A Second Order Penalized Direct Forcing for Hybrid Cartesian/Immersed Boundary Flow Simulations

Clement Introini, Michel Belliard, Clarisse Fournier

A Second Order Penalized Direct Forcing for Hybrid Cartesian/Immersed Boundary Flow Simulations

C. Introïni[a], M. Belliard[a], C. Fournier[b]

[a] *Commissariat à l'Énergie Atomique et aux énergies alternatives*
CEA, DEN/DANS/DM2S/STMF/LMEC
CEA Cadarache, Bât. 238*, F-*13108 *St Paul-lez-Durance*
[b] *Commissariat à l'Énergie Atomique et aux énergies alternatives*
CEA, DEN/DANS/DM2S/STMF/LMSF
CEA Grenoble, Bât 1005*,* 17 *rue des Martyrs, F-*38054 *Grenoble Cedex* 9

Abstract

In this paper, we propose a second order penalized direct forcing method to deal with fluid-structure interaction problems involving complex static or time-varying geometries. As this work constitutes a first step toward more complicated problems, our developments are restricted to Dirichlet boundary condition in purely hydraulic context. The proposed method belongs to the class of immersed boundary techniques and consists in immersing the physical domain in a Cartesian fictitious one of simpler geometry on fixed grids. A penalized forcing term is added to the momentum equation to take the boundary conditions around/inside the obstacles into account. This approach avoids the tedious task of re-meshing and allows us to use fast and accurate numerical schemes. In contrary, as the immersed boundary is described by a set of Lagrangian points that does not generally coincide with those of the Eulerian grid, numerical procedures are required to reconstruct the velocity field near the immersed boundary. Here, we develop a second order linear interpolation scheme and we compare it to a simpler model of order one. As far as the governing equations are concerned, we use a particular fractional-step method in which the penalized forcing term is distributed both in prediction and correction equations. The accuracy of the proposed method is assessed through 2-D numerical experiments involving static and rotating solids. We show in particular that the numerical rate of convergence of our method is quasi-quadratic.

Keywords: immersed boundaries, penalized direct forcing, projection scheme, Cartesian

grids, interpolation scheme

1. Introduction

Fluid flow with heat and mass transfer around complex stationary or moving geometries (solid or flexible) appears in a large number of situations of practical interest including biological fluid mechanics (blood flow in human heart for instance) or in life-science context (the fish-like swimming *e.g.*). Fluid-structure interaction problems are also of importance in many engineering applications, as for example, to design industrial heat exchangers, aerospace vehicles or in nuclear safety context. In this latter case, the vitrification process for the radioactive waste storage is an example. In this process, a viscous multiphase multicomponent flow at high temperature (gas bubbles and molten glass incorporating the ultimate waste) interacts with both static (*e.g.* the vessel structure, the apparatus of measurement, ...) and moving (*e.g.* the mechanical stirrer) bodies of more or less complex geometries.

The numerical treatment of these kinds of problem appears to be a challenging task because of time-varying geometries, often combined with complex flow regimes. To tackle numerically these complex problems, the well-known body-fitted approach is usually followed. Such an approach consists in discretizing the governing equations on a non-structured mesh for which the boundaries of the computational domain lie on those of the physical domain. Thereby, boundary conditions are directly (and so, exactly) imposed on the physical domain boundary. However, the main drawback of the body-fitted like techniques lies in their lack of ability to handle complex industrial problems involving moving bodies which require the development of specific numerical schemes to deal with the difficult issue of re-gridding.

Another approach consists in using non-boundary conforming techniques in which the physical domain is immersed in a fixed fictitious one of simpler geometry on a Cartesian grid. Such techniques allow us to use efficient, fast and accurate numerical methods avoiding the tedious task of the re-meshing caused by time-varying geometries. In contrast, as the immersed boundaries are described by a set of Lagrangian points (or the zero of a level-set function) that do not generally coincide with those of the Eulerian grid, numerical methods have to account the immersed boundary conditions at their right places. The non-boundary

conforming techniques proposed in the literature may be classified into two categories.

The first category, including for instance Cartesian methods (*e.g.* [1, 2]), the Immersed Interface Method (IIM, [3]) or the Jump Embedded Boundary Condition method (JEBC, [4]), mimics the presence of embedded geometries by modifying the numerical scheme in the immediate vicinity of the immersed boundary or interface. The two latter methods introduce jump conditions across the interface in the solve of the partial differential equations. Such an approach leads to a sharp representation of the immersed interface but, for the Cartesian method, extending it to three-dimensional problems may appear to be a challenging task, particularly regarding the coding logistic.

In the second category (rather than locally modifying the numerical scheme) a supplementary term, referred to as the forcing term, is added to the governing equations. This class of non-boundary conforming techniques dates back to Peskin's works in which an Immersed Boundary Method (IBM) has been developed to numerically simulate blood flows in a human heart [5]. In this case, the immersed boundaries correspond to muscular heart walls and extra forces acting in these boundaries are modeled by a vectorial forcing term added to the continuous Navier-Stokes equations. A Lagrangian coordinate system is employed to track the interface and to calculate the vectorial forcing term. The IB method has been successfully applied to problems with elastic geometries but, in the rigid limit, it generally leads to very stiff problems. Moreover, in order to ensure the stability of the numerical scheme, the forcing term based on a Dirac delta function must be smeared over a stencil of few Cartesian nodes. Following the ideas introduced by Peskin, several IB-like methods with different forcing terms (or forcing strategies) have been proposed in the literature. In [6], Goldstein *& al.* propose the Feedback Forcing (FF) method in which the forcing term can be viewed as a force density that brings the fluid velocity to zero near the immersed boundary. Similarly to what is done in [5], the numerical scheme used in [6] requires a spreading of the forcing term over the interface. Moreover, the FF method suffers from the fact that the forcing term highly depends on flow properties. Whether the Peskin's IB method or the FF method, their application to flows at high Reynolds number is limited by the spreading of the forcing term over the immersed boundary. In this case, local mesh refinement techniques

can be a solution [7]. An alternative approach to the aforementioned techniques, referred to as Direct Forcing (DF) method, has been proposed by Mohd-Yusof [8] and then adapted by Fadlun *& al.* [9]. This immersed boundary technique consists in directly applying the desired boundary conditions on Cartesian nodes close to the interface leading to a quasi sharp representation of the interface (through one cell layer). In that sense, using the terminology employed by Gilmanov *& al.* in [10], the DF method may be referred to as a Hybrid Cartesian/Immersed Boundary (HCIB) approach and may be conceptually related to the IIM. Moreover, one of the interest of the DF method is that the forcing term can be easily computed and it does not depend on the flow properties. Therefore, the stability of the numerical scheme is not affected. However, the accuracy of the DF method is partially dependent on the numerical scheme because the calculation of the forcing term is in particular based on the discretized form of the governing equations. Since its development by Mohd-Yusof [8], the DF method has gained in popularity and has been successfully applied to various fluid-structure interaction problems (*e.g.* [10–15]) or turbulent flow simulations (*e.g.* [16, 17]) using mesh refinement or mesh stretching techniques. It is also worth to mention the immersed boundary method of Pinelli *& al.* [18], that has roots in both IBM and DF methods, and which is suitable for general grid systems including curvilinear ones. More recently, Belliard & Fournier [19] have proposed a variant of HCIB techniques, called Penalized Direct Forcing (PDF) method, that combines both the basic features of the DF method and those of L^2-penalty methods (*e.g.* [20]). Links can be found with the works of Sarthou *& al.* [21] and those of Bergmann & Iollo [22]. As for the DF method, the unknowns are locally enforced on the grid nodes nearest the immersed interface. However, the PDF method appears to be a more versatile approach than the DF method because the forcing term expresses as a L^2-penalty term that is independent on the discrete governing equations.

In the present paper, after introducing the discretization of the Navier-Stokes governing equations in the Section 2, the PDF algorithm is detailed in the Section 3, including a specific treatment of the pressure near the immersed boundaries. The PDF method itself is presented in Section 3.1. Interesting for practical purposes, the non-boundary conforming

approaches are often coupled with fractional-step schemes (*e.g.* [23, 24]) but, as emphasized by Ikeno & Kajishima [25] or Taira & Colonius [26], most of them take account of the forcing term only in the prediction equation leading to inconsistent schemes. Here, an original fractional-step scheme leading to a consistent (in the sense of [25]) PDF method is developed and presented in Section 3.2. Homogeneous Neumann IBCs for the pressure are recovered through a particular treatment of the pressure equation coefficients near the immersed interface.

Whatever the non-boundary conforming method involved, an important issue concerns the reconstruction of the velocity field close to the immersed boundary and the accuracy of the numerical method developed for this purpose. These points have received a particular attention in the literature with, most of the time, the development of interpolation schemes. Most of the classical approaches consist in interpolating or extrapolating the velocity field in a preferred direction (*e.g.* [10, 12]). In this work, we have developed an original robust interpolation scheme, second-order accurate in space, that is not guided by particular direction. It relies mainly on an averaged reconstruction of the velocity gradient near the IB and on an approximate projection operator onto the IB. Without loss of generality, we restrict our presentation to Dirichlet's IBCs for the velocity[1]. This is the object of Section 4.

Finally, in Section 5, some 2-D numerical experiments are performed for steady and unsteady incompressible laminar flows at very moderate Reynolds number (up to 100) around/between static and rotating solids to assess the validity, accuracy and the ability of the proposed method for both uniform and analytically velocity prescribed at IBs. We show in particular that the numerical rate of convergence is (quasi-)quadratic for all studied cases. To highlight the ability of our method to deal with 3-D moving geometries, we also present an illustration of flows induced by a stirrer .

[1]Neumann IBCs can be also considered by interpolations involving the prescribed flux at the boundary and the velocities of the surrounding flow.

2. Governing equations and numerical method

The section is devoted to the numerical method. It is structured in two parts. The first part focuses on the governing equations whereas the numerical scheme is the object of the second part.

2.1. Governing equations

The governing equations used to describe unsteady incompressible flows are given by:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \nabla P - \nu \nabla^2 \mathbf{u} = \mathbf{f} \quad \text{in } \Omega \tag{1a}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \tag{1b}$$

$$\mathbf{u} = \mathbf{u}_D \quad \text{on } \partial\Omega \text{ and } \mathbf{u}(t_0) \text{ given in } \Omega \tag{1c}$$

where Ω denotes the computational domain, $\partial\Omega$ its boundary, $\mathbf{u}$ the solenoidal velocity and ν the kinematic viscosity. Here-above, P is the total pressure defined by:

$$\rho \nabla P = \nabla p - \rho \mathbf{g} \tag{2}$$

where p is the hydrodynamic pressure, ρ the constant density and $\mathbf{g}$ the gravity force. For the sake of clarity, we assume full Dirichlet boundary conditions and we consider the volume force $\mathbf{f} = 0$ in the sequel of the paper.

2.2. Numerical scheme

Here, we focus on the numerical scheme used in this work. On the first hand, we present the space and time discretizations. On the second hand, we discuss about the fractional-step method used to solve the Navier-Stokes equations.

2.2.1. Time and space discretizations

The time marching of the velocity is performed by means of a variant of the degenerate fourth-order explicit Runge-Kutta scheme of Williamson [27]. Given an initial condition

$\mathbf{u}^0 = \mathbf{u}(t_0)$ and $\frac{\partial \mathbf{u}}{\partial t} = f(\mathbf{u})$, the new velocity $\mathbf{u}^{n+1} = \mathbf{u}(t_0 + (n+1)\Delta t)$ is obtained by:

$$\begin{aligned} \forall\, n \in \mathbb{N},\ \mathbf{u}^{n+1} &= \mathbf{u}^n + \beta_1 q_1 + \beta_2 q_2 + \beta_3 q_3 \\ \text{where} \qquad & \beta_1 = \frac{1}{2} \quad \text{and } q_1 = \Delta t f\left(\mathbf{u}^n\right), \\ & \beta_2 = 1 \quad \text{and } q_2 = \Delta t f\left(\mathbf{u}^n + \frac{1}{2} q_1\right) - \frac{1}{2} q_1, \\ & \beta_3 = \frac{1}{6} \quad \text{and } q_3 = \Delta t f\left(\mathbf{u}^n + \frac{1}{2} q_1 + q_2\right) - 2 q_2. \end{aligned} \tag{3}$$

This scheme is degenerated because it only uses two storage vectors and three computational steps like the third-order Runge-Kutta scheme. In our variant, at each step k of the Runge-Kutta scheme, we get $f(\mathbf{u}^n + \ldots)$ by solving the couple $(\mathbf{u}, P)$ using a fractional-step method (*cf.* Section 2.2.2) for which the temporal discretization is based on a semi-implicit scheme which relies on an explicit discretization of the convection term and an implicit discretization of the diffusion.

As far as the space discretization is concerned, it is based on a finite volume approximation with a staggered grid arrangement of the primitive variables $(\mathbf{u}, P)$. As a result, the pressure degrees of freedom are located at the cell centers whereas those of each velocity component are placed at the middle of the cell edges.

[Figure 1 about here.]

Fig. 1 presents an example of the staggered arrangement of the unknowns on a 2-D Cartesian cell $\Omega_{i,j}$. In this figure, $\Omega_{i+1/2,j}$ and $\Omega_{i,j+1/2}$ are respectively the control volumes of the components $u_{i,j}$ and $v_{i,j}$ of the velocity field. Note that the control volume of the pressure $P_{i,j}$ coincide with the cell $\Omega_{i,j}$. In this frame, the governing equations Eqs. (1) are integrated over each control volume ensuring the conservation of mass and momentum balances. Moreover, the convection and diffusion terms are respectively approached by the QUICK and the centered schemes [28].

Now, denoting by $\star_h$ a discrete space operator associated to $\star$, at each step $k = 1, \ldots, 3$

of the Runge-Kutta scheme (3), the discrete form of the governing equation Eqs. (1) reads:

$$f(\mathbf{u}^{k-1}) := \frac{\mathbf{u}^\star - \mathbf{u}^{k-1}}{\Delta t} = -\nabla_h \cdot \left(\mathbf{u}^{k-1} \otimes \mathbf{u}^{k-1}\right) - \nabla_h P^\star + \nu \nabla_h^2 \mathbf{u}^\star \quad \text{in } \Omega \tag{4a}$$

$$\nabla_h \cdot \mathbf{u}^\star = 0 \quad \text{in } \Omega \tag{4b}$$

$$\mathbf{u}^\star = \mathbf{u}_D \quad \text{on } \partial\Omega \tag{4c}$$

where $\mathbf{u}^k = \mathbf{u}^n + \sum_{i=1}^{k} \beta_i q_i$, $\mathbf{u}^0 = \mathbf{u}^n$ and $(\mathbf{u}^{n+1}, P^{n+1}) = (\mathbf{u}^3, P^\star)$. Here, $(\mathbf{u}^\star, P^\star)$ stands for a solenoidal velocity and a pressure obtained by a projection method as described in the following part.

2.2.2. A fractional step algorithm to solve the Navier-Stokes equations

The fractional-step method or projection method has been introduced by Chorin and Temam in 1968 for incompressible flows [23, 24]. On the basis of their work, many variants have been proposed, as for example, the incremental projection method for incompressible flows [29], the projection schemes for dilatable or barotropic fluids [30, 31] or, more recently, the novel fractional time stepping technique massively parallel for incompressible Navier-Stokes equations developed by Guermond & Minev [32], to cite among others. We refer the reader to [33] for a recent review of these methods.

In this paper, we use a non-incremental fractional-step scheme. The first step consists in solving a predicted velocity $\widetilde{\mathbf{u}}$ without pressure gradient term as follows[2]:

$$\frac{\widetilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \nabla_h \cdot (\mathbf{u}^n \otimes \mathbf{u}^n) - \nu \nabla_h^2 \widetilde{\mathbf{u}} = 0 \quad \text{in } \Omega \tag{5a}$$

$$\widetilde{\mathbf{u}} = \mathbf{u}_D \quad \text{on } \partial\Omega. \tag{5b}$$

Then, the second step corresponds to a correction stage which consists in computing a new pressure P^{n+1} and recovering a new solenoidal velocity $\mathbf{u}^{n+1}$. By assuming that $\nabla_h^2 \widetilde{\mathbf{u}} \sim \nabla_h^2 \mathbf{u}^{n+1}$, this step reads:

$$\frac{\mathbf{u}^{n+1} - \widetilde{\mathbf{u}}}{\Delta t} = -\nabla_h P^{n+1} \text{ in } \Omega \tag{6a}$$

$$\nabla_h \cdot \mathbf{u}^{n+1} = 0 \text{ in } \Omega \tag{6b}$$

$$\mathbf{n} \cdot \mathbf{u}^{n+1} = \mathbf{n} \cdot \widetilde{\mathbf{u}} = \mathbf{n} \cdot \mathbf{u}_D \text{ on } \partial\Omega. \tag{6c}$$

[2]But it should be with the pressure at the previous time step or an extrapolation of the pressure as well.

Using Eq. (6*b*) and Eq. (6*c*), one can write:

$$\nabla_h \cdot \left(\Delta t \nabla_h P^{n+1}\right) = \nabla_h \cdot \widetilde{\mathbf{u}} \text{ in } \Omega \tag{7a}$$

$$\mathbf{n} \cdot \left(\Delta t \nabla_h P^{n+1}\right) = 0 \text{ on } \partial\Omega. \tag{7b}$$

Finally, the correction Eqs. (6) allows us to compute the new velocity $\mathbf{u}^{n+1}$ as follows:

$$\mathbf{u}^{n+1} = \widetilde{\mathbf{u}} - \Delta t \nabla_h P^{n+1} \text{ in } \Omega. \tag{8}$$

3. Immersed boundary method

The root concept of immersed boundary approaches is depicted in Fig. 2. In the system illustrated in this figure, the physical domain Ω_f corresponds to a fluid flow around a time-varying obstacle of boundary surface Σ. Its velocity is denoted by $\mathbf{u}_s$. As previously mentioned, the immersed boundary techniques consists in immersing this physical domain Ω_f in a fixed fictitious one Ω. Here, the computational domain Ω corresponds to an Eulerian grid with a fluid domain Ω_f and an embedded solid domain Ω_s. The physical boundary surface Σ is numerically approached by a discrete surface Σ_h which is described by a set of Lagrangian points. In the numerical experiments presented in the present paper, the interface tracking is performed by means of a Front-Tracking technique. The description of such method is out of work of the present study. Therefore, for details about Front-Tracking approach, we refer the interested reader to [34, 35], to cite among others.

[Figure 2 about here.]

The remain of this section is structured as follows. In the first part, we introduce the concept of the penalized direct forcing and, in the second part, the consistent fractional-step algorithm used with the PDF method. The interpolation scheme developed in order to reconstruct accurately the velocity field near the immersed boundary is presented in Section 4. Let us remark that, even if the algorithm is illustrated on fixed rigid bodies on uniform Cartesian grids, it also works on moving bodies and on non-uniform grids. Furthermore, it could be extended to computational grids that can be mapped into Cartesian ones or to deformable bodies as in [22].

