Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface

Armando Coco, Giovanni Russo

January 6, 2018

Abstract

In this paper we propose a second-order accurate numerical method to solve elliptic problems with discontinuous coefficients (with general non homogeneous jumps in the solution and its gradient) in 2D and 3D. The method consists of a finite-difference method on a Cartesian grid in which complex geometries (boundaries and interfaces) are embedded, and is second order accurate in the solution and the gradient itself. In order to avoid the drop in accuracy caused by the discontinuity of the coefficients across the interface, two numerical values are assigned on grid points that are close to the interface: a real value, that represents the numerical solution on that grid point, and a ghost value, that represents the numerical solution extrapolated from the other side of the interface, obtained by enforcing the assigned non-homogeneous jump conditions on the solution and its flux. The method is also extended to the case of matrix coefficient. The linear system arising from the discretization is solved by an efficient multigrid approach. Unlike the 1D case, grid points are not necessarily aligned with the normal derivative and therefore suitable stencils must be chosen to discretize interface conditions in order to achieve second order accuracy in the solution and its gradient. A proper treatment of the interface conditions will allow the multigrid to attain the optimal convergence factor, comparable with the one obtained by Local Fourier Analysis for rectangular domains. The method is robust enough to handle large jump in the coefficients: order of accuracy, monotonicity of the errors and good convergence factor are maintained by the scheme.

1 Introduction

Elliptic equations with discontinuous coefficients arise from the mathematical modelling of a large number of real-life applications. Examples include the steady-state solution of diffusion problems, for instance in the context of solidification processes of materials with different diffusion coefficients across a complex interface [64]; the Poisson equation arising from the projection method for incompressible two-phase fluids with different physical characteristics [59]; the study of electrostatic phenomena such as those encountered in the simulation of biomolecules' electric potential [42], and many more [40, 13]. All these problems may be characterized by complex moving interfaces across which the jump of the solution and its flux must be prescribed for well-posedness. While for interfaces aligned with the grid lines the discretization results straightforward, for complex interfaces it is necessary to adopt a suitable numerical approach.

Among the different approaches that have been proposed to numerically solve this problem, interface-fitted grid methods such as those based on the finite element methods (FEM) [6, 7, 33, 31, 19, 27, 17] are computationally expensive especially for complex moving geometries, since at each time step a new mesh fitting the moving interface must be generated. The computational burden is even more exasperated in three dimensions. This negative aspect is partially alleviated by adopting XFEM methods [16, 35, 24, 37, 29], where the solution space for elements crossed by the interface includes discontinuous functions. For complex moving interfaces a suitable re-meshing procedure

still encumber the computational cost, and therefore the problems of this class are most efficiently approached by methods that embed the interface in a fixed Cartesian grid.

The Immersed Boundary Method [51] is a pioneering work for this class of numerical methods, where Peskin proposed a first-order accurate method derived from the discretization of the δ-function to model blood flows in the heart.

An extension of [51] to second order accuracy is proposed first by LeVeque and Li in [36] (Immersed Interface Methods), where the authors use a suitable six-point stencil to discretize the elliptic operator in grid points close to the interface and find the coefficients of the stencil by Taylor expansion of the solution. The jump condition of the solution is used to modify the coefficients related to nodes close to the interface in order to attain second-order accuracy overall. Second order accuracy using boundary integral techniques was achieved also by Mayo in [43] to solve Poisson (and biharmonic) equations on complex geometries.

More recently, Fedkiw *et al.* introduced the Ghost-Fluid Method in [22], where the authors solve a two-phase problem with an irregular interface between the two phases. In this method, the values on the grid points on the other side of the interface (*ghost points*) is not the physical value (which refers to the other phase of the problem, leading to a discontinuity in the stencil), but a fictitious value (*ghost value*) found by a continuous extrapolation. The problem is finally reduced to two sub-problems by adopting a multi-domain formulation, and each sub-problem is solved by the same technique adopted to solve a single problem with Dirichlet or Neumann boundary conditions.

Liu *et al.* [41] developed a first-order accurate ghost-fluid method for the elliptic equations in the presence of an irregular interface across which the variable coefficient, the solution and its derivative may have jumps. The discretization of the equation results in a symmetric linear system, thus allowing the use of fast iterative solvers. In the case of Dirichlet boundary conditions, instead of interface jump conditions, a second-order symmetric discretization of the ghost-fluid method is proposed by Gibou *et al.* in [25]. A fourth-order (non-symmetric) version can be found in [26], and several features concerning the ghost-fluid method to solve Poisson problems on complex domains with Dirichlet boundary conditions are detailed in [45].

In [46] the authors studied a simple and efficient method for the Navier-Stokes equations on arbitrary shaped domains. Sharp-edge interface problems are also solved by other techniques such as the matched interface and boundary (MIB) method [65], the finite volume method [48], and the non-symmetric positive definite finite element method for matrix coefficient [32]. Other methods accounting for complex interfaces are the arbitrary Lagrangian Eulerian method (ALE) [23, 18], the penalization methods [55, 5, 10] and the class of Immersed Finite Volume Methods (IFVM) [21].

Variable high-jump coefficients across the interface have been presented in recent methods, where only numerical tests with jump ratio up to the order of one thousand have been explored [38, 65, 47, 48], although they show some unsolved robustness issues (lack of monotonicity of the error when the grid is refined [38] or accuracy lower than second order [65]). Jump ratios up to one million may occur in some applications, for example in the fluid-structure interaction where the structure is modelled as an extremely high viscosity fluid.

In some applications it is desirable to achieve higher accuracy in the gradient of the solution as well, as for example in the case of the Stefan problem, where the gradient drives the motion of the interface, or in the incompressible Navier-Stokes equations, where the gradient of the pressure is used to enforce the incompressibility condition in the projection method [11, 44]. In [50] Gibou *et al.* propose an efficient finite-volume discretization for Neumann boundary conditions based on cut-cell methods. The method is second-order accurate for the Poisson and heat equation, while it downgrades to first-order accuracy for the Stefan-type problem. In [28] the authors introduce the Voronoi Interface Method to solve elliptic equation with discontinuous coefficients in arbitrary domains. The Cartesian grid is modified close to the interface by adopting a Voronoi mesh in order to discretize straightforwardly normal derivatives. The final linear system arising form the discretization is symmetric positive definite and the accuracy is second order for the solution and first order for the gradient. In [12] the authors introduce additional grid points on the intersections between Cartesian lines and interface to discretize the jump conditions in the solution and its flux adopting a dimension-by-dimension approach, attaining second order accuracy.

Second order accuracy in both the solution and in the gradient is achieved in [15], where we present a ghost-point method to solve elliptic problems in arbitrary domains with mixed (Dirichlet and Neumann) boundary conditions. Some extensions to the case of sharp-edge boundary and variable matrix coefficient is also presented. A suitable

geometric multigrid approach is introduced to solve the problem, based on a proper relaxation of the boundary conditions on ghost points, with a relaxation parameter chosen in order to achieve the convergence of the iterative method. The extension of the method for the discontinuous coefficient case in one dimension is presented in [14].

Multigrid method is among the most efficient iterative numerical methods to solve the linear system that arises from the discretization of a class of partial differential equations. Initially designed for elliptic problems, nowadays multigrid methods have been adopted to solve a large variety of problems [8, 61, 30]. Regarding the scope of this paper, namely multigrid for elliptic equations with discontinuous coefficients, a detailed survey can be found in [9]. Several methods have been proposed for the simpler case where the interface is aligned with grid lines. As example, we mention [4, 34], where an operator-dependent interpolation is performed assuming the continuity of the flux (instead of the solution gradient), and [53], where an algebraic multigrid method is proposed based on Galerkin Coarse Grid Operator. For Cartesian grids and arbitrary interfaces, i.e. not aligned with grid lines, we mention the paper [3], where a multigrid approach for solving the linear system arising from the discretization of interface conditions described in [3, 39] 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 [1] this multigrid has been improved by modifying the interpolation and restriction operators in such a way the coarse-grid matrices are M-matrices. A comparison of both geometric multigrid methods [3, 1] with Algebraic Multigrid solvers is performed in [2] for the underlying discretization, showing that the multigrid in [1] is the most efficient. Other recent developments of multigrid solvers for non-smooth coefficients can be found in [62], where a geometric multigrid method for multiple interfaces in higher dimensions is proposed, where an accurate interpolation captures the correct boundary conditions at the interfaces via a level set function, and the issues coming from the storage of the coarse-grid matrix are avoided. In [63] 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.

In this paper, the method proposed in [15] for continuous coefficients is extended to the case of discontinuous coefficients across an arbitrary interface, with general non homogeneous jumps in the solution and its gradient. The method is also extended to the case of matrix coefficient. Boundary and interface are implicitly described by level-set functions. The resulting linear system is not symmetric, and a proper multigrid solver is proposed. Numerical results show that second-order accuracy is achieved in both the solution and its gradient. Unlike other existing methods in the literature and mentioned above, variable high-jump coefficients do not affect (i) the monotonic decreasing of the error, which decays with second order accuracy in both the solutions and its gradient, or (ii) the multigrid efficiency, where the convergence factor does not depend on the coefficient ratio.

The plan of the paper is the following: in the next section we describe the model problem for the 2D problem and introduce the level-set function and some notation. The third section is devoted to the description of the discretization of the model problem, with particular attention to the discretization of the boundary and interface conditions. Section 4 describes the multigrid approach with all the multigrid ingredients, consisting mainly of the relaxation scheme and the transfer (restriction and interpolation) operators. Section 5 presents the extension to the matrix coefficient case. Section 6 discuss the extension of the method for 3D problems. The last Sections are devoted to numerical tests, showing the second-order accuracy of the method and the optimal convergence factor of the multigrid approach, and limitations and conclusions.

2 Model Problem

In this paper we mostly refer to the 2D case, although the method can be easily extended to higher dimensions, as described in Sect. 6 and 7.6. 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 $\Omega = \left\{\Omega^+, \Omega^-\right\}$, i.e. Ω^+ and Ω^- are two non-empty domains such that $\Omega^+ \cap \Omega^- = \emptyset$ and $\Omega = \overline{\Omega}^+ \cup \overline{\Omega}^-$, where $\overline{\Omega}^{\pm}$ denotes the closure of $\Omega^{\pm}$. We restrict our study to the case $\partial\Omega^- \cap \partial\Omega = \emptyset$ (see Fig. 1). Let Γ be the *interface* separating the two subdomains, i.e. $\Gamma = \partial\Omega^- \cap \partial\Omega^+$, while the

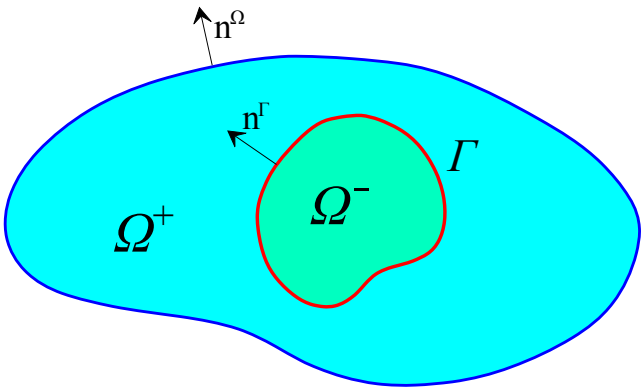


Fig. 1: Domain partition $\Omega = \Omega^- \cup \Omega^+$ and the interface Γ separating the two subdomains along which we impose the jump conditions.

boundary of Ω is $\partial\Omega$. We assume that both $\partial\Omega$ and Γ are smooth curves. Consider the following problem:

$$\begin{cases} -\nabla \cdot \left(\beta^{\pm} \nabla u^{\pm}\right) &= f^{\pm} \text{ in } \Omega^{\pm} \\ [\![u]\!] &= g_D \text{ on } \Gamma \\ \left[\!\left[\beta \dfrac{\partial u}{\partial \mathbf{n}^{\Gamma}}\right]\!\right] &= g_N \text{ on } \Gamma \\ u &= g \text{ on } \partial\Omega \end{cases} \tag{1}$$

where β^- and β^+ are positive smooth functions bounded away from zero, and $\mathbf{n}^{\Gamma}$ is the normal vector to Γ pointing from Ω^- to Ω^+. We denote by $[\![\cdot]\!]$ the jump across the interface Γ, i.e.

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

for any $(\bar{x}, \bar{y}) \in \Gamma$.

Remark 1 *Although we assign Dirichlet boundary conditions on $\partial\Omega$, the numerical method proposed in this paper can be easily extended to the more general case of Neumann or Robin boundary conditions, as proposed in [15]. Since the main focus of this paper is on interface conditions on Γ, we prescribe only Dirichlet boundary conditions on $\partial\Omega$.*

2.1 Level-set functions

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

$$\begin{aligned} \Omega &= \{(x,y) \colon \phi(x,y) < 0\}, \\ \Omega^- &= \left\{(x,y) \colon \phi^{\Gamma}(x,y) < 0 \text{ and } \phi(x,y) < 0\right\}, \\ \Omega^+ &= \left\{(x,y) \colon \phi^{\Gamma}(x,y) > 0 \text{ and } \phi(x,y) < 0\right\}, \\ \Gamma &= \left\{(x,y) \colon \phi^{\Gamma}(x,y) = 0 \text{ and } \phi(x,y) < 0\right\}. \end{aligned} \tag{2}$$

Level set methods for tracking interfaces are introduced and largely discussed, for example, in [49, 57]. For a fixed geometry, the level-set functions are not uniquely defined. A particular case of level-set function is the signed

distance function:

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

where $d\left((x,y),\partial\Omega\right) = \inf_{(\bar{x},\bar{y})\in\partial\Omega} d_e\left((x,y),(\bar{x},\bar{y})\right)$ is the distance between a point and the set $\partial\Omega$ (d_e is the Euclidean distance between points). The signed distance function ϕ can be obtained, for example, by the reinitialization approach [58, 54, 20], namely as the steady-state solution of the following problem

$$\frac{\partial\phi}{\partial t} = \text{sign}(\phi_0)\left(1 - |\nabla\phi|\right), \quad \phi = \phi_0 \text{ at } t = 0, \tag{3}$$

where ϕ_0 is a generic level-set function representing Ω, and t a fictitious time that represents an iterative parameter. A signed distance function is usually numerically more stable than a generic level-set function, since the latter may develop too steep (or too shallow) gradients. From now on we assume that ϕ is the signed-distance function, possibly computed by Eq. (3). We observe that it is sufficient to solve Eq. (3) only for a few time steps, since we need to know the distance function only in a narrow band surrounding the interface/boundary.

The same argument can be repeated for the level-set function ϕ^Γ, referred to the interface Γ.

The normal unit vectors to $\partial\Omega$ and Γ are, respectively:

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

where ϕ and ϕ^Γ are generic level-set functions (not necessarily signed distance functions).

2.2 Notation

Let $N \geq 1$ be an integer and $h = 2/N$ the spatial step. Let $D_h = \left\{\mathbf{j}h, \mathbf{j} = (i,j) \in \{-N, -N+1, \ldots, N-1, N\}^2\right\}$ be the discrete versions of D. D_h is the set of the *grid points*. Let $\Omega_h^+ = \Omega^+ \cap D_h$ and $\Omega_h^- = \Omega^- \cap D_h$ be the discrete versions of Ω^+ and Ω^- respectively. Let $\partial\Omega_h^+$ be the set of the *ghost points for* Ω^+, namely the grid points outside Ω^+ and such that one of the four neighbour grid points is inside Ω^+, i.e.:

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

Similarly, we define $\partial\Omega_h^-$ the set of the *ghost points for* Ω^-, and $\partial\Omega_h$ the set of the *ghost points for* Ω. Let us define $\Gamma_h^- = \partial\Omega_h^- \backslash \partial\Omega_h$ and $\Gamma_h^+ = \partial\Omega_h^+ \backslash \partial\Omega_h$. We call $N_i^+ = \left|\Omega_h^+\right|$, $N_g^+ = \left|\Gamma_h^+\right|$, $N_i^- = \left|\Omega_h^-\right|$, $N_g^- = \left|\Gamma_h^-\right|$, $N_g = |\partial\Omega_h|$. Refer to Fig. 2 for clarity. Since $\partial\Omega^- \cap \partial\Omega = \emptyset$ (see Fig. 1), we have $\Gamma_h^- = \partial\Omega_h^-$. We will use the following notation for discrete functions: $w_{i,j} \approx w(i\,h, j\,h)$, $w_P \approx w(P)$.

3 Discretization of the problem

The final linear system coming from the discretization of the problem will consist of 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 discretizing the first Eq. of (1) by usual central differences:

$$\beta^-_{i+1/2,j}\left(u^-_{i,j} - u^-_{i+1,j}\right) + \beta^-_{i-1/2,j}\left(u^-_{i,j} - u^-_{i-1,j}\right) + \beta^-_{i,j+1/2}\left(u^-_{i,j} - u^-_{i,j+1}\right) + \beta^-_{i,j-1/2}\left(u^-_{i,j} - u^-_{i,j-1}\right) = f^-_{i,j}\,h^2 \tag{4}$$

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^+:

$$\beta^+_{i+1/2,j}\left(u^+_{i,j} - u^+_{i+1,j}\right) + \beta^+_{i-1/2,j}\left(u^+_{i,j} - u^+_{i-1,j}\right) + \beta^+_{i,j+1/2}\left(u^+_{i,j} - u^+_{i,j+1}\right) + \beta^+_{i,j-1/2}\left(u^+_{i,j} - u^+_{i,j-1}\right) = f^+_{i,j}\,h^2 \tag{5}$$

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

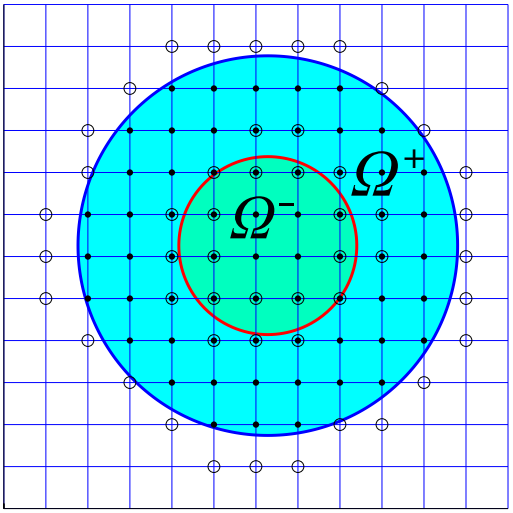


Fig. 2: Inside grid points and ghost points introduced in Sect. 2.2. In particular, the dot grid points inside Ω^- represent the grid points of Ω_h^-, the dot grid points inside Ω^+ and outside Ω^- represent the grid points of Ω_h^+, the unfilled circle grid points inside Ω^- represent the grid points of Γ_h^+, the unfilled circle grid points inside Ω^+ and outside Ω^- represent the grid points of Γ_h^-, the unfilled circle grid points outside Ω^- and Ω^+ represent the grid points of $\partial\Omega_h$.

