The algebraic immersed interface and boundary method for elliptic equations with jump conditions

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

The algebraic immersed interface and boundary method for elliptic equations with jump conditions

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

January 27, 2010

[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, Immersed boundary method, Penalty methods, Finite volumes, Elliptic equations, Jump Embedded conditions.

Abstract

A new 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 accurately. The main idea is to create auxiliary unknowns at existing grid locations which increases the degrees of freedom of the initial problem. These auxiliary unknowns 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 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.

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. 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 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 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 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 the following model scalar immersed boundary problem with a Dirichlet boundary condition (BC) on the interface Σ (see Fig. 1):

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

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.* [10]. Lagrange multipliers are introduced into the weak formulation of the initial elliptic equation to ensure the immersed boundary condition.

Cartesian grid [14, 21] and Cut-cell CITE 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 [24, 25]. 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 [22]. The idea here is to impose a no-slip condition directly on the boundary using a mirrored flow over the boundary. In [4, 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 [34], 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 [30], this kind of approach seems more accurate than [4, 35].

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), [2, 1] requires the addition of a penalty term $\frac{\chi}{\varepsilon}(u - u_D)$ in the conservation equations, such that:

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

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 ([15, 16]) authors add a 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 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 [26]. In [30, 29], Sarthou et al. have discretized the volume penalty term with a second order using implicit interpolations as in [34]. 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 [7, 36].
Applied to problem $\mathcal{P}_b$, the ghost cell immersed boundary method [34] and SMP method [29] used the first cells in Ω_1 to enhance the accuracy of the solution in Ω_0. An other approach which considers the extension of the solution is considered in [9, 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 "by-hand".

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

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

A first class of method is the immersed interface methods (IIM) initially introduced by LeVeque and Li [18]. 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 applications, such as moving interfaces [12] or Navier-Stokes equations [17]. In [19], 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, 20], 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 [39, 38, 37] 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 [19], 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 [19] to discretize additional equations.

The present method solves elliptic problems using an augmented method coupled with an auxiliary unknown approach. Contrary to ghost nodes, auxiliary unknowns are present in the linear system. 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

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 requires interpolations, we use the following interpolations in 2D: $\mathbb{P}_1^2(x,y) = p_1 + p_2 x + p_3 y$ and $\mathbb{Q}_1^2(x,y) = p_1 + p_2 x + p_3 y + p_4 xy$. In 3D, we use $\mathbb{P}_1^3(x,y,z) = p_1 + p_2 x + p_3 y + p_4 z$ and $\mathbb{Q}, \digamma_1^3(x,y) = p_1 + p_2 x + p_3 y + p_4 z + p_5 xy + p_6 yz + p_7 zx + p_8 xyz$. An additional interpolation, $\mathbb{L}_1^1(x) = p_1 + p_2 x$, is use for 2D and 3D problems. The superscript is the dimension of the interpolation while the subscript is the order of spatial accuracy.

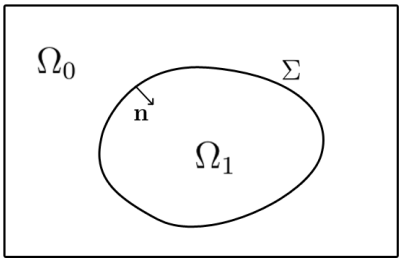

Figure 1: Definition of the subdomains and the interface

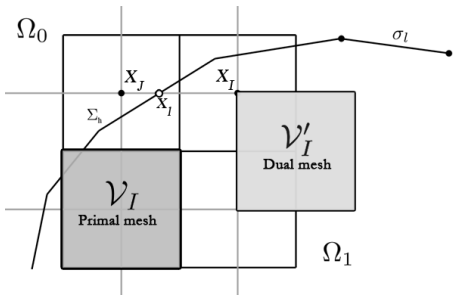

Figure 2: Definition of the discretization kernels for the AIIB method

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 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 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 : $\Sigma_h = \{\sigma_l \subset \mathbb{R}^{d-1}, l \in \mathcal{L}_f\}$, $\mathcal{L}_f$ being the set of index 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 $i = 1, d$ 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.
We shall use the following Eulerian volume fonctions :

- 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] :

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

New sets of Eulerian points x_I are defined near the interface so that each one has a 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 $\mathcal{N}_0 = \{I, x_I \in \Omega_0, \chi_I \neq \chi_J, x_J \in \Omega_1\}$ and $\mathcal{N}_1 = \{I, x_I \in \Omega_1, \chi_I \neq \chi_J, x_J \in \Omega_0\}$.
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 on the Eulerian physical grid.
In order to obtain the discrete Heaviside function χ, one have 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 [30,

16], the authors used a global methodology partly based on [32] where χ is obtained thanks to a PDE. This method suffers from a lack of accuracy and robustness and a Ray-casting method based on the Jordan curve theorem 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. Some details of the implementation and a short review of point in solid strategies can be found in [23].

Algorithm 1 describes a pseudo-code performing a basic computation of $\chi_I = \chi(x_I)$ for all x_I. 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 (for computational efficiency).

