Scientific Research Publishing

Open Journal of Fluid Dynamics, 2020, 10, 239-269
https://www.scirp.org/journal/ojfd
ISSN Online: 2165-3860
ISSN Print: 2165-3852

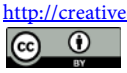
The Algebraic Immersed Interface and Boundary Method for Elliptic Equations with Jump Conditions

Arthur Sarthou[1*], Stéphane Vincent[2], Philippe Angot[3], Jean-Paul Caltagirone[4]

[1]Buf Compagnie, Paris, France
[2]Laboratoire MSME, Université Gustave Eiffel, Champs-sur-Marne, France
[3]Institut de Mathématiques de Marseille, Centrale Marseille, Marseille, France
[4]Institut Polytechnique de Bordeaux, Université de Bordeaux, Talence, France
Email: *arthur.sarthou@gmail.com

How to cite this paper: Sarthou, A., Vincent, S., Angot, P. and Caltagirone, J.-P. (2020) The Algebraic Immersed Interface and Boundary Method for Elliptic Equations with Jump Conditions. *Open Journal of Fluid Dynamics*, **10**, 239-269.
https://doi.org/10.4236/ojfd.2020.103015

Received: April 23, 2020
Accepted: August 16, 2020
Published: August 19, 2020



Abstract

A new simple fictitious domain method, the algebraic immersed interface and boundary (AIIB) method, is presented for elliptic equations with immersed interface conditions. This method allows jump conditions on immersed interfaces to be discretized with a good accuracy on a compact stencil. Auxiliary unknowns are created at existing grid locations to increase the degrees of freedom of the initial problem. These auxiliary unknowns allow imposing various constraints to the system on interfaces of complex shapes. For instance, the method is able to deal with immersed interfaces for elliptic equations with jump conditions on the solution or discontinuous coefficients with a second order of spatial accuracy. As the AIIB method acts on an algebraic level and only changes the problem matrix, no particular attention to the initial discretization is required. The method can be easily implemented in any structured grid code and can deal with immersed boundary problems too. Several validation problems are presented to demonstrate the interest and accuracy of the method.

Keywords

Fictitious Domain, Immersed Interface Method, Immersed Boundary Method, Penalty Methods, Finite Volumes, Elliptic Equations, Jump Embedded Conditions

1. Introduction and General Motivations

Simulating flows and heat transfer interacting with complex objects on Cartesian

structured grids requires an efficient coupling between such grids and the corresponding numerical methods and complex shape interfaces. Such a coupling is often performed thanks to fictitious domain methods, where the computational domain does not match the physical domain. The advantages of this approach are numerous. A second-order accurate discretization of the spatial operators is simple to obtain; grid generation is trivial, and furthermore there is no need to remesh the discretization grid in the case of moving or deformable boundaries. Concerning this last point, fictitious domain methods can be useful even on unstructured grids: Eulerian fixed unstructured grids can fit immobile obstacles, (e.g. a stator of an aircraft motor) while mobile objects (a rotor) are treated with fictitious domain methods. Two particular classes of problems can be drawn: the immersed boundary problems and the immersed interface problems. The first deals with complex boundaries, such as flow past objects, where no attention has to be paid to the solution inside the obstacles. The immersed interface problems consider subdomains delimited by interfaces, and the solution is required in both sides of the interface. As particular conditions, such as jump conditions, can be required on the interface, this second class of problems is often more difficult to treat.

Let us consider a domain of interest Ω_0, typically a fluid domain, which is embedded inside a computational domain of simple shape $\Omega \subset \mathbb{R}^d$, d being the spatial dimension of the problem. The auxiliary domain Ω_1, typically a solid particle or an obstacle, is such that as shown in Figure 1: $\Omega = \Omega_0 \bigcup \Sigma \bigcup \Omega_1$ where Σ is an immersed interface, and $\boldsymbol{n}$ is the unit outward normal vector to Ω_0 on Σ. Let us consider the following model scalar immersed boundary problem in Ω_0 with a Dirichlet boundary condition (BC) on the interface Σ:

$$\mathcal{P}_b \quad \begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega_0 \\ u_{|\Sigma} = u_D & \text{on } \Sigma \end{cases}$$

A boundary condition is also required on the other part of the boundary $\partial\Omega_0$ so that the whole problem is well-posed.

A first approach dealing with immersed boundaries is the distributed Lagrange Multiplier method proposed by Glowinski *et al.* [1]. Lagrange multipliers are introduced into the weak formulation of the initial elliptic equation to ensure the immersed boundary condition.

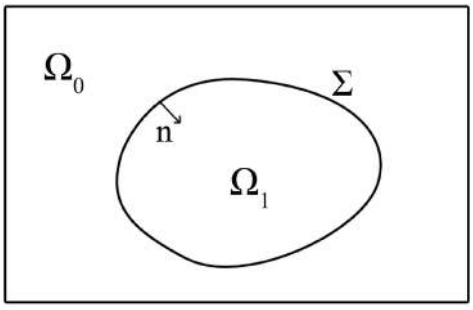


Figure 1. Definition of the subdomains and the interface.

Cartesian grid [2] [3] and Cut-cell [4] methods use a structured grid in the whole domain except near obstacles where unstructured cells are created from structured cells. These methods are hard to implement due to the numerous different space configurations of the intersections between cells and objects. Furthermore, the existence of small cells can induce solver troubles.

The immersed boundary method (IBM) was initially presented by Peskin [5] [6]. Fictitious boundaries are taken into account through a singular source term defined only near the boundaries. As the source term is weighted with a discrete Dirac function smoothed on a non-zero support, the interface influence is spread over some grid cells. This method is first-order in space and explicit. Another class of IBM, the direct-forcing (DF) method, was initially proposed by Mohd-Yusof [7]. The idea here is to impose a no-slip condition directly on the boundary using a mirrored flow over the boundary. In [8] [9], the correct boundary velocity is obtained by interpolating the solution on the boundary and far from the boundary on grid points in the near vicinity of the interface. In [10], Tseng *et al.* use the same principle but extrapolate the solution in ghost cells outside the domain. This approach can be seen as a generalization of the mirror boundary conditions used in Cartesian staggered grids to impose a velocity Dirichlet condition on pressure nodes. As discussed in [11], this kind of approach seems to be more accurate than [8] [9].

The penalty methods for fictitious domains consist in adding specific terms in the conservation equations to play with the order of magnitude of existing physical contributions so as to obtain at the same time and with the same set of equations two different physical properties. The volume penalty method (VPM) [12] requires the addition of a penalty term $\frac{\chi}{\varepsilon}\left(u-u_D\right)$ in the conservation equations, such that:

$$\begin{cases} -\nabla\cdot\left(a\nabla u\right)+\dfrac{\chi}{\varepsilon}\left(u-u_D\right)=f \text{ in } \Omega \\ \text{with } \chi_{\left|\Omega_0\right.}=0, \chi_{\left|\Omega_1\right.}=1, \text{ for } 0<\varepsilon\ll 1 \end{cases} \tag{1}$$

where ε denotes the penalty parameter which tends to 0. Hence, in Ω_1 the original equation becomes negligible and $u=u_D$ is imposed. In [13] [14] authors add a Darcy term $\frac{\mu}{K}\boldsymbol{u}$ to the Navier-Stokes (NS) equations where μ is the dynamic viscosity and K the permeability. In the fluid medium, $K\to\infty$ so the Darcy term is then negligible and the original set of NS equations is retrieved. In the solid medium, $K\to 0$ and consequently the NS equations tend to $\boldsymbol{u}=0$. Classical discretizations of the penalty terms are of first order only since they consider the projected shape of the interface on the Eulerian grid to define the penalty parameters [15]. In [16] [17], Sarthou *et al.* have discretized the volume penalty term with a second order using implicit interpolations as in [10]. This method is called the sub-mesh penalty (SMP) method and has been applied to both elliptic and NS equations.

Applied to problem $\mathcal{P}_b$, the ghost cell immersed boundary method [10] and SMP method [16] used the cells in Ω_1 in the vicinity of the interface to enhance the accuracy of the solution in Ω_0.

Another approach which considers the extension of the solution is considered in [18] [19] by Gibou and Fedkiw. Ghost nodes and simple interpolations are considered, but contrary to the SMP and the IBM-DF methods, only 1D interpolations are used and the operators are rediscretized "by-hand".

Let us now consider a model immersed interface problem with jump interface conditions:

$$(\mathcal{P}_i)\quad \begin{cases} -\nabla\cdot(a\nabla u) = f & \text{in } \Omega \\ [\![u]\!]_\Sigma = \varphi & \text{on } \Sigma \\ [\![(a\cdot\nabla u)\cdot \boldsymbol{n}]\!]_\Sigma = \psi & \text{on } \Sigma \end{cases}$$

A first class of method is the immersed interface methods (IIM) initially introduced by LeVeque and Li [20] and widely described in [21]. This group of methods uses Taylor series expansion of the solution at discretization points in the vicinity of Σ to modify the discrete operators at these points. Much work has been devoted to the immersed interface method and its numerous applications, such as moving interfaces [22] or Navier-Stokes equations [23]. In [24], Li uses an augmented variables approach. Additional variables and interface equations are added to the initial linear system. The new variables are the values of jumps at some interface points. This method has been extended to the incompressible Stokes [25] and Navier-Stokes [26].

The Ghost Fluid Method, originally developed by Fedkiw *et al.* [27] [28], introduces ghost nodes where the solution is extended from one side of the interface to the other side. As for IIM, the operator discretization must be modified "by-hand". Zhou *et al.* overcome this drawback with the matched interface and boundary (MIB) method [29] [30] [31] by using interface conditions to express the solution at ghost nodes with respect to the solution on physical nodes. Hence, the discretization is automatically performed whatever the discretization scheme. Contrary to [24], the additional equations for these two last methods are not written at "random" points of the interface but at the intersections between the Eulerian grid and the immersed interface. Furthermore, simple Lagrange polynomials are used whereas a more complicated weighted least squares approach is used in [24] to discretize additional equations. In [32], Cisternino and Weynans propose a quite simple method with additional unknowns located at the interface. Interfaces conditions are discretized at these points and are added to the final linear system.

The method presented in this work solves elliptic problems using an augmented method coupled with an auxiliary unknown approach. Its main contribution is its simplicity that eases its implementation, and its generality that allows the method to be applied to various boundary and interface conditions. Contrary to ghost nodes, auxiliary unknowns are present in the linear system and the discretization of the interface conditions is independant from the discre-

tization of the initial equation.

Compact interpolations are used to discretize the additional interface constraints. The method is simple to implement even for interfaces of complex shapes, *i.e.* not described by analytical equations. Except for the discretization of interface conditions, all operations are automatically performed with algebraic modification or directly by the "black-box" matrix solver. This new method is called the algebraic immersed interface and boundary (AIIB) method. In Section 2, the method is presented for immersed boundary problems. Then, the method is extended to immersed interface problems with known solution on the interface. Finally, the method is applied to immersed interfaces with transmission and jump conditions. A special attention is paid to the management of the discretized interface, especially the way to project it onto the Eulerian grid using a fast ray-casting method. In Section 3, validation tests and convergence studies are presented. Conclusions and perspectives are finally drawn in Section 4.

2. The Algebraic Immersed Interface and Boundary Method

The AIIB method is now presented. The method is first formulated for immersed boundary problems when a Dirichlet or a Neumann boundary condition is required. The method is then extended to simple immersed interface problems where the solution is *a priori* known on the interface. Finally, an extension to jump and transmission conditions is described.

2.1. Definitions and Notations

Our objective is to numerically impose the adequate conditions on the interface Σ for problems similar to $\mathcal{P}_b$ and $\mathcal{P}_i$. These conditions will be discretized in space on an Eulerian structured mesh covering Ω. As the discretization of the interface or boundary conditions requires interpolations, the following interpolations in 2D: $\mathbb{P}_1^2(x,y) = p_1 + p_2 x + p_3 y$ and $\mathbb{Q}_1^2(x,y) = p_1 + p_2 x + p_3 y + p_4 xy$ are used. In 3D, we use $\mathbb{P}_1^3(x,y,z) = p_1 + p_2 x + p_3 y + p_4 z$ and $\mathbb{Q}_1^3(x,y,z) = p_1 + p_2 x + p_3 y + p_4 z + p_5 xy + p_6 yz + p_7 zx + p_8 xyz$. An additional interpolation, $\mathbb{L}_1^1(x) = p_1 + p_2 x$, is also possible to be chosen for 2D and 3D problems. The superscript is the dimension of the interpolation while the subscript is the order of spatial accuracy.

