A parallel second order Cartesian method for elliptic interface problems

Marco Cisternino[1] and Lisl Weynans[2,*]

[1] *Dipartimento di Ingegneria Aeronautica e Spaziale, Politecnico di Torino C.so Duca degli Abruzzi 24, 10129 Torino, Italy.*
[2] *Univ. Bordeaux, IMB, UMR 5251, F-33400 Talence, France. CNRS, IMB, UMR 5251, F-33400 Talence, France. INRIA, F-33400 Talence, France.*
Fax number: 33 5 40 00 21 23

Abstract. We present a parallel Cartesian method to solve elliptic problems with complex immersed interfaces. This method is based on a finite difference scheme and is second order accurate in the whole domain. The originality of the method lies in the use of additional unknowns located on the interface, allowing to express straightforwardly the interface transmission conditions. We describe the method 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.

AMS subject classifications: 65N06, 65N12, 65Y05

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

*Corresponding author. *Email addresses:* marco.cisternino@polito.it (M. Cisternino), lisl.weynans@math.u-bordeaux1.fr (L. Weynans)

1 Introduction

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

$$\nabla.(k\nabla u) = f \text{ on } \Omega = \Omega_1 \cup \Omega_2 \tag{1.1}$$

$$[\![u]\!] = \alpha \text{ on } \Sigma \tag{1.2}$$

$$[\![k\frac{\partial u}{\partial n}]\!] = \beta \text{ on } \Sigma \tag{1.3}$$

assorted with boundary conditions on $\delta\Omega$ defined as the boundary of Ω, and where $[\![\cdot]\!]$ means $\cdot_1 - \cdot_2$. As illustrated on Figure 1, Ω consists in the union of 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 [11], or in the modelling of electric potential in biological cells [12]. In this latter case the jump of the solution across the interface is proportional to the interior normal derivative.

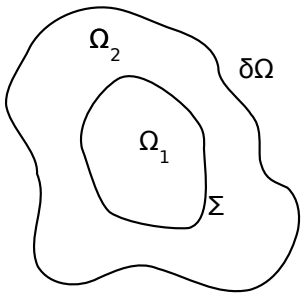

Figure 1: Geometry considered: two subdomains Ω_1 and Ω_2 separated by a complex interface Σ.

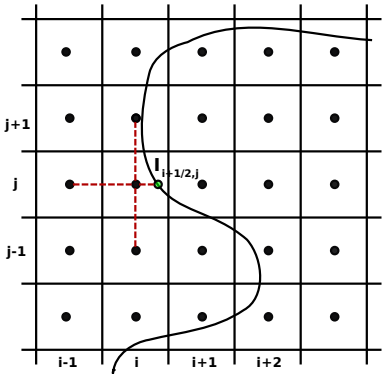

Figure 2: Example of stencil for the discretization of the elliptic operator

To solve an elliptic interface problem in the case of a complex interface, an alternative approach to body-fitted methods (see for instance [5], [10] and [13]) 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 need of remeshing if the interface moves.

The first Cartesian grid method for elliptic problems was designed by Mayo in 1984 [30], and developed further in [31] and [32]. In that work an integral equation was derived to solve elliptic interface problems with piecewise coefficients to second order accuracy in maximum norm. Then LeVeque and Li (1994) [25] devised the very well known

Immersed Interface Method (IIM). This method 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. Li [26] developed 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 [42] 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 [27] 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) [9]. 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: [46], [45], [43]. 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. These fictitious values are computed by iteratively enforcing the lowest order interface jump conditions. Finally, Chern and Shu [14] 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. ([18], [19]) developed methods inspired by Fedkiw's Ghost-Fluid Method ([17], [16]) 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 [4] and Angot et al [3], 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.

All the methods cited before can be considered as finite differences methods, and this is the context in which we aim to present our study. However, Cartesian method for elliptic interface problems also exist in the finite-volume community: 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 ([24] and [33]). Among the finite-element community, let us give some references: [5], [10], [13], [22], [15], [23], [34] and [29].

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 problem defined by equations (1.1)-(1.3) 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. 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 2 we firstly discuss the influence of the truncation error on the convergence order. Then we present the method in section 3 and its parallelization performed with the PETSC library in section 4. Finally we present numerical results validating the order of convergence of the method and its scalability in section 5.

2 Convergence rate dependence on truncation error

In this section, we consider the Laplacian equation in 1D and analyse of the convergence error in terms of the truncation error. This will allow us to explain the choice of a discretization near the interface in order to get a second order accuracy.

First of all, let us point out that the truncation error on a given point has an effect on the convergence rate depending of the point location. We assume that a boundary or an interface lies between grid points i and $i+1$. To approximate the Laplacian operator at point i we consider a ghost-cell formula. We create a fictitious value at point $i+1$ taking into account the boundary or interface condition $u(x_{int}) = u_{int}$. u_{int} may be a fixed value if we consider a Dirichlet boundary condition or may depend on the solution on the other side of the interface if we consider interface flux transmissions. If we use a linear

extrapolation the ghost value at point $i+1$will be:

$$\tilde{u}_{i+1} = u_i + dx\frac{u_{int}-u_i}{x_{int}-x_i} \tag{2.1}$$

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

$$\epsilon_i = \frac{dx\frac{(x_{int}-x_i)}{2}u''(x_i)+\frac{dx^2}{2}u''(x_i)+O(dx^3)}{dx^2} = O(1) \tag{2.2}$$

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

In the case of boundary conditions, theoretical studies performed by Gustafsson ([20] and [21]) and more recently by Svard and Nordstrom [41] show that under certain assumptions a discretization less accurate at the boundary than in the rest of the domain does not deteriorate the order of convergence. In fact the numerical methods developed in the ghost-cell spirit ([18], [19]) are practically at least second order accurate for Dirichlet boundary conditions. However, in the case of immersed interface problems, only first order convergence is reached for ghost cell methods, like for instance in [28] and [38]. For this reason, we study in a simple case the influence of the truncation error near the interface.

We consider the one-dimensional Laplace equation on the segment $[0,1]$, with Dirichlet boundary conditions. The following jump conditions are satisfied at the interface Σ located at $x = x_{int}$.

$$[\![u]\!] = 0 \text{ on } \Sigma \tag{2.3}$$

$$[\![k\frac{\partial u}{\partial n}]\!] = 0 \text{ on } \Sigma \tag{2.4}$$

We assume that $k = k'$ in $[0,x_{int}]$ and $k = k''$ in $[x_{int},1]$. Grid points are defined on locations $x_m = mdx$, $0 \leq m \leq N+1$ with $dx = \frac{1}{N+1}$ and x_{int} belongs to the segment $]x_i,x_{i+1}[$. The interface stands inside the domain so we can write $i \sim aN$, with a a real between 0 and 1, independent of dx. For the grid points inside the domain and far enough from the interface, we use the second order discretization:

$$\frac{u_{m+1}-2u_m+u_{m-1}}{dx^2} = f_m \tag{2.5}$$

with f_m the value of f at point x_m. The local error $e_m = u(x_m)-u_m$ satisfies the same linear relationship as u_k with the local truncation error ϵ_m as a source term:

$$\frac{e_{m+1}-2e_m+e_{m-1}}{dx^2} = \epsilon_m. \tag{2.6}$$

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 numerical solution at the

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

$$\begin{aligned} \frac{u_{int}-u_i}{x_{int}-x_i} - \frac{u_i-u_{i-1}}{dx} &= dx f_i \quad (2.7) \\ \frac{u_{i+2}-u_{i+1}}{dx} - \frac{u_{i+1}-u_{int}}{x_{i+1}-x_{int}} - &= dx f_{i+1}. \quad (2.8) \end{aligned}$$

