A parallel second order Cartesian method for elliptic interface problems

Marco Cisternino, Lisl Weynans

▶ **To cite this version:**

Marco Cisternino, Lisl Weynans. A parallel second order Cartesian method for elliptic interface problems. [Research Report] RR-7573, 2011, pp.27. inria-00577874v1

HAL Id: inria-00577874

https://inria.hal.science/inria-00577874v1

Submitted on 17 Mar 2011 (v1), last revised 13 Apr 2021 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A parallel second order Cartesian method for elliptic interface problems

Marco Cisternino — Lisl Weynans

N° 7573

Mars 2011

Computational models and simulation

ISSN 0249-6399 ISRN INRIA/RR--7573--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
BORDEAUX - SUD OUEST

A parallel second order Cartesian method for elliptic interface problems

Marco Cisternino*, Lisl Weynans†

Theme : Computational models and simulation
Applied Mathematics, Computation and Simulation
Équipe-Projet MC2

Rapport de recherche n° 7573 — Mars 2011 — ?? pages

Abstract: We present in this report a parallel Cartesian method to solve elliptic problems with complex immersed interfaces. This method is based on a finite difference scheme and second order accurate in the whole domain. We use a standard five point scheme for the discretization of the elliptic operator on all grid points, coupled with the discretization of transmission conditions across the interface. The originality of the method lies in the use of additional unknowns located on the interface, where the transmission conditions are imposed. We firstly describe the method itself and the details of its parallelization performed with the PETSc library. Then we present numerical validations in two dimensions, assorted with comparisons to other related methods, and a numerical study of the parallelized method.

Key-words: elliptic interface problem, Cartesian method, second order scheme.

* Dipartimento di Ingegneria Aeronautica e Spaziale, Politecnico di Torino
† INRIA Bordeaux Sud-Ouest, Equipe MC2

Centre de recherche INRIA Bordeaux – Sud Ouest
Domaine Universitaire - 351, cours de la Libération 33405 Talence Cedex
Téléphone : +33 5 40 00 69 00

A parallel second order Cartesian method for elliptic interface problems

Résumé : Nous présentons dans ce rapport une méthode cartésienne parallèle pour résoudre des problèmes elliptiques avec des interfaces complexes. Cette méthode est basée sur un schéma aux différences finies et est d'ordre deux dans tout le domaine. Nous utilisons un schéma standart à cinq points pour la discrétization de l'opérateur elliptique sur tous les points de grille, couplé à une discrétisation des conditions de transmission à travers l'interface. L'originalité de la méthode consiste en l'utilisation d'inconnues additionnelles situées sur l'interface, là où sont imposées les conditions de transmission. Nous décrivons d'abord la méthode elle-même et les détails de sa parallélisation réalisée avec la librairie PETSc. Puis nous présentons des validations numériques en deux dimensions, accompagnées de comparaisons avec d'autres méthodes du même type, et une étude numérique de sa version parallèle.

Mots-clés : problème elliptique avec interface, méthode cartésienne, schéma d'ordre deux.

1 Introduction

In this paper we aim to solve on Cartesian grids with an order two accuracy the following problem :

$$(P)\begin{cases} \nabla.(k\nabla u) = f \text{ on } \Omega = \Omega_1 \cup \Omega_2 \\ [\![u]\!] = \alpha \text{ on } \Sigma \\ [\![k\dfrac{\partial u}{\partial n}]\!] = \beta \text{ on } \Sigma \end{cases}$$

assorted with boundary conditions on $\delta\Omega$ defined as the boundary of Ω. As illustrated on Figure **??**, Ω is splitted into two subdomains Ω_1 and Ω_2, separated by a complex interface Σ. This elliptic problem with discontinuities across an interface appears in numerous physical or biological models. Among the well-known applications are heat transfer, electrostatics, fluid dynamics, but similar elliptic problems arise for instance in tumour growth modelling, where one has to solve a pressure equation [**?**], or in the modelling of electric potential in biological cells [**?**]. In this latter case the jump of the solution across the

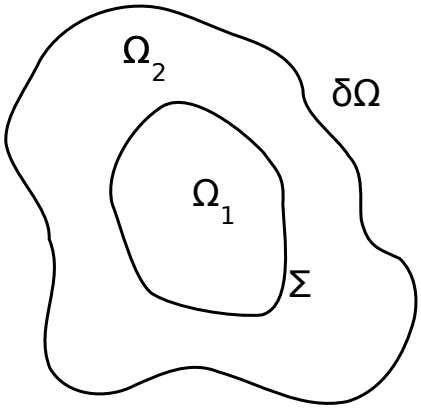

Figure 1: Geometry considered: two subdomains Ω_1 and Ω_2 separated by a complex interface Σ.

interface is proportional to the interior normal derivative.

To solve an elliptic interface problem in the case of a complex interface, an alternative approach to body-fitted methods (see for instance [**?**], [**?**] and [**?**]) is to discretize and solve the problem on a Cartesian grid. In this case, one takes into account the influence of the complex interface through modifications of the numerical scheme near the interface, without remeshing if the interface moves.

The first Cartesian grid method for elliptic problems was designed by Mayo in 1984 [**?**], and developed further in [**?**] and [**?**]. In that work an integral equation was derived to solve elliptic interface problems with piecewise coefficients to second order accuracy in maximum norm. Then came the very well known Immersed Interface Method (IIM) of LeVeque and Li (1994) [**?**]. This method

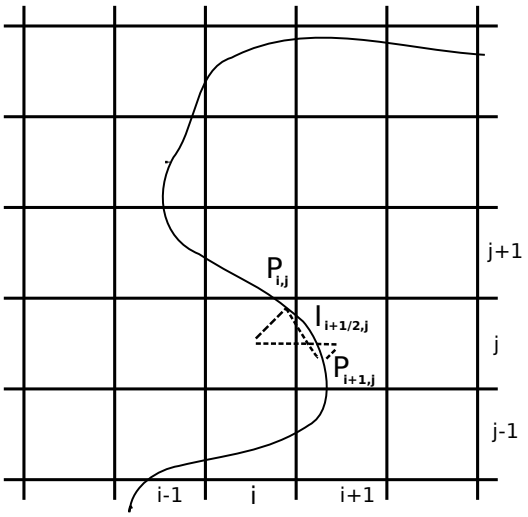

Figure 2: Local reconstruction of the interface between i and $i+1$ through the interface points $P_{i,j}$ and $P_{i+1,j}$ and computation of the intersection point $I_{i+1/2,j}$

relies on Taylor expansions of the solution on each side of the interface, with a local coordinate transformation near the interface to express the jump conditions in an appropriate frame. The elliptic operator is discretized on each grid point near the interface with formulas accounting for the jumps across the interface. In order to find these formulas a linear system with six unknowns needs to be solved for each of the concerned grid points. The method is also second order accurate in maximum norm. Numerous developments of the IIM have been performed. In the following lines we briefly evoke the most relevant. In [**?**] Li introduced a fast IIM algorithm for elliptic problems with piecewise constant coefficients. This version of IIM used auxiliary unknowns expressing the normal derivative at the interface. The fast IIM algorithm was generalized by Wiegmann and Bube in [**?**] under the name of Explicit Jump Immersed Interface Method (EJIIM). The EJIIM considers a classical finite difference discretization and uses corrective terms added to the right hand side of the linear system to take the interface into account. The corrective terms involve jumps of the solution and high order derivatives of it across the interface. Then, Li and Ito [**?**] proposed to solve a quadratic optimization problem for each point near the interface in order to choose finite difference coefficients on a nine point stencil leading to a maximum principle preserving scheme (MIIM). Bethelsen devised the Decomposed Immersed Interface Method (DIIM) [**?**]. He used an iterative procedure to compute successive right hand side correction terms accounting for the jump conditions at the interface, associated to a nine point interpolation stencil on each side of the interface.

Another class of Cartesian method recently introduced by Zhou et al. is the Matched Interface and Boundary (MIB) method: [**?**], [**?**], [**?**]. This method can provide finite-difference schemes of arbitrary high order. The solution on each side of the interface is extended on fictitious points on the other side. The

fictitious values are computed by iteratively enforcing the lowest order interface jump conditions. Finally, Chern and Shu [**?**] proposed a Coupling Interface Method, where the discretizations on each subdomain are coupled through a dimension by dimension approach using the jump conditions. All the methods cited above are second order accurate.

