Journal of Computational Physics 336 (2017) 608–626

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Continuity constrained least-squares interpolation for SFO suppression in immersed boundary methods

Diogo M.C. Martins, Duarte M.S. Albuquerque *, José C.F. Pereira

LAETA, IDMEC, Instituto Superior Técnico, Univerisidade de Lisboa, Lisboa, Portugal

ARTICLE INFO

Article history:
Received 16 July 2016
Received in revised form 30 January 2017
Accepted 8 February 2017
Available online 14 February 2017

Keywords:
Immersed boundary (IB)
Spurious force oscillations (SFO)
Discrete forcing
Flow reconstruction
Least-squares
Moving body problems

ABSTRACT

A new immersed boundary interpolation for discrete forcing methods is presented. The method decreases spurious oscillations in the pressure field and consequently in the body force calculations which are a common issue in several immersed boundary methods. The method applies a continuity constraint in the least-squares interpolation, and guarantees that adjacent interpolation polynomials are continuous between each other. This approach strictly enforces continuity in the flow reconstruction domain, reducing time discontinuities caused by the boundary conditions applied at the immersed boundary. Several moving body problems test cases are simulated, including a three dimensional one, to demonstrate the new method's capability of computing stable pressure fields, even for coarse grids and small CFL numbers, which are known to increase the pressure oscillations.

Nomenclature

a_i, b_i, c_i, l_i – least-squares coefficients
A_m – movement amplitude
a_p, a_l – momentum matrices entries
$[B]$ – least squares coefficients vector (WLS)
D – reference diameter
$\mathbf{d}$ – distance vector between cells P and P_n
f – movement frequency
$\mathbf{f}$ – distance vector between cell P and face f's centroid
$[\mathbf{M}]$ – least squares rectangular matrix (WLS)
$\mathbf{n}, \mathbf{t}$ – normal and tangent versors of solid surface
P – cell or volume control (mesh)
p – pressure
p' – pressure correction
S_f – cell's face (mesh)
$\mathbf{sp}_i$ – position of ith solid point
$\mathbf{S}_f$ – cell face's normal vector
T – movement period

* Corresponding author.
E-mail addresses: diogomatiasmartins@tecnico.ulisboa.pt (D.M.C. Martins), duartealbuquerque@tecnico.ulisboa.pt (D.M.S. Albuquerque), jcfpereira@tecnico.ulisboa.pt (J.C.F. Pereira).

http://dx.doi.org/10.1016/j.jcp.2017.02.026
0021-9991/

U_f – face velocity
$\mathbf{u}$ – velocity vector
u, v, w – velocity components
V_P – cell volume
x, y, z – Cartesian coordinates
α_u, α_p – momentum and pressure relaxation factors
$\beta(x, y)$ – source term
ε – variable error
η – convective scheme weighing factor
ϕ – transported variable
$[\phi]$ – least squares RHS vector (WLS)
ϕ_f – quantity ϕ at face f
ρ – density
θ_p – movement inclination angle with horizontal
ν – kinematic viscosity
ζ_1, ζ_2, ϕ_1, ϕ_2, ψ – analytic functions

Operators
$\nabla\cdot$ – Divergence
∇ – Gradient
$\frac{\partial}{\partial x_i}$ – derivative
$\otimes$ – tensor product
$\frac{D\phi}{Dt}$ – Material derivative

Subscripts and Superscripts
$\overline{a}$ – interpolated values of a
n – iteration number
$*$ – approximated velocity

Acronyms
$CCLS$ – Continuity constrained least-squares
IB – Immersed boundary
SFO – Spurious force oscillation
CFL – Courant–Friedrichs–Lewy number
DFD – Domain-free discretization
CDS – Central differencing scheme

1. Introduction

During the last decades immersed boundary (IB) methods have been used for fluid flow computation with complex geometries because the meshing process is significantly simplified and remeshing becomes unnecessary in moving or deforming body problems.

The IB scene is very diverse, since no single cohesive approach exists and different methods are used for the same conditions. Mittal and Iaccarino [1] divided IB methods in two categories, continuous forcing and discrete forcing. In the continuous forcing approach the solid boundary is represented by adding a source term in the Navier–Stokes equations [2–9], typically distributed over a few fluid nodes near the boundary region. This has the effect of producing a smeared solid boundary, and not strictly being capable of enforcing the no-slip condition [1]. This approach is usually applied to simulate elastic boundaries and has the advantage of being completely independent of the fluid discretization schemes and simple to implement in a variant of fluid solvers.

Discrete forcing methods are popular for rigid boundaries and problems where the boundary-layer is of greater relevance due to its ability to enforce the no-slip condition. The solid boundary is represented by boundary conditions imposed directly at the immersed boundary. Discrete forcing methods are capable of providing a sharp-interface of the immersed boundary, as well as strictly enforce the no-slip condition, unlike continuous forcing methods. Another advantage is the larger time step allowed in discrete methods when compared to continuous forcing methods [1,10]. However the discrete methods are usually intrinsically related to the fluid flow discretization schemes, therefore being potentially more complicated to implement.

A problem that has been often reported in IB simulations of moving boundaries is the existence of oscillations in pressure, see e.g. [10–17], that result in spurious forces oscillations, or SFOs. These oscillations have been reported for virtually all families of IB methods, such has continuous forcing [11,18] and discrete forcing in both ghost-cell type methods [10,13]

and cut-cell methods [16]. In continuous forcing approaches this effect can be significantly reduced by simply increasing the size of the forcing stencil [18], which smoothens discontinuities over the computational grid. However no remedy has been found to eliminate them with discrete forcing methods.

The origin of these SFOs is associated with a spatial discontinuity in the pressure field and a temporal discontinuity in the velocity field. This occurs when cells in the fluid domain are moved into the solid domain (fresh cells) or vice-versa (dead cells) between two time steps [14]. SFOs decrease when the grid size is decreased or the time step is increased. A method developed by Kim and Choi [19] consisting of mass source/sinks enforcing continuity has been successively implemented to dampen pressure oscillations. Seo and Mittal [13] concluded that the violation of the geometric conservation law is the main culprit of the pressure oscillations. They suppressed SFOs with a hybrid ghost/cut-cell method for the continuity equation, which more strictly enforced continuity. Luo et al. [15] argued that an inconsistent treatment between boundary nodes and the bulk flow is responsible for the oscillations, and was capable of suppressing them by smoothing the treatment of the immersed boundary over a few grid cells. Similarly Meinke et al. [16] identified abrupt changes in the discrete body representation as the problem in a cut-cell method. Using again a mass source/sink method Lee and You [10] achieved SFO suppression in a ghost-cell method and used a time-backward integration technique with enhanced stability domain allowing CFL numbers as high as 5.0, which provides benefits in decreasing pressure oscillations. A similar approach was then implemented by Zhang [17] in a local DFD method.

In this work a new interpolation technique is proposed in the context of a discrete-forcing, least-squares flow reconstruction IB method. Immersed boundary methods based on least-squares [20–24] have recently gained popularity due to the great inherent flexibility of the least-squares method. The method proposed in this work has competitive advantages over other methods like ghost-cell and cut-cell because it can be implemented completely independently from the fluid solver. This is similar to continuous forcing techniques, but unlike other discrete forcing methods and results in the method being suited for 3D and unstructured-grids implementation, despite these being beyond the scope of the present work. Furthermore the proposed method avoids the discretization problems and the grid quality issues of cut-cell methods related to the existence of small-cells and the need for mass-source sinks in fluid cells, see e.g. [10,14,17,19].

The IB method proposed in this work resembles the ghost-cell technique used by many authors [10,13]. The method performs a conservative cut in the domain and only cells completely inside the fluid domain are used in the fluid solver, with an added boundary at the interface. The method is extremely versatile because the body influence is represented only by the boundary conditions applied at the interface between fluid and non-fluid cells. The conditions applied at the boundary are computed with a least-squares algorithm featuring a stencil of neighboring fluid cells and solid points. A new flow reconstruction technique is proposed which eliminates continuity errors at the interpolated domain seen in other flow reconstruction methods, therefore eliminating the problem of spurious force oscillations.

The following section describes the numerical discretization techniques used in the developed incompressible Navier–Stokes solver. Section 3 presents the new immersed boundary treatment and this is followed in section 4 by results from several well studied and well documented benchmark test cases demonstrating the new method's capabilities in suppressing pressure oscillations. The paper closes with summarizing conclusions.

2. Numerical method

The numerical method for the bulk flow used in the present work has been thoroughly verified in several flow conditions [25,26]. A detailed description of the numerical method can be found in the original works, and so only a succinct explanation is presented in this section. The finite volume method solves the fluid equations implicitly in a collocated grid arrangement with second order accuracy. The pressure coupling is performed with the PISO algorithm [27].

2.1. Governing equations and discretization methods

The two dimensional unsteady Navier–Stokes equations for incompressible isothermal Newtonian fluid flow are solved numerically. The continuity and momentum equations read as:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\frac{\partial(\mathbf{u})}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = -\frac{1}{\rho}\nabla (p) + \nabla \cdot (\nu \nabla \mathbf{u}) \tag{2}$$

The convective fluxes are discretized by a two point linear interpolation.

$$\phi_f = (1-\eta)\phi_{P_0} + \eta\phi_{P_1} \tag{3}$$

where η depends of distance factors, ϕ_{P_0} and ϕ_{P_1} are the dependent variables at cells P_0 and P_1, respectively. In this work only structured grids are used and therefore η stands for the distance averaging of the cell face to the centroid values of the neighbor cells.

$$\eta = \frac{||\mathbf{f} - \mathbf{P}_0||}{||\mathbf{P}_1 - \mathbf{P}_0||} \tag{4}$$

where $\mathbf{f}$ is the face centroid, and $\mathbf{P}_0$ and $\mathbf{P}_1$ the cell positions of cells P_0 and P_1, respectively.

The central differencing scheme (CDS) is used for discretizing the diffusive terms. This scheme is second order accurate in Cartesian grids and is represented by:

$$(\nabla\phi)_f = \frac{\phi_{P_1} - \phi_{P_0}}{||\mathbf{P}_1 - \mathbf{P}_0||} \tag{5}$$

The Gauss method is used to compute the pressure gradient, which is required for solving the fluid equations.

$$(\nabla p)_P = \frac{1}{V_P} \sum_{f \in \mathcal{F}(P)} p_f \mathbf{S}_f \tag{6}$$

where the second order assumption inside the cell and the Gauss–Legendre quadrature were considered. For the calculation of the face pressure (p_f) the previously mentioned convective scheme is used.

2.2. Pressure velocity coupling algorithm

The bulk flow is solved using a fully implicit formulation of the PISO algorithm [27], in a collocated grid arrangement. PISO is a pressure-based algorithm which improves the pressure coupling of the SIMPLE [28] algorithm by performing an additional pressure correction step.

The algorithm begins by estimating a velocity field $\mathbf{u}^*$ that satisfies the momentum equation, as well as an initial pressure field p^*. This is done using the previous iteration's velocity and pressure fields. The resulting discretized momentum equations can be written in the form:

$$\frac{1}{\alpha_u} a_p \mathbf{u}_P^* + \sum_{l=1}^{F} a_l \mathbf{u}_l^* = -\frac{V_p}{\rho}(\nabla p^n)_P + \frac{1-\alpha_u}{\alpha_u} a_p \mathbf{u}_P^n + \left[C_{low}(\mathbf{u}^n) + C_{high}(\mathbf{u}^n)\right] + \frac{V_P}{\Delta t}\mathbf{u}_P^0 \tag{7}$$

where α_u is the under relaxation factor for the momentum equations, a_P is the main diagonal coefficient of the momentum matrix and a_l the other momentum matrix' elements, which represent the contribution of the P cell's neighbors to the momentum equations, V_P is the volume of cell P, Δt is the time step and $\mathbf{u}_P^0$ is the velocity field from the previous time step. The terms C_{low} and C_{high} appear due to the use of deferred correction iterative procedure and correspond to an upwind convective scheme and the linear convective scheme respectively. The under relaxation factor is required to stabilize the system, by increasing the diagonal dominance of the momentum matrix. Solving eq. (7) implicitly provides the predicted velocity field $\mathbf{u}^*$.

Before performing the pressure correction, a face interpolation is required. Since this method uses a collocated grid approach, a Rhie–Chow interpolation [29] is used in order to eliminate pressure field checkerboarding. The Rhie–Chow interpolation makes use of the cell-centered pressure gradient and the momentum equations, and it is defined as:

$$U_f^* = \mathbf{u}_f^* . \mathbf{S}_f - \overline{\left[\frac{\alpha_u V_P}{\rho a_p}\right]}_f \left[(\nabla p^n)_f - (\overline{\nabla p^n})_f\right] . \mathbf{S}_f \tag{8}$$

where $\mathbf{u}_f^*$ is a prediction of the face centered velocity which is computed with the convective scheme with the velocity values at the cells. The over-line denotes an interpolation between cell-centered quantities using the convective scheme. The velocity field at the face centroid $\mathbf{u}_f^*$ is only conservative after the continuity equation is satisfied, which occurs in the end of the PISO iteration. The factor $\frac{\alpha_u V_P}{\rho a_p}$ relates the pressure gradient with the velocity field, and results from the momentum equation. The pressure gradient at the face $(\nabla p)_f$ is obtained with the diffusive scheme from the pressure values at the adjacent cells and the $(\overline{\nabla p^n})_f$ quantity is obtained by interpolation of the cell centered pressure gradients $(\nabla p)_{P_0}$ and $(\nabla p)_{P_1}$.

The velocity field must be corrected in order to satisfy the continuity equation. To do this, a pressure correction p' is first computed which projects the velocity field to a conservative one. The assembly of the pressure correction equation, which relates the pressure field with the velocity field, results in the following equation:

$$\sum_{f=1}^{F} \overline{\left[\frac{\alpha_u V_P}{\rho a_p}\right]}_f (\nabla p')_f . \mathbf{S}_f = \sum_{f=1}^{F} U_f^* \tag{9}$$

After solving the pressure correction equation for $\nabla p'$, the face velocity correction U_f' and cell velocity correction $\mathbf{u}_P'$ can be computed from the equation:

$$U_f' = -\overline{\left[\frac{\alpha_u V_P}{\rho a_p}\right]}_f (\nabla p')_f . \mathbf{S}_f \tag{10}$$

$$\mathbf{u}_P' = -\frac{\alpha_u V_P}{\rho a_p}(\nabla p')_P \tag{11}$$

The main issue with the velocity correction field is that it does not satisfy the momentum equations. This can be forced by computing a second prediction of the velocity field $\mathbf{u}^{**}$:

$$\mathbf{u}_P^{**} = \frac{1}{a_p}\left[-\sum_{l=1}^{F} a_l \mathbf{u}_l' + C_{low}(\mathbf{u}') - C_{high}(\mathbf{u}')\right] \tag{12}$$

where the values of the matrix come from the momentum matrix used in the first velocity prediction. Afterwards, a face interpolation is required:

$$U_f^{**} = (\mathbf{u}^{**})_f . \mathbf{S}_f \tag{13}$$

and the second pressure correction p'' equation is assembled:

$$\sum_{f=1}^{F} \overline{\left[\frac{\alpha_u V_P}{\rho a_p}\right]}_f (\nabla p'')_f . \mathbf{S}_f = \sum_{f=1}^{F} U_f^{**} \tag{14}$$

where the matrix of this pressure correction is the same as the one used in the previous correction step, so it is not necessary to recompute the matrix values. The right hand side of the system denotes the cell out mass balance of the second predicted velocity field.

Finally, with the second pressure correction p'' the velocity fields at the faces and the cells centers can be updated by:

$$U_f^{n+1} = U_f^{*} + U_f^{**} - \overline{\left[\frac{\alpha_u V_P}{\rho a_p}\right]}_f (\nabla p')_f . \mathbf{S}_f - \overline{\left[\frac{\alpha_u V_P}{\rho a_p}\right]}_f (\nabla p'')_f . \mathbf{S}_f \tag{15}$$

$$\mathbf{u}_P^{n+1} = \mathbf{u}_P^{*} + \mathbf{u}_P^{**} - \frac{\alpha_u V_P}{\rho a_p}(\nabla p')_P - \frac{\alpha_u V_P}{\rho a_p}(\nabla p'')_P \tag{16}$$

and the pressure can be computed by:

$$p^{n+1} = p^n + p' + p'' \tag{17}$$

The PISO algorithm does not require pressure relaxation due to the strong pressure–velocity coupling, even if only two correction steps are being used. In all simulations of this work the under relaxation factor was $\alpha_u = 0.95$.

3. Immersed boundary treatment

In this section the steps involved in the numerical IB method are presented, together with two interpolation techniques used, the standard version and the new constrained interpolation with SFO elimination. The first step of the IB method consists of isolating the fluid domain consisting of all the cells completely immersed in the fluid region. A boundary is created at the interface of the fluid domain, separating it from the remaining computational domain. At this boundary a Dirichlet boundary condition for velocity is imposed and the usual discretization schemes used in Dirichlet boundaries are applied. The imposed velocity at the interface is computed in a way that the body influence is accurately represented in the overall flow. This is achieved using a least-squares interpolation for each immersed boundary face.

3.1. IB categorization

For notation purposes cells which contain all vertices outside the solid body are labeled as *fluid cells*. All non-*fluid cells* which contain a neighboring face with a *fluid cell* are labeled *immersed boundary cells* (or *ib cells* for short) and the neighboring face is labeled as an *ib face*. The remaining cells are labeled as *solid cells* (Fig. 1). The momentum and pressure correction equations are only solved for the *fluid cells*, meaning that the number of *ib* and *solid cells* does not have a significant computational cost associated. An advantage of this kind of categorization is that it does not create grid quality problems as those reported in cut-cell methods. The conservative cut performed in a Cartesian grid around a generic 2D body is shown in Fig. 1, where the fluid domain, *ib faces* and *ib cells* are presented.

The solid body's surface is described by discrete points, with known velocity and position at each time instant (prescribed motion). The solid point's position and velocity could also be calculated from rigid body equations or even from a coupled structural solver, in the case of deforming bodies. Only the closest solid point to each *ib face* is used, which leads to a linearization of the solid boundary. If the solid point spacing is much smaller than Δx, the used solid point can be assumed to be the actual closest point of the solid boundary to each *ib face*'s center, and the spacing between two consecutive solid points is always close to Δx.

At the *ib faces*, a Dirichlet boundary condition is applied for the velocity components and a Neumann boundary condition is applied for the pressure correction. The velocity components to be applied in each *ib face* are found through a least-squares interpolation. The two different interpolation techniques used in this work are described in the following subsections.

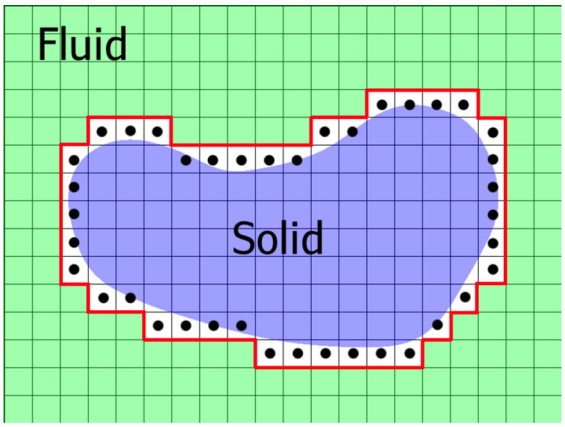

Fig. 1. Example of the conservative cut around a generic solid body (• – *ib cells*).

3.2. The unconstrained interpolation method

The standard method, that does not constrain continuity, interpolates each velocity component by a polynomial of the form:

$$u = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 y^2 + a_5 xy \tag{18}$$

$$v = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 y^2 + b_5 xy \tag{19}$$

Notice that a quadratic polynomial is used, meaning that a third order accuracy interpolation is obtained in the domain. This is done to guarantee that the immersed boundary method has an error lower than the bulk flow for a significant fine grid and therefore not introducing additional error in the simulation, as will be shown in section 4.1.

For the least-squares interpolation a stencil with a relatively large number of points (six or higher) is required for each *ib face*. To assemble the stencil first the *fluid cell* containing the *ib face* is identified. Then all *fluid cells* that share vertices with that cell are added to the stencil. If any of the *fluid cells* in the stencil contains an *ib face*, then its associated solid point is also added to the stencil. The stencil for a generic *ib face* is shown in Fig. 2.

The least-squares problem, for example for the first component of velocity can be written in matrix notation as:

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & y_1^2 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2^2 & y_2^2 & x_2 y_2 \\ & & & \vdots & & \\ 1 & x_n & y_n & x_n^2 & y_n^2 & x_n y_n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \tag{20}$$

where n is the total number of points in the stencil including both fluid and solid ones. Each matrix line corresponds to one stencil point, where x_i and y_i are coordinates of point i, and u_i and v_i are the velocities values for the mentioned point.

The system can also be written in the form:

$$[\mathbf{M}][B] = [\phi] \tag{21}$$

The least-squares problem is solved by calculating the coefficients of the vector $[B]$:

$$[B] = ([\mathbf{M}]^T [\mathbf{M}])^{-1} [\mathbf{M}]^T [\phi] \tag{22}$$

Once the polynomials' coefficients are known, both velocity components can be computed and applied to the *ib face* as a Dirichlet boundary condition. This process is repeated for each *ib face*. Because the *fluid cell*'s velocity changes each iteration, it is necessary to repeat this step at the beginning of each spatial iteration. However because the geometric matrix $[\mathbf{M}]$ remains the same in each temporal iteration the vector $([\mathbf{M}]^T [\mathbf{M}])^{-1} [\mathbf{M}]$ needs only to be computed once.

3.3. Proposed continuity constrained least squares (CCLS) interpolation method

A new method for velocity interpolation is described in this subsection. It consists of a modification of the previously explained least-squares method. The modifications consist in guaranteeing that the least-squares polynomials verify the continuity equation locally, and additionally each pair of neighboring interpolation polynomials is continuous between each other, which improves global continuity of the IB domain.

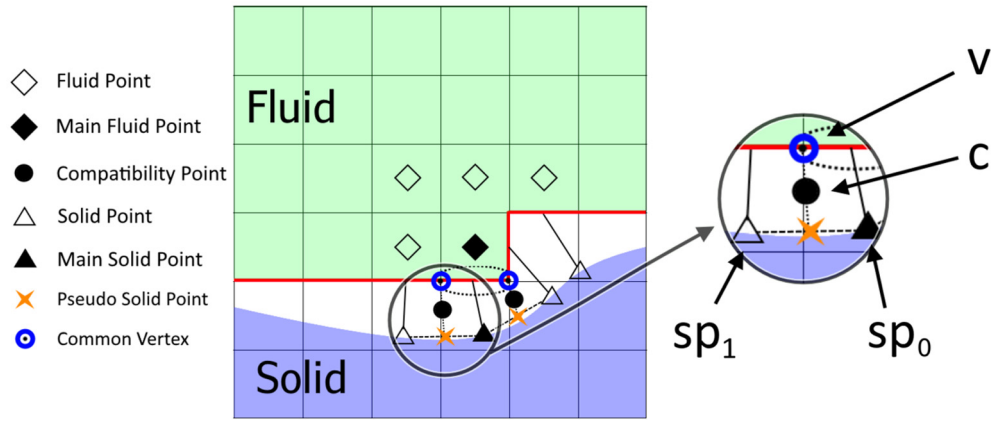

Fig. 2. Diagram of a stencil used in the least-squares interpolation for the circled *ib face*, with details of the compatibility points computation.

To strictly enforce continuity locally the two polynomials for the velocity components u and v are coupled and computed simultaneously in a single least-squares problem. The coefficients of the two polynomials which appear in the divergence equation are replaced by common coefficients, in a way to guarantee that $\frac{\partial u}{\partial x} = -\frac{\partial v}{\partial y}$. To achieve a sharper interface a constraint is applied to the velocity of the main solid point (the solid point associated with the current *ib face*), by fixing the stencil's origin on the *solid point* and setting a_0 and b_0 to the solid point's velocity. This results in two polynomials of the form:

$$(u - u_s) = a_1 x + a_2 y + a_3 x^2 + a_4 y^2 + a_5 xy + a_6 x^2 y \tag{23}$$

$$(v - v_s) = b_1 x - a_1 y + b_3 x^2 - \frac{a_5}{2} y^2 - 2a_3 xy - a_6 xy^2 \tag{24}$$

Notice that two extra terms were added to the polynomials which allows a better overall fit without computational cost since Gauss elimination was already manually applied to take into account both the continuity constraint and the solid point's velocity constraint.

The least-squares matrix is assembled by compiling the polynomials of both velocity components for each point of the stencil, resulting in:

$$\begin{bmatrix} x_1 & y_1 & x_1^2 & y_1^2 & x_1 y_1 & x_1^2 y_1 & 0 & 0 \\ -y_1 & 0 & -2x_1 y_1 & 0 & -y_1^2/2 & -x_1 y_1^2 & x_1 & x_1^2 \\ x_2 & y_2 & x_2^2 & y_2^2 & x_2 y_2 & x_2^2 y_2 & 0 & 0 \\ -y_2 & 0 & -2x_2 y_2 & 0 & -y_2^2/2 & -x_2 y_2^2 & x_2 & x_2^2 \\ & & & & \vdots & & & \\ x_n & y_n & x_n^2 & y_n^2 & x_n y_n & x_n^2 y_n & 0 & 0 \\ -y_n & 0 & -2x_n y_n & 0 & -y_n^2/2 & -x_n y_n^2 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ b_1 \\ b_3 \end{bmatrix} = \begin{bmatrix} u_1 - u_s \\ v_1 - v_s \\ u_2 - u_s \\ v_2 - v_s \\ \vdots \\ u_n - u_s \\ v_n - v_s \end{bmatrix} \tag{25}$$

Note that in this matrix each two consecutive rows correspond to a single stencil point, one row for each velocity component.

Besides enforcing local continuity it is also verified that guaranteeing that the polynomials are piecewise continuous in all the ib domain has a significant effect in suppressing SFOs. To achieve this, a *compatibility point* is generated between each two neighboring *ib faces*. In 2D each *ib face* has always two neighboring *ib faces*, resulting in two *compatibility points* which are added to each least-squares stencil. At these points the two polynomials containing them are forced to interpolate the same value through an iterative method. By making all the polynomials continuous between themselves, because each polynomial is locally divergence free, global continuity in the interpolated domain is improved when applying the compatibility points procedure.

To compute these *compatibility points* firstly the medium point between the two solid points (associated with each of the neighboring *ib faces*) is found. Then the *compatibility point* is found by computing the medium point between this *pseudo solid point* and the common vertex of the two neighboring *ib faces*. Fig. 2 illustrates the complete stencil for a generic *ib face*, including the *compatibility points* and auxiliary geometry to compute them. Each *compatibility point*'s position is computed by:

$$\mathbf{c} = 0.5\mathbf{v} + 0.5 \underbrace{(0.5\mathbf{sp}_0 + 0.5\mathbf{sp}_1)}_{\textbf{pseudo solid point}} \tag{26}$$

where **c** is the *compatibility point*, **v** is the common vertex between the two *ib faces*, and $\mathbf{sp}_0$ and $\mathbf{sp}_1$ are the solid points associated with each *ib face* (see zoom of Fig. 2 for more details).

The velocity at each *compatibility point* is unknown and the interpolation is performed iteratively. In the first iteration the least-squares problem is solved without *compatibility points*. Afterwards the velocity at the *compatibility points* is estimated by averaging the interpolation result from each of the polynomials of the two neighboring *ib faces*. A second iteration is then computed, with the *compatibility points* added to the least-squares problem with a low interpolation weight. This process is repeated, increasing the interpolation weight of the *compatibility points* until the two neighboring polynomials are continuous at these points. In this work the least-squares weight used for compatibility points in each iteration is:

$$w = i_c^2 \tag{27}$$

where i_c is zero in the first iteration (where the *compatibility points* are not used), and increases unitarily in each iteration. It was found that high least-squares weight produces faster compatibility convergence by producing continuous polynomials in the least amount of compatibility iterations and resulting in an overall method that suppresses SFOs. This increasing weight function allows the proposed algorithm to converge in a few compatibility iterations, as will be shown in the results section.

When a cell in one iteration undergoes from being a *non-fluid cell* to being a *fluid-cell*, henceforth called a *fresh cell*, a problem arises because there is no time history from this cell. To solve this problem the previous iteration least-squares polynomials are used to calculate a value of u and v at the *fresh cell*'s centroid at the previous time instant. In the case where a *fresh cell* has more than one *ib face* in the previous iteration, an algebraic average of the value calculated by each *ib face*'s polynomial is used.

A starting value for the pressure at the cell centroid is also required for the PISO algorithm and so a new least squares polynomial is obtained for the pressure field. This interpolation re-uses the same stencil used in the velocity components' interpolation with the exclusion of the *compatibility points*. At *solid points* where the pressure value is unknown, a pressure normal derivative condition is used. From the momentum equation, and neglecting the viscous terms, $\frac{\partial p}{\partial n} = -\rho \frac{Du_n}{Dt}$ [15,16,30], where the material derivative is the solid point's acceleration. The least-squares interpolation can be described by two different equations, one for *fluid points* (eq. (28)) and one for *solid points* (eq. (29)), these are respectively:

$$l_0 + l_1 x + l_2 y + l_3 x^2 + l_4 y^2 + l_5 xy = p \tag{28}$$

$$l_1 n_x + l_2 n_y + 2l_3 x n_x + 2l_4 y n_y + l_5 x n_x y n_y = -\rho \frac{D(u_s n_x + v_s n_y)}{Dt} \tag{29}$$

The least-squares matrix is then assembled with all fluid and solid points' information, and the system is solved. A pressure value at the cell centroid is then interpolated and used as a first estimate in the PISO algorithm.

3.4. Calculation of viscous and pressure force

The calculation of the force acting on solid walls is of crucial relevance for fluid structure interaction. The force is calculated using the discretization of the solid body at the *solid points*, each associated with an *ib face*, where each force component is computed.

For computing the viscous force the velocity spatial derivatives at the discrete solid points' position can be obtained directly from the following set of equations:

$$\frac{\partial u}{\partial x} = a_1 + 2a_3 x + a_5 y + 2a_6 xy \tag{30}$$

$$\frac{\partial u}{\partial y} = a_2 + 2a_4 y + a_5 x + a_6 x^2 \tag{31}$$

$$\frac{\partial v}{\partial x} = b_1 + 2b_3 x - 2a_3 y - a_6 y^2 \tag{32}$$

$$\frac{\partial v}{\partial y} = -a_1 - 2a_3 x - a_5 y - 2a_6 xy \tag{33}$$

where a_i and b_i are the least-squares coefficients of the *solid point*'s *ib face*. Note that since the polynomials are calculated in a solid point centered referential, these terms become trivial to calculate ($x = y = 0$). A similar set of equations can be obtained from the polynomials used in the standard unconstrained method.

The normal derivative of the tangent velocity component can then be calculated with the following equation:

$$\frac{\partial U_t}{\partial n} = [\mathbf{t}]^T \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} [\mathbf{n}] \tag{34}$$

where **t** and **n** are, respectively, the tangent and normal versors at the solid point's location. The viscous force acting on the discretized body surface described by each solid point is calculated simply by:

$$d\mathbf{F_v} = \mu \frac{\partial U_T}{\partial n} S_s \mathbf{t} \tag{35}$$

where S_s is the body surface area associated with the solid point, which is calculated as half of the distance between the two neighboring solid points.

The total viscous force acting on the body is equal to the sum of the viscous force acting on each discrete solid point.

To compute the pressure force, a similar method is employed. The pressure at the solid point is obtained from the least squares pressure polynomial used for providing a starting pressure at fresh-cells (described previously) and the pressure force acting on the discretized body surface described by each solid point is calculated by:

$$d\mathbf{F_p} = (P_s S_s)\mathbf{n} \tag{36}$$

where P_s is the interpolated pressure value at the solid point.

4. Results and discussion

4.1. 2D analytical cavity

The analytical solution of the 2D lid driven cavity test case is used to verify the proposed method's second order spatial convergence. The problem consists of a lid-driven cavity with rigid wall boundary conditions for the right, left and lower boundaries and an imposed velocity at the upper boundary, together with a zero derivative Neumann condition for the pressure correction at all four outer boundaries. This test case was chosen because an analytical solution to the manufactured incompressible Navier–Stokes equations is known [26,31–33]. The velocity prescribed on the upper boundary is:

$$u(x, 1) = 16\zeta_1(x) \tag{37}$$

$$v(x, 1) = 0 \tag{38}$$

A source term β is required in the v momentum equation:

$$\beta(x, y) = \frac{8}{Re}\left[24\int \zeta_1(x)dx + 2\zeta_1'(x)\zeta_2''(y) + \zeta_1'''(x)\zeta_2(y)\right] - 64\left[\Phi_2(x)\Psi(y) - \zeta_2(y)\zeta_2'(y)\Phi_1(x)\right] \tag{39}$$

where $'$ is the differential operator and the functions $\zeta_1(x)$, $\zeta_2(y)$, $\Phi_1(x)$, $\Phi_2(x)$ and $\Psi(y)$ are defined by:

$$\zeta_1(x) = x^4 - 2x^3 + x^2 \tag{40}$$

$$\zeta_2(y) = y^4 - y^2 \tag{41}$$

$$\Phi_1(x) = \zeta_1(x)\zeta_1''(x) - \zeta_1'(x)\zeta_1'(x) \tag{42}$$

$$\Phi_2(x) = \int \zeta_1(x)\zeta_1'(x)dx \tag{43}$$

$$\Psi(y) = \zeta_2(y)\zeta_2'''(y) - \zeta_2'(y)\zeta_2''(y) \tag{44}$$

The domain consists of a 1×1 m^2 box and the chosen Reynolds number was 1 in order to give an equal weight to both convective and diffusive terms.

The analytical solution to this problem is:

$$u(x, y) = 8\zeta_1(x)\zeta_2'(y) \tag{45}$$

$$v(x, y) = -8\zeta_1'(x)\zeta_2(y) \tag{46}$$

Four Cartesian grids of uniformly distributed grid-points, with 20×20, 40×40, 80×80 and 160×160 cells are used. A cylinder of radius 0.2 is introduced in the center of the domain (see Fig. 4) and it is discretized using the proposed immersed-boundary method. In this cylinder's *solid points* the analytical values for u and v are imposed, in the same way they would be imposed for a moving body. In the center of the cylinder the fluid equations are not solved.

The errors, ε, are evaluated by subtracting the computational result from the analytical solution. The maximum cell error, $||\varepsilon||_\infty$, as well as the $||\varepsilon||_2$ error for the two velocity components are shown in Fig. 3.

The results in Fig. 3 demonstrate a second order spatial convergence for the developed method for both $||\varepsilon||_2$ and $||\varepsilon||_\infty$ errors. This is the same convergence rate of the PISO algorithm used for the bulk flow. Furthermore it can be seen in Fig. 4 that the *immersed boundary* isn't responsible for the highest local errors because the maximum error appear detached from the immersed boundary. This comes from the immersed boundary interpolation being third order accurate and so, for a sufficient fine grid the overall error will be limited to second order caused by the bulk solver of the fluid domain.

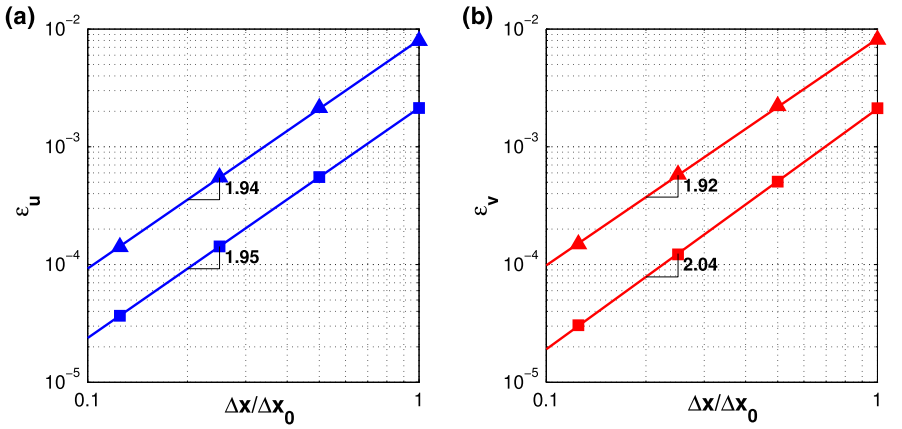

Fig. 3. Spatial convergence of the present method, for errors in velocity component u (a) and v (b) ($\square$ – $||\varepsilon||_2$ error, $\triangle$ – $||\varepsilon||_\infty$ error).

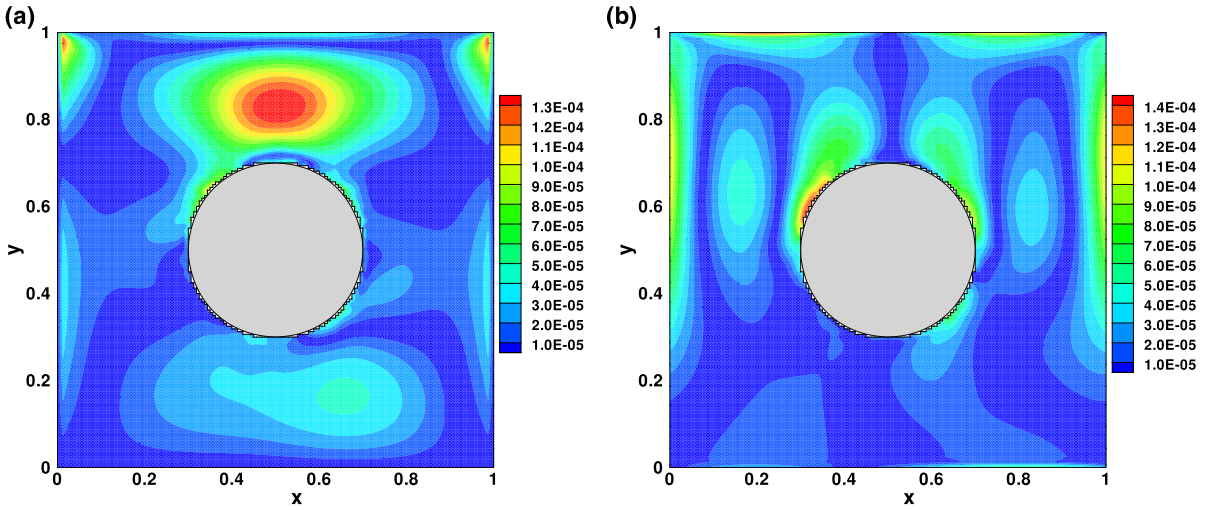

Fig. 4. Local error of the two velocity components for the 160×160 grid (u (a) and v (b)).

4.2. Oscillating cylinder in static flow

After the verification of the method for a steady-state problem, a moving body problem is simulated to demonstrate the method's accuracy. The problem chosen was the flow around an oscillating cylinder in static flow that was studied extensively by Dutsch et al. [34] and used in immersed boundary contexts by many other authors [14,17].

The cylinder oscillates in static fluid, with the position over time defined as:

$$x(t) = \frac{A_m}{2} sin(2\pi f t) \tag{47}$$

$$y(t) = 0 \tag{48}$$

The Reynolds number based on the maximum translational velocity is 100 and the Keulegan–Carpenter number 5. These values are the same used in one of the test cases in [34], providing comparative results. The values of A_m and f are chosen to achieve a Strouhal number equal to the one used in the original article:

$$A_m = 1.6D \tag{49}$$

$$f = 0.05 \text{ Hz} \tag{50}$$

The domain consists of a rectangular box of $10D \times 10D$ with a 400×400 non-uniformly distributed cells, with the cylinder oscillating around the domain center. The finest mesh interval, $\Delta x = D/60$, corresponds to the solid-body being discretized by roughly 240 *ib faces*. Pressure outlet boundary conditions are used at all four boundaries. The time step used was $\Delta t = T/720$ (where $T = 1/f$ is the movement period), which results in a maximum CFL number of 0.42. The CFL number is based on the solid body velocity and the grid spacing.

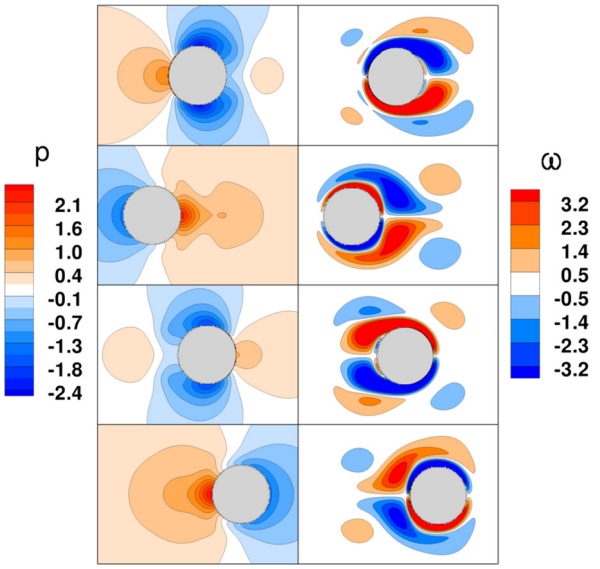

Fig. 5. Pressure and vorticity (left – pressure, right – vorticity) nondimensionalized contour plots at four different movement phases (0°, 96°, 192°, 288°).

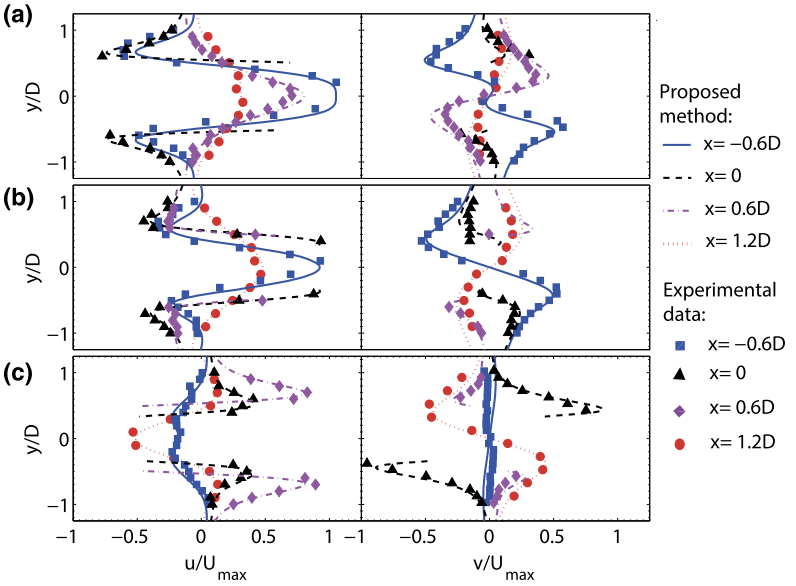

Fig. 6. Comparison with experimental data from [34] of the velocity components at cross-sections with constant x values at different phases of the movement ((a) 180°, (b) 210°, (c) 330°).

$$CFL = ||\mathbf{u}_{solid}|| \frac{\Delta t}{\Delta x} \tag{51}$$

Fig. 5 shows the pressure and vorticity contours at four different movement phases. These contours are in excellent agreement with [34], notably in the pressure field which validates the proposed pressure calculation method in moving body conditions, overcoming the standard unconstrained method's limitations.

Fig. 6 shows a comparison of the velocity fields computed with the proposed method and experimental data, and Fig. 7 shows the comparison with computational results that were not obtained with an IB method, but with a body-fitted moving mesh [34]. The IB computations are in good agreement with the experiments and excellent agreement with the reference's computational results from [34]. This lends credit to the argument that the observed small differences in Fig. 6 between prediction and experimental data are from experimental data uncertainties.

The calculated forces acting on the body, shown in Fig. 8(a), are also in excellent agreement with the body-fitted results [34]. Fig. 8(b) shows the results obtained with both the standard unconstrained method and the improved CCLS IB method for the same grid and time step. The oscillations present in the results using the standard method are suppressed with the new CCLS proposed method.

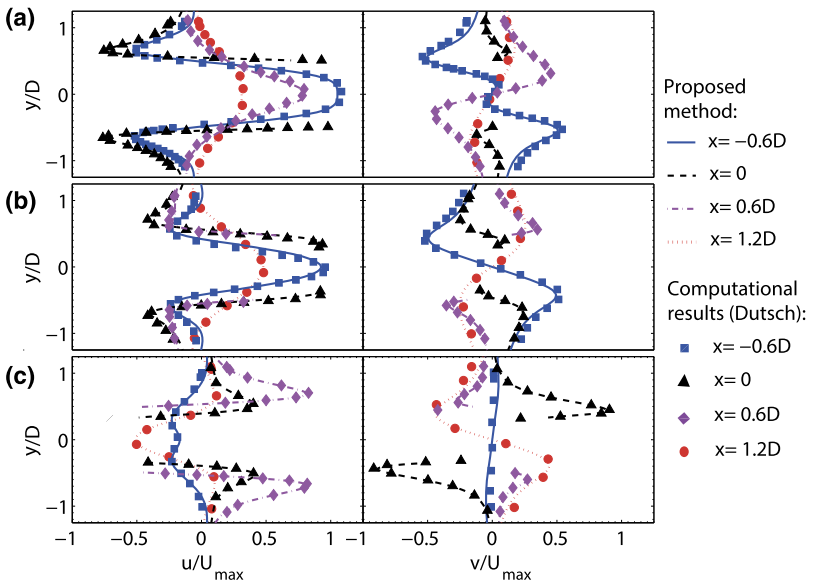

Fig. 7. Comparison with computational results from [34] of the velocity components at cross-sections with constant x values at different phases of the movement ((a) 180°, (b) 210°, (c) 330°).

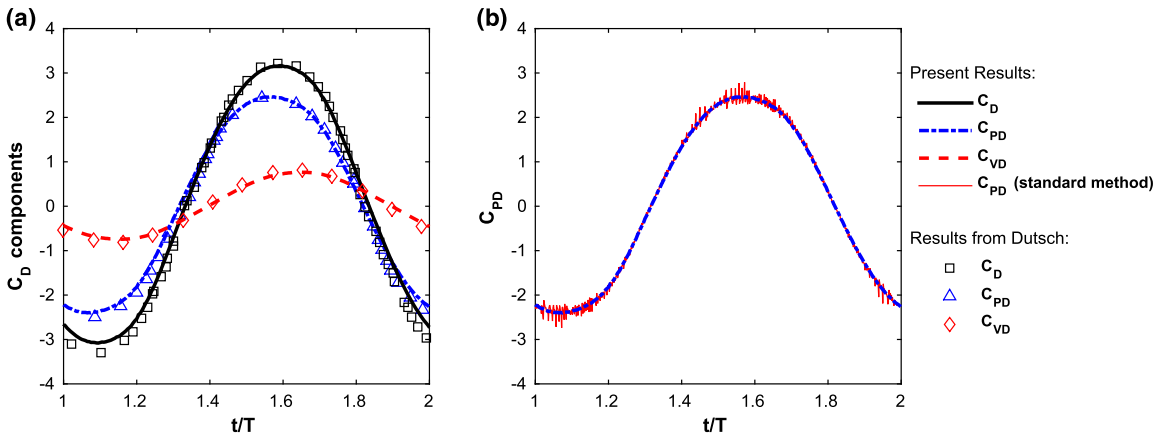

Fig. 8. Time history of adimensionalized force components (a) and comparison in pressure force obtained from standard and CCLS methods (b).

4.3. SFO study in an oscillating cylinder

To compare the proposed method with other SFO suppression related studies, a similar test case is selected to the one in the previous section of an oscillating cylinder in static fluid. The cylinder oscillates in a domain of 4D × 4D, with pressure outlet boundary conditions on the left and right outer boundaries and fixed wall boundary conditions in the upper and lower boundaries. The cylinder equation constants were chosen in accordance to [13], and are:

$$A_m = \frac{D}{4} \tag{52}$$

$$f = 0.995 \text{ Hz} \tag{53}$$

The Reynolds number in relation to the maximum translational velocity is equal to 78.5. Four Cartesian grids 64 × 64, 128 × 128, 192 × 192 and 256 × 256 were considered and four time steps were used: $\Delta t = 0.002T, 0.004T, 0.008T$ and $0.016T$. These grids and time steps are consistent with the ones used in [13], providing a way to compare the proposed method with an already established SFO suppression method.

Fig. 9 shows the pressure force over time for both the proposed and the standard unconstrained method. The proposed method provides reasonably smooth results for the different grids and time steps, showing the SFOs have been eliminated. The standard unconstrained method shows strong oscillations for all tested grids and time steps. It is noticeable for both methods a significant decrease in SFOs with grid refinement and a slight decrease in SFOs with increasing time step. The observed behavior is in agreement with other authors [13,14].

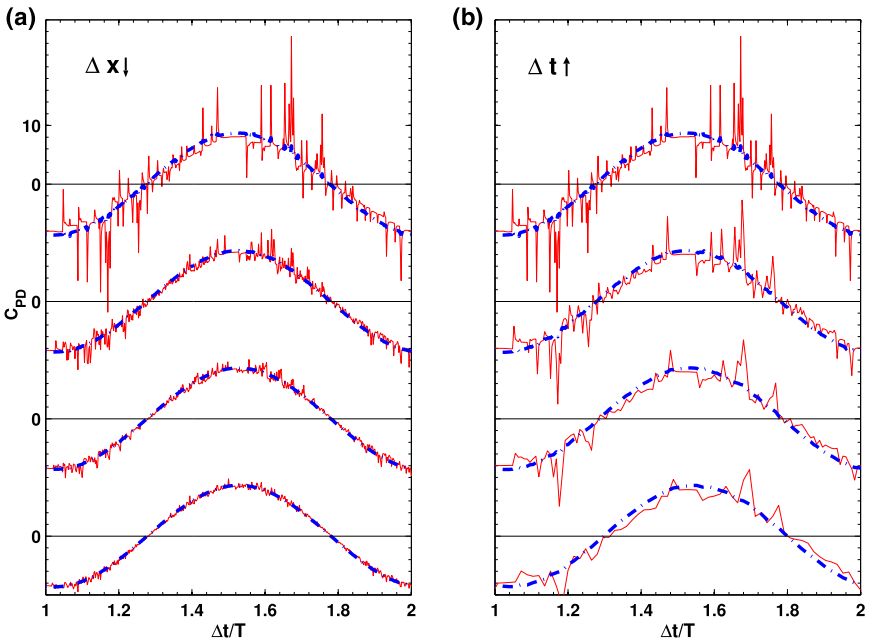

Fig. 9. Time history of C_{PD} for four grids with constant time step (a) and four different time steps with constant grid (b) (— standard method, – · – proposed method).

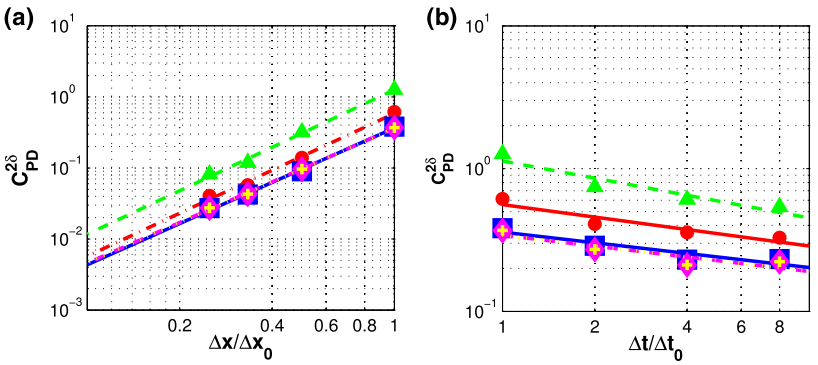

Fig. 10. Variation of $C^{2\delta}_{PD_{rms}}$ for different number of compatibilization iterations for four grids with constant time step (a), and four different time steps with constant grid (b) (number of compatibilization iterations: △ – 1, ● – 3, □ – 5, ◇ – 10, + – 20).

To measure the decrease in oscillations when varying Δx and Δt the metric proposed by Seo et al. [13] is considered. $C^{2\delta}_{PD}$ is defined in the following way:

$$C^{2\delta}_{PD} = |C^{n+1}_{PD} - 2C^{n}_{PD} + C^{n-1}_{PD}| \tag{54}$$

The root mean square of this function is subsequently used to provide a comparative value on the magnitude of the oscillations in the force calculation. These results can be plotted in function of Δx, Δt and CFL number, providing measurements on the suppression of the observed SFOs.

Because the proposed method requires compatibilization iterations, a stopping criteria is required. Fig. 10 shows the oscillations' metric plotted for different numbers of compatibilization iterations. The study on the effect of the number of compatibilization iterations showed that there is a limit after which no more benefits are obtained. Notice that the case of only one compatibilization iteration corresponds to a method where no compatibility of neighboring polynomials is performed, only the continuity and solid point velocity constraints are applied to the least-squares polynomials. This simple modification over the standard method produces significant suppression of oscillations, showing a decrease in the oscillations by a factor of four for the coarse grid and smallest time step.

A similar study was performed on the effects of compatibility iterations together when using the unconstrained interpolation method. The results showed that the produced oscillations were not suppressed independently of the number of compatibilization iterations. The reason for the compatibilization capability to suppress oscillations is because it improves global continuity in the reconstructed domain, however that only happens when each of the interpolated polynomials is

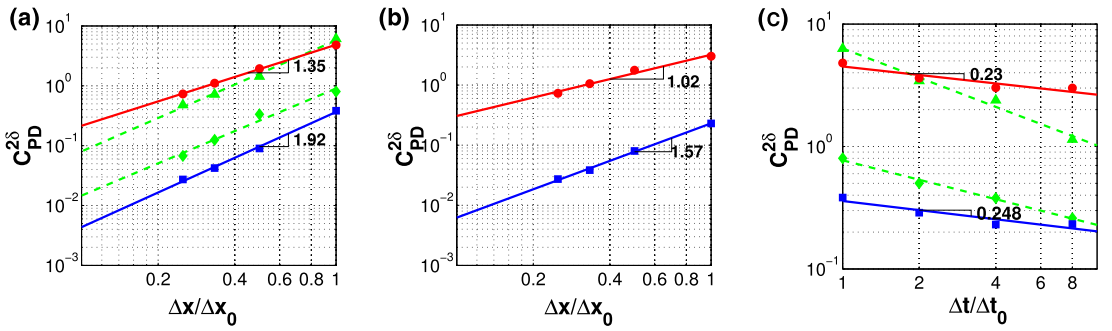

Fig. 11. Variation of $C^{2\delta}_{PD_{rms}}$ for four grids with constant time step (a), four grids with constant CFL number (b) and four different time steps with constant grid (c) (• – standard method, ■ – proposed CCLS method, △ – Seo standard method, ◇ – Seo proposed method).

already divergence free. Because in the unconstrained method the polynomials are not, the compatibility algorithm alone has a neglectable effect on the spurious force oscillations.

Fig. 10 shows also that five iterations are sufficient to achieve the desired effect. Further compatibilization iterations provide no apparent benefit, indicating that the interpolated polynomials are continuous between each other by the fifth iteration. When five (or more) compatibilization iterations are performed the SFOs present correspond roughly to the continuity constrained interpolation method with no compatibilization for a grid with Δx halved. This correspondence validates the iterations' effectiveness.

Fig. 11 shows a comparison between the proposed and the standard method, together with the results of the original and proposed methods in [13]. The SFO suppression provided by the proposed CCLS method is higher than the one obtained by Seo in [13]. With the proposed method the SFOs decrease faster with grid refinement than with the standard method and show roughly a second order slope in agreement with many authors' observations [13–15]. However the SFOs first order trend with increasing time step is not found. This was also the case in [13], which reports the SFOs decrease with $\Delta t^{0.5}$, instead of an unexplained linear (Δt^1) behavior reported by other authors. Fig. 11(b) shows for the present calculations that keeping constant the CFL number SFOs decrease with a slope $\Delta x^{1.57}$. These results show that the proposed method is less sensitive to the time step used, and is capable of achieving smooth results faster when grid refinements are performed with CFL constraints in comparison to other SFO suppression methods.

4.4. 2D flapping wing

The flow over a flapping wing is simulated to test the robustness of the proposed method. This flow has been extensively studied, see e.g. Wang [35], Kim and Choi [36], Eldredge [37] and Albuquerque [38], and has also been commonly used in SFO investigations in IB methods [10,14,15,17]. This test case consists of a 2D elliptical wing, translating in a sinusoidal movement on a plane with inclination θ_p relative to the horizontal axis, while simultaneously rotating around it's center. For the two variations of this movement presented in this section, the wing's position and pitch angle are given by:

$$x(t) = \frac{A_m}{2} cos(2\pi f t) \cdot cos(\theta_p) \tag{55}$$

$$y(t) = \frac{A_m}{2} cos(2\pi f t) \cdot sin(\theta_p) \tag{56}$$

$$\theta(t) = \frac{\pi}{4} sin(2\pi f t) + \theta_0 \tag{57}$$

The first variation of this movement corresponds to the one studied by Wang in [35], Kim and Choi in [36] and Albuquerque in [38]. In this variation the movement constants are: $\theta_p = 60°$, $A_m = 2.5c$, $f = 40$ Hz and $\theta_0 = -\pi/4$. The wing consists of an ellipse with a thickness ratio of 8 ($c/h = 8$). The Reynolds number, which is based on the maximum translational velocity, is equal to 157.

This test case was simulated on a grid of $20c \times 20c$, with 400×400 non-uniformly distributed grid points. In the finest region, which encompasses the whole rigid-body movement area, the grid spacing is $\Delta x = c/80$. A time step of $\Delta t = T/800$ is used, which results in a CFL number of 0.79 based on the maximum translational velocity. In all outer boundaries pressure outlet boundary conditions are used.

Fig. 12 shows the comparison between the forces obtained with the proposed method and the results from [36], that uses an IB method but in a non-inertial reference frame fixed to the body and consequently does not have the SFOs problem, and the results from [38] and [35]. The present calculations show a good agreement with those works.

The vorticity contours at different time instants, obtained with the new CCLS method and shown in Fig. 13, are virtually identical to those represented by [36] and [35].

Another flow case was simulated corresponding to the flapping wing movement constants $\theta_p = 0°$, $A_m = 2.8c$, $f = 0.25$ Hz and $\theta_0 = \pi/2$, for a Reynolds number of 75 and a wing of elliptical cross-section with a thickness ratio of 10. This