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

$$\begin{aligned} \frac{e_{int}-e_i}{x_{int}-x_i} - \frac{e_i-e_{i-1}}{dx} &= dx\,\epsilon_i \quad (2.9) \\ \frac{e_{i+2}-e_{i+1}}{dx} - \frac{e_{i+1}-e_{int}}{x_{i+1}-x_{int}} &= dx\,\epsilon_{i+1} \quad (2.10) \end{aligned}$$

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

$$k_{i+1}\frac{u_{i+1}-u_{int}}{x_{i+1}-x_{int}} - = k_i \frac{u_{int}-u_i}{x_{int}-x_i} \quad (2.11)$$

with k_i the value of coefficient k at point x_i. Thus the truncation error ϵ_{int} related to this equation satisfies:

$$k_{i+1}\frac{e_{i+1}-e_{int}}{x_{i+1}-x_{int}} - k_i \frac{e_{int}-e_i}{x_{int}-x_i} = \epsilon_{int}. \quad (2.12)$$

The truncation error of the Laplacian discretizations (2.7) and (2.8) are zero order accurate, and the truncation error of the flux equation (2.11) is order one accurate, as a Taylor expansion would show it in the same way as in (2.2) . In order to close the linear system, we assume that Dirichlet boundary conditions are imposed exactly:

$$e_0 = 0 \text{ and } e_{N+1} = 0. \quad (2.13)$$

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:

$$e_{m+n} = (n+1)e_m - ne_{m-1} + dx^2 \sum_{j=1}^{n} j\epsilon_{m+n-j} \quad (2.14)$$

$$e_{m-n} = (n+1)e_m - ne_{m+1} + dx^2 \sum_{j=1}^{n} j\epsilon_{m-n+j} \quad (2.15)$$

Using the boundary conditions, we deduce from (2.14) and (2.15) that:

$$0 = (N+1-m)e_{m-1}-(N+2-m)e_m-dx^2\sum_{j=1}^{N+1-m} j\epsilon_{N+1-j} \text{ for } m\geq i+2 \quad (2.16)$$

$$0 = (m+1)e_m-me_{m+1}+dx^2\sum_{j=1}^{m} j\epsilon_j \text{ for } m\leq i-1 \quad (2.17)$$

Therefore:

$$e_{m+1} = \frac{(N-m)e_m}{(N+1-m)}-dx^2\sum_{j=1}^{N-m}\frac{j}{(N+1-m)}\epsilon_{N+1-j} \text{ for } m\geq i+1 \quad (2.18)$$

$$e_{m+1} = \frac{(m+1)}{m}e_m+dx^2\sum_{j=1}^{m}\frac{j}{m}\epsilon_j \text{ for } m\leq i-1 \quad (2.19)$$

Using the latter equations with (2.9), (2.10) and (2.12) we can prove that:

$$e_{int}(\frac{k_1}{x_{int}}+\frac{k_2}{1-x_{int}}) = -\epsilon_{int}-dx^2\frac{k_1}{x_{int}}\sum_{j=1}^{N-i} j\epsilon_{N+1-j}-dx^2\frac{k_2}{1-x_{int}}\sum_{j=1}^{i} j\epsilon_j \quad (2.20)$$

$$e_i = \frac{x_i}{x_{int}}e_{int}-\frac{(x_{int}-x_i)}{x_{int}}dx^2\sum_{j=1}^{i} j\epsilon_j. \quad (2.21)$$

Therefore if the truncation error of the flux equation ϵ_{int} is order one, or if the truncation error of the Laplacian is order zero near the interface: $\epsilon_i = O(1)$ and $\epsilon_{i+1} = O(1)$, then the errors e_{int} and e_i are a priori order one. We deduce that to obtain an order 2 accuracy, we need to:

- use a discretization of Laplacian near the interface with a truncation error of order 1, thus avoid linear extrapolations,
- use a discretization of the flux transmission equations at the interface with a truncation error of order 2.

This analysis is 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 in the expression of the error. However numerical results with ghost-cell like methods in [28] and [38] corroborate this reasoning.

3 Description of the method

In this section we firstly describe the method in the case $\alpha = \beta = 0$, where α and β are the jumps of the solution and its normal derivative, as presented in equations (1.2) and (1.3). The case $\alpha \neq 0$, $\beta \neq 0$ is handled in the last part of the section.

3.1 Interface description and classification of grid points

In order to improve accuracy in the vicinity of the interface we need additional geometric information. This information, mainly the distance from the interface and the normal to the interface, can be provided by the level set method, introduced by Osher and Sethian [36]. We refer the interested reader to [39], [40] and [35] for recent reviews of this method. The zero isoline of the level set function, defined by:

$$\varphi(x) = \begin{cases} dist_\Sigma(x) & \text{outside of the interface} \\ -dist_\Sigma(x) & \text{inside of the interface} \end{cases} \tag{3.1}$$

represents implicitely the interface Σ immersed in the computational domain. A useful property of the level set function is:

$$\mathbf{n}(x) = \nabla\varphi(x) \tag{3.2}$$

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.

We chose to use a level-set function to represent the interface because with the level-set the information is carried by grid points, which is convenient in our case, especially for parallelization. But in fact, we could use any representation of the interface allowing to know to which subdomain each grid point belongs and to compute interface points and normals to the interface.

The points on the Cartesian grid are defined by $M_{i,j} = (x_i, y_j) = (idx, jdy)$. 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 to approximate 1.1:

$$\begin{aligned} \nabla\cdot(k\nabla u)(x_i,y_j) \quad \approx \quad & \frac{k_{i+1/2,j}(u_{i+1,j}-u_{i,j}) - k_{i-1/2,j}(u_{i,j}-u_{i-1,j})}{dx^2} \\ & + \frac{k_{i,j+1/2}(u_{i,j+1}-u_{i,j}) - k_{i,j-1/2}(u_{i,j}-u_{i,j-1})}{dy^2} \end{aligned} \tag{3.3}$$

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

3.3 Discrete elliptic operator near the interface

On grid points neighbouring the interface, the latter approximation is not accurate enough, because of the discontinuity of the coefficient k. To discretize more accurately the term

$\nabla(k\nabla u)(x_i,y_j)$, we create new unknowns, located at what we call "intersection points". The intersection points are defined the following way: if the intersection of the interface and $[M_{i,j}M_{i+1,j}]$ exists, then we define the intersection point $I_{i+1/2,j} = (\tilde{x}_{i+1/2,j}, y_j)$ as this intersection. The intersection point $I_{i,j+1/2} = (x_i, \tilde{y}_{j+1/2,j})$ is similarly defined as the intersection, if it exists, of the interface and the segment $[M_{i,j}M_{i,j+1}]$.

For a grid point $M_{i,j}$ neighbouring the interface, we discretize $\nabla\cdot(k\nabla u)(x_i,y_j)$ with the values on $M_{i,j}$ and the closest points (grid or intersection points) to $M_{i,j}$ in each direction. For instance, in the case illustrated on Figure 2, we get:

$$\begin{aligned}\nabla\cdot(k\nabla u)(x_i,y_j) = &\frac{\tilde{k}_{i+1/2,j}\frac{\tilde{u}_{i+1/2,j}-u_{i,j}}{\tilde{x}_{i+1/2,j}-x_i} - k_{i-1/2,j}\frac{u_{i,j}-u_{i-1,j}}{dx}}{\frac{(\tilde{x}_{i+1/2,j}-x_i)}{2}+\frac{dx}{2}} \\ &+\frac{k_{i,j+1/2}(u_{i,j+1}-u_{i,j}) - k_{i,j-1/2}(u_{i,j}-u_{i,j-1})}{dy^2}\end{aligned} \tag{3.4}$$

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 $\tilde{x}_{i-1/2,j}$ coincides with x_{i+1}, and first order otherwise.

