

A global procedure for flow and heat transfer simulation with complex obstacles on curvilinear grids

Arthur Sarthou, Stéphane Vincent, Jean-Paul Caltagirone*,
Philippe Angot[†]

July 20, 2009

Abstract

A global methodology for simulating multiphase flows and heat transfers interacting with complex objects or interfaces is presented. Elliptic equations or Navier-Stokes equations are resolved on a fixed structured curvilinear grid and the solid objects are initially represented by triangular surface elements. Several difficulties arise as soon as these two non-conforming grids have to interact in the same physical problem, and accurate methods are presented for each issues: Lagrangian/Eulerian grid projection, immersed boundary of interface problems, and finally visualization. Hence, a new fast point-in-solid method for curvilinear grid is presented to project Lagrangian shapes on Eulerian grid. A new immersed boundary and interface method is presented, the Algebraic Immersed Interface method. Several validation and application problems are presented to demonstrate the interest and accuracy of the method.

1 Introduction and motivations

Simulation of real heat and mass transfers often implies interactions between multiphase flows and complex obstacles, or between heat transfers through irregular interfaces. Many simulation codes based on structured grids have shown their ability to deal with a large amount of physical phenomena. However, structured grids can not generally match complex interfaces due to their lack of flexibility, what makes complex shape problems unnatural and uneasy to treat with these codes. Many methods have been designed to improve the performances of structured grid codes with complex shapes. Such methods can be designated as fictitious domain methods. This term was originally defined in [14].

Two main classes of problems exist : the immersed boundary problems and the immersed interface problems. The first ones can be solved with the immersed

*Université de Bordeaux, TREFLE-ENSCPB, UMR CNRS 8508, Bordeaux, France (sarthou@enscpb.fr).

[†]Université de Provence, LATP-CMI, UMR CNRS 6632, Marseille, France

boundary method (IBM). Charles Peskin [8] designed the original method to treat flow problem in heart valves. His method uses discrete Dirac functions to distribute a singular interface force on the Eulerian grid. Recently, a direct-forcing IBM has been presented by Tseng et al. [9]. In [11, 12], Sarthou et al. use a quite similar approach coupled with a penalty method framework. Designed by Roland Glowinski, the Distributed Lagrangian Multiplier (DLM) method [4] uses Lagrange multiplier to enforce a solid behaviour in objects. Concerning problems with immersed interface, i.e. when the solution must be computed in both sides on the interface where special conditions are imposed, the immersed interface method (IIM) of Leveque and Li [6] has shown its ability to deal with such problems. The principle of IIM is to use Taylor series expansions to rewrite the operators discretization so as to accurately take into account interface conditions. Recently, the Matched Interface and Boundary (MIB) method has been presented by Zhou et al. [16]. Contrary to the IIM approach, the operators discretization is indirectly modified and the procedure remains the same no matter the initial discretization. However, fictitious domain methods are not the only required tools needed to treat problems with complex shapes, and some difficulties lies in the shape projection, i.e. the interpretation of a Lagrangian mesh on an Eulerian grid. The aim here is to obtain a fast and accurate projection for curvilinear grids for which the computational cost is negligible against the cost of the inversion of the main matrix of the conservation equation.

Many methods have been proposed to deal with immersed boundaries([10], [2]...). A global methodology is presented to treat complex shape immersed boundaries or interface problems in a curvilinear framework, from shape generation to visualization. Objects and interfaces are discretized as Lagrangian surface meshes composed of linear (in 2D) or triangular (in 3D) elements. Lagrangian meshes are then projected onto Eulerian grids using a curvilinear to Cartesian projection. This method allows the use of a fast Ray-Casting method to obtain a phase indicator of the Lagrangian mesh (a VOF function can be obtained using a level set function). Equations are then discretized, and linear systems are then modified using the Algebraic Immersed Interface method (AII), which is presented here for the first time. After the matrix inversion, special post-treatments can be applied to solutions allowing multiple uses, from scientific visualisation to general public movies.