3.1. Penalized direct forcing

In the frame of immersed boundary-like methods [5, 8, 9], the presence of embedded time-varying geometries Ω_s in the computational domain Ω is taken into account by a supplementary source term $\mathbf{F}$ added to the Navier-Stokes equations given by Eqs. (1). This

leads to the following equations:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \nabla P - \nu \nabla^2 \mathbf{u} = \mathbf{F} \quad \text{in } \Omega \tag{9a}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega. \tag{9b}$$

The term $\mathbf{F}$, here after referred to as penalized direct forcing term, is defined by [19]:

$$\mathbf{F} = \frac{\alpha \chi_s}{\eta} (\mathbf{u}_{imp} - \mathbf{u}) \text{ with } \alpha > 0,\ 0 < \eta \ll 1 \text{ and } \chi_s \rightarrow [0, 1] \tag{10}$$

where χ_s is the characteristic function or the volume ratio function of the obstacle [9], η a penalty coefficient, $0 < \alpha$ a real parameter and $\mathbf{u}_{imp}$ the imposed fluid velocity around/inside the obstacle Ω_s.

The imposed fluid velocity $\mathbf{u}_{imp}$ depends not only on the solid velocity $\mathbf{u}_s$ that may be uniform (*i.e.* constant in time and space) or analytic (*i.e.* $\mathbf{u}_s := \mathbf{u}_s(\mathbf{x}, t)$) but also on the fluid velocity $\mathbf{u}$ depending on the requested accuracy. This dependency leads to a change of the mass matrix when solving the equation Eq. (9*a*). But in practice, such a modification appears to be a complex and tedious task as suggested by the relationship Eq. (26) of Section 4. In this work, rather than tackle directly this full difficulty appearing in equation Eq. (9*a*), our objective is to calculate the velocity $\mathbf{u}^{n+1}$ using an explicit approximation of $\mathbf{u}_{imp}$ in the forcing term. This is achieved by solving first the following free-obstacle explicit N.-S.-like equations[3]:

$$\frac{\tilde{\mathbf{u}}^\star - \mathbf{u}^n}{\Delta t^\star} + \nabla_h \cdot (\mathbf{u}^n \otimes \mathbf{u}^n) - \nu \nabla_h^2 \mathbf{u}^n = 0 \quad \text{in } \Omega. \tag{11}$$

Then, the resulting provisional velocity $\tilde{\mathbf{u}}^\star$ is used to calculate an approximate imposed fluid velocity $\mathbf{u}_{imp}^{n+1}$ that depends on $\mathbf{u}_s^{n+1}$ and $\tilde{\mathbf{u}}^\star$. A similar strategy can be found in [18].

Remark 1. Eq. (10) *can be viewed as an implicit limit version* ($\Delta t = \eta/\alpha \ll 1$) *of the direct forcing expression* $\mathbf{F}_{DF} = \chi_s(\frac{\mathbf{u}_{imp} - \mathbf{u}^{k-1}}{\Delta t} - f(\mathbf{u}^{k-1}))$ *[8] applied to the Eq.* (4*a*)*:*

$$\mathbf{F}_{DF} = \frac{\chi_s}{\Delta t}\{\mathbf{u}_{imp} - (\mathbf{u}^{k-1} + \Delta t f(\mathbf{u}^{k-1}))\}. \tag{12}$$

[3]This equation is similar to the velocity prediction equation involved in the first step of a non-incremental projection method.

Remark 2. As far as the penalty coefficient η is concerned, Angot & al. [20] have shown that the solution of the PDE system given by Eqs. (9) *converges toward the Navier-Stokes body-fitted solution in $L^2(\Omega)$ norm with $\mathcal{O}\left(\eta^{3/4}\right)$ in the solid and $\mathcal{O}\left(\eta^{1/4}\right)$ in the fluid (but numerical tests show $\mathcal{O}(\eta)$ whatever the subdomain is). In this work, η is set to 10^{-12}.*

The penalized forcing term $\mathbf{F}$ is only applied on Cartesian nodes near or inside the immersed boundary (*i.e.* $\chi_s > 0$; at the discrete level χ_s is taken as an extension of the cell volume ratio function of the obstacle strictly positive for the penalized faces). On these nodes, thanks to the penalty coefficient η^{-1}, the momentum equation Eq. (9a) boils down to the following relationship:

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}_{imp}(\mathbf{x}), \text{ for } \mathbf{x} \in \Omega \text{ such that } \chi_s(\mathbf{x}) > 0. \tag{13}$$

Note also that the classical incompressible Navier-Stokes equations are recovered in the (computed) fluid domain Ω_f when χ_s is set to zero in Eq. (10).

At this point of the analysis, it remains to define the imposed fluid velocity $\mathbf{u}_{imp}$ at Cartesian nodes nearest Σ_h. Regarding those that are located into the solid domain Ω_s (depicted by $\diamond$ in Fig. 3), the imposed velocity $\mathbf{u}_{imp}$ is exactly set to be the obstacle velocity $\mathbf{u}_s$. This can be express as:

$$\mathbf{u}_{imp}(\mathbf{x}) = \mathbf{u}_s(\mathbf{x}), \text{ for } \mathbf{x} \in \Omega \text{ such that } \chi_s(\mathbf{x}) > 0 \text{ and } d(\mathbf{x}) < 0 \tag{14}$$

where $d(\mathbf{x})$ denotes a signed distance to the immersed boundary Σ_h[4].

[Figure 3 about here.]

As far as the other nodes are concerned, namely the fluid nodes belonging to the interfacial region with a positive signed distance (*i.e.* $\chi_s(\mathbf{x}) > 0$ and $d(\mathbf{x}) > 0$), a numerical procedure is required to reconstruct accurately the velocity field. This is the object of Section 4.

[4] $d(\mathbf{x}) < 0$ for the solid domain and $d(\mathbf{x}) > 0$ for the fluid region.

3.2. A consistent fractional-step method

In the literature, most of the fractional-step schemes, employed in the frame of immersed boundary methods, do not take account of the forcing term in the correction step. As emphasized by Ikeno and Kajishima [25], such an approach leads to inconsistent schemes in the sense that the immersed boundary conditions are well satisfied by the predicted velocities in the prediction stage but not by the new velocities at the end of the projection stage. See also [26] for a projection approach of the immersed boundary methods.

In this section, we propose a modification of the fractional-step method presented in Section 2 in order to obtain a consistent scheme in the sense defined in [25].

The new feature of our algorithm [19] is that the forcing term is distributed both in the prediction and the correction stages of the projection. Under these circumstances, the first step of our scheme consists in solving a predicted velocity $\widetilde{\mathbf{u}}$ without the pressure gradient term as follows (setting $\alpha = 1/\Delta t$):

$$\frac{\widetilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \nabla_h \cdot (\mathbf{u}^n \otimes \mathbf{u}^n) - \nu \nabla_h^2 \widetilde{\mathbf{u}} = \frac{\chi_s}{\eta \Delta t}(\mathbf{u}_{imp}^{n+1} - \widetilde{\mathbf{u}}) \quad \text{in } \Omega. \tag{15}$$

Here, as in the standard fractional-step algorithm, $\widetilde{\mathbf{u}}$ respects the following boundary conditions on the obstacles:

$$\widetilde{\mathbf{u}} = \mathbf{u}_D \text{ on } \Sigma_h. \tag{16}$$

The second step of our modified algorithm corresponds to the correction stage to recover the pressure P^{n+1} and the solenoidal velocity $\mathbf{u}^{n+1}$, but including the remaining part of the forcing term:

$$\frac{\mathbf{u}^{n+1} - \widetilde{\mathbf{u}}}{\Delta t} = -\nabla_h P^{n+1} + \frac{\chi_s}{\eta \Delta t}(\widetilde{\mathbf{u}} - \mathbf{u}^{n+1}) \text{ in } \Omega \tag{17a}$$

$$\nabla_h \cdot \mathbf{u}^{n+1} = \chi_s \nabla_h \cdot \mathbf{u}_{imp}^{n+1} \text{ in } \Omega. \tag{17b}$$

The equation Eq. (17a) can be simplified as:

$$\check{\rho} \frac{\mathbf{u}^{n+1} - \widetilde{\mathbf{u}}}{\Delta t} = -\nabla_h P^{n+1} \text{ in } \Omega. \tag{18}$$

Here, it is interesting to remark that the equation Eq. (18) is similar to the classical correction equation Eq. (6a) with $\check{\rho} = (1 + \chi_s/\eta)$. This justifies our choice $\alpha = 1/\Delta t$. Now, using

Eq. (17b), we obtain[5]:

$$\nabla_h \cdot \left(\frac{\Delta t}{\check{\rho}} \nabla_h P^{n+1} \right) \;=\; \nabla_h \cdot \widetilde{\mathbf{u}} - \chi_s \nabla_h \cdot \mathbf{u}_{imp}^{n+1} \text{ in } \Omega. \tag{19}$$

In the liquid Ω_f (*i.e.* $\chi_s = 0$), we have $\check{\rho} = 1$. But in the solid (*i.e.* $\chi_s > 0$), it becomes $\check{\rho}(\eta) \sim \mathcal{O}\left(\eta^{-1}\right)$. Therefore, $1/\check{\rho}(\eta)$ can be viewed as an effective coefficient which satisfies:

$$\frac{1}{\check{\rho}(\eta)} \sim \mathcal{O}\left(\eta\right) \ll 1. \tag{20}$$

By doing this, we enforce the local homogeneous Neumann boundary conditions of Eq. (19) on obstacles in a natural way. The use of small values for the diffusion coefficients to simulate immersed homogeneous Neumann BCs can also be found in [36].

In Eq. (19), in order to avoid numerical difficulties caused by these low diffusion coefficients inside the immersed boundary (when all the velocity degrees of freedom are penalized), we may add a pressure L^2-penalty term leading to the following modified pressure correction equation:

$$\nabla_h \cdot \left(\frac{\Delta t}{\check{\rho}} \nabla_h P^{n+1} \right) + \frac{\chi_s}{\eta} \left(P^0 - P^{n+1} \right) \;=\; \nabla_h \cdot \widetilde{\mathbf{u}} - \chi_s \nabla_h \cdot \mathbf{u}_{imp}^{n+1} \text{ in } \Omega. \tag{21}$$

where P^0 is a prescribed pressure correction. This only affects the cells totally included in the obstacle for which the velocities and the pressures are imposed but not the consistency of the scheme in the fluid sub-domain. In practice, P^0 is set to zero (*cf.* Sec. 5).

Remark 3. To clarify this point, let us consider as example an 1D case with the cell i in fluid and the cell $i+1$ (cut by Σ) in the solid. The face $i+1/2$ coincides with Σ_h and, following Remark 2, we have: $\widetilde{\mathbf{u}}_{\Sigma_h} = \mathbf{u}^{n+1}_{imp\,\Sigma_h} + \mathcal{O}(\eta^\zeta)$ with $\zeta = 3/4$ in the approximated-solid region. On one hand, Eq. (19) and Eq. (21) are similar for the fluid cell i and give us: $\widetilde{\mathbf{u}}_{i-1/2} - \frac{\Delta t}{\rho}(\nabla_h P^{n+1})_{i-1/2} = \mathbf{u}^{n+1}_{imp\,\Sigma_h} + \mathcal{O}(\eta^\zeta) - \frac{\Delta t}{\rho}\mathcal{O}(\eta)(\nabla_h P^{n+1})_{\Sigma_h}$. This does not break the imposed boundary condition on Σ_h if $(\nabla_h P^{n+1})_{\Sigma_h}$ is $\mathcal{O}(\eta^{\zeta-1})$ or higher order.

On other hand, Eq. (21) applied to the solid cell $i+1$ gives us: $\frac{\Delta t}{\rho}\mathcal{O}(\eta)(\nabla_h P^{n+1})_{i+3/2} -$

[5] $\nabla_h \cdot \mathbf{u}^{n+1} = 0$ in Ω_f and $\nabla_h \cdot \mathbf{u}^{n+1} = \nabla_h \cdot \mathbf{u}^{n+1}_{imp}$ in Ω_s. We recall that for translations or solid rotations, $\nabla \cdot \mathbf{u}^{n+1}_{imp}$=0.

$\frac{\Delta t}{\rho}\mathcal{O}(\eta)(\nabla_h P^{n+1})_{\Sigma_h}+\frac{1}{\eta}\left(P^0-P_{i+1}^{n+1}\right)=\mathcal{O}(\eta^\zeta)$ *leading to* $P_{i+1}^{n+1}=P^0+\mathcal{O}(\eta^{1+\zeta})$ *and* $(\nabla_h P^{n+1})_{\Sigma_h}$ *as well as* $(\nabla_h P^{n+1})_{i+3/2}=\mathcal{O}(\eta^{\zeta-1})$ *or higher orders.*

Finally, the correction equation (18) allows us to compute the new velocity $\mathbf{u}^{n+1}$ as follows:

$$\mathbf{u}^{n+1}=\widetilde{\mathbf{u}}-\frac{\Delta t}{\check{\rho}}\nabla_h P^{n+1} \text{ in } \Omega. \tag{22}$$

As a whole, we get a consistent fractional-step scheme in the sense that the immersed Dirichlet boundary condition is well satisfied on the obstacles by the new velocity $\mathbf{u}^{n+1}$, see Eqs. (16) and (22):

$$\mathbf{n}\cdot\mathbf{u}^{n+1}=\mathbf{n}\cdot\widetilde{\mathbf{u}}=\mathbf{n}\cdot\mathbf{u}_{imp}^{n+1} \text{ on } \Sigma_h \tag{23}$$

as well as the immersed Neumann boundary condition by the pressure, see Eq. (20):

$$\frac{\Delta t}{\check{\rho}}\nabla_h P^{n+1}\cdot\mathbf{n}=\mathcal{O}\left(\eta\right) \text{ on } \Sigma_h. \tag{24}$$

4. Reconstruction of the velocity field : interpolation schemes

The immersed boundaries are described by a set of Lagrangian points which do not generally coincide with the nodes of the Eulerian mesh. Under these circumstances, the velocity field must be reconstructed near the immersed boundary in order to take account of the immersed boundary conditions. This issue, which constitutes one of the major difficulty of the immersed boundary methods, has received a particular attention in the literature.

[Figure 4 about here.]

The simplest approach consists in prescribing directly the velocity of the obstacle $\mathbf{u}_s$ on the Cartesian nodes closest to the immersed boundary Σ_h without any interpolation scheme [9]. This numerical procedure, referred as the base model, is first-order accurate in space and reads:

$$\mathbf{u}_{imp}(\mathbf{x}):=\mathbf{u}_s(\mathbf{x})+\mathcal{O}\left(h\right), \text{ for } \mathbf{x}\in\Omega \text{ such that } \chi_s(\mathbf{x})>0 \text{ and } d(\mathbf{x})>0 \tag{25}$$

where h is the size of a Cartesian cell. As illustrated in Fig. 4, for instance with a staggered grid, the base model leads to a step-wise description of the immersed interface Σ_h and thus, to a larger obstacle.

Improved numerical algorithms have been proposed in the literature to reconstruct the velocity field in the interfacial region. Among them, the most widely employed to improve the accuracy of the solution close to Σ_h are those based on an interpolation or extrapolation procedure (*e.g.* [9–13, 25]). Roughly speaking, the interpolation or extrapolation procedures involve solid and fluid velocity contributions to calculate the velocities at the forcing nodes. The solid contribution corresponds to a node located on the immersed boundary Σ_h and is required when the solid velocity is analytic (as $\mathbf{u}_s$ depends on the position considered on Σ_h). As mentioned in the introduction of this paper, a classical approach consists in interpolating or extrapolating along a specific direction. For instance, in [10], the solid contribution corresponds to the orthogonal projection of the forcing node onto Σ_h. And the fluid contribution is defined as the intersection of the nearest fluid cell with the line passing through the forcing node and the solid contribution. Another possibility, proposed by Liao *& al.* [12], consists in defining the fluid contribution by the nearest fluid node along a grid line direction or a diagonal direction. Similarly, the solid contribution is defined by the intersection of Σ_h with the line passing through the forcing node and the fluid contribution. In the aforementioned examples, the determination of the solid contribution depends essentially on the preferred direction while the fluid contribution does not take account of the other fluid nodes in the immediate vicinity of the forcing node.

In this paper, we develop a linear interpolation scheme, second-order accurate in space, in which the fluid contribution is built following similar ideas used in [25] through an averaged reconstruction of the velocity gradient near Σ_h. The solid contribution is determined by means of a minimization problem. The scheme consists in an estimation of the imposed fluid velocity $\mathbf{u}_{imp}$ at the forcing or penalized node $\mathbf{x}$ according to:

$$\begin{aligned} \mathbf{u}_{imp}(\mathbf{x}) \;&:=\; \mathbf{u}_s(\Pi_\Sigma(\mathbf{x})) + \frac{d(\mathbf{x})}{N}\sum_{p=1}^{N}\frac{\mathbf{u}(\mathbf{x}^p)-\mathbf{u}_s(\Pi_\Sigma(\mathbf{x}^p))}{d(\mathbf{x}^p)} + \mathcal{O}\left(h^2\right) \\ &\qquad \text{for } \mathbf{x}\in\Omega \text{ such that } \chi_s(\mathbf{x})>0 \text{ and } d(\mathbf{x})>0 \end{aligned} \tag{26}$$

where $\mathbf{x}^p = \mathbf{x} \pm h$ is the p-th fluid node in the immediate vicinity of the penalized node $\mathbf{x}$. Fig. 5 shows a schematic representation of the penalized nodes $\mathbf{x}$ (red crosses $\times$) and their associated fluid nodes $\mathbf{x}^p$ (blue circles $\circ$) involved in our interpolation procedure. The relationship of Eq. (26) is similar to the one proposed in [25]. It can be viewed as an averaged linear interpolation over all the fluid nodes $\circ$ located in the immediate vicinity of the penalized node $\mathbf{x}$. Therefore, contrary to the classical approach found in [10] and [12], the local influence of the fluid flow at $\mathbf{x}$ is fully taken into account.

[Figure 5 about here.]

In Eq. (26), $\Pi_\Sigma(\mathbf{x})$ denotes the projection of $\mathbf{x}$ into the immediate neighborhood $\mathcal{V}^{\Sigma_h}$ of the immersed boundary Σ_h and corresponds also to the solid contribution of our interpolation scheme. It is defined throughout an algorithm based on the following minimization problem.

$$\text{Find } \mathbf{z}\,(=\Pi_\Sigma(\mathbf{x})) \in \mathcal{V}^{\Sigma_h} \text{ such that } J(\mathbf{z}) \;=\; \inf_{\mathbf{y}\in\mathcal{V}^{\Sigma_h}} J(\mathbf{y}) \tag{27}$$

where $J(\cdot)$ is defined by:

$$\forall\, \mathbf{y} \in \mathcal{V}^{\Sigma_h},\; J(\mathbf{y}) \;=\; \|\mathbf{y}-\mathbf{x}\|^2. \tag{28}$$

To tackle this minimization problem, regardless of the Cartesian node involved $\mathbf{x}$ (generic name for forcing nodes or fluid nodes), the first step consists in the partial reconstruction of the immersed boundary in the immediate vicinity of $\mathbf{x}$. To do so, we collect all the Lagrangian facets that belong to the cells intersected by Σ_h and located in the immediate vicinity of $\mathbf{x}$. If we consider an interfacial node $\mathbf{x}$ (*i.e.* $\chi(\mathbf{x}) > 0$ and $d(\mathbf{x}) > 0$, *cf.* Fig. 6), we begin to detect the first level of neighboring cells intersected by Σ_h. Then, we identify the "neighbors of neighbors" themselves cut by Σ_h and, for each of them, we collect the associated Lagrangian facets. If we consider a fluid node $\mathbf{x}$ (*i.e.* $\chi(\mathbf{x}) = 0$), the collecting procedure differs slightly because, in such a case, the neighboring cells of $\mathbf{x}$ are always purely fluid cells (*cf.* Fig. 7). Therefore, the collecting procedure starts with the identification of the interfacial nodes located in the neighboring cells of $\mathbf{x}$. Then, for each interfacial nodes, we identify the neighboring cells cut by Σ_h and we collect the associated Lagrangian facets.

