 5.12: Visualization of imposed deformations in example 5.7.2.

compares the predicted and actual changes in energy for each iteration. The predicted values match the actual values very well almost everywhere, expect for two spikes that correspond to topology changes in the polymer morphologies and that cannot be described within the assumption of smooth behaviors.

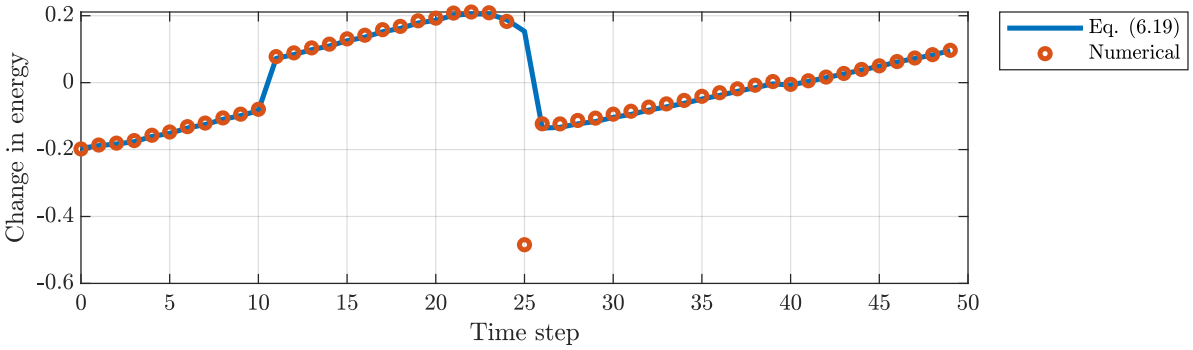


Figure 5.13: Comparison between changes in energy computed using (5.19) and numerically.

Substrate-supported BCP droplets

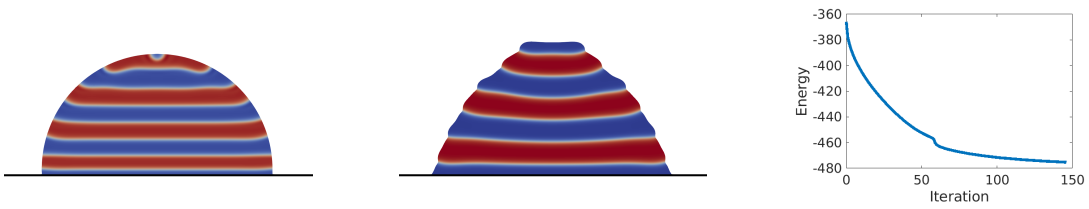
To demonstrate the capabilities of the proposed framework we consider several representative examples, specifically:

1. A droplet of lamella-forming diblock copolymer ($\chi_{AB}N = 30,\ f = 0.5$) on a substrate with lamellae oriented parallel to the substrate;

2. A droplet of lamella-forming diblock copolymer ($\chi_{AB}N = 30,\ f = 0.5$) on a substrate with lamellae oriented perpendicular to the substrate;

3. A droplet of cylinder-forming diblock copolymer ($\chi_{AB}N = 30,\ f = 0.3$) on a substrate;

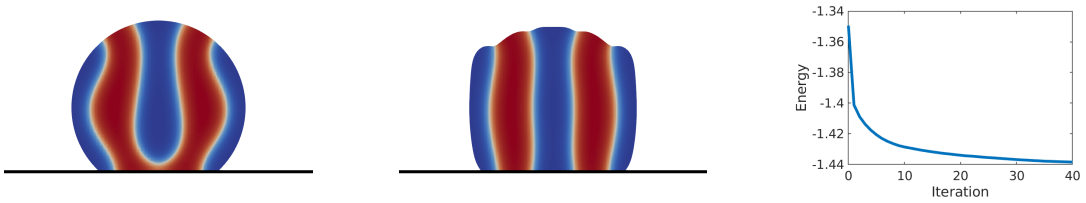
4. A droplet of cylinder-forming diblock copolymer ($\chi_{AB}N = 30,\ f = 0.3$) in a groove.



(a) (b) (c)

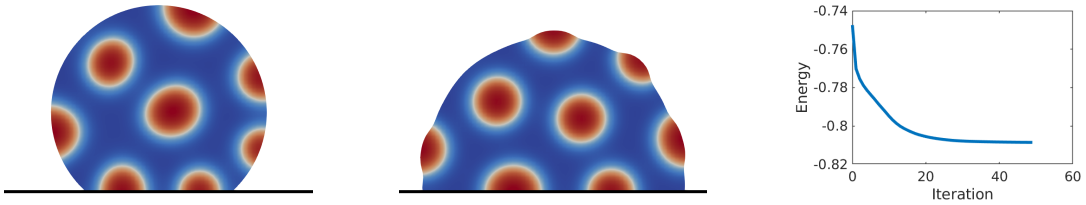
Figure 5.14: Equilibration of a droplet of a lamella-forming BCP on a substrate with lamellae oriented horizontally: (a) initial state, (b) final state, (c) energy evolution.

The simulation results for these cases are presented in figures 5.16, 5.14, 5.15, and 5.17. Specifically, they demonstrate the initial shape of a polymer droplet, the final shape, and the energy evolution. As one can see, in all cases the simulated system reaches a local energetic minimum.



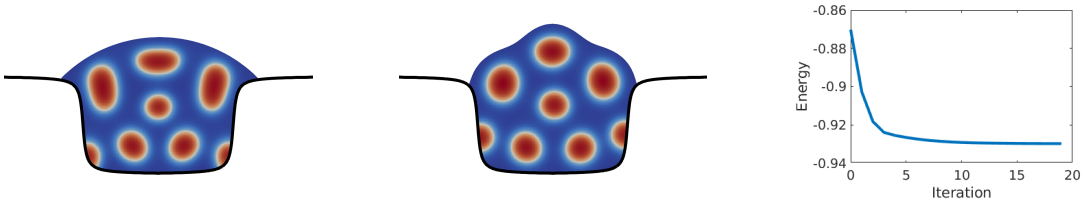
(a) (b) (c)

Figure 5.15: Equilibration of a droplet of a lamella-forming BCP on a substrate with lamellae oriented vertically: (a) initial state, (b) final state, (c) energy evolution.



(a) (b) (c)

Figure 5.16: Equilibration of a droplet of a cylinder-forming BCP on a substrate: (a) initial state, (b) final state, (c) energy evolution.



(a) (b) (c)

Figure 5.17: Equilibration of a droplet of a cylinder-forming BCP in a groove: (a) initial state, (b) final state, (c) energy evolution.

Meniscus formation in graphoepitaxy applications

Finally, we apply the presented method to study the meniscus formation in graphoepitaxy applications of block copolymers [76, 77]. Specifically, we consider vertically oriented lamellar-forming and horizontally oriented cylindrical-forming block copolymers in grooves and analyze the influence of the polymer-air surface tension as well as the value

of the contact angle on the resulting self-assembling structures of polymer material. The parameters of the problem are chosen in the following way: the width of the groove is $20R_g$, the height of the unperturbed film is $5R_g$, the interaction strength between A and B polymer species is $\chi_{AB}N = 30$, the fraction of the A component is $f = 0.5$ and $f = 0.3$ in the case of lamellae- and cylinder-forming diblock copolymers, respectively. For simplicity, the surface tensions between the polymer species and the groove's walls are assumed to be equal (i.e., no preferential attraction).

Figures 5.18 and 5.19 demonstrate the stable shapes of the polymer films obtained in cases when the contact angle formed by the polymer material with the groove's wall is 60^o, 90^o, or 120^o and for values of polymer-air surface tensions corresponding to $\chi_{ap}N = 1$, 9, 25, and 100.

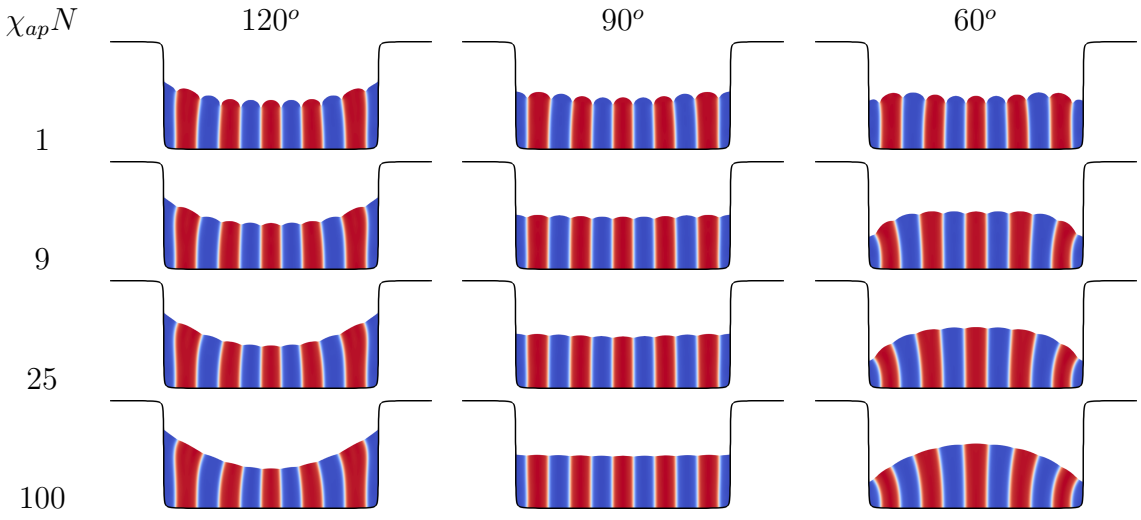


Figure 5.18: Equilibrium shapes of a lamellae-forming diblock copolymer film of height $4R_g$ in a groove of width $20R_g$ for several values of the polymer-air surface tension and contact angle.

These equilibrium shapes can be qualitatively explained in terms of the balance be-

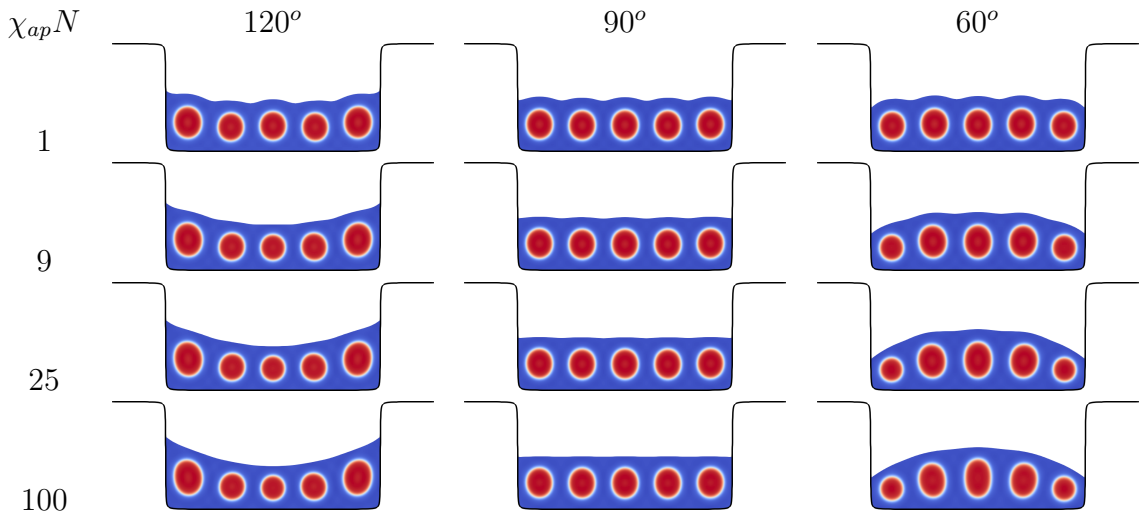


Figure 5.19: Equilibrium shapes of a cylinder-forming diblock copolymer film of height $4R_g$ in a groove of width $20R_g$ for several values of the polymer-air surface tension and contact angle.

tween the internal polymer stiffness and the polymer-air surface tension. In cases when the polymer-air surface tension $\chi_{ap}N = 1$ is much lower than the interaction strength between the polymer components $\chi_{AB}N = 30$, the overall film shape remains almost flat with the curvature caused by the meniscus propagating only a very small distance away from the walls. Note that the low surface tension also allows a small-scale corrugation of polymer-air interface due to the internal film morphology. In the other limit when the polymer-air surface tension $\chi_{ap}N = 100$ is much higher than the polymer interaction strength $\chi_{AB}N = 30$, the overall film shapes are very close to the spherical ones expected for simple liquids with only a slight small-scale corrugations. Note that in this case, the polymer domains next to groove's walls also undergo significant deformations. Finally, in intermediate cases $\chi_{ap}N = 9$ and $\chi_{ap}N = 25$, which are comparable to $\chi_{AB}N = 30$, the curvature due to the meniscus formation propagates a significant distance away from

the walls, while the center region of the film is still flat.