2 Methodology

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 $\Omega \subset \mathbb{R}^d$. The auxiliary domain Ω_1 , typically a solid particle or an obstacle, is then such that : $\Omega = \Omega_0 \cup \Sigma \cup \Omega_1$ where Σ is an immersed interface (see Fig. 1 left). 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 with second order schemes on an Eulerian structured mesh covering Ω .

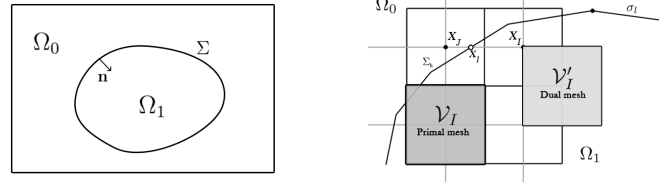


Figure 1: Definition of the domains and discretization kernels for the SMP method

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 indices of the Eulerian orthogonal curvilinear structured 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 discretize the conservation equations. A dual grid is introduced for the management of the AII 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 : $\Sigma_h = \{\sigma_l \in \mathbb{P}_1^{d-1}, l \in \mathcal{L}_f\}$, \mathcal{L}_f being the set of indices 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. 1 right). 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. As the discretization of the interface or boundary conditions require interpolations, we define: $\mathbb{P}_1^1(x) = \alpha_1 + \alpha_2 x$, $\mathbb{P}_1^2(x, y) = \alpha_1 + \alpha_2 x + \alpha_3 y$ and $\mathbb{Q}_1^2(x, y) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy$.

The cell centers x_I are sorted according to their location inside Ω_0 or Ω_1 thanks to the characteristic or color function C define hereafter. New sets of Eulerian points x_I are defined near the interface such as it exists one neighbor x_J verifying $C_J \neq C_I$, 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 so obtained, where $\mathcal{N}_0 = \{I, x_I \in \Omega_0, C_I \neq C_J, x_J \in \Omega_1\}$ and $\mathcal{N}_1 = \{I, x_I \in \Omega_1, C_I \neq C_J, x_J \in \Omega_0\}$.

Let us define now *auxiliary* entities which are superscripted with $*$, and 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 $\mathcal{N}_0^* = \{I, x_I^* \in \Omega_0, C_I \neq C_J, x_J^* \in \Omega_1\}$ and $\{x_I^*, I \in \mathcal{N}_1^*\}$ with $\mathcal{N}_1^* = \{I, x_I^* \in \Omega_1, C_I \neq C_J, x_J^* \in \Omega_0\}$. However, 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 $u(x_I) = u_I \neq u(x_I^*) = u_I^*$.

Let now define three useful functions on curvilinear grids:

- The discrete Heaviside function χ , defined as $\chi(x) = 1$ if $x_i \in \Omega_1$, 0 otherwise

This function is the binary indicator of the presence of an Eulerian point in Ω_1 and is built with a *point in solid* method presented below. The function χ will be used to perform fictitious domain algorithm and to build level-set functions.

- The level-set function ϕ , with

$$\phi_S(x_i) = \text{dist}_\Sigma(x_i) \text{ if } \chi(x) > 0.5, -\text{dist}_\Sigma(x_i) \text{ otherwise} \quad (1)$$

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

- The color phase functions C_i , which are the ratio of a given phase in a control volume.

This function is directly obtained from the ϕ function by using the formula proposed by Sussman and Fatemi [15] :