[Figure 6 about here.]

[Figure 7 about here.]

After collecting a set of Lagrangian facets, the next step is to determine the equation of the plan passing through each of them. For example, if we consider the i-th Lagrangian facet associated with $\mathbf{x}$, such an equation reads:

$$\sum_{j=1}^{d} c_{ij} y_j = f_i \tag{29}$$

where d is the spatial dimension of the problem, c_{ij} the coordinates of the unit normal vector at the centroid of the i-th facet and y_j the coordinates of one vertex belonging to the i-th facet. Now, by applying Eq. (29) on all the facets associated with the Cartesian node $\mathbf{x}$, the minimization problem given by Eq. (27) can be expressed as:

$$\text{Find } \mathbf{z}\,(=\Pi_{\Sigma}(\mathbf{x})) \in \mathcal{V}^{\Sigma_h} \text{ such that } J(\mathbf{z}) \;=\; \inf_{\mathbf{C}\cdot\mathbf{y}\approx\mathbf{f}} J(\mathbf{y}). \tag{30}$$

To solve this new minimization problem, we use an Uzawa algorithm which can be summarized as follows.

1. Initialization $(k=0)$: we assume $\boldsymbol{\lambda}^0 = 0$
2. k-th iteration : by assuming λ^k known, we are able to compute
   - $\mathbf{y}^{k+1}$ by solving $\mathbf{y}^{k+1} = \mathbf{x} - \dfrac{1}{2}\mathbf{C}\cdot\boldsymbol{\lambda}^k$
   - $\boldsymbol{\lambda}^{k+1}$ by solving $\boldsymbol{\lambda}^{k+1} = \max\left[\boldsymbol{\lambda}^k + \varrho\left(\mathbf{C}\cdot\mathbf{y}^{k+1} - \mathbf{f}\right), 0\right]$ with $\varrho = \left(\|\mathbf{C}\|_1\|\mathbf{C}\|_\infty\right)^{-1}$

where $\boldsymbol{\lambda}^k$ is the Lagrange multiplier vector at the k-th iteration associated with the constraint $\mathbf{C}\cdot\mathbf{y}^{k+1} = \mathbf{f}$.

Regarding the coding logistic, the computational cost of the Uzawa algorithm depends on the size of the coordinates matrix $\mathbf{C}$. Thus, it depends on the number of Lagrangian facets collected for a Cartesian node $\mathbf{x}$. The collecting procedure, based on a two-level detection of the Cartesian cells crossed by Σ_h, contributes to reduce the number of facets without

limiting the accuracy of the proposed immersed boundary method (*cf.* Sec. 5). But, there may be cases in which several Cartesian cells contain a significant number of Lagrangian facets, for instance when the mesh is generated by CAD. To overcome this difficulty, we have implemented a numerical procedure that selects the most representative facets and also detects those that are collinear in order to remove the duplicate. In this study, the most representative facet means the most large. It is worthy to mention that different criteria can be chosen. Our collecting approach also limits the approximation errors in the Uzawa algorithm. Indeed, as the projection of the Cartesian node $\mathbf{x}$ is performed onto a set of plans and not onto a particular facet, increasing the number of level (and thus the number of facets) would necessarily increase the number of equations that must be satisfied by the projected node. The proposed algorithm is very robust and $\mathbf{z}\,(=\Pi_{\Sigma}(\mathbf{x}))\in\mathcal{V}^{\Sigma_h}$ converges toward $\mathbf{z}\in\Sigma_h$ when the space step decreases. Its implementation is a bit less straightforward than in the case of an usual local interpolation algorithm but it allows a great robustness for the complex immersed boundary geometries that can be found in industrial processes.

5. Numerical experiments

This section is devoted to the numerical validation of the proposed Penalized Direct Forcing method. On the one hand, a Poiseuille flow in an inclined channel, a cylindrical Couette flow and a steady flow around static/rotating cylinders are considered to assess the validity and the ability of our method both for uniform and analytic Dirichlet's IBCs. All these academic numerical tests have been performed with both the base model (*cf.* Eq. (25)) and the linear interpolation scheme (*cf.* Eq. (26)). Moreover, a grid convergence study has been done in order to obtain the numerical rate of convergence of the method. To do so, we have calculated the relative $\mathrm{L}^2(\Omega_f)$ norm ε_2 and the absolute $\mathrm{L}^\infty(\Omega_f)$ norm ε_∞ of the error of the velocity component u in the fluid domain Ω_f as follows:

$$\varepsilon_2 = \sqrt{\frac{\sum_{i=0}^{N_f} \left(u_i - u_{i,ref}\right)^2}{\sum_{i=0}^{N_f} {u_{i,ref}}^2}} \quad \text{and} \quad \varepsilon_\infty = \max_{0 \leq i \leq N_f} \left(|u_i - u_{i,ref}|\right) \tag{31}$$

where the lowerscript i denotes the i-th face of the Cartesian grid, N_f the total number of faces in the fluid region Ω_f, u_i the **u**-velocity component at the center of the i-th face and $\mathbf{u}_{i,ref}$ a reference velocity at the same location. This latter term is defined either by the analytic solution in the case of the Poiseuille and Couette flows or by the third-order extrapolation of the velocity computed on the finest grid in the case of steady flows around static/rotating cylinders. On the other hand, interestingly for practical purposes, our method coupled with the linear interpolation scheme is also applied to 2-D calculations involving a unsteady laminar flow and to 3-D calculations of the flow induced by a stirrer illustrating the ability to deal with moving geometries.

All the calculations have been performed using the CFD code Trio_U, which is a modular (*i.e.* finite differences or finite volumes) software package for thermal-hydraulic computations, developed at "Commissariat à l'Énergie atomique et aux énergies alternatives" [37]. As a whole, the linear interpolation scheme is about 10%-20% more expensive in CPU time than the base model. Unless specifically mentioned, we solve the arising linear systems from

Eqs. (15) and (19) with the conjugate gradient (CG) method. Considering the DF method, it is enough to precondition the pressure's iterative solver by a SSOR. But for the PDF method, due to badder matrix condition numbers, we need more powerful preconditioners. We use the diagonal inverse one (low cost) and the algebraic multigrid one for respectively Eqs. (15) and (19). This can induce a CPU time over-cost of about 10% respectively to the DF method, but the algebraic multigrid preconditioner is also very useful for the DF method, saving many CG iterations in case of a large number of degrees of freedom.

5.1. Poiseuille flow in an inclined channel

Here, the problem under consideration is the well-known test case of a Poiseuille flow in an inclined channel ($\theta = \pi/4$). We refer to Fig. 8 in which we have shown the geometrical features of the computational domain $\Omega = \Omega_s \cup \Omega_f$ with the immersed boundary $\Sigma = \Omega_s \cap \Omega_f$.

[Figure 8 about here.]

Regarding the boundary conditions imposed on Γ_s and Γ_{out}, we prescribe a null pressure. While, we prescribed on Γ_{in} the following parabolic velocity:

$$\mathbf{u}_{in} = (u_{in}, v_{in})^{\mathrm{T}} = \left(U_\infty - Y^2, 0\right)^{\mathrm{T}} \text{ with } Y = \frac{y - x}{\sqrt{2}} \tag{32}$$

where $U_\infty = 0.605$ m.s^{-1} is the maximum velocity. Here-above, (X, Y) is the system of coordinates in the frame associated with the inclined channel. For this numerical test, a no-slip boundary condition is imposed on the immersed boundary Σ (*i.e.* $\mathbf{u}_s = \mathbf{0}$).

[Figure 9 about here.]

Results are the following. Fig. 9 shows the stationary velocity and pressure fields in the computational domain obtained by the proposed penalized direct forcing method with the linear interpolation model. As mentioned previously, we have performed a grid convergence study. Four computational grid sizes have been considered ranging over $25.10^{-3}\text{m} \leq h \leq 2.10^{-1}\text{m}$. Fig. 10 presents the evolution of the $\mathrm{L}^2(\Omega_f)$ norm ε_2 and the $\mathrm{L}^\infty(\Omega_f)$ norm ε_∞ of the error given by Eq. (31). As expected, the graphs of Fig. 10 show that the proposed linear interpolation model leads to a quadratic numerical rate of convergence while the numerical order of the method using the base model is about one.

[Figure 10 about here.]

Here, it is interesting to remark that, in the case of the linear interpolation model, the last three errors in $\mathrm{L}^2(\Omega_f)$ and $\mathrm{L}^\infty(\Omega_f)$ norms are at least one order of magnitude lower than all the ones obtained with the base model. These results are confirmed by the graphs presented in Fig. 11. Indeed, we remark that the velocity profile obtained in the case of the coarse grid with the linear interpolation model is very close to the one computed with the base model on the fine grid. In contrary, the result obtained with the base model on the coarsest grid is very bad. The fact is that the immersed interfaces do not coincide with the grid lines and the base model consists in directly imposing the solid velocity $\mathbf{u}_s$ on the edges of the cells cut by the immersed interfaces (*cf.* Eq. (25)). This results in an erroneous step-wise description of the walls with a lower flow section. This is no longer the case using the linear interpolation model.

In conclusion, for this test, the results clearly show that the proposed method coupled with the linear interpolation model allows us to obtain an accurate solution at a low computational cost.

[Figure 11 about here.]

5.2. Taylor-Couette problem

Here, we focus on a steady flow between two rotating concentric cylinders, also called Taylor-Couette flow. This problem has already been considered in the frame of immersed boundary-like methods (*e.g.* [11, 38]). It allows us to numerically estimate the order of accuracy of the proposed method on problems involving rotating geometries.

Fig. 12 presents the geometrical features of the computational domain $\Omega = \Omega_s \cup \Omega_f$ with the immersed boundaries Σ_1 and Σ_2 mimicking the inner-and outer cylinders, respectively. $\Omega = [0, L] \times [0, L]$ corresponds here to a square where L will be defined later. The inner cylinder rotates clockwise ($\omega_1 > 0$) while the outer cylinder rotates counterclockwise ($\omega_2 < 0$). In overall calculations, we assume that the Reynolds number, defined by $\mathrm{Re} = |\omega_1| r_1^2/\nu$, is set to 1. Such an assumption allows us to write:

$$\mathrm{Ta} < \mathrm{Ta}_c \tag{33}$$

where Ta = 3/2 is the Taylor number defined by $\mathrm{Ta} = 0.5\mathrm{Re}^2(r_1 + r_2)(r_2 - r_1)^3/r_1^4$ and $\mathrm{Ta}_c = 1.712$ is the theoretical critical one [39]. Eq. (33) implies that the following simulations are still strictly planar (2-D).

[Figure 12 about here.]

The numerical parameters of the computations are summarized in Tab. 1.

[Table 1 about here.]

Regarding the boundary conditions, we prescribed symmetry conditions on the boundaries of the computational domain Γ and the following analytic velocities on the immersed boundaries Σ_1 and Σ_2:

$$\forall\, i = 1, 2 \quad \mathbf{u}_s = \omega_i r_i \mathbf{e}_r \quad \text{on}\, \Sigma_i \tag{34}$$

where $\mathbf{e}_r$ is the radial unit vector. The flow is initially at rest and converges toward a steady state. Fig. 13 presents an example of uniform Cartesian grid used to perform the calculations (left) and the steady velocity field (right).

[Figure 13 about here.]

Here again, a grid convergence study has been done in order to estimate the numerical order of the proposed method on problems involving moving bodies. For this problem, five meshes have been considered over the range $25.10^{-3} \leq h/r_1 \leq 10^{-1}$ where h denotes the step size. In order to reduce the computational time of these simulations, we have defined the length L of the computational domain Ω as follows:

$$L = 2\,(r_1 + r_2 + Nh) \tag{35}$$

where N is the number of cells in the outer cylinder (here, $N = 3$). This strategy is justified because, in the solid, the imposed fluid velocity is equal to the solid velocity (*cf.* Eq. (14)). As previously done, the linear interpolation model is compared to the base model. Fig. 14 presents the $\mathrm{L}^2(\Omega_f)$ norm ε_2 and the $\mathrm{L}^\infty(\Omega_f)$ norm ε_∞ of the error defined by Eq. (31). As in the case of an uniform imposed velocity (*cf.* part 5.1), a quasi-linear numerical

rate of convergence is obtained with the base model and a quasi-quadratic rate with the linear interpolation scheme. These good results confirm the enhancement of the accuracy as expected with the linear interpolation model. However, it is more difficult to explain why, in this case, the numerical rate of convergence is slightly lower than two (*cf.* Fig. 14). Such a result might be a consequence of two difficulties associated with our numerical scheme. Firstly, as the imposed fluid velocity $\mathbf{u}_{imp}$ depends on a provisional velocity $\widetilde{\mathbf{u}}^{\star}$ (*cf.* Eq. (11)), a time splitting error is made during its calculation which may slightly affect the rate of convergence. Secondly, in the case of fluid domain Ω_f totally embedded in the computational domain Ω, the calculation of the new pressure P^{n+1} by solving a Poisson like problem (*cf.* Eq. (21)), leads to numerical difficulties (ill-posed problem) as there is no pressure Dirichlet boundary condition imposed on the fluid domain boundary. In order to encompass this difficulty, a partial solution may consist in prescribing the pressure on one cell in the fluid domain Ω_f. In doing so, we get a smooth pressure field but the velocity field is locally disturbed around the place where the pressure is prescribed.
Finally, as observed in the case of a Poiseuille flow, the graphs presented in Fig. 14 show that the calculations performed on coarse grids with the linear interpolation scheme give better results (in terms of the errors) than those done on finer grids with the base model.

[Figure 14 about here.]

5.3. Laminar flows around a cylinder

In this part, we are concerned with laminar flows around a circular cylinder. This problem has been the object of many experimental and numerical studies. In the latter case, problems involving steady and unsteady fluid flows past a static cylinder have received a particular attention. In contrast, to our knowledge, the case of a fluid flow around a rotating cylinder has been rarely investigated.

In this paper, we consider laminar fluid flows around static and rotating cylinders. The flow is characterized by the Reynolds number $\mathrm{Re} = \frac{UD}{\nu}$ where U is the oncoming velocity, D the cylinder diameter and ν the kinematic viscosity of the fluid. Our simulations are performed both in the steady laminar regime (*i.e.* $\mathrm{Re} \leq 47$ [39]) with $\mathrm{Re} = 20$ and the

unsteady laminar regime with Re = 100. Tab. 2 gathers the test case parameters. We also introduce the dimensionless number β in order to characterize the rotation of the solid. It is defined as the ratio of the rotational velocity ω over the oncoming velocity and reads:

$$\beta = \frac{D\omega}{2U} \tag{36}$$

[Table 2 about here.]

[Figure 15 about here.]

As illustrated in Fig. 15, the computational domain Ω corresponds here to a square of length L with an immersed cylinder centered at the point of coordinates $(0, 0)$. The boundaries of the computational domain $\partial\Omega$ must be located sufficiently far enough to reduce the impact of boundary conditions on vortex development behind the cylinder [40]. In this work, the ratio of the length L over the diameter of the cylinder D is set to 60. This allows us to obtain, with a reasonable computational cost, results in good agreement with those given in the literature. Symmetry conditions are prescribed on Γ_s while a null pressure is imposed on Γ_{out}. As far as the boundary condition on Γ_{in} is concerned, we consider an uniform normal velocity $u_{in} = U$ and a null tangential velocity. Regarding the IBCs, the following Dirichlet boundary condition is considered:

$$\mathbf{u}_s = \frac{\omega D}{2}\mathbf{e}_r \text{ on } \Sigma \tag{37}$$

where $\mathbf{e}_r$ is the radial unit vector. Note that in the case of non-rotating cylinder (*i.e.* $\omega = 0$), Eq. (37) reduces to a no-slip boundary condition.

5.3.1. Steady laminar case : Re = 20

The simulations presented and discussed herein have been performed with the linear interpolation model and the base one on uniform Cartesian grids. Qualitatively, these two models provide similar results.

Fig. 16(a) and Fig. 17(a) show respectively the streamlines and the vorticity contours obtained with a static cylinder. In this case, the flow pattern is characterized by a pair of two steady symmetric vortices attached to the surface of the cylinder. These results are in very good agreement with those proposed in the literature (*e.g.* [1, 14, 26, 41–43]). The rotation of the cylinder disturbs the flow pattern. Indeed, as illustrated in Fig. 16(b) with the streamlines and in Fig. 17(b) with the vorticity contours, the flow becomes asymmetric with especially, a complete disappearing of the two vortices located behind the cylinder in the static case. These results are in very good agreement with [2] and also with those published in [40]. At this stage, we refer the reader to the work of Stojković *& al.* [40] in order to understand the effect of the rotation on the flow around the cylinder. In our case, namely for Reynolds numbers lower than the critical one ($\mathrm{Re} \leq 47$), Stojković *& al.* [40] explain that the upper vortex disappears and the lower one goes away from the surface of the cylinder with a significant decrease in size. But, in the considered regime (*i.e.* $\mathrm{Re} = 20$) the steady vortex obtained in the static case is small (*cf.* Fig. 16(a)) and thus, due to the rotation of the cylinder, it completely disappears.

[Figure 16 about here.]

[Figure 17 about here.]

To conclude the flow description, a comparison of the results presented in Fig. 18(a) and Fig. 18(b) clearly shows that the rotation also impacts the pressure distribution around the cylinder. Indeed, the pressure force rotates in the opposite direction to the cylinder which is also in good agreement with [40].

[Figure 18 about here.]

The results provided by the linear interpolation model and the base one are also compared quantitatively in terms of drag coefficient $C_d = F_x/0.5U^2D$ (static and rotating cases), lift coefficient $C_l = F_y/0.5U^2D$ (rotating case), recirculation length L_w (static case) and

direction of the total force $\theta = \tan^{-1}(C_l/C_d)$ (rotating case). This comparison is done using three levels of refinement ranging from $D/h = 10$ to $D/h = 40$ with h the size of Cartesian cells. Here-above, F_x and F_y correspond respectively to the tangential and normal components of the total force $\mathbf{F}_t$ defined by:

$$\mathbf{F}_t = \oint (-p\mathbf{I} + \nu\nabla\mathbf{u}) \cdot \mathbf{n} \, dS \tag{38}$$

where $\mathbf{n}$ is the outward normal unit along the surface Γ.