The computational domain Ω is approximated with a curvilinear mesh T_h composed of $N \times M$ ($\times L$ in 3D) cell-centered finite volumes ($\mathcal{V}_I$) for $I \in \mathcal{E}$, $\mathcal{E}$ being the set of indexes of the Eulerian orthogonal curvilinear structured mesh. Let x_I be the vector coordinates of the center of each volume $\mathcal{V}_I$. In 2D, the horizontal and vertical mesh steps are respectively h_x and h_y. This grid is used to discretize the conservation equations. A dual grid is introduced for the management of the AIIB method. The grid lines of this dual cell-vertex mesh are defined by the network of the cell centers x_I. The volumes of the dual mesh are denoted by ($\mathcal{V}_I'$). The Eulerian unknowns are noted u_I which are the approximated values of $u(x_I)$, *i.e.* the solution at the cell centers x_I.

The discrete interface Σ_h, hereafter called the Lagrangian mesh, is given by a discretization of the original interface Σ. It is described by a piecewise linear approximation of Σ: $\Sigma_h = \{\sigma_l \subset \mathbb{R}^{d-1}, l \in \mathcal{L}_f\}$, $\mathcal{L}_f$ being the set of indexes of the Lagrangian mesh and K being the cardinal of $\mathcal{L}_f$. Typically, σ_l are segments in 2D and triangles in 3D. The vertices of each face σ_l are denoted by $x_{l,i}$ for $i = 1, \cdots, d$ and the set of all vertices is $\{x_l, l \in \mathcal{L}_v\}$, $\mathcal{L}_v$ being the set of indexes of the vertices. The intersection points between the grid lines of the Eulerian dual mesh and the faces σ_l of the Lagrangian mesh are denoted by $\{x_i, i \in \mathcal{I}\}$ (see Figure 2). Our objective is to discretize Dirichlet, Neumann, transmission and jump conditions at these interface points to build a general fictitious domain approach. This method is expected to reach a global second-order spatial accuracy.

We shall use the following Eulerian volume functions in order to implicitly locate Σ_h:

- The Heaviside function χ, defined as:

$$\chi(x) = \begin{cases} 1 & \text{if } x \in \Omega_1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

This function is built with a point in solid method presented below. The function χ will be used to perform fictitious domain algorithms and to build a level-set function.

- The level-set function ϕ, with:

$$\phi(x) = \begin{cases} -\text{dist}_\Sigma(x) & \text{if } x \in \Omega_1 \\ \text{dist}_\Sigma(x) & \text{otherwise} \end{cases} \tag{3}$$

and $\text{dist}_\Sigma(p) = \inf_{x \in \Sigma} \|x - p\|$. The unsigned distance is computed geometrically. The sign is directly obtained with the discrete Heaviside function χ.

- The colour phase functions C, which is the ratio of a given phase in a control volume. We denote $C(x_I)$ the phase ratio in the control volume centered in x_I. This function is approximated from the ϕ function by using the formula proposed by Sussman and Fatemi [33]:

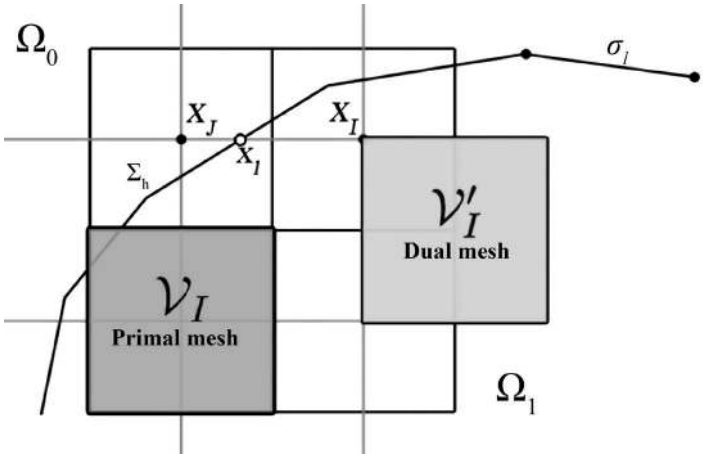


Figure 2. Definition of the discretization kernels for the AIIB method.

$$C(x) \approx \begin{cases} 1 & \text{if } \phi(x) > h \\ 0 & \text{if } \phi(x) < -h \\ \frac{1}{2}\left(1 + \frac{\phi(x)}{h} + \frac{1}{\pi}\sin\left(\pi\phi(x)/h\right)\right) & \text{otherwise} \end{cases} \quad (4)$$

New sets of Eulerian points x_I are defined near the interface so that each one has a neighbor x_J verifying $\chi_J \neq \chi_I$ (with $\chi_I = \chi(x_I)$ and $\chi_J = \chi(x_J)$), *i.e.* the segment $[x_I; x_J]$ is cut by Σ_h. These Eulerian "interface" points are also sorted according to their location inside Ω_0 or Ω_1. Two sets $\{x_I, I \in \mathcal{N}_0\}$ and $\{x_I, I \in \mathcal{N}_1\}$ are thus obtained, where
$\mathcal{N}_0 = \{I, x_I \in \Omega_0, x_I \in Neighb(x_J), \chi_I \neq \chi_J, x_J \in \Omega_1\}$ and
$\mathcal{N}_1 = \{I, x_I \in \Omega_1, x_J \in Neighb(x_I), \chi_I \neq \chi_J, x_J \in \Omega_0\}$.

For each x_I, $I \in \mathcal{N}_0$ or $I \in \mathcal{N}_1$, we associate two unknowns: the *physical* one denoted as u_I and the *auxiliary* one u_I^*.

2.2. Projection of the Lagrangian Shape on the Eulerian Grid

The generation of the Lagrangian mesh of the interface is achieved using a computer graphics software. Specific algorithms have been developed to project this Lagrangian grid onto the Eulerian physical grid. In order to obtain the discrete Heaviside function χ, one has to determine which Eulerian points are inside the domain Ω_1 defined by a Lagrangian surface. Such a surface must be closed and not self-intersecting. In [11] [14], the authors used a global methodology partly based on [34] where χ is obtained thanks to a PDE. This method suffers from a lack of accuracy and robustness. A Ray-casting method based on the Jordan curve theorem is more adapted and is used in the present work. The principle is to cast a ray from each Eulerian point to infinity and to test the number of intersections between the ray and the Lagrangian mesh. If the number of intersections is odd, the Eulerian point is inside the object, and outside otherwise. The Ray-casting method can be enhanced by classifying elements of the Lagrangian mesh with an octree sub-structure which recursively subdivides the space in boxes. If a ray does not intersect a box, it does not intersect the triangles inside the box. A fast and simple optimization is to test if a given point is in the box bounding the Lagrangian mesh. An improvement of the Ray-casting algorithm, the Thread Ray-casting, can be found in [35] [36]. Instead of casting rays in any direction, rays are all aligned on a principal direction of the grid such as each ray passes through a line of points x_I. Hence, only one ray is cast for each line of points allowing the complexity of the algorithm to be reduced by one order.

The Lagrangian points x_l of Σ_h, $l \in \mathcal{I}$ are required to couple the Lagrangian surface and the Eulerian grid used to solve the conservation equations. These points can be obtained with two methods. A geometrical computation of the intersections gives the most accurate result. If not optimized the computational cost of this method is not always negligible for some cases.

Using the Level-set function is a faster but less accurate way to obtain the in-

tersection points. Let us consider two Eulerian points $x_I \in \Omega_0$ and $x_J \in \Omega_1$. We denote by $d_I = d(x_I, \Sigma_h)$ and $d_J = d(x_J, \Sigma_h)$ the unsigned distances between Eulerian points and the interface Σ_h. Then, $x_I = (x_I d_J + x_J d_I)/(d_I + d_J)$.

Algorithmic problems can be encountered if the Lagrangian mesh is too complex compared to the Eulerian mesh. In some cases, more than one segment of the Lagrangian mesh can intersect a same grid segment $[x_I; x_J]$, and more than one intersecting points can then be found between x_I and x_J with the geometric method. In this case, only one intersecting point is considered.

Concerning the use of the Level-set, this function is a projection of the shape on a discrete grid. The local curvature of the projected shape is thus limited by the accuracy of the Eulerian grid. Consequently, no more than one intersecting point can be found between x_I and x_J with the Level-set.

2.3. The AIIB for Dirichlet and Neumann Conditions

2.3.1. General Principle

Once the shape informations are available on the Eulerian grid, the problem discretization has to be modified to take into account the fictitious domain delimited by an immersed boundary or an immersed interface. The sub-mesh penalty (SMP) method [11] [16] was originally designed to treat immersed boundary problems. It could be extended to treat immersed interface problems by symmetrization of the algorithm with introduction of auxiliary unknowns as in the AIIB method. This new method is an enhancement of the SMP method which is also able to solve immersed interface problems. The main idea of the AIIB method is to embed an interface into a given domain by modifying the final matrix only. As no modification of the discretization of the operators is required (contrary to [18] [19] and the immersed interface methods [20]), the AIIB method is thus simple to implement.

Let $\mathcal{P}$ be a model problem discretized in the whole domain Ω as $Au = b$ where A is a square matrix of order m, u the solution vector and b a source term. The basic idea of the AIIB method is to add new unknowns and equations to the initial linear system so as to take into account additional interface constraints. The new unknowns, so-called the auxiliary or fictitious unknowns and labeled with *, are defined as being the extrapolation of the solution from one side of the interface to the other, and are used to discretize the interface conditions. Hence, the original problem $Au = b$ becomes $A'u' = b'$, with A' a square matrix of order $m+n$, with n the number of auxiliary constraints related to the interface conditions. The solution u' is decomposed such as $u' = (u, u^*)^{\mathrm{T}}$ and the source term as $b' = (b, b^*)^{\mathrm{T}}$. The interface constraints are discretized with a $(n, m+n)$ block matrix C and the source term b^*. The discretizations of the constraints for various interface and boundary conditions are given in Sections 2.3.2 and 2.4.

According to the interface conditions, the regularity of the solution on the interface is often lower than in the rest of the domain. Hence, the discretization of operators with a stencil cutting the interface can induce a great loss of accuracy.

The first idea is to consider unknowns $u_I^*, I \in \mathcal{N}_1$ (resp. $u_I^*, I \in \mathcal{N}_0$) as the extension of the solution in Ω_0 (resp. Ω_1). The initial algebraic link between unknowns from both sides of the interface is cut, and the new link over the interface is obtained thanks to auxiliary unknowns. Practically, matrix coefficients must be modified to take into account the new connectivities. Let $\alpha_{I,J}$ be a coefficient of A at row I, column J and $\alpha'_{I,J}$ the new coefficient in A'. If $I \in \mathcal{N}_0$ and $J \in \mathcal{N}_1$, $\alpha'_{I,J} = 0$.

Hence, there is no more direct link between the initial unknowns in Ω_0 and Ω_1. However, the initial value of $\alpha_{I,J}$ is reused to express the new relation between the original unknown u_I and its new neighbor u_J^*. Consequently, $\alpha'_{I,J^*} = \alpha_{I,J}$, where J^* is the index corresponding to u_J^*.

This is exactly the way how we proceed for the practical algorithm. However, this modification can be expressed algebraically with permutation and mask matrices as follows.

We define the two following mask matrices I_1 of dimensions $(m, m+n)$ and I_2 of size $(n, m+n)$:

$$I_1 = \begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & & \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 & & \ddots & \vdots \\ 0 & \ldots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \tag{5}$$

$$I_2 = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & & 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{6}$$

The matrices A_0 and A_1 are defined such as $A_0 + A_1 = A$, $A_0(I,J) = A(I,J)$ if $I \in \mathcal{N}_0$, else $A_0(I,J) = 0$. Similarly $A_1(I,J) = A(I,J)$ if $I \in \mathcal{N}_1$ else $A_1(I,J) = 0$. Finally, the connectivities are changed using the permutation matrices P_0 and P_1: P_0 is defined to switch row I with row J if $I \in \mathcal{N}_0$, $J \in \mathcal{N}_1$ and P_1 to switch row I with row J if $I \in \mathcal{N}_1$, $J \in \mathcal{N}_0$. Hence, the new problem matrix is now defined by:

$$A' = I_1^{\mathrm{T}} \left(P_0 \left(A_0 I_1 \right) + P_1 \left(A_1 I_1 \right) \right) + I_2^{\mathrm{T}} C \tag{7}$$

The new problem is $A'u' = b'$ with A' written with 4 blocks of various sizes: $\tilde{A}(m,m)$, $B(m,n)$, $C_1(n,m)$, $C_2(n,n)$. The matrix $\tilde{A}$ is thus the modification of the initial matrix A by setting to zero the coefficient $\alpha_{I,J}$ if $\chi(x_I) \neq \chi(x_J)$, and C_1 and C_2 are the two sub-matrices of the matrix C. The problem can be written as:

$$\begin{pmatrix} \tilde{A} & B \\ C_1 & C_2 \end{pmatrix} \begin{pmatrix} u \\ u^* \end{pmatrix} = \begin{pmatrix} b \\ b^* \end{pmatrix} \tag{8}$$

The entire problem can then be solved to obtain $u' = \left(u, u^*\right)^{\mathrm{T}}$. However, u^* being the auxiliary solution is not required to be computed explicitly. Hence, the Schur complement method can be used to calculate the solution for the physical

unknowns only. The final problem is now:

$$\left(\tilde{A} - BC_2^{-1}C_1\right)u = b - BC_2^{-1}b^* \tag{9}$$

The opportunity of such a reduction will be discussed later.

