The algebraic immersed interface and boundary method for elliptic equations with discontinuous coefficients

Arthur Sarthou, Stéphane Vincent, Philippe Angot, Jean-Paul Caltagirone

The algebraic immersed interface and boundary method for elliptic equations with discontinuous coefficients

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

May 31, 2009

[1] *Université de Bordeaux, Laboratoire TREFLE, UMR-CNRS 8508, ENSCPB, 16 avenue Pey Berland, Pessac Cedex, F33607, France*
[2] U*niversité de Provence, Laboratoire LATP-CMI, UMR-CNRS 6632, Technopôle Château-Gombert, 39 rue F. Joliot Curie, 13453 Marseille Cedex 13, France*
[3] *CNRS, France*

Keywords: Fictitious domain, Immersed interface method, Penalty methods, Heat transfert, Finite volumes, Elliptic equations, Immersed boundary method.

Abstract

A new immersed interface method, the algebraic immersed interface and boundary (AIIB) method, is presented for elliptic equations with immersed interface problems. This method allows accurate discontinuous conditions on immersed boundaries and interfaces to be imposed. The main idea is to create auxiliary unknowns at existing grid locations in order to increase the degrees of freedom of the initial problem. These auxiliary nodes allow to impose 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 at 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 or unstructured grid code. Several validation problems are presented to demonstrate the interest and accuracy of the method.

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 numerical domain does not match the physical domain. The advantages of this approach are numerous. Standard discrete operators are simple, especially in finite volume methods, grid generation is trivial, and furthermore there is no need to remesh the discretization grid in the case of moving 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 corresponding to two classes of method can be drawn : the immersed boundary problems and the immersed interface problems. The firsts deal 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 on 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 the following model scalar immersed boundary problem with a Dirichlet boundary condition (BC) on the interface Σ (see Fig. 1):

$$\mathcal{P}_{ib} \quad \left\{ \begin{array}{rl} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega_0 \\ u_{|\Sigma} = u_D & \text{on } \Sigma \end{array} \right.$$

Some boundary conditions are also imposed on the other part of the boundary $\partial\Omega_0$ so the whole problem is well-posed.

A first approach dealing with immersed boundaries is the distributed Lagrangian Multiplier method proposed by Glowinski [9]. Used for the Navier-Stokes (NS) equations with finite-element methods, the coupling between

fluid and solid media is ensured thanks to Lagrange multipliers introduced into the weak formulation of the NS equations. The method is second order accurate in space but only globally and does not locally preserves mass and momentum.

Cartesian grid methods [13, 20] propose using a structured grid in the whole domain except near obstacles where unstructured cells are created from structured cells. This method is quite 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 methods (IBM) was initially presented by Peskin [23, 24]. Fictitious boundaries are taken into account through a volumic source term defined only near the boundaries. As the source term is weighted with a discrete Dirac function with 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) methods, was initially proposed by Mohd-Yusof [21]. The idea here is to impose a no-slip condition directly on the boundary using a mirrored flow over the boundary. In [3, 35], 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 [33], Tseng et al. use the same principle but extrapolate the solution in ghost cells inside the boundary. This approach can be seen as a generalization of the mirror boundary conditions used in Cartesian Staggered grids to impose a Dirichlet condition on pressure nodes. As discussed in [28], this kind of approach seems more accurate than [3, 35].
Originally presented in [1], 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 volumic penalty method (VPM) ([14] and the references therein) requires the addition of a penalty term $b(u-u_D)$ in the conservation equations, such that:

$$\begin{cases} -\nabla \cdot (a\nabla u) + b(u - u_D) = f \text{ in } \Omega \\ \text{with } b_{|\Omega_0} = 0, b_{|\Omega_1} = \frac{1}{\varepsilon}, \text{ for } 0 < \varepsilon \ll 1 \end{cases} \tag{1}$$

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. The Darcy penalty method ([15]) consists in adding the Darcy term $\frac{\mu}{K}\mathbf{u}$ to the 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 $\mathbf{u} = 0$. Classical penalty methods are of first order only since they consider the projected shape of the interface on the Eulerian grid to define the penalty parameters [25]. In [28, 29], Sarthou et al. have extended penalty methods to higher orders by modifying the expression of $b(u - u_D)$ using implicit interpolations as in [33]. This new method is called the sub-mesh penalty (SMP) method, and has been applied to Navier-Stokes equations with augmented Lagrangian velocity/pressure coupling [4].
Applied to problem $\mathcal{P}$, the ghost cell immersed boundary method [33] and SMP method [29] used the first cells in Ω_1 to enhance the accuracy of the solution in Ω_0. The solution is coherent in Ω_0 only, so these methods are suitable for immersed boundary problems, but not for immersed interface problems. As the solution in Ω_1 is not relevant, nodes in Ω_1 can be considered as auxiliary nodes.
A not so different approach is considered in [7, 8] 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.

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

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

A first class of method is the immersed interface methods (IIM) initially introduced by LeVeque and Li [17]. This groupe of methods use 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 application, such as moving interfaces [11] or Navier-Stokes equations [16]. In [18], Li uses an augmented 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.

The Ghost Fluid Method, originally developed by Fedkiw et al. [5, 19], introduces ghost nodes where the solution is extended from one side of the interface to the other. 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 [37, 38, 36] 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 [18], 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 when a more complicated weighted least squares approach is used in [18] to discretize additional equations.

The present method solves elliptic problems using an augmented method coupled with an auxiliary node approach (contrary to ghost nodes, auxiliary nodes are present in the linear system). Hence, compact interpolations are used to discretize additional interface constraints. The method is very 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. 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 method

The algebraic immersed interface (AIIB) method is now presented. The method for immersed boundary problems, when a Dirichlet or a Neumann boundary condition is required, is first formulated. 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

Let us consider the original domain of interest denoted by Ω_0, typically the fluid domain, which is embedded inside a simple computational domain $\Omega \subset \mathbb{R}^d$, d being the spatial dimension of the problem. The auxiliary domain Ω_1, typically a solid particle or an obstacle, is such that : $\Omega = \Omega_0 \cup \Sigma \cup \Omega_1$ where Σ is an immersed interface (see Fig. 1). Let $\mathbf{n}$ be the unit outward normal vector to Ω_0 on Σ. Our objective is to numerically impose the adequate boundary conditions on the interface Σ. These conditions will be discretized in space on an Eulerian structured mesh covering Ω. As the discretization of the interface or boundary conditions require interpolations, we define: $\mathbb{L}^1_1(x) = \alpha_1 + \alpha_2 x$, $\mathbb{P}^2_1(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y$ and $\mathbb{Q}^2_1(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy$. As can be seen, the superscript is the dimension of the interpolation while the subscript is the order.

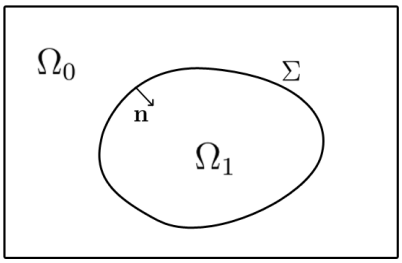

Figure 1: Definition of the domains and the interface

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 index of the Eulerian orthogonal curvilinear structured

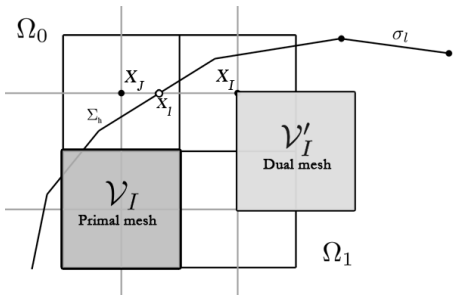

Figure 2: Definition of the discretization kernels for the AIIB method

mesh. Let x_I be the vector coordinates of the center of each volume $\mathcal{V}_I$. The local space step of the volume $\mathcal{V}_I$ defined as the maximum length of $\mathcal{V}_I$ in each direction is denoted by h_I, whereas h denotes the Eulerian mesh step: $h = \sup_{I\in\mathcal{E}} h_I$. This grid is used to discretized 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 : $\Sigma_h = \{\sigma_l \in \mathbb{P}_1^{d-1}, l \in \mathcal{L}_f\}$, K being the cardinal of $\mathcal{L}_f$ and $\mathcal{L}_f$ being the set of index of the Lagrangian mesh. 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 $i = 1, d$ and the set of all vertices is : $\{x_l, l \in \mathcal{L}_v\}$. 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 Fig. 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.
The projection of the Lagrangian mesh on the Eulerian grid generates the following Eulerian functions :

- The discrete Heaviside function χ, defined as :

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

This function is the basic indicator of the presence of an Eulerian point in Ω_1 and is build 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 :

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

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_i, 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 directly obtained from the ϕ function by using the formula proposed by Sussman and Fatemi [32] :

$$C_i(x) = \begin{cases} 1 \text{ if } \phi(x) > h \\ 0 \text{ if } \phi(x) < -h \\ \frac{1}{2}(1 + \frac{\phi}{h} + \frac{1}{\pi}\sin(\pi\phi/h)) \text{ otherwise} \end{cases} \tag{4}$$

New sets of Eulerian points x_I are defined near the interface so that each one has a x_J neighbor verifying $\chi_J \neq \chi_I$, *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 $\mathcal{N}_0 = \{I, x_I \in \Omega_0, \chi_I \neq \chi_J, x_J \in \Omega_1\}$ and $\mathcal{N}_1 = \{I, x_I \in \Omega_1, \chi_I \neq \chi_J, x_J \in \Omega_0\}$.
Let us now define *auxiliary* entities which are superscripted with $*$, corresponding to *physical* interface entities, point, control volume or unknown. Hence, two sets of points are defined: $\{x_I^*, I \in \mathcal{N}_0^*\}$ with $\mathcal{N}_0^* = \{I, x_I^* \in \Omega_0, \chi_I \neq \chi_J, x_J^* \in \Omega_1\}$ and $\{x_I^*, I \in \mathcal{N}_1^*\}$ with $\mathcal{N}_1^* = \{I, x_I^* \in \Omega_1, \chi_I \neq \chi_J, x_J^* \in \Omega_0\}$. It is important to note that physical and auxiliary locations are the same, $x_I = x_I^*$ and $\mathcal{V}_I = \mathcal{V}_I^*$. The main difference lies in the magnitude of the solution at these point, as $u(x_I) = u_I \neq u(x_I^*) = u_I^*$.

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 interpret this Lagrangian grid on the Eulerian physical grid.
A first issue is to determine which Eulerian points are inside the domain Ω_1 defined by the Lagrangian surface. In [28, 15], the authors used a global methodology partly based on [31].

A Ray Casting method based on the Jordan Curve Theorem is implemented. 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, otherwise outside. The Ray-casting method can be enhanced by classifying elements of the Lagrangian mesh with an octree sub-structure. If a ray does not intersect a cube, it does not intersect the triangles inside. A fast and simple optimization is to test if a given point is in a box bounding the Lagrangian mesh. Some details of the implementation of the method and a short review of points in solid strategies can be found in [22].

Algorithm 1 describe a pseudo-code performing a basic computation of the discrete Heaviside function χ. To avoid numerical errors due to the presence of great numbers to simulate $+\infty$, the ray is only cast to a point $x_{\infty i}$ which is far enough to be outside the object and the grid. To optimize the intersection calculation, $x_{\infty i}$ is different for all x_i and parallel to a gridline.