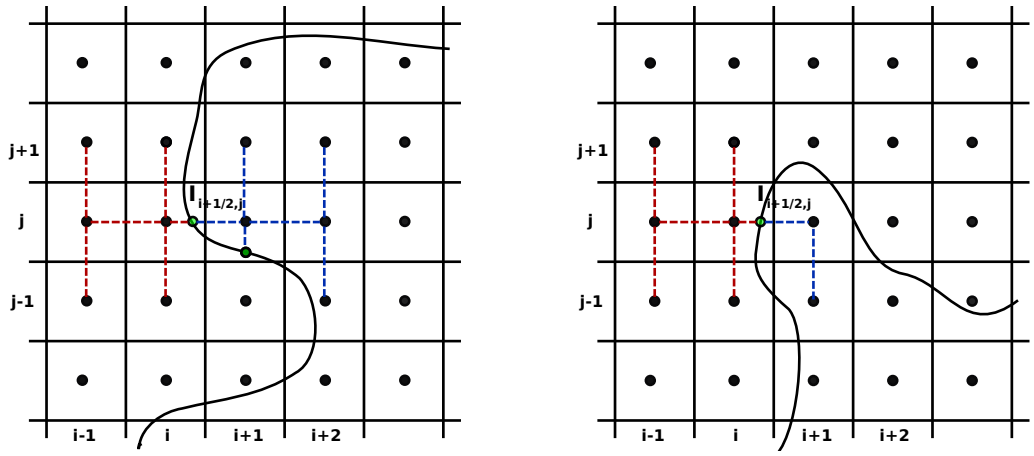

Figure 3: Example of order two (left) and one (right) flux discretization at point $I_{i+1/2,j}$.

3.4 Discrete flux transmission conditions

As we have seen in the last section, we want the truncation error of the discretization of flux equality (1.3) to be second order accurate. On Figure 3 (left side) 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 with three a priori 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:

$$\begin{aligned}\frac{\partial u^l}{\partial x}(\tilde{x}_{i+1/2,j},y_j) \approx & \frac{(u_{i-1,j}-\tilde{u}_{i+1/2,j})(x_i-\tilde{x}_{i+1/2,j})}{dx(x_{i-1}-\tilde{x}_{i+1/2,j})}+ \\ & -\frac{(u_{i,j}-\tilde{u}_{i+1/2,j})(x_{i-1}-\tilde{x}_{i+1/2,j})}{dx(x_i-\tilde{x}_{i+1/2,j})}\end{aligned} \tag{3.5}$$

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 parallel to the y-axis passing by $I_{i+1/2,j}$. Therefore we use an extended stencil including 6 points. We use a linear combination of $(\frac{\partial u}{\partial y})_{i,j}$ and $(\frac{\partial u}{\partial y})_{i-1,j}$, defined respectively as second order approximations of the y-derivative on $M_{i,j}$ and $M_{i-1,j}$ to approximate the derivative on $I_{i+1/2,j}$ with second order accuracy:

$$\frac{\partial u^l}{\partial y}(\tilde{x}_{i+1/2,j},y_j) \approx \frac{\tilde{x}_{i+1/2,j}-x_{i-1}}{dx}(\frac{\partial u}{\partial y})_{i,j}-\frac{\tilde{x}_{i+1/2,j}-x_i}{dx}(\frac{\partial u}{\partial y})_{i-1,j} \tag{3.6}$$

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 (3.5). The formulas (3.5) and (3.6) are consistent if the point $M_{i-1,j}$ belongs to the same domain as $M_{i,j}$. We thus need that there are at least two adjacent points in each direction belonging to the same domain. If on one side of the interface the two closest grid points aligned with the intersection point do not belong to the same subdomain, the second order discretization is not possible anymore. In this case, we use instead for this side of the interface a first order discretization involving only three points. Such a case is illustrated on Figure 3 (left side).

At the end the flux equality (1.1) corresponding to the case described in Figure 3 (right side) is discretized by:

$$[k^l(\frac{\partial u^l}{\partial x})_{i+1/2,j}-k^r(\frac{\partial u^r}{\partial x})_{i+1/2,j}]\,n_x+[k^l(\frac{\partial u^l}{\partial y})_{i+1/2,j}-k^r(\frac{\partial u^r}{\partial y})_{i+1/2,j}]\,n_y = 0 \tag{3.7}$$

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 $I_{i+1/2,j}$.

The stencil used in this discretization of the flux equality contains 13 points. Actually, it is possible to use only 8 points. As other authors noticed ([42], [9], [46] and [14]) the jump condition on u can be differentiated in the direction tangential to the interface:

$$[\![\frac{\partial u}{\partial \tau}]\!] = \frac{\partial \alpha}{\partial \tau}. \tag{3.8}$$

The latter equation is a linear relationship between 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 $\frac{\partial u^r}{\partial y}$. For instance $\frac{\partial u^r}{\partial y}$ can be expressed as a linear combination of the others partial derivatives, and does not need to be discretized, removing

4 points from the stencil used for the discretization of the flux. Additionally, one can use only 5 points instead of 6 to discretize $\frac{\partial u^l}{\partial y}$ with second order accuracy. However, when we compared the 13 points and the 8 points versions, we noticed that the amplitude of the error was sensibly higher when using the 8 points stencil, while the computational time was almost the same. This observation is illustrated for Problem 3 of the numerical validation, in subsection 5.1.

Most second order cartesian methods of the literature use iterative procedures: [26] and [9], or higher order derivatives: [42], or local changes of coordinates near the interface: [25] and [26], or need to solve local linear systems: [25], [27]. Compared to these latters, the formulation of our method seems more straightforward, because we directly express the jump conditions and use them to built one linear system to solve. The MIB method [46] and the CIM [14] also built their linear systems straightforwardly. But the MIB method suffers some limitations in the case of sharp angles, overcomed at the price of additionnal complexity, through the use of primary and secondary auxiliary points. The CIM couples the values of the numerical solution on each side of the interface, resulting in an more complex formulation.

3.5 Stabilization

In our first numerical tests, we noticed oscillations in the convergence plots. These instabilities appeared when the discretization of a flux equation at an interface point involved another intersection point located very close to a grid point. This situation is illustrated on Figure 4. On this figure one can understand why such a configuration leads to numerical instabilities: the finite differences formula creates a coupling between the two interface unknowns and is ill-conditioned due to the small distance between the grid point and the intersection point.

To avoid these oscillations we use in these cases a decentered discretization. As illustrated in Figure 4, instead of using in the flux discretization the second intersection point, we use the closest grid point located in the opposite direction. For instance, in Eq. (3.7) the term $(\frac{\partial u^r}{\partial y})_{i+1/2,j}$ is computed with a second order finite differences formula involving the grid points $(i+1,j)$, $(i+1,j+1)$ and $(i+1,j+2)$ instead of $(i+1,j)$, $(i+1,j+1)$ and $I_{i+1,j-1/2}$. We decided to use the modified stencil in a systematic way every time that there are two intersection points involved in the same flux equation. The effect of the stabilization is illustrated for Problem 1 of the numerical validation in subsection 5.1.