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
```

Concerning the ray-triangle intersection, [23] 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.

Algorithm 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 $S(I, nsect_I)$. For each $x_I \in \mathcal{S}_{xy}$, the number $nsect_I$ of intersections by rows, is not known *a priori*. If S is an array, a first pass has to be performed to determine the size of S. 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, no octree structure).

In 3D, the shape is discretized with triangles which have common points and vertices. If a ray pass through one of these entities, more than one intersection are detected. It generally occurs when the computational domain has rounded or symmetrical dimensions, and when the object is centered on it (see Fig. 3). The intersection algorithm can be modified to avoid this but a simple solution is to shift the Lagrangian object by a distance ϵ which is small enough to do not provide additional error.

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 intersection 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, $x_l = (x_I d_J + x_J d_I)/(d_I + d_J)$.

Algorithmic problems can be encountered if the Lagrangian mesh is too complex compared to the Eulerian mesh. For example, two intersecting points x_l can 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.

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 $S(I, nsect)$
      $nsect := nsect + 1$
    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 $x_j(3) > S(I, nsect_{tmp})$ **do**
      Switch In_state
      $nsect_{tmp} := nsect_{tmp} + 1$
    end while
    $\chi(x_J) := In\_state$
  end for
end for

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.

2.3 The algebraic immersed interface and boundary method

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

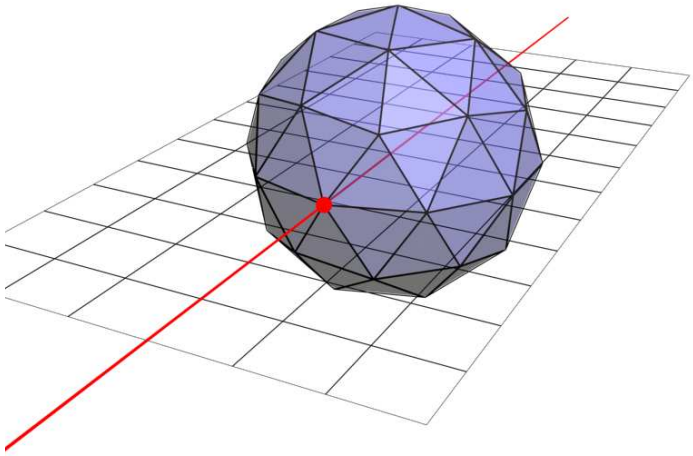

Figure 3: Illustration of a typical error during the ray-casting process. The ray intersect the interface at the common vertex of many triangles

(SMP) method [30, 29] 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 [9, 8] and the immersed interface methods [18]), 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 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 the interface conditions. The solution u' is

decomposed such as $u' = (u, u^*)^T$ and the source term as $b' = (b, b^*)^T$. The interface constraints are discretized with a $(n, m+n)$ block matrix C and the source term b^*.

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 $J \in \mathcal{N}_1$, $\alpha'_{I,J} = 0$ and $\alpha'_{I,J^*} = \alpha_{I,J}$, 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 $(n, m+n)$:

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

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:

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

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:

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

The entire problem can be then solved to obtain $u' = (u, u^*)^T$. 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:

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

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 $\mathcal{P}_b$ 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 (Fig. 1 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 :

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

As noticed in [34, 8], a linear interpolation only is required to reach a second order of accuracy. 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 Fig. 4 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 :

$$u_p = \alpha_I u_I^* + \alpha_J u_J + \alpha_K u_K \ , \ 0 < \alpha_I, \alpha_J, \alpha_K < 1 \ , \ \alpha_I + \alpha_J + \alpha_K = 1 \tag{10}$$

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 Fig. 4 left). As a third choice, two independent linear 1D interpolations can be used (one for each direction) for an almost equivalent result. It produces :