Concerning the ray-triangle intersection, [22] announces that the Feito-Torres [6] algorithm seems to be one of the fastest. An optimization for the Jordan-based method on orthogonal structured grids that greatly improves the performances of the algorithm is now proposed. In the Jordan based-method, Ray direction is indifferent. If all rays are launched in the same direction, Ox for instance, many intersection tests are done more than once for a set of points in a same Eulerian mesh row in the Ox direction. Hence, only one ray can be cast per row. If rays are cast in a given direction (the best choice is the one with the most cells), computational cost is divided by the number of cells in this direction.

Alg. 2 now describes an enhanced version of algorithm 1. Rays are cast from points x_i included in a boundary slice $\mathcal{S}_{xy}$ of the Eulerian mesh. For each starting point x_i, the intersections are stored and sorted according to their z component in a two entry structure $PTZ(i, nsect_i)$. For each $x_i \in \mathcal{S}_{xy}$,

Algorithm 1 Simple computation of the discrete Heaviside function

```
for i = 1, m do
  nsect := 0
  for k = 1, K do
    if Segment [x_i; x_∞i] intersects σ_k then
      nsect := nsect + 1
    end if
  end for
  if nsect is even then
    χ(x_i) := 0
  else
    χ(x_i) := 1
  end if
end for
```

nsect, the number of intersection by rows, is not known *a priori*. If PTZ is an array, a first pass has to be performed to determine the size of PTZ. A better choice is to use chained lists.

For the sake of clarity, the algorithm is not the fully optimized one (no bounding box test for instance), and can be extended to octree approaches for example.

The binary χ_i functions obtained are used to build an Eulerian Level-Set function near the interface by estimating the distance between the Eulerian points and the neighboring Lagrangian points.

The Lagrangian points used to couple the Lagrangian surface of the complex object and the Eulerian grid used to solve the conservation equations are the points x_l of Σ_h. Two different methods presented below to determine these points are used.

Algorithm 2 Optimized computation of the discrete Heaviside function in 3D

for $i = 1, m$ with $x_i \in \mathcal{S}_{xy}$ **do**
  $nsect := 0$
  for $k = 1, K$ **do**
    if Segment $[x_i; x_{\infty i}]$ intersects σ_k **then**
      Store the intersection in $PTZ(i, nsect)$
      $nsect := nsect + 1$
    end if
  end for
  if nsect is even **then**
    $\chi(x_i) := 0$
  else
    $\chi(x_i) := 1$
  end if
  $In\_state := boolean(\chi(x_i))$
  $nsect_{tmp} := 0$
  for $j = 1, m_z$ **do**
    while $nsect_{tmp} < nsect$ and $x_j(3) > PTZ(i, nsect_{tmp})$ **do**
      Switch In_state
      $nsect_{tmp} := nsect_{tmp} + 1$
    end while
    $\chi(x_j) := In\_state$
  end for
end for

2.3 The algebraic immersed interface and boundary method for immersed boundary problems

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 (an immersed boundary or an immersed interface). The sub mesh penalty (SMP) method [28] has been previously used to treat immersed boundaries, but cannot deal with immersed interfaces. The AIIB method is an enhancement of the SMP method [29] 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 the ghost point method [7, 8] and the immersed interface methods [17]), the AIIB method is quite simple to implement.

Let $\mathcal{P}$ be a model problem discretized 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 equations to the initial linear system so as to take into account additional interface constraints. Our approach here is to increase the number of unknowns to obtained a new square matrix. The new unknowns, so-called the auxiliary unknowns and labeled with $*$, are considered as the extrapolation of the solution from one side of the interface to the other, and are used to discretize interface conditions. The initial algebraical link between unknowns from both sides of the interface is cut, and the new link over the interface is performed thanks to auxiliary unknowns. Hence, the orignal 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 interface conditions. The solution u' is decomposed such as $u' = (u, u^*)^T$ and the source term as $b' = (b, b^*)^T$. Interface constraints are discretized in a $(n, m + n)$ matrix C.

According to the interface conditions, the regularity of the solution through 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). Practically, matrix coefficients must be modified to take into account the new connectivities. Let $\alpha_{I,J}$ be a coefficient of A at line I, row J and $\alpha'_{I,J}$ the new coefficient in A'. If $I, J \in \mathcal{N}$ and $C_I \neq C_J$, $\alpha'_{I,J} = 0$ and $\alpha'_{I^*,J} = \alpha_{I,J}$. This modification can be performed thanks to permutation and mask matrixes. The initial matrix A is extended to $A_E = I_1 A$ with I_1 a $(m, m + n)$ matrix and C is extended to $C_E = I_2 C$ with I_2 a $(n, m + n)$ matrix, such as

$$I_1 = \begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & & \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 & & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}, \quad I_2 = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & & 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & & \ddots & 0 \\ 0 & \dots & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{5}$$

A_0 and A_1 are defined such as $A_0 + A_1 = A$, $A_0(i,j) = A(i,j)$ if $i \in \mathcal{N}_0$ and $A_1(i,j) = A(i,j)$ if $i \in \mathcal{N}_1$. Finally, connectivities are changed using 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 defined as:

$$A' = I_1^T(P_0 A_0 I_1 + P_1 A_1 I_1) + I_2^T C \tag{6}$$

The new problem is $A'u' = b'$. A' is now cut in 4 new sub-matrices of various size: $\tilde{A}(m,m)$, $B(m,n)$, $C_1(n,m)$, $C_2(n,n)$. $\tilde{A}$ is the modification of the initial matrix A, C_1 and C_2 are two sub-parts of the initial matrix C. The problem can be written as:

$$\begin{pmatrix} \tilde{A} & B \\ C_1 & C_2 \end{pmatrix} \begin{pmatrix} u \\ u^* \end{pmatrix} = \begin{pmatrix} b \\ b^* \end{pmatrix} \tag{7}$$

The entire problem can be solved as it is to obtain $u' = (u, u^*)^T$. However, u^* is the solution at the auxiliary nodes and is generally not required. Hence, the Schur complement method can be used to obtain the solution for the physical nodes only. The final problem is:

$$(\tilde{A} - BC_2^{-1}C_1)u = b - BC_2^{-1}u^* \tag{8}$$

The opportunity of such a reduction will be discussed later.

2.3.2 AIIB algorithm for a scalar equation with Dirichlet boundary conditions

For sake of clarity, let us first describe in $2D$ the AIIB method for the model scalar problem $\mathcal{P}$ 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 nodes are created in Ω_1 only. Let us consider a point x_I with $I \in \mathcal{N}_0^*$. 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 (Fig. 1 right). Then, the solution u_l at the interface is approximated by the $\mathbb{P}_1^1$ interpolation between the Eulerian unknowns u_I^* and u_J :

$$u_l = \lambda_I u_I^* + \lambda_J u_J \text{ with } 0 < \lambda_I, \lambda_J < 1 \text{ and } \lambda_I + \lambda_J = 1 \tag{9}$$

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. We choose x_p, a new point of Σ_h between x_l and x_m (see Fig. 3 left). The solution $u_p(x_p)$ is then approximated using a $\mathbb{P}_1^2$-interpolation of the values u_I^*, u_J and u_K :

$$u_p = \lambda_I u_I^* + \lambda_J u_J + \lambda_K u_K \ , \ 0 < \lambda_I, \lambda_J, \lambda_K < 1 \ , \ \lambda_I + \lambda_J + \lambda_K = 1 \quad (10)$$

A $\mathbb{Q}_1^2$ interpolation of u_I, u_J, u_K and u_L can be 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 Fig. 3 left). As a third choice, two independent linear 1D interpolations can be used (one for each direction) for an almost equivalent result. It produces :

$$\begin{cases} u_l = \lambda_I u_I^* + \lambda_J u_J \text{ with } 0 < \lambda_I, \lambda_J < 1 \text{ and } \lambda_I + \lambda_J = 1 \\ u_m = \lambda'_I u_I^{*'} + \lambda_K u_K \text{ with } 0 < \lambda'_I, \lambda_K < 1 \text{ and } \lambda'_I + \lambda_J = 1 \end{cases} \quad (11)$$

In this case, two auxiliary unknowns are created.

A simple choice for x_p is the barycenter between x_l and x_m where $u_p = (u_l + u_m)/2$. This particular case enables an easy implementation since we have :

$$\lambda_I u_I^* + \lambda_J u_J = u_l \quad (12)$$

$$\lambda'_I u_I^* + \lambda_K u_K = u_m \quad (13)$$

A summation of these two constraints gives :

$$\lambda_I u_I^* + \lambda_J u_J + \lambda'_I u_I^* + \lambda_K u_K = u_l + u_m \quad (14)$$

what is equivalent to the constraint in u_m built with $\mathbb{P}_1^2$ interpolation :

$$\frac{(\lambda_I + \lambda'_I)u_I^* + \lambda_J u_J + \lambda_K u_K}{2} = u_p \ , \ 0 < \frac{\lambda_I + \lambda'_I}{2}, \frac{\lambda_J}{2}, \frac{\lambda_K}{2} < 1 \ , \ \frac{\lambda_I + \lambda'_I}{2} + \frac{\lambda_J}{2} + \frac{\lambda_K}{2} = 1$$

Hence, an easy implementation consists in summing the penalty terms corresponding to each direction. This method can be applied to unstructured grids were x_I can have many neighbors. 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 high/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 maximum second order precision of

our discretization. The convergence of this additional error is numerically tested in section (3.3.1). If Σ_h is regular enough, x_I almost never has a third or a fourth neighbor in Ω_0. However, if it is the case, the L^2-penalty term $b(u_I - u_D(x_I))$ is used. 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. Hence, the present method is suitable to impose a Dirichlet boundary condition on Σ for Ω_0, when the solution in Ω_1 has no interest. The solution $u_I(x_I)$ for $I \in \mathcal{N}_1$ is an extrapolation of the solution in Ω_0 through Σ and so is non-physical. It is important to notice that u_I depends only on the solution in Ω_0. Hence, the solution at the nodes of Ω_1 far from the interfaces does not impact on the solution in Ω_0. Nevertheless, the fictitious domain approach induces the estimate of the solution in Ω_1. It is naturally obtained with the initial set of equations, and for nodes in this domain, a penalty method such as VPM [14] can be used. The imposed solution can be analytical when possible, or a constant value. The uselessness of the computation of the solution for these nodes can be treated numerically by switching the solving of $u(x), x \in \{\Omega_1\}$ off, or by totally removing these nodes in the solving matrix.

2.3.3 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 :

$$\begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega_0 \\ (a \cdot \nabla u) \cdot \mathbf{n} = g_D & \text{on } \Sigma \end{cases} \tag{15}$$

The principle is about the same as for Dirichlet BC, and the same interpolations, once derived, can be used to approximate the quantity $(a \cdot \nabla u) \cdot \mathbf{n}$. Hence,

$$(a \cdot \nabla u_l) \cdot \mathbf{n} \approx (a \cdot \nabla p(x_l) \cdot \mathbf{n}) \tag{16}$$

For $p \in \mathbb{Q}_1^2$, $\nabla p(x,y) \cdot \mathbf{n} = (a_3 y + a_2) n_x + (a_3 x + a_1) n_y$ is obtained whereas for $p \in \mathbb{P}_1^2$, $\nabla p(x,y) \cdot \mathbf{n} = a_2 n_x + a_1 n_y$ is obtained which means that the normal gradient is approximated as being constant over the whole support. Practically, the absolute value of the norm is used. For $p \in \mathbb{P}_1^2$, we have :

$$\nabla p(x,y) \cdot |\mathbf{n}| = \frac{u_J - u_I}{h_x}|n_x| + \frac{u_K - u_I}{h_y}|n_y| = u_I(-\frac{1}{h_x}|n_x| - \frac{1}{h_y}|n_y|) + u_J \frac{1}{h_x}|n_x| + u_K \frac{1}{h_y}|n_y| \tag{17}$$

If $n_x \approx -n_y$ and $h_x \approx h_y$, the diagonal coefficient of the line related to $u_I \approx 0$ which leads to numerical instabilities. Hence, we use $|\mathbf{n}|$ instead of $\mathbf{n}$ to avoid numerical instabilities, and the source term of the flux constraint has to be modified according to the sign changes of each component of the normal. In [33] a similar interpolation is prone to induce the same instabilities. However, the way the authors avoid such instabilities is not mentioned. 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 suits 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 Fig. 3 right). This point is a neighbor of x_J and is taken as $[x_I, x_J] \perp [x_J, x_K]$. As in 2D two choices generally appear, the point being so that the angle $(\mathbf{n}, x_K - x_J)$ is in $[-\pi/2; \pi/2]$ is taken.