2.3.2. AIIB Algorithm for a Scalar Equation with Dirichlet Boundary Conditions

For the sake of clarity, let us first describe in $2D$ the AIIB method for the model scalar problem $\mathcal{P}_b$ with a Dirichlet boundary condition on the interface Σ. For this version of the AIIB algorithm, Ω_0 is the domain of interest and auxiliary unknowns are created in Ω_1 only. Let us consider a point $x_I, I \in \mathcal{N}_1$. At location x_I, two unknowns coexist: a physical one u_I and an auxiliary one u_I^*. We first describe the case when x_I has only one neighbor x_J in Ω_0. The Lagrangian point x_l is the intersection between $[x_I; x_J]$ and Σ_h (**Figure 3** right). Then, the solution $u_l = u_D(x_l)$ at the interface is approximated by the $\mathbb{P}_1^1$ interpolation between the Eulerian unknowns u_I^* and u_J:

$$u_l = \alpha_I u_I^* + \alpha_J u_J \text{ with } 0 < \alpha_I, \alpha_J < 1 \text{ and } \alpha_I + \alpha_J = 1 \tag{10}$$

To our knowledge, it has been verified that for schemes close to the one presented here (including ghost-like schemes [19] and IBM-Direct Forcing schemes [10]) only a linear interpolation is required to reach a second order of accuracy as a Dirichlet *BC* is imposed here. It has been theoretically demonstrated in [37] that under the assumption of pointwise stability of the scheme the discretization at the boundary can be two orders less accurate than in the rest of the domain without reducing the global convergence order. It is important to notice this does not apply for immersed interface conditions [32].

If now x_I has a second neighbor x_K in Ω_0, the intersection x_m between $[x_I; x_K]$ and Σ_h is considered with $u_m = u_D(x_m)$. We choose x_p, a new point of Σ_h between x_l and x_m (see **Figure 3** left). The solution $u_p = u_D(x_p)$ is then imposed using a $\mathbb{P}_1^2$ -interpolation of the values u_I^*, u_J and u_K:

$$u_p = \alpha_I u_I^* + \alpha_J u_J + \alpha_K u_K,\ 0 < \alpha_I, \alpha_J, \alpha_K < 1,\ \alpha_I + \alpha_J + \alpha_K = 1 \tag{11}$$

A $\mathbb{Q}_1^2$ interpolation of u_I, u_J, u_K and u_L can be also used by extending the interpolation stencil with the point x_L which is the fourth point of the cell of the dual mesh defined by x_I, x_J and x_K (see **Figure 3** left). As a third

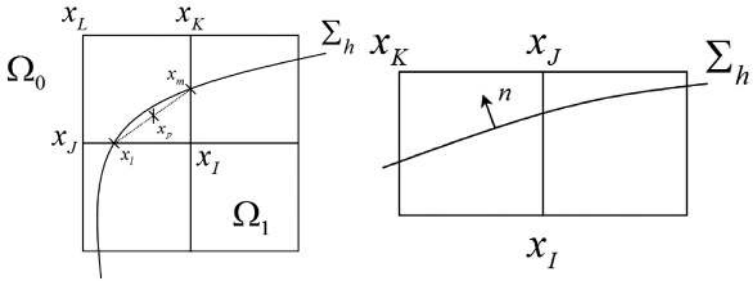


Figure 3. Example of selection of points for Dirichlet (a) and Neumann (b) constraints.

choice, two independent $\mathbb{L}_1^2$ linear 1D interpolations can be used (one for each direction) for an almost equivalent result. It produces:

$$\begin{cases} u_l = \alpha_I u_I^* + \alpha_J u_J \text{ with } 0 < \alpha_I, \alpha_J < 1 \text{ and } \alpha_I + \alpha_J = 1 \\ u_m = \alpha_I' u_I^{*'} + \alpha_K u_K \text{ with } 0 < \alpha_I', \alpha_K < 1 \text{ and } \alpha_I' + \alpha_J = 1 \end{cases} \tag{12}$$

In this case, two auxiliary unknowns are created.

A simple choice for x_p is the barycenter between x_l and x_m where $u_p = (u_l + u_m)/2$. A summation of the two constraints from (12) gives:

$$\alpha_I u_I^* + \alpha_J u_J + \alpha_{I'} u_I^* + \alpha_K u_K = u_l + u_m \tag{13}$$

which is equivalent to build a constraint imposing u_p at x_p with a $\mathbb{P}_1^2$ interpolation:

$$\begin{gathered} \frac{(\alpha_I + \alpha_I')u_I^* + \alpha_J u_J + \alpha_K u_K}{2} = u_p, \\ \text{with } 0 < \frac{\alpha_I + \alpha_{I'}}{2}, \frac{\alpha_J}{2}, \frac{\alpha_K}{2} < 1, \frac{\alpha_I + \alpha_{I'}}{2} + \frac{\alpha_J}{2} + \frac{\alpha_K}{2} = 1 \end{gathered} \tag{14}$$

Hence, an easy general implementation consists in summing the constraints corresponding to each direction, no matter the number of neighbors of x_I. If the elements σ_l of Σ_h used to define x_l and x_m are not the same, the barycenter x_p of these two points is not necessarily on Σ_h, especially for interfaces of strong curvature. However, the distance $d(x_p, \Sigma_h)$ between x_p and Σ_h varies like $\mathcal{O}(h^2)$ and so this additional error does not spoil the second-order precision of our discretization. The convergence of this additional error is numerically tested in Section (3.3.1). If the curvature of Σ_h is small enough relatively to the Eulerian mesh, *i.e.* if the Eulerian mesh is sufficiently fine, x_I almost never has a third or a fourth neighbor in Ω_0. However, if this case appears, a simple constraint $u_I^* = u_B$ is used with u_B being an average of u_D at the neighbor intersection points. In any case, by decreasing the Eulerian mesh step *h*, the number of points x_I having more than two neighbors in Ω_0 also decreases.

Concerning the C submatrix, it is filled with the coefficients α. One line corresponds to the discretization of the boundary constraint related to one auxiliary node. Each value of the second member b' is the boundary value u_l related on auxiliary point.

Hence, the present method is suitable to impose a Dirichlet boundary condition on Σ for Ω_0, when the solution in Ω_1 is of no interest. The solution u_I^* for $I \in \mathcal{N}_1$ is an extrapolation of the solution in Ω_0 in order to satisfy the boundary condition on Σ and thus is non-physical. Hence, the solution at the nodes of Ω_1 far from the interface does not impact on the solution in Ω_0 as the unknowns u_I for $I \in \mathcal{N}_1$ have no more algebraic relations with the nodes u_I for $I \in \mathcal{N}_0$. Nevertheless, if nothing specific is performed, a solution with no interest is computed by default in Ω_1. A prescribed solution inside Ω_1 can be obtained easily with a volume penalty method such as VPM [13] where initial equations are penalized to impose a desired analytical solution. However, a de-

sired solution in Ω_1 can also be obtained by modifying the solution obtained by solver. Then, the computational cost of the approach can be reduced by switching the solving of $u_I, x_I \in \Omega_1$ off, or by totally removing these nodes in the solving matrix.

2.3.3. Symmetric Version for Dirichlet Interface Conditions

The next step is to allow for multiple Dirichlet boundary conditions on both sides of the immersed interface. Thin objects could be treated with this approach. The problem is now:

$$\begin{cases} -\nabla\cdot\left(a\nabla u\right) = f & \text{in } \Omega \\ u_{|\Sigma}^{-} = u_D & \text{on } \Sigma \\ u_{|\Sigma}^{+} = u_G & \text{on } \Sigma \end{cases} \tag{15}$$

The problem (15) requires for each point x_I a physical unknown u_I as well as an auxiliary unknown u_I^* on both sides of the interface.

Practically, the algebraic modification part of the AIIB algorithm for a Dirichlet *BC* is applied a first time with Ω_0 as domain of interest, and auxiliary unknowns are created near Σ_h in Ω_1. As a second step, the Heaviside function is modified as $\chi := 1-\chi$ and the algorithm is applied a second time. Now, Ω_1 is the domain of interest and auxiliary unknowns are created near Σ in Ω_0.

Then, both problems are solved at the same time with the resolution of the linear system.

2.3.4. AIIB Algorithm for a Scalar Equation with Neumann Boundary Conditions

Let us now consider the following model scalar problem with a Neumann *BC* on the interface Σ:

$$\begin{cases} -\nabla\cdot\left(a\nabla u\right) = f & \text{in } \Omega_0 \\ \left(a\nabla u\right)\cdot\boldsymbol{n} = g_N & \text{on } \Sigma \end{cases} \tag{16}$$

The principle is about the same as for Dirichlet *BC*, and the same interpolations, once derived, can be used to approximate the quantity $\left(a\nabla u\right)\cdot\boldsymbol{n}$. Hence, at any point $x_l, l \in \mathcal{I}$ on Σ_h we use

$$\left(a\nabla u_l\right)\cdot\boldsymbol{n} \approx \left(a\nabla p\left(x_l\right)\cdot\boldsymbol{n}\right). \tag{17}$$

For $p \in \mathbb{Q}_1^2$, we get $\nabla p(x,y)\cdot\boldsymbol{n} = \left(p_3 y + p_2\right)n_x + \left(p_3 x + p_1\right)n_y$ whereas for $p \in \mathbb{P}_1^2$, $\nabla p(x,y)\cdot\boldsymbol{n} = p_2 n_x + p_1 n_y$ is obtained which means that the normal gradient is approximated by a constant over the whole support.

For example, in the configuration of **Figure 3(a)**, with $p \in \mathbb{P}_1^2$, we have:

$$\nabla p(x,y)\cdot\boldsymbol{n} = \frac{u_I^* - u_J}{h_x}n_x + \frac{u_K - u_I^*}{h_y}n_y = u_I^*\left(\frac{n_x}{h_x} - \frac{n_y}{h_y}\right) + u_J\frac{n_x}{h_x} + u_K\frac{n_y}{h_y} \tag{18}$$

The diagonal coefficient of the row related to u_I^* in C_2 is $\left(\frac{n_x}{h_x} - \frac{n_y}{h_y}\right)$. The case where $\frac{n_x}{h_x} \approx \frac{n_y}{h_y}$ leads to numerical instabilities. However, it can happens

for this configuration case only if n_x and n_y have the same sign, which will not happen if the interface is sufficiently smooth. For a smooth enough interface, $n_x > 0$ and $n_y < 0$. A smoothed interface can be simply obtained using the normal vector of the segment $[x_I, x_m]$ implies that the signs of n_x and n_y are always different so the diagonal coefficient is always dominant. The same property occurs for the other stencil configurations.

When x_I has only one neighbor x_J in Ω_0, the $\mathbb{Q}_1^2$ and $\mathbb{P}_1^2$ interpolations degenerate to $\mathbb{L}_1^1$ interpolations which suit for Dirichlet *BC*. For Neumann *BC*, this loss of dimension no longer allows the interface orientation to be accurately taken into account, as one of the components of the normal unit vector disappears from the interfacial constraint. Hence, a third point x_K in Ω_0 is caught to build $\mathbb{P}_1^2$ interpolations (see **Figure 3** right). This point is a neighbor of x_J and is taken as $[x_I, x_J] \perp [x_J, x_K]$. In 2D, two choices generally appear, and the point being so that the angle $(\boldsymbol{n}, x_K - x_J)$ is in $[-\pi/2; \pi/2]$ is taken.