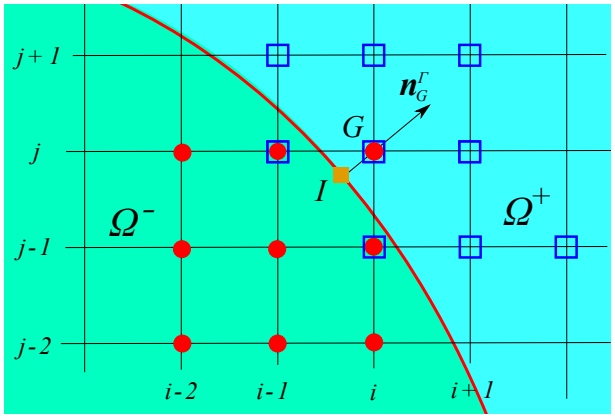


Fig. 3: In this figure $G \in \Gamma_h^-$. The nine-point stencil contained in $\Omega_h^+ \cup \Gamma_h^+$ that serves to interpolate $\tilde{u}^+$ is represented by unfilled squares (modified with respect to the standard central stencil); the nine-point stencil contained in $\Omega_h^- \cup \Gamma_h^-$ that serves to interpolate $\tilde{u}^-$ is represented by filled circles. Interface point I is represented by a filled square.

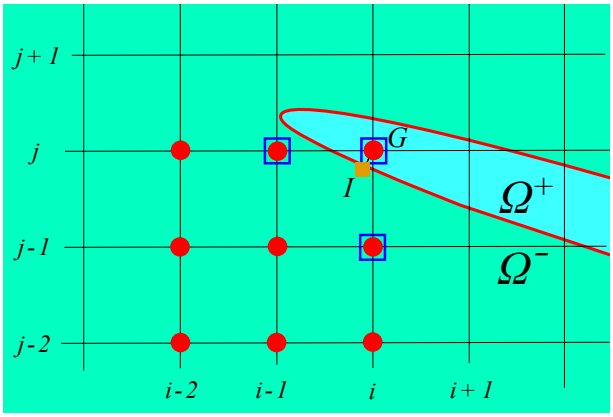


Fig. 4: In this figure $G \in \Gamma_h^-$. The nine-point stencil contained in $\Omega_h^+ \cup \Gamma_h^+$ serving to interpolate $\tilde{u}^+$ has been reduced to the three-point stencil represented by unfilled squares.

3.1 Discretization of the interface conditions

Let $G \in \Gamma_h^- \cup \Gamma_h^+$. In order to find an extrapolated value for G, we discretize the interface conditions (second and third equations of (1)). Let us explain such discretization in detail.

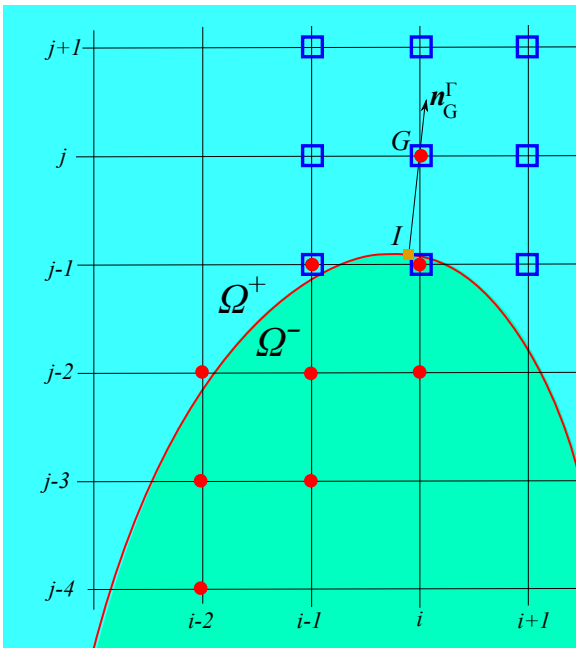


Fig. 5: In this figure $G \in \Gamma_h^-$. The nine-point stencil contained in $\Omega_h^+ \cup \Gamma_h^+$ that serves to interpolate $\tilde{u}^+$ is represented by unfilled squares; the nine-point stencil contained in $\Omega_h^- \cup \Gamma_h^-$ that serves to interpolate $\tilde{u}^-$ is modified with respect to the standard upwind stencil and is represented by filled circles.

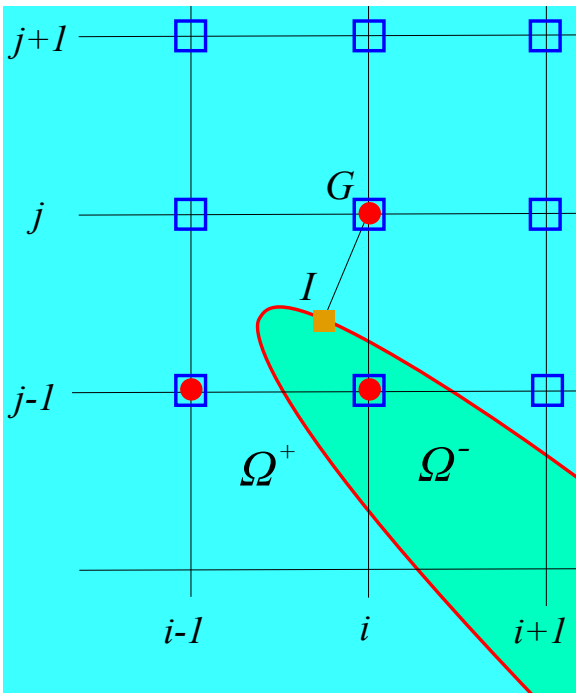


Fig. 6: In this figure $G \in \Gamma_h^-$. The nine-point stencil contained in $\Omega_h^- \cup \Gamma_h^-$ serving to interpolate $\tilde{u}^-$ has been reduced to the three-point stencil (filled circles).

First, we compute an approximation of the unit normal vector to Γ in G pointing from Ω^- to Ω^+, that is $\mathbf{n}_G^\Gamma = (n_{G,x}^\Gamma, n_{G,y}^\Gamma) = \left(\nabla\phi^\Gamma / |\nabla\phi^\Gamma|\right)\big|_G$, using a central finite-difference approximation for $\nabla\phi^\Gamma$. Since ϕ^Γ is the signed distance function, the closest interface point to G, that we call I, is approximated by:

$$I = G - \mathbf{n}_G^\Gamma \cdot \phi^\Gamma(G). \tag{6}$$

Then, the equation of the linear system for the ghost point G is obtained discretizing one of the jump conditions (second and third equation of (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 (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. To 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 (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 $\beta^+ > \beta^-$. The presence of larger off-diagonal values may lead to an 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 $[\![u]\!](I) = g_D(I)$:

$$\boxed{\tilde{u}_h^+(I) - \tilde{u}_h^-(I) = g_D(I)} \tag{7}$$

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

$$\boxed{\left(\beta^+\nabla\tilde{u}_h^+ - \beta^-\nabla\tilde{u}_h^-\right)\big|_I \cdot \left.\left(\frac{\nabla\tilde{\phi}_h^\Gamma}{\left|\nabla\tilde{\phi}_h^\Gamma\right|}\right)\right|_I = g_N(I)} \tag{8}$$

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 $\tilde{\phi}_h^\Gamma$ is the biquadratic interpolant of ϕ^Γ in a nine-point stencil surrounding I. What is left is the choice of the nine-point stencils contained in $\Omega_h^- \cup \Gamma_h^-$ and $\Omega_h^+ \cup \Gamma_h^+$.

Let us recall that the biquadratic interpolant of a function $\omega(x, y)$ is a polynomial:

$$\tilde{\omega}(x, y) = a_1x^2y^2 + a_2x^2y + a_3xy^2 + a_4x^2 + a_5y^2 + a_6xy + a_7x + a_8y + a_9$$

whose coefficients a_i are obtained imposing $\tilde{\omega}(\bar{x}, \bar{y}) = \omega(\bar{x}, \bar{y})$, for each $(\bar{x}, \bar{y})$ belonging to the nine-point stencil.

3.1.1 Choice of the stencil

Let us suppose that $G \in \Gamma_h^-$ (if $G \in \Gamma_h^+$ the procedure is the same, provided that subscripts $+$ and $-$ are interchanged).

3.1.1.1 Stencil contained in $\Omega_h^- \cup \Gamma_h^-$. The nine-point stencil contained in $\Omega_h^- \cup \Gamma_h^-$ is chosen in upwind direction, using the same technique described in [15, 13], i.e.:

$$St_9^U = \left\{G + h(s_x\, k_1, s_y\, k_2) \colon (k_1, k_2) \in \{0, 1, 2\}^2\right\}, \tag{9}$$

where $s_x = \text{sign}(x_I - x_G)$ and $s_y = \text{sign}(y_I - y_G)$, with $G \equiv (x_G, y_G)$ and $I \equiv (x_I, y_I)$. We call the nine-point stencil (9) *standard upwind stencil* and it is contained in $\Omega_h^- \cup \Gamma_h^-$ provided that the grid is sufficiently fine (i.e. the spatial step h is sufficiently small) with respect to the curvature of the interface (such as in Fig. 3). For coarser grids (that must be considered in the multigrid approach, see Sect. 4), it may happen that the stencil is not entirely contained in $\Omega_h^- \cup \Gamma_h^-$ (see Fig. 5). In this case, a *modified* nine-point stencil is chosen when available (Fig. 5), or in the worst case the stencil is reduced to a first-order accurate three-point stencil (Fig. 6), as explained later. Since this latter reduction occurs quite rarely in a fine grid, the overall second-order accuracy is not affected (as shown in numerical tests).

Assuming $n_{G,x}^\Gamma > 0$ and $n_{G,y}^\Gamma > 0$ (the other cases are treated similarly), the (modified) nine-point stencil is chosen using the following algorithm. If $|x_I - x_G| < |y_I - y_G|$ (as in Fig. 5), the nine-point stencil will be composed by three points of each of the columns i, $i-1$ and $i-2$; while if $|x_I - x_G| \geq |y_I - y_G|$ it will be composed by three points of each of the rows j, $j-1$ and $j-2$. Let us suppose $|x_I - x_G| < |y_I - y_G|$ (the opposite case is treated similarly).

- The three points of the column i are:

  $$(i, j)h,\ (i, j-1)h,\ (i, j-2)h.$$

  These three points belong to $\Omega_h^- \cup \Gamma_h^-$, since $(i, j-1)h \equiv G \in \Omega_h^-$.

- The three points of the column $i-1$ are

  $$(i-1, j)h,\ (i-1, j-1)h,\ (i-1, j-2)h$$

  if they belong to $\Omega_h^- \cup \Gamma_h^-$, otherwise we choose:

  $$(i-1, j-1)h,\ (i-1, j-2)h,\ (i-1, j-3)h \tag{10}$$

  if they belong to $\Omega_h^- \cup \Gamma_h^-$, otherwise we reduce the stencil as described later.

- The three points of the column $i-2$ are

$$(i-2,j)h,\ (i-2,j-1)h,\ (i-2,j-2)h$$

if they belong to $\Omega_h^- \cup \Gamma_h^-$, otherwise we choose:

$$(i-2,j-1)h,\ (i-2,j-2)h,\ (i-2,j-3)h$$

if they belong to $\Omega_h^- \cup \Gamma_h^-$, otherwise, if the three points for the column $i-1$ were those indicated in Eq. (10), we choose:

$$(i-2,j-2)h,\ (i-2,j-3)h,\ (i-2,j-4)h$$

if they belong to $\Omega_h^- \cup \Gamma_h^-$ (this is the case illustrated in Fig. 5), otherwise we reduce the stencil as described later.

Reduction of the stencil contained in $\Omega_h^- \cup \Gamma_h^-$. If it is not possible to build the nine-point stencil, we revert to a more robust (less accurate) three-point stencil (Fig. 6):

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

In this case a linear interpolation is used instead of a biquadratic interpolation. Note that these three points belong to $\Omega_h^- \cup \Gamma_h^-$, since $G \equiv (i,j-1)h \in \Omega_h^-$.

3.1.1.2 Stencil contained in $\Omega_h^+ \cup \Gamma_h^+$.

The nine-point stencil contained in $\Omega_h^+ \cup \Gamma_h^+$ is:

$$St_9^C = \left\{G + h(k_1,k_2)\colon (k_1,k_2) \in \{-1,0,1\}^2\right\}, \tag{11}$$

We call the nine-point stencil (11) *standard central stencil* (see Figs. 5 and 6). If it is not contained in $\Omega_h^+ \cup \Gamma_h^+$, a *modified* nine-point stencil is chosen (Fig. 3) or, in the worst case, the stencil is reduced to a three-point stencil (Fig. 4).

The (modified) 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 Fig. 3) it will be composed by three points of each of the rows $j-1$, j and $j+1$; while if $|x_G - x_I| < |y_G - y_I|$ it will be composed by three points of each of the columns $i-1$, i and $i+1$. 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.

Reduction of the stencil contained in $\Omega_h^+ \cup \Gamma_h^+$. If it is not possible to build the nine-point stencil, we revert to a more robust (less accurate) three-point stencil (Fig. 4):

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

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

3.2 Discretization of the boundary conditions

Let $G \in \partial\Omega_h$. In order to find an extrapolated value for G, we discretize the boundary condition on Ω, i.e. the fourth equation of (1). First, we approximate the outward unit normal in G by the formula $\mathbf{n}_G = (n_G^x, n_G^y) = \nabla\phi / |\nabla\phi|$, where $\nabla\phi$ is discretized by central difference in G. Then, the closest boundary point to G, that we call B, is computed similarly to (6):

$$B = G - \phi(G)\mathbf{n}_G. \tag{12}$$

Finally, the equation of the linear system related to the ghost point G is:

$$\tilde{u}(B) = g(B) \tag{13}$$

where $\tilde{u}$ is the biquadratic interpolant of u on the upwind nine-point stencil St_9^U:

$$St_9^U = \left\{ G + h(s_x\, k_1, s_y\, k_2) \colon (k_1,k_2) \in \{0,1,2\}^2 \right\}, \tag{14}$$

where $s_x = \mathrm{sign}(x_B - x_G)$ and $s_y = \mathrm{sign}(y_B - y_G)$, with $G \equiv (x_G, y_G)$ and $B \equiv (x_B, y_B)$. This stencil may not be contained entirely in $\Omega_h^+ \cup \Gamma_h^+$. In such cases it is modified in the same way as in Sect. 3.1.1.1.

4 Multigrid approach

The discretization presented in Sect. 3 leads to a sparse non-symmetric linear system that will be solved efficiently by a multigrid approach. We refer the reader to [61] for a comprehensive presentation of multigrid methods. In this paper we confine the description on how the ingredients of standard multigrid (relaxation scheme, transfer operators) are modified for this specific problem. The first step consists of providing a proper relaxation scheme. Gauss-Seidel and (weighted) Jacobi schemes do not converge for this specific problems unless a special treatment is performed for the relaxation scheme on the ghost points, as described in the following section.

4.1 Relaxation scheme

The relaxation scheme aims at maintaining a Jacobi- or Gauss-Seidel-like iteration scheme for internal points, and propose a proper relaxation scheme for ghost points. For the clarity of description, we describe the scheme for the Jacobi-like scheme (for internal points), although easy generalization can be obtained for weighted-Jacobi or Gauss-Seidel schemes (which are the schemes that we use in practice, since they are more suitable for multigrid techniques due to the smoothing property, see [61, Ch. 2.1]). The whole scheme is obtained discretizing the following *associate time-dependent problem* in space and (fictitious) time:

$$\begin{aligned}
\frac{\partial u^{\pm}}{\partial t} &= \mu^{\pm}\left(f^{\pm} + \nabla\cdot\left(\beta^{\pm}\nabla u^{\pm}\right)\right) \text{ in } \Omega^{\pm} \\
\frac{\partial u^{s_1}}{\partial t} &= \mu_D\left(g_D - [\![u]\!]\right) \text{ on } \Gamma \\
\frac{\partial u^{s_2}}{\partial t} &= \mu_N\left(g_N - \left[\!\!\left[\beta\frac{\partial u}{\partial n}\right]\!\!\right]\right) \text{ on } \Gamma \\
\frac{\partial u}{\partial t} &= \mu_B\left(g - u\right) \text{ on } \partial\Omega
\end{aligned} \tag{15}$$

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 have a better preconditioner for the linear system, as explained in Sect. 3.2. In detail:

$$s_1 = +, \quad s_2 = - \qquad \text{if } \beta^+ \leq \beta^-,$$

$$s_1 = -, \quad s_2 = + \qquad \text{if } \beta^+ > \beta^-.$$

Let us describe this relaxation scheme in detail. For a grid point $(i,j)h \in \Omega_h^-$, the iterative scheme is obtained discretizing the first equation of (15) by forward Euler in time and by (4) in space:

$$u_{i,j}^{-\,(n+1)} = u_{i,j}^{-\,(n)} + \mu_{i,j}^- \, \Delta t \, f_{i,j}^- + \frac{\mu_{i,j}^- \, \Delta t}{h^2} \left(\beta_{i+1/2,j}^- \left(u_{i,j}^{-\,(n)} - u_{i+1,j}^{-\,(n)} \right) + \beta_{i-1/2,j}^- \left(u_{i,j}^{-\,(n)} - u_{i-1,j}^{-\,(n)} \right) \right.$$
$$\left. + \beta_{i,j+1/2}^- \left(u_{i,j}^{-\,(n)} - u_{i,j+1}^{-\,(n)} \right) + \beta_{i,j-1/2}^- \left(u_{i,j}^{-\,(n)} - u_{i,j-1}^{-\,(n)} \right) \right) \quad (16)$$

where $\mu_{i,j}^-$ is chosen in such a way (16) becomes a Jacobi-like scheme, i.e.:

$$\mu_{i,j}^- \Delta t = \frac{h^2}{\beta_{i-1/2,j}^- + \beta_{i+1/2,j}^- + \beta_{i,j-1/2}^- + \beta_{i,j+1/2}^-}. \quad (17)$$

If $(i,j)h \in \Omega_h^+$ the iteration scheme is similar and is obtained by replacing the subscript $-$ with $+$:

$$u_{i,j}^{+\,(n+1)} = u_{i,j}^{+\,(n)} + \mu_{i,j}^+ \, \Delta t \, f_{i,j}^+ + \frac{\mu_{i,j}^+ \, \Delta t}{h^2} \left(\beta_{i+1/2,j}^+ \left(u_{i,j}^{+\,(n)} - u_{i+1,j}^{+\,(n)} \right) + \beta_{i-1/2,j}^+ \left(u_{i,j}^{+\,(n)} - u_{i-1,j}^{+\,(n)} \right) \right.$$
$$\left. + \beta_{i,j+1/2}^+ \left(u_{i,j}^{+\,(n)} - u_{i,j+1}^{+\,(n)} \right) + \beta_{i,j-1/2}^+ \left(u_{i,j}^{+\,(n)} - u_{i,j-1}^{+\,(n)} \right) \right) \quad (18)$$

We observe that in practice we do not need to define both values $\mu_{i,j}^-$ and Δt for Eqs. (16), (18), but only the product $\mu_{i,j}^{\pm\,\Delta t} := \mu_{i,j}^{\pm} \, \Delta t$ according to Eq. (17).