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

2.1.1 The curvilinear to Cartesian projection

Some optimisation in computing Eulerian functions can be performed when the grid lines of the Eulerian mesh are straight. The approach proposed here allow methods used on Cartesian grids to work in a curvilinear structured framework. The main idea is to first unfold the orthogonal curvilinear grid to obtain a dual Cartesian grid. Then, the objects are projected onto this new grid. The method is presented in 2D but can be generalized in 3D without difficulties. Let \hat{T}_h be the Cartesian structured mesh composed by elements $\hat{\mathcal{V}}'_i$, T_h being the primal orthogonal curvilinear grid. Let Π be the projector from T_h to \hat{T}_h . Hence, the discrete interface $\hat{\Sigma}_h$ is the projected interface such as $\hat{\Sigma}_h = \Pi(\Sigma_h)$. Each element $\hat{\mathcal{V}}'_i$ is a unit square, such as $\hat{\Omega}_h = [0, N] \times [0, M]$. The projection of Σ_h is performed by projecting each node of elements σ_i , denoted by σ_{ij} , $j = 1, 2$. Let (x_l, y_l) be the position of a node σ_{ij} and (x_k, y_k) , $k = 1, \dots, 4$ the position of each node K_{ij} of the element K_i containing σ_{ij} (see Fig. (2) for notations). Two Q_1 interpolations Q_x and Q_y are defined such as $Q_x(x_k, y_k) = \hat{x}_k$ and $Q_y(x_k, y_k) = \hat{y}_k$. The determination of the coefficients requires to solve two linear systems. The analytical solution is used in 2D and a BiCG-Stab method is used in 3D to obtain the projector coefficients. At last, $(Q_x(x_l), Q_y(y_l))$ gives the position of $\hat{\sigma}_{ij}$.

2.2 Point in solid algorithm

Many works (see [7] for a review) have been devoted to the point in solid problem which is a basic operation in computer graphics. The Jordan curve theorem-based method is chosen. From each Eulerian point x_i , a ray r_i is cast to infinity. An intersection test is performed between r_i and each element σ_i of Σ_h . If the ray intersects the Lagrangian surface Σ_h an even number of time, the point is

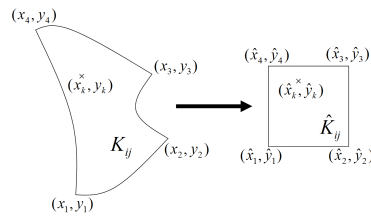


Figure 2: Notations and principle of the curvilinear to Cartesian projection. Original element K_{ij} and projected element \hat{K}_{ij} are described

outside the object. Else, the point is inside. A simple intersection algorithm can be found in [1]. Every ray can be cast in a same direction. On Cartesian structured grids, rays can follow grid lines which induces that many rays overlap themselves. Hence, a simple and efficient optimization is to send one ray per Eulerian points row. This optimization is not *a priori* possible on curvilinear structured grids if grid line are not straight, and a key point of our method rely on the curvilinear to Cartesian projection to allow this very useful optimisation. Many other optimizations can be performed to obtain a negligible computational cost, and contrary to the front-tracking technique used in [10], the ray-casting can deal with large surface elements and works even if Σ_h intersect the boundary of the numerical domain.

2.3 The Algebraic Immersed Interface 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). Sub-Mesh Penalty method [11] has been previously used to treat immersed boundaries, but cannot deal with immersed interfaces. The Algebraic Immersed Interface (AII) method is an enhancement of the SMP method [12] which is able to solve immersed interface problems too. The main idea of the AII 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 Ghost Fluid [3] and Immersed Interface Methods [6]), the AII 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 AII 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 original problem $Au = b$ becomes $A'u' = b'$, with A' a square matrix of order $M + N$, with N the number of auxiliary constraints related to interface conditions. The solution u' is decomposed as $u' = u + u^*$ and the source term as $b' = b + b^*$.

2.3.2 AII algorithm for a scalar equation with Dirichlet boundary conditions

A model scalar problem is first considered with a Dirichlet boundary conditions (BC) on the interface Σ :

$$\begin{cases} -\nabla \cdot (a\nabla u) = f & \text{in } \Omega_0 \\ u = u_D & \text{on } \Sigma \end{cases} \quad (3)$$

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

For sake of clarity, let us first describe in $2D$ the AII method for the model scalar problem \mathcal{P} with a Dirichlet boundary condition on the interface Σ . For this version of the AII 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 $I \in \mathcal{N}_1^*$. 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 interface unknown u_l is approximated by the \mathbb{P}_1^1 -interpolation between the Eulerian unknowns u_I^* and u_J :

$$u_l = \lambda_I u_I^* + \lambda_J u_J \text{ with } 0 < \lambda_I, \lambda_J < 1 \text{ and } \lambda_I + \lambda_J = 1 \quad (4)$$