2.3.5. Algebraic Elimination Using the Schur Complement

The Schur complement method allows an algebraic reduction to be performed. For a Dirichlet or Neumann *BC*, each constraint is written in such a way that only one auxiliary unknown is needed:

$$u_I^* = \sum_{J \in \mathcal{N}} \alpha_J u_J + u_S \tag{19}$$

where u_S is the source term. In this case, the matrix C_2 in (8) is diagonal and thus the Schur complement $\left(\tilde{A} - BC_2^{-1}C_1\right)$ is easy to calculate. Practically, when the algebraic reduction is made, $\tilde{A}$ is built directly by the suitable modification of A without considering the extended matrix A'. The part $-BC_2^{-1}C_1$ is then added to $\tilde{A}$ whereas $-BC_2^{-1}b^*$ is added to b. As it will be subsequently demonstrated, the algebraic reduction decreases the computational cost of the solver by 10% - 20%.

If only $\mathbb{L}_1^1$ interpolations are used with the algebraic elimination, the matrix obtained with this method is similar to the one obtained in [18] for a Dirichlet problem. However, in this last paper, the auxiliary unknowns are taken into account before the discretization of the operator which requires additional calculations for each discretization scheme.

If $\mathbb{P}_1^2$ interpolations are used, the computed solution in Ω_0 are very similar (within machine error if the solver perfectly converges) to those that can be obtained with the SMP [16] method (when the penalty parameter is sufficiently small) and the DF-IB method [10]. These methods discretize equivalent boundary constraints, but the way the constraint is applied to the initial discretization of the equation for the nodes $x_I, I \in \mathcal{N}_1$ differs. The SMP method uses a penalty term and the DF-IB method uses terms of opposite signs to erase some part of the initial equation. The discretization matrix obtained with both methods is not equivalent to the one obtained with the AIIB method, with or without algebraic reduction. With algebraic reduction, the discretization for the nodes

$x_I, I \in \mathcal{N}_0$ is modified, and without algebraic reduction, both auxiliary and physical unknowns coexist at $x_I, I \in \mathcal{N}_1$. The accuracy of these methods will be discussed in the next section.

The present algorithm seems simpler, as the standard discretization of the operators is automatically modified in an algebraic manner. So, various discretization schemes of the spatial operators can be used. However, the discretization of an operator at $x_I \in \Omega_0$ can only use in Ω_1 the fictitious unknowns and not the physical ones. Hence, the only limitation concerns the stencil of these operators which have to be limited, if centered, to three points by direction.

2.4. The AIIB for Immersed Interface Problems with Jump Conditions

With the symmetric method described in (2.3.3), the problem can be solved on both sides of the interface when explicit Dirichlet *BC* are imposed. For many problems, the solution is not *a priori* known on the interface and some jump transmission conditions on the interface Σ are required. Let us now consider the problem:

$$(\mathcal{P}_i) \quad \begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega \\ +\text{Interface conditions} & \text{on } \Sigma \end{cases}$$

where the interface conditions are:

$$[\![u_\Sigma]\!] = \varphi \text{ on } \Sigma \tag{20}$$

$$[\![(a \nabla u) \cdot \boldsymbol{n}]\!]_\Sigma = \psi \text{ on } \Sigma \tag{21}$$

The notation $[\![\,]\!]_\Sigma$ denotes the jump of a quantity over the interface Σ. In the symmetric version of the AIIB method, a given intersection point $x_l, l \in \mathcal{I}$, is associated with two auxiliary unknowns, one on each side of the interface. Hence, the interface constraints (20) and (21) of $(\mathcal{P}_i)$ can be imposed at each intersection point x_l by using the two auxiliary unknowns. For example, the row I of the matrix A' with $u_I^*, I \in \mathcal{N}_0$, can be used to impose the constraint (20) and the line J of the matrix with $u_J^*, J \in \mathcal{N}_1$, is then used to impose the constraint (21).

2.4.1. The Solution Constraint

The symmetrized AIIB methods for Dirichlet *BC* reads:

$$\begin{cases} u_\Sigma^+ = \alpha_1 u_I + \alpha_2 u_J^* \\ u_\Sigma^- = \alpha_1 u_I^* + \alpha_2 u_J \end{cases} \tag{22}$$

when $\mathbb{L}_1^1$ interpolations are used. With $u_\Sigma = u_\Sigma^+ - u_\Sigma^- = \varphi$, we obtain:

$$\alpha_1 u_I + \alpha_2 u_J^* - \alpha_1 u_I^* - \alpha_2 u_J = \varphi \tag{23}$$

which is the first constraint to be imposed.

2.4.2. The Flux Constraint

Following the same idea and using $\mathbb{P}_1^2$ interpolations,

$$\begin{cases} \left(a\cdot\nabla u_{\Sigma}^{+}\right)\cdot\boldsymbol{n} = a^{+}\left(\dfrac{u_I - u_J^*}{h_x}n_x + \dfrac{u_K^* - u_I}{h_y}n_y\right) \\ \left(a\cdot\nabla u_{\Sigma}^{-}\right)\cdot\boldsymbol{n} = a^{-}\left(\dfrac{u_I^* - u_J}{h_x}n_x + \dfrac{u_K - u_I^*}{h_y}n_y\right) \end{cases} \tag{24}$$

for the case presented in **Figure 3** left. The normal is pointing from the + to the − sides. Using (21), we obtain:

$$a^{+}\left(\frac{u_I - u_J^*}{h_x}n_x + \frac{u_K^* - u_I}{h_y}n_y\right) - a^{-}\left(\frac{u_I^* - u_J}{h_x}n_x - \frac{u_K - u_I^*}{h_y}n_y\right) = \psi \tag{25}$$

which is the second constraint to be imposed. With such an interpolation, the solution gradient is constant over the whole stencil. As demonstrated later, the second-order accuracy can be reached on Cartesian grids when $\psi = 0$.

Three auxiliary unknowns are thus involved in the discretizations (23) and (25). The auxiliary unknown u_K^* is also involved in the discretization of (20) and (21) at another intersection point on Σ_h. Hence, the whole system $A'u' = b'$ is closed.

One can notice that in [32], the same kind of augmented system is considered. Contrary to the AIIB method, no auxiliary nodes are used and the spatial discretization at the grid points at the vicinity of the interface has to be modified "by-hand".

2.4.3. Algebraic Reduction

Since we need more than one auxiliary unknown to discretize each constraint, the matrix C_2 is not diagonal and a solver has to be used to compute C_2^{-1}. This matrix is not diagonally dominant. For instance, if the discretized solution constraint (23) is inserted in C_2 for the row related to u_I^*, column I_* has a coefficient α_1 while the coefficient of the column J^* has a coefficient α_2. In many situations, α_2 is greater that α_1, and the situation can be critical if $\alpha_2 \gg \alpha_1$. The equivalent behaviour can happen with the discretized flux constraint (25). This point is discussed later in Section 3.4.

For the matched interface and boundary (MIB) method, Zhou *et al.* [29] use a different discretization of the interface conditions which allows an easy algebraic reduction which is directly performed row by row.

The algebraic reduction for the immersed interface problems has not been yet implemented. However, the standard discretization of the AIIB method requires a more compact stencil than for the MIB method and then the need for a reduction seems to be less critical. More generally, as auxiliary points are in the vicinity of a surface immersed in a volume, their count is one order below the total count of grid points. It suggests that the number of unknowns added to the initial matrix is low compared to the total number of unknowns.

3. Numerical Results for Scalar Problems

Elliptic equations are discretized using the standard second-order centered Lap-

lacian. For all problems, similar results have been obtained with a PARDISO direct solver [38], and an iterative BiCGSTAB solver [39], preconditioned under a ILUK method [40]. Unless otherwise mentioned, a numerical domain $[-1;1]\times[-1;1]$ is used for every simulation. Two discrete errors are used.

The discrete relative L^2 error in a domain Ω is defined as:

$$\left\|u\right\|_{L^2_{rel}(\Omega)} = \frac{\left\|u-\tilde{u}\right\|_{L^2(\Omega)}}{\left\|\tilde{u}\right\|_{L^2(\Omega)}} \tag{26}$$

$$= \left(\frac{\sum_{x_I \in \Omega} meas\left(\mathcal{V}_I\right)\left|u_I - \tilde{u}\left(x_I\right)\right|^2}{\sum_{x_I \in \Omega} meas\left(\mathcal{V}_I\right)\left|\tilde{u}\left(x_I\right)\right|^2} \right)^{\frac{1}{2}} \tag{27}$$

with $\tilde{u}$ the analytical solution.

The discrete L^∞ error is defined as:

$$\left\|u\right\|_{L^\infty(\Omega)} = \max_{x_I \in \Omega}\left|u_I - \tilde{u}\left(x_I\right)\right| \tag{28}$$

without AIIB method, the present code only reach a first-order spatial error convergence rate for cases with irregular interfaces. With the AIIB method, the code is expected to reach a second-order convergence rate as linear interpolations are used to approximate the solution at the vicinity of the interface.

Only Ω_0 is taken into account for the immersed boundary problems. For all cases, Σ is the boundary of Ω_1 as illustrated in **Figure 1**. If not explicitly written, φ and ψ have a default value of 0. For all cases, the analytical solution is imposed for the cells at the boundary of the numerical domain.

3.1. Immersed Boundary Problems

3.1.1. Problem 1

The homogenous 2D Laplace equation is solved. The interface Σ is a centered circle of radius $R_1 = 0.5$ with a Dirichlet condition of $U_1 = 10$. An analytical solution accounts for the presence of a second circle with a radius $R_2 = 2$ and $U_2 = 0$ is imposed on the boundary conditions. The analytical solution is:

$$u\left(r\right) = \left(U_2 - U_1\right)\frac{1-\ln\left(R_1\right)}{\ln\left(R_2\right)-\ln\left(R_1\right)}\ln\left(r\right) + U_1 \tag{29}$$

Accuracy tests are performed with $\mathbb{L}^1_1$, $\mathbb{P}^2_1$ and $\mathbb{Q}^2_1$ interpolations. **Figure 4** shows the solution and the error map for a 32 × 32 mesh with $\mathbb{P}^2_1$ interpolations. The same results are always obtained with and without algebraic reduction. **Figure 5** shows the convergence of the error for the L^2 and L^∞ norms. For all interpolations, the convergence slopes are approximately 2 for the relative L^2 error. For the L^∞ error, the slopes are about 1.8. The $\mathbb{P}^2_1$ interpolation is the more accurate, followed by the $\mathbb{L}^1_1$ interpolation although it uses more auxiliary points (but a smaller stencil). However, the differences of accuracy between the different interpolations remain small. The performances of the ILUK-BiCG-Stab

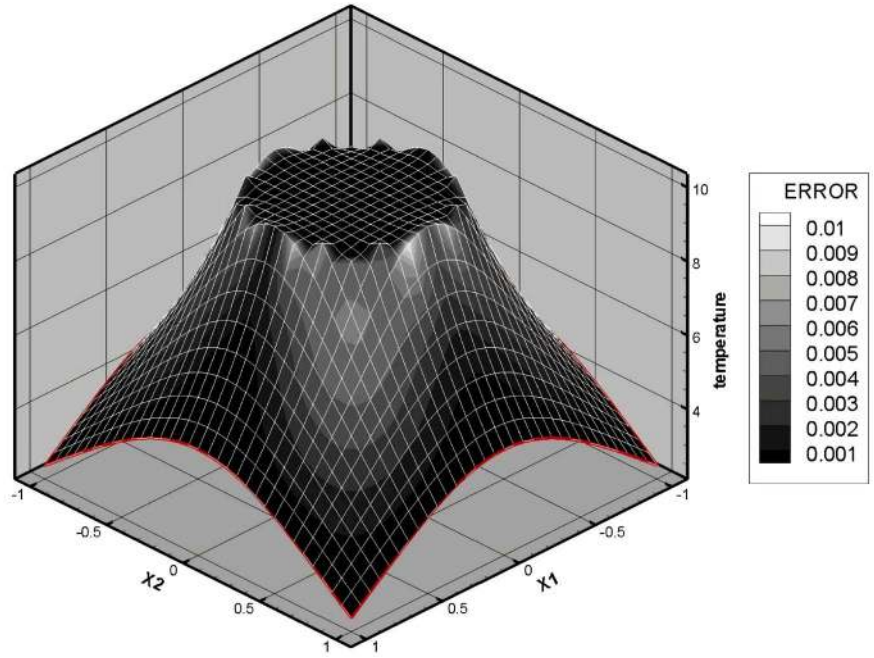


Figure 4. Solution and error map for problem 1 on a 32 × 32 grid.