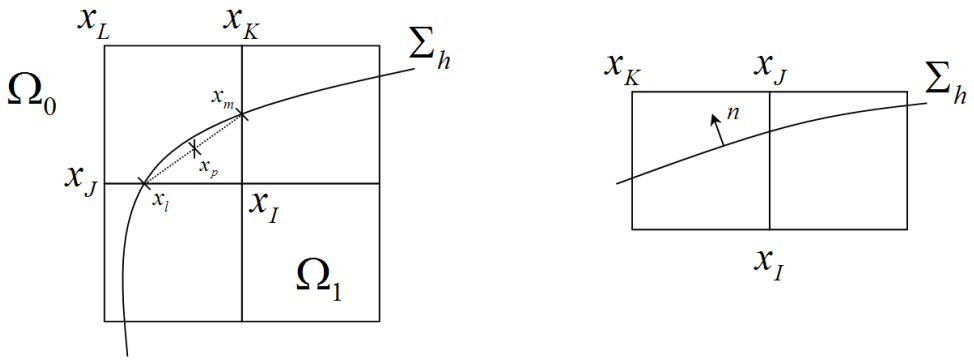

Figure 3: Example of selection of points for Dirichlet (left) and Neumann (right) constraints

2.3.4 Symmetric version for Dirichlet interface conditions

The next step is to allow multiple Dirichlet boundary conditions on both sides of the immersed interface. This method allows for instance to treat thin objects. The problem is now :

$$\begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega \\ u^-_{|\Sigma} = u_D & \text{on } \Sigma \\ u^+_{|\Sigma} = u_G & \text{on } \Sigma \end{cases} \tag{18}$$

This problem cannot be solved with the SMP method which uses nodes in Ω_1 to increase the accuracy of the solution in Ω_0. The problem (18) requires a physical solution on both sides of the interface. For each node and their unknowns near the interface, two properties are required (for instance, the unknown u_I at a node x_I in Ω_1 is taken) :

- As the physical solution is required in the whole domain, u_I must be physical too
- The solution u_I must be an extrapolation of the solution in Ω_0 through the interface and cannot be physical

These two requirements are incompatibles, so the standard SMP method is not suitable for problems with immersed interfaces. The solution is to create auxiliary nodes near the interface. Hence, if u_I is an unknown in Ω_0 near the interface with a neighbor in Ω_1, an auxiliary unknown at x_I denoted u_I^* is created. Let us consider a node x_I in Ω_0 with a neighbor x_J in Ω_1. With a standard numerical resolution, u_I and u_J are algebraically connected in the solved matrix. As a Dirichlet boundary condition is imposed at Σ, the connection between u_I and u_J, which are physical unknowns, is irrelevant. Hence, this connection is broken, and the new neighbor of u_I (resp. u_J) is now u_J^* (resp. u_I^*).

Practically, 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 in Ω_1. As a second step, the color function is modified as $C := 1 - C$ 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.

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 such as only auxiliary node is needed:

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

The Schur complement of C_2, $(\tilde{A} - BC_2^{-1}C_1)$ is easy to determine as C^2 is a diagonal matrix. Practically, if the algebraic reduction is considered, $\tilde{A}$ is not built. The part $-BC_2^{-1}C_1$ is directly added to A, and $-BC_2^{-1}b^*$ is added to

b. As will be subsequently demonstrated, the algebraic reduction decreases by $10-20\%$ the computational cost of the solver.

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 with [7]. 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. The present algorithm seems simpler, as the standard discretization of the operators is automatically modified in an algebraic manner. So, no particular attention has to be paid to the discretization scheme used.

If $\mathbb{P}_1^2$ interpolations are used, the computed solution is the same as for the SMP [29] method and the DF-IB method [33]. These methods change the discretization of the initial equation for the nodes in $\mathcal{N}_1$. 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 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 in $\mathcal{N}_0$ is modified, and without algebraic reduction, auxiliary nodes in $\mathcal{N}_1^*$ are superimposed with nodes in $\mathcal{N}_1$.

The accuracy of these methods will be discussed in the next section.

2.3.6 A word on the application to the Navier-Stokes equations

The SMP method has been applied to the Navier-Stokes equations in [29]. For immersed boundary problems, the SMP and the AIIB methods give equivalent results and the AIIB method can be used to immerge obstacles in fluid flows. Both methods can be used with the Navier-Stokes equations and scalar equations, the difference being that for the Naviers-Stokes equations the procedure is independently repeated for each component of the velocity. However, the AIIB method with $\mathbb{L}_1^1$ interpolations only cannot be applied to Navier-Stokes equations. An illustration is given Fig. 4. With such interpolations, two auxiliary unknowns u_I^* and $u_I^{*'} \in \mathcal{N}_1^*$ can coexist in a same location x_I. Hence, u_I^* is the natural neighbor of u^J and $u_I^{*'}$ is the natural neighbor of u^K. A problem occurs for the discretization of the inertial term as a node of a given velocity component has to use an auxiliary point of an other velocity component. In this case, neither u_I^* or $u_I^{*'}$ is a natural neighbor for v_l, a velocity unknown in the y direction. Whether a point, its neighbor in the other subdomain or an averaging of the two is taken, it does not

work. Particular attention has to be given to the velocity pressure coupling.

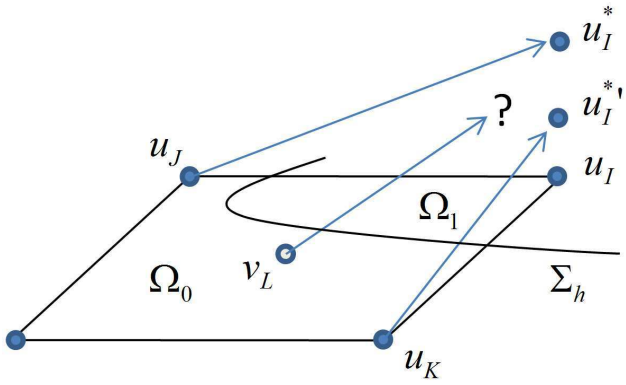

Figure 4: Illustration of the application to the Navier-Stokes equation

SMP and direct-forcing immersed boundary methods are not well suited for time splitting methods, as they basically modify the prediction step only. However, the correction step, if not modified, changes the velocity without taking into account the interface constraint [2]. Generally, the authors have made minor modifications or consider that the modification of the velocity is negligible. In fact, the projection step has to be rewritten considering the forcing term, as can be seen in [12]. In [29], the authors use an augmented Lagragian method which add a Lagrange multiplier in the predictor step. As this multiplier enforces the incompressibility, no correction step is required.

2.4 AIIB for immersed interface problems

With the precedent symmetrical method, the physical problem can be solved on both sides of the interface, and an explicit Dirichlet BC is imposed on the interface. As for many problems, the solution is not *a priori* known on the interface, the final step of our method is to allow transmission or jump conditions on the interface Σ. Now, the problem is :

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

where interface conditions are :

$$[\![u]\!]_\Sigma = \varphi \quad \text{on } \Sigma \tag{20}$$

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

The notation $[\![\ ]\!]_\Sigma$ denotes the jump of a quantity over the interface Σ. As can be seen with the symmetric version of the AIIB method, a given intersection point $x_l, l \in \mathcal{I}$, can be associated with two auxiliary unknowns. In the last algorithms, auxiliary unknowns were associated to Dirichlet or Neumann BC, but each auxiliary unknown can be related to any type of constraint, especially jump or transmission constraints. Hence, an intersection point can be associated to the two interface constraints (20) and (21) of $(\mathcal{P}_{ii})$. For instance, the I^{nth} line of the matrix with $x_I^* \in \Omega_0$ can be used to impose the constraint (20) and the J^{nth} line of the matrix with $x_J^* \in \Omega_1$ is then used to impose constraint (21).

2.4.1 The solution constraint

The 1D symmetrized AIIB methods for Dirichlet BC holds :

$$\left\{ \begin{array}{l} u_\Sigma^+ = \alpha u_I + \beta u_J^* \\ u_\Sigma^- = \alpha u_I^* + \beta u_J \end{array} \right. \tag{22}$$

One can notice that the interpolation coefficients α and β are the same for both constraints due to the superimposition of the implied nodes. As $[\![u]\!]_\Sigma = u_\Sigma^+ - u_\Sigma^- = \varphi$, we obtain :

$$\alpha u_I + \beta u_J^* - \alpha u_I^* - \beta u_J = \varphi \tag{23}$$

which is the first constraint to be imposed.

2.4.2 The flux constraint

Following the same idea as for the solution constraint, and using $\mathbb{P}_1^2$ interpolation,

$$\left\{ \begin{array}{l} (a \cdot \nabla u_\Sigma^+) \cdot |\mathbf{n}| = a^+ (\frac{u_J^* - u_I}{h_x}|n_x| + \frac{u_K^* - u_I}{h_y}|n_y|) \\ (a \cdot \nabla u_\Sigma^-) \cdot |\mathbf{n}| = a^- (\frac{u_J - u_I^*}{h_x}|n_x| + \frac{u_K - u_I^*}{h_y}|n_y|) \end{array} \right. \tag{24}$$

and using (21), we obtain:

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

which is the second constraint to be imposed. Contrary to the solution constraint, the precise location of the interface is not taken into account. However, as demonstrated later, the second order in space is possible to reach on Cartesian grids when $\psi = 0$.

2.4.3 Algebraic reduction

Each constraint written to impose an interface constraint involves more than one auxiliary unknown, so contrary to the immersed boundary constraints, C_2 is not a diagonal matrix and a solver has to be used to compute C_2^{-1}.

For the matched interface and boundary (MIB) method, Zhou et al. [37] used a different discretization of the interface conditions which allows an easy algebraical reduction which is directly performed line by line.

The algebraic reduction for the immersed interface problems has not been implemented yet. However, the standard discretization of the AIIB method requires a more compact stencil than for the MIB method, and the additional computational time generated by the auxiliary nodes is small. Hence, the lack of algebraic reduction does not seem to be problematic.

2.4.4 Grid intersections and interface localization

Two main approaches are generally used to locate an immersed interface. The level-set function indicates the signed distance of an Eulerian point from the interface and is often coupled with the Ghost Fluid method. Using this Eulerian function, the intersection between an edge of $\mathcal{V}_i'$ and the interface can be quite accurately determined. For example, 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, $x_l = (x_I d_J + x_J d_I)/(d_I + d_J)$ is obtained. If the Level-Set is calculated not only to find the intersections, this approach has no additional cost.
A second method calculates the geometric intersection between an edge of $\mathcal{V}_i'$ crossing the interface and the interface. This last method is more accurate than the level-set method but can be slow if not optimized. Algorithmic problems can be encountered if the Lagrangian mesh is too complex compared to the Eulerian mesh, and for instance if a segment between two neighboring Eulerian points is crossed more than one time by the interface. Conversely, the relative lack of precision of the Level-Set produces a smoothed projection of a shape on a mesh. Hence, there cannot be more than one intersection between Σ_h and a grid segment.