$$\begin{cases} u_l = \alpha_I u_I^* + \alpha_J u_J \text{ with } 0 < \alpha_I, \alpha_J < 1 \text{ and } \alpha_I + \alpha_J = 1 \\ u_m = \alpha_I' u_I^{*'} + \alpha_K u_K \text{ with } 0 < \alpha_I', \alpha_K < 1 \text{ and } \alpha_I' + \alpha_J = 1 \end{cases} \tag{11}$$

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 :

$$\alpha_I u_I^* + \alpha_J u_J = u_l \tag{12}$$

$$\alpha_I' u_I^* + \alpha_K u_K = u_m \tag{13}$$

A summation of these two constraints gives :

$$\alpha_I u_I^* + \alpha_J u_J + \alpha_I' u_I^* + \alpha_K u_K = u_l + u_m \tag{14}$$

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

$$\begin{gathered} \frac{(\alpha_I + \alpha_I')u_I^* + \alpha_J u_J + \alpha_K u_K}{2} = u_p \ , \\ \text{with } 0 < \frac{\alpha_I + \alpha_I'}{2}, \frac{\alpha_J}{2}, \frac{\alpha_K}{2} < 1 \ , \ \frac{\alpha_I + \alpha_I'}{2} + \frac{\alpha_J}{2} + \frac{\alpha_K}{2} = 1 \end{gathered} \tag{15}$$

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.
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^* 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. Nevertheless, the fictitious domain approach computes a non-physical solution in Ω_1. It is naturally obtained with the initial set of equations together with a volume penalty method such as VPM [15]. The imposed solution can be analytical when possible, or an arbitrary constant value. The computational cost of this 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 :

$$\begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega \\ u^-_{|\Sigma} = u_D & \text{on } \Sigma \\ u^+_{|\Sigma} = u_G & \text{on } \Sigma \end{cases} \tag{16}$$

The problem (16) 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 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.

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 :

$$\begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega_0 \\ (a \cdot \nabla u) \cdot \mathbf{n} = g_N & \text{on } \Sigma \end{cases} \tag{17}$$

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, at any point $x_l, l \in \mathcal{I}$ on Σ_h we use

$$(a \cdot \nabla u_l) \cdot \mathbf{n} \approx (a \cdot \nabla p(x_l) \cdot \mathbf{n}). \tag{18}$$

For $p \in \mathbb{Q}_1^2$, we get $\nabla p(x,y) \cdot \mathbf{n} = (p_3 y + p_2) n_x + (p_3 x + p_1) n_y$ whereas for $p \in \mathbb{P}_1^2$, $\nabla p(x,y) \cdot \mathbf{n} = p_2 n_x + p_1 n_y$ is obtained which means that the normal gradient is approximated by a constant over the whole support. For example, in the configuration of Fig. 4.left, with $p \in \mathbb{P}_1^2$, we have :

$$\nabla p(x,y) \cdot \mathbf{n} = \frac{u_I^* - u_J}{h_x} n_x + \frac{u_K - u_I^*}{h_y} n_y = u_I^* (\frac{n_x}{h_x} - \frac{n_y}{h_y}) + u_J \frac{n_x}{h_x} + u_K \frac{n_y}{h_y} \tag{19}$$

The diagonal coefficient of the raw related to u_I^* in C_2 is $(\frac{n_x}{h_x} - \frac{n_y}{h_y})$. The case when $\frac{n_x}{h_x} \approx \frac{n_y}{h_y}$ leads to numerical instabilities. If we consider the configuration of Fig. 4.left, using the normal vector of the segment $[x_l, x_m]$ 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 cases. 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 Fig. 4 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.

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 one auxiliary unknown is needed:

$$u_I^* = \sum_{J \in \mathcal{N}} \alpha_J u_J + u_S \tag{20}$$

where u_S is the source term. In this case, the matrix C_2 in (7) is diagonal and thus the Schur complement $(\tilde{A} - B C_2^{-1} C_1)$ is easy to calculate. Practically,

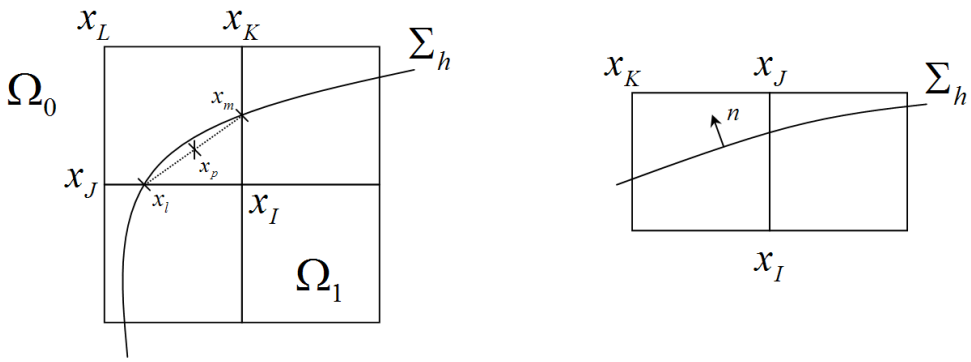

Figure 4: Example of selection of points for Dirichlet (left) and Neumann (right) constraints

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 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 [9] 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 is the same as for the SMP [29] method (when the penalty parameter tends to zero) and the DF-IB method [34]. These methods change the discretization of the initial equation for the nodes $x_I, I \in \mathcal{N}_1$. 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 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.