Now, let us consider a ghost point $G \in \partial\Omega_h$. The iterative scheme for G is obtained discretizing the fourth equation of (15) by forward Euler in time and by (13) in space:

$$u_G^{(n+1)} = u_G^{(n)} + \mu_B \, \Delta t \left(g(B) - \tilde{u}^{(n)}(B) \right), \quad (19)$$

where B is the projection point on the boundary $\partial\Omega$ obtained by (12).

If $G \in \Gamma_h^-$, the iterative scheme for G is obtained discretizing the second or third equation of (15), 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 Sect. 3.2. Recalling the choice (7) or (8), and the (6), we summarize the iteration as follows:

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

$$u_G^{-\,(n+1)} = u_G^{-\,(n)} + \mu_D \, \Delta t \left(g_D(I) - \left(\tilde{u}_h^{+\,(n)}(I) - \tilde{u}_h^{-\,(n)}(I) \right) \right) \quad (20)$$

- otherwise, it is:

$$u_G^{-\,(n+1)} = u_G^{-\,(n)} + \mu_N \, \Delta t \left(g_N(I) - \left(\beta^+ \nabla \tilde{u}_h^{+\,(n)} - \beta^- \nabla \tilde{u}_h^{-\,(n)} \right) \Big|_I \cdot \left(\frac{\nabla \tilde{\phi}_h^\Gamma}{\left| \nabla \tilde{\phi}_h^\Gamma \right|} \right) \Big|_I \right) \quad (21)$$

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:

$$u_G^{+\,(n+1)} = u_G^{+\,(n)} + \mu_N\,\Delta t \left(g_N(I) - \left(\beta^+ \nabla \tilde{u}_h^{+\,(n)} - \beta^- \nabla \tilde{u}_h^{-\,(n)}\right)\Big|_I \cdot \left(\frac{\nabla \tilde{\phi}_h^\Gamma}{\left|\nabla \tilde{\phi}_h^\Gamma\right|} \right)\Bigg|_I \right) \tag{22}$$

- otherwise, it is:

$$u_G^{+\,(n+1)} = u_G^{+\,(n)} + \mu_D\,\Delta t \left(g_D(I) - \left(\tilde{u}_h^{+\,(n)}(I) - \tilde{u}_h^{-\,(n)}(I) \right) \right) \tag{23}$$

Up to now, nothing has been said about the sign of the constants μ_D, μ_N and μ_B. This is a crucial point in order to make the whole iterative process convergent. What we request for stability is, in fact, that the derivative of the right-hand side of (20)-(23) with respect to $u_G^{\pm\,(n)}$ is positive and less than one. For example, considering the iteration (20), the derivative of the right-hand side, that we call c_G^-, is:

$$c_G^- = 1 + \mu_D\,\Delta t\,\frac{\partial\,\tilde{u}_h^-(I)}{\partial\,u_G^{-\,(n)}}.$$

The stability is attained for $0 < c_G^- \leq 1$. It can be proved that see (Eq. (69) in Appendix 1)

$$\frac{\partial\,\tilde{u}_h^-(I)}{\partial\,u_G^{-\,(n)}} \geq 0. \tag{24}$$

Therefore, the condition $c_G^- \leq 1$ is ensured by $\mu_D < 0$, while the condition $0 < c_G^-$ implies

$$|\mu_D|\,\Delta t < \left(\frac{\partial\,\tilde{u}_h^-(I)}{\partial\,u_G^{-\,(n)}} \right)^{-1}.$$

This condition must hold for every possible value of $\dfrac{\partial\,\tilde{u}_h^-(I)}{\partial\,u_G^{-\,(n)}}$, which in fact depends on the vector $G - I$. It can be shown that (see Eq. (70) in Appendix 1):

$$\sup_{|G-I| \leq h} \left(\frac{\partial\,\tilde{u}_h^-(I)}{\partial\,u_G^{-\,(n)}} \right) = 1 \tag{25}$$

and then the stability conditions for (20) finally read:

$$\mu_D < 0, \quad |\mu_D|\Delta t < 1. \tag{26}$$

Let us now consider the iteration (21). The derivative c_G^- of the right-hand side with respect to $u_G^{-\,(n)}$ is:

$$c_G^- = 1 + \mu_N\,\Delta t\,\frac{\partial\,\left(\beta^-\,\nabla \tilde{u}_h^-(I) \cdot \tilde{\mathbf{n}}_I^\Gamma\right)}{\partial\,u_G^{-\,(n)}}. \tag{27}$$

where $\tilde{\mathbf{n}}_I^\Gamma = \dfrac{\nabla \tilde{\phi}_h^\Gamma(I)}{\left|\nabla \tilde{\phi}_h^\Gamma(I)\right|}$. The stability condition is $0 < c_G^- \leq 1$. Since $\tilde{\mathbf{n}}_I^\Gamma$ points from Ω^- to Ω^+ and the stencil is chosen in upwind direction, then it can be shown that (see Eq. (72) in Appendix 1)

$$\frac{\partial\,\left(\beta^-\,\nabla \tilde{u}_h^-(I) \cdot \tilde{\mathbf{n}}_I^\Gamma\right)}{\partial\,u_G^{-\,(n)}} \geq 0. \tag{28}$$

Therefore, the condition $c_G^- \leq 1$ is satisfied when $\mu_N < 0$. Condition $0 < c_G^-$ implies

$$|\mu_N|\,\Delta t < \left(\frac{\partial\,\left(\beta^-\,\nabla \tilde{u}_h^-(I)\cdot \tilde{\mathbf{n}}_I^\Gamma\right)}{\partial\, u_G^{-\,(n)}} \right)^{-1}.$$

As before, this condition must be satisfied for each possible value of $\dfrac{\partial\,\left(\beta^-\,\nabla \tilde{u}_h^-(I)\cdot \tilde{\mathbf{n}}_I^\Gamma\right)}{\partial\, u_G^{-\,(n)}}$, that depends on $G-I$. It can be shown that (see Eq. (73) in Appendix 1):

$$\sup_{|G-I|\leq h} \left(\frac{\partial\,\left(\beta^-\,\nabla \tilde{u}_h^-(I)\cdot \tilde{\mathbf{n}}_I^\Gamma\right)}{\partial\, u_G^{-\,(n)}} \right) \leq \frac{3\,\beta^-}{\sqrt{2}\,h}. \tag{29}$$

Finally, the stability conditions for (21) read:

$$\mu_N < 0, \quad \frac{|\mu_N|\,\Delta t}{h} < \frac{\sqrt{2}}{3\,\beta^-}. \tag{30}$$

By the same arguments, the stability conditions for the iteration (22) are:

$$\mu_N < 0, \quad \frac{|\mu_N|\,\Delta t}{h} < \frac{\sqrt{2}}{3\,\beta^+}., \tag{31}$$

while, for the iteration (23) are:

$$\mu_D > 0, \quad \mu_D\,\Delta t < 1. \tag{32}$$

Finally, observe that the conditions on μ_B are (see (19)):

$$\mu_B > 0, \quad \mu_B\,\Delta t < 1. \tag{33}$$

4.1.1 Changing of notation

For simplicity, 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 (15) as follows:

$$\begin{aligned}
\frac{\partial u^\pm}{\partial t} &= \mu^\pm \left(f^\pm + \nabla\cdot\left(\beta^\pm \nabla u^\pm\right)\right) \text{ in } \Omega^\pm \\
\frac{\partial u^{s_1}}{\partial t} &= s_1\,\mu_D\,(g_D - [\![u]\!]) \text{ on } \Gamma \\
\frac{\partial u^{s_2}}{\partial t} &= \mu_N \left(\left[\!\!\left[\beta \frac{\partial u}{\partial n} \right]\!\!\right] - g_N \right) \text{ on } \Gamma \\
\frac{\partial u}{\partial t} &= \mu_B\,(g-u) \text{ on } \partial\Omega.
\end{aligned} \tag{34}$$

The iteration equations of the interface conditions (20)-(23) become:

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

$$u_G^{-\,(n+1)} = u_G^{-\,(n)} - \mu_D\,\Delta t \left(g_D(I) - \left(\tilde{u}_h^{+\,(n)}(I) - \tilde{u}_h^{-\,(n)}(I) \right) \right) \tag{35}$$

- otherwise, it is:

$$u_G^{-\,(n+1)} = u_G^{-\,(n)} - \mu_N\,\Delta t \left(g_N(I) - \left(\beta^+ \nabla \tilde{u}_h^{+\,(n)} - \beta^- \nabla \tilde{u}_h^{-\,(n)} \right)\Big|_I \cdot \left(\frac{\nabla \tilde{\phi}_h}{\left|\nabla \tilde{\phi}_h\right|} \right)\Bigg|_I \right) \tag{36}$$

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

$$u_G^{+\,(n+1)} = u_G^{+\,(n)} - \mu_N\,\Delta t \left(g_N(I) - \left(\beta^+ \nabla \tilde{u}_h^{+\,(n)} - \beta^- \nabla \tilde{u}_h^{-\,(n)}\right)\Big|_I \cdot \left(\frac{\nabla \tilde{\phi}_h}{\left|\nabla \tilde{\phi}_h\right|} \right)\Bigg|_I \right) \tag{37}$$

- otherwise, it is:

$$u_G^{+\,(n+1)} = u_G^{+\,(n)} + \mu_D\,\Delta t \left(g_D(I) - \left(\tilde{u}_h^{+\,(n)}(I) - \tilde{u}_h^{-\,(n)}(I) \right) \right) \tag{38}$$

4.1.2 Choosing constants μ_B, μ_D and μ_N

According to the new notation introduced in the previous section 4.1.1, the conditions (26), (30), (31) and (32) on the constants μ_D and μ_N become:

$$\mu_D\,\Delta t < 1, \quad \frac{\mu_N\,\Delta t}{h} < \frac{\sqrt{2}}{3\,\beta^{\pm}}. \tag{39}$$

The conditions on the positive constant μ_B remain the (33).

We observe that in practice we do not need to define all four values μ_D, μ_N, μ_B and Δt for Eqs. (19), (35), (36), (37), (38), but only the products $\mu_B^{\Delta t} = \mu_B\,\Delta t$, $\mu_D^{\Delta t} = \mu_D\,\Delta t$ and $\mu_N^{\Delta t} = \mu_N\,\Delta t$. In the numerical tests of Sect. 7 we use $\mu_B^{\Delta t} = \mu_D^{\Delta t} = 0.9$ and $\mu_N^{\Delta t} = 0.9\,\dfrac{\sqrt{2}\,h}{3\,\max\{\beta^+, \beta^-\}}$ (that satisfy the stability conditions (33), (39)).

4.1.3 Smoothing property

As mentioned at the beginning of Sect. 4.1, it is known that the Jacobi scheme is not a proper smoother for the multigrid algorithm (see, for instance, [61, Ch. 2.1]). Therefore, we need to replace the Jacobi-like scheme introduced in Sect. 4.1 with a relaxation scheme that holds the smoothing property, such as the Gauss-Seidel scheme or the weighted Jacobi scheme (with weight $\omega = 4/5$ in 2D). In the following, we revert as example to a Gauss-Seidel relaxation scheme. The smoothing property of the Gauss-Seidel scheme depends on the order chosen for the variables. It is well known (see [61, Ch. 2.1]) that the Gauss-Seidel Red-Black (GS-RB) scheme is a better smoother with respect to the Gauss-Seidel Lexicographic (GS-LEX) scheme. Anyway we study for simplicity the smoothing properties of the GS-LEX scheme and we compare the convergence factor with the one predicted by the Local Fourier Analysis for rectangular domains (see [61, Ch. 4.6.1] for more details), since the main goal of this paper is to show the efficiency of the method for complex geometries and not the choice of the best relaxation scheme. In order to obtain a more efficient multigrid method, a GS-RB can be easily employed instead of the GS-LEX (this is behind the scope of the paper).

Finally, we switch from the relaxation scheme described in Sect. 4.1 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 order of the iterations. We use a lexicographic order for inner equations, and any order for interface and boundary conditions.

In detail, we order the grid points according to the following list:

$$\left\{\partial\Omega_h, \Gamma_h^-, \Gamma_h^+, \Omega_h^-, \Omega_h^+\right\}.$$

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.:

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

In order to avoid that the boundary effects degrade the convergence factor, we perform some extra-relaxations on two suitable layers surrounding respectively the interface and the boundary. In detail, we choose a positive integer λ and a positive parameter δ, and we introduce two additional sets of grid points:

$$\Omega_h^{(\delta)} = \{P \in \Omega_h \text{ such that } d(P, \partial\Omega) < \delta\},$$

$$\Omega_h^{\pm\,(\delta)} = \left\{P \in \Omega_h^{\pm} \text{ such that } d(P, \Gamma) < \delta\right\}.$$

One single relaxation includes some over-relaxations on $\Omega_h^{(\delta)}$ and $\Omega_h^{\pm\,(\delta)}$ and is represented by the Algorithm 1. Similarly to [15], we experienced that a good choice is:

$$\lambda = 5, \quad \delta = 5\,h. \tag{40}$$

4.2 Multigrid components

Let us extend the notation of Sect. 2.2. For a grid with spatial step h, we denote:

$$\Omega_h^{--} = \Omega_h^- \cup \Gamma_h^-, \qquad \Omega_h^{++} = \Omega_h^+ \cup \Gamma_h^+.$$

Let us define the set of functions defined on a subset I_h of the grid:

$$S(I_h) = \{w_h \colon I_h \to \mathbb{R}\}, \text{ for any } I_h \subseteq D_h,$$

and then the set of functions defined on internal points:

$$\bar{S}(\Omega_h) = S(\Omega_h^- \cup \Gamma_h^-) \times S(\Omega_h^+ \cup \Gamma_h^+).$$

The discrete differential operators of the elliptic equation can be expressed by:

$L_h^- \colon S(\Omega_h^{--}) \times S(\Omega_h^{--}) \to S(\Omega_h^-)$ such that

$$L_h^-(\beta_h^-, u_h^-)_{i,j} = \frac{1}{h^2}\Big(\beta_{i+1/2,j}^-\left(u_{i,j}^- - u_{i+1,j}^-\right) + \beta_{i-1/2,j}^-\left(u_{i,j}^- - u_{i-1,j}^-\right) + \beta_{i,j+1/2}^-\left(u_{i,j}^- - u_{i,j+1}^-\right) + \beta_{i,j-1/2}^-\left(u_{i,j}^- - u_{i,j-1}^-\right)\Big) \text{ for any } (i,j)h \in \Omega^-,$$

$L_h^+ \colon S(\Omega_h^{++}) \times S(\Omega_h^{++}) \to S(\Omega_h^+)$ such that

$$L_h^+(\beta_h^+, u_h^+)_{i,j} = \frac{1}{h^2}\Big(\beta_{i+1/2,j}^+\left(u_{i,j}^+ - u_{i+1,j}^+\right) + \beta_{i-1/2,j}^+\left(u_{i,j}^+ - u_{i-1,j}^+\right) + \beta_{i,j+1/2}^+\left(u_{i,j}^+ - u_{i,j+1}^+\right) + \beta_{i,j-1/2}^+\left(u_{i,j}^+ - u_{i,j-1}^+\right)\Big) \text{ for any } (i,j)h \in \Omega^+,$$

and then they can be summarized as a unique operator L_h:

$L_h \colon \bar{S}(\Omega_h) \times \bar{S}(\Omega_h) \to S(\Omega_h)$ such that

$$L_h(\beta_h, u_h)(P) = \begin{cases} L_h^-(\beta_h^-, u_h^-) \text{ if } P \in \Omega_h^- \\ L_h^+(\beta_h^+, u_h^+) \text{ if } P \in \Omega_h^+ \end{cases}$$

$$\text{where } \beta_h = (\beta_h^-, \beta_h^+), \quad u_h = (u_h^-, u_h^+).$$

Algorithm 1 One single relaxation includes some over-relaxations in the vicinity of the interface and boundary.

for $i = 1 \to \lambda$ **do**

  for all $G \in \partial\Omega_h$ **do**
    perform the iteration equation (19);
  end for
  for all $G \in \Gamma_h^-$ **do**
    perform the iteration equation (35) or (36);
  end for
  for all $G \in \Gamma_h^+$ **do**
    perform the iteration equation (37) or (38);
  end for
  for all $P \in \Omega_h^{-\,(\delta)}$ **do**
    perform the iteration equation (16);
  end for
  for all $P \in \Omega_h^{+\,(\delta)}$ **do**
    perform the iteration equation (18);
  end for
end for

for all $G \in \partial\Omega_h$ **do**
  perform the iteration equation (19);
end for
for all $G \in \Gamma_h^-$ **do**
  perform the iteration equation (35) or (36);
end for
for all $G \in \Gamma_h^+$ **do**
  perform the iteration equation (37) or (38);
end for
for all $P \in \Omega_h^-$ **do**
  perform the iteration equation (16);
end for
for all $P \in \Omega_h^+$ **do**
  perform the iteration equation (18);
end for

The discrete jump operators become:

$[\cdot,\cdot]_h^- : \bar{S}(\Omega_h) \times \bar{S}(\Omega_h) \to S(\Gamma_h^-)$ such that

$$[\beta_h, u_h]_h^-(G) = \begin{cases} \tilde{u}_h^+(I) - \tilde{u}_h^-(I) \\ \text{or} \\ \left(\beta^+ \nabla \tilde{u}_h^+ - \beta^- \nabla \tilde{u}_h^-\right)\big|_I \cdot \left(\dfrac{\nabla \tilde{\phi}_h^\Gamma}{\left|\nabla \tilde{\phi}_h^\Gamma\right|} \right)\Bigg|_I \end{cases}$$

for any $G \in \Gamma_h^-$, according to the choice (7) or (8),

$[\cdot,\cdot]_h^+ : \bar{S}(\Omega_h) \times \bar{S}(\Omega_h) \to S(\Gamma_h^+)$ such that

$$[\beta_h, u_h]_h^+(G) = \begin{cases} \tilde{u}_h^+(I) - \tilde{u}_h^-(I) \\ \text{or} \\ \left(\beta^+ \nabla \tilde{u}_h^+ - \beta^- \nabla \tilde{u}_h^-\right)\big|_I \cdot \left(\dfrac{\nabla \tilde{\phi}_h^\Gamma}{\left|\nabla \tilde{\phi}_h^\Gamma\right|} \right)\Bigg|_I \end{cases}$$

for any $G \in \Gamma_h^+$, according to the choice (7) or (8).

