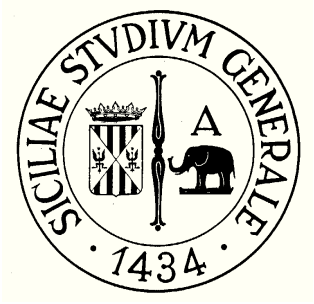
Università degli Studi di Catania

Facoltà di Scienze Matematiche, Fisiche e Naturali

Dipartimento di Matematica e Informatica

Dottorato di Ricerca in Matematica per la Tecnologia

XXIV ciclo



Finite-Difference Ghost-Cell Multigrid Methods for Elliptic Problems

with Mixed Boundary Conditions and Discontinuous Coefficients

Armando Coco

Advisor

Prof. Giovanni Russo

Anno Accademico 2010-2011

Università degli Studi di Catania

Facoltà di Scienze Matematiche, Fisiche e Naturali

Dipartimento di Matematica e Informatica

Dottorato di Ricerca in Matematica per la Tecnologia

XXIV ciclo

Finite-Difference Ghost-Cell Multigrid Methods for Elliptic Problems

with Mixed Boundary Conditions and Discontinuous Coefficients

Relatore
Chiar.mo Prof. Giovanni Russo

Coordiantore
Prof. Giovanni Russo

Candidato
Armando Coco

Anno Accademico 2010-2011

A Laura

Per ogni giorno, ogni istante, ogni attimo che sto vivendo: Grazie Mille.
M. Pezzali

Contents

Acknowledgement

Ho sempre considerato il dottorato come l'*adolescenza* del ricercatore. Oggi, ad un passo dal traguardo, posso affermare che, nonostante i tanti dubbi che affioravano nella mia testa sul ramo da scegliere, questa mia adolescenza è trascorsa in maniera *felice* e *spensierata*. Per questo ringrazio vivamente il mio tutor, prof. Giovanni Russo, che mi ha incoraggiato fin dall'inizio e mi ha guidato scrupolosamente nella formazione, facendomi scoprire passo dopo passo la passione per la ricerca. Il suo essere all'avanguardia ha contribuito a trasmettermi l'importanza che nella ricerca rivestono il confronto con gli altri, l'allargamento degli orizzonti, la collaborazione senza limiti geografici.

Ringrazio i miei genitori, che sono sempre stati un solido sostegno morale, fornendomi un appoggio concreto in ogni momento e permettendomi di mantenere viva la concentrazione nella ricerca.

Ringrazio mia sorella, con la quale ho condiviso una infanzia meravigliosa, e che non smette mai di farmi sentire importante.

Ringrazio Moghinetta, l'unica persona che riesce ad intenerirmi anche nei momenti di rabbia e nervosismo, capace di farmi riscoprire l'assoluta importanza delle cose semplici.

Infine, grazie Laura. Tu sei la linfa del mio percorso, la soluzione ai miei problemi, la ragione per cui tutto questo ha un senso.

Introduction

Elliptic equation in arbitrary domain (possibly with moving boundary) is central to many applications, such as diffusion phenomena, fluid dynamics, charge transport in semiconductors, crystal growth, electromagnetism and many others. The wide range of applications may require different kinds of boundary conditions. Let us look for instance at the temperature distribution in a medium of arbitrary shape satisfying stationary heat equation: we may have Dirichlet (the temperature is fixed at the boundary), Neumann (heat flux is prescribed), or mixed boundary conditions (namely different boundary conditions on different parts of the boundary). More general Robin boundary conditions may also be considered, as in Stefan-type problem, in which a combination of temperature and heat flux is prescribed at the boundary (e.g. see [47, 28]).

The work of this thesis is devoted to the development of an original and general numerical method for solving elliptic equations in an arbitrary domain (described by a level-set function) with general boundary conditions (Dirichlet, Neumann, Robin, ...) on Cartesian grids. It can be then considered an immersed boundary method, and the scheme we use is based on a finite-difference ghost-cell technique. The entire problem is solved by an effective multigrid solver, whose components have been suitably constructed in order to be applied to the scheme. The method is extended to the more challenging case of discontinuous coefficients, and the multigrid is suitable modified in order to attain the optimal convergence factor of the entire iteration procedure. The development of a multigrid solvers for discontinuous coefficients maintaining the optimal convergence factor independently on the jump in the coefficient and on the problem size is a challenging problem and it is recently studied in the literature. However, the development of the multigrid is only based on a practical implementation, and then it is not provided with a rigorous analytical study. The method is second order accurate in the

solution and its gradient. A convergence proof for the first order scheme is provided, while second order is confirmed by several numerical tests.

Survey of the literature for elliptic equations

Numerous techniques have been developed to treat such problems. Interface-fitted grid methods such as those based on the finite element methods [13, 18, 52, 65] are difficult to use in the case of moving interfaces, because of the computationally expensive meshing procedures needed at each time step. In such cases an approach treating the interface as embedded in a Cartesian grid may be preferred. Since the interface may be not aligned with the grid, a special treatment is needed for the discretization of the equation near the interface. The simplest method makes use of the Shortley-Weller discretization [102], that discretizes the Laplacian operator with usual central difference away from the interface, and makes use of a non symmetric stencil in the grid points close to the interface, adding extra-grid points on the interface. While the treatment of jump conditions in the solution is straightforward to discretize in the case where grid points lie on the interface, the discretization of the jump condition in the flux (involving the normal derivative) is not straightforward in more than one spatial dimension. In fact, Shortley-Weller discretization requires that the value of the normal derivative of the solution on both sides of the interface is suitably reconstructed at the intersection between the grid and the interface. This approach is adopted, for example, by Hackbusch in [56] to first-order accuracy, and by other authors (see [20] and references therein) to second-order accuracy. However, the method proposed by Bramble in [20] for second-order accuracy is quite involved and may not be recommended for all practical purposes.

Methods based on embedding the domain in a Cartesian grid are derived from the pioneering work of Peskin [88], where the Immersed Boundary Methods is introduced to model blood flows in the heart. In that paper a source term is localized on the interface and the method makes use of a discretized δ-function to model the exchange of force between the interface and the fluid, leading to a first-order accurate method. A second-order accurate extension to jump coefficients is the Immersed Interface Methods, first developed by LeVeque and Li in [67], where non-homogeneous jump conditions are allowed on the function and on the normal flux. Another method which achieves second-order accuracy was previously proposed by Mayo in [75] for solving Poisson or biharmonic equations on irregular domains using boundary integral techniques. Liu *et al.* [70] developed a first-order accurate ghost-fluid method for the variable coefficient Poisson equation in the presence of an irregular interface across which the variable coefficient, the solution and the

derivative of the solution may have jumps. This method leads to a symmetric linear system that allows the use of fast iterative solvers.

In the case where Dirichlet boundary conditions are imposed, instead of jump conditions, a second-order accurate ghost-fluid method can be built [46]. The value at the ghost nodes is assigned by linear extrapolation, and the whole discretization leads to a symmetric linear system. A fourth-order accurate version is also proposed by Gibou *et al.* in [47] and several characteristics concerning Poisson solvers on irregular domains with Dirichlet boundary conditions using the ghost-fluid method can be found in [81].

The treatment of boundary conditions on irregular interfaces is also found in methods modeling the interaction between multiphase flows and solid obstacles. Examples are the arbitrary Lagrangian Eulerian method (ALE) [44, 39], distribute Lagrangian multiplier (DLM) [51] and the penalization methods [98, 10]. For example, in [31] a combination of penalization and level-set methods is presented to solve inverse or shape optimization problems on uniform Cartesian meshes. In [82] Ng *et al.* studied a simple and efficient method for the Navier-Stokes equations on arbitrary shaped domains. Several methods have been recently developed for sharp-edge interface, such as the matched interface and boundary (MIB) method [110], the finite volume method [85], and the non-symmetric positive definite finite element method [58], where the more general case of variable matrix coefficient is treated.

In [87], an efficient discretization based on cut-cell method to impose Robin conditions is proposed. This method provides second-order accuracy for the Poisson and heat equation and first-order accuracy for Stefan-type problems. The drop in accuracy for the Stefan problem is due to the fact that the solution gradient, driving the free moving boundary, is only first-order accurate. Other approaches based on cut-cell methods obtained by a finite volume discretization are presented by Colella *et al.* in [60]. Cells that are cut by the boundary requires a special treatment, such as cell-merging and rotated-cell, in order to avoid too strict of a restriction of the time step dictated by the CFL condition.

In some applications, it is desirable to obtain second-order accuracy in the solution's gradient, as it is the case of the Stefan problem or in the case of two-way fluid-solid coupling. Also high-order accuracy may be required, for instance when turbulence and shock interact, or high frequency wave propagation are presented in inhomogeneous media [16].

Survey of the literature for related multigrid

Multigrid technique is one of the most efficient strategy to solve a class of partial differential equations, using a hierarchy of discretizations. It accelerates the convergence of an existing iterative method, which otherwise slowly converges toward the solution of the discrete problem, due to the bad convergence rate for the low frequency components of the error. The idea of multigrid methods is to solve such low frequency components in a coarser grid. Most iterative schemes to solve Elliptic equations can be speeded up by a multigrid technique. Some multigrid textbooks are for example [27, 105, 55].

Multigrid has been developed since the 1960s, when first papers appeared [42, 43, 15], while it has been studied more carefully since the 1970s [21, 54, 83, 6]. Several surveys on multigrid exist in literature, in particular for different kinds of applications [24, 22, 23], for multilevel-unstructured grids [29], parallel implementation [62, 1], discontinuous coefficients [104].

The Algebraic Multigrid (AMR) is a multigrid procedure that use only purely algebraic information, and it is mainly used in problems where a supported grid is not available, and then a geometric relation between unknowns cannot be derived. It was first proposed in [25] and later popularized in [94, 26]. Among the other numerous multigrid techniques, we recall the Black-box multigrid of Dendy [38, 63], the BPX method [19], the matrix-dependent interpolation [93, 111, 101], the energy-minimization [108, 30, 73].

Regarding the scope of this thesis, namely multigrid for elliptic equations with mixed boundary conditions and discontinuous coefficients, a detailed survey can be found in [104]. Several multigrid approaches exist in literature to treat the jumping coefficient problem in 2D when the interface is aligned with the Cartesian grid. We mention the method based on operator-dependent interpolation [6, 59], where the interpolation is carried out by exploiting the continuity of the flux instead of the gradient of the solution, and the method based on Galerkin Coarse Grid Operator [94], which makes the algebraic problem more expensive from a computational point of view and does not take advantage from the fact that the discrete problem comes from a continuous problem. For Cartesian grids and arbitrary interfaces, i.e. not aligned with line grids, we mention the paper [4], where a multigrid approach for solving the linear system arising from the discretization of interface conditions described in [4, 69] is provided. In this multigrid technique a Black-box multigrid interpolation is used for grid points away from the interface, while the interpolation weights for grid points near the interface are derived from a Taylor expansion (with a change of coordinates). In [2] such a multigrid has been improved, modifying the interpolation and restriction operators in such a way the coarse-grid matrices are M-matrices. A com-

parison of both multigrid method [4, 2] with Algebraic Multigrid solvers is performed in [3] for the underlying discretization, showing that the multigrid in [2] is the most efficient. Other recent developments of multigrid solvers for non-smooth coefficients can be found in [106], where a geometric multigrid method for multiple interfaces in higher dimensions is proposed. Here an accurate interpolation which captures the correct boundary conditions at the interfaces via a level set function is provided, and the issues coming from the storage of the coarse-grid matrix are avoided. In [107] the coarse grid points are selected in such a way the irregular interfaces are resolved as much as possible: only linear interpolation is needed to obtain fast convergence.

Survey of the literature for Adaptive Mesh Refinement

Many physical problems have different scales and very often only small portions of the computational domain require fine resolution. Uniform grids become inefficient in this case in terms of memory storage and CPU usage. Adaptive mesh strategies for elliptic linear partial differential equations, such as the Poisson equations, are often associated with the finite element method (see e.g. [14, 61]). Young *et al.* [109] introduced a finite element method employing adaptive mesh refinements for second-order variable coefficient elliptic equations using a cut-cell representation of irregular domains. The finite element method has the advantage of a rigorous theoretical framework and a vast number of optimized commercial implementations. However, two factors that must be considered are the adaptive mesh generation for complicated domains and the efficiency of the organization of the resulting data structure. Generally, when applying the finite element method to moving boundary problems, one must take great care that the mesh generated is of good quality everywhere (see e.g. [99]). Moreover, even though we are not considering this case in this thesis, one is often interested in solving the variable coefficient Poisson equation on time dependent irregular domains, where it is hard to avoid the difficulty and cost associated with frequent mesh generations. This is where finite difference discretizations can be advantageous (see e.g. [48]).

Johansen and Colella [60] presented a cell-centered finite volume method for solving the variable coefficient Poisson equation on irregular domains using a multigrid approach and a block-grid algorithm related to the adaptive mesh refinement scheme of Berger and Oliger [17]. McCorquodale *et al.* [76] presented a node-centered finite difference approach for solving the variable coefficient Poisson equation on irregular domains using the block-structured adaptive mesh refinement and the multigrid solver of Almgren [7, 9, 8]. However, these patch based adaptive mesh refinement (AMR) techniques restrict

the adaptivity to block structured meshes, as is made clear by the following quote "The obvious first choice, suggested by the nested hierarchical nature of the grid itself, is to use an octree in 3D or quadtree in 2D", from [5].

The quadtree spatial discretization has been used in a variety of approaches for solving the Poisson equation. Greengard *et al.* [53] presented a kind of domain decomposition and spectral element method to solve the Poisson equation using the quadtree data structure. Popinet [89] proposed a second-order nonsymmetric numerical method to study the incompressible Navier-Stokes equations using a quadtree data structure for spatial discretization. In this method, a Poisson equation for the pressure needs to be solved to account for the incompressibility condition using a standard projection method. Only graded trees (trees in which the ratio between two adjacent cell sizes does not exceed two) were considered in [89].

Non-graded octrees have been used in [72] to obtain a first order accurate symmetric discretization of the Poisson equation, which was extended to second-order accuracy in [71]. In [79] Min *et al.* solved the variable coefficient Poisson equation on a rectangular domain using quadtrees (in 2D) and octrees (in 3D) to represent the non-graded Cartesian grids, and second-order accuracy in both the solution and its gradients were obtained. In [32] Chen *et al.* extended this approach for Poisson and heat equations on irregular domains with Dirichlet boundary conditions. At internal T-junction nodes, the value at the missing direct neighbor is linearly interpolated and the discretization in the other direction is properly weighted to compensate for the spurious error induced by the linear interpolation. When the node is next to the interface, the interface point location and the Dirichlet boundary value at this interface point are found by quadratic interpolation. In [78] Min *et al.* proposed a second-order accurate level-set method on non-graded adaptive Cartesian grids. Incompressible Navier-Stokes equations on non-graded adaptive Cartesian grids can be found in [50, 45, 49, 77].

Structure of the thesis

Chapter 1. In this chapter we present a rather simple numerical method to solve the Poisson equation in an arbitrary domain Ω, identified by a level set function ϕ (i.e. $\Omega = \left\{x \in \mathbb{R}^d \colon \phi(x) < 0\right\}$), with mixed boundary conditions. The method is based on a finite difference discretization on a regular Cartesian grid using ghost points.

The plan of the chapter is the following: the first section introduces the level-set function and some notations, while the second section presents the description of the first and second order accurate method. The third section is devoted to the extensions of the method to Robin boundary conditions,

variable coefficient, anisotropic case and boundary with kinks. In the last section, several numerical tests are presented, confirming the accuracy and efficiency of the approach.

The results of this chapter can also be found in [34].

Chapter 2. In this chapter we present a multigrid approach to solve the linear system coming from the discretization of the elliptic problem described in Chapter 1. In the first section we motivate the necessity to find a proper relaxation scheme, illustrating that naive Jacobi iteration scheme fails to converge. The second section is devoted to the description of the novel relaxation scheme, which consists of a transformation of the Poisson problem into a fictitious-time dependent problem, that leads to an iterative scheme converging to the solutions of the originary problem. A convergence proof for the first-order accurate method is provided in Section 3. The other sections describe the relaxation scheme for the extensions presented in Chapter 1 (Robin boundary conditions, variable coefficient, anisotropic case and boundary with kinks), and the transfer grid operators for one and two dimensions.

This multigrid strategy can be applied to more general problems where a non-eliminated boundary condition approach is used. Arbitrary domains make the definition of the restriction operator for boundary conditions hard to find. In this thesis a suitable restriction operator is provided, together with a proper treatment of the boundary smoothing, in order to avoid degradation of the convergence factor of the multigrid due to boundary effects. Several numerical tests confirm the good convergence properties of the new method.

The results of this chapter can also be found in [35].

Chapter 3. In this chapter we provide a 1D discretization of the discontinuous coefficient case, together with a proper multigrid approach. We use the standard interpolation operator and discretize the operator in the coarser grid in the same way as in the fine grid, without making use of Galerkin conditions. But, since the defect may jump crossing the interface, a separated restriction for both sub-problems is needed. This approach provides a good convergence factor, comparable with the ones measured for no-jumping case. We also show that the convergence factor does not depend on the magnitude of the jump in the coefficient, nor on the problem size. Interface conditions are relaxed, then have to be transferred to the coarse grid as well. In one-dimensional case this task is trivial, since such conditions are just two real values that can be copied to the coarse grid.

This chapter is divided in 3 sections. In the first section we describe the second order accurate discretization of the model problem and the iterative scheme obtained by discretizing the fictitious time-dependent problem. The second section is devoted to the multigrid approach, with a careful description

of the transfer operators. In section 3 some numerical tests are performed, to show the second order accuracy in the solution and in its first derivative as well. We also measure the convergence factor and compare it with the convergence factor obtained by other methods of the literature.

The results of this chapter can also be found in [36].

Chapter 4. The discretization of the 2D discontinuous coefficient case is derived from the 1D discretization of Chapter 3 , using the ghost-cell structure of Chapter 1. In practice, new additional unknowns are added to the linear system and they are related to ghost points close to the interface and from both sides. In such ghost points two values of the solution are defined: one for the solution of the sub-domain to which such grid points belongs, and one for the ghost value related to the sub-domain on the other side of the interface. For the multigrid ingredients, the restriction is performed separately for the defect of inner equations of both subdomains, and of interface conditions. For inner equations, the stencil of the restriction is suitable reduced for grid points close to the interface, namely using only values from one side of the interface, in the same manner as we restrict the defect of inner equations in Chapter 2. For the interface conditions, in higher dimension the defect is stored in ghost points, which can show a complex structure for arbitrary interfaces. The restriction of the defect of interface conditions can be carried out in the same manner of the restriction of the defect of boundary conditions described in Chapter 2 for problems with non-eliminated boundary conditions: the defect is first extrapolated outside the subdomain and then transferred to the coarse grid in the same manner as the restriction of the defect of inner equations, i.e., without using values from the other side of the boundary. Numerical tests of the last section prove the second order accuracy and the efficiency of the multigrid solver, namely the optimal convergence factor is attained and it does not depends on the jump in the coefficient nor on the problem size.

Chapter 5. The discretization proposed in Chapter 1 to solve elliptic problems with mixed boundary conditions (i.e. Dirichlet on a part of the boundary and Neumann on the other part) and in Chapter 4 to treat the discontinuous coefficient case are embedded in the adaptive grid framework presented by Gibou *et al.* in [79, 32], where the adaptive grid generation and the finite difference discretization on quadtree were introduced for the Poisson equation in an arbitrary domain with continuous coefficient and Dirichlet boundary condition. We obtain a second-order method in the solution and the gradient for elliptic equation with discontinuous coefficient and mixed boundary conditions in a fully adaptive non-graded grid, which as far as we know is new in the literature. In order to easily implement the discretization of the boundary/interface conditions illustrated in Chapters 1 and 4, a

uniform grid is however required close to the boundary/interface. To this purpose, the refinement criterion is modified with respect to the one used for example in [79, 32], allowing uniform grid in a larger band surrounding the boundary/interface. In this chapter only the accuracy of the discretization is presented, relegating to a future work the implementation of the iterative scheme, consisting in an extension of the multigrid presented in Chapters 2 and 4. A proper fast direct solver is used, such as BiCGSTAB or the Matlab direct solver. Numerical results show that second-order accuracy is achieved in both the solution and its gradient.

In the first section the adaptive grid generation and the discretization of the elliptic equation on quadtrees are described. In Section 2 several numerical tests (some of which taken from [84, 57, 70]) are illustrated, which confirm the second-order accuracy of the solution and its gradient in L^2 and L^∞ norms.

The results of this chapter can also be found in [33].

Chapter 1

Elliptic equations

We start describing the numerical method for the simplest model problem, namely the Poisson equation with mixed boundary conditions in a smooth domain. Therefore we extend the technique for more general elliptic problems, such as the variable coefficient case, anisotropic problems, sharp-edged domains.

Let $d \geq 1$ an integer, $D = [-1, 1]^d$ the computational domain, $\Omega \subset D$ a domain such that $\partial\Omega \cap \partial D = \emptyset$. We assume Ω is a smooth domain, i.e. the boundary $\partial\Omega \in C^1$. Let Γ_D, Γ_N a partition of $\partial\Omega$ (i.e. $\Gamma_D \cup \Gamma_N = \partial\Omega, \overset{\circ}{\Gamma}_D \cap \overset{\circ}{\Gamma}_N = \emptyset$, where the interior points are computed in the $d-1$ dimensional topological space).

Model problem 1 *Consider the model problem:*

$$-\Delta u = f \quad \text{in } \Omega \tag{1.1}$$

$$u = g_D \quad \text{on } \Gamma_D \tag{1.2}$$

$$\frac{\partial u}{\partial n} = g_N \quad \text{on } \Gamma_N \tag{1.3}$$

where $\hat{\mathbf{n}}$ *is the outward unit normal,* $f \colon \Omega \to \mathbb{R}$, $g_D \colon \Gamma_D \to \mathbb{R}$, $g_N \colon \Gamma_N \to \mathbb{R}$ *are assigned functions.*

1.1 Notation

Let $N \geq 1$ an integer and $h = 2/N$ the spatial step. Let $D_h = \mathbf{j}h, \mathbf{j} = (j_1, \ldots, j_d) \in \{-N, N\}^d$ and $\Omega_h = \Omega \cap D_h$ be the discrete versions of D and Ω respectively. D_h is the set of the *grid points*. Let Γ_h the set of the

so-called *ghost points*, namely the grid points outside to Ω and belonging to some five-point stencil centered in a grid point inside to Ω, i.e.

$$(x,y) \in \Gamma_h \Longleftrightarrow (x,y) \in D_h \backslash \Omega_h \text{ and } \{(x \pm h, y), (x, y \pm h)\} \cap \Omega_h \neq \emptyset$$

Let us denote $N_i = |\Omega_h|$ and $N_g = |\Gamma_h|$ the cardinality of sets Ω_h and Γ_h. We will use the following notation for discrete functions: $w_{j_1,\ldots,j_d} \approx w(j_1\, h, \ldots, j_d\, h)$, $w_P \approx w(P)$.

1.1.1 Level-set function

In order to keep track of the boundary Γ, we introduce the level set function $\phi_0 \colon D \to \mathbb{R}$, in such a way:

$$(x,y) \in \overset{\circ}{\Omega} \Longleftrightarrow \phi_0(x,y) < 0, \quad (x,y) \in \partial\Omega \Longleftrightarrow \phi_0(x,y) = 0.$$

The outward unit normal to the boundary is

$$\mathbf{n} = \frac{\nabla \phi_0}{|\nabla \phi_0|}. \tag{1.4}$$

General references on the level set method for tracking interfaces are, for examples, [86] or [100]. The signed distance function ϕ is a particular case of level-set function:

$$\phi(x,y) = \begin{cases} -d\left((x,y),\Gamma\right) & \text{if } (x,y) \in \Omega, \\ d\left((x,y),\Gamma\right) & \text{if } (x,y) \notin \Omega, \end{cases}$$

where $d\left((x,y),\Gamma\right) = \inf\limits_{(\bar{x},\bar{y})\in\Gamma} d_e\left((x,y),(\bar{x},\bar{y})\right)$ is the distance function between a point and a set and d_e denotes the Euclidean distance between points. From the level set function ϕ_0, we can obtain the signed distance function ϕ by fast marching methods [100] or by the reinitialization procedure based on the numerical solution of the following PDE

$$\frac{\partial \phi}{\partial t} = \operatorname{sgn}(\phi_0)\left(1 - |\nabla \phi|\right), \tag{1.5}$$

as we can see, for instance, in [103, 95, 40]. A signed distance function is preferred to a simple level-set function because sharp gradients are avoided and it is simpler to compute the boundary closest point to a given ghost point. Now we assume that $|\nabla \phi| = 1$ and suppose we know the signed distance function just at the grid nodes. In practice, Eq. (1.5) has to be solved for a few time steps, in order to compute the distance function a few grid points away from the boundary.

1.2 Discretization of the problem

Let us start by describing the space discretization of the problem, i.e. the discretization of model problem 1. Our goal is to write a $(N_i+N_g)\times(N_i+N_g)$ linear system, where $N_i = |\Omega_h|$ and $N_g = |\Gamma_h|$. In this section we refer to the 2D case, but easy generalization can be obtained in higher dimension. The N_i equations coming from the inside grid points of Ω are obtained from the discretization of the Laplace operator by the 5-point stencil (Fig. 1.1):

$$\frac{4u_{i,j} - (u_{i,j-1} + u_{i,j+1} + u_{i-1,j} + u_{i+1,j})}{h^2} = f_{i,j}. \tag{1.6}$$

To close the linear system, we must write an equation for each ghost point G.

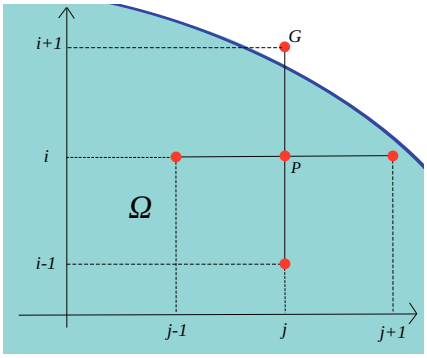


Fig. 1.1: Five-point stencil centered at P. The grid point G is outside the domain and it is named *ghost point*.

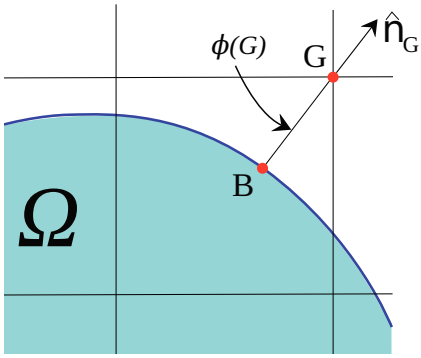


Fig. 1.2: Computation of the boundary closest point B to the ghost point G from the normal unit vector, obtained from the signed distance function.

1.2.1 Discretization of boundary conditions

In order to close the linear system of equations (1.6) for inside grid points, we must write an equation for each ghost point. Let G be a ghost point. We compute the outward unit normal in G, that is $\hat{\mathbf{n}}_G = (n_G^x, n_G^y) = \nabla\phi/|\nabla\phi|$, using a second order accurate discretization for $\nabla\phi$, such as central difference in G. Now we can compute the closest boundary point to G, that we call B, by the signed distance function (see Fig. 1.2):

$$B = G - \hat{\mathbf{n}}_G \cdot \phi(G). \tag{1.7}$$

We observe that indeed B depends on G, then it should be written as B_G, but we omit the subscript for clarity. Now, we discretize in some way the

Dirichlet or the Neumann boundary condition if B lies respectively on the Dirichlet or Neumann boundary. In particular, if $B \in \Gamma_D$, the equation for the system becomes:

$$u_h(B) = g_D(B) \tag{1.8}$$

while, if $B \in \Gamma_N$, it becomes:

$$\frac{\partial u_h}{\partial n}(B) = g_N(B)$$

where $u_h(B)$ and $\dfrac{\partial u_h}{\partial n}(B)$ are suitable reconstructions of the exact solution and its normal derivative by the numerical solution u_h.

We obtain a first or second order accurate solution of the model problem 1 according to the accuracy order of the reconstructions. Let us choose the reconstruction in detail.

1.2.1.1 First order accuracy

We can simply choose

$$u_h(B) = u_G, \tag{1.9}$$

while $\partial u/\partial n$ is discretized by first order upwind (Fig. 1.3)

$$\frac{\partial u_h}{\partial n}(B) = \frac{1}{h}\left(\left(u_G^{(n)} - u_{Q_x}^{(n)}\right)|n_x| + \left(u_G^{(n)} - u_{Q_y}^{(n)}\right)|n_y|\right) \tag{1.10}$$

where Q_x and Q_y are the two upwind close points to G (we say that the grid point Q is an *upwind close point* to G if $d(G,Q) = h$ and $(G-Q)\cdot\hat{\mathbf{n}} > 0$), while $\hat{\mathbf{n}} = (n_x, n_y)$ is the outward unit normal to $\partial\Omega$ in B, computed using (1.4) and discretizing $\nabla\phi$ also in upwind fashion, i.e.

$$\phi_x = \frac{\phi_G - \phi_{Q_x}}{h}\operatorname{sgn}\left(G^{(x)} - Q_x^{(x)}\right) \text{ and } \phi_y = \frac{\phi_G - \phi_{Q_y}}{h}\operatorname{sgn}\left(G^{(y)} - Q_y^{(y)}\right)$$

1.2.1.2 Second order accuracy

We choose

$$u_h(B) = \tilde{u}(B), \quad \frac{\partial u_h}{\partial n}(B) = (\nabla\tilde{u}\cdot\hat{\mathbf{n}})|_B = \left(\nabla\tilde{u}\cdot\nabla\tilde{\phi}/|\nabla\tilde{\phi}|\right)\Big|_B \tag{1.11}$$

where $\tilde{u}$ and $\tilde{\phi}$ are biquadratic interpolant respectively of u and ϕ on the Upwind nine-point stencil St_9 represented in Figure 1.4 ($n_G^x > 0$, $n_G^y > 0$):

$$St_9 = \left\{G + h(s_x\, k_1, s_y\, k_2)\colon (k_1,k_2) \in \{0,1,2\}^2\right\}, \tag{1.12}$$

where $s_x = \operatorname{sgn}(x_B - x_G)$ and $s_y = \operatorname{sgn}(y_B - y_G)$, with the notation $P \equiv (x_P, y_P)$, for $P = G, B$.

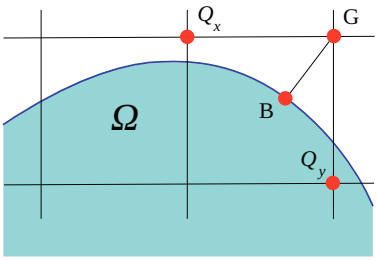


Fig. 1.3: discretization of Ω in two space dimensions. B is the boundary closest point to G, while Q_x and Q_y are the two upwind close points to G.

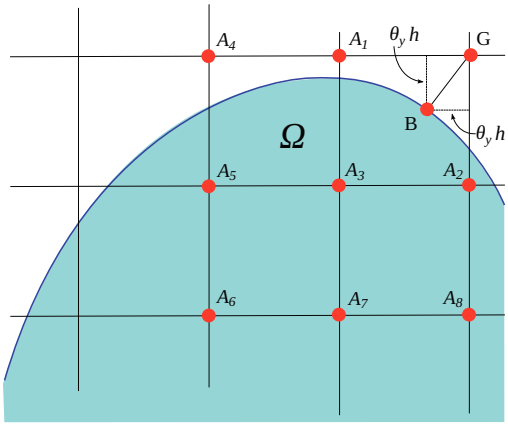


Fig. 1.4: nine-point stencil in upwind direction for the discretization of the Neumann boundary condition concerning the ghost point G

1.2.1.3 Remarks

Remark 1. (Error in the gradient) If we were interested in second order accuracy just of the solution, we could use a linear interpolation for $u_h(B)$ in grid points P, A_1, A_2, A_3 (see Fig. 1.4). Since in general we want to obtain second order accuracy also in the gradient, we use biquadratic interpolation for both Dirichlet and Neumann conditions. We show in Example 1.4.2 that we obtain second order accuracy also in the gradient. The accuracy of the gradient is computed by the seminorm of the Sobolev space $W^{1,q}(\Omega)$, i.e.

$$|u - u_h|_{W^{1,q}(\Omega)} = \|\nabla u - \nabla_h u_h\|_{L^q(\Omega)}$$

where $\nabla_h u_h$ is computed by central difference and for $q = 2$.

Remark 2. (Efficient stencil) We propose a more efficient stencil to reconstruct the normal derivative. We observe we just need to reconstruct the gradient of the solution u at the closest point B. We can reconstruct for instance the partial derivative $\partial u/\partial x$ as follows (for $\partial u/\partial y$ is similar). Consider the six-point stencil depicted in Figure 1.5. Then

$$\left.\frac{\partial u_h}{\partial x}\right|_B = \vartheta_y \left.\frac{\partial u_h}{\partial x}\right|_{H'} + (1 - \vartheta_y) \left.\frac{\partial u_h}{\partial x}\right|_H$$

Now, relating to Figure 1.5, we reconstruct $\partial u_h/\partial x$ in H' $[H]$ using a quadratic interpolation of u in P_1, P_2, P $[P_1', P_2', P']$ and differentiating it in H $[H']$. Ultimately, we use a six-point stencil to reconstruct the x-derivative, that is

more efficient than using a nine-point stencil. The computational cost improvement is more evident in three dimension, where a 12-point stencil is used instead of a 27-point stencil.

Remark 3. (Outside points) In some case, we cannot have all nine points of the upwind stencil in Figure 1.4 at our disposal, because some of these points may be neither a point of Ω_h nor a (ghost) point of Γ_h. We call them *outside points* (e.g. point P_2 in Fig. 1.6). Then, when an *outside point* is one of the nine points of the stencil of Figure 1.4, we use a different approach to discretize the Neumann condition. The six-point stencil described in the previous remark is easier to deal if some grid point is an outside point. For example, if we are in the case of Figure 1.6, point P_2 is an *outside point*. Then, we just use a first-order reconstruction in H:

$$\left.\frac{\partial u_h}{\partial x}\right|_H = \frac{u_P - u_{P_1}}{h},$$

leaving the second order reconstruction in H' unchanged. We will see numerically in Section 1.4 (Ex. 1.4.3) that even if we are in the case of Figure 1.6 (P_2 is an *outside point*), we do not loose the overall second order accuracy of the solution.

We suppose the spatial step h is less than the minimum radius of curvature of the boundary $\partial\Omega$, in order to capture the profile of the boundary. With this assumption, the reduced stencil rarely appears, and when it appears the closest point B is almost aligned with P', P'_1, P'_2. Therefore, the second order reconstruction in H' is weighted much more than the first order reconstruction in H. This may be the reason for which we do not loose the overall second order accuracy (this aspect may seem not evident in Figure 1.6, but we have to keep in mind that spatial step h is usually much smaller than mean curvature radius of the boundary).

1.2.2 1D discretization

In order to study some properties of the discretization, such as ill conditioned extrapolation, we analyze the easier 1D problem:

Model problem 2

$$\begin{aligned} -u'' &= f \ in \ [a,b] \subseteq [-1,1] && (1.13)\\ u(a) &= g_a && (1.14)\\ u'(b) &= g_b && (1.15) \end{aligned}$$

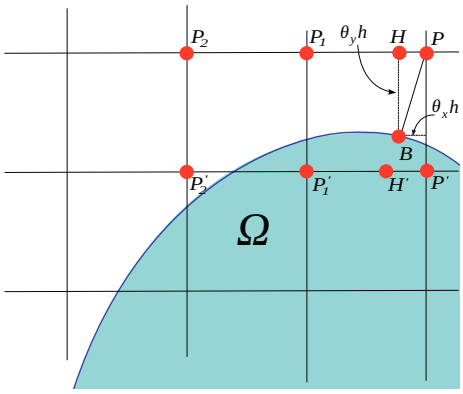


Fig. 1.5: Six-point stencil for the second order reconstruction of $\partial u/\partial x$ in H_P by linear interpolation of 1D reconstructions of $\partial u/\partial x$ in H and H'.

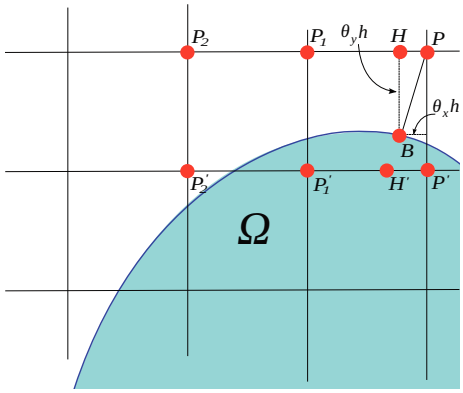


Fig. 1.6: P_2 is an *outside point*, then is not involved in calculus of the reconstruction. Anyway, the point H_P is closer to the line $P'-P_2'$ with respect to the line $P-P_2$ and the first-order reconstruction in H has a very small weight in the linear interpolation.

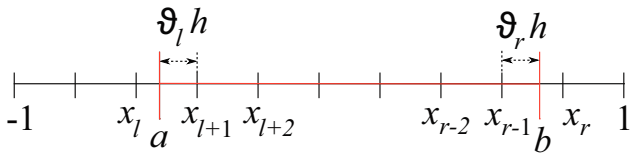


Fig. 1.7: Discretization of the domain in 1D.

Let l and r be such that $x_l \leq a < x_{l+1}$, $x_{r-1} < b < x_r$, and $\vartheta_l = (x_{l+1} - a)/h$, $\vartheta_r = (b - x_{r-1})/h$ (see Figure 1.7).

In one dimension, the nine-point stencil of Fig. 1.4 reduces to a three-point stencil, and the discretization of Section 1.2 consists in a simple 1D extrapolation. The linear system becomes:

$$-\frac{1}{h^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) = f_i \quad i = l+1, \ldots, r-1 \tag{1.16}$$

$$(1+\vartheta_l)\frac{\vartheta_l}{2}u_l + (1+\vartheta_l)(1-\vartheta_l)u_{l+1} - (1-\vartheta_l)\frac{\vartheta_l}{2}u_{l+2} = g_a \tag{1.17}$$

$$\frac{1}{h}\left(u_{r-1} - u_{r-2} + (u_{r-2} - 2u_{r-1} + u_r)\left(\frac{1}{2} + \vartheta_r\right)\right) = g_b \tag{1.18}$$

We observe that Eq. 1.17 is like performing a quadratic extrapolation of the numerical solution in x_{l+1} and a to obtain the value in x_l. When ϑ_l is very small, this quadratic extrapolation is not too much accurate and leads to an ill-conditioned system. Then, the error is quite over the best-fit line

(which maintains a second order accuracy slope), as we can observe in Figure 1.13 of Example 1.4.1.

1.3 Some extensions

In this section we show how the approach can be used in more general cases, which are of great relevance in practical applications. In particular, we consider extension of the method to variable coefficients, anisotropic operators, and domains with singularities.

1.3.1 Robin boundary conditions

So far we have considered the model problem 1 with mixed boundary conditions. Indeed, the same approach to discretize the boundary conditions works with the more general Robin boundary conditions. In such case, the model problem 1 becomes:

Model problem 3

$$-\Delta u = f \quad \text{in } \Omega \tag{1.19}$$

$$\alpha u + \beta \frac{\partial u}{\partial n} = g \quad \text{on } \Gamma \tag{1.20}$$

where $\Gamma = \partial\Omega$*, while* $\alpha, \beta \colon \Gamma \to \mathbb{R}$ *and* $g \colon \Gamma \to \mathbb{R}$ *are assigned functions, with* $\alpha, \beta \geq 0$.

The model problem 1 is a particular case of the model problem 3, choosing $\alpha = \chi_{\Gamma_D}$ and $\beta = \chi_{\Gamma_N}$, where $\chi_A \colon \Gamma \to \mathbb{R}$ denotes the characteristic function over $A \subseteq \Gamma$. The discretization of the model problem 3 is obtained straightforwardly from the discretization of the model problem 1 described in Section 1.2.

1.3.2 Variable diffusion coefficient

The whole procedure (numerical method and convergence proof) can be extended to the case of variable diffusion coefficient, i.e. we replace the Laplacian operator Δu by a more general operator $\nabla \cdot (\gamma \nabla u)$. The model problem 1 becomes:

Model problem 4

$$\begin{aligned} -\nabla \cdot (\gamma \nabla u) &= f \quad && in\ \Omega \\ u &= g_D \quad && on\ \Gamma_D \\ \frac{\partial u}{\partial n} &= g_N \quad && on\ \Gamma_N \end{aligned}$$

where $\gamma \colon \Omega \to \mathbb{R}$ *is an assigned positive function.*

Observe that actually Neumann boundary condition should be $\gamma \partial u/\partial n = g_N$, but we can rename $g_N \colon = g_N/\gamma$ and consider the usual Neumann condition we dealt up to now.

Let us suppose we know the function γ only in the grid points and let $\gamma_{i,j} = \gamma(jh, ih)$. The discretized system is obtained in the same manner as the one described in Sec. 1.2, according to replace (1.6) with:

$$\begin{aligned} -\frac{1}{h^2} \big(& \gamma_{i+1/2,j} \left(u_{i+1,j} - u_{i,j}\right) + \gamma_{i-1/2,j} \left(u_{i-1,j} - u_{i,j}\right) \\ & + \gamma_{i,j+1/2} \left(u_{i,j+1} - u_{i,j}\right) + \gamma_{i,j-1/2} \left(u_{i,j-1} - u_{i,j}\right) \big) = f_{i,j} \end{aligned}$$

where $\gamma_{i\pm 1/2,j} = (\gamma_{i,j} + \gamma_{i\pm 1,j})/2$, $\gamma_{i,j\pm 1/2} = (\gamma_{i,j} + \gamma_{i,j\pm 1})/2$.

In the case of smooth coefficient γ, the method maintains the prescribed second order accuracy (see Examples 1.4.3, 1.4.4, 1.4.7, 1.4.13). If the coefficient γ is not smooth, then a loss of accuracy is observed. The important case of piecewise smooth coefficient, which models, for example, a system composed by different materials separated by an interface, and second order accuracy is desired, is described in Chapter 3.

1.3.3 Anisotropic case

In this section we describe how to extend this technique to the case of variable matrix coefficient. Let us consider the Model Problem:

Model problem 5

$$\begin{aligned} -\nabla \cdot (A \nabla u) &= f \ in\ \Omega && (1.21) \\ u &= g_D \ on\ \Gamma_D && (1.22) \\ \nabla u \cdot \mathbf{N} &= g_N \ on\ \Gamma_N && (1.23) \end{aligned}$$

where $A = (a_{i,j})_{i,j=1,2} : \Omega \to \mathbb{R}^{2\times 2}$ *is a symmetric positive definite variable matrix, and* $\mathbf{N} = A \cdot \mathbf{n}$ *is the co-normal vector to the boundary.*

Expanding 1.23, we obtain:

$$-\left(\frac{\partial a_{11}}{\partial x}\frac{\partial u}{\partial x}+\frac{\partial a_{12}}{\partial x}\frac{\partial u}{\partial y}+\frac{\partial a_{12}}{\partial y}\frac{\partial u}{\partial x}+\frac{\partial a_{22}}{\partial y}\frac{\partial u}{\partial y}\right.$$
$$\left.+\,a11\frac{\partial^2 u}{\partial x^2}+2a12\frac{\partial^2 u}{\partial x\partial y}+a22\frac{\partial^2 u}{\partial y^2}\right)=f. \quad (1.24)$$

First, we observe that we need to know the value of A also in ghost points, due to the presence of coefficient derivatives. If we know the matrix A just in inside grid nodes, its ghost value can be obtained by second order accurate extrapolation [12], or if we know A also on the boundary, we can enforce such boundary value as a boundary condition, obtaining ghost values by biquadratic extrapolation, using the usual nine-point stencil in Upwind direction.

All the spatial derivatives appearing in (1.24) can be discretized by central differences. In this case, the stencil will be composed by nine points instead of five, due to the presence of the mixed derivative term $\partial^2 u/\partial x\partial y$, which can be discretized by the standard stencil:

$$\frac{\partial^2 u}{\partial x\partial y}\approx\frac{1}{4\,h^2}\begin{bmatrix}1&0&1\\0&0&0\\1&0&1\end{bmatrix}u_{i,j}=\frac{u_{i+1,j+1}+u_{i-1,j-1}-u_{i+1,j-1}-u_{i-1,j+1}}{4\,h^2}.$$

The whole stencil for an inside grid point results in a nine-points stencil. In order to keep the same set Γ_h of ghost points as in the isotropic case, we modify this nine-point stencil into a seven-point stencil for inside grid points close to the boundary (see Fig. 1.8). In details, if a grid point $(x,y)\in\Omega_h$ satisfies $|\phi(x,y)|<h$, then we compute the normal $\mathbf{n}\equiv(n_x,n_y)=\nabla\phi/|\nabla\phi|$. Then, if $n_x\cdot n_y\geq 0$, we use the following discretization for the mixed derivative:

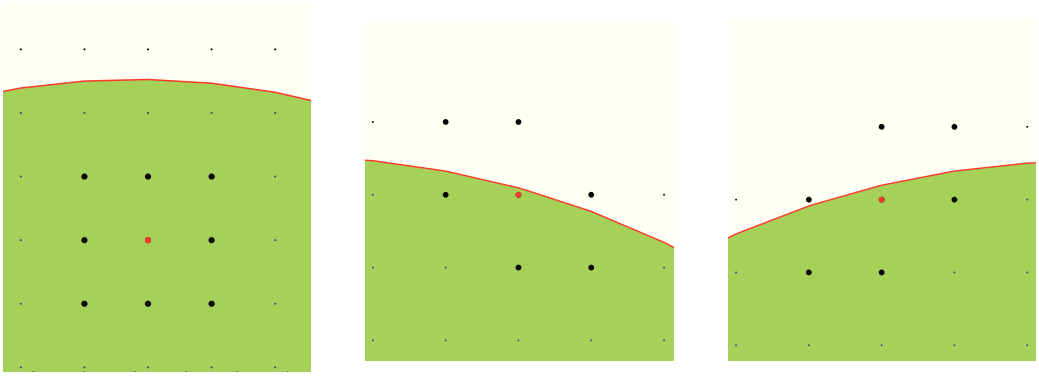
$$\frac{\partial^2 u}{\partial x\partial y}\approx\frac{1}{2\,h^2}\begin{bmatrix}-1&1&0\\1&-2&1\\0&1&-1\end{bmatrix}u_{i,j}$$

while, if $n_x\cdot n_y<0$, we use:

$$\frac{\partial^2 u}{\partial x\partial y}\approx\frac{1}{2\,h^2}\begin{bmatrix}0&1&1\\-1&2&1\\1&-1&0\end{bmatrix}u_{i,j}.$$

For an explanation of such stencils see for instance [105, pag. 264].

We observe that, for the Neumann boundary condition, we always find the *closest boundary point to the ghost point* and set the nine-point stencil in Upwind direction with respect to $\mathbf{n}$, but we discretize the numerical derivative along the co-normal direction $\mathbf{N}$ (see Fig. 1.9). This mismatching may lead to a non-Upwind discretization for the co-normal derivative, especially for strongly anisotropic operators.



$$\frac{1}{4\,h^2}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{4\,h^2}\begin{bmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1 \end{bmatrix} \qquad \frac{1}{4\,h^2}\begin{bmatrix} 0 & 1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$

Fig. 1.8: The stencil for the mixed derivative changes accordingly to the distance from the boundary and to the normal direction.

1.3.4 Sharp-edged domain

The method is designed for C^1 boundaries. Some extension is however possible when the boundary is only Lipschitz continuous, i.e. in presence of kinks due to the intersection or union of two smooth domains. We observe that, although it is possible to have an exact solution with singular gradients (due to the lack of regularity of the boundary, even if boundary conditions and sources are smooth), the numerical tests of Section 1.4 (1.4.10, 1.4.11 and 1.4.7) are limited to cases where the solution is smooth.

Let us describe how to modify the method near kink points due to intersection of two domains (if we are near the union of two domains the approach is similar). In particular, let us consider the two-dimensional case. Let

$$\Omega_1 = \left\{(x,y) \in \mathbb{R}^2 \colon \phi_1(x,y) < 0\right\}, \quad \Omega_2 = \left\{(x,y) \in \mathbb{R}^2 \colon \phi_2(x,y) < 0\right\}$$

be two domains with non-empty intersection, and let $\Omega = \Omega_1 \cap \Omega_2$ as in Fig. 1.10.

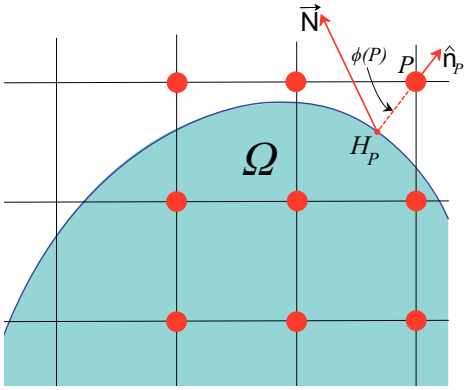


Fig. 1.9: The stencil is in Upwind direction with respect to the normal vector $\mathbf{n}_P$ computed in P, while the discretization is performed along the co-normal direction $\mathbf{N}$.

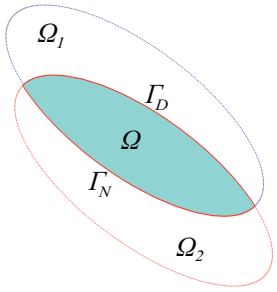


Fig. 1.10: Domain as intersection of two smooth domains (Section 1.3.4).

The domain Ω is described by the level set function $\phi = \max\{\phi_1, \phi_2\}$ ($\phi = \min\{\phi_1, \phi_2\}$ in case of union of domains) and the approach proposed so far easily applies also in this case, but a special treatment is needed close to the kink points of the intersection $\partial\Omega_1 \cap \partial\Omega_2$, because if we use the numerical method proposed so far using the level-set function $\phi = \max\{\phi_1, \phi_2\}$, the slope of the best-fit line in the accuracy test is just first order (even for L^1-error), as we can see in the Ex. 1.4.10, and this is due to the discontinuity of $\nabla\phi$ near the intersection between the two domains. In fact, referring to the Fig. 1.12, let us consider the ghost point P. Computing the boundary closest point using the level-set function ϕ we obtain H and the segment PH is not normal to the boundary. Moreover, the normal to the boundary in the point H computed using the stencil $P, Q, P_1, \ldots, P_7$ is incorrect (the arrow is normal to the approximation of the zero level-set of ϕ showed by the dashed red line). We must use a different approach if we want to preserve second order accuracy.

We refer to Fig. 1.11. Let us suppose we want to compute the solution in the ghost point P_3. To do that, we must use the (reduced) stencil P, P_1, P_2, P_3. We involve the grid point P_2 in our computation and we must write an additional equation for it. Computing the boundary closest point using the level-set function ϕ may produce an incorrect result, because ϕ has a discontinuous gradient near the kink point K. Using separately the two level-set functions ϕ_1 and ϕ_2, we can compute the two boundary closest points (respectively to $\partial\Omega_1$ and $\partial\Omega_2$) M and N. Neither M nor N belongs to $\partial\Omega$, then we cannot use any boundary condition from them. Therefore,

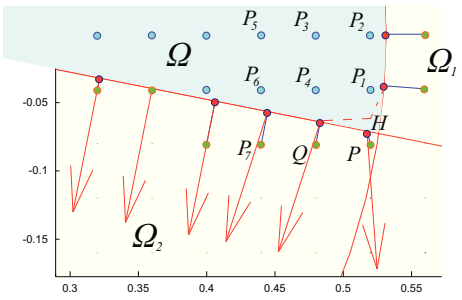


Fig. 1.11: Incorrect direction of the normal near the intersection between the two domains. This is due to the discontinuity of $\nabla\phi$.

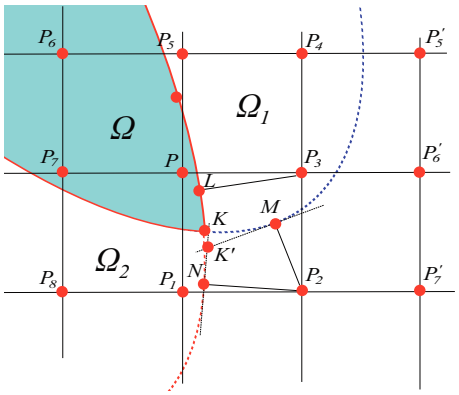


Fig. 1.12: Treatment of the kink points described in Section 1.3.4.

we find the point K' from P_2, N, M solving the system:

$$\begin{cases} (M - K') \cdot (M - P_2) &= 0 \\ (N - K') \cdot (N - P_2) &= 0 \end{cases}$$

The point K' is a second order accurate approximation of the exact kink point K. Using the nine-point stencil $P, P_1, \ldots, P_8$, we enforce the boundary condition in K to match the biquadratic interpolation in K'. In details, if $K' \in \partial\Gamma_D$, we enforce $\tilde{u}(K') = g_D(K)$, otherwise $(\nabla\tilde{u} \cdot \mathbf{n}_i)(K') = g_N(K)$, where $\mathbf{n}_i = \nabla\tilde{\phi}_i / \left|\nabla\tilde{\phi}_i\right|$, while $\tilde{u}$ and $\tilde{\phi}_i$ are the biquadratic interpolant respectively of u and ϕ_i in the nine-point stencil. We have to choose either $i = 1$ or $i = 2$. In order to prevent as much as possible the Upwind direction of the stencil, we choice i such that $\mathbf{n}_i$ is closest to $P_2 - K'$, i.e., the scalar product $\mathbf{n}_i \cdot (P_2 - K')$ is minimum. A numerical test over a cardioid shaped domain with one level-set is performed in Ex. 1.4.9. The accuracy order in L^∞ is about 1.65. In Ex. 1.4.10 we compare the two approaches (one unique level-set with sharp gradient and two smooth level-set functions), showing that the second one improves the accuracy, reaching second order.

A numerical test (Ex. 1.4.10) on the use of this technique is provided, while a more complicated geometry with both union and intersection of domains (the pentagon star) is treated in Ex. 1.4.11.

1.4 Numerical tests

This section provides numerical tests showing second order accuracy in 1D, 2D and 3D. We will start from 1D case, showing that very small ϑ does not degrade the convergence (Sec. 1.2.2).