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

Now we present how to take into account non-zero jumps of the function and 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. We define as interface unknowns at the points $I_{i+1/2,j}$ or $I_{i,j+1/2}$ the limit values of u in the domain defined by $\phi < 0$: $\tilde{u}_{i+1/2,j}$ or $\tilde{u}_{i,j+1/2}$. The values at the same points but for the domain where $\phi > 0$ are defined by

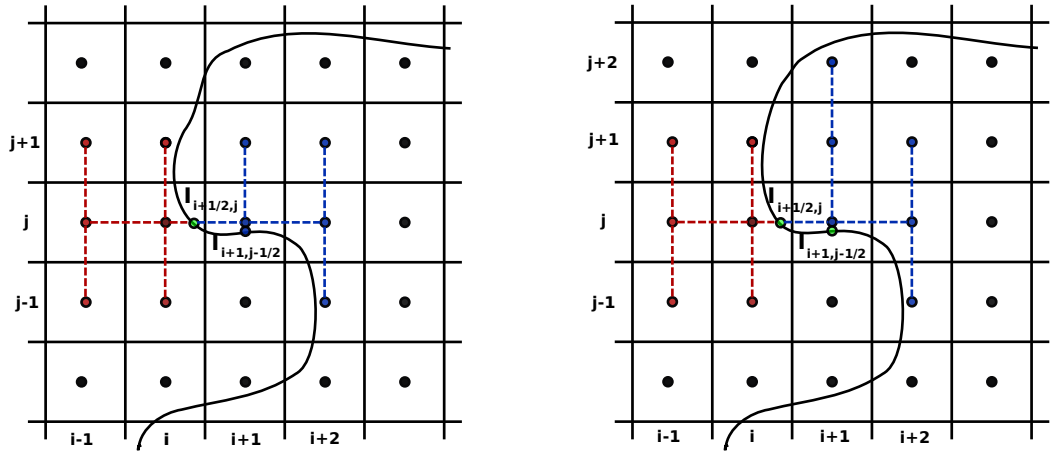

Figure 4: On left: Centered stencil: the discretization of the flux on point $I_{i+1/2,j}$ involves intersection point $I_{i+1,j-1/2}$ and is not numerically well conditionned. On right: Decentered stencil: the discretization of the flux on point $I_{i+1/2,j}$ involves grid point $(i+1,j+2)$ instead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

$\tilde{u}_{i+1/2,j}+\alpha(I_{i+1/2,j})$ or $\tilde{u}_{i,j+1/2}+\alpha(I_{i,j+1/2})$. Only the right hand side term of the linear system 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 coefficient used 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. (3.4) remains unchanged, but if $\varphi(M_{i,j})>0$ it becomes:

$$\frac{\tilde{k}_{i+1/2,j}\frac{\tilde{u}_{i+1/2,j}-u_{i,j}}{\tilde{x}_{i+1/2,j}-x_i}-k_{i-1/2,j}\frac{u_{i,j}-u_{i-1,j}}{dx}}{\frac{(\tilde{x}_{i+1/2,j}-x_i)}{2}+\frac{dx}{2}}+\frac{k_{i,j+1/2}(u_{i,j+1}-u_{i,j})-k_{i,j-1/2}(u_{i,j}-u_{i,j-1})}{dy^2} = -\frac{\frac{\tilde{k}_{i+1/2,j}}{(\tilde{x}_{i+1/2,j}-x_i)}}{\frac{(\tilde{x}_{i+1/2,j}-x_i)}{2}+\frac{dx}{2}}\alpha(I_{i+1/2,j})+f(x_i,y_j). \tag{3.9}$$

If the jump term β is non-zero, the coefficient $\beta(I_{i+1/2,j})$ or $\beta(I_{i,j+1/2})$ appears in the right hand side of the equation 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. (3.7) becomes:

$$\begin{aligned}&[k_{i,j}(\frac{\partial u^l}{\partial x})_{i+1/2,j}-k_{i+1,j}(\frac{\partial u^r}{\partial x})_{i+1/2,j}]\,n_x+\\&+[k_{i,j}(\frac{\partial u^l}{\partial y})_{i+1/2,j}-k_{i+1,j}(\frac{\partial u^r}{\partial y})_{i+1/2,j}]\,n_y \quad =\beta(I_{i+1/2,j}).\end{aligned} \tag{3.10}$$

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 to solve 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.

4.1 Parallelization model and PETSc library

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). The API (Application Program Interface) chosen to implement the former is MPI (Message Passing Interface, [1]). The computational domain is decomposed among the processes. The management of the sub-domains boundaries is approached by extending the sub-domain of a processor to points of the adjacent sub-domain, making up a new area, called ghost region. Between sub-domain boundary regions and ghost regions communications take place, in order to keep the latter up to date.

Using the PETSc library (Portable, Extensible Toolkit for Scientific Computation, see [7], [6], [8] for details) we were able to avoid explicit communications. PETSc 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.

4.2 Parallel Implementation

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

$$Au = f \tag{4.1}$$

where A is the matrix discretizing the differential operator on the Cartesian grid points and the conditions on the interface fluxes, according to the equations in Sections 3.2-3.4; 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 the values on the grid points and on the interface points. Therefore, the code searches the intersections, assembles A and f, solves the linear system and extracts the solution at the grid points.

4.2.1 The matrix

The matrix non-zeros pattern strongly depends on the position of the interface in the computational domain and on the grid points enumeration introduced by PETSc. Figure 6 shows the local and global enumeration of grid points and intersections. Without loss of generality we want to show what happen in the matrix for points near to the interface

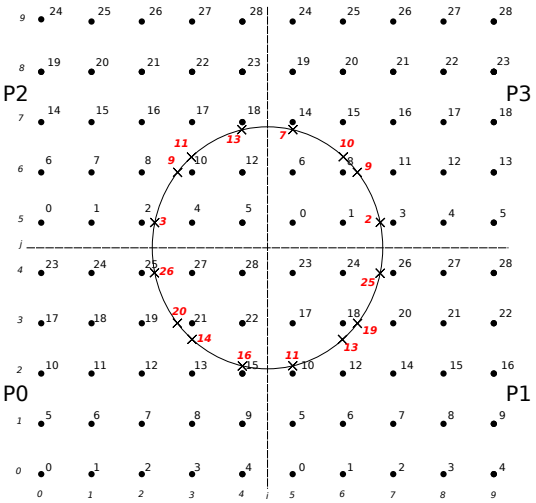

Figure 5: Example of intersections search (4 processors case). The local arrangements; the sequential vector stores the values (29,29,29,29). Intersections in red.

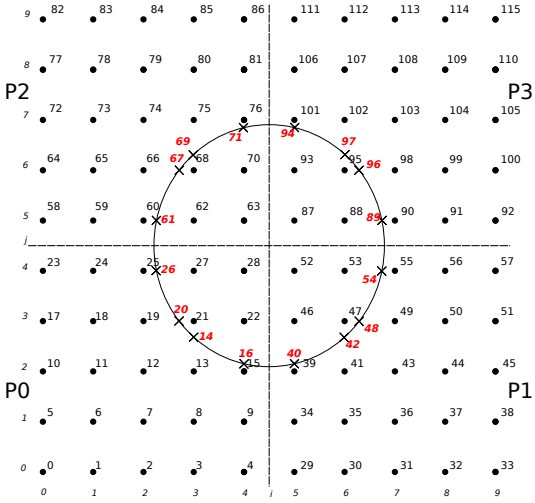

Figure 6: Example of intersections search (4 processors case). The final global arrangement. Intersections in red.

Figure 7 illustrates the matrix structure in the case of Figure 6 for grid points $0 \leq i \leq 9$ and $2 \leq j \leq 3$, and intersection points in the same area, in the case of Dirichlet boundary conditions.

The matrix is row partitioned: a number of rows equal to the sum of grid points and intersections in its own sub-domain is given to each processor. The solution and the right hand side vectors are partitioned in the same way. This distribution guarantees that communications take place only along the sub-domain boundaries.