It is also interesting to see that the degree of the film deformation is slightly higher in the case of lamellae-forming diblock copolymer as demonstrated in figure 5.20. This indicates that the stiffness of the cylindrical morphology is higher and the deformation of a cylindrical arrangement of polymer chains causes a higher energetic penalty compared to the deformation of the lamellar arrangements.

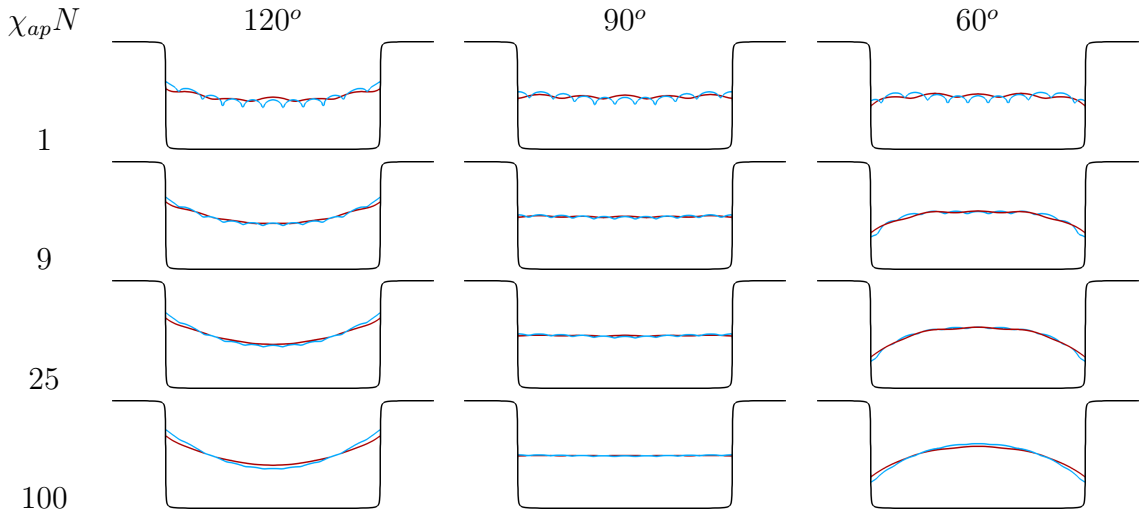


Figure 5.20: Comparison of the equilibrium shapes of lamellae-forming (blue line) and cylinder-forming (red line) diblock copolymer films of height $4R_g$ in a groove of width $20R_g$ for several values of the polymer-air surface tension and contact angle.

To summarize, the result presented in this example demonstrate that the overall equilibrium shapes of copolymer films in guiding grooves are defined by the balance between the internal stiffness of polymer morphologies and the polymer-air surface tension. These results indicate that in graphoepitaxy applications of block copolymer, the surface energies of guiding templates must be carefully selected to prevent significant deformations of the self-assembling films.

5.7.3 Block copolymer nanocomposites

Validation test

Similarly to the case of block copolymers with a free surface, we validate the expression for the energy sensitivity to particles' positions and orientations by considering a synthetic example with imposed velocity, which allows us to compare the predicted and the actual changes in energy. Specifically, we consider three particles – a disk, a rod, and a star – rotating in a circular trajectory and spinning around their center of mass as illustrated in figure 5.21. Note that the rod has a uniform surface energy, while the disk and the star have variable surface energies along their boundaries. Figure 5.22 compares the predicted and actual values of the energy change for each iteration and confirms the validity of the derived expressions.

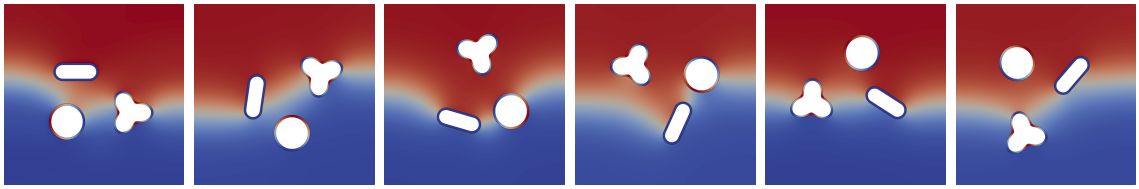


Figure 5.21: Visualization of imposed motion in example 5.7.3.

Placement of Janus rods in lamellae-forming BCP

We now apply the proposed computational approach for investigating the co-assembly of lamellar diblock copolymer ($\chi_{AB}N = 30$, $f = 0.5$) with rod-like Janus particles, that is, particles that are attracted to the A polymer component on the one end and to the the B polymer component on the other end. On the one hand, the elongated shape

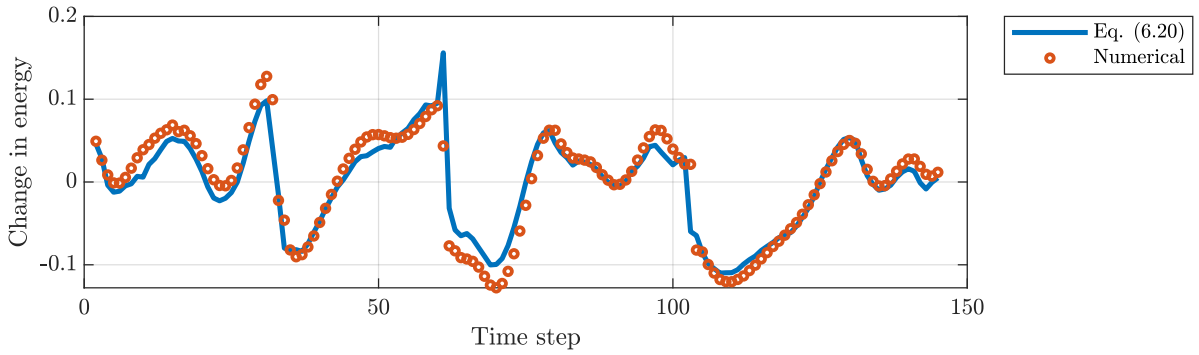


Figure 5.22: Comparison between changes in energy computed using (5.20) and numerically.

of such particles promotes the orientation parallel to the AB interface to reduce the system's energy, while the selective attraction on opposite sides of such particles favors the perpendicular orientation. As the result, one can expect the resulting placement of the nanoparticle to be a function of its length and the magnitude of its surface energy. Another factor expected to affect the particle placement is the particles density because the crowding may result in insufficient space for nanoparticles to assume their equilibrium orientation.

In numerical simulations, we consider particles of capsule-like shapes with radius of curvature $r_0 = 0.1R_g$ and nonuniform surface energies given by:

$$\gamma_{Ap} = \frac{1}{\sigma}\frac{1}{2}\sqrt{\chi_{ap}N}\,(1+\cos(\theta))\,,$$
$$\gamma_{Bp} = \frac{1}{\sigma}\frac{1}{2}\sqrt{\chi_{ap}N}\,(1-\cos(\theta))\,,$$

where θ is the angle of rotation from the particle's axis and $\chi_{ap}N$ characterizes the strength of preferential attraction between the nanoparticle and polymer species and, for convenience, is also referred to as *the polarization strength*.

Figure 5.23 depicts a representative simulation result for the co-assembly of lamellae-forming diblock copolymer ($\chi_{AB}N = 30$, $f = 0.5$) with Janus nanorods of length $0.4R_g$ in a computational domain of dimensions $3R_g \times 3R_g$. One observes that, while a small number of nanoparticles are trapped inside of A or B domains the majority of nanorods tend to aggregate at the interface between A and B polymer components and form a certain angle with the interface. In order to investigate this co-assembly behavior in

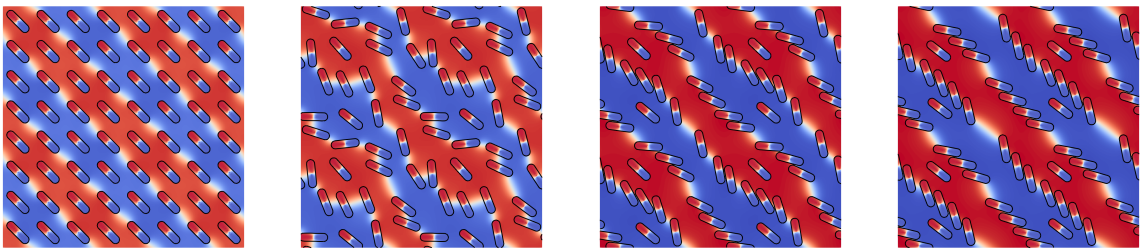


Figure 5.23: Representative simulation results for co-assembly of a lamellar diblock copolymer ($\chi_{AB}N = 30$, $f = 0.5$) and Janus nanorods of length $0.4R_g$ (periodically tiled for visualization purposes).

greater detail, we first consider a single nanorod in a smaller rectangular domain with dimension $1.8 \times 1.8R_g$, which approximately corresponds to the single diblock lamellar spacing. Periodic boundary conditions are imposed in the x-direction and homogeneous Neumann boundary conditions are imposed in the y-direction. Figure 5.24 depicts the equilibrium placements of nanorods of different lengths and for several values of the polarization strength $\chi_{ap}N$, while figure 5.25 summarizes the quantitative values for the equilibrium angles. It is clear from these results that the stable orientation of an isolated Janus nanorod in a lamellar diblock copolymer with respect to the interface between polymer components is a smooth function of the particle's length and of the polarization

strength.

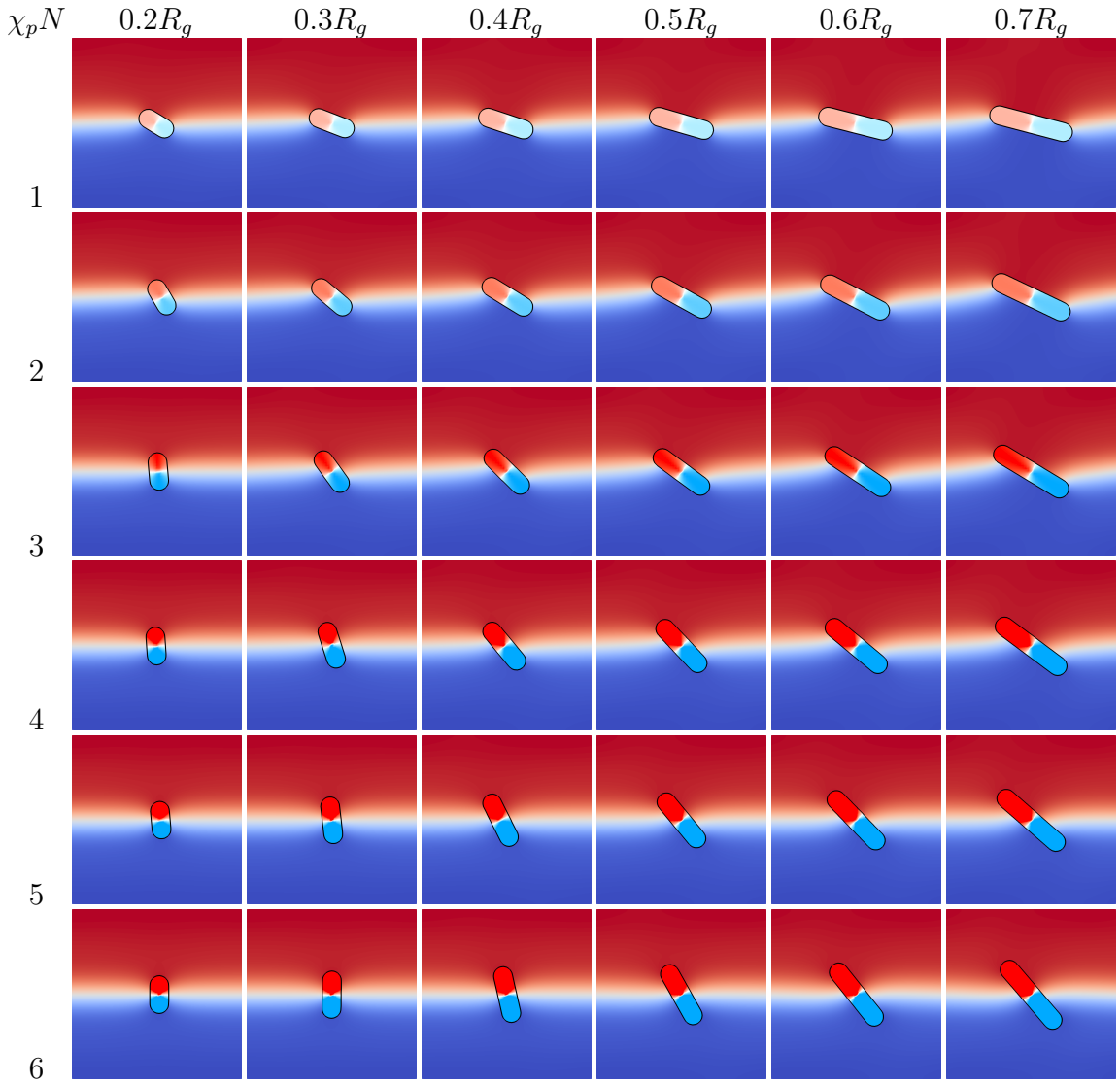


Figure 5.24: Equilibrium orientation of a Janus nanorod at the interface between A and B blocks obtained for different values of the particle's lengths and of the polarization strength.