Then we switch to 2D case, starting with a simple geometry, such as the circular domain with both mixed (Example 1.4.2) and Robin (Example 1.4.3) boundary condition. In the first example, we also show the second order accuracy of the gradient of the numerical solution (Remark of the Section 1.2.1.2); in the second example we will see how reduction of the stencil described in Section 1.2.1.3 does not degrade the second order accuracy. After, we provide examples with flower-shaped domain (Fig. 1.20) with mixed and Robin boundary conditions (Ex. 1.4.4). Another example over a stronger flower-shaped domain (Fig. 1.23) is provided (Ex. 1.4.6). In Example 1.4.5 a comparison between our method and the method proposed in [46] is provided. Then, a numerical test over a domain with a saddle point (Ex. 1.4.7) is provided as well. The anisotropic case with a variable matrix coefficient described in Section 1.3.3 is tested in Example 1.4.8. Therefore, we analyze the case of domain with boundary kink points, making a comparison between the method with a unique level-set function (whose gradient is discontinuous near kink points) and the approach with two smooth level-set functions (described in Section 1.3.4). First example is the cardioid-shaped domain (see Figure 1.29). Even though we use a unique level-set function, we observe an accuracy close to 1.65 for the L^∞-error. In case of more complicated geometries we observe a substantial loss of accuracy: in Ex. 1.4.10 the domain is the intersection between two smooth domains and a comparison between the two methods (unique level-set function and two smooth level-set functions) is provided, showing as the second one provides second order. Then, in Ex. 1.4.11 a test in presence of more kink points (pentagon star) taken from [110] is adopted. Last two numerical tests are in the 3D case (Examples 1.4.12, 1.4.13).

In all tests we choose an analytical expressions of the exact solution u and diffusion coefficient γ and reconstruct the data of the problem (f, g_D and g_N for mixed boundary conditions, or f and g for Robin boundary conditions). We also choose an analytical expression of the level-set function ϕ_0 and reinitialize it to signed distance function ϕ. Recall that we suppose to know ϕ only in grid nodes.

Most of the numerical tests was taken from [46], [47], [87].
Numerical tests (in 2D and 3D) are related to as general as possible case, namely variable diffusion coefficient problem with Robin boundary conditions:

$$-\nabla \cdot (\gamma \nabla u) = f \quad \text{in } \Omega \tag{1.25}$$

$$\alpha u + \beta \frac{\partial u}{\partial n} = g \quad \text{on } \Gamma \tag{1.26}$$

where $\Gamma = \partial\Omega$, while $f\colon \Omega \to \mathbb{R}$, $\gamma\colon \Omega \to \mathbb{R}$, $\alpha\colon \Gamma \to \mathbb{R}$, $\beta\colon \Gamma \to \mathbb{R}$ and $g\colon \Gamma \to \mathbb{R}$ are assigned functions. In each test it is specified if it reduces to mixed boundary condition (we always choose Dirichlet boundary condition for all points $(x, y) \in \Gamma$ such that $x \leq 0$, Neumann boundary condition otherwise), or constant diffusion coefficient ($\gamma = 1$).

The linear system is solved by the multigrid technique described later in Chapter 2.
In all the following tables, the values in the column of the accuracy order are computed as

$$\frac{\log\left(e_{i-1}/e_i\right)}{\log\left(N_i/N_{i-1}\right)},$$

where e_i is the error indicated in the i-th row.

1.4.1 1D Numerical test

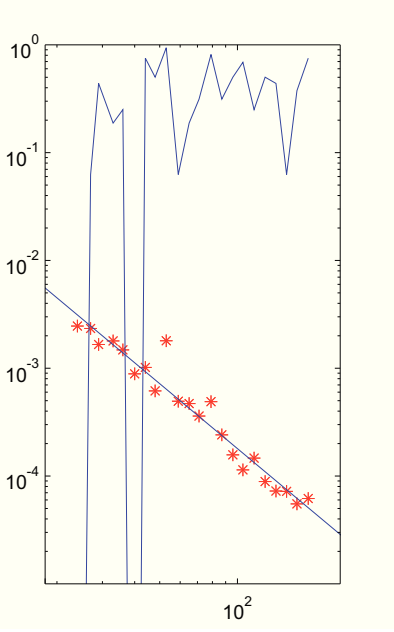


Fig. 1.13: Representation of the L^∞-error reported in Table 1.14 (Example 1.4.1). The slope of the best-fit lines is $s = -2.00$. The piece-wise straight line is ϑ_l. When ϑ_l is very small, the error is larger.

N	L^∞-error	order	ϑ_l
24	$2.46 \cdot 10^{-3}$	-	$1.19 \cdot 10^{-13}$
27	$2.33 \cdot 10^{-3}$	0.44	0.06
29	$1.66 \cdot 10^{-3}$	4.79	0.44
33	$1.79 \cdot 10^{-3}$	-0.59	0.19
36	$1.48 \cdot 10^{-3}$	2.20	0.25
40	$8.85 \cdot 10^{-4}$	4.85	$2.02 \cdot 10^{-13}$
44	$1.01 \cdot 10^{-3}$	-1.43	0.75
48	$6.15 \cdot 10^{-4}$	5.76	0.50
53	$1.79 \cdot 10^{-3}$	-10.81	0.94
59	$4.93 \cdot 10^{-4}$	12.04	0.06
65	$4.69 \cdot 10^{-4}$	0.52	0.19
71	$3.59 \cdot 10^{-4}$	3.02	0.31
79	$4.88 \cdot 10^{-4}$	-2.88	0.81
87	$2.40 \cdot 10^{-4}$	7.35	0.31
96	$1.57 \cdot 10^{-4}$	4.34	0.50
105	$1.14 \cdot 10^{-4}$	3.58	0.69
116	$1.46 \cdot 10^{-4}$	-2.53	0.25

Fig. 1.14: Example 1.4.1.

Let us consider the one-dimensional model problem 2 (1.13)-(1.15) with an exact solution $u = e^{x^2}$, over $[-a, a] \subseteq [-1, 1]$, with $a = 0.3725 + 10^{-15}$, in order to have a very small ϑ_l when the number of grid points is $n = 24$ or $n = 40$. Table 1.14 shows the numerical results, and the convergence is guaranteed even with very small ϑ_l (but the accuracy order computed by subsequent error data is small as ϑ_l is small). Figure 1.13 shows the second order slope of the best-fit line for the L^∞-error. In this figure we also report the values of ϑ_l, showing that the error is larger for smaller values of ϑ_l.
Remark Table 1.14 shows some negative convergence order in fifth column. In fact, small values of ϑ cause inaccuracy that leads to negative convergence order. Anyway, in this numerical test two subsequent grids have almost the same number of points. Then, the accuracy of the local approximation oscillates due to interpolation errors in points that are not grid points. Such errors decrease on average when the grid is refined, but may fluctuate for grids with almost the same number of points, leading to a non-monotone behavior.

1.4.2 2D Numerical test: Circular domain with mixed boundary condition

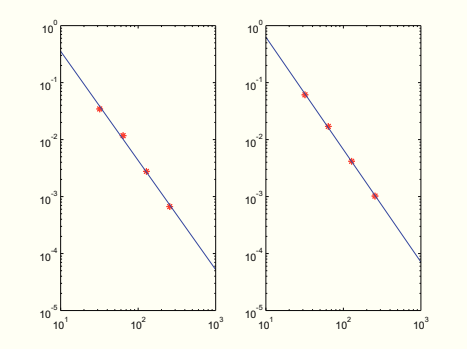


Fig. 1.15: Representation of the L^∞-error for the solution (left) and the gradient (right) reported in Table 1.1 (Example 1.4.2). The slope of the best-fit lines is respectively $s = -1.92$ and $s = -1.97$.

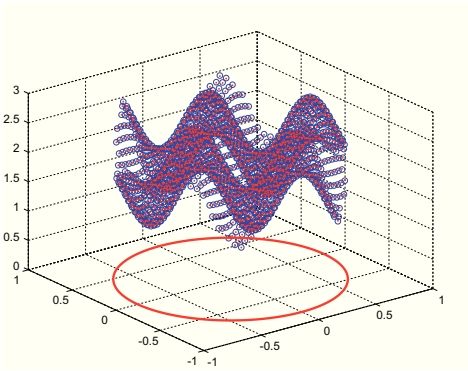


Fig. 1.16: Numerical solution of the Example 1.4.2 with $n = 64$ over a circular domain. The (blue) circle points represent the exact solution, while (red) dot points represent the numerical solution.

Let us consider the two-dimensional problem (1.25),(1.26) with $\gamma = 1$ and with an exact solution $u = 2 + \cos(2\pi x) * \sin(2\pi y)$, over a circular domain centered at the origin and with radius $r = 0.813$. We test the case of mixed

boundary condition and show the second order accuracy of the gradient. Table 1.1 shows the numerical results, while in Figure 1.16 we depict the solution. Figure 1.15 shows the second order slope of the best-fit line for the L^∞-error of the solution and the gradient. The errors in the gradient are computed in the seminorm of the Sobolev space $W^{1,q}(\Omega)$, i.e.

$$|u - u_h|_{W^{1,q}(\Omega)} = \|\nabla u - \nabla_h u_h\|_{L^q(\Omega)} .$$

for $q = 2$. The gradient of the numerical solution is computed in inside points by central differences.

Table 1.1: Example 1.4.2.

No. of grid points	$\|\mathbf{u} - u_h\|_\infty$	order	$\|\nabla\mathbf{u} - \nabla u_h\|_2$	order
$32 \cdot 32$	$3.45 \cdot 10^{-2}$	-	$6.10 \cdot 10^{-2}$	-
$64 \cdot 64$	$1.17 \cdot 10^{-2}$	1.56	$1.70 \cdot 10^{-2}$	1.84
$128 \cdot 128$	$2.77 \cdot 10^{-3}$	2.08	$4.16 \cdot 10^{-3}$	2.03
$256 \cdot 256$	$6.67 \cdot 10^{-4}$	2.05	$1.02 \cdot 10^{-3}$	2.03

1.4.3 Circular domain with Robin boundary condition

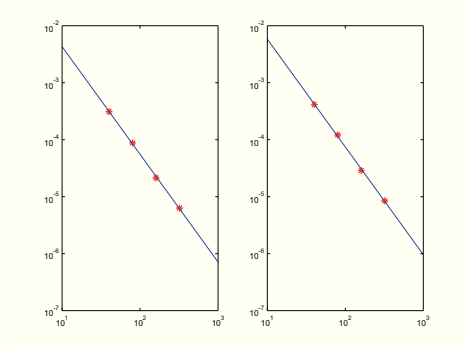


Fig. 1.17: Representation of the L^1-error (left) and L^∞-error (right) reported in Table 1.2 (Example 1.4.3). The slope of the best-fit lines is respectively $s = -2.07$ and $s = -2.06$.

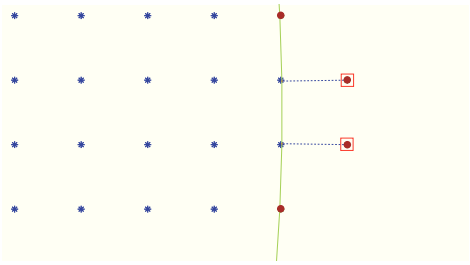


Fig. 1.18: Reduction of the stencil (described in Section 1.2.1.3). The star-shaped grid points are the inside grid points, the circle points are the ghost points, the solid line is the boundary of Ω, the dashed lines are the normal vectors to the boundary. Stencil for ghost points must be reduced as described in Section 1.2.1.3

In this Example we show that the stencil reduction described in Section 1.2.1.3 does not degrade the second order accuracy. In fact, when $n = 80$, the discretization requires a reduction of the stencil in a ghost point, as we

can see in Figure 1.18.
Let us consider the two-dimensional problem (1.25),(1.26) with $\gamma = 2 + \sin(xy)$ and with an exact solution $u = e^{xy}$, over a circular domain centered at the origin and with radius $r = 0.813$. We have Robin boundary condition with $\alpha(x, y) = \beta(x, y) = 0.5$. Table 1.2 shows the numerical results. Figure 1.17 shows the second order slope of the best-fit line for the L^1-error and L^∞-error.

Table 1.2: Example 1.4.3.

No. of grid points	L^1-error	order	L^∞-error	order
$40 \cdot 40$	$3.10 \cdot 10^{-4}$	-	$4.13 \cdot 10^{-4}$	-
$80 \cdot 80$	$8.81 \cdot 10^{-5}$	1.82	$1.21 \cdot 10^{-4}$	1.77
$160 \cdot 160$	$2.14 \cdot 10^{-5}$	2.04	$2.88 \cdot 10^{-5}$	2.07
$320 \cdot 320$	$6.31 \cdot 10^{-6}$	1.76	$8.46 \cdot 10^{-6}$	1.77

1.4.4 Flower shaped domain with mixed boundary condition

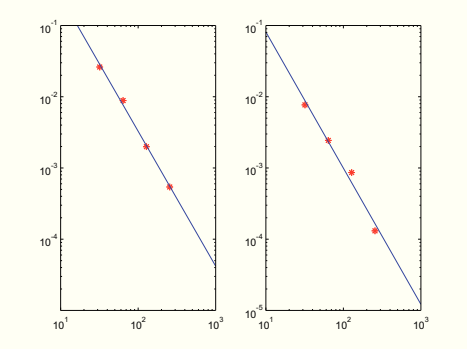


Fig. 1.19: Representation of the L^∞-error reported in Table 1.3 (Example 1.4.4) in case of mixed (left) and Robin (right) boundary conditions. The slope of the best-fit lines is respectively $s = -1.89$ and $s = -1.91$.

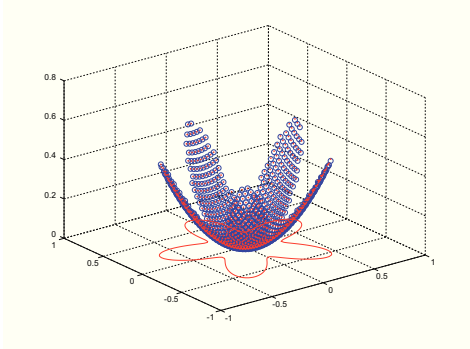


Fig. 1.20: Numerical solution of the Example 1.4.4 with $n = 64$ over the flower-shaped domain of Figure 1.20. The (blue) circle points represent the exact solution, while (red) dot points represent the numerical solution.

Let us consider the two-dimensional problem (1.25),(1.26) over a flower-shaped domain represented in Figure 1.20 and defined by the zero level-set

$$\phi = r - 0.5 - \frac{y^5 + 5x^4y - 10x^2y^3}{5r^5}, \quad r = \sqrt{x^2 + y^2}.$$

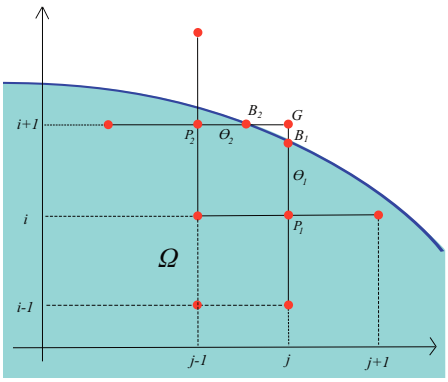


Fig. 1.21: Comparison between our method and the method proposed in [46].

We test the case of mixed boundary conditions (test (M)) with $\gamma = 1$ and with an exact solution $u = x^2+y^2$, and the case of Robin boundary conditions with $\alpha(x, y) = \beta(x, y) = 0.5$ (test (R)), with $\gamma = 2 + \sin(xy)$ and with an exact solution $u = e^{xy}$. Table 1.3 shows the numerical results. Figure 1.19 shows the second order slope of the best-fit line for the L^∞-error. Fig. 1.20 shows the numerical solution over the domain.

Table 1.3: Example 1.4.4.

No. of grid pts	L^∞-error $((M))$	order	L^∞-error $((R))$	order
$32 \cdot 32$	$2.60 \cdot 10^{-2}$	-	$7.65 \cdot 10^{-3}$	-
$64 \cdot 64$	$8.80 \cdot 10^{-3}$	1.56	$2.43 \cdot 10^{-3}$	1.66
$128 \cdot 128$	$1.98 \cdot 10^{-3}$	2.15	$8.59 \cdot 10^{-4}$	1.50
$256 \cdot 256$	$5.39 \cdot 10^{-4}$	1.88	$1.31 \cdot 10^{-4}$	2.72

1.4.5 Flower shaped domain with Dirichlet boundary condition: comparison with the method [46]

In this Example we compare the numerical results between our method and the method proposed in [46]. We recall the method proposed in [46] apply to Dirichlet boundary condition, then we have to reduce our technique to the case of pure Dirichlet condition to make comparison. We will see that our method, even though is second order accurate, provide a greater error than in [46]. A possible explanation is that the method proposed in [46] may define more than one value in each ghost point, unlike our method, which defines just one value for each ghost point. In details, referring to the Fig.

1.21, the method [46] consists in discretizing the Laplace operator (1.6) in the inside grid points P_1 and P_2, and using the linear interpolation to define the numerical value in the ghost points belonging to the five-point stencil. For instance, the ghost point G belongs either to the stencil centered in P_1 or in P_2. Then we have two value in G:

$$\vartheta_1 u_G^{(1)} + (1-\vartheta_1) u_{P_1} = g(B_1)$$

$$\vartheta_2 u_G^{(2)} + (1-\vartheta_2) u_{P_2} = g(B_2)$$

Both equations are solved respectively for $u_G^{(1)}$ and $u_G^{(2)}$, and these value are plugged in (1.6), written resp. for P_1 and P_2.

Let us consider the two-dimensional problem (1.25),(1.26) with $\gamma = 1$ and with an exact solution $u = x^2 + y^2$, over the flower-shaped domain of Ex. 1.4.4. We have Dirichlet boundary condition on all the boundary. Table 1.4 shows the numerical results and compare them with the results taken from [46].

Table 1.4: These two tables refer to the Example 1.4.5 (top) and the numerical results from [46] (bottom).

No. of grid points	L^1-error	order	L^∞-error	order
$101 \cdot 101$	$1.8536 \cdot 10^{-4}$	-	$5.8390 \cdot 10^{-4}$	-
$201 \cdot 201$	$5.4324 \cdot 10^{-5}$	1.7707	$1.0523 \cdot 10^{-4}$	2.4722
$401 \cdot 401$	$1.3104 \cdot 10^{-5}$	2.0515	$2.5508 \cdot 10^{-5}$	2.0445

No. of grid points	L^1-error	order	L^∞-error	order
$101 \cdot 101$	$7.329 \cdot 10^{-5}$	-	$9.777 \cdot 10^{-5}$	-
$201 \cdot 201$	$1.776 \cdot 10^{-5}$	2.04	$2.427 \cdot 10^{-5}$	2.01
$401 \cdot 401$	$4.714 \cdot 10^{-6}$	1.92	$6.178 \cdot 10^{-6}$	1.97

1.4.6 Strong flower-shaped domain

Let us consider the two-dimensional problem (1.25),(1.26) with $\gamma = 1$ and with an exact solution $u = x^2 + y^2$, over a flower-shaped domain showed in Figure 1.23 and defined by the zero level-set of

$$\phi = r - 0.5 - \frac{y^5 + 5x^4 y - 10x^2 y^3}{3r^5}, \quad r = \sqrt{x^2 + y^2}.$$

We have Dirichlet boundary condition for all points $(x, y) \in \partial\Omega$ such that $x \leq 0$, Neumann boundary condition otherwise. Table 1.5 shows the numerical results, while in Figure 1.23 we depict the solution. Figure 1.22 shows the second order slope of the best-fit line for the L^1-error and L^∞-error.

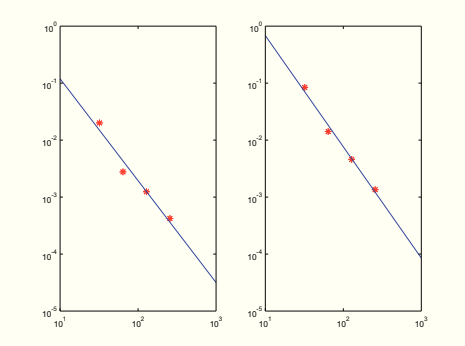


Fig. 1.22: Representation of the L^1-error (left) and L^∞-error (right) reported in Table 1.5 (Example 1.4.6). The slope of the best-fit lines is respectively $s = -1.79$ and $s = -1.95$.

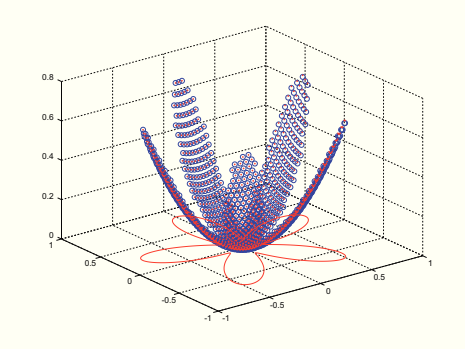


Fig. 1.23: Numerical solution of the Example 1.4.6 with $n = 64$ over the flower-shaped domain of Figure 1.23. The (blue) circle points represent the exact solution, while (red) dot points represent the numerical solution.

Table 1.5: Example 1.4.6.

No. of grid points	L^1-error	order	L^∞-error	order
$32 \cdot 32$	$2.00 \cdot 10^{-2}$	-	$8.42 \cdot 10^{-2}$	-
$64 \cdot 64$	$2.77 \cdot 10^{-3}$	2.86	$1.41 \cdot 10^{-2}$	2.57
$128 \cdot 128$	$1.24 \cdot 10^{-3}$	1.16	$4.59 \cdot 10^{-3}$	1.62
$256 \cdot 256$	$4.21 \cdot 10^{-4}$	1.56	$1.35 \cdot 10^{-3}$	1.77

1.4.7 Domain with a saddle point

Let us consider the two-dimensional problem (1.25),(1.26) with $\gamma = 2 + \sin(\pi x)\cos(\pi y)$ and with an exact solution $u = 2 + \cos(\pi x)\sin(\pi y)$ over the domain showed in Figure 1.25 and defined by the zero level-set of

$$\phi(x, y) = 16y^4 - x^4 - 32y^2 + 9x^2.$$

We test the case of mixed boundary conditions (test (M)) and the case of Robin boundary condition with $\alpha(x, y) = \beta(x, y) = 0.5$ for each $(x, y) \in \Gamma$ (test (R)). Table 1.6 shows the numerical results. Figure 1.24 shows the second order slope of the best-fit line for the L^∞-error.

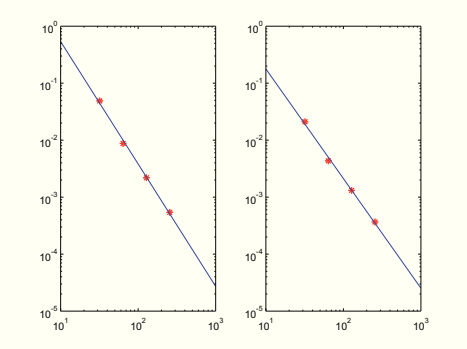


Fig. 1.24: Representation of the L^∞-error reported in Table 1.6 (Example 1.4.7) in case of mixed (left) and Robin (right) boundary conditions. The slope of the best-fit lines is respectively $s = -2.15$ and $s = -1.92$.

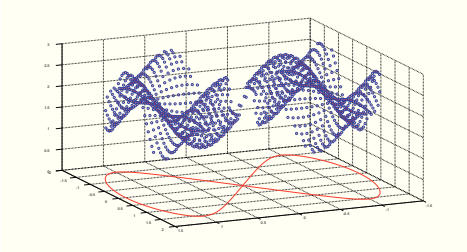


Fig. 1.25: Numerical solution of the Example 1.4.7 with $n = 64$ over a domain with a saddle point. The (blue) circle points represent the exact solution, while (red) dot points represent the numerical solution.

Table 1.6: Example 1.4.7.

No. of grid pts	L^∞-error (M)	order	L^∞-error (R)	order
$32 \cdot 32$	$4.87 \cdot 10^{-2}$	-	$2.09 \cdot 10^{-2}$	-
$64 \cdot 64$	$8.74 \cdot 10^{-3}$	2.48	$4.33 \cdot 10^{-3}$	2.27
$128 \cdot 128$	$2.18 \cdot 10^{-3}$	2.00	$1.31 \cdot 10^{-3}$	1.72
$256 \cdot 256$	$5.40 \cdot 10^{-4}$	2.01	$3.65 \cdot 10^{-4}$	1.84

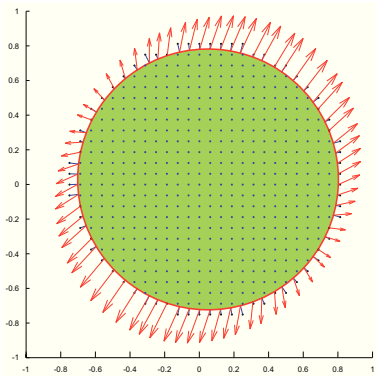


Fig. 1.26: Representation of the direction of the co-normal vectors (red arrays) with respect to the boundary.

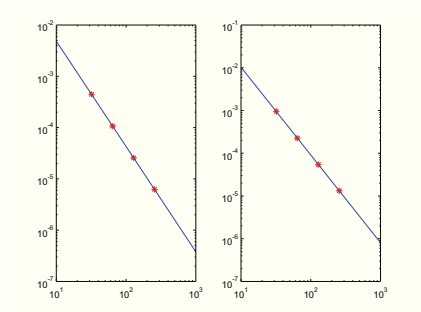


Fig. 1.27: Representation of the L^1-error (left) and L^∞-error (right) reported in Table 1.7 (Example 1.4.8). The slope of the best-fit lines is respectively $s = -2.05$ and $s = -2.05$.

1.4.8 Anisotropic case

This example is taken from [58] but it is applied to a simpler geometry. Let us consider the two-dimensional problem of Sec. 1.3.3 with

$$A = \begin{pmatrix} 2 + xy & 1 + xy \\ 1 + xy & 3 + xy \end{pmatrix}$$

and with an exact solution $u = \sin(\pi x)\cos(\pi y)$, over a circular domain centered at the origin and with radius $r = 0.813$. We have Dirichlet boundary condition for all points $(x, y) \in \partial\Omega$ such that $x \leq 0$, Neumann boundary condition otherwise. Table 1.7 shows the numerical results, while Figure 1.27 shows the second order slope of the best-fit line for the L^1-error and L^∞-error. In Fig. 1.26 we can observe the direction of the co-normal direction with respect to the boundary.

Table 1.7: Example 1.4.8.

No. of grid points	L^1-error	order	L^∞-error	order
$32 \cdot 32$	$4.4740 \cdot 10^{-4}$	-	$9.5728 \cdot 10^{-4}$	-
$64 \cdot 64$	$1.0732 \cdot 10^{-4}$	2.0596	$2.2661 \cdot 10^{-4}$	2.0787
$128 \cdot 128$	$2.5801 \cdot 10^{-5}$	2.0565	$5.4547 \cdot 10^{-5}$	2.0546
$256 \cdot 256$	$6.2621 \cdot 10^{-6}$	2.0427	$1.3355 \cdot 10^{-5}$	2.0301

1.4.9 Cardioid-shaped domain

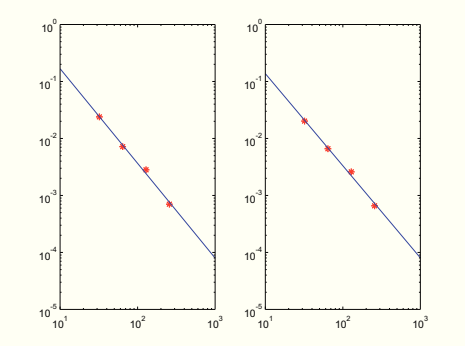


Fig. 1.28: Representation of the L^∞-error reported in Table 1.8 (Example 1.4.9) in case of mixed (left) and Robin (right) boundary conditions. The slope of the best-fit lines is respectively $s = -1.94$ and $s = -1.66$.

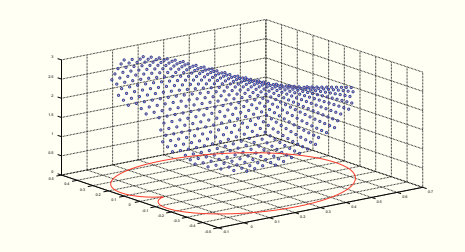


Fig. 1.29: Numerical solution of the Example 1.4.9 with $n = 80$ over the cardioid-shaped domain. The (blue) circle points represent the exact solution, while (red) dot points represent the numerical solution.

Let us consider the two-dimensional problem (1.25),(1.26) with $\gamma = e^{xy}$ and with an exact solution $u = 2 + \cos(\pi x)\sin(\pi y)$, over a cardioid-shaped domain (see Fig. 1.4.9) and defined by the zero level-set of

$$\phi(x, y) = \left(3\left(x^2 + y^2\right) - x\right)^2 - x^2 - y^2.$$

We test the case of mixed boundary condition (test (M)) and the case of Robin boundary condition with $\alpha(x,y) = \beta(x,y) = 0.5$ (test (R)). Table 1.8 shows the numerical results. Figure 1.28 shows a loss of accuracy for the L^∞-error because the kink in the boundary. In case of kink points the approach of using two level-set functions instead of one near kink points (see Sec. 1.3.4) may improve the accuracy (Ex. 1.4.10 and 1.4.11).

Table 1.8: Example 1.4.9.

No. of grid pts	L^∞-error (M)	order	L^∞-error (R)	order
$32 \cdot 32$	$4.87 \cdot 10^{-2}$	-	$2.09 \cdot 10^{-2}$	-
$64 \cdot 64$	$8.74 \cdot 10^{-3}$	2.48	$4.33 \cdot 10^{-3}$	2.27
$128 \cdot 128$	$2.18 \cdot 10^{-3}$	2.00	$1.31 \cdot 10^{-3}$	1.72
$256 \cdot 256$	$5.40 \cdot 10^{-4}$	2.01	$3.65 \cdot 10^{-4}$	1.84

1.4.10 Domain as intersection between two domains

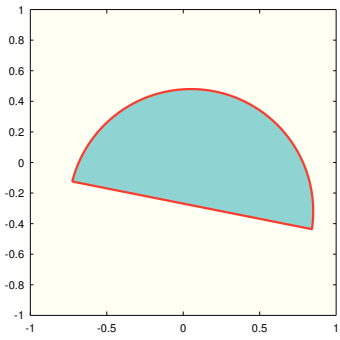


Fig. 1.30: Domain of the Ex. 1.4.10

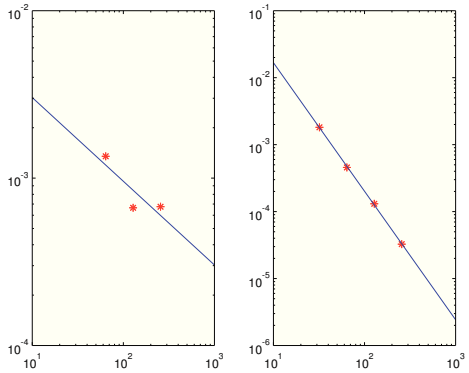


Fig. 1.31: Representation of the L^∞-error for test (a) (left) and test (b) (right), reported in Table 1.9 (Example 1.4.10). The slope of the best-fit lines is respectively $s = -0.50$ and $s = -1.92$.

Let us consider the two-dimensional problem

$$\begin{aligned} -\Delta u &= f \quad && \text{in } \Omega \\ u &= g_D \quad && \text{on } \Gamma_D \\ \frac{\partial u}{\partial n} &= g_N \quad && \text{on } \Gamma_N \end{aligned}$$

and with an exact solution $u = x^2 + y^2$, over the domain Ω depicted in Fig. 1.30, intersection between the circle centered at $(0, 0)$ and radius $r = 0.5$ and the half-plane of equation $0.0313 - y\cos(1/16) - x\sin(1/16) < 0$. We choose

$$\Gamma_D = \left\{(x, y)\colon x^2 + y^2 = r^2\right\} \cap \bar{\Omega}$$

$$\Gamma_N = \{(x, y)\colon 0.0313 - y\cos(1/16) - x\sin(1/16) = 0\} \cap \bar{\Omega}$$

In this example the domain has two kink points and the treatment is described in Section 1.3.4. A comparison between the method with one unique level-set function $\phi = \max\{\phi_1, \phi_2\}$ (test (a)) and two smooth level-set functions (test (b)) is provided.
Table 1.9 shows the numerical results, while Figure 1.31 shows the low order slope of the best-fit line for the L^1-error for test (a), while a better accuracy is observed for test (b) for the L^1-error and the L^∞-error. Loss of accuracy for test (a) is due to the inaccurate normal direction computed close to the intersection between the two domains. In fact, referring to the figure 1.12, the normal to the boundary in the point H is completely wrong (due to the discontinuity of $\nabla\phi$) and the method impose that along this direction the derivative of the solution should be g_N, while such derivative should be along the normal derivative. This lead to an incorrect value for the numerical solution in P (which even is $O(1)$) and to a completely wrong solution.

Table 1.9: Example 1.4.10.

test (a)

No. of grid points	L^1-error	order
$40 \cdot 40$	$1.35 \cdot 10^{-3}$	-
$80 \cdot 80$	$6.65 \cdot 10^{-4}$	1.02
$160 \cdot 160$	$6.75 \cdot 10^{-4}$	-0.021

test (b)

No. of grid points	L^1-error	order	L^∞-error	order
$20 \cdot 20$	$1.0320 \cdot 10^{-3}$	-	$1.8140 \cdot 10^{-3}$	-
$40 \cdot 40$	$2.6300 \cdot 10^{-4}$	1.9723	$4.5600 \cdot 10^{-4}$	1.9921
$80 \cdot 80$	$6.9000 \cdot 10^{-5}$	1.9304	$1.3100 \cdot 10^{-4}$	1.7995
$160 \cdot 160$	$1.7000 \cdot 10^{-5}$	2.0211	$3.3000 \cdot 10^{-5}$	1.9890

1.4.11 Boundary with more kink points: pentagon star

The geometry of this example is taken, for instance, from [110, 58]. Let us consider a more complicated geometry, given by the pentagon star depicted

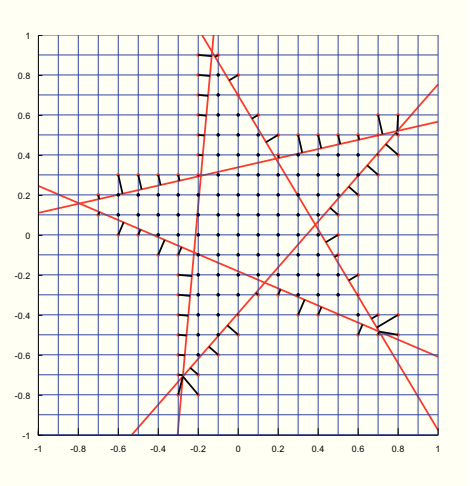


Fig. 1.32: Pentagon star of the Ex. 1.4.11.

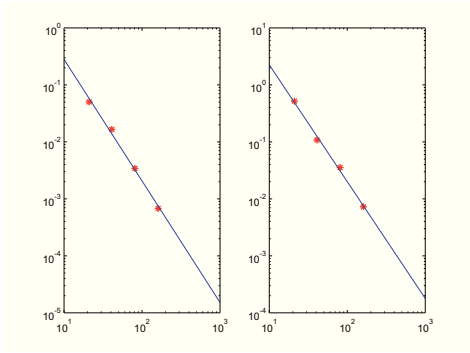


Fig. 1.33: Representation of the L^1-error (left) and L^∞-error (right) reported in Table 1.10 (Example 1.4.11). The slope of the best-fit lines is respectively $s = -2.13$ and $s = -2.05$.

in Fig. 1.32. Note that some kink points are due to the intersection of domains, while some others are due to the union of domains. The special treatment of the kink points is described in Sec. 1.3.4. The boundary of the star is composed by ten sides, which are aligned in groups of two. Then, the whole boundary may be described by five level-set functions:

$$\phi_i(x,y) = (x-x_0)\cos\vartheta_i + (y-y_0)\sin\vartheta_i - R\,\sin\vartheta_t/2, \quad i = 1,\dots,5$$

where $x_0 = 0.06$, $y_0 = 0.08$, $R = 6/7$, $\vartheta_t = \pi/5$ and $\vartheta_i = \vartheta_r + \vartheta_t/2 + 2(i-1)\vartheta_t$ with $\vartheta_r = \pi/14$. In this case, the domain can be identified by:

$$\Omega = \left\{ (x,y) \in D \colon \sum_{i=1}^{5} \phi_i(x,y) \geq 4 \right\}.$$

Let us choose an exact solution $u = \sin(2\pi x)\cos(2\pi y)$ and a coefficient $\gamma = 1$. We have Dirichlet boundary condition for all points $(x,y) \in \partial\Omega$ such that $x \leq 0$, Neumann boundary condition otherwise. Table 1.10 shows the numerical results. Figure 1.33 shows the second order slope of the best-fit line for the L^1-error and L^∞-error.

1.4.12 3D Numerical tests: mixed boundary condition

Let us consider the three-dimensional problem (1.25),(1.26) with $\gamma = 1$ and with an exact solution $u = e^{-(x+y+z)}$, over a spheric domain centered at the origin and with radius $r = 0.713$. We have Dirichlet boundary condition

Table 1.10: Example 1.4.11.

No. of grid points	L^1-error	order	L^∞-error	order
$20 \cdot 20$	$5.0140 \cdot 10^{-2}$	-	$5.1696 \cdot 10^{-1}$	-
$40 \cdot 40$	$1.6493 \cdot 10^{-2}$	1.6619	$1.0771 \cdot 10^{-1}$	2.3444
$80 \cdot 80$	$3.4183 \cdot 10^{-3}$	2.3114	$3.5336 \cdot 10^{-2}$	1.6369
$160 \cdot 160$	$6.7910 \cdot 10^{-4}$	2.3526	$7.2322 \cdot 10^{-3}$	2.3093

for all points $(x, y) \in \partial\Omega$ such that $x \leq 0$, Neumann boundary condition otherwise. Table 1.11 shows the numerical results. Figure 1.34 shows the second order slope of the best-fit line for the L^1-error and L^∞-error.

Table 1.11: Example 1.4.12.

No. of grid points	L^1-error	order	L^∞-error	order
$32 \cdot 32 \cdot 32$	$8.7107 \cdot 10^{-4}$	-	$4.5800 \cdot 10^{-3}$	-
$48 \cdot 48 \cdot 48$	$3.4747 \cdot 10^{-4}$	2.2667	$2.4111 \cdot 10^{-3}$	1.5824
$64 \cdot 64 \cdot 64$	$2.2000 \cdot 10^{-4}$	1.5887	$1.1274 \cdot 10^{-3}$	2.6424

1.4.13 3D Numerical tests: Robin boundary condition

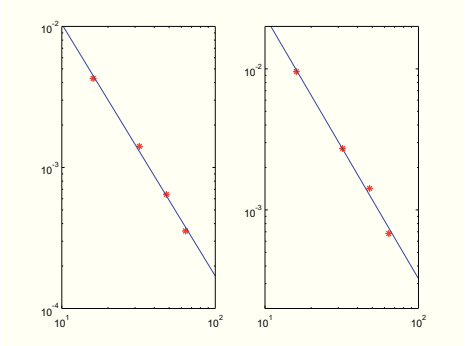


Fig. 1.34: Representation of the L^1-error (left) and L^∞-error (right) reported in Table 1.11 (Example 1.4.12). The slope of the best-fit lines is respectively $s = -2.00$ and $s = -1.99$.

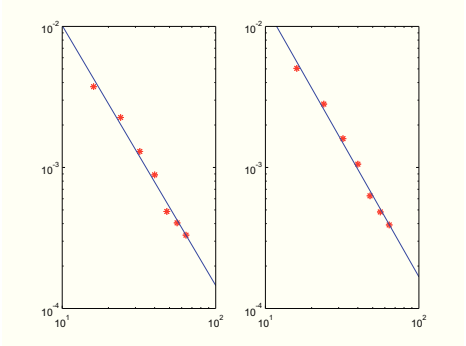


Fig. 1.35: Representation of the L^1-error (left) and L^∞-error (right) reported in Table 1.12 (Example 1.4.13). The slope of the best-fit lines is respectively $s = -2.06$ and $s = -2.10$.

Let us consider the three-dimensional problem (1.25),(1.26) with $\gamma = 2 + \sin(x + y + z)$ and with an exact solution $u = e^{-(x+y+z)}$, over a spheric

domain centered at the origin and with radius $r = 0.713$. We have Robin boundary condition with $\alpha(x, y, z) = \beta(x, y, z) = 0.5$. Table 1.12 shows the numerical results. Figure 1.35 shows the second order slope of the best-fit line for the L^1-error and L^∞-error.

Table 1.12: Example 1.4.13.

No. of grid points	L^1-error	order	L^∞-error	order
$32 \cdot 32 \cdot 32$	$1.2958 \cdot 10^{-3}$	1.9282	$1.5971 \cdot 10^{-3}$	1.9621
$40 \cdot 40 \cdot 40$	$8.8416 \cdot 10^{-4}$	1.7130	$1.0532 \cdot 10^{-3}$	1.8661
$48 \cdot 48 \cdot 48$	$4.8639 \cdot 10^{-4}$	3.2778	$6.2784 \cdot 10^{-4}$	2.8371
$56 \cdot 56 \cdot 56$	$4.0368 \cdot 10^{-4}$	1.2091	$4.8267 \cdot 10^{-4}$	1.7058
$64 \cdot 64 \cdot 64$	$3.3051 \cdot 10^{-4}$	1.4979	$3.9041 \cdot 10^{-4}$	1.5886

Chapter 2

Multigrid approach

In this chapter we provide a multigrid strategy to solve the linear system arising from the second order discretization described in Chapter 1. The main steps consists of the introduction of a suitable smoother, and the definition of the transfer (interpolation and restriction) operator. The multigrid we are going to describe is a geometric multigrid, and the Galerkin conditions (required in the Algebraic multigrid) are not satisfied. In practice, the interpolation and restriction operator are not the transpose each other (multiplied by a suitable constant) and the coarse grid operator in constructed in the same way as in the fine grid, without taking into account the transfer operators.

The main goal of this study is to find a general strategy to make the multigrid convergent and to attain the optimal convergent factor (i.e. the one predicted by the Local Fourier Analysis). We are not interested in this thesis in constructing the most efficient multigrid strategy, that is, we only want to describe how to treat the ingredients of the multigrid in presence of an arbitrary domain and in relation with the boundary conditions, in order to limit the degradation of the convergence factor due to boundary effects. In a few words, we describe the boundary treatment in order to gain the convergence factor predicted by the Local Fourier Analysis for the smoother used for inner equations, independently on the kind of smoother. Therefore, we just use for simplicity a Gauss-Seidel Lexicographic smoother, in place of the more efficient Red-Black Gauss-Seidel smoother, although it can be easily implemented with our method. For the same argument, we only study the convergence factor of the W-cycle algorithm, without implement the more efficient Full Multigrid.

2.1 Failure of the Jacobi scheme

Since we want to provide the multigrid solver with a good smoother, we want to use a Jacobi-like iterative scheme to solve the $(N_i + N_g) \times (N_i + N_g)$ linear system. Here we show that, in general, a naive implementation of Jacobi iteration does not guarantee the convergence. For simplicity, we analyze the discretization in one dimension, described in Section 1.2.2.

The matrix of the linear system (1.16)-(1.18) is not diagonally dominant and then the convergence of Jacobi schemes cannot be guaranteed, as we can see in the following counterexample: let us consider the homogeneous elliptic problem (1.13)-(1.15) with

$$f = g_a = g_b = 0, \quad [a, b] = [-0.813, 0.813], \quad N = 100.$$

Starting from the initial guess $u = 1$, let us show the maximum absolute value of the numerical solution every ten iterations (Table 2.1).

Table 2.1: Divergence of the numerical solution for the Jacobi iterative scheme applied to the non-eliminated boundary condition system (1.16)-(1.18)

Iteration no.	$\max_{i=l,\dots,r} \lvert u_i \rvert$
10	$2.86 \cdot 10^{+3}$
20	$4.10 \cdot 10^{+8}$
30	$5.87 \cdot 10^{+12}$
40	$8.40 \cdot 10^{+16}$

We could eliminate the boundary equations (1.17) and (1.18) solving them with respect to u_l and u_r respectively, and plugging the result into (1.16). Such elimination leads to a diagonally dominant system, but is very hard to perform in higher dimensions, since the boundary equations are strongly coupled each other.

An alternative strategy could be the following. We perform one iteration over the inside grid points, and after that we iterate until convergence over the ghost points, and repeat both steps until convergence. Even in this case, the convergence is not guaranteed (observe that in one dimension such scheme reduces to Jacobi scheme when boundary conditions are not eliminated). Therefore, we have to find a different approach if we still want to use an iterative Jacobi-like scheme.

2.2 Relaxation scheme: fictitious time

The main idea is to keep a Jacobi-type iteration for inner equations, and to *relax* in some way the boundary conditions. To do this, the model problem 1 (1.1)-(1.3) is transformed into the following *associate time-dependent problem*:

$$\frac{\partial \tilde{u}}{\partial t} = \Delta \tilde{u} + f \qquad \text{in } \Omega \tag{2.1}$$

$$\frac{\partial \tilde{u}}{\partial t} = \mu_D(g_D - \tilde{u}) \qquad \text{on } \Gamma_D \tag{2.2}$$

$$\frac{\partial \tilde{u}}{\partial t} = \mu_N \left(g_N - \frac{\partial \tilde{u}}{\partial n} \right) \qquad \text{on } \Gamma_N \tag{2.3}$$

$$\tilde{u} = \tilde{u}_0 \quad \text{in } \Omega, \text{ when } t = 0 \tag{2.4}$$

where μ_D and μ_N are two positive constants. If the solution of the problem (2.1)-(2.4) is stationary (i.e. $\partial\tilde{u}/\partial t \to 0$, as $t \to +\infty$), then $u(x) = \lim_{t\to+\infty} \tilde{u}(t,x)$ is a solution of the model problem 1 (1.1)-(1.3). A relaxation scheme can be therefore obtained discretizing the problem (2.1)-(2.4), where now the time t represents an iterative parameter.

Below we give a description of the method, starting from the easier 1D case.

2.2.1 One-dimensional case

Consider the model problem 2 (1.13)-(1.15). The associate *time-dependent* problem is:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f \quad \text{in } \Omega \tag{2.5}$$

$$\frac{\partial u(t,a)}{\partial t} = \mu_D \left(g_a - u(t,a) \right) \tag{2.6}$$

$$\frac{\partial u(t,b)}{\partial t} = \mu_N \left(g_b - \frac{\partial u(t,b)}{\partial x} \right) \tag{2.7}$$

$$u(0,x) = u_0(x) \quad \text{in } \Omega \tag{2.8}$$

Let l and r be such that $x_l \le a < x_{l+1}$, $x_{r-1} < b < x_r$ (see Figure 1.7). In order to discretize the time-dependent problem, we use central difference in space and forward Euler in time for (2.5) obtaining:

$$u_i^{(n+1)} = u_i^{(n)} + \frac{\Delta t}{h^2} \left(u_{i-1}^{(n)} - 2u_i^{(n)} + u_{i+1}^{(n)} \right) + \Delta t f_i, \quad i = l+1, \ldots, r-1$$

Taking the maximum time step consented by CFL condition (see [37, 90] for more details about CFL condition), i.e. $\Delta t = h^2/2$, we obtain:

$$u_i^{(n+1)} = \frac{1}{2}\left(u_{i-1}^{(n)} + u_{i+1}^{(n)} + h^2 f_i\right), \quad i = l+1, \ldots, r-1. \tag{2.9}$$

Note that if we discretize directly (1.13) using central difference for the Laplacian operator, and use Jacobi iterative scheme for such discretization, we obtain exactly (2.9).

For a **first order** method, we use first order accurate discretization of the boundary conditions. This is obtained by discretizing (2.6) and (2.7) directly in x_l and x_r respectively. We use forward Euler in time and upwind scheme in space, obtaining:

$$u_l^{(n+1)} = u_l^{(n)} + \mu_D \Delta t \left(g_a - u_l^{(n)}\right), \tag{2.10}$$

$$u_r^{(n+1)} = u_r^{(n)} - \frac{\mu_N \Delta t}{h}\left(u_r^{(n)} - u_{r-1}^{(n)}\right) + \mu_N \Delta t g_b. \tag{2.11}$$

Equation (2.8) is simply the initial condition for the iterative method and we can choose it arbitrarily. We will set $u_i^{(0)} = 0$, for $i = l, \ldots, r$. The iterative scheme (2.9), (2.10), (2.11) is first order accurate.

To obtain a **second order** accuracy, we discretize $u(t, a)$ in (2.6) and the spatial derivative $\partial u(t, b)/\partial x$ in (2.7) to second order (note that we can again discretize to first order in time, because we are just interested at the accuracy in space as $t \to +\infty$). Then we can use linear interpolation of u in nodes x_l, x_{l+1} to compute $u(t, a)$ in (2.6) (that is second order accurate, as we can see in [46]) and quadratic interpolation of u in nodes x_r, x_{r-1}, x_{r-2} to compute the derivative $\partial u(t, b)/\partial x$ in (2.7). However, since we are interested in the second order accuracy not only for the unknown function, but also for its gradient, we use quadratic interpolation also for the Dirichlet condition, i.e. quadratic interpolation of u in nodes x_l, x_{l+1}, x_{l+2} to compute $u(t, a)$, obtaining:

$$\begin{aligned} u_l^{(n+1)} &= u_l^{(n)} \\ &- \mu_D \Delta t \left((1+\vartheta_l)\frac{\vartheta_l}{2} u_l^{(n)} + (1+\vartheta_l)(1-\vartheta_l) u_{l+1}^{(n)} - (1-\vartheta_l)\frac{\vartheta_l}{2} u_{l+2}^{(n)} - g_a\right) \end{aligned} \tag{2.12}$$

$$\begin{aligned} u_r^{(n+1)} &= u_r^{(n)} \\ &- \frac{\mu_N \Delta t}{h}\left(u_{r-1}^{(n)} - u_{r-2}^{(n)} + \left(u_{r-2}^{(n)} - 2u_{r-1}^{(n)} + u_r^{(n)}\right)\left(\frac{1}{2} + \vartheta_r\right)\right) + \mu_N \Delta t g_b, \end{aligned} \tag{2.13}$$

where $\vartheta_l = (x_{l+1} - a)/h$ and $\vartheta_r = (b - x_{r-1})/h$. Note that the discretization (2.9) is already second order accurate in space, hence (2.9), (2.12) and (2.13) provide a second order accurate iterative scheme.

Constants μ_D and μ_N of (2.10) and (2.11) (first order accuracy) and of (2.12) and (2.13) (second order accuracy) are obtained by CFL condition, i.e. in such a way that the coefficient of $u_{l,r}^{(n)}$ in the right-hand side of (2.10) and (2.11) (first order accuracy) and of (2.12) and (2.13) (second order accuracy) is positive. Such conditions read:

$$\mu_D \Delta t < 1, \quad \frac{\mu_N \Delta t}{h} < 1 \text{ for first order accuracy;}$$

$$\mu_D \Delta t < 1, \quad \frac{\mu_N \Delta t}{h} < \frac{2}{3} \text{ for second order accuracy.} \tag{2.14}$$

In numerical tests of Section 2.7, we choose $\mu_D \Delta t = 0.9$ and $\mu_N \Delta t = 0.9 \cdot 2h/(3)$. Since $\Delta t = h^2/2$, then $\mu_D = 1.8/h^2$ and $\mu_N = 3.6/(3h)$.

Anyway, if ϑ_l is very small, the discretization error is slightly greater than what we expect, but the convergence is guaranteed.

2.2.2 Two-dimensional case

Consider the model problem 1 (1.1)-(1.3) and the associate time-dependent problem (2.1)(2.4). For any grid point (jh, ih) of Ω_h, we write an equation obtained from the discretization of (2.1) in such point, using again forward Euler in time and central difference in space and taking the maximum time step consented by the CFL condition (i.e. $\Delta t = h^2/4$):

$$u_{i,j}^{(n+1)} = 1/4 \left(h^2 f_{i,j} + u_{i-1,j}^{(n)} + u_{i+1,j}^{(n)} + u_{i,j-1}^{(n)} + u_{i,j+1}^{(n)} \right). \tag{2.15}$$

As in the one-dimensional case, Eq. (2.15) is equivalent to directly discretizing (1.1) using central difference in space and Jacobi iteration.

For any ghost point $G \in \Gamma_h$ we compute the projection point B on the boundary by (1.7) and discretize (2.2) or (2.3) if respectively $G \in \Gamma_D$ or $G \in \Gamma_N$. We use forward Euler in time and the discretizations (1.9), (1.10) (first order accuracy) and (1.11) (second order accuracy) in space. We obtain the iterative scheme:

$$u_G^{(n+1)} = u_G^{(n)} + \mu_D \Delta t \left(g_D(B) - u_h^{(n)}(B) \right) \tag{2.16}$$

if $G \in \Gamma_D$, or

$$u_G^{(n+1)} = u_G^{(n)} + \mu_N \Delta t \left(g_D(B) - \frac{\partial u_h}{\partial n}^{(n)}(B) \right) \tag{2.17}$$

if $G \in \Gamma_N$.

The reconstructions $u_h^{(n)}(B)$ and $\dfrac{\partial u_h}{\partial n}^{(n)}(B)$ are the ones described in Sec. 1.2.1, more precisely they are (1.9) and (1.10) for **first order** accuracy (of the solution) and (1.11) for **second order** accuracy (of the solution and in the gradient).

Constants μ_D and μ_N of (2.16) and (2.17) are obtained by CFL condition, i.e. in such a way that the coefficient of $u_P^{(n)}$ in the right-hand side of (2.16) and (2.17) is positive. Such conditions read:

$$\mu_D \Delta t < 1, \quad \frac{\mu_N \Delta t}{h} < \frac{1}{\sqrt{2}} \text{ for first order accuracy;}$$

$$\mu_D \Delta t < 1, \quad \frac{\mu_N \Delta t}{h} < \frac{2}{3\sqrt{2}} \text{ for second order accuracy.} \tag{2.18}$$

In numerical tests of Section 2.7, we choose $\mu_D \Delta t = 0.9$ and $\mu_N \Delta t = 0.9 \cdot 2h/(3\sqrt{2})$. Since $\Delta t = h^2/4$, then $\mu_D = 3.6/h^2$ and $\mu_N = 7.2/(3\sqrt{2}h)$.