The physical coefficients obtained with a static cylinder are summarized in Tab. 3 and are confronted with the set of data proposed in [1, 14, 26, 41–43]. Fig. 19 illustrates the time convergence of these hydrodynamic coefficients.

[Table 3 about here.]

[Figure 19 about here.]

It is interesting to note that all the couples of coefficients $(C_d, L_w/D)$ corresponding to the linear interpolation model are in good agreement with those presented in the literature whatever the number of cells D/h is, whereas the base model requires the finest resolution.

[Table 4 about here.]

Tab. 4 summarizes the values of C_d, C_l and θ obtained with the linear interpolation scheme and the base model in the rotating case. These coefficients are compared to those given in [2, 40, 44, 45]. The trend observed in Tab. 4 is similar to the one previously highlighted in the static case and, as expected, the linear interpolation model yields better results than the base model.

Now, a grid convergence study is conducted in order to estimate the numerical rate of convergence of the proposed method. For this purpose, without loss of generality, a smaller computational domain is chosen, namely $\Omega = [-10D, 10D] \times [-10D, 10D]$. Resizing the computational domain allows us to consider fine grids with a reasonable computational cost. At this stage, this approach is justified because our goal is not to calculate physical

coefficients but to study the behavior of our penalized direct forcing method. As no analytical solution exists, we compute the error norms ε_2 and ε_∞ (*cf.* Eq. (31)) assuming that the solution obtained on the finest mesh is the reference solution. Five levels of refinement are used ranging from $h/D = 10^{-1}$ to $h/D = 6.25 \times 10^{-3}$. For the sake of clarity, we begin by discussing the case of the static cylinder. Fig. 20 and Fig. 21 present, respectively, the evolution of the error norms ε_2 and ε_∞ measured over the whole fluid domain. As expected, Fig. 20 shows that the numerical rate of convergence in L^2 norm of the PDF method employed with the linear interpolation scheme is consistent with the second order of accuracy. The one calculated with the base model is slightly higher than one. However, as illustrated in Fig. 21, the convergence in L^∞ norm is much lower with a numerical rate of convergence close to one.

[Figure 20 about here.]

[Figure 21 about here.]

[Figure 22 about here.]

Fig. 22 presents the evolution of ε_∞ calculated over 90% of the fluid domain. In this case, it is interesting to notice that the numerical rate of convergence tends toward the second order of accuracy. This indicates that the maximum of the error is located in the immediate vicinity of the immersed interface. Such a behavior has already been pointed out in the frame of the Cartesian methods by Cheny and Botella [38] which explain it by the piecewise approximation of the pressure in the Cartesian cells near the immersed interface. Actually, it is difficult for us to explain why the proposed PDF method behaves like that. Such a study will be the object of future improvements. In Fig. 20 and Fig. 22, it is also interesting to remark that the values of ε_2 and ε_∞ supplied by the linear interpolation scheme on coarse grids are lower than those measured with the base model on the finest grid. This trend is similar to the one observed in Section 5.1 and Section 5.2.

Now, we focus on the results obtained with a rotating cylinder. The error norms ε_2 and ε_∞ measured over the whole fluid domain are depicted on Fig. 23 and Fig. 24, respectively. In

Fig. 23, we observe that the rate of convergence in L^2 norm of the proposed method used with the linear interpolation model is close to two while it is slightly superior to one with the base model. In Fig. 24 and Fig. 25, the graphs clearly show that a calculation of the L^∞ norm over 90% of the fluid domain yields a better rate of convergence. Again, this means that the maximum of the error is very close to the immersed interface. These results confirm the trend observed with a static cylinder. Here again, the values of ε_2 and ε_∞ calculated on coarse grids with the linear interpolation scheme are lower than those computed with the base model.

[Figure 23 about here.]

[Figure 24 about here.]

[Figure 25 about here.]

5.3.2. Unsteady laminar case : Re = 100

This part is devoted to simulations performed in the unsteady regime with Re = 100. Here, we remind the reader that we only use the linear interpolation model. The geometrical features are identical to those previously considered in the steady case. By cons, all the calculations have been done using a non-uniform grid with fifty Cartesian cells in the diameter of the cylinder.

Fig. 26, Fig. 27 and Fig. 28 present, respectively, an instantaneous of the streamlines, the vorticity contours and the pressure distribution. In these figures, we remark that the flow pattern is characterized by the well-known Von Kármán vortex street which corresponds to a periodic shedding of vortex behind the cylinder. Such a phenomenon, well captured by our immersed boundary technique, is in good agreement with the results provided in the literature (*e.g.* [2, 13–15, 38, 40, 41] in the static case and [2, 40, 46] in the rotating case).

[Figure 26 about here.]

[Figure 27 about here.]

[Figure 28 about here.]

Moreover, as previously done, our results are also compared in terms of hydrodynamic coefficients and confronted to those of the references formerly cited. We have computed the time averaging drag coefficient $\overline{C_d}$, the time averaging lift coefficient $\overline{C_l}$, the amplitude of their fluctuations $C_d^{'}$ and $C_l^{'}$ and the Strouhal number $\mathrm{St} = fD/U$ which is the dimensionless number used to characterize the shedding frequency f. In this work, f is estimated from the periodic variation of the lift coefficient C_l. As shown in Tab. 5 (static case) and Tab. 6 (rotating case), the values of the hydrodynamic coefficients obtained with the proposed penalized direct forcing method are in good agreement with those published in the literature. Fig. 29 illustrates the time evolution of these hydrodynamic coefficients.

[Table 5 about here.]

[Table 6 about here.]

[Figure 29 about here.]

5.4. 3-D flow induced by an ellipsoidal stirrer

In order to illustrate the ability of the PDF method with the linear interpolation model to deal with 3-D moving geometries, we describe here the flow induced by a stirrer in a cylindrical chemical reactor. The context is the vitrification process for the storage of radioactive wastes involving viscous flows (molten glass at high temperature incorporating the ultimate waste). In industrial (confidential) simulations both the vessel structure, the apparatus of measurement and the mechanical stirrer are modeled by IBCs. Here, we present a more academic test case for which the industrial stirrer is replaced by an ellipsoid in rotation around an inclined axis in the y-z plan in a cylindrical vessel. This computation was run using a pretty rough 15x15x10 Cartesian mesh. Fig. 30 shows the velocity and pressure distributions. The dynamic viscosity of the fluid is about several units. The angular velocity of the stirrer is about 60 rounds per minute, starting impulsively in a fluid at rest. Homogeneous Dirichlet BCs are prescribed on the immersed boundary describing the vessel and on the boundary of the computational domain, except on the upper surface. On this latter one, a symmetry condition is imposed. As we face to moving IBCs, an important issue concerns the treatment of the so-called "freshly fluid or solid cells". Without going into details (this will be the subject of a future paper), this is achieved by a local averaging process in the fresh fluid cells left by the moving solid. Under a CFL limitation, we set the *new* value of a fresh fluid cell using an arithmetic average of its neighbor cells that are in the fluid at the current and the previous time steps.

[Figure 30 about here.]

For real world applications, more complex geometries are imported from Computer-Aided Design files. For instance, these computations can be useful to improve the design of the

stirrer's geometry. Actually, the Trio_U software has also the possibility to describe the stirrer by an IBC using in the same time a body-fitted space discretization based on the vessel. But the use of IBCs to model this vessel allows us to drastically reduce the computational CPU time because fast solvers can be used (as multigrid solvers on Cartesian grids).

6. Conclusion

In this paper, we have developed a Penalized Direct Forcing (PDF) method for unsteady laminar flows around complex time-varying geometries. Such a method allows us to solve on a Cartesian grid the Navier-Stokes equations with a penalized forcing term that mimics the presence of solids through immersed Dirichlet boundary conditions. One of the purposes of this work was to develop a robust linear interpolation scheme in order to accurately reconstruct the velocity field near the immersed interfaces. The originality of our approach is that the interpolation process does not depend on a preferred direction, as often done in the literature, but is based on an averaged reconstruction of the velocity gradient around the interface and on a minimization problem that relies on a local reconstruction of the immersed interface. Regarding to the numerical scheme, the proposed method is based on a finite volume approximation with a staggered grid arrangement of the variables. Furthermore, in order to solve the governing equations, we have used a fractional step scheme that is modified in such a way that the Dirichlet immersed boundary condition for velocity is verified not only in the prediction equation but also in the correction one. Moreover, the associated homogeneous Neumann boundary conditions are also enforced on the immersed boundaries. It is worth to notice that Neumann immersed boundary conditions could be also considered.

Several numerical experiments have been carried out using this new method. All the results demonstrate the efficiency and the potentialities of applications of our PDF method. On the one hand, the validity and the accuracy in space of our Hybrid Cartesian/Immersed Boundary technique have been assessed throughout two academical problems involving uniform and analytic Dirichlet IBCs. From these two tests, it results that the numerical rate of convergence is (quasi-)quadratic in L^2- and L^∞-norms. On the other hand, the problem of a laminar flow around static and rotating cylinders has been studied both in steady and unsteady regimes. Whatever the case considered, our results are in good agreement with those published in the literature. Furthermore, a grid convergence study performed in the steady case confirms that the numerical rate of convergence of our method tends to the second order of accuracy in space.

Extension of our method to problems involving 3-D geometries is trivial and requires no coding efforts as illustrated in this paper. In fact, in the nuclear safety context, 3-D purely hydraulic numerical simulations are in progress to study vitrification processes for the storage of radioactive wastes. In addition, our method has been only assessed on 2-D problems involving static and rotating solids. In the future, it is planned to improve this assessment on 3-D problems with moving geometries. This work is already underway for the flow induced by a mixer during the vitrification process. In future works, it will be also interesting to investigate the impact of the approximation of pressure on the results and, more particularly, on the convergence of the method in the immediate vicinity of the immersed interface. Equipping our method with adaptive techniques of mesh refinement would be also useful for this task. In an other direction, we plan to work in the future on *immersed wall laws* for turbulent flow applications as it is often the case in the industrial context. This enhancement, that is an alternative to the mesh refinement techniques, was already mentioned by Iaccarino and Verzicco in [16] where LES computations were done using immersed boundaries and mesh enrichment techniques.

Acknowledgement

This work was granted access to the HPC resources of CINES under the allocation 2012-c2012026025 made by GENCI (Grand Equipement National de Calcul Intensif).

References

[1] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, Journal of Computational Physics 156 (1999) 209-240.

[2] M.-H. Chung, Cartesian cut cell approach for simulating incompressible flows with rigid bodies of arbitrary shape, Computers & Fluids 35 (2006) 607–623.

[3] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019-1044.

[4] I. Ramière, Ph. Angot, M. Belliard: A general fictitious domain method with immersed jumps and non-conforming structured mesh, Journal of Computational Physics 225:2 (2007) 1347-1387.

[5] C. Peskin, Numerical Analysis of blood flow in heart, Journal of Computational Physics 25 (1977) 220-252.

[6] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, Journal of Computational Physics 105 (1993) 354-366.

[7] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, Journal of Computational Physics 153 (1999) 509-534.

[8] J. Mohd-Yusof, Combined Immersed Boundaries/B-Splines Methods for Simulations of Flows in Complex Geometries, CTR Annual Research Briefs, NASA Ames/Stanford University, 1997.

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

[10] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, Journal of Computational Physics 207 (2005) 457-492.

[11] N. Peller, A. Le Duc, F. Tremblay, M. Manhart, High-order stable interpolations for immersed boundary methods, Int. J. Numer. Meth. Fluids 52 (2006) 1175-1193.

[12] C.-C. Liao, Y.-W. Chang, C.-A. Lin, J.M. McDonough, Simulating flows with moving rigid boundary using immersed-boundary method, Computers & Fluids 39 (2010) 152-167.

[13] P.H. Chiu, R.K. Lin, T.W.H. Sheu, A differentially interpolated direct forcing immersed boundary method for predicting incompressible Navier-Stokes equations in time-varying complex geometries, Journal of Computational Physics 229 (2010) 4476-4500.

[14] J.-I. Choi, R.C. Oberoi, J.R. Edwards, J.A. Rosati, An immersed boundary method for complex incompressible flows, Journal of Computational Physics 224 (2007) 757-784.

[15] C. Ji, A. Munjiza, J.J.R. Williams, A novel iterative direct-forcing immersed boundary method and its finite volume applications, Journal of Computational Physics 231 (2012) 1797–1821.

[16] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations, Appl. Mech.

Rev. 56, 3 (2003) 331-347.

[17] J.Y. Yang, E. Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, Journal of Computational Physics 215 (2006) 12-40.

[18] A. Pinelli, I. Z. Naqavi, U. Piomelli, J. Favier. Immersed-boundary methods for general finite-difference and finite-volume solvers, Journal of Computational Physics 229 (2010) 9073-9091.

[19] M. Belliard, C. Fournier, Penalized direct forcing and projection schemes for Navier–Stokes, C. R. Acad. Sci. Paris, Ser. I 348 (2010) 1133-1136.

[20] Ph. Angot, Ch.-H. Bruneau, P. Fabrie, A penalization method to take into account obstacles in incompressible viscous flows, Numerische Mathematik 81 (1999) 497-520.

[21] A. Sarthou, S. Vincent, J. P. Caltagirone, Consistant velocity-pressure coupling for second-order L^2-penalty and direct-forcing methods, hal-00592079 version 1 (2011).

[22] M. Bergmann, A. Iollo, Modeling and simulation of fish-like swimming, Journal of Computational Physics 230 (2011) 329-348.

[23] A. Chorin, Numerical Solution of the Navier-Stokes Equations, Mathematics of Computation 22 (1968) 745-762.

[24] R. Temam, Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires, Arch.Rational Mech. Anal. 32 (1969) 135-153.

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

[26] K. Taira, T. Colonius, The immersed boundary method: a projection approach, Journal of Computational Physics 225 (2007) 2118-2137.

[27] J. H. Williamson, Low-storage Runge-Kutta schemes, Journal of Computational Physics 35 (1980) 48-56.

[28] J. H. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, Springer, New York, USA (1996).

[29] J. Shen, On Error Estimates of some Higher Order Projection and Penalty-Projection Methods for Navier-Stokes Equations, Numerische Mathematik. 62 (1992) 49-73.

[30] M. Jobelin, B. Piar, Ph. Angot, J.-C. Latché, Une méthode de pénalité-projection pour les écoulements dilatables, European J. Comput. Mech. 17 (2008) 453-480.

[31] Th.Gallouët, L. Gastaldo, R. Herbin, J.-C. Latché, An unconditionally stable pressure correction scheme for the compressible barotropic Navier-Stokes equations, M2AN 42 (2008) 303-331.

[32] J.L. Guermond, P.D. Minev, A new class of massively parallel direction splitting for the incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 200 (2011) 2083–2093.

[33] J.L. Guermond, P.D. Minev, J. Shen, An overview of projection methods for incompressible flows, Comput. Methods Appl. Mech. Engrg. 195 (2006) 6011–6045.

[34] S.O. Unverdi, G. Tryggvason, A Front-Tracking method for viscous, incompressible, multi-fluid flows, Journal of Computational Physics 100 (1992) 25-37.

[35] G. Tryggvason,B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas,Y.-J. Jan, A Front-Tracking method for the computations of multiphase flow, Journal of Computational Physics 169 (2001) 708–759

[36] B. Kadoch,D. Kolomenskiy, Ph. Angot, K. schneider, A volume penalization method for incompressible flows and scalar advection-diffusion with moving obstacles, Journal of Computational Physics 231 (2012) 4365–4383

[37] C. Calvin, A development framework for parallel CFD applications: Trio_U project, SuperComputing in Nuclear Applications (SNA-2003), Paris, France (2003). http://www-trio-u.cea.fr

[38] Y. Cheny, O. Botella, The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties, Journal of Computational Physics 229 (2010) 1043–1076.

[39] E. Guyon, J.P. Hulin, L. Petit, Physical Hydrodynamics, Oxford University Press, Oxford (2001).

[40] D. Stojković, M. Breuer, F. Durst, Effect of high rotation rates on the laminar flow around a circular cylinder, Physics of Fluids 14, 9 (2002) 3160-3178.

[41] M. N. Linnick, H.F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, Journal of Computational Physics 204 (2005) 157-192.

[42] B. Fornberg, A numerical study of steady viscous flow past a circular cylinder, Journal of Fluids Mechanic 98, 4 (1980) 819-855.

[43] D. J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds number, Journal of Fluids Mechanic 6 (1959) 547-567.

[44] D. B. Ingham, T. Tang, A numerical investigation into steady flow past a rotating circular cylinder at low and intermediate Reynolds numbers, Journal of Computational Physics 87 (1990) 91-107.

[45] H. M. Badr, S. C. R. Dennis, P. J. S. Young, Steady and unsteady flow past a rotating cylinder at low Reynolds numbers, Computers & Fluids 17, 4 (1989) 579-609.

[46] S. Kang, H. Choi, S. Lee, Laminar flow past a rotating circular cylinder, Phys. Fluids 11, 11 (1999) 3312-3321.

List of Figures

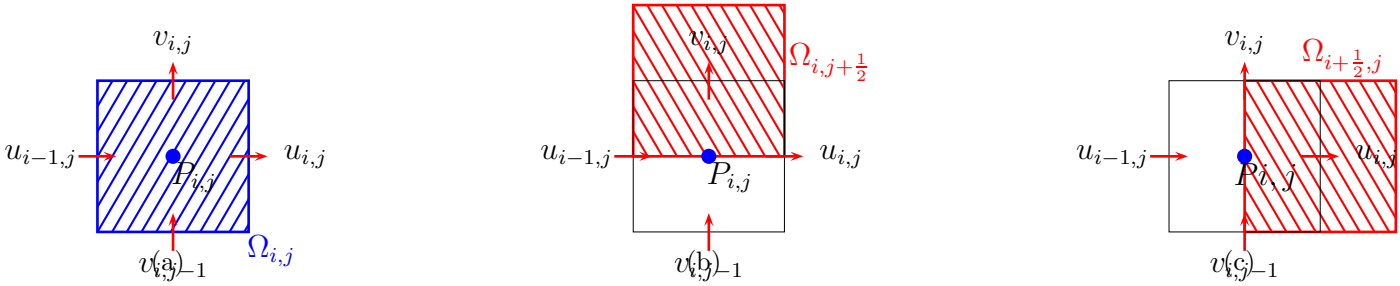


Figure 1: Staggered arrangement of the unknowns on a 2-D Cartesian cell (i, j) with the CVs of pressure (blue □) and velocity components (red □).