The following algorithm used to find intersections conserves the robustness of the global methodology even if an interface segment crosses the interface more than one time:

Hence, taking the first intersection and then switching to another neigh-

Algorithm 3 Research of grid-interface intersections

```
n := 0
for i = 1, m do
  if C(x_i) > 0.5 then
    for all neighbors x_j of x_i do
      if C(x_j) < 0.5 then
        compute and store the first intersection between Σ_h and [x_i; x_j]
        cycle j
      end if
    end for
  end if
end for
```

bor avoids having two intersections for a given neighbor.
Except for the Dirichlet BC, the interpolations degenerates if the interface lies on an Eulerian point. Hence, the position of the Lagrangian mesh is shifted by an ϵ. With $\epsilon = 10^{-10}$ (depending on the size of the problem), the accuracy of the method is not altered. Another reason to shift the Lagangian mesh is to avoid errors during the ray-casting process. For a domain with rounded or symmetrical dimensions, for example a zero centered numerical domain, if the Lagrangian object is zero centered, the configuration where the edge between two triangles intersects a grid-line is frequent. If no attention is paid to this particular configuration, the ray-casting method can detect more or less than one intersection when crossing the interface at this neighboring edge (see Fig. 5). The slight shift of the Lagrangian object will avoid this.

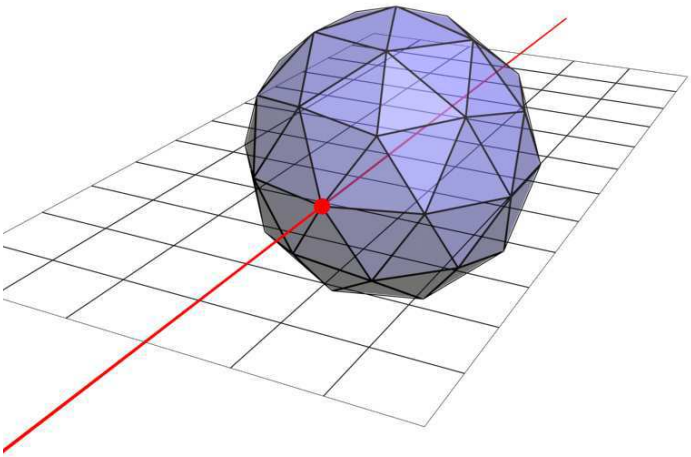

Figure 5: Illustration of a typical error during the ray-casting process. The ray intersect the interface at the common vertex of many triangles

3 Numerical results for scalar problems

Elliptic equations are discretized using the standard three point Laplacian. For all problems, similar results have been obtained with a PARDISO direct solver [30], and an iterative BiCGSTAB solver [10], preconditioned under a ILUK method [27]. Unless otherwise mentioned, a numerical domain of unit length is used for every simulation. Two discrete errors are used.
The discrete relative L^2 error is defined as:

$$\|u\|_{L^2(\Omega)} = \frac{\|u-\tilde{u}\|_{L^2_{abs}(\Omega)}}{\|\tilde{u}\|_{L^2_{abs}(\Omega)}} = \Big(\sum_{I\in\mathcal{N}} meas(\mathcal{V}_I)|u_I-\tilde{u}(x_I)|^2\Big)^{\frac{1}{2}} / \Big(\sum_{I\in\mathcal{N}} meas(\mathcal{V}_I)|\tilde{u}(x_I)|^2\Big)^{\frac{1}{2}} \tag{26}$$

where $\mathcal{N} = \mathcal{N}_0 \cup \mathcal{N}_1$ and $\tilde{u}$ is the analytical solution.
The discrete L^∞ error is defined as:

$$\|u\|_{L^\infty(\Omega)} = max_{I\in\mathcal{N}}|u_I - \tilde{u}x_I| \tag{27}$$

One can notice that only Ω_0 is taken into account for the immersed boundary problems.

3.1 Immersed boundary problems

3.1.1 Problem 1

The homogenous 2D Laplace equation is solved. Interface Σ is a centered circle of radius $R_1 = 0.5$ with a Dirichlet condition of $U_1 = 10$. An analytical solution which 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:

$$u(r) = \frac{U_2 - U_1}{ln(R_2) - ln(R_1)} ln(r) + U_1 - (U_2 - U_1)\frac{ln(R_1)}{ln(R_2) - ln(R_1)} \tag{28}$$

Accuracy tests are performed with $\mathbb{L}^1_1$, $\mathbb{P}^2_1$ and $\mathbb{Q}^2_1$ interpolations. Fig. 6 shows the solution and the error map for a 32×32 mesh with $\mathbb{P}^2_1$ interpolation. The same results are always obtained with and without algebraic reduction. Fig. 7 shows the convergence of the error for the L^2 and L^∞ norms. For all interpolations, the convergence slopes are approximatively 2 for the relative L^2 error. For the L^∞ error, the slopes are about 1.8. $\mathbb{P}^2_1$ interpolation is the more accurate, followed by the $\mathbb{L}^1_1$ interpolation although

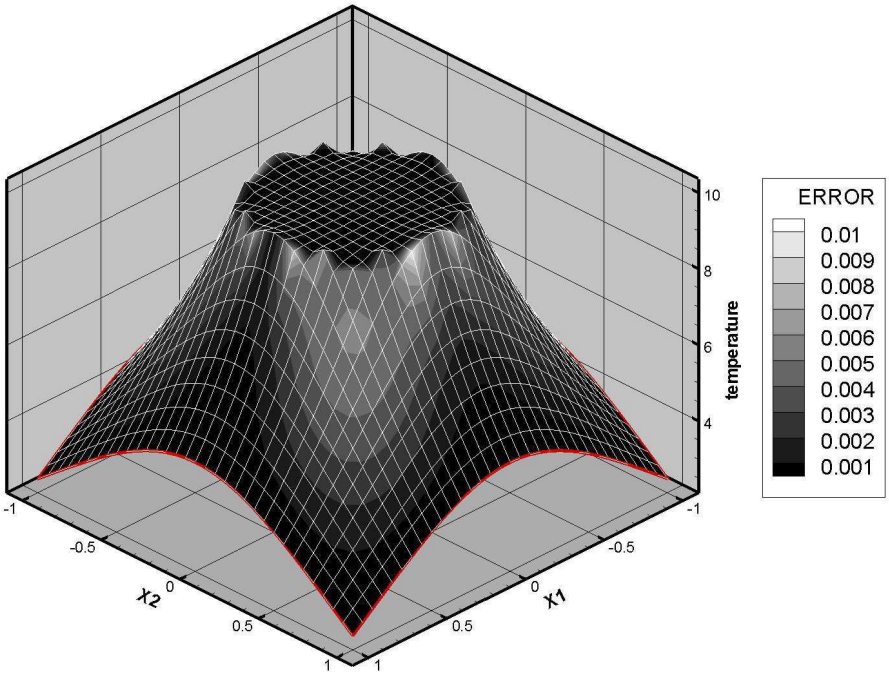

Figure 6: Solution and error map for problem 1

it uses more auxiliary points (but smaller stencils). However, the differences of accuracy between the different interpolations remain small. The same cases with algebraic reduction give the same accuracy. The performances of the ILUK-BiCG-Stab solver are now benchmarked for the three interpolations with and without algebraic reduction and for the SMP method. Tab. 1 shows the computational times of the matrix inversion (average time in seconds for 25 matrix inversions) and Tab. 2 shows the time ratio between the standard and the reduced matrix. Except for the $\mathbb{Q}_1^1$ 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 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 AIIB method with algebraic reduction. Figures 8, 9, 10, 11 shows the convergence of the ILUK-BiCG-Stab solver for the seven configurations. The type of interpolation does not significantly impact on solver performances.

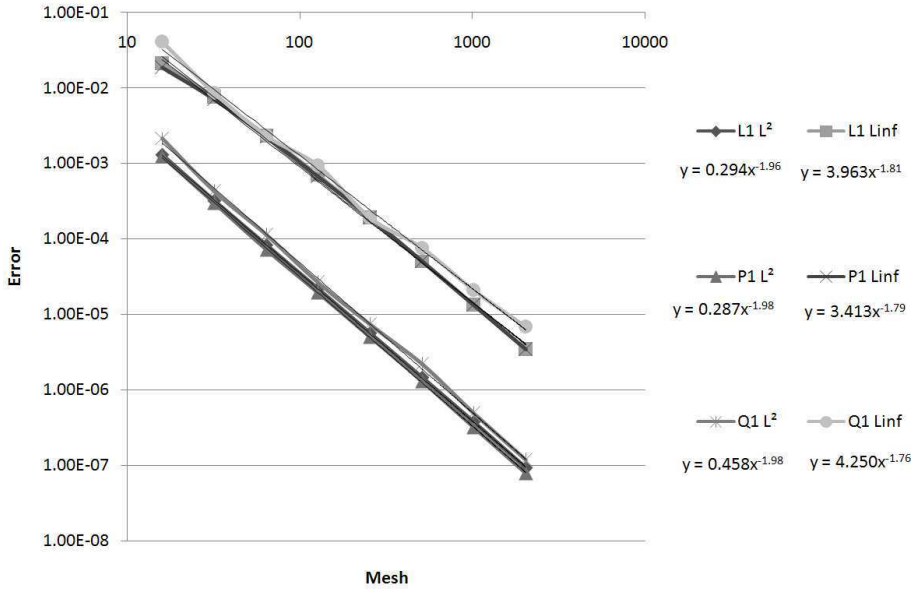

Figure 7: Curves of errors for section 3.1.1

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

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	$\mathbb{L}_1^1$	$\mathbb{P}_1^2$	$\mathbb{Q}_1^2$ 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%

Table 2: Ratio of computational times for reduced and standard matrices for section 3.1.1

3.1.2 Problem 2

The 3D equation $\Delta T = 6$ is solved in a unit box. The solution is $T(r) = r^2$. The solution is imposed on an immersed centered sphere of radius 0.2. As

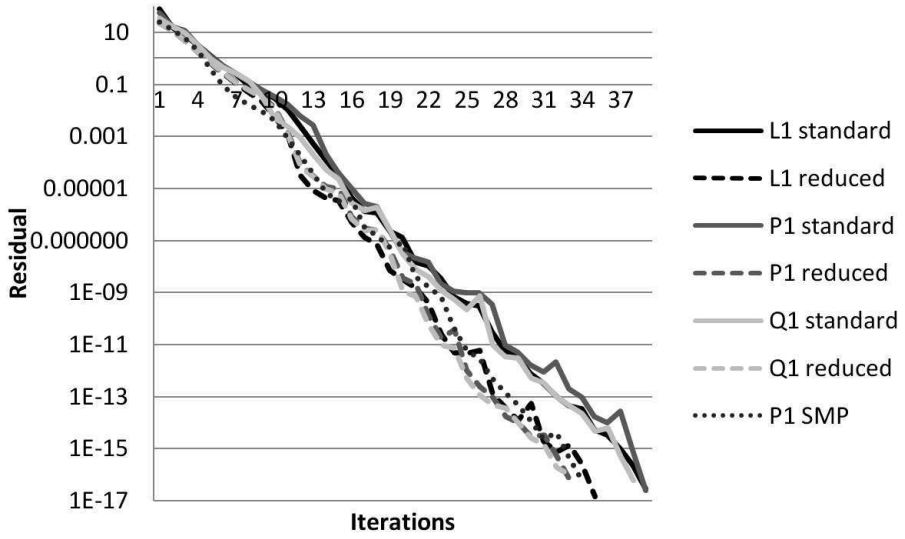

Figure 8: Residual against iterations of ILUK solver for problem 1 with a 128×128 mesh