Other classes of Cartesian methods also exist, less accurate in the case of interface problems, but probably simpler to implement: Gibou et al. ([**?**], [**?**]) developed methods inspired by Fedkiw's Ghost-Fluid Method ([**?**], [**?**]) for multiphase flows. These methods are second order accurate for Dirichlet boundary conditions on arbitrary domains, but only first order accurate for interface problems. In the same spirit is the AIIB method of Sarthou et al (submitted), second order accurate for Dirichlet boundary conditions and between order one and order two for interface problems. The penalty method, introduced by Arquis and Caltagirone [**?**] and Angot et al [**?**], consists in approximating fluid and solid by porous media with porosity tending respectively to infinity or zero. It can also be used to solve elliptic problems on arbitrary domains and is order one accurate. An improvement has been recently proposed by Chantalat et al [**?**], to obtain second order accuracy for Dirichlet boundary conditions with an iterative scheme. But for the while it does not deal with interface problems

All the methods cited before can be considered as finite differences methods, and this is the context in which we aim to present our method. However, Cartesian method for elliptic interface problems also exist in finite-volume and finite-element communities: Collela and his group have notably devised methods in a finite volume spirit, where an interface reconstruction is applied to the cells near the interface (sometimes thus referenced as "cut cells"), in order to preserve conservativity properties ([**?**] and [**?**]). Among the finite-element community, let us give some references: [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**] and [**?**].

To our knowledge, none of the second order Cartesian methods cited above has been implemented in a parallel code. One advantage of using Cartesian grids is to allow an easy parallelization, at least provided that the specific treatment of interfaces does not increase to much the complexity of the method. In this paper we propose a parallel second order Cartesian method for elliptic interface problems. The method is based on a finite differences discretization and a dimension by dimension approach.

In order to solve accurately the elliptic problem near the interface, we introduce additional unknowns located at the intersections of the interface with the grid. These interface unknowns are used in the discretization of the elliptic operator near the interface, and avoid us to derive specific finite differences formulas containing jump terms, corrective terms, or needing the inversion of a linear system, as in many other second order Cartesian methods. In order to solve the interface unknowns we discretize and solve the flux jump conditions. Jump conditions and the coupling of the solution in the different subdomains are thus handled independently of the discretization of the elliptic equation, as it is the case in the MIB method, and the methods of Gibou and Sarthou too, through the use of additional unknowns. But contrary to them, these additional unknowns are located at the interface and not at grid points of the other side of the interface. The simplicity of the method guarantees its easy parallelization.

In section **??** we firstly discuss the influence of the truncation error on the convergence order. Then we present the method in section **??** and its parallelization performed with the PETSC library in section **??**. Finally we present

numerical results validating the order of convergence of the method and its scalability in section **??**.

2 Convergence rate dependence on truncation error near the interface

In this section, we perform an analysis of the convergence error in terms of the truncation error for the Laplacian equation in 1D. This will allow us to explain the choice of a discretization near the interface in order to get an order two accuracy.

First of all, let us remark that ghost-cell like formulas at boundaries, that is, finite difference formulas where one point has a "fictitious value" imposed in order to account for the boundary conditions, are not necessarily consistent for the discretization of Laplace equation: For instance, let us assume that the boundary lies between grid points i and $i+1$. In order to compute an approximation of the Laplacian at point i, a ghost-cell like method needs a fictitious value at point $i+1$ that takes into account the Dirichlet boundary condition $u(x_{int}) = u_{int}$. If we use a linear extrapolation the ghost value at point $i+1$will be:

$$\tilde{u}_{i+1} \quad = \quad u_i + dx \frac{u_{int} - u_i}{x_{int} - x_i}$$

The truncation error of the discretization of the Laplacian at point i is therefore:

$$\frac{\tilde{u}_{i+1} - 2u_i + u_{i-1}}{dx^2} \quad = \quad \frac{dx \frac{(x_{int} - x_i)}{2} u''(x_i) + \frac{dx^2}{2} u''(x_i) + O(dx^3)}{dx^2} = O(1)$$

Thus the scheme is not formally consistent in the finite differences sense at point i.

However, theoretical studies performed by Gustafsson ([**?**] and [**?**]) and more recently by Svard and Nordstrom [**?**] show that under certain assumptions it is possible to have at the boundary a discretization less accurate than in the rest of the domain without deteriorating the order of convergence. Furthermore, the numerical methods developed in the ghost-cell spirit ([**?**], [**?**]) are actually at least second order accurate for Dirichlet boundary conditions. But what works for boundary conditions does not necessarily work for an interface immersed inside the domain. Let us notice for instance that in [**?**] and [**?**], only first order convergence is reached for some interface problems. For this reason, in the following, we study in a simple case the influence of the truncation error on the convergence rate of the numerical solution.

We consider the one-dimensional Laplace equation on the segment $[0, 1]$, with Dirichlet boundary conditions. We assume that there is an interface Σ located at $x = x_{int}$ inside the domain, where the following jump conditions are satisfied.

$$\begin{cases} [\![u]\!] = 0 & \text{on } \Sigma \\ [\![k \frac{\partial u}{\partial n}]\!] = 0 & \text{on } \Sigma \end{cases}$$

We assume that $k = k_1$ in $[0, x_{int}]$ and $k = k_2$ in $[x_{int}, 1]$ Grid points are defined on locations $x_k = kdx, 0 \leq k \leq N+1$ with $dx = \frac{1}{N+1}$ and x_{int} belongs to the

segment $]x_i, x_{i+1}[$. The interface is located inside the domain so we can write $i \sim aN$, where a is a real between 0 and 1, independent of dx.

For the grid points inside the domain and far enough from the interface, the finite difference approximation is the second order centered three points formula:

$$\frac{u_{k+1} - 2u_k + u_{k-1}}{dx^2} = f_k. \tag{1}$$

The local error $e_k = u(x_k) - u_k$ satisfies the same kind of linear relationship as u_k, but with the local truncation error ϵ_k as a source term:

$$\frac{e_{k+1} - 2e_k + e_{k-1}}{dx^2} = \epsilon_k.$$

We aim to solve explicitly the linear system satisfied by the truncation error. By two recurrences, one forward and one backward, we can show that:

$$e_{k+n} = (n+1)e_k - ne_{k-1} + dx^2 \sum_{j=1}^{n} j\epsilon_{k+n-j} \tag{2}$$

$$e_{k-n} = (n+1)e_k - ne_{k+1} + dx^2 \sum_{j=1}^{n} j\epsilon_{k-n+j} \tag{3}$$

Now we assume that a Ghost-Cell technique based on a linear extrapolation is used to discretize the Laplacian near the interface. Let denote u_{int} the value of u at the interface. Two fictitious values are created at points i and $i+1$ respectively:

$$\begin{aligned} \tilde{u}_{i+1} &= u_i + dx\frac{u_{int} - u_i}{x_{int} - x_i} \\ \tilde{u}_i &= u_{i+1} - dx\frac{u_{i+1} - u_{int}}{x_{i+1} - x_{int}} \end{aligned}$$

We obtain the following finite difference formulas for the discretization of the Laplace equation at points i and $i+1$:

$$\frac{u_{int} - u_i}{x_{int} - x_i} - \frac{u_i - u_{i-1}}{dx} = dx f_i \tag{4}$$

$$\frac{u_{i+2} - u_{i+1}}{dx} - \frac{u_{i+1} - u_{int}}{x_{i+1} - x_{int}} - = dx f_{i+1} \tag{5}$$

Then the truncation errors ϵ_i and ϵ_{i+1} on points i and $i+1$ satisfy:

$$\frac{e_{int} - e_i}{x_{int} - x_i} - \frac{e_j - e_{i-1}}{dx} = dx\ \epsilon_i \tag{6}$$

$$\frac{e_{i+2} - e_{i+1}}{dx} - \frac{e_{i+1} - e_{int}}{x_{i+1} - x_{int}} = dx\ \epsilon_{i+1} \tag{7}$$

with $e_{int} = u(x_{int}) - u_{int}$. The flux equation at interface is expressed by:

$$k_{i+1}\frac{u_{i+1} - u_{int}}{x_{i+1} - x_{int}} - = k_i\frac{u_{int} - u_i}{x_{int} - x_i} \tag{8}$$

and thus the truncation error ϵ_{int} related to this equation satisfies:

$$k_{i+1}\frac{e_{i+1}-e_{int}}{x_{i+1}-x_{int}} - k_i\frac{e_{int}-e_i}{x_{int}-x_i} = \epsilon_{int} \tag{9}$$

The truncation error of the Laplacian discretizations (**??**) and (**??**) are zero order accurate, as seen before. The truncation error of the flux equation (**??**) is order one accurate, as a Taylor expansion shows it. In order to close the linear system, we assume that at the boundary of the computational domain Dirichlet boundary conditions are imposed exactly:

$$e_0 = 0 \text{ and } e_{N+1} = 0.$$

Other types of boundary conditions are possible and would lead to linear relationships between e_0 or e_{N+1} and the other e_i. At the end the conclusion would be the same.