2.3.6 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 immerse obstacles in fluid flows. Both methods can be used for the scalar and the Navier-Stokes equations. In the latter, the procedure is done componentwise for the velocity vector. However, the AIIB method, with $\mathbb{L}^1_1$ interpolations only, cannot be applied to the Navier-Stokes equations on staggered grid (no tests have been performed for a collocated approach). An illustration is given Fig. 5. With such interpolations, two auxiliary unknowns u_I^* and $u_I^{*'}, I \in \mathcal{N}_1$ can coexist at the same location x_I. Hence, u_I^* is the natural neighbor of u_J and $u_I^{*'}$ is the natural neighbor of u_K. So a problem occurs for the discretization of the inertial term since a node of a given velocity component has to use an auxiliary unknown of an other velocity component. In this case, neither u_I^* nor $u_I^{*'}$ are natural neighbors for v_l, a velocity unknown in the y direction. No matter which unknown is used, or an average of the two collocated unknowns, the simulation is instable outside 'the Stokes regime.

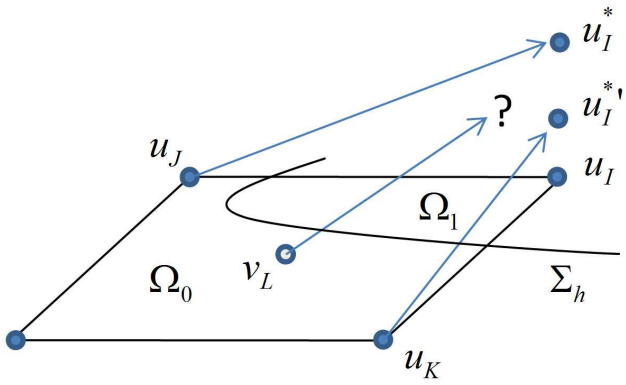

Figure 5: Illustration of the application to the Navier-Stokes equations on staggered grid

A particular attention has also to be given to the velocity pressure coupling. If a fractional step method is used, the prediction step is modified by any fictitious domain method to impose an immersed boundary condition for the velocity. Thus, the projection step has to be modified according to the prediction step to remain consistent with the overall problem.

However, some authors do not consider at all this modification of the correction step [34] or have only made minor modifications. In fact, the projection step has to be rewritten considering the forcing term, as can be seen in [13, 3]. In [29], the authors use an iterative augmented Lagrangian method [36] which adds a penalty term in the momentum equation to enforce the divergence free constraint.

2.4 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 :

$$[\![u]\!]_\Sigma = \varphi \quad \text{on } \Sigma \tag{21}$$

$$[\![(a \cdot \nabla u) \cdot \mathbf{n}]\!]_\Sigma = \psi \quad \text{on } \Sigma \tag{22}$$

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 on both sides of the interface. Hence, the interface constraints (21) and (22) of $(\mathcal{P}_i)$ can be imposed at each intersection point x_l by using the two auxiliary unknowns. For example, the I^{nth} row of the matrix A' with $u_I^*, I \in \mathcal{N}_0$ can be used to impose the constraint (21) and the J^{nth} line of the matrix with $u_J^*, J \in \mathcal{N}_1$ is then used to impose the constraint (22).

2.4.1 The solution constraint

The symmetrized AIIB methods for Dirichlet BC reads :

$$\begin{cases} u_\Sigma^+ = \alpha_1 u_I + \alpha_2 u_J^* \\ u_\Sigma^- = \alpha_1 u_I^* + \alpha_2 u_J \end{cases} \tag{23}$$

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

$$\alpha_1 u_I + \alpha_2 u_J^* - \alpha_1 u_I^* - \alpha_2 u_J = \varphi \tag{24}$$

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,

$$\begin{cases} (a \cdot \nabla u_\Sigma^+) \cdot \mathbf{n} = a^+ (\frac{u_I - u_J^*}{h_x} n_x + \frac{u_K^* - u_I}{h_y} n_y) \\ (a \cdot \nabla u_\Sigma^-) \cdot \mathbf{n} = a^- (\frac{u_I^* - u_J}{h_x} n_x + \frac{u_K - u_I^*}{h_y} n_y) \end{cases} \tag{25}$$

for the case presented in Fig. 4.left. Using (22), we get:

$$a^+ \left(\frac{u_I - u_J^*}{h_x} n_x + \frac{u_K^* - u_I}{h_y} n_y \right) - a^- \left(\frac{u_I^* - u_J}{h_x} n_x - \frac{u_K - u_I^*}{h_y} n_y \right) = \psi \tag{26}$$

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 (24) and (26). The auxiliary unknown u_K^* is also involved in the discretization of (21) and (22) at another intersection point on Σ_h. Hence, the whole system $A'u' = b'$ is closed.

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}.