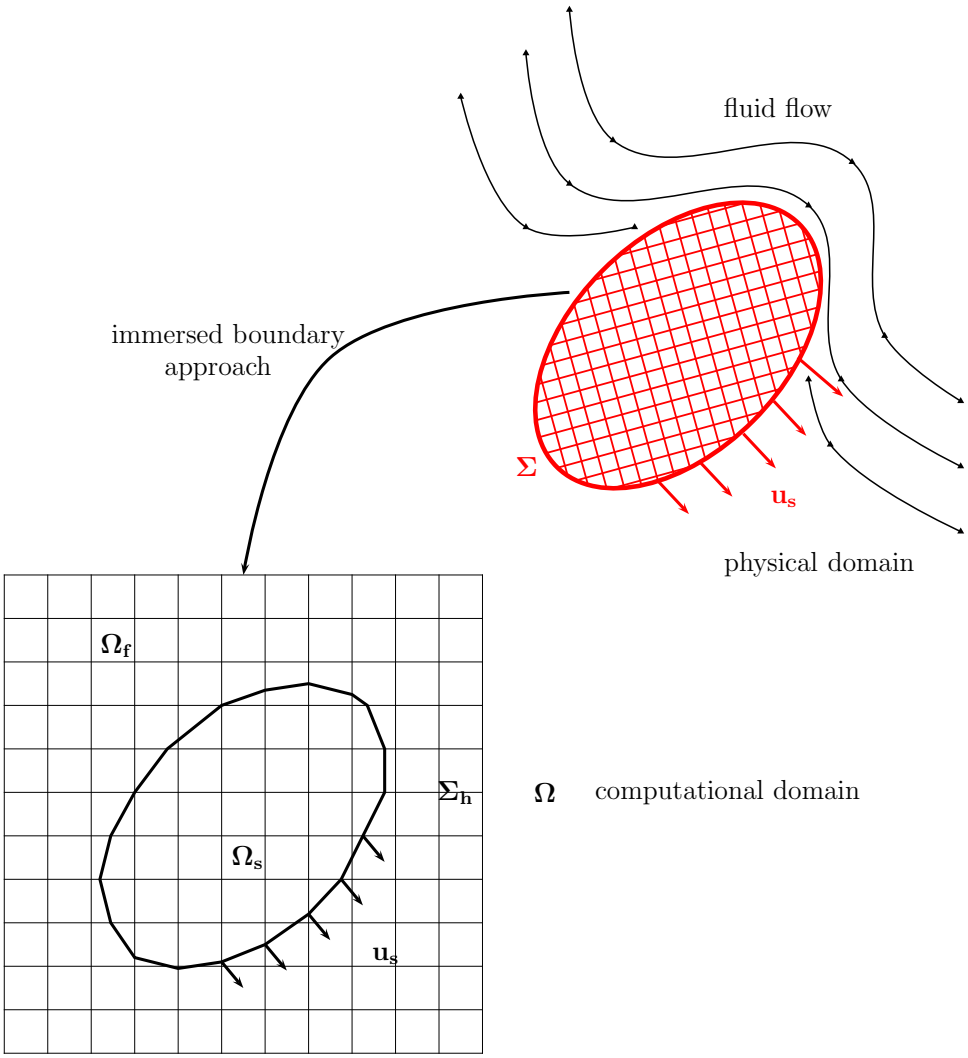


Figure 2: Schematic representation of a fictitious computational domain Ω on a Cartesian grid with the solid domain Ω_s, the fluid domain Ω_f and the immersed boundary Σ. Here, $\mathbf{u}_s$ denotes the velocity of the obstacle Ω_s.

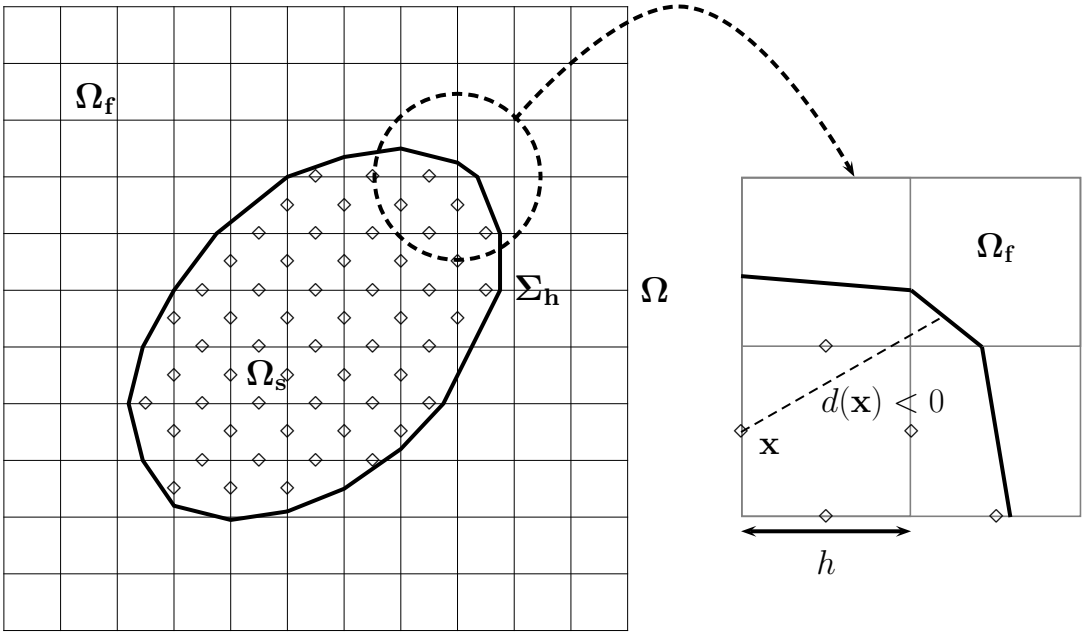


Figure 3: Schematic representation of solid nodes $\mathbf{x}$ where $\mathbf{u}_{imp}(\mathbf{x}) = \mathbf{u}_s(\mathbf{x})$.

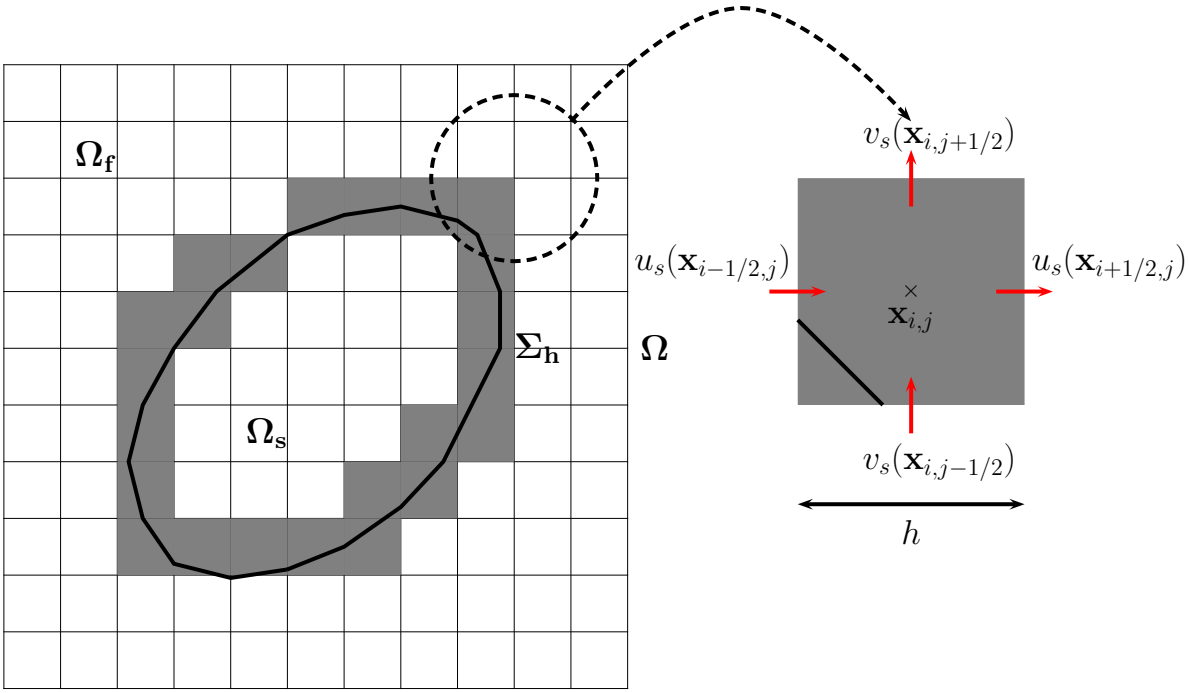


Figure 4: Schematic representation of the base model on a staggered grid

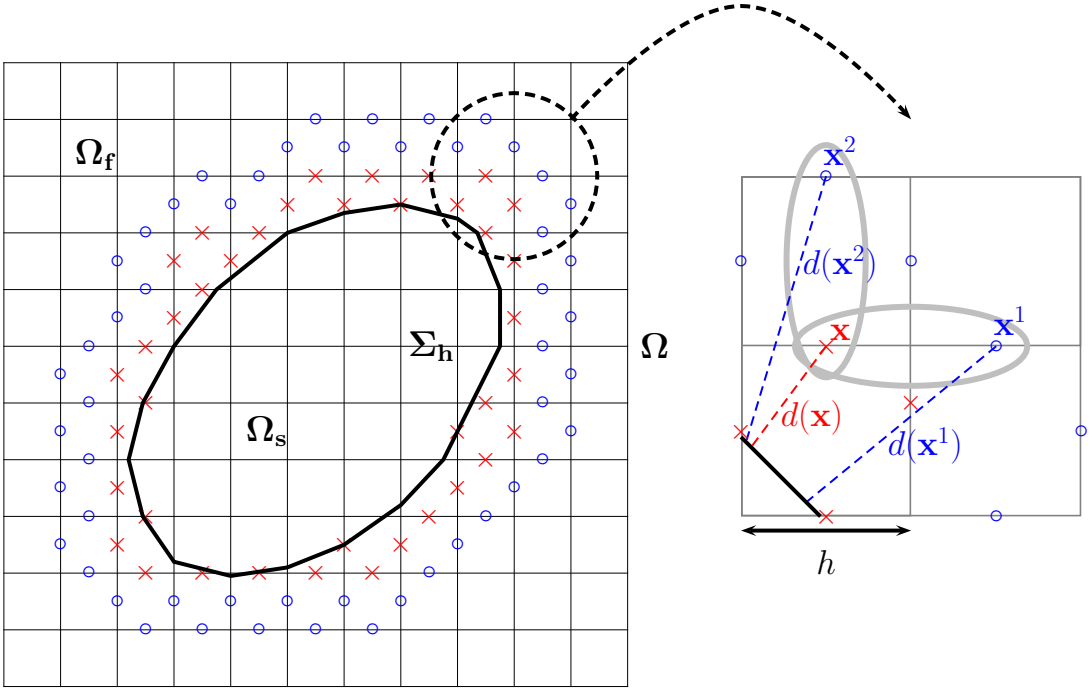


Figure 5: Schematic representation of the linear interpolation model on a staggered grid

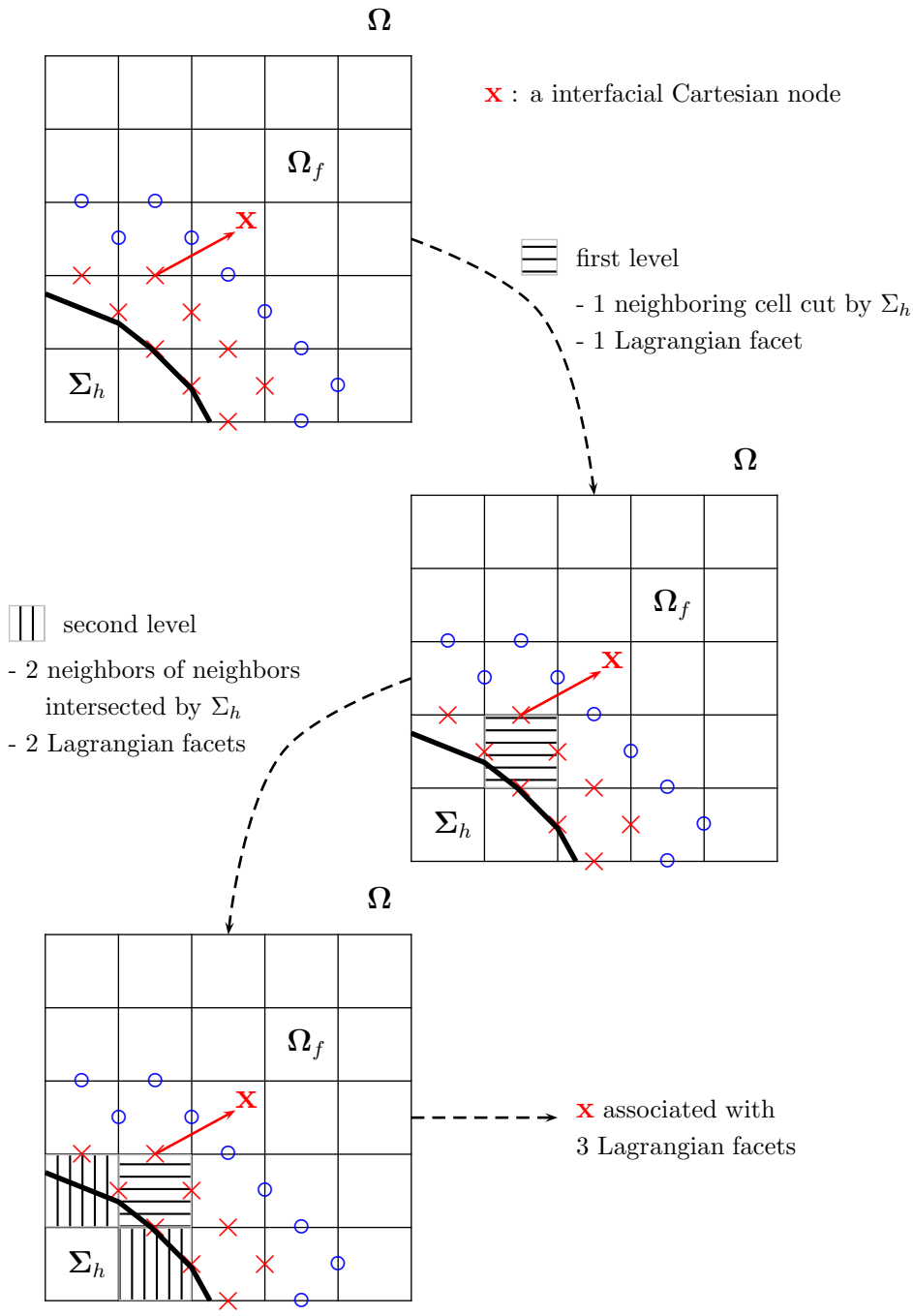


Figure 6: Schematic representation of the collecting procedure of Lagrangian facets in the case of a Cartesian node $\mathbf{x}$ located in the interfacial region. In the example, three Lagrangian facets are collected. One of them is obtained during the first level where only one neighboring cells is intersected by Σ_h. The last two are collected in the two "neighbors of neighbors" identified during the second level.

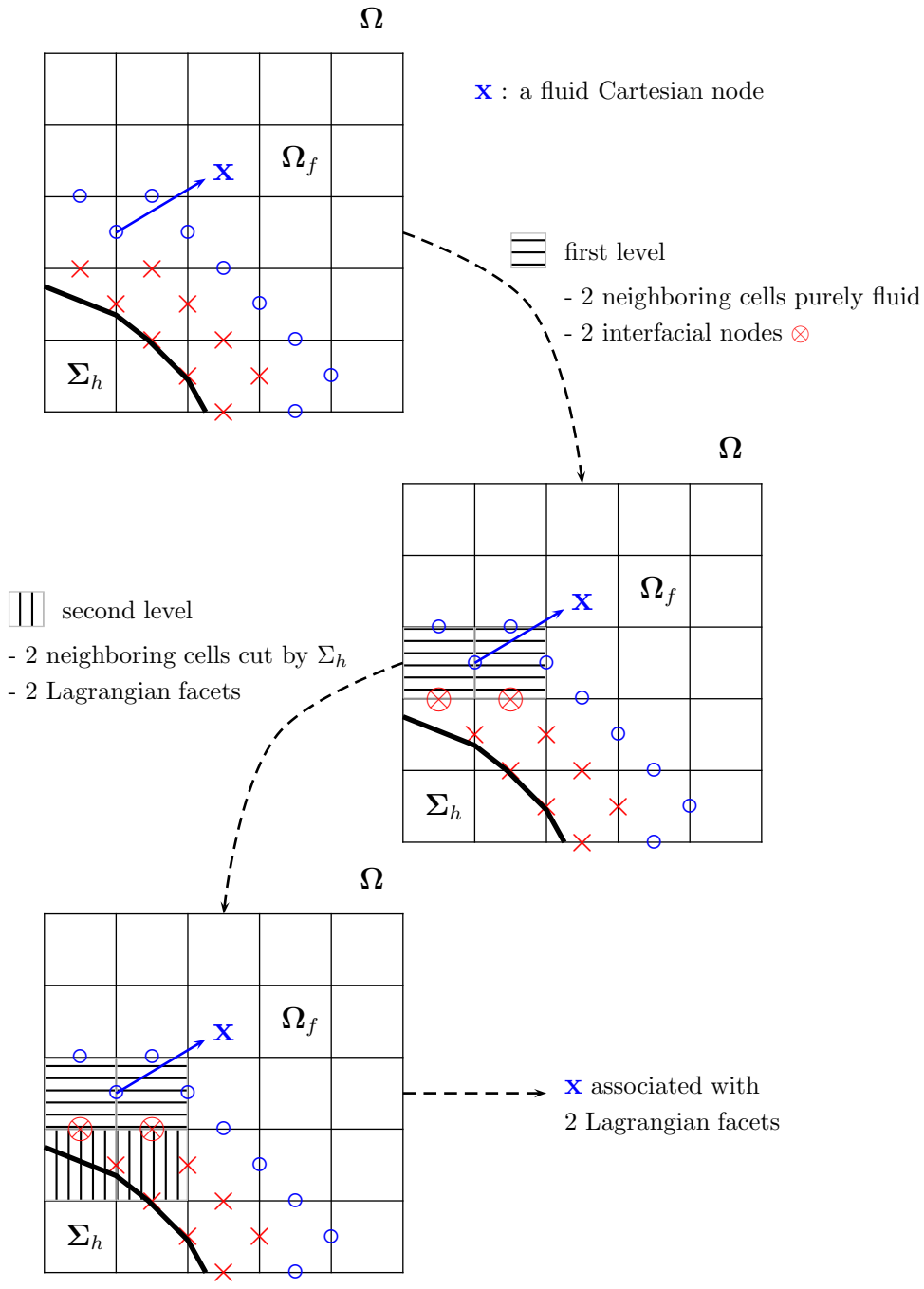


Figure 7: Schematic representation of the collecting procedure of Lagrangian facets in the case of a fluid Cartesian node $\mathbf{x}$. In the example, two interfacial points (red $\otimes$) are detected in the first level of the collecting procedure and the second level results in two cells with two associated Lagrangian facets.