2.2.3 Three-dimensional case

One of the strength of the method is the easy generalization in higher dimension. In fact, the description of the method is exactly the same in all dimensions. Then we are able to provide a description holding in the general case of dimension d. Anyway, for the sake of the argument, we just describe the method in fixed dimension. In 3D, for instance, the description depicted in Section 2.2.2 for 2D case still holds, but the equations of the linear system have to be changed. For inner points, the equations (2.15) becomes:

$$u_{i,j,k}^{(n+1)} = \frac{1}{6}\left(h^2 f_{i,j,k} + u_{i-1,j,k}^{(n)} + u_{i+1,j,k}^{(n)} + u_{i,j-1,k}^{(n)} + u_{i,j+1,k}^{(n)} + u_{i,j,k-1}^{(n)} + u_{i,j,k+1}^{(n)}\right)$$

The equations (2.16) and (2.17) are:

$$u_P^{(n+1)} = u_P^{(n)} + \mu_D \Delta t \left(g_D(H_P) - u_P^{(n)}\right)$$

$$\begin{aligned} u_P^{(n+1)} &= u_P^{(n)} + \mu_N \Delta t g_N(H_P) \\ &\quad - \mu_N \frac{\Delta t}{h}\left(\left(u_P^{(n)} - u_{Q_x}^{(n)}\right)|n_x| + \left(u_P^{(n)} - u_{Q_y}^{(n)}\right)|n_y| + \left(u_P^{(n)} - u_{Q_z}^{(n)}\right)|n_z|\right) \end{aligned}$$

for first order accuracy (in the solution), and:

$$u_P^{(n+1)} = u_P^{(n)} + \mu_D \Delta t \left(g_D(H_P) - \tilde{u}_{27}^{(n)}(H_P)\right)$$

$$u_P^{(n+1)} = u_P^{(n)} + \mu_N \Delta t \left(g_N(H_P) - (\nabla \tilde{u}_{27} \cdot \hat{\mathbf{n}})|_{H_P} \right)$$

for second order accuracy (in the solution and its gradient). where $\tilde{u}_{27}$ is the triquadratic interpolation in the 27−point stencil $P_1 - P_{27}$ in upwind direction.

As in Sec. 2.2.2, constants μ_D and μ_N are obtained by CFL condition. In the general d-dimensional case they read:

$$\mu_D \Delta t < 1, \quad \frac{\mu_N \Delta t}{h} < \frac{1}{\sqrt{d}} \text{ for first order accuracy;}$$

$$\mu_D \Delta t < 1, \quad \frac{\mu_N \Delta t}{h} < \frac{2}{3\sqrt{d}} \text{ for second order accuracy.} \tag{2.19}$$

Finally, by a numerical point of view, it is time-consuming to reconstruct $\nabla \tilde{u}_{27} \cdot \hat{\mathbf{n}}$ in H_P using a 27−point stencil. To avoid it, we can use the approach described in 2D case in Section 1.2.1.3 and easily adapt it to 3D case. We will use this approach not only when we would like to reduce the stencil (because some point of the stencil is an *outside point*), but everywhere. The computation then uses just a 12−point stencil for any derivative and the scheme is more efficient. We use this approach in the 3D numerical tests described in Section 2.7.

2.3 Convergence proof

Note that the iterative scheme can be seen as a preconditioned iterative method with a diagonal preconditioner P, whose diagonal is:

$$diag(P) = \left[\ldots, 4/h^2, \ldots, (\mu_D \Delta t)^{-1}, \ldots, (\mu_N \Delta t)^{-1}, \ldots \right],$$

where the entries refer respectively to the inner equations, the Dirichlet and the Neumann boundary conditions. Therefore, the iterative scheme for the linear system $Au = f$ reads:

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + P^{-1} \left(f - A\mathbf{u}^{(n)} \right)$$

and the iteration matrix is $B = I - P^{-1}A$, where I is the identity matrix. In this section we shall prove the convergence in all space dimensions only for the first order method. For the second order method, numerical experiments confirm the convergence of the method, i.e. the spectral radius of the iteration matrix B is less than one (as we can see, for instance, in numerical tests 1.1 and 1.5).

2.3.1 One-dimensional case

Consider the notation of Sections 1.2.2 and 2.2.1. Let $\tilde{N} = r - l + 1$. Then we can write the iterative scheme (2.9), (2.10), (2.11) in matrix fashion:

$$\mathbf{u}^{(n+1)} = B\mathbf{u}^{(n)} + F \tag{2.20}$$

where $\mathbf{u}^{(n)}$, $F \in \mathbb{R}^{\tilde{N}}$, $B \in \mathbb{R}^{\tilde{N}\times\tilde{N}}$ are defined as follows:

$$\mathbf{u}^{(n)} = \begin{pmatrix} u_l^{(n)} \\ u_{l+1}^{(n)} \\ \vdots \\ u_r^{(n)} \end{pmatrix}, \quad F = \begin{pmatrix} \Delta t g_a \\ h^2 f_{l+1} \\ \vdots \\ h^2 f_{r-1} \\ \Delta t g_b \end{pmatrix}$$

$$B = \begin{pmatrix} 1-\mu_D\Delta t & 0 & & & \\ 1/2 & 0 & 1/2 & & \\ & \ddots & \ddots & \ddots & \\ & & 1/2 & 0 & 1/2 \\ & & & c & (1-c) \end{pmatrix},$$

where $c = \mu_N \Delta t/h$ is the Courant number. A necessary and sufficient condition for the convergence of (2.20) is $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of the matrix B (i.e. the maximum eigenvalue of B in absolute value). In order to prove this condition we shall make use of Gershgorin-Hadamard theorems (that can be founded in any good basic text of Numerical Analysis, such as [80] or [91]).

The first theorem states that if $B \in \mathbb{C}^{m\times m}$, with $m \in \mathbb{N}$, then every eigenvalue λ of B satisfies:

$$\lambda \in S_C = \cup_{i=1}^m C_i, \quad C_i = \{z \in \mathbb{C} : |z - b_{ii}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{m} |b_{ij}|\}$$

and the sets C_i are said *Gershgorin circles*.

The second theorem states that, if B is an irreducible matrix, every eigenvalue of B belonging to the boundary of S_C, belongs to the boundary of all Gershgorin circles.

Applying the first Gershgorin-Hadamard theorem to the matrix B, we have $\tilde{N}-2$ coincident circles $\mathcal{C}\left((0,0),1\right)$ centered at the origin and with radius 1, one circle reduced to a single point $(1-\Delta t, 0)$, and one circle centered at $(1-c, 0)$ with radius c. Since $0 < c < 1$ (CFL condition), the union of all

Gershgorin circles is $\mathcal{C}((0,0),1)$. Then $\rho(B) \leq 1$. If $\rho(B) = 1$, there exists an eigenvalue λ such that $\lambda \in \partial\mathcal{C}((0,0),1)$. By second Gershgorin-Hadamard theorem, this eigenvalue must belong to all Gershgorin circles, and this is not possible because the presence of the circle $\mathcal{C}((1-\Delta t,0),0)$. Therefore, $\rho(B) < 1$ and the iterative scheme (2.20) converges.

2.3.2 Two-dimensional case

Consider the notation of Sections 1.2 and 2.2.2. Let N_i be the number of the inside points to Ω and N_g the number of ghost points. The first order accurate relaxation scheme is the following (by (2.15), (2.16), (2.17), (1.9), (1.10)):

$$u_{i,j}^{(n+1)} = \frac{1}{4}\left(h^2 f_{i,j} + u_{i-1,j}^{(n)} + u_{i+1,j}^{(n)} + u_{i,j-1}^{(n)} + u_{i,j+1}^{(n)}\right) \text{ if } (ih, jh) \in \Omega_h, \tag{2.21}$$

$$u_G^{(n+1)} = u_G^{(n)} + \mu_D \Delta t \left(g_D(B) - u_G^{(n)}\right) \text{ if } G \in \Gamma_D, \tag{2.22}$$

$$\begin{aligned} u_P^{(n+1)} &= u_P^{(n)} + \mu_N \Delta t g_N(H_P) \\ &\quad - \mu_N \frac{\Delta t}{h}\left(\left(u_P^{(n)} - u_{Q_x}^{(n)}\right)|n_x| + \left(u_P^{(n)} - u_{Q_y}^{(n)}\right)|n_y|\right) \text{ if } G \in \Gamma_N. \end{aligned} \tag{2.23}$$

By numbering all this points (inside and ghost) in some order in a single vector, we can write the iteration scheme (2.21)-(2.23) in matrix fashion:

$$\mathbf{u}^{(n+1)} = B\mathbf{u}^{(n)} + F$$

where $\mathbf{u}^{(n)}, F \in \mathbb{R}^{N_i+N_g}$, $B = (b_{i,j}) \in \mathbb{R}^{(N_i+N_g)\times(N_i+N_g)}$. As in the previous section, we shall make use of Gershgorin-Hadamard theorems. Let us check that all the Gershgorin circles are subset (in the complex plane) of the unit circle $B((0,0),1)$ centered at the origin. For any row k, we define

$$c_k = b_{k,k}, \quad r_k = \sum_{\substack{l=1 \\ l\neq k}}^{N_i+N_g} |b_{k,l}|$$

respectively the center and the radius of the associated Gershgorin circle $B((c_k,0),r_k)$. Then:

$$B((c_k,0),r_k) \subseteq B((0,0),1) \Longleftrightarrow r_k \leq 1 - |c_k| \tag{2.24}$$

Our goal is therefore to check that the right-hand side of (2.24) holds for any $k = 1,\ldots,N_i+N_g$. If k is related to one of the N_i equations of the kind

(2.15), then $c_k = 0$, $r_k = 1$ and the right hand-side of (2.24) is satisfied. If k is related to an equation of the kind (2.22), then $c_k = 1 - \mu_D \Delta t$, $r_k = 0$ and the right hand-side of (2.24) is (strictly) satisfied, since we assume $\mu_D \Delta t < 1$. If k is related to an equation of the kind (2.23), then

$$c_k = 1 - \frac{\mu_N \Delta t}{h} \left(|n_x| + |n_y| \right) \tag{2.25}$$

$$r_k = \frac{\mu_N \Delta t}{h} \left(|n_x| + |n_y| \right) \tag{2.26}$$

and the right hand-side of (2.24) is (strictly) satisfied if and only if $r_k \leq 1$. This is true because

$$\mu_N \Delta t / h < 1/\sqrt{2} \tag{2.27}$$

that is the CFL condition (2.18), and

$$|n_x| + |n_y| \leq \sqrt{2} \tag{2.28}$$

(because $n_x^2 + n_y^2 = 1$). This leads us to claim $\rho(B) \leq 1$. Thanks to the strictly inequality $r_{\tilde{k}} < 1 - |c_{\tilde{k}}|$ for at least one $\tilde{k}$, we can ensure by (2.24) that $B\left((c_{\tilde{k}}, 0), r_{\tilde{k}}\right) \subset B\left((0,0), 1\right)$ and then $\rho(B) < 1$ by the second Gershgorin-Hadamard theorem. □

2.3.3 Three-dimensional case

Another strength of the method is the easy generalization of the 2D-convergence proof (for first order accuracy) in higher dimension. For instance, let us consider three-dimensional case. The proof will be the same as that provided in 2D case in the previous section, but the center and the radius of the Gershgorin circle (2.25), (2.26) become respectively:

$$c_k = 1 - \mu_N \Delta t / h \left(|n_x| + |n_y| + |n_z| \right)$$

$$r_k = \mu_N \Delta t / h \left(|n_x| + |n_y| + |n_z| \right)$$

while the convergence conditions (2.27), (2.28) will be replaced by:

$$\mu_N \Delta t / h < 1/\sqrt{3}$$

$$|n_x| + |n_y| + |n_z| \leq \sqrt{3} \ \left(\text{ because } n_x^2 + n_y^2 + n_z^2 = 1 \right)$$

The generalization in higher dimension is straightforward.

2.4 Relaxation scheme for some extensions

In this section we describe how to modify the associate time-dependent problem (2.1)-(2.4) for the cases introduced in Section 1.3, namely Robin boundary conditions, variable coefficient diffusion, anisotropic operators, sharp-edged domains.

2.4.1 Robin boundary conditions

Let us consider the Model Problem 3 (1.19)-(1.20). The associate time-dependent problem is:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \Delta u + f && \text{in } \Omega \\ \frac{\partial u}{\partial t} &= \mu \left(g - \left(\alpha u + \beta \frac{\partial u}{\partial n} \right) \right) && \text{on } \Gamma \\ u &= u_0 && \text{in } \Omega, \text{ when } t = 0 \end{aligned}$$

For any ghost point $G \in \Gamma_h$, let $\tilde{u}$ and $\tilde{\Phi}$ be the biquadratic interpolants of u and Φ in the Upwind nine-point stencil (see Fig. 1.4).

Therefore, the second order accurate iterative scheme for the problem (1.19), (1.20) will be:

$$u_{i,j}^{(n+1)} = 1/4 \left(h^2 f_{i,j} + u_{i-1,j}^{(n)} + u_{i+1,j}^{(n)} + u_{i,j-1}^{(n)} + u_{i,j+1}^{(n)} \right).$$

for any grid point $(jh, ih) \in \Omega_h$, and

$$u_P^{(n+1)} = u_P^{(n)} - \mu \Delta t \left(\alpha \tilde{u}^{(n)}(H_P) + \beta \mu \Delta t \nabla \tilde{u}^{(n)} \cdot \hat{\mathbf{n}} \big|_{H_P} \right) + \mu \Delta t \; g \qquad (2.29)$$

for any ghost point G, where B is the closest boundary point to G, $\hat{\mathbf{n}} = \nabla \tilde{\phi} / \left| \nabla \tilde{\phi} \right|$, $\alpha = \alpha(B)$, $\beta = \beta(B)$, $g = g(B)$, and μ is chosen in order to satisfy the CFL condition:

$$\frac{\mu \Delta t}{h} < \frac{1}{\alpha h + \beta \sqrt{2}}. \qquad (2.30)$$

2.4.1.1 Convergence proof

First order accurate scheme is obtained substituting (2.29) with (see Fig. 1.3)

$$\begin{aligned} u_P^{(n+1)} = u_P^{(n)} &+ \mu \Delta t \; g \\ &- \mu \Delta t \left(\alpha u_P^{(n)} + \frac{\beta}{h} \left(u_P^{(n)} - u_{Q_x}^{(n)} \right) |n_x| + \left(u_P^{(n)} - u_{Q_y}^{(n)} \right) |n_y| \right) \end{aligned}$$

The convergence proof is similar to the one presented in the case of mixed boundary condition in Section 2.3.2, provided to replace (2.25) and (2.26) by respectively:

$$c_k = 1 - \mu\Delta t \left(\alpha + \frac{\beta}{h} \left(|n_x| + |n_y| \right) \right),$$

$$r_k = \frac{\mu\Delta t}{h} \beta \left(|n_x| + |n_y| \right).$$

The CFL condition (2.30) ensures that $c_k \geq 0$ and the right hand-side of (2.24) reads:

$$\frac{\mu\Delta t}{h} \beta \left(|n_x| + |n_y| \right) \leq \mu\Delta t \left(\alpha + \frac{\beta}{h} \left(|n_x| + |n_y| \right) \right)$$

which is strictly satisfied for at least one ghost node since α is not identically equal to zero (i.e. the problem has not pure Neumann conditions). This leads us to claim $\rho(B) < 1$ by the same argument of Sec. 2.3.2. $\square$

2.4.2 Variable diffusion coefficient

Let us consider the Model Problem 4 of Sec. 1.3.2. The associate time-dependent problem is:

$$\begin{aligned}
\frac{\partial \tilde{u}}{\partial t} &= \mu \left(\nabla \cdot (\gamma \nabla \tilde{u}) + f \right) && \text{in } \Omega \\
\frac{\partial \tilde{u}}{\partial t} &= \mu_D (g_D - \tilde{u}) && \text{on } \Gamma_D \\
\frac{\partial \tilde{u}}{\partial t} &= \mu_N \left(g_N - \frac{\partial \tilde{u}}{\partial n} \right) && \text{on } \Gamma_N \\
\tilde{u} &= \tilde{u}_0 && \text{in } \Omega, \text{ when } t = 0
\end{aligned}$$

where μ is a positive function depending only on the space, in such a way in grid points (ih, jh) we have:

$$\mu_{i,j}\Delta t = \frac{h^2}{\gamma_{i-1/2,j} + \gamma_{i+1/2,j} + \gamma_{i,j-1/2} + \gamma_{i,j+1/2}},$$

so that we can regain the Jacobi scheme for inner equations. Constants μ_D and μ_N are defined as in Sec. 2.2. In practice, the relaxation scheme is the same as the one described in Sec. 2.2.2, provided to replace (2.15) by:

$$u_{i,j}^{(n+1)} = \frac{\left(h^2 f_{i,j} + \gamma_{i-1/2,j} u_{i-1,j}^{(n)} + \gamma_{i+1/2,j} u_{i+1,j}^{(n)} + \gamma_{i,j-1/2} u_{i,j-1}^{(n)} + \gamma_{i,j+1/2} u_{i,j+1}^{(n)} \right)}{\gamma_{i-1/2,j} + \gamma_{i+1/2,j} + \gamma_{i,j+1/2} + \gamma_{i,j-1/2}}$$

where $\gamma_{i\pm1/2,j} = (\gamma_{i,j} + \gamma_{i\pm1,j})/2$, $\gamma_{i,j\pm1/2} = (\gamma_{i,j} + \gamma_{i,j\pm1})/2$.

2.4.3 Anisotropic case

Let us consider the Model Problem 5 of Sec. 1.3.3. The associate time-dependent problem is:

$$\begin{aligned}\frac{\partial u}{\partial t} &= \mu\left(\nabla\cdot(A\nabla u)+f\right) \text{ in } \Omega\\ \frac{\partial u}{\partial t} &= \mu_D\left(g_D-u\right) \text{ on } \Gamma_D\\ \frac{\partial u}{\partial t} &= \mu_N\left(g_N-\nabla u\cdot\mathbf{N}\right) \text{ on } \Gamma_N\end{aligned}$$

where μ is a positive function having the same meaning as in the previous Sec. 2.4.2, i.e. in such a way the iteration scheme is Jacobi-like for inner equations. The relaxation scheme is straightforward to obtain. We must observe that the convergence of the relaxation scheme is gained only if the anisotropy is sufficiently small, i.e. if the ratio between the maximum and minimum eigenvalues of the matrix A is less than a suitable threshold.

2.4.4 Sharp-edged domain

Since the the only difference is the spatial discretization, the relaxation scheme is straightforwardly obtained.

2.5 Multigrid components: one dimensional case

In this section (1D) and in Section 2.6 (2D) we provide a multigrid approach to speed up the convergence of the relaxation scheme (2.1)-(2.4) In all the description, we consider the second order accuracy discretization. For one-dimensional case, the multigrid approach in arbitrary domain is a natural extension of the basic multigrid strategy that can be found in any good basic text about multigrid, such as [105, 27, 55]. Although we can eliminate the boundary conditions from the linear system obtained by discretizing the problem, we always want to treat the case of non-eliminated boundary conditions in order to straightforwardly extend the method to more than one dimension, where the elimination of the boundary conditions from the system is hard to perform.

Let us refer to the notation of Secs. 1.2.2 and 2.2.1. We call Γ_h the set of ghost points, i.e. $\Gamma_h = \{x_l, x_r\}$. Let I_h be a general subset of D_h. We introduce the linear space of grid functions over I_h and we denote it

$S(I_h) = \{\mathbf{w}_h \colon I_h \to \mathbb{R}\}$. For any $\mathbf{w}_h \in S(I_h)$, we pose $w_i^h = \mathbf{w}_h(x_i)$. Let $\mathbf{f}_h \in S(\Omega_h)$ such that $f_i^h = f(x_i)$. The iterative scheme (2.9), (2.12), (2.13) converges to the exact solution of the discretized system

$$-\Delta_h \mathbf{u}_h = \mathbf{f}_h \tag{2.31}$$
$$g_D^h(\mathbf{u}_h) = g_a \tag{2.32}$$
$$g_N^h(\mathbf{u}_h) = g_b, \tag{2.33}$$

where $\Delta_h \colon S(\Omega_h \cup \Gamma_h) \to S(\Omega_h)$ is defined by:

$$\Delta_h \mathbf{u}_h(x_i) = \frac{u_{i-1}^h - 2u_i^h + u_{i+1}^h}{h^2}, \quad x_i \in \Omega_h,$$

while $g_D^h, g_N^h \colon S(\Omega_h \cup \Gamma_h) \to \mathbb{R}$ are the discrete versions of the boundary conditions:

$$g_D^h(\mathbf{u}_h) = (1+\vartheta_l)\frac{\vartheta_l}{2}u_l^h + (1+\vartheta_l)(1-\vartheta_l)u_{l+1}^h - (1-\vartheta_l)\frac{\vartheta_l}{2}u_{l+2}^h,$$

$$g_N^h(\mathbf{u}_h) = \frac{u_{r-1}^h - u_{r-2}^h}{h} + \frac{u_{r-2}^h - 2u_{r-1}^h + u_r^h}{h}\left(\frac{1}{2} + \vartheta_r\right).$$

System (2.31)-(2.33) can be interpreted in general as a discrete system of a Poisson equation with non-eliminated boundary conditions.

Let us consider an arbitrary grid function $\tilde{\mathbf{u}}^h \in S(\Omega_h \cup \Gamma_h)$ and let

$$\begin{aligned} \mathbf{r}_h &= \mathbf{f}_h + \Delta_h \tilde{\mathbf{u}}^h \\ \tilde{g}_a &= g_a - g_D^h(\tilde{\mathbf{u}}^h) \\ \tilde{g}_b &= g_b - g_N^h(\tilde{\mathbf{u}}^h) \end{aligned}$$

be the defects of (2.31), (2.32), (2.33) respectively. Because of the linearity of Δ_h, g_D^h, g_N^h, if we solve exactly the so-called *residual problem*

$$\begin{aligned} -\Delta_h \mathbf{e}_h &= \mathbf{r}_h \\ g_D^h(\mathbf{e}_h) &= \tilde{g}_a \\ g_N^h(\mathbf{e}_h) &= \tilde{g}_b \end{aligned}$$

in the unknown $\mathbf{e}_h \in S(\Omega_h \cup \Gamma_h)$, then $\mathbf{u}_h = \tilde{\mathbf{u}}^h + \mathbf{e}_h$ is the exact solution of the system (2.31), (2.32), (2.33). In the basic idea of multigrid one needs to solve the residual problem in a grid coarser than the original one.
We can summarize the iterative scheme (2.9), (2.12), (2.13) as follows:

$$\mathbf{u}_h^{(m+1)} = \Re_h\left(\mathbf{u}_h^{(m)}, \mathbf{f}_h, g_a, g_b\right) \tag{2.34}$$

$$\Re_h \colon S(\Omega_h \cup \Gamma_h) \times S(\Omega_h) \times \mathbb{R}^2 \longrightarrow S(\Omega_h \cup \Gamma_h). \tag{2.35}$$

Note that the iterative scheme (2.9), (2.12), (2.13) is of a Jacobi kind. In order to provide a multigrid strategy, we just require that the iteration operator (2.35) has the *smoothing property*, i.e. after few iteration steps (2.34), the error becomes smooth (not necessarily small). Roughly speaking, the high-frequency components of the error reduce quickly. We call *smoothers* any operator (2.35) with this property. Many iterators have this property, such as Gauss-Seidel or weighted Jacobi (with weight $\omega = 2/3$ in 1D or $\omega = 4/5$ in 2D), but not Jacobi (see [105, pag. 30–32] for more details). From now on, by (2.34) we shall intend the Gauss-Seidel version of (2.9), (2.12), (2.13), i.e.:

$$\begin{aligned} u_l^{(m+1)} &= u_l^{(m)} + \mu_D\, \Delta t\, g_a \\ &\quad - \mu_D\, \Delta t \left((1+\vartheta_l)\frac{\vartheta_l}{2} u_l^{(m)} + (1+\vartheta_l)(1-\vartheta_l) u_{l+1}^{(m)} - (1-\vartheta_l)\frac{\vartheta_l}{2} u_{l+2}^{(m)} \right) \end{aligned}$$

$$u_i^{(m+1)} = \frac{1}{2}\left(u_{i-1}^{(m+1)} + u_{i+1}^{(m)} + h^2 f_i \right), \quad i = l+1, \ldots, r-1$$

$$\begin{aligned} u_r^{(m+1)} &= u_r^{(m)} + \mu_N\, \Delta t\, g_b \\ &- \frac{\mu_N\, \Delta t}{h} \left(u_{r-1}^{(m+1)} - u_{r-2}^{(m+1)} + \left(u_{r-2}^{(m+1)} - 2u_{r-1}^{(m+1)} + u_r^{(m+1)} \right) \left(\frac{1}{2} + \vartheta_r \right) \right). \end{aligned}$$

In order to explain the multigrid approach, we just describe the two-grid correction scheme (TGCS), because all the other schemes, such as V-cycle, W-cycle, F-cycle or Full multigrid cycle, can be easily derived from it (see [105, Sections 2.4, 2.6] for more details). The TGCS consists into the following algorithm:

1. Set initial guess $\mathbf{u}_h = 0$

2. Relax ν_1 times on the finest grid: for k from 1 to ν_1 do

$$\mathbf{u}_h := \Re_h \left(\mathbf{u}_h, \mathbf{f}_h, g_a, g_b \right)$$

3. Compute the defects

$$\begin{aligned} \mathbf{r}_h &= \mathbf{f}_h + \Delta_h \mathbf{u}_h \\ \tilde{g}_a &= g_a - g_D^h(\tilde{\mathbf{u}}^h) \\ \tilde{g}_b &= g_b - g_N^h(\tilde{\mathbf{u}}^h) \end{aligned}$$

4. Transfer the defect $\mathbf{r}_h$ to a coarser grid with spatial step $2h$ by a suitable *restriction operator*
$$\mathbf{r}_{2h} = I_{2h}^{h}(\mathbf{r}_h)$$

5. Solve exactly the residual problem on the coarser grid
$$\begin{aligned} -\Delta_{2h}\mathbf{e}_{2h} &= \mathbf{r}_{2h} \\ g_D^{2h}(\mathbf{e}_{2h}) &= \tilde{g}_a \\ g_N^{2h}(\mathbf{e}_{2h}) &= \tilde{g}_b \end{aligned}$$
in the unknown $\mathbf{e}_{2h} \in S(\Omega_{2h} \cup \Gamma_{2h})$

6. Transfer the error to the finest grid by a suitable *interpolation operator*
$$\mathbf{e}_h = I_h^{2h}(\mathbf{e}_{2h})$$

7. Correct the fine-grid approximation
$$\mathbf{u}_h := \mathbf{u}_h + \mathbf{e}_h$$

8. Relax ν_2 times on the finest grid: for k from 1 to ν_2 do
$$\mathbf{u}_h := \Re_h(\mathbf{u}_h, \mathbf{f}_h, g_a, g_b)$$

We have just to explain the steps concerning grid migration (steps 4 and 6).

2.5.1 Transfer grid operators

In this section, we describe the transfer grid operators for vertex-centered grid. We observe that our approach is based on the discretization of the equations on the various grids (both for inner and ghost points). This approach is very different from algebraic multigrid. As a consequence, the interpolation and the restriction operators are not the transpose of each other.

2.5.1.1 Restriction operator

Since such operator will act on the defect $\mathbf{r}_h \in S(\Omega_h)$ (step 4), we must determine $I_{2h}^{h}\mathbf{r}_h(x)$ for any $x \in \Omega_{2h}$ using only values inside Ω_h. This is justified by the fact that the defect of the inside grid points (referred to the Poisson equation) may be very different (after few relaxations) from the defects $\tilde{g}_a$, $\tilde{g}_b$ (referred to the boundary conditions and stored computationally in the ghost points), because the operators (for inner equations and for

boundary conditions) scale with different powers of h. Then, let $x \in \Omega_{2h}$ and refer to Fig. 2.1 (upper part). If x is not near an outside grid point, i.e. $\min\{|x-a|, |x-b|\} \geq h$, then we will use the standard full-weighting restriction operator (FW):

$$I_{2h}^{h}\mathbf{r}_h(x) = \frac{1}{4}\left(\mathbf{r}_h(x-h) + 2\,\mathbf{r}_h(x) + \mathbf{r}_h(x+h)\right), \tag{2.36}$$

while if $x - h < a$ or $x + h > b$ we set respectively

$$I_{2h}^{h}\mathbf{r}_h(x) = \frac{1}{2}\left(\mathbf{r}_h(x) + \mathbf{r}_h(x+h)\right) \tag{2.37}$$

or

$$I_{2h}^{h}\mathbf{r}_h(x) = \frac{1}{2}\left(\mathbf{r}_h(x-h) + \mathbf{r}_h(x)\right). \tag{2.38}$$

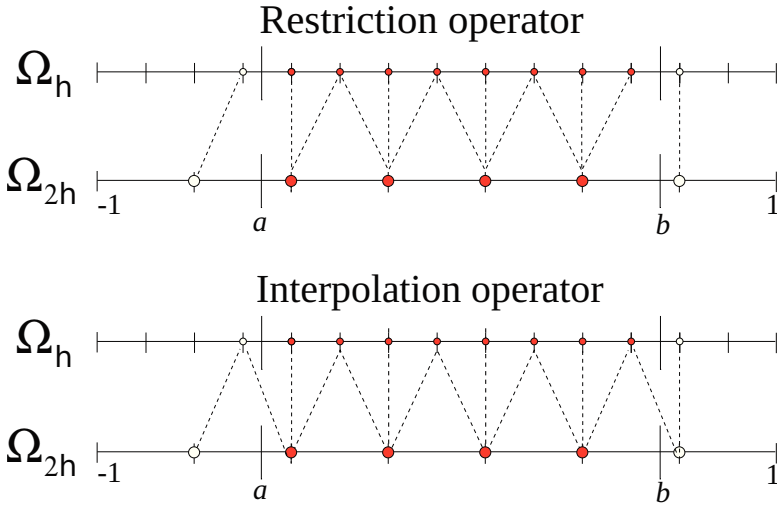


Fig. 2.1: Vertex-centered discretization in 1D. Inner grid nodes (red circles) and ghost points (empty circles) on the fine and coarse mesh. The dashed lines represent the action of the restriction (up) and the interpolation (down) operators.

2.5.1.2 Interpolation operator

Since the interpolation operator acts on the error (step 6), which is continuous across the boundary, we do not need to separate the interpolation for inner equations from the interpolation of ghost points, and then we just use the standard linear interpolation operator (see the lower part of Fig. 2.1):

$$\begin{aligned} I_{h}^{2h}\mathbf{e}_{2h}(x_j) &= \mathbf{e}_{2h}(x_j) && \text{if } j \text{ is even} \\ I_{h}^{2h}\mathbf{e}_{2h}(x_j) &= \tfrac{1}{2}\left(\mathbf{e}_{2h}(x_{j-1}) + \mathbf{e}_{2h}(x_{j+1})\right) && \text{if } j \text{ is odd.} \end{aligned}$$

Remark. 1 (V-cycle) The V-cycle algorithm is easily obtained from the TGCS recursively, namely applying the same algorithm to solve the residual equation in step 5. To terminate the recursion, an exact solver is used to solve the residual problem when the grid becomes too coarse.

Remark. 2 (W-cycle) The W-cycle is similar to the V-cycle, with the only difference that the residual problem is solved recursively two times instead of one (in general schemes, γ times, but $\gamma > 2$ is considered useless for practical purpose).

Remark. 3 (Coarser operator) We observe that the discrete operator Δ_{2h} in step 5 is just the operator obtained discretizing directly the continuous operator in the coarser grid, and not the operator obtained by the Galerkin condition

$$\Delta_{2h} = I_{2h}^{h}\, \Delta_h\, I_{h}^{2h}.$$

The latter approach, typical of algebraic multigrid, makes the algebraic problem more expensive from a computational point of view and does not take advantage of the fact that the discrete problem comes from a continuous problem.

2.6 Multigrid components: High-dimensional case

In this case the defect of the boundary conditions has to be transferred in a suitable way to a coarser grid. The restriction has to be performed separately from the restriction of the interior equations, since these defects may show a sharp gradient crossing the boundary, because the discrete operators scale with different powers of h.

In case of arbitrary domain, ghost points may have a complex structure and the restriction cannot be defined straightforwardly as in the rectangular case, where ghost points are aligned with the grid and the restriction can be performed by a one dimensional operator.

For arbitrary domain we first need to extend the defect in a narrow band outside the domain constant along normal direction, and then we can operate the restriction as in the interior of the domain. For the sake of clarity, we describe the multigrid strategy in the two-dimensional case, but the procedure can be extended straightforwardly in more dimensions.

Let us refer to the notation of Secs. 1.2 and 2.2.2 and add some further notation. Let I_h be a general subset of D_h. We introduce the linear space of grid functions over I_h and we denote it $S(I_h) = \{\mathbf{w}_h \colon I_h \to \mathbb{R}\}$. From now

on, we shall consider the two space dimension, i.e. $d = 2$, but the results are valid also for $d > 2$.

Using the simplified notation, the iterative scheme (2.15)-(2.17) converges to the solution of the problem:

$$\begin{cases} -\Delta_h \mathbf{u}_h = \mathbf{f}_h \\ L_h \mathbf{u}_h = \mathbf{g}_h \end{cases} \tag{2.39}$$

where:

- $\mathbf{u}_h \in S(\Omega_h \cup \Gamma_h)$ is the unknown;
- $\Delta_h \colon S(\Omega_h \cup \Gamma_h) \to S(\Omega_h)$ is the standard discrete version of the Laplacian operator, namely:

$$\Delta_h \mathbf{w}_h(x,y) = \frac{1}{h^2}\left(\mathbf{w}_h(x+h,y) + \mathbf{w}_h(x-h,y) - 4\mathbf{w}_h(x,y) \right. \\ \left. + \mathbf{w}_h(x,y+h) + \mathbf{w}_h(x,y-h)\right)$$

  for any $\mathbf{w}_h \in S(\Omega_h \cup \Gamma_h)$ and $(x,y) \in \Omega_h$;

- $\mathbf{f}_h \in S(\Omega_h)$ is defined by $f_h(P) = f(P)$ for any grid point $P \in \Omega_h$;
- $L_h \colon S(\Omega_h \cup \Gamma h) \to S(\Gamma_h)$ is the discrete version of boundary conditions, namely:

$$L_h \mathbf{w}_h(G) = \begin{cases} \mathcal{L}_{St_G}[u](B) & \text{if} \quad B \in \Gamma_D \\ \left(\nabla \mathcal{L}_{St_G}[u] \cdot \dfrac{\nabla \mathcal{L}_{St_G}[\phi]}{|\nabla \mathcal{L}_{St_G}[\phi]|}\right)\Big|_B & \text{if} \quad B \in \Gamma_N \end{cases} \tag{2.40}$$

  for any $\mathbf{w}_h \in S(\Omega_h \cup \Gamma_h)$ and $G \in \Gamma_h$;

- $\mathbf{g}_h \in S(\Gamma_h)$ is defined by:

$$\mathbf{g}_h(G) = \begin{cases} g_D(B) & \text{if} \quad B \in \Gamma_D \\ g_N(B) & \text{if} \quad B \in \Gamma_N \end{cases}$$

  for any ghost point $G \in \Gamma_h$.

Let us introduce, for any spatial step h, an exact solver

$$\mathbf{u}_h = \mathcal{S}_h\left(\mathbf{f}_h, \mathbf{g}_h\right)$$

of the system (2.39), and denote by

$$\Re_h \colon S(\Omega_h \cup \Gamma_h) \times S(\Omega_h) \times S(\Gamma_h) \longrightarrow S(\Omega_h \cup \Gamma_h) \tag{2.41}$$

the relaxation operator, namely the iterative scheme

$$\mathbf{u}_h^{(m+1)} = \Re_h \left(\mathbf{u}_h^{(m)}, \mathbf{f}_h, \mathbf{g}_h \right) \tag{2.42}$$

converges to the solution of (2.39) as $n \to +\infty$. In details, the iteration (2.42) summarize the iterative scheme (2.15)-(2.17).

As in one dimensional case, we will intend by (2.42) the Gauss-Seidel version of (2.15)-(2.17), in order to deal with a proper *smoother*, and we have to order all points of $\Omega_h \cup \Gamma_h$ in some way. Let us choose the lexicographic ordering (GS-LEX):

$$(x', y') \leq (x'', y'') \Longleftrightarrow x' < x'' \text{ or } x' = x'',\ y' < y''.$$

The relaxation scheme can be easily extended to more efficient kinds of smoothers, such as Red-Black Gauss-Seidel (see [105]): however, we limit ourselves to study of the GS-LEX smoother.

In order to explain the multigrid approach, we just describe the two-grid correction scheme (TGCS), because all the others schemes, such as V-cycle, W-cycle or Full multigrid, can be easily derived from it (see [105, Sections 2.4, 2.6] for more details). The TGCS consists into the following algorithm:

1. Set initial guess $\mathbf{u}_h = 0$

2. Relax ν_1 times on the finest grid: for k from 1 to ν_1 do

$$\mathbf{u}_h := \Re_h \left(\mathbf{u}_h, \mathbf{f}_h, \mathbf{g}_h \right)$$

3. Compute the following defects:

$$\mathbf{r}_h^{\Omega} = f_h + \Delta_h \, \mathbf{u}_h$$

$$\mathbf{r}_h^{\Gamma} = g_h - L_h \mathbf{u}_h$$

4. Transfer the defects to a coarser grid with spatial step $2h$ by a suitable *restriction operator*

$$\mathbf{r}_{2h}^{\Omega} = I_{2h}^h \left(\mathbf{r}_h^{\Omega} \right)$$

$$\mathbf{r}_{2h}^{\Gamma} = I_{2h}^h \left(\mathbf{r}_h^{\Gamma} \right)$$

5. Solve exactly the residual problem in the coarser grid

$$\mathbf{e}_{2h} = \mathcal{S}_{2h} \left(\mathbf{r}_{2h}^{\Omega}, \mathbf{r}_{2h}^{\Gamma} \right)$$

6. Transfer the error to the finest grid by a suitable *interpolation operator*

$$\mathbf{e}_h = I_h^{2h}(\mathbf{e}_{2h})$$

7. Correct the fine-grid approximation

$$\mathbf{u}_h := \mathbf{u}_h + \mathbf{e}_h$$

8. Relax ν_2 times on the finest grid: for k from 1 to ν_2 do

$$\mathbf{u}_h := \Re_h(\mathbf{u}_h, \mathbf{f}_h, \mathbf{g}_h)$$

We have just to explain the steps concerning grid migration (steps 4 and 6). All the other steps are clear.

2.6.1 Transfer grid operators

2.6.1.1 Restriction operator

Let us consider the total defect $\mathbf{r}_h = (\mathbf{r}_{2h}^{\Omega}, \mathbf{r}_{2h}^{\Gamma})$. If (2.41) has the smoothing property, after ν_1 relaxations (step 2 of the algorithm) we have a smooth defect $\mathbf{r}_h$. Therefore, we can hope to transfer this defect to a coarser grid without losing much information. Abusing of notation, we can say that the defect $\mathbf{r}_h$ belongs to $S(\Omega_h \cup \Gamma_h)$. In order to transfer it to a coarser grid, one strategy could be to extend in some way this defect in the whole computational domain D_h (i.e. $\mathbf{r}_h \in S(D_h)$), in such a way we can use the standard full-weighting stencil for the restriction operator $I_{2h}^h : S(D_h) \to S(D_{2h})$, that is (see [105, pag. 42])

$$I_{2h}^h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{2h}^{h}. \tag{2.43}$$

In general, by the stencil notation

$$I_{2h}^h = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & t_{-1,-1} & t_{-1,0} & t_{-1,1} & \cdots \\ \cdots & t_{0,-1} & t_{0,0} & t_{0,1} & \cdots \\ \cdots & t_{1,-1} & t_{1,0} & t_{1,1} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}_{2h}^{h}$$

we will intend the restriction operator I_{2h}^h defined by:

$$I_{2h}^h \mathbf{w}_h(x,y) = \sum_{(i,j)\in R_k} t_{i,j}\mathbf{w}_h(x+jh, y+ih),$$

where only a finite number of coefficients $t_{i,j}$ is different from zero, and $R_k \equiv \{-k,\dots,k\}^2$ for some positive integer k. In practice $k=1$ allows second order restriction operator.

Let us suppose we have extended the defect to the whole computational domain D_h (as it is carefully described in Sec. 2.6.1.2). Anyhow, since we have different operators for inner equations and for boundary conditions, the defect is smooth separately inside Ω_h and along the ghost point Γ_h (or $D_h - \Omega_h$, because of the extension), but it is not smooth in all $\Omega_h \cup \Gamma_h$.

For this reason, it is convenient to transfer separately to the coarse grid the defects $\mathbf{r}_h^\Omega$ and $\mathbf{r}_h^\Gamma$. Let us define the new restriction operator:

$$\mathcal{I}_{2h}^h \colon S(Z_h) \longrightarrow S(Z_{2h}), \tag{2.44}$$

where Z_h is an arbitrary subset of D_h, and where we intend $Z_{2h} = Z_h \cap \Omega_{2h}$. Let $(x,y) \in Z_{2h}$. We focus our attention to the neighborhood of (x,y), that is $\mathcal{N}(x,y) = \{(x+jh, y+ih) \colon j,i = -1,0,1\}$.

Now consider the maximum full rectangle $\mathcal{T}$ with vertices belonging to $\mathcal{N}(x,y)$ and such that $\mathcal{T} \cap D_h \subseteq Z_h$ (see Fig. 2.2, where $Z_h = \Omega_h$). Therefore, the stencil we use in (x,y) to transfer $\mathbf{w}_h$ to a coarse grid depends on the size of $\mathcal{T}$. In fact, let $\mathcal{T} \cap D_h$ be a 3×3 points (i.e. $\mathcal{N}(x,y) \subseteq Z_h$). In this case we can use the standard full-weighting stencil (2.43).

Now let $\mathcal{T} \cap D_h$ be a 3×2 points. Without loss of generality, we can suppose the vertices of $\mathcal{T}$ are $(x+jh, y+ih)$, with $j \in \{-1,0\}$, $i \in \{-1,1\}$. In this case, the stencil we will use is:

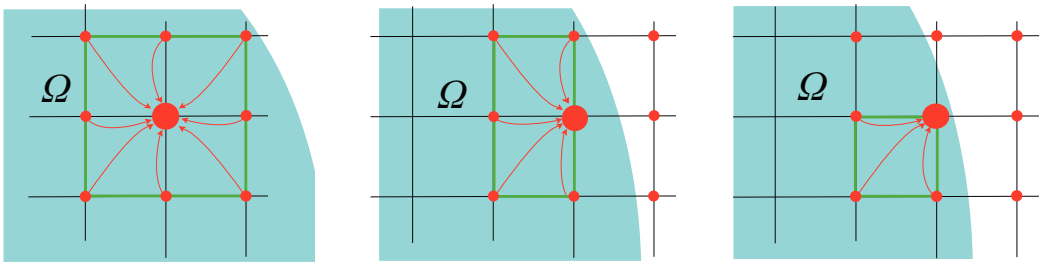
$$\left(I_{2h}^h \mathbf{w}_h\right)(x,y) = \frac{1}{16}\begin{bmatrix} 2 & 2 & 0 \\ 4 & 4 & 0 \\ 2 & 2 & 0 \end{bmatrix}_{2h}^{h} (x,y), \tag{2.45}$$

while, if $\mathcal{T}$ is a 2×2 points, with vertex $(x+jh, y+ih)$, $j,i \in \{-1,0\}$, the stencil will be:

$$\left(I_{2h}^h \mathbf{w}_h\right)(x,y) = \frac{1}{16}\begin{bmatrix} 0 & 0 & 0 \\ 4 & 4 & 0 \\ 4 & 4 & 0 \end{bmatrix}_{2h}^{h} (x,y), \tag{2.46}$$

This three case are summarized in Fig. 2.2 (where $Z_h = \Omega_h$).

Note that the stencils (2.43), (2.46), (2.45) can be derived as tensor products of the one-dimensional restrictions (2.36), (2.37), (2.38).



$$\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}^h_{2h} \qquad \frac{1}{16}\begin{bmatrix}2&2&0\\4&4&0\\2&2&0\end{bmatrix}^h_{2h} \qquad \frac{1}{16}\begin{bmatrix}0&0&0\\4&4&0\\4&4&0\end{bmatrix}^h_{2h}$$

Fig. 2.2: Upper, the nine points of $\mathcal{N}(x,y)$ and the green boundary of the rectangle $\mathcal{T}$. The bold red point is on the coarser and finer grids, while the little red points are on the finer grid. The arrows represent the action of the restriction operators. Below, the respective stencil to be used.

2.6.1.2 Extension of the defect

The goal of this section is to extend the defect $\mathbf{r}_h^\Gamma$ from $S(\Gamma_h)$ to $S(D_h - \Omega_h)$. In every ghost point we store the defect of the boundary condition concerning that ghost point. In formulas, we have seen in step 3 of the TGCS algorithm that $\mathbf{r}_h^\Gamma(G) = (\mathbf{g}_h - L_h\mathbf{u}_h)(G)$, for any ghost point G. But $\mathbf{g}_h(G) = g(B)$ and $L_h\mathbf{u}_h(G)$ is the reconstructed boundary condition in B of the boundary operator L (see (2.40)), where B is the closest boundary point to G (i.e. the orthogonal projection on the boundary). In summary, the defect is stored in a ghost point G, but it is geometrically referred to a boundary point B placed along the normal direction. When we switch to a coarse grid, some ghost point G_1 may not be ghost point in the fine grid, i.e. $\Gamma_{2h} \subseteq \Gamma_h$ is not true (see Fig. 2.3).

Then, no acceptable value of the defect is stored in G_1. Indeed, we expect that $\mathbf{r}_{2h}$ has in the ghost point G_1 the defect of the boundary conditions referred to B_1. Hence, if we extend the defect $\mathbf{r}_h$ outside Ω_h constant along the normal lines to the boundary, we will find $\mathbf{r}_h^\Gamma(G_1)$ as an approximation of the defect of the boundary conditions in B_1. After coarsening (performed using only points outside Ω_h, as described before), the ghost points of the coarser grid will contain the expected values of the defect.

The extension of the defect $\mathbf{r}_h^\Gamma$ is performed by solving the transport equation

$$\frac{\partial r^\Gamma}{\partial \tau} + \nabla r^\Gamma \cdot \mathbf{n} = 0$$

in a few steps of a fictitious time τ, where $\mathbf{n} \equiv (n_x, n_y) = \nabla\phi/|\nabla\phi|$ is the

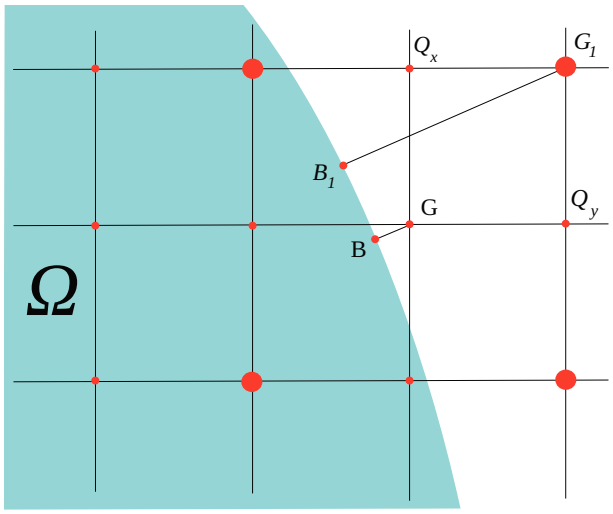


Fig. 2.3: Red bold and small points are grid points of Ω_h, while red bold points are grid points of Ω_{2h}. G_1 is a ghost point on the coarser grid, but not on the finer grid, then no value of the defect is stored in it. Q_x and Q_y are the two upwind near points to G_1.

unit normal vector to the level-set. In details, we compute few steps of the following iteration scheme:

$$\begin{aligned}\mathbf{r}_h^{\Gamma\,(m+1)}(P) &= \mathbf{r}_h^{\Gamma\,(m)}(P) \\ &+ \frac{\Delta\tau}{h}\left(\left(\mathbf{r}_h^{\Gamma\,(m)}(P) - \mathbf{r}_h^{\Gamma\,(m)}(Q_x)\right)|n_x| + \left(\mathbf{r}_h^{\Gamma\,(m)}(P) - \mathbf{r}_h^{\Gamma\,(m)}(Q_y)\right)|n_y|\right)\end{aligned} \tag{2.47}$$

for all $P \in D_h - (\Omega_h \cup \Gamma_h)$, where Q_x and Q_y are the two upwind near points to P, i.e.

$$Q_x = P - \operatorname{sgn}(n_x)\, h\, \mathbf{i}, \quad Q_y = P - \operatorname{sgn}(n_y)\, h\, \mathbf{j}.$$

However, it is sufficient to perform the iteration (2.47) only in a narrow band with width $3\,h$. In order to speed up the extension, we can perform (2.47) in a Gauss-Seidel fashion, sorting points in $D_h - (\Omega_h \cup \Gamma_h)$ by the distance from the boundary (it can be done using the distance function ϕ), and starting from the closest ghost point to the boundary P.

Finally, we can resume the extension process introducing an extension operator, which in practice depends only on the set of ghost point Γ_h and on the discretized signed distance function ϕ_h. Therefore:

$$\mathcal{E}[\Gamma_h; \phi_h] \colon S(\Gamma_h) \longrightarrow S(D_h - \Omega_h), \tag{2.48}$$

and then the extension procedure and the restriction of the defect $\mathbf{r}_h^\Gamma$ can be resumed as follows:

$$I_{2h}^h(\mathcal{E}[\Gamma_h;\phi_h](\mathbf{r}_h^\Gamma)).$$

2.6.1.3 Interpolation

Since the interpolation operator acts on the error, which is continuous across the boundary, we just use the standard bilinear interpolation operator:

$$I_h^{2h} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h}. \tag{2.49}$$

2.7 Numerical tests

In all the following numerical tests we always choose the Dirichlet and Neumann parts of $\Gamma = \partial\Omega$ as:

$$\Gamma_D = \{(x,y) \in \Gamma \colon x \leq 0\}, \quad \Gamma_N = \{(x,y) \in \Gamma \colon x > 0\}.$$

The Local Fourier Analysis (LFA) is a powerful tool to obtain the theoretically convergence factor by analyzing separately the action of different parts of the multigrid algorithm to high and low frequency components of the error. For a detailed explanation of the LFA, we refer to [105, Ch. 4].

Before to apply the LFA, one has to be sure that the relaxation operator (2.41) has the *smoothing property*. Roughly speaking, the *smoothing property* is the property to dump high frequency components of the error, in order to make it smooth after few relaxation sweeps.

When the multigrid algorithm applies to a regular rectangular domain, the LFA and smoothing analysis are well studied. In the case of arbitrary domain, as Achi Brandt points out in [105, pag. 587], there are some boundary related difficulties about the discretization and relaxation near the boundary:

- There is no a general smoothing analysis when the boundary is not aligned with the grid;
- The residuals should be reduced near boundaries more than in the interior;
- The coarsest grid has not to be too coarse, because it should catch the curvature of the boundary in order to guarantee the convergence.

Now, we perform a numerical test in order to check if the convergence factor is close to the predicted one by LFA, which is obtained for rectangular domain with periodic boundary conditions, i.e. without taking into account boundary effects. Note that the multigrid algorithm described before may be seen as an iterative scheme:

$$\mathbf{u}_h^{(m+1)} = M_h\, \mathbf{u}_h^{(m)} + \mathbf{b}_h$$

for some matrix M_h and vector $\mathbf{b}_h$. We call ρ the convergence factor, which is the spectral radius of the matrix M_h. For rectangular domain with periodic boundary conditions and constant coefficients, the convergence factor is said to be local and it is denoted by ρ_{loc}. The convergence factors predicted by LFA for Gauss-Seidel LEX relaxation and FW restriction operator are listed in Table 2.2 (see [105, pag. 117]).

Table 2.2: Predicted convergence factor ρ_{loc} by LFA for GS-LEX and FW restriction operator.

$\nu = \nu_1 + \nu_2$	1	2	3	4
ρ_{loc}	0.400	0.193	0.119	0.084

In all the numerical tests we perform, the convergence factor is estimated as the ratio of consecutive defects, i.e.:

$$\rho = \rho^{(m)} = \frac{\left\| \mathbf{r}_h^{(m)} \right\|_\infty}{\left\| \mathbf{r}_h^{(m-1)} \right\|_\infty}$$

for m very large. In order to avoid difficulties related to numerical instability related to the machine precision, we will always use the homogeneous model problem, namely (1.1) with $f = g_D = g_N = 0$, and perform the multigrid algorithm starting from an initial guess different from zero. Since we are just interested at the convergence factor and not at the numerical solution itself (which approaches zero indefinitely for homogeneous problem), a reasonable stopping criterion will be

$$\frac{\left|\rho^{(m)} - \rho^{(m-1)}\right|}{\rho^{(m)}} < 10^{-3}.$$

Note that, since we want to study the efficiency of the multigrid components proposed in this paper (smoother, restriction, ...), it is sufficient to study basic kind of multigrid such as V-cycle and W-cycle, while a more efficient algorithm (such as FMG) can be easily derived.

2.7.1 1D numerical test

Referring to Sec. 1.2.2, let us choose $[a, b] = [-0.743, 0.843] \subseteq [-1, 1]$. The finest grid is obtained dividing the whole computational domain $[-1, 1]$ into $N = 64$ subintervals, while the coarsest grid is obtained dividing $[-1, 1]$ into $Nc = 8$ subintervals. The computed convergence factors are very close to those ones predicted by LFA (Table 2.2), namely $\rho = 0.185$ for $\nu = 2$ and $\rho = 0.122$ for $\nu = 3$.

2.7.2 An initial test in 2D

We start testing the multigrid algorithm on a circular domain Ω with center $(\sqrt{2}/20, \sqrt{3}/30)$ and radius $r = 0.563$ (Fig. 2.4).

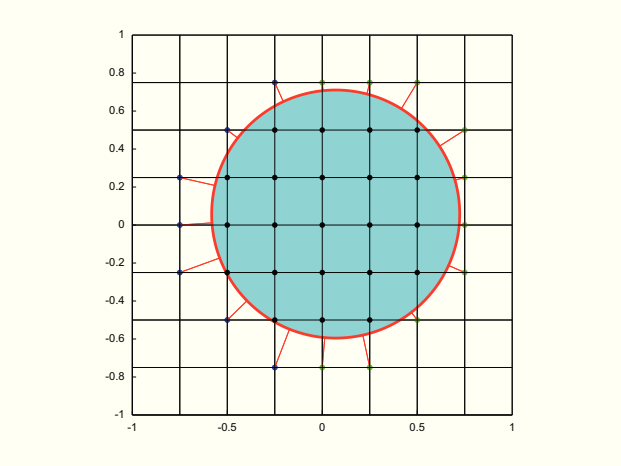


Fig. 2.4: Circular domain and the coarsest grid used to capture the curvature. Blue ghost points refer to Dirichlet condition, while green ghost points refer to Neumann condition. Red lines are normal to the boundary.