4.2.2 Notes on intersections

In the parallel implementation of the method the search for intersection is local, i.e. every processor looks for intersection in its own sub-domain and fills parallel vectors defined on the regular grid in with information about existence and position of the intersections. This guarantees the best performance but needs communications to update the ghost regions. However, without a change in the arrangement every processor would number the intersections in an incompatible way compared with other processors. In order to avoid any ambiguity, every processor starts the search in its own domain, Figure 5; then, a sequential vector owned by all the processors is updated: every position in this vector is associated to a processor and stores the number of unknowns (grid points plus intersections) in every sub-domain. 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, and a global numeration of the intersections is obtained, Figure 6. By this way we can get as less communications as we can in solving the linear system.

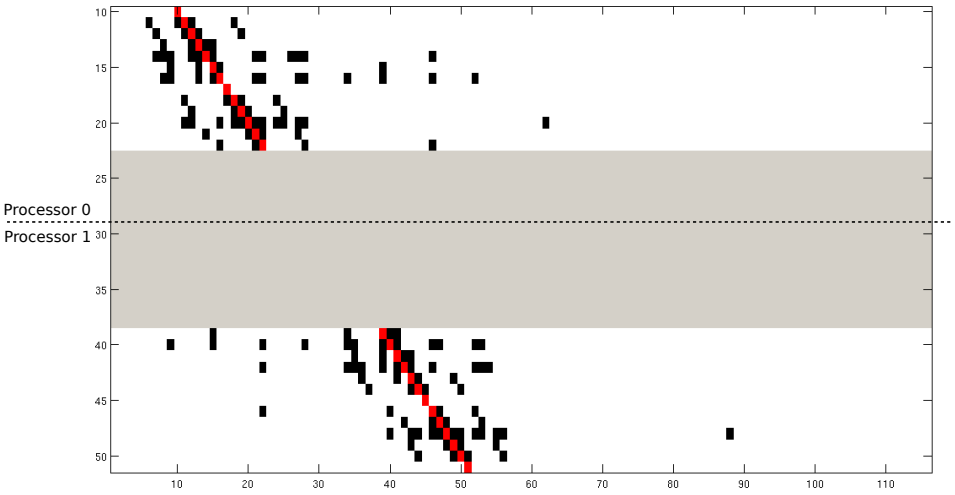

Figure 7: A-matrix rows relative to points (i,j) in the ranges $0 \leq i \leq 9$ and $2 \leq j \leq 3$, in Figure 6. White means zero valued matrix elements, black and red mean non-zeros (red for elements on the principal diagonal). Grey is for omitted rows (corresponding to points outside the considered ranges). The separation between Processor 0 and Processor 1 is shown.

4.3 Aspects of the method simplifying parallelization through PETSc library

Some aspects of this method make the parallelization easy to be accomplished, comparing the present one to the already existing methods. The use of the Cartesian grid and the level set function, putting information about the intersections on the Cartesian grid points make easy to distribute the interface among the processors, just partitioning the computational domain. Moreover, the discretization of the transmission conditions is performed without introducing local reference systems or local sub-problems, such as matrix inversions ([25]), optimization problems ([27]) or further jump conditions calculations ([9] and [42]). Surely these methods can simplify the matrix of the linear system, but it may be hard to preserve a good balance in processors computational load.

Concerning the use of the PETSc library, the most important structure to manage cartesian grids is the *DA*, the distributed array. Even if we have to deal with the intersection points, storing informations about them on the grid nodes makes the use of the *DA* possible to manage these points. Although the augmented nature of the linear system seems to complicate the extraction of the solution on the grid nodes, efficient routines and the partitioning introduced make the scattering between parallel vectors easy to be implemented with few code lines. 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). All these aspects makes parallel code quite similar to the sequential one, provided the right use of the PETSc structures and routines.

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. The linear systems are solved with routines from the SPARSKIT library [37]: a GMRES algorithm with a ILU preconditionning. In all the following test cases, we consider a square domain Ω consisting in the union of 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 equations (1.1 - 1.3) are solved on domain $\Omega = [-2,2] \times [-2,2]$ with the analytical solution:

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

with $k = k_1$ for $1 < r < 1.5$, $k = k_2$ for $r > 1.5$ and Dirichlet boundary conditions:

$$u(x,y) = 100 \text{ if } r = 1.$$

Numerical results for $k_1 = 2$ or 1000 and $k_2 = 1$ for the discrete L^∞ norm are presented on Table 1 and Table 3. On Table 2 are presented the results without stabilization for $k_1 = 2$. Plots of the numerical solution and the numerical error for $n_x = n_y = 120$ are presented in Figures 8 and 9. We observe global second order numerical convergence if the stabilization is applied. If not, the convergence order oscillates, and the accuracy is deteriorated for some values of grid points.

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
360	3.079×10^{-3}	1.99	7.435×10^{-4}	1.91
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	2.735×10^{-1}	-	6.651×10^{-2}	-
60	5.528×10^{-2}	2.31	3.386×10^{-2}	0.97
120	1.648×10^{-1}	0.36	5.784×10^{-2}	0.20
240	5.915×10^{-3}	1.84	1.638×10^{-3}	1.78
360	4.915×10^{-3}	1.62	1.905×10^{-3}	1.43
480	1.279×10^{-3}	1.94	4.287×10^{-4}	1.82

Table 2: Numericals results for Problem 1, for $k_1 = 2$ and $k_2 = 1$, without stabilization.

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

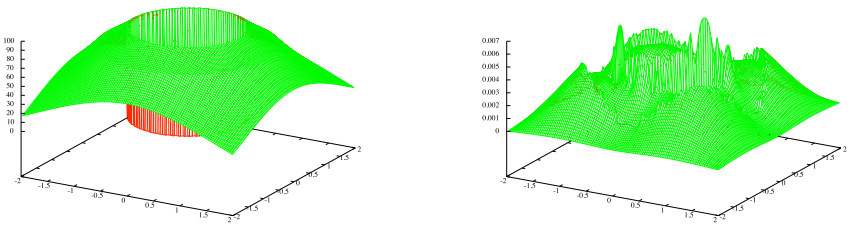

Figure 8: Numerical solution (left) and error (right) for $N = 120$ for Problem 1 with $k_1 = 2$.

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 $[-2,2]\times[-2,2]\backslash\Omega_1$. 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 in Ω_1 and $k_2 = 1$ in Ω_2 for the discrete L^∞ norm are

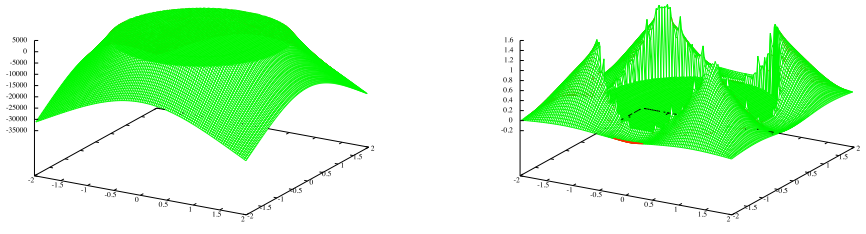

Figure 9: Numerical solution (left) and error (right) for $N=120$ for Problem 1 with $k_1=1000$.

presented on Table 4. Plots of the numerical solution and the numerical error for $N=80$ are presented on Figure 10. This test-case shows that our method maintains 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 4: Numericals results for Problem 2.