Discrete right-hand sides of the jump conditions are summarized by a grid function $g_h^+ \in S(\Gamma_h^+)$ $[g_h^- \in S(\Gamma_h^-)]$ such that $g_h^+(G) = g_D(I)$ or $g_N(I)$ $[g_h^-(G) = g_D(I)$ or $g_N(I)]$, for any $G \in \Gamma_h^+$ $[G \in \Gamma_h^-]$, according to the choice (7) or (8). Discrete boundary condition operator can be expressed by:

$\mathcal{B}_h \colon \bar{S}(\Omega_h) \to S(\partial\Omega_h)$ such that $\mathcal{B}_h(u_h) = \tilde{u}(B)$ according to the Eq. (13),

and the respective discrete right-hand side is a grid function $g_h \in S(\partial\Omega_h)$ such that $g_h(G) = g(B)$ according to the Eq. (13). With this notation, we can write the linear system on the grid with spatial step h in the following compact form:

$$\begin{aligned} L_h(\beta_h, u_h) &= f_h \\ [\beta_h, u_h]_h^- &= g_h^- \\ [\beta_h, u_h]_h^+ &= g_h^+ \\ \mathcal{B}(u_h) &= g_h \end{aligned} \tag{41}$$

In order to describe the multigrid algorithm, we first introduce the extension operator and the transfer (restriction and interpolation) operators.

4.2.1 Extension operator

Let us consider the whole domain Ω (the argument can be easily repeated with the two subdomains Ω^- and Ω^+). This domain is defined by a level-set function ϕ and it defines a set of inner grid points Ω_h and a set of ghost points $\partial\Omega_h$. Let us suppose we know a grid function ω_h only on grid nodes of $\partial\Omega_h$ (i.e. $\omega_h \in S(\partial\Omega_h)$) and we want to extend ω_h in the whole domain $D_h \backslash \Omega_h$, namely we want to obtain a new grid function $\omega_h^{ext} \in S(D_h \backslash \Omega_h)$. The function ω_h^{ext} can be obtained by extrapolating ω_h constant along the normal direction to $\partial\Omega$, i.e. solving the transport equation

$$\frac{\partial \omega}{\partial \tau} + \nabla\omega \cdot \mathbf{n} = 0 \tag{42}$$

for a few steps of a fictitious time τ, where $\omega = \omega_h$ in $\partial\Omega_h$, and $\mathbf{n} \equiv (n_x, n_y) = \nabla\phi/|\nabla\phi|$ is the unit normal vector. The procedure is analogous to the one presented in [15] for the case of continuous coefficients.

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

$$\mathcal{E}[\partial\Omega_h; \phi_h] \colon S(\partial\Omega_h) \longrightarrow S(D_h \backslash \Omega_h). \tag{43}$$

4.2.2 Restriction operator

We want to define a suitable restriction operator:

$$\mathcal{I}_{2h}^h \colon S(Z_h) \to S(Z_{2h}), \text{ where } Z_h \subseteq D_h, \quad Z_{2h} = Z_h \cap D_{2h}.$$

We perform the usual full-weighting restriction away from the boundary/interface, while we modify the restriction for inner equations close to the boundary/interface and for the boundary conditions. We recall the full-weighting restriction operator (see [61, Ch. 2.3.3]):

$$\mathcal{I}_{2h}^{h} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{2h}^{h}. \tag{44}$$

In general, by the stencil notation

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

we will intend the restriction operator $\mathcal{I}_{2h}^{h}$ defined by:

$$\mathcal{I}_{2h}^{h} w_h(x,y) = \sum_{(i,j)\in R_k} t_{i,j} w_h(x+jh, y+ih),$$

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.

Following the same technique of [15], we modify the restriction operator when we are close to the interface/boundary in such a way we only use values coming from the same side of the interface/boundary. The modified restriction operator acting on a general subset $Z_h \subseteq D_h$ is therefore defined as follows. Let $(x,y) \in Z_{2h}$ and let $\mathcal{N}(x,y) = \{(x+jh, y+ih)\colon j,i = -1,0,1\}$ be the neighborhood of (x,y). We define $\mathcal{T}$ as the maximum full rectangle with vertices belonging to $\mathcal{N}(x,y)$ and such that $\mathcal{T} \cap D_h \subseteq Z_h$ (see Fig. 7, where $Z_h = \Omega_h^-$). The stencil used in (x,y) to transfer w_h to a coarser grid depends on the size of $\mathcal{T}$. In particular, if $\mathcal{T} \cap D_h$ is a 3×3 point stencil (i.e. $\mathcal{N}(x,y) \subseteq Z_h$), then we can use the standard full-weighting stencil (44). Now let $\mathcal{T} \cap D_h$ be a 3×2 point stencil (the case 2×3 is similar). Without loss of generality, we can suppose that the vertices of $\mathcal{T}$ are $(x+jh, y+ih)$, with $j \in \{-1,0\}$, $i \in \{-1,1\}$. In this case, the restriction operator is:

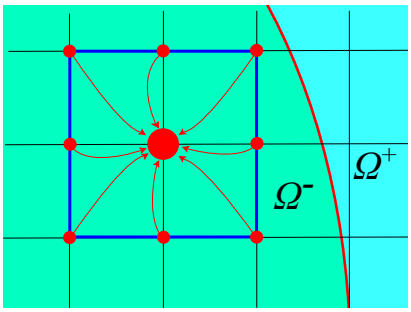
$$\left(I_{2h}^{h} w_h\right)(x,y) = \frac{1}{16} \begin{bmatrix} 2 & 2 & 0 \\ 4 & 4 & 0 \\ 2 & 2 & 0 \end{bmatrix}_{2h}^{h} (x,y), \tag{45}$$

while, if $\mathcal{T}$ is a 2×2 point stencil, with vertex $(x+j\,h, y+i\,h)$, $j,i \in \{-1,0\}$, the restriction operator is:

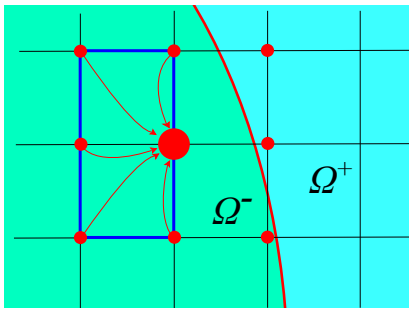
$$\left(I_{2h}^{h} w_h\right)(x,y) = \frac{1}{16} \begin{bmatrix} 0 & 0 & 0 \\ 4 & 4 & 0 \\ 4 & 4 & 0 \end{bmatrix}_{2h}^{h} (x,y), \tag{46}$$

These three cases are summarized in Fig. 7 (where $Z_h = \Omega_h^-$).

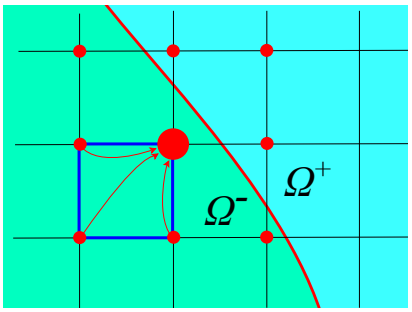
The restriction of the boundary/interface conditions is performed similarly, provided extending the defect of the boundary/interface conditions away from the boundary/interface (by the procedure described in Sect. 4.2.1) and then performing the same restriction procedure, using only the values coming from the same side of the boundary/interface. The procedure will be described in detail in Sect. 4.2.4.



$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{2h}^{h}$$



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



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

Fig. 7: Top: nine point stencil $\mathcal{N}(x,y)$ (circles) and the boundary of the rectangle $\mathcal{T}$ (bold line). The bold circle is on both the coarser and finer grids, while the smaller circles are only on the finer grid. The arrows represent the action of the restriction operators. Bottom: the respective stencils in matrix form used by the restriction operator.

4.2.3 Interpolation operator

The interpolation operator of the multigrid algorithm acts on the error, which is continuous across the boundary/interface. Therefore, we do not need to modify the stencil for particular cases and we are allowed to use the standard linear interpolation operator:

$$\mathcal{I}_h^{2h} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{h}^{2h}. \tag{47}$$

4.2.4 Two-Grid Correction scheme

Let us suppose we have an exact solver $\mathcal{S}$ of the linear system (41) for a grid with an arbitrary spatial step h:

$$u_h = \mathcal{S}(\beta_h, f_h^-, f_h^+, g_h^-, g_h^+, g_h).$$

Now, in order to describe the multigrid technique to solve the linear system (41), it is sufficient to describe the TGCS (Two-Grid Correction Scheme), since any other multigrid algorithm (such as for example V-cycle, W-cycle, Full Multigrid) can be easily derived from it (see [61, Ch. 2.4, 2.6] for more details). The TGCS consists into the following algorithm:

1. Set initial guess $u_h = 0$;
2. Relax ν_1 times (by the Algorithm 1) on the grid with spatial step h
3. Compute the following defects:

$$\begin{aligned} r_h^{\Omega^-} &= f_h^- - L_h^-(\beta_h^-, u_h^-) \\ r_h^{\Omega^+} &= f_h^+ - L_h^+(\beta_h^+, u_h^+) \\ r_h^{\Gamma^-} &= g_h^- - [\beta_h, u_h]_h^- \\ r_h^{\Gamma^+} &= g_h^+ - [\beta_h, u_h]_h^+ \\ r_h^{\partial\Omega} &= g_h - \mathcal{B}(u_h) \end{aligned}$$

4. Extend the defects $r_h^{\Gamma^-}$, $r_h^{\Gamma^+}$ and r_h^{Γ} using the extension operator defined in (43):

$$\begin{aligned} r_h^{\Gamma^-,ext} &= \mathcal{E}[\Gamma_h^-;\phi_h^{\Gamma}](r_h^{\Gamma^-}), \\ r_h^{\Gamma^+,ext} &= \mathcal{E}[\Gamma_h^+;-\phi_h^{\Gamma}](r_h^{\Gamma^+}), \\ r_h^{\partial\Omega,ext} &= \mathcal{E}[\partial\Omega_h;\phi_h](r_h^{\partial\Omega}). \end{aligned}$$

5. Transfer these defects to a coarser grid with spatial step $2h$ by the *restriction operator* defined in Sect. 4.2.2:

$$\begin{aligned} r_{2h}^{\Omega^-} &= I_{2h}^h\left(r_h^{\Omega^-}\right) \\ r_{2h}^{\Omega^+} &= I_{2h}^h\left(r_h^{\Omega^+}\right) \\ r_{2h}^{\Gamma^-} &= I_{2h}^h\left(r_h^{\Gamma^-,ext}\right) \\ r_{2h}^{\Gamma^+} &= I_{2h}^h\left(r_h^{\Gamma^+,ext}\right) \\ r_{2h}^{\partial\Omega} &= I_{2h}^h\left(r_h^{\partial\Omega,ext}\right) \end{aligned}$$

6. Solve the residual problem in the coarser grid

$$e_{2h} = \mathcal{S}(\beta_{2h}, r_{2h}^{\Omega^-}, r_{2h}^{\Omega^+}, r_{2h}^{\Gamma^-}, r_{2h}^{\Gamma^+}, r_{2h}^{\partial\Omega})$$

7. Transfer the error to the finer grid by the interpolation operator (47):

$$e_h = I_h^{2h}(e_{2h})$$

8. Correct the fine-grid approximation

$$u_h :\, = u_h + e_h$$

9. Relax ν_2 times (by the Algorithm 1) on the grid with spatial step h.

As mentioned above, the other multigrid algorithms are based on the recursive application of the TGCS. For example, the $V-$cycle algorithm is obtained by substituting the exact solver of step 6 with the application of the TGCS in the coarser grid. The recursive procedure continues until a suitable coarse grid is reached, where the exact solver is employed.

One multigrid iteration consists of an entire $V-$cycle. Iterations are performed until a suitable tolerance is satisfied, chosen in such a way the algebraic error (due to this stopping criterion) is negligible with respect to the discretization error.

5 Matrix coefficient

In this section we discuss the extension of the numerical method to the case of matrix coefficients. Numerical tests are provided in Sect. 7.5. The problem reads:

$$\begin{cases} -\nabla\cdot\left(\beta^{\pm}\nabla u^{\pm}\right) &= f^{\pm} \text{ in } \Omega^{\pm} \\ [\![u]\!] &= g_D \text{ on } \Gamma \\ \left[\!\left[\beta\,\nabla u\cdot\mathbf{n}^{\Gamma}\right]\!\right] &= g_N \text{ on } \Gamma \\ u &= g \text{ on } \partial\Omega \end{cases}, \tag{48}$$

where $\beta^{\pm}$ is a symmetric positive definite matrix

$$\beta^{\pm} = \beta^{\pm}(x,y) = \begin{pmatrix} \beta_{11}^{\pm}(x,y) & \beta_{12}^{\pm}(x,y) \\ \beta_{12}^{\pm}(x,y) & \beta_{22}^{\pm}(x,y) \end{pmatrix}, \tag{49}$$

i.e. $\beta_{11}^{\pm} > 0$ and $\beta_{11}^{\pm}\beta_{22}^{\pm} > (\beta_{12}^{\pm})^2$. Observe that $\beta\,\nabla u \cdot \mathbf{n}^{\Gamma} = \nabla u \cdot (\beta\,\mathbf{n}^{\Gamma})$ (since β is symmetric), and therefore the interface condition $[\![\beta\,\nabla u \cdot \mathbf{n}^{\Gamma}]\!] = g_N$ may be written

$$\left[\!\left[\mathbf{n}_{\mathrm{co}}^{\Gamma} \cdot \nabla u\right]\!\right] = g_N,$$

where $\mathbf{n}_{\mathrm{co}}^{\Gamma\pm} = \beta^{\pm}\,\mathbf{n}^{\Gamma}$ is the co-normal vector. For simplicity, we omit the Γ superscript in the notation of this section, though we always refer to the normal to the interface Γ.

We follow the same idea of [15] to discretize (48). In particular, we expand the first equation of (48)

$$-\left(\frac{\partial\beta_{11}^{\pm}}{\partial x}\frac{\partial u^{\pm}}{\partial x} + \frac{\partial\beta_{12}^{\pm}}{\partial x}\frac{\partial u^{\pm}}{\partial y} + \frac{\partial\beta_{12}^{\pm}}{\partial y}\frac{\partial u^{\pm}}{\partial x} + \frac{\partial\beta_{22}^{\pm}}{\partial y}\frac{\partial u^{\pm}}{\partial y} + \beta_{11}^{\pm}\frac{\partial^2 u^{\pm}}{\partial x^2} + 2\beta_{12}^{\pm}\frac{\partial^2 u^{\pm}}{\partial x\partial y} + \beta_{22}^{\pm}\frac{\partial^2 u^{\pm}}{\partial y^2}\right) = f^{\pm}$$

and we discretize the derivatives using standard central difference. Special attention must be posed to the mixed derivative $\dfrac{\partial^2 u^{\pm}}{\partial x\partial y}$. In detail, we use standard central difference away from the interface

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

and a modified stencil near the interface (in order to maintain the same set of ghost points as in the scalar coefficient case):

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

if $n_x \cdot n_y \geq 0$, and

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

if $n_x \cdot n_y < 0$, where $\mathbf{n} = (n_x, n_y)$. An example is shown in Fig. 8. An explanation of these discretizations can be found, for example, in [61, page 264].

The discretization of the interface conditions is a straightforward extension of the scalar coefficient case.

Some additional aspects must be discussed for the matrix coefficient case, i.e. the condition $\beta^{+}(I) > \beta^{-}(I)$ (to select between (7) and (8)) and the condition (39). Condition $\beta^{+}(I) > \beta^{-}(I)$ was introduced to improve the condition number of the linear system. The idea of ensuring that the diagonal term is greater than the off-diagonal terms cannot be extended to the matrix coefficient case in a straightforward manner. A possible solution, that is confirmed by numerical tests, is to compare (upper bounds of) the coefficients of u_G^{-} and u_G^{+} in Eq. (8) (see (70) and (73) for the case of scalar coefficient), which in this case can be written as:

$$\left(\left\|\tilde{\mathbf{n}}_{\mathrm{co}}^{+}\right\|_2 \nabla\tilde{u}_h^{+} \cdot \hat{\tilde{\mathbf{n}}}_{\mathrm{co}}^{+} - \left\|\tilde{\mathbf{n}}_{\mathrm{co}}^{-}\right\|_2 \nabla\tilde{u}_h^{-} \cdot \hat{\tilde{\mathbf{n}}}_{\mathrm{co}}^{-}\right)\Big|_I = g_N(I)$$

where

$$\tilde{\mathbf{n}}_{\mathrm{co}}^{\pm} = \beta^{\pm}\,\frac{\nabla\tilde{\phi}_h^{\Gamma}}{\left|\nabla\tilde{\phi}_h^{\Gamma}\right|}.$$

is the approximation of the co-normal vector and $\hat{\tilde{\mathbf{n}}}_{\mathrm{co}}^{\pm}$ is the corresponding unit vector. Absolute value of the coefficients of $u_G^{\pm}$ are then expressed by

$$\frac{\partial\left(\left\|\tilde{\mathbf{n}}_{\mathrm{co}}^{\pm}\right\|_2 \nabla\tilde{u}_h^{\pm} \cdot \hat{\tilde{\mathbf{n}}}_{\mathrm{co}}^{\pm}\right)}{\partial u_G^{\pm}} \tag{50}$$

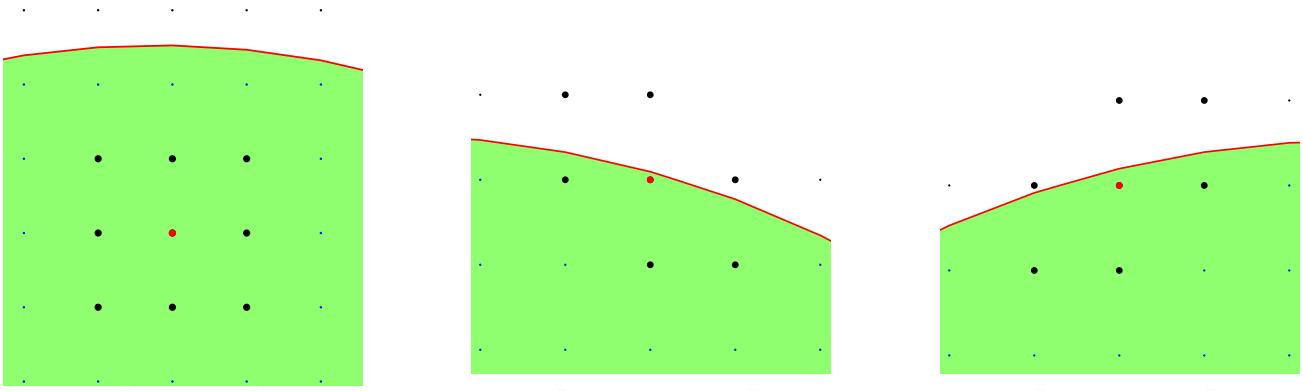


Fig. 8: The stencil for the mixed derivative changes accordingly to the distance from the boundary and to the normal direction.