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 :

$$u_p = \lambda_I u_I^* + \lambda_J u_J + \lambda_K u_K, \quad 0 < \lambda_I, \lambda_J, \lambda_K < 1, \quad \lambda_I + \lambda_J + \lambda_K = 1 \quad (5)$$

We can also use a \mathbb{Q}_1^2 -interpolation of u_I , u_J , u_K and u_L , 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 \mathbb{P}_1^1 -interpolations can be used (one for each direction) for a quasi equivalent result. It produces :

$$\begin{cases} u_l = \lambda_I u_I^* + \lambda_J u_J \text{ with } 0 < \lambda_I, \lambda_J < 1 \text{ and } \lambda_I + \lambda_J = 1 \\ u_m = \lambda'_I u_I^* + \lambda'_K u_K \text{ with } 0 < \lambda'_I, \lambda'_K < 1 \text{ and } \lambda'_I + \lambda'_K = 1 \end{cases} \quad (6)$$

In this case, two auxiliary unknowns are created. If x_I has a third neighbor, this last method is directly usable. For the other methods, the solution is directly imposed on x_I , and $u_i = u_D$. This case happens rarely, then the overall second order of convergence is not affected by this first order approximation.

2.3.3 AII algorithm for a scalar equation with Neumann boundary conditions

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

$$\begin{cases} -\nabla \cdot (a\nabla u) = f & \text{in } \Omega_0 \\ (a \cdot \nabla u) \cdot \mathbf{n} = g_D & \text{on } \Sigma \end{cases} \quad (7)$$

The principle is quite 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,

$$(a \cdot \nabla u_I) \cdot \mathbf{n} \approx (a \cdot \nabla p(x_I) \cdot |\mathbf{n}|) \tag{8}$$

For $p \in \mathbb{Q}_1^2$, we obtain $\nabla p(x, y) \cdot |\mathbf{n}| = (a_3 y + a_2)|n_x| + (a_3 x + a_1)|n_y|$. For $p \in \mathbb{P}_1^2$, we obtain $\nabla p(x, y) \cdot |\mathbf{n}| = a_2|n_x| + a_1|n_y|$ which means that the normal gradient is approximated as being constant over the whole support. Nevertheless, a second order in space can be obtained on Cartesian meshes for the solution with such an interpolation.

As can be noticed, $|\mathbf{n}|$ is used instead of \mathbf{n} to avoid numerical instabilities: if $n_x \approx -n_y$ and $h_x \approx h_y$, the diagonal coefficient of the line related to u_I tends to 0.

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{Q}_1^1 and \mathbb{P}_1^1 interpolations which suits well for Dirichlet BC. For Neumann BC, this loss of dimension no more allows to take into account the interface orientation, 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 appears, we take the point such that the angle $(\mathbf{n}, x_K - x_J)$ is in $[-\pi/2; \pi/2]$.

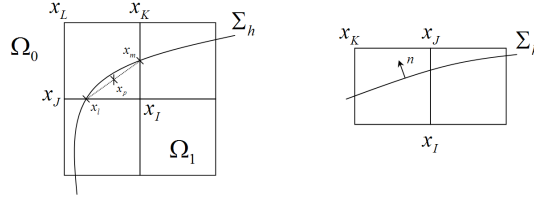


Figure 3: Example of selection of point for Dirichlet (left) and Neumann (right) constraints

2.3.4 Symmetric version for Dirichlet boundary conditions

The next step is to allow multiple Dirichlet BC on both sides of the immersed interface. The problem is now :

$$\begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega \\ u|_{\Sigma}^- = u_D & \text{on } \Sigma \\ u|_{\Sigma}^+ = u_G & \text{on } \Sigma \end{cases} \tag{9}$$

Contrary to the SMP method, the AII method do not use physical nodes in one side of the interface to increase the accuracy of the solution on the other side. The AII method always creates auxiliary nodes, so the two precedent requirements on Σ can be respected. Hence, the AII algorithm for immersed boundaries is performed two times for both sides of the interface. Practically, the algorithm is easily performed one time using C and a second time using $C' = 1 - C$.