Next, we perform similar simulations but this time with different numbers of nanoparticles inside the computational domain. Figures 5.26 illustrate the obtained particles configurations and figure 5.27 presents the quantitative results describing these configurations. One can observe that, as long as the density of particles is low (1-3 particles)

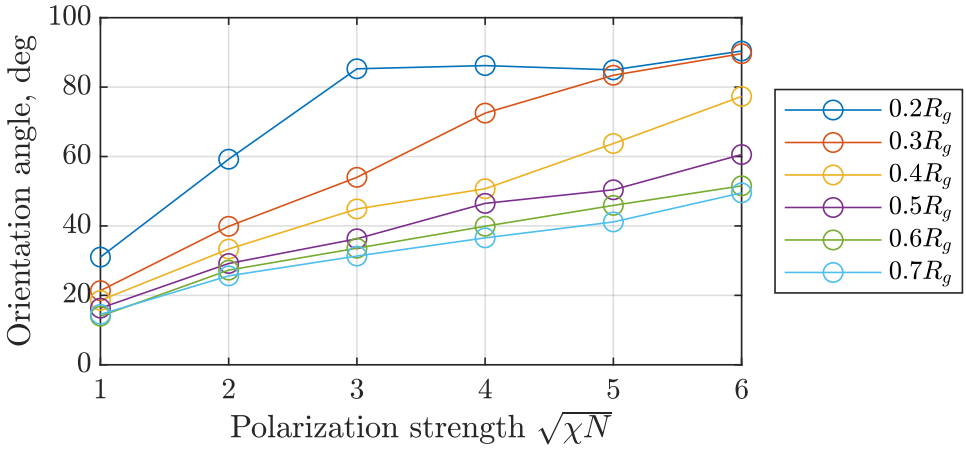


Figure 5.25: Dependence of the Janus nanorod orientation on the polarization strength for several values of the particle's length.

and no crowding occurs, the equilibrium orientations of nanorods are almost identical to those of a isolated nanorod. As the density of particles is increased, the equilibrium orientations of nanorods dramatically changes due to crowding effects. First, the equilibrium orientation angle increases for each polarization strength value as the particle density increases. Second, as the polarization strength decreases to low values the equilibrium orientation converges to a minimal nonzero value, which is dictated by the geometrical constraint due to crowding, and, thus, the accessible range of the particles' orientation is reduced.

To summarize the findings from numerical simulations of the co-assembly of Janus nanorods in a lamellar-forming diblock copolymer, we observed an aggregation of nanoparticles along interfaces between dissimilar polymer species oriented at a well defined angle with respect to these interfaces. The angle value of the equilibrium orientation continuously depend on the nanorod's length and on the polarization strength. High particle densities influence the equilibrium placement of nanorods due to the crowding effect,

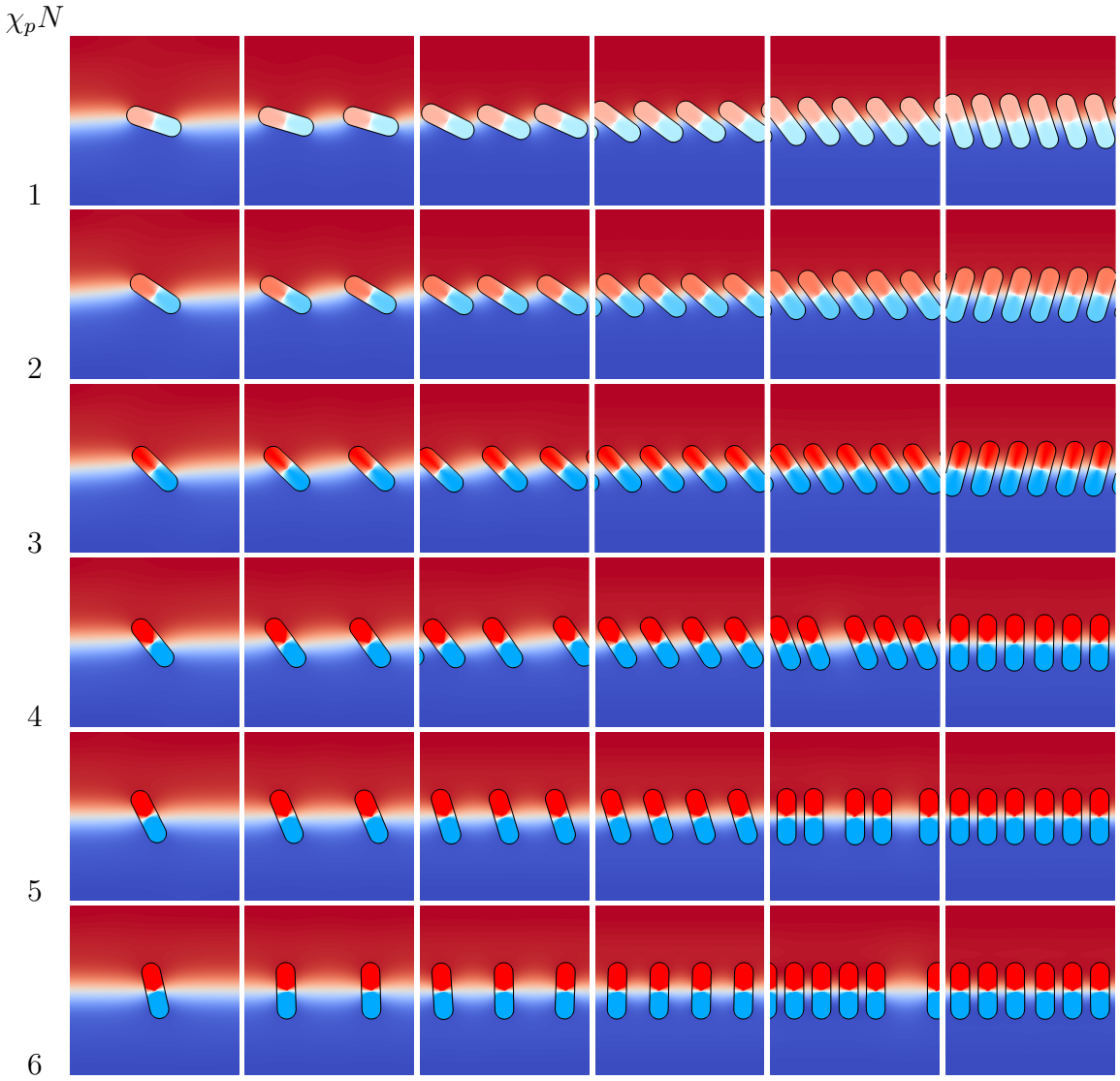


Figure 5.26: Equilibrium orientation of Janus nanorods at the interface between A and B blocks obtained for different values of the particles density and the polarization strength.

limiting the minimum achievable angle values.

5.7.4 Inverse Design for Directed Self-Assembly

In all the numerical examples of this section, we consider a cylinder-forming diblock copolymer described by parameters $f = 0.3$ and $\chi_{AB}N = 30$. All target patterns consist

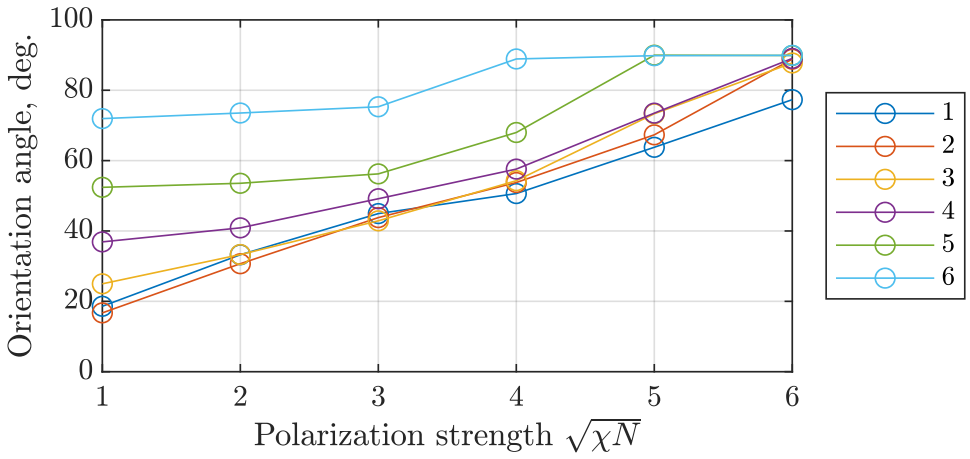


Figure 5.27: Dependence of the Janus nanorod orientation on the polarization strength for several values of the particles density.

of cylindrical domains and the target field μ_t of each cylindrical domain is taken as:

$$\mu_t = \frac{1}{2}\chi_{AB}N\text{sgn}(\phi)\left(\exp\left(-|\phi|\sqrt{\chi_{AB}N}\right)-1\right),$$

where

$$\phi(x,y) = \sqrt{(x-x_c)^2+(y-y_c)^2} - r_0,$$

and r_0 and (x_c, y_c) are the target domain's radius and center, respectively.

Validation test

Before presenting some practical applications of the proposed algorithm for the inverse design problem in DSA, we first validate it on a synthetic test. Specifically, we consider a circular confining domain of radius $3.5R_g$, a circular target domain of radius $1.0R_g$, and impose deformation velocities according to (5.22). Figure 5.28 illustrate the self-assembly under an imposed deformation of the surface. Figure 5.29 shows the comparison between

the change in the cost functional calculated directly and as predicted by the analytical expression (5.21). The close match between the actual change and the predicted one illustrates the accuracy of the proposed approach.

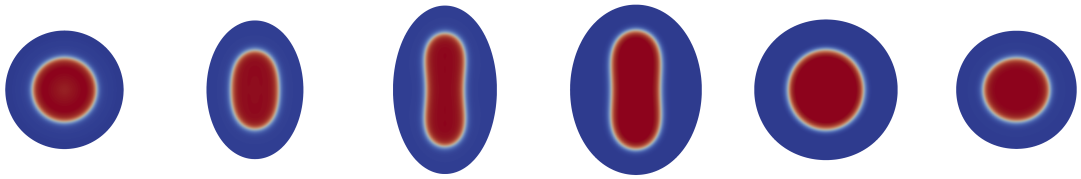


Figure 5.28: Visualization of imposed motion in example 5.7.4.

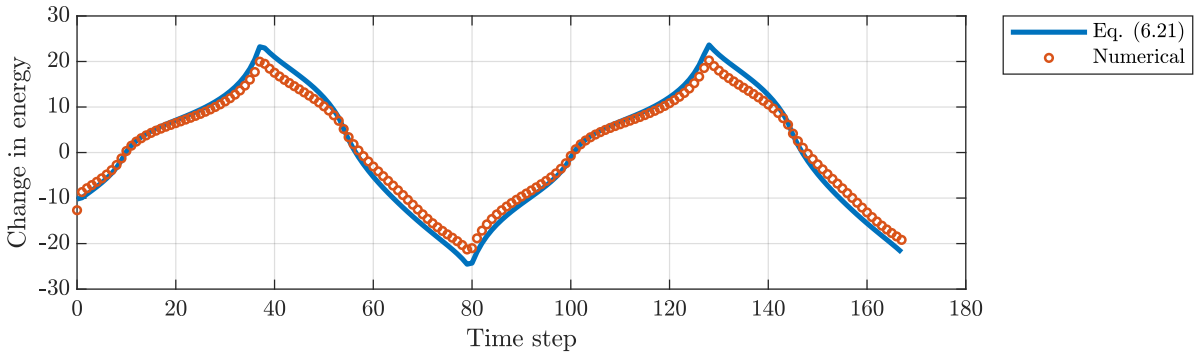


Figure 5.29: Comparison between changes in energy computed using (5.21) and numerically.