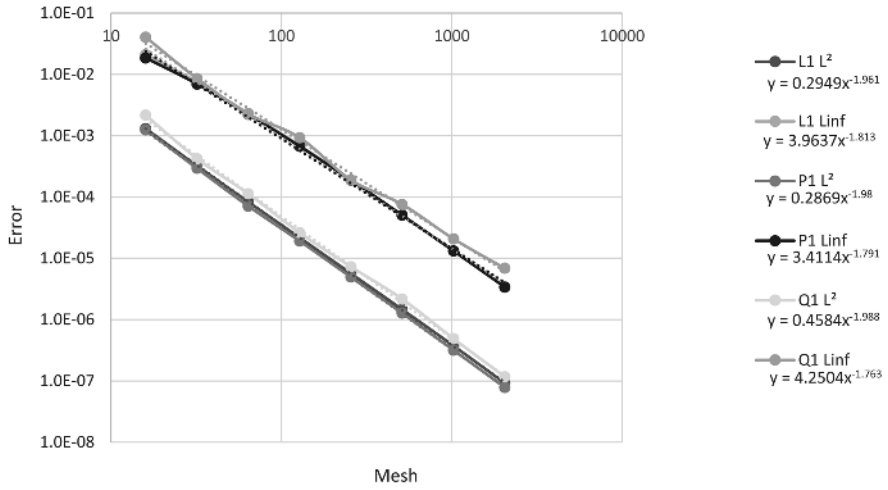


Figure 5. Curves of errors for Section 3.1.1.

solver are now benchmarked for the three interpolations with and without algebraic reduction and for the SMP method. Table 1 shows the computational times of the matrix inversions (average time in seconds for 25 matrix inversions) and Table 2 shows the time ratio between the standard and the reduced matrix. Except for the $\mathbb{Q}_1^2$ interpolation on the 1024 × 1024 mesh, the differences between the two methods seem to decrease with the size of the matrix. In fact, as interfaces are $d-1$ manifolds, the number of intersection points does not increase as fast as the Eulerian points. Hence, the ratio between the size of a reduced and a complete matrix tends to 1. The computational time for the SMP method is quite similar to the one obtained with the AIIB method and algebraic reduction. Figure 6 shows the convergence of the ILUK-BiCG-Stab solver for

Table 1. Computational times in seconds for problem 1. Tests are performed with three different interpolations with (red) and without (std) algebraic reduction, and compared to the SMP method.

Mesh	$\mathbb{L}_1^1$ std	$\mathbb{L}_1^1$ red	$\mathbb{P}_1^2$ std	$\mathbb{P}_1^2$ red	$\mathbb{Q}_1^2$ std	$\mathbb{Q}_1^2$ red	$\mathbb{P}_1^2$ SMP
128	0.215	0.189	0.216	0.182	0.208	0.181	0.181
256	2.18	1.89	2.14	1.83	2.14	1.88	1.88
512	19.7	17.6	19.5	17.1	20.3	18.4	16.9
1024	168	159	171	156	173	141	168

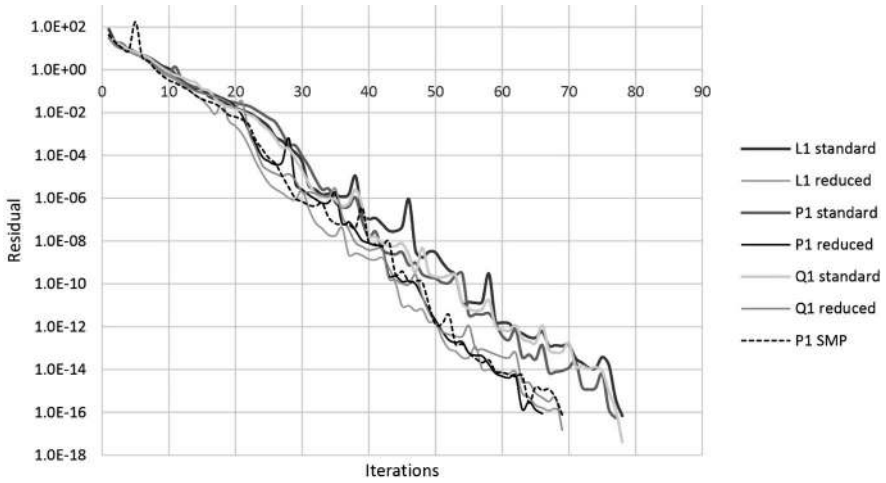


Figure 6. Residual against iterations of ILUK solver for problem 1 with a 256 × 256 mesh.

the seven interpolations with a 256 × 256 mesh. The same behavior has been observed for other grid resolutions. The type of interpolation does not significantly impact on solver performances. As expected, the number of solver iterations has to be increased to reach a given residual when the number of computational nodes is also increased.

As a conclusion on the interpolation type, $\mathbb{L}_1^1$ interpolation seems to be the most desirable as it is generally slightly easier to implement, and requires less memory than a $\mathbb{Q}_1^1$ interpolation. To explain that, one can suppose that the boundary constraint forces the points of the stencil to follow a polynomial solution. As the real solution is not polynomial, adding points to the stencil increases the zone where the analytical solution is more difficult to match. For the next problems, $\mathbb{P}_1^1$ will be used for constraints on the solution.

3.1.2. Problem 2

The 3D equation $\Delta T = 6$ is solved. The solution is $T(r) = r^2$. The solution is imposed on an immersed centered sphere of radius 0.2. Results of the numerical accuracy test with the spherical inner boundary are presented in **Figure 7**. The average slope for the L^2 norm is 2.33 and increases for the denser meshes. Even if the method has a convergence order similar to the order of the polynomial solution, computer error accuracy cannot be expected because the immersed boundary Σ_h is an approximation of the initial interface Σ. The analytical solution

Table 2. Ratio of computational times for reduced and standard matrices for Section 3.1.1.

Mesh	$\mathbb{L}_1^1$	$\mathbb{P}_1^2$	$\mathbb{Q}_1^2$ std
128	88.3%	84.5%	87.3%
256	86.9%	85.5%	88.2%
512	89.4%	87.5%	90.9%
1024	94.6%	91.2%	81.5%

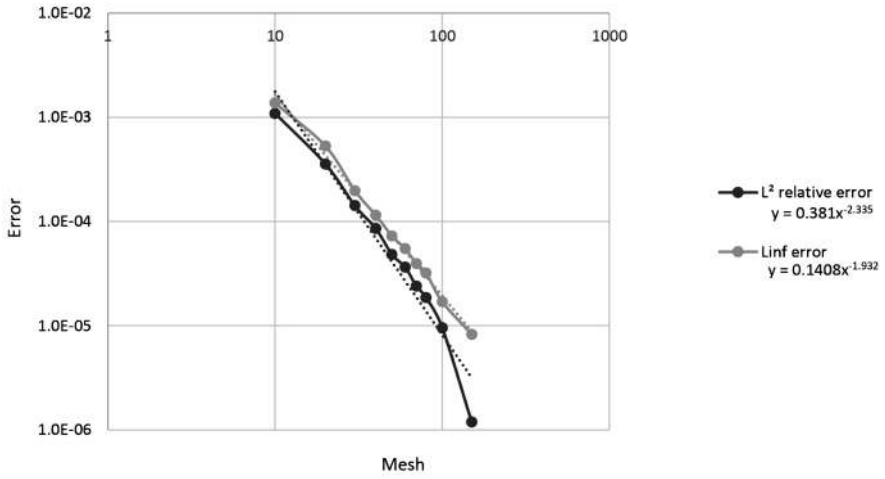


Figure 7. Curves of errors for problem 2.

(the value at the boundary) is imposed on the boundary, but the boundary location is approximated so an error which has the order of the discretization error of the interface is introduced.

3.1.3. Problem 3

The 3D equation $\Delta T = 12r^2$ is solved. The solution is $T(r) = x^4 + y^4 + z^4$. Results with and without an immersed interface (a circle of radius $r = 1$) with the analytical solution imposed on it. The results are presented in **Figure 8**. For the L^∞ norm, the second order is regularly obtained. For the L^2 norm, the convergence order tends to a second order with the size of the mesh. As can be noticed by comparing results with and without the AIIB method, this last one does not spoil the convergence order of the code, and the presence of the immersed interface with an analytical solution imposed on Σ_h improves the accuracy. For both cases, the numerical solution tends to a second order in space.

3.1.4. Problem 4

The 2D equation $\Delta T = 4$ is solved. The analytical solution is imposed on the boundaries of the domain and a Neuman *BC* is imposed on a centered circle of radius $R = 0.5$. As can be seen in **Figure 9**, the global convergence has an average slope of 1.10 and can be explained by the simplicity of the discretization of the Neumann *BC*.

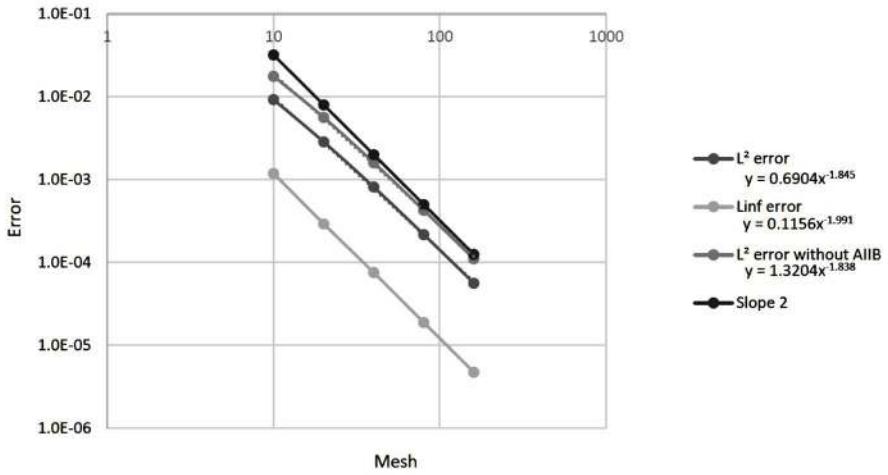


Figure 8. Curves of errors for problem 3.

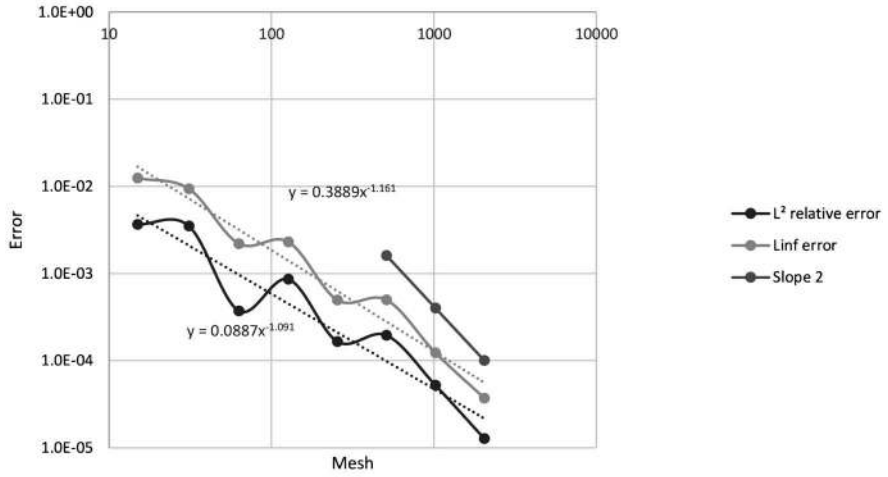


Figure 9. Curves of errors for problem 4.

3.2. Immersed Interface Problems

3.2.1. Problem 5

The 2D problem $\mathcal{P}_i$ with $f=-4$ and $a=1$ is solved. As the equation remains the same in both domains, this problem can be solved without immersed interface method. The analytical solution is $u=r^2$. As can be expected with our second-order code, computer error is reached for all meshes with or without AIIB method. The difference with problem 2, where the solution is a second-order polynomial too, is that instead of imposing the solution on an approximated shape, we impose a solution jump. If the numerical nodes stay in the correct side of the interface (where they should be if the interface were not approximated), the location of the jump does not influence the numerical solution.

In **Figure 10** the same result is obtained with a solution jump $\varphi=-1$ on a circular interface such as $u=r^2$ for $r>0.5$ (Ω_1) and $u=r^2+1$ otherwise (Ω_0).