Using these boundary conditions, we deduce from (**??**) and (**??**) that:

$$\begin{aligned} 0 &= (N+1-k)e_{k-1} - (N+2-k)e_k - dx^2 \sum_{j=1}^{N+1-k} j\epsilon_{N+1-j} \text{ for } k \geq i+2 \\ 0 &= (k+1)e_k - ke_{k+1} + dx^2 \sum_{j=1}^{k} j\epsilon_j \text{ for } k \leq i-1 \end{aligned}$$

Therefore:

$$\begin{aligned} e_{k+1} &= \frac{(N-k)e_k}{(N+1-k)} - dx^2 \sum_{j=1}^{N-k} \frac{j}{(N+1-k)}\epsilon_{N+1-j} \text{ for } k \geq i+2 \\ e_{k+1} &= \frac{(k+1)}{k}e_k + dx^2 \sum_{j=1}^{k} \frac{j}{k}\epsilon_j \text{ for } k \leq i-1 \end{aligned}$$

Using the latter equations with (**??**), (**??**) and (**??**) we show that:

$$e_{int}(C_i + C_{i+1}) = -\epsilon_{int} - dx^2\, C_i \sum_{j=1}^{N-i} j\ \epsilon_{N+1-j} - dx^2\, C_{i+1} \sum_{j=1}^{i} j\ \epsilon_j$$

with:

$$\begin{aligned} C_i &= \frac{k_1}{x_{int} - x_i + idx} \\ C_{i+1} &= \frac{k_2}{x_{i+1} - x_{int} + (N-i)dx} \end{aligned}$$

Because $i \sim aN$ then $C_i = O(1)$ and $C_{i+1} = O(1)$. We therefore conclude that if the truncation error of the flux equation discretization ϵ_{int} is order one then the error on the interface e_{int} is a priori order one too. Moreover, if the truncation error of the Laplacian discretization is order 2 for all points excepted for the two points neighbouring the interface: $\epsilon_i = O(1)$ and $\epsilon_{i+1} = O(1)$, then the error on the interface point is a priori order one too.

We deduce from the above considerations that, to obtain an order 2 accuracy at the interface, we need to:

- use a discretization of Laplacian near the interface with a truncation error of order 1, thus avoid linear Ghost Cell techniques. One can for instance either use a quadratic extrapolation to determine the ghost cell value, or, as we do in the following, use another discretization of the elliptic operator.
- use a discretization of the flux transmission equations at the interface with a truncation error of order 2.

This analysis is of course not a proof that if the truncation error of the Laplacian is zero, then the numerical error will never be second order accurate, because there can be compensation effects in the sums involved in the expression of the error. It is well known in the finite volume community that a scheme that is not consistent in the finite difference sense can in fact converge. However numerical tests that we performed corroborate this reasoning. Other existing second order Cartesian methods often explicitly take care to have a second order truncation error for the first order derivatives, and a first order truncation error for the second order derivatives near the interface.

3 Description of the method

In this section we firstly describe the method in the case where $\alpha = \beta = 0$. The case $\alpha \neq 0$, $\beta \neq 0$ will be treated in the last part of the section.

3.1 Interface description and classification of grid points

In order to improve accuracy where the interface crosses grid cells we need additional geometric information. This information, mainly the distance from the interface and the interface normal, is provided by the distance function. The level set method, introduced by Osher and Sethian [**?**], is used to implicitely represent the interface in the computational domain. We refer the interested reader to [**?**], [**?**] and [**?**] for recent reviews of this method. The zero isoline of the level set function represents the interface Σ immersed in the computational domain. The level set function is defined by:

$$\varphi(x) = \begin{cases} dist_\Sigma(x) & \text{outside of the interface} \\ -dist_\Sigma(x) & \text{inside of the interface} \end{cases}$$

The interface is implicitly represented by the zero level set of this function. A useful property of the level set function is:

$$\mathbf{n}(x) = \nabla\varphi(x)$$

where $\mathbf{n}(x)$ is the outward normal vector of the isoline of ϕ passing on x. In particular, this allows to compute the values of the normal to the interface, represented by the isoline $\phi = 0$.

The points on the Cartesian grid are defined by $M_{i,j} = (x_i, y_j) = (i\,dx, j\,dy)$. We denote by $u_{i,j}$ the approximation of u at the point (x_i, y_j). We say that a grid point is neighbouring the interface if ϕ changes sign between this point and at least one of its neighbours. In the contrary, regular grid points designate grid points that are not neighbours of the interface.

3.2 Discrete elliptic operator for regular grid points

On regular grid points we use the standard centered second order finite differences scheme:

$$\begin{aligned}\nabla(k\nabla u)(x_i,y_j) &\approx \frac{k_{i+1/2,j}(u_{i+1,j}-u_{i,j})-k_{i-1/2,j}(u_{i,j}-u_{i-1,j})}{dx^2} \\ &+\frac{k_{i,j+1/2}(u_{i,j+1}-u_{i,j})-k_{i,j-1/2}(u_{i,j}-u_{i,j-1})}{dy^2} \quad (10)\end{aligned}$$

where $k_{i+1/2,j}$ is a second order approximation of k at point $\frac{M_{i,j}+M_{i+1,j}}{2}$.

On grid points neighbouring the interface, this approximation is not accurate enough, because of the discontinuity of the coefficient k. In order to cope with this problem, we create new unknowns, located at what we call "intersection points": the intersections of the interface with the segment joining two grid points having opposite signs of ϕ. We will use these new unknowns to discretize more accurately the term $\nabla(k\nabla u)(x_i,y_j)$. But firstly we need to compute the coordinates of these intersection points.

3.3 Computation of intersection points

We define an intersection point each time that ϕ changes its sign between two adjacent grid points. In [**?**] Sarthou et al. proposed to compute the barycentre of two grid points weighted with their respective distance to the interface. This formula is exact in one dimension, but in two dimensions it assumes implicitly that the interface point closest to these grid points is located on the segment joining them, which is not always the case. We propose here another way to compute the intersection points:

Let $M_{i,j}$ and $M_{i+1,j}$ two adjacent grid points such that $\phi_{i,j}\,\phi_{i+1,j}<0$. We define $P_{i,j}$ and $P_{i+1,j}$ as the points belonging to the interface nearest to $M_{i,j}$ and $M_{i+1,j}$ respectively:

$$P_{i,j}=M_{i,j}+\phi_{i,j}n_{i,j} \text{ and } P_{i+1,j}=M_{i+1,j}+\phi_{i+1,j}n_{i+1,j}$$

We compute $n_{i,j}$, an approximation of the normal to the interface at point $P_{i,j}$, for example with an second order discretization of the gradient of ϕ. Then we define $I_{i+1/2,j}=(\tilde{x}_{i+1/2,j},y_j)$, the intersection point, as the intersection if it exists of $[P_{i,j}P_{i+1,j}]$ and $[M_{i,j}M_{i+1,j}]$. If this intersection does not exist, we define instead $I_{i+1/2,j}$ as the middle of $[M_{i,j}M_{i+1,j}]$. This construction of the interface points is illustrated on Figure **??**.