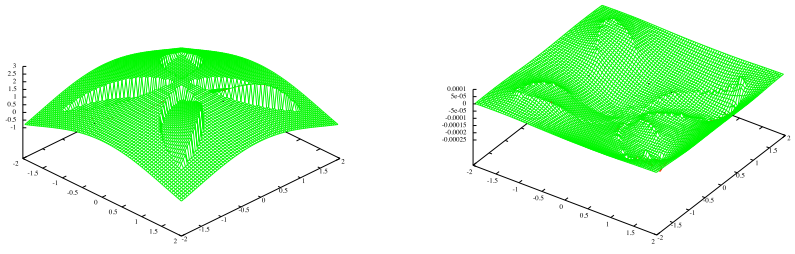

Figure 10: Numerical solution (left) and error (right) for $N=80$ for Problem 2.

5.1.3 Problem 3

It is a test case appearing in [46] (MIB method, case 3 of the tests on irregular interfaces) and [14] (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 [46] we fix the diffusion coefficient k 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, as presented in Tables 5 and 6. The numerical solution and error are plotted on Figures 11 and 12. For $k=1000$ our method provides smaller errors than the two others Cartesian methods. For $k=10$ we obtain more accurate results than with the MIB method, and slightly less accurate than with the CIM. In the case $k=1000$ inside the interface, we also present in Table 6 the numerical results obtained with the 8 points version of the method. We see that the 8 points version provides less accurate results than the 13 points version. The computational time is with the 13 points version 0.051 s for 80^2 grid points, 0.191s for 160^2 grid points and 0.763 s for 320^2 grid points, while it is 0.049 s for 80^2 grid points, 0.189s for 160^2 grid points and 0.769 s for 320^2 grid points with the 8 points version.

N	L^∞ error	order	L^∞ error for MIB [46]	L^∞ error for CIM [14]
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 5: Numericals results for Problem 3, for $k=10$ inside the interface.

5.1.4 Problem 4

It is a test case studied in several references: [25], [9], [42], [27] and [14]. A slightly different problem is also considered in [44]. We consider an spherical interface Σ defined by:

$$r^2=1/4$$

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

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

N	L^∞ error (13 points version)	order	L^∞ error (8 points version)	L^∞ error for MIB [46]	L^∞ error for CIM [14]
20	1.083×10^{-1}	-	1.005×10^{-1}	9.130×10^{-2}	3.539×10^{-1}
40	4.094×10^{-2}	1.40	4.715×10^{-2}	2.764×10^{-2}	1.100×10^{-1}
80	7.045×10^{-3}	1.97	2.967×10^{-2}	7.524×10^{-3}	2.028×10^{-2}
160	1.824×10^{-3}	1.96	7.979×10^{-3}	2.169×10^{-3}	6.462×10^{-3}
320	4.671×10^{-4}	1.97	2.120×10^{-3}	4.841×10^{-4}	1.437×10^{-3}

Table 6: Numericals results for Problem 3, for $k=1000$ inside the interface.

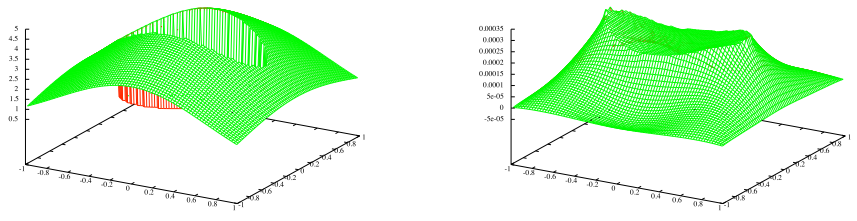

Figure 11: Numerical solution (left) and error (right) for $N=80$ for Problem 3 with $k=10$ inside the interface.

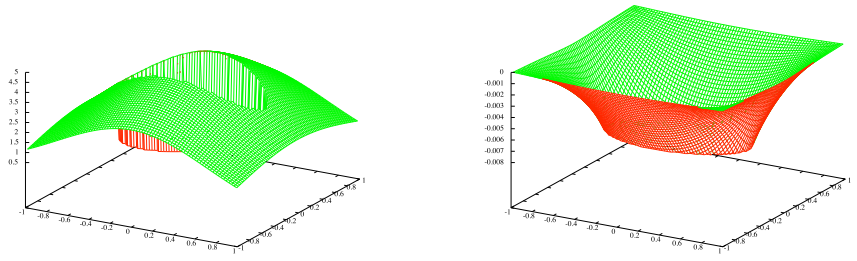

Figure 12: Numerical solution (left) and error (right) for $N=80$ for Problem 3 with $k=1000$ inside the interface.

The exact solution is:

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

with b a parameter appearing in the formula for the coefficient k and for the solution u, that we make vary: $b=10$, 1000 and 0.001. The source term is :

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

The numerical results and orders of convergence are presented in Tables 7, 8 and 9. The numerical solution and error are plotted in Figures 13, 14 and 15. For this test we again observe second order accuracy. If we compare to the results of other cartesian methods found in the literature, the current method is globally one of the most accurate.

N	Current method	Order	IIM [25]	DIIM [9]
20	4.204×10^{-4}	-	3.5195×10^{-3}	5.378×10^{-4}
40	1.161×10^{-4}	1.86	7.5613×10^{-4}	1.378×10^{-4}
80	3.757×10^{-5}	1.74	1.6512×10^{-4}	3.470×10^{-5}
160	5.332×10^{-6}	2.10	3.6002×10^{-5}	8.704×10^{-6}
320	1.581×10^{-6}	2.02	8.4405×10^{-6}	2.177×10^{-6}
N	EJIIM [42]	MIIM [27]	CIM [14]	
20	7.6×10^{-4}	-	1.259×10^{-3}	
40	2.4×10^{-4}	4.864×10^{-4}	2.565×10^{-4}	
80	7.9×10^{-5}	1.448×10^{-4}	5.215×10^{-5}	
160	2.2×10^{-5}	3.012×10^{-5}	1.142×10^{-5}	
320	5.3×10^{-6}	8.226×10^{-6}	2.725×10^{-6}	

Table 7: Numericals results in L^∞ norm for Problem 4, $b = 10$.

N	Current method	Order	DIIM [9]	MIIM [27]	CIM [14]
32	1.825×10^{-4}	-	2.083×10^{-4}	5.136×10^{-4}	2.732×10^{-4}
64	4.965×10^{-5}	1.88	5.296×10^{-5}	8.235×10^{-5}	3.875×10^{-5}
128	1.304×10^{-5}	1.90	1.330×10^{-5}	1.869×10^{-5}	5.337×10^{-6}
256	3.333×10^{-6}	1.92	3.330×10^{-6}	4.026×10^{-6}	7.241×10^{-7}

Table 8: Numericals results in L^∞ norm for Problem 4, $b = 1000$.

5.2 Numerical study of the parallelized method

In order to describe the performances of the parallel code implementing the method, we conduct here some numerical experiments on a fixed grid, varying the number of processors, and on finer and finer grids to study the error convergence rate.

5.2.1 Problem 1

We study the Problem 1, presented in section 5.1, with $k_1 = 1000$ and $k_2 = 1$. 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 and ILU on

N	Current method	Order	DIIM [9]	MIIM [27]	CIM [14]
32	2.036×10^{0}	-	4.971×10^{0}	9.346×10^{0}	4.278×10^{-1}
64	3.522×10^{-1}	2.53	1.176×10^{0}	2.006×10^{0}	1.260×10^{-1}
128	7.255×10^{-2}	2.41	2.900×10^{-1}	5.808×10^{-1}	3.773×10^{-2}
256	1.807×10^{-2}	2.27	7.086×10^{-2}	1.374×10^{-1}	1.365×10^{-2}

Table 9: Numericals results in L^{∞} norm for Problem 4, $b = 0.001$.