We observe that Eq. (73) of Appendix 1 is valid for a generic unit vector, and therefore can be applied in (50) with the unit vector $\hat{\tilde{\mathbf{n}}}^{\pm}_{\text{co}}$. Therefore, using Eq. (73) and the bound $\left\|\tilde{\mathbf{n}}^{\pm}_{\text{co}}\right\|_2 = \left\|\beta^{\pm}\,\tilde{\mathbf{n}}^{\pm}\right\|_2 \leq \left\|\beta^{\pm}\right\|_2$, we obtain the following bound for (50)

$$\left\|\beta^{\pm}\right\|_2 \frac{3}{\sqrt{2}\,h}.$$

Therefore, the condition $\beta^+(I) > \beta^-(I)$ now reads:

$$\left\|\beta^{+}\right\|_2 > \left\|\beta^{-}\right\|_2.$$

Since $\beta^{\pm}$ is a symmetric positive definite matrix, then $\left\|\beta^{\pm}\right\|_2$ corresponds to the dominant eigenvalue, which can be easily computed by hand, leading to the final condition:

$$\lambda^{+}_{\beta} > \lambda^{-}_{\beta},$$

where

$$\lambda^{\pm}_{\beta} = \frac{\beta^{\pm}_{11} + \beta^{\pm}_{22} + \sqrt{(\beta^{\pm}_{11} - \beta^{\pm}_{22})^2 + 4(\beta^{\pm}_{12})^2}}{2}$$

is the dominant eigenvalue of $\beta^{\pm}$. Observe that this condition reverts to $\beta^+(I) > \beta^-(I)$ in the case of scalar coefficient (since $\beta_{11} = \beta_{22}$ and $\beta_{12} = 0$).

Regarding the multigrid approach, condition (39) must be modified accordingly. In particular, We observe that the 9-point stencil for the discretization of the interface conditions (jumps in the solution and in the flux) is chosen in the Upwind direction, which may not correspond to the co-normal direction. For this reason, condition (28) is not guaranteed and an additional check on the sign of (71) must be carried out in order to corresponding sign of μ_N and ensuring that $c^-_G \leq 1$ in (71). Finally, Eq. (29) can be easily extended by using (73) (by the same argument described above) and then condition (39) now reads

$$\mu_D\,\Delta t < 1, \quad \frac{\mu_N\,\Delta t}{h} < \frac{\sqrt{2}}{3\,\lambda^{\pm}_{\beta}}.$$

6 Extension to the 3D case

Although the numerical method has been entirely described for 2D problems, the extension to 3D problems is mostly straightforward and one test is presented in this paper (Sect. 7.6). Some crucial aspects of the method must be adapted to the 3D case with care. For example, 3D versions of central difference discretizations (4) and (5) use 7-point stencils rather than 5-point stencils, while the stencils (9) and (11) used to discretize the interface/boundary conditions have $3^3 = 27$ points in 3D rather than $3^2 = 9$ points (in general, the stencil is made by 3^d points in d dimensions), and can be expressed by:

$$St_{27}^U = \left\{ G + h(s_x\, k_1, s_y\, k_2, s_z\, k_3) \colon (k_1, k_2, k_3) \in \{0, 1, 2\}^3 \right\}, \tag{51}$$

$$St_{27}^C = \left\{ G + h(k_1, k_2, k_3) \colon (k_1, k_2, k_3) \in \{-1, 0, 1\}^3 \right\}, \tag{52}$$

where $s_x = \text{sign}(x_I - x_G)$, $s_y = \text{sign}(y_I - y_G)$ and $s_z = \text{sign}(z_I - z_G)$, with $G \equiv (x_G, y_G, z_G)$ (ghost point) and $I \equiv (x_I, y_I, z_I)$ (interface point).

Moreover, the choice of the different discretization configurations of the interface stencils presented in Sections 3.1.1.1 and 3.1.1.2 (Figs. 3 and 5) can be extended to the 3D case as follows. For simplicity, we only describe the case of the stencil (51), since the case of the stencil (52) can be implemented similarly.

Let G be a ghost point and s_{ijk} the compact representation of the stencil (51), where $i = j = k = 0$ refer to the ghost point G (see Fig. 15 for a 2D version of the stencil). Without loss of generality, assume that the normal vector $\mathbf{n} = (n_x, n_y, n_z)$ is such that $n_x, n_y, n_z < 0$ (as in Fig. 15). Observe that in 2D the stencil may be shifted only in one direction with respect to the standard configuration 9, as can be seen in Fig. 5, where the unfilled square stencil is shifted in the $x-$direction, or in Fig. 3, where the filled circle stencil is shifted in the $y-$direction. In 3D the approach is analogous and the direction along which the stencil may be switched is chosen as follows: if $|x_I - x_G| > |y_I - y_G|$ and $|x_I - x_G| > |z_I - z_G|$, then the stencil is possibly shifted along the $x-$direction, else if $|y_I - y_G| > |z_I - z_G|$, then the stencil is possibly shifted along the $y-$direction, else the stencil is possibly shifted along the $z-$direction.

Assume that the direction along which the stencil may be shifted is the $x-$direction (the other instances are analogous). In this case, the code checks that each point s_{0jk} belongs to $\Omega_h^- \cup \Gamma_h^-$. If so, no shifting is performed. If not, then the points $s_{0j^*k^*}$ that do not belong to $\Omega_h^- \cup \Gamma_h^-$ are marked and the stencil is shifted for each of these points as described below. The stencil point $s_{0j^*k^*}$ is overridden by:

$$s_{0j^*k^*} \coloneqq s_{3j^*k^*}.$$

Then, the interpolation coefficients for the interface conditions, that are already computed for the standard non-shifted configuration (51), must be updated as well. Referring to the notation adopted in (67) for the 2D case, we update the coefficients by:

$$c_i^x \coloneqq c_i^x + d_i\, c_0^x, \qquad c_i'^x \coloneqq c_i'^x + d_i\, c_0'^x, \qquad i = 2, 1, 0,$$

where $d_0 = 1$, $d_1 = 3$ and $d_2 = -3$.

In some cases, the point s_{1jk} does not belong to $\Omega_h^- \cup \Gamma_h^-$ either, and then the following additional update of stencil and coefficients must be performed:

$$s_{1j^*k^*} \coloneqq s_{4j^*k^*}.$$
$$c_i^x \coloneqq c_i^x + \tilde{d}_i\, c_0^x, \qquad c_i'^x \coloneqq c_i'^x + \tilde{d}_i\, c_0'^x, \qquad i = 2, 1, 0,$$

where $\tilde{d}_0 = -3$, $\tilde{d}_1 = 1$ and $\tilde{d}_2 = 3$.

Once the stencil and coefficients are updated, then the standard interpolation formulas can be used:

$$\begin{aligned}
\tilde{u}_h(I) &= \sum_{0 \le i,j,k \le 2} c_i^x\, c_j^y\, c_k^z\, u_{ijk}, & \frac{\partial \tilde{u}_h(I)}{\partial x} &= \sum_{0 \le i,j,k \le 2} c_i'^x\, c_j^y\, c_k^z\, u_{ijk}, \\
\frac{\partial \tilde{u}_h(I)}{\partial y} &= \sum_{0 \le i,j,k \le 2} c_i^x\, c_j'^y\, c_k^z\, u_{ijk}, & \frac{\partial \tilde{u}_h(I)}{\partial z} &= \sum_{0 \le i,j,k \le 2} c_i^x\, c_j^y\, c_k'^z\, u_{ijk},
\end{aligned} \tag{53}$$

where u_{ijk} is the grid function over the (updated) stencil s_{ijk}.

Although this strategy may seem cumbersome, from the implementation point of view it is very compact and efficient.

Finally, observe that conditions (39) become:

$$\mu_D \, \Delta t < 1, \quad \frac{\mu_N \, \Delta t}{h} < \frac{2}{3\sqrt{3}\,\beta^{\pm}}, \tag{54}$$

since the supremum of the 3D version of (71) is obtained for $\vartheta_x = \vartheta_y = \vartheta_z = 0$ and $|n_x| = |n_y| = |n_z| = \sqrt{3}/3$. In the general $d-$dimensional case, the conditions are:

$$\mu_D \, \Delta t < 1, \quad \frac{\mu_N \, \Delta t}{h} < \frac{2}{3\sqrt{d}\,\beta^{\pm}}. \tag{55}$$

7 Numerical tests

In this section we perform different numerical tests to show the second-order accuracy in the solution and its gradient and the efficiency of the multigrid. The gradient is computed in the inner grid points with the regular central difference approximation of the derivatives (eventually making use of the value computed at ghost points). In detail, Examples 1 (circular domains) and 2 (flower-shaped domains) show the accuracy of the discretization. The tolerance for the stopping criterion of the multigrid is small enough in order to make the error of the iterative scheme negligible with respect to the one associated with the discretization error. Therefore, the error of the numerical results is (almost) entirely due to the discretization error, which decays with second order accuracy. Although second order accuracy in both the solution and the gradient is achieved, large errors are observed when $\beta^- \gg \beta^+$ due to the artificiality of the numerical tests, as demonstrated in Example 3 (where more realistic data is chosen). Example 4 is related to the efficiency of the multigrid for high-jump coefficients. In order to avoid numerical instability associated with the machine precision, we test the multigrid efficiency on the homogeneous problem (starting with an initial guess different from zero) and analyze the convergence factor towards the exact (null) solution. Example 5 shows the performance of the method in the case of matrix coefficient, while Example 6 presents some results in 3D.

The implementation of the numerical method has been carried out in Matlab for 2D problems and C++ for 3D problems.

7.1 Example 1: circular domains

Let us consider the model problem (1) with the following data:

$$\phi^{\Gamma}(x,y) = \sqrt{(x-x_0)^2 + (y-y_0)^2} - R_1,$$
$$\phi(x,y) = \sqrt{(x-x_0)^2 + (y-y_0)^2} - R_2,$$
$$\beta^- = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y), \quad \beta^+ = 1 + 0.5\sin(2\pi x)\cos(4\pi y) \tag{56}$$

or

$$\beta^- = 1 + 0.5\sin(2\pi x)\cos(4\pi y), \quad \beta^+ = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y). \tag{57}$$

Functions $f^{\pm}$, g_D, g_N and g are chosen in such a way the exact solution is the following:

$$u^- = \sin(4\pi x)\cos(6\pi y), \quad u^+ = \cos(2\pi x)\sin(3\pi y).$$

We choose $x_0 = \sqrt{2}/30$, $y_0 = \sqrt{3}/40$, $R_1 = 0.353$ and $R_2 = 0.753$. The domain is represented in Fig. 9 (left side). We perform one test with (56) and one test with (57). In Tables 1 and 2 we list the errors of the solution and its gradient in the L^1 and L^∞ norms, as well as the condition number κ of the linear system, while Fig. 10 shows the related bestfit lines. Second order accuracy is attained in both the solution and its gradient, and the errors are almost aligned with the best-fit line, highlighting the robustness of the method even with variable coefficients (with jump ratio up to one million).

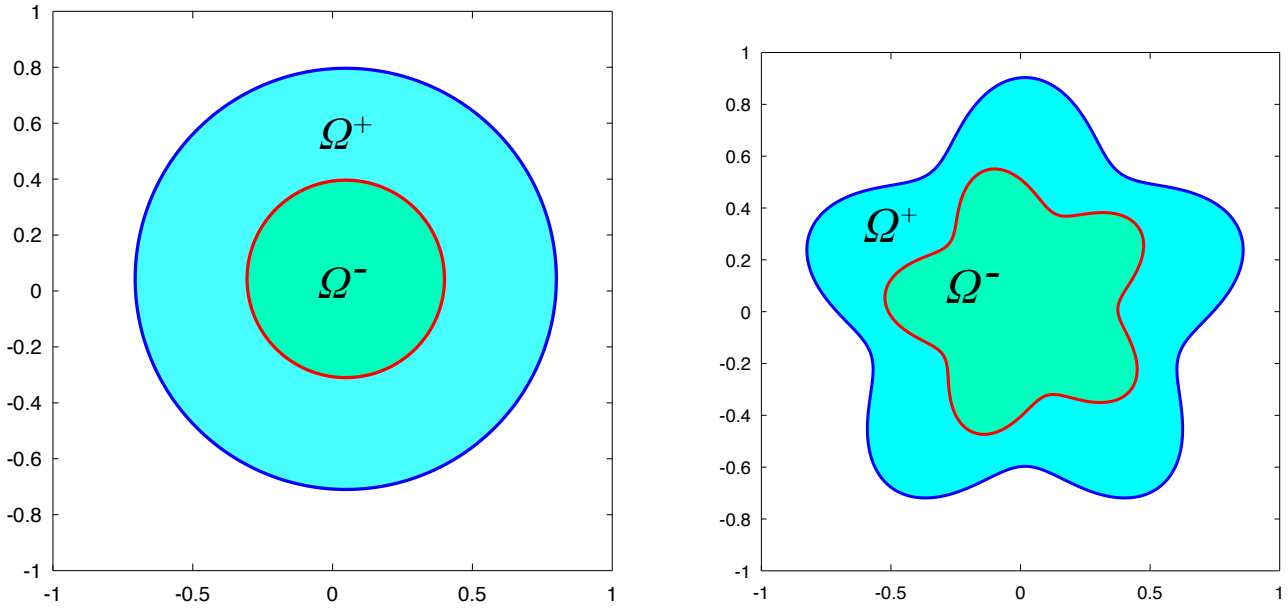


Fig. 9: Domains Ω^- and Ω^+ of the Examples 7.1 and 7.3 (left), 7.2 and 7.4 (right).

7.1.1 Large errors in artificial tests

We observe in Table 1 that, although second order accuracy is observed, very large errors are actually displayed compared to the exact solution, even for a reasonable number of grid points. This phenomenon is attributable to the artificial aspects of the test rather than to an issue of the numerical method, since data $f^{\pm}$, g_D, g_N and g are defined artificially from the chosen exact solution. In fact, since we choose an exact solution u of $O(1)$ and the coefficient β^- of the internal domain Ω^- is much larger than the coefficient β^+ of the domain Ω^+, some data (such as f) may develop high jumps across the interface, leading to large errors compared to the exact solution, although decaying with second order of accuracy (a possible explanation is given in Appendix 2). This phenomenon is only present when the coefficient β^- of the internal domain Ω^- is much larger than the coefficient β^+ of the domain Ω^+ (in fact it does not appear in Table 2). A similar behavior is observed, for example, in [47]. In real-life applications, $f^{\pm}$, g_D, g_N and g are set up according to the specific application and then they do not show high jumps (they are usually of $O(1)$). In these cases, the phenomenon of large errors is not observed, as supported by the numerical test proposed in Sect. 7.3.

7.2 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:

$$\text{flower}_0(r_0, r_1, x_0, y_0; x, y) = r - r_0 - r_1 \frac{(y-y_0)^5 + 5(x-x_0)^4(y-y_0) - 10(x-x_0)^2(y-y_0)^3}{r^5}.$$

Table 1: Example 7.1. Accuracy order in the solution (top) and in the gradient (bottom) for the case (56).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order	κ
32×32	$8.34 \cdot 10^{3}$	-	$7.70 \cdot 10^{4}$	-	$6.98 \cdot 10^{8}$
64×64	$2.07 \cdot 10^{3}$	2.01	$1.85 \cdot 10^{4}$	2.06	$2.43 \cdot 10^{9}$
128×128	$5.79 \cdot 10^{2}$	1.84	$5.10 \cdot 10^{3}$	1.86	$9.21 \cdot 10^{10}$
256×256	$1.46 \cdot 10^{2}$	1.99	$1.28 \cdot 10^{3}$	2.00	$3.66 \cdot 10^{10}$
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order	κ/n^2
32×32	$1.46 \cdot 10^{5}$	-	$3.90 \cdot 10^{5}$	-	$6.82 \cdot 10^{5}$
64×64	$3.49 \cdot 10^{4}$	2.06	$1.06 \cdot 10^{5}$	1.88	$5.94 \cdot 10^{5}$
128×128	$9.56 \cdot 10^{3}$	1.87	$2.96 \cdot 10^{4}$	1.84	$5.62 \cdot 10^{5}$
256×256	$2.39 \cdot 10^{3}$	2.00	$7.45 \cdot 10^{3}$	1.99	$5.58 \cdot 10^{5}$

Table 2: Example 7.1. Accuracy order in the solution (top) and in the gradient (bottom) for the case (57).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order	κ
32×32	$4.40 \cdot 10^{-3}$	-	$1.22 \cdot 10^{-1}$	-	$3.33 \cdot 10^{9}$
64×64	$1.02 \cdot 10^{-3}$	2.11	$2.93 \cdot 10^{-2}$	2.06	$9.85 \cdot 10^{9}$
128×128	$3.29 \cdot 10^{-4}$	1.64	$7.61 \cdot 10^{-3}$	1.95	$3.45 \cdot 10^{10}$
256×256	$8.24 \cdot 10^{-5}$	2.00	$2.14 \cdot 10^{-3}$	1.83	$1.29 \cdot 10^{11}$
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order	κ/n^2
32×32	$3.93 \cdot 10^{-1}$	-	$3.52 \cdot 10^{0}$	-	$3.25 \cdot 10^{6}$
64×64	$9.95 \cdot 10^{-2}$	1.98	$9.71 \cdot 10^{-1}$	1.86	$2.40 \cdot 10^{6}$
128×128	$2.63 \cdot 10^{-2}$	1.92	$2.80 \cdot 10^{-1}$	1.80	$2.10 \cdot 10^{6}$
256×256	$6.60 \cdot 10^{-3}$	1.99	$7.32 \cdot 10^{-2}$	1.93	$1.96 \cdot 10^{6}$

where $r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$. A rotated version (counter-clockwise by an angle ϑ_0) of this interface is obtained by considering the following level-set function:

$$\text{flower}(r_0, r_1, x_0, y_0, \vartheta_0; x, y) = \text{flower}_0(r_0, r_1, x_0, y_0; x^*, y^*),$$

where

$$x^* = \cos(\vartheta_0)x - \sin(\vartheta_0)y,$$
$$y^* = \sin(\vartheta_0)x + \cos(\vartheta_0)y.$$