The measured convergence factors for TGCS, V-cycle and W-cycle are listed in Table 2.3.

As we can see, the measured convergence factor is far from the predicted one by LFA (Table 2.2). Then, some boundary effect degrades the convergence factor. Note that in 1D such boundary effects do not degrade the convergence factor (Ex. 2.7.1), because we have only two boundary points, and the degradation is due to the oscillating behavior of the residual on the tangential direction to the boundary, that does not exist in 1D. Then, in 2D we must smooth the error also along the tangential direction to the boundary. To overcome this difficulty, we apply, after a single relaxation and at

Table 2.3: Measured convergence factor ρ with $\nu_1 = \nu_2 = 1$ on the left and with $\nu_1 = 2$, $\nu_2 = 1$ on the right.

N	TGCS	V-cycle	W-cycle	N	TGCS	V-cycle	W-cycle
64	0.67	0.68	0.71	64	0.58	0.72	0.58
128	0.68	0.73	0.68	128	0.58	0.73	0.59
256	0.70	0.71	0.70	256	0.61	0.83	0.60

each grid level, λ extra relaxation sweeps on all ghost points Γ_h and on all inside grid points of Ω_h within $\delta > 0$ distance from the boundary (the extra computational work is $O(N)$, then negligible as $N \to \infty$). It can be proved numerically that a good choice of these parameters will be:

$$\lambda = 5, \quad \delta = 3\,h. \tag{2.50}$$

The explanation of the optimal value $\lambda = 5$ is the following: the degradation observed in Table 2.3 is an indication that the error decays much slower at the boundary. Assuming that the convergence factor in Table 2.3 is essentially the convergence factor at the boundary, ρ_B, we want to match it with the convergence factor at the bulk, therefore λ_{opt} is the smallest value of λ for which $\rho_B^{\lambda+1} \leq \rho_I$. The value ρ_I, in turn, can be computed as the convergence factor for large value of λ.

Investigating the smoothing property, we observe that choosing the initial error as an high frequency component, the error is not smoothed after few relaxation sweeps. While, if we add the extra-relaxations, the error become sufficiently smooth (Figs. 2.5-2.6).

2.7.3 Numerical tests in 2D

In this section we confirm numerically the improvement of the convergence factor if we apply extra-relaxations, and we compare the relaxation scheme with other well-known alternatives such as the Kaczmarz and the block relaxation. In all numerical tests, we choose an arbitrary domain Ω assigning a level-set function ϕ_0. Then we reinitialize it by the procedure described in Section 1.1.1, obtaining the signed distance function ϕ. Afterwards, we perform the multigrid technique applying the W-cycle algorithm instead of the V-cycle, to ensure the independence of the convergence factor ρ from the step size h (as explained for example in [105, pag. 78]). Several tests are performed for each domain, based on the different size of the finest and coarsest grids. The finest grid is obtained dividing the whole computational domain D into N subintervals in each Cartesian direction, while the coarsest

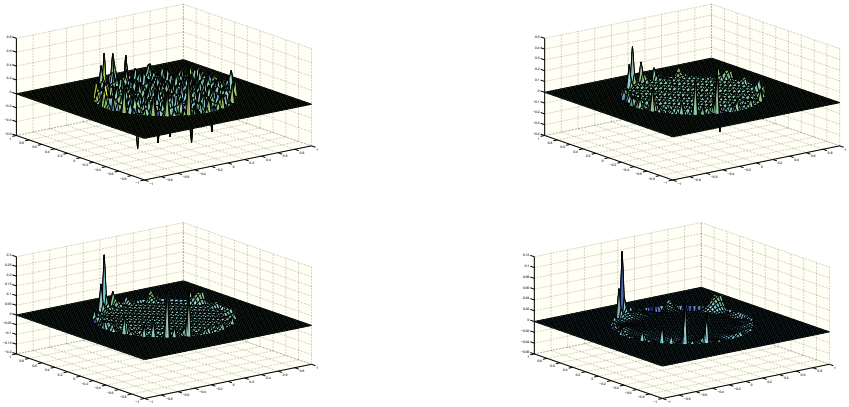


Fig. 2.5: High frequency initial error after 1 (up-left), 3 (up-right), 5 (down-left), 10 (down-right) relaxation sweeps and $\lambda = 0$ extra-relaxations.

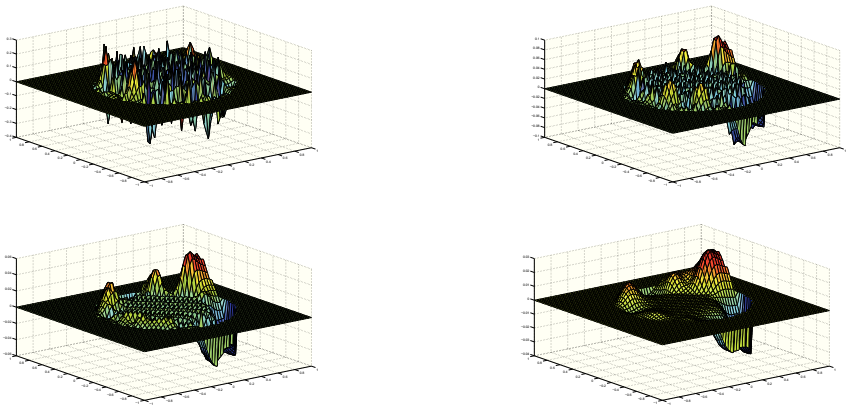


Fig. 2.6: High frequency initial error after 1 (up-left), 3 (up-right), 5 (down-left), 10 (down-right) relaxation sweeps and $\lambda = 5$ extra-relaxations.

grid is obtained replacing N with N_c. The solution on the coarsest grid is obtained by a direct solver.

2.7.3.1 Circular domain

In this case we can choose as a level-set function directly the signed distance function, which is known analytically:

$$\phi(x, y) = \sqrt{(x - \sqrt{2}/20)^2 + (y - \sqrt{3}/30)^2} - 0.563.$$

The zero level-set is represented in Fig 2.7 (top-left). Different value of the convergence factor are listed in Table 2.4 (for $\nu = \nu_1 + \nu_2 = 2$ and

$\nu = 3$). They are really improved with respect to those obtained without extra-relaxations (Table 2.3) for the same test.

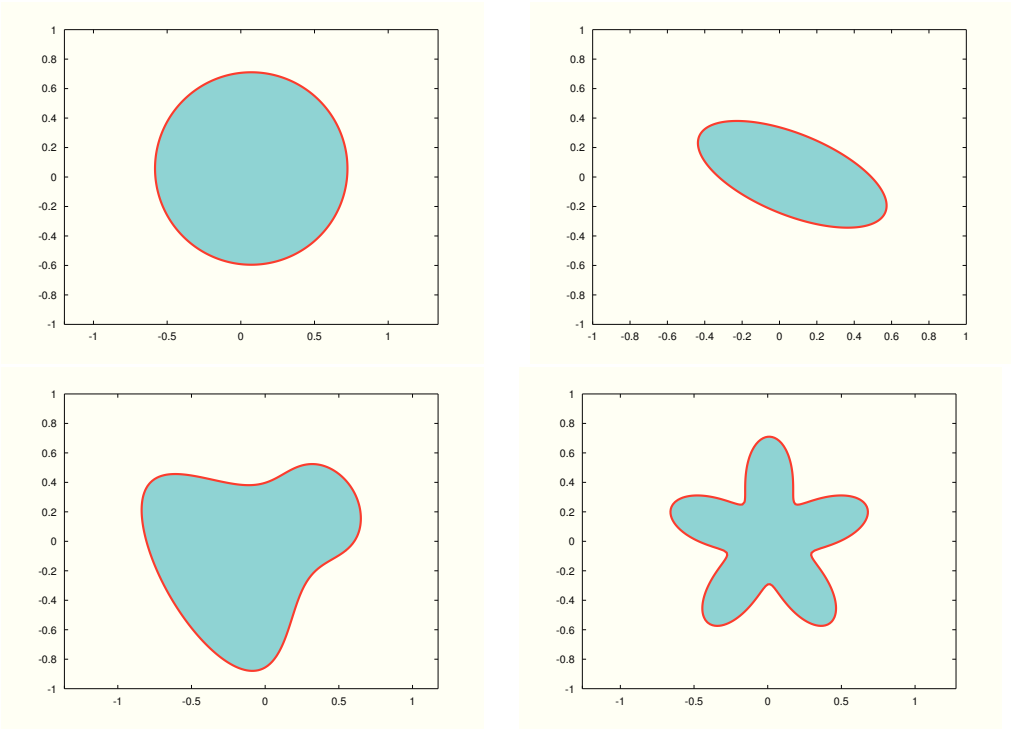


Fig. 2.7: Different domains used in the numerical tests: Example 2.7.3.1 (top-left), 2.7.3.3 (top-right), 2.7.3.4 (bottom-left), 2.7.3.5 (bottom-right).

2.7.3.2 Comparison with the Kaczmarz and the block relaxations

Note that the relaxation scheme (2.9), (2.12), (2.13) is composed by a Gauss-Seidel iteration over inner grid points and a suitable relaxation over ghost points (boundary conditions). As an alternative to the relaxation of the boundary condition, we can use the Kaczmarz relaxation [64] near the boundary, which is known to be unconditionally convergent. Let us recall the Kaczmarz iteration scheme for a subset of equations $\mathcal{J} \subseteq \{\dots, N_i + N_g\}$ of a linear system $Lu = f$:

$$
\begin{aligned}
&u^{TEMP} = u^{(m)},\\
&\text{for } j \in \mathcal{J} \text{ do:} \quad u^{TEMP} := u^{TEMP} + \frac{f_j - < l_j, u^{TEMP} >}{\|l_j\|_2^2} l_j^T,\\
&u^{(m+1)} = u^{TEMP}.
\end{aligned}
$$

Table 2.4: Convergence factor of the numerical test of Section 2.7.3.1. We use $N \times N$ number of grid points in the finest grid; $N_c \times N_c$ number of grid points in the coarsest grid. Top: $\nu = \nu_1 + \nu_2 = 2$, bottom: $\nu = \nu_1 + \nu_2 = 3$.

N / N_c	16	32	64	128	256
8	0.052	0.053	0.11	0.13	0.14
16		0.061	0.11	0.13	0.14
32			0.11	0.13	0.14
64				0.13	0.14
128					0.14

N / N_c	16	32	64	128	256
8	0.059	0.035	0.09	0.08	0.08
16		0.041	0.09	0.08	0.08
32			0.09	0.08	0.08
64				0.09	0.08
128					0.09

The symbol $< \cdot, \cdot >$ denotes the inner product operator and l_j is the j-th row of the matrix L. If we choose $J = \{\ldots, N_i + N_g\}$ then we obtain the classical Kaczmarz relaxation scheme for the solution of the linear system $Lu = f$, and the iteration scheme is equivalent to a Gauss-Seidel relaxation for the system $L^T Lu = L^T f$. In our case, one iteration of the alternative relaxation we want to study is composed as follows: we perform a Gauss-Seidel sweep in the interior of the domain, followed by λ Kaczmarz iterations over ghost points and inner points close to the boundary (say within δ distance from the boundary).

Another alternative is represented by the block relaxation [38]. As we point out in Sec. 2.5, the elimination of the boundary conditions is hard to perform in high dimensions, while in one dimension it is a trivial task and leads to a diagonally dominant linear system. A middle ground between the elimination of the boundary conditions and the relaxation operator we use in this paper is the block relaxation. Let us describe it in details. For each grid point $P \in \Omega_h \cup \Gamma_h$ we choose a stencil $St_P \subseteq \Omega_h \cup \Gamma_h$. For instance, if $P \in \Omega_h$ we choose $St_P = St_{P9} \cap (\Omega_h \cup \Gamma_h)$, where St_{P9} is the 3×3 stencil centered at P, else if $P \in \Gamma_h$ we choose the stencil St_P defined in (1.12). One iteration of the alternative relaxation is composed as follows. We perform a Gauss-Seidel sweep in the interior of the domain except in grid points within

δ distance from the boundary. For each grid point $P \in \Omega_h \cup \Gamma_h$ within δ distance from the boundary we rewrite the linear system $Lu = f$ as follows (by a permutation of rows):

$$Lu = f \Leftrightarrow \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

where u_1 is referred to those grid points belonging to St_P. Therefore, we update the values of u_1 as:

$$u_1 = A_{1,1}^{-1} \left(f_1 - A_{1,2} u_2 \right).$$

We perform a comparison between the relaxation proposed in this paper (that we call *new iteration* in the following plots) and the two alternative relaxation described above. Such a comparison is carried out in terms of smoothing factor and convergence factor. We perform the comparison using the TGCS for the test case of the circular domain 2.7.3.1 with $N = 64$.

In Fig. 2.8 we plot the smoothing factor μ for the three iteration schemes, which is estimated by the ratio of subsequent defects after each iteration, i.e.

$$\mu^{(m)} = \frac{\left\| \mathbf{r}_h^{(m)} \right\|_\infty}{\left\| \mathbf{r}_h^{(m-1)} \right\|_\infty}.$$

In practice, we perform only the iteration schemes, without taking into account the effects of the multigrid procedure. In order to better capture the behavior of the smoothing factor, we choose an initial guess being highly oscillant, for example $u = \sin(40\pi x)\sin(50\pi y)$.

In Fig. 2.9 we depict the convergence factor ρ for the Kaczmarz and the new iteration against the number of extra-relaxations λ (for comparison, we also plot the convergence factor of the block relaxation as an horizontal line, since it does not depend on λ). After five extra-relaxations, the new iteration reaches a plate configuration, since it achieves the convergence factor of the Gauss-Seidel smoother for inner equations, i.e. the convergence factor predicted by the LFA (see Table 2.2). The Kaczmarz iteration falls down slower, while the block iteration already provides the optimal convergence factor. The computational cost of five point-iterations of the new method is considerably lower than the cost of one block-iteration.

2.7.3.3 Ellipsoidal domain

The level-set function is:

$$\phi(x, y) = \frac{(X(x, y) - \sqrt{2}/20)^2}{0.563^2} + \frac{(Y(x, y) - \sqrt{3}/30)^2}{0.263^2} - 1$$

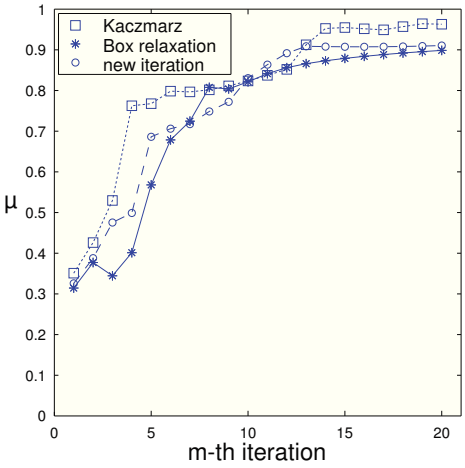


Fig. 2.8: Smoothing factor μ against the number of iterations for the three iterative schemes: Kaczmarz relaxation, Block relaxation, new iteration.

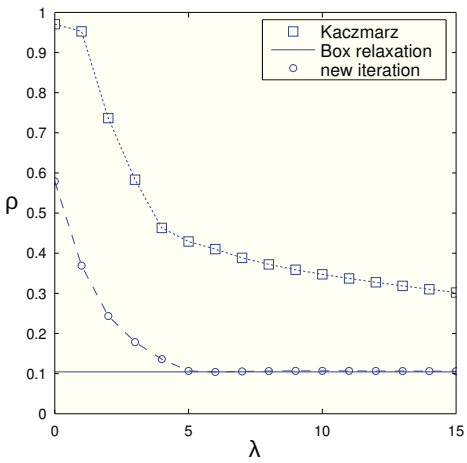


Fig. 2.9: Convergence factor ρ of the entire multigrid against the number of extra-relaxations λ for the three iterative schemes: Kaczmarz relaxation, Block relaxation, new iteration.

where

$$X(x,y) = \cos(\pi/6)\, x - \sin(\pi/6)\, y, \quad X(x,y) = \sin(\pi/6)\, x + \cos(\pi/6)\, y,$$

and the zero level-set is represented in Fig. 2.7 (top-right). The convergence factor obtained are listed in Table 2.5 (for $\nu = \nu_1 + \nu_2 = 2$ and $\nu = 3$). We observe as the convergence factor degrade choosing a coarsest grid too much coarse, but starting from a certain level of coarsest grid it is relatively close to the predicted convergence factor by LFA (Table 2.2).

2.7.3.4 Saddle-shaped domain

The level-set function is:

$$\phi(x,y) = 9\left(\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right)^2 + \left(\frac{3\sqrt{3}}{2}x + \frac{3}{2}y - 1\right)^2 \sin\left(\frac{3\sqrt{3}}{2}x + \frac{3}{2}y - 1\right) - 1$$

and the zero level-set is represented in Fig. 2.7 (bottom-left). The convergence factor obtained for $\nu = \nu_1 + \nu_2 = 3$ are listed in Table 2.6 (left). Also in this case, only if we choose $N = 16$ and $N_c = 8$ (which actually is TGCS) the convergence factor is degraded.

Table 2.5: Convergence factor of the numerical test of Section 2.7.3.3. We use $N \times N$ number of grid points in the finest grid; $N_c \times N_c$ number of grid points in the coarsest grid. Top: $\nu = \nu_1 + \nu_2 = 2$, bottom $\nu = \nu_1 + \nu_2 = 3$.

N / N_c	16	32	64	128	256
8	0.34	0.09	0.14	0.14	0.15
16		0.65	0.45	0.19	0.15
32			0.14	0.14	0.15
64				0.15	0.15
128					0.15

N / N_c	16	32	64	128	256
8	0.44	0.06	0.12	0.11	0.09
16		0.55	0.30	0.09	0.09
32			0.13	0.10	0.09
64				0.12	0.08
128					0.09

2.7.3.5 Flower-shaped domain

The level-set function is:

$$\phi = r - 0.5 - \frac{y^5 + 5x^4y - 10x^2y^3}{5r^5}, \quad r = \sqrt{x^2 + y^2}$$

and the zero level-set is represented in Fig. 2.7 (bottom-right). The convergence factor obtained for $\nu = \nu_1 + \nu_2 = 3$ are listed in Table 2.6 (right). This is the hardest numerical test, because of the indentation of the boundary. We need to start from a coarsest level $N_c = 32$ to correctly capture the boundary profile and to make the discretization accurate.

Table 2.6: Convergence factor of the numerical test of Sections 2.7.3.4 (top) and 2.7.3.5 (bottom). We use $N \times N$ number of grid points in the finest grid; $N_c \times N_c$ number of grid points in the coarsest grid. In this test we use $\nu = \nu_1 + \nu_2 = 3$. For the flower-shaped domain, if $N_c = 8$ the multigrid does not converge.

N / N_c	16	32	64	128	256
8	0.36	0.08	0.09	0.12	0.09
16		0.12	0.09	0.12	0.09
32			0.09	0.12	0.09
64				0.13	0.09
128					0.09

N / N_c	16	32	64	128	256
8	n.c.	n.c.	n.c.	n.c.	n.c.
16		0.89	0.75	0.50	0.25
32			0.49	0.25	0.12
64				0.24	0.11
128					0.09

Chapter 3

Discontinuous coefficents: 1D case

In this section we obtain a second order accurate numerical method to solve an elliptic equation with discontinuous coefficients. After introducing the model problem, we provide a discretization and an iterative solver of the linear system. In some applications one may be interested in second order accuracy also for the derivative of the solution. In numerical tests of Sec. 3.3 we show that the method is second order accurate in the solution and in its first derivative.

3.1 Model problem

Let us consider the model problem

$$\begin{aligned} -\frac{d}{dx}\left(\gamma\frac{du}{dx}\right) &= f \text{ in } \Omega = [0,1], \\ u(0) = g_0, \qquad & u(1) = g_1. \end{aligned} \tag{3.1}$$

where the diffusion coefficient $\gamma\colon [0,1] \to \mathbb{R}$ jumps on an interface $\alpha \in]0,1[$, i.e., is a smooth function in $[0,\alpha[$ and in $]\alpha,1]$, but may be discontinuous across α. We assume $\gamma > \epsilon > 0$ in all the domain. If we solve this problem by standard central differences on a uniform grid, the accuracy of the method degrades to first order.

Let

$$u^L = u|_{[0,\alpha[}, \quad u^R = u|_{]\alpha,1]}, \quad \gamma^L = \gamma|_{[0,\alpha[}, \quad \gamma^R = \gamma|_{]\alpha,1]}$$

be the restriction functions of the solution and of the coefficient on the two

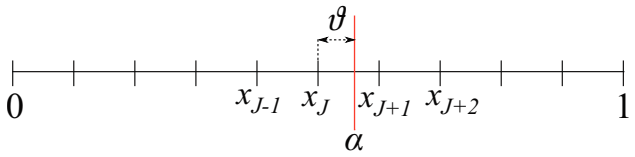


Fig. 3.1: Computational domain Ω with an arbitrary interface α.

subdomains. We split the problem into the following subproblems:

$$\begin{aligned} -\frac{d}{dx}\left(\gamma^L \frac{du^L}{dx}\right) &= f \text{ in } [0,\alpha[\\ u^L(0) &= g_0, \end{aligned} \tag{3.2}$$

$$\begin{aligned} -\frac{d}{dx}\left(\gamma^R \frac{du^R}{dx}\right) &= f \text{ in }]\alpha,1] \\ u^R(1) &= g_1. \end{aligned} \tag{3.3}$$

In order to close the problem, we must provide an additional boundary condition for each of u^L and u^R on the interface α. This additional conditions are inferred to the requirement that the solution u and the flux $\gamma u'$ are continuous across α. Introducing the *jumping operator* on α

$$[w] = \lim_{x\to\alpha^+} w - \lim_{x\to\alpha^-} w,$$

the additional boundary conditions may be resumed as

$$[u] = 0, \quad [\gamma u'] = 0$$

and are called *transmission conditions* [92]. They can be inferred by a physical requirement: for instance, in steady-state diffusion problems in two materials, the temperature and its flux are required to be continuous across α. Non-homogeneous interface conditions may appear, for example, in presence of a delta-function on the right hand side $f = f_1 + \delta_\alpha$, with $f_1 \in C^0([0,1])$. Precisely, the two following problems are equivalent:

$$\begin{aligned} -\frac{d}{dx}(\gamma\,\frac{du}{dx}) &= f_1 + C\delta_\alpha \text{ in } [0,1] \\ u(0) = g_0, \qquad & u(1) = g_1, \end{aligned}$$

$$\begin{aligned} -\frac{d}{dx}(\gamma\,\frac{du}{dx}) &= f_1 \text{ in } [0,\alpha[\cup]\alpha,1] \\ u(0) = g_0, \qquad & u(1) = g_1 \\ [u] = 0, \qquad & [\gamma\, u'] = -C. \end{aligned}$$

In the following we suppose the right-hand side is a regular function in the two sub-regions, and non-homogeneous interface conditions are allowed:

$$[u] = g_D, \quad [\gamma u'] = g_N. \tag{3.4}$$

Such general case is relevant for some applications, for example pressure equation for incompressible flow in presence of surface tension at the interface.
The two subproblems (3.2) and (3.3) are then coupled on α and cannot be solved separately. The whole problem becomes

Model problem 6

$$-\frac{d}{dx}\left(\gamma^L \frac{d\,u^L}{dx}\right) = f \ in\ [0,\alpha[\tag{3.5}$$

$$-\frac{d}{dx}\left(\gamma^R \frac{d\,u^R}{dx}\right) = f \ in\ ]\alpha,1] \tag{3.6}$$

$$u^L(0) = g_0, \qquad u^R(1) = g_1 \tag{3.7}$$

$$[u] = g_D, \qquad [\gamma u'] = g_N. \tag{3.8}$$

3.1.1 Discretization

Let N be an integer, $h = 1/(N+1)$ be the spatial step and $x_0, x_1, \ldots, x_N, x_{N+1}$ be the equally spaced grid points, with $x_j = j\,h$. Let J be such that $x_J \leq \alpha < x_{J+1}$ (see Fig. 3.1). We write $J = \lfloor \alpha \rfloor$, where $\lfloor \cdot \rfloor$ denotes the integer part. We will denote by $\mathcal{L}_j[w]$ the quadratic interpolant of w in nodes $\{x_{j-1}, x_j, x_{j+1}\}$. By u_j^L $[u_j^R]$ we denote the component of the numerical solution which approximates $u^L(x_j)$ $[u^R(x_j)]$, while we intend $f_j = f(x_j)$, $\gamma_j^L = \gamma^L(x_j)$, $\gamma_j^R = \gamma^R(x_j)$.
Let us discretize the system (3.7). Discretizing Equation (3.5) on nodes $x_1, x_2, \ldots, x_J$ using central differences for the solution u^L and linear interpolation for the coefficient function γ^L, we obtain:

$$\frac{1}{h^2}\left(\gamma^L_{j-\frac{1}{2}}\left(u_j^L - u_{j-1}^L\right) + \gamma^L_{j+\frac{1}{2}}\left(u_j^L - u_{j+1}^L\right)\right) = f_j, \quad j = 1, \ldots J, \tag{3.9}$$

where $\gamma^L_{j+1/2} = (\gamma_j^L + \gamma_{j+1}^L)/2$. In Eq. (3.9) for $j = 1$ the value u_0^L is given by the Dirichlet condition (3.7): $u_0^L = g_0$. It can be easily eliminated from (3.9), but we will leave it in the system just for simplicity. The same applies for u_N^R discretizing Eq. (3.6) in node x_N.
Eq. (3.9) for $j = J$ needs to know the values of u^L and γ^L in node x_{J+1}. Since u' and γ are discontinuous, we cannot use respectively u_{J+1}^R and γ_{J+1}^R, because

this may result in a loss of accuracy, since it smears out the coefficient γ and the numerical solution itself, while both jump on the interface. Then we need to add an additional grid point value for the numerical solution $u^L(x_{J+1})$, called *ghost point value*, and to extrapolate γ^L up to the first ghost point x_{J+1}. The same argument holds for u^R and γ^R in their ghost point x_J, when discretizing Eq. (3.6) in node x_{J+1}.
The unknowns of the numerical method are therefore the $N+4$ quantities

$$u_0^L, \dots, u_{J+1}^L, u_J^R, \dots, u_{N+1}^R. \tag{3.10}$$

This approach has been called *Ghost Fluid Method* and used in the context of multi-fluid flows [41]. The two additional unknowns u_{J+1}^L and u_J^R require two additional boundary conditions to close the system, which are given by the transmission conditions (3.4), resulting in a 2×2 sub-system. We will not solve this sub-system for u_{J+1}^L and u_J^R, but we instead leave it in the whole linear system, which will be solved iteratively. The extrapolation for the coefficient functions γ^L and γ^R is simple linear extrapolation:

$$\gamma_{J+1}^L = 2\,\gamma_J^L - \gamma_{J-1}^L, \quad \gamma_J^R = 2\,\gamma_{J+1}^L - \gamma_{J+2}^L.$$

Using then central differences to discretize (3.5) and (3.6), linear and quadratic interpolation to discretize respectively the two conditions (3.8), we obtain the following second order $(N+4) \times (N+4)$ linear system:

$$u_0^L = g_0 \tag{3.11}$$

$$\frac{1}{h^2}\left(\gamma_{j-\frac{1}{2}}^L \left(u_j^L - u_{j-1}^L\right) + \gamma_{j+\frac{1}{2}}^L \left(u_j^L - u_{j+1}^L\right)\right) = f_j \qquad j = 1, \dots J \tag{3.12}$$

$$\left((1-\vartheta)u_J^R + \vartheta u_{J+1}^R\right) - \left((1-\vartheta)u_J^L + \vartheta u_{J+1}^L\right) = g_D \tag{3.13}$$

$$\gamma_\alpha^R\, \mathcal{L}'_J[u^R](\alpha) - \gamma_\alpha^L\, \mathcal{L}'_{J-1}[u^L](\alpha) = g_N \tag{3.14}$$

$$\frac{1}{h^2}\left(\gamma_{j-\frac{1}{2}}^R \left(u_j^R - u_{j-1}^R\right) + \gamma_{j+\frac{1}{2}}^R \left(u_j^R - u_{j+1}^R\right)\right) = f_j \qquad j = J+1, \dots N \tag{3.15}$$

$$u_N^R + 1 = g_1, \tag{3.16}$$

with γ_α^L and γ_α^R obtained by linear interpolation:

$$\gamma_\alpha^L = (1-\vartheta)\gamma_J^L + \vartheta\gamma_{J+1}^L, \quad \gamma_\alpha^R = (1-\vartheta)\gamma_J^R + \vartheta\gamma_{J+1}^R$$

and $\vartheta = (\alpha - x_J)/h \in [0,1]$.
If we apply a simple iterative method such as Gauss-Seidel or Jacobi to this linear system, in general it will not converge, unless we solve the 2×2 sub-system of transmission conditions, eliminating them from the whole system.

This elimination is easy to perform in one dimension, but becomes quite involved in higher dimension. Therefore, we prefer to work with the whole linear system without eliminate transmission conditions from it, in order to extend the method to higher dimension in Chapter 4. Then we have to find a different approach to solve iteratively the previous linear system. This can be done by relaxing the transmission conditions.

3.1.2 Relaxation scheme

In order to find a convergent iterative method to solve the linear system (3.11)-(3.16), following the approach introduced in Sec. 2.2, we solve the *associate time-dependent* problem in the unknowns $u^L(x,t)$ and $u^R(x,t)$ for $(x,t) \in [0,1] \times (0,+\infty)$:

$$
\begin{aligned}
u^L(0,t) &= g_0 && (3.17)\\
\frac{\partial u^L}{\partial t} &= \mu\left(\frac{\partial}{\partial x}\left(\gamma^L \frac{\partial u^L}{\partial x}\right) + f\right), \quad x \in [0,\alpha[&& (3.18)\\
\left.\frac{\partial u^L}{\partial t}\right|_{x=\alpha} &= \mu_N\left(\left[\gamma \frac{\partial u}{\partial x}\right] - g_N\right) && (3.19)\\
\left.\frac{\partial u^R}{\partial t}\right|_{x=\alpha} &= \mu_D\left(g_D - [u]\right) && (3.20)\\
\frac{\partial u^R}{\partial t} &= \mu\frac{\partial}{\partial x}\left(\gamma^R \frac{\partial u^R}{\partial x}\right) + f, \quad x \in]\alpha,1] && (3.21)\\
u^R(1,t) &= g_1. && (3.22)
\end{aligned}
$$

where μ is a positive function, and μ_D and μ_N are two positive constants, that will be set in Sec. 3.1.3 to satisfy some stability condition.
The choice of the sign of the two constants μ_D and μ_N is crucial and requires some explanation. Roughly speaking, when replacing a vector equation $F(w) = 0$ for $F : \mathbb{R}^m \to \mathbb{R}^m$ by $\dot{\omega} = F(\omega)$, we have to be sure that the solution is asymptotically stable, i.e. that $\lambda(\nabla_\omega F) < 0$. Eq. (3.20) will be used to compute u_J^R, therefore the derivative of the right hand side of Eq. (3.20) with respect to u_J^R has to be negative, to ensure convergence to equilibrium. Eq. (3.19) is used to determine u_{J+1}^L by a transport equation on $u^L(x,t)$. Since $x_{J+1} > \alpha$ the propagation speed $\mu_N \gamma^L$ associated to $u^L(x,t)$, has to be positive.
We are obviously interested in the steady-state solution and the time t represents an iterative parameter. We observe that transmission conditions (3.19)

and (3.20) can be replaced by

$$\begin{aligned} \left.\frac{\partial u^R}{\partial t}\right|_{x=\alpha} &= \mu_N \left(g_N - \left[\gamma \frac{\partial u}{\partial x} \right] \right) \\ \left.\frac{\partial u^L}{\partial t}\right|_{x=\alpha} &= \mu_D \left([u] - g_D \right) \end{aligned}$$

because both choices lead to the same steady state conditions.
To obtain a second order accurate solution in space we are allowed to discretize first order accurate the time derivative. Using forward Euler in time and central differences in space for (3.18) and (3.21), we obtain (superscripts L and R are omitted):

$$u_j^{(m+1)} = u_j^{(m)} + \mu_j \, \Delta t \left(f_j - \frac{\gamma_{j-\frac{1}{2}} \left(u_j^{(m)} - u_{j-1}^{(m)} \right) + \gamma_{j+\frac{1}{2}} \left(u_j^{(m)} - u_{j+1}^{(m)} \right)}{h^2} \right), \tag{3.23}$$

where $j = 1, \ldots, J$ for u^L and $j = J+1, \ldots, N$ for u^R. Choosing the maximum time step allowed by the CFL condition for diffusion equation, i.e., $\mu_j \, \Delta t = h^2/(\gamma_{j+1/2} + \gamma_{j-1/2})$, Eq. (3.23) becomes:

$$u_j^{(m+1)} = \frac{1}{\gamma_{j-\frac{1}{2}} + \gamma_{j+\frac{1}{2}}} \left(f_j \, h^2 + \gamma_{j-\frac{1}{2}} \, u_{j-1}^{(m)} + \gamma_{j+\frac{1}{2}} \, u_{j+1}^{(m)} \right), \tag{3.24}$$

where $j = 1, \ldots, J$ for u^L and $j = J+1, \ldots, N$ for u^R. Observe that such equation is the one obtained by applying Jacobi iteration to Eqs. (3.12) and (3.15).
Let us discretize Eq. (3.19). The time derivative is discretized by forward Euler at the ghost point x_{J+1}, which is the quantity we want to compute. The jump is discretized as in (3.14), so it is second order accurate. We obtain the iteration:

$$u_{J+1}^{L,(m+1)} = u_{J+1}^{L,(m)} + \mu_N \Delta t \left(\gamma_\alpha^R \, \mathcal{L}'_J[u^{R,(m)}](\alpha) - \gamma_\alpha^L \, \mathcal{L}'_{J-1}[u^{L,(m)}](\alpha) - g_N \right). \tag{3.25}$$

Likewise, in Eq. (3.20) we discretize the time derivative in x_J, obtaining:

$$\begin{aligned} u_J^{R,(m+1)} &= u_J^{R,(m)} \\ &+ \mu_D \, \Delta t \left((1-\vartheta) u_J^{L,(m)} + \vartheta u_{J+1})^{L,(m)} - (1-\vartheta) u_J^{R,(m)} + \vartheta u_{J+1}^{R,(m)} + g_D \right). \end{aligned} \tag{3.26}$$

Iterations (3.24), (3.25) and (3.26) constitute the iterative scheme to solve problem (3.1) to second order accuracy.

3.1.3 Choosing constants μ_D and μ_N for transmission conditions

In (3.25) and (3.26) two arbitrary constants μ_D and μ_N appear. Following the same argument as in Sec. 2.2, such constants will be chosen in order to satisfy some stability condition for the equation where they appear. This procedure is not rigorous because it does not take into account the coupling between the equations, and does not consist in a convergence proof. However, in all numerical tests we performed, the conditions we find seem to guarantee convergence.
Constant μ_D is introduced in Eq. (3.20), which is just a relaxation of the jump condition. Then we require:

$$\mu_D \, \Delta t < 1. \tag{3.27}$$

This condition will ensure positivity, and is a factor 2 more stringent than just stability restriction. For practical purpose, we set $\mu_D \, \Delta t = 0.9$. In order to obtain a condition on μ_N, we rewrite Eq. (3.19) as follows (we have supposed for simplicity homogeneous jump $g_N = 0$):

$$\frac{\partial u^L}{\partial t} + \mu_N \, \gamma^L \, \frac{\partial u^L}{\partial x} = \mu_N \, \gamma^R \, \frac{\partial u^R}{\partial x}, \quad t \in (0, \infty). \tag{3.28}$$

This is a simple convection equation with speed $\mu_N \, \gamma^L$. Then a simple CFL condition for convection equation might be

$$\mu_N \Delta t \leq \frac{h}{\gamma^L}.$$

Numerical experiments show that this condition is not enough, especially in the case $\gamma^R / \gamma^L \gg 1$. An explanation of this behavior may be that the right-hand side of (3.28) is not stationary when the convection evolves in time, but it depends on time itself by u^R. An acceptable condition is

$$\mu_N \, \Delta t \leq \frac{h}{\max\left\{\gamma^L, \gamma^R\right\}}. \tag{3.29}$$

For practical purpose we choose $\mu_N \, \Delta t = 0.9 \, h / \max\left\{\gamma^L, \gamma^R\right\}$. Numerical tests show that conditions (3.27) and (3.29) are sufficient for guarantee convergence, but not necessary. A more detailed analysis is in progress.
Notice that $\mu \, \Delta t = O(h^2)$, $\mu_N \, \Delta t = O(h)$, $\mu_D \, \Delta t = O(1)$. Furthermore, only the product of the constants times Δt enters into the conditions, therefore we may imagine that $\Delta t = 1$.

3.2 Multigrid approach

The convergence of the iterative method proposed in Sec. 3.1.2 is usually very slow. To accelerate the convergence we use a multigrid strategy. To make the iteration scheme (3.24)-(3.26) a building block for an efficient multigrid solver, we must be sure that such iteration (*relaxation scheme*) has the *smoothing property*, i.e. that after few steps, the error becomes smooth (not necessarily small). Roughly speaking, the high-frequency components of the error reduce quickly. We do not explain all multigrid features, but just what is different from classical multigrid approach, remanding to the literature for more details (e.g., see [105, 55, 27]). The iteration scheme (3.24)-(3.26) is a Jacobi-like scheme, as mentioned in Sec. 3.1.2. Jacobi scheme is not a good smoother, since high-frequency components of the error reduce slowly. A good smoother is instead the Gauss-Seidel scheme. Then, we use a Gauss-Seidel version of (3.24)-(3.26) as a relaxation scheme, i.e.

$$u_j^{L,(m+1)} = \frac{1}{\gamma_{j-\frac{1}{2}} + \gamma_{j+\frac{1}{2}}} \left(f_j \, h^2 + \gamma_{j-\frac{1}{2}} \, u_{j-1}^{L,(m+1)} + \gamma_{j+\frac{1}{2}} \, u_{j+1}^{L,(m)} \right), \qquad j = 1, \ldots, J \quad (3.30)$$

$$u_{J+1}^{L,(m+1)} = u_{J+1}^{L,(m)} + \mu_N \Delta t \left(\gamma_\alpha^R \, \mathcal{L}'_J[u^{R,(m)}](\alpha) - \gamma_\alpha^L \, \mathcal{L}'_{J-1}[\tilde{u}^L](\alpha) - g_N \right) \quad (3.31)$$

$$\begin{aligned} u_J^{R,(m+1)} &= u_J^{R,(m)} \\ &+ \mu_D \Delta t \left((1-\vartheta) u_J^{L,(m+1)} + \vartheta u_{J+1}^{L,(m+1)} - (1-\vartheta) u_J^{R,(m)} - \vartheta u_{J+1}^{R,(m)} + g_D \right) \end{aligned} \quad (3.32)$$

$$u_j^{R,(m+1)} = \frac{1}{\gamma_{j-\frac{1}{2}} + \gamma_{j+\frac{1}{2}}} \left(f_j \, h^2 + \gamma_{j-\frac{1}{2}} \, u_{j-1}^{R,(m+1)} + \gamma_{j+\frac{1}{2}} \, u_{j+1}^{R,(m)} \right), \qquad j = J+1, \ldots, N \quad (3.33)$$

where in (3.31) we intend $\tilde{u}^L$ such that $\tilde{u}_j^L = u_j^{L,(m+1)}$ for $j < J+1$ and $\tilde{u}_{J+1}^L = u_{J+1}^{L,(m)}$. The unknowns are updated in the same order reported in (3.10).

In order to explain the multigrid approach, we just describe the two-grid correction scheme (TGCS), because all the other schemes, such as V-cycle, W-cycle, F-cycle or Full Multigrid cycle, can be easily derived from it (see [105,

Sections 2.4 and 2.6] for more details). Let us introduce teh following notation. For a grid of spatial step h, we denote:

$$J = \left\lfloor \frac{\alpha}{h} \right\rfloor, \quad \vartheta = \frac{\alpha}{h} - J$$

$$S(\Omega_h) = \big\{ \mathbf{w}_h = (w^L, w^R) \text{ such that } w^L \colon \{x_0, \ldots, x_{J+1}\} \to \mathbb{R},\ w^R \colon \{x_J, \ldots, x_{N+1}\} \to \mathbb{R} \big\}$$

$$\overset{\circ}{S}(\Omega_h) = \big\{ \mathbf{w}_h = (w^L, w^R) \text{ such that } w^L \colon \{x_1, \ldots, x_J\} \to \mathbb{R},\ w^R \colon \{x_{J+1}, \ldots, x_N\} \to \mathbb{R} \big\}$$

$$\mathbf{u}_h = ((u_j^L)_{j=0,\ldots,J+1}, (u_j^R)_{j=J,\ldots,N+1}) \in S(\Omega_h)$$

$$\gamma_h = ((\gamma_j^L)_{j=0,\ldots,J+1}, (\gamma_j^R)_{j=J,\ldots,N+1}) \in S(\Omega_h)$$

$$\mathbf{f}_h \in \overset{\circ}{S}(\Omega_h) \text{ such that } \mathbf{f}_h(x_j) = f_j$$

$$L_h \colon S(\Omega_h) \times S(\Omega_h) \longrightarrow \overset{\circ}{S}(\Omega_h) \text{ such that}$$

$$(L_h(\gamma_h, \mathbf{u}_h))_j = \frac{1}{h^2} \left(\gamma^L_{j-\frac{1}{2}} \left(u_j^L - u_{j-1}^L \right) + \gamma^L_{j+\frac{1}{2}} \left(u_j^L - u_{j+1}^L \right) \right) \text{ if } j \le J$$

$$(L_h(\gamma_h, \mathbf{u}_h))_j = \frac{1}{h^2} \left(\gamma^R_{j-\frac{1}{2}} \left(u_j^R - u_{j-1}^R \right) + \gamma^R_{j+\frac{1}{2}} \left(u_j^R - u_{j+1}^R \right) \right) \text{ if } j \ge J{+}1$$

$$[\,\cdot\,]_h^D : S(\Omega_h) \longrightarrow \mathbb{R} \text{ such that}$$

$$[\mathbf{u}_h]_h^D = \big((1-\vartheta) u_J^R + \vartheta u_{J+1}^R \big) - \big((1-\vartheta) u_J^L + \vartheta u_{J+1}^L \big)$$

$$[\,\cdot\,,\,\cdot\,]_h^N : S(\Omega_h) \times S(\Omega_h) \longrightarrow \mathbb{R} \text{ such that}$$

$$[\gamma_h, \mathbf{u}_h]_h^N = \gamma_\alpha^R \, \mathcal{L}'_J[u^R](\alpha) - \gamma_\alpha^L \, \mathcal{L}'_{J-1}[u^L](\alpha)$$

The linear system (3.11)-(3.16) can be resumed as follows:

$$\begin{aligned} L_h(\gamma_h, \mathbf{u}_h) &= \mathbf{f}_h \\ [\mathbf{u}_h]_h^D &= g_D \\ [\gamma_h, \mathbf{u}_h]_h^N &= g_N \\ u_0^L &= g_0 \\ u_N^R &= g_1. \end{aligned}$$

For simplicity we assume that $N + 1 = 1/h$ is a power of 2. The TGCS consists into the following algorithm:

1. Set initial guess $\mathbf{u}_h = 0$.

2. Relax ν_1 times on the finest grid: for k from 1 to ν_1 do (3.30), (3.31), (3.32).

3. Compute the defects $\mathbf{r}_h \in \overset{\circ}{S}(\Omega_h)$, $\tilde{g}_D, \tilde{g}_N \in \mathbb{R}$:

$$\begin{aligned} \mathbf{r}_h &= \mathbf{f}_h + L_h(\gamma_h, \mathbf{u}_h) \\ \tilde{g}_D &= g_D - [\mathbf{u}_h]_h^D \\ \tilde{g}_N &= g_N - [\gamma_h, \mathbf{u}_h]_h^N \end{aligned}$$

4. Transfer the defect $\mathbf{r}_h$ to a coarser grid with spatial step $2h$ by a suitable *restriction operator*

$$\mathbf{r}_{2h} = I_{2h}^h (\mathbf{r}_h).$$

5. Solve exactly the residual problem on the coarser grid in the unknown $\mathbf{e}_{2h} \in S(\Omega_{2h})$

$$\begin{aligned} L_h(\gamma_{2h}, \mathbf{e}_{2h}) &= \mathbf{r}_{2h} \\ [\mathbf{e}_{2h}]_h^D &= \tilde{g}_D \\ [\gamma_{2h}, \mathbf{u}_{2h}]_h^N &= \tilde{g}_N \\ e_0^L &= 0 \\ e_{(N+1)/2}^R &= 0 \end{aligned}$$

6. Transfer the error to the finest grid by a suitable *interpolation operator*

$$\mathbf{e}_h = I_h^{2h} (\mathbf{e}_{2h}).$$

7. Correct the fine-grid approximation

$$\mathbf{u}_h = \mathbf{u}_h + \mathbf{e}_h.$$

8. Relax ν_2 times on the finest grid: for k from 1 to ν_2 do (3.30), (3.31), (3.32).

To complete the description of TGCS, we have just to explain the steps concerning grid migration (steps 4 and 6).

3.2.1 Transfer grid operators

In this section, we describe the transfer grid operators for vertex-centered grid. Observe that coefficients γ^L and γ^R can be transferred in an exact manner by a simple injection operator.

3.2.1.1 Restriction operator

Since such operator will act on the defect $\mathbf{r}_h = (\mathbf{r}_h^L, \mathbf{r}_h^R) \in \overset{\circ}{S}(\Omega_h)$ (step 4), we perform the restriction from a fine grid to a coarser grid separately for $\mathbf{r}_h^L$ and $\mathbf{r}_h^R$. This is justified by the fact that the defect $\mathbf{r}_h^L$ of the left domain may be very different (after few relaxations) from the defect $\mathbf{r}_h^R$ of the right domain, especially in the case of high jumping coefficient, i.e., $\max\left\{\gamma_\alpha^L/\gamma_\alpha^R, \gamma_\alpha^R/\gamma_\alpha^L\right\} >> 1$. In addition, these defects are very different also from the defects of jumping conditions $\tilde{g}_D$ and $\tilde{g}_N$, because the operators scale with different power of h.
Let us describe the restriction of $\mathbf{r}_h^L$ by the operator $\left(I_{2h}^h\right)^L$ (see Fig. 3.2). Let x_J be the closest grid point to α from the left in the fine grid (see Fig. 3.1). Let x be a grid point of the coarse grid. If $x < x_J$ we will use the standard full-weighting restriction operator (FW):

$$\left(I_{2h}^h\right)^L \mathbf{r}_h^L(x) = \frac{1}{4}\left(\mathbf{r}_h(x-h)^L + 2\,\mathbf{r}_h(x)^L + \mathbf{r}_h(x+h)^L\right), \tag{3.34}$$

while if $x = x_J$ we reduce to an *upwind* linear convex combination from the left direction:

$$\left(I_{2h}^h\right)^L \mathbf{r}_h^L(x) = \omega_1\,\mathbf{r}_h^L(x) + (1-\omega_1)\mathbf{r}_h^L(x-h), \tag{3.35}$$

since in $x+h$ only $\mathbf{r}_h^R$ is defined and not $\mathbf{r}_h^L$. In our tests we found that $\omega_1 = 1/2$ gives better results than $\omega_1 = 3/4$.
The operator $\left(I_{2h}^h\right)^R$ works in a similar manner: let x_{J+1} the closest grid point

to α from the right in the fine grid. If $x > x_{J+1}$ we will use the standard full-weighting restriction operator (FW):

$$\left(I_{2h}^{h}\right)^{R} \mathbf{r}_{h}^{R}(x) = \frac{1}{4}\left(\mathbf{r}_{h}(x-h)^{R} + 2\,\mathbf{r}_{h}(x)^{R} + \mathbf{r}_{h}(x+h)^{R}\right), \tag{3.36}$$

while if $x = x_J$ we reduce to an *Upwind* mean value from the left direction:

$$\left(I_{2h}^{h}\right)^{R} \mathbf{r}_{h}^{R}(x) = \frac{1}{2}\left(\mathbf{r}_{h}^{R}(x) + \mathbf{r}_{h}^{R}(x+h)\right). \tag{3.37}$$

The whole restriction reads

$$I_{2h}^{h}\mathbf{r}_{h} = \left(\left(I_{2h}^{h}\right)^{L} \mathbf{r}_{h}^{L}, \left(I_{2h}^{h}\right)^{R} \mathbf{r}_{h}^{R}\right).$$

In the upper part of Fig. 3.2 is represented the case in which we have to use (3.35) and (3.36). The only other possible case is that we have to use (3.34) and (3.37).

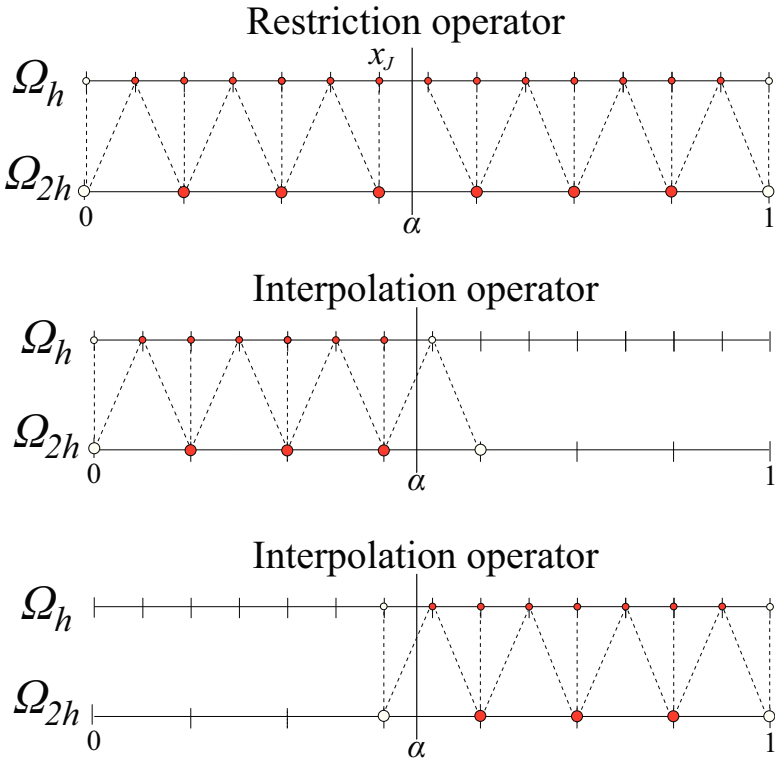


Fig. 3.2: Fine and coarse grid for transfer operators. The dashed lines represent the action of the restriction (top) and the interpolation (middle and bottom) operators.

3.2.1.2 Interpolation operator

Since such operator will act on the correction $\mathbf{e}_{2h} = (\mathbf{e}^L_{2h}, \mathbf{e}^R_{2h}) \in S(\Omega_{2h})$ (step 4), we perform the interpolation from a coarse grid to a finer grid separately for $\mathbf{e}^L_{2h}$ and $\mathbf{e}^R_{2h}$ (see middle and lower part of Fig. 3.2), but always using the standard linear interpolation:

$$\begin{cases} \left(I_h^{2h}\right)^L \mathbf{e}^L_{2h}(x_j) &= \mathbf{e}^L_{2h}(x_j) & \text{if } j \text{ is even} \\ \left(I_h^{2h}\right)^L \mathbf{e}^L_{2h}(x_j) &= \frac{1}{2}\left(\mathbf{e}^L_{2h}(x_{j-1}) + \mathbf{e}^L_{2h}(x_{j+1})\right) & \text{if } j \text{ is odd.} \end{cases}$$

$$\begin{cases} \left(I_h^{2h}\right)^R \mathbf{e}^R_{2h}(x_j) &= \mathbf{e}^R_{2h}(x_j) & \text{if } j \text{ is even} \\ \left(I_h^{2h}\right)^R \mathbf{e}^R_{2h}(x_j) &= \frac{1}{2}\left(\mathbf{e}^R_{2h}(x_{j-1}) + \mathbf{e}^R_{2h}(x_{j+1})\right) & \text{if } j \text{ is odd.} \end{cases}$$

The whole interpolation reads

$$I_h^{2h}\mathbf{e}_{2h} = \left(\left(I_h^{2h}\right)^L \mathbf{e}^L_{2h}, \left(I_h^{2h}\right)^R \mathbf{e}^R_{2h}\right).$$

Remark. 1 (Coarser operator) We observe that the discrete operator L_{2h} on the coarser grid (step 5) is just the operator obtained discretizing directly the continuous operator in the grid with spatial step $2h$, and not the operator obtained by Galerkin condition

$$L_{2h} = I^h_{2h}\, L_h\, I^{2h}_h.$$

The last approach, typical of algebraic multigrid, makes the algebraic problem more expensive from a computational point of view and does not take advantage of the fact that the discrete problem comes from a continuous problem.

Remark. 2 (V-cycle) The V-cycle algorithm is easily obtained from the TGCS recursively, namely applying the same algorithm to solve the residual equation in step 5. To terminate the recursion, an exact solver is used to solve the residual problem when the grid achieves a fixed level of coarsening. We denote by $V(\nu_1, \nu_2)$-cycle the V-cycle performed with ν_1 pre-relaxations and ν_2 post-relaxations.

Remark. 3 (W-cycle) The W-cycle is similar to the V-cycle, with the only difference that the residual problem is solved recursively two times instead of one (in general schemes, δ times, but $\delta > 2$ is considered useless for practical purposes).