For the matched interface and boundary (MIB) method, Zhou et al. [39] use a different discretization of the interface conditions which allows an easy algebraic reduction which is directly performed raw by raw.

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 the additional computational time generated by the auxiliary nodes is small. Hence, the lack of algebraic reduction does not seem to be problematic.

3 Numerical results for scalar problems

Elliptic equations are discretized using the standard second-order centered Laplacian. For all problems, similar results have been obtained with a PARDISO direct solver [31], and an iterative BiCGSTAB solver [11], preconditioned under a ILUK method [28]. 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:

$$\|u\|_{L^2_{rel}(\Omega)} = \frac{\|u-\tilde{u}\|_{L^2(\Omega)}}{\|\tilde{u}\|_{L^2(\Omega)}} = \Big(\sum_{x_I\in\Omega} meas(\mathcal{V}_I)|u_I-\tilde{u}(x_I)|^2\Big)^{\frac{1}{2}} / \Big(\sum_{x_I\in\Omega} meas(\mathcal{V}_I)|\tilde{u}(x_I)|^2\Big)^{\frac{1}{2}} \tag{27}$$

whith $\tilde{u}$ is the analytical solution.
The discrete L^∞ error is defined as:

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

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. The 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:

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

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 $\mathbb{P}^2_1$ interpolations. 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. The $\mathbb{P}^2_1$

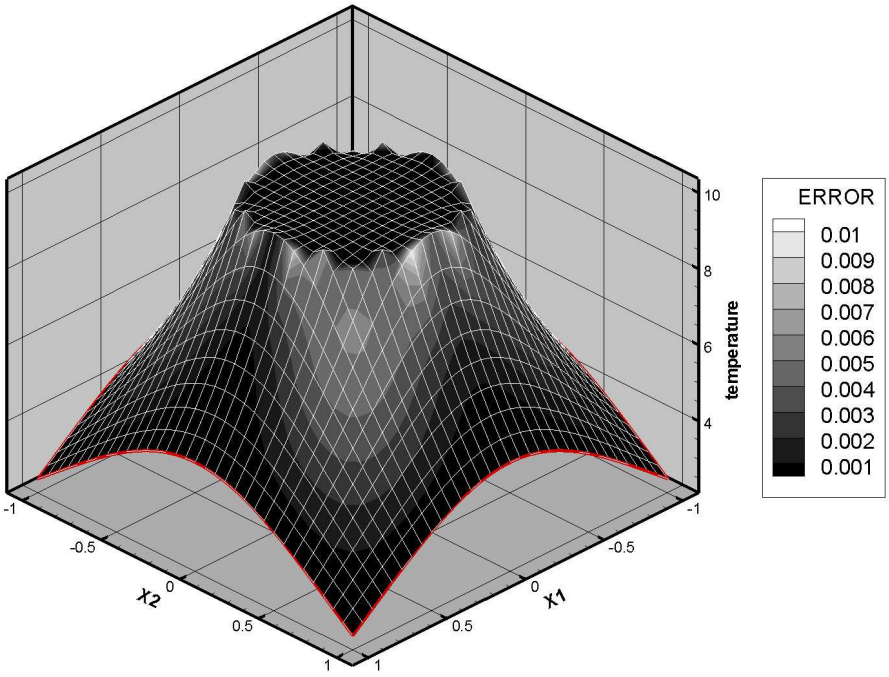

Figure 6: Solution and error map for problem 1

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 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 inversions (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 $\mathbb{Q}_1^1$ 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 AIIB method with algebraic reduction. Figures 8, 9, 10, 11 shows the convergence of the ILUK-BiCG-Stab solver for the seven config-

urations. The type of interpolation does not significantly impact on solver performances.

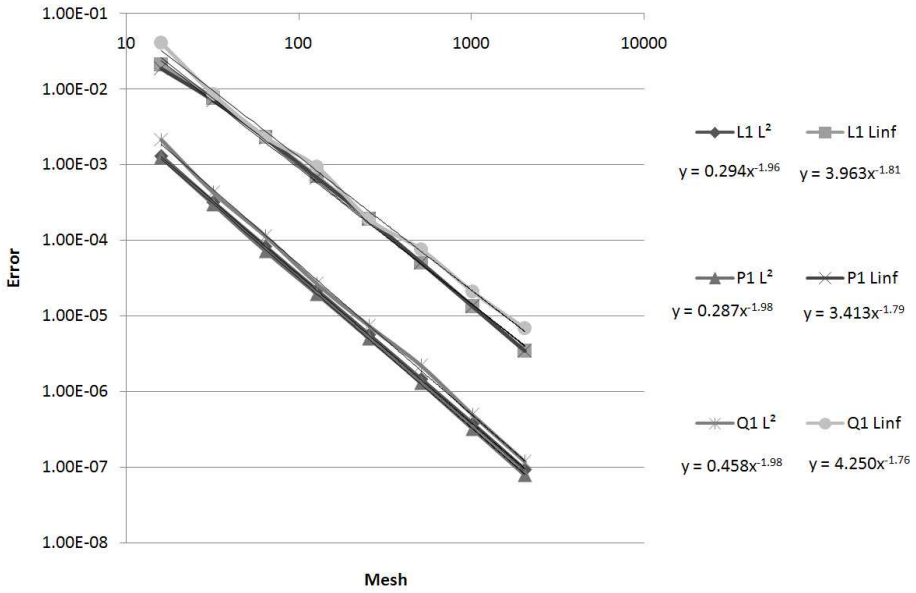

Figure 7: Curves of errors for section 3.1.1

Mesh	$\mathbb{L}^1_1$ std	$\mathbb{L}^1_1$ red	$\mathbb{P}^2_1$ std	$\mathbb{P}^2_1$ red	$\mathbb{Q}^2_1$ std	$\mathbb{Q}^2_1$ red	$\mathbb{P}^2_1$ 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

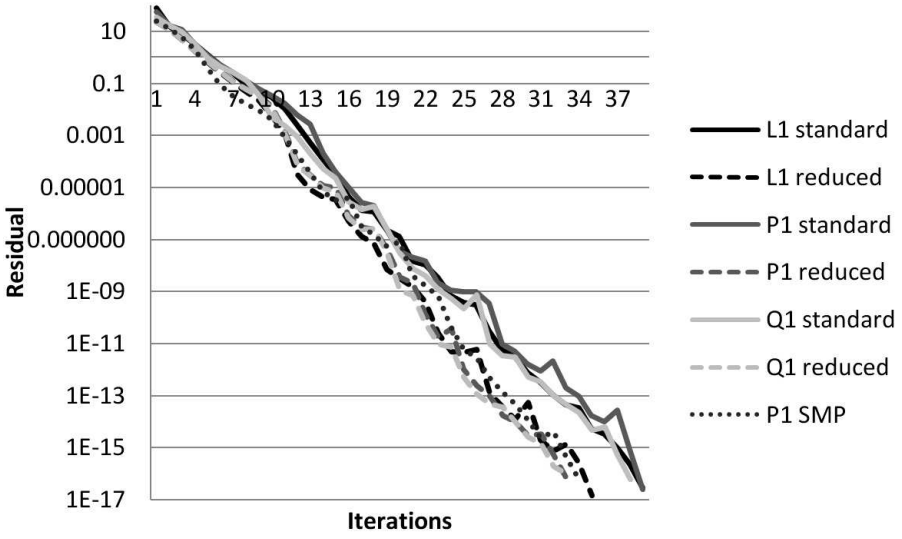

Figure 8: Residual against iterations of ILUK solver for problem 1 with a 128×128 mesh

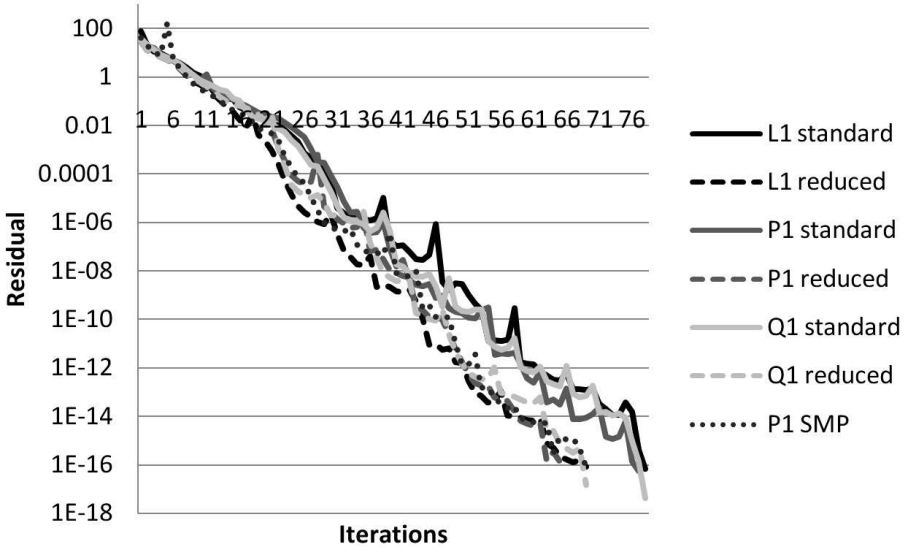

Figure 9: Residual against iterations of ILUK solver for problem 1 with a 256×256 mesh

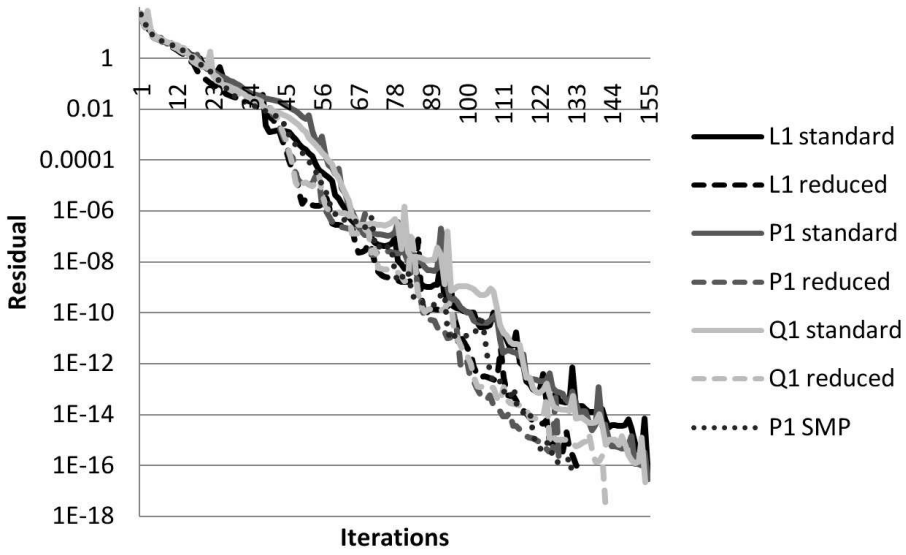

Figure 10: Residual against iterations of ILUK solver for problem 1 with a 512×512 mesh

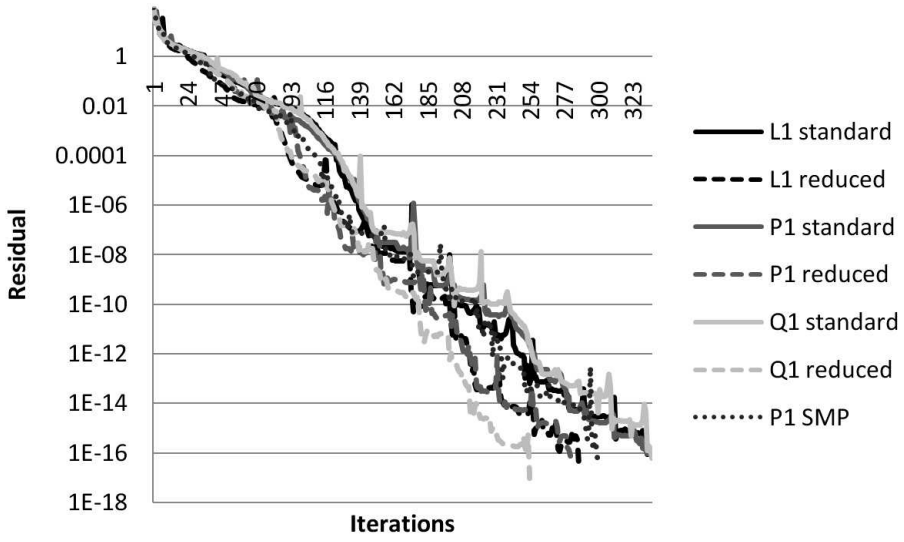

Figure 11: Residual against iterations of ILUK solver for problem 1 with a 1024×1024 mesh

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. As expected, the second-order code gives the exact solution to almost computer-error accuracy 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

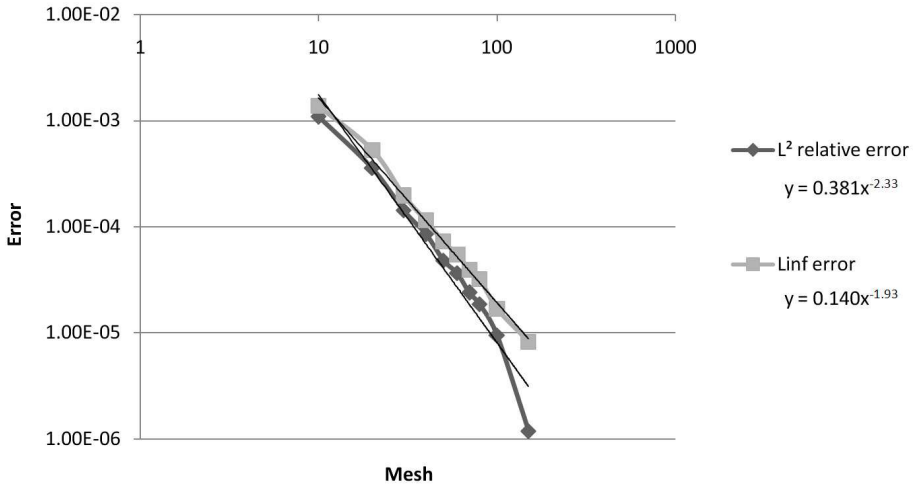

Figure 12: Curves of errors for problem 2

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 Σ_h is an approximation of the initial sphere Σ.

3.1.3 Problem 3

The 3D equation $\Delta T = 12r^2$ is solved. The solution is $T(r) = x^4 + y^4 + z^4$. 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 one 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. For both cases the numerical solution tends to a second order in space.

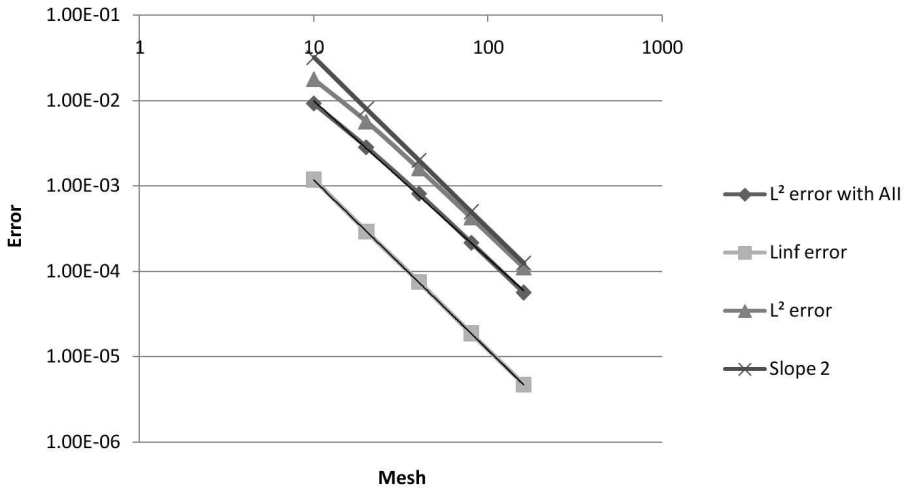

Figure 13: Curves of errors for problem 3

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 Fig. 14, the global convergence has an average slope of 1.10. However, the convergence for the three biggest meshes reaches a slope of 2.

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 still 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.

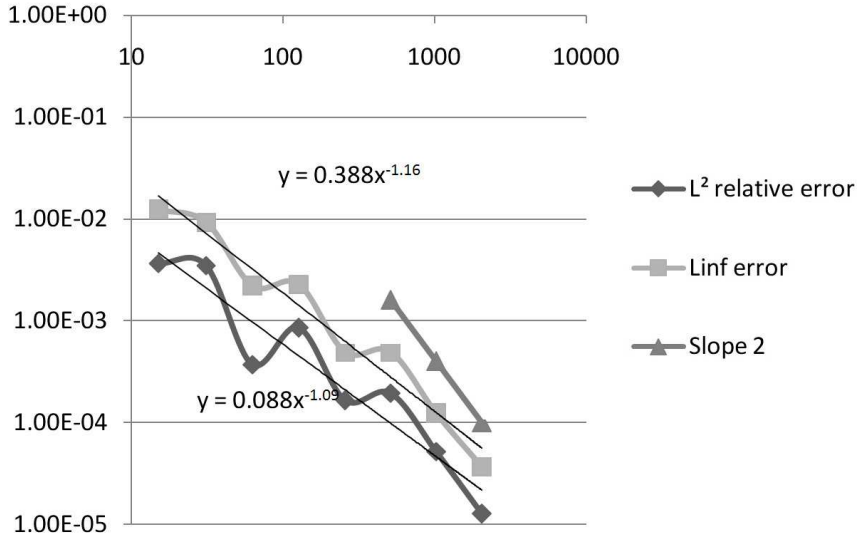

Figure 14: Curves of errors for problem 4

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