Let us consider the model problem (1) with the following data:

$$\phi^\Gamma(x,y) = \text{flower}(0.45, 1/12, 0.01\sqrt{3}, 0.02\sqrt{2}, -\pi/12; x, y),$$
$$\phi(x,y) = \text{flower}(0.75, 1/8, 0.01\sqrt{3}, 0.02\sqrt{2}, 0; x, y),$$
$$\beta^- = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y), \quad \beta^+ = 1 + 0.5\sin(2\pi x)\cos(4\pi y) \tag{58}$$

or

$$\beta^- = 1 + 0.5\sin(2\pi x)\cos(4\pi y), \quad \beta^+ = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y). \tag{59}$$

Functions $f^\pm$, g_D, g_N and g are chosen in such a way the exact solution is the following:

$$u^- = \sin(4\pi x)\cos(6\pi y), \quad u^+ = \cos(2\pi x)\sin(3\pi y),$$

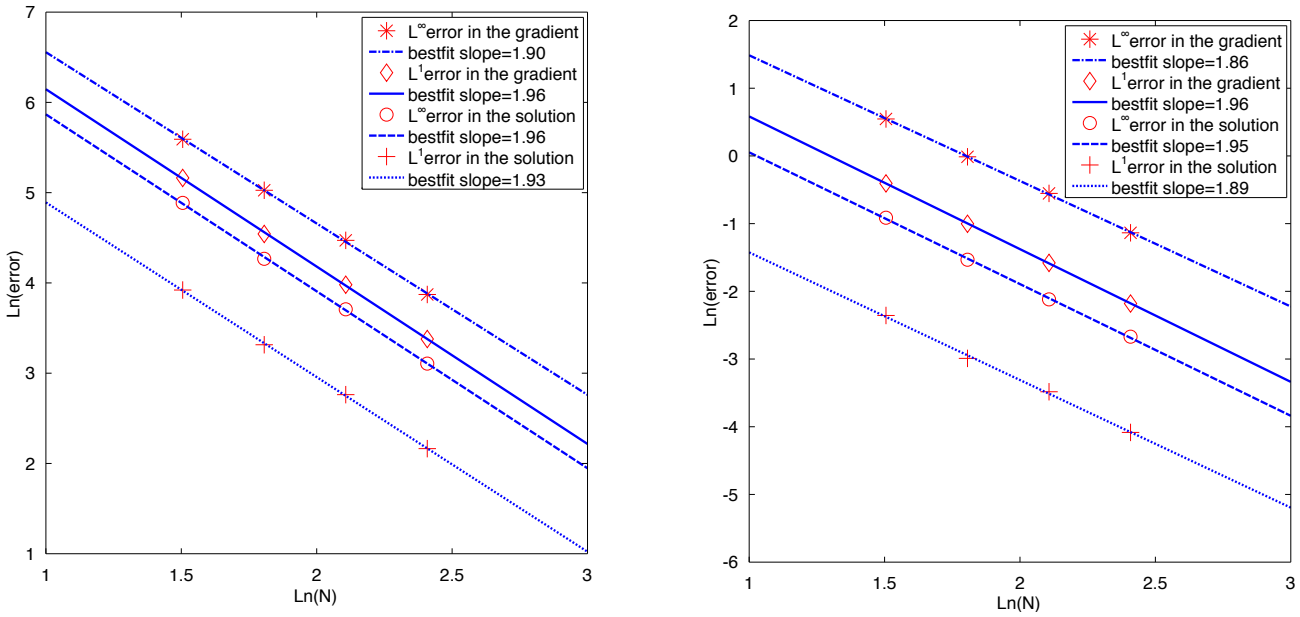


Fig. 10: Example 7.1. Bestfit lines of the errors in the solution and in the gradient (Tables 1 and 2) in both the L^1 and L^∞ norms. Left: β^- and β^+ are given by (56); Right: β^- and β^+ are given by (57).

The domain is represented in Fig. 9 (right side). We perform one test with (58) and one test with (59). In Tables 3 and 4 we list the errors of the solution and its gradient in the L^1 and L^∞ norms, as well as the condition number κ of the linear system. Fig. 11 shows the related bestfit lines.

As in the previous case, also in the presence of complex interface/boundary the method is robust and second order accurate in both the solution and its gradient. The phenomenon of large errors in Table 3 is similar to the one observed in Example 1 and the explanation is given in Sect. 7.1.1 and Appendix 2. The numerical test in Sect. 7.3 demonstrates that this issue is related with the artificial aspect of the numerical test rather than with the discretization method.

Table 3: Example 7.2. Accuracy order in the solution (top) and in the gradient (bottom) for the case (58).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order	κ
32×32	$6.99 \cdot 10^3$	-	$5.77 \cdot 10^4$	-	$4.53 \cdot 10^8$
64×64	$1.17 \cdot 10^3$	2.58	$9.27 \cdot 10^3$	2.64	$1.66 \cdot 10^9$
128×128	$5.69 \cdot 10^2$	1.04	$4.32 \cdot 10^3$	1.10	$6.01 \cdot 10^9$
256×256	$7.61 \cdot 10^1$	2.90	$5.65 \cdot 10^2$	2.93	$2.39 \cdot 10^{10}$
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order	κ/n^2
32×32	$1.23 \cdot 10^5$	-	$5.02 \cdot 10^5$	-	$4.43 \cdot 10^5$
64×64	$1.95 \cdot 10^4$	2.65	$8.69 \cdot 10^4$	2.53	$4.05 \cdot 10^5$
128×128	$9.30 \cdot 10^3$	1.07	$4.38 \cdot 10^4$	0.99	$3.67 \cdot 10^5$
256×256	$1.23 \cdot 10^3$	2.92	$6.26 \cdot 10^3$	2.80	$3.65 \cdot 10^5$

Table 4: Example 7.2. Accuracy order in the solution (top) and in the gradient (bottom) for the case (59).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order	κ
32×32	$6.63 \cdot 10^{-3}$	-	$2.51 \cdot 10^{-1}$	-	$4.93 \cdot 10^{9}$
64×64	$2.49 \cdot 10^{-3}$	1.41	$8.18 \cdot 10^{-2}$	1.62	$1.60 \cdot 10^{10}$
128×128	$5.02 \cdot 10^{-4}$	2.31	$1.66 \cdot 10^{-2}$	2.30	$5.83 \cdot 10^{10}$
256×256	$1.28 \cdot 10^{-4}$	1.98	$4.03 \cdot 10^{-3}$	2.04	$2.17 \cdot 10^{11}$
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order	κ/n^2
32×32	$6.70 \cdot 10^{-1}$	-	$4.30 \cdot 10^{0}$	-	$4.82 \cdot 10^{6}$
64×64	$1.91 \cdot 10^{-1}$	1.81	$1.19 \cdot 10^{0}$	1.86	$3.91 \cdot 10^{6}$
128×128	$4.64 \cdot 10^{-2}$	2.04	$3.31 \cdot 10^{-1}$	1.84	$3.56 \cdot 10^{6}$
256×256	$1.18 \cdot 10^{-2}$	1.97	$1.23 \cdot 10^{-1}$	1.43	$3.31 \cdot 10^{6}$

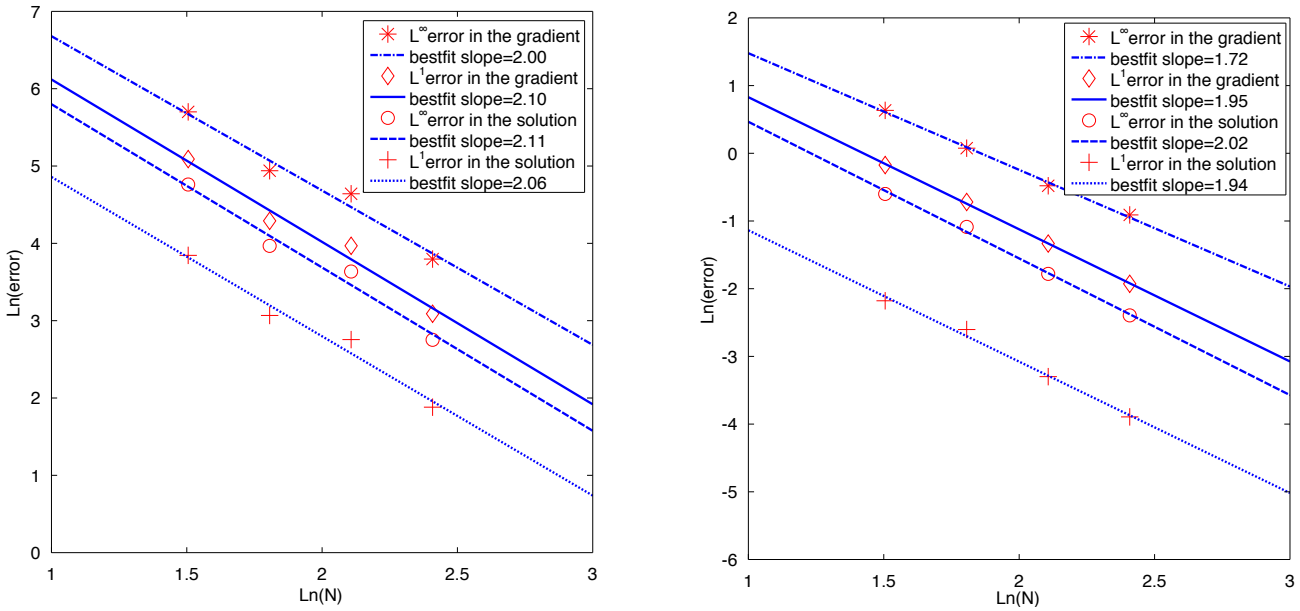


Fig. 11: Example 7.2. Bestfit lines of the errors in the solution and in the gradient (Tables 3 and 4) in both the L^1 and L^∞ norms. Left: β^- and β^+ are given by (58); Right: β^- and β^+ are given by (58).

7.3 Example 3: accuracy test with more realistic data $f^\pm$, g_D, g_N and g

In this test we show that the large errors present in Tables 1 and 3 are actually a consequence of the artificial aspect of the numerical tests rather than an issue of the discretization method itself. To this purpose, we choose more realistic values for the right-hand sides $f^\pm$, g_D, g_N and g, rather than computing them by the exact solution (which can lead to high jumps in f and then to large numerical errors). In absence of the exact solution, we compute the errors by comparing the numerical solution with a reference solution rather than with the exact solution (which is not available here). In this test, the reference solution is the numerical solution with a sufficiently large number of grid points (1024×1024 in our case). The reference gradient of the solution is computed by standard central difference schemes on the reference solution.

Let us consider the model problem (1) with the following data:

$$\begin{gathered}\phi^{\Gamma}(x,y) = \sqrt{(x-x_0)^2+(y-y_0)^2} - R_1,\\ \phi(x,y) = \sqrt{(x-x_0)^2+(y-y_0)^2} - R_2,\\ \beta^- = 10^6 + 10^5\sin(\pi x)\cos(3\pi y), \quad \beta^+ = 1 + 0.5\sin(2\pi x)\cos(4\pi y),\\ f^- = \sin(4\pi x)\cos(6\pi y), \quad f^+ = \cos(2\pi x)\sin(3\pi y),\\ g_D = g_N = g = 0.\end{gathered} \tag{60}$$

As in Example 7.1, we choose $x_0 = \sqrt{2}/30$, $y_0 = \sqrt{3}/40$, $R_1 = 0.353$ and $R_2 = 0.753$ (the domain is represented in Fig. 9, left side).

Table 5: Example 7.3. Accuracy order in the solution (top) and in the gradient (bottom).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order
32×32	$4.97 \cdot 10^{-5}$	-	$6.26 \cdot 10^{-4}$	-
64×64	$8.43 \cdot 10^{-6}$	2.56	$1.05 \cdot 10^{-4}$	2.58
128×128	$1.92 \cdot 10^{-6}$	2.13	$2.22 \cdot 10^{-5}$	2.24
256×256	$3.63 \cdot 10^{-7}$	2.40	$4.60 \cdot 10^{-6}$	2.27
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
32×32	$1.88 \cdot 10^{-3}$	-	$1.30 \cdot 10^{-2}$	-
64×64	$4.52 \cdot 10^{-4}$	2.06	$4.76 \cdot 10^{-3}$	1.45
128×128	$1.10 \cdot 10^{-4}$	2.04	$1.06 \cdot 10^{-3}$	2.17
256×256	$2.57 \cdot 10^{-5}$	2.09	$2.48 \cdot 10^{-4}$	2.09

As can be seen in Table 5, relative errors are now reasonable without compromising the second order accuracy in the solution and the gradient. Fig. 12 (left) shows the associated bestfit lines. The reference solution (obtained with 1024×1024 grid points) is displayed in Fig. 12 (right).

7.4 Example 4: High-jump coefficients and multigrid efficiency

In this example we show that the asymptotic convergence factor of the multigrid algorithm does not depend on the jump of the coefficient nor on the size of the problem. In particular, we will see that the convergence factor is close to the one predicted by the Local Fourier Analysis for inner equations and detailed in Table 6. As we pointed out in Sect. 4.1.3, we know that more efficient smoothers than GS-LEX exist (such as GS-RB), but the goal of this work is to show that the optimal convergence factor is attained, regardless on the smoother adopted. 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 values of Table 6. 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 it slightly degrades when reaching asymptotic convergence.

Table 6: Predicted convergence factor ρ_{loc} by LFA for GS-LEX and FW restriction operator (see, for instance, [61, Ch. 4.6.1]).

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

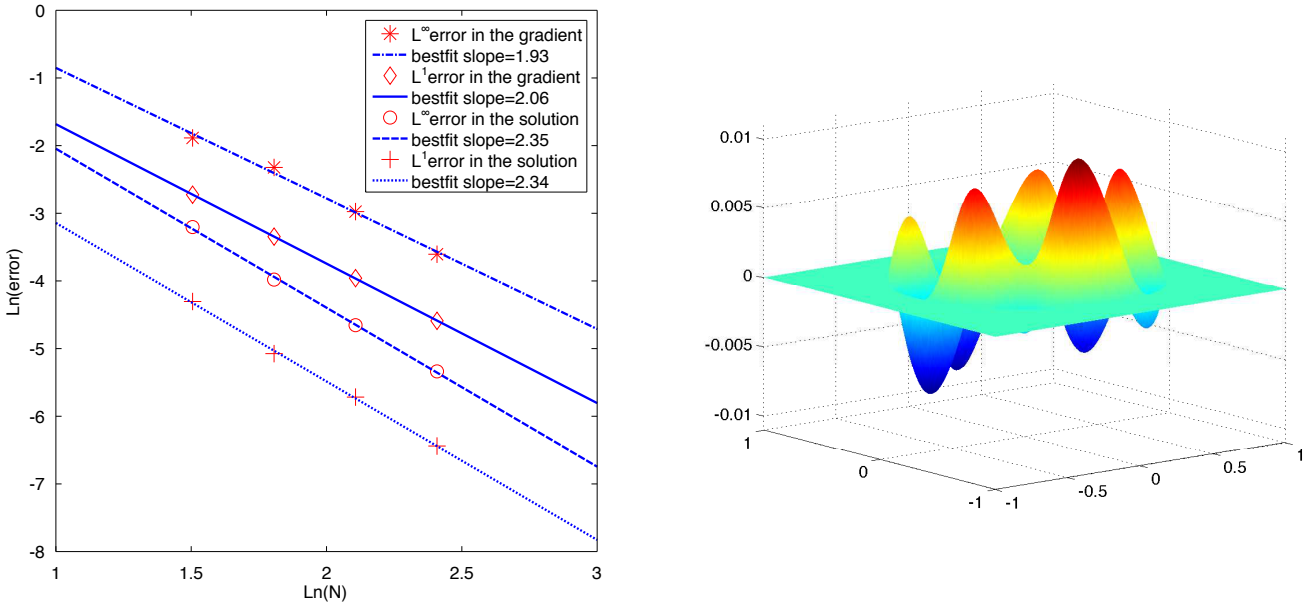


Fig. 12: Example 7.3. Left: bestfit lines of the errors in the solution and in the gradient (Table 5) in both the L^1 and L^∞ norms. Right: reference solution obtained with 1024×1024 grid points.

Let us recall that we estimate the asymptotic convergence factor as:

$$\rho = \lim_{m\to\infty} \rho^{(m)} = \lim_{m\to\infty} \frac{\left\| r_h^{(m)} \right\|_\infty}{\left\| r_h^{(m-1)} \right\|_\infty},$$

where $r_h = \left(r_h^{\Omega^-}, r_h^{\Omega^+}, r_h^{\Gamma^-}, r_h^{\Omega^+}, r_h^{\Gamma} \right)$. In practice, we compute $\rho^{(m)}$ until the following stopping criterion is satisfied:

$$\frac{\left| \rho^{(m)} - \rho^{(m-1)} \right|}{\rho^{(m)}} < 10^{-3}. \tag{61}$$

We compare this convergence factor with the average convergence factor of the first ten W-cycle iterations, computed as follows:

$$\bar{\rho} = \sqrt[9]{\prod_{m=2}^{10} \rho^{(m)}}. \tag{62}$$

In this example, we use the same geometry as in Example 7.2 (namely the flower-shaped domains), with coefficients:

$$\beta^- = 10^p, \quad \beta^+ = 1.$$

We solve the homogeneous problem (starting with an initial guess different from zero), namely the Problem (1) with $f^\pm = g_D = g_N = g = 0$, in order to avoid numerical instability associated with the machine precision. We use the W-cycle algorithm with $\nu_1 = 2$ pre-smoothing and $\nu_2 = 1$ post-smoothing relaxations (therefore $\nu = 3$ in Table 6), and with a coarsest grid of 16×16 grid points. Tables 7 and 8 show the estimated convergence factors for different numbers of grid points and jumps in the coefficient. As we can see from Tables 7 and 8, the average

convergence factor of the method is almost comparable with the result predicted by the Local Fourier Analysis, and then the method is very effective on such problems. We observe that in some cases the convergence factor is even less than the predicted one. A possible explanation for this phenomenon lies in the choice of the parameter λ and δ in Eq. (40). In fact, this choice is performed at every level of the multigrid, even for coarser grids, where the choice $\delta = 5\,h$ leads to perform the extra-relaxation steps on the whole domain, and then the actual value of iteration steps $\nu = \nu_1 + \nu_2$ is higher than three on those coarser levels. This phenomenon improves the overall efficiency of the multigrid with some extra computational cost.