2.3.5 Algebraic elimination

The symmetrical AII method creates N auxiliary unknowns for which a boundary constraint is written. Hence, the method increases the size of the solved matrix. However, the discretization of the problem at lines $I \in \mathcal{N}^*$ contains only one auxiliary unknown at the same time, and we have :

$$u_I^* = \sum_{J \in \mathcal{N}} \alpha_J u_J \quad (10)$$

Hence, occurrences of u_I^* at lines $J \in \mathcal{N}$ can be easily replaced by $\sum_{J \in \mathcal{N}} \alpha_J u_J$.

The matrix obtained with this method is quite similar to the ones proposed in [3]. However in this last paper the auxiliary unknowns are taken into account before the discretization of the operator which is then modified, and a particular work is needed for each discretization scheme. Our algorithm seems simpler, as the standard discretization of the operators is automatically modified in an algebraic way. So, no particular attention has to be paid to the discretization scheme used.

2.4 AII 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 condition 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 :

$$(\mathcal{P}_{ii}) \quad \begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega \\ + \text{Interface condition} & \text{on } \Sigma \end{cases} \quad (11)$$

where interface conditions are :

$$\llbracket u \rrbracket_{\Sigma} = \varphi \quad \text{on } \Sigma \quad (12)$$

$$\llbracket (a \cdot \nabla u) \cdot \mathbf{n} \rrbracket_{\Sigma} = \psi \quad \text{on } \Sigma \quad (13)$$

where $\llbracket \cdot \rrbracket_{\Sigma}$ denote the jump of a quantity over the interface Σ . As can be seen with the symmetric version of the AII 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 constraint. Hence, an intersection point can be associated to the two interface constraints (12) and (13) 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 (12) and the J^{nth} line of the matrix with $x_J^* \in \Omega_1$ is then used to impose constraint (13).

2.4.1 The solution constraint

The 1D symmetrized AII methods for Dirichlet BC holds :

$$\begin{cases} u_{\Sigma}^+ = \alpha u_I + \beta u_J^* \\ u_{\Sigma}^- = \alpha u_I^* + \beta u_J \end{cases} \quad (14)$$

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 $\llbracket u \rrbracket_\Sigma = u_\Sigma^+ - u_\Sigma^- = \varphi$, we obtain :

$$\alpha u_I + \beta u_J^* - \alpha u_I^* - \beta u_J = \varphi \quad (15)$$

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,

$$\begin{cases} (a \cdot \nabla u_\Sigma^+) \cdot \mathbf{n} = a^+ \left(\frac{u_J^* - u_I}{h_x} |n_x| + \frac{u_K^* - u_I}{h_y} |n_y| \right) \\ (a \cdot \nabla u_\Sigma^-) \cdot \mathbf{n} = a^- \left(\frac{u_J - u_I^*}{h_x} |n_x| + \frac{u_K - u_I^*}{h_y} |n_y| \right) \end{cases} \quad (16)$$

and using (13), we obtain:

$$a^+ \left(\frac{u_J^* - u_I}{h_x} |n_x| + \frac{u_K^* - u_I}{h_y} |n_y| \right) - a^- \left(\frac{u_J - u_I^*}{h_x} |n_x| - \frac{u_K - u_I^*}{h_y} |n_y| \right) = \psi \quad (17)$$

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.

2.4.3 Algebraic reduction

As interpolation functions are of various dimensions, a given auxiliary unknown can be involved in the discretization of many constraints. Hence, auxiliary unknowns are strongly interdependent and a simple algebraic reduction of the matrix is no more possible. For the Matched Interface and Boundary (MIB) method, Zhou et al. [16] used a different discretization of the interface conditions. Combined with such a discretization and then algebraically reduced, the AII method provides the same matrix as the MIB method. However, the standard discretization of the AII method requires a more compact stencil, and the additional computational time generated by the auxiliary nodes is small.