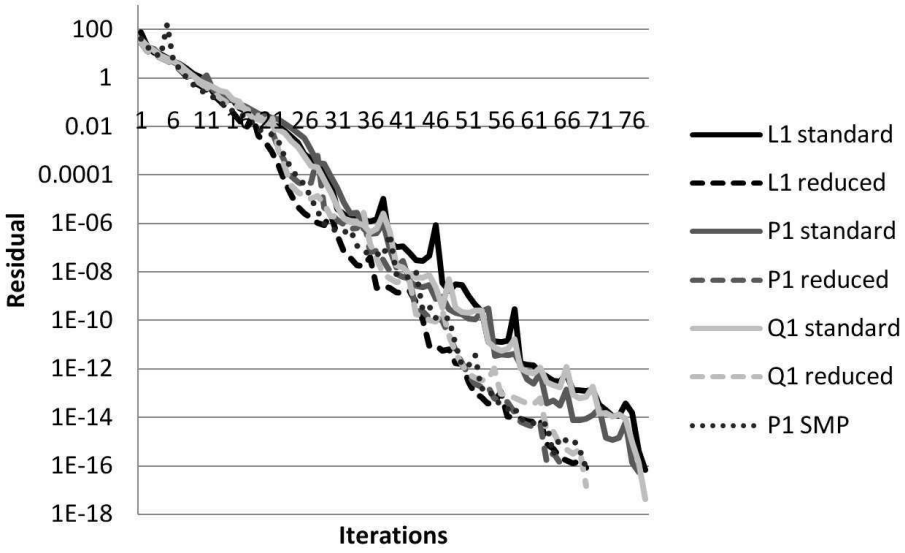

Figure 9: Residual against iterations of ILUK solver for problem 1 with a 256×256 mesh

expected, the code gives the exact solution to almost computer error accuracy

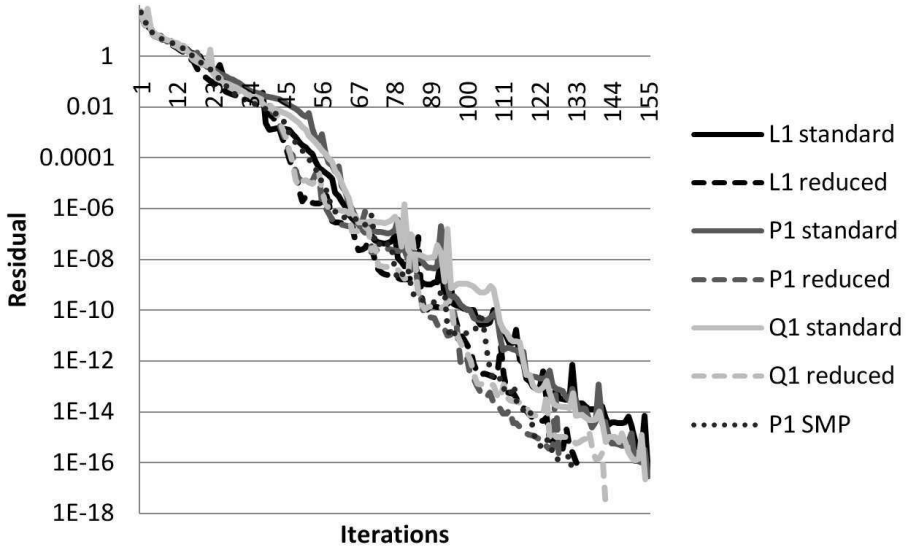

Figure 10: Residual against iterations of ILUK solver for problem 1 with a 512×512 mesh

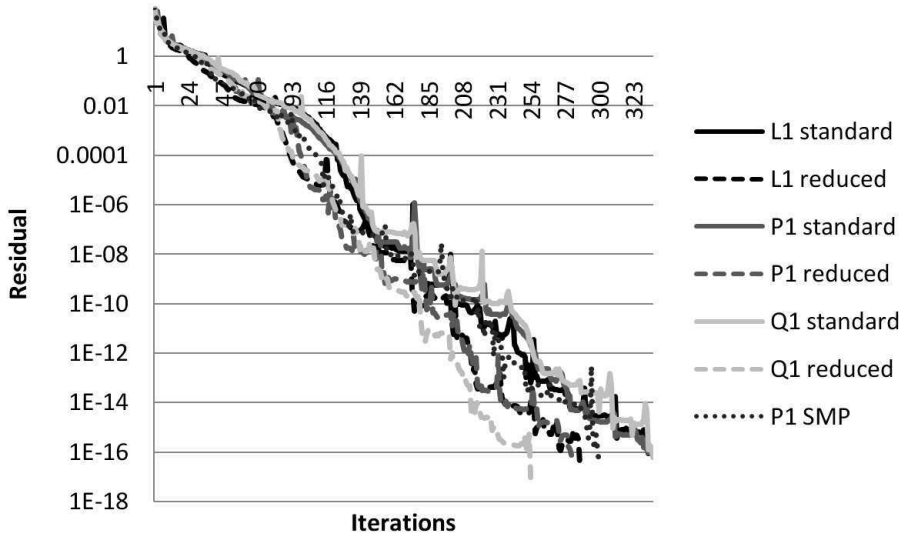

Figure 11: Residual against iterations of ILUK solver for problem 1 with a 1024×1024 mesh

without this inner boundary. Results of the numerical accuracy test with the

spherical inner boundary are presented in Fig. 12. The average slope for the

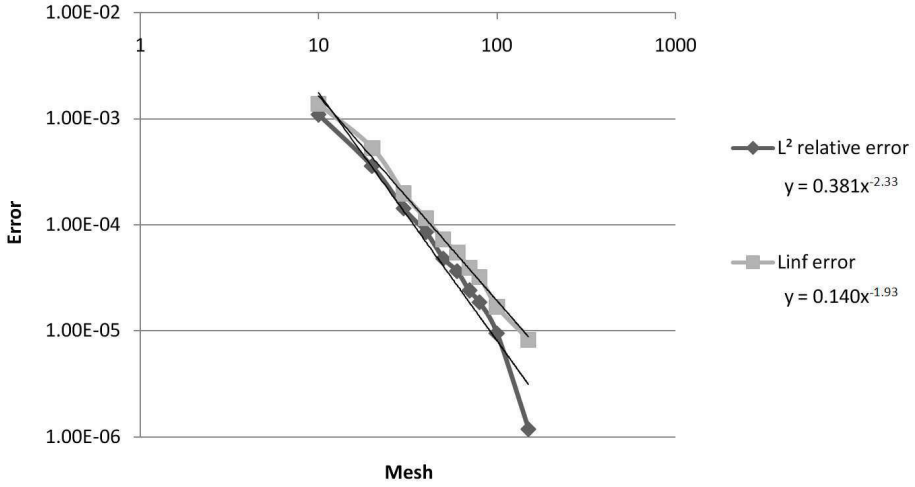

Figure 12: Curves of errors for problem 2

L^2 norm is 2.33 and increases for the denser meshes. Even if the method is of second order in space, computer error accuracy can not be expected as the immersed boundary is an approximation of a sphere.

3.1.3 Problem 3

The 3D equation $\Delta T = 12r^2$ is solved in a unit box. The solution is $T(r) = x^4 + y^4 + z^4$. The results are presented in Fig. 13. For the L^∞ norm, the second order is regularly obtained. For the L^2 norm, the second order is not obtained for the coarsest meshes as the code has not reached its asymptotical convergence domain. As can be noticed by comparing results with and without the AIIB method, this last method does not spoil the convergence order of the code, and the presence of the immersed interface with an analytical solution imposed in Σ_h improves the accuracy of the code. For both cases the numerical solution tends to an order two in space.

3.1.4 Problem 4

The 2D equation $\Delta T = 4$ is solved in a unit square. 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 Fig. 14, the global

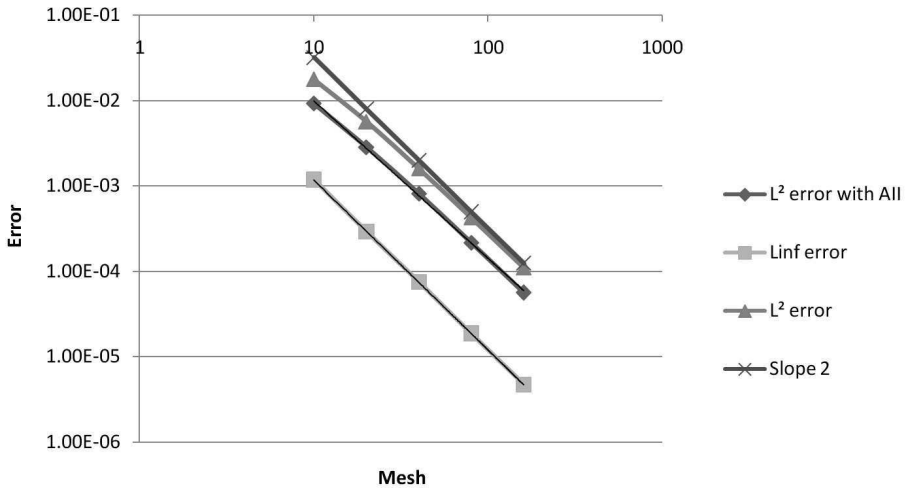

Figure 13: Curves of errors for problem 3

convergence has an average slope of 1.10. However, the convergence for the three biggest meshes reaches a slope of 2.

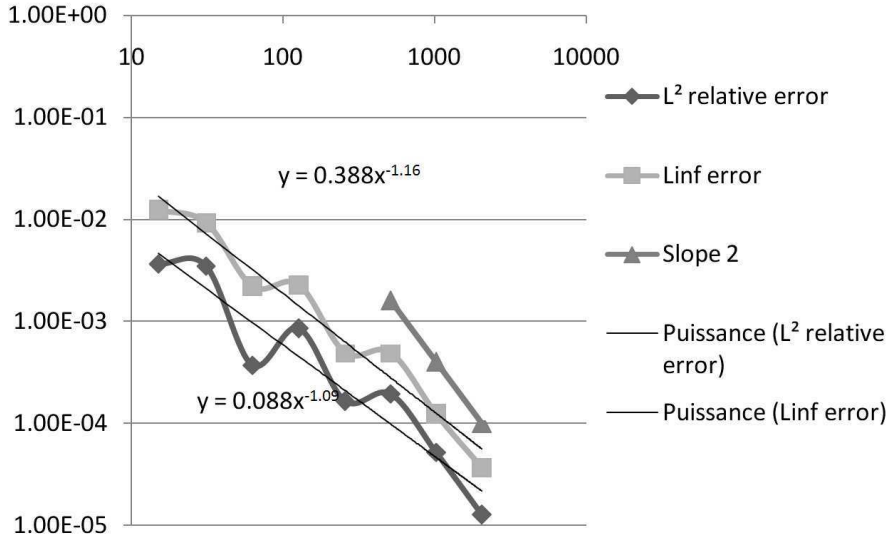

Figure 14: Curves of errors for problem 4

3.2 Immersed interface problems

3.2.1 Problem 5

The 2D problem $\mathcal{P}_{ii}$ 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 the solution is not explicitly imposed at a given location. In the present case, the interface condition is correct anywhere in the domain so the approximation of the interface position does not generate errors.