3.3 Numerical tests

In this section we confirm numerically the second order accuracy of the discretization of Sec. 3.1.1 and compute the convergence factor ρ of the multigrid approach for several examples, to confirm the independence of ρ from the spatial step h and the magnitude of the jumping coefficient.
Second order accuracy is gained also for first derivative of the solution, as it is shown by the comparison between exact first derivative and the numerical derivative obtained by central difference of the numerical solution.
In all numerical tests, we choose an arbitrary interface $\alpha \in]0,1[$ and an analytical expression of the exact solution $u = (u^L, u^R)$ and of diffusion coefficient $\gamma = (\gamma^L, \gamma^R)$. Then we reconstruct the data f, g_D and g_N, perform the multigrid technique, and compare the numerical solution with the exact solution to compute the order of accuracy by the slope of the best-fit line.
In all our tests we use the following stopping criterion for the $V-$cycle

$$\frac{\left\|\mathbf{u}_h^{(m+1)} - \mathbf{u}_h^{(m)}\right\|_\infty}{\left\|\mathbf{u}_h^{(m+1)}\right\|_\infty} \leq TOL.$$

This will ensure that the actual relative error satisfies

$$\frac{\left\|\mathbf{e}_h^{(m+1)}\right\|_\infty}{\|\mathbf{e}_h\|_\infty} \leq \rho \frac{TOL}{1-\rho}.$$

The tolerance we used is $TOL = 10^{-6}$, which ensures that the error in the solution of the algebraic system is always lower than truncation error. For each example we show a table in which we list the errors, and the value in the third [fifth] column and i-th row of the table indicates the accuracy order, computed as $\log_2(e_{i-1}/e_i)$, where e_i is the L^∞-error of the numerical solution [derivative] indicated in the second [fourth] column and i-th row.
To compute the asymptotic convergence factor, we use the following estimate:

$$\rho = \rho^{(m)} = \frac{\left\|\mathbf{r}_h^{(m)}\right\|_\infty}{\left\|\mathbf{r}_h^{(m-1)}\right\|_\infty},$$

which is reliable for m large. In order to avoid difficulties related to numerical instability due to machine precision, we will always use the homogeneous model problem as a test when we want to compute the asymptotic convergence factor, namely Eq. (3.1) with $f = g_0 = g_1 = 0$ and homogeneous

jump conditions, and perform the multigrid algorithm starting from an initial guess different from zero. Since in this case we are just interested in the convergence factor and not in the numerical solution itself (which approaches zero), a reasonable stop criterion will be

$$\frac{\left|\rho^{(m)} - \rho^{(m-1)}\right|}{\rho^{(m)}} < 10^{-2}.$$

Several tests are performed for each example, based on the different size of the finest and coarsest grids. The finest grid is obtained dividing the domain $[0,1]$ into $N+1$ intervals, while the coarsest grid is obtained dividing the domain into N_c+1 intervals.

3.3.1 Example 1

We choose (see Fig. 3.3)

$$\alpha = 0.343, \quad \begin{cases} u^L &= e^{\sin(5\pi x)} \\ u^R &= e^{x^2} \end{cases}, \quad \begin{cases} \gamma^L &= 3 + \cos(5\pi x) \\ \gamma^R &= 10^9\,(10 + \sin(5\pi x)) \end{cases}.$$

Fig. 3.4 shows the numerical results and the second order slope of the best-fit line for the L^∞-error of the numerical solution and its derivative. Table 3.1 shows the convergence factor for different values of N and N_c.

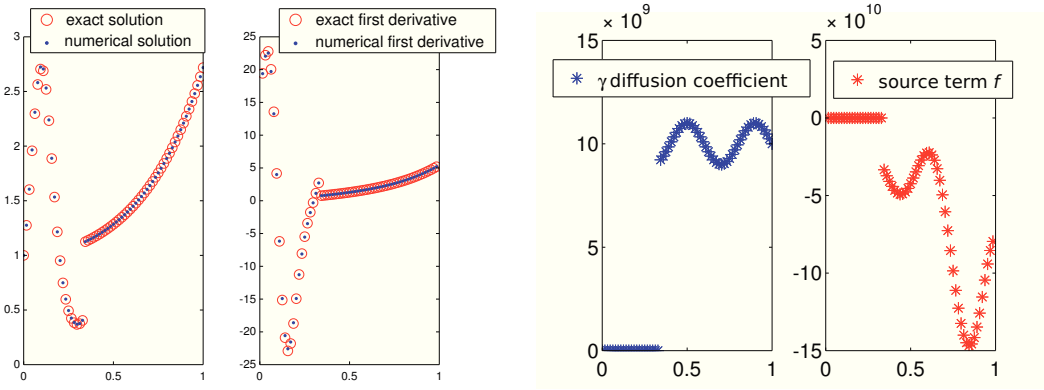
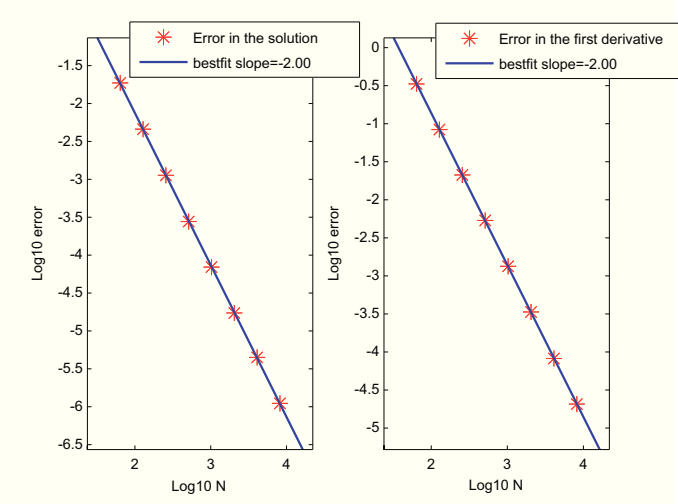


Fig. 3.3: We refer to Ex. 3.3.1. The data are computed for $N = 64$.

3.3.2 Example 2

We choose (see Fig. 3.5)

$$\alpha = 0.743, \quad \begin{cases} u^L &= e^{\sin(5\pi x)} \\ u^R &= e^{x^2} \end{cases}, \quad \begin{cases} \gamma^L &= 3 + \cos(5\pi x) \\ \gamma^R &= 10^9\,(10 + \sin(5\pi x)) \end{cases}.$$



$N+1$	$\|\mathbf{u}-\mathbf{u}_h\|_\infty$	order	$\|\mathbf{u}'-\mathbf{u}'_h\|_\infty$	order
64	$1.87 \cdot 10^{-2}$	-	$3.33 \cdot 10^{-1}$	-
128	$4.59 \cdot 10^{-3}$	2.03	$8.38 \cdot 10^{-2}$	1.99
256	$1.13 \cdot 10^{-3}$	2.02	$2.12 \cdot 10^{-2}$	1.98
512	$2.77 \cdot 10^{-4}$	2.03	$5.35 \cdot 10^{-3}$	1.99
1024	$6.95 \cdot 10^{-5}$	2.00	$1.34 \cdot 10^{-3}$	2.00
2048	$1.73 \cdot 10^{-5}$	2.01	$3.36 \cdot 10^{-4}$	1.99
4096	$4.48 \cdot 10^{-6}$	1.95	$8.21 \cdot 10^{-5}$	2.03
8192	$1.11 \cdot 10^{-6}$	2.01	$2.07 \cdot 10^{-5}$	1.99

Fig. 3.4: We refer to Ex 3.3.1. Left: Representation of the L^∞-error of the numerical solution and its derivative. The slope of the best-fit lines is respectively $s = -2.00$ and $s = -2.00$. Right: List of errors and order of accuracy computed by subsequent errors.

Table 3.1: Measured $V(1,1)$-cycle convergence factor for the numerical test of Ex. 3.3.1. We use $N+2$ number of grid points in the finest grid; N_c+2 number of grid points in the coarsest grid.

N+1 / Nc+1	32	64	128	256	512	1024	2048	4096
16	0.15	0.16	0.15	0.17	0.19	0.15	0.15	0.15
32		0.14	0.12	0.17	0.19	0.15	0.15	0.15
64			0.07	0.16	0.19	0.15	0.15	0.15
128				0.11	0.17	0.15	0.15	0.15

The only difference with respect to the previous example is the value of α. Fig. 3.6 shows the numerical results and the second order slope of the best-fit

line for the L^∞-error of the numerical solution and its derivative. Table 3.2 shows the convergence factor for different values of N and N_c.

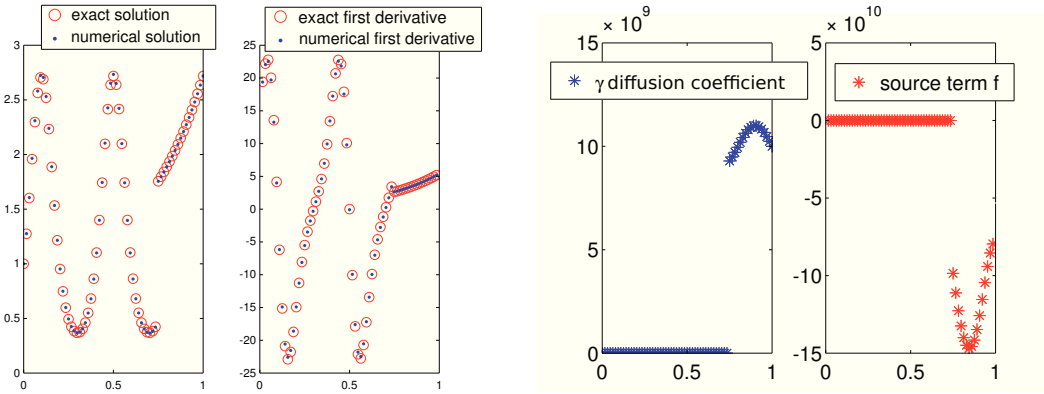


Fig. 3.5: We refer to Ex. 3.3.2. The data are computed for $N = 64$.

Table 3.2: Measured $V(1,1)$-cycle convergence factor for the numerical test of Ex. 3.3.2. We use $N+2$ number of grid points in the finest grid; N_c+2 number of grid points in the coarsest grid.

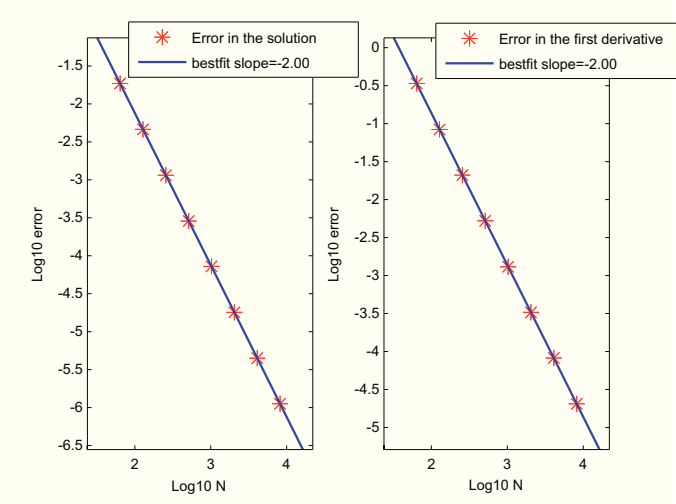
N+1 / Nc+1	32	64	128	256	512	1024	2048	4096
16	0.13	0.13	0.11	0.14	0.15	0.15	0.15	0.15
32		0.13	0.11	0.15	0.15	0.15	0.15	0.15
64			0.11	0.13	0.15	0.15	0.15	0.15
128				0.15	0.15	0.15	0.15	0.15

3.3.3 Example 3

We choose (see Fig. 3.7)

$$\alpha = 0.283 \quad \begin{cases} u^L = e^{\sin(5\pi x)} \\ u^R = e^{x^2} \end{cases}, \quad \begin{cases} \gamma^L = 10^9\,(10 + \sin(5\pi x)) \\ \gamma^R = 3 + \cos(5\pi x) \end{cases}.$$

Fig. 3.8 shows the numerical results and the second order slope of the best-fit line for the L^∞-error of the numerical solution and its derivative. Table 3.3 shows the convergence factor for different values of N and N_c.



$N+1$	$\|\mathbf{u}-\mathbf{u}_h\|_\infty$	order	$\|\mathbf{u}'-\mathbf{u}'_h\|_\infty$	order
64	$1.86 \cdot 10^{-2}$	-	$3.38 \cdot 10^{-1}$	-
128	$4.63 \cdot 10^{-3}$	2.01	$8.38 \cdot 10^{-2}$	2.01
256	$1.15 \cdot 10^{-3}$	2.01	$2.10 \cdot 10^{-2}$	2.00
512	$2.86 \cdot 10^{-4}$	2.01	$5.26 \cdot 10^{-3}$	2.00
1024	$7.24 \cdot 10^{-5}$	1.98	$1.30 \cdot 10^{-3}$	2.01
2048	$1.80 \cdot 10^{-5}$	2.01	$3.28 \cdot 10^{-4}$	1.99
4096	$4.48 \cdot 10^{-6}$	2.01	$8.21 \cdot 10^{-5}$	2.00
8192	$1.12 \cdot 10^{-6}$	2.00	$2.05 \cdot 10^{-5}$	2.00

Fig. 3.6: We refer to Ex 3.3.2. Left: Representation of the L^∞-error of the numerical solution and its derivative. The slope of the best-fit lines is respectively $s = -2.00$ and $s = -2.00$. Right: List of errors and order of accuracy computed by subsequent errors.

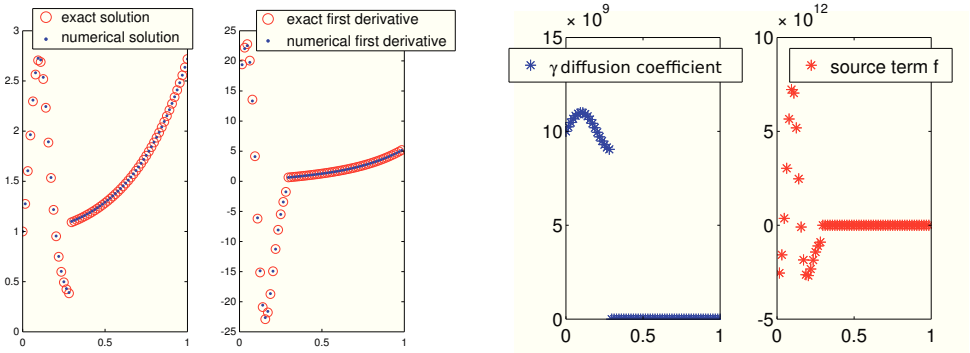
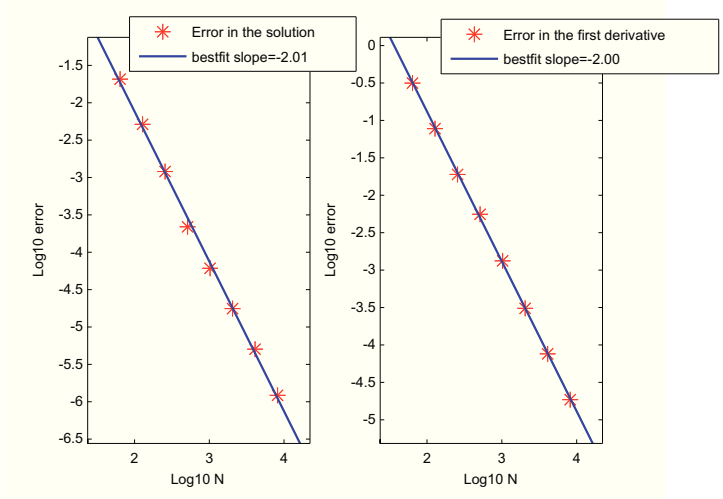


Fig. 3.7: We refer to Ex. 3.3.3. The data are computed for $N = 64$.

3.3.4 Example 4

We choose (see Fig. 3.9)

$$\alpha = 0.813, \quad \begin{cases} u^L &= e^{x^2} \\ u^R &= e^{\sin(5\pi x)} \end{cases}, \quad \begin{cases} \gamma^L &= 10^9\,(10+\sin(5\pi x)) \\ \gamma^R &= 3+\cos(5\pi x) \end{cases}.$$



$N+1$	$\|\mathbf{u}-\mathbf{u}_h\|_\infty$	order	$\|\mathbf{u}'-\mathbf{u}'_h\|_\infty$	order
64	$2.07 \cdot 10^{-2}$	-	$3.15 \cdot 10^{-1}$	-
128	$5.15 \cdot 10^{-3}$	2.01	$7.76 \cdot 10^{-2}$	2.02
256	$1.20 \cdot 10^{-3}$	2.10	$1.90 \cdot 10^{-2}$	2.03
512	$2.19 \cdot 10^{-4}$	2.46	$5.57 \cdot 10^{-3}$	1.77
1024	$6.10 \cdot 10^{-5}$	1.84	$1.33 \cdot 10^{-3}$	2.07
2048	$1.76 \cdot 10^{-5}$	1.79	$3.09 \cdot 10^{-4}$	2.11
4096	$5.05 \cdot 10^{-6}$	1.80	$7.60 \cdot 10^{-5}$	2.02
8192	$1.22 \cdot 10^{-6}$	2.05	$1.86 \cdot 10^{-5}$	2.03

Fig. 3.8: We refer to Ex 3.3.3. Left: Representation of the L^∞-error of the numerical solution and its derivative. The slope of the best-fit lines is respectively $s = -2.01$ and $s = -2.00$. Right: List of errors and order of accuracy computed by subsequent errors.

Table 3.3: Measured $V(1,1)$-cycle convergence factor for the numerical test of Ex. 3.3.3. We use $N+2$ number of grid points in the finest grid; N_c+2 number of grid points in the coarsest grid.

N+1 / Nc+1	32	64	128	256	512	1024	2048	4096
16	0.09	0.10	0.12	0.15	0.15	0.15	0.15	0.15
32		0.09	0.10	0.15	0.15	0.15	0.15	0.15
64			0.12	0.15	0.15	0.15	0.15	0.15
128				0.13	0.15	0.15	0.15	0.15

Fig. 3.10 shows the numerical results and the second order slope of the best-fit line for the L^∞-error of the numerical solution and its derivative. Table 3.4 shows the convergence factor for different values of N and N_c.

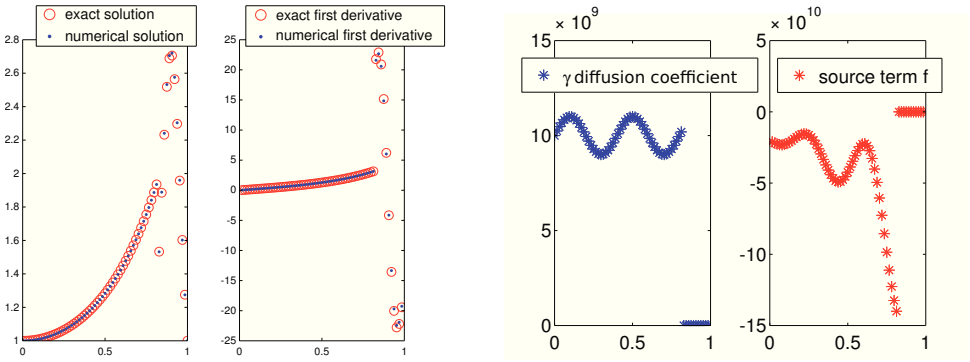
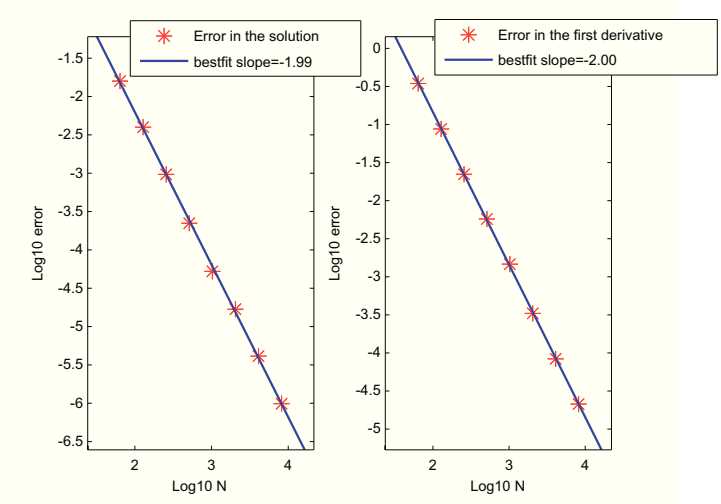


Fig. 3.9: We refer to Ex. 3.3.4. The data are computed for $N = 64$.



$N+1$	$\|\mathbf{u}-\mathbf{u}_h\|_\infty$	order	$\|\mathbf{u}'-\mathbf{u}'_h\|_\infty$	order
64	$1.59 \cdot 10^{-2}$	-	$3.46 \cdot 10^{-1}$	-
128	$3.99 \cdot 10^{-3}$	2.00	$8.74 \cdot 10^{-2}$	1.98
256	$9.66 \cdot 10^{-4}$	2.05	$2.22 \cdot 10^{-2}$	1.98
512	$2.23 \cdot 10^{-4}$	2.12	$5.74 \cdot 10^{-3}$	1.95
1024	$5.25 \cdot 10^{-5}$	2.08	$1.47 \cdot 10^{-3}$	1.97
2048	$1.68 \cdot 10^{-5}$	1.64	$3.31 \cdot 10^{-4}$	2.15
4096	$4.12 \cdot 10^{-6}$	2.03	$8.36 \cdot 10^{-5}$	1.98
8192	$9.87 \cdot 10^{-7}$	2.06	$2.13 \cdot 10^{-5}$	1.97

Fig. 3.10: We refer to Ex 3.3.4. Left: Representation of the L^∞-error of the numerical solution and its derivative. The slope of the best-fit lines is respectively $s = -1.99$ and $s = -2.00$. Right: List of errors and order of accuracy computed by subsequent errors.

3.3.5 Independence of convergence factor from the jump in the coefficient

In this section we show that the convergence factor does not depend on the jump in the coefficient. We choose

$$\alpha = 0.543, \quad \begin{cases} u^L = 0 \\ u^R = 0 \end{cases}, \quad \begin{cases} \gamma^L = 10^p \\ \gamma^R = 1 \end{cases}$$

Table 3.4: Measured $V(1,1)$-cycle convergence factor for the numerical test of Ex. 3.3.4. We use $N+2$ number of grid points in the finest grid; N_c+2 number of grid points in the coarsest grid.

N+1 / Nc+1	32	64	128	256	512	1024	2048	4096
16	0.17	0.12	0.14	0.18	0.17	0.15	0.16	0.15
32		0.11	0.14	0.16	0.15	0.15	0.15	0.15
64			0.06	0.14	0.15	0.15	0.15	0.15
128				0.12	0.15	0.15	0.15	0.15

and start the multigrid process with an initial guess different from zero, in order to compute the asymptotic convergence factor. We list the results in Table 3.5.

Table 3.5: Measured $V(1,1)$ asymptotic convergence factors for a problem with a jumping coefficient of the order 10^p

p	0	1	2	3	4	5
ρ	0.11	0.10	0.11	0.11	0.11	0.10

Remark. (Comparison with Domain Decomposition Method) Domain Decomposition Method (DDM) is another iterative method to solve elliptic problems with discontinuous coefficient, based on solving iteratively the two subproblems

$$\begin{cases} -\dfrac{\partial}{\partial x}\left(\gamma^L \dfrac{\partial u^{L,(m+1)}}{\partial x}\right) = f & \text{in } [0,\alpha[\\ u^{L,(m+1)}(0) = g_0 \\ u^{L,(m+1)}(\alpha) = u^{R,(m)}(\alpha) \end{cases}$$

$$\begin{cases} -\dfrac{\partial}{\partial x}\left(\gamma^R \dfrac{\partial u^{R,(m+1)}}{\partial x}\right) = f & \text{in }]\alpha,1] \\ \gamma^R \dfrac{\partial u^{R,(m+1)}(\alpha)}{\partial x} = \gamma^L \dfrac{\partial u^{L,(m+1)}(\alpha)}{\partial x} \\ u^{R,(m+1)}(1) = g_1 \end{cases}$$

until convergence. A little drawback of this method is that, in order to guarantee the convergence, it must be $\alpha > 0.5$ (see [92, pag. 12]). Our method may be regarded as a DDM, but in place of solving a subproblem to provide the right-hand side for the other subproblem (and so on iteratively), we just perform a relaxation on a subproblem, and with the guess obtained

we build the right-hand side of the other subproblem, as it can be seen in Sec. 3.1.2. With this relaxing strategy, the convergence is always guaranteed, as showed in numerical tests.

Chapter 4

Discontinuous coefficients: 2D case

In this chapter we describe the discretization and the multigrid technique to solve the elliptic problem with discontinuous coefficient in higher dimension. Some general aspects are a straightforward extension from the one dimensional case described in the previous chapter, such as the discretization of inner equations, the ghost-technique for interface transmission conditions, the kind of multigrid cycling. On the other hand, some specific aspects have to be adapted in a suitable way to the high dimensional case, such as the choice of the nine-point stencil for the transmission conditions (that, as we will see, it is not a straightforward extension of the continuous coefficient case), the defect extension far from the interface for both sides, the coarsest grid direct solver.

4.1 Model Problem

Let $D = [-1, 1]^2$ be the computational domain and $\Omega \subset D$ be a domain such that $\partial\Omega \cap \partial D = \emptyset$. Let us consider a partition $\Omega = \Omega^- \cup \Omega^+$, i.e. Ω^+ and Ω^- are two non-empty domains such that $\overset{\circ}{\Omega}{}^- \cap \overset{\circ}{\Omega}{}^+ = \emptyset$ (see Fig. 4.1). Let Γ be the *interface* separating the two subdomains, i.e. $\Gamma = \partial\Omega^- \cap \partial\Omega^+$, while the *boundary* is $\partial\Omega$. Considering the description of Chapter 3, the equivalent problem of the Model Problem 6 (3.5)-(3.8) is the following:

Model problem 7

$$\begin{cases} -\nabla \cdot (\beta^{\pm} \nabla u^{\pm}) &= f^{\pm} \ in \ \Omega^{\pm} \\ [\![u]\!] &= g_D \ on \ \Gamma \\ \left[\!\!\left[\beta \dfrac{\partial u}{\partial n}\right]\!\!\right] &= g_N \ on \ \Gamma \\ u &= g \ on \ \partial\Omega \end{cases} \tag{4.1}$$

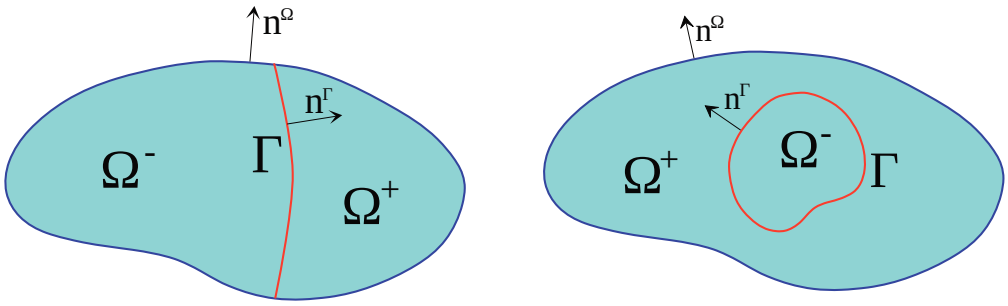


Fig. 4.1: Domain partition $\Omega = \Omega^- \cup \Omega^+$ and the interface Γ separating the two subdomains along which we impose the jump conditions.

With $[\![\cdot]\!]$ we denote the jump across the interface Γ, i.e.

$$[\![\omega]\!](\bar{x},\bar{y}) = \lim_{\Omega^+ \ni (x,y)\to(\bar{x},\bar{y})} \omega^+(x,y) - \lim_{\Omega^- \ni (x,y)\to(\bar{x},\bar{y})} \omega^-(x,y).$$

The domains and the interface are implicitly known by two level set functions ϕ and ϕ^Γ in such a way:

$$\begin{aligned} \Omega &= \{(x,y)\colon \phi(x,y) < 0\}, \\ \Omega^- &= \left\{(x,y)\colon \phi^\Gamma(x,y) < 0 \text{ and } \phi(x,y) < 0\right\}, \\ \Omega^+ &= \left\{(x,y)\colon \phi^\Gamma(x,y) >= 0 \text{ and } \phi(x,y) < 0\right\}, \\ \Gamma &= \left\{(x,y)\colon \phi^\Gamma(x,y) = 0 \text{ and } \phi(x,y) < 0\right\}. \end{aligned} \tag{4.2}$$

The level-set treatment of Sec. 1.1.1 can be repeated here for both level-set functions ϕ and ϕ^Γ. The normal vectors are:

$$\mathbf{n}^\Omega = \frac{\nabla\phi}{|\nabla\phi|}, \quad \mathbf{n}^\Gamma = \frac{\nabla\phi^\Gamma}{|\nabla\phi^\Gamma|}.$$

4.1.1 Notation

Let us consider the notation introduced in Section 1.1 and add some further definitions. Let $\Omega_h^+ = \Omega^+ \cap D_h$ and $\Omega_h^- = \Omega^- \cap D_h$ be the discrete versions of Ω^+ and Ω^- respectively. Let Γ_h^{++} be the set of the *ghost points for* Ω^+, namely the grid points outside Ω^+ and belonging to some five-point stencil centered in a grid point inside Ω^+, i.e.

$$(x,y) \in \Gamma_h^{++} \iff (x,y) \in D_h \backslash \Omega_h^+ \text{ and } \{(x\pm h, y), (x, y\pm h)\} \cap \Omega_h^+ \neq \emptyset.$$

Similarly we define Γ_h^{--} the set of the *ghost points for* Ω^-, and Γ_h the set of the *ghost points for* Ω. Let us define $\Gamma_h^- = \Gamma_h^{--} \backslash \Gamma_h$ and $\Gamma_h^+ = \Gamma_h^{++} \backslash \Gamma_h$. We call $N_i^+ = \left|\Omega_h^+\right|$, $N_g^+ = \left|\Gamma_h^+\right|$, $N_i^- = \left|\Omega_h^-\right|$, $N_g^- = \left|\Gamma_h^-\right|$, $N_g = |\Gamma_h|$, $N_i^{++} = \left|\Gamma_h^{++}\right|$, $N_i^{--} = \left|\Gamma_h^{--}\right|$.

4.2 Discretization of the problem

The final linear system coming from the discretization of the problem will consist in a $(N_i^+ + N_g^+ + N_i^- + N_g^- + N_g) \times (N_i^+ + N_g^+ + N_i^- + N_g^- + N_g)$ linear system. The N_i^- equations coming from the grid points of Ω_h^- are obtained by usual central differences:

$$\beta^-_{i+1/2,j}\left(u^-_{i,j} - u^-_{i+1,j}\right) + \beta^-_{i-1/2,j}\left(u^-_{i,j} - u^-_{i-1,j}\right) + \beta^-_{i,j+1/2}\left(u^-_{i,j} - u^-_{i,j+1}\right) + \beta^-_{i,j-1/2}\left(u^-_{i,j} - u^-_{i,j-1}\right) = f^-_{i,j}\, h^2 \quad (4.3)$$

where $\beta^-_{i\pm1/2,j} = (\beta^-_{i,j} + \beta^-_{i\pm1,j})/2$, $\beta^-_{i,j\pm1/2} = (\beta^-_{i,j} + \beta^-_{i,j\pm1})/2$. Similarly, we write an equation for each grid point of Ω_h^+.

$$\beta^+_{i+1/2,j}\left(u^+_{i,j} - u^+_{i+1,j}\right) + \beta^+_{i-1/2,j}\left(u^+_{i,j} - u^+_{i-1,j}\right) + \beta^+_{i,j+1/2}\left(u^+_{i,j} - u^+_{i,j+1}\right) + \beta^+_{i,j-1/2}\left(u^+_{i,j} - u^+_{i,j-1}\right) = f^+_{i,j}\, h^2 \quad (4.4)$$

Therefore, to close the linear system, we must write an equation for each ghost point $G \in \Gamma_h \cup \Gamma_h^+ \cup \Gamma_h^-$.

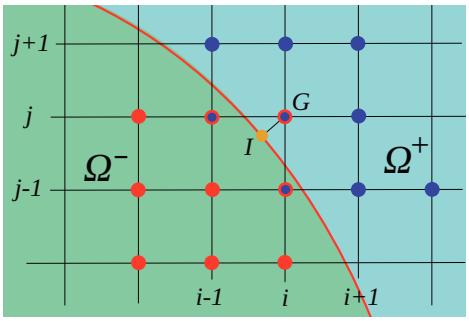


Fig. 4.2: In this figure $G \in \Gamma_h^-$. The blue nine-point stencil is contained in $\Omega_h^+ \cup \Gamma_h^+$ and serves for interpolating $\tilde{u}^+$; the red nine-point stencil is contained in $\Omega_h^- \cup \Gamma_h^-$ and serves for interpolating $\tilde{u}^-$.

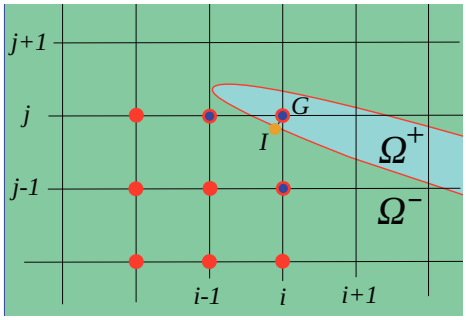


Fig. 4.3: In this figure $G \in \Gamma_h^-$. The nine-point stencil contained in $\Omega_h^+ \cup \Gamma_h^+$ serving to interpolate $\tilde{u}^+$ has been reduced to the blue three-point stencil.

4.2.1 Discretization of interface/boundary conditions

Let $G \in \Gamma_h$. Then, we discretize the boundary condition on Ω, i.e. the fourth equation of (4.1). For such discretization we follow the algorithm described in Sec. 1.2.1.

Let $G \in \Gamma_h^- \cup \Gamma_h^+$, we discretize the interface conditions (second and third equations of (4.1)). Let us explain such a discretization in details.

We compute an approximation of the unit normal vector to Γ in G pointing from Ω^- to Ω^+, that is $\mathbf{n}_G^\Gamma = \left(\nabla\phi^\Gamma / \left|\nabla\phi^\Gamma\right|\right)\Big|_G$, using a second order accurate discretization for $\nabla\phi^\Gamma$, such as central differences in G. Now we can compute the closest interface point to G, that we call I, as:

$$I = G - \mathbf{n}_G \cdot \phi(G). \tag{4.5}$$

The equation of the linear system for the ghost point G is obtained discretizing one of the jump conditions (second and third equation of (4.1)): more precisely, if $G \in \Gamma_h^-$ we use one of the two jump conditions, while if $G \in \Gamma_h^+$ we use the other jump condition. Which jump condition has to be used in each case constitutes a choice, that can be based, for example, on the condition number of the resulting linear system. In fact, it is preferred to use the jump in the flux (third equation in (4.1)) if G is the ghost point for the domain where the coefficient β is greater, in order to obtain a better conditioned linear system. In order to better explain this fact, let us suppose we want to discretize the equation for the ghost point $G \in \Gamma^-$ and that $\beta^- < \beta^+$. If we discretize the jump in the flux (third equation of (4.1)) to construct the equation of the linear system, then the diagonal entry is multiplied by β^-, while some of the off-diagonal entries are multiplied by $\beta^+ > \beta^-$, leading to ill-conditioned system.

Therefore:

- if $\left\{G \in \Gamma_h^+ \text{ and } \beta^+(I) < \beta^-(I)\right\}$ or $\left\{G \in \Gamma_h^- \text{ and } \beta^+(I) > \beta^-(I)\right\}$, then the equation for the ghost point G is obtained from $[\![u]\!](I) = g_D(I)$:

$$\boxed{\tilde{u}_h^+(I) - \tilde{u}_h^-(I) = g_D(I)} \tag{4.6}$$

- otherwise, it is obtained from $\left[\!\left[\beta\dfrac{\partial u}{\partial n}\right]\!\right](I) = g_N(I)$:

$$\boxed{\left(\beta^+\nabla\tilde{u}_h^+ - \beta^-\nabla\tilde{u}_h^-\right)\Big|_I \cdot \left.\left(\frac{\nabla\tilde{\phi}_h}{\left|\nabla\tilde{\phi}_h\right|}\right)\right|_I = g_N(I)} \tag{4.7}$$

where $\tilde{u}_h^+$ (resp. $\tilde{u}_h^-$) is the biquadratic interpolant of u_h^+ (resp. u_h^-) in a suitable nine-point stencil contained in $\Omega_h^+ \cup \Gamma_h^{++}$ (resp. $\Omega_h^- \cup \Gamma_h^{--}$), and $\tilde{\phi}_h$ is the biquadratic interpolant of ϕ in a any nine-point stencil surrounding I. What is left is the choice of the nine-point stencils contained in $\Omega_h^- \cup \Gamma_h^{--}$ and $\Omega_h^+ \cup \Gamma_h^{++}$.

4.2.1.1 Choice of the nine-point stencil

Let us suppose that $G \in \Gamma_h^-$ (the case $G \in \Gamma_h^+$ is treated similarly). The nine-point stencil contained in $\Omega_h^- \cup \Gamma_h^{--}$ is chosen in upwind direction, i.e.:

$$St_9 = \left\{G + h(s_x\, k_1, s_y\, k_2)\colon (k_1, k_2) \in \{0,1,2\}^2\right\}, \tag{4.8}$$

where $s_x = \operatorname{sgn}(x_I - x_G)$ and $s_y = \operatorname{sgn}(y_I - y_G)$, with $G \equiv (x_G, y_G)$ and $I \equiv (x_I, y_I)$. Such stencil can be modified according to the Remark 3 of Sec. 1.2.1.3.

The nine-point stencil contained in $\Omega_h^+ \cup \Gamma_h^{++}$ will be set as follows: if $|x_G - x_I| \geq |y_G - y_I|$ (as in Figs. 4.2 and 4.3) it will be composed by three points of the row $j-1$, three points of the row j, three points of the row $j+1$; while if $|x_G - x_I| < |y_G - y_I|$ it will be composed by three points of the column $i-1$, three points of the column i, three points of the column $i+1$. Let us suppose $|x_G - x_I| \geq |y_G - y_I|$ (the opposite case is treated similarly). Then:

- The three points of the row j are:

  $$(i-1, j)h,\ (i, j)h,\ (i+1, j)h.$$

  Since $(i, j)h \equiv G \in \Omega_h^+$, such three points belong to $\Omega_h^+ \cup \Gamma_h^{++}$.

- The three points of the row $j+1$ are

  $$(i-1, j+1)h,\ (i, j+1)h,\ (i+1, j+1)h$$

  if all of them belong to $\Omega_h^+ \cup \Gamma_h^{++}$, otherwise we choose one of the following two triples:

  $$\{(i-2, j+1)h,\ (i-1, j+1)h,\ (i, j+1)h\} \text{ or}$$
  $$\{(i, j+1)h,\ (i+1, j+1)h,\ (i+2, j+1)h\}$$

  if one of them is contained in $\Omega_h^+ \cup \Gamma_h^{++}$, otherwise we reduce the stencil as described later.

- The three points of the row $j-1$ are

  $$(i-1, j-1)h,\ (i, j-1)h,\ (i+1, j-1)h$$

  if all of them belong to $\Omega_h^+ \cup \Gamma_h^{+}$, otherwise we choose one of the following two triples:

  $$\{(i-2, j-1)h,\ (i-1, j-1)h,\ (i, j-1)h\} \text{ or}$$
  $$\{(i, j-1)h,\ (i+1, j-1)h,\ (i+2, j-1)h\}$$

  if one of them is contained in $\Omega_h^+ \cup \Gamma_h^{++}$. otherwise we reduce the stencil as described later.

If it is not possible to build the nine-point stencil, we revert to a more robust (less accurate) three-point stencil (Fig. 4.3):

$$(i,j)h,\ (i-1,j)h,\ (i,j-1)h.$$

Note that these three points belong to $\Omega_h^+ \cup \Gamma_h^{++}$, since $G \equiv (i,j)h \in \Omega_h^+$.

If $G \in \Gamma_h^-$ the procedure is the same, provided to interchange the subscripts $+$ and $-$.

4.3 Multigrid approach

4.3.1 Relaxation scheme

As in Sec. 3.1.2 for 1D problems, the relaxation scheme is obtained discretizing the following *associate time-dependent problem* in space and time:

$$\begin{aligned}
\frac{\partial u^{\pm}}{\partial t} &= \mu^{\pm}\left(f^{\pm} + \nabla\cdot\left(\beta^{\pm}\nabla u^{\pm}\right)\right) \text{ in } \Omega^{\pm} \\
\frac{\partial u^{s_1}}{\partial t} &= \mu_D\left(g_D - [\![u]\!]\right) \text{ on } \Gamma \\
\frac{\partial u^{s_2}}{\partial t} &= \mu_N\left(g_N - \left[\!\left[\beta\frac{\partial u}{\partial n}\right]\!\right]\right) \text{ on } \Gamma \\
\frac{\partial u}{\partial t} &= \mu_B\left(g - u\right) \text{ on } \partial\Omega
\end{aligned} \tag{4.9}$$

where $s_1, s_2 \in \{-,+\}$ and $s_1 \neq s_2$. The choice of s_1 and s_2 depends on the value of β in order to better precondition the linear system, as explained in Sec. 4.2.1. In details:

$$\begin{aligned}
s_1 = +, \quad s_2 = - \qquad &\text{if } \beta^+ < \beta^-, \\
s_1 = -, \quad s_2 = + \qquad &\text{if } \beta^+ \geq \beta^-.
\end{aligned}$$

Let us describe such a relaxation scheme in details. For inner equations, the relaxation scheme is straightforward. Let us consider, for instance, a grid point $(i,j)h \in \Omega_h^-$. The iterative scheme for such a point is obtained by the first equation of (4.9) and by (4.3):

$$\begin{aligned}
u_{i,j}^{-\,(n+1)} &= u_{i,j}^{-\,(n)} + \mu_{i,j}^- \,\Delta t\, f_{i,j}^- \\
&\quad + \frac{\mu_{i,j}^- \,\Delta t}{h^2}\left(\beta_{i+1/2,j}^-\left(u_{i,j}^{-\,(n)} - u_{i+1,j}^{-\,(n)}\right) + \beta_{i-1/2,j}^-\left(u_{i,j}^{-\,(n)} - u_{i-1,j}^{-\,(n)}\right)\right. \\
&\qquad \left. + \beta_{i,j+1/2}^-\left(u_{i,j}^{-\,(n)} - u_{i,j+1}^{-\,(n)}\right) + \beta_{i,j-1/2}^-\left(u_{i,j}^{-\,(n)} - u_{i,j-1}^{-\,(n)}\right)\right)
\end{aligned} \tag{4.10}$$

where $\mu^-_{i,j}$ is chosen in such a way it becomes a Jacobi-like scheme, i.e.:

$$\mu^-_{i,j}\Delta t = \frac{h^2}{\beta^-_{i-1/2,j} + \beta^-_{i+1/2,j} + \beta^-_{i,j-1/2} + \beta^-_{i,j+1/2}}.$$

If $(i,j)h \in \Omega^-_h$ the iteration scheme is straightforward to obtain, according to replace the subscript $-$ with $+$:

$$\begin{aligned} u^{+\,(n+1)}_{i,j} &= u^{+\,(n)}_{i,j} + \mu^+_{i,j}\,\Delta t\, f^+_{i,j} \\ &+ \frac{\mu^+_{i,j}\,\Delta t}{h^2}\Big(\beta^+_{i+1/2,j}\left(u^+_{i,j} - u^+_{i+1,j}\right) + \beta^+_{i-1/2,j}\left(u^+_{i,j} - u^+_{i-1,j}\right) \\ &\quad + \beta^+_{i,j+1/2}\left(u^+_{i,j} - u^+_{i,j+1}\right) + \beta^+_{i,j-1/2}\left(u^+_{i,j} - u^+_{i,j-1}\right)\Big) \end{aligned} \tag{4.11}$$

Now, let us consider a ghost point $G \in \Gamma_h$. Therefore, the iterative scheme for G is obtained from the fourth equation of (4.9) and it is similar to Eq. (2.16), i.e.:

$$u^{(n+1)}_G = u^{(n)}_G + \mu_B\,\Delta t\left(g(B) - u^{(n)}_h(B)\right), \tag{4.12}$$

where B is the projection point on the boundary $\partial\Omega$ starting from the ghost point G and obtained by the signed distance function ϕ:

$$B = G - \phi(G)\frac{\nabla\phi}{|\nabla\phi|},$$

where $\nabla\phi$ is discretized by central differences in G. Note that the value of u_G indeed refers to u^-_G or u^+_G if respectively $B \in \partial\Omega^-$ or $B \in \partial\Omega^+$.

If $G \in \Gamma^-_h$, the iterative scheme for G is obtained by the second or third equation of (4.9), more precisely, the second equation if $s_1 = -$, the third equation if $s_2 = -$. This choice is in accord with the discretization of the interface conditions described in Sec. 4.2.1. Recalling the choice (4.6) or (4.7), and the (4.5), we summarize as follows:

- if $\beta^+(I) > \beta^-(I)$, then the equation for the ghost point $G \in \Gamma^-_h$ is:

$$u^{-\,(n+1)}_G = u^{-\,(n)}_G + \mu_D\,\Delta t\left(g_D(I) - \left(\tilde{u}^+_h(I) - \tilde{u}^-_h(I)\right)\right) \tag{4.13}$$

- otherwise, it is:

$$u^{-\,(n+1)}_G = u^{-\,(n)}_G + \mu_N\Delta t\left(g_N(I) - \left(\beta^+\nabla\tilde{u}^+_h - \beta^-\nabla\tilde{u}^-_h\right)\big|_I \cdot \left(\frac{\nabla\tilde{\phi}_h}{\left|\nabla\tilde{\phi}_h\right|}\right)\Bigg|_I\right) \tag{4.14}$$

On the contrary, the iterative equation for a ghost point $G \in \Gamma_h^+$ will be set as follows.

- if $\beta^+(I) > \beta^-(I)$, then the equation for the ghost point $G \in \Gamma_h^+$ is:

$$u_G^{+\,(n+1)} = u_G^{+\,(n)} + \mu_N \Delta t \left(g_N(I) - \left(\beta^+ \nabla \tilde{u}_h^+ - \beta^- \nabla \tilde{u}_h^-\right)\Big|_I \cdot \left(\frac{\nabla \tilde{\phi}_h}{\left|\nabla \tilde{\phi}_h\right|} \right)\Bigg|_I \right) \tag{4.15}$$

- otherwise, it is:

$$u_G^{+\,(n+1)} = u_G^{+\,(n)} + \mu_D \, \Delta t \left(g_D(I) - \left(\tilde{u}_h^+(I) - \tilde{u}_h^-(I) \right) \right) \tag{4.16}$$

Up to now, nothing has been said about the sign of the constants μ_D, μ_N and μ_B. Actually, this is a crucial point in order to make the whole iterative process convergent. What we request for stability is, in fact, that the coefficient of the right-hand side with respect to the variable on which we are iterating is positive and less than one. For example, let us consider the iteration (4.13). We are iterating on the variable u_G^- and the coefficient of the right-hand side with respect to this variable is:

$$c_G^- = 1 + \mu_D \, \Delta t \; coeff(\tilde{u}_h^-(I), u_G^-).$$

Then, we impose that $0 < c_G^- < 1$. The condition $c_G^- < 1$ is ensured by $\mu_D < 0$, while the condition $0 < c_G^-$ implies

$$-\mu_D \, \Delta t < \frac{1}{coeff(\tilde{u}_h^-(I), u_G^-)}.$$

This condition must holds for every possible value of $coeff(\tilde{u}_h^-(I), u_G^-)$ (such value depends in fact on the vector $G - I$), then for its maximum value, which is one. Ultimately, the conditions are:

$$\mu_D < 0, \quad -\mu_D \Delta t < 1. \tag{4.17}$$

By the same argument, let us consider the iteration (4.14). We are iterating on the variable u_G^- and the coefficient of the right-hand side with respect to this variable is:

$$c_G^- = 1 + \mu_N \, \Delta t \; c_G^{interp}, \text{ with } c_G^{interp} = coeff \left(\beta^- \nabla \tilde{u}_h^-(I) \cdot \frac{\nabla \tilde{\phi}_h(I)}{\left|\nabla \tilde{\phi}_h(I)\right|}, u_G^- \right).$$

Let us impose $0 < c_G^- < 1$. The condition $c_G^- < 1$ is satisfied if $\mu_N < 0$. In fact, we observe that, since the nine-point stencil for the biquadratic interpolation $\tilde{u}_h$ is chosen in Upwind direction (see (4.8)), therefore the coefficient c_G^{interp} is positive if and only if the normal vector

$$\tilde{\mathbf{n}}_I = \frac{\nabla\tilde{\phi}_h(I)}{\left|\nabla\tilde{\phi}_h(I)\right|}$$

points from Ω^- to Ω^+, i.e. points outside the domain related to the variable we are interpolating (in this case $\tilde{u}_h^-$, and the related domain is therefore Ω^-). Since we have chosen such a direction for the normal vector (i.e. from Ω^- to Ω^+, see Fig. 4.1 and Eq. (4.2)), we have $c_G^{interp} > 0$, which implies $\mu_N < 0$. Condition $0 < c_G^-$ implies $-\mu_N\,\Delta t < \left(c_G^{interp}\right)^{-1}$. As before, since it has to be satisfied for all possible values of c_G^{interp}, the final conditions read:

$$\mu_N < 0, \quad -\frac{\mu_N\,\Delta t}{h} < \frac{2}{3\sqrt{2}\,\beta^-}, \tag{4.18}$$

By the same argument, the conditions for the iterative equation (4.15) are (now we are itearting on the variable u_G^+):

$$\mu_N < 0, \quad -\frac{\mu_N\,\Delta t}{h} < \frac{2}{3\sqrt{2}\,\beta^+}, \tag{4.19}$$

while, for the iteration (4.16) they are:

$$\mu_D > 0, \quad \mu_D\,\Delta t < 1. \tag{4.20}$$

Finally, observe that the condition on μ_B is (see (4.12)):

$$\mu_B > 0, \quad \mu_B\,\Delta t < 1. \tag{4.21}$$

4.3.1.1 Changing of notation

For the clarity, we want to keep a suitable notation such that constants μ_D and μ_N are always positive. To this purpose, we change the associate time-dependent problem (4.9) and the iteration equations of the interface conditions (4.13)-(4.16) as follows:

$$\begin{aligned}
\frac{\partial u^{\pm}}{\partial t} &= \mu^{\pm}\left(f^{\pm} + \nabla\cdot\left(\beta^{\pm}\nabla u^{\pm}\right)\right) \text{ in } \Omega^{\pm} \\
\frac{\partial u^{s_1}}{\partial t} &= s_1\,\mu_D\left(g_D - [\![u]\!]\right) \text{ on } \Gamma \\
\frac{\partial u^{s_2}}{\partial t} &= \mu_N\left(\left[\!\left[\beta\frac{\partial u}{\partial n}\right]\!\right] - g_N\right) \text{ on } \Gamma \\
\frac{\partial u}{\partial t} &= \mu_B\left(g - u\right) \text{ on } \partial\Omega
\end{aligned} \tag{4.22}$$

- if $\beta^+(I) > \beta^-(I)$, then the equation for the ghost point $G \in \Gamma_h^-$ is:

$$u_G^{-\,(n+1)} = u_G^{-\,(n)} - \mu_D\,\Delta t\left(g_D(I) - \left(\tilde{u}_h^+(I) - \tilde{u}_h^-(I)\right)\right) \tag{4.23}$$

- otherwise, it is:

$$u_G^{-\,(n+1)} = u_G^{-\,(n)} - \mu_N\Delta t\left(g_N(I) - \left(\beta^+\nabla\tilde{u}_h^+ - \beta^-\nabla\tilde{u}_h^-\right)\big|_I\cdot\left(\frac{\nabla\tilde{\phi}_h}{\left|\nabla\tilde{\phi}_h\right|}\right)\Bigg|_I\right) \tag{4.24}$$

- if $\beta^+(I) > \beta^-(I)$, then the equation for the ghost point $G \in \Gamma_h^+$ is:

$$u_G^{+\,(n+1)} = u_G^{+\,(n)} - \mu_N\Delta t\left(g_N(I) - \left(\beta^+\nabla\tilde{u}_h^+ - \beta^-\nabla\tilde{u}_h^-\right)\big|_I\cdot\left(\frac{\nabla\tilde{\phi}_h}{\left|\nabla\tilde{\phi}_h\right|}\right)\Bigg|_I\right) \tag{4.25}$$

- otherwise, it is:

$$u_G^{+\,(n+1)} = u_G^{+\,(n)} + \mu_D\,\Delta t\left(g_D(I) - \left(\tilde{u}_h^+(I) - \tilde{u}_h^-(I)\right)\right) \tag{4.26}$$

Observe that we abused of notation: in fact, at the beginning of the right-hand side of the second equation of (4.22), we intend by $s_1 = +1$ if $s_1 = 1$, and $s_1 = -1$ if $s_1 = -$.

4.3.2 Choosing constants μ_B, μ_D and μ_N

The condition on the positive constant μ_B remains the (4.21).

With the new notation introduced in the previous section 4.3.1.1, the conditions on the constants μ_D and μ_N (4.17), (4.18), (4.19) and (4.19) change

accordingly. As we pointed out in Sec. 3.1.3 for 1D problems, also in 2D the conditions (4.18) and (4.19) does not turn out to be sufficient for the convergence, as we proved by numerical experiments. Therefore, also in this case we have to switch to a more stringent condition, which is equivalent to the 1D condition (3.29). The final conditions on the positive constants μ_D and μ_N are then:

$$\mu_D \, \Delta t < 1, \quad \frac{\mu_N \, \Delta t}{h} < \frac{2}{3\sqrt{2} \, \max\{\beta^-, \beta^+\}}.$$

4.3.3 Smoothing property

As we pointed out at the beginning of Sec. 3.2 for 1D problems, since we want the relaxation scheme to be a good smoother, we cannot use a Jacobi-like scheme, such as the one introduced in Sec. 4.3.1. Instead, we must use a relaxation scheme having the smoothing property, such as the Gauss-Seidel scheme or the weighted Jacobi scheme, with the weight $\omega = 4/5$ in 2D. In the following, we revert for simplicity to a Gauss-Seidel relaxation scheme. The smoothing property of the Gauss-Seidel scheme depends on the ordering chosen for the variables. It is well known (see [105]) that the Red-Black Gauss-Seidel (RB-GS) scheme is a better smoother with respect to the Lexicographic Gauss-Seidel (GS-LEX) scheme, but, again for simplicity, we just study the smoothing properties of the GS-LEX scheme, and we compare the convergence factor with the predicted one by the Local Fourier Analysis (see [105] for more details) for Gauss-Seidel scheme. Once we prove (numerically) that the attained convergence factor is the optimal one for the GS-LEX smoother, then we could straightforwardly switch to a more efficient smoother, such as the RB-GS. This is not done in this work, because the only goal is to construct an effective multigrid solver able to gain the optimal convergence factor for inner equations (GS-LEX in this case), independently on the kind of smoother used for such inner equations.