$$\begin{cases} x(\alpha) = (.5 + .2\sin(5\alpha))\cos(\alpha) \\ y(\alpha) = (.5 + .2\sin(5\alpha))\sin(\alpha) \end{cases} \tag{30}$$

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

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:

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

Accuracy tests are first performed with the interface almost passing by some grid points (called odd mesh). The interface does not strictly lies on these points, as the shape is shifted by an ϵ. This configuration is difficult as the interpolations degenerates. Accuracy tests are then performed with a box of length 1.0001 (called even mesh). In this configuration, the interface never passes by a grid point. The results of the numerical accuracy test are

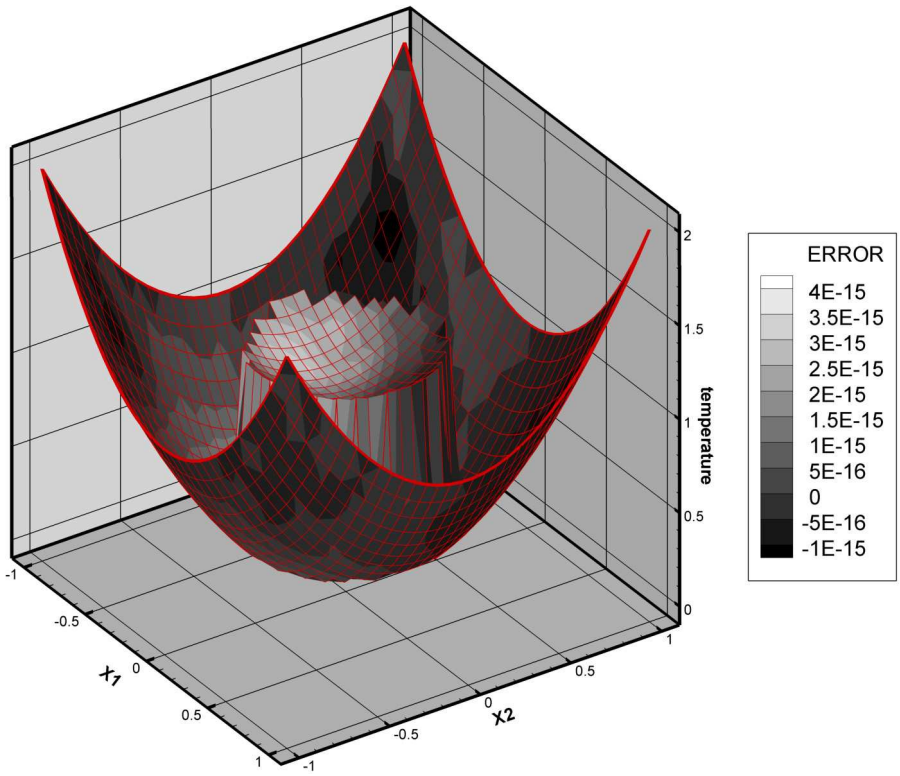

Figure 15: The solution and the error for problem 5 with a 32×32 mesh

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.

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 subdomain.

3.2.3 Problem 7

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

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

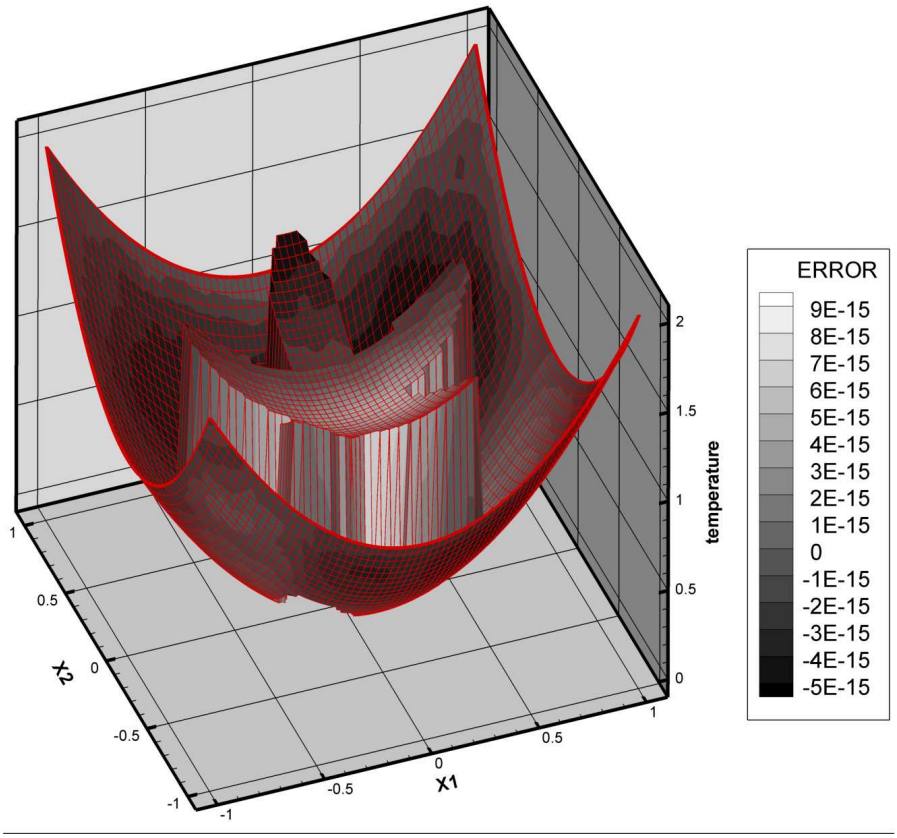

Figure 16: The solution and the error for problem 5 with a 64×64 mesh

where Ω_0 and Ω_1 are delimited by Σ 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. The Fig. 20 shows the numerical solution (which is not so different from the analytical solution) and the error map for a 32×32 mesh. 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 the first order of convergence.

In [34], the authors seems to have encountered the same difficulties as they explain how to impose Neumann BC with a quite similar method without performing showing a convergence test.

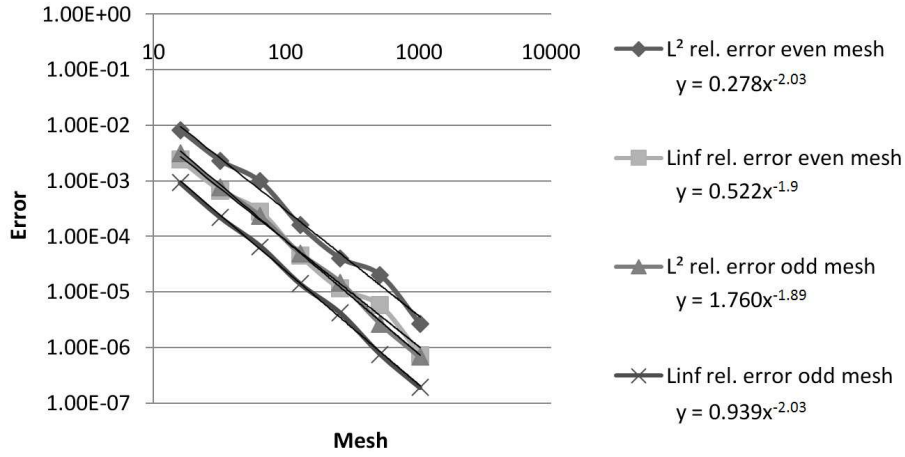

Figure 17: Curves of errors for odd and even meshes the problem 6

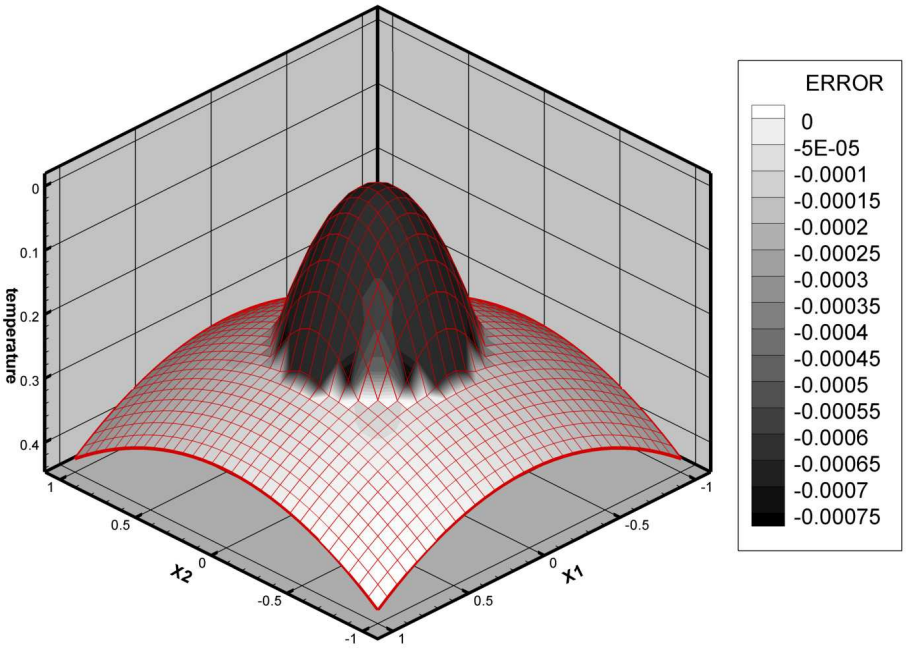

Figure 18: The solution and the error for problem 6 with a 33×33 mesh

3.3 Shape management

3.3.1 Convergence

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

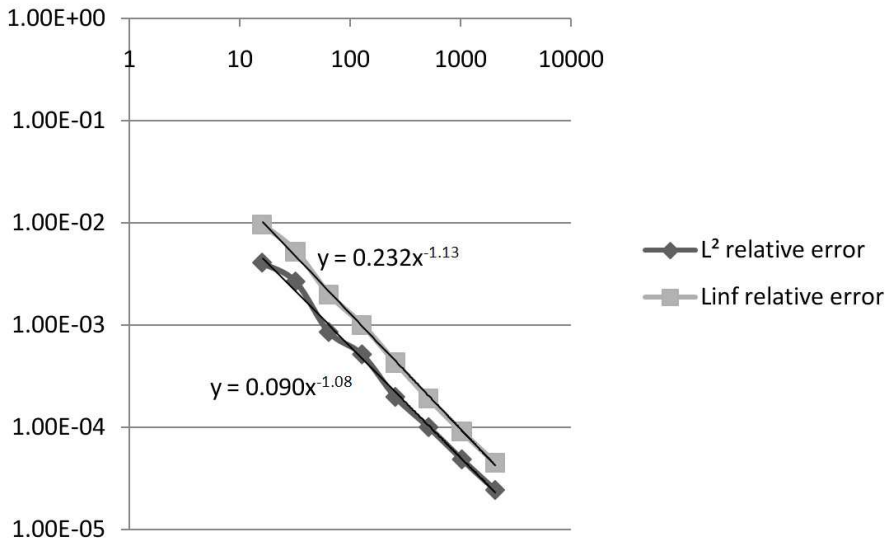

Figure 19: Convergence of the L^2 relative error and the L^∞ error for problem 7

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 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 second-order method enhances the representation of the boundary condition compared to a first-order method. 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. 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.

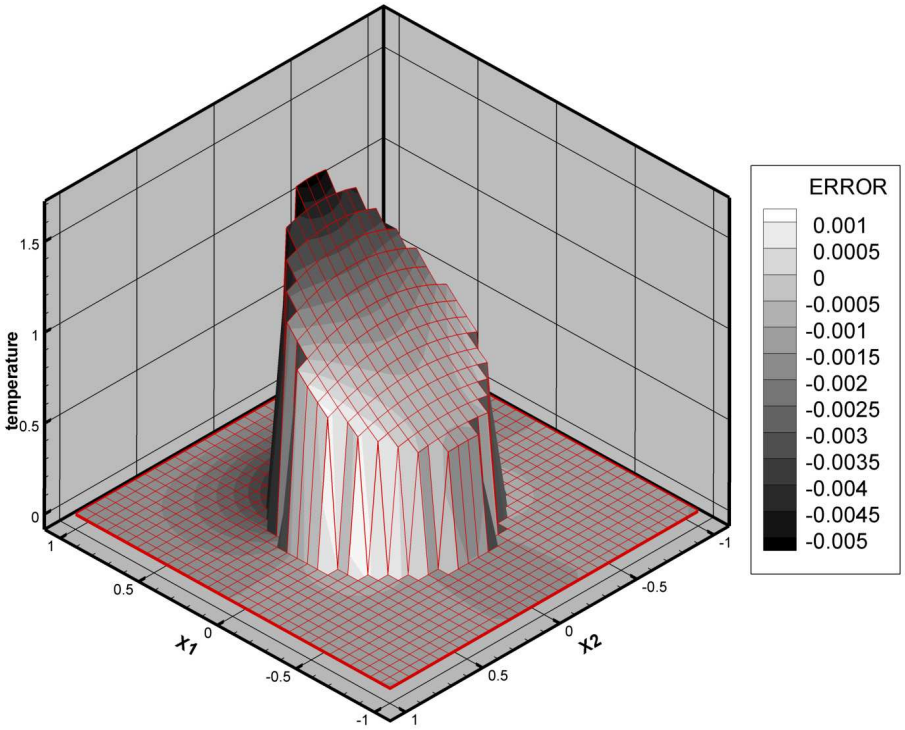

Figure 20: The solution and the L^2 relative error for problem 7 with a 32×32 mesh

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 kind of interpolation function used and the position of the interface have an impact on the final discretization matrix C', especially on its conditionning. Let us consider an intersection x_l 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_l is imposed on it. The constraint constructed with a $\mathbb{L}_1^1$ interpolation is $(1-\alpha)u_J + \alpha u_I^*$, with $\alpha = \frac{x_l - x_J}{x_I - x_J}$. Hence, $\frac{\alpha}{1-\alpha}$ tends to 0 when x_l tends to x_J. As the matrix loses its diagonal dominance, solver problems can be encountered. Tseng et al. [34] proposed

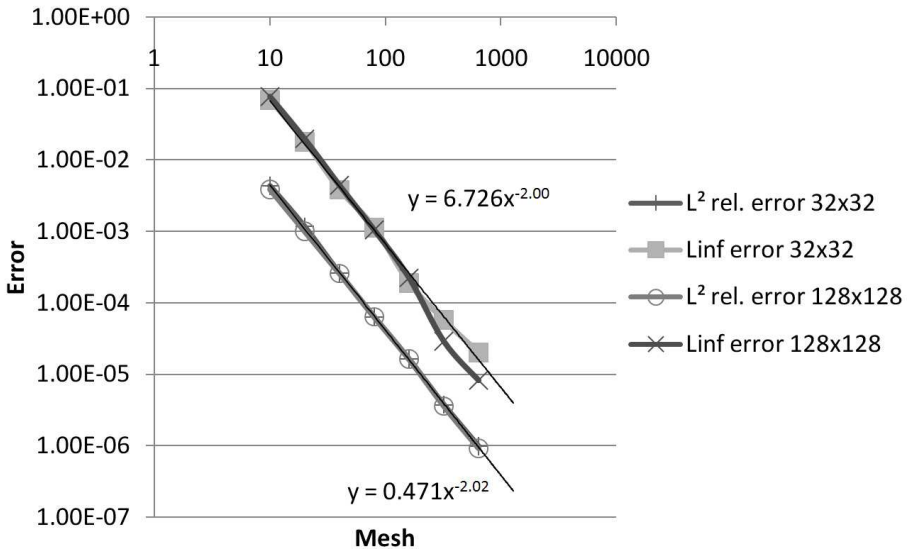

Figure 21: Convergence of the error with respect to the accuracy of the lagrangian shape problem 1

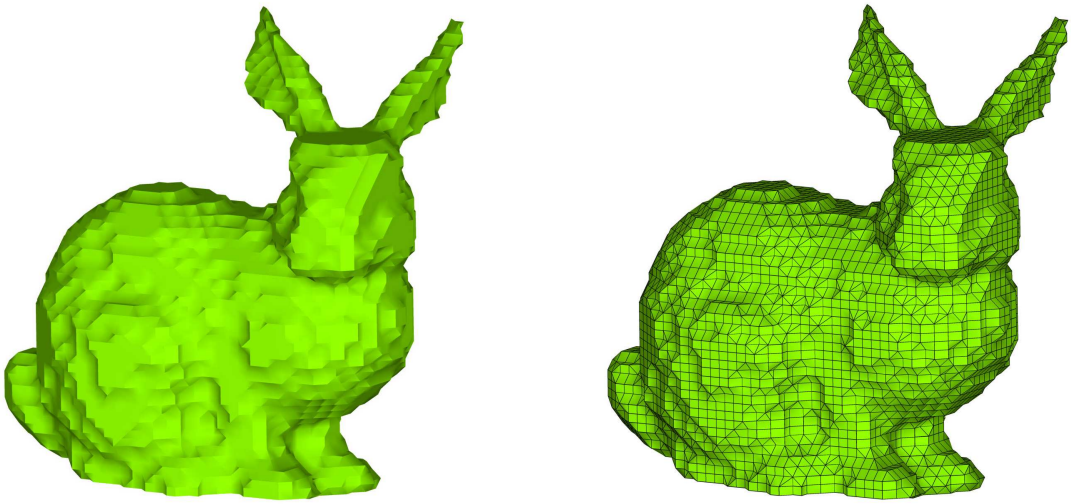

Figure 22: Iso-surface $T = 10$ for the Stanford bunny with a first order method

changing the interpolation by using a new node which is the image of x_I through the interface. In [9, 8], authors pointed out this problem and sug-

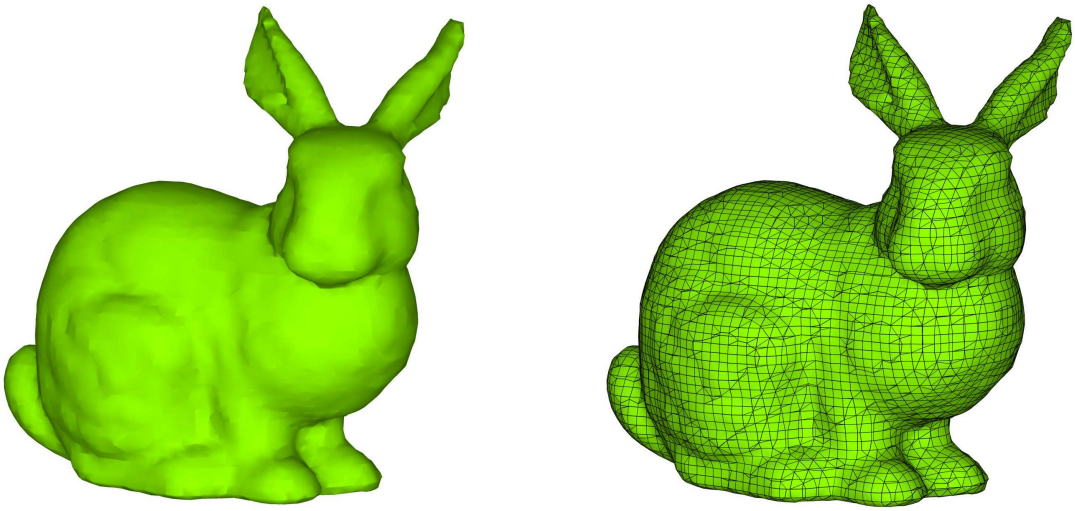

Figure 23: Iso-surface $T = 10$ for the Stanford bunny with a second order method

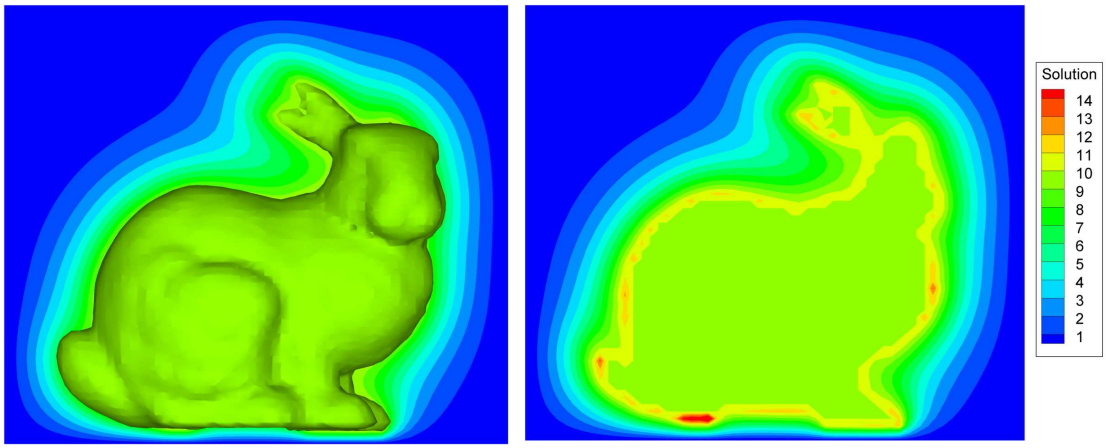

Figure 24: Iso-surface $T = 10$ and a slice of the solution

gest to slightly move the interface to a neighboring point (in our case x_J) if x_I is too close to Σ_h.

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