An equivalent quality of result is obtained (see **Figure 11**) with Σ such as:

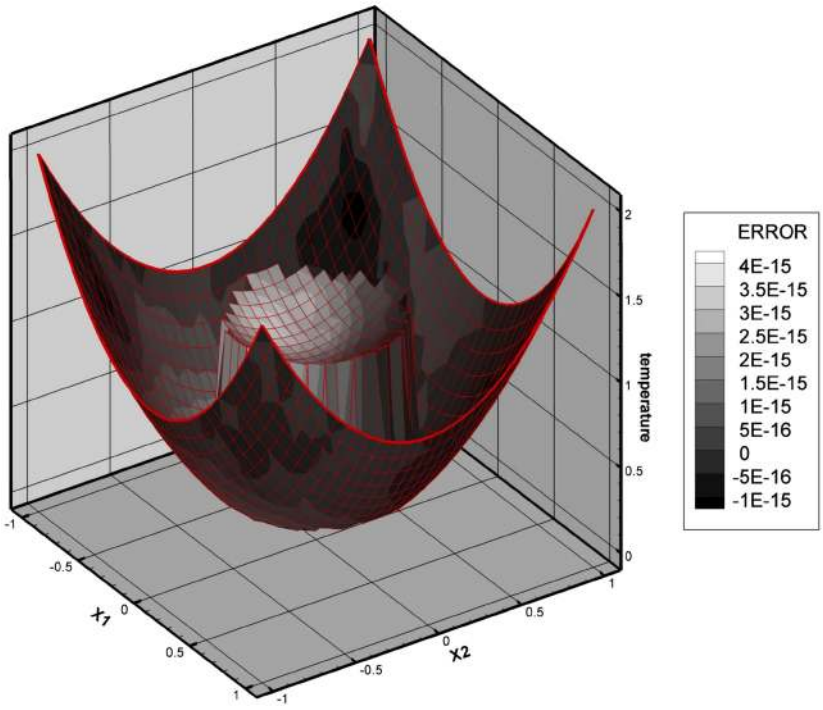


Figure 10. The solution and the error for problem 5 with a 32 × 32 mesh.

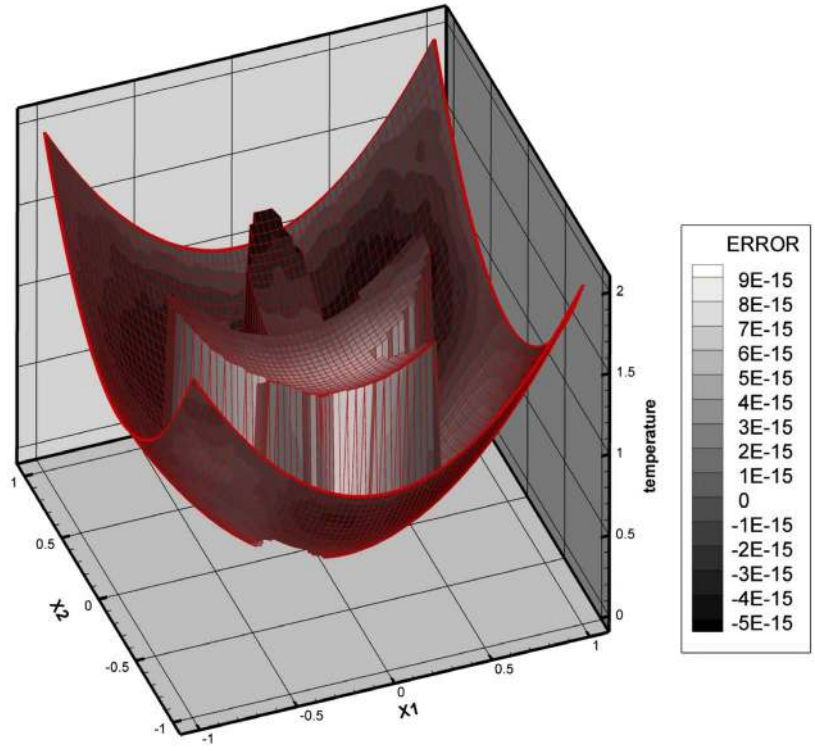


Figure 11. The solution and the error for problem 5 with a 64 × 64 mesh.

$$\begin{cases} x(\alpha) = (0.5 + 0.2\sin(5\alpha))\cos(\alpha) \\ y(\alpha) = (0.5 + 0.2\sin(5\alpha))\sin(\alpha) \end{cases} \tag{30}$$

with $\alpha \in [0, 2\pi]$. The small stencil of the method allows interfaces with relatively strong curvatures to be used.

3.2.2. Problem 6

For this case, $(\mathcal{P}_i)$ is now considered with a discontinuous coefficient *a* such as $a = 10$ in Ω_0 and $a = 1$ in Ω_1, involving the following analytical solution:

$$u(r) = \begin{cases} r^2 & \text{in } \Omega_0 \\ \dfrac{r^2}{10} + \dfrac{0.9}{4} & \text{in } \Omega_1 \end{cases} \tag{31}$$

Accuracy tests are first performed with the interface almost passing by some grid points (the mesh is referred as centered). The interface does not strictly lies on these points, as the shape is shifted by a value of 10^{-10}. This configuration is difficult as the interpolations degenerates. Accuracy tests are then performed with a numerical domain of size 1.0001 (called shifted mesh as the center of Σ does not match the center of the numerical domain). In this configuration, the interface never passes by a grid point. The numerical errors on the solution are presented in Figure 12. For the centered mesh, the slope is 1.86 for the L^2 and L^∞ errors. For the even series, where no geometrical singularity is present, the slope for both errors is 2.04.

Figure 13 shows the solution and the L^2 relative error for a 32 × 32 mesh. As the analytical solution is imposed on the numerical boundary, the error is principally located in the interior subdomain.

3.2.3. Problem 7

The homogenous 2D Laplace equation is considered with the following analytical solution:

$$u(x, y) = \begin{cases} 0 & \text{in } \Omega_0 \\ e^x \cos(y) & \text{in } \Omega_1 \end{cases} \tag{32}$$

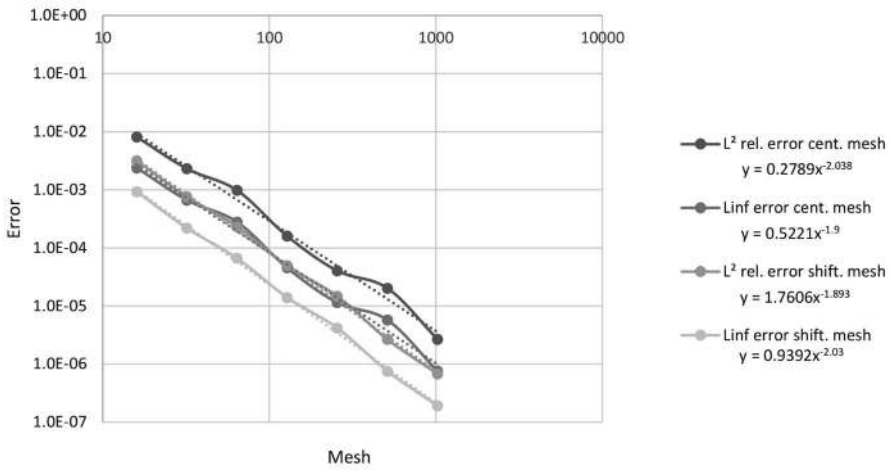


Figure 12. Curves of errors for the centered and shifted meshes for problem 6.

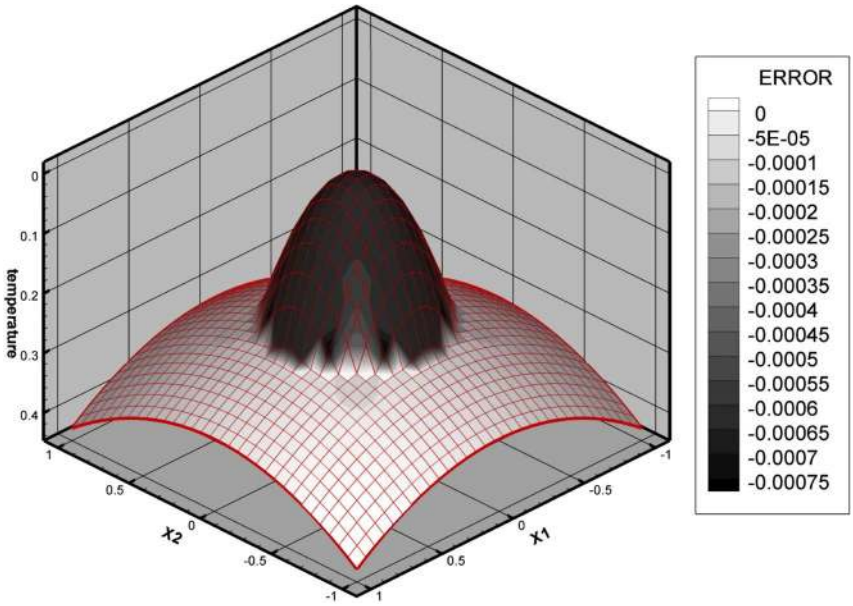


Figure 13. The solution and the error for problem 6 with a 33 × 33 mesh.

where Ω_0 and Ω_1 are delimited by Σ a centered circle of radius 0.5. The jumps coefficients are $\varphi = -\mathrm{e}^x \cos(y)$, $\psi = \mathrm{e}^x \cos(y) n_x - \mathrm{e}^x \sin(y) n_y$ and $a = 1$.

Figure 14 shows that the convergence for both L^2 and L^∞ error are of first order only. **Figure 15** shows the numerical solution (which is not so different from the analytical solution) and the error map for a 32 × 32 mesh.

Hence, the drawback of the compacity of the discretization is a first-order convergence in space only for cases with flux jump. As for the Neumann *BC*, the simplicity of the discretization of the flux condition seems to be the cause of this low order. Contrary to the discretization of the solution jump, the precise position of the interface is not present which induces a first order of convergence. Generally, other approaches [20] [29] [32] use the precise position of the interface (at the expanse of a larger stencil) and can reach higher orders with all flux conditions.

3.3. Shape Management

3.3.1. Convergence

We measure the sensibility of the method with the accuracy of the discretization of the immersed interface. Problem 1 is solved on 32 × 32 and 128 × 128 meshes. **Figure 16** shows the accuracy of the solution with respect to the number of points used to discretize the interface which is here a circle. The reference solutions (**Figure 5**) have been computed with an analytical circle. As can be seen, a second order in space is globally obtained. The numerical solutions of reference for the 32 × 32 and 128 × 128 meshes are different but the sensitivity of the error to the number of points in the Lagrangian mesh is almost the same.

3.3.2. The Stanford Bunny

This last case demonstrates how a method with a second-order convergence rate

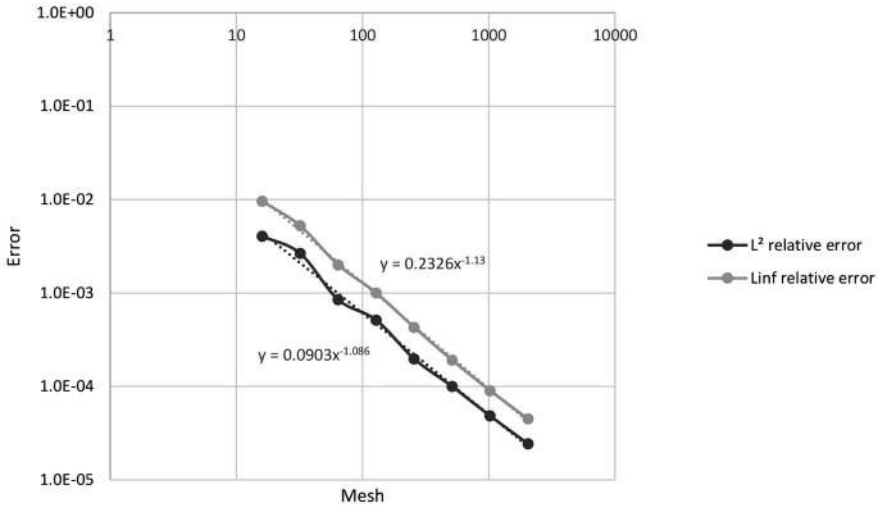


Figure 14. Convergence of the L^2 relative error and the L^∞ error for problem 7.