Fig. 15 shows that the same result is obtained with an interface jump such as $u = r^2$ for $r > 0.5$ and $u = r^2 + 1$ otherwise.

An equivalent quality of result is obtained with Σ such as:

$$\begin{cases} x(\alpha) = (.5 + .2\sin(5\alpha))\cos(\alpha) \\ y(\alpha) = (.5 + .2\sin(5\alpha))\sin(\alpha) \end{cases} \tag{29}$$

with $\alpha \in [0, 2\pi]$. The small stencil of the method allows to treat interfaces with strong curvatures. However, geometrical singularities (*i.e.* exceptional interface/Eulerian grid conformations resulting from a too coarse grid) are not taken into account for now.

3.2.2 Problem 6

The same problem as in 3.2.1 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:

$$u(r) = \begin{cases} r^2 \text{ in } \Omega_0 \\ \frac{r^2}{10} + \frac{0.9}{4} \text{ in } \Omega_1 \end{cases} \tag{30}$$

Accuracy tests are first performed with the interface lying on some grid points (odd mesh). Of course, the interface does not strictly lies on these points, as the shape is shifted by an ϵ. Accuracy tests are then performed with a box of length 1.0001 (even mesh). In this configuration, the interface never lies on a grid point. Results of the numerical accuracy test are presented in Fig. 17. For the odd series of test, 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.

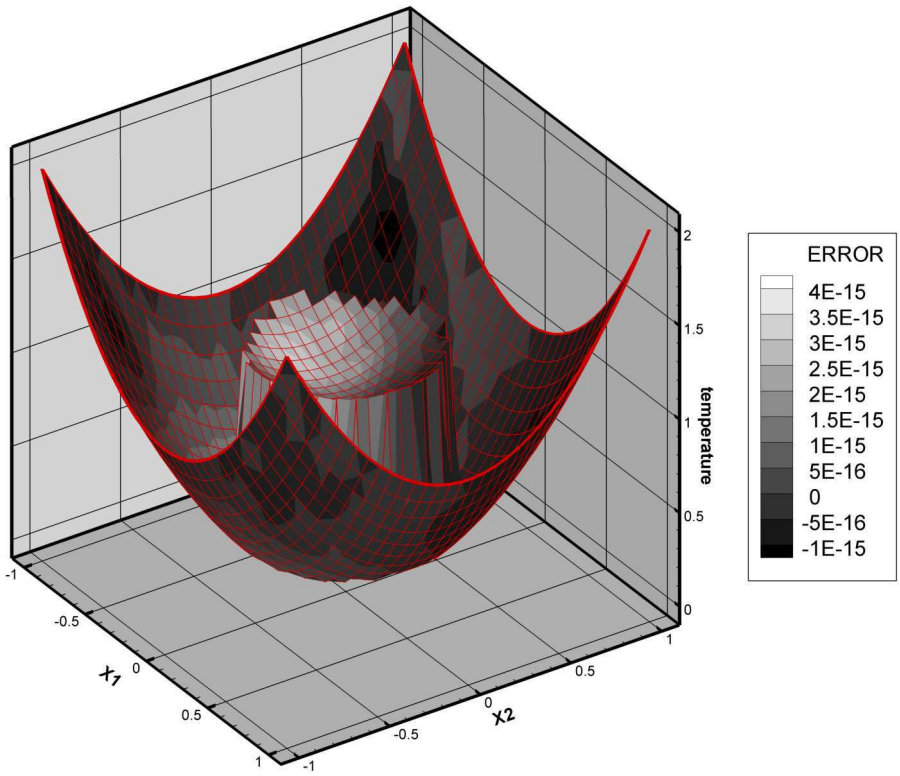

Figure 15: The solution and the error for problem 5 with a 32×32 mesh

Figures 18 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 sub-domain.

3.2.3 Problem 7

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

$$u(x,y) = \begin{cases} 0 \text{ in } \Omega_0 \\ e^x \cos(y) \text{ in } \Omega_1 \end{cases} \tag{31}$$

were Ω_0 and Ω_1 are delimited by Σ defined as before as a centered circle of radius 0.5. Fig. 19 shows that the convergence for both L^2 and L^∞ error are of first order only. Anyway, as can be seen on Fig. 20, which shows the solution and the error map for a 32×32 mesh, the solution is not so

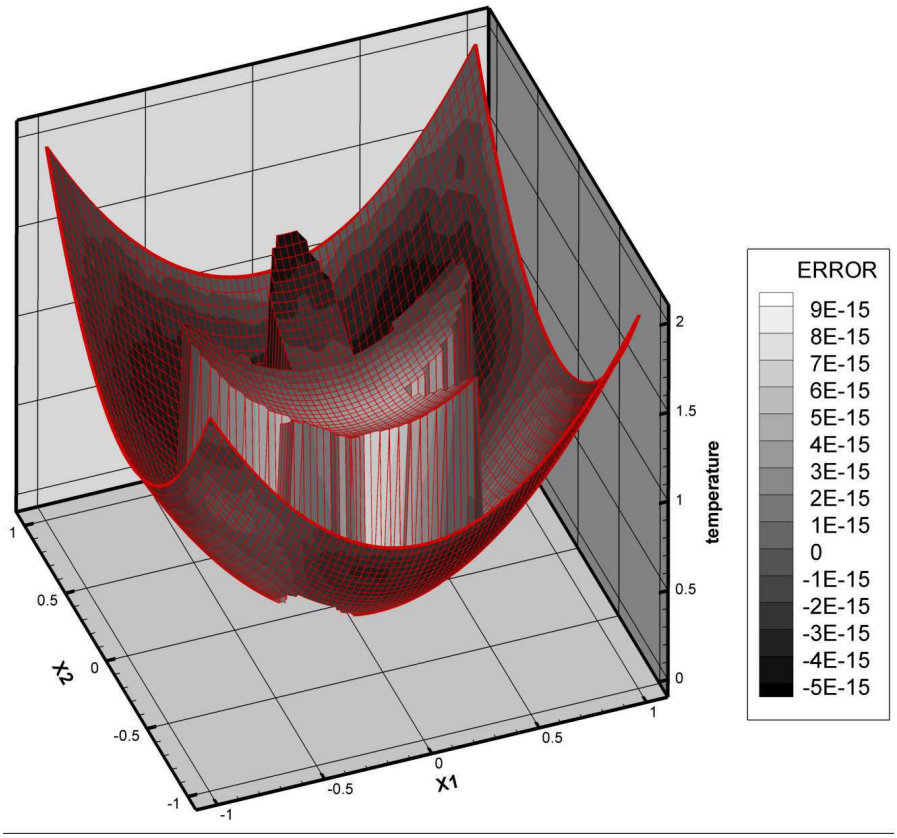

Figure 16: The solution and the error for problem 5 with a 64×64 mesh

different from the analytical solution. In section (3.1.4), a first global order is observed too, even if a second order is reached for the three last meshes. Hence, the convergence is not as good as expected when a condition on the normal flux with a source term ($\psi \neq 0$) is imposed. Numerous trials implying interpolations of higher orders have lead to similar results, so, for now, we cannot explain this lack of accuracy.

In [33], the authors seems to have encountered the same difficulties as they explain how to impose Neumann BC with a quite similar method without performing a precise convergence test.

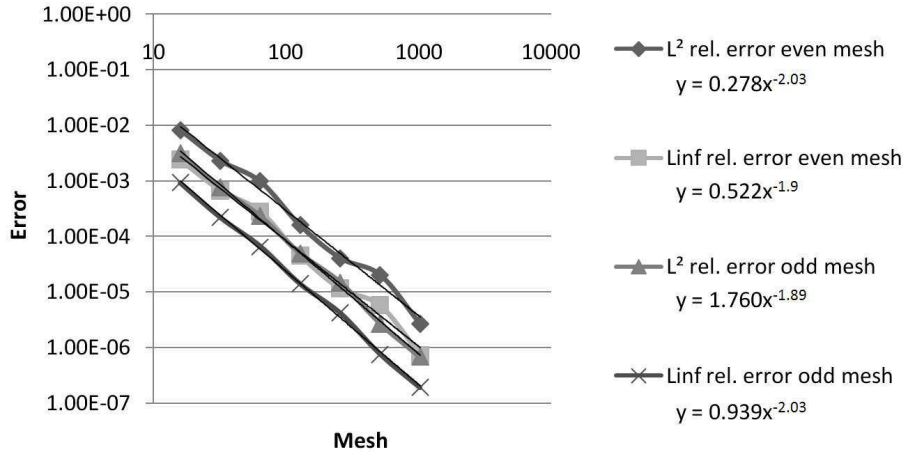

Figure 17: Curves of errors for odd and even meshes the problem 6

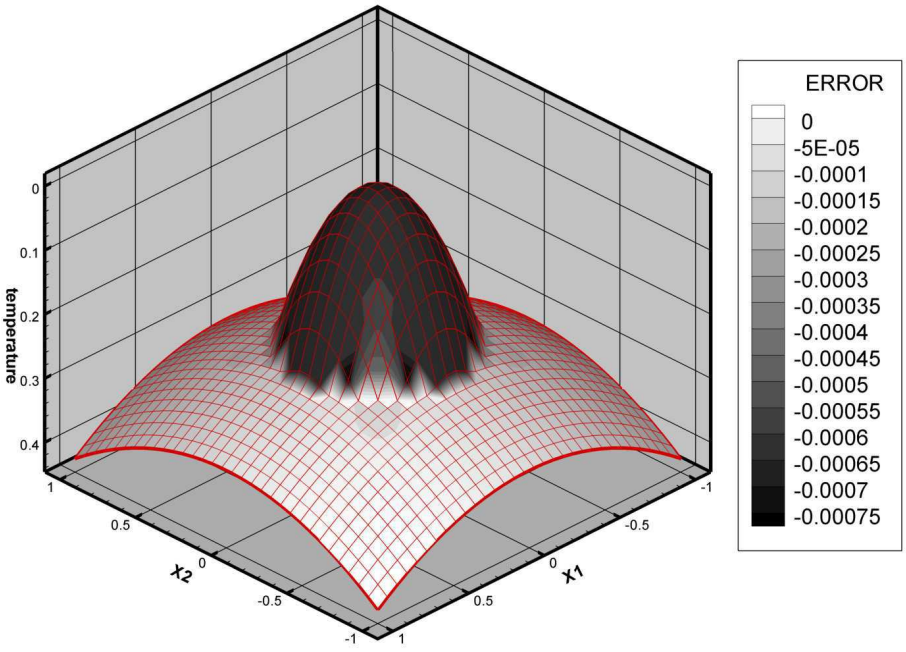

Figure 18: The solution and the error for problem 6 with a 33×33 mesh

3.3 Shape management

3.3.1 Convergence

Our aim here is to measure the sensibility of the method with the accuracy of the Lagrangian mesh discretizing the immersed interface. Problem 1 is