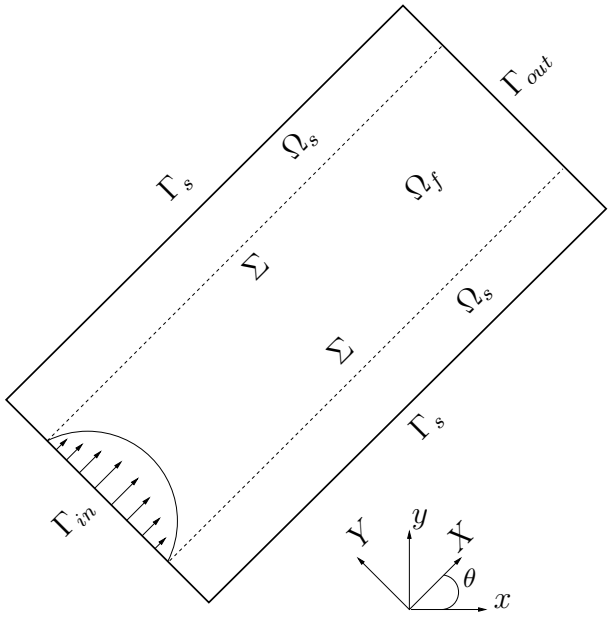


Figure 8: Poiseuille flow in an inclined channel. Computational domain Ω with the fluid domain Ω_f, the solid domain Ω_s and the immersed boundaries Σ.

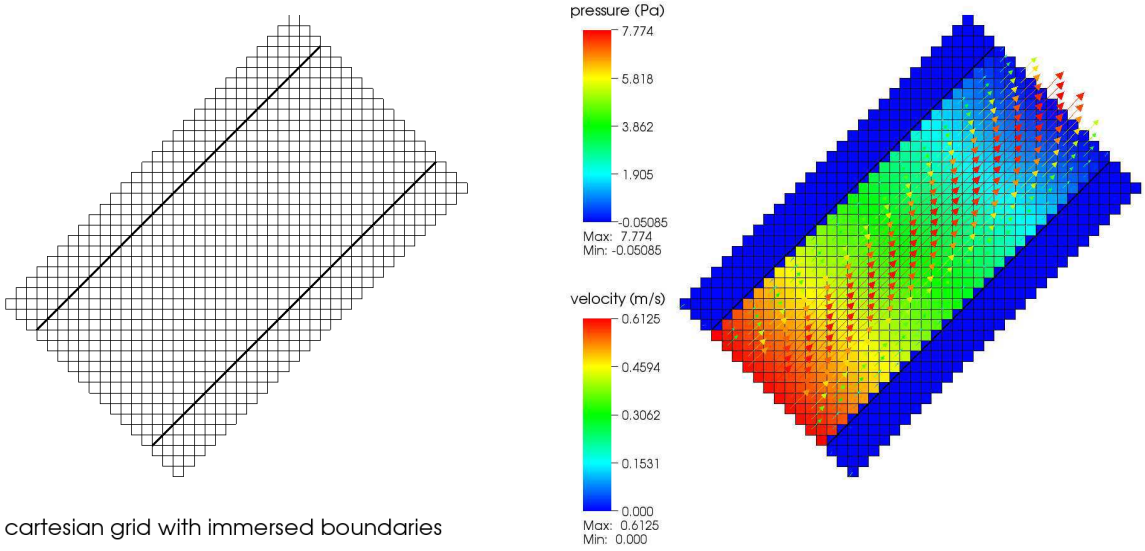


Figure 9: Poiseuille flow in an inclined channel. Example of stationary velocity and pressure fields (right) plotted on Cartesian grid with immersed boundaries (left).

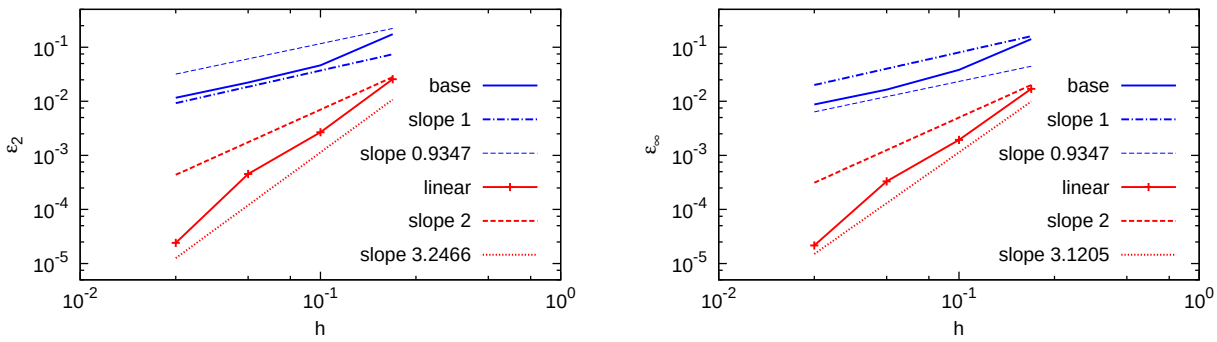


Figure 10: Poiseuille flow in an inclined channel. $\mathrm{L}^2(\Omega_f)$ norm (left) and $\mathrm{L}^\infty(\Omega_f)$ norm (right) of the error *vs.* the computational grid size h. Comparison of the linear interpolation model with the base model.

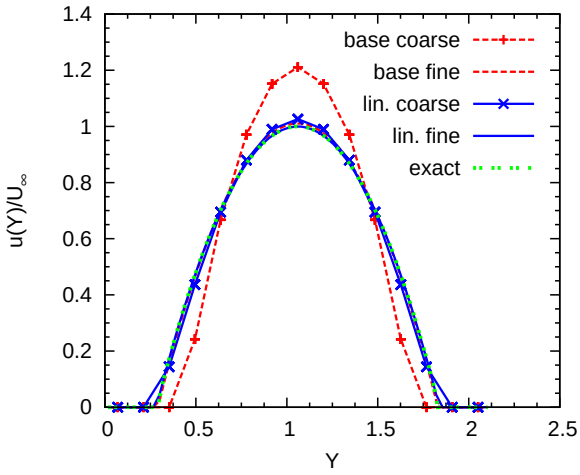


Figure 11: Poiseuille flow in an inclined channel. Plot on $X = 2.9$ of the ratio $u(Y)$ over the maximum velocity U_∞. Comparison of the linear interpolation model with the base model in the coarse and fine grid cases.

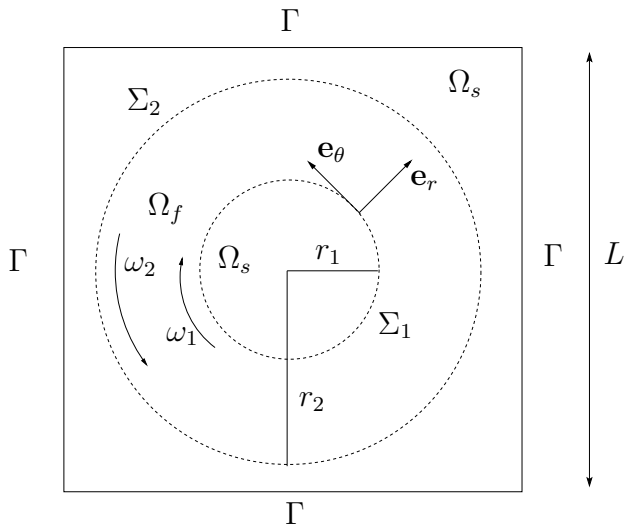


Figure 12: Taylor-Couette flow. Computational domain with immersed boundaries

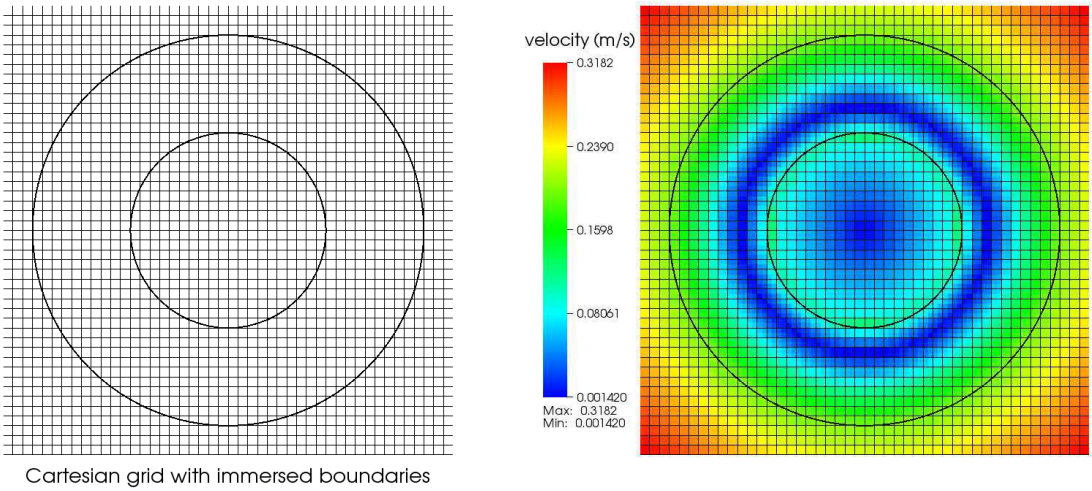


Figure 13: Taylor-Couette problem. Example of the stationary velocity field (right) plotted on a Cartesian grid with immersed boundaries (left).

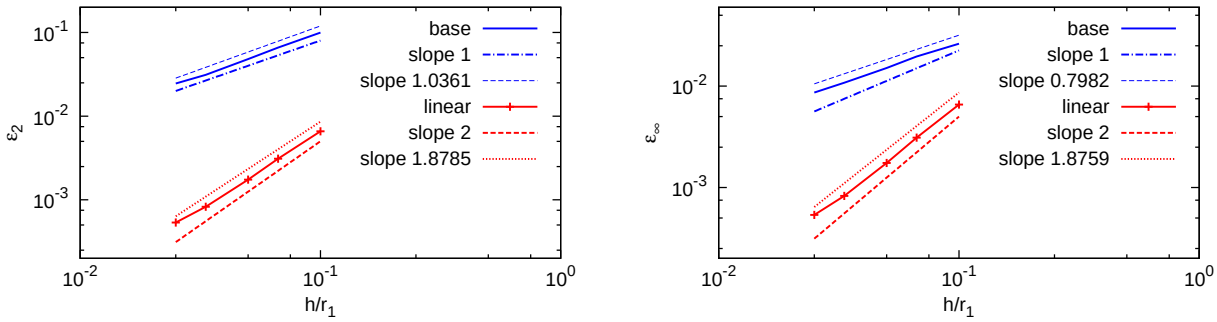


Figure 14: Taylor-Couette problem. $\mathrm{L}^2(\Omega_f)$ norm (left) and $\mathrm{L}^\infty(\Omega_f)$ norm (right) of the error *vs.* the ratio of the computational grid size h over the smallest radius. Comparison of the linear interpolation model with the base model.

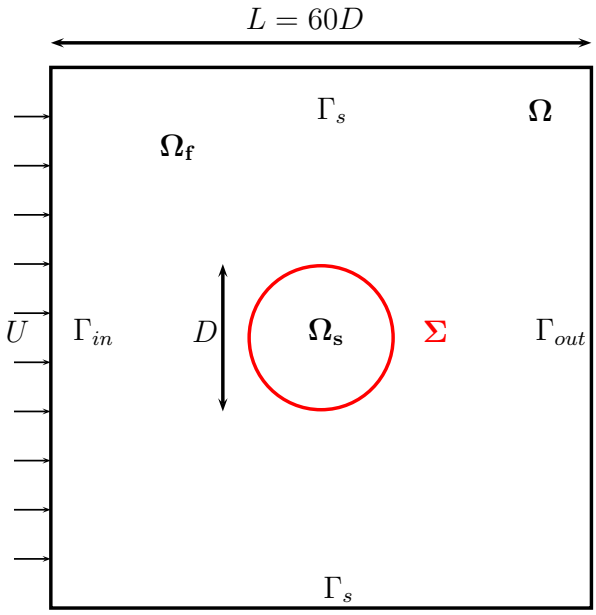
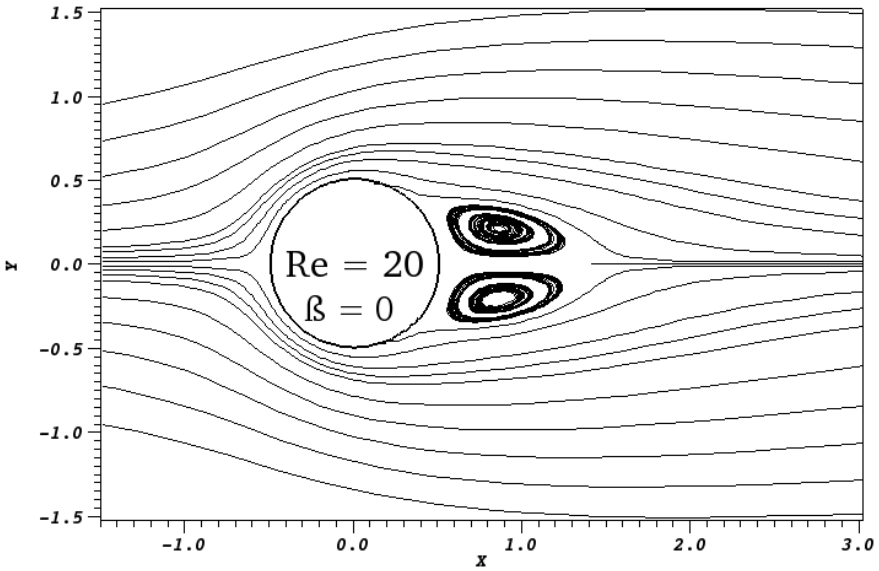
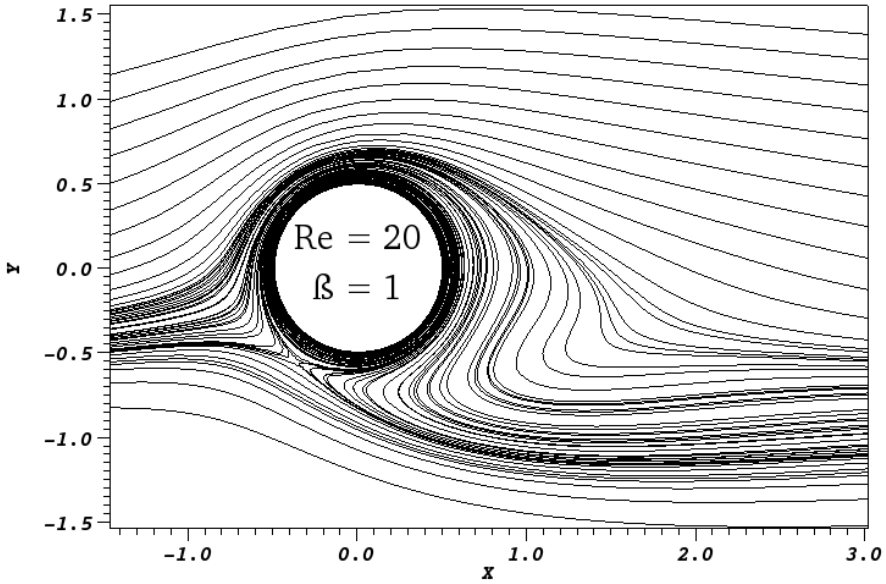


Figure 15: Schematic representation of a fictitious computational domain Ω on a Cartesian grid with the solid domain Ω_s, the liquid domain Ω_f and the immersed boundary Σ mimicking the cylinder. The oncoming velocity is denoted by U.

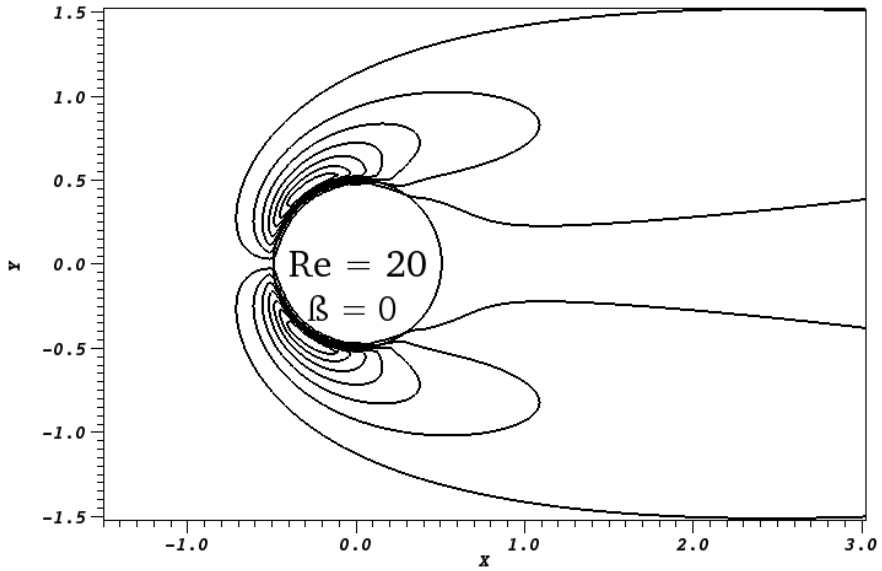


(a) Static cylinder

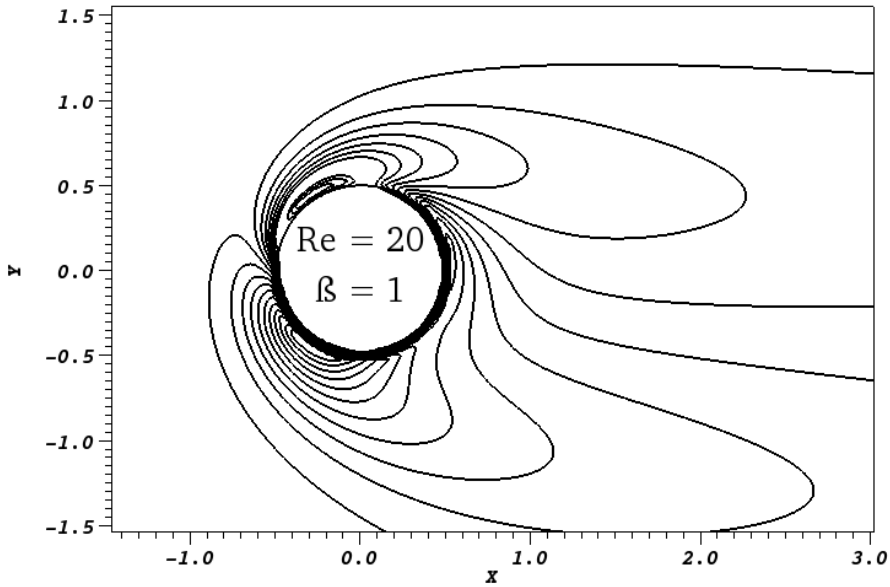


(b) Rotating cylinder

Figure 16: Streamlines around a rotating cylinder obtained with the linear interpolation scheme : Re = 20

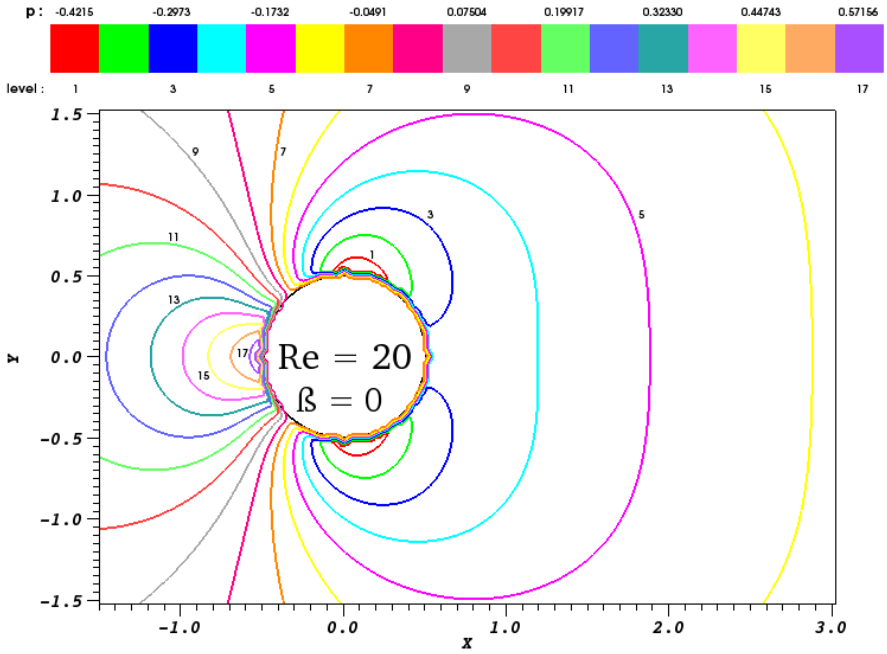


(a) Static cylinder

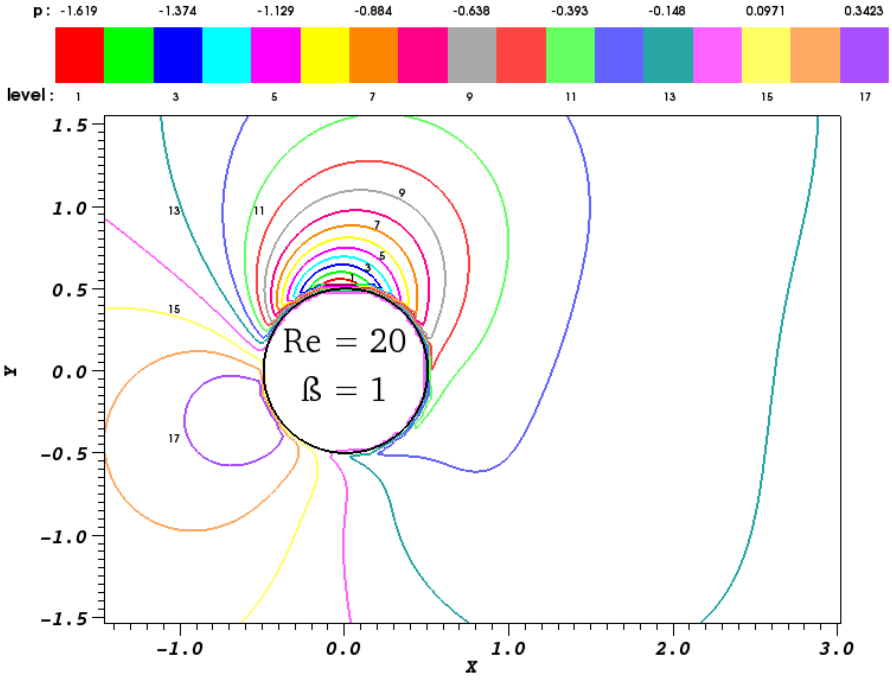


(b) Rotating cylinder

Figure 17: Vorticity contours around a rotating cylinder obtained with the linear interpolation scheme : Re = 20



(a) Static cylinder



(b) Rotating cylinder

Figure 18: Pressure distribution around a rotating cylinder obtained with the linear interpolation scheme : Re = 20

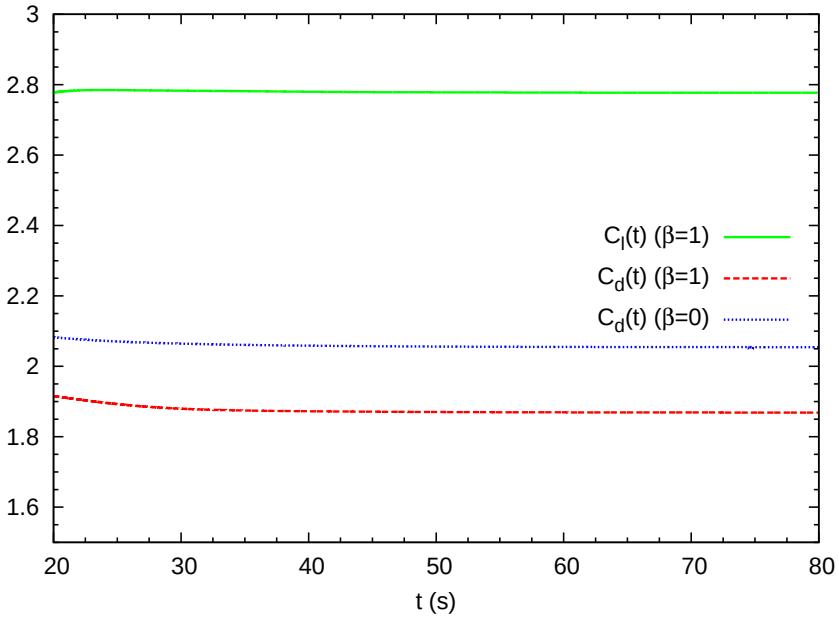


Figure 19: Time convergence of the hydrodynamic coefficients for static ($\beta = 0$) and rotating ($\beta = 1$) cylinders : Re = 20

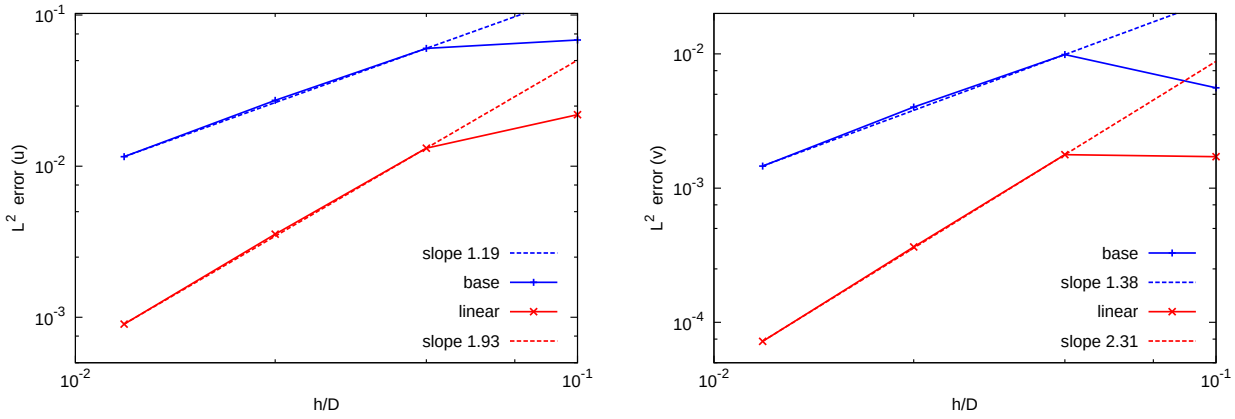


Figure 20: Flow around a static cylinder: $L^2(\Omega_f)$ norm error of the streamwise u (left) and the spanwise v (right) velocity components *vs.* the ratio of the computational grid size h over the smallest radius. Errors calculated over the whole fluid domain. Comparison of the linear interpolation model with the base model.

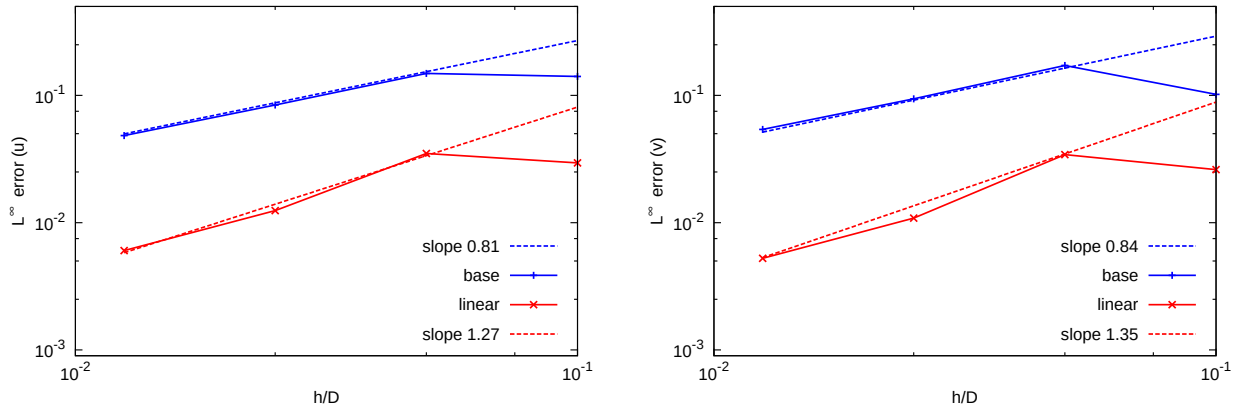


Figure 21: Flow around a static cylinder: $\mathrm{L}^{\infty}(\Omega_f)$ norm error of the streamwise u (left) and the spanwise v (right) velocity components *vs.* the ratio of the computational grid size h over the smallest radius. Errors calculated over the whole fluid domain. Comparison of the linear interpolation model with the base model.

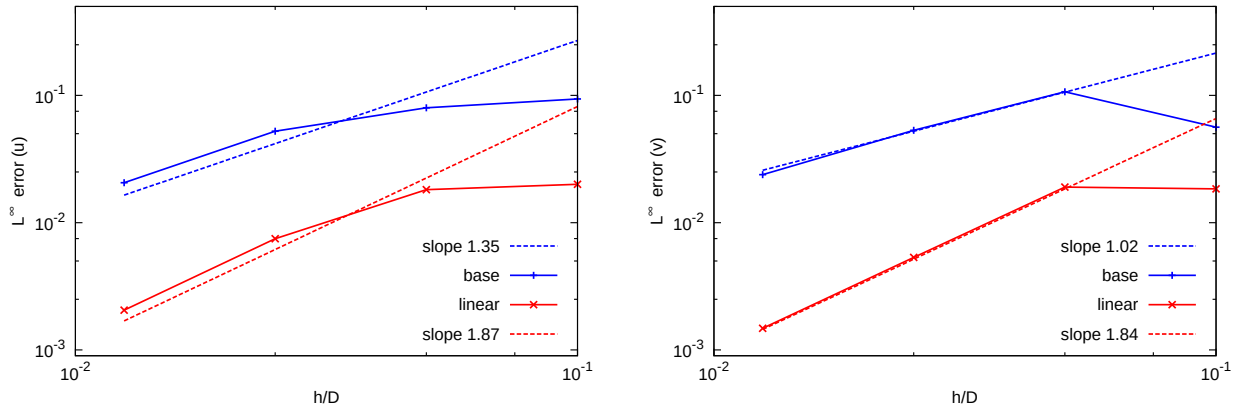


Figure 22: Flow around a static cylinder: $\mathrm{L}^\infty(\Omega_f)$ norm error of the streamwise u (left) and the spanwise v (right) velocity components *vs.* the ratio of the computational grid size h over the smallest radius. Errors calculated over 90% of the fluid domain. Comparison of the linear interpolation model with the base model.

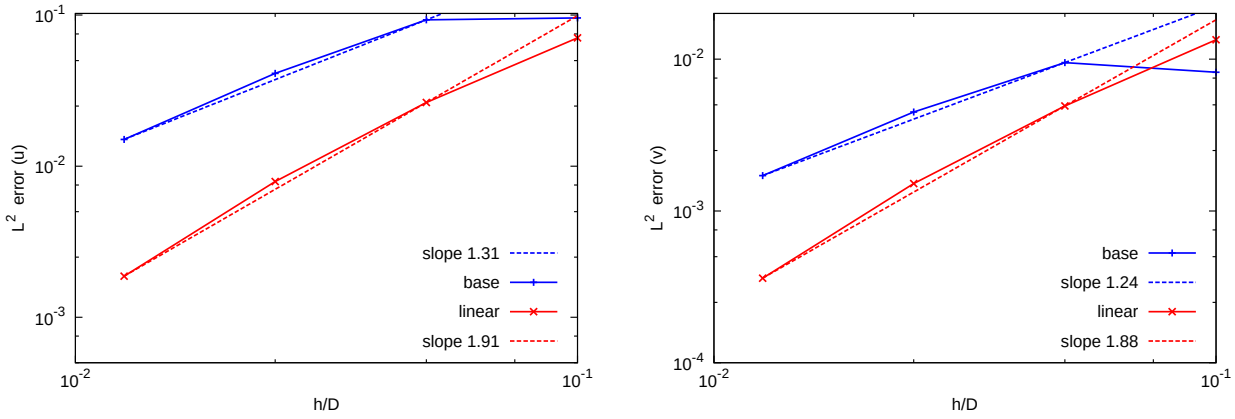


Figure 23: Flow around a rotating cylinder: $L^2(\Omega_f)$ norm error of the streamwise u (left) and the spanwise v (right) velocity components *vs.* the ratio of the computational grid size h over the smallest radius. Errors calculated over the whole fluid domain. Comparison of the linear interpolation model with the base model.

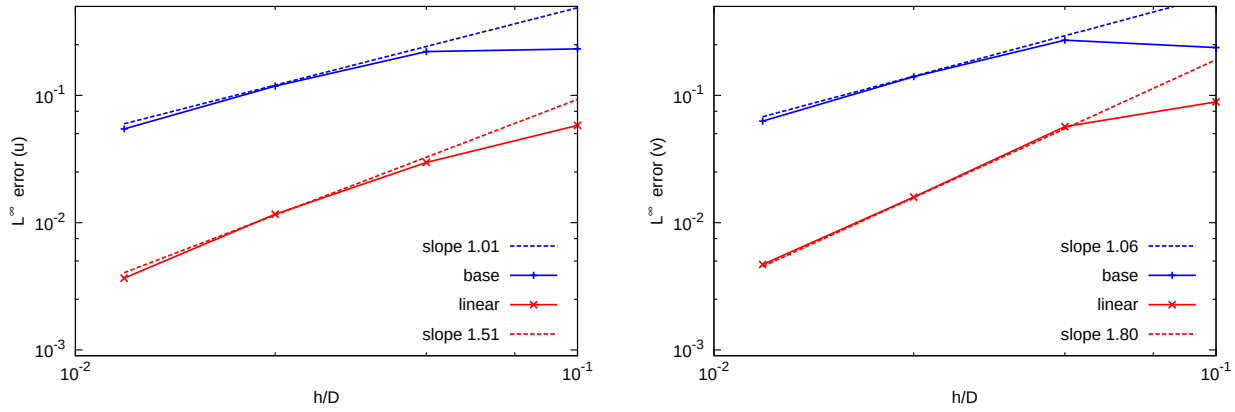


Figure 24: Flow around a rotating cylinder: $\mathrm{L}^\infty(\Omega_f)$ norm error of the streamwise u (left) and the spanwise v (right) velocity components *vs.* the ratio of the computational grid size h over the smallest radius. Errors calculated over the whole fluid domain. Comparison of the linear interpolation model with the base model.

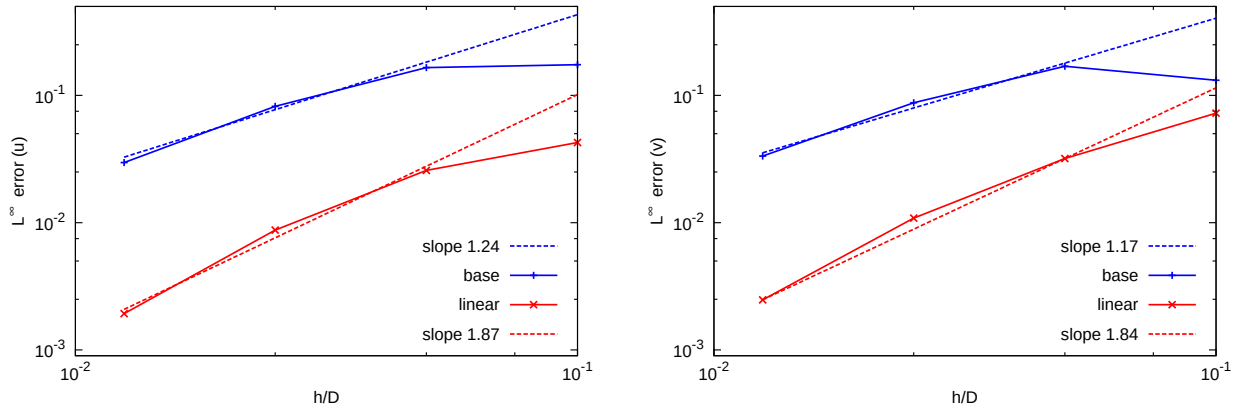
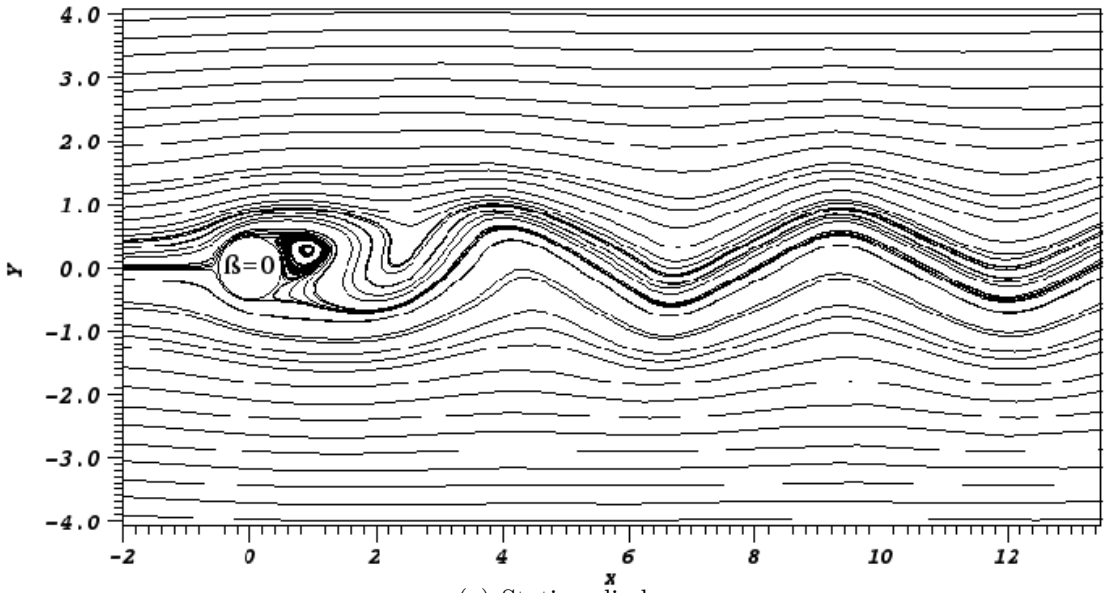
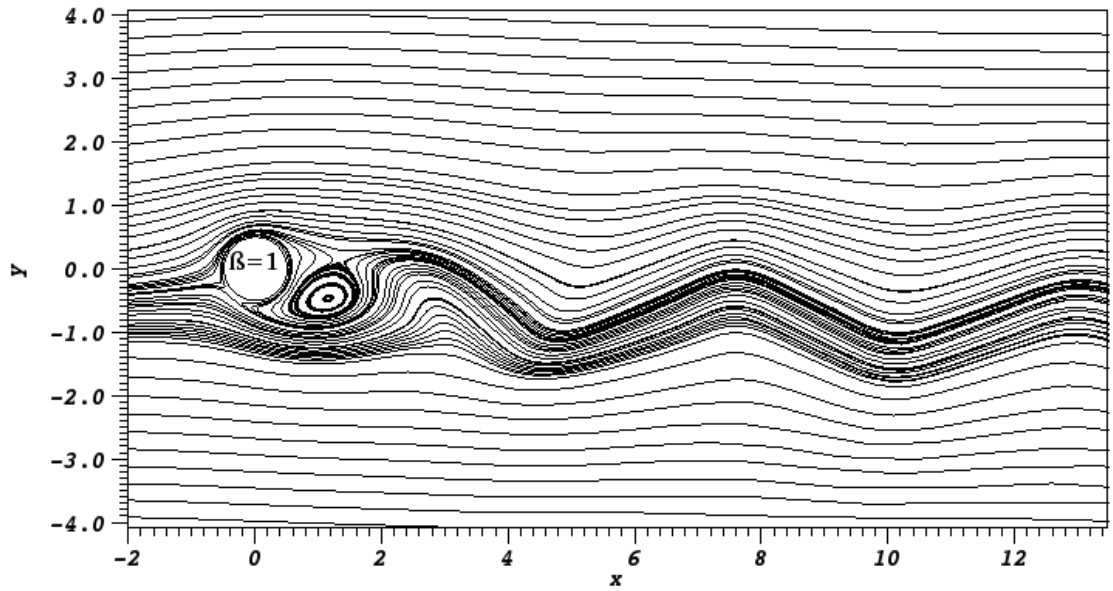


Figure 25: Flow around a rotating cylinder: $\mathrm{L}^\infty(\Omega_f)$ norm error of the streamwise u (left) and the spanwise v (right) velocity components *vs.* the ratio of the computational grid size h over the smallest radius. Errors calculated over 90% of the fluid domain. Comparison of the linear interpolation model with the base model.

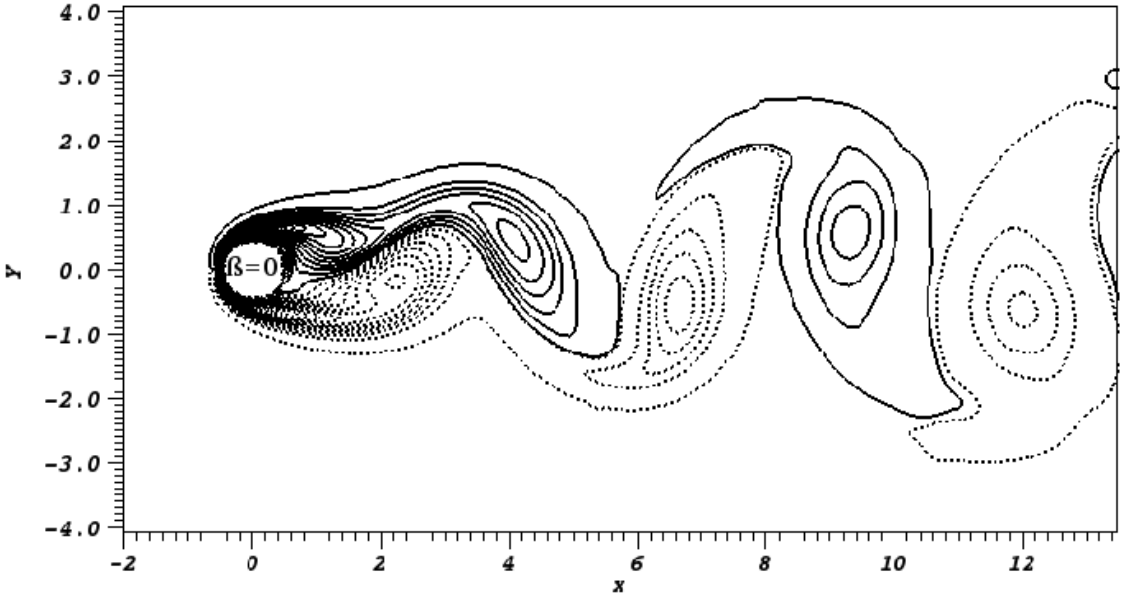


(a) Static cylinder

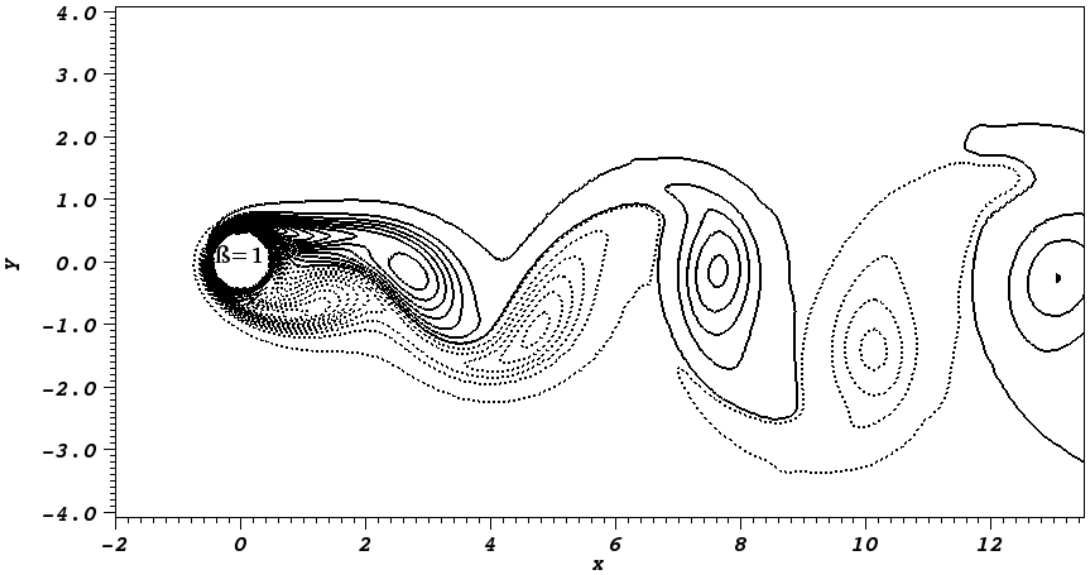


(b) Rotating cylinder

Figure 26: Streamlines around static and rotating cylinders : Re = 100

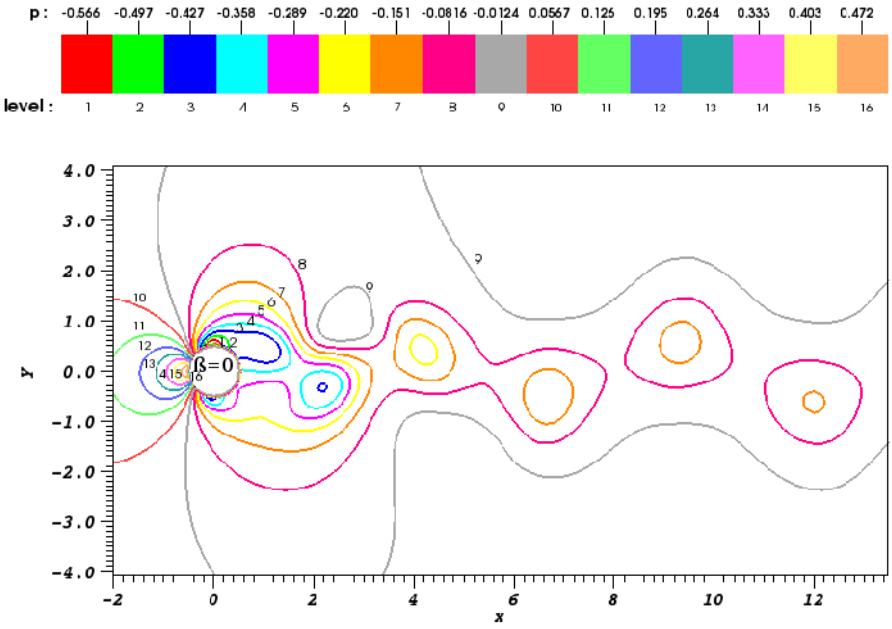


(a) Static cylinder (solid lines : negative contours ; dotted lines : positive contours)

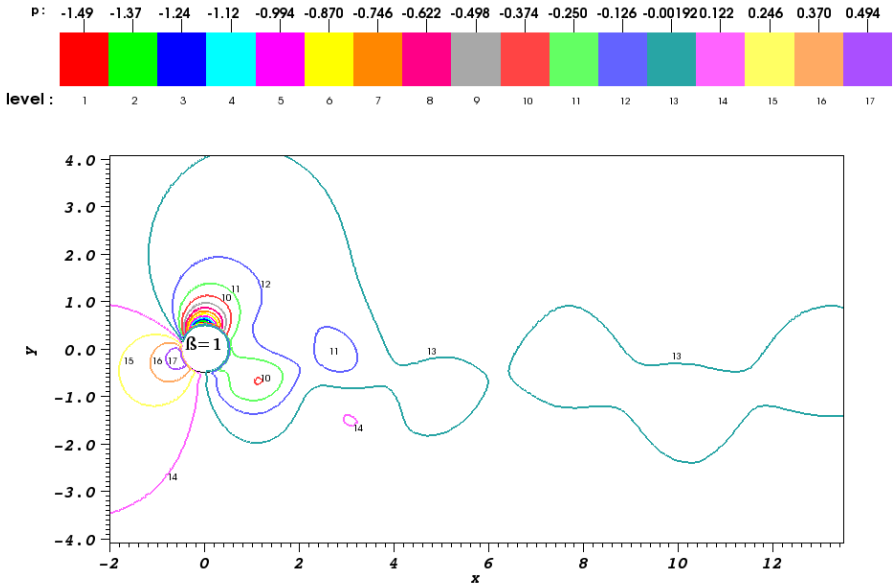


(b) Rotating cylinder (solid lines : negative contours ; dotted lines : positive contours)

Figure 27: Vorticity contours around static and rotating cylinders : Re = 100

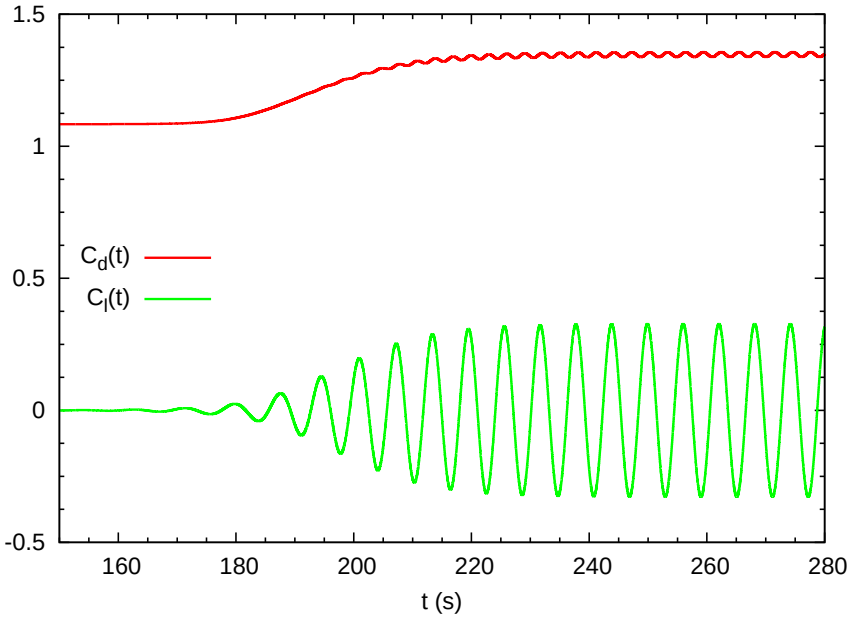


(a) Static cylinder

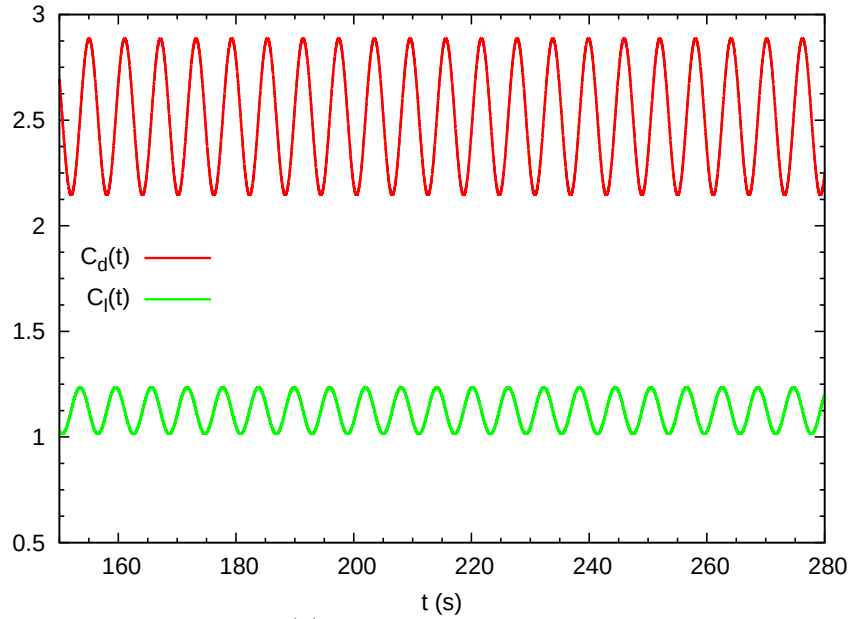


(b) Rotating cylinder

Figure 28: Pressure distribution around static and rotating cylinders : Re = 100

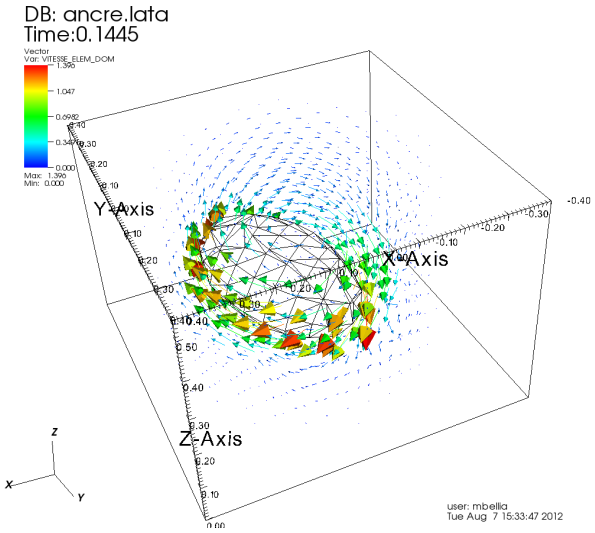


(a) Static cylinder

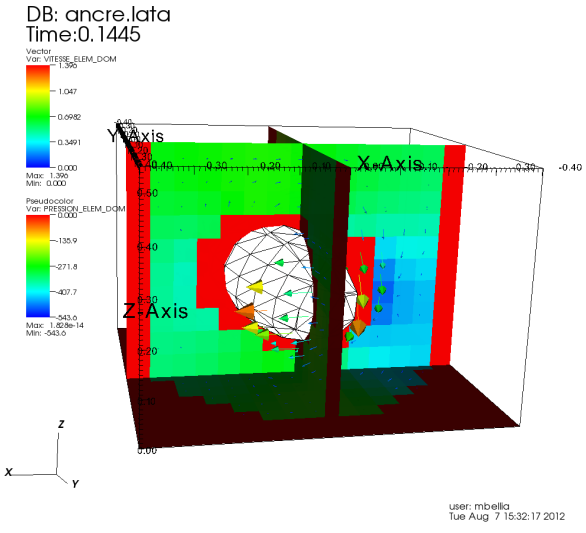


(b) Rotating cylinder

Figure 29: Time evolution of the hydrodynamic coefficients for static and rotating cylinders : Re = 100



(a) Velocities



(b) Velocities and pressure

Figure 30: 3-D flow induced by an ellipsoidal stirrer.

List of Tables

ω_1 (s^{-1})	ω_2 (s^{-1})	r_1 (m)	r_2 (m)	ρ ($kg.m^{-3}$)	μ (Pa.s)	Re	Ta
1	-1	0.1	0.2	1	r_1/Re	1	1.5

Table 1: Numerical parameters used in the Taylor-Couette problem

ω (s^{-1})	U ($m.s^{-1}$)	D (m)	L (m)	ρ ($kg.m^{-3}$)	μ (Pa.s)	Re	β
2	1	1	$60D$	1	1/Re	20 & 100	1

Table 2: Numerical parameters used for the test of a laminar flow around a circular cylinder

		D/h			References					
		10	20	40	[1]	[14]	[26]	[41]	[42]	[43]
C_d	base	2.066	2.094	2.059	2.03	2.02	2.06	2.06	2.00	2.09
	linear	2.085	2.071	2.054						
$\frac{L_w}{D}$	base	0.82	0.98	0.925	0.92	0.9	0.94	0.93	0.91	-
	linear	0.92	0.91	0.9						

Table 3: Hydrodynamic coefficients associated with the problem of steady flows around a static cylinder

		D/h			References			
		10	20	40	[2]	[40]	[44]	[45]
C_d	base	1.8968	1.8703	1.8608	1.888	~ 1.85	1.925	2.000
	linear	1.9104	1.8746	1.8679				
C_l	base	3.0284	3.1097	2.9419	2.629	~ 2.75	2.617	2.740
	linear	2.6248	2.7740	2.7745				
θ	base	57.93°	58.97°	57.68°	54.31°	$\sim 56^\circ$	53.66°	53.87°
	linear	53.95°	55.95°	56.05°				

Table 4: Hydrodynamic coefficients associated with the problem of steady flows around a rotating cylinder

	Present results	References						
		[40]	[41]	[2]	[14]	[13]	[38]	[15]
$\overline{C_d}$	1.347	1.3371	1.34	1.392	1.34	1.35	1.317	1.3757
$C_d^{'}$	$\pm$0.009	$\pm$0.0091	$\pm$0.009	$-$	$\pm$0.011	$\pm$0.012	$\pm$0.009	$\pm$0.0096
$C_l^{'}$	0.326	0.3259	0.333	$-$	0.315	0.303	0.349	0.3393
St	0.165	0.165	0.166	0.172	0.164	0.167	0.170	0.1692

Table 5: Hydrodynamic coefficients associated with the problem of unsteady flows around a static cylinder

	Present results	References		
		[40]	[2]	[46]
$\overline{C_d}$	1.12	1.1080	1.189	1.0979
C_d'	$\pm$0.11	$\pm$0.0986	$\pm$0.1195	$\pm$0.0988
$\overline{C_l}$	2.51	2.504	2.405	2.4833
C_l'	$\pm$0.37	$\pm$0.3616	$\pm$0.4427	$\pm$0.3603
St	0.165	0.1658	0.1732	0.1650

Table 6: Hydrodynamic coefficients associated with the problem of unsteady flows around a rotating cylinder