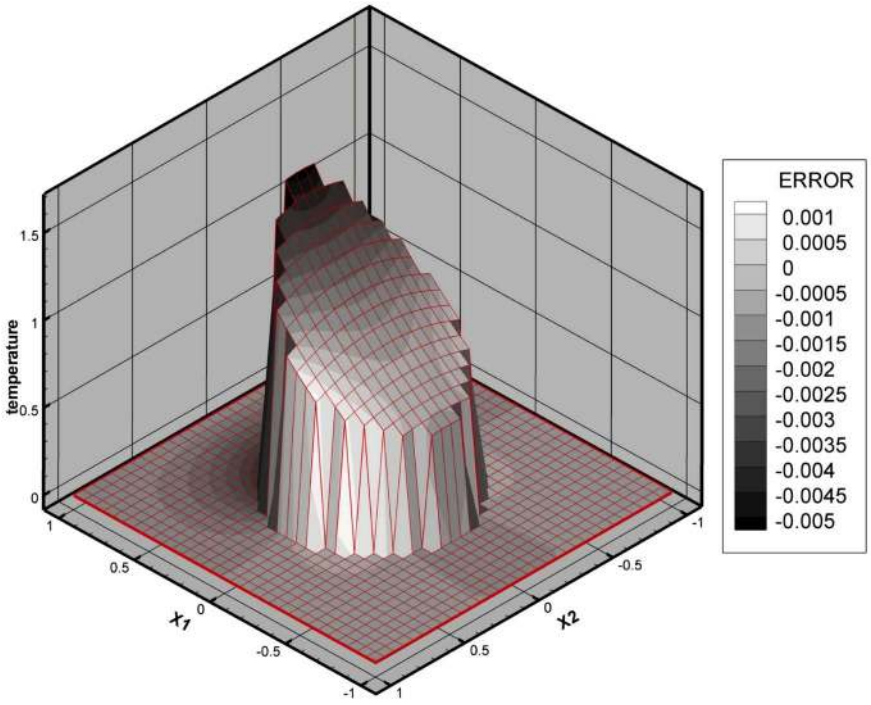


Figure 15. The solution and the L^2 relative error for problem 7 with a 32 × 32 mesh.

on spatial accuracy enhances the representation of the boundary condition compared to a first-order method such as the VPM which imposes a prescribed solution at the center of the control volume as shown in Equation (1).

The homogenous Laplace problem with a Dirichlet BC $T_\Sigma = 10$ is solved on a $60 \times 60 \times 50$ mesh bounding an obstacle of complex shape (the Stanford bunny). The extension of the solution in Ω_1 is used for the post treatment. Thus, all $u_J, J \in \mathcal{N}_1$, are replaced by u_J^*. Then, the iso-surface $T = T_\Sigma$ gives an idea of the approximation of the boundary condition. Figure 17 shows the iso-surface for a first order method. As can be seen, the shape of the obstacle endures a

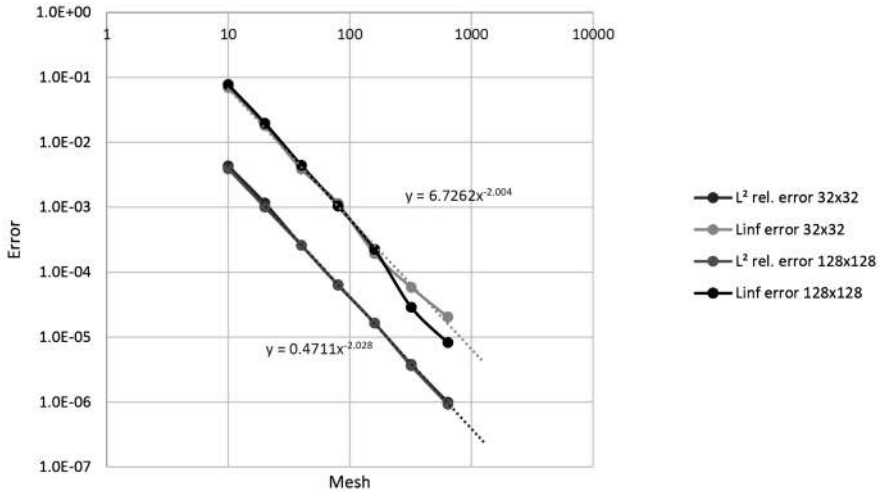


Figure 16. Convergence of the error with respect to the number of elements forming the Lagrangian shape for problem 1.

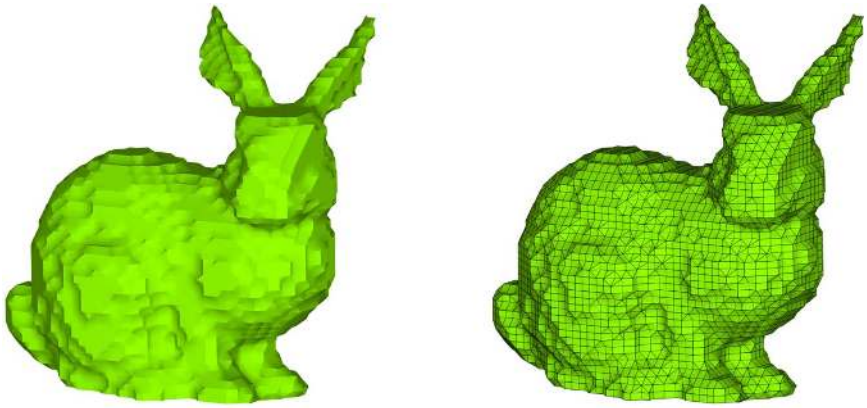


Figure 17. Iso-surface $T=10$ for the Stanford bunny with a first order method.

rasterization effect as the solution is imposed in the entire control volumes. **Figure 18** shows the iso-surface for the second order AIIB method. The improvement brought by this approach is straightforward. The graphic processing has been applied to both methods. **Figure 19** shows a slice of the solution passing through the bunny. As can be seen, overshoots are present inside the shape which corresponds to the auxiliary values allowing the correct solution at the Lagrangian interface points to be obtained.

3.4. Some Remarks about the Solvers

The kind of interpolation function used and the position of the interface have an impact on the final discretization matrix A', especially on its conditioning. Let us consider an intersection x_I of Σ_h between two points $x_J, J \in \mathcal{N}_0$ and $x_I, I \in \mathcal{N}_1$. A Dirichlet BC u_I is imposed on it. The constraint constructed with a $\mathbb{L}_1^1$ interpolation is $(1-\alpha)u_J + \alpha u_I^*$, with $\alpha = \frac{x_I - x_J}{x_I - x_J}$. Hence, $\frac{\alpha}{1-\alpha}$ tends to 0 when x_I tends to x_J. As the matrix loses its diagonal dominance, solver

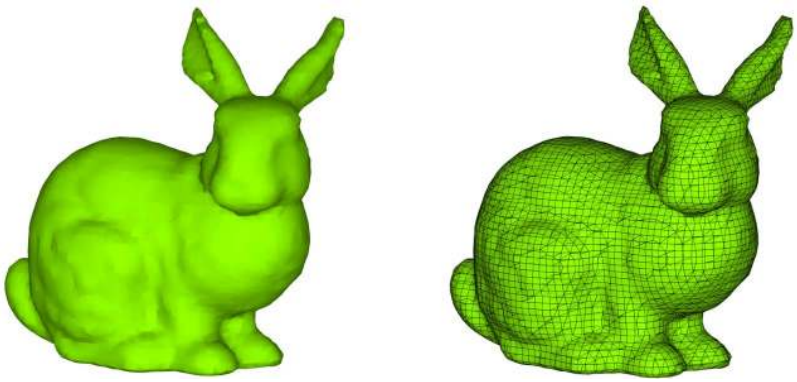


Figure 18. Iso-surface $T=10$ for the Stanford bunny with a second order method.

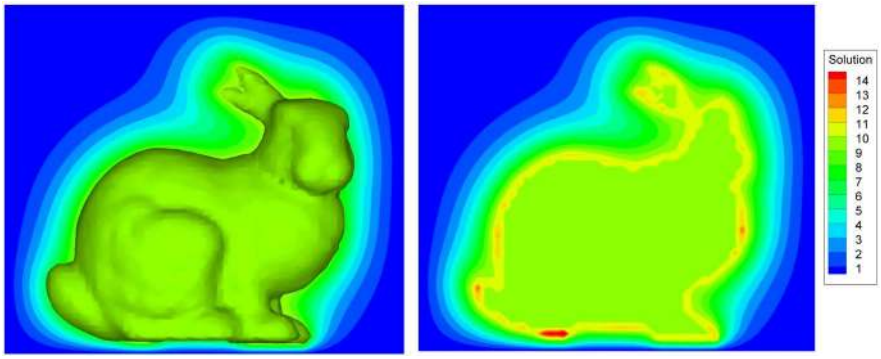


Figure 19. Iso-surface $T=10$ and a slice of the solution.

problems can be encountered. Tseng *et al.* [10] proposed changing the interpolation by using a new node which is the image of x_I through the interface. In [18] [19], authors pointed out this problem and suggest to slightly move the interface to a neighboring point (in our case x_J) if x_I is too close to Σ_h.

In our approach, a simple solution is to shift the interface such as the interface matches x_I. The loss of accuracy depends on the threshold of distance between x_I and Σ_h under which the solver is unstable. In our case, test 3.2.2 suggests that a threshold below the error of the discretization of Σ into the piecewise linear interface Σ_h can be taken.

In the case of interface shift, for the Dirichlet *BC*, an unknown u_J^* is created, and the equation at x_J is simply $u_J^* = u_I$. For the Neumann *BC*, the standard interpolation is written at x_J with u_J^* and its neighbor unknowns in Ω_0.

For the transmission conditions (20)-(21), if $\phi = 0$ and $\psi = 0$, no auxiliary unknown is created and the standard finite-volume centered discretization is used. However, for this case, or for $\phi \neq 0$ and $\psi \neq 0$, our implementation using ILUK preconditioner or a PARDISO direct solver does not necessarily require such methods, even if $\frac{\alpha}{1-\alpha} \approx 10^{-10}$. Correct solutions are always obtained.

4. Discussion and Conclusions

A new immersed interface method, the algebraic immersed interface and boun-

dary (AIIB) method, which uses algebraic manipulations and compact stencil discretizations, has been presented. This method is able to treat elliptic equations with discontinuous coefficients and solution jumps over complex interfaces.

For the immersed boundary problems with a Dirichlet *BC*, the method has shown a second order of convergence in space for various kinds of interpolations. An algebraic reduction has been applied to accelerate the convergence of the solver. For the Neumann *BC*, a first order has been obtained. For the immersed interfaces, a second order of convergence in space is obtained when the jump of the normal flux is zero, even if the equation has discontinuous coefficients. For a non-zero flux jump, the method reaches a first order.

The main advantage of the method is its simplicity and its ease of implementation. First, the method does not require to modify the original discretization of the equation and works only at the level of the discretization matrix. Second, the method has a compact stencil and interfaces with high curvature can be treated with no particular attention.

Future work could be devoted to increase the accuracy of the method when the jump of the normal flux is not zero. It is the main drawback of the method compared to the IIM, MIB method or [32]. A challenge will be to keep a compact stencil. A study of the matrix conditioning would be important too as a strong solver is required for the present method. Another interesting point would be to couple the method with alternative interface conditions such as the Jump Embedded Boundary Conditions proposed in [41] which are:

$$\left[\!\left[\left(\boldsymbol{a}\cdot\nabla u\right)\cdot\boldsymbol{n}\right]\!\right]_{\Sigma} = \alpha\overline{u}_{|\Sigma} - h \quad \text{on } \Sigma \tag{33}$$

$$\overline{\left(\boldsymbol{a}\cdot\nabla u\right)\cdot\boldsymbol{n}}_{\Sigma} = \beta u - g \quad \text{on } \Sigma \tag{34}$$

where $\overline{u}_{\Sigma} = \left(u^{+} + u^{-}\right)/2$ denotes the arithmetic mean of the traces of a quantity on both sides of the interface; α, β, h and g are scalar values which can be chosen in order to obtain Dirichlet, Neumann, Fourier transmission conditions on the interface.

As the method acts on an algebraic level, it can be extended to other implicit approaches where the solution is obtained through the resolution of a linear system. It would be interesting to apply this method to finite element approaches. The interface conditions currently written for finite volumes could be discretized in a finite element manner.

At last, a long-term goal is to extend the method to the Navier-Stokes equations with immersed interfaces. To perform fluid/structure coupling, the implicit tensorial penalty method [42] [43] can be considered. With this approach, the solid medium is treated as a fluid with a high viscosity. At the fluid/solid interface, average physical quantities are imposed. Such strategy is generally less accurate than methods using polynomial interpolations so a coupling with the AIIB method is desirable.