Influence of domain size and spacing

We start with a simple case of designing confining masks for the placement of two cylindrical domains formed by the minority component. Specifically, we investigate the influence of the domains' size and spacing on the resulting mask shapes. Figure 5.30 illustrates mask shapes obtained using the proposed optimization algorithm for cylindrical radii ranging from $r_0 = 0.9R_g$ to $r_0 = 1.2R_g$ and domain spacings ranging from $\Delta r = 2.75R_g$ to $\Delta r = 4R_g$, where R_g is the radius of gyration of the block copolymer chains. The

value of $\alpha = 0.1$ is used for curvature penalization. The results indicate that the proposed

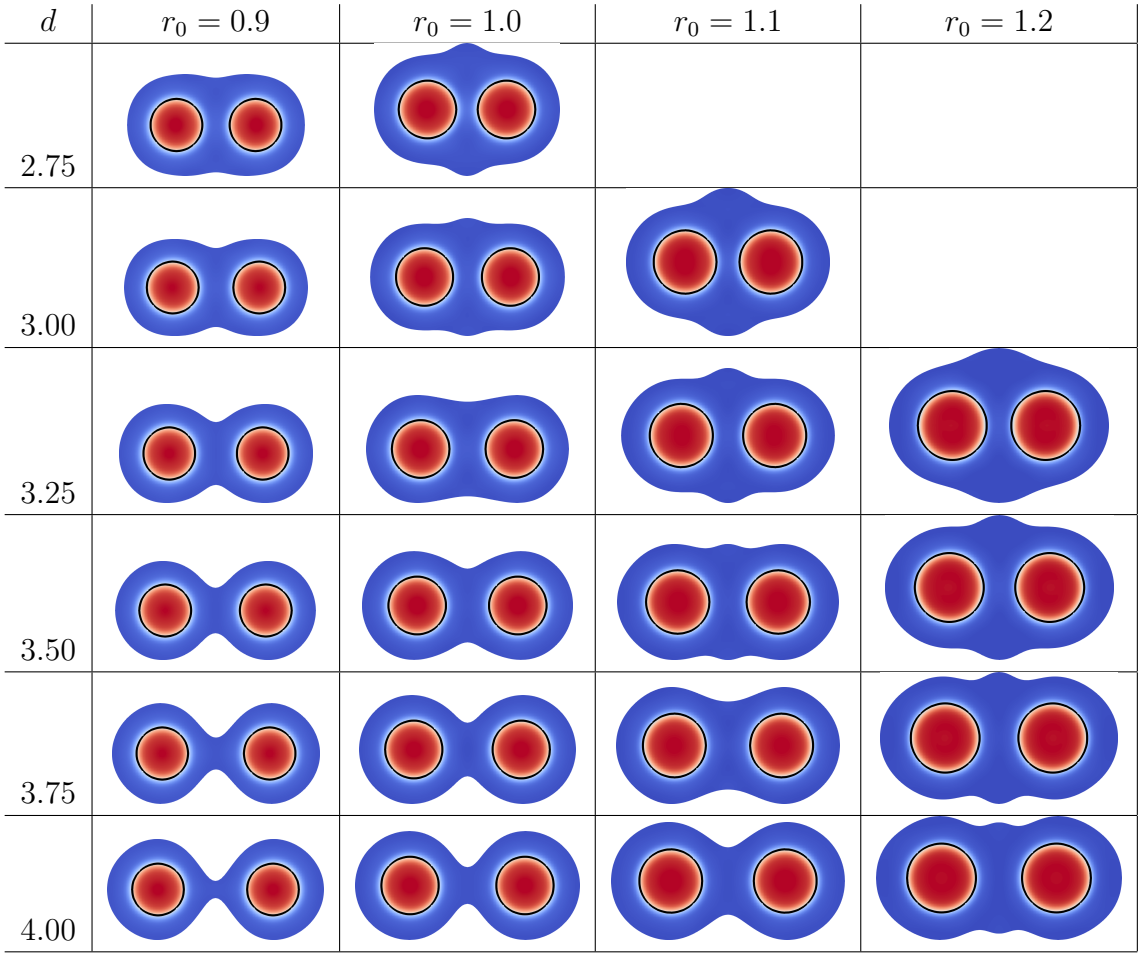
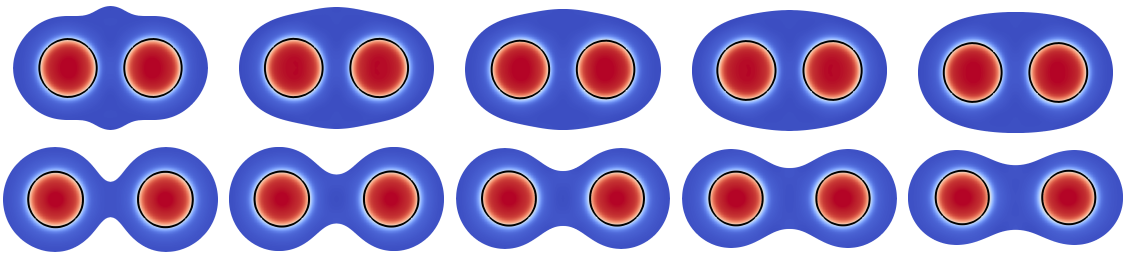


Figure 5.30: Confining masks for placement of two cylinders for different design parameters. Coloring shows the density configuration of the self-assembled polymer while the solid black line represent the target template.

method is successful for almost every case considered and produces confining geometries that results not only in accurate placement of domain centers but their overall shapes as well. In some cases, the algorithm is enable to find a mask geometry that will guide the self-assembly towards the target design. However, those cases consider small well distances and/or large cylinders radii, which forces the cylindrical polymer domains to be placed too close to each other and eventually merge. Thus, it should be interpreted

as the non-existence of solution in these cases rather than the deficiency of the method.

One can notice that depending on the target pattern geometry, the resulting confining masks can have either concave or convex features to accommodate for the polymer chains jammed between cylindrical domains. In either cases, such features have scales smaller than the target pattern, which makes such confining masks of a little practical value. However, we now show how the mask's smoothness can be controlled through adjusting the curvature penalization parameter α. Specifically, we consider a case with concave features ($d = 4R_g$, $r_0 = R_g$) and a case with convex features ($d = 3R_g$, $r_0 = 1.1R_g$). Figure 5.31 demonstrates the resulting confining mask in the cases where the curvature penalization parameter take the values 0.1, 0.2, 0.4, 0.8, and 1.6. As one can see, the



(a) $\alpha = 0.1$ (b) $\alpha = 0.2$ (c) $\alpha = 0.4$ (d) $\alpha = 0.8$ (e) $\alpha = 1.6$

Figure 5.31: Confining masks for two cylindrical domains placed $3R_g$ (top row) and $4R_g$ (bottom row) apart and varying the parameter α controlling the smoothness of the mask. Coloring shows the density configuration of the self-assembled polymer while the solid black line represent the target template.

proposed approach performs well under curvature constraints producing confining masks of desired smoothness that, at the same time, guide the polymer self-assembly very precisely to its target design.

Influence of relative orientation

Now we turn our attention to designing confining masks for the placement of a line of five cylinders making a turn at a specified angle. We consider angles ranging from $\theta = 60^o$ to $\theta = 150^o$. In addition, we investigate several cylinder spacing values ranging from $\Delta r = 3R_g$ to $\Delta r = 4R_g$. Figure 5.30 illustrates the resulting confining masks. As one can see,

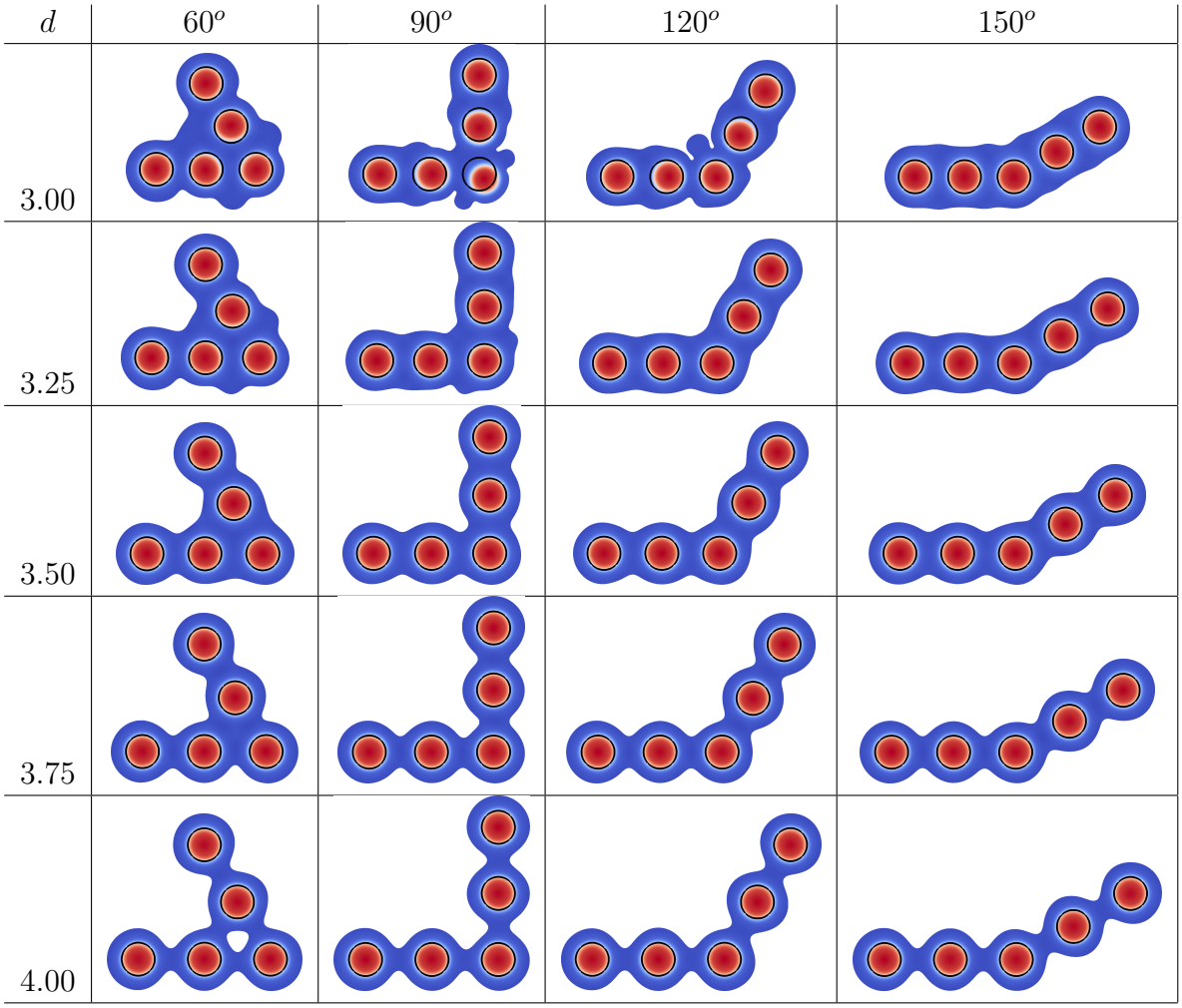


Figure 5.32: Confining mask for the placement of a line of cylinders making a specified turn. Coloring shows the density configuration of the self-assembled polymer domains while the solid black line represent the target template.

the proposed algorithm was able to find confining masks that result in very close matches

between actual polymer morphologies and the desired ones, except for the case $\theta = 90^o$ and domain spacing $\Delta r = 3R_g$. However, we do not interpret this as a deficiency of the numerical method but as a more fundamental incommensurability between the desired template and the intrinsic properties of the polymer material considered. Indeed, in the cases of angles $\theta = 60^o$ and $\theta = 120^o$, which are more commensurate with the intrinsically favorable hexagonally packed polymer morphology, the inverse design is successful.

Confining mask for more complex templates

Finally, in order to demonstrate the robustness and the flexibility of the proposed optimization algorithm, we apply it to the design of confining masks for more complex templates. Specifically, we consider the C, A, S, L, U, and B shaped patterns guiding 6, 9, 7, 6, 8, and 9 cylindrical domain, correspondingly. In all cases, the characteristic distance between the cylindrical domains and their size are chosen to be $\Delta r = 3.5R_g$ and $r_0 = R_g$. Figure 5.33 illustrates the output of the optimization algorithm and demonstrates that successful and non-trivial confining masks are obtained for all patterns.

5.8 Conclusion

In this chapter we have presented an adjoint-state based approaches for three moving boundary problems relevant to the physics of inhomogeneous polymers: the self-assembly of free surface BCPs, the co-assembly of BCP nanocomposites, and the inverse design problem for DSA. These numerical approaches are based on exact analytical shape deriva-

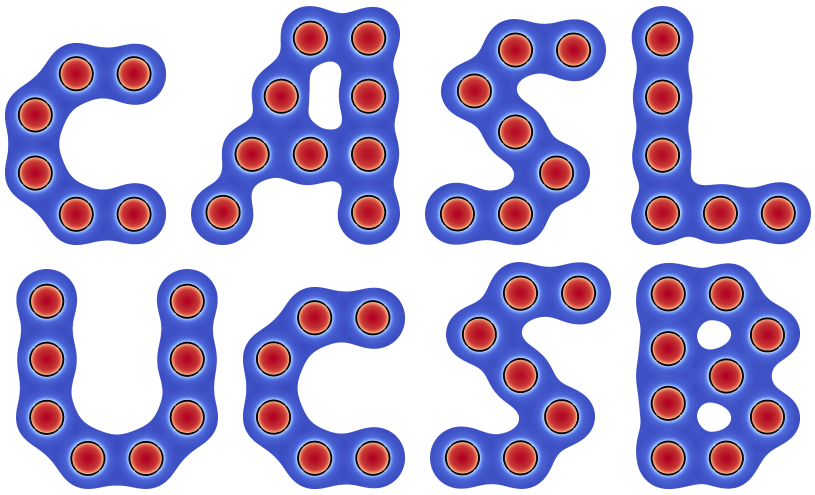


Figure 5.33: Confining masks for guiding cylindrical domains in the C, A, S, L, U, and B shaped patterns. Coloring shows the density configuration of the self-assembled polymer domains while the solid black line represent the target template.

tives of the underlying quantity for minimization (the system's energy in the first two cases, and the deviation of the actual density field from the desired one in the third case). In all cases, the polymer-air and the polymer-solid interfaces are treated in a sharp, physically meaningful fashion. The surface tensions between the polymer material and its surroundings are imposed using a novel consistent approach that results in pressure fields free of numerical singularities. All proposed methods have been validated on benchmark examples with imposed velocities and demonstrated a very close match between predicted and actual changes in the minimization quantity.