For the transmission conditions (21)-(22), 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 preconditionner or a PARDISO direct solver does not necessarily require such methods, even if $\frac{\alpha}{1-\alpha} \approx 10^{-10}$.

4 Discussion and conclusion

A new 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. A second order in space is reached for several configurations with minor modifications of the original code. The interface conditions are discretized 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. In general, 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. Compared to the MIB method [39] or to the IIM [18], this new method is easier to implement and uses a smaller stencil.

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 [27] to obtain an accurate two-way coupling.

Appendix : Definition of the interpolation

We 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 :

$$Q = \begin{pmatrix} 1 & x_I & y_I & x_I y_I \\ 1 & x_J & y_J & x_J y_J \\ 1 & x_K & y_K & x_K y_K \\ 1 & x_L & y_L & x_L y_L \end{pmatrix} \tag{33}$$

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

$$\xi = \frac{x_l - x_I}{x_J - x_I} \text{ and } \eta = \frac{y_l - x_I}{x_K - x_I}. \tag{34}$$

Fig. 25 illustrates the projection. Then, a unique trivial Q^{-1} matrix has to be found for each kind of interpolation. The coefficients for the $p \in \mathbb{Q}_1^2$ interpolation are the following :

$$\alpha_I = (1-\xi)(1-\eta) \tag{35}$$
$$\alpha_J = (1-\xi)\eta \tag{36}$$
$$\alpha_K = \xi\eta \tag{37}$$
$$\alpha_L = (1-\xi)\eta. \tag{38}$$

In [34], the authors uses three Eulerian points and an interface point to

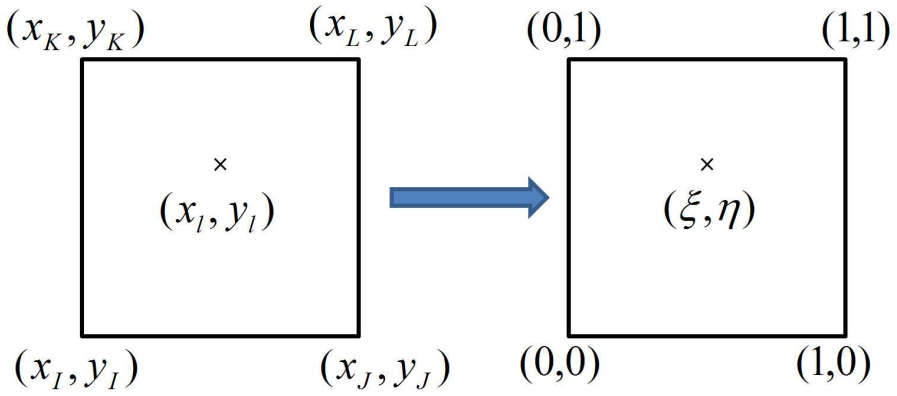

Figure 25: The projection of the initial square defined by the Eulerian mesh to an unit square

construct the interpolation providing a more complex linear system which has to be solved for each intersection point.

Acknowledgment

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

References

[1] Philippe Angot, Charles-Henri Bruneau, and Pierre Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81:497–520, 1999.

[2] Philippe Angot and J.-P. Caltagirone. In *Proceedings of the 2nd World Congress on Computational Mechanics, Stuttgart*, volume 1, pages 973–970, 1990.

[3] Federico Domenichini. On the consistency of the direct forcing method in the fractional step solution of the navier-stokes equations. *Journal of Computational Physics*, 227:6372–6384, 2008.

[4] E. A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, 161(1):35 – 60, 2000.

[5] Ronald P. Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics*, 152(2):457 – 492, 1999.

[6] F. R. Feito and J. C. Torres. Inclusion test for general polyhedra. *Computers & Graphics*, 21(1):23–30, 1997.

[7] M. Fortin and R. Glowinski. *Méthodes de lagrangien augmenté. Application à la résolution numérique de problèmes aux limites*. Dunod Paris, 1982.

[8] Frédéric Gibou and Ronald Fedkiw. 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(2):577–601, 2005.

[9] Frédéric Gibou, Ronald P. Fedkiw, Li-Tien Cheng, and Myungjoo Kang. A Second-Order-Accurate symmetric discretization of the poisson equation on irregular domains. *Journal of Computational Physics*, 176(1):205–227, February 2002.

[10] R. Glowinski, T. W. Pan, T. I. Hesla, and D. D. Joseph. A distributed lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25(5):755 – 794, 1999.

[11] I. Gustafsson. *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. la, 1978.

[12] T.Y. Hou, Z.L. Li, S. Osher, and H. Zhao. A hybrid method for moving interface problems with application to the hele-shaw flow. *Journal of Computational Physics*, 134:236–252, 1997.

[13] T. Ikeno and T. Kajishima. Finite-difference immersed boundary method consistent with wall conditions for incompressible turbulent flow simulations. *Journal of Computational Physics*, 226(2):1485–1508, 2007.

[14] Hans Johansen and Phillip Colella. A cartesian grid embedded boundary method for poisson's equation on irregular domains. *Journal of Computational Physics*, 147(1):60 – 85, 1998.