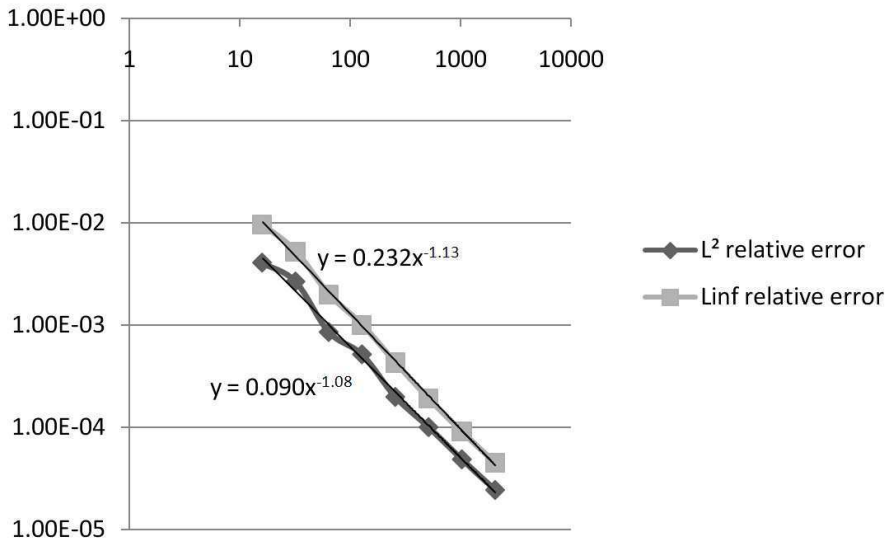

Figure 19: Convergence of the L^2 relative error and the L^∞ error for problem 7

solved on 32×32 and 128×128 meshes. Fig. 21 shows the accuracy of the solution with respect to the number of points used to discretized the interface which is here a circle. The reference solutions (Fig. 7) have been computed with an analytical circle. As can be seen, a second order in space is globally obtained. The reference numerical solutions 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

The purpose of this last case is to demonstrate the enhancement of the interface shape as it is perceived by the problem. 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). As explained before, if Ω_0 is the domain of interest the AIIB method extends the numerical domain Ω_0 in Ω_1 thanks to auxiliary unknowns $u_I, I \in \mathcal{N}_1^*$. The Dirichlet constraints are built using unknowns of $\mathcal{N}_0$ and $\mathcal{N}_1^*$. The interpolations are linear, so for two neighboring points $x_I \in \Omega_0$ and $x_J \in \Omega_1$, the constraints insure $u_I \leq T_\Sigma \leq u_J^*$ or $u_J^* \leq T_\Sigma \leq u_I$. So, to retrieve the shape of the interface as it is considered by the discretization, the solution is post-treated

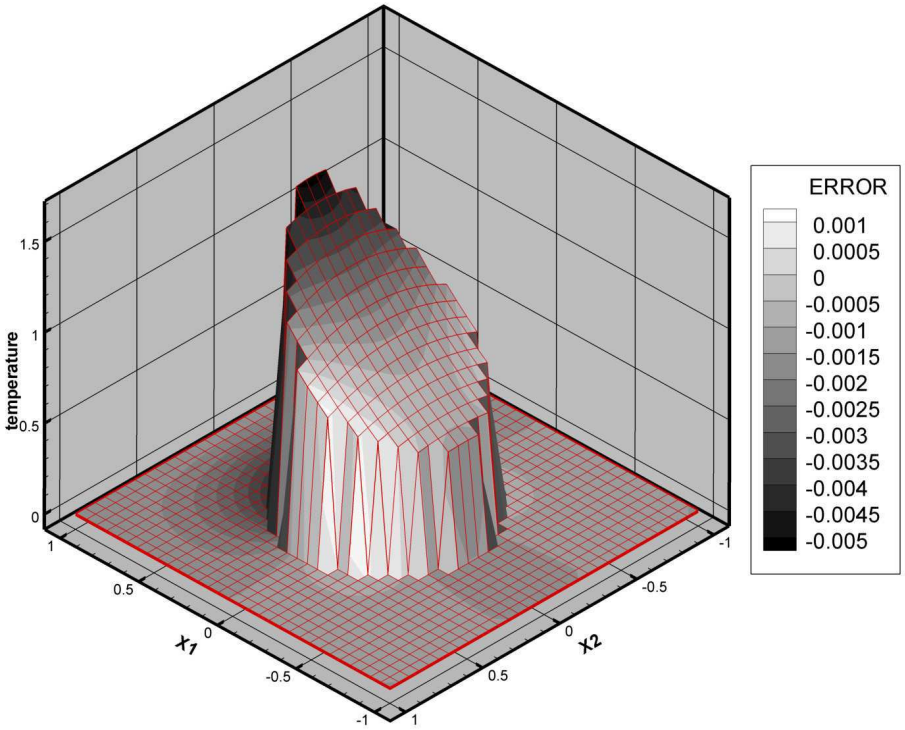

Figure 20: The solution and the L^2 relative error for problem 7 with a 32×32 mesh

with all $u_J, J \in \mathcal{N}_1$ replaced by $u_J^*, J \in \mathcal{N}_1^*$. Then, the iso-surface $T = T_\Sigma$ is plotted thanks to a visualization software (Tecplot 360). Fig. 22 shows the iso-surface for a first order method. As can be seen, the shape of the obstacle endures a rasterization effect as the solution is imposed in the entire control volumes. Fig. 23 shows the iso-surface for the second order AIIB method. Fig. 24 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 coefficients in the lines of the matrix created by the AIIB method depends on the kind of interpolation function used and the position of the

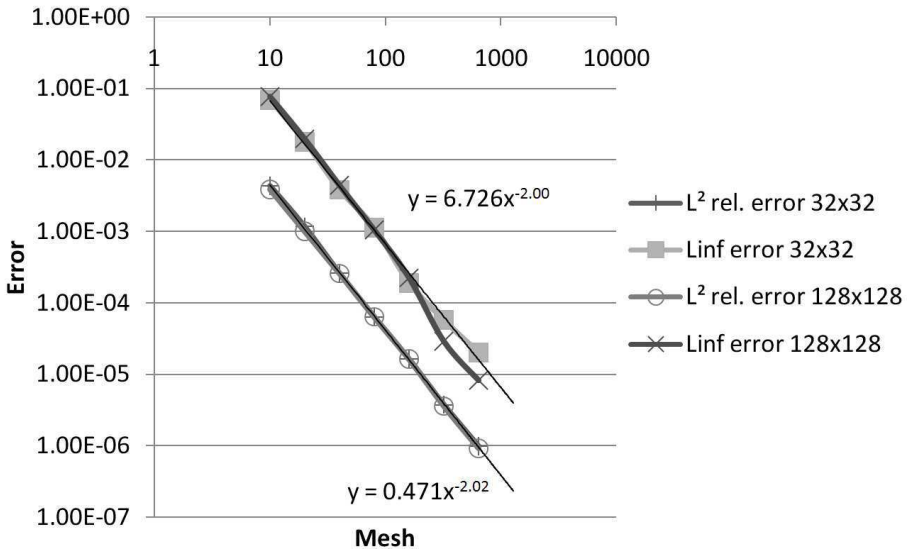

Figure 21: Convergence of the error with respect to the accuracy of the lagrangian shape problem 1

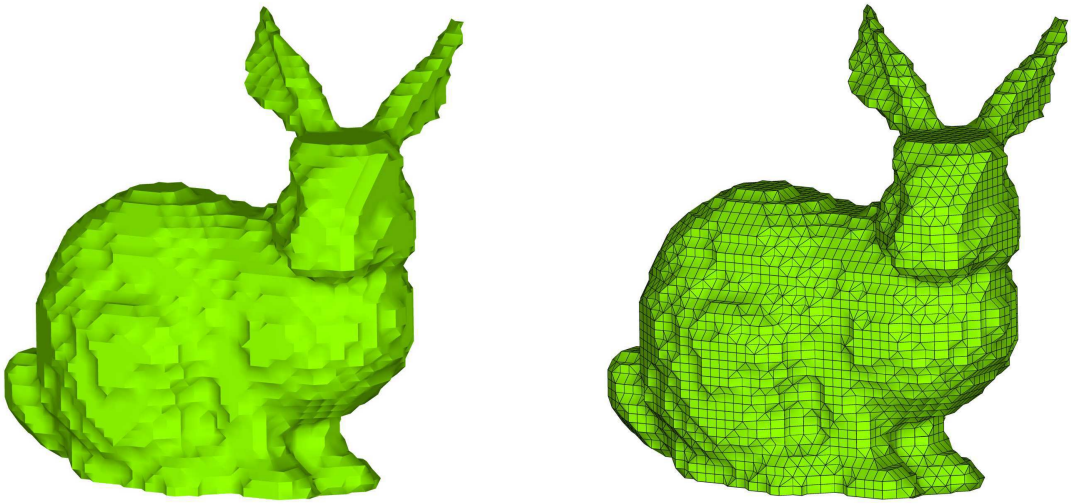

Figure 22: Iso-surface $T = 10$ for the Stanford bunny with a first order method

interface. Let us consider an interface passing between two points x_I and x_J, with x_J an interior point. A Dirichlet BC u_l is imposed on it. The constraint

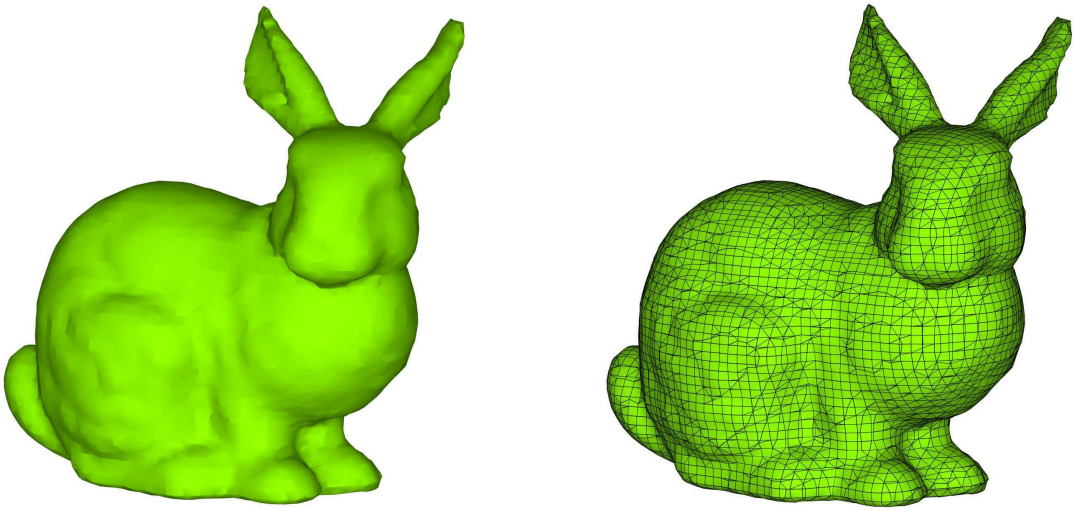

Figure 23: Iso-surface $T = 10$ for the Stanford bunny with a second order method

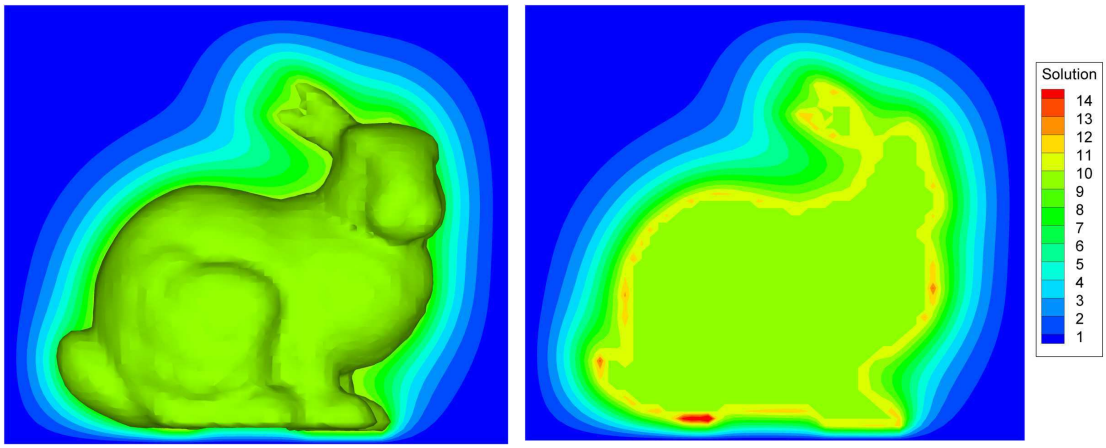

Figure 24: Iso-surface $T = 10$ and a slice of the solution

constructed with a $\mathbb{L}^1_1$ interpolation is $(1-\lambda)u_I + \lambda u^*_J$, with $\lambda = \frac{x_l - x_I}{x_J - x_I}$. Hence, $\frac{\lambda}{1-\lambda}$ tends to 0 when x_l tends to match x_I (One can notice that if $x_l = x_I$, u^*_I is created, the interpolation is simply $u_I = u_l$). Tseng et al. [33] proposed changing the interpolation by using a new node which is the image of x_I through the interface. In [7, 8], authors pointed out this problem.