The computational method for modeling free surface BCPs was applied to the investigation of the meniscus formation in the context of graphoepitaxy applications for lamellar and cylindrical diblock copolymers. It was shown that in case of the lamella-forming copolymer the polymer-air surface tension, as well as the contact angle value at the air-polymer-wall triple junction point affect the distribution of polymer material

inside the guiding groove and also cause the bending of polymer lamellar domains. At the same time the cylinder-forming polymer morphology demonstrated less sensitivity to these parameters.

The computational method for the co-assembly of BCP nanocomposites was applied to investigate the placement and orientation of "polarized" nanorod particle in lamellar diblock copolymer. It was demonstrated that such particles aggregate at the interface between dissimilar polymer blocks and that their resulting orientation is controlled by the balance of the particles' length, the "polarization" strength, and the particles density.

Finally, the computational method for the inverse design problem for DSA was applied to obtain confining masks for placement of cylindrical domains of BCP in a number of different patterns. The method was shown to be able to produce masks that result not only in an accurate placement of cylindrical domains but accurate dimensions as well. Additionally, it was applied to cases of rather complicated patterns to demonstrate the method's robustness.

Interesting future work will include the application of the presented methods to cases of grafted polymers, polymers of more complex architectures, and blends of copolymers. Another direction of future work will be the extension of the presented methods to a more accurate fluctuating field-theoretic description of block copolymers.

5.A Weak formulations and equivalence of modified diffuion equations

To demonstrate the equivalence of (5.2) and (5.8) we invoke their weak forms. Multiplying the diffusion equation in (5.2) by an test function $\psi(s, \boldsymbol{r})$ and integrating over domain Ω and contour variable range $s \in [0, 1]$ we obtain:

$$\int_{\Omega} \int_{0}^{1} \left(\partial_s q + (\mu(s) + \sigma\gamma(s)\delta_{\Gamma})\, q - D(s)\nabla^2 q \right) \psi \, d\boldsymbol{r} \, ds = 0. \tag{5.23}$$

Using integration by parts and the divergence theorem it is straightforward to show that:

$$\begin{aligned} \int_{\Omega} \int_{0}^{1} \partial_s q \psi \, d\boldsymbol{r} \, ds &= \int_{\Omega} \partial_s \left(q\psi \right) \, d\boldsymbol{r} \, ds - \int_{\Omega} \int_{0}^{1} q \partial_s \psi \, d\boldsymbol{r} \, ds \\ &= \int_{\Omega} \left(q(1, \boldsymbol{r})\psi(1, \boldsymbol{r}) - q(0, \boldsymbol{r})\psi(0, \boldsymbol{r}) \right) \, d\boldsymbol{r} \, ds - \int_{\Omega} \int_{0}^{1} q \partial_s \psi \, d\boldsymbol{r} \, ds \end{aligned} \tag{5.24}$$

and

$$\begin{aligned} \int_{\Omega} \int_{0}^{1} D(s)\nabla^2 q \psi \, d\boldsymbol{r} \, ds &= \int_{\Omega} \int_{0}^{1} D(s) \nabla \cdot \left(\nabla q \psi \right) \, d\boldsymbol{r} \, ds - \int_{\Omega} \int_{0}^{1} D(s) \nabla q \cdot \nabla \psi \, d\boldsymbol{r} \, ds \\ &= \int_{\Gamma} \int_{0}^{1} D(s) \partial_{\boldsymbol{n}} q \psi \, d\boldsymbol{r} \, ds - \int_{\Omega} \int_{0}^{1} D(s) \nabla q \cdot \nabla \psi \, d\boldsymbol{r} \, ds. \end{aligned} \tag{5.25}$$

Using the above two expressions in (5.23) along with the initial and boundary conditions for q produces the following weak form of (5.2):

$$\int\limits_{\Omega} \left(q(1,\boldsymbol{r})\psi(1,\boldsymbol{r}) - \psi(0,\boldsymbol{r})\right)\, d\boldsymbol{r}\, ds$$

$$+ \int\limits_{\Omega}\int\limits_{0}^{1} \left(-q\partial_s\psi + \left(\mu(s) + \sigma\gamma(s)\delta_{\Gamma}\right) q\psi + D(s)\nabla q\cdot\nabla\psi\right)\, d\boldsymbol{r}\, ds = 0, \quad \forall\psi$$

Converting the term containing surface delta function δ_{Γ} into a boundary integral as

$$\int\limits_{\Omega}\int\limits_{0}^{1} \sigma\gamma(s)\delta_{\Gamma} q\psi\, d\boldsymbol{r}\, ds = \int\limits_{\Gamma}\int\limits_{0}^{1} \sigma\gamma(s) q\psi\, d\boldsymbol{r}\, ds$$

the obtained weak form can be transformed into

$$\int\limits_{\Omega} \left(q(1,\boldsymbol{r})\psi(1,\boldsymbol{r}) - \psi(0,\boldsymbol{r})\right)\, d\boldsymbol{r}\, ds$$

$$+ \int\limits_{\Omega}\int\limits_{0}^{1} \left(-q\partial_s\psi + \mu(s) q\psi + D(s)\nabla q\cdot\nabla\psi\right)\, d\boldsymbol{r}\, ds$$

$$+ \int\limits_{\Gamma}\int\limits_{0}^{1} \sigma\gamma(s) q\psi\, d\boldsymbol{r}\, ds = 0, \quad \forall\psi,$$

which represents a weak form an altered diffusion equation. Indeed, using formulas (5.24) and (5.25) in the reverse way, one can write the above expression as

$$\int\limits_{\Omega} \left(q(0,\boldsymbol{r}) - 1\right) \psi(0,\boldsymbol{r})\, d\boldsymbol{r}\, ds$$

$$+ \int\limits_{\Omega}\int\limits_{0}^{1} \left(\partial_s q + \mu(s) q - D(s)\nabla^2 q\right) \psi\, d\boldsymbol{r}\, ds$$

$$+ \int\limits_{\Gamma}\int\limits_{0}^{1} \left(D(s)\partial_{\boldsymbol{n}} q + \sigma\gamma(s) q\right) \psi\, d\boldsymbol{r}\, ds = 0, \quad \forall\psi$$

from which it immediately follows that q satisfies (5.8). The equivalence of (5.3)-(5.9), (5.8)-(5.10), and (5.9)-(5.11) can be demonstrated in the analogous fashion.

5.B Shape derivatives of integral quantities

In order to calculate derivatives of integral quantities (5.17) we invoke the Level-Set methodology [121]. Specifically, we assume, first, that domain $\Omega(\tau)$ can be represented

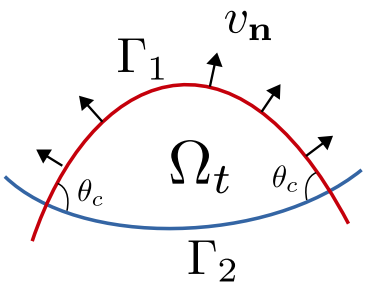


Figure 5.34: Problem geometry and notation used in section 5.B.

as the intersection of two smooth domains $\Omega_a(\tau)$ and Ω_w as demonstrated in figure 5.34, and, second, that boundaries of domains $\Omega_a(\tau)$ and Ω_w are described as zero-isocontours of higher-dimensional functions $\phi_a(\tau)$ and ϕ_w, called *level-set functions*, such that:

$$\phi_a(\tau)\begin{cases} < 0, & \boldsymbol{r} \in \Omega_a, \\ > 0, & \boldsymbol{r} \notin \Omega_a, \end{cases} \quad \text{and} \quad \phi_w(\tau)\begin{cases} < 0, & \boldsymbol{r} \in \Omega_w, \\ > 0, & \boldsymbol{r} \notin \Omega_w. \end{cases}$$

In this setting the evolution of interface $\Gamma_a(\tau)$ in the velocity field $v_{\boldsymbol{n}}(\tau)$ corresponds to the following advection equation for function $\phi_a(\tau)$:

$$\frac{\partial \phi_a}{\partial \tau} + v_{\boldsymbol{n}} \left|\nabla \phi_a\right| = 0. \tag{5.26}$$

The description of the problem geometry in this way allows to write the boundary and domain integrals using the Heaviside step functions $H(x)$, Dirac delta distribution $\delta(x)$ and the co-area formula as

$$\begin{aligned}
\int_{\Omega} f\, d\boldsymbol{r} &= \int_{\Re^{\mathcal{D}}} f H(-\phi_a) H(-\phi_w)\, d\boldsymbol{r}, \\
\int_{\Gamma_w} f\, d\boldsymbol{r} &= \int_{\Re^{\mathcal{D}}} f H(-\phi_a) \delta(\phi_w) \left|\nabla \phi_w\right|\, d\boldsymbol{r}, \\
\int_{\Gamma_a} f\, d\boldsymbol{r} &= \int_{\Re^{\mathcal{D}}} f \delta(\phi_a) \left|\nabla \phi_a\right| H(-\phi_w)\, d\boldsymbol{r},
\end{aligned}$$

where $\mathcal{D}$ denotes the problem's dimensionality. Taking the derivative of these expression with respect to τ leads to

$$\begin{aligned}
\frac{\partial}{\partial \tau}\int_{\Omega} f\, d\boldsymbol{r} &= \int_{\Re^{\mathcal{D}}} \frac{\partial f}{\partial \tau} H(-\phi_a) H(-\phi_w)\, d\boldsymbol{r} - \int_{\Re^{\mathcal{D}}} f \delta(-\phi_a) \frac{\partial \phi_a}{\partial \tau} H(-\phi_w)\, d\boldsymbol{r}, \\
\frac{\partial}{\partial \tau}\int_{\Gamma_w} f\, d\boldsymbol{r} &= \int_{\Re^{\mathcal{D}}} \frac{\partial f}{\partial \tau} H(-\phi_a) \delta(\phi_w) \left|\nabla \phi_w\right|\, d\boldsymbol{r} - \int_{\Re^{\mathcal{D}}} f \delta(-\phi_a) \frac{\partial \phi_a}{\partial \tau} \delta(\phi_w) \left|\nabla \phi_w\right|\, d\boldsymbol{r}, \\
\frac{\partial}{\partial \tau}\int_{\Gamma_a} f\, d\boldsymbol{r} &= \int_{\Re^{\mathcal{D}}} \frac{\partial f}{\partial \tau} \delta(\phi_a) \left|\nabla \phi_a\right| H(-\phi_w)\, d\boldsymbol{r} + \int_{\Re^{\mathcal{D}}} f \delta'(\phi_a) \frac{\partial \phi_a}{\partial \tau} \left|\nabla \phi_a\right| H(-\phi_w)\, d\boldsymbol{r} \\
&\quad + \int_{\Re^{\mathcal{D}}} f \delta(\phi_a) \frac{\nabla \phi_a}{|\nabla \phi_a|} \cdot \nabla \frac{\partial \phi_a}{\partial \tau} H(-\phi_w)\, d\boldsymbol{r},
\end{aligned} \tag{5.27}$$

Applying integration by parts to the last term in the third expression it can be shown that

$$\int_{\Re^{\mathcal{D}}} f\delta(\phi_a)\frac{\nabla\phi_a}{|\nabla\phi_a|}\cdot\nabla\frac{\partial\phi_a}{\partial\tau}H(-\phi_w)\,d\boldsymbol{r}$$

$$= -\int_{\Re^{\mathcal{D}}} \frac{\partial\phi_a}{\partial\tau}\nabla\cdot\left(f\delta(\phi_a)\frac{\nabla\phi_a}{|\nabla\phi_a|}H(-\phi_w)\right)\,d\boldsymbol{r}$$

$$= -\int_{\Re^{\mathcal{D}}} \frac{\partial\phi_a}{\partial\tau}\Bigg(\nabla f\cdot\frac{\nabla\phi_a}{|\nabla\phi_a|}\delta(\phi_a)H(-\phi_w) + f\delta'(\phi_a)\,|\nabla\phi_a|\,H(-\phi_w)$$

$$+ f\delta(\phi_a)\nabla\cdot\left(\frac{\nabla\phi_a}{|\nabla\phi_a|}\right)H(-\phi_w) - f\delta(\phi_a)\delta(-\phi_w)\nabla\phi_w\cdot\frac{\nabla\phi_a}{|\nabla\phi_a|}\Bigg)\,d\boldsymbol{r}.$$