2.4.4 Application to physical equations

All the derivations of the AII method can be applied to elliptic equations for solving numerous heat transfers problems or velocity projection. Concerning the Navier-Stokes equations, [11] shown the ability of this approach to treat fluid flow problems with immersed boundaries. For vector equations, the method used for scalar equations is repeated for each component. No other modification are required if the velocity-pressure coupling is an augmented Lagrangian method. Modifications are required for time-splitting methods as they require a predictor and a projection steps that induce the definition of different boundary constraints in the two steps. Accuracy of the two approaches will be investigated in a future work.

2.5 Post-treatment and scientific visualization

Multiphase flows across obstacles are well suited for general public movies. Such animations are generated as follows: first, a realistic scene is designed with a computer graphics (CG) software such as 3D Studio Max or Blender. The scene is composed of triangularized surfaces. The surfaces interacting with fluid simulation are extracted and defined as immersed boundaries in a CFD code. The simulation of the fluid flow is then performed and at some regular time step, a triangularized surface is extracted from the Eulerian description C_f of the fluid free surface. These surfaces are then exported into the same CG software. At last, a realistic rendering is computed (see Fig. 4). Many animations can be found in [13].



Figure 4: Dam break flow over an obstacle (left)-Dam break flow on a realistic topography (right)

3 Validation and examples

As AII and SMP methods are equivalent for IB problems, validations for such cases on Cartesian grids can be found in [11] and [12]. New validations for IB problems are performed on two curvilinear grids (see Fig. 5). Grid A is an orthogonal mesh with exponential periodic steps. Grid B is a converging pipe. The homogenous Laplace equation between two circles of radius $R_1 = 0.5$ and

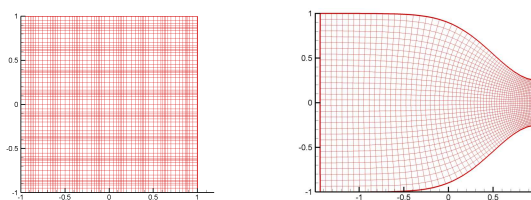


Figure 5: Curvilinear grids used for validation : Grid A (left) - Grid B (right)

$R_2 = 4$ is first solved. The solution is $u_1 = 10$ on the first circle and $u_2 = 0$ on the second circle. The boundary condition on the inner circle is imposed with AII method and the analytical solution which account for the exterior circle is a second order is reached for the L^2 norm. For the L^∞ norm the accuracy is imposed on the boundary of the Eulerian grid. As can be seen in Tab. 1, globally of order 1.7.

An II problem is now solved on a Cartesian grid. The interface Σ is a centered circle of radius $r = 0.5$. We solve the problem \mathcal{P}_{ii} with $f = -4$ and $a = 1$. As the equation remains exactly the same in both domains, this

Mesh A	L^2 relative error	Order	L^∞ error	Order
32×32	7.40×10^{-4}	---	1.64×10^{-2}	---
64×64	1.88×10^{-4}	1.98	5.19×10^{-3}	1.66
128×128	4.48×10^{-5}	2.07	1.72×10^{-3}	1.59
256×256	1.81×10^{-5}	1.31	6.82×10^{-4}	1.34
512×512	3.84×10^{-6}	2.23	1.42×10^{-4}	2.26
1024×1024	8.20×10^{-7}	2.23	3.80×10^{-5}	1.90

Mesh B	L^2 relative error	Order	L^∞ error	Order
32×64	3.60×10^{-4}	---	1.07×10^{-2}	---
64×128	6.82×10^{-5}	2.40	2.49×10^{-3}	2.10
128×256	2.26×10^{-5}	1.59	7.80×10^{-4}	1.67
256×512	5.10×10^{-6}	2.15	2.80×10^{-3}	1.48