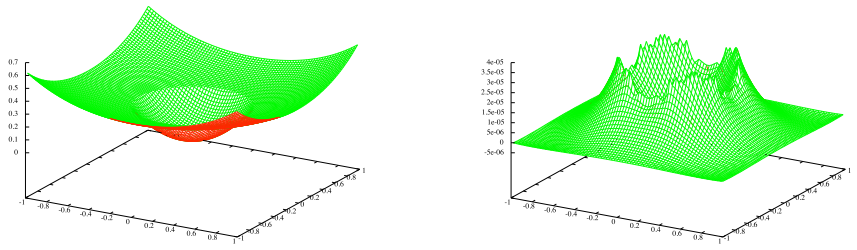

Figure 13: Numerical solution (left) and error (right) for $N = 80$ for Problem 4 with $b = 10$.

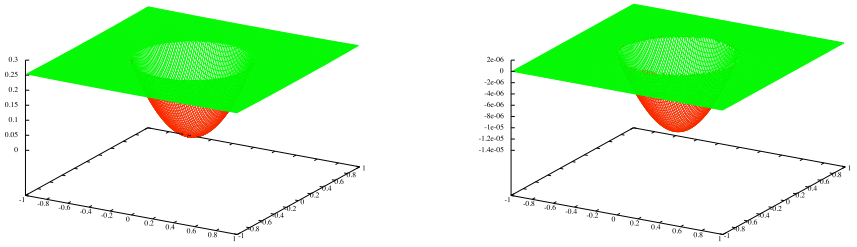

Figure 14: Numerical solution (left) and error (right) for $N = 128$ for Problem 4 with $b = 1000$.

each block. The scalability results are presented in Figure 16. 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, $b = -1$. The error convergence rate results are presented in Table 10. A global second order numerical convergence is observed for the parallel implementation of the method.

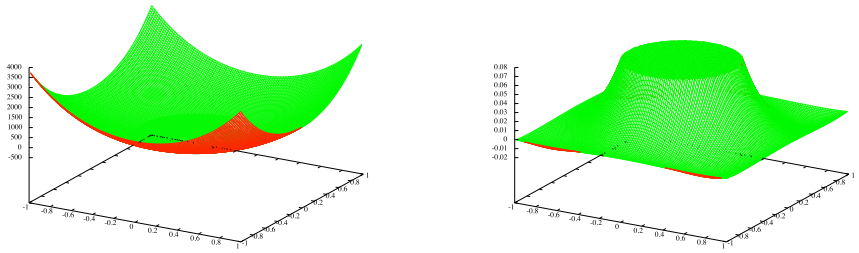

Figure 15: Numerical solution (left) and error (right) for $n_x = n_y = 128$ for Problem 4 with $b = 0.001$.

N	L^1 error	order	L^∞ error	order
800	7.897×10^{-3}	-	1.559×10^{-3}	-
1600	1.977×10^{-3}	2.00	3.917×10^{-4}	1.99
3200	4.960×10^{-4}	2.00	9.836×10^{-5}	1.99
6400	1.280×10^{-4}	1.98	2.546×10^{-5}	1.98

Table 10: Parallel numericals results for Problem 1, for $k_1 = 1000$ and $k_2 = 1$.

5.2.2 Problem 5

It is a test case mentionned in [26] and [14]. We consider a flower-like interface Σ:

$$\phi(r,\theta) = r - r_0 - 0.2\sin(\omega\theta).$$

with $r = \sqrt{(x-x_c)^2 + (y-y_c)^2}$, $\theta = arctan((y-y_c)/(x-x_c))$, $x_c = y_c = 0.2\sqrt{20}$. The coefficient k is defined by:

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

The exact solution is:

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

We study two cases: $\omega = 5$, $r_0 = 0.5$, $k^- = 1000$, and $\omega = 12$ and $r_0 = 0.4$, $k^- = 100$. On Figure 17 are presented numerical results for N varying from 270 to 3500, with an increment 10 from 270 to 1090, as in [26] and [14], then an increment 100 from 1100 to 3500. The numerical solution and error are plotted in Figures 18 and 19. We observe in both cases a second order convergence. More precisely, for $\omega = 5$ the linear regression slope is 2.06, and for $\omega = 12$ the slope is 2.64, but in the latter case the convergence rate is overestimated by the presence of oscillations for the smallest values of N. The overall accuracy is slightly better than CIM [14] for $\omega = 5$ and slightly worse for $\omega = 12$, the CIM being itself globally more accurate than the method in [26].

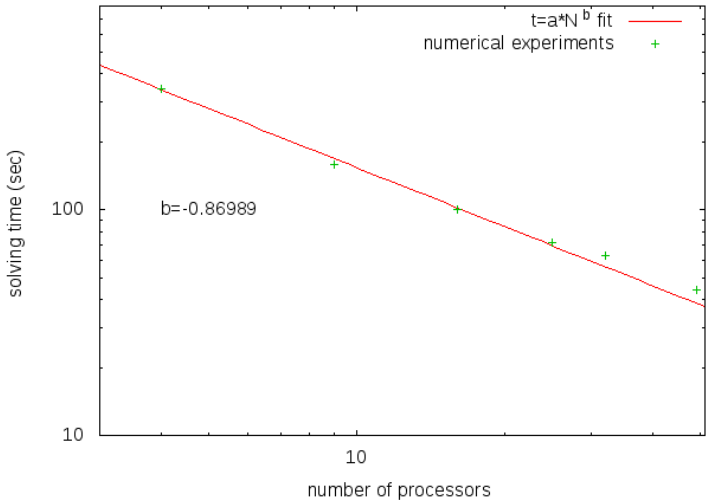

Figure 16: This figure shows how the calculation time scales with the number of processors. Crosses: experimental data. Line: least square fit of the data. The experiments have been conducted on the machine Fourmi at PlaFRIM (see, [2])

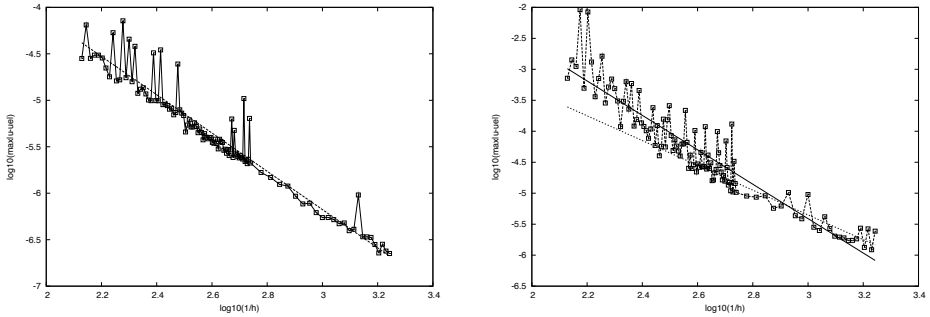

Figure 17: Convergence tests for Problem 5, with $\omega = 5$, $r_0 = 0.5$, $k^- = 1000$ (left), and $\omega = 12$ and $r_0 = 0.4$, $k^- = 100$ (right). Dashed line illustrates the slope of order two accuracy. Solid line is the slope of the linear regression.

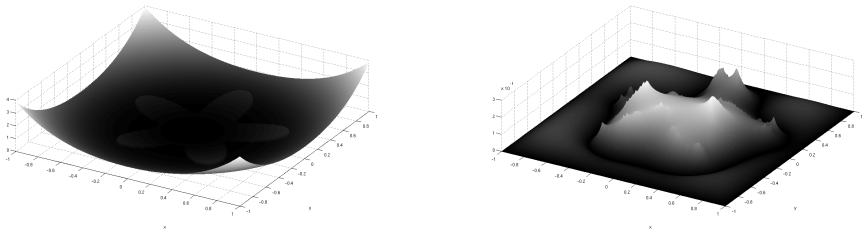

Figure 18: Numerical solution (left) and error (right) for $n_x = n_y = 270$ for Problem 5 for $\omega = 5$.