Substituting the above expression into (5.27), using advection equation (5.26) and the fact that the normal vectors and curavtures of Γ_a and Γ_b are equal to

$$\boldsymbol{n}_a = \frac{\nabla\phi_a}{|\nabla\phi_a|}, \quad \kappa_a = \nabla\cdot\left(\frac{\nabla\phi_a}{|\nabla\phi_a|}\right),$$
$$\boldsymbol{n}_w = \frac{\nabla\phi_w}{|\nabla\phi_w|}, \quad \kappa_w = \nabla\cdot\left(\frac{\nabla\phi_w}{|\nabla\phi_w|}\right),$$

results in the following formulas

$$\frac{\partial}{\partial\tau}\int\limits_{\Omega} f\,d\boldsymbol{r} = \int\limits_{\Re^{\mathcal{D}}} \frac{\partial f}{\partial\tau} H(-\phi_a)H(-\phi_w)\,d\boldsymbol{r} + \int\limits_{\Re^{\mathcal{D}}} f v_{\boldsymbol{n}}\delta(\phi_a)\,|\nabla\phi_a|\,H(-\phi_w)\,d\boldsymbol{r},$$

$$\begin{aligned}\frac{\partial}{\partial\tau}\int\limits_{\Gamma_w} f\,d\boldsymbol{r} &= \int\limits_{\Re^{\mathcal{D}}} \frac{\partial f}{\partial\tau} H(-\phi_a)\delta(\phi_w)\,|\nabla\phi_w|\;d\boldsymbol{r} \\ &\quad + \int\limits_{\Re^{\mathcal{D}}} f v_{\boldsymbol{n}}\delta(\phi_a)\,|\nabla\phi_a|\,\delta(\phi_w)\,|\nabla\phi_w|\;d\boldsymbol{r},\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\tau}\int\limits_{\Gamma_a} f\,d\boldsymbol{r} &= \int\limits_{\Re^{\mathcal{D}}} \frac{\partial f}{\partial\tau}\delta(\phi_a)\,|\nabla\phi_a|\,H(-\phi_w)\,d\boldsymbol{r} \\ &\quad + \int\limits_{\Re^{\mathcal{D}}} v_{\boldsymbol{n}}\left(\kappa f + \partial_{\boldsymbol{n}} f\right)\delta(\phi_a)\,|\nabla\phi_a|\,H(-\phi_w)\,d\boldsymbol{r} \\ &\quad - \int\limits_{\Re^{\mathcal{D}}} f v_{\boldsymbol{n}}\delta(\phi_a)\,|\nabla\phi_a|\,\delta(\phi_w)\,|\nabla\phi_w|\,\boldsymbol{n}_a\cdot\boldsymbol{n}_w\,d\boldsymbol{r},\end{aligned}$$

which after applying the coarea formulas and converting integrals into more traditional notation become (5.18). Note that these expressions contain integrals over intersection of Γ_a and Γ_w. These terms can be interpreted as additional surface generation as Γ_a travels along Γ_w (see figure 5.35).

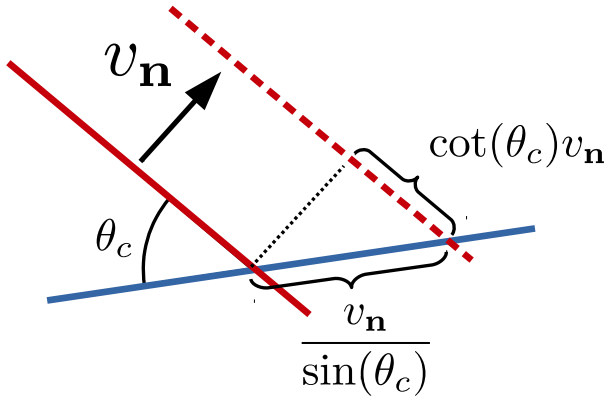


Figure 5.35: Interpretation of resulting expression for shape derivatives in case of sharp features.

Bibliography

[1] L. Adams and T. Chartier. New geometric immersed interface multigrid solvers. *SIAM J. Sci. Comput.*, 25:1516–1533, 2004.

[2] L. Adams and T. Chartier. A comparison of algebraic multigrid and geometric immersed interface multigrid methods for interface problems. *SIAM J. Sci. Comput.*, 26:762–784, 2005.

[3] L. Adams and Z. Li. The immersed interface/multigrid methods for interface problems. *SIAM J. Sci. Comput.*, 24(2):463–479, 2002.

[4] Grégoire Allaire, François Jouve, and Anca-Maria Toader. Structural optimization using sensitivity analysis and a level-set mehtod. *J. Comput. Phys.*, 194(1):363–393, 2004.

[5] Victoria Arias, Daniil Bochkov, and Frederic Gibou. Poisson equations in irregular domains with Robin boundary conditions - solver with second-order accurate gradients. *J. Comput. Phys.*, 365:1–6, 2018.

[6] Tariq Aslam. A partial differential equation approach to multidimensional extrapolation. *J. Comput. Phys.*, 193(1):349–355, 2004.

[7] Tariq Aslam, Songting Luo, and Hongkai Zhao. A static pde approach for multidimensional extrapolation using fast sweeping methods. *SIAM J. Sci. Comput.*, 36(6):A2907–A2928, 2014.

[8] Ivo Babuška. The finite element method for elliptic equations with discontinuous coefficients. *Computing*, 5:207–213, 1970.

[9] Satish Balay, Jed Brown, , Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. *PETSc Users Manual*. Argonne National Laboratory, 2012.

[10] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. Petsc web page, 2012.

[11] Zhaosheng Bao, Jeong-Mo Hong, Joseph Teran, and Ronald Fedkiw. Fracturing rigid materials. *IEEE Trans. Vis. Comput. Graph*, 13:370–378, 2007.

[12] John W. Barrett, Harald Garcke, and Robert Nürnberg. On stable parametric finite element methods for the Stefan problem and the Mullins–Sekerka problem with applications to dendritic growth. *J. Comput. Phys.*, 229(18):6270 – 6299, 2010.

[13] Ted Belytschko, Nicolas Moës, Shuji Usui, and Chandu Parimi. Arbitrary discontinuities in finite elements. *Int. J. Numer. Methods Eng.*, 50:993–1013, 2001.

[14] P. A. Berthelsen. A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions. *J. Comput. Phys.*, 197:364–386, 2004.

[15] Wurigen Bo, Xingtao Liu, James Glimm, and Xiaolin Li. A robust front tracking method: Verification and application to simulation of the primary breakup of a liquid jet. *SIAM J. Sci. Comput.*, 33(4):1505–1524, 2011.

[16] Daniil Bochkov and Frederic Gibou. Pde-based multidimensional extrapolation of scalar fields over interfaces with kinks and high curvatures. *Accepted in SIAM J. Sci. Comput.*, 2019.

[17] Daniil Bochkov and Frederic Gibou. A sharp computational method for the simulation of the solidification of multicomponent alloys. *In Preparation*, 2019.

[18] Daniil Bochkov and Frederic Gibou. Solving poisson-type equations with robin boundary conditions on piecewise smooth interfaces. *J. Comput. Phys.*, 376:1156–1198, 2019.

[19] Daniil Bochkov and Frederic Gibou. Solving elliptic interface problems with jump conditions on cartesian grids. *J. Comput. Phys.*, 407:109269, 2020.

[20] Michael R Bockstaller, Rafal A Mickiewicz, and Edwin L Thomas. Block copolymer nanocomposites: perspectives for tailored functional materials. *Advanced materials*, 17(11):1331–1349, 2005.

[21] François Bouchon and Gunther H. Peichl. A second-order immersed interface technique for an elliptic Neumann problem. *Numer. Meth. Part. D. E.*, 23(2):400–420, mar 2007.

[22] J. H. Bramble and B. E. Hubbard. Approximation of solutions of mixed boundary value problems for Poisson's equation by finite differences. *J. ACM*, 12(1):114–123, 1965.

[23] Christopher E. Brennen. *Fundamentals of Multiphase Flows*, volume ISBN 0521 848040. Cambridge University Press, 2005.

[24] Carsten Burstedde, Lucas C Wilcox, and Omar Ghattas. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM J. Sci. Comput.*, 33(3):1103–1133, 2011.

[25] Corinne L. Carpenter, Kris T. Delaney, Nabil Laachi, and Glenn H. Fredrickson. Directed self-assembly of diblock copolymers in cylindrical confinement: effect of underfilling and air-polymer interactions on configurations. *SPIE Advanced Lithography*, 9423:94231Z, 2015.

[26] Adam Chacon and Alexander Vladimirsky. A parallel two-scale method for Eikonal equations. *SIAM J. Sci. Comput.*, 37(1):A156–A180, 2015.

[27] Han Chen, Chohong Min, and Frederic 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.*, 31:19–60, 2007.

[28] Han Chen, Chohong Min, and Frederic Gibou. A numerical scheme for the Stefan problem on adaptive Cartesian grids with supralinear convergence rate. *J. Comput. Phys.*, 228(16):5803–5818, 2009.

[29] Susan Chen, Barry Merriman, Stanley Osher, and Peter Smereka. A simple level set method for solving Stefan problems. *J. Comput. Phys.*, 135:8–29, 1997.

[30] T. Chen and J. Strain. Piecewise-polynomial discretization and krylov-accelerated multigrid for elliptic interface problems. *J. Comput. Phys.*, 227(16):7503–7542, 2008.

[31] Marco Cisternino and Lisl Weynans. A parallel second order cartesian method for elliptic interface problems. *Comm. Comput. Phys.*, 12:1562–1587, 2012.

[32] A. Coco and G. Russo. Second order multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface, i: One dimensional problems. *Numer. Math.-Theory Me.*, 5:19, 2012.

[33] Armando Coco and Giovanni Russo. Finite-difference ghost-point multigrid methods on Cartesian grids for elliptic problems in arbitrary domains. *J. Comput. Phys.*, 241:464–501, 2013.

[34] R.K. Crockett, P. Colella, and D.T. Graves. A cartesian grid embedded boundary method for solving the poisson and heat equations with discontinuous coefficients in three dimensions. *J. Comput. Phys.*, 230(7):2451 – 2469, 2011.

[35] Andrew B. Croll, Michael V. Massa, Mark W. Matsen, and Kari Dalnoki-Veress. Droplet shape of an anisotropic liquid. *Physical Review Letters*, 97(November):204502, 2006.

[36] Christophe Daux, Nicolas Moës, John Dolbow, Natarajan Sukumar, and Ted Belytschko. Arbitrary branched and intersecting cracks with the extended finite element method. *Int. J. Numer. Methods Eng.*, 48:1741–1760, 2000.

[37] Stephen H Davis. *Theory of solidification*. Cambridge University Press, 2001.

[38] Charles Cleret de Langavant, Arthur Guittet, Maxime Theillard, Fernando Temprano-Coleto, and Frédéric Gibou. Level-set simulations of soluble surfactant driven flows. *J. Comput. Phys.*, 348:271 – 297, 2017.

[39] Miles Detrixhe and Frédéric Gibou. Hybrid Massively Parallel Fast Sweeping Method for static Hamilton-Jacobi Equations. *J. Comput. Phys.*, 322:199–223, 2016.

[40] Miles Detrixhe, Frédéric Gibou, and Chohong Min. A parallel fast sweeping method for the eikonal equation. *J. Comput. Phys.*, 237:46–55, 2013.

[41] Dharshi Devendran, Daniel Graves, Hans Johansen, and Terry Ligocki. A fourth-order cartesian grid embedded boundary method for poissons equation. *Comm. App. Math. Com. Sc.*, 12(1):51–79, 2017.

[42] Raphael Egan and Frédéric Gibou. Fast and scalable algorithms for constructing solvent-excluded surfaces of large biomolecules. *J. Comput. Phys.*, 374:91 – 120, 2018.

[43] B Engquist, A K Tornberg, and R Tsai. Discretization of Dirac delta functions in level set methods. *J. Comput. Phys.*, 207:28–51, 2005.

[44] Doug Enright, Duc Nguyen, Frederic Gibou, and Ron Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proceedings of the ASME/JSME 2003 4th Joint Fluids Summer Engineering Conference. Volume 2: Symposia, Parts A, B, and C.*, volume 36975, pages 337–342, 2003.

[45] R. D. Falgout and U. M. Yang. *Hypre: A library of high performance preconditioners*, volume 2331. Springer Berlin Heidelberg, 2002.