Finally, it is worth to observe that if the choice (35)–(38) is performed in the opposite way (i.e. (36),(38) if $\beta^+(I) > \beta^-(I)$, and (35),(37) otherwise), then the convergence factor degrades to $\rho \approx 1$ (not shown).

Table 7: Example 7.4. Asymptotic convergence factor, computed with the stop criterion (61) ($\nu = \nu_1 + \nu_2 = 3$).

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 8: Example 7.4. Average convergence factor for the first ten W-cycle iterations, computed by the formula (62) ($\nu = \nu_1 + \nu_2 = 3$).

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

7.5 Example 5: Matrix coefficient case

In this section we perform a numerical test in the case of a matrix coefficient (described in Sect. 5).

Let us consider the model problem (48) with the following data:

$$\phi^{\Gamma}(x,y) = \sqrt{(x-x_0)^2 + (y-y_0)^2} - R_1,$$

$$\phi(x,y) = \sqrt{(x-x_0)^2 + (y-y_0)^2} - R_2,$$

$$f^- = \sin(4\pi x)\cos(6\pi y), \quad f^+ = \cos(2\pi x)\sin(3\pi y),$$
$$g_D = g_N = g = 0.$$

The matrix coefficient β is expressed by the following coefficients (see (49)):

$$\beta_{11}^- = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y), \quad \beta_{12}^- = 10^5 + 10^4 \sin(3\pi x)\cos(2\pi y), \quad \beta_{22}^- = 10^6 + 10^5 \sin(2\pi x)\cos(4\pi y),$$
$$\beta_{11}^+ = 1 + 0.5\sin(2\pi x)\cos(4\pi y), \quad \beta_{12}^+ = 0.1 + 0.05\sin(3\pi x)\cos(2\pi y), \quad \beta_{22}^+ = 1 + 0.5\sin(4\pi x)\cos(\pi y), \tag{63}$$

or

$$\beta_{11}^- = 1 + 0.5\sin(2\pi x)\cos(4\pi y), \quad \beta_{12}^- = 0.1 + 0.05\sin(3\pi x)\cos(2\pi y), \quad \beta_{22}^- = 1 + 0.5\sin(4\pi x)\cos(\pi y),$$
$$\beta_{11}^+ = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y), \quad \beta_{12}^+ = 10^5 + 10^4 \sin(3\pi x)\cos(2\pi y), \quad \beta_{22}^+ = 10^6 + 10^5 \sin(2\pi x)\cos(4\pi y). \tag{64}$$

As in Example 7.1, we choose $x_0 = \sqrt{2}/30$, $y_0 = \sqrt{3}/40$, $R_1 = 0.353$ and $R_2 = 0.753$ (the domain is represented in Fig. 9, left side).

Table 9: Example 7.5. Accuracy order in the solution (top) and in the gradient (bottom) for the case (63).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order	κ
32×32	$3.71 \cdot 10^{-5}$	-	$6.31 \cdot 10^{-4}$	-	$6.65 \cdot 10^{8}$
64×64	$5.57 \cdot 10^{-6}$	2.74	$7.94 \cdot 10^{-5}$	2.99	$2.21 \cdot 10^{9}$
128×128	$2.43 \cdot 10^{-6}$	1.20	$2.76 \cdot 10^{-5}$	1.53	$8.26 \cdot 10^{9}$
256×256	$3.22 \cdot 10^{-7}$	2.91	$4.49 \cdot 10^{-6}$	2.62	$3.29 \cdot 10^{10}$
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order	κ/n^2
32×32	$1.69 \cdot 10^{-3}$	-	$1.39 \cdot 10^{-2}$	-	$6.49 \cdot 10^{5}$
64×64	$3.96 \cdot 10^{-4}$	2.10	$5.28 \cdot 10^{-3}$	1.39	$5.41 \cdot 10^{5}$
128×128	$1.00 \cdot 10^{-4}$	1.98	$1.26 \cdot 10^{-3}$	2.07	$5.04 \cdot 10^{5}$
256×256	$2.27 \cdot 10^{-5}$	2.14	$4.09 \cdot 10^{-4}$	1.62	$5.02 \cdot 10^{5}$

Table 10: Example 7.5. Accuracy order in the solution (top) and in the gradient (bottom) for the case (64).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order	κ
32×32	$4.53 \cdot 10^{-6}$	-	$4.91 \cdot 10^{-4}$	-	$3.45 \cdot 10^{9}$
64×64	$1.02 \cdot 10^{-6}$	2.16	$1.15 \cdot 10^{-4}$	2.10	$1.03 \cdot 10^{10}$
128×128	$2.54 \cdot 10^{-7}$	2.00	$2.81 \cdot 10^{-5}$	2.03	$3.52 \cdot 10^{10}$
256×256	$6.13 \cdot 10^{-8}$	2.05	$6.65 \cdot 10^{-6}$	2.08	$1.34 \cdot 10^{11}$
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order	κ/n^2
32×32	$6.95 \cdot 10^{-4}$	-	$1.08 \cdot 10^{-2}$	-	$3.37 \cdot 10^{6}$
64×64	$1.73 \cdot 10^{-4}$	2.01	$2.89 \cdot 10^{-3}$	1.90	$2.50 \cdot 10^{6}$
128×128	$4.37 \cdot 10^{-5}$	1.98	$9.10 \cdot 10^{-4}$	1.67	$2.15 \cdot 10^{6}$
256×256	$1.04 \cdot 10^{-5}$	2.07	$2.08 \cdot 10^{-4}$	2.13	$2.05 \cdot 10^{6}$

In Tables 9 and 10 we list the errors of the solution and its gradient in the L^1 and L^∞ norms, as well as the condition number κ of the linear system, while Fig. 13 shows the related bestfit lines. The reference solution is the numerical solution with a sufficiently fine grid (1024×1024 grid points). Second order accuracy is attained in both the solution and its gradient.

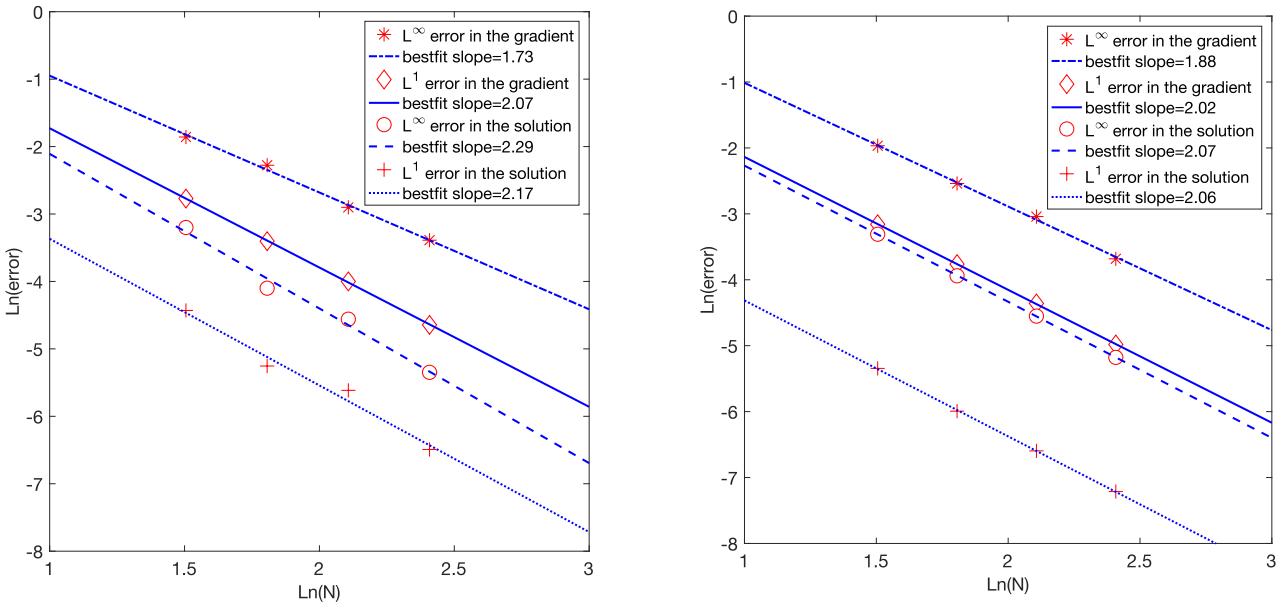


Fig. 13: Example 7.5. Bestfit lines of the errors in the solution and in the gradient (Tables 9 and 10) in both the L^1 and L^∞ norms. Left: β^- and β^+ are given by (63); Right: β^- and β^+ are given by (64).

7.6 Example 6: A numerical test in 3D

In this section we consider the extension to the 3D case. The domains Ω^- and Ω are two spheres and the respective level set functions are expressed by:

$$\phi^\Gamma(x,y,z) = \sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2} - R_1,$$

$$\phi(x,y,z) = \sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2} - R_2.$$

Let us consider the 3D version of the model problem (1) with the following data:

$$\beta^- = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y)\sin(2\pi z), \quad \beta^+ = 1 + 0.5\sin(2\pi x)\cos(4\pi y)\sin(3\pi z) \tag{65}$$

or

$$\beta^- = 1 + 0.5\sin(2\pi x)\cos(4\pi y)\sin(3\pi z), \quad \beta^+ = 10^6 + 10^5 \sin(\pi x)\cos(3\pi y)\sin(2\pi z). \tag{66}$$

Functions $f^\pm$, g_D, g_N and g are chosen in such a way the exact solution is the following:

$$u^- = \sin(3\pi x)\sin(\pi y)\sin(2\pi z), \quad u^+ = \sin(\pi x)\sin(4\pi y)\sin(3\pi z).$$

We choose $x_0 = y_0 = z_0 = 0$, $R_1 = 0.653$ and $R_2 = 0.873$. We perform one test with (65) and one test with (66). Due to the high computational cost of 3D problems, numerical tests with many grid points can only be investigated if the numerical method is implemented through parallel programming. The parallelization of the numerical method is under investigation and beyond the scope of this paper. To investigate the numerical accuracy using a non-parallel code, we do not use more than 150^3 grid points. Therefore, in order to have a sufficient number of tests to compute the accuracy order, we choose the number of grid points $(N+1)^3$ by the following formula:

$$N+1 = \lceil 25 \times 1.4^j \rceil, \qquad j = 0, \ldots, 5,$$

Table 11: Example 7.6. Accuracy order in the solution (top) and in the gradient (bottom) for the case (65).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order
$25 \times 25 \times 25$	$2.99 \cdot 10^{-3}$	-	$1.06 \cdot 10^{-1}$	-
$35 \times 35 \times 35$	$1.43 \cdot 10^{-3}$	2.18	$5.07 \cdot 10^{-2}$	2.20
$49 \times 49 \times 49$	$7.11 \cdot 10^{-4}$	2.08	$2.58 \cdot 10^{-2}$	2.01
$69 \times 69 \times 69$	$3.57 \cdot 10^{-4}$	2.05	$1.24 \cdot 10^{-2}$	2.17
$97 \times 97 \times 97$	$1.82 \cdot 10^{-4}$	2.00	$6.56 \cdot 10^{-3}$	1.90
$135 \times 135 \times 135$	$9.86 \cdot 10^{-5}$	1.83	$3.47 \cdot 10^{-3}$	1.89
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
$25 \times 25 \times 25$	$3.76 \cdot 10^{-1}$	-	$2.24 \cdot 10^{0}$	-
$35 \times 35 \times 35$	$1.95 \cdot 10^{-1}$	1.94	$1.09 \cdot 10^{0}$	2.14
$49 \times 49 \times 49$	$9.94 \cdot 10^{-2}$	2.01	$6.10 \cdot 10^{-1}$	1.72
$69 \times 69 \times 69$	$5.05 \cdot 10^{-2}$	2.01	$3.81 \cdot 10^{-1}$	1.40
$97 \times 97 \times 97$	$2.55 \cdot 10^{-2}$	2.03	$1.63 \cdot 10^{-1}$	2.52
$135 \times 135 \times 135$	$1.33 \cdot 10^{-2}$	1.93	$1.01 \cdot 10^{-1}$	1.44

Table 12: Example 7.6. Accuracy order in the solution (top) and in the gradient (bottom) for the case (66).

No. of points (n)	L^1 error of u	order	L^∞ error of u	order
$25 \times 25 \times 25$	$2.69 \cdot 10^{-3}$	-	$1.05 \cdot 10^{-1}$	-
$35 \times 35 \times 35$	$1.48 \cdot 10^{-3}$	1.77	$6.35 \cdot 10^{-2}$	1.49
$49 \times 49 \times 49$	$8.56 \cdot 10^{-4}$	1.63	$3.79 \cdot 10^{-2}$	1.54
$69 \times 69 \times 69$	$4.53 \cdot 10^{-4}$	1.89	$1.97 \cdot 10^{-2}$	1.94
$97 \times 97 \times 97$	$2.45 \cdot 10^{-4}$	1.82	$9.81 \cdot 10^{-3}$	2.08
$135 \times 135 \times 135$	$1.25 \cdot 10^{-4}$	2.00	$5.01 \cdot 10^{-3}$	2.00
No. of points (n)	L^1 error of $\lvert\nabla u\rvert$	order	L^∞ error of $\lvert\nabla u\rvert$	order
$25 \times 25 \times 25$	$4.18 \cdot 10^{-1}$	-	$2.14 \cdot 10^{0}$	-
$35 \times 35 \times 35$	$2.10 \cdot 10^{-1}$	2.04	$1.17 \cdot 10^{0}$	1.80
$49 \times 49 \times 49$	$1.11 \cdot 10^{-1}$	1.89	$8.13 \cdot 10^{-1}$	1.08
$69 \times 69 \times 69$	$5.58 \cdot 10^{-2}$	2.05	$3.66 \cdot 10^{-1}$	2.37
$97 \times 97 \times 97$	$2.86 \cdot 10^{-2}$	1.99	$2.15 \cdot 10^{-1}$	1.58
$135 \times 135 \times 135$	$1.46 \cdot 10^{-2}$	2.01	$1.16 \cdot 10^{-1}$	1.85

where $\lceil \cdot \rceil$ is the ceiling function. The exponential formula guarantees that the values of N are almost uniformly distributed in logarithmic scale (see Fig. 14).

In Tables 11 and 12 we list the errors of the solution and its gradient in the L^1 and L^∞ norms, while Fig. 14 shows the related bestfit lines. Second order accuracy is attained in both the solution and its gradient, and the errors are almost aligned with the best-fit line, highlighting the robustness of the method even with variable coefficients (with jump ratio up to one million).

8 Limitations and Conclusion

A ghost-point finite difference method to solve elliptic equations with discontinuous coefficients (with general non-homogeneous jumps in the solutions and its gradient) is presented. The method is second order accurate both in the

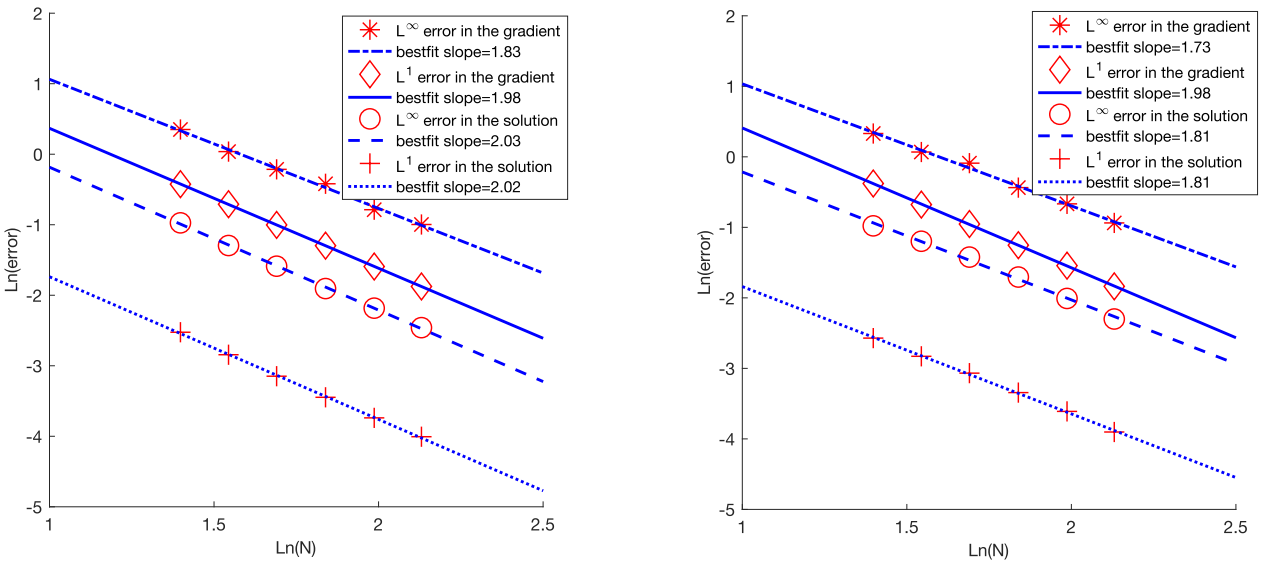


Fig. 14: Example 7.6. Bestfit lines of the errors in the solution and in the gradient (Tables 11 and 12) in both the L^1 and L^∞ norms. Left: β^- and β^+ are given by (65); Right: β^- and β^+ are given by (66).

solution and in the gradient, and therefore it is suitable for real-life applications that require additional accuracy also in the gradient, such as Stefan problems or incompressible Navier-Stokes equations. The accuracy order is not influenced by high-jump coefficients and can be straightforwardly increased by using a higher order interpolation procedure on the interface and boundary. The linear system arising from the discretization is solved by a proper multigrid approach, whose convergence factor is close to the optimal one achieved by the Local Fourier Analysis for rectangular domain and it is not affected by high-jump coefficients. Numerous applications may benefit from the higher accuracy of this method and the efficiency of the multigrid solver, such as those mentioned in the introduction, especially for the 3D case. However, it is important to identify possible limitations of the numerical method in its current form, in order to drive future developments of the code. For example, the second order accuracy is observed only when the computational grid is sufficiently fine with respect to the interface and boundary curvatures. This means that a very fine grid is needed for very complex geometries, such as the flower-shaped domain with much sharper petals. The refined grid is needed only in the vicinity of the interface/boundary, and therefore an adaptive mesh refinement approach can be adopted in this case [25, 28, 56]. Moreover, parallelization of the code can drastically decrease the overall computational cost, especially for 3D problems. Cartesian meshes are well suitable for parallel implementations [52], although some aspects of the multigrid approach and the interface discretizations may need a dedicated development[60]. Extensions of the method to the case of adaptive Cartesian grids and High Performance Computing is subject of future research.