3.4 Discrete elliptic operator near the interface

For a grid point $M_{i,j}$ neighbouring the interface, we discretize the term

$$\nabla\cdot(k\nabla u)(x_i,y_j)$$

with the values on $M_{i,j}$ and on the four points (grid points or intersection points) that are the closest to $M_{i,j}$ in each direction. For instance, in the case illustrated

on Figure **??**, the linear equation to solve at point $M_{i,j}$ is:

$$\begin{aligned}&\frac{\tilde{k}_{i+1/2,j}\frac{\tilde{u}_{i+1/2,j}-u_{i,j}}{\tilde{x}_{i+1/2,j}-x_i} - k_{i-1/2,j}\frac{u_{i,j}-u_{i-1,j}}{dx}}{\frac{(\tilde{x}_{i+1/2,j}-x_i)}{2}+\frac{dx}{2}}+\\ &+\frac{k_{i,j+1/2}(u_{i,j+1}-u_{i,j})-k_{i,j-1/2}(u_{i,j}-u_{i,j-1})}{dy^2} = f(x_i,y_j)\end{aligned} \tag{11}$$

where $\tilde{u}_{i+1/2,j}$ denotes the value of u at point $I_{i+1/2,j}$, and $\tilde{k}_{i+1/2,j}$ is an approximation of the value of k at the middle between $I_{i+1/2,j}$ and $M_{i,j}$. This discretization is second order accurate if the point $\tilde{x}_{i-1/2,j}$ coincide with x_{i+1}, and first order accurate otherwise.

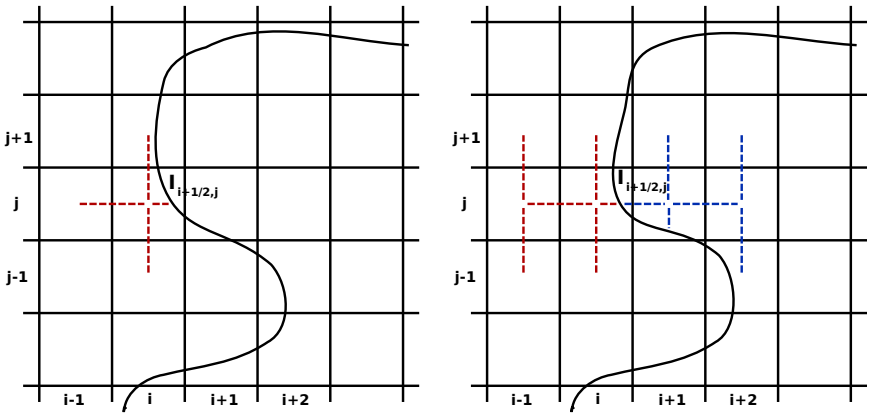

Figure 3: Example of stencils for the discretization of the elliptic operator at point (i,j) (left) and for the discretization of the flux equality at point $I_{i+1/2,j}$ (right).

3.5 Discrete flux transmission conditions

As we have seen in the last section, we want the truncation error of the flux equality discretization to be second order accurate. On Figure **??** a possible configuration of the interface is illustrated. In the x-direction, it is straightforward to compute a second order approximation of the x-derivative by using a finite difference formula with three non equidistant points. For example we approximate the flux on the left side of the interface with the points $M_{i-1,j}$, $M_{i,j}$ and $I_{i+1/2,j}$ by:

$$\begin{aligned}\frac{\partial u^l}{\partial x}(\tilde{x}_{i+1/2,j},y_j) \approx &\frac{(u_{i-1,j}-\tilde{u}_{i+1/2,j})(x_i-\tilde{x}_{i+1/2,j})}{dx(x_{i-1}-\tilde{x}_{i+1/2,j})}+\\ &-\frac{(u_{i-1,j}-\tilde{u}_{i+1/2,j})(x_{i-1}-\tilde{x}_{i+1/2,j})}{dx(x_i-\tilde{x}_{i+1/2,j})}\end{aligned} \tag{12}$$

The right x-derivative $\frac{\partial u^r}{\partial x}(\tilde{x}_{i+1/2,j},y_j)$ is approximated in the same way.

For the derivative along the y-direction, we do not have unknowns located on the line passing by $I_{i+1/2,j}$ and parallel to the y-axis. Therefore we use an extended stencil including 6 points. We compute $(\frac{\partial u}{\partial y})_{i,j}$ and $(\frac{\partial u}{\partial y})_{i-1,j}$, defined respectively as second order approximations of the y-derivative on points $M_{i,j}$, $M_{i-1,j}$, and we use a linear combination of them to approximate the derivative on $I_{i+1/2,j}$ with second order accuracy.

$$\frac{\partial u^l}{\partial y}(\tilde{x}_{i+1/2,j}, y_j) \approx \frac{\tilde{x}_{i+1/2,j} - x_{i-1}}{dx} (\frac{\partial u}{\partial y})_{i,j} - \frac{\tilde{x}_{i+1/2,j} - x_i}{dx} (\frac{\partial u}{\partial y})_{i-1,j} \quad (13)$$

The formulas for $(\frac{\partial u}{\partial y})_{i,j}$ and $(\frac{\partial u}{\partial y})_{i-1,j}$ depend on the local configuration on the interface, but are based on the same principle as for (**??**).

The above formulas are consistent provided that the point $M_{i-1,j}$ belongs to the same domain as $M_{i,j}$. We thus need to make the assumption that there are at least two adjacent points in each direction belonging to the same domain. Otherwise, the second order discretization is not possible with this method. We believe this condition is not very restrictive.

Eventually the flux equality corresponding to the case described in Figure **??** is discretized by:

$$\begin{aligned} &[k^l(\frac{\partial u^l}{\partial x})_{i+1/2,j} - k^r(\frac{\partial u^r}{\partial x})_{i+1/2,j}]\ n_x + \\ &+ [k^l(\frac{\partial u^l}{\partial y})_{i+1/2,j} - k^r(\frac{\partial u^r}{\partial y})_{i+1/2,j}]\ n_y \quad = 0 \end{aligned} \quad (14)$$

with (n_x, n_y) an approximation of the vector normal to the interface at point $I_{i+1/2,j}$, k^l and k^r respectively the left and right limit values of k on interface point $I_{i+1/2,j}$.

The stencil used in our discretization of the flux equality across the interface contains then 13 points. Actually, it is possible to maintain an order two accuracy with a stencil containing only 8 points. Let assume for instance that there is a change of sign of the level-set function between $M_{i,j}$ and $M_{i+1,j}$. As other authors noticed it ([**?**], [**?**], [**?**] and [**?**]) the jump condition on u can be differentiated in the direction tangential to the interface, giving the relationship:

$$[\![\frac{\partial u}{\partial \tau}]\!] = \frac{\partial \alpha}{\partial \tau}$$

The latter equation is in fact a linear relationship involving the partial derivatives on each side on the interface $\frac{\partial u^l}{\partial x}$, $\frac{\partial u^r}{\partial x}$, $\frac{\partial u^l}{\partial y}$ and $\frac{\partial u^r}{\partial y}$. With this equation $\frac{\partial u^r}{\partial y}$ for instance can be expressed as a linear combination of the others partial derivatives. This term thus does not need to be discretized, which removes 4 points from the stencil used for the discretization of the flux. Additionally, one does not actually need 6 points to discretize $\frac{\partial u^l}{\partial y}$ with second order accuracy, but only 5. Therefore, one can discretize the flux equality at the interface at second order accuracy with only 8 points instead of 13. However, when we performed numerical comparisons between the 13 points version and the 8 points version, it appeared to us that the amplitude of the error in the maximum norm was sensibly higher when using the 8 points stencil, while the time needed to solve the linear system was not noticeably higher. For this reason, we decided to present the 13 points discretization of the flux rather than the 8 points version.