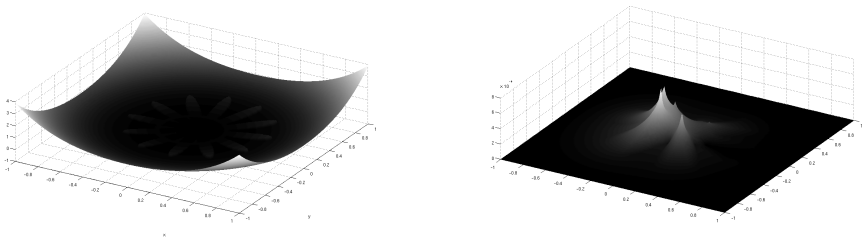

Figure 19: Numerical solution (left) and error (right) for $n_x = n_y = 270$ for Problem 5 for $\omega = 12$.

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 the interface unknowns, which are themselves determined by solving discretized flux transmissions at the interface. The use of interface unknowns allows to decouple the discretization of the elliptic operator from the treatment of the interface jump 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 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, even on complex interfaces, with an absolute error competitive with the other methods of the literature, and its parallel implementation shows good scalability properties.

Acknowledgments

The authors would like to thank Angelo Iollo for his helpful advices.

References

[1] The Message Passing Interface Forum Home Page. http://www.mpi-forum.org.

[2] Plateforme Fédérative pour la Recherche en Informatique et Mathématiques en Aquitaine. https://plafrim.bordeaux.inria.fr.

[3] P. Angot, C.-H. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible flows. *Numer. Math.*, 81(4):497–520, 1999.

[4] E. Arquis and J.P. Caltagirone. Sur les conditions hydrodynamiques au voisinage d'une interface milieu fluide-milieux poreux: application la convection naturelle. *C.R. Acad. Sci. Paris II*, 299:1–4, 1984.

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

[6] Satish Balay, Jed Brown, , Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[7] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2009. http://www.mcs.anl.gov/petsc.

[8] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[9] P. Bethelsen. A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions. *J. Comput. Phys.*, 197:364–386, 2004.

[10] 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.

[11] D. Bresch, T. Colin, E. Grenier, B. Ribba, and O. Saut. Computational modeling of solid tumor growth: the avascular stage, 2009.

[12] F. Buret, N. Faure, L. Nicolas, R. Perussel, and C. Poignard. Numerical studies on the effect of electric pulses on an egg-shaped cell with a spherical nucleus. Technical Report 7270, INRIA, 2010.

[13] Z. Chen and J. Zou. Finite element methods and their convergence for elliptic and parabolic interface problems,. *Numer. Math.*, 79:175–202, 1998.

[14] I. Chern and Y.-C. Shu. A coupling interface method for elliptic interface problems. *J. Comput. Phys.*, 225:2138–2174, 2007.

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

[16] R. P. Fedkiw. Coupling an eulerian fluid calculation to a lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys.*, 175:200–224, 2002.

[17] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492, 1999.

[18] F. Gibou, R. P. Fedkiw, L.T. Cheng, and M. Kang. A second order accurate symmetric discretization of the poisson equation on irregular domains. *J. Comput. Phys.*, 176:205–227, 2002.

[19] F. Gibou and R.P. Fedkiw. A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem. *J. Comput. Phys.*,

202:577–601, 2005.

[20] B. Gustafsson. A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem. *SIAM Journal of Numerical Analysis*, 39:396–406, 1975.

[21] B. Gustafsson. The convergence rate for difference approximations to general initial boundary value problems. *SIAM Journal of Numerical Analysis*, 18:179–190, 1981.

[22] J. Huang and J. Zou. A mortar element method for elliptic problems with discontinuous coefficients. *A mortar element method for elliptic problems with discontinuous coefficients*, 22:549–576, 2002.

[23] J.-S. Huh and J.A. Sethian. Exact subgrid interface correction schemes for elliptic interface problems. *Proceedings of the National Academy of Sciences of the United States of America*, 105:9874, 2008.

[24] H. Johansen and P. Colella. A cartesian grid embedded boundary method for poisson's equation on irregular domains. *J. Comput. Phys.*, 147:60–85, 1998.

[25] R. J. Leveque and L.Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Numerical Analysis*, 31(4):1019–1044, 1994.

[26] Z.L. Li. A fast iterative algorithm for elliptic interface problems. *SIAM J. Numer. Anal.*, 35:230–254, 1998.

[27] Z.L. Li and K. Ito. Maximum principle preserving schemes for interfacxe problems with discontinuous coefficients. *SIAM J. Sci. Comput.*, 23:339–361, 2001.

[28] X.-D. Liu, R. P. Fedkiw, and M. Kang. A boundary capturing method for poisson's equation on irregular domains. *J. Comput. Phys.*, 160:151–178, 2000.

[29] B. Maury. A fat boundary method for the poisson problem in a domain with holes. *Journal of Scientific Computing*, 16:319–339, 2001.

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

[31] A. Mayo. The rapid evaluation of volume integrals of potential theory on general regions. *J. Comput. Phys.*, 100:236–245, 1992.

[32] A. Mayo and A. Greenbaum. Fast parallel iterative solution of poisson's and the biharmonic equations on irregular regions. *SIAM J. Sci. Stat. Comput.*, 13:101–118, 1992.

[33] P. McCorquodale, P. Collela, and H. Johansen. A cartesian grid embedded boundary method for the heat equation on irregular domains. *J. Comput. Phys.*, 173:620–635, 2001.

[34] M. Oevermann, C. Scharfenberg, and R. Klein. A sharp interface finite volume method for elliptic equations on cartesian grids. *J. Comput. Phys.*, 228:5184–5206, 2009.

[35] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003.

[36] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamiltonjacobi formulations. *J. Comput. Phys.*, 79(12), 1988.

[37] Y. Saad. Sparskit a basic tool-kit for sparse matrix computations. http://www-users.cs.umn.edu/ saad/software/SPARSKIT/sparskit.html.

[38] A. Sarthou, S. Vincent, P. Angot, and J.P. Caltagirone. The algebraic immersed interface and boundary method for elliptic equations with discontinuous coefficients. submitted, 2009.

[39] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, UK, 1999.

[40] J. A. Sethian. Evolution, implementation, and application of level set and fast marching methods for advancing fronts. *J. Comput. Phys.*, 169:503–555, 2001.

[41] M. Svard and J. Nordstrom. On the order of accuracy for difference approximations of initial-boundary value problems. *J. Comput. Phys.*, 218:333–352, 2006.

[42] A. Wiegmann and K. Bube. The explicit jump immersed interface method: finite difference method for pdes with piecewise smooth solutions. *SIAM J. Numer. Anal.*, 37(3):827–862, 2000.
[43] S. a Yu and G.W. Wei. Three-dimensional matched interface and boundary (mib) method for treating geometric singularities. *J. Comput. Phys.*, 227:602–632, 2007.
[44] X. Zhong. A new high-order immersed interface method for solving elliptic equations with imbedded interface of discontinuity. *J. Comput. Phys.*, 225:1066–1099, 2007.
[45] Y. C. Zhou and G. W. Wei. On the fictitious-domain and interpolation formulations of the matched interface and boundary (mib) method. *J. Comput. Phys.*, 219:228–246, 2006.
[46] Y. C. Zhou, S. Zhao, M. Feig, and G. W. Wei. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *J. Comput. Phys.*, 213:1–30, 2006.