Therefore, we switch from the relaxation scheme described in Sec. 4.3.1 to a Gauss-Seidel version, namely we update the variable on which we are iterating and we use such updated value for the following iterations on the other variables. The only thing is left to choose is the ordering of the updating sweep. As we said before, we use a lexicographic ordering for inner equations, and any ordering for interface and boundary conditions.

In details, we order the grid points according to the following list:

$$\left\{\Gamma_h, \Gamma_h^-, \Gamma_h^+, \Omega_h^-, \Omega_h^+\right\}.$$

The order within any set of grid points of this list is arbitrary, except for grid

points of Ω_h^- and Ω_h^+, where the lexicographic order is used. i.e.:

$$(x', y') \leq (x'', y'') \Longleftrightarrow \begin{cases} x' < x'' \\ \text{ or} \\ x' = x'' \text{ and } y' < y''. \end{cases}$$

By the same argument of Sec. 2.7.2, in order to avoid that the boundary effects degrade the convergence factor, we add some extra-relaxations on two suitable layers surrounding respectively the interface and the boundary. In details, chosen two parameter λ and δ, by one single relaxation on the whole problem we mean the Algorithm 1, where we have introduced two additional sets of grid points:

$$\Omega_h^{(\delta)} = \{P \in \Omega_h \text{ such that } d(P, \partial\Omega) < \delta\},$$
$$\Omega_h^{\pm\,(\delta)} = \left\{P \in \Omega_h^{\pm} \text{ such that } d(P, \Gamma) < \delta\right\}.$$

We experienced that in this case a good choice of the parameters λ and δ is:

$$\lambda = 5, \quad \delta = 5\,h,$$

in comparison with the case of continuous coefficient (2.50).

4.4 Multigrid components

Let us consider the notation of Sec. 4.1.1 and add some further notation. For a grid of spatial step h, we denote:

$S(I_h) = \{\mathbf{w}_h \colon I_h \to \mathbb{R}\}$, for any $I_h \subseteq D_h$,

$\bar{S}(\Omega_h) = S(\Omega_h^{--}) \times S(\Omega_h^{++})$,

$L_h^- \colon S(\Omega_h^{--}) \times S(\Omega_h^{--}) \to S(\Omega_h^-)$ such that

$$L_h^-(\beta_h^-, \mathbf{u}_h^-)_{i,j} = \frac{1}{h^2}\Big(\beta^-_{i+1/2,j}\left(u^-_{i,j} - u^-_{i+1,j}\right) + \beta^-_{i-1/2,j}\left(u^-_{i,j} - u^-_{i-1,j}\right)$$
$$+\beta^-_{i,j+1/2}\left(u^-_{i,j} - u^-_{i,j+1}\right) + \beta^-_{i,j-1/2}\left(u^-_{i,j} - u^-_{i,j-1}\right)\Big) \text{ for any } (i,j)h \in \Omega^-$$

$L_h^+ \colon S(\Omega_h^{++}) \times S(\Omega_h^{++}) \to S(\Omega_h^+)$ such that

$$L_h^+(\beta_h^+, \mathbf{u}_h^+)_{i,j} = \frac{1}{h^2}\Big(\beta^+_{i+1/2,j}\left(u^+_{i,j} - u^+_{i+1,j}\right) + \beta^+_{i-1/2,j}\left(u^+_{i,j} - u^+_{i-1,j}\right)$$
$$+\beta^+_{i,j+1/2}\left(u^+_{i,j} - u^+_{i,j+1}\right) + \beta^+_{i,j-1/2}\left(u^+_{i,j} - u^+_{i,j-1}\right)\Big) \text{ for any } (i,j)h \in \Omega^+$$

Algorithm 1 One single relaxation on the whole problem include some over-relaxations in the vicinity of the interface and boundary.

for $i = 1 \to \lambda$ **do**

  for all $G \in \Gamma_h$ **do**
    perform the iteration equation (4.12);
  end for
  for all $G \in \Gamma_h^-$ **do**
    perform the iteration equation (4.23) or (4.24);
  end for
  for all $G \in \Gamma_h^+$ **do**
    perform the iteration equation (4.25) or (4.26);
  end for
  for all $P \in \Omega_h^{-\,(\delta)}$ **do**
    perform the iteration equation (4.10);
  end for
  for all $P \in \Omega_h^{+\,(\delta)}$ **do**
    perform the iteration equation (4.11);
  end for
end for

for all $G \in \Gamma_h$ **do**
  perform the iteration equation (4.12);
end for
for all $G \in \Gamma_h^-$ **do**
  perform the iteration equation (4.23) or (4.24);
end for
for all $G \in \Gamma_h^+$ **do**
  perform the iteration equation (4.25) or (4.26);
end for
for all $P \in \Omega_h^-$ **do**
  perform the iteration equation (4.10);
end for
for all $P \in \Omega_h^+$ **do**
  perform the iteration equation (4.11);
end for

$L_h \colon \bar{S}(\Omega_h) \times \bar{S}(\Omega_h) \to S(\Omega_h)$ such that

$$L_h(\beta_h, \mathbf{u}_h)(P) = \begin{cases} L_h^-(\beta_h^-, \mathbf{u}_h^-) \text{ if } P \in \Omega_h^- \\ L_h^+(\beta_h^+, \mathbf{u}_h^+) \text{ if } P \in \Omega_h^+ \end{cases}$$
$$\text{where } \beta_h = (\beta_h^-, \beta_h^+), \quad \mathbf{u}_h = (\mathbf{u}_h^-, \mathbf{u}_h^+).$$

$[\cdot,\cdot]_h^- : \bar{S}(\Omega_h) \times \bar{S}(\Omega_h) \to S(\Gamma_h^-)$ such that

$$[\beta_h, \mathbf{u}_h]_h^-(G) = \begin{cases} \tilde{u}_h^+(I) - \tilde{u}_h^-(I) \\ \text{or} \\ \left(\beta^+ \nabla \tilde{u}_h^+ - \beta^- \nabla \tilde{u}_h^-\right)\big|_I \cdot \left(\dfrac{\nabla \tilde{\phi}_h}{\left|\nabla \tilde{\phi}_h\right|} \right)\Bigg|_I \end{cases}$$

for any $G \in \Gamma_h^-$, according to the choice (4.6) or (4.7).

$g_h^+ \in S(\Gamma_h^+)$ such that $g_h^+(G) = g_D(I)$ or $g_N(I)$, for any $G \in \Gamma_h^+$, according to the choice (4.6) or (4.7).

$[\cdot,\cdot]_h^+ : \bar{S}(\Omega_h) \times \bar{S}(\Omega_h) \to S(\Gamma_h^+)$ such that

$$[\beta_h, \mathbf{u}_h]_h^+(G) = \begin{cases} \tilde{u}_h^+(I) - \tilde{u}_h^-(I) \\ \text{or} \\ \left(\beta^+ \nabla \tilde{u}_h^+ - \beta^- \nabla \tilde{u}_h^-\right)\big|_I \cdot \left(\dfrac{\nabla \tilde{\phi}_h}{\left|\nabla \tilde{\phi}_h\right|} \right)\Bigg|_I \end{cases}$$

for any $G \in \Gamma_h^+$, according to the choice (4.6) or (4.7).

$g_h^- \in S(\Gamma_h^-)$ such that $g_h^-(G) = g_D(I)$ or $g_N(I)$, for any $G \in \Gamma_h^-$, according to the choice (4.6) or (4.7).

$\mathcal{B}_h \colon \bar{S}(\Omega_h) \to S(\Gamma_h)$ such that $\mathcal{B}_h(\mathbf{u}_h) = u_h(B)$ according to the Eq. (1.8) .

$g_h \in S(\Gamma_h)$ such that $g_h(B)$ is defined according to the Eq. (1.8).

With this notation, we can write the linear system on the grid with spatial step h in the following compact form:

$$\begin{aligned} L_h(\beta_h, \mathbf{u}_h) &= f_h \\ [\beta_h, \mathbf{u}_h]_h^- &= g_h^- \\ [\beta_h, \mathbf{u}_h]_h^+ &= g_h^+ \\ \mathcal{B}(\mathbf{u}_h) &= g_h \end{aligned} \tag{4.27}$$

Let us suppose we have an exact solver of such linear system (4.27) for a grid with an arbitrary spatial step h:

$$\mathbf{u}_h = \mathcal{S}(\beta_h, f_h, g_h^-, g_h^+, g_h).$$

Now, in order to describe the multigrid technique to solve the linear system (4.27), we just describe the TGCS (Two-Grid Correction Scheme), since any other basic multigrid algorithm (such as V-cycle, W-cycle, Full Multigrid, and so on) can be easily derived from it (see [105] for more details). The TGCS consists into the following algorithm:

- Set initial guess $\mathbf{u}_h = 0$;
- Relax ν_1 times (by the Algorithm 1) on the grid with spatial step h
- Compute the following defects:

$$\begin{aligned}
\mathbf{r}_h^{\Omega^-} &= f_h^- - L_h^-(\beta_h^-, \mathbf{u}_h^-) \\
\mathbf{r}_h^{\Omega^+} &= f_h^+ - L_h^+(\beta_h^+, \mathbf{u}_h^+) \\
\mathbf{r}_h^{\Gamma^-} &= g_h^- - [\beta_h, \mathbf{u}_h]_h^- \\
\mathbf{r}_h^{\Gamma^+} &= g_h^+ - [\beta_h, \mathbf{u}_h]_h^+ \\
\mathbf{r}_h^{\Gamma} &= g_h - \mathcal{B}(\mathbf{u}_h)
\end{aligned}$$

- Extend the defects $\mathbf{r}_h^{\Gamma^-}$, $\mathbf{r}_h^{\Gamma^+}$ and $\mathbf{r}_h^{\Gamma}$ using the extension operator defined in (2.48):

$$\begin{aligned}
\mathbf{r}_h^{\Gamma^-,ext} &= \mathcal{E}[\Gamma_h^-; -\phi_h^{\Gamma}](\mathbf{r}_h^{\Gamma^-}), \\
\mathbf{r}_h^{\Gamma^+,ext} &= \mathcal{E}[\Gamma_h^+; \phi_h^{\Gamma}](\mathbf{r}_h^{\Gamma^+}), \\
\mathbf{r}_h^{\Gamma_h,ext} &= \mathcal{E}[\Gamma_h; \phi_h](\mathbf{r}_h^{\Gamma}).
\end{aligned}$$

- Transfer these defects to a coarser grid with spatial step $2h$ by the *restriction operator* defined in (2.44):

$$\begin{aligned}
\mathbf{r}_{2h}^{\Omega^-} &= I_{2h}^h \left(\mathbf{r}_h^{\Omega^-}\right) \\
\mathbf{r}_{2h}^{\Omega^+} &= I_{2h}^h \left(\mathbf{r}_h^{\Omega^+}\right) \\
\mathbf{r}_{2h}^{\Gamma^-} &= I_{2h}^h \left(\mathbf{r}_h^{\Gamma^-,ext}\right) \\
\mathbf{r}_{2h}^{\Gamma^+} &= I_{2h}^h \left(\mathbf{r}_h^{\Gamma^+,ext}\right) \\
\mathbf{r}_{2h}^{\Gamma} &= I_{2h}^h \left(\mathbf{r}_h^{\Gamma,ext}\right)
\end{aligned}$$

- Solve exactly the residual problem in the coarser grid

$$\mathbf{e}_{2h} = \mathcal{S}(\beta_{2h}, \mathbf{r}_{2h}^{\Omega}, \mathbf{r}_{2h}^{\Gamma^-}, \mathbf{r}_{2h}^{\Gamma^+}, \mathbf{r}_{2h}^{\Gamma})$$

where $\mathbf{r}_{2h}^{\Omega} = (\mathbf{r}_{2h}^{\Omega^-}, \mathbf{r}_{2h}^{\Omega^+})$.

- Transfer the error to the finer grid by the interpolation operator (2.49):

$$\mathbf{e}_h = I_h^{2h}(\mathbf{e}_{2h})$$

- Correct the fine-grid approximation

$$\mathbf{u}_h := \mathbf{u}_h + \mathbf{e}_h$$

- Relax ν_2 times (by the Algorithm 1) on the grid with spatial step h.

4.5 Numerical tests

Numerical tests have been performed in the simpler case $\partial\Omega^- \cap \partial\Omega = \emptyset$ (see right side of Fig. 4.1).

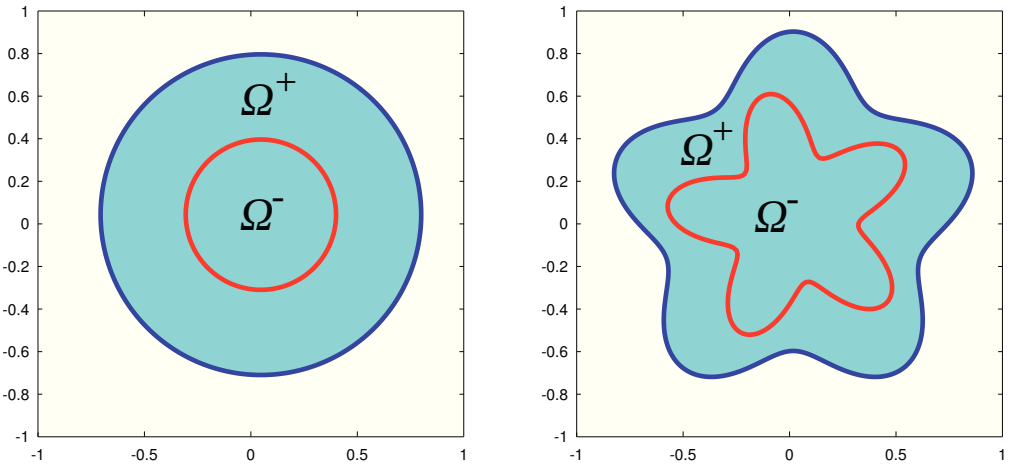


Fig. 4.4: Domains Ω^- and Ω^+ of the Example 4.5.1 (left) and of the Example 4.5.2 (right).

4.5.1 Example 1: circular domains

Let us consider the following data:

$$\phi^\Gamma(x,y) = \sqrt{(x-x0)^2+(y-y0)^2} - R_1,$$

$$\phi(x,y) = \sqrt{(x-x0)^2+(y-y0)^2} - R_2,$$

$$u^- = \sin(4\pi x)\cos(6\pi y), \quad u^+ = \cos(2\pi x)\sin(3\pi y),$$

$$\beta^- = 10^6 + 10^5\sin(\pi x)\cos(3\pi y), \quad \beta^+ = 1 + 0.5\sin(2\pi x)\cos(4\pi y). \quad (4.28)$$

or

$$\beta^- = 1 + 0.5\sin(2\pi x)\cos(4\pi y), \quad \beta^+ = 10^6 + 10^5\sin(\pi x)\cos(3\pi y). \quad (4.29)$$

We choose $x_0 = 30\sqrt{2}$, $y_0 = 40\sqrt{3}$, $R_1 = 0.353$ and $R_2 = 0.753$. The domain is represented in Fig. 4.4 (left side). We performed one test with (4.28) and one test with (4.29). We list in Tables 4.1 and 4.2 the errors of the solution and its gradient in the L^1 and L^∞ norms, while Fig. 4.5 shows the related bestfit lines.

Table 4.1: Example 4.5.1. Accuracy order in the solution (top) and in the gradient (bottom) for the case (4.28).

No. of points	L^1 error of u	order	L^∞ error of u	order
32×32	$8.34 \cdot 10^3$	-	$7.70 \cdot 10^4$	-
64×64	$2.07 \cdot 10^3$	2.01	$1.85 \cdot 10^4$	2.06
128×128	$5.79 \cdot 10^2$	1.84	$5.10 \cdot 10^3$	1.86
256×256	$1.46 \cdot 10^2$	1.99	$1.28 \cdot 10^3$	2.00
No. of points	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
32×32	$1.46 \cdot 10^5$	-	$3.90 \cdot 10^5$	-
64×64	$3.49 \cdot 10^4$	2.06	$1.06 \cdot 10^5$	1.88
128×128	$9.56 \cdot 10^3$	1.87	$2.96 \cdot 10^4$	1.84
256×256	$2.39 \cdot 10^3$	2.00	$7.45 \cdot 10^3$	1.99

Table 4.2: Example 4.5.1. Accuracy order in the solution (top) and in the gradient (bottom) for the case (4.29).

No. of points	L^1 error of u	order	L^∞ error of u	order
32×32	$4.40 \cdot 10^{-3}$	-	$1.22 \cdot 10^{-1}$	-
64×64	$1.02 \cdot 10^{-3}$	2.11	$2.93 \cdot 10^{-2}$	2.06
128×128	$3.29 \cdot 10^{-4}$	1.64	$7.61 \cdot 10^{-3}$	1.95
256×256	$8.24 \cdot 10^{-5}$	2.00	$2.14 \cdot 10^{-3}$	1.83
No. of points	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
32×32	$3.93 \cdot 10^{-1}$	-	$3.52 \cdot 10^0$	-
64×64	$9.95 \cdot 10^{-2}$	1.98	$9.71 \cdot 10^{-1}$	1.86
128×128	$2.63 \cdot 10^{-2}$	1.92	$2.80 \cdot 10^{-1}$	1.80
256×256	$6.60 \cdot 10^{-3}$	1.99	$7.32 \cdot 10^{-2}$	1.93

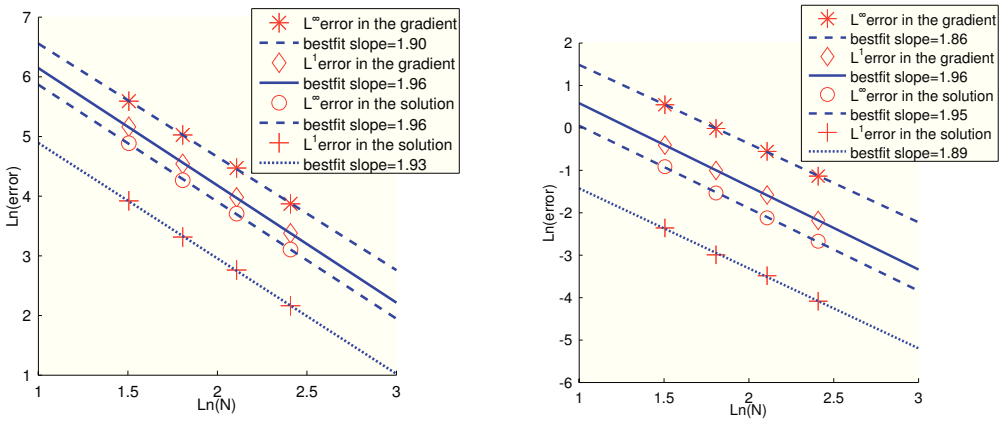


Fig. 4.5: Example 4.5.1. Bestfit lines of the errors in the solution and in the gradient (Tables 4.1 and 4.2) in both the L^1 and L^∞ norms. Left: β^- and β^+ are given by (4.28); Right: β^- and β^+ are given by (4.29).

4.5.2 Example 2: flower-shaped domains

Let us consider the general flower-shaped interface with parametric equations:

$$X(\vartheta) = r(\vartheta)\cos(\vartheta) + x_0,$$
$$Y(\vartheta) = r(\vartheta)\sin(\vartheta) + y_0,$$

with $\vartheta \in [0, 2\pi]$ and $r(\vartheta) = r_0 + r_1 \sin(\omega\vartheta)$. Let us consider $\omega = 5$. The level-set representation of this interface is:

$$flower(r_0, r_1, x_0, y_0; x, y) = r - r_0 - r_1 \frac{(y-y_0)^5 + 5(x-x_0)^4(y-y_0) - 10(x-x_0)^2(y-y_0)^3}{r^5}.$$

where $r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$. Let us choose the following data:

$$\phi^\Gamma(x,y) = flower(0.45, 1/7, 0.01\sqrt{3}, 0.02\sqrt{2}; x, y),$$
$$\phi(x,y) = flower(0.75, 1/8, 0.01\sqrt{3}, 0.02\sqrt{2}; x, y),$$
$$u^- = \sin(4\pi x)\cos(6\pi y), \quad u^+ = \cos(2\pi x)\sin(3\pi y),$$
$$\beta^- = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y), \quad \beta^- = 1 + 0.5\sin(2\pi x)\cos(4\pi y). \quad (4.30)$$

or

$$\beta^- = 1 + 0.5\sin(2\pi x)\cos(4\pi y), \quad \beta^- = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y). \quad (4.31)$$

The domain is represented in Fig. 4.4 (right side). We performed one test with (4.30) and one test with (4.31). We list in Tables 4.3 and 4.4 the errors of the solution and its gradient in the L^1 and L^∞ norms, while Fig. 4.6 shows the related bestfit lines.

Table 4.3: Example 4.5.2. Accuracy order in the solution (top) and in the gradient (bottom) for the case (4.30).

No. of points	L^1 error of u	order	L^∞ error of u	order
32×32	$1.41 \cdot 10^{4}$	-	$1.03 \cdot 10^{5}$	-
64×64	$2.54 \cdot 10^{3}$	2.48	$1.78 \cdot 10^{4}$	2.53
128×128	$8.07 \cdot 10^{2}$	1.65	$5.58 \cdot 10^{3}$	1.67
256×256	$1.57 \cdot 10^{2}$	2.36	$1.07 \cdot 10^{3}$	2.38
No. of points	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
32×32	$2.49 \cdot 10^{5}$	-	$8.38 \cdot 10^{5}$	-
64×64	$4.34 \cdot 10^{4}$	2.52	$1.57 \cdot 10^{5}$	2.42
128×128	$1.35 \cdot 10^{4}$	1.68	$4.94 \cdot 10^{4}$	1.67
256×256	$2.60 \cdot 10^{3}$	2.38	$9.65 \cdot 10^{3}$	2.36

Table 4.4: Example 4.5.2. Accuracy order in the solution (top) and in the gradient (bottom) for the case (4.31).

No. of points	L^1 error of u	order	L^∞ error of u	order
32×32	$6.63 \cdot 10^{-3}$	-	$2.51 \cdot 10^{-1}$	-
64×64	$2.49 \cdot 10^{-3}$	1.41	$8.18 \cdot 10^{-2}$	1.62
128×128	$5.02 \cdot 10^{-4}$	2.31	$1.66 \cdot 10^{-2}$	2.30
256×256	$1.28 \cdot 10^{-4}$	1.98	$4.03 \cdot 10^{-3}$	2.04
No. of points	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
32×32	$6.70 \cdot 10^{-1}$	-	$4.30 \cdot 10^{0}$	-
64×64	$1.91 \cdot 10^{-1}$	1.81	$1.19 \cdot 10^{0}$	1.86
128×128	$4.64 \cdot 10^{-2}$	2.04	$3.31 \cdot 10^{-1}$	1.84
256×256	$1.18 \cdot 10^{-2}$	1.97	$1.23 \cdot 10^{-1}$	1.43

4.5.3 Example 3: Convergence factor of the multigrid

In this example we show that the asymptotic convergence factor does not depend on the jump of the coefficient and on the size of the problem. The study we want to carry out about the convergence factor concerns how it is close to the optimal one (see Table 2.2), i.e. the convergence factor predicted

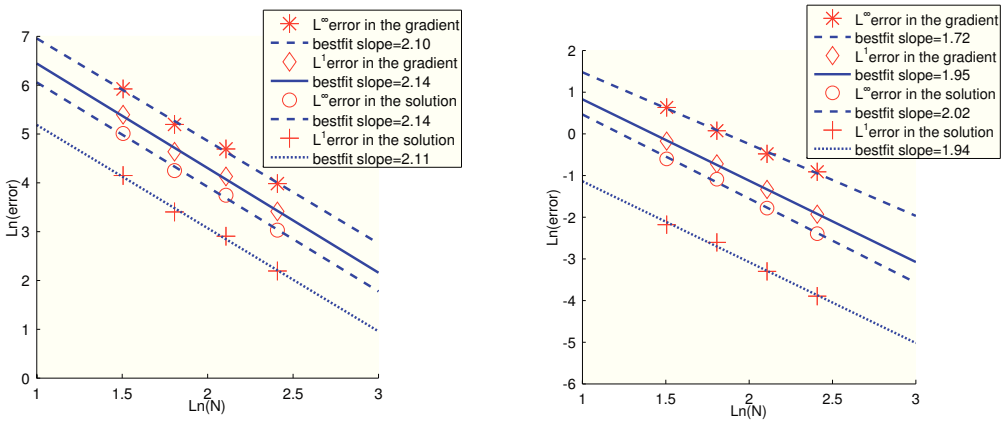


Fig. 4.6: Example 4.5.2. Bestfit lines of the errors in the solution and in the gradient (Tables 4.3 and 4.4) in both the L^1 and L^∞ norms. Left: β^- and β^+ are given by (4.30); Right: β^- and β^+ are given by (4.30).

by the Local Fourier Analysis for inner equations. As we pointed out in Sec. 4.3.3, we know that more efficient smoothers than LEX-GS for inner equations exist (such as RB-GS), but the goal of this work is to show that the optimal convergence factor is attained, regardless on the smoother for inner equations. The same argument holds for the multigrid algorithm: even if the Full Multigrid is more efficient, we limit ourselves to study the convergence factor for the W-cycle algorithm, in order to compare results with the well-known Table 2.2.

However, we experienced that the convergence factor is close to the optimal one in the first few cycles of the entire algorithm (say the first ten), while the asymptotic convergence factor slightly degrades. We believe that some more effort can be done in order to lead the asymptotic convergence factor to be the optimal one, and such a work will be done in near future.

Let us recall that we estimate the asymptotic convergence factor as:

$$\rho = \lim_{m\to\infty} \rho^{(m)} = \lim_{m\to\infty} \frac{\left\| \mathbf{r}_h^{(m)} \right\|_\infty}{\left\| \mathbf{r}_h^{(m-1)} \right\|_\infty},$$

where $\mathbf{r}_h = \left(\mathbf{r}_h^{\Omega^-}, \mathbf{r}_h^{\Omega^+}, \mathbf{r}_h^{\Gamma^-}, \mathbf{r}_h^{\Omega^+}, \mathbf{r}_h^{\Gamma}\right)$. In practice, we compute $\rho^{(m)}$ until the following stop criterion is satisfied:

$$\frac{\left|\rho^{(m)} - \rho^{(m-1)}\right|}{\rho^{(m)}} < 10^{-3}. \tag{4.32}$$

We compare this convergence factor with the averaged convergence factor of

the first ten W-cycle iterations, computed as follows:

$$\bar{\rho} = \sqrt[9]{\prod_{m=2}^{10} \rho^{(m)}}. \tag{4.33}$$

We perform the homogeneous model problem, namely the Model Problem 7 (4.1) with $f^{\pm} = g_D = g_N = g = 0$ (starting with an initial guess different from zero), in order to avoid difficulties related to numerical instability related to the machine precision.

The numerical tests have been performed by a W-cycle algorithm with $\nu_1 = 2$ pre-smoothing and $\nu_2 = 1$ post-smoothing (therefore $\nu = 3$ in Table 2.2), and with the coarsest grid having 16×16 grid points. Tables 4.5 and 4.6 show the estimated convergence factors for different numbers of grid points and jump in the coefficient. We performed such tests in the more complicated geometry of Example 4.5.2. We choose the following coefficients:

$$\beta^- = 10^p, \quad \beta^+ = 1.$$

Table 4.5: Example 4.5.3. Asymptotic convergence factor, computed with the stop criterion (4.32) ($\nu = \nu_1 + \nu_2 = 3$).

p / N^2	-9	-7	-5	-3	-1
32^2	0.0875	0.0875	0.0875	0.0872	0.1019
64^2	0.1723	0.1723	0.1722	0.1553	0.1103
128^2	0.1616	0.1616	0.1616	0.1616	0.1616
p / N^2	1	3	5	7	9
32^2	0.2302	0.2411	0.2411	0.2411	0.2411
64^2	0.2176	0.2442	0.2445	0.2445	0.2445
128^2	0.1617	0.1618	0.1947	0.1947	0.1947

Table 4.6: Example 4.5.3. Average convergence factor for the first ten W-cycle iterations, computed by the formula (4.33) ($\nu = \nu_1 + \nu_2 = 3$).

p / N^2	-9	-7	-5	-3	-1
32^2	0.0776	0.0776	0.0776	0.0773	0.0486
64^2	0.0930	0.0930	0.0930	0.0930	0.1107
128^2	0.1544	0.1544	0.1544	0.1544	0.1544
p / N^2	1	3	5	7	9
32^2	0.1563	0.1586	0.1585	0.1585	0.1585
64^2	0.0931	0.1027	0.1029	0.1029	0.1029
128^2	0.1543	0.1543	0.1544	0.1544	0.1544

Chapter 5

Grid adaptivity

In this chapter we describe how to use adaptive grid to effectively reduce the computational effort, without losing the level of accuracy. In practice, it is well known that in elliptic problems the error concentrates on the vicinity of the interface/boundary. Therefore, a good strategy to reduce the computational work maintaining the accuracy is to refine the grid close to the interface/boundary and leaving a coarser grid far from it. To this purpose we introduce the adaptive grid.

In order to straightforwardly implement the treatment of boundary/interface conditions proposed in Sections 1.2 and 4.2, we aim to obtain a uniform grid in a suitable layer surrounding the boundary/interface, more precisely, in a layer with width $2\,h$.

5.1 Domain discretization: quadtree data structure

n this section we describe how we obtain the adaptive grid by the quadtree structure. Let us recall the domain discretization described in [79, 32]. The domain $D = [-1, 1]^2$ is discretized into squares, and a quadtree data structure is used to represent this discretization. As depicted in Figure 5.1, the entire domain is originally associated with the root of the tree which has level zero by definition. Then it is split into four children cells of equal size which have level one. This discretization proceeds recursively, i.e. each cell can be in turn split into four children which have one more level than their parent cell. A cell with no children is called a leaf. Two cells are called neighbors if they share a common face or part of a face. The discretization in 3D is similar, except that each cube is split into eight cubes of the same size (octree data

structure). The interested reader is referred to [79, 32, 96, 97] for more on quadtree and octree data structures.

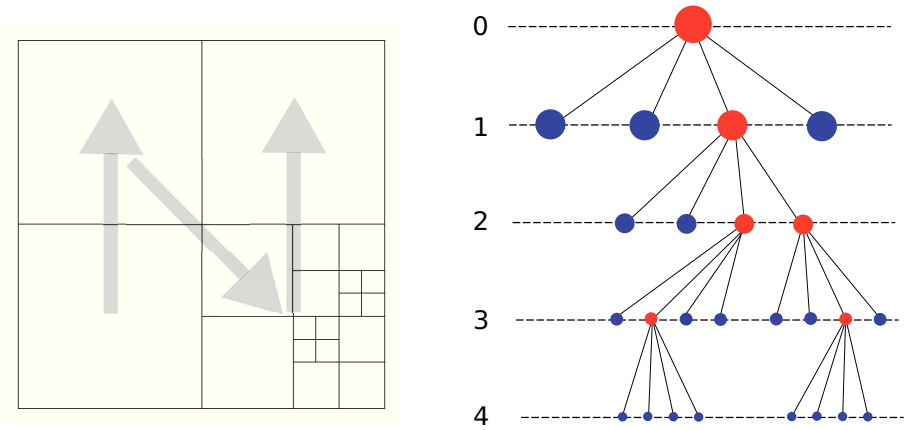


Fig. 5.1: Discretization of a two dimensional domain (left) and its quadtree representation (right). The entire domain corresponds to the root of the tree (level 0), and each cell is subdivided into four children, in the order of lower-left, upper-left, lower-right, upper-right.

Several criteria may be adopted to decide where the grid has to be refined. Here we assume that the solution may vary more dramatically near the boundary, which is likely to be the case, for example, if the boundary itself has a complex shape. Because of the possible irregularity of Γ, we assume for the moment that the refinement is needed only near the boundary. For this reason, we discretize the computational domain in such a way that the cell size is proportional to the absolute value of the signed distance function ϕ, i.e., the distance to the boundary. We split a cell c if (see the left side of Fig. 5.2):

$$\min_{v \in V} |\phi(v)| < \frac{diag}{2}, \tag{5.1}$$

where v is a vertex of cell c, V is the set of all vertices of cell c and $diag$ is the diagonal length of the cell. Finally, the finest resolution is obtained at cells cut by the boundary. Condition (5.1) is the refinement criterion used in [79, 32].

We note that using the refinement criterion (5.1) leads us to an infinite recursive algorithm, since we always must refine cells cut by the boundary, obtaining smaller and smaller cells close to the boundary. Then, a stopping criterion is required, in order to obtain a uniform (finer) grid along the boundary. On the other hand, we want to avoid big cells, even far from the boundary. Then, we start the refinement algorithm from an established uniform (coarser) grid. In details, we proceed as follows: we choose

two levels of refinement: *min_level* and *max_level*. Accordingly, we define $h_{\min} = 2/(2^{max\_level})$ and $h_{\max} = 2/(2^{max\_level})$. Then we split a cell if its side h satisfies $h > h_{\max}$ or if (5.1) is satisfied and $h > h_{\min}$.

We say that a grid is *graded* if the difference between two adjacent cells is at most one. By definition, a quadtree is said to be graded if it corresponds to a graded grid. In this paper, we sample the solution at the cell vertices and pose no limitation on the level difference between two adjacent cells, allowing for fully adaptive non-graded grids, which allow more flexibility and efficiency in the adaptation.

In order to straightforwardly use the treatment of boundary/interface conditions proposed in Sections 1.2 and 4.2, we want to obtain a uniform grid in a suitable layer surrounding the boundary/interface, more precisely, in a layer with width $2h$. To this purpose, we modify the refinement criterion (5.1) into the following (see the right side of Fig. 5.2):

$$\min_{v \in V} \{|\phi(v)|, |\phi(v) - h|, |\phi(v) + h|\} < \frac{diag}{2}. \tag{5.2}$$

Remark. For practical purpose, in numerical tests we often deal with a level-set function ϕ_0 in place of a signed distance function ϕ. In such cases, we use ϕ_0 instead of ϕ in the refinement criteria (5.1) and (5.2) and multiply the right-hand side by the Lipschitz constant of ϕ_0.

5.1.1 Finite difference discretization

We follow the technique of [79, 32, 78]. Let us consider the finite difference discretization for a T-junction node without a direct right neighboring node as depicted in Fig. 5.3. Discretization at T-junction nodes without another direct neighboring node (left, top or bottom) can be derived in the same manner. The equation for a node P obtained from the discretization of the first equation of (4.1) is:

$$\begin{aligned} \frac{u_P - u_A}{d_{PA}} \left(\frac{\beta_P + \beta_A}{2} \right) + \frac{u_P - u_B}{d_{PB}} \left(\frac{\beta_P + \beta_B}{2} \right) \\ + \frac{u_P - u_C}{d_{PC}} \left(\frac{\beta_P + \beta_C}{2} \right) + \frac{u_P - u_D}{d_{PD}} \left(\frac{\beta_P + \beta_D}{2} \right) = f_P \end{aligned} \tag{5.3}$$

where d_{HK} is the distance between points H and K. All values in (5.3) are defined, except u_A and β_A, since A is not a grid point. We obtain such values from the Taylor expansion formula

$$u(A) = \tilde{u}_A - \frac{d_{DE} d_{DF}}{2} u_{yy}(P) + O(h^3),$$

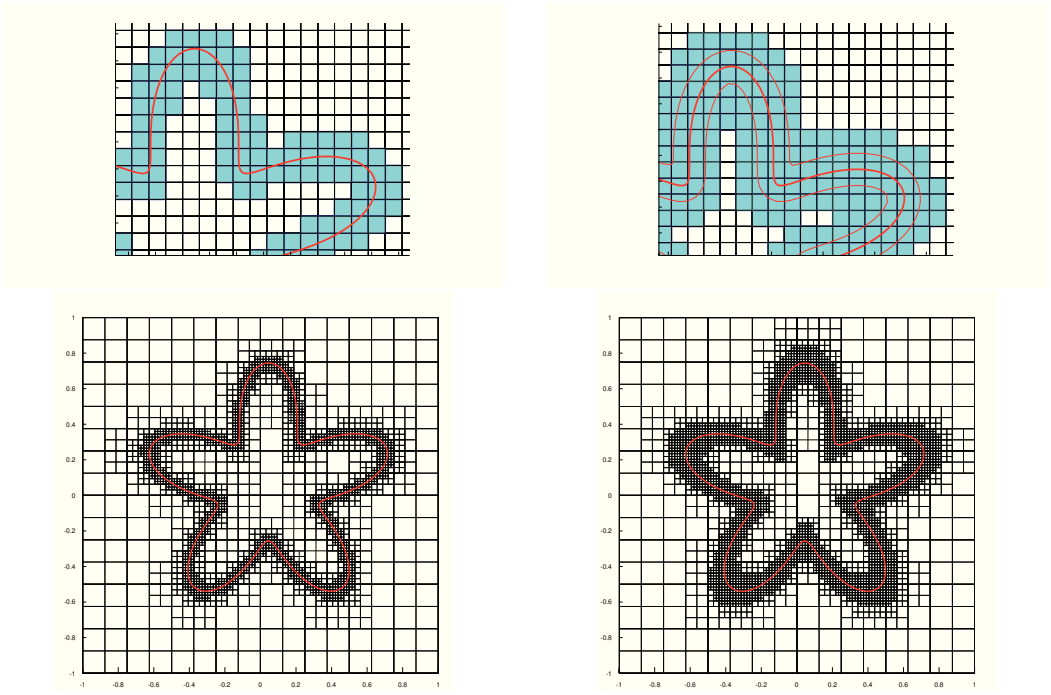


Fig. 5.2: Adaptive grid refinement for the flower-shaped domain (see numerical test 5.2.4). Top: cells to be refined according to the refinement criterion (5.1) (left) and (5.2) (right). Using (5.2) is equivalent to using (5.1) for three level-sets: $\phi = 0$ (red bolder central line) and $\phi = \pm h$ (red smaller lines). Bottom: adaptive grid computed with $max\_level = 7$, $min\_level = 4$ and with the refinement criterion (5.1) (left) and (5.2) (right). Adopting (5.2) allows us to obtain a uniform grid along the boundary/interface in a wider band.

where $\tilde{u}_A$ denotes the linear interpolation in D between nodes E and F. Because only a first order accurate formula is needed for the second derivative, we obtain the following third order interpolation formula [78]:

$$\begin{aligned} u_A &= \tilde{u}_A - \frac{d_{DE}d_{DF}}{d_{CD}}\left(\frac{u_C - u_P}{d_{PC}} + \frac{u_D - u_P}{d_{PD}}\right) \\ &= \frac{u_E d_{DF} + u_F d_{DE}}{d_{EF}} - \frac{d_{DE}d_{DF}}{d_{CD}}\left(\frac{u_C - u_P}{d_{PC}} + \frac{u_D - u_P}{d_{PD}}\right). \end{aligned} \tag{5.4}$$

The same interpolation formula applies for β_D.

The discretization (5.3) along with the (5.4) is first order accurate for the second derivatives, but it allows to obtain second order for the solution (see e.g. [66, 74, 102]).

Now, we can easily define the first order derivatives to compute the gra-

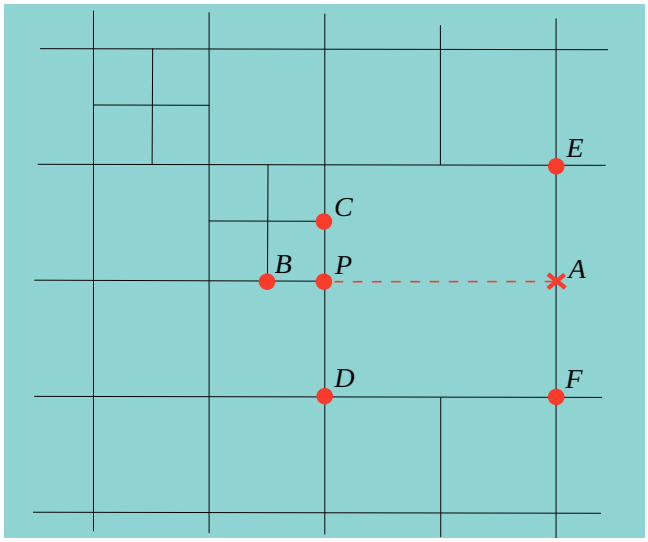


Fig. 5.3: A configuration illustrating the nodes involved in the discretization at a T-junction node v_0.

dient:

$$\begin{aligned} D_x u_P &= \frac{u_A - u_P}{d_{PA}} \frac{d_{PB}}{d_{AB}} + \frac{u_P - u_B}{d_{PB}} \frac{d_{PA}}{d_{AB}}, \\ D_y u_P &= \frac{u_C - u_P}{d_{PC}} \frac{d_{PD}}{d_{CD}} + \frac{u_P - u_D}{d_{PD}} \frac{d_{PC}}{d_{CD}}. \end{aligned}$$

To close the linear system, we must write an equation for each ghost point, imposing the boundary condition.

5.2 Numerical tests

In this section we present numerical tests that confirm the second order accuracy of the scheme described in this paper both for the solution and its gradient. For each test we set the minimum and maximum level of refinement: *min_level* and *max_level*. Let $N_{\min} = 2^{min\_level}$ and $N_{\max} = 2^{max\_level}$. The maximum resolution of the adaptive grid is determined by a spatial step $h_{\min} = 2/N_{\max}$, while the minimum resolution is $h_{\max} = 2/N_{\min}$. If not specified, we intend $h = h_{\min}$. Most numerical tests are taken from [84, 57, 70]. In all the tests, all best-fit line figure are performed in a loglog scale plot, using the natural logarithm. The BiCGSTAB solver is used to solve the resulting linear system for numerical test 5.2.1 (continuous coefficient). For the numerical tests regarding the discontinuous coefficient case, the BiCGSTAB

fails to converge, especially for linear systems with bigger dimension. In this case the MATLAB solver is used.

5.2.1 Example 1: mixed boundary conditions

We start with two numerical tests to solve the Poisson equation with continuous coefficient described in Chapter 1. The Dirichlet and Neumann part of the boundary are chosen as follows:

$$\Gamma_D = \{(x,y) \in \partial\Omega \colon x \leq 0\}, \quad \Gamma_N = \partial\Omega \backslash \Gamma_D.$$

In this first numerical test we choose the exact solution u, the coefficient β, and the domain (given by the level-set function) as follows:

$$\begin{aligned} u &= \sin(\pi x)\cos(\pi y), \\ \beta &= 2 + \sin(xy), \\ \phi &= \sqrt{(x-0.0413)^2 + (y+0.0613)^2} - 0.783. \end{aligned}$$

In Table 5.1 we list the errors obtained in the L^2 and L^∞ norms for the solution and the gradient. In Fig. 5.4 we depict the solution (left) and the bestfit lines of the errors (right).

Table 5.1: Example 5.2.1. Accuracy order in the solution (top) and in the gradient (bottom).

$(N_{\min},N_{\max})$	L^2 error of u	order	L^∞ error of u	order
(64,256)	$2.07 \cdot 10^{-3}$	-	$3.58 \cdot 10^{-3}$	-
(128,512)	$5.38 \cdot 10^{-4}$	1.94	$9.60 \cdot 10^{-4}$	1.90
(256,1024)	$1.35 \cdot 10^{-4}$	2.00	$2.53 \cdot 10^{-4}$	1.93
(512,2048)	$2.73 \cdot 10^{-5}$	2.30	$6.46 \cdot 10^{-5}$	1.97
$(N_{\min},N_{\max})$	L^2 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
(64,256)	$3.90 \cdot 10^{-3}$	-	$4.50 \cdot 10^{-2}$	-
(128,512)	$1.06 \cdot 10^{-3}$	1.89	$1.47 \cdot 10^{-2}$	1.61
(256,1024)	$2.74 \cdot 10^{-4}$	1.95	$4.25 \cdot 10^{-3}$	1.79
(512,2048)	$6.03 \cdot 10^{-5}$	2.18	$1.73 \cdot 10^{-3}$	1.30

5.2.2 Example 2: discontinuous coefficient case

This example is taken from [84, 57].
The interface Γ is a simple circle with radius 0.5 and midpoint at $(0,0)$. The

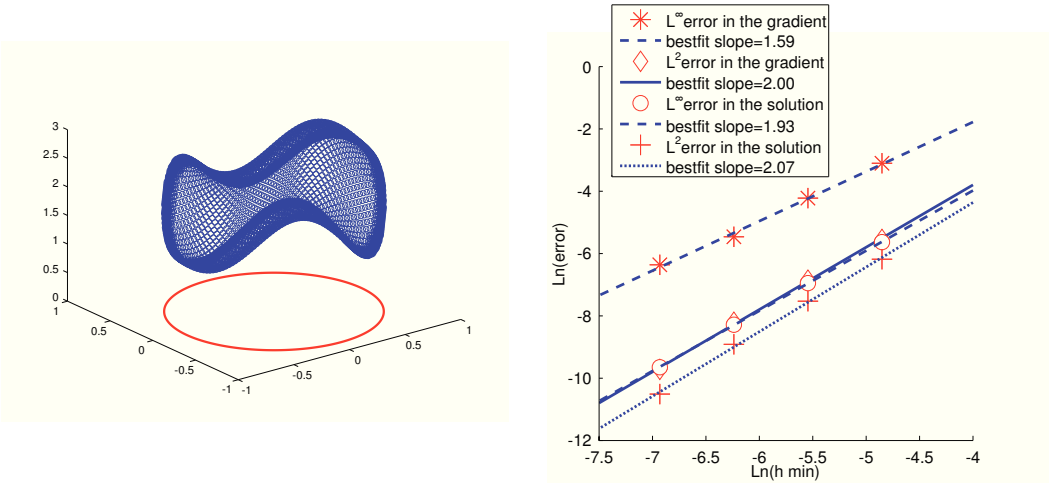


Fig. 5.4: Example 5.2.1. Left: Numerical solution for $(N_{min}, N_{max}) = (64, 256)$. Right: bestfit lines of the errors in the L^2 and L^∞ norms.

analytic solutions $u^\pm$, the coefficients $\beta^\pm$, and the level set function are given as follows:

$$u^+ = \ln(x^2 + y^2), \quad u^- = \sin(x + y)$$
$$\beta^+ = \sin(x + y) + 2, \quad \beta^- = \cos(x + y) + 2$$
$$\phi = \sqrt{x^2 + y^2} - 0.5$$

Table 5.2: Example 5.2.2. Accuracy order in the solution (top) and in the gradient (bottom).

$(N_{\min}, N_{\max})$	L^2 error of u	order	L^∞ error of u	order
(64,256)	$1.04 \cdot 10^{-4}$	-	$2.51 \cdot 10^{-4}$	-
(128,512)	$2.62 \cdot 10^{-5}$	1.99	$6.43 \cdot 10^{-5}$	1.96
(256,1024)	$7.02 \cdot 10^{-6}$	1.90	$1.75 \cdot 10^{-5}$	1.87
(512,2048)	$1.69 \cdot 10^{-6}$	2.05	$4.36 \cdot 10^{-6}$	2.01
$(N_{\min}, N_{\max})$	L^2 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
(64,256)	$9.71 \cdot 10^{-4}$	-	$3.57 \cdot 10^{-3}$	-
(128,512)	$2.82 \cdot 10^{-4}$	1.78	$1.12 \cdot 10^{-3}$	1.68
(256,1024)	$7.77 \cdot 10^{-5}$	1.86	$3.09 \cdot 10^{-4}$	1.85
(512,2048)	$2.04 \cdot 10^{-5}$	1.93	$8.34 \cdot 10^{-5}$	1.89

In Table 5.2 we list the errors obtained in the L^2 and L^∞ norms for the solution and the gradient. In Fig. 5.5 we depict the solution (left) and the bestfit lines of the errors (right).

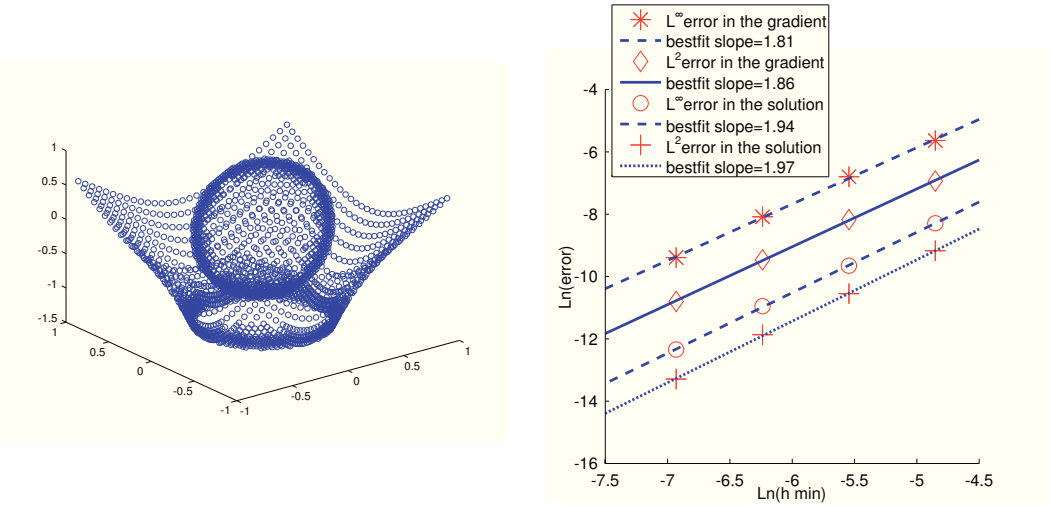


Fig. 5.5: Example 5.2.2. Left: Numerical solution for $(N_{min}, N_{max}) = (64, 256)$. Right: bestfit lines of the errors in the L^2 and L^∞ norms.

From Fig. 5.6 it can be inferred that an adaptive refinement with $max\_level = min\_level+2$ should be enough for damping the interface errors for such jump in the coefficient. In fact, while for uniform grid the error is concentrated on the interface, with an adaptive grid it is distributed on all the domain smoothly.

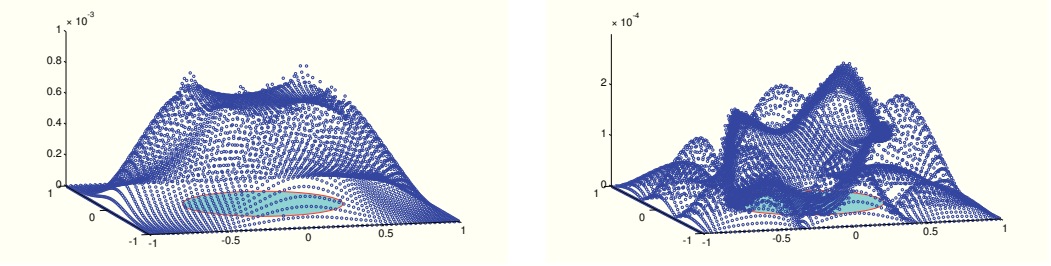


Fig. 5.6: Example 5.2.2. Left: error obtained in a uniform grid with $(N_{\min},N_{\max}) = (64, 64)$. The error is concentrated on the interface. Right: error obtained with an adaptive grid with $(N_{\min},N_{\max}) = (64, 256)$. The error varies smoothly.

5.2.3 Example 3

In this test we compare the results between the scheme proposed in this paper and the scheme presented by Fedkiw *et al.* in [70]. The computational domain is $D = [0, 1]^2$. We choose the following data for the analytic solutions

$u^{\pm}$, the coefficients $\beta^{\pm}$, and the level set function (ellipse):

$$u^+ = 0, \quad u^- = 5y$$

$$\beta^+ = 1, \quad \beta^- = 1$$

$$\phi = \sqrt{35(x-0.5)^2 + (y-0.5)^2} - 0.45.$$

Table 5.3: Example 5.2.3. Accuracy order in the solution (top) and in the gradient (bottom).

$(N_{\min},N_{\max})$	L^2 error of u	order	L^∞ error of u	order
(16,64)	$1.30 \cdot 10^{-4}$	-	$6.31 \cdot 10^{-4}$	-
(32,128)	$3.16 \cdot 10^{-5}$	2.04	$1.19 \cdot 10^{-4}$	2.40
(64,256)	$1.07 \cdot 10^{-5}$	1.57	$3.50 \cdot 10^{-5}$	1.77
(128,512)	$2.85 \cdot 10^{-6}$	1.90	$9.84 \cdot 10^{-6}$	1.83
(256,1024)	$5.02 \cdot 10^{-7}$	2.50	$2.05 \cdot 10^{-6}$	2.27
(512,2048)	$9.51 \cdot 10^{-8}$	2.40	$4.74 \cdot 10^{-7}$	2.11
$(N_{\min},N_{\max})$	L^2 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
(16,64)	$2.88 \cdot 10^{-3}$	-	$1.43 \cdot 10^{-2}$	-
(32,128)	$6.42 \cdot 10^{-4}$	2.17	$4.02 \cdot 10^{-3}$	1.83
(64,256)	$2.00 \cdot 10^{-4}$	1.68	$1.09 \cdot 10^{-3}$	1.89
(128,512)	$5.21 \cdot 10^{-5}$	1.94	$3.87 \cdot 10^{-4}$	1.49
(256,1024)	$9.85 \cdot 10^{-6}$	2.40	$9.86 \cdot 10^{-5}$	1.97
(512,2048)	$2.02 \cdot 10^{-6}$	2.29	$2.57 \cdot 10^{-5}$	1.94

In Table 5.3 we list the errors obtained in the L^2 and L^∞ norms for the solution and the gradient. In Fig. 5.7 we depict the solution (left) and the bestfit lines of the errors for the scheme described in [70] (middle) and the scheme proposed in this paper (right). Looking at the middle figure, we can observe that the error decreases as expected until approximately 40 grid points in each direction are introduced. After this point, the error no longer decreases. By inspection of the exact solution, it is clear that over a majority of the interface, the jump in the normal derivative is approximatively zero, while the jump in the tangential derivative is significant. The method of Fedkiw *et al.* [70] assumes the tangential derivative is zero. In cases (like the one here) in which the tangential derivative is significant, the method proposed in [70] fails to converge to the exact solution below some critical mesh size.