In this paper we have only presented the method in the case of a two-dimensional geometry. But extending it to three-dimensional problems would not increase very much its complexity, because the method is based on a dimensional splitting. Therefore, adding one dimension only requires to add to the formulas for the flux equality or for the elliptic operator the discretization of the first or second order derivative in the z-direction, following the same principles than in the x- or y-directions. The discretization of the elliptic operator on a given grid point would need the grid point itself, and the closest point (grid point or intersection point) in each direction, leading to a 7 point stencil. The discretization of the flux equality, for instance at an intersection point between $M_{i,j}$ and $M_{i+1,j}$, would need to approximate the z-derivative similarly to the y-derivative. It would lead to a stencil with 3 to 8 more points than in 2D, depending on which kind of discretization is chosen (between the 8 and 13 points versions mentioned above).

3.6 Stabilization

In the first numerical tests that we performed, we noticed oscillations in the convergence plots, spoiling the expected order of convergence. These instabilities appeared exactly when the discretization of a flux transmission equation at an interface point involved another intersection point located very near from one grid point. The nearer the interface point was to the grid point, the stronger the oscillations were. This situation is illustrated on Fig **??**. On this figure one can understand why such a configuration leads to numerical instabilities: the centered finite differences formula is ill-conditioned due to the small distance between the grid point and the intersection point. However, in the case of a flux equality involving one only intersection point, the fact that this intersection point may be very near from a grid point did not seem to deteriorate in a sensible way the numericals results in our experiments.

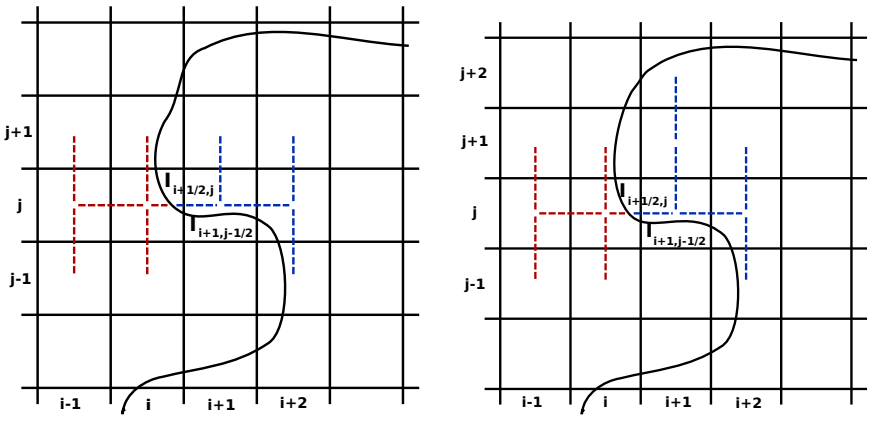

Figure 4: On left: Centered stencil: the discretization of the flux on point $I_{i+1/2,j}$ involves intersection point $I_{i+1,j-1/2}$ and is not numerically well-conditioned. On right: Decentered stencil: the discretization of the flux on point $I_{i+1/2,j}$ involves now grid point $(i+1, j+2)$ instead of intersection point $I_{i+1,j-1/2}$

The solution that we choose to avoid these unwanted oscillations without lowering the order of the discretization is to use in these cases a decentered discretization of the fluxes at the interface. We use the same number of points, but we change the stencil, as illustrated in Figure **??**. Instead of using in the flux discretization the intersection point causing the instability, we use the nearest grid point located in the opposite direction. For instance, in Eq. (**??**):

$$\begin{aligned}&[k_{i,j}(\frac{\partial u^l}{\partial x})_{i+1/2,j} - k_{i+1,j}(\frac{\partial u^r}{\partial x})_{i+1/2,j}]\ n_x + \\ &+ [k_{i,j}(\frac{\partial u^l}{\partial y})_{i+1/2,j} - k_{i+1,j}(\frac{\partial u^r}{\partial y})_{i+1/2,j}]\ n_y \quad = 0\end{aligned}$$

the term $(\frac{\partial u^r}{\partial y})_{i+1/2,j}$ is computed with a finite differences formula involving the grid points $(i+1,j)$, $(i+1,j+1)$ and $(i+1,j+2)$ instead of the grid points $(i+1,j)$, $(i+1,j+1)$ and the intersection point $I_{i+1,j-1/2}$.

At the beginning, we aimed to use this modified stencil only when the interface point was extremely near the grid point. But we realized that the modified stencil did not seem to deteriorate the accuracy of the results, and thus we decided to use the modified stencil in a systematic way everytime that there are two intersection points involved in the same flux equation.

3.7 Case $\alpha \neq 0$, $\beta \neq 0$

Now we present how to take into account non-zero jumps in the values of the function and/or its normal derivative across the interface.

The jump term α being non-zero means that there are in fact two unknowns at each intersection point, with each one of these unknowns that can be expressed as a sum of the other unknown and the jump term. We define as interface unknowns $\tilde{u}_{i+1/2,j}$ or $\tilde{u}_{i,j+1/2}$ the limit values of u in the domain defined by $\phi < 0$ at the points $I_{i+1/2,j}$ or $I_{i,j+1/2}$. The values at the same intersection points but for the domain where $\phi > 0$ are defined by $\tilde{u}_{i+1/2,j} + \alpha(I_{i+1/2,j})$ or $\tilde{u}_{i,j+1/2} + \alpha(I_{i,j+1/2})$. The discretization of the linear system remains the same, only the right hand side term changes. For the lines of the linear system involving the interface unknowns of the subdomain $\phi > 0$, the right hand side term receives the value $-\alpha(I_{i+1/2,j})$ or $-\alpha(I_{i,j+1/2})$ multiplied by the same coefficient than for the interface unknowns $\tilde{u}_{i+1/2,j}$ or $\tilde{u}_{i,j+1/2}$ themselves. For instance, if $\varphi(M_{i,j}) < 0$ then Eq. (**??**) remains unchanged, but if $\varphi(M_{i,j}) > 0$ it becomes:

$$\begin{aligned}&\frac{\tilde{k}_{i+1/2,j}\frac{\tilde{u}_{i+1/2,j}-u_{i,j}}{\tilde{x}_{i+1/2,j}-x_i} - k_{i-1/2,j}\frac{u_{i,j}-u_{i-1,j}}{dx}}{\frac{(\tilde{x}_{i+1/2,j}-x_i)}{2}+\frac{dx}{2}} + \\ &+\frac{k_{i,j+1/2}(u_{i,j+1}-u_{i,j}) - k_{i,j-1/2}(u_{i,j}-u_{i,j-1})}{dy^2} = \\ &= -\frac{\frac{\tilde{k}_{i+1/2,j}}{(\tilde{x}_{i+1/2,j}-x_i)}}{\frac{(\tilde{x}_{i+1/2,j}-x_i)}{2}+\frac{dx}{2}}\alpha(I_{i+1/2,j}) + f(x_i,y_j)\end{aligned}$$

If the jump term β is non-zero, it is simpler: the coefficient $\beta(I_{i+1/2,j})$ or $\beta(I_{i,j+1/2})$ appears in the right hand side of the equation of the linear system discretizing the flux equality on point $I_{i+1/2,j}$ or $I_{i,j+1/2}$. For instance, if

$\varphi(M_{i,j}) > 0$ Eq. (**??**) becomes:

$$\begin{aligned}&[k_{i,j}(\frac{\partial u^l}{\partial x})_{i+1/2,j} - k_{i+1,j}(\frac{\partial u^r}{\partial x})_{i+1/2,j}]\ n_x + \\ &+ [k_{i,j}(\frac{\partial u^l}{\partial y})_{i+1/2,j} - k_{i+1,j}(\frac{\partial u^r}{\partial y})_{i+1/2,j}]\ n_y \quad = \beta(I_{i+1/2,j})\end{aligned}$$

4 Parallelization of the method

Growing in interface topology complexity involves the need for a large number of points in order to catch the near interface details of the solution. Moreover, if the method is employed in the frame of time integration methods, which need the solution of an immersed interface elliptic problem at every time step, the efficiency and the performance of the solver become crucial. The parallel implementation of the method allows to deal with both the matters. However, the discretization scheme complexity may spoil these advantages.

4.1 Paradigm and Model

The parallelization of the method has been handled using the local memory parallel programming paradigm, Message Passing, and the SPMD philosophy (Single Program Multiple Data). According to the latter, every processor executes the same set of instructions and the execution flow varies as a function of the local ambient (data, enumeration of the processor, etcetera...). The API (Application Program Interface) chosen to implement the Message Passing is MPI (Message Passing Interface, [**?**]).