Gibou et al. suggest slightly moving the interface to a neighboring point if x_I is too close to Σ_h. However, as demonstrated here, our implementation using ILUK preconditioner or PARDISO direct solver does not require such methods, even if $\frac{\lambda}{1-\lambda} \approx 10^{-10}$.

4 Discussion and conclusion

A new simple immersed interface method, the algebraic immersed interface and boundary (AIIB) method, using algebraic manipulations has been presented. This method is able to treat elliptic equations with discontinuous coefficients and solution jumps over complex interfaces. Second order in space is reached for several configurations with minor modifications of the original code. The interface conditions are dicretized with a compact stencil, so the AIIB approach is directly able to treat interfaces with strong curvatures even if a particular treatment of geometric singularities can be required. As the modified matrix looses its diagonal dominance, efficient solvers are required.

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 second order seems to be reachable for the densest meshes.

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. An algebraic reduction is not possible, but compared to the MIB method [37] or to IIM [17], this new method is easier to implement and uses a smaller stencil. An adaptation of the AIIB method to curvilinear grids will be presented in a coming paper.

Future work will be devoted increasing the accuracy of the method when the jump of the normal flux is not zero, and to extend the method to the Navier-Stokes equations with immersed interfaces. Our general aim is to treat complex moving fluid/solid and fluid/fluid interfaces using both AIIB method and ITP method [26] to obtain an accurate two-way coupling.

Appendix : Definition of the interpolation

Let now explain the calculation of the interpolation coefficients for $2D$ problems. Let us consider x_I, x_J, x_K and x_L which define a dual control cell $\mathcal{V}'_I$. A $p \in \mathbb{Q}_1^2$ interpolation over $\mathcal{V}'_I$ is such that $p(x_i) = u_i$ for $i = I, J, K, L$, and $p(x, y) = a_0 + a_1 x + a_2 y + a_3 xy$. The following coordinates matrix can be

defined :

$$Q = \begin{pmatrix} 1 & x_I & y_I & x_I y_I \\ 1 & x_J & y_J & x_J y_J \\ 1 & x_K & y_K & x_K y_K \\ 1 & x_L & y_L & x_L y_L \end{pmatrix} \tag{32}$$

If a is the vector of the interpolation coefficient, $p(x, y) = aQ$ and $a = Q^{-1}p$.

As each term a_i of a is a linear combination of u_i, one can write $p(x, y) = \sum_{i=I,J,K,L} \alpha_i u_i$ with (x, y) the coordinates of the Lagrangian intersection point x_l, $l \in \mathcal{I}$. Practically, the four Eulerian points are the four corners of an unit square and x_l is easily projected in this new frame by insuring

$$\xi = \frac{x_l - x_I}{x_J - x_I} \text{ and } \eta = \frac{y_l - x_I}{x_K - x_I} \tag{33}$$

Fig. 25 illustrates the projection. Then, a unique Q^{-1} matrix has to be found for all interpolations. At last, the matrix coefficients are the following :

$$C_I = (1 - \xi)(1 - \eta) \tag{34}$$
$$C_J = (1 - \xi)\eta \tag{35}$$
$$C_K = \xi\eta \tag{36}$$
$$C_L = (1 - \xi)\eta \tag{37}$$

In [33], the authors include the interface point instead of the auxiliary node providing a more complex linear system. Hence, a matrix has to be inverted for each interpolation.

Acknowledgment

The authors would like to acknowledge the Aquitaine Region Council for its financial support concerning the computational resources.

References

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

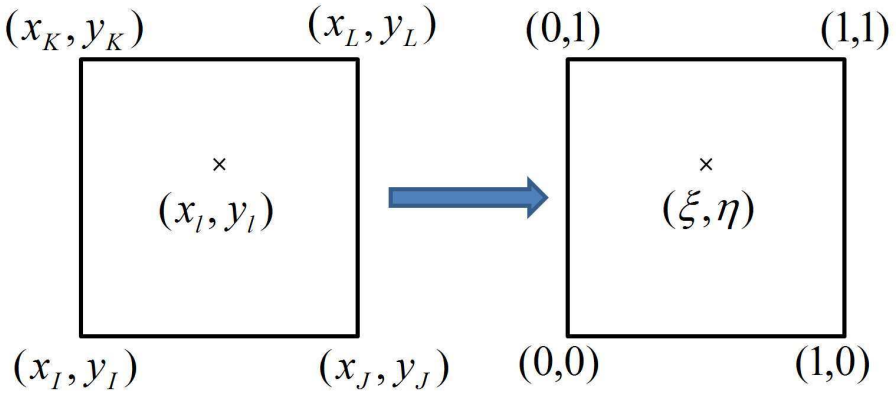

Figure 25: The projection of the initial square defined by the Eulerian mesh to an unit square

[2] F. Domenichini, On the consistency of the direct forcing method in the fractional step solution of the Navier-Stokes equations, J. Comput. Phys. 227 (2008) 6372–6384.

[3] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for threedimensional complex flow simulations, J. Comput. Phys. 161 (2000) 30–60.

[4] M. Fortin, R. Glowinski, Méthodes de Lagrangien augmenté. Application à la résolution numérique de problèmes aux limites, Paris, Dunod (1982).

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

[6] F. Feito, J.C. Torres, Inclusion test in general polyhedra, Computer & Graphics 21(1) (1997) 23–30.

[7] F. Gibou, R. Fedkiw, L.T. Cheng, M. Kang, A Second Order Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains, J. Comput. Phys. 176 (2002) 1–23.

[8] F. Gibou, R. Fedkiw, A Fourth Order Accurate Discretization for the Laplace and Heat Equations on Arbitrary Domains, with Applications to the Stefan Problem, J. Comput. Phys., vol. 202, 2005, p. 577-601.

[9] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, Int. J. of Multiphase Flow 25 (1999) 755–794.

[10] I. Gustafsson, On first and second order symmetric factorization methods for the solution of elliptic difference equations, Technical report, Chalmers University of Technology, 1978.

[11] T.Y. Hou, Z.L. Li, S. Osher, H. Zhao, A hybrid method for moving interface problems with application to the Hele-Shaw flow, J. Comput. Phys. 134 (1997) 236–252.

[12] T. Ikeno, T. Kajishima, Finite-difference immersed boundary method consistent with wall conditions for incompressible turbulent flow simulations, J. of Comput. Phys. 226 (2007) 1485–1508

[13] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, J. Comp. Phys. 147 (1998) 60–85.

[14] K. Khadra, P. Angot, S. Parneix, J.P. Caltagirone, Fictitious domain approach for numerical modelling of Navier-Stokes equations, Int. J. for Num. Meth. in Fl. 34 (2000) 651–684.

[15] D. Lacanette, S. Vincent, A. Sarthou, P. Malaurent, J.P. Caltagirone, 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 (2009) 2528–2542.

[16] M.-C. Lai, H.-C. Tseng, A simple implementation of the immersed interface methods for Stokes flows with singular forces, Comp. and Flu. 37 (2008) 99–106

[17] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (2004) 1001–1025.

[18] Z. Li, A fast iterative algorothm for elliptic interface problems, SIAM J. Numer. Anal. 35 (1998) 230-254.

[19] X.-D. Liu, R. P. Fedkiw, M. Kang A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains J. of Comput. Phys. 160 (2000), 151–178.

[20] P. McCorquodale, P. Colella, H. Johansen, A Cartesian grid embedded boundary method for the heat equation on irregular domains, J. Comp. Phys. 173 (2001) 620–635.

[21] J. Mohd-Yusof, Combined immersed-boundary/b-spline methods for simulation of flow in complex geometries, Technical report, Center for Turbulence, Annuel Research Briefs, 1997

[22] C. J. Ogayar, R. J. Segura, F. R. Feito, Point in solid strategies, Comp. and Graph. 29 (2005) 616–624.

[23] C.S. Peskin, Flow patterns around heart valves, A numerical method, J. Comput. Phys. 10 (1972) 252–271.

[24] C.S. Peskin, The Immersed Boundary Method, Acta Numerica 11 (2000) 479–517.

[25] I. Ramière, P. Angot, M. Belliard, A general fictitious domain method with immersed jumps and multilevel nested structured meshes, J. of Comput. Phys. 225 (2007) 1347–1387.

[26] T.N. Randrianarivelo, G. Pianet, S. Vincent, J.P. Caltagirone, Numerical modelling of solid particle motion using a new penalty method, Int. J. of Num. Meth. in Fluids 47 (2005) 1245–1251.

[27] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual method for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986), 856–869.

[28] A. Sarthou, S. Vincent, J.P. Caltagirone, P. Angot, Eulerian-Lagrangian grid coupling and penalty methods for the simulation of multiphase flows interacting with complex objects, Int. J. of Num. Meth. in Fl., 56 (2008) 1093-1099.

[29] A. Sarthou, S. Vincent, P. Angot, J.P. Caltagirone, The sub-mesh penalty method, Finite Volumes for Complex Applications V (2008) 633–640.

[30] O. Schenk and K. Gärtner, Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO, J. of Fut. Gen. Comp. Sys. 20 (2004) 475–487.

[31] S. Shin, D. Juric, Modeling Three-Dimensional Multiphase Flow Using a Level Contour Reconstruction Method for Front Tracking without Connectivity, J. Comput. Phys. 180 (2002) 427–470.

[32] M. Sussman, E. Fatemi, An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow, SIAM J. Sci. Comput. 20 (1999) 1165–1191.

[33] Y.H. Tseng, J.H. Ferzinger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput Phys. 623 (2003) 192–593.

[34] H. A. Van Der Vorst, Bi-cgstab: a fast and smoothly converging variant of bi-cg for the solution of non-symmetric linear systems, SIAM J. Sci. Stat. Comput. 13 (1992) 631–644.

[35] R. Verzicco, G. Iaccarino, M. Fatica, P. Orlandi, Flow in an impeller stirred tank using an immersed boundary method, in: Annual Research Briefs, NASA Ames Research Center/Stanford University Center for Turbulence Research, Stanford, CA, 2000, pp. 251–261.

[36] S. Yu, G.W. Wei, Three-dimensional matched interface and boundary (MIB) method for treating geometric singularities, J. Comput. Phys. 227 (2007) 602–632.

[37] Y.C. Zhou, S. Zhao, M. Feig, G.W. Wei, High order matched interface and boundary method for lelliptic equations with discontinuous coefficients and singular sources, J. Comput. Phys. 213 (2006) 1–30.

[38] Y.C. Zhou, G.W. Wei, On the fictitious-domain and interpolation formulations of the matched interface and boundary (MIB) method, J. Comput. Phys. 219 (2006) 228–246.