[46] Ronald P Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152(2):457–492, 1999.

[47] Glenn H. Fredrickson. *The equilibrium theory of inhomogeneous polymers*, volume 134. Oxford University Press, USA, 2006.

[48] Thomas-Peter Fries. Higher-order conformal decomposition fem (cdfem). *Comput. Method. Appl. M.*, 328:75–98, 2018.

[49] Thomas-Peter Fries and Ted Belytschko. The intrinsic XFEM: a method for arbitrary discontinuities without additional unknowns. *Int. J. Numer. Methods Eng.*, 68:1358–1385, 2006.

[50] Thomas-Peter Fries and Samir Omerović. Higher-order accurate integration of implicit geometries. *Int. J. Numer. Methods Eng.*, 106(5):323–371, may 2016.

[51] Thomas-Peter Fries, Samir Omerović, Daniel Schöllhammer, and Jakob Steidl. Higher-order meshing of implicit geometries—Part I: Integration and interpolation in cut elements. *Comput. Method. Appl. M.*, 313:759–784, 2017.

[52] Thomas-Peter Fries and Daniel Schöllhammer. Higher-order meshing of implicit geometries, part ii: Approximations on manifolds. *Comput. Method. Appl. M.*, 326:270–297, 2017.

[53] Olivier Gallinato and Clair Poignard. Superconvergent Cartesian methods for Poisson type equations in 2d – domains. Technical report, 2015.

[54] Olivier Gallinato and Clair Poignard. Superconvergent second order Cartesian method for solving free boundary problem for invadopodia formation. *J. Comput. Phys.*, 339:412–431, 2017.

[55] Frederic Gibou, Liguo Chen, Duc Nguyen, and Sanjoy Banerjee. A level set based sharp interface method for the multiphase incompressible navier–stokes equations with phase change. *J. Comput. Phys.*, 222(2):536–555, March 2007.

[56] Frédéric Gibou and Ronald Fedkiw. A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem. *J. Comput. Phys.*, 202(2):577 – 601, 2005.

[57] Frédéric Gibou, Ronald Fedkiw, Russel Caflisch, and Stanley Osher. A level set approach for the numerical simulation of dendritic growth. *J. Sci. Comput.*, 19(1-3):183–199, 2003.

[58] Frederic Gibou, Ronald Fedkiw, Li-Tien Cheng, and Myngjoo Kang. A second-order accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comput. Phys.*, 176(1):205–227, 2002.

[59] Frederic Gibou, Ronald Fedkiw, and Stanley Osher. A review of level-set methods and some recent applications. *J. Comput. Phys.*, 353:82 – 109, 2018.

[60] Frederic Gibou and Chohong Min. Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions. *J. Comput. Phys.*, 231:3246–3263, 2012.

[61] Frederic Gibou, Chohong Min, and Ronald Fedkiw. High resolution sharp computational methods for elliptic and parabolic problems in complex geometries. *J. Sci. Comput.*, 54:369–413, 2013.

[62] J Glimm, J Grove, X L Li, K Shyue, Y.Zeng, and Q Zhang. Three dimensional front tracking. *SIAM J. Sci. Comput.*, 1998:703–729, 1998.

[63] James Glimm, John W. Grove, Xiaolin Li, and Nailiang Zhao. Simple front tracking. *Contemporary Math.*, 238:133–149, 1999.

[64] D Greenspan. On the numerical solution of problems allowing mixed boundary conditions. *J. Franklin I.*, 277(1):11–30, 1964.

[65] S. Groí and A. Reusken. An extended pressure finite element space for two-phase incompressible flows with surface tension. *J. Comput. Phys.*, 224:40–58, 2007.

[66] Arthur Guittet, Mathieu Lepilliez, Sebastien Tanguy, and Frédéric Gibou. Solving elliptic problems with discontinuities on irregular domains – the Voronoi interface method. *J. Comput. Phys.*, 298:747 – 765, 2015.

[67] Arthur Guittet, Clair Poignard, and Frederic Gibou. A voronoi interface approach to cell aggregate electropermeabilization. *J. Comput. Phys.*, 332:143 – 159, 2017.

[68] Arthur Guittet, Maxime Theillard, and Frédéric Gibou. A stable projection method for the incompressible Navier–Stokes equations on arbitrary geometries and adaptive Quad/Octrees. *J. Comput. Phys.*, 292:215 – 238, 2015.

[69] Grégory Guyomarc'h, Chang-Ock Lee, and Kiwan Jeon. A discontinuous Galerkin method for elliptic interface problems with application to electroporation. *Comm. Numer. Meth. Eng.*, 25(10):991–1008, 2009.

[70] J Heinrich and P Zhao. Front tracking finite element method for dendritic solidification. *J. Comput. Phys.*, 173:765–796, 2001.

[71] Ásdís Helgadóttir and Frederic Gibou. A Poisson-Boltzmann solver on irregular domains with Neumann or Robin boundary conditions on non-graded adaptive grid. *J. Comput. Phys.*, 230:3830–3848, 2011.

[72] Jeffrey Lee Hellrung, Luming Wang, Eftychios Sifakis, and Joseph M. Teran. A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions. *J. Comput. Phys.*, 231(4):2015 – 2048, 2012.

[73] Thomas Y Hou, John S Lowengrub, and Michael J Shelley. Removing the stiffness from interfacial flows with surface tension. *J. Comput. Phys.*, 114(2):312–338, 1994.

[74] Hanqiong Hu, Manesh Gopinadhan, and Chinedum O Osuji. Directed self-assembly of block copolymers: a tutorial review of strategies for enabling nanotechnology with soft matter. *Soft matter*, 10(22):3867–3889, 2014.

[75] Wei-Fan Hu, Ming-Chih Lai, and Yuan-Nan Young. A hybrid immersed boundary and immersed interface method for electrohydrodynamic simulations. *J. Comput. Phys.*, 282:47–61, 2015.

[76] Seong-Jun Jeong, Ji Eun Kim, Hyoung-Seok Moon, Bong Hoon Kim, Su Min Kim, Jin Baek Kim, and Sang Ouk Kim. Soft graphoepitaxy of block copolymer assembly with disposable photoresist confinement. *Nano letters*, 9(6):2300–2305, 2009.

[77] Seong-Jun Jeong, Ju Young Kim, Bong Hoon Kim, Hyoung-Seok Moon, and Sang Ouk Kim. Directed self-assembly of block copolymers for next generation nanolithography. *Materials today*, 16(12):468–476, 2013.

[78] Espen Jettestuen, Johan O. Helland, and Masa Prodanovic. A level set method for simulating capillary-controlled displacements at the pore scale with nonzero contact angles. *Water Resour. Res.*, 49(8):4645–4661, 2013.

[79] H. Ji and J. Dolbow. On strategies for enforcing interfacial constraints and evaluating jump conditions with extended finite element method. *Int. J. Numer. Methods Eng.*, 61(14):2508–2535, 2004.

[80] H. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson equation on irregular domains. *J. Comput. Phys.*, 147:60–85, 1998.

[81] Z. Jomaa and C. Macaskill. The Shortley-Weller embedded finite-difference method for the 3d Poisson equation with mixed boundary conditions. *J. Comput. Phys.*, 229(10):3675–3690, 2010.

[82] Ziad Jomaa and Charlie Macaskill. Numerical solution of the 2-d poisson equation on an irregular domain with robin boundary conditions. In *Proceedings of the 14th Biennial Computational Techniques and Applications Conference*, volume 50, pages 1–10, 2008.

[83] Damir Juric and Grétar Tryggvason. A front tracking method for dendritic solidification. *J. Comput. Phys*, 123:127–148, 1996.

[84] Alain Karma. Phase-field formulation for quantitative modeling of alloy solidification. *Physical Review Letters*, 87(11):115701, 2001.

[85] Jaeup U. Kim and Mark W. Matsen. Droplets of structured fluid on a flat substrate. *Soft Matter*, 5(15):2889–2895, 2009.

[86] Seong Gyoon Kim. A phase-field model with antitrapping current for multicomponent alloys with arbitrary thermodynamic properties. *Acta Materialia*, 55(13):4391–4399, 2007.

[87] Jason Koski, Huikuan Chao, and Robert A Riggleman. Field theoretic simulations of polymer nanocomposites. *The Journal of chemical physics*, 139(24):244911, 2013.

[88] W. Kurz and D. J. Fisher. *Fundamentals of Solidification*. Trans Tech Publication, 1998.

[89] Azat Latypov. Computational solution of inverse directed self-assembly problem. In *Alternative Lithographic Technologies V*, volume 8680, page 86800Z. International Society for Optics and Photonics, 2013.

[90] Mathieu Lepilliez, Elena Roxana Popescu, Frederic Gibou, and Sébastien Tanguy. On two-phase flow solvers in irregular domains with contact line. *J. Comput. Phys.*, 321:1217–1251, 2016.

[91] Adrián J Lew and Gustavo C Buscaglia. A discontinuous-Galerkin-based immersed boundary method. *Int. J. Numer. Methods Eng.*, 76:427–454, 2008.

[92] Zhilin Li. A fast iterative algorithm for elliptic interface problems. *SIAM J. Numer. Anal.*, 35:230–254, 1998.

[93] Zhilin Li and Kazufumi Ito. *The Immersed Interface Method – Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*, volume 33. SIAM Frontiers in Applied mathematics, 2006.

[94] Zi-Cai Li and Tzon-Tzer Lu. Singularities and treatments of elliptic boundary value problems. *Math. Comput. Modell.*, 31(8-9):97–145, 2000.

[95] X.-D. Liu, R. P. Fedkiw, and M. Kang. A boundary capturing method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 160:151–178, 2000.

[96] Frank Losasso, Frederic Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 457–462, 2004.

[97] Michael J Maher, Jeffrey L Self, Pawel Stasiak, Gregory Blachut, Christopher J Ellison, Mark W Matsen, Christopher M Bates, and C Grant Willson. Structure, stability, and reorganization of 0.5 l 0 topography in block copolymer thin films. *ACS nano*, 10(11):10152–10160, 2016.

[98] Mark W Matsen. The standard gaussian model for block copolymer melts. *Journal of Physics: Condensed Matter*, 14(2):R21, 2001.

[99] Chohong Min and Frederic Gibou. Geometric integration over irregular domains with application to level-set methods. *J. Comput. Phys.*, 226:1432–1443, 2007.

[100] Chohong Min and Frederic Gibou. A second order accurate level set method on non-graded adaptive Cartesian grids. *J. Comput. Phys.*, 225(1):300–321, 2007.

[101] Chohong Min and Frederic Gibou. Robust second-order accurate discretizations of the multi-dimensional Heaviside and Dirac delta functions. *J. Comput. Phys.*, 227(22):9686–9695, November 2008.

[102] Chohong Min, Frédéric Gibou, and Hector D Ceniceros. A supra-convergent finite difference scheme for the variable coefficient poisson equation on non-graded grids. *J. Comput. Phys.*, 218(1):123–140, 2006.

[103] Mohammad Mirzadeh and Frédéric Gibou. A conservative discretization of the Poisson–Nernst–Planck equations on adaptive cartesian grids. *J. Comput. Phys.*, 274:633–653, 2014.

[104] Mohammad Mirzadeh, Arthur Guittet, Carsten Burstedde, and Frederic Gibou. Parallel level-set methods on adaptive tree-based grids. *J. Comput. Phys.*, 322:345 – 364, 2016.

[105] Mohammad Mirzadeh, Maxime Theillard, and Frédéric Gibou. A second-order discretization of the nonlinear Poisson–Boltzmann equation over irregular geometries using non-graded adaptive Cartesian grids. *J. Comput. Phys.*, 230(5):2125 – 2140, 2011.

[106] Mohammad Mirzadeh, Maxime Theillard, Asdís Helgadöttir, David Boy, and Frédéric Gibou. An adaptive, finite difference solver for the nonlinear Poisson-Boltzmann equation with applications to biomolecular computations. *Comm. Comput. Phys.*, 13(1):150–173, 2012.

[107] Pouria Mistani, Arthur Guittet, Daniil Bochkov, Joshua Schneider, Dionisios Margetis, Christian Ratsch, and Frederic Gibou. The island dynamics model on parallel quadtree grids. *J. Comput. Phys.*, 361:150–166, 2018.

[108] Pouria Mistani, Arthur Guittet, Clair Poignard, and Frederic Gibou. A parallel voronoi-based approach for mesoscale simulations of cell aggregate electropermeabilization. *J. Comput. Phys.*, 380:48 – 64, 2019.

[109] Nicolas Moës, Mathieu Cloirec, Patrice Cartraud, and Jean-Francois Remacle. A computational approach to handle complex microstructure geometries. *Comput. Method. Appl. M.*, 192:3162–3177, 2003.

[110] Nicolas Moës, John Dolbow, and Ted Belytschko. A finite element method for crack growth without remeshing. *Int. J. Numer. Methods Eng.*, 46:131–150, 1999.