The implemented parallelization model, the parallelism of the data, is typical of problems coming from Partial Differential Equations, because it allows a good scalability compared to the functional one. The computational domain is decomposed and every single part is assigned to a processor which, then, executes the program on its sub-domain.

The management of the sub-domains boundaries is crucial for a good efficiency of the parallel code, in particular if the values of the solution in the points of the sub-domain near to the boundary depend on points belonging to the adjacent sub-domain. The most common approach to the boundary management is extending the sub-domain of a processor to points of the adjacent sub-domain. These points make up a new area of the single processor computational domain, called Ghost Region. The values of the solution in this region are not modified by the processor they belong to, as Ghost Points, but they are updated through communications with contiguous processors. An example for 4 processors is given in Figure **??** showing Ghost Regions and communication fluxes. The Cartesian topology has been chosen in analogy to the computational grid and it ensures, at least in the general cases, the smallest amount of communications to update the Ghost Regions.

4.2 PETSc library

Using the PETSc library (Portable, Extensible Toolkit for Scientific Computation, see [**?**], [**?**], [**?**] for details) we were able to avoid explicit communications.

PETSc is a suite of data structures and routines for parallel solution of scientific applications modelled by Partial Differential Equations. It employs the MPI standard for all the Message Passing communications; it is a free use software provided a copyright notice is retained, see [**?**]; it supplies the user with parallel data structures (vectors, matrices, index sets and more), efficient access and assembling operations for these structures, and in particular several iterative linear and non-linear solvers.

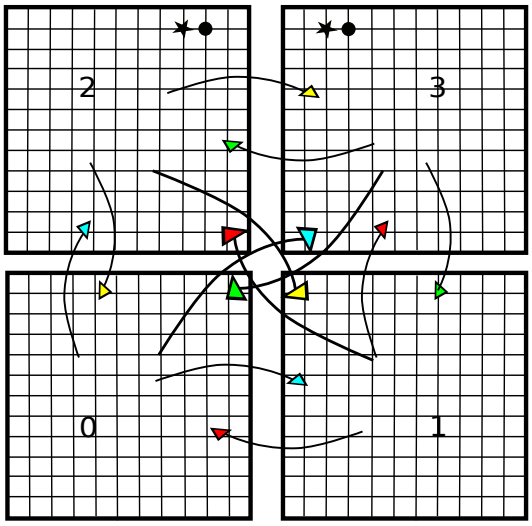

Figure 5: Example of domain decomposition for 4 processors. Light colours: Ghost Regions. Dark colours: Computational Domain. The arrows show the communications fluxes. Same symbols mean same points.

4.3 Parallel Implementation

In this section we document how the algorithmic simplicity of the method introduced in the previous sections allows a straightforward parallelization of the code implementing the method through the employ of the PETSc library.

The aim of the code is to solve the following linear system

$$Au = f \tag{15}$$

where A is the matrix discretizing the differential operator on the Cartesian grid points augmented with the conditions on the interface fluxes and f is the right hand side on the grid points augmented with the jump values of the fluxes on the interface points. The solution is stored in u, which contains not only the values on the grid points but also the values on the interface points. Therefore, the code assembles A and f, solves the linear system and extracts the solution on the grid points.

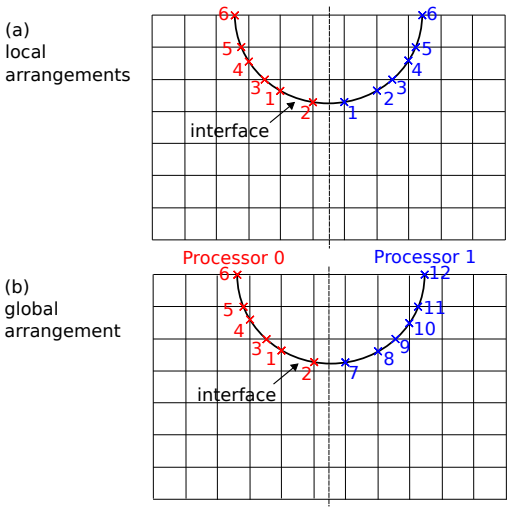

Figure 6: Example of intersections search (two processors case). (a): the local arrangements; the sequential vector stores the values $(6, 6)$. (b): the final global arrangement.

4.3.1 Aspects of the method simplifying the parallelization through PETSc library

The use of Cartesian grid, the choice of putting information about the intersections on the Cartesian grid points, the need for scattering only to extract the solution and having only one linear system to solve are all important advantages in this parallel implementation.

Despite the PETSc library capacity to manage any kind of computational grid, it provides powerful means to deal with structured grids. The most important structure to manage this kind of grids is the DA, the distributed array. It is a logical regular and rectangular structure and it is not intended to store data but to contains information about the layout of the parallel data and about the communications. The storage of vectors and matrices is accomplished by the joint use of DA, Vec and Mat in a straightforward and implicit way (no explicit calls to communication routines is needed). The details on Vec, Mat, and DA structures can be found in [**?**].

Moreover, even if we have to deal with the intersections of the interface with the grid axes, storing informations about them on the grid nodes makes the use of the DA possible to manage these points which are generally out of the nodes. No complex geometrical treatment of the intersections is required (as we will see in the next section), referencing them as properties of each grid point, stored in some Vec structures.

Although the augmented nature of the linear system seems to complicate the extraction of the solution on the grid nodes, efficient structures (such as IS (index set), VecScatter, see [**?**]) and routines relative to their use make the scattering between parallel vectors easy to be implemented with few code lines and fairly performing. Furthermore, we have to implement this operation just

for one time at the end of the code with almost no computational cost (compared to the one needed to solve the linear system).

The need for solving a single linear system (**??**) is eventually fully satisfied by the use of the PETSc core routines, included in KSP interface (see [**?**]). They allow not only the choice and the set up of iterative solvers and preconditioners, but even the possibility to do this at runtime, giving the code an important flexibility.

All these aspects makes parallel coding very similar to the sequential one, provided the right use of the PETSc structures and routines. The less trivial part of the code concerns the enumeration of the intersections points, but a very simple approach can efficiently and shortly deal with this matter.

4.3.2 Notes on intersections

This is the only section of the parallel code where we decide to use explicit calls to MPI standard in. The search for intersection is local, i.e. every processor looks for intersection in its own sub-domain and it fills parallel vectors defined on the regular grid in with information about existence and position of the intersections. This guarantees the best performance but it needs communications to update the ghost regions. However, without a change in the arrangement every processor would number the intersections starting from 1. This would make impossible to unambiguously recognize any single intersection.

In order to avoid any ambiguity, every processor starts the search in its own domain (**??**(a)); then, through a call to MPI_ALLGATHER(), a sequential vector of length equal to the number of processors and owned by all the processors is updated: every position in this vector is associated to a processor and it stores the number of intersections in every sub-domain. Therefore, the sum of the vector elements, which come before the position associated to a processor, is used to increase the intersections arrangement of that processor. At the end the ghost regions are updated by a call to the DALocalToLocal() routine [**?**] and a global numeration of the intersections is obtained (**??**(b)).

These important remarks and the PETSc library have made possible to write a parallel code absolutely similar to the sequential one and for some aspects even more flexible, without any compromise in term of performance.

5 Numerical validation

5.1 Sequential validation of the method in two dimensions

In this section we present the convergence results for several test cases in two dimensions. Some of them have already been studied by others authors, which allow us to compare our results to theirs, in terms of convergence order as well as error amplitude. The linear systems are solved with routines from the SPARSKIT library [**?**]: a GMRES algorithm with a ILU preconditionning.

In all the following test cases, we consider a square domain Ω splitted into two subdomains Ω_1 and Ω_2 separated by an interface Σ. If not specified otherwise, $\Omega = [-1, 1] \times [-1, 1]$. We impose exact Dirichlet boundary conditions on the outer boundary of Ω.