Table 1: Accuracy results for 2D Poisson equation with immersed boundary condition for grids A and B

problem can be solved without immersed interface method. The analytical solution is $u = r^2$. As can be expected with our second order code, the error machine is reached for all meshes with or without AII method. Similar results are obtained with an interface Σ parametrized by $(x(\theta), y(\theta))$ where: $x(\theta) = .02\sqrt{5} + (.5 + .2 \sin(5\theta)) \cos(\theta)$, $y(\theta) = .02\sqrt{5} + (.5 + .2 \sin(5\theta)) \sin(\theta)$, with $\theta \in [0, 2\pi]$.

We consider now the same problem with a discontinuous coefficient: $a = 10$ in Ω_0 , and $a = 1$ in Ω_1 with the following analytical solution : $u = r^2$ in Ω_0 and $u = r^2/10 + 0.9/4$ in Ω_1 . As can be seen on Tab. 2, a second order is reached for

Mesh	L^2 relative error	Order	L^∞ error	Order
16×16	3.22×10^{-3}	---	3.08×10^{-3}	---
32×32	7.75×10^{-4}	2.06	6.97×10^{-4}	2.14
64×64	2.39×10^{-4}	1.69	2.03×10^{-4}	1.78
128×128	5.02×10^{-5}	2.25	4.29×10^{-5}	2.24
256×256	1.47×10^{-5}	1.77	1.37×10^{-5}	1.65
512×512	2.73×10^{-6}	2.43	2.87×10^{-6}	2.26
1024×1024	7.08×10^{-7}	1.95	6.98×10^{-7}	2.04

Table 2: Accuracy results for 2D Poisson equation with discontinuous coefficient Cartesian grids.

4 Conclusion and perspective

A new and easy to implement fictitious domain method for IB and II problems has been designed. On Cartesian grids, the method reaches a second order in space for both problems. On curvilinear grids, a second order is obtained with IB problems only. Hence, future works will be devoted to obtain a second order in space for Curvilinear grids with II problems. Extension of the AII method in 3D for II problems will be performed too.

References

- [1] F. Badouel, *An efficient Ray-Polygon intersection*, Graphic Gems, Academic Press (1990), pp. 390–393.

- [2] L. Ge, F. Sotiropoulos, *A numerical method for solving the 3D unsteady incompressible Navier-Stokes equations in curvilinear domains with complex immersed boundaries*, J. Comput. Phys. 225 (2007), pp. 1792–1809.
- [3] 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), pp. 1–23.
- [4] 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), pp. 755–794.
- [5] 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), pp. 651–684.
- [6] R.J. LeVeque, Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal. 31(2004), pp. 1001–1025.
- [7] C. J. Ogayar, R. J. Segura, F. R. Feito, *Point in solid strategies*, Comp. and Graph. 29 (2005), pp. 616–624.
- [8] C.S. Peskin, *The Immersed Boundary Method*, Acta Numerica 11 (2000), pp. 479–517.
- [9] Y.H. Tseng, J.H. Ferziger, *A ghost-cell immersed boundary method for flow in complex geometry*, J. Comput. Phys. 623 (2003), pp. 192–593.
- [10] 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*, Proceedings of ICFD07, Reading, UK (2007).
- [11] 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 (8) (2008), pp. 1093–1099.
- [12] A. Sarthou, S. Vincent, P. Angot, J.P. Caltagirone, *The Sub-Mesh Penalty Method*, Proceedings of FVCA5 (2008), pp. 633–640.
- [13] A. Sarthou, S. Vincent, *The MKF project*, <http://projetmkf.free.fr> (in French)
- [14] V.K. Saul'ev, *On the solution of some boundary value problems on high performance computers by fictitious domain method*, Siberian Math. J. 4 (1963), pp. 912–925.
- [15] 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), pp. 1165–1191.
- [16] Y.C. Zhou, S. Zhao, M. Feig, G.W. Wei, *High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources*, J. Comput. Phys. 213 (2006), pp. 1–30.