Remark. Since the exact solutions u^+ and u^- are polynomials of degree at most one, we should observe discretization errors of the order of

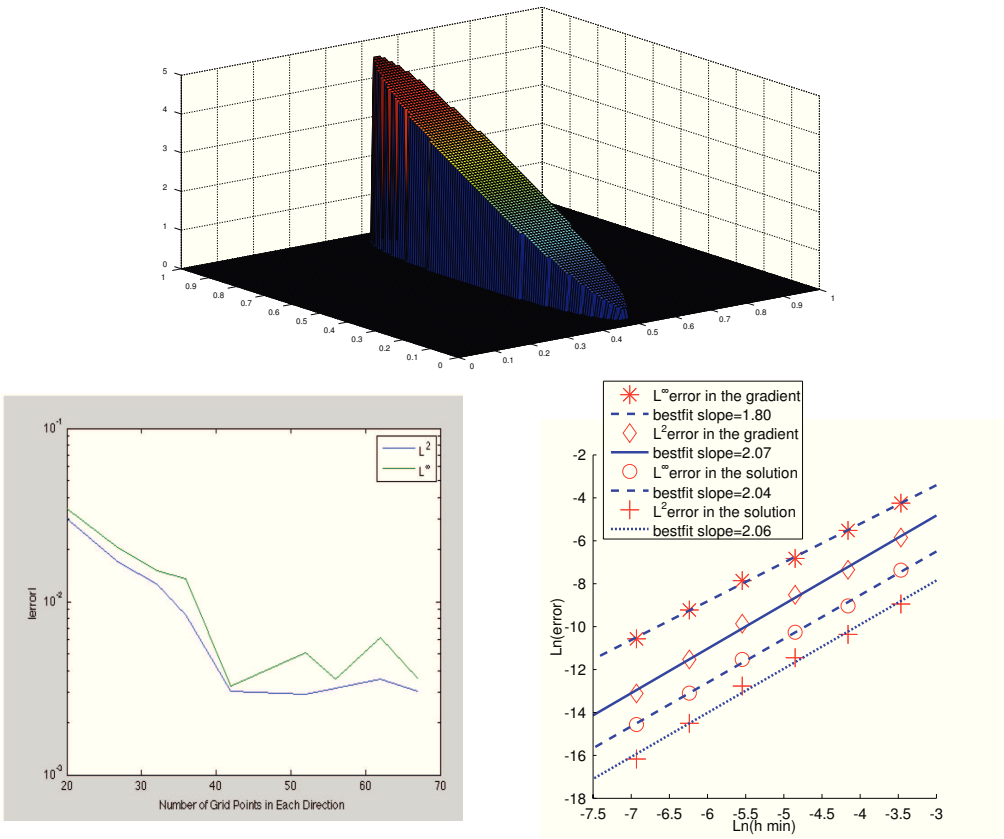


Fig. 5.7: Example 5.2.3. Top: Exact solution. Bottom-left: bestfit lines of the errors for the scheme proposed in [70]. Bottom-right: bestfit lines of the errors for the scheme proposed in this paper.

machine accuracy. Indeed, the representation of the ellipsoidal domain is not exact to the machine precision, and its error affects the discretization error of the jump of the flux, and consequently the accuracy of the method, which appears to be second order.

5.2.4 Example 4

This example is taken from [84, 68]. In this case, Γ is a flower-shaped interface with parametric equations (see the right side of Fig. 5.2):

$$X(\vartheta) = r(\vartheta)\cos(\vartheta) + x_0,$$
$$Y(\vartheta) = r(\vartheta)\sin(\vartheta) + y_0,$$

with $\vartheta \in [0, 2\pi]$ and $r(\vartheta) = r_0 + r_1 \sin(\omega\vartheta)$. The parameters are set to $r_0 = 0.5$, $r_1 = 0.2$, $\omega = 5$ and $x_0 = y_0 = 0.2/\sqrt{20}$. The level-set representation of

this interface is:

$$\phi(x,y) = r - r_0 - r_1 \frac{(y-y_0)^5 + 5(x-x_0)^4(y-y_0) - 10(x-x_0)^2(y-y_0)^3}{r^5}.$$

where $r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$. The analytic solutions $u^{\pm}$ and the coefficients $\beta^{\pm}$ are given as follows:

$$u^+ = \frac{r^4 + C_0 \log(2r)}{\beta^+}, \quad u^- = \frac{r^2}{\beta^-}$$

$$\beta^+ = \text{const.}, \quad \beta^- = \text{const.}$$

where $C_0 = -0.1$.

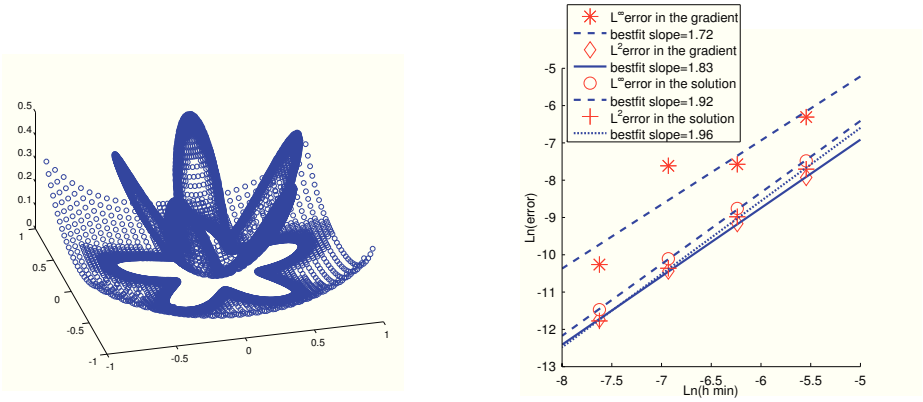


Fig. 5.8: Example 5.2.4. Left: numerical solution, Right: bestfit lines of the errors. The results are obtained for $\beta^- = 1$ and $\beta^+ = 10$.

In Tables 5.4, 5.5, 5.6 we list the errors obtained in the L^2 and L^∞ norms for the solution and the gradient. In Figs. 5.8 and 5.9 we depict the solutions (left) and the bestfit lines of the errors (right) for different choices of the coefficients.

5.2.5 Example 5

The interface Γ is a simple circle with radius 0.5 and midpoint at $(x_0, y_0) \equiv (0.0413, -0.0613)$. The analytic solutions $u^{\pm}$ and the coefficients $\beta^{\pm}$ are given as follows:

$$u^+ = e^x \left(x^2 \sin(y) + y^2\right); u^- = -(x^2 + y^2);$$

$$\beta^+ = 1000(xy + 5), \quad \beta^- = 1 + x^2 + y^2.$$

In Table 5.7 we list the errors obtained in the L^2 and L^∞ norms for the solution and the gradient. In Fig. 5.10 we depict the solution (left) and the bestfit lines of the errors (right).

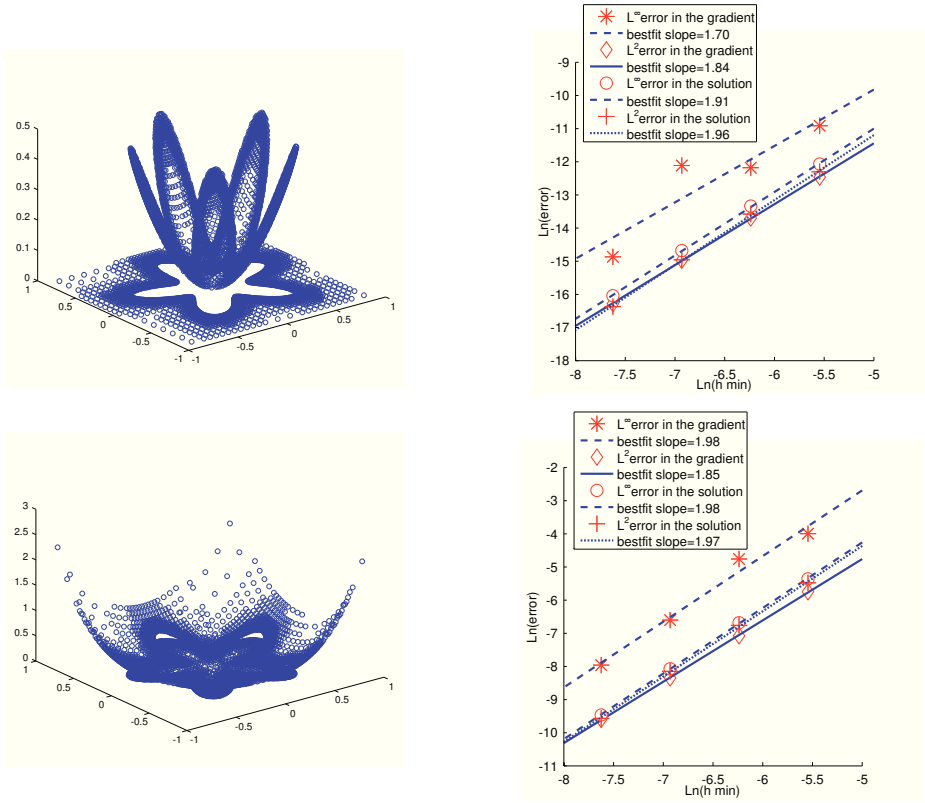


Fig. 5.9: Example 5.2.4. Left: numerical solution, Right: bestfit lines of the errors. The results are obtained for different values of the coefficient: $\beta^- = 1$ and $\beta^+ = 1000$ (top), $\beta^- = 1000$ and $\beta^+ = 1$ (bottom).

Table 5.4: Example 5.2.4. Accuracy order in the solution (top) and in the gradient (bottom) for $\beta^- = 1$, $\beta^+ = 10$.

$(N_{\min},N_{\max})$	L^2 error of u	order	L^∞ error of u	order
(32,512)	$4.51 \cdot 10^{-4}$	-	$5.63 \cdot 10^{-4}$	-
(64,1024)	$1.26 \cdot 10^{-4}$	1.84	$1.57 \cdot 10^{-4}$	1.84
(128,2048)	$3.16 \cdot 10^{-5}$	1.99	$4.08 \cdot 10^{-5}$	1.95
(256,4096)	$7.70 \cdot 10^{-6}$	2.04	$1.04 \cdot 10^{-5}$	1.97
$(N_{\min},N_{\max})$	L^2 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
(32,512)	$3.61 \cdot 10^{-4}$	-	$1.82 \cdot 10^{-3}$	-
(64,1024)	$1.05 \cdot 10^{-4}$	1.79	$5.14 \cdot 10^{-4}$	1.83
(128,2048)	$2.93 \cdot 10^{-5}$	1.83	$4.93 \cdot 10^{-4}$	0.06
(256,4096)	$7.99 \cdot 10^{-6}$	1.88	$3.49 \cdot 10^{-5}$	3.82

Table 5.5: Example 5.2.4. Accuracy order in the solution (top) and in the gradient (bottom) for $\beta^- = 1$, $\beta^+ = 1000$.

$(N_{\min},N_{\max})$	L^2 error of u	order	L^∞ error of u	order
(32,512)	$4.55 \cdot 10^{-6}$	-	$5.74 \cdot 10^{-6}$	-
(64,1024)	$1.27 \cdot 10^{-6}$	1.84	$1.62 \cdot 10^{-6}$	1.83
(128,2048)	$3.20 \cdot 10^{-7}$	1.99	$4.22 \cdot 10^{-7}$	1.94
(256,4096)	$7.79 \cdot 10^{-8}$	2.04	$1.08 \cdot 10^{-7}$	1.96
$(N_{\min},N_{\max})$	L^2 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
(32,512)	$3.87 \cdot 10^{-6}$	-	$1.82 \cdot 10^{-5}$	-
(64,1024)	$1.12 \cdot 10^{-6}$	1.78	$5.14 \cdot 10^{-6}$	1.83
(128,2048)	$3.15 \cdot 10^{-7}$	1.84	$5.49 \cdot 10^{-6}$	-0.10
(256,4096)	$8.50 \cdot 10^{-8}$	1.89	$3.49 \cdot 10^{-7}$	3.97

Table 5.6: Example 5.2.4. Accuracy order in the solution (top) and in the gradient (bottom) for $\beta^- = 1000$, $\beta^+ = 1$.

$(N_{\min},N_{\max})$	L^2 error of u	order	L^∞ error of u	order
(32,512)	$4.19 \cdot 10^{-3}$	-	$4.72 \cdot 10^{-3}$	-
(64,1024)	$1.16 \cdot 10^{-3}$	1.85	$1.25 \cdot 10^{-3}$	1.91
(128,2048)	$2.90 \cdot 10^{-4}$	2.00	$3.13 \cdot 10^{-4}$	2.00
(256,4096)	$7.02 \cdot 10^{-5}$	2.05	$7.77 \cdot 10^{-5}$	2.01
$(N_{\min},N_{\max})$	L^2 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
(32,512)	$3.19 \cdot 10^{-3}$	-	$1.84 \cdot 10^{-2}$	-
(64,1024)	$8.46 \cdot 10^{-4}$	1.92	$8.58 \cdot 10^{-3}$	1.10
(128,2048)	$2.37 \cdot 10^{-4}$	1.83	$1.36 \cdot 10^{-3}$	2.66
(256,4096)	$6.78 \cdot 10^{-5}$	1.81	$3.50 \cdot 10^{-4}$	1.96

Table 5.7: Example 5.2.5. Accuracy order in the solution (top) and in the gradient (bottom).

$(N_{\min},N_{\max})$	L^2 error of u	order	L^∞ error of u	order
(32,128)	$1.12 \cdot 10^{-3}$	-	$3.54 \cdot 10^{-3}$	-
(64,256)	$2.69 \cdot 10^{-4}$	2.06	$8.98 \cdot 10^{-4}$	1.98
(128,512)	$6.49 \cdot 10^{-5}$	2.05	$2.25 \cdot 10^{-4}$	1.99
(256,1024)	$1.59 \cdot 10^{-5}$	2.03	$5.64 \cdot 10^{-5}$	2.00
(512,2048)	$3.91 \cdot 10^{-6}$	2.02	$1.41 \cdot 10^{-5}$	2.00
$(N_{\min},N_{\max})$	L^2 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
(32,128)	$8.91 \cdot 10^{-3}$	-	$1.96 \cdot 10^{-1}$	-
(64,256)	$3.08 \cdot 10^{-3}$	1.53	$6.99 \cdot 10^{-2}$	1.49
(128,512)	$9.33 \cdot 10^{-4}$	1.72	$2.12 \cdot 10^{-2}$	1.72
(256,1024)	$2.60 \cdot 10^{-4}$	1.84	$5.91 \cdot 10^{-3}$	1.85
(512,2048)	$6.92 \cdot 10^{-5}$	1.91	$1.57 \cdot 10^{-3}$	1.91

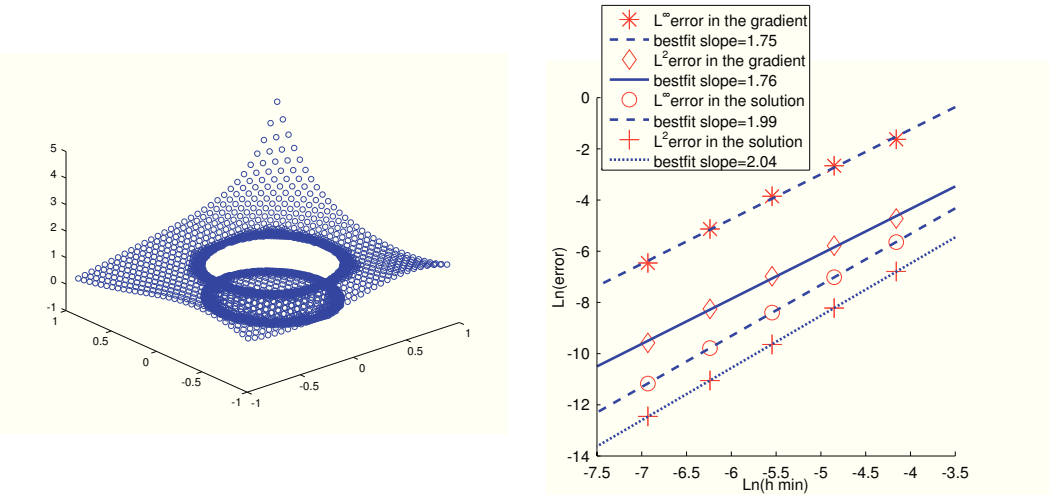


Fig. 5.10: Example 5.2.5. Left: Numerical solution for $(N_{min}, N_{max}) = (64, 256)$. Right: bestfit lines of the errors in the L^2 and L^∞ norms.

Appendix A

A tentative of convergence proof for second order accuracy

n Chapter 2 a convergence proof of the first order method is provided by Gershgorin-Hadamard theorems. Although the same proof cannot be carried out for the second order accurate scheme, we always observed convergence in all numerical tests we have performed. In this chapter we slightly modify the one dimensional second order accurate scheme in order to provide a convergence proof by Gershgorin-Hadamard theorems as well.

Section A.2 presents an alternative second order scheme and provides a convergence proof. In Section A.3 a detailed Taylor analysis is carried out in order to compare the original scheme with the modified one: as expected, the original scheme is more accurate. In the last section, some numerical tests are presented.

A.1 Description of the iterative scheme

The second order accurate iterative scheme for the one dimensional case has been described in Chapter 2, in particular it consists of the relaxation scheme (2.9), (2.12), (2.13). However, this scheme is second order accurate in the solution and gradient as well, while if we want to limit to the second order accuracy only in the solution we can replace the iteration (2.12) by the following:

$$u_l^{(n+1)} = u_l^{(n)} - \mu_D \Delta t \left(\vartheta_l u_l^{(n)} + (1 - \vartheta_l) u_{l+1}^{(n)} - g_a \right). \tag{A.1}$$

We study the simpler second order accurate (only in the solution) iterative scheme (2.9), (A.1), (2.13).

The constants μ_D and μ_N are chosen in order to satisfy the CFL conditions, i.e. $\mu_D \Delta t < 1$ and $\mu_N \Delta t/h < 2/3$ (see 2.14). Since $\Delta t = h^2/2$, we then can choose (for instance) $\mu_D = 1.8/h^2$, $\mu_N = 3.6/(3h)$. In summary, our second order accurate iterative method is composed by Eqs. (2.9), (A.1) and (2.13), with the choice of constants

$$\Delta t = h^2/2, \quad \mu_D = 1.8/h^2, \quad \mu_N = 3.6/(3h).$$

A.2 Convergence proof

In all numerical tests we performed we always observed convergence of the second order iterative scheme (2.9), (2.12) and (2.13). However, in order to provide a convergence proof, we slightly modify the scheme, still maintaining second order accuracy. A convergence proof of the first order accurate version of the method can be found in [34].

Let $\tilde{N} = r - l + 1$. Let us rewrite the scheme in matrix fashion:

$$\mathbf{u}^{(n+1)} = B\mathbf{u}^{(n)} + F \tag{A.2}$$

where $\mathbf{u}^{(n)}$, $F \in \mathbb{R}^{\tilde{N}}$, $B \in \mathbb{R}^{\tilde{N}\times\tilde{N}}$ are defined as follows:

$$\mathbf{u}^{(n)} = \begin{pmatrix} u_l^{(n)} \\ u_{l+1}^{(n)} \\ \vdots \\ u_r^{(n)} \end{pmatrix}, F = \begin{pmatrix} \Delta t g_a \\ h^2 f_{l+1} \\ \vdots \\ h^2 f_{r-1} \\ \Delta t g_b \end{pmatrix}$$

$$B = \begin{pmatrix} 1 - \mu_D\,\Delta t\,\vartheta_l & \mu_D\,\Delta t\,(1-\vartheta_l) & & & & \\ 1/2 & 0 & 1/2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1/2 & 0 & 1/2 & \\ & & c\,(0.5-\vartheta_r) & 2\,c\,\vartheta_r & 1 - c\,(0.5+\vartheta_r) \end{pmatrix}. \tag{A.3}$$

where $c = \mu_N \Delta t/h$ is the Courant number. A necessary and sufficient condition for the convergence of (A.2) is $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of the matrix B (i.e. the maximum eigenvalue of B in absolute value). In order to prove this condition we shall make use of Gershgorin-Hadamard theorems (that can be founded in any good basic text of Numerical Analysis, such as [80] or [91]).

The first theorem states that if $B \in \mathbb{C}^{m\times m}$, with $m \in \mathbb{N}$, then every eigenvalue λ of B satisfies:

$$\lambda \in S_C = \cup_{i=1}^{m} C_i, \quad C_i = \{z \in \mathbb{C} : |z - b_{ii}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{m} |b_{ij}|\}$$

and the sets C_i are said *Gershgorin circles*.
The second theorem states that, if B is an irreducible matrix, every eigenvalue of B belonging to the boundary of S_C, belongs to the boundary of all Gershgorin circles. Applying Gershgorin-Hadamard theorems, we proof the convergence if all Gershgorin circles are contained in the circle $\mathcal{C}((0,0),1)$ centered at the origin and with radius 1, and at least one circle is contained in the interior of $\mathcal{C}((0,0),1)$. Letting $c_i = b_{ii}$ and $r_i = \sum_{\substack{j=1 \\ j\neq i}}^{m} |b_{ij}|$ be respectively the center and the radius of the $i-$th Gershgorin circle for $i = 1, \ldots, \tilde{N}$, proving convergence is equivalent to proof that

$$|c_i| \leq 1 - r_i \tag{A.4}$$

for all $i = 1, \ldots, \tilde{N}$ and $|c_j| < 1 - r_j$ for at least one $j \in \left\{1, \ldots, \tilde{N}\right\}$. While this condition is satisfied for $i = 2, \ldots, \tilde{N} - 1$, unfortunately it is not satisfied for $i = 1$ and $i = \tilde{N}$. more precisely, recalling the CFL conditions $\mu_D \Delta t < 1$ and $\mu_N \Delta t/h < 1$, condition (A.4) is satisfied for $i = 1$ if and only if $\vartheta_l \geq 0.5$, while it is satisfied for $i = \tilde{N}$ if and only if $\vartheta_r \leq 0.5$.

In order to always satisfies $\vartheta_l \geq 0.5$ and $\vartheta_r \leq 0.5$, we modify the scheme as follows. Iteration (2.12) use a linear interpolation in nodes x_l and x_{l+1} for the spatial discretization. In case of $\vartheta_l < 0.5$, we instead use grid points x_l and x_{l+2} for the linear interpolation. In such case, iteration (2.12) is replaced by:

$$u_l^{(n+1)} = u_l^{(n)} - \mu_D \Delta t \left(\tilde{\vartheta}_l u_l^{(n)} + (1 - \tilde{\vartheta}_l) u_{l+2}^{(n)} - g_a\right)$$

where $\tilde{\vartheta}_l = (1 + \vartheta_l)/2$.

Iteration (2.13) use a quadratic interpolation in nodes $\{x_{r-2}, x_{r-1}, x_r\}$ for the spatial discretization. In case of $\vartheta_r > 0.5$, we use x_{r+1} as ghost point instead of x_r, using a quadratic interpolation in nodes $\{x_{r-3}, x_{r-1}, x_{r+1}\}$ to reconstruct the spatial derivative. Iteration (2.13) is replaced by:

$$u_{r+1}^{(n+1)} = u_{r+1}^{(n)} - \frac{\mu_N \Delta t}{2h}\left(u_{r-1}^{(n)} - u_{r-3}^{(n)} + \left(u_{r-3}^{(n)} - 2u_{r-1}^{(n)} + u_{r+1}^{(n)}\right)\left(\frac{1}{2} + \tilde{\vartheta}_r\right)\right) + \mu_N \Delta t\, g_b$$

where $\tilde{\vartheta}_r = \vartheta_r/2$. In this case we also must replace iteration (2.9) for $i = r-1$ with:

$$u_{r-1}^{(n+1)} = \frac{1}{2}\left(u_{r-3}^{(n)} + u_{r+1}^{(n)} + (2h)^2 f_i\right).$$

In these cases, the iteration matrix B and vectors $\mathbf{u}^{(n)}$ and F of (A.2) become:

$$\mathbf{u}^{(n)} = \begin{pmatrix} u_l^{(n)} \\ u_{l+1}^{(n)} \\ \vdots \\ u_{r-1}^{(n)} \\ u_{r+1}^{(n)} \end{pmatrix}, F = \begin{pmatrix} \Delta t g_a \\ h^2 f_{l+1} \\ \vdots \\ h^2 f_{r-2} \\ (2h)^2 f_{r-1} \\ \Delta t g_b \end{pmatrix}$$

$$B = \begin{pmatrix} 1 - \mu_D \Delta t \, \tilde{\vartheta}_l & 0 & \mu_D \Delta t \, (1 - \tilde{\vartheta}_l) & & \\ 1/2 & 0 & 1/2 & & \\ & \ddots & \ddots & \ddots & \\ & 1/2 & 0 & 0 & 1/2 \\ & \frac{c}{2}(0.5 - \tilde{\vartheta}_r) & 0 & c\,\tilde{\vartheta}_r & 1 - \frac{c}{2}(0.5 + \tilde{\vartheta}_r) \end{pmatrix}. \tag{A.5}$$

Conditions $\tilde{\vartheta}_l \geq 0.5$ and $\tilde{\vartheta}_r \leq 0.5$ are always satisfied; therefore $\rho(B) < 1$.

A.3 Comparison between the two methods: Taylor analysis

In this section we analyze the difference between the errors obtained with the iteration scheme (A.2)-(A.3) (that we call method (A)) and with the iteration scheme (A.2)-(A.5) (that we call method (B)). In order to avoid oscillation in the error, when we perform method (B) we always use the matrix (A.5), even if $\vartheta_l \geq 0.5$ or $\vartheta_r \leq 0.5$.

Method (A) converges to the solution of the linear system:

$$\begin{cases} \dfrac{1}{h^2}(u_{i-1} - 2\,u_i + u_{i+1}) = f_i, \quad i = l+1, \ldots, r-1 \\ \vartheta_l u_l + (1 - \vartheta_l) u_{l+1} = g_a \\ \dfrac{1}{h}(u_{r-1} - u_{r-2} + (u_r - 2\,u_{r-1} + u_{r-2})(0.5 + \vartheta_r)) = g_b \end{cases} \tag{A.6}$$

that we summarize as $L^h \mathbf{u}^h = f^h$, while method (B) converges to the solution

of the linear system:

$$\begin{cases} \dfrac{1}{h^2}\,(u_{i-1} - 2\,u_i + u_{i+1}) = f_i, \quad i = l+1, \ldots, r-2 \\ \dfrac{1}{4\,h^2}\,(u_{i-2} - 2\,u_i + u_{i+2}) = f_i, \quad i = r-1 \\ \tilde{\vartheta}_l u_l + (1 - \tilde{\vartheta}_l) u_{l+2} = g_a \\ \dfrac{1}{2h}\left(u_{r-1} - u_{r-3} + (u_{r+1} - 2\,u_{r-1} + u_{r-3})(0.5 + \tilde{\vartheta}_r)\right) = g_b \end{cases} \tag{A.7}$$

that we summarize as $\tilde{L}^h\,\tilde{\mathbf{u}}^h = \tilde{f}^h$.

Let u be the exact solution. Let us start studying the error for the linear system (A.6). The error $e^h := \mathbf{u}_h - u$ satisfies the defect equation:

$$L^h\,e^h = d^h := f^h - L^h\,u. \tag{A.8}$$

By a Taylor analysis, we can compute the discretization error d^h. For equations $i = l+1, \ldots, r-1$, we obtain:

$$d_i^h = -u''(x_i) + \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} = \frac{h^2}{2}u^{IV}(x_i) + O(h^4), \quad i = l+1, \ldots, r-1 \tag{A.9}$$

By the same argument, we can compute the discretization error for boundary conditions. For the Dirichlet boundary condition, performing a Taylor expansion of the solution, we obtain:

$$\begin{aligned} u(x_{l+1}) &= u(a) + (\vartheta_l\,h)u'(a) + \frac{(\vartheta_l h)^2}{2}u''(a) + O(h^3) \\ u(x_l) &= u(a) - ((1-\vartheta_l)\,h)u'(a) + \frac{((1-\vartheta_l)h)^2}{2}u''(a) + O(h^3) \end{aligned}$$

Multiplying the first equation by $1 - \vartheta_l$ and second equation by ϑ_l, summing and collecting terms, we obtain:

$$d_l^h = u(a) - (\vartheta_l u(x_l) + (1-\vartheta_l)u(x_{l+1})) = -\frac{\vartheta_l(1-\vartheta_l)h^2}{2}u''(a) + O(h^3) \tag{A.10}$$

For the Neumann boundary conditions:

$$\begin{aligned} u(x_r) &= u(b) + (1-\vartheta_r)h\,u'(b) + \frac{((1-\vartheta_r)h)^2}{2}u''(b) + \frac{((1-\vartheta_r)h)^3}{6}u'''(b) + O(h^4) \\ u(x_{r-1}) &= u(b) - \vartheta_r\,h\,u'(b) + \frac{(\vartheta_r\,h)^2}{2}u''(b) - \frac{(\vartheta_r\,h)^3}{6}u'''(b) + O(h^4) \\ u(x_{r-2}) &= u(b) - (1+\vartheta_r)h\,u'(b) + \frac{((1+\vartheta_r)h)^2}{2}u''(b) - \frac{((1+\vartheta_r)h)^3}{6}u'''(b) + O(h^4) \end{aligned}$$

After some algebra, we obtain the discretization error:

$$\begin{aligned} d_r^h &= u'(b) - \frac{1}{h}\left(u(x_{r-1}) - u(x_{r-2}) + (u(x_r) - 2u(x_{r-1}) + u(x_{r-2}))(0.5 + \vartheta_r)\right) \\ &= \left(\vartheta_r^2 - \frac{1}{3}\right)\frac{h^2}{2}u'''(b) + O(h^3) \end{aligned} \tag{A.11}$$

Neglecting higher order terms, we can claim from (A.8), (A.9), (A.10) and (A.11) that the error e^h is a second order accurate numerical solution of the differential problem:

$$\begin{cases} -e'' = \dfrac{h^2}{12}u^{IV} \text{ in }]a,b[\\ e(a) = -\dfrac{\vartheta_l(1-\vartheta_l)h^2}{2}u''(a) \\ e'(b) = \left(\vartheta_r^2 - \dfrac{1}{3}\right)\dfrac{u'''(b)}{2}h^2 \end{cases} \tag{A.12}$$

Similarly, we can analyze the behavior of the error of (A.7). For simplicity we do not take into account the difference between the first two equations of (A.7) in the computation of the defect, and then we assume that the defect for inner equations is again (A.9). At the end, we will obtain the following differential problem for the error $\tilde{e}^h$:

$$\begin{cases} -e'' = \dfrac{h^2}{12}u^{IV} \text{ in }]a,b[\\ e(a) = -\dfrac{\tilde{\vartheta}_l(1-\tilde{\vartheta}_l)(2h)^2}{2}u''(a) \\ e'(b) = \left(\tilde{\vartheta}_r^2 - \dfrac{1}{3}\right)\dfrac{u'''(b)}{2}(2h)^2 \end{cases} \tag{A.13}$$

We can summarize (A.12) and (A.13) in a unique differential problem:

$$\begin{cases} -e'' = \dfrac{h^2}{12}u^{IV} \text{ in }]a,b[\\ e(a) = -\dfrac{\bar{\vartheta}_l(1-\bar{\vartheta}_l)(jh)^2}{2}u''(a) \\ e'(b) = \left(\bar{\vartheta}_r^2 - \dfrac{1}{3}\right)\dfrac{u'''(b)}{2}(jh)^2 \end{cases} \tag{A.14}$$

where $\bar{\vartheta}_I = \vartheta_I$ ($I \in \{l,r\}$) and $j = 1$ for the method (A), while $\bar{\vartheta}_I = \tilde{\vartheta}_I$ ($I \in \{l,r\}$) and $j = 2$ for the method (B). From the first equation of (A.14)

we obtain

$$e(x) = -\frac{h^2}{12}u''(x) + m\,x + q. \tag{A.15}$$

We found the parameters m and q imposing second and third equation of (A.14). From the third equation:

$$-\frac{h^2}{12}u'''(b)+m = \left(\bar{\vartheta}_r^2 - \frac{1}{3}\right)\frac{u'''(b)}{2}(jh)^2 \Longrightarrow \boxed{m = \frac{h^2u'''(b)}{2}\left(j^2\left(\bar{\vartheta}_r^2 - \frac{1}{3}\right) + \frac{1}{6}\right)} \tag{A.16}$$

From the second equation of (A.14):

$$-\frac{h^2}{12}u''(a) + m\,x + q = -\frac{\bar{\vartheta}_l(1-\bar{\vartheta}_l)(jh)^2}{2}u''(a) \Longrightarrow$$
$$\Longrightarrow \boxed{q = \frac{h^2}{2}\left(u''(a)\left(\frac{1}{6} - \bar{\vartheta}_l(1-\bar{\vartheta}_l)j^2\right) - a\,u'''(b)\left(\frac{1}{6} + j^2\left(\bar{\vartheta}_r^2 - \frac{1}{3}\right)\right)\right)} \tag{A.17}$$

In order to compare the errors for the two methods, we define the ratio

$$\alpha = \frac{\|\text{error with method (B)}\|}{\|\text{error with method (A)}\|}. \tag{A.18}$$

When h is very small, we can approximate the error $e(x)$ with a line, i.e., from (A.15), $e(x) \approx m\,x + q$. With this approximation, if we compute α by (A.18) with the L^1 norm or with the L^∞ norm we obtain the same result. Let us compute it using the L^∞ norm. Let us suppose the greater error is produced by the discretization of the Neumann condition, then

$$\text{Assumption: } \max_{[a,b]}|e(x)| = |e(b)| \approx |m\,b + q|. \tag{A.19}$$

Then, from (A.16) and (A.17) we can compute the error for the method (A) (choosing $\bar{\vartheta}_I = \vartheta_I$ for $I \in \{l, r\}$ and $j = 1$) and the error for the method (B) (choosing $\bar{\vartheta}_I = \tilde{\vartheta}_I$ for $I \in \{l, r\}$ and $j = 2$). Recalling that $\tilde{\vartheta}_l = (\vartheta_l + 1)/2$ and $\tilde{\vartheta}_r = \vartheta_r/2$, we obtain:

$$\alpha = \left|\frac{u''(a)\left(\frac{1}{6} - (1+\vartheta_l)(1-\vartheta_l)\right) + u'''(b)\left(\vartheta_r^2 - \frac{7}{6}\right)}{u''(a)\left(\frac{1}{6} - \vartheta_l(1-\vartheta_l)\right) + u'''(b)\left(\vartheta_r^2 - \frac{1}{6}\right)}\right|. \tag{A.20}$$

In order to compute the value α under grid refinement for a fixed problem, we should compute the expected value:

$$\bar{\alpha} = \int_{[0,1]^2} \alpha d\vartheta_l d\vartheta_r.$$

Even if the discretization error of the method (B) is four time greater than the discretization error of the method (A), the value (A.20) may be greater than 4, as we will see in the numerical tests of the next section.

A.4 Numerical results

In this section we compare the accuracy order and the absolute value of the error between iteration schemes (A.2)-(A.3) (that we call method (A)) and (A.2)-(A.5) (that we call method (B)). In order to avoid oscillation in the error, when we perform method (B) we always use the matrix (A.5), even if $\vartheta_l \geq 0.5$ or $\vartheta_r \leq 0.5$.

In all numerical tests we choose the exact solution u and the domain $[a, b] \subset [-1, 1]$, and from these we compute f, g_a and g_b.

Let $u^{(n)}$ be the numerical solution obtained at the iteration n. All numerical computations in the following numerical tests are performed with a very strict tolerance on the residue

$$\left\| f_h - L_h \, u^{(n)} \right\| \leq \epsilon \, \| f_h \|$$

with $\epsilon = 10^{-12}$, so that the discretization error is the dominant cause of error.

For each numerical test we also compute the ratio α, defined in (A.18).

A.4.1 Example 1

We choose:

$$u = \sin(2\pi x), \quad \Omega = [-0.5, 0.5 + 10^{-12}].$$

In this case we get $u''(a) = 0$ and, if we choose a grid with a number of node equal to a power of two, $\vartheta_l = 1$ and $\vartheta_r \approx 0$. Since $\vartheta_l = 1$, the second equation of (A.14) gives $e(a) = 0$ and then the assumption (A.19) is valid. Then, from (A.20) we obtain $\alpha = 7$. This value is confirmed by the numerical results of Fig. A.1 and Table A.1. In Fig. 5.6 we observe the behavior of the error for both methods (A) and (B).

A.4.2 Example 2

We choose:

$$u = \sin(2\pi x), \quad \Omega = [-0.843, 0.813].$$

In Fig. A.3 and in Table A.2 we report the errors for both methods (A) and (B) in L^1 and L^∞ norms and compute the respective values of α.

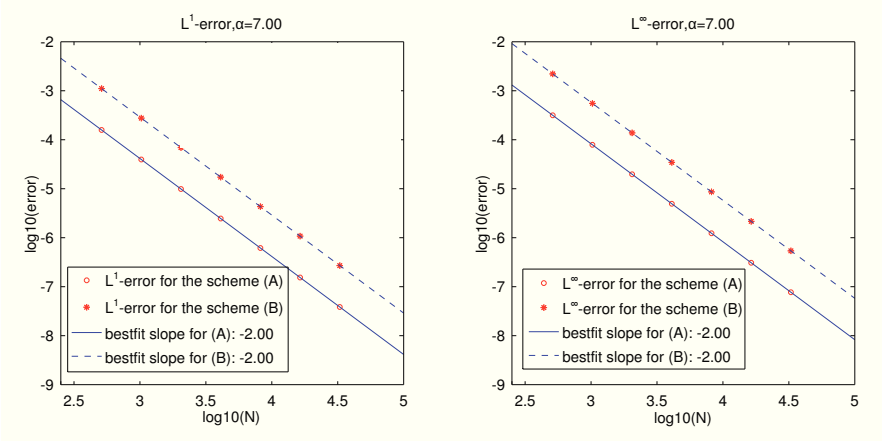


Fig. A.1: Example A.4.1. Errors obtained with methods (A) and (B) computed with L^1 and L^∞ norms and the respective best-fit lines. Left: L^1 norm, mean value $\alpha = 7.00$; Right: L^∞ norm, mean value $\alpha = 7.00$.

Table A.1: L^1 and L^∞ errors for methods (A) and (B) and respective values of α.

N	(A): $(\vartheta_l, \vartheta_r)$	(B): $(\tilde{\vartheta}_l, \tilde{\vartheta}_r)$	(A): L^1	(B): L^1	α
512	(1.00,0.00)	(1.00,0.00)	$1.58 \cdot 10^{-4}$	$1.11 \cdot 10^{-3}$	7.00
1024	(1.00,0.00)	(1.00,0.00)	$3.95 \cdot 10^{-5}$	$2.77 \cdot 10^{-4}$	7.00
2048	(1.00,0.00)	(1.00,0.00)	$9.87 \cdot 10^{-6}$	$6.91 \cdot 10^{-5}$	7.00
4096	(1.00,0.00)	(1.00,0.00)	$2.47 \cdot 10^{-6}$	$1.73 \cdot 10^{-5}$	7.00
8192	(1.00,0.00)	(1.00,0.00)	$6.16 \cdot 10^{-7}$	$4.31 \cdot 10^{-6}$	7.00
16384	(1.00,0.00)	(1.00,0.00)	$1.54 \cdot 10^{-7}$	$1.08 \cdot 10^{-6}$	7.00
32768	(1.00,0.00)	(1.00,0.00)	$3.85 \cdot 10^{-8}$	$2.69 \cdot 10^{-7}$	7.00
N	(A): $(\vartheta_l, \vartheta_r)$	(B): $(\tilde{\vartheta}_l, \tilde{\vartheta}_r)$	(A): L^∞	(B): L^∞	α
512	(1.00,0.00)	(1.00,0.00)	$3.15 \cdot 10^{-4}$	$2.21 \cdot 10^{-3}$	7.00
1024	(1.00,0.00)	(1.00,0.00)	$7.89 \cdot 10^{-5}$	$5.52 \cdot 10^{-4}$	7.00
2048	(1.00,0.00)	(1.00,0.00)	$1.97 \cdot 10^{-5}$	$1.38 \cdot 10^{-4}$	7.00
4096	(1.00,0.00)	(1.00,0.00)	$4.93 \cdot 10^{-6}$	$3.45 \cdot 10^{-5}$	7.00
8192	(1.00,0.00)	(1.00,0.00)	$1.23 \cdot 10^{-6}$	$8.62 \cdot 10^{-6}$	7.00
16384	(1.00,0.00)	(1.00,0.00)	$3.08 \cdot 10^{-7}$	$2.16 \cdot 10^{-6}$	7.00
32768	(1.00,0.00)	(1.00,0.00)	$7.70 \cdot 10^{-8}$	$5.39 \cdot 10^{-7}$	7.00

A.4.3 Example 3

We choose:

$$u = e^x, \quad \Omega = [-0.843, 0.813].$$

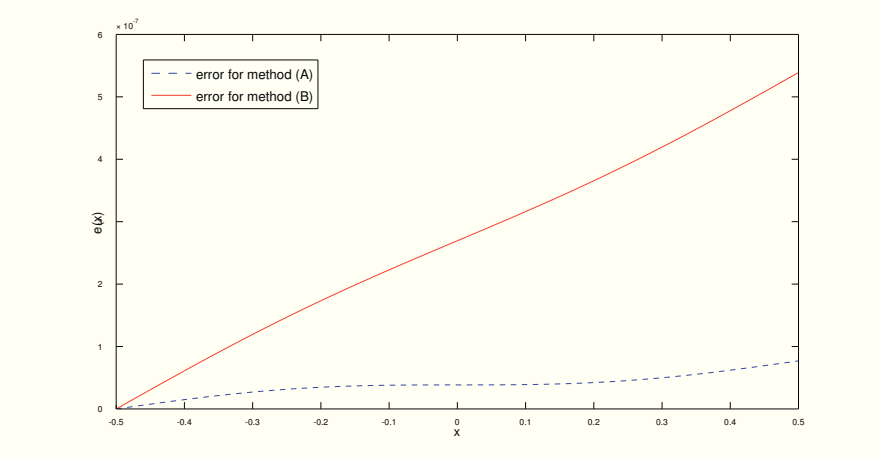


Fig. A.2: Example A.4.1. Behavior of the error for both methods (A) and (B).

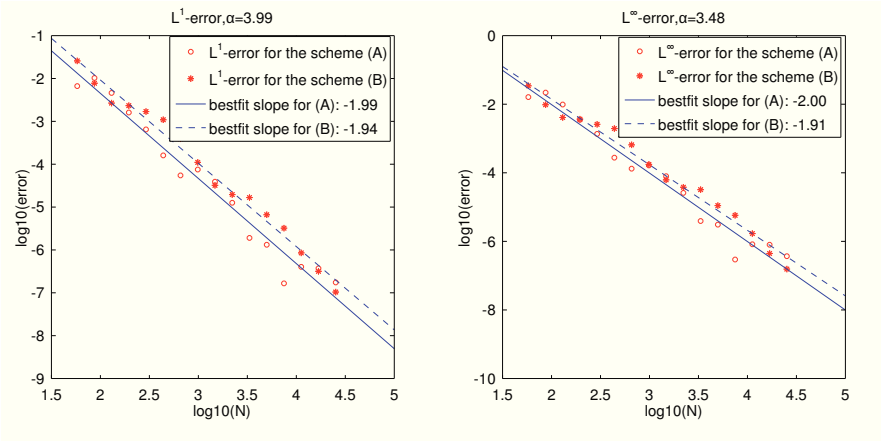


Fig. A.3: Example A.4.2. Errors obtained with methods (A) and (B) computed with L^1 and L^∞ norms and the respective best-fit lines. Left: L^1 norm, mean value $\alpha = 3.99$; Right: L^∞ norm, mean value $\alpha = 3.48$.

In Fig. A.4 and in Table A.3 we report the errors for both methods (A) and (B) in L^1 and L^∞ norms and compute the respective values of α.

A.4.4 Example 4

We choose:

$$u = e^{x^2}\sin(2\pi x), \quad \Omega = [-0.823, 0.787].$$

In Fig. A.4 and in Table A.3 we report the errors for both methods (A) and (B) in L^1 and L^∞ norms and compute the respective values of α.

Table A.2: Example A.4.2: L^1 and L^∞ errors for methods (A) and (B) and respective values of α.

N	(A): $(\vartheta_l, \vartheta_r)$	(B): $(\tilde{\vartheta}_l, \tilde{\vartheta}_r)$	(A): L^1	(B): L^1	α
58	(0.45,0.58)	(0.72,0.29)	$6.65 \cdot 10^{-3}$	$2.57 \cdot 10^{-2}$	3.87
87	(0.17,0.87)	(0.59,0.43)	$1.03 \cdot 10^{-2}$	$7.75 \cdot 10^{-3}$	0.75
130	(0.79,0.84)	(0.90,0.42)	$4.61 \cdot 10^{-3}$	$2.67 \cdot 10^{-3}$	0.58
195	(0.69,0.77)	(0.85,0.38)	$1.61 \cdot 10^{-3}$	$2.33 \cdot 10^{-3}$	1.45
292	(0.08,0.70)	(0.54,0.35)	$6.48 \cdot 10^{-4}$	$1.69 \cdot 10^{-3}$	2.61
438	(0.62,0.05)	(0.81,0.02)	$1.61 \cdot 10^{-4}$	$1.09 \cdot 10^{-3}$	6.78
657	(0.43,0.57)	(0.71,0.29)	$5.48 \cdot 10^{-5}$	$3.91 \cdot 10^{-4}$	7.14
986	(0.60,0.81)	(0.80,0.40)	$7.52 \cdot 10^{-5}$	$1.10 \cdot 10^{-4}$	1.47
1478	(0.98,0.81)	(0.99,0.40)	$3.91 \cdot 10^{-5}$	$3.22 \cdot 10^{-5}$	0.82
2217	(0.97,0.71)	(0.98,0.36)	$1.26 \cdot 10^{-5}$	$1.96 \cdot 10^{-5}$	1.55
3326	(0.91,0.02)	(0.95,0.01)	$1.91 \cdot 10^{-6}$	$1.67 \cdot 10^{-5}$	8.75
4988	(0.44,0.62)	(0.72,0.31)	$1.32 \cdot 10^{-6}$	$6.62 \cdot 10^{-6}$	5.01
7482	(0.66,0.43)	(0.83,0.22)	$1.66 \cdot 10^{-7}$	$3.23 \cdot 10^{-6}$	19.45
11223	(0.99,0.65)	(1.00,0.32)	$4.03 \cdot 10^{-7}$	$8.53 \cdot 10^{-7}$	2.12
16835	(0.45,0.93)	(0.73,0.46)	$3.72 \cdot 10^{-7}$	$3.17 \cdot 10^{-7}$	0.85
25252	(0.72,0.94)	(0.86,0.47)	$1.75 \cdot 10^{-7}$	$1.04 \cdot 10^{-7}$	0.60
N	(A): $(\vartheta_l, \vartheta_r)$	(B): $(\tilde{\vartheta}_l, \tilde{\vartheta}_r)$	(A): L^∞	(B): L^∞	α
58	(0.45,0.58)	(0.72,0.29)	$1.61 \cdot 10^{-2}$	$3.45 \cdot 10^{-2}$	2.13
87	(0.17,0.87)	(0.59,0.43)	$2.18 \cdot 10^{-2}$	$9.66 \cdot 10^{-3}$	0.44
130	(0.79,0.84)	(0.90,0.42)	$9.79 \cdot 10^{-3}$	$4.10 \cdot 10^{-3}$	0.42
195	(0.69,0.77)	(0.85,0.38)	$3.52 \cdot 10^{-3}$	$3.67 \cdot 10^{-3}$	1.04
292	(0.08,0.70)	(0.54,0.35)	$1.36 \cdot 10^{-3}$	$2.59 \cdot 10^{-3}$	1.90
438	(0.62,0.05)	(0.81,0.02)	$2.76 \cdot 10^{-4}$	$1.95 \cdot 10^{-3}$	7.07
657	(0.43,0.57)	(0.71,0.29)	$1.32 \cdot 10^{-4}$	$6.53 \cdot 10^{-4}$	4.95
986	(0.60,0.81)	(0.80,0.40)	$1.64 \cdot 10^{-4}$	$1.76 \cdot 10^{-4}$	1.07
1478	(0.98,0.81)	(0.99,0.40)	$7.95 \cdot 10^{-5}$	$6.22 \cdot 10^{-5}$	0.78
2217	(0.97,0.71)	(0.98,0.36)	$2.59 \cdot 10^{-5}$	$3.79 \cdot 10^{-5}$	1.46
3326	(0.91,0.02)	(0.95,0.01)	$3.92 \cdot 10^{-6}$	$3.22 \cdot 10^{-5}$	8.21
4988	(0.44,0.62)	(0.72,0.31)	$3.08 \cdot 10^{-6}$	$1.10 \cdot 10^{-5}$	3.59
7482	(0.66,0.43)	(0.83,0.22)	$2.95 \cdot 10^{-7}$	$5.77 \cdot 10^{-6}$	19.6
11223	(0.99,0.65)	(1.00,0.32)	$8.22 \cdot 10^{-7}$	$1.69 \cdot 10^{-6}$	2.05
16835	(0.45,0.93)	(0.73,0.46)	$7.98 \cdot 10^{-7}$	$4.44 \cdot 10^{-7}$	0.56
25252	(0.72,0.94)	(0.86,0.47)	$3.69 \cdot 10^{-7}$	$1.56 \cdot 10^{-7}$	0.42

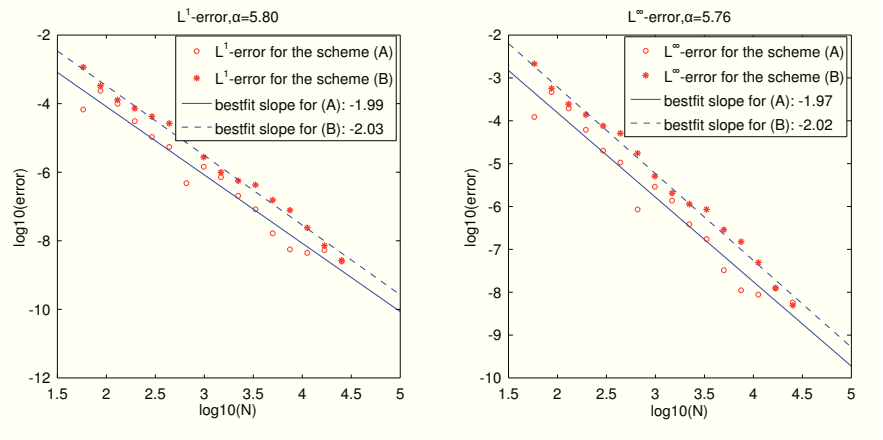


Fig. A.4: Example A.4.3. Errors obtained with methods (A) and (B) computed with L^1 and L^∞ norms and the respective best-fit lines. Left: L^1 norm, mean value $\alpha = 5.80$; Right: L^∞ norm, mean value $\alpha = 5.76$.

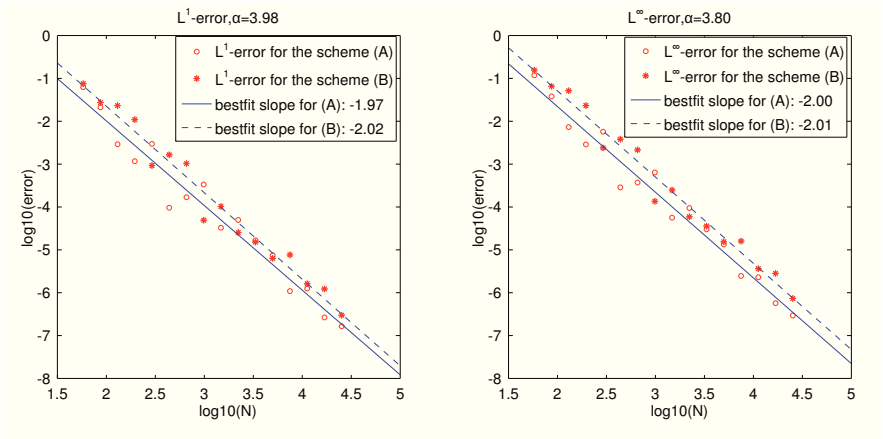


Fig. A.5: Example A.4.4. Errors obtained with methods (A) and (B) computed with L^1 and L^∞ norms and the respective best-fit lines. Left: L^1 norm, mean value $\alpha = 3.98$; Right: L^∞ norm, mean value $\alpha = 3.80$.

Table A.3: Example A.4.3: L^1 and L^∞ errors for methods (A) and (B) and respective values of α.