5.1.1 Problem 1

The homogeneous Laplace equation is considered with the analytical solution:

$$u(x,y) = \begin{cases} 100 \text{ if } r = \sqrt{x^2+y^2} < 1 \\ 100 + 50\ ln(1/r) \text{ if } 1 < r < 1.5 \\ 100 + 50\dfrac{k_1}{k_2} ln(1/r) + 50\ (1 - \dfrac{k_1}{k_2})\ ln(1/1.5) \text{ if } r > 1.5 \end{cases}$$

with k_1 and k_2 varying coefficients. Numerical results for $k_1 = 2$ or 1000 and $k_2 = 1$ for the discrete L^∞ norm are presented on Table **??** and Table **??**. We observe global second order numerical convergence. In the case of $k_1 = 1000$ the error in maximum norm may seem very big for the low resolutions. But the maximum norm of the solution itself depends also of the value of k_1. If the error was expressed in a normalized maximum norm error, the contrast with the error obtained in the case $k_1 = 2$ would be attenuated.

N	L^1 error	order	L^∞ error	order
30	4.390×10^{-1}	-	8.590×10^{-2}	-
60	1.364×10^{-1}	1.69	3.474×10^{-2}	1.31
120	2.581×10^{-2}	2.04	6.351×10^{-3}	1.88
240	7.580×10^{-3}	1.95	1.653×10^{-3}	1.90
480	2.333×10^{-3}	1.89	4.269×10^{-4}	1.91

Table 1: Numericals results for Problem 1, for $k_1 = 2$ and $k_2 = 1$.

N	L^1 error	order	L^∞ error	order
30	56.567×10^{0}	-	19.147×10^{0}	-
60	13.502×10^{0}	2.07	5.110×10^{0}	1.91
120	3.128×10^{0}	2.09	1.459×10^{0}	1.86
240	7.638×10^{-1}	2.07	3.436×10^{-1}	1.93
480	2.009×10^{-1}	2.04	9.651×10^{-2}	1.91

Table 2: Numericals results for Problem 1, for $k_1 = 1000$ and $k_2 = 1$.

5.1.2 Problem 2

We compute now the solution for a more complex interface. Ω_1 is defined by the intersections of the circles of radius 1.5 whose centers are located on points $(1.6, 1.6)$, $(1.6, -1.6)$, $(-1.6, 1.6)$ and $(-1.6, -1.6)$. Ω_2 consists in $[-2, 2] \times [-2, 2] \setminus \Omega_2$. Σ is the interface between Ω_1 and Ω_2.

The exact solution is:

$$u(x,y) = \begin{cases} cos(x) + cos(y) \text{ in } \Omega_1 \\ cos(y)e^x \text{ in } \Omega_2. \end{cases}$$

Numerical results for $k_1 = 2$ or 100 and $k_2 = 1$ for the discrete L^∞ norm are presented on Table **??**. For the smaller values of grid points, the observed convergence rate differs somewhat from second order, but when the number of points increases the asymptotic convergence is really second order. This test-case shows that our method maintains its second order accuracy in the case of a complex interface with sharp edges.

N	L^∞ error	order
20	7.191×10^{-3}	-
40	7.799×10^{-4}	3.22
80	2.334×10^{-4}	2.47
160	6.852×10^{-5}	2.24
240	2.886×10^{-5}	2.22
320	1.711×10^{-5}	2.18
400	1.057×10^{-5}	2.18

Table 3: Numericals results for Problem 2

5.1.3 Problem 3

It is a test case appearing in [**?**] (MIB method, case 3 of the tests on irregular interfaces) and [**?**] (CIM, example 4). We consider an elliptical interface Σ defined as:

$$(\frac{x}{18/27})^2 + (\frac{y}{10/27})^2 = 1.$$

The exact solution is:

$$u(x,y) = \begin{cases} e^x cos(y), \text{ inside } \Sigma \\ 5e^{-x^2 - \frac{y^2}{2}}, \text{ otherwise.} \end{cases}$$

As in [**?**] we fix the diffusion coefficient to be 1 ouside the interface, and we choose 10 or 1000 inside the interface.
For this test case we still observe the second order convergence. For $b = 1000$ our method provides smaller errors than the two others Cartesian methods. For $b = 10$ we obtain more accurate results than with the MIB method, and slightly less accurate than with the CIM.

N	L^∞ error	order	L^∞ error for MIB [**?**]	L^∞ error for CIM [**?**]
20	8.115×10^{-3}	-	2.659×10^{-2}	4.067×10^{-3}
40	9.152×10^{-4}	3.15	5.206×10^{-3}	6.171×10^{-4}
80	3.221×10^{-4}	2.33	1.487×10^{-3}	1.682×10^{-4}
160	6.335×10^{-5}	2.33	3.746×10^{-4}	3.975×10^{-5}
320	1.212×10^{-5}	2.35	7.803×10^{-5}	7.390×10^{-6}

Table 4: Numericals results for Problem 3, for $k = 10$ inside the interface

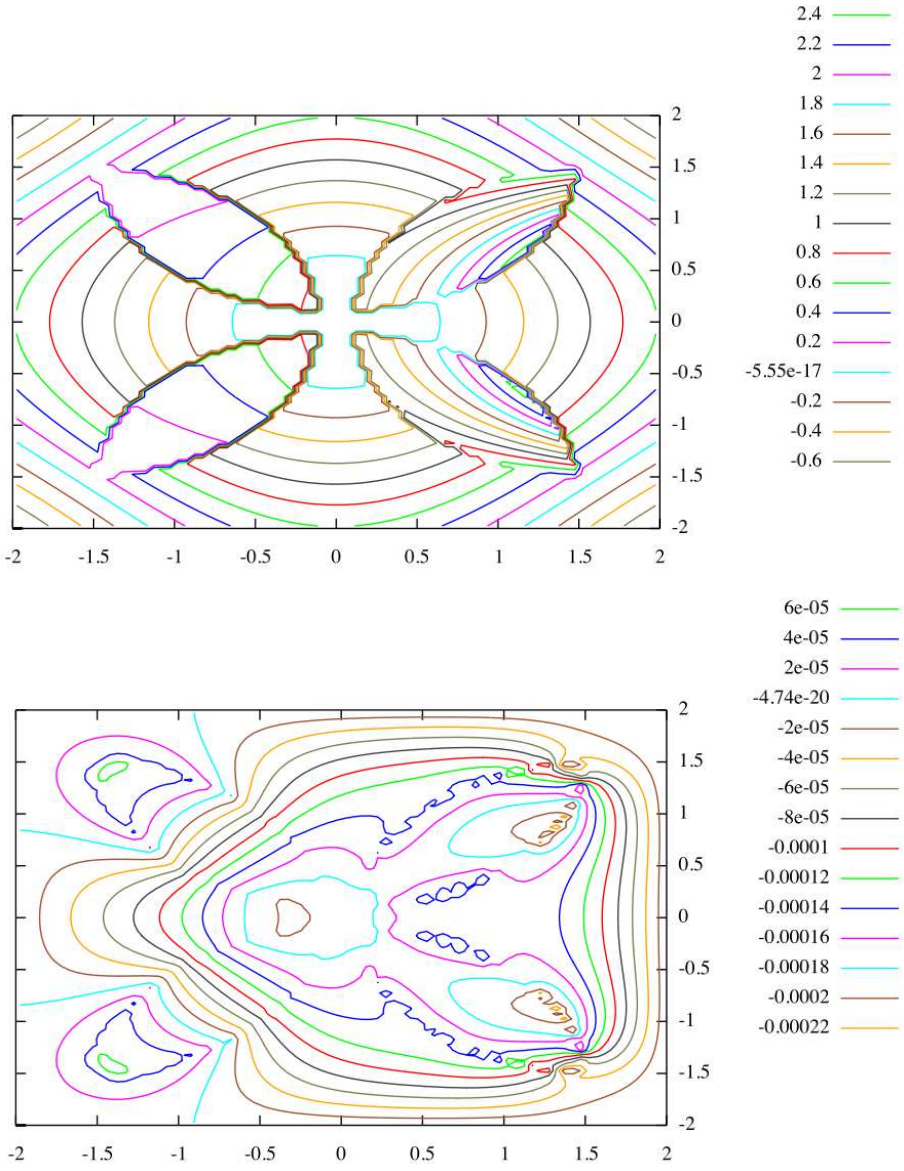

Figure 7: Numerical solution (top) and error (bottom) for $n_x = n_y = 80$ for Problem 2