Acknowledgements

The authors greatly thank the reviewers for their accurate and pertinent ques-

tions and remarks that helped to improve this article. The authors also thank Lisl Weynans for the helpful discussions.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Glowinski, R., Pan, T.W., Hesla, T.I. and Joseph, D.D. (1999) A Distributed Lagrange Multiplier/Fictitious Domain Method for Particulate Flows. *International Journal of Multiphase Flow*, **25**, 755-794. https://doi.org/10.1016/S0301-9322(98)00048-2

[2] Johansen, H. and Colella, P. (1998) A Cartesian Grid Embedded Boundary Method for Poisson's Equation on Irregular Domains. *Journal of Computational Physics*, **147**, 60-85. https://doi.org/10.1006/jcph.1998.5965

[3] McCorquodale, P., Colella, P. and Johansen, H. (2001) A Cartesian Grid Embedded Boundary Method for the Heat Equation on Irregular Domains. *Journal of Computational Physics*, **173**, 620-635. https://doi.org/10.1006/jcph.2001.6900

[4] Ye, T., Mittal, R., Udaykumar, H.S. and Shyy, W. (1999) An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries. *Journal of Computational Physics*, **156**, 209-240. https://doi.org/10.1006/jcph.1999.6356

[5] Peskin, C.S. (1972) Flow Patterns around Heart Valves: A Numerical Method. *Journal of Computational Physics*, **10**, 252-271. https://doi.org/10.1016/0021-9991(72)90065-4

[6] Peskin, C.S. (2002) The Immersed Boundary Method. *Acta Numerica*, **11**, 479-517. https://doi.org/10.1017/S0962492902000077

[7] Mohd-Yusof, J. (1997) Combined Immersed Boundary/B-Spline Methods for Simulations of Flows in Complex Geometries. Technical Report, NASA ARS/Stanford University CTR, Stanford.

[8] Fadlun, E.A., Verzicco, R., Orlandi, P. and Mohd-Yusof, J. (2000) Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations. *Journal of Computational Physics*, **161**, 35-60. https://doi.org/10.1006/jcph.2000.6484

[9] Verzicco, R., Iaccarino, G., Fatica, M. and Orlandi, P. (2001) Flow in an Impeller-Stirred Tank Using an Immersed Boundary Method. *AIChE Journal*, **50**, 1109-1118.

[10] Tseng, Y.-H. and Ferziger, J.H. (2003) A Ghost-Cell Immersed Boundary Method for Flow in Complex Geometry. *Journal of Computational Physics*, **192**, 593-623. https://doi.org/10.1016/j.jcp.2003.07.024

[11] Sarthou, A., Vincent, S., Caltagirone, J.-P. and Angot, P. (2008) Eulerian-Lagrangian Grid Coupling and Penalty Methods for the Simulation of Multiphase Flows Interacting with Complex Objects. *International Journal for Numerical Methods in Fluids*, **56**, 1093-1099. https://doi.org/10.1002/fld.1661

[12] Angot, P., Bruneau, C.-H. and Fabrie, P. (1999) A Penalization Method to Take into Account Obstacles in Incompressible Viscous Flows. *Numerische Mathematik*, **81**, 497-520. https://doi.org/10.1007/s002110050401

[13] Khadra, K., Angot, P., Parneix, S. and Caltagirone, J.-P. (2000) Fictitious Domain Ap-

proach for Numerical Modelling of Navier-Stokes Equations. *International Journal for Numerical Methods in Fluids*, **34**, 651-684. https://doi.org/10.1002/1097-0363(20001230)34:8%3C651::AID-FLD61%3E3.0.CO;2-D

[14] Lacanette, D., Vincent, S., Sarthou, A., Malaurent, P. and Caltagirone, J.P. (2009) An Eulerian/Lagrangian Method for the Numerical Simulation of Incompressible Convection Flows Interacting with Complex Obstacles: Application to the Natural Convection in the Lascaux Cave. *International Journal of Heat and Mass Transfer*, **52**, 2528-2542. https://doi.org/10.1016/j.ijheatmasstransfer.2008.12.028

[15] Ramière, I., Angot, P. and Belliard, M. (2007) A General Fictitious Domain Method with Immersed Jumps and Multilevel Nested Structured Meshes. *Journal of Computational Physics*, **225**, 1347-1387. https://doi.org/10.1016/j.jcp.2007.01.026

[16] Sarthou, A., Vincent, S., Angot, P. and Caltagirone, J.-P. (2008) Finite Volumes for Complex Applications V, Chapter the Sub-Mesh-Penalty Method. Wiley, Hoboken, 633-640.

[17] Sarthou, A., Vincent, S. and Caltagirone, J.-P. (2020) Consistent Velocity-Pressure Coupling for Direct-Forcing and Penalty Methods. *Fluids*, **5**, 92.

[18] Gibou, F., Fedkiw, R.P., Cheng, L.-T. and Kang, M. (2002) A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *Journal of Computational Physics*, **176**, 205-227. https://doi.org/10.1006/jcph.2001.6977

[19] Gibou, F. and Fedkiw, R. (2005) A Fourth Order Accurate Discretization for the Laplace and Heat Equations on Arbitrary Domains, with Applications to the Stefan Problem. *Journal of Computational Physics*, **202**, 577-601. https://doi.org/10.1016/j.jcp.2004.07.018

[20] Leveque, R.J. and Li, Z. (1994) The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources. *SIAM Journal of Numerical Analysis*, **31**, 1001-1025. https://doi.org/10.1137/0731054

[21] Li, Z.L. and Ito, K. (2006) The Immersed Interface Method. Numerical Solutions of PDEs Involving Interfaces and Irregular Domains. SIAM, Philadelphia. https://doi.org/10.1137/1.9780898717464

[22] Hou, T.Y., Li, Z.L., Osher, S. and Zhao, H. (1997) A Hybrid Method for Moving Interface Problems with Application to the Hele-Shaw Flow. *Journal of Computational Physics*, **134**, 236-252. https://doi.org/10.1006/jcph.1997.5689

[23] Lai, M.-C. and Tseng, H-.C. (2008) A Simple Implementation of the Immersed Interface Methods for Stokes Flows with Singular Forces. *Computers and Fluids*, **37**, 99-106. https://doi.org/10.1016/j.compfluid.2007.04.003

[24] Li, Z. (1998) A Fast Iterative Algorithmfor Elliptic Interfaces Problems. *SIAM Journal of Numerical Analysis*, **35**, 230-254. https://doi.org/10.1137/S0036142995291329

[25] Li, Z.L., Ito, K. and Lai, M.-C. (2007) An Augmented Approach for Stokes Equations with a Discontinuous Viscosity and Singular Forces. *Computers & Fluids*, **36**, 622-635. https://doi.org/10.1016/j.compfluid.2006.03.003

[26] Ito, K., Lai, M.-C. and Li, Z.L. (2009) A Well-Conditioned Augmented System for Solving Navier-Stokes Equations in Irregular Domains. *Journal of Computational Physics*, **228**, 2616-2628. https://doi.org/10.1016/j.jcp.2008.12.028

[27] Fedkiw, R.P., Aslam, T., Merriman, B. and Osher, S. (1999) A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (The Ghost Fluid Method). *Journal of Computational Physics*, **152**, 457-492. https://doi.org/10.1006/jcph.1999.6236

[28] Liu, X.-D., Fedkiw, R.P. and Kang, M. (2000) A Boundary Condition Capturing

Method for Poisson's Equation on Irregular Domains. *Journal of Computational Physics*, **160**, 151-178. https://doi.org/10.1006/jcph.2000.6444

[29] Zhou, Y.C., Zhao, S., Feig, M. and Wei, G.W. (2006) High Order Matched Interface and Boundary Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources. *Journal of Computational Physics*, **213**, 1-30. https://doi.org/10.1016/j.jcp.2005.07.022

[30] Zhou, Y.C. and Wei, G.W. (2006) On the Fictitious-Domain and Interpolation Formulations of the Matched Interface and Boundary (MIB) Method. *Journal of Computational Physics*, **219**, 228-246. https://doi.org/10.1016/j.jcp.2006.03.027

[31] Yu, S.N., Zhou, Y.C. and Wei, G.W. (2007) Matched Interface and Boundary (MIB) Method for Elliptic Problems with Sharp-Edged Interfaces. *Journal of Computational Physics*, **224**, 729-756. https://doi.org/10.1016/j.jcp.2006.10.030

[32] Cisternino, M. and Weynans, L. (2012) A Parallel Second Order Cartesian Method for Elliptic Interface Problems. *Communications in Computational Physics*, **12**, 1562-1587. https://doi.org/10.4208/cicp.160311.090112a

[33] Sussman, M. and Fatemi, E. (1999) An Efficient, Interface-Preserving Level Set Redistancing Algorithm and Its Application to Interfacial Incompressible Fluid Flow. *SIAM Journal of Scientific Computing*, **20**, 1165-1191. https://doi.org/10.1137/S1064827596298245

[34] Shin, S. and Juric, D. (2002) Modelling Three-Dimensional Multiphase Flow Using a Level Contour Reconstruction Method for Front Tracking without Connectivity. *Journal of Computational Physics*, **180**, 427-470. https://doi.org/10.1006/jcph.2002.7086

[35] Sarthou, A., Vincent, S. and Caltagirone, J.P. (2010) A Second-Order Curvilinear to Cartesian Transformation of Immersed Interfaces and Boundaries. Application to Fictitious Domains and Multiphase Flows. *Computers & Fluids*, **46**, 422-428. https://doi.org/10.1016/j.compfluid.2010.11.008

[36] Sarthou, A. (2009) Fictitious Domain Methods for the Elliptic and Navier-Stokes Equations. Application to Fluid-Structure Coupling. PhD Thesis, Université Bordeaux, Bordeaux.

[37] Svard, M. and Nordstrom, J. (2006) On the Order of Accuracy for Difference Approximations of Initial-Boundary Value Problems. *Journal of Computational Physics*, **352**, 219-333. https://doi.org/10.1016/j.jcp.2006.02.014

[38] Schenk, O. and Gärtner, K. (2004) Solving Unsymmetric Sparse Systems of Linear Equations with Pardiso. *Journal of Future Generation Computing Systems*, **20**, 475-487. https://doi.org/10.1016/j.future.2003.07.011

[39] Gustafsson, I. (1978) On First and Second Order Symmetric Factorization Methods for the Solution of Elliptic Difference Equations. Research Report, Department of Computer Sciences, Chalmersuniversity of Technology and the University of Gateborg, Gateborg.

[40] Saad, Y. and Schultz, M.H. (1986) Gmres: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, **7**, 856-869. https://doi.org/10.1137/0907058

[41] Angot, P. (2005) A Unified Fictitious Domain Model for General Embedded Boundary Conditions. *Comptes Rendus Mathematique*, **341**, 683-688. https://doi.org/10.1016/j.crma.2005.09.046

[42] Randrianarivelo, T.N., Pianet, G., Vincent, S. and Caltagirone, J.-P. (2005) Numerical Modelling of Solid Particle Motion Using a New Penalty Method. *International*

Journal for Numerical Methods in Fluids, **47**, 1245-1251.
https://doi.org/10.1002/fld.914

[43] Stéphane, V., De Motta, J.C.B., Sarthou, A., Estivalezes, J.-L., Simonin, O. and Climent, E. (2012) A Lagrangian VOF Tensorial Penalty Method for the DNS of Resolved Particle-Laden Flows. *Journal of Computational Physics*, **256**, 582-614.

Definition of the Parameters

Name	Definition
Ω	Computational domain
Ω_0	Subdomain of Ω
Ω_1	Subdomain of Ω
Σ	Immersed boundary of the interface with $\Omega = \Omega_0 \cup \Sigma \cup \Omega_1$
$\boldsymbol{n}$	Normal unit outward vector to Ω_0 on Σ
$\mathcal{P}_b$	Model immersed boundary problem
$\mathcal{P}_i$	Model immersed interface problem
ε	Penalty parameter
χ	Indicator function of Ω_1
φ	Value of the solution jump on Σ
ψ	Value of the flux jump on Σ
$\mathcal{N}_0, \mathcal{N}_1$	Set of index of the Eulerian interface points
$\mathcal{V}_l$	Set of cell-centered finite volume
$\mathcal{V}_l'$	Set of dual finite volume
$\mathcal{I}$	Set of index of the intersection points between σ_e and the faces σ_l
$\mathcal{L}_f$	Set of index of the elements of the Lagrangian mesh
$\mathcal{L}_v$	Set of index of the vertices of the Lagrangian mesh
σ_e	Faces of the Eulerian dual mesh
σ_l	Faces of the Lagrangian mesh
$x_{l,i}$	Vertices of face σ_l for $i = 1, d$
Σ_h	Piecewise linear approximation of Σ such as $\Sigma_h = \left\{\sigma_l \in \mathbb{P}_1^{d-1}, l \in \mathcal{L}_f\right\}$
$C(x_I)$	Phase ratio in the control volume centered in x_I
d_I	Unsigned distances between Eulerian points x_I and the interface Σ_h