[15] Kodor Khadra, Philippe Angot, Sacha Parneix, and Jean-Paul Caltagirone. Fictitious domain approach for numerical modelling of navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 34:651–684, 2000.

[16] D. Lacanette, S. Vincent, A. Sarthou, P. Malaurent, and 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*, 52:2528–2542, 2009.

[17] Ming-Chih Lai and H-.C. Tseng. A simple implementation of the immersed interface methods for stokes flows with singular forces. *Computers and Fluids*, 37:99–106, 2008.

[18] Randall J. Leveque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal of Numerical Analysis*, 31:1001–1025, 1994.

[19] Z. Li. A fast iterative algorithmfor elliptic interfaces problems. *SIAM Journal of Numerical Analysis*, 35:230–254, 1998.

[20] Xu-Dong Liu, Ronald P. Fedkiw, and Myungjoo Kang. A boundary condition capturing method for poisson's equation on irregular domains. *Journal of Computational Physics*, 160(1):151 – 178, 2000.

[21] Peter McCorquodale, Phillip Colella, and Hans Johansen. A cartesian grid embedded boundary method for the heat equation on irregular domains. *Journal of Computational Physics*, 173(2):620 – 635, 2001.

[22] J. Mohd-Yusof. Combined immersed boundary/b-spline methods for simulations of flows in complex geometries. Technical report, NASA ARS/Stanford University CTR, 1997.

[23] Carlos J. Ogayar, Rafael J. Segura, and Francisco R. Feito. Point in solid strategies. *Computers & Graphics*, 29(4):616–624, 2005.

[24] Charles S. Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, 10(2):252–271, 1972.

[25] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[26] Isabelle Ramière, Philippe Angot, and Michel Belliard. A general fictitious domain method with immersed jumps and multilevel nested structured meshes. *Journal of Computational Physics*, 225(2):1347–1387, 2007.

[27] T.N. Randrianarivelo, G. Pianet, S. Vincent, and J.-P. Caltagirone. Numerical modelling of solid particle motion using a new penalty method. *International Journal for Numerical Methods in Fluids*, 47:1245–1251, 2005.

[28] Y. Saad and M.H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

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

[30] A. Sarthou, S. Vincent, J.-P. Caltagirone, and P. Angot. Eulerian-Lagrangian grid coupling and penalty methods for the simulation of multiphase flows interacting with complex objects. *International Journal for Numerical Methods in Fluids*, 56(8):1093–1099, 2008.

[31] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with pardiso. *Journal of Future Generation Computing Systems*, 20:475–487, 2004.

[32] S. Shin and D. Juric. Modelling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity. *Journal of Computational Physics*, 180:427–470, 2002.

[33] M. Sussman and E. Fatemi. An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *Siam Journal of Scientific Computing*, 20:1165–1191, 1999.

[34] Yu-Heng Tseng and Joel H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, 192(2):593–623, December 2003.

[35] R. Verzicco, G. Iaccarino, M. Fatica, and P. Orlandi. Flow in a impeller stirred tank using an immersed boundary method. Annual research brief, NASA Center for Turbulence Research, 2001.

[36] S. Vincent, A. Sarthou, J.-P. Caltagirone, F. Sonilhac, P. Février, C. Mignot, and G. Pianet. Augmented Lagrangian and penalty methods for the simulation of two-phase flows interacting with moving solids. Application to hydroplaning flows interacting with real tire tread patterns. Submitted to JCP.

[37] Sining Yu, Yongcheng Zhou, and G.W. Wei. Matched interface and boundary (mib) method for elliptic problems with sharp-edged interfaces. *Journal of Computational Physics*, 224(2):729 – 756, 2007.

[38] Y.C. Zhou and G.W. Wei. On the fictitious-domain and interpolation formulations of the matched interface and boundary (mib) method. *Journal of Computational Physics*, 219(1):228–246, 2006.

[39] Y.C. Zhou, Shan Zhao, Michael Feig, and G.W. Wei. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *Journal of Computational Physics*, 213(1):1 – 30, 2006.