5.1.4 Problem 4

It is a test case studied in several references: [?], [?], [?], [?] and [?]. A slightly different problem is also considered in [?]. We consider an spherical interface Σ defined by:

$$r^2 = 1/4$$

N	L^∞ error	order	L^∞ error for MIB [?]	L^∞ error for CIM [?]
20	1.083×10^{-1}	-	9.130×10^{-2}	3.539×10^{-1}
40	4.094×10^{-2}	1.40	2.764×10^{-2}	1.100×10^{-1}
80	7.045×10^{-3}	1.97	7.524×10^{-3}	2.028×10^{-2}
160	1.824×10^{-3}	1.96	2.169×10^{-3}	6.462×10^{-3}
320	4.671×10^{-4}	1.97	4.841×10^{-4}	1.437×10^{-3}

Table 5: Numericals results for Problem 3, for $k = 1000$ inside the interface

with $r = \sqrt{x^2 + y^2}$. The coefficient k varies in space:

$$k(x, y) = \begin{cases} r^2 + 1 \text{ inside } \Sigma \\ b \text{ otherwise.} \end{cases}$$

The exact solution is:

$$u(x, y) = \begin{cases} r^2 \text{ inside } \Sigma \\ (1 - \frac{1}{8b} - \frac{1}{b})/4 + (\frac{r^4}{2} + r^2)/b + C\, log(2\, r)/b \text{ otherwise,} \end{cases}$$

with b a parameter that we make vary: $b = 10, 1000$ and 0.001. The source term is :

$$f(x, y) = 8(x^2 + y^2) + 4.$$

For this test, we simply observe that our method produces results similar to the other methods of the literature.

N	Current method	IIM [?]	DIIM [?]	EJIIM [?]	MIIM [?]	CIM[?]
20	4.623×10^{-4}	3.5195×10^{-3}	5.378×10^{-4}	7.6×10^{-4}	-	1.259×10^{-3}
40	1.364×10^{-4}	7.5613×10^{-4}	1.378×10^{-4}	2.4×10^{-4}	4.864×10^{-4}	2.565×10^{-4}
80	4.431×10^{-5}	1.6512×10^{-4}	3.470×10^{-5}	7.9×10^{-5}	1.448×10^{-4}	5.215×10^{-5}
160	1.568×10^{-5}	3.6002×10^{-5}	8.704×10^{-6}	2.2×10^{-5}	3.012×10^{-5}	1.142×10^{-5}
320	7.053×10^{-6}	8.4405×10^{-6}	2.177×10^{-6}	5.3×10^{-6}	8.226×10^{-6}	2.725×10^{-6}

Table 6: Numericals results in L^∞ norm for Problem 4, $b = 10$.

N	Current method	DIIM [?]	MIIM [?]	CIM [?]
32	3.393×10^{-4}	2.083×10^{-4}	5.136×10^{-4}	2.732×10^{-4}
64	1.020×10^{-4}	5.296×10^{-5}	8.235×10^{-5}	3.875×10^{-5}
128	2.231×10^{-5}	1.330×10^{-5}	1.869×10^{-5}	5.337×10^{-6}
256	7.391×10^{-6}	3.330×10^{-6}	4.026×10^{-6}	7.241×10^{-7}

Table 7: Numericals results in L^∞ norm for Problem 4, $b = 1000$.

N	Current method	DIIM [?]	MIIM [?]	CIM [?]
32	2.572×10^{0}	4.971×10^{0}	9.346×10^{0}	4.278×10^{-1}
64	3.014×10^{-1}	1.176×10^{0}	2.006×10^{0}	1.260×10^{-1}
128	1.940×10^{-1}	2.900×10^{-1}	5.808×10^{-1}	3.773×10^{-2}
256	6.233×10^{-2}	7.086×10^{-2}	1.374×10^{-1}	1.365×10^{-2}

Table 8: Numericals results in L^{∞} norm for Problem 4, $b = 0.001$.

5.2 Numerical study of the parallelized method

In order to describe the performances of the parallel code implementing the method, we conducted some numerical experiments on a fixed grid, varying the number of processors. We chose a case presented in section **??**, specifically the one in Problem 1.

As we have anticipated in section **??**, PETSc allows to choose at runtime among a certain number of linear iterative solvers and preconditioners. For our tests we chose the restarted GMRES solver with Additive Schwarz Method (ASM) preconditioner with overlapping between matrix blocks with one block per process, each of which is solved with ILU(n). Therefore, several tests were conducted to explore the parameters space (defined by the dimension of the Krylov space, i.e. the GMRES restart, the ASM overlapping and the ILU level over the single block) and to find the best performance for every number of processors.

The scalability results are presented in Figure **??**. It shows the relationship between the number of processors and the calculation time. Data have been fitted with a power law, $t = aN^b$, where t is the calculation time, N is the number of processors and an estimation of the parameters a and b is given. The trend of the data implies we are not so far from a perfect parallelism, $b = -1$.

6 Conclusion

In this paper we have presented a parallel second order Cartesian method to solve elliptic problems with discontinuous coefficients on interfaces. The method is based on a dimensional splitting and on the use of new unknowns located on the interface. The discretization of the elliptic operator near the interface is performed with standard finite differences formulas involving these interface unknowns, which are themselves determined by solving discretized flux transmissions equations at the interface. The use of interface unknowns allows to decouple the discretization of the elliptic operator from the treatment of the interface transmission conditions. This decoupling makes our method particularly simple to implement, and allow easy modifications in the discretization of the elliptic operator or of the interface jump conditions. Moreover, if the interface transmission conditions themselves are modified, only their discretization needs to be changed to solve the new problem. The method is parallelized with the PETSc library in a straightforward manner. It is is second order accurate, with an absolute error competitive with the others method of the literature, and its parallel implementation shows good scalability properties. We have only presented the method in the two-dimensional case, but as we have explained in

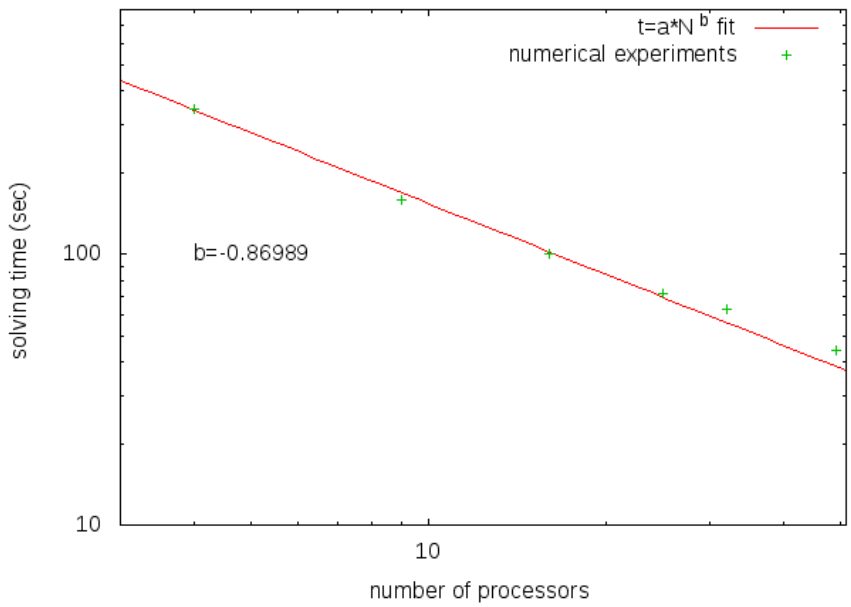

Figure 8: This figure shows how the calculation time scales with the number of processors. Crosses: experimental data. Line: least square fit of the data. The experiments have been conducted on the machine Fourmi at PlaFRIM (see, [**?**])

subsection **??**, extending it to three-dimensional problems would not especially increase its complexity. The method is indeed based on a dimensional splitting, and adding one dimension only requires to discretize the first and second derivatives along the new direction with the same principles than for the two others.

Centre de recherche INRIA Bordeaux – Sud Ouest
Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex (France)

Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399