N	(A): $(\vartheta_l, \vartheta_r)$	(B): $(\tilde{\vartheta}_l, \tilde{\vartheta}_r)$	(A): L^1	(B): L^1	α
58	(0.45,0.58)	(0.72,0.29)	$6.75 \cdot 10^{-5}$	$1.16 \cdot 10^{-3}$	17.14
87	(0.17,0.87)	(0.59,0.43)	$2.37 \cdot 10^{-4}$	$3.32 \cdot 10^{-4}$	1.40
130	(0.79,0.84)	(0.90,0.42)	$9.87 \cdot 10^{-5}$	$1.28 \cdot 10^{-4}$	1.30
195	(0.69,0.77)	(0.85,0.38)	$3.09 \cdot 10^{-5}$	$7.33 \cdot 10^{-5}$	2.37
292	(0.08,0.70)	(0.54,0.35)	$1.07 \cdot 10^{-5}$	$4.20 \cdot 10^{-5}$	3.94
438	(0.62,0.05)	(0.81,0.02)	$5.43 \cdot 10^{-6}$	$2.66 \cdot 10^{-5}$	4.89
657	(0.43,0.57)	(0.71,0.29)	$4.79 \cdot 10^{-7}$	$9.40 \cdot 10^{-6}$	19.61
986	(0.60,0.81)	(0.80,0.40)	$1.44 \cdot 10^{-6}$	$2.76 \cdot 10^{-6}$	1.91
1478	(0.98,0.81)	(0.99,0.40)	$7.17 \cdot 10^{-7}$	$9.99 \cdot 10^{-7}$	1.39
2217	(0.97,0.71)	(0.98,0.36)	$2.06 \cdot 10^{-7}$	$5.60 \cdot 10^{-7}$	2.72
3326	(0.91,0.02)	(0.95,0.01)	$8.28 \cdot 10^{-8}$	$4.27 \cdot 10^{-7}$	5.16
4988	(0.44,0.62)	(0.72,0.31)	$1.65 \cdot 10^{-8}$	$1.54 \cdot 10^{-7}$	9.32
7482	(0.66,0.43)	(0.83,0.22)	$5.62 \cdot 10^{-9}$	$7.79 \cdot 10^{-8}$	13.85
11223	(0.99,0.65)	(1.00,0.32)	$4.44 \cdot 10^{-9}$	$2.39 \cdot 10^{-8}$	5.38
16835	(0.45,0.93)	(0.73,0.46)	$5.35 \cdot 10^{-9}$	$7.24 \cdot 10^{-9}$	1.35
25252	(0.72,0.94)	(0.86,0.47)	$2.49 \cdot 10^{-9}$	$2.68 \cdot 10^{-9}$	1.08
N	(A): $(\vartheta_l, \vartheta_r)$	(A): L^∞	(B): L^∞	α	
58	(0.45,0.58)	(0.72,0.29)	$1.23 \cdot 10^{-4}$	$2.13 \cdot 10^{-3}$	17.39
87	(0.17,0.87)	(0.59,0.43)	$4.67 \cdot 10^{-4}$	$5.72 \cdot 10^{-4}$	1.23
130	(0.79,0.84)	(0.90,0.42)	$1.94 \cdot 10^{-4}$	$2.46 \cdot 10^{-4}$	1.26
195	(0.69,0.77)	(0.85,0.38)	$6.15 \cdot 10^{-5}$	$1.38 \cdot 10^{-4}$	2.25
292	(0.08,0.70)	(0.54,0.35)	$2.01 \cdot 10^{-5}$	$7.58 \cdot 10^{-5}$	3.77
438	(0.62,0.05)	(0.81,0.02)	$1.06 \cdot 10^{-5}$	$5.12 \cdot 10^{-5}$	4.81
657	(0.43,0.57)	(0.71,0.29)	$8.52 \cdot 10^{-7}$	$1.75 \cdot 10^{-5}$	20.58
986	(0.60,0.81)	(0.80,0.40)	$2.90 \cdot 10^{-6}$	$5.11 \cdot 10^{-6}$	1.76
1478	(0.98,0.81)	(0.99,0.40)	$1.37 \cdot 10^{-6}$	$2.05 \cdot 10^{-6}$	1.50
2217	(0.97,0.71)	(0.98,0.36)	$3.84 \cdot 10^{-7}$	$1.14 \cdot 10^{-6}$	2.97
3326	(0.91,0.02)	(0.95,0.01)	$1.74 \cdot 10^{-7}$	$8.56 \cdot 10^{-7}$	4.93
4988	(0.44,0.62)	(0.72,0.31)	$3.26 \cdot 10^{-8}$	$2.86 \cdot 10^{-7}$	8.76
7482	(0.66,0.43)	(0.83,0.22)	$1.11 \cdot 10^{-8}$	$1.50 \cdot 10^{-7}$	13.52
11223	(0.99,0.65)	(1.00,0.32)	$8.79 \cdot 10^{-9}$	$4.90 \cdot 10^{-8}$	5.58
16835	(0.45,0.93)	(0.73,0.46)	$1.23 \cdot 10^{-8}$	$1.26 \cdot 10^{-8}$	1.03
25252	(0.72,0.94)	(0.86,0.47)	$5.73 \cdot 10^{-9}$	$4.97 \cdot 10^{-9}$	0.87

Table A.4: Example A.4.4: L^1 and L^∞ errors for methods (A) and (B) and respective values of α.

N	(A): $(\vartheta_l, \vartheta_r)$	(B): $(\tilde{\vartheta}_l, \tilde{\vartheta}_r)$	(A): L^1	(B): L^1	α
58	(0.87,0.82)	(0.93,0.41)	$6.24 \cdot 10^{-2}$	$7.50 \cdot 10^{-2}$	1.20
87	(0.30,0.73)	(0.65,0.37)	$2.12 \cdot 10^{-2}$	$2.74 \cdot 10^{-2}$	1.29
130	(0.49,0.15)	(0.75,0.08)	$2.91 \cdot 10^{-3}$	$2.32 \cdot 10^{-2}$	8.00
195	(0.74,0.23)	(0.87,0.12)	$1.17 \cdot 10^{-3}$	$1.09 \cdot 10^{-2}$	9.33
292	(0.16,0.90)	(0.58,0.45)	$2.97 \cdot 10^{-3}$	$9.26 \cdot 10^{-4}$	0.31
438	(0.24,0.35)	(0.62,0.18)	$9.61 \cdot 10^{-5}$	$1.65 \cdot 10^{-3}$	17.14
657	(0.86,0.03)	(0.93,0.01)	$1.69 \cdot 10^{-4}$	$1.03 \cdot 10^{-3}$	6.13
986	(0.74,0.99)	(0.87,0.50)	$3.35 \cdot 10^{-4}$	$4.91 \cdot 10^{-5}$	0.15
1478	(0.20,0.59)	(0.60,0.30)	$3.28 \cdot 10^{-5}$	$1.03 \cdot 10^{-4}$	3.13
2217	(0.80,0.89)	(0.90,0.44)	$4.98 \cdot 10^{-5}$	$2.53 \cdot 10^{-5}$	0.51
3326	(0.65,0.78)	(0.82,0.39)	$1.66 \cdot 10^{-5}$	$1.52 \cdot 10^{-5}$	0.91
4988	(0.56,0.78)	(0.78,0.39)	$7.43 \cdot 10^{-6}$	$6.34 \cdot 10^{-6}$	0.85
7482	(0.84,0.17)	(0.92,0.08)	$1.09 \cdot 10^{-6}$	$7.64 \cdot 10^{-6}$	7.01
11223	(0.76,0.75)	(0.88,0.38)	$1.25 \cdot 10^{-6}$	$1.61 \cdot 10^{-6}$	1.29
16835	(0.10,0.07)	(0.55,0.04)	$2.65 \cdot 10^{-7}$	$1.22 \cdot 10^{-6}$	4.62
25252	(0.20,0.66)	(0.60,0.33)	$1.65 \cdot 10^{-7}$	$2.99 \cdot 10^{-7}$	1.81
N	(A): $(\vartheta_l, \vartheta_r)$	(B): $(\tilde{\vartheta}_l, \tilde{\vartheta}_r)$	(A): L^∞	(B): L^∞	α
58	(0.87,0.82)	(0.93,0.41)	$1.18 \cdot 10^{-1}$	$1.56 \cdot 10^{-1}$	1.32
87	(0.30,0.73)	(0.65,0.37)	$3.81 \cdot 10^{-2}$	$6.55 \cdot 10^{-2}$	1.72
130	(0.49,0.15)	(0.75,0.08)	$7.31 \cdot 10^{-3}$	$5.16 \cdot 10^{-2}$	7.06
195	(0.74,0.23)	(0.87,0.12)	$2.88 \cdot 10^{-3}$	$2.33 \cdot 10^{-2}$	8.09
292	(0.16,0.90)	(0.58,0.45)	$5.70 \cdot 10^{-3}$	$2.39 \cdot 10^{-3}$	0.42
438	(0.24,0.35)	(0.62,0.18)	$2.87 \cdot 10^{-4}$	$3.83 \cdot 10^{-3}$	13.35
657	(0.86,0.03)	(0.93,0.01)	$3.72 \cdot 10^{-4}$	$2.15 \cdot 10^{-3}$	5.78
986	(0.74,0.99)	(0.87,0.50)	$6.40 \cdot 10^{-4}$	$1.36 \cdot 10^{-4}$	0.21
1478	(0.20,0.59)	(0.60,0.30)	$5.64 \cdot 10^{-5}$	$2.48 \cdot 10^{-4}$	4.40
2217	(0.80,0.89)	(0.90,0.44)	$9.46 \cdot 10^{-5}$	$5.90 \cdot 10^{-5}$	0.62
3326	(0.65,0.78)	(0.82,0.39)	$3.01 \cdot 10^{-5}$	$3.59 \cdot 10^{-5}$	1.19
4988	(0.56,0.78)	(0.78,0.39)	$1.33 \cdot 10^{-5}$	$1.54 \cdot 10^{-5}$	1.15
7482	(0.84,0.17)	(0.92,0.08)	$2.46 \cdot 10^{-6}$	$1.60 \cdot 10^{-5}$	6.50
11223	(0.76,0.75)	(0.88,0.38)	$2.29 \cdot 10^{-6}$	$3.63 \cdot 10^{-6}$	1.58
16835	(0.10,0.07)	(0.55,0.04)	$5.71 \cdot 10^{-7}$	$2.83 \cdot 10^{-6}$	4.96
25252	(0.20,0.66)	(0.60,0.33)	$2.93 \cdot 10^{-7}$	$7.34 \cdot 10^{-7}$	2.50

Appendix B

Diffusion Equation with moving interface

n this chapter we describe how to extend the method described in this thesis to solve the Diffusion Equation with discontinuous coefficients across a moving interface. The method is implemented in 1D, but we think it can be easily extended in higher dimensions. This chapter consists mainly of a work in progress.

Model problem 8 *Consider the model problem:*

$$\begin{aligned} \frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T &= \mu \, \Delta T + f \ \textit{in} \ \Omega = [a, b] && \text{(B.1)} \\ [\![T]\!] &= g_D \ \textit{on} \ \alpha \in \Omega && \text{(B.2)} \\ [\![\gamma \, T']\!] &= g_N \ \textit{on} \ \alpha \in \Omega && \text{(B.3)} \\ & && \text{(B.4)} \end{aligned}$$

in the unknown $T \colon \Omega \to \mathbb{R}$*, where* $\mathbf{u}(t, x)$ *is an assigned velocity, and the coefficient* γ *is discontinuous across* $\alpha(t)$*, which consists of a moving interface.*

Let $\Omega_1 = [a, \alpha]$ and $\Omega_2 = [\alpha, b]$. Eq. (B.1) is split in two equations:

$$\frac{\partial T_i}{\partial t} + \mathbf{u} \cdot \nabla T_i = \mu_i \, \Delta T_i + f_i \text{ in } \Omega_i, \quad i = 1, 2. \tag{B.5}$$

Now the coefficient μ_i is continuous in Ω_i and the problem can be solved with the usual technique of ghost cells described in Chapter 3. The moving interface can cause some troubles, especially when it crosses a grid point. The velocity u is defined in all Ω, and consists on the velocity by which the interface α moves in time:

$$\alpha(t + \Delta t) = \alpha(t) + \Delta t \, u(\alpha(t)).$$

Of course we can use a more accurate discretization of the interface evolution $\dot{\alpha} = u$. In higher dimension, the evolution will be performed by the level-set method.

B.1 Discretization

Let N_1 and N_1^{new} be as in Fig. B.1, where $\alpha = \alpha(t^n)$ and $\alpha^{new} = \alpha(t^{n+1})$.

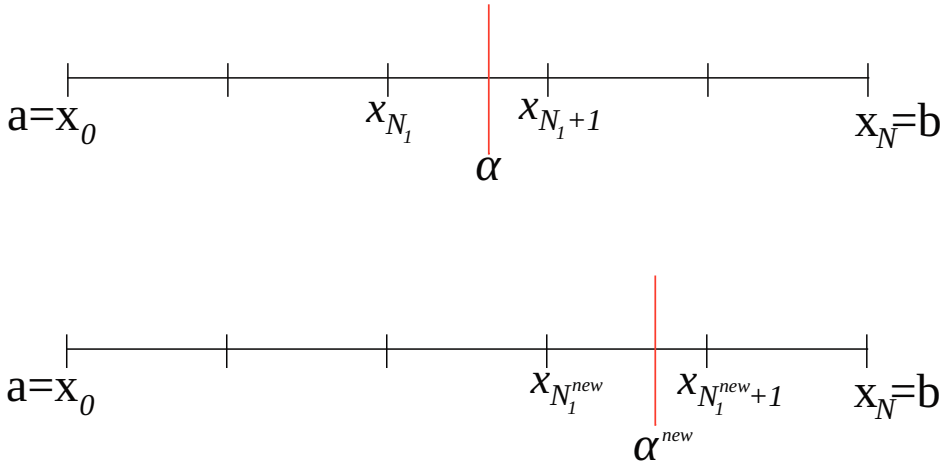


Fig. B.1: Discretization of the domain Ω at time t^n (top) and t^{n+1} (bottom).

Let us suppose that we know at time t^n the values $T_1(x_j)$ for $j = 0 \dots N_1 + 1$ and $T_2(x_j)$ for $j = N_1 \dots N$, and from this values we want to know at time t^{n+1} the values $T_1^{new}(x_j)$ for $j = 0 \dots N_1^{new} + 1$ and $T_2^{new}(x_j)$ for $j = N_1^{new} \dots N$. Let us build a linear system by writing an equation for each unknown $T_i^{new}(x_j)$. If $j = 0$ or $j = N$ the equation can be easily constructed from the boundary conditions in a and b. If $i = 1, \dots, N_1^{new} - 1$ we use Crank-Nicolson, i.e. we discretize Eq. (B.5) for $i = 1$ in $P \equiv x_j^{n+1/2}$ (see Fig. B.2):

$$\frac{T_1^{new}(x_j) - T_1(x_j)}{\Delta t} + \frac{\mathbf{u}^{new}(x_j) + \mathbf{u}(x_j)}{2} \cdot \frac{T_1^{new}(x_{j+1}) - T_1^{new}(x_{j-1}) + T_1(x_{j+1}) - T_1(x_{j-1})}{4\,\Delta x} = \frac{\mu_1^{new}(x_j) + \mu_1(x_j)}{4\,\Delta x^2} \left(\delta_j^2 T_1^{new} + \delta_j T_1\right) + \frac{f^{new}(x_j) + f(x_j)}{2} \quad \text{(B.6)}$$

where $\delta_j^2\, W = W(x_{j+1}) - 2\, W(x_j) + W(x_{j+1})$.

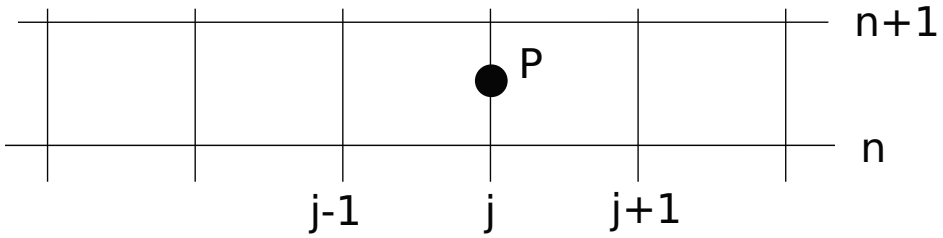


Fig. B.2: We discretize the diffusion equation by Crank-Nicolson if we are far from the interface, i.e. we discretize the equation in P.

If $j = N_1^{new}$ and $x_j < \alpha$ at time t^nm the equation is obtained as Eq. (B.6), else we have to find a different approach since we do not know the value $T_1(x_{j+1})$. Of course we can extrapolate T_1 far from the interface. In 1D is a trivial task, while in 2D we can follow for example the method described by Aslam in [12]. However, we here propose an alternative technique. In practice, we discretize (B.5) for $i = 1$ in $P \equiv x_{j+1/2}^{n+1/2}$ (see Fig. B.3).

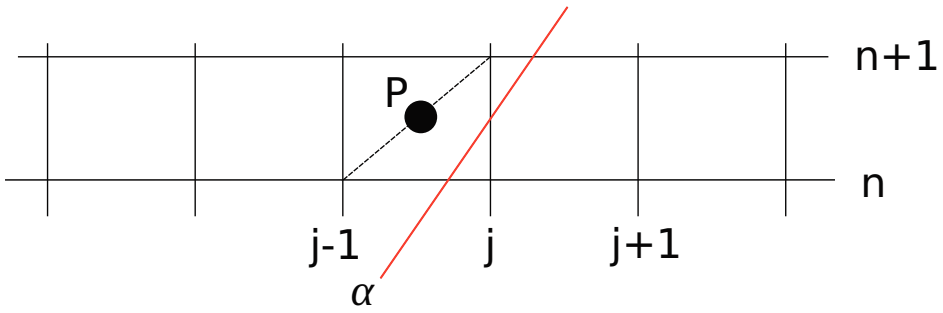


Fig. B.3: If the interface crosses a grid point from left to right, we discretize Eq. B.5 for $i = 1$ in P.

Let $\mathcal{L}_j W$ the quadratic interpolation of W in nodes x_{j-1}, x_j, x_{j+1}. Let us discretize every term of (B.5) for $i = 1$.

$$\frac{\partial T_1}{\partial t} \approx \frac{\mathcal{L}_j T_1^{new}(x_{j-1/2}) - \mathcal{L}_{j-1} T_1(x_{j-1/2})}{\Delta t}$$

$$\mathbf{u} \approx \frac{\mathbf{u}^{new}(x_j) + \mathbf{u}(x_{j-1})}{2}$$

$$\frac{\partial T_1}{\partial x} \approx \frac{1}{4\,\Delta x}\left(T_1^{new}(x_{j+1}) - T_1^{new}(x_{j-1}) + T_1(x_j) - T_1(x_{j-2})\right)$$

$$\mu_1 \approx \frac{\mu_1^{new}(x_j) + \mu_1(x_{j-1})}{2}$$

$$\frac{\partial^2 T_1}{\partial x^2} \approx \frac{1}{2\,\Delta x^2}\left(\delta_j^2 T_1^{new} + \delta_{j-1}^2 T_1\right)$$

$$f \approx \frac{f^{new}(x_j) + f(x_{j-1})}{2}.$$

The whole procedure can be repeated similarly for $T_2(x_j)$, for $j = N_1^{new} + 1, \ldots, N$. Only the ghost values $T_1(x_{N_1^{new}+1})$ and $T_2(x_{N_1^{new}})$ are left. They are found by discretizing the interface conditions (B.2) and (B.3):

$$\mathcal{L}_{j+1} T_2(\alpha^{new}) - \mathcal{L}_j T_1(\alpha^{new}) = g_D$$
$$\mu_2 \mathcal{L}'_{j+1} T_2(\alpha^{new}) - \mu_1 \mathcal{L}'_j T_1(\alpha^{new}) = g_N.$$

B.2 Multigrid approach

In this section we describe the multigrid solver, with particular attention to the relaxation scheme, since all the other components (transfer grid operators, ...) can be easily extended from those defined in Chapter 3.

The whole linear system can be resumed in matrix form:

$$A\,\mathbf{x} = \mathbf{b},$$

where now the unknown is $\mathbf{x}$. Following the observation at the beginning of Sec. 2.3, the iterative scheme derived from the associate fictitious time problem can be seen as a Richardson method with a suitable diagonal preconditioner. In details, the iteration scheme becomes:

$$\mathbf{x}^{m+1} = \mathbf{x}^m + P^{-1}\left(\mathbf{b} - M\mathbf{x}^m\right).$$

The diagonal entries p_k of P^{-1} are:

$$p_k = \frac{\Delta t}{1 + 2\,c_1} \text{ with } c_1 = \frac{\mu_1\,\Delta t}{2\,\Delta x^2} \tag{B.7}$$

for the inner equation of T_1,

$$\frac{\Delta t}{1 + 2\,c_2} \text{ with } c_2 = \frac{\mu_2\,\Delta t}{2\,\Delta x^2} \tag{B.8}$$

for the inner equation of T_2,

$$p_k = 1 \text{ and } p_k = \frac{\Delta x}{\max\{\mu_1, \mu_2\}}$$

for respectively the transmission conditions (B.2) and (B.3). The values (B.7) and (B.8) are obtained in order to zero out the related diagonal entry of the iteration matrix $B = I - P^{-1}M$, in such a way it appears to be a Jacobi iteration for inner equation. In order to have a proper multigrid smoother, we switch to a Gauss-Seidel version, that can be resumed, after some algebra, as follows:

$$\mathbf{x}^{n+1} = \mathbf{x}^n + (P - E)^{-1}(\mathbf{b} - M\,\mathbf{x}^n).$$

B.3 Alternative approaches

In this section we propose some alternative approaches that we can use when the interface cross a grid point.

Alternative 1: Fictitious velocity

Let us suppose we are in the case of Fig. B.3, i.e. the interface cross the grid point x_j from the left to the right. In order to write the linear equation for the unknown $T_1^{new}(x_j)$ (and only for this unknown) we introduce in the Eq. B.1 a fictitious velocity $\mathbf{v}$ in the following way (we have omitted for simplicity the convection velocity $\mathbf{u}$, i.e. we suppose $\mathbf{u} = 0$):

$$\frac{\partial T_1}{\partial t} + \mathbf{v}\frac{\partial T_1}{\partial x} = \mu\frac{\partial^2 T_1}{\partial x^2} + \mathbf{v}\frac{\partial T}{\partial x}. \tag{B.9}$$

We choose $\mathbf{v} = \Delta x/\Delta t$, in such a way the discretization of the left-hand side of (B.9) is:

$$\frac{dT_1}{dt} :\ = \frac{\partial T_1}{\partial t} + \mathbf{v}\frac{\partial T_1}{\partial x} \approx \frac{T_j^{n+1} - T_{j-1}^n}{\Delta t}.$$

Discretizing the other terms of (B.9) and collecting the terms in $n+1$ and n, we obtain (with an obvious notation of the stencil):

$$\frac{\frac{1}{4}[1 \quad 4 \quad -1]_j\, T_1^{new} - \frac{1}{4}[-1 \quad 4 \quad 1]_{j-1}\, T_1}{\Delta t} = \frac{\mu_1}{2\,\Delta x^2}\left(\delta_j^2 T_1^{new} + \delta_{j-1}^2 T_1\right). \tag{B.10}$$

Let us introduce the following notation: by $w|_j^n$ we intend a second order discretization of w in (x_j, t^n), i.e. $w|_j^n = w(x_j, t^n) + \mathcal{O}(h^2)$, with $h = \max\{\Delta t, \Delta x\}$. Analyzing the first-hand side of Eq. B.10, which indeed must be a consistent discretization of $\partial T_1/\partial t$ in P of Fig. B.3 (from (B.9)), we obtain:

$$\begin{aligned}
&\frac{\frac{1}{4}[1 \quad 4 \quad -1]_j\, T_1^{new} - \frac{1}{4}[-1 \quad 4 \quad 1]_{j-1}\, T_1}{\Delta t} \\
&= \frac{T_1^{new}(x_j) - T_1(x_j)}{\Delta t} + \frac{1}{\Delta t}\left([3/4 \quad -1 \quad 1/4]_{j-1}\, T_1 - [-1 \quad 0 \quad 1]_j\, T_1^{new}\right) \\
&= \left.\frac{\partial T_1}{\partial t}\right|_j^{n+1/2} - \frac{\Delta x}{2\,\Delta t}\left(\left.\frac{\partial T_1}{\partial x}\right|_j^{n+1} - \left.\frac{\partial T_1}{\partial x}\right|_j^{n}\right) \\
&= \left.\frac{\partial T_1}{\partial t}\right|_j^{n+1/2} - \frac{\Delta x}{2}\left.\frac{\partial^2 T_1}{\partial t \partial x}\right|_j^{n+1/2}.
\end{aligned}$$

In other words, we have obtained the Taylor expansion of $\partial T_1/\partial t$ centered in $(x_j, t^{n+1/2})$ and with spatial step $-\Delta/2$.

Alternative 2: Shifted stencil

By this name we intend the discretization described in Sec. B.1, namely:

$$\left.\frac{\partial T_1}{\partial t}\right|_{j-1/2}^{n+1/2} \approx \frac{\mathcal{L}_j T_1^{new}(x_{j-1/2}) - \mathcal{L}_{j-1} T_1(x_{j-1/2})}{\Delta\, t},$$

that can be resumed in the stencil notation:

$$\left.\frac{\partial T_1}{\partial t}\right|_{j-1/2}^{n+1/2} \approx \frac{\frac{1}{8}[3 \quad 6 \quad -1]_j\, T_1^{new} - \frac{1}{8}[-1 \quad 6 \quad 3]_{j-1}\, T_1}{\Delta t}.$$

Alternative 3: Least square

We can compute a quadratic interpolation $\tilde{T}_1$ of T_1 in the six-point stencil $(x_{j-2}, t^n), (x_{j-1}, t^n), (x_j, t^n), (x_{j-1}, t^{n+1}), (x_j, t^{n+1}), (x_{j+1}, t^{n+1})$, and then compute the discretization as:

$$\left.\frac{\partial T_1}{\partial t}\right|_{j-1/2}^{n+1/2} \approx \frac{\partial \tilde{T}_1}{\partial t}(x_{j-1/2}, t^{n+1/2}).$$

Although the linear system to find the quadratic interpolation is composed by 6 equations and 6 unknowns, such linear system is singular (observe that we can compute a suitable discretization of T_{tt} with such a stencil). However, if we solve this linear system by least squares, we obtain the following approximation:

$$\left.\frac{\partial T_1}{\partial t}\right|_{j-1/2}^{n+1/2} \approx \frac{\frac{1}{12}[7 \quad 4 \quad 1]_j\, T_1^{new} - \frac{1}{12}[1 \quad 4 \quad 7]_{j-1}\, T_1}{\Delta t}.$$

B.3.1 Comparison of the three discretizations

These three discretizations can be resumed as follows:

$$\left.\frac{\partial T_1}{\partial t}\right|_{j-1/2}^{n+1/2} \approx \frac{1}{\Delta t}\left(\frac{1}{24}[9 \quad 18 \quad -3]_j\, T_1^{new} + \beta[1 \quad -2 \quad 1]_j\, T_1^{new} \right.$$
$$\left. -\frac{1}{24}[-3 \quad 18 \quad 9]_{j-1}\, T_1 - \beta[1 \quad -2 \quad 1]_{j-1}\, T_1\right), \quad \text{(B.11)}$$

where $\beta = -3/24$ for the Alternative 1, $\beta = 0$ for the Alternative 2, $\beta = 5/24$ for the Alternative 3. Observe that (B.11) is second order accurate, independently on β, since we have:

$$\beta\frac{[1 \quad -2 \quad 1]_j\, T_1^{new} - [1 \quad -2 \quad 1]_{j-1}\, T_1}{\Delta t} = \mathcal{O}(h^2).$$

B.4 Numerical tests

We want to test the second order accuracy of the Alternative 2. We choose $[a, b] = [0, 1]$ and the following exact solutions:

$$T_1 = \sin(x + \frac{1}{t+1}), \quad T_2 = \cos(x + \frac{10}{t+1})$$

and a velocity

$$\mathbf{u} = \dot{\alpha}(t)\frac{(x-a)(x-b)}{(\alpha(t)-a)(\alpha(t)-b)},$$

where $\alpha(t) = 0.543 + 0.05\sin(t)$. In Figs. B.4 and B.5 we report the bestfit line of the errors in the L^∞ norm at the final time $t^{fin} = 0.25$ respectively for the alternatives 1 and 2.

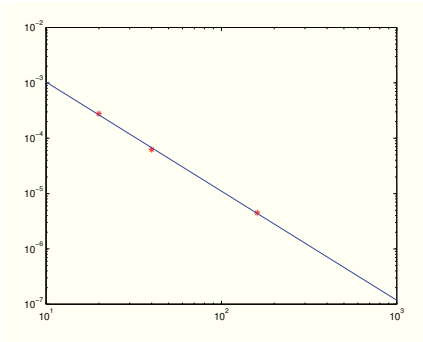


Fig. B.4: In the x-axis we report N, while in the y-axis we report the L^∞ error. The slope of the bestfit line is $s = -1.97$.

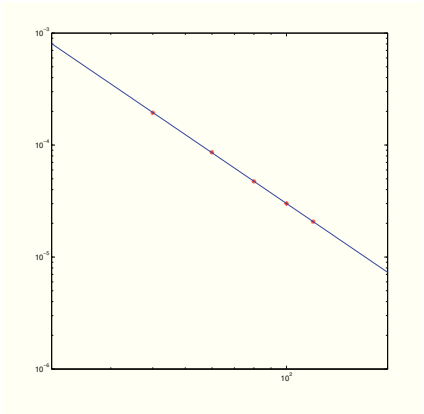


Fig. B.5: In the x-axis we report N, while in the y-axis we report the L^∞ error. The slope of the bestfit line is $s = -2.04$.

Conclusion and Work in progress

In this thesis we have proposed a numerical method to solve the elliptic equation in an arbitrary domain (described by a level-set function) with mixed boundary conditions and discontinuous coefficients. The problem is discretized on a Cartesian grid by a finite-difference ghost-cell method. The value in each ghost point is computed by a biquadratic extrapolation of the numerical solution in Upwind direction. Such a interpolation is computed in the projection of the ghost point on the boundary, and it may involve other ghost points. Therefore, the extrapolation equations for the ghost points are strongly coupled each other, and cannot be solved by a proper sub-system in order to eliminate the ghost equation from the system. This is the main strength and drawback of the discretization. It is a strength because it allows a simpler discretization, in particular for the Neumann boundary conditions, since it gives information in both Cartesian directions. It is a drawback because it adds additional unknown to the linear system. Such complication is overcame by adopting an iterative solver based on multigrid. First, we construct a good smoother for the multigrid, namely a convergent Gauss-Seidel-like relaxation scheme. Since a simple Gauss-Seidel applied to the whole linear system fails to converge, a suitable strategy has been adopted: the equations have been transformed into a fictitious time-dependent problem, whose time step represents an iterative parameter, which has been set up in order to guarantee the convergence. Such a time step is chosen also in such a way that the Gauss-Seidel iteration scheme for inner equations is recovered. We observe that the relaxation scheme by the fictitious time can be seen as a Richardson method with a diagonal preconditioner. Once the smoother is set up, the transfer (restriction and interpolation) operators must be built. The interpolation acts on the error, which is continuous across the boundary, therefore we chose the usual bilinear interpolation operator. On the other hand, the restriction operator acts on the residual,

which is supposed to be discontinuous across the boundary, since it refers to differential operators scaling with different powers of the spatial step h: the residual of inner equations is $\mathcal{O}(h^{-2})$, of Neumann boundary conditions is $\mathcal{O}(h^{-1})$, of Dirichlet boundary condition is $\mathcal{O}(1)$. Therefore, the restriction operator must be modified in the vicinity of the boundary, namely the inside coarse value is a weighted mean of neighboring fine values, but only of the ones belonging to the same side of the boundary, since ghost values refer to boundary conditions. For the ghost coarse value the argument is the same, provided to suitably extrapolate the residual outside the boundary constant along the normal direction. In summary, the restriction operator is performed separately for inner equations and for boundary conditions. The final convergence factor is close to the optimal one (i.e. the predicted one by the Local Fourier Analysis for inner relaxations), provided to add at each relaxation sweep on each grid level some extra-relaxations over boundary conditions and over inner equations close to the boundary (observe that the additional computational effort is negligible as the problem increases, since it is $\mathcal{O}(h^{-1})$). The boundary treatment of the iteration process has been compared with other techniques of the relevant literature, such as the Box and Kaczmarz relaxations (Example 2.7.3.2), showing that our method outperforms the Kaczmarz relaxation and performs nearly as well as the Box relaxation (which however results more expansive).

We must observe that, in the case of complex domains such as the flower-shaped domains, the convergence of the multigrid is observed only when the coarsest grid is fine enough to capture the curvature of the domain, as we can see in Example 2.7.3.5.

Other strengths of the whole method (discretization and multigrid) are the easy implementation in higher dimension, the second order accuracy in the solution and its gradient (and the method can be easily extended to higher accuracy), and the power of the solver when embedded in a moving boundary time dependent problem, such as those arising from applications, for instance, the incompressible Navier-Stokes equations. The motion of the domain can be easily handled by the level-set function, which implicitly describes the boundary, and then the computational effort of the grid regeneration at each time step is avoided.

The main motivation of the whole work is to embed the elliptic solver in the incompressible fluid dynamic framework, in order to model the motion of a fluid contained in a tank of arbitrary shape. In order to deal the two-phase flow problems, such as air-water, or the more challenging fluid-solid model, a discontinuous coefficient numerical method has been developed. In practice, the fluid-solid interaction would be described by a two-phase flow problem, where the solid is modeled as an highly viscous fluid with

respect to the real fluid. To this purpose, we extended the method to the case of discontinuous coefficients: we impose the continuity of the solution and of its flux across the interface separating the two fluids (indeed, the method can be applied to the more general case of non-homogeneous interface conditions). The discretization is obtained by adding ghost values close to the interface. In practice, in grid points close to the interface we define two values: one for the real solution in that grid point, and one for the ghost value related to the other side of the interface. The discrete equation of the ghost value for one sub-domain is obtained discretizing one of the jump conditions, and for the other sub-domain discretizing the other jump condition. The correlation between sub-domain and jump condition makes a choice, and finally we found a criterion based on the conditioning of the problem. The multigrid strategy adopted in the continuous coeffficient case can be extended to this case, provided to suitably extrapolate the residual of the jump conditions far from the interface (without overwrite real values) and to solve exactly the problem in the coarsest grid in place of simply relax few times (by just relaxing few times on the coarsest grid we do not regain the optimal convergence factor, especially for highly jump in the coefficient). We observed that the convergence factor is close to the optimal one, and non-depending on the jump in the coefficient nor on the size of the problem. Such features are recently investigated in the relevant literature.

Even though such applications to fluid dynamics have been performed, with satisfactory results, they are not reported in this thesis (we remind to another Ph.D. thesis [11] for more details).

In Chapter 5, the boundary/interface discretization has been embedded in the framework of the Adaptive Mesh refinement, in order to consider problems with small scale details, and better control the computational cost in terms of memory storage.

Work in progress

Let us describe the work in progress we have in mind, grouping them according to the subject.

Adaptive Mesh refinement

This future work is intended in collaboration with the research group of F. Gibou of University of Santa Barbara, California.

- Up to now, we have studied only the accuracy of the discretization in an adaptive Cartesian grid, solving the resulting (non-symmetric) linear

system with a (probably not the most efficient) direct solver. However, we recognize that future work will need to focus on the design of efficient multigrid solvers for our linear systems.

- Up to now, the refinement process is performed only close to the interface/boundary and depends only on the distance from it. In future works we aim to extend the method to more general criteria, for instance based on the solution (in order to homogenize the discretization error for all cells) or, more challenging, based on the curvature of the interface/boundary, in order to capture small scale effects arising only in some parts of the interface/boundary.

- The Poisson-Boltzmann/Poisson equation for biomolecules where one needs a method for the Poisson equation with jump conditions is currently underway.

Improvement of the method for particular cases

- In case of multiple interfaces, some problem may occur when two or more interfaces are very close each other. Let us think at the case of a drop falling into a still water: when the drop is very close to the water, some ghost points may refer at the same time to two inner grid points, one inside the drop, and one inside the rest of water. In such cases, the ghost-method described in this thesis may fail to gain the second order accuracy. One possible solution is to define two ghost values in the same grid points, referred to two different areas of the fluid (the drop and the rest of water).

- In the case of anisotropic operators, we have observed (Sec. 2.4.3) that the convergence of the relaxation scheme is not guaranteed when a strong anisotropy occurs. The reason is that the stencil we use is not in Upwind direction with respect to the co-normal derivative, but it is in Upwind direction with the normal derivative. However, taking a stencil in Upwind direction with respect the co-normal derivative is not always allowed, since in most cases the grid points of such a stencil are not unknowns of the linear system. A possible strategy can be obtained by choosing three different stencils for the reconstruction respectively of the solution (Dirichlet), the x-derivative and the y-derivative (Neumann). Since it is not needed to use a proper Upwind stencil for the reconstruction of the solution, we can just use the usual Upwind stencil with respect to the normal direction. While, for the gradient reconstruction, we need a co-normal Upwind stencil, which we think can

be always obtained, providing to use two different stencils (for x- and y- directions) possibly reduced, but still maintaining the whole second order accuracy.

- In presence of sharp-edged domains (Sec. 1.3.4), the accuracy slightly degrades when the corner is very small. Applying a strategy close to the one described in the previous point, namely using different stencil for the reconstructions, we guess that we can obtain a more stable second order discretization.

- In presence of convex zones of the domain, sometimes can be very difficult to reconstruct the correct normal derivative on the boundary by the signed-distance function ϕ, since it may presents a discontinuity in the gradient. In such cases we think to use a WENO reconstruction of the gradient of ϕ, namely we properly choose the stencil in such a way it does not crosses the discontinuity in $\nabla\phi$.

Analytical studies

Even though we developed a purely practical multigrid solver, we hope in future to provide the method with an analytical study of all multigrid components. The first step will be the convergence proof of the relaxation scheme for the second order accurate method, while the proof of the first order method has been presented in Sec. 2.3.

- The convergence proof for the second order accurate discretization is not a trivial task, since the resulting linear system is not diagonally dominant. Starting from the easier 1D case, we gave an alternative discretization of the method in order to prove the convergence. However, such idea is very simple and still not satisfactory in our opinion. We described such a technique in Appendix A.

- For interface conditions, we observed that the CFL condition for the jump in the flux must take into account the value of the coefficient of both sides of the interface (Sec. 3.1.3). We think we can give in future a suitable explanation of the condition (3.29) by the Fourier modes analysis.

Implementation

Such work in progress will be carried out in near future.

- We are extending all the methods presented in this thesis to the 3D case.

- The code has being parallelized during a 13-week study period in Bordeaux, France, among the research group of A. Iollo, and with the HPC-Europa2 fellowship.

Applications

- The extension of the method to the more general case of a convection-diffusion equation with discontinuous coefficient across a moving interface is currently underway:

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \mu \, \Delta T + f$$

  in the unknown $T \colon \Omega \to \mathbb{R}$, where the coefficient μ is discontinuous across a lower dimensional interface $\Gamma(t) \subseteq \Omega$ which evolves in time. The time derivative is discretized by Crank-Nicolson. When the interface cross a grid point, the two 5-point stencils at time t^n and t^{n+1} must be suitably modified, since some grid values may be not available. Several kinds of strategies are currently under investigation, all providing encouraging results. Some 1D results are described in Appendix B.

- The Stefan problem models the phase change of a medium describing its temperature distribution. It is important to extend the method proposed in this thesis to a Stefan-type problem in our research group since one of the main applications is the crust-formation embedded in the lava-crust interaction. The Stefan problem, combined with the incompressible fluid dynamic framework, should be able to describe this phenomena.

Bibliography

[1]

[2] L. Adams and T. P. Chartier. New geometric immersed interface multigrid solvers. *SIAM Journal of Scientific Computing*, 25:1516–1533, 2004.

[3] L. Adams and T. P. Chartier. A comparison of algebraic multigrid and geometric immersed interface multigrid methods for interface problems. *SIAM Journal of Scientific Computing*, 26:762–784, 2005.

[4] L. Adams and Z. Li. The immersed interface/multigrid methods for interface problems. *Journal of Scientific Computing*, 24:463–479, 2002.

[5] M. J. Aftosmis, M. J. Berger, and J. E. Melton. *In CRC Handbook of Mesh Generation (Contributed Chapter)*, chapter Adaptive Cartesian mesh generation, pages 199–208.

[6] R. E. Alcouffe, A. Brandt, J. Dendy, J. E., and J. W. Painter. The multigrid method for the diffusion equation with strongly discontinuous coefficients. *Journal on Scientific and Statistical Computing*, 2:430–454, 1981.

[7] A. Almgren. *A Fast Adaptive Vortex Method Using Local Corrections.* PhD thesis, University of California, Berkeley, 1991.

[8] A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome. A conservative adaptive projection method for the variable density incompressible navier-stokes equations. *J. Comput. Phys.*, 142:1–46, 1998.

[9] A. Almgren, R. Buttke, and P. Colella. A fast adaptive vortex method in three dimensions. *J. Comput. Phys.*, 113:177–200, 1994.

[10] P. Angot, C.-H. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numer. Math.*, 81, 1999.

[11] V. Artale. *Level-Set Ghost Fluid Methods for Free Boundary Problems in Incompressible Euler and Navier-Stokes Equations.* PhD thesis, University of Catania, Italy, 2011.

[12] T. D. Aslam. A partial differential equation approach to multidimensional extrapolation. *Journal of Computational Physics*, 193:349–355, 2003.

[13] I. Babuška. The finite element method for elliptic equations with discontinuous coefficients. *Computing*, 5:207–213, 1970.

[14] I. Babuška, J. E. Flaherty, W. D. Henshaw, J. E. Hopcroft, J. E. Oliger, , and T. Tezduyar. *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations.* Springer Verlag, Berlin, 1995.

[15] N. Bachvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Comput. Math. Phys.*, 6:101–135, 1966.

[16] G. Bao, G. Wei, and S. Zhao. Numerical solution of the Helmholtz equation with high wave numbers. *J. Numer. Methods Engng.*, 59:389–408, 2004.

[17] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 92:484–512, 1984.

[18] J. Bramble and J. King. A finite element method for interface problems in domains with smooth boundaries and interfaces. *Adv. Comput. Math.*, 6:109–138, 1996.

[19] J. Bramble, J. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comp.*, 55:1–22, 1990.

[20] J. H. Bramble and B. E. Hubbard. Approximation of solutions of mixed boundary value problems for Poisson's equation by finite differences. *J. Assoc. Comput. Mach.*, 12:114–123, 1965.

[21] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31:333–390, 1977.

[22] A. Brandt. Guide to multigrid developments. In W. Hackbusch and U. Trottenberg, editors, *Multigrid Methods, Lectures Notes in Mathematics*, volume 960, pages 220–312, Berlin, 1982. Springer.

[23] A. Brandt. *Multigrid*, chapter Recent Developments in Multigrid Efficiency in Computational Fluid Dynamics, pages 573–589. Academic Press, 2001. Authors of the book: U.Trottemberg, C.W. Oosterlee, A. Schuller.

[24] A. Brandt. *Multigrid techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition*. SIAM, St. Augustin, 2011.

[25] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Technical Report, Inst. for Computational Studies, Fort Collins, CO, 1982.

[26] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, and J. Ruge. Algebraic multigrid based on element interpolation (AMGe). Technical Report UCRL-JC-131752, Lawrence Livermore National Laboratory, 1988.

[27] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, 2000.

[28] R. E. Caflisch, M. F. Gyure, B. Merriman, S. J. Osher, C. Ratsch, D. D. Vvedensky, and J. J. Zinck. Island dynamics and the level set method for epitaxial growth. *Applied Mathematics Letters*, 4:13–22, 1999.

[29] T. Chan, S. Go, and L. Zikatanov. Lecture Notes on Multilevel Methods for Elliptic Problems on Unstructured Grids. In *Lecture Course 28th Computational Fluid Dynamics*, von Karman Institute for Fluid Dynamics, Belgium, March 37 1997.

[30] T. Chan, J. Xu, and L. Zikatanov. An agglomeration multigrid method for unstructured grids. Technical Report CAM 98-8, Department of Mathematics, UCLA, Los Angeles, CA, February 1998.

[31] F. Chantalat, C.-H. Bruneau, C. Galusinski, and A. Iollo. Level-set, penalization and cartesian meshes: A paradigm for inverse problems and optimal design. *Journal of Computational Physics*, 228:6291–6315, 2009.

[32] H. Chen, C. Min, and F. Gibou. A supra-convergent finite difference scheme for the Poisson and heat equations on irregular domains and non-graded adaptive Cartesian grids. *Journal of Scientific Computing*, 31:19–60, 2007.

[33] A. Coco, F. Gibou, and G. Russo. Adaptive solvers for Poisson problems with discontinuous coefficients on Cartesian grids. Submitted.

[34] A. Coco and G. Russo. A fictitious time method for the solution of Poisson equation in an arbitrary domain embedded in a square grid. *Journal of Computation Physics*. Under revision.

[35] A. Coco and G. Russo. Multigrid approach for Poisson's equation with mixed boundary condition in an arbitrary domain. Submitted. Preprint available in http://arxiv.org/pdf/1111.0983.

[36] A. Coco and G. Russo. Second order multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface, I: one dimensional problems. *Numerical Mathematics: Theory, Methods and Applications*. In press. Pre-print available in http://arxiv.org/pdf/1111.1167.

[37] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM J. Res. Develop.*, 11:215–234, 1967.

[38] J. Dendy. Black box multigrid. *Journal of Computational Physics*, 48:366–386, 1982.

[39] J. Donea. An arbitrary Lagrangian-Eulerian finite element method for transient fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33:689–723, 1982.

[40] A. du Chéné, C. Min, and F. Gibou. Second-Order Accurate Computation of Curvatures in a Level Set Framework Using Novel High Order Reinitialization Schemes. *Journal of Scientific Computing archive*, 35:114–131, 2008.

[41] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (The Ghost Fluid Method). *Journal of Computational Physics*, 152:457–492, 1999.

[42] R. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Comput. Math. Phys.*, 1:1092–1096, 1961.

[43] R. Fedorenko. The speed of convergence of one iterative process. *USSR Comput. Math. Phys.*, 4:227–235, 1964.

[44] L. Formaggia and F. Nobile. Stability analysis of second-order time accurate schemes for ALE-FEM. *Computer Methods in Applied Mechanics and Engineering*, 193:4097–4116, 2004.

[45] F. Gibou, L. Chen, D. Nguyen, and S. Banerjee. A level set based sharp interface method for the multiphase incompressible Navier-Stokes equations with phase change. *J. Comput. Phys.*, 2:536–555, 2007.

[46] F. Gibou and R. Fedkiw. A second-order-accurate symmetric discratization of the poisson equation on irregular domains. *Journal of Computational Physics*, 176:205–227, 2002.

[47] F. Gibou and R. Fedkiw. A fourth order accurate discretization for the laplace and heat equations on arbitary domains, with applications to the stefan problem. *Journal of Computational Physics*, 202:577–601, 2005.

[48] F. Gibou, R. Fedkiw, and R. Caflisch. A Level Set Approach for the Numerical Simulation of Dendritic Growth. *J. Sci. Comput.*, 19:183–199, 2003.

[49] F. Gibou, C. Min, and H. Ceniceros. *Free boundary problems, Internat. Ser. Numer. Math.*, chapter Finite difference schemes for incompressible flows on fully adaptive grids, pages 199–208. Birkhuser, Basel, 2007.

[50] F. Gibou, C. Min, and H. D. Ceniceros. Non-graded adaptive grid approaches to the incompressible Navier-Stokes equations. *FDMP Fluid Dyn. Mater. Process.*, 3:37–48, 2007.

[51] R. Glowinski, T. W. Pan, T. I. Hesla, and D. D. Joseph. A distributed Lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25:755–794, 1999.

[52] I. G. Graham and M. J. Hagger. Unstructured additive Schwarzconjugate gradient method for elliptic problems with highly discontinuous coefficients. *SIAM J. Sci. Comput.*, 20:2041–2066, 1999.

[53] L. Greengard and J. Y. Lee. A direct adaptive Poisson solver of arbitrary order accuracy. *J. Comput. Phys.*, 125:415–424, 1996.

[54] W. Hackbusch. A fast iterative method solving Poisson's equation in a general region. In R. Bulirsch, R. Grigorie, and J. Schroder, editors, *Lecture Notes in Mathematics*, volume 631, pages 51–62, Oberwolfach, July 1976. Springer, Berlin, 1978.

[55] W. Hackbusch. *Multi-grid methods and applications.* Springer, 1985.

[56] W. Hackbusch. *Elliptic Differential Equations: Theory and Numerical Treatment.* Springer, 2003.

[57] S. Hou and X. Liu. A numerical method for solving variable coefficient elliptic equation with interfaces. *Journal of Computational Physics*, 202:411–445, 2005.

[58] S. Hou, W. Wang, and L. Wang. Numerical method for solving matrix coefficient elliptic equation with sharp-edged interfaces. *Journal of Computational Physics*, 229:7162–7179, 2010.

[59] J. J. E. Dendy. Black Box Multigrid. *Journal of Computational Physics*, 48:366–386, 1982.

[60] H. Johansen and P. Colella. A Cartesian Grid Embedded Boundary Method for Poisson Equation on Irregular Domains. *Journal of Computational Physics*, 147:60–85, 1998.

[61] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Cambridge University Press, New York, NY, 1987.

[62] J. Jones and S. McCormick. Parallel multigrid methods. In D. Keyes, A. Sameh, and V. Venkatakrishnan, editors, *Parallel Numerical Algorithms*, Kluwer, Dordrecht, 1997. NASA=LaRC Interdisciplinary Series in Science and Engineering.

[63] J. D. Jr. Black box multigrid for nonsymmetric problems. *Appl. Math. Comput.*, 13:261283, 1983.

[64] S. Kaczmarz. Angenherte Auflsung von Systemen linearer Gleichungen. *Bulletin International de l'Acadmie Polonaise des Sciences et des Lettres. Classe des Sciences Mathmatiques et Naturelles. Srie A, Sciences Mathmatiques*, 35:355–357, 1937.

[65] B. Koobus, M. H. Lallemand, and A. Derieux. Unstructured Volume-Agglomeration MG: Solution of the Poisson Equation. Technical Report 1946, INRIA, Sophia Antipolis, France, 1993.

[66] H. O. Kreiss, H.-O. Manteuffel, T. A. Schwartz, B. Wendroff, and J. White, A. B. Supra-convergent schemes on irregular grids. *Math. Comput.*, 47:537–554, 1986.

[67] R. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 31:1019–1044, 1994.

[68] Z. Li. A fast iterative algorithm for elliptic interface problems. *SIAM Journal of Numerical Analysis*, 35:230–254, 1998.

[69] Z. Li and K. Ito. Maximum principle preserving schemes for interface problems with discontinuous coefficients. *SIAM Journal of Scientific Computing*, 23:339–361, 2001.

[70] X. Liu, R. Fedkiw, and M. Kang. A Boundary Condition Capturing Method for Poissons Equation on Irregular Domains. *Journal of Computational Physics*, 160:151–178, 2000.

[71] F. Losasso, R. Fedkiw, , and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *J. Comput. Phys.*, 35:995–1010, 2006.

[72] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *SIGGRAPH 2004, ACM TOG*, 23:457–462, 2004.

[73] J. Mandel, M. Brezina, and P. Vaněk. Energy optimization of algebraic multigrid bases. *Computing*, 62:205–228, 1999.

[74] T. Manteuffel and A. White. The numerical solution of second-order boundary value problems on nonuniform meshes. *Math. Comput.*, 47:511–535, 1986.

[75] A. Mayo. The fast solution of Poisson's and the biharmonic equations on irregular regions. *SIAM J. Numer. Anal.*, 21:285–299, 1984.

[76] P. McCorquodale, P. Colella, D. Grote, and J.-L. Vay. A node-centered local refinement algorithm for poisson's equation in complex geometries. *Journal of Computational Physics*, 201:34–60, 2004.

[77] C. Min and F. Gibou. A second order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids. *J. Comput. Phys.*, 219:912–929, 2006.

[78] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive cartesian grids. *Journal of Computational Physics*, 225:300–321, 2007.

[79] C. Min, F. Gibou, and H. D. Ceniceros. A supra-convergent finite difference scheme for the variable coefcient poisson equation on non-graded grids. *Journal of Computational Physics*, 218:123–140, 2006.

[80] G. Naldi, L. Pareschi, and G. Russo. *Introduzione al Calcolo Scientifico: metodi e applicazioni con Matlab* . McGraw-Hill, 2001.

[81] Y. T. Ng, H. Chen, C. Min, and F. Gibou. Guidelines for Poisson solvers on irregular domains with Dirichlet boundary conditions using the ghost fluid method. *J. Sci. Comput.*, 41:300–320, 2009.

[82] Y. T. Ng, C. Min, and F. Gibou. An efficient fluid-solid coupling algorithm for single-phase flows. *J. Comput. Phys.*, 228:8807–8829, 2009.

[83] R. Nicolaides. On multiple grid and related techniques for solving discrete elliptic systems. *J. Comput. Phys.*, 19:418–431, 1975.

[84] M. Oevermann and R. Klein. A Cartesian grid finite volume method for elliptic equations with variable coefficients and embedded interfaces. *Journal of Computational Physics*, 219:749–769, 2006.

[85] M. Oevermann, C. Scharfenberg, and R. Klein. A sharp interface finite volume method for elliptic equations on Cartesian grids. *Journal of Computational Physics*, 228:5184–5206, 2009.

[86] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag New York, Applied Mathematical Sciences, 2002.

[87] J. Papac, F. Gibou, and C. Ratsch. Efficient Symmetric Discretization for the Poisson, Heat and Stefan-Type Problems with Robin Boundary Conditions. *Journal of Computational Physics*, 229:875–889, 2010.

[88] C. S. Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25:220–252, 1977.

[89] S. Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *J. Comput. Phys.*, 190:572–600, 2003.

[90] A. Quarteroni and R. Sacco. *Numerical approximation of partial differential equations*. Springer, 1994.

[91] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*. Springer, 2000.

[92] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Numerical Mathematics and Scientific Computation, 1999.

[93] A. Reusken. Multigrid with matrix-dependent transfer operators for a singular perturbation problem. *Computing*, 50:199–211, 1993.

[94] J. W. Ruge and Stüben. *Multigrid methods*, chapter Algebraic multigrid, pages 73–130. SIAM, Philadelphia, 1987.

[95] G. Russo and P. Smereka. A remark on computing distance functions. *Journal of Computational Physics*, 163:51–67, 2000.

[96] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, New York., 1989.

[97] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS.* Addison-Wesley, New York., 1990.

[98] A. Sarthou, S. Vincent, J. Caltagirone, and P. Angot. Eulerian-Lagrangian grid coupling and penalty methods for the simulation of multiphase flows interacting with complex objects. *International Journal for Numerical Methods in Fluids*, 00:1–6, 2007.

[99] A. Schmidt. Computation of three dimensional dendrites with finite elements. *Journal of Computational Physics*, 125:293–312, 1996.

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

[101] Y. Shapira. *Matrix-Based Multigrid: Theory and Applications. Second Edition*, volume 2 of *Numerical Methods and Algorithms*. Springer, New York, 2008.

[102] G. H. Shortley and R. Weller. The numerical solution of laplace's equation. *J. Appl. Phys.*, 9:334–348, 1938.

[103] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible 2-phase flow. *Journal of Computational Physics*, 114:146–159, 1994.

[104] W. W. Tony F. Chan. Robust multigrid methods for nonsmooth coefficient elliptic linear systems. *Journal of Computational and Applied Mathematics*, 123:323–352, 2000.

[105] U.Trottemberg, C. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2000.

[106] J. W. L. Wan and X.-D. Liu. A boundary condition-capturing multigrid approach to irregular boundary problems. *Journal of Scientific Computing*, 25:1982–2003, 2004.

[107] W. L. Wan. Interface preserving coarsening multigrid for elliptic problems with highly discontinuous coefficients. *Numer. Linear Algebra Appl.*, 7:727–741, 2000.

[108] W. L. Wan, T. F. Chan, and B. Smith. An energy-minimizing interpolation for robust multi-grid methods. *SIAM J. Sci. Comput.*, 21:1632–1649, 2000.

[109] D. Young, R. Melvin, M. Bieterman, F. Johnson, S. Samant, and J. Bussoletti. A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics. *J. Comput. Phys.*, 92:1–66, 1991.

[110] S. Yu, Y. Zhou, and G. Wei. Matched Interface and Boundary (MIB) method for elliptic problems with sharp-edged interfaces. *Journal of Computational Physics*, 224:729–756, 2007.

[111] P. M. D. Zeeuw. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *J. Comput. Appl. Math.*, 33:1–27, 1990.