[111] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:385–392, 2004.

[112] Timothy J Moroney, Dylan R Lusmore, Scott W McCue, and DL Sean McElwain. Extending fields in a level set method by solving a biharmonic equation. *J. Comput. Phys.*, 343:170–185, 2017.

[113] Mohammed Moumnassi, Salim Belouettar, Éric Béchet, Stéphane P.A. Bordas, Didier Quoirin, and Michel Potier-Ferry. Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces. *Comput. Method. Appl. M.*, 200(5-8):774–796, jan 2011.

[114] Björn Müller, Florian Kummer, and Martin Oberlack. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *Int. J. Numer. Methods Eng.*, 96(8):512–528, nov 2013.

[115] Yen Ting Ng, Han Chen, Chohong Min, and Frederic Gibou. Guidelines for Poisson solvers on irregular domains with Dirichlet boundary conditions using the Ghost Fluid Method. *Journal of Scientific Computing*, 41(2):300–320, May 2009.

[116] Yen Ting Ng, Chohong Min, and Frédéric Gibou. An efficient fluid–solid coupling algorithm for single-phase flows. *J. Comput. Phys.*, 228(23):8807 – 8829, 2009.

[117] Duc Nguyen, Frederic Gibou, and Ronald Fedkiw. A Fully Conservative Ghost Fluid Method and Stiff Detonation Waves. In *12th Int. Detonation Symposium, San Diego, CA*, 2002.

[118] M. Oevermann, C. Scharfenberg, and R. Klein. A sharp interface finite volume method for elliptic equations on Cartesian grids. *J. Comput. Phys.*, 228:5184–5206, 2009.

[119] Samir Omerović and Thomas-Peter Fries. Conformal higher-order remeshing schemes for implicitly defined interface problems. *Int. J. Numer. Methods Eng.*, 109(6):763–789, 2017.

[120] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002. New York, NY.

[121] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003.

[122] Stanley Osher and James A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[123] Gaddiel Ouaknin, Nabil Laachi, Daniil Bochkov, Kris Delaney, Glenn H Fredrickson, and Frederic Gibou. Functional level-set derivative for a polymer self consistent field theory hamiltonian. *J. Comput. Phys.*, 345:207–223, 2017.

[124] Gaddiel Ouaknin, Nabil Laachi, Kris Delaney, Glenn H. Fredrickson, and Frederic Gibou. Self-consistent field theory simulations of polymers on arbitrary domains. *J. Comput. Phys.*, 327:168 – 185, 2016.

[125] Gaddiel Ouaknin, Nabil Laachi, Kris Delaney, Glenn H Fredrickson, and Frederic Gibou. Level-set strategy for inverse DSA-lithography. *J. Comput. Phys.*, 375:1159–1178, 2018.

[126] A Pandolfi, P Krysl, and M Ortiz. Finite element simulation of ring expansion and fragmentation. *Int. J. Fract.*, 95(1–4):279–297, 1999.

[127] Joe Papac, Dionisios Margetis, Frederic Gibou, and Christian Ratsch. Island-dynamics model for mound formation: Effect of a step-edge barrier. *Phys. Rev. E*, 90(2):022404, 2014.

[128] Joseph Papac, Frederic Gibou, and Christian Ratsch. Efficient symmetric discretization for the Poisson, heat and Stefan-type problems with Robin boundary conditions. *J. Comput. Phys.*, 229:875–889, 2010.

[129] Joseph Papac, Asdis Helgadottir, Christian Ratsch, and Frederic Gibou. A level set approach for diffusion and Stefan-type problems with Robin boundary conditions on Quadtree/Octree adaptive Cartesian grids. *J. Comput. Phys.*, 233:241–261, 2013.

[130] J. Qian, G. Tryggvason, and C.K. Law. A front tracking method for the motion of premixed flames. *J. Comput. Phys.*, 144(1):52–69, July 1998.

[131] K Reuther and M Rettenmayr. Perspectives for cellular automata for the simulation of dendritic solidification–a review. *Computational materials science*, 95:213–220, 2014.

[132] Casey L Richardson, Jan Hegemann, Eftychios Sifakis, Jeffrey Hellrung, and Joseph M Teran. An XFEM method for modeling geometrically elaborate crack propagation in brittle materials. *Int. J. Numer. Methods Eng.*, 88:1042–1065, 2011.

[133] Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph.*, 27(3):1–9, 2008.

[134] Annette Rösler, Guido WM Vandermeulen, and Harm-Anton Klok. Advanced drug delivery devices via self-assembly of amphiphilic block copolymers. *Advanced drug delivery reviews*, 64:270–279, 2012.

[135] Giovanni Russo and Peter Smereka. A remark on computing distance functions. *J. Comput. Phys.*, 163(1):51–67, 2000.

[136] Chris H Rycroft and Frédéric Gibou. Simulations of a stretching bar using a plasticity model from the shear transformation zone theory. *J. Comput. Phys.*, 231(5):2155–2179, 2012.

[137] G Ryskin and LG Leal. Numerical solution of free-boundary problems in fluid mechanics. part 1. the finite-difference technique. *J. Fluid Mech.*, 148:1–17, 1984.

[138] Rajiv Sampath and Nicholas Zabaras. Numerical study of convection in the directional solidification of a binary alloy driven by the combined action of buoyancy, surface tension, and electromagnetic forces. *J. Comput. Phys.*, 168(2):384 – 411, 2001.

[139] R. I. Saye. High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangle. *SIAM J. Sci. Comput.*, 37(2):A993–A1019, jan 2015.

[140] Alfred Schmidt. Computation of three dimensional dendrites with finite elements. *J. Comput. Phys.*, 125:293–312, 1996.

[141] Peter Schwartz, Michael Barad, Phillip Colella, and Terry Ligocki. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *J. Comp. Phys.*, 211(2):531–550, 2006.

[142] Peter Schwartz, Julie Percelay, Terry Ligocki, Hans Johansen, Daniel Graves, Dharshi Devendran, Phillip Colella, and Eli Ateljevich. High-accuracy embedded boundary grid generation using the divergence theorem. *Comm. App. Math. Com. Sc.*, 10(1):83–96, mar 2015.

[143] Guus Segal, Kees Vuik, and Fred Vermolen. A conserving discretization for the free boundary in a two-dimensional Stefan problem. *J. Comput. Phys.*, 141(1):1 – 21, 1998.

[144] James A Sethian. *Level set methods*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 1996. Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science.

[145] James A. Sethian. *Level set methods and fast marching methods*. Cambridge University Press, 1998.

[146] James A Sethian and Alexander Vladimirsky. Ordered upwind methods for static hamilton-jacobi equations. *Proc. Natl. Acad. Sci*, 98/20:11069–11074, 2001.

[147] George H Shortley and Royal Weller. The numerical solution of laplace's equation. *J. Appl. Phys.*, 9(5):334–348, 1938.

[148] Scott W Sides, Bumjoon J Kim, Edward J Kramer, and Glenn H Fredrickson. Hybrid particle-field simulations of polymer nanocomposites. *Physical review letters*, 96(25):250601, 2006.

[149] Eftychios Sifakis, Kevin G Der, and Ronald Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, volume 2007, pages 73–80, 2007.

[150] P Smereka. The numerical approximation of a delta function with application to level set methods. *J. Comput. Phys.*, 211:77–90, 2006.

[151] Gihun Son. A level set method for incompressible two-fluid flows with immersed solid boundaries. *Numer. Heat. Tr. B-Fund.*, 47(5):473–489, apr 2005.

[152] David P. Starinshak, Smadar Karni, and Philip L. Roe. A New Level-Set Model for the Representation of Non-Smooth Geometries. *J. Sci. Comput.*, 61(3):649–672, 2014.

[153] David P. Starinshak, Smadar Karni, and Philip L. Roe. A new level set model for multimaterial flows. *J. Comput. Phys.*, 262:1–16, 2014.

[154] P. Stasiak, J. D. McGraw, K. Dalnoki-Veress, and M. W. Matsen. Step edges in thin films of lamellar-forming diblock copolymer. *Macromolecules*, 45(23):9531–9538, 2012.

[155] Morgan Stefik, Stefan Guldin, Silvia Vignolini, Ulrich Wiesner, and Ullrich Steiner. Block copolymer self-assembly for nanophotonics. *Chemical Society Reviews*, 44(15):5076–5091, 2015.

[156] Ingo Steinbach. Phase-field models in materials science. *Modelling and simulation in materials science and engineering*, 17(7):073001, 2009.

[157] Mark Sussman, Peter Smereka, and Stanley Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114:146–159, 1994.

[158] Maxime Theillard, Landry Fokoua Djodom, Jean-Léopold Vié, and Frédéric Gibou. A second-order sharp numerical method for solving the linear elasticity equations on irregular domains and adaptive grids – application to shape optimization. *J. Comput. Phys.*, 233:430–448, 2013.

[159] Maxime Theillard, Frédéric Gibou, and Tresa Pollock. A sharp computational method for the simulation of the solidification of binary alloys. *J. Sci. Comput.*, 63(2):330–354, 2015.

[160] Russell B Thompson, Valeriy V Ginzburg, Mark W Matsen, and Anna C Balazs. Predicting the mesophases of copolymer-nanoparticle composites. *Science*, 292(5526):2469–2472, 2001.

[161] A.-K. Tornberg and B Engquist. Numerical approximations of singular source terms in differential equations. *J. Comput. Phys.*, 200:462–488, 2004.

[162] J Towers. Two methods for discretizing a delta function supported on a level set. *J. Comput. Phys.*, 220:915–931, 2007.

[163] John D. Towers. Finite difference methods for approximating Heaviside functions. *J. Comput. Phys.*, 228(9):3478–3489, 2009.

[164] G Tryggvason, B Bunner, A Esmaeeli, D Juric, N Al-Rawahi, W Tauber, J Han, S Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, 169:708–759, 2001.

[165] Masafumi Tsunekane, Akane Suzuki, and Tresa M Pollock. Single-crystal solidification of new co–al–w-base alloys. *Intermetallics*, 19(5):636–643, 2011.

[166] S O Unverdi and G Tryggvason. A front-tracking method for viscous, incompressible, multifluid flows. *J. Comput. Phys.*, 100:25–37, 1992.

[167] F. van der Bos and V. Gravemeier. Numerical simulation of premixed combustion using an enriched finite element method. *J. Comput. Phys.*, 228:3605–3624, 2009.

[168] H. J. van Linde. High-order finite-difference methods for Poisson's equation. *Math. Comput.*, 28(126):369–369, may 1974.

[169] Richard S Varga. *Matrix iterative analysis*, volume 27. Springer Science & Business Media, 2009.

[170] J. Villain. Continuum models of crystal growth from atomic beams with and without desorption. *J. Phys. I*, 1, 1991.

[171] Xin Wen. High order numerical methods to two dimensional delta function integrals in level set methods. *J. Comput. Phys.*, 228(11):4273–4290, jun 2009.

[172] Xin Wen. High order numerical methods to three dimensional delta function integrals in level set methods. *SIAM J. Sci. Comput.*, 32(3):1288–1309, jan 2010.

[173] Xin Wen. High order methods to Heaviside function integrals. *J. Comput. Math.*, 29(3):305–323, 2011.

[174] A. Wiegmann and K. Bube. The explicit jump immersed interface method: finite difference method for pdes with piecewise smooth solutions. *SIAM J. Numer. Anal.*, 37(3):827–862, 2000.

[175] Neil M Wigley. An efficient method for subtracting off singularities at corners for laplace's equation. *J. Comput. Phys.*, 78(2):369–377, 1988.

[176] Kelin Xia, Xin Feng, Zhan Chen, Yiying Tong, and Guo-Wei Wei. Multiscale geometric modeling of macromolecules i: Cartesian representation. *J. Comput. Phys.*, 257:912 – 936, 2014.

[177] Jian-Jun Xu, Weidong Shi, Wei-Fan Hu, and Jun-Jie Huang. A level-set immersed interface method for simulating the electrohydrodynamics. *J. Comput. Phys.*, 400:108956, 2020.

[178] Sheng Xu. An iterative two-fluid pressure solver based on the immersed interface method. *Comm. Comput. Phys.*, 12(2):528–543, 2012.

[179] Yi Yang and HS Udaykumar. Sharp interface cartesian grid method iii: Solidification of pure materials and binary solutions. *Journal of Computational Physics*, 210(1):55–74, 2005.

[180] Sara Zahedi and Anna Karin Tornberg. Delta function approximations in level set methods by distance function extension. *J. Comput. Phys.*, 229(6):2199–2219, 2010.

[181] Hongkai Zhao. A fast sweeping method for eikonal equations. *Math. Comput.*, 74:603–627, 2004.

[182] P. Zhao, M. Vénere, J.C. Heinrich, and D.R. Poirier. Modeling dendritic growth of a binary alloy. *J. Comput. Phys.*, 188(2):434–461, July 2003.