Acknowledgment

The work of A. Coco has been partially supported by the London Mathematical Society Computer Science Small Grants - Scheme 7. The work of G. Russo was supported in part by ITN-ETN Horizon 2020 Project ModCompShock, Modeling and Computation on Shocks and Interfaces, Project Reference 642768.

Appendix 1: Upper bounds for 2D interpolation coefficients

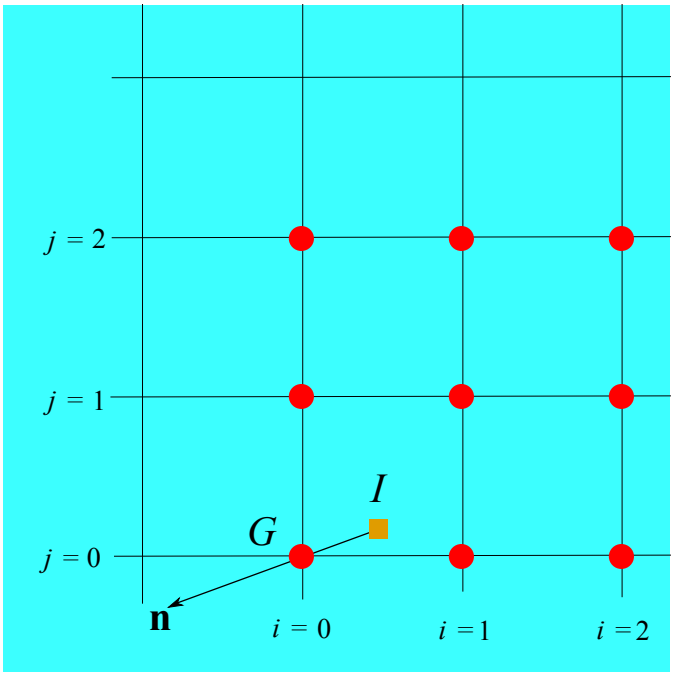


Fig. 15: Generic 3×3 point stencil (circles) where a grid function u_{ij} is interpolated. Point G (ghost point) is at the bottom-left corner point. Point I (square point) is where the interpolation and its derivatives are evaluated. Unit vector $\tilde{\mathbf{n}}$ represents the normal vector in Sect. 3 and then is almost parallel to $G - I$. In the Appendix, point I is in the bottom-left quadrant and $\mathbf{n} = (n_x, n_y)$ is such that $n_x, n_y \leq 0$, but Eqs. (69), (70), (72), (73) are also valid in all the other three cases.

In this appendix we prove the equations (24), (25), (28) and (29). Let St_9 be the 3×3 point stencil of Fig. 15 (we assume that G is at the bottom left of the stencil, since the other three cases are analogous)

$$St_9 = \left\{ G + h(i,j) \colon (i,j) \in \{0,1,2\}^2 \right\},$$

and $I \equiv (x_I, y_I)$ be a point such that $x_G \leq x_I \leq x_G + h$, $y_G \leq y_I \leq y_G + h$. Let u_{ij}, $(i,j) \in \{0,1,2\}^2$, be a grid function defined on the stencil St_9 and $\tilde{u}_h$ be the biquadratic interpolant of u_{ij} on the stencil St_9. Let $\vartheta_x = \dfrac{x_I - x_G}{h}$ and $\vartheta_y = \dfrac{y_I - y_G}{h}$. Observe that $0 \leq \vartheta_x, \vartheta_y \leq 1$. After some algebra, we have

$$\tilde{u}_h(I) = \sum_{0 \leq i,j \leq 2} c_i^x \, c_j^y \, u_{ij}, \qquad \frac{\partial \tilde{u}_h(I)}{\partial x} = \sum_{0 \leq i,j \leq 2} c_i'^x \, c_j^y \, u_{ij}, \qquad \frac{\partial \tilde{u}_h(I)}{\partial y} = \sum_{0 \leq i,j \leq 2} c_i^x \, c_j'^y \, u_{ij}, \tag{67}$$

where

$$(c_0^x, c_1^x, c_2^x) = \left(\frac{(1-\vartheta_x)(2-\vartheta_x)}{2}, \vartheta_x(2-\vartheta_x), \frac{(\vartheta_x - 1)\vartheta_x}{2} \right),$$

$$(c_0^y, c_1^y, c_2^y) = \left(\frac{(1-\vartheta_y)(2-\vartheta_y)}{2}, \vartheta_y(2-\vartheta_y), \frac{(\vartheta_y - 1)\vartheta_y}{2} \right),$$

$$(c_0'^x, c_1'^x, c_2'^x) = \frac{1}{h} \left(\vartheta_x - \frac{3}{2}, 2(1-\vartheta_x), \vartheta_x - \frac{1}{2} \right),$$

$$(c_0'^y, c_1'^y, c_2'^y) = \frac{1}{h}\left(\vartheta_y - \frac{3}{2}, 2(1-\vartheta_y), \vartheta_y - \frac{1}{2}\right).$$

Therefore (observe that $u_G = u_{00}$):

$$\frac{\partial \tilde{u}_h(I)}{\partial u_G} = c_0^x\, c_0^y = \frac{(1-\vartheta_x)(2-\vartheta_x)(1-\vartheta_y)(2-\vartheta_y)}{4}. \tag{68}$$

Since $\vartheta_x, \vartheta_y \leq 1$, we have:

$$\frac{\partial \tilde{u}_h(I)}{\partial u_G} \geq 0. \tag{69}$$

The supremum of (68) is attained for $\vartheta_x = \vartheta_y = 0$. Therefore:

$$\sup_{|G-I|\leq h} \frac{\partial \tilde{u}_h(I)}{\partial u_G} = 1. \tag{70}$$

Now, consider a generic unit vector $\tilde{\mathbf{n}} = (n_x, n_y)$ pointing towards the bottom-left quadrant, i.e. $n_x, n_y \leq 0$ and $n_x^2 + n_y^2 = 1$. Then:

$$\frac{\partial\, (\nabla \tilde{u}_h(I) \cdot \tilde{\mathbf{n}})}{\partial\, u_G} = \frac{\partial}{\partial\, u_G}\left(\frac{\partial \tilde{u}_h(I)}{\partial x} n_x + \frac{\partial \tilde{u}_h(I)}{\partial y} n_y\right) = c_0'^x\, c_0^y\, n_x + c_0^x\, c_0'^y\, n_y$$

$$= \frac{1}{h}\left(\frac{(3-2\vartheta_x)\,(1-\vartheta_y)(2-\vartheta_y)}{4}\,|n_x| + \frac{(1-\vartheta_x)(2-\vartheta_x)\,(3-2\vartheta_y)}{4}\,|n_y|\right). \tag{71}$$

Since $\vartheta_x, \vartheta_y \leq 1$, we have:

$$\frac{\partial\, (\nabla \tilde{u}_h(I) \cdot \tilde{\mathbf{n}})}{\partial\, u_G} \geq 0. \tag{72}$$

The supremum of (71) is obtained for $\vartheta_x = \vartheta_y = 0$ and $|n_x| = |n_y| = \sqrt{2}/2$. Therefore

$$\sup_{|G-I|\leq h} \frac{\partial\, (\nabla \tilde{u}_h(I) \cdot \tilde{\mathbf{n}})}{\partial\, u_G} = \frac{3}{\sqrt{2}\, h}. \tag{73}$$

Finally, we observe that the assumption that G is at the bottom-left corner does not lead the validity of Eqs. (69), (70), (72), (73), which are valid also in the three other cases.

Appendix 2: Error upper bounds for 1D elliptic equations with discontinuous coefficients

In this appendix we aim at justifying the large errors observed in Tables 1 and 3 (although they decay with second order of accuracy). The focus of this section is on the behaviour of the errors in presence of high jumps in the coefficients, which should not depend on the dimension of the problem (1D, 2D or 3D). Therefore, we simplify the analysis by focussing on the 1D problem and computing an upper bound for the error of elliptic equations with high jump coefficients.

Let us consider the 1D problem:

$$\begin{cases} -\dfrac{d}{dx}\left(\beta^{\pm}\dfrac{du^{\pm}}{dx}\right) &=& f^{\pm} \text{ in } \Omega^{\pm} \\ [\![u]\!] &=& g_D \text{ on } x=-1 \text{ and } x=1 \\ \left[\!\left[\text{sign}(x)\,\beta\dfrac{du}{dx}\right]\!\right] &=& g_N \text{ on } x=-1 \text{ and } x=1 \\ u &=& g \text{ on } x=-2 \text{ and } x=2. \end{cases} \tag{74}$$

In this 1D problem we have $\Omega^- = (-1,1)$ and $\Omega^+ = (-2,2)\backslash\Omega^-$. Let u_h be the discrete solution obtained by the 1D version of (41):

$$\begin{aligned} L_h(\beta_h, u_h) &= f_h \\ [\beta_h, u_h]_h^- &= g_h^- \\ [\beta_h, u_h]_h^+ &= g_h^+ \\ \mathcal{B}(u_h) &= g_h \end{aligned} \tag{75}$$

and $e_h = u - u_h$. Due to the linearity of the operators, we have:

$$\begin{aligned} L_h(\beta_h, e_h) &= L_h(\beta_h, u) - f_h \\ [\beta_h, e_h]_h^- &= [\beta_h, u]_h^- - g_h^- \\ [\beta_h, e_h]_h^+ &= [\beta_h, u]_h^+ - g_h^+ \\ \mathcal{B}(e_h) &= \mathcal{B}(u) - g_h \end{aligned} \tag{76}$$

Right-hand sides of Eq. (76) are the discretization errors of the operators. Therefore, e_h is a numerical approximation of the solution of the following problem:

$$\left\{ \begin{array}{rcl} -\dfrac{d}{dx}\left(\beta^{\pm}\dfrac{de^{\pm}}{dx}\right) &=& \beta^{\pm} C_1\, h^2 \text{ in } \Omega^{\pm} \\ [\![e]\!] &=& C_2\, h^3 \text{ on } x=-1 \text{ and } x=1 \\ \left[\!\left[\text{sign}(x)\,\beta\dfrac{de}{dx}\right]\!\right] &=& \max\left\{\beta^+,\beta^-\right\} C_3\, h^2 \text{ on } x=-1 \text{ and } x=1 \\ e &=& C_4\, h^3 \text{ on } x=-2 \text{ and } x=2 \end{array} \right. \tag{77}$$

Let us assume for simplicity that β^+ and β^- are two (possibly different) constants, and that the solution u of Eq. (74) and its derivatives up to order four are of $O(1)$, so that $C_1 - C_4$ are constants of $O(1)$. Then, the exact solutions of (77) are:

$$e^+(x) = -\frac{C_1\, h^2}{2}(2-|x|)^2 + \alpha(2-|x|)h^2 + C_4\, h^3, \qquad e^-(x) = -\frac{C_1\, h^2}{2}x^2 + (C_4 - C_2)h^3 + \alpha\, h^2, \tag{78}$$

with

$$\alpha = \frac{(\beta^+ + \beta^-)C_1 - \max\left\{\beta^+,\beta^-\right\} C_3}{\beta^+}.$$

From (78) we can infer that the error is in general $O(\alpha\, h^2)$. If $\beta^+ > \beta^-$, then α is $O(1)$, while if $\beta^- > \beta^+$ we observe that α is $O(\beta^-/\beta^+)$. Finally, the error is $O(h^2)$ when $\beta^+ > \beta^-$, and $O(\beta^-\, h^2)$ when $\beta^- > \beta^+$. If $\beta^- \gg \beta^+$ we observe that the error, although decays with second order of accuracy, may be very large for high value of h, and this explain the results of Tables 1 and 3.

References

[1] L. Adams and T. P. Chartier. New geometric immersed interface multigrid solvers. *SIAM Journal of Scientific Computing*, 25:1516–1533, 2004.

[2] 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.

[3] L. Adams and Z. Li. The immersed interface/multigrid methods for interface problems. *Journal of Scientific Computing*, 24:463–479, 2002.

[4] 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.

[5] P. Angot, C.-H. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numer. Math.*, 81, 1999.

[6] I. Babuška. The finite element method for elliptic equations with discontinuous coefficients. *Computing*, 5:207–213, 1970.

[7] 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.

[8] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, 2000.

[9] T. F. Chan and W. Wan. Robust multigrid methods for nonsmooth coefficient elliptic linear systems. *Journal of Computational and Applied Mathematics*, 123:323–352, 2000.

[10] 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.

[11] A. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 135:115–125, 1997.

[12] M. Cisternino and L. Weynans. A parallel second order cartesian method for elliptic interface problems. *Communications in Computational Physics*, 12(05):1562–1587, 2012.

[13] A. Coco, G. Currenti, C. D. Negro, and G. Russo. A Second Order Finite-Difference Ghost-Point Method for Elasticity Problems on unbounded domains with applications to Volcanology. *Communications in Computational Physics*, 16:983–1009, 2014.

[14] 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*, 5:19–42, 2012.

[15] A. Coco and G. Russo. Finite-Difference Ghost-Point Multigrid Methods on Cartesian Grids for Elliptic Problems in Arbitrary Domains. *Journal of Computational Physics*, 241:464–501, 2013.

[16] J. Dolbow and T. Belytschko. A finite element method for crack growth without remeshing. *International journal for numerical methods in engineering*, 46(1):131–150, 1999.

[17] J. Dolbow and I. Harari. An efficient finite element method for embedded interface problems. *Int. J. for Num. Meth. in Eng.*, 78(229–252), 2009.

[18] 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.

[19] M. Dryja. A neumann-neumann algorithm for mortar discretization of elliptic problems with discontinuous coefficients. *Num. Math.*, 99(645–656), 2005.

[20] 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.

[21] R. E. Ewing, Z. Li, T. Lin, and Y. Lin. The immersed finite volume element methods for the elliptic interface problems. *Mathematics and Computers in Simulation*, 50(1):63–76, 1999.

[22] 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.

[23] 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.

[24] T.-P. Fries and T. Belytschko. The intrinsic xfem: a method for arbitrary discontinuities without additional unknowns. *International journal for numerical methods in engineering*, 68(13):1358–1385, 2006.

[25] 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.

[26] 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.

[27] S. Groí and A. Reusken. An extended pressure finite element space for two-phase incompressible flows with surface tension. *Journal of Computational Physics*, 224(40–58), 2007.

[28] A. Guittet, M. Lepilliez, S. Tanguy, and F. Gibou. Solving elliptic problems with discontinuities on irregular domains - the voronoi interface method. *Journal of Computational Physics*.

[29] G. Guyomarc'h, C.-O. Lee, and K. Jeon. A discontinuous galerkin method for elliptic interface problems with application to electroporation. *Communications in numerical methods in engineering*, 25(10):991–1008, 2009.

[30] W. Hackbusch. *Multi-grid methods and applications*. Springer, 1985.

[31] A. Hansbo and P. Hansbo. A finite element method for the simulation of strong and weak discontinuities in solid mechanics. *Comput. Meth. in Appl. Mech. and Eng.*, 1993(3523–3540), 2004.

[32] 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.

[33] J. Huang and J. Zou. A mortar element method for elliptic problems with discontinuous coefficients. *IMA J. Numer. Anal.*, 22(549–576), 2001.

[34] J. J. E. Dendy. Black Box Multigrid. *Journal of Computational Physics*, 48:366–386, 1982.

[35] H. Ji and J. Dolbow. On strategies for enforcing interfacial constraints and evaluating jump conditions with the extended finite element method. *International Journal for Numerical Methods in Engineering*, 61(14):2508–2535, 2004.

[36] 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.

[37] A. J. Lew and G. C. Buscaglia. A discontinuous-galerkin-based immersed boundary method. *International Journal for Numerical Methods in Engineering*, 76(4):427–454, 2008.

[38] Z. Li. A fast iterative algorithm for elliptic interface problems. *SIAM Journal of Numerical Analysis*, 35:230–254, 1998.

[39] Z. Li and K. Ito. Maximum principle preserving schemes for interface problems with discontinuous coefficients. *SIAM Journal of Scientific Computing*, 23:339–361, 2001.

[40] Z. Li and K. Ito. *The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains*. SIAM, 2006.

[41] 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.

[42] B. Lu, Y. Zhou, M. Holst, and J. McCammon. Recent progress in numerical methods for the poisson-boltzmann equation in biophysical applications. *Commun Comput Phys*, 3(5):973–1009, 2008.

[43] A. Mayo. The fast solution of Poisson's and the biharmonic equations on irregular regions. *SIAM J. Numer. Anal.*, 21:285–299, 1984.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag New York, Applied Mathematical Sciences, 2002.

[50] 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.

[51] C. S. Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25:220–252, 1977.

[52] A. Raeli, M. Bergmann, and A. Iollo. A finite-difference method for the variable coefficient poisson equation on hierarchical cartesian meshes. *Journal of Computational Physics*, 355:59–77, 2018.

[53] J. W. Ruge and Stüben. *Multigrid methods*, chapter Algebraic multigrid, pages 73–130. SIAM, Philadelphia, 1987.

[54] G. Russo and P. Smereka. A remark on computing distance functions. *Journal of Computational Physics*, 163:51–67, 2000.

[55] 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.

[56] M. Semplice, A. Coco, and G. Russo. Adaptive mesh refinement for hyperbolic systems based on third-order compact weno reconstruction. *Journal of Scientific Computing*, 66(2):692–724, 2016.

[57] 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.

[58] 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.

[59] M. Sussman, K. M. Smith, M. Y. Hussaini, M. Ohta, and R. Zhi-Wei. A sharp interface method for incompressible two-phase flows. *Journal of computational physics*, 221(2):469–505, 2007.

[60] M. Theillard, C. H. Rycroft, and F. Gibou. A multigrid method on non-graded adaptive octree and quadtree cartesian grids. *Journal of Scientific Computing*, 55(1):1–15, 2013.

[61] U.Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2001.

[62] 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.

[63] W. L. Wan. Interface preserving coarsening multigrid for elliptic problems with highly discontinuous coefficients. *Numer. Linear Algebra Appl.*, 7:727–741, 2000.

[64] Y. Yang and H. Udaykumar. Sharp interface cartesian grid method iii: Solidification of pure materials and binary solutions. *Journal of Computational Physics*, 210(1):55–74, 2005.

[65] 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.