City Research Online

City, University of London Institutional Repository

Citation: Pinelli, A., Naqavi, I. Z., Piomelli, U. & Favier, J. (2010). Immersed Boundary Method for Generalised Finite Volume and Finite Difference Navier-Stokes Solvers. Journal of Computational Physics, 1(PARTS), pp. 2361-2370. doi: 10.1115/fedsm-icnmm2010-30529

This is the accepted version of the paper.

This version of the publication may differ from the final published version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/6940/

Link to published version: https://doi.org/10.1115/fedsm-icnmm2010-30529

Copyright: City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

Reuse: Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

City Research Online: http://openaccess.city.ac.uk/ publications@city.ac.uk

Immersed-Boundary Methods for General Finite-Difference and Finite-Volume Navier-Stokes Solvers

A. Pinelli[a,*], I. Naqavi[b], U. Piomelli[b], J. Favier[a]

[a] *CIEMAT, Unidad de Modelización y Simulación Numérica, 28040 Madrid, Spain*
[b] *Dept. of Mechanical and Materials Engineering, Queen's University, Kingston (Ontario) K7L 3N6, Canada*

Abstract

We present an immersed-boundary algorithm for incompressible flows with complex boundaries, suitable for Cartesian or curvilinear grid system. The key stages of any immersed-boundary technique are the interpolation of a velocity field given on a mesh onto a general boundary (a line in 2D, a surface in 3D), and the spreading of a force field from the immersed boundary to the neighboring mesh points, to enforce the desired boundary conditions on the immersed-boundary points. We propose a technique that uses the Reproducing Kernel Particle Method [Liu *et al.*, *Int. J. Num. Meth. Fluids* **20**(8) (1995) 1081–1106] for the interpolation and spreading. Unlike other methods presented in the literature, the one proposed here has the property that the integrals of the force field and of its moment on the grid are conserved, independent of the grid topology (uniform or non-uniform, Cartesian or curvilinear). The technique is easy to implement, and is able to maintain the order of the original underlying spatial discretization. Applications to two- and three-dimensional flows in Cartesian and non-Cartesian grid system, with uniform and non-uniform meshes are presented.

1. Introduction

Over the last few years several authors have turned their attention to immersed-boundary methods (IBMs) for their ability to handle moving or deforming bodies with complex surface geometry embedded in a fluid flow. The key feature of IBMs is the fact that the Eulerian grid is not required to conform to the immersed-body geometry, since the no-slip boundary conditions are enforced on the body surface by appropriate boundary forces.

Peskin [1] presented the first application of this method. The flow motion was solved using a uniform mesh (referred to as the *Eulerian grid* in the following). The immersed surface was represented by a set of Lagrangian markers, and the boundary forces took the form of singular functions along the immersed surface in the continuous equations.

*Corresponding author
Email addresses: a.pinelli@ciemat.es (A. Pinelli), ugo@me.queensu.ca (U. Piomelli), julien.favier@ciemat.es (J. Favier)

They were introduced in the discretized equations in conjunction with regularized delta functions that spread (regularize) the force field over the neighboring Eulerian cells. Peskin originally used the IBM to simulate blood flow inside a heart with flexible valves, and the forcing function was computed using Hooke's law [1, 2]. By considering a large value of the spring stiffness, this method could also be applied to rigid boundaries [3, 4].

Other authors obtain the singular forces by a feed-back mechanism [5, 6, 7, 8, 9]: a deviation from the local desired value of velocity (or position) generates a force in the opposite direction, determined by a system of virtual springs and dampers attached to the Lagrangian markers. Undesirable features of these formulations are the introduction of additional free parameters and the compromise that must be taken, when dealing with rigid objects, between severe restrictions on the time step for negligible deformations or larger time steps accepting mild deformation of the embedded body [3, 9].

To avoid the drawbacks of the original IBM formulation for rigid objects, Fadlun *et al.* [10] introduced a direct formulation of the force term. The direct forcing modifies the discretized momentum equation so that the interpolated velocity at the Lagrangian points takes the desired values. Further developments of this technique were proposed, among others, by Kim *et al.* [11], Yang and Balaras [12], and Taira and Colonius [13]. Kim *et al.* [11] used an explicit variant of the direct forcing method that maintains the matrix structure of a standard finite-difference method. Yang and Balaras [12] developed a multi-dimensional interpolation for complex boundary shapes. Taira and Colonius [13] proposed the immersed-boundary projection method, in which the pressure and singular forces are treated together as Lagrangian multipliers in the framework of a projection method.

Fadlun *et al.* [10] present an example of a flow involving moving boundaries; during the relative motion, however, the boundary force is not smooth. This was recognized by Uhlmann [14], who showed that the interpolation procedure used to relate the force at fixed grid nodes and the arbitrarily located boundaries, in simulations with moving bodies, can lead to undesirable force oscillations. This observation led to the formulation of an alternative direct-forcing scheme in which the force is first computed on the Lagrangian markers, then spread onto the neighboring Eulerian nodes. Both interpolation and spreading use the same discrete Dirac kernel. The algorithm conveys very smooth hydrodynamic forces while preserving the global order of the spatial scheme. Recently, Vanella and Balaras [15] have presented an extension of the aforementioned method that yields sharp boundary resolution similar to Eulerian direct-forcing schemes and boundary conforming methods, while keeping the simplicity of the original technique.

In most applications of IBMs the underlying Eulerian grid is Cartesian. Curvilinear grids have been used in fewer cases [16, 17]. The common argument against using immersed-boundary methods with curvilinear grids is the increased cost of curvilinear codes, compared to Cartesian ones. However, when the grid points must be clustered close to solid boundaries that are not aligned with grid lines, Cartesian grids may be suboptimal, as the refinement is extended to regions of the flow in which it is not needed, or even to the solid region. Furthermore, in wall-bounded turbulent flows it is generally desirable to use grid cells that are longer in the flow direction than in the other two. Here, the use of meshes in which the grid lines are nearly aligned with the streamlines can lead to very significant savings [17]. Therefore, one of the principal goals of the present contribution is to develop a method that can be applied to both Cartesian (uniform or non-uniform) and curvilinear Eulerian meshes.

One of the reasons that has limited the use of the IBM techniques to Cartesian grids is due to the discrete delta function, which cannot be trivially extended to more complex grid systems without losing some fundamental properties of the regularized force field. In particular, consider the integral conservation of the force (spread onto the Eulerian grid) and of its first moment:

$$\sum_{i,j,k} \mathbf{f}(\mathbf{x}_{i,j,k})\Delta v_{i,j,k} = \sum_{m} \mathbf{F}(\mathbf{X}_m)\ \Delta V_m, \tag{1}$$

$$\sum_{i,j,k} \mathbf{x}_{i,j,k} \times \mathbf{f}(\mathbf{x}_{i,j,k})\Delta v_{i,j,k} = \sum_{m} \mathbf{X}_m \times \mathbf{F}(\mathbf{X}_m)\Delta V_m. \tag{2}$$

In (1-2), $\mathbf{x}_{i,j,k}$ are the Eulerian grid nodes (over which the force has been spread), and $\mathbf{X}_m$ are the Lagrangian markers; $\Delta v_{i,j,k}$ is the volume of the (i,j,k) Eulerian cell and ΔV_m the volume defined about the m-th Lagrangian marker. The sums are the discrete counterparts of three-dimensional integrals. These properties ensure that the forces and moments exerted by the fluid on the solid can be calculated correctly by integrating the force field.

Equations (1-2) are verified if the regularized delta function $\delta_h(\mathbf{s})$ has the following properties [18]:

$$\sum_{i,j,k} \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X})\Delta v_{i,j,k} = 1 \tag{3}$$

$$\sum_{i,j,k} (\mathbf{x}_{i,j,k} - \mathbf{X})\delta_h(\mathbf{x}_{i,j,k} - \mathbf{X})\Delta v_{i,j,k} = 0. \tag{4}$$

Reproducing conditions (3-4) are easily met on an uniform Cartesian grid by a number of regularized delta functions available in the literature. It will be shown later that correction terms must be introduced in the delta approximant if (1-2) are to be satisfied on an arbitrary underlying mesh.

In this paper we will follow the ideas originated by Liu *et al.* [19] to build locally regularized functions that verify any number of integral conditions. These local approximants will be used both for interpolating the velocity field and for spreading the singular force field in the framework of a pressure-correction scheme for the incompressible Navier-Stokes equations. We will also demonstrate that the method conserves the order of accuracy of the spatial discretization both in Cartesian (uniform or non-uniform) meshes and in curvilinear ones. It is worth mentioning that, when higher order reproducing conditions are required, the approximant tends toward an exact delta function, and the resolution of the boundary becomes sharper.

The idea of using modified window functions to extend the immersed-boundary method to a general grid system is not new. Indeed it was originally proposed by Zhang et al. [25] in the context of the finite element method (*i.e.*, the *immersed finite element method*). Later on, it was extended to a number of applications by several authors [24, 26, 27]. Here we formulate the method in a finite difference/ finite volume context, and most importantly, we clarify the necessary conditions to be met for assembling discrete interpolation and spreading operators that are genuinely dual.

In the following, we will first present the mathematical formulation of the proposed immersed-boundary method and describe the numerical model. We then present the

results of the calculation of two- and three-dimensional flows, using both Cartesian (uniform and non-uniform) and curvilinear grids. We will finally draw some conclusions and make recommendations for future work.

2. Mathematical formulation

2.1. The time advancement method

We start by introducing the general framework of a time-discretized formulation based on a fractional step method [20, 21] for the incompressible Navier-Stokes equations. We focus on the pressure-correction method [22] with the only aim of introducing the general idea of the proposed immersed-boundary formulation. More details on the actual numerical implementation, including the temporal and spatial discretizations used for each numerical experiment will be given later.

The standard sequence used to solve the incompressible Navier-Stokes equation by a fractional step method is:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\mathcal{N}_l(\mathbf{u}^n, \mathbf{u}^{n-1}) - \mathcal{G}\phi^{n-1} + \frac{1}{Re}\mathcal{L}(\mathbf{u}^*, \mathbf{u}^n) \tag{5}$$

$$\mathcal{L}\phi = \frac{1}{\Delta t}\mathcal{D}\mathbf{u}^* \tag{6}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \mathcal{G}\phi^n, \tag{7}$$

where $\mathbf{u}^*$ is the predicted (non-solenoidal) velocity field, $\mathbf{u}^n$ is the divergence-free velocity field at time-step n, Δt is the time step, $\mathcal{N}_l$ is the discrete non-linear operator, $\mathcal{G}$ and $\mathcal{D}$ are, respectively, the discrete gradient and divergence operators, $\mathcal{L}$ is the discrete Laplacian, ϕ is a projection variable (related to the pressure field). The operators include coefficients that are specific to the selected time scheme. Note that we assume explicit integration of the advective term, and either explicit or implicit integration of the viscous diffusion, following the technique most often used for the direct or large-eddy simulation of turbulent flows.

Following Uhlmann [14], the sequence above is modified to impose the desired boundary values on the embedded geometry. The time advancement of the momentum equations is carried out in two stages: First, a fully explicit step analogous to (5) is performed, without any constraint on the embedded boundary:

$$\mathbf{u}^* = \mathbf{u}^n - \Delta t \left[\mathcal{N}_l(\mathbf{u}^n, \mathbf{u}^{n-1}) - \mathcal{G}\phi^{n-1} + \frac{1}{Re}\mathcal{L}(\mathbf{u}^n)\right] \tag{8}$$

The velocity field obtained from (8) is then interpolated onto the embedded geometry Γ, which is discretized through a number of Lagrangian marker points with coordinates $\mathbf{X}_k$:

$$\mathbf{U}^*(\mathbf{X}_k, t^n) = \mathcal{I}(\mathbf{u}^*); \tag{9}$$

The form of interpolation operator $\mathcal{I}$ will be specified later.

The values of $\mathbf{U}^*(\mathbf{X}_k, t^n)$ are used to determine a distribution of singular forces along Γ, that restore the prescribed boundary values $\mathbf{U}^\Gamma(\mathbf{X}_k, t^n)$ on Γ:

$$\mathbf{F}^*(\mathbf{X}_k, t^n) = \frac{\mathbf{U}^\Gamma(\mathbf{X}_k, t^n) - \mathbf{U}^*(\mathbf{X}_k, t^n)}{\Delta t}. \tag{10}$$

The singular force field defined over Γ is then transformed by a spreading operator $\mathcal{C}$ into a volume force-field defined on the mesh points $\mathbf{x}_{i,j,k}$:

$$\mathbf{f}^*(\mathbf{x}_{i,j,k}, t^n) = \mathcal{C}\left[\mathbf{F}^*(\mathbf{X}_k, t^n)\right]. \tag{11}$$

The regularized force given by (11) is now used as (5) is solved again (in explicit or implicit form):

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\mathcal{N}_l(\mathbf{u}^n, \mathbf{u}^{n-1}) - \mathcal{G}\phi^{n-1} + \frac{1}{Re}\mathcal{L}(\mathbf{u}^*, \mathbf{u}^n) + \mathbf{f}^* \tag{12}$$

Finally, the algorithm completes the time step, with the usual solution of the pressure Poisson equation and the consequent projection step (7).

In the next Section we will discuss the key points of the algorithm presented above. They are the definitions of the interpolation operator $\mathcal{I}$, and of the convolution kernel involved in $\mathcal{C}$.

2.2. Interpolation and Convolution: Dirac's delta approximants

We propose to use the *Reproducing Kernel Particle Method* (RKPM) to define interpolation and spreading operators that satisfy the conservation properties (1-2). We begin this subsection by outlining the aspects of the method that have a direct bearing on the definition of the interpolation and spreading operators that we propose. A complete review and analysis of the technique can be found in [19, 23], while applications of the RKPM embedded domain technique in a finite element framework are illustrated in [25].

The approximation $f^a(x)$ of the value of a given smooth function $f(s)$ at point $x \in \Omega$ can be expressed as a *kernel* approximation:

$$f^a(x) = \int_\Omega w_d(x-s) f(s) ds \tag{13}$$

where w_d is the kernel (or weighting) function, and the subscript indicates that the kernel depends on an additional parameter d, the dilation parameter. The non-negative kernel function is assumed to be of compact support (*i.e.*, nonzero in a subdomain Ω_I of Ω and zero outside in $\Omega \backslash \Omega_I$). The dilation parameter d determines the dimensions of the support Ω_I. Note that if the kernel function is the delta function, $f^a(x) = f(x)$ and the function is reproduced exactly.

Roma *et al.* [18] proposed a discrete approximation of the delta function

$$w_d(r) = \begin{cases} \frac{1}{6}\left(5 - 3|r| - \sqrt{-3(1-|r|)^2 + 1}\right) & 0.5 \le |r| \le 1.5 \\ \frac{1}{3}\left(1 + \sqrt{-3r^2 + 1}\right) & |r| \le 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

(where $r = (x-s)/d$) that satisfies the following properties:

1. $w_d(r)$ is continuous $\forall r \in \mathbb{R}$;
2. $w_d(r) = 0$ if $|r| \ge 1.5$;
3. $\sum_l w_d(r-l) = 1,\ \forall l \in \mathbb{N}$;
4. $\sum_l (r-l) w_d(r-l) = 0,\ \forall r, l$;

5. $\sum_l \left[w_d(r-l)\right]^2 = 1/2,\ \forall r, l.$

Since the last three properties involve the integers l, the conservation properties (3-4) can be met by a function interpolated using (13) and (14) only if the nodes are equispaced.

To extend this approach to a non-uniform sequence of nodes, following Liu *et al.* [23], we introduce a modified window function

$$\widetilde{w}_d(x-s) = \sum_{i=0}^{n} b_i(y,d)(x-s)^i w_d(x-s), \tag{15}$$

where the unknown polynomial coefficients $b_i(y,d)$ are determined by imposing the reproducing conditions

$$\widetilde{m}_0(x) = \int_\Omega \widetilde{w}_d(x-s)\,ds \quad = \quad 1 \tag{16}$$

$$\widetilde{m}_i(x) = \int_\Omega (x-s)^i\,\widetilde{w}_d(x-s)\,ds \quad = \quad 0 \qquad (i=1,...,N), \tag{17}$$

which are the continuous equivalent of the third and fourth conditions met by $w_d(x-s)$ on an uniform mesh.

It is interesting to observe that conditions (16-17) imply the exact representation of the elements of the canonical polynomial base $\left\{1, x, x^2, ...\right\}$. For $i = 0$, and 1, in fact conditions (16) and (17) yield:

$$\int_\Omega \widetilde{w}_d(x-s)\,ds \quad = \quad 1 \tag{18}$$

$$\int_\Omega s\,\widetilde{w}_d(x-s)\,ds \quad = \quad x; \tag{19}$$

For $i = 2$, then, we have

$$x^2 \int_\Omega \widetilde{w}_d(x-s)\,ds + \int_\Omega s^2\,\widetilde{w}_d(x-s)\,ds - 2x\int_\Omega s\,\widetilde{w}_d(x-s)\,ds = 0 \tag{20}$$

and, therefore,

$$x^2 = \int_\Omega s^2\,\widetilde{w}_d(x-s)\,ds. \tag{21}$$

Following this procedure it is possible to show, by induction, that the moment conditions (16-17) are equivalent to the polynomial-reproducing conditions

$$x^i = \int_\Omega s^i\,\widetilde{w}_d(x-s)\,ds, \quad i=0,1,...,N. \tag{22}$$

If we insert in (22) the definition of the polynomial correction (15), it is possible to write the conditions on the modified moments $\widetilde{m}_i(x)$ in terms of the original moments

$m_i(x) = \int_\Omega (x-s)^i\, w_d(x-s)\, ds = \delta_{i0}$ (where δ_{ij} is Kronecker's delta):

$$\begin{aligned}
\widetilde{m}_0(x) &= \int_\Omega \widetilde{w}_d(x-s)\, ds = \sum_{i=0}^{n} b_i(x) m_i(x) = 1 \\
\widetilde{m}_1(x) &= \int_\Omega (x-s)\, \widetilde{w}_d(x-s)\, ds = \sum_{i=0}^{n} b_i(x) m_{i+1}(x) = 0 \\
\cdots &= \cdots \\
\widetilde{m}_j(x) &= \int_\Omega (x-s)^j\, \widetilde{w}_d(x-s)\, ds = \sum_{i=0}^{n} b_i(x) m_{i+j}(x) = 0 \\
\cdots &= \cdots \\
\widetilde{m}_N(x) &= \int_\Omega (x-s)^N\, \widetilde{w}_d(x-s)\, ds = \sum_{i=0}^{n} b_i(x) m_{i+N}(x) = 0.
\end{aligned}$$

which results in a symmetric linear system

$$\begin{pmatrix} m_0 & m_1 & \cdots & m_N \\ \cdots & \cdots & \cdots & \cdots \\ m_j & m_{j+1} & \cdots & m_{N+j} \\ \cdots & \cdots & \cdots & \cdots \\ m_N & m_{N+1} & \cdots & m_{2N} \end{pmatrix} \begin{pmatrix} b_0 \\ \cdot \\ b_j \\ \cdot \\ b_N \end{pmatrix} = \begin{pmatrix} 1 \\ \cdot \\ 0 \\ \cdot \\ 0 \end{pmatrix} \tag{23}$$

for the unknown polynomial coefficients b_i, $i = 0, 1, ..., N$.

Modified window functions that verify the reproducing conditions in higher dimensions can be obtained by imposing the exact representation of a complete polynomial basis. Note that in 1D to ensure the integral conservation of the force and of its moments, one needs all the linear combinations of $\{1, x\}$ to be exactly represented; in 2D, the linear combinations of $\{1, x, y\}$ and, in 3D, the ones related with $\{1, x, y, z\}$. The exact representation of higher degree polynomials provides a sharper definition of the boundary [15]. Of course, the exact representation of higher order polynomials should be also considered if the underlying order of the *Eulerian* scheme is increased.

In 2D or 3D problems the *mother* window function can be given as a Cartesian product of (14) with itself:

$$w_{\delta,\eta}(x-s, y-t) = w_\delta(x-s)\, w_\eta(y-t) \tag{24}$$

or

$$w_{\delta,\eta,\sigma}(x-s, y-t, z-v) = w_\delta(x-s)\, w_\eta(y-t)\, w_\sigma(z-v) \tag{25}$$

Here, δ, η and σ are again the dilatation parameters in the three coordinate directions.

Next we look for corrected window functions

$$\begin{aligned}
\widetilde{w}_{\delta,\eta}(x-s, y-t) = \big[&b_0 + (x-s)b_1 + (y-t)b_2 + (x-s)(y-t)b_3 \\
&+(x-s)^2 b_4 + (y-t)^2 b_5\big] w_{\delta,\eta}(x-s, y-t)
\end{aligned} \tag{26}$$

(where $b_i = b_i(\delta, \eta, x, y)$ for $i = 0, ..., 5$) in 2D, and similarly,

$$\begin{aligned}
\widetilde{w}_{\delta,\eta,\sigma}(x-s, y-t, z-v) = \big[&b_0 + (x-s)b_1 + (y-t)b_2 + (z-v)b_3 + (x-s)(y-t)b_4 \\
&+(y-t)(z-v)b_5 + (z-v)(x-s)b_6 + (x-s)^2 b_7 \\
&+(y-t)^2 b_8 + (z-v)^2 b_9\big] w_{\delta,\eta,\sigma}(x-s, y-t, z-v)
\end{aligned} \tag{27}$$

in 3D (where $b_i = b_i(\delta, \eta, \sigma, x, y, z)$ and $i = 0, \ldots, 10$). By imposing the exact representation of the members of the polynomial basis, a symmetric linear system $\mathsf{M}_{2D/3D}\ \vec{b} = \vec{e}_1$ is obtained. Where

$$\mathsf{M}_{2D} = \begin{pmatrix} m_{0,0} & m_{1,0} & m_{0,1} & m_{1,1} & m_{2,0} & m_{0,2} \\ m_{1,0} & m_{2,0} & m_{1,1} & m_{2,1} & m_{3,0} & m_{0,2} \\ m_{0,1} & m_{1,1} & m_{0,2} & m_{1,2} & m_{2,1} & m_{0,3} \\ m_{1,1} & m_{2,1} & m_{1,2} & m_{2,2} & m_{3,1} & m_{2,3} \\ m_{2,0} & m_{3,0} & m_{2,1} & m_{3,1} & m_{4,0} & m_{2,2} \\ m_{0,2} & m_{1,2} & m_{0,3} & m_{1,3} & m_{2,2} & m_{0,4} \end{pmatrix} \tag{28}$$

and

$$\mathsf{M}_{3D} = \begin{pmatrix} m_{0,0,0} & m_{1,0,0} & m_{0,1,0} & m_{0,0,1} & m_{1,1,0} & m_{0,1,1} & m_{1,0,1} & m_{2,0,0} & m_{0,2,0} & m_{0,0,2} \\ m_{1,0,0} & m_{2,0,0} & m_{1,1,0} & m_{1,0,1} & m_{2,1,0} & m_{1,1,1} & m_{2,0,1} & m_{3,0,0} & m_{1,2,0} & m_{1,0,2} \\ m_{0,1,0} & m_{1,1,0} & m_{0,2,0} & m_{0,1,1} & m_{1,2,0} & m_{0,2,1} & m_{1,1,1} & m_{2,1,0} & m_{0,3,0} & m_{0,1,2} \\ m_{0,0,1} & m_{1,0,1} & m_{0,1,1} & m_{0,0,2} & m_{1,1,1} & m_{0,1,2} & m_{1,0,2} & m_{2,0,1} & m_{0,2,1} & m_{0,0,3} \\ m_{1,1,0} & m_{2,1,0} & m_{1,2,0} & m_{1,1,1} & m_{2,2,0} & m_{1,2,1} & m_{2,1,1} & m_{3,1,0} & m_{1,3,0} & m_{1,1,2} \\ m_{0,1,1} & m_{1,1,1} & m_{0,2,1} & m_{0,1,2} & m_{1,2,1} & m_{0,2,2} & m_{1,1,2} & m_{2,1,1} & m_{0,3,1} & m_{0,1,3} \\ m_{1,0,1} & m_{2,0,1} & m_{1,1,1} & m_{1,0,2} & m_{2,1,1} & m_{1,1,2} & m_{2,0,2} & m_{3,0,1} & m_{1,2,1} & m_{1,0,3} \\ m_{2,0,0} & m_{3,0,0} & m_{2,1,0} & m_{2,0,1} & m_{3,1,0} & m_{2,1,1} & m_{3,0,1} & m_{4,0,0} & m_{2,2,0} & m_{2,0,2} \\ m_{0,2,0} & m_{1,2,0} & m_{0,3,0} & m_{0,2,1} & m_{1,3,0} & m_{0,3,1} & m_{1,2,1} & m_{2,2,0} & m_{0,4,0} & m_{0,2,2} \\ m_{0,0,2} & m_{1,0,2} & m_{0,1,2} & m_{0,0,3} & m_{1,1,2} & m_{0,1,3} & m_{1,0,3} & m_{2,0,2} & m_{0,2,2} & m_{0,0,4} \end{pmatrix} \tag{29}$$

Here, $\vec{b}$ is the array of unkown polynomial coefficients and $\vec{e}_1$ is the first unit vector of the canonical basis in $\mathbb{R}^6$ in 2D ($\mathbb{R}^{10}$ in 3D). The elements of the matrices are:

$$m_{i,j} = \int_{\Omega_I} (x-s)^i (y-t)^j \, w_{\delta,\eta}(x-s, y-t)\, ds\, dt \tag{30}$$

in 2D, and

$$m_{i,j,k} = \int_{\Omega_I} (x-s)^i (y-t)^j (z-v)^k \, w_{\delta,\eta,\sigma}(x-s, y-t, z-v)\, ds\, dt\, dv \tag{31}$$

in 3D. The choice of the support Ω_I and the solution procedure required to ensure that the system (23) is not singular will be described in the following Subsection.

2.3. Interpolation and convolution: discrete approach

We now discuss the implementation of the IB technique based on RKPM in a discrete, generalized-coordinate finite-difference or finite-volume code. We limit ourselves to the 2D case with a second-order correction polynomial, since the 3D extension is straightforward.

First, the embedded-boundary curve must be discretized into a number of nodes $\mathbf{X}_I,\ \ I = 1, \ldots, N_e$. Around each node $\mathbf{X}_I$ we define a rectangular cage Ω_I that contains at least three nodes of the underlying mesh in each direction, while at the same time minimizing the number of grid nodes contained in the cage. The cage and underlying mesh are sketched in Figure 1. Following the definition (14), the edges of the rectangle measure 3δ in x and 3η in y (recall that δ and η are the dilation parameters in the coordinate directions). Including at least three nodes in each direction in the cage is required to avoid a singular moment matrix when considering second-order polynomials for the correction of the original window function.

To determine the dimensions of the rectangular support centred in $\mathbf{X}_I$, first we look for the grid node closest to $\mathbf{X}_I$, $\mathbf{x}_{\hat{\imath},\hat{\jmath}} = (x_{\hat{\imath},\hat{\jmath}}, y_{\hat{\imath},\hat{\jmath}})$; next, we consider (in a structured mesh)

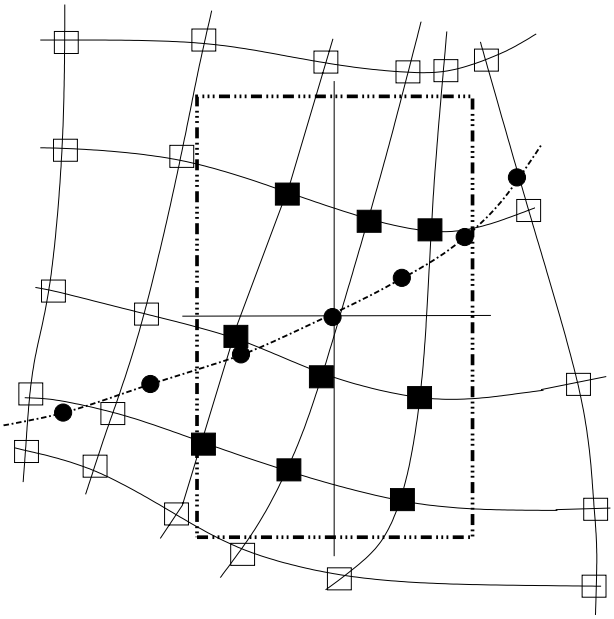

Figure 1: Definition of the support cage. The dashed line is the embedded curve and the dashed rectangle is the support cage Ω_I (centered about X_I). • Lagrangian markers on the embedded curve, □ grid points, ■ grid points within the support.

the set of nodes neighboring $\mathbf{x}_{\hat{\imath},\hat{\jmath}}$, $\mathcal{N}_I = \{\mathbf{x}_{\hat{\imath}+k,\hat{\jmath}+l}\}$, for k and $l = -1, 0, 1$; we then evaluate $h_x^{\pm}$ and $h_y^{\pm}$ as:

$$\begin{cases} h_x^+(\mathbf{X}_I) = \max\left\{|x_{i,j} - x_{i-1,j}| : x_{i,j}, x_{i-1,j} \in \mathcal{N}_I\right\}, \\ h_x^-(\mathbf{X}_I) = \min\left\{|x_{i,j} - x_{i-1,j}| : x_{i,j}, x_{i-1,j} \in \mathcal{N}_I\right\}, \\ h_y^+(\mathbf{X}_I) = \max\left\{|y_{i,j} - y_{i,j-1}| : y_{i,j}, y_{i,j-1} \in \mathcal{N}_I\right\}, \\ h_y^-(\mathbf{X}_I) = \min\left\{|y_{i,j} - y_{i,j-1}| : y_{i,j}, y_{i,j-1} \in \mathcal{N}_I\right\}. \end{cases} \tag{32}$$

Based on these values, we define the length of the edges of the rectangle ($3\delta_I$ and $3\eta_I$) through the local dilation factors:

$$\delta_I = \left(\frac{5}{6}h_x^+(\mathbf{X}_I) + \frac{1}{6}h_x^-(\mathbf{X}_I) + \epsilon_x(\mathbf{X}_I)\right) \tag{33}$$

$$\eta_I = \left(\frac{5}{6}h_y^+(\mathbf{X}_I) + \frac{1}{6}h_y^-(\mathbf{X}_I) + \epsilon_y(\mathbf{X}_I)\right) \tag{34}$$

where $\epsilon_x(\mathbf{X}_I)$ and $\epsilon_y(\mathbf{X}_I)$ are small fractions of the local mesh spacing, and are added to avoid the support boundary touching some of the support nodes (to be defined later); in this case the window function would be zero at those nodes, making the discretized moment matrix singular.

Finally, a set of mesh nodes that fall within the cage is sought:

$$\mathcal{S}_I = \left\{\mathbf{x}_{i,j} : |x_{\hat{\imath},\hat{\jmath}} - x_{i,j}| < \frac{3}{2}\delta_I \quad \text{and} \quad |y_{\hat{\imath},\hat{\jmath}} - y_{i,j}| < \frac{3}{2}\eta_I\right\}. \tag{35}$$

We have verified that with this particular choice, when the underlying mesh is reasonably smooth, the set of nodes within the support is at least nine almost everywhere (27 in 3D).

The elements of the moment matrix (30) must be evaluated numerically to assemble the local window function centered in $\mathbf{X}_I$. We approximate the entries in the moment matrix relative to node $\mathbf{X}_I = (X_I, Y_I)$ using the mid-point quadrature rule:

$$m^I_{i,j} = \sum_{k,l \in \mathcal{S}_I} (x_{k,l} - X_I)^i (y_{k,l} - Y_I)^j \, w_{\delta_I,\eta_I}(x_{k,l} - X_I, y_{k,l} - Y_I) \, \Delta A_{k,l}, \tag{36}$$

where $\Delta A_{k,l}$ is the area of the cell centered at $\mathbf{x}_{k,l}$. The extension of such an approximation to the 3D case (31) is trivial. Once the discrete moment matrix is assembled for each Lagrangian node $\mathbf{X}_I$, the coefficients of the correction polynomials are found by solving, at each Lagrangian point, the symmetric linear system: $\mathsf{M}^I \vec{b}^{\,I} = \vec{e}_1$, for $I = 1, ..., N_e$. M^I are the discrete equivalent of (28) or (29) (i.e., 6×6 in a 2D case and 10×10 in 3D, with a second-order correction polynomial).

Due to the very low values that the window function may take at the nodes close to the boundary of the support cage, the moment matrix may become ill-conditioned. This problem can be avoided by rescaling the linear system, solving the equivalent one $\mathsf{H}^I \mathsf{M}^I (\mathsf{H}^I)^{-1} \, \vec{b}^I = \vec{e}_1$, where

$$\mathsf{H}^I = \mathrm{diag}\left(1, \frac{1}{\delta_I}, \frac{1}{\eta_I}, \frac{1}{\delta_I \eta_I}, \frac{1}{\delta_I^2}, \frac{1}{\eta_I^2}\right) \tag{37}$$

(the 3D form of H^I is a trivial extension of the one above) in two stages:

$$\mathsf{H}^I \mathsf{M}^I \vec{c}^{\,I} = \vec{e}_1, \text{ and } \mathsf{H}^I \vec{c}^{\,I} = \vec{b}^I. \tag{38}$$

The coefficient matrix of the first linear system can be equivalently obtained by normalizing the distances $(x_{k,l} - X_I)$ and $(y_{k,l} - Y_I)$ appearing in (36) with the dilation parameters (δ_I and η_I). The matrix product that follows in (38) is needed to undo the scaling.

The methodology developed so far allows the definition of a localized window function $\widetilde{w}_{\delta_I,\eta_I}(\mathbf{x} - \mathbf{X}_I)$ to be used in the convolution integrals symbolically introduced in (9) and (11). In particular, given a component of the velocity field $u_i(x, y)$ known at the mesh nodes $\mathbf{x}_{k,l} \in \mathcal{S}_I$, the interpolated value at node $\mathbf{X}_I$ on the embedded line can be approximated by:

$$U_i(\mathbf{X}_I) = \mathcal{I}(u_i) = \sum_{k,l \in \mathcal{S}_I} u_i(\mathbf{x}_{l,k}) \, \widetilde{w}_{\delta_I,\eta_I}(x_{k,l} - X_I, y_{k,l} - Y_I) \, \Delta A_{k,l}, \tag{39}$$

having used the same quadrature rule as in (36). Once the force component $F_i(\mathbf{X}_I)$ is found from (10), the distribution of the singular forces over the mesh nodes can be obtained using a discrete counterpart of (11) as convolution operator. This operator can be determined by using a quadrature formula over a strip surrounding Γ:

$$f_i(\mathbf{x}_{k,l}) = \mathcal{C}(F_i) = \sum_{I=1}^{N_e} F_i(\mathbf{X}_I) \, \widetilde{w}_{\delta_I,\eta_I}(x_{k,l} - X_I, y_{k,l} - Y_I) \, \varepsilon_I \, \Delta s_I \tag{40}$$

where Δs_I is length of the arc joining $\mathbf{X}_{I+1/2}$ to $\mathbf{X}_{I-1/2}$, and ε_I is a characteristic strip-width related to the local dilation coefficients of the window function $\widetilde{w}_{\delta_I,\eta_I}$.

To determine the correct value of ε_I we start by considering the value of the force on the markers obtained by interpolation from the nodes of the underlying grid:

$$F_i(\mathbf{X}_I) = \sum_{k,l\in\mathcal{S}_I} f_i(\mathbf{x}_{l,k})\,\widetilde{w}_{\delta_I,\eta_I}(x_{k,l}-X_I, y_{k,l}-Y_I)\,\Delta A_{k,l} \tag{41}$$

Next we replace the values of $f_i(\mathbf{x}_{l,k})$ with those obtained from the spreading step (40):

$$F_i(\mathbf{X}_I) = \left[\sum_{k,l\in\mathcal{S}_I} \Delta A_{k,l}\,\widetilde{w}_{\delta_I,\eta_I}(\mathbf{x}_{k,l}-\mathbf{X}_I)\right] \times \left[\sum_{K=1}^{N_e} F_i(\mathbf{X}_K)\,\widetilde{w}_{\delta_K,\eta_K}(\mathbf{x}_{k,l}-\mathbf{X}_K)\,\varepsilon_K\,\Delta s_K\right]. \tag{42}$$

Rearranging (42) the following conditions are obtained:

$$F_i(\mathbf{X}_I) = \sum_{K=1}^{N_e} a_{I,K}\,\varepsilon_K\,F_i(\mathbf{X}_K), \quad I = 1, ..., N_e \tag{43}$$

where $a_{I,K}$ is the (discrete) integral of the product of the I^{th} and K^{th} window functions over the support of the former one multiplied by Δs_K:

$$a_{I,K} = \Delta s_K \sum_{k,l\in\mathcal{S}_I} \widetilde{w}_{\delta_I,\eta_I}(\mathbf{x}_{k,l}-\mathbf{X}_I)\,\widetilde{w}_{\delta_K,\eta_K}(\mathbf{x}_{k,l}-\mathbf{X}_K)\,\Delta A_{k,l} \tag{44}$$

In matrix notation, the system (43) can be written as:

$$[\mathsf{A}\ \mathrm{diag}(\varepsilon_1,\ldots,\varepsilon_{N_e})]\ \vec{F}_i = \vec{F}_i; \tag{45}$$

By requiring that the local width ε_i is independent of the actual force distribution $\vec{F}_i$, the condition

$$\det\left[\mathsf{A}\ \mathrm{diag}(\varepsilon_1,\ldots,\varepsilon_{N_e}) - \mathsf{I}\right] = 0 \tag{46}$$

(were I is the identity matrix) is obtained. The array $\vec{\varepsilon}$ that verifies the constraint (46) can be found by solving the linear system:

$$\mathsf{A}\,\vec{\varepsilon} = \vec{1} \tag{47}$$

with $\vec{1} = (1, 1, \ldots, 1)^T$.

It is worth noting that, by virtue of (41) and (44), $a_{I,K}/\Delta s_K$ can be interpreted as the value of $\widetilde{w}_{\delta_K,\eta_K}(\mathbf{x}-\mathbf{X}_K)$ evaluated at $\mathbf{x} = \mathbf{X}_I$. Therefore, each I^{th} equation in (47) is a condition imposing that the sum of all the K window functions evaluated at node $\mathbf{X}_I$, weighted by $\varepsilon_K\Delta s_K$, is one. In other words, the weighted window functions $\varepsilon_K\,\Delta s_K\,\widetilde{w}_{\delta_K,\eta_K}(\mathbf{x}-\mathbf{X}_K)$ used to carry out the *spreading* step, must satisfy a partition-of-unity condition [28].

The conditioning of matrix A depends on the ratios between the distances of the Lagrangian nodes Δs_K and the dilation factors δ_I and η_I, or equivalently the ratio between Δs_K and the local Eulerian grid size. To determine an optimal criterion for the selection of the number of Lagrangian nodes to be used to discretize an embedded contour

N_e	$\Delta s/\Delta x$	$\min(\lambda)$	$\max(\lambda)$	$\|\|\text{error}\|\|_\infty$	$\langle\varepsilon\rangle$	$\text{rms}(\varepsilon)$
105	1.1818	1.48	9.85	0.0131	0.103	0.103
115	1.0791	0.714	9.83	0.0131	0.103	0.103
125	0.9921	0.0648	9.81	0.0175	0.102	0.103
135	0.9209	7.85×10^{-3}	9.80	0.0235	0.103	0.116
145	0.8577	2.72×10^{-3}	9.80	0.0349	0.103	0.197
155	0.8024	1.11×10^{-3}	9.81	0.103	0.103	0.401
165	0.7510	4.27×10^{-4}	9.81	0.175	0.104	1.30
175	0.7115	1.61×10^{-4}	9.80	0.313	0.105	2.90
185	0.6719	4.94×10^{-5}	9.80	0.486	0.106	4.46
195	0.6364	1.39×10^{-5}	9.81	1.84	0.119	21.1

Table 1: Spreading and interpolation error for a uniform, 2D Cartesian Eulerian grid covering the surface $[0,5]\times[0,5]$ and a circle of unitary radius with center at $(2.5, 2.5)$. The first column contains the number of Lagrangian markers uniformly distributed along Γ. The second and third columns show the values of the two extreme eigenvalues of A: note that the condition number ($\lambda_{max}/\lambda_{min}$) grows from 6.67 up to 7.01×10^5 in the worst case. The fourth column shows the error after the spreading-interpolation procedure. The last two columns give the average value of the solution of system (47) and its root-mean-square. As a reference value, the interpolation error (only from Eulerian to Lagrangian mesh) for the case $\Delta s/\Delta x = 0.9873$ is approximately 0.0273.

on a given Eulerian mesh we have carried out numerical experiments by considering a circle Γ embedded in a uniform mesh (with the same spacings in the x and y directions) and in a non-uniform one. The circle is discretized with equispaced Lagrangian nodes in the former case and non-equispaced nodes in the latter. We have defined a test function $g(x,y) = \sin\pi x\cos\pi y$ to be used to measure the error induced by the *spreading* operation. We then evaluate the infinity norm of the difference between $g(X_I, Y_I)$ and $g^{\mathcal{IC}}(X_I, Y_I)$, *i.e.,* the values of the test function at the same nodes obtained by first spreading $g(X_I, Y_I)$ onto the Eulerian mesh and then re-interpolating the discrete values over Γ using (39):

$$g^{\mathcal{IC}}(X_I, Y_i) = \mathcal{I}\left\{\mathcal{C}\left[g(X_I, Y_I)\right]\right\}. \tag{48}$$

Table 1 presents a summary of the results obtained for the case of the circle embedded onto an uniform Cartesian Eulerian mesh. The maximum distance between the equispaced Lagrangian nodes is small enough that the function can be represented accurately. The error increases when Δs becomes smaller than the Eulerian mesh spacing. Furthermore, when the Lagrangian nodes become too close to each other, the solution of (47) becomes oscillatory (as shown by $\min(\lambda)\to 0$, while $\max(\lambda)$, which depends on the dimension of the support, remains approximately constant). The deterioration of the solution observed as the number of Lagrangian nodes is increased depends on the fact that we are re-interpolating a function that had been sampled on Γ at a higher resolution, resulting in a Gibbs-like phenomenon. A similar behavior was observed when considering a non-uniform Cartesian mesh. We conclude that the spacing of Lagrangian nodes should be approximately equal to the local grid size. If this requirement is satisfied, the linear system (47) is well-conditioned and the resulting solution $\varepsilon(s)$ turns out to be smooth and positive at all Lagrangian nodes. Note that the positiveness of $\varepsilon(s)$ is consistent with the interpretation of the discrete spreading (40) as an integral over a finite stripe.

When the underlying grid spacing is coarser than the Lagrangian spacing between

neighboring markers, it is still possible to obtain an acceptable solution for $\varepsilon(s)$ either using a singular-value-decomposition when solving (47) (by filtering our the smallest singular values, or by solving a regularized version of the given linear system). We will not discuss further this issue, since the criterion for the distribution of the Lagrangian markers given above appears sufficient to impose the desired boundary value on the immersed boundary accurately.

Finally, it is worth noting that the conservation properties (1) and (2) are verified independently of the value assigned to $\vec{\varepsilon}$, whose role is only to enforce a partition-of-unity condition. Furthermore, the actual entries in matrix A do not need to be computed explicitly, since the action of A over a given vector $\vec{\varepsilon}^{\,p}$ is simply a sequence of a spreading action of the unity function over Γ, followed by an interpolation according to (40) (using $\vec{\varepsilon}^{\,p}$) and (39), respectively. As shown above, if the Lagrangian spacing is comparable to the local Eulerian grid size, any Krylov-type iterative method, applied to (47), converges in few iterations (typically 4-6) without any preconditioning.

2.4. The Navier-Stokes solver

The immersed-boundary method discussed so far has been implemented in a curvilinear finite-volume solver for the incompressible Navier-Stokes equations [29]. The time-advancement follows the procedure roughly described above in Equations (5–7).

In particular, the equations are discretized on a non-staggered grid system using a curvilinear finite volume code. The method of Rhie and Chow [30] is used to avoid pressure oscillations. Both convective and diffusive fluxes are approximated by second-order central differences. A second-order semi-implicit fractional-step procedure [31] is used for the temporal discretization. The Crank-Nicolson scheme is used for the temporal discretization of wall-normal diffusive terms, and the second-order Adams-Bashforth scheme for all the other terms. Fourier transforms are used to reduce the three-dimensional Poisson equation into a series of two-dimensional Helmoltz equations in wavenumber space, which are then solved iteratively using the biconjugate gradient stabilized (BiCGStab) method. The code is parallelized using the MPI message-passing library and the domain-decomposition technique, and has been widely tested [29, 32, 33, 34] in simulations of turbulent flows using curvilinear, body-fitted grids.

3. Results

In this section the accuracy, convergence and robustness of the RKPM-based immersed-boundary method proposed here will be investigated. We consider first the laminar flow around a circular cylinder; we then examine the flow over a two-dimensional hill to show the implementation of the scheme on non-uniform and non-orthogonal grids. Then we implement and test the scheme in a three-dimensional case: the flow around the sphere is simulated in a range of Reynolds numbers that covers both steady and unsteady cases. Finally, we consider a case in which the body is moving, by examining the impulsive start of a two-dimensional flat plate orthogonal to its direction of motion.

3.1. Flow around a circular cylinder

The steady flow around a circular cylinder is considered at two Reynolds numbers (based on the free-stream velocity U_∞ and the cylinder diameter D), $Re_D = 30$ and

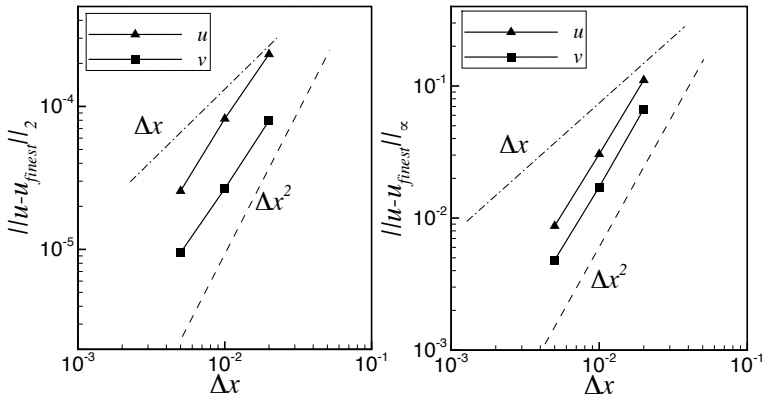

Figure 2: Grid convergence study. *(a)* L_2 and *(b)* L_∞ norms of the error.

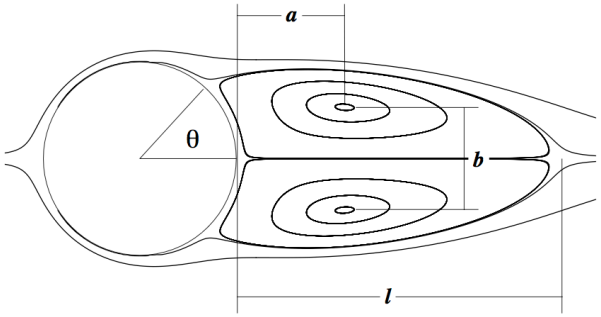

Figure 3: Definition of different parameters of the wake.

$Re_D = 185$. At the low Reynolds number the flow remains steady, while at $Re_D = 185$ periodic vortex shedding is expected; in this case the results depend critically on the accurate reproduction of the vorticity field in the vicinity of the cylinder, which, in turn, depends on a correct prescription of pressure and shear forces at the solid boundary.

The flow around the cylinder is simulated using a domain of dimensions $[-9D, 40D]$ in the streamwise (x) direction and $[-17D, 17D]$ in the vertical (y) direction; the center of the cylinder is at $(0, 0)$. The dimensions of the domain are comparable to those used in other numerical studies [14, 15]. A steady uniform velocity is assigned at the inlet plane, while at the outlet plane convective boundary condition are specified. Free-slip conditions are applied on the top and the bottom of the domain.

The grid is uniform near the cylinder, in the region $-0.6D \leq x \leq D$ and $-D \leq y \leq D$. Outside this region, it is stretched with a stretching ratio of less than 1.02. Four different grids (with $\Delta x = \Delta y = 0.0025D$, $0.005D$, $0.01D$ and $0.02D$ in the uniform region) are considered with 1260, 629, 316 and 153 uniformly spaced Lagrangian points on the boundary of the cylinder. The results for the finest grid are considered accurate, and the L_2 and L_∞ norms of the error obtained on the coarser grids are calculated and shown in Figure 2. The results demonstrate the second-order accuracy of the method.

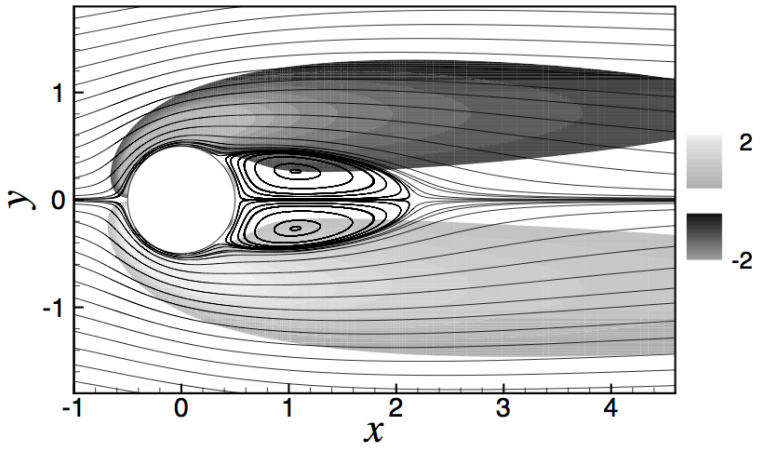

Figure 4: Streamlines and vorticity contours for the steady-state flow around a cylinder at $Re_D = 30$.

	l/D	a/D	b/D	θ	C_D
Present	1.70	0.56	0.52	48.05^o	1.80
Coutanceau and Bouard [35]	1.55	0.54	0.54	50.00^o	—
Tritton [36]	—	—	—	—	1.74

Table 2: Comparison of wake parameters and drag coefficient for steady-state flow around a cylinder at $Re_D = 30$ with experimental data.

The simulation results are also compared with the experimental data. The most important physical feature of this flow is the presence of a recirculating region in the wake of the cylinder. The important parameters associated with the wake are shown in Figure 3; l is the length of the wake, a is the distance from the cylinder to the center of the wake vortex, b is the distance between the centres of the wake vortices, and θ is the angle of separation measured from x-axis. Another important flow parameter is the drag coefficient, $C_D = 2\mathcal{D}/\rho U_\infty^2 D^2$ (where $\mathcal{D}$ is the drag force). Since the interpolation and spreading operators conserve the force, the drag force $\mathcal{D}$ can be calculated directly from the summation of the forces at all Lagrangian points:

$$\mathcal{D} = \sum_{l=1}^{N_e} \mathbf{F}^*(\mathbf{X}_l) \cdot \mathbf{e}_1 \, \Delta s_l, \tag{49}$$

where $\mathbf{e}_1$ is the unit vector in the flow direction.

The streamlines and vorticity contours for this flow are shown in Figure 4. The wake parameters computed from the simulation are in good agreement with the experimental data, as shown in Table 2. To illustrate the effect of the forcing in the vicinity of the immersed body, in Figure 5 we show profiles of pressure p and of the u and v velocity components near the cylinder along three lines: a horizontal one, a vertical one, and one inclined by 45^o from the horizontal (and oriented upstream). Note that the region from $[-0.5, 0.5]$ along the x and y-directions is within the cylinder. We observe some

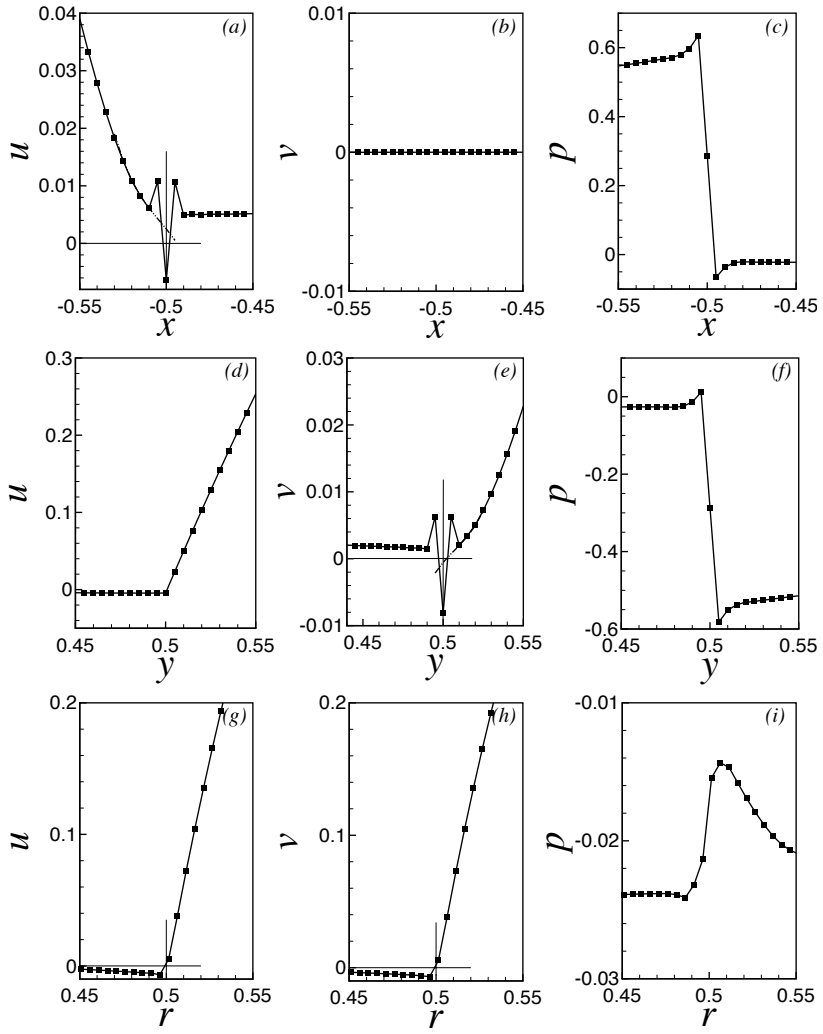

Figure 5: Behavior of u, v, and p near the body. Stagnation point region: *(a)* u, *(b)* v, *(c)* p. Top of the cylinder: *(d)* u, *(e)* v and *(f)* p. Along a line at 45^o from point of stagnation: *(g)* u, *(h)* v and *(i)* p.

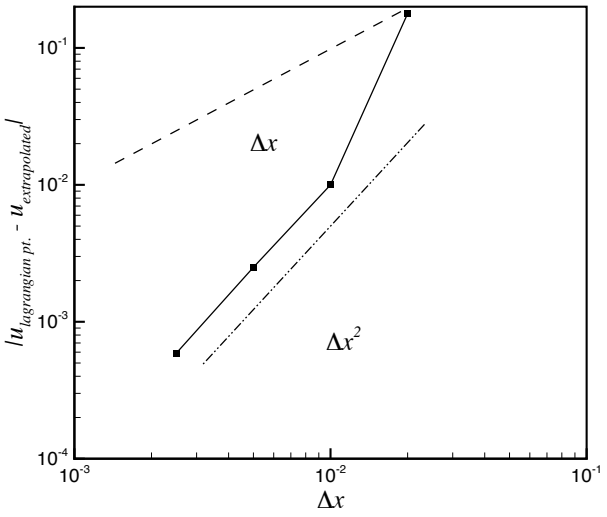

Figure 6: Variation of absolute difference between extrapolated velocity and Lagrangian point value with the grid size.

fluctuations in the velocity component normal to the boundary [u at the stagnation point, Figure 5*(a)*, and v at the top of the cylinder, Figure 5*(e)*], while the velocity component parallel to the boundary goes to zero smoothly. The oscillations, however, are confined to the three points of the support, where the flow is expected to be unphysical, and are not corrupting the rest of the flow field. Also, notice that the interpolation of the velocities to the Lagrangian point results in an exact enforcement of the no-slip condition even when oscillations are present. The pressure shows a large jump, again restricted to the support points, and varies smoothly away from the body. Figures 5 *(g)*, *(h)* and *(i)* show the velocities u, v and pressure p variation along a line at 45^o from the stagnation point. There are no fluctuations in the velocity components, and pressure and velocities go to zero at the cylinder wall smoothly. To further quantify the behavior of the flow in the vicinity of the cylinder boundary a third-order spline extrapolation of the velocity near the boundary is also plotted on the curves (as a dash-dot-dot line) in Figure 5*(a)* and *(e)*. The computation gives zero velocity at the Lagrangian points; however, the extrapolated velocity is not zero. The absolute difference between the extrapolated velocity and the Lagrangian-point value (*i.e.*, the no-slip condition) reduces at second-order rate as the grid is refined, as shown in Figure 6. Also notice that there is motion inside the cylinder (since we do not enforce a zero-velocity condition there); the inner flow is, however, decoupled from the outer one.

We then examined a higher Re_D case, to verify that the unsteadiness of the flow is captured correctly. The domain and the boundary conditions for the higher Reynolds number case were similar to the low-Re_D case. Only two grids were considered, with $\Delta x = \Delta y = 0.01D$ and $0.005D$, respectively, in the region of interest ($-0.6D \leq x \leq D$ and $D \leq y \leq D$).

Figure 7 shows instantaneous vorticity contours and a time series of the drag and lift coefficients, C_D and C_L (where $C_L = 2\mathcal{L}/\rho U_\infty^2 D^2$, and $\mathcal{L}$ is the lift force on the cylinder)

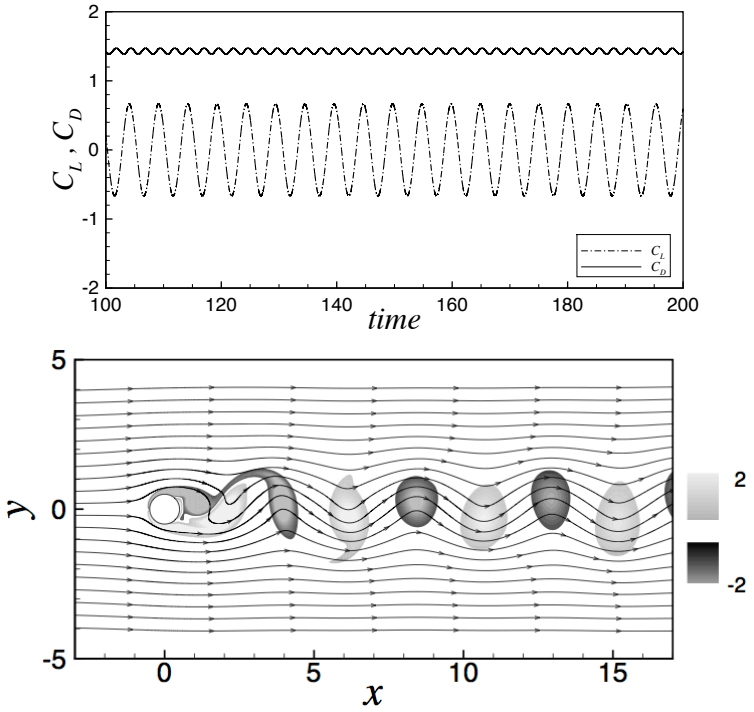

Figure 7: Flow around a cylinder at $Re_D = 185$. *(a)* Time-series of lift and drag coefficients; *(b)* instantaneous vorticity contours and streamlines.

obtained using the finer grid. The periodic fluctuations of drag and lift coefficients indicate a stable vortex shedding behind the cylinder. The instantaneous vorticity contours for this flow are shown in Figure 7*(b)*. The coefficients of drag, lift and Strouhal number $St = D\omega/U_\infty$ (where ω is the shedding frequency) are compared with the values reported in the literature in Table 3. The results obtained with the present method are well within the range of values obtained by other researchers.

3.2. Flow over a two-dimensional hill

An important property of the current forcing scheme is its applicability to non-uniform and non-orthogonal grids without loss of accuracy. To demonstrate this property a two dimensional flow over a small hill in a channel is considered here with a Reynolds number of $Re = 600$ based on the channel height H. The domain size is $35H \times H$. The shape of the hill is defined by $y_h = 0.15H \sin^2[\pi(x - 2.0)]$; thus, the maximum height is 15% of the channel height, its peak is at $x = 2.5$. The no-slip boundary condition is enforced on the top and bottom walls of the channel and a periodic boundary condition is assumed in the streamwise direction. The domain, however, is long enough that the flow returns to a fully developed state, with a parabolic profile, before the end of the computational domain.

Four different grid topologies, shown in Figure 8, are considered. First a non-orthogonal body-fitted grid is used, which conforms to the hill. In this case hill becomes

	C_D	C_L^{rms}	St
Present $\Delta x = 0.005D$	1.430	0.423	0.196
Present $\Delta x = 0.01D$	1.509	0.428	0.199
Vanella and Balaras [15]	1.377	0.461	-
Guilmineau and Queutey [37]	1.280	0.443	0.195
Lu and Dalton [38]	1.310	0.422	0.195
Williamson [39]	-	-	0.193

Table 3: Comparison of coefficients of drag and lift, and Strouhal number for the flow around a cylinder at $Re_D = 185$.

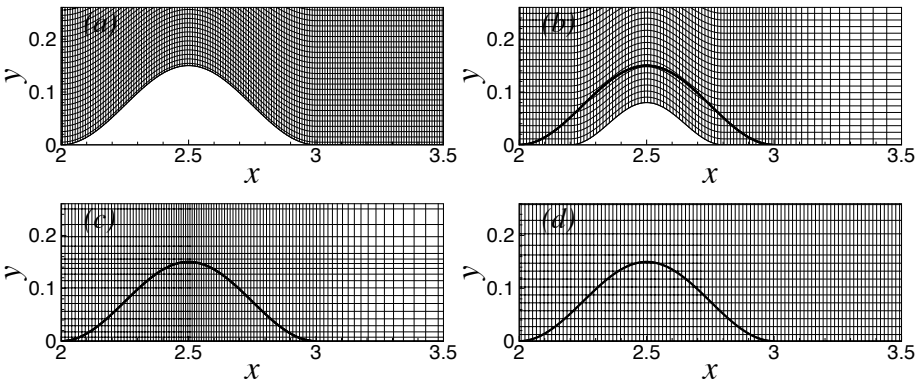

Figure 8: Grids for two-dimensional hill test. *(a)* Body-fitted non-orthogonal grid. *(b)* Non-orthogonal grid with immersed boundary. *(c)* Non-uniform orthogonal grid with immersed body. *(d)* Uniform orthogonal grid with immersed boundary. Every fourth grid line is shown except in (a), where every line is shown. Only part of the domain is shown.

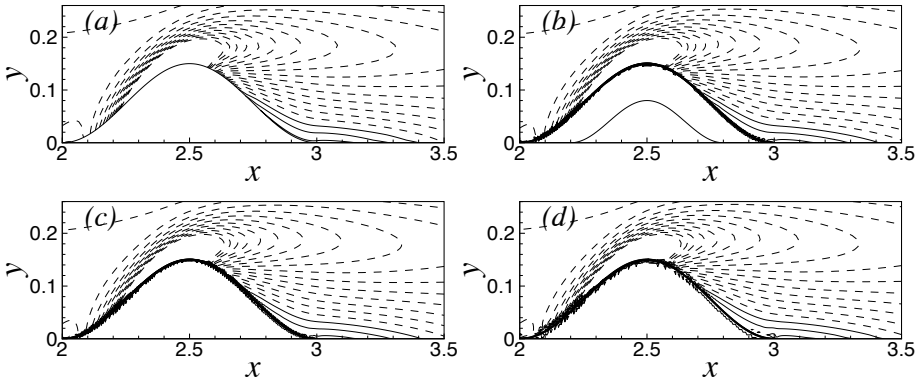

Figure 9: Spanwise vorticity in the flow over the 2D hill. *(a)* Body-fitted non-orthogonal grid. *(b)* Non-orthogonal grid with immersed boundary. *(c)* Non-uniform orthogonal grid with immersed body. *(d)* Uniform orthogonal grid with immersed boundary. 21 equally spaced contour levels from 2 to -8 are shown.

an integral part of the domain boundary and direct no-slip boundary condition is applied. This is considered the reference case, with which the others are compared. The maximum grid size in the region $2H \leq x \leq 4H$ and $0 \leq y \leq 0.25H$ is $0.01H$. This is the region of interest for this flow, since it covers the hill and the recirculation zone. The second grid is also a non-orthogonal grid, but the hill does not coincide with a grid line, and the no-slip conditions on the hill itself are imposed by the immersed-boundary method. The maximum grid size in the region of interest for this case is also $0.005H$. The third grid is a non-uniform orthogonal grid, clustered in y near the top and bottom of the hill, and in x near its leading and trailing edges. In the central part of the hill the grid is stretched in both directions. Here the grid size in the region of interest varies between $0.0025H$ and $0.0075H$. The fourth grid is a uniform orthogonal grid, also with a grid size $\Delta x = \Delta y = 0.005H$ in the region of interest. Except for the first case, the hill is defined as an immersed boundary with 650 uniformly distributed Lagrangian points along the geometry, and the no-slip condition on the hill is applied through the forcing described previously.

The vorticity contours on the hill and in the recirculation region for the four grids are shown in Figure 9. The non-orthogonality and non-uniformity of the grid has no significant effect on the large-scale flow features. The vorticity contours for all the three grids with the IBM compare well with the body-fitted non-orthogonal grid. A quantitative comparison of the u and v velocity profiles is shown in Figure 10. The velocity profiles are compared at three locations, $x = 2.5$ (the peak), $x = 2.75$ (50% along the downward slope) and $x = 3.2$ (in the recirculation region). The u and v velocity profiles for all the four grids are in very good agreement with each other. The maximum error appears to be on the v velocity component, at the hill crest. Even there, the maximum difference between the velocity calculated using the boundary-fitted grid and the methods with the immersed boundary and the uniform grid is less than 7%, decreasing to 3% when non-uniform mesh is used, and the flow gradients are resolved better.

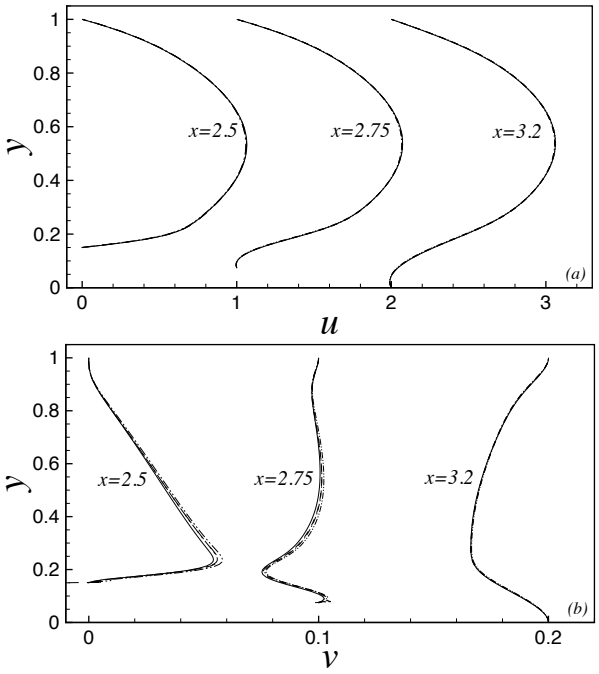

Figure 10: Comparison of velocity profiles at $x = 2.5$, $x = 2.75$ and $x = 3.2$ over the hill. *(a)* u, *(b)* v. ——— Body fitted non-orthogonal grid; − − − Non-orthogonal grid with IB; —·— Non-uniform orthogonal grid with IB; —··— Uniform orthogonal grid with IB.

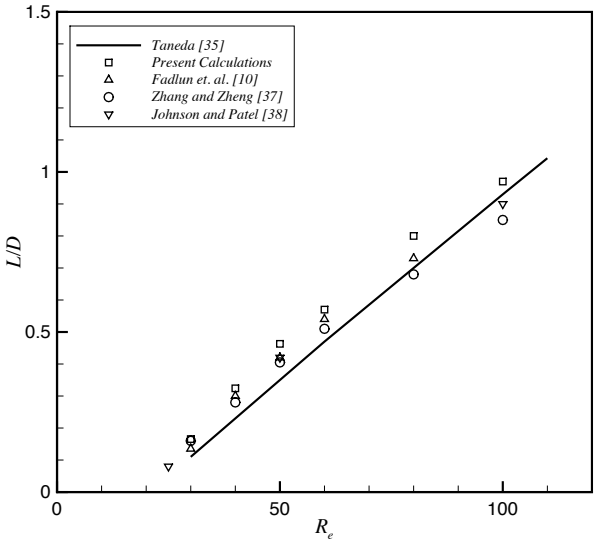

Figure 11: Length of the separation bubble at different Reynolds number.

3.3. Flow around a sphere

The examples discussed above show that the current scheme is able to simulate two-dimensional flows with complex immersed geometries very well. The application of the scheme to three-dimensional immersed objects is tested next by simulating the flow around a sphere at low Reynolds numbers. A Cartesian domain $[-4D, 10D]\times[-4D, 4D]\times[-4D, 4D]$ is used. The sphere is placed with its center at $(0, 0, 0)$. A uniform grid of $\Delta x = \Delta y = \Delta z = 0.05D$ is used around the sphere; the grid is stretched away from the sphere. The sphere is defined using 1258 equally spaced Lagrangian points on its surface. A range of Reynolds numbers Re_D from 30 to 100 is considered. The length of the separation bubble for these Reynolds numbers is compared with the data by Taneda [40] presented in Batchelor [41] and with two other immersed-boundary techniques (by Fadlun *et al.* [10] and Zhang and Zheng [42]) in Figure 11, which shows that the prediction of the length of the separation bubble compares well with other immersed-boundary methods. Another important parameter associated with this flow is the coefficient of pressure $C_p = 2(p - p_\infty)/\rho U_\infty^2$, where p is the pressure at the sphere wall and p_∞ is pressure in undisturbed flow field. The C_p from the present simulation is in very good agreement with the results of Fadlun *et al.* [10] and Zhang and Zheng [42] at $Re_D = 100$, as shown in Figure 12.

The flow around a sphere at low Reynolds numbers remains symmetric and no shedding occurs. However, at a slightly higher Reynolds number the flow become asymmetric; at $Re_D = 280$ vortex shedding is started (Johnson and Patel [43]). We performed a calculation in the shedding region, at $Re_D = 300$, using a Cartesian domain with dimensions $[-4D, 11D]\times[-4.5D, 4.5D]\times[-4.5D, 4.5D]$. The grid size in the vicinity of the sphere is $\Delta x = \Delta y = \Delta z = 0.018D$, and is stretched away from the sphere. The sphere is defined using 10,000 equally spaced Lagrangian points on its surface.

The mean streamwise velocity along the $x-$axis, U_{avg}, in the wake region is compared

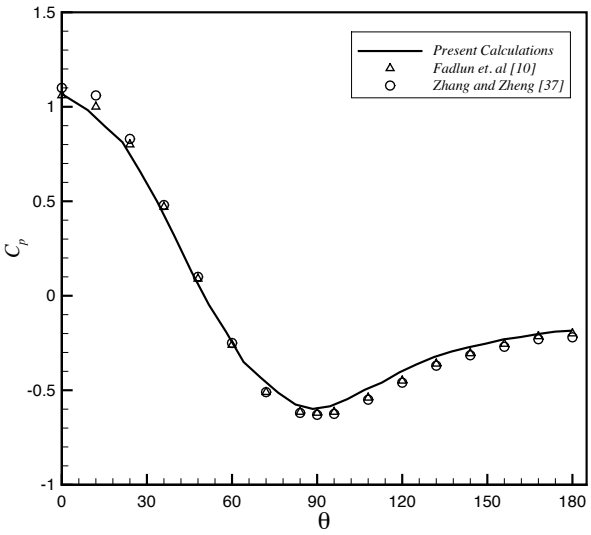

Figure 12: Coefficient of pressure C_p at Reynolds number $Re_D = 100$.

in Figure 13 with the data obtained by Johnson and Patel [43] using a body-fitted grid. There is good agreement between the present calculation and the reference results. The shedding from the sphere results in some unique coherent structures shown in Figure 14. The vortices are visualized as isosurfaces of the second invariant of the velocity gradient tensor, $Q = (u_{i,j}u_{j,i})/2$ and the frames are 20 time units apart. The vortices appear like hairpin vortices. The heads of the shed vortex form very close to the sphere in the wake region and its legs stretch as it moves away from the sphere. The legs are attached to the head of the next vortex. The head of the alternate shedded vortices are inclined in vertically opposite directions with respect to the $x-$axis.

3.4. Suddenly accelerated normal flat plate

As a last example, we consider the case of a moving body, an infinitesimally thin finite flat plate of height h, suddenly accelerated from rest to a constant velocity U_o in the direction normal to its surface in a fluid at rest. The Reynolds number is $Re_h = U_o h/\nu = 1000$. This value matches two simulations of this case by Mittal *et al.* [44], who used of a direct-forcing method, and Koumoutsakos and Shiels [45] who employed a vortex-particle method. The latter is particularly well suited for this problem, since, at least at early times, the vorticity is confined to a small subregion of the domain, thereby easing the computational requirements for the vortex simulations.

In our calculation, the grid is uniform in the direction of the plate motion, while in the normal direction a stretched grid has been employed (with a stretching ratio of 1.045) to cluster the nodes near the ends of the plate. The computational domain, $12h \times 9h$ (in the streamwise and normal direction), has been discretized using 2550×260 points, giving a minimum grid spacing $\Delta x_{min} = \Delta y_{min} = 0.0012h$. The Lagrangian markers are clustered according to the local vertical mesh spacing.

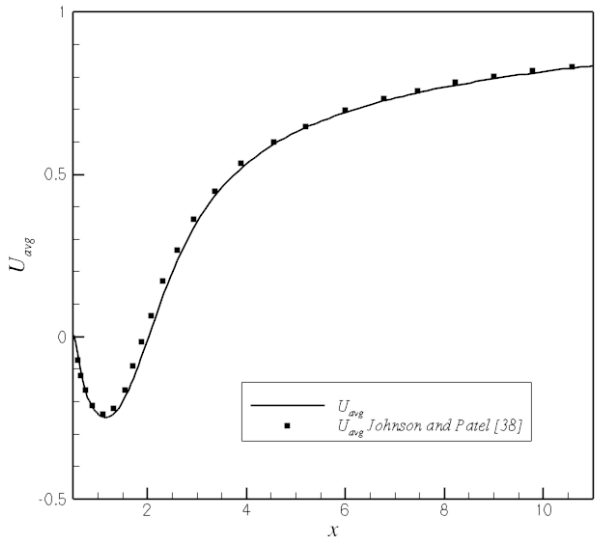

Figure 13: Mean streamwise velocity in the wake region.

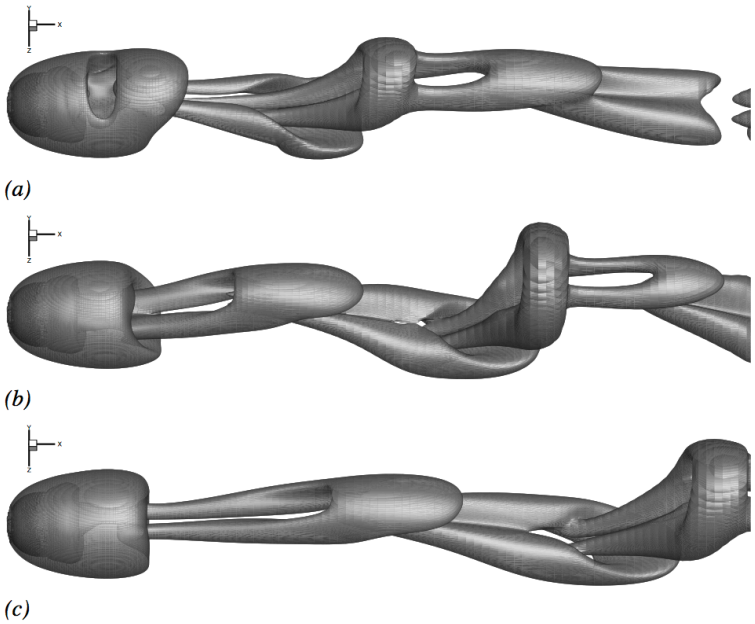

Figure 14: Visualization of the coherent structures shedding from the sphere at $Re_D = 300$. Frames *(a)*, *(b)* and *(c)* are $20D/U_\infty$ time units apart.

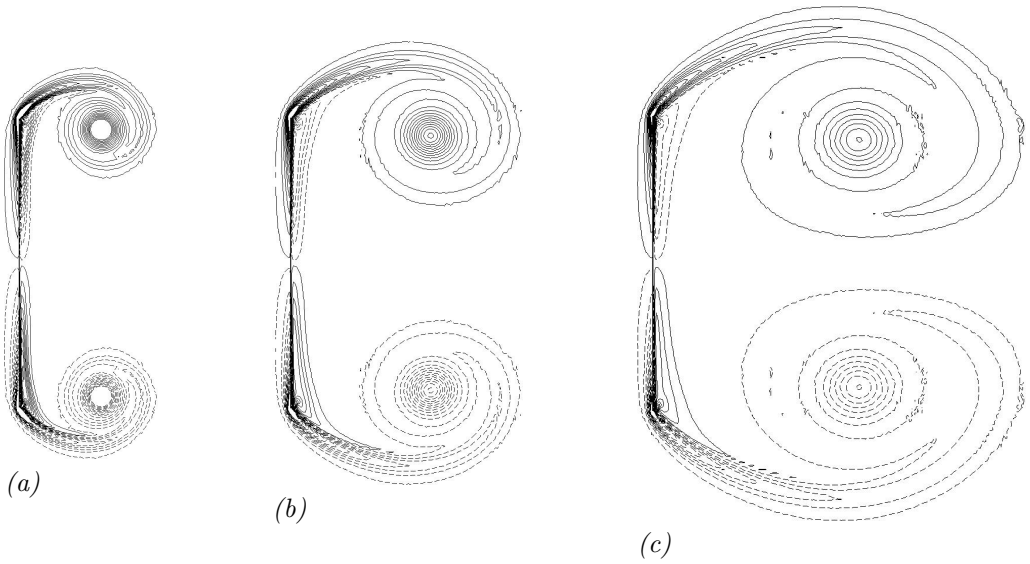

Figure 15: Spanwise vorticity contours at three dimensionless times tU_o/h: *(a)* 0.5; *(b)* 1 and *(c)* 2. Positive vorticity corresponds to solid lines, negative values to dashed lines.

Figure 15 shows the evolution of the wake behind the plate at three times. Very good agreement with the results in the literature (see Figure 18 of [44] and Figure 5 of [45]) is found.

Figure 16 shows the temporal variation of the computed non dimensional bubble length (*i.e.*, the length of the region of reverse flow s/h measured on the centerline behind the plate, normalized by plate height) obtained from the current simulations. Again, an excellent agreement is found with the results of Koumotsakos and Sheils [45]. It is worth noting that this case demonstrates the ability of the forcing algorithm to handle infinitesimally thin bodies including the singular points at the two extremities, with no need to handle this configuration through the use of auxiliary ghost-cell.

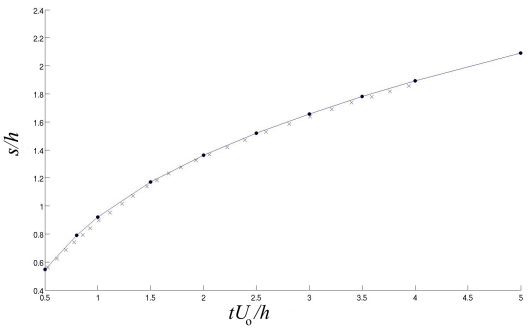

Figure 16: Dimensionless length of the reversed-flow region behind the plate. Solid line and circles corresponds to the present results, $\times$ are values obtained by Koumotsakos and Shiels [45].

4. Conclusions

We have presented a novel interpolation-spreading procedure in the context of immersed-boundary type methods. The global algorithm follows the ideas originally introduced by Peskin [1], and lately extended to rigid moving bodies by Uhlmann [14] and Vanella and Balaras [15], among others. These approaches differ from the direct-forcing methodology introduced by [10], since the evaluation of the forcing function is done on the Lagrangian markers distributed on the embedded geometry, rather than on the Eulerian grid points surrounding it. In principle, the main advantages of this strategy compared to existing direct-forcing schemes is the ease and versatility of its implementation with almost any available solver for parabolic equations. This feature allows decoupling the computation of the required forcing from the computational grid itself. The forcing procedures proposed so far in the literature, however, do not fully exploit the theoretical advantages of the general methodology. The present contribution aims to develop a numerical method leading to a genuine *plug-and-play* module that can be applied to existing solvers based on any spatial discretization.

The method proposed here draws heavily from ideas developed for meshless methods, particularly from the Reproducing Kernel Particle Method (RKPM) [19], its application as an immersed boundary tool in the finite element context [25], and from the partition-of-unity concept [28], for the design of kernel functions that preserve the order of accuracy of the underlying spatial scheme. In particular, the main improvements with respect to other techniques are:

1. It can be applied to codes relying upon any spatial discretization (Cartesian non-uniform, structured body fitted and unstructured grids).
2. One can preserve the underlying order of the spatial scheme while improving the sharpness of the immersed boundaries, in principle, by selecting the order of the reproducing conditions.
3. The spreading and interpolation operators are dual of each other.
4. It can be applied to moving solid objects
5. The computational cost of the forcing procedure is a small fraction of the overall cost required by the original solver.
6. The forces and moments caused by the fluid are calculated in a straightforward manner by integrating the forcing field.

These qualities of the proposed algorithm have been demonstrated numerically applying the proposed forcing method in a curvilinear, finite-volume incompressible Navier-Stokes solver for steady and unsteady flow over two-dimensional and three-dimensional objects, both at rest and in motion. Cartesian uniform and non-uniform grids have been used, as well as curvilinear non-orthogonal meshes. In all cases the numerical results were in excellent agreement with data in the literature, showing the robustness, accuracy, and range of applicability of the proposed methodology.

No tests using unstructured grids have been performed in this study. Nevertheless, the method is inherently general and the only issues that should be faced for this class of discretizations would concern a strategy to determine the support of the window function and a quadrature rule consistent with a cell centered or cell vertex approach.

Acknowledgements

This research was began while AP was visiting the Department of Mechanical and Materials Engineering at Queen's University. The computational support of the High Performance Computing and Virtual Laboratory, Queen's University site, is gratefully acknowledged. AP and JF acknowledge the financial support of the Spanish Ministry of Innovation and Science through the *Consolider* grant *Supercomputación y e-Ciencia*. UP acknowledges the support of the Natural Science and Engineering Research Council of Canada (NSERC) under the Discovery Grant program.

References

[1] C. S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (1972) 252–271.
[2] C. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys 25 (3) (1977) 220–252.
[3] M. Lai, C. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comput. Phys 160 (2) (2000) 705–719.
[4] R. P. Beyer, R. J. LeVeque, Analysis of a one-dimensional model for the immersed boundary method, SIAM J. Num. Anal. 29 (2) (1992) 332–364.
[5] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, J. Comput. Phys. 105 (1993) 354–366.
[6] K. Höfler, S. Schwarzer, Navier–stokes simulation with constraint forces: finite-difference method for particle-laden flows and complex geometries, Phys. Rev. E 61 (6) (2000) 7146–7160.
[7] Z. G. Feng, E. E. Michaelides, The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems, J. Comput. Phys 195 (2) (2004) 602–628.
[8] E. M. Saiki, S. Biringen, Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method, J. Comput. Phys. 123 (2) (1996) 450–465.
[9] C. Lee, Stability characteristics of the virtual boundary method in three-dimensional applications, J. Comput. Phys. 184 (2) (2003) 559–591.
[10] E. A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J. Comput. Phys. 161 (2000) 35–60.
[11] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, J. Comput. Phys. 171 (2001) 132–150.
[12] J. Yang, E. Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, J. Comput. Phys. 215 (2006) 12–40. doi:10.1016/j.jcp.2005.10.035.
[13] K. Taira, T. Colonius, The immersed boundary method: A projection approach, J. Comput. Phys. 225 (2007) 2118–2137.
[14] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2) (2005) 448–476.
[15] M. Vanella, E. Balaras, A moving-least-squares reconstruction for embedded-boundary formulations, J. Comput. Phys. 228 (18) (2009) 6617–6628.
[16] P. Moin, Advances in large eddy simulation methodology for complex flows, Int. J. Heat Fluid Flow 23 (2002) 710–720.
[17] F. Roman, E. Napoli, B. Milici, V. Armenio, An improved immersed boundary method for curvilinear grids, Comput. Fluids 38 (2009) 1510–1527.
[18] A. M. Roma, C. S. Peskin, M. J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (1999) 509–534.
[19] W. K. Liu, Y. Chen, R. A. Uras, C. T. Chang, Generalized multiple scale reproducing kernel particle methods, Comput. Meth. Appl. Mech. Eng. 139 (1-4) (1996) 91–157.
[20] A. J. Chorin, Numerical solution of Navier-Stokes equations, Math. Comput. 22 (104) (1968) 745–762.
[21] R. Temam, Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (I), Arch. Rat. Mech. Anal. 32 (2) (1969) 135–153.

[22] J. Van Kan, A second-order accurate pressure-correction scheme for viscous incompressible flow, SIAM J. Sci. Stat. Comput. 7 (1986) 870–891.
[23] W. K. Liu, S. Jun, Y. F. Zhang, Reproducing kernel particle methods, Int. J. Num. Meth. Fuids 20 (8) (1995) 1081–1106.
[24] X. Wang, W. K. Liu, Extended immersed boundary method using FEM and RKPM, Comput. Meth. Appl. Mech. Eng. 193 (12-14) (2004) 1305–1321.
[25] L. Zhang, A. Gerstenberger, X. Wang, W. K. Liu, Immersed finite element method, Comput. Meth. Appl. Mech. Eng. 193 (21-22) (2004) 2051–2067.
[26] Y. Liu, W. K. Liu, T. Belytschko, N. Patankar, A.C. To, A. Kopacz, J.H. Chung, Immersed electrokinetic finite element method, International Journal for Numerical Methods in Engineering 71 (4) (2006) 379–405.
[27] W.K. Liu, Y. Liu, D. Farrell, L. Zhang, X. S. Wang, Y. Fukui, N. Patankar, Y. Zhang, C. Bajaj, J. Lee, J. Hong, X. Chen, H. Hsu, Immersed finite element method and its applications to biological systems, Comput. Meth. Appl. Mech. Eng. 195 (13-16) (2006) 1722–1749.
[28] I. Babuška, J. Melenk, The partition of unity method, International Journal for Numerical Methods in Engineering 40 (4) (1997) 727–758.
[29] A. Silva Lopes, J. M. L. M. Palma, Simulations of isotropic turbulence using a non-orthogonal grid system, J. Comput. Phys. 175 (2) (2002) 713–738.
[30] C. M. Rhie, W. L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, AIAA J. 21 (1983) 1525–1532.
[31] J. Kim, P. Moin, Application of a fractional step method to incompressible Navier-Stokes equations, J. Comput. Phys. 59 (1985) 308–323.
[32] A. Silva Lopes, U. Piomelli, J. M. L. M. Palma, Large-eddy simulation of the flow in an S-duct, J. Turbul. 7 (11) (2006) 1–24.
[33] S. Radhakrishnan, U. Piomelli, A. Keating, A. Silva Lopes, Reynolds-averaged and large-eddy simulations of turbulent non-equilibrium flows, J. Turbul. 7 (63) (2006) 1–30.
[34] S. Radhakrishnan, U. Piomelli, A. Keating, Wall-modeled large-eddy simulations of flows with curvature and mild separation, ASME J. Fluids Eng. 130 (101203).
[35] M. Coutanceau, R. Bouard, Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow., J. Fluid Mech. 79 (2) (1977) 231–256.
[36] D. J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds numbers, J. Fluid Mech. 6 (1959) 547–567.
[37] E. Guilmineau, P. Queutey, A numerical simulation of vortex shedding from an oscillating circular cylinder, J. Fluids Struct. 16 (6) (2002) 773–794.
[38] X. Y. Lu, C. Dalton, Calculation of the timing of vortex formation from an oscillating cylinder, J. Fluids Struct. 10 (5) (1996) 527–541.
[39] C. H. K. Williamson, Defining a universal and continuous Strouhal–Reynolds number relationship for the laminar vortex shedding of a circular cylinder, Phys. Fluids 31 (10) (1988) 2742–2744.
[40] S. Taneda, Experimental investigation of the wake behind a sphere at low Reynolds numbers, J. Phys. Soc. Japan 1110 (1104–1108).
[41] G. K. Batchelor, An Introduction to Fluid Mechanics, Cambridge Univ. Press, 1967.
[42] N. Zhang, Z. C. Zheng, An improved direct-forcing immersed-boundary method for finite difference applications, J. Comput. Phys. 221 (2007) 250–268.
[43] T. A. Johnson, V. C. Patel, Flow past a sphere up to a reynolds number of 300, J. Fluid Mech. 378 (1999) 19–70.
[44] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas, A. Von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, J. Comput. Phys. 227 (10) (2008) 4825–4852.
[45] P. Koumoutsakos, D. Shiels, Simulations of the viscous flow normal to an impulsively started and uniformly accelerated flat plate, J. Fluid Mech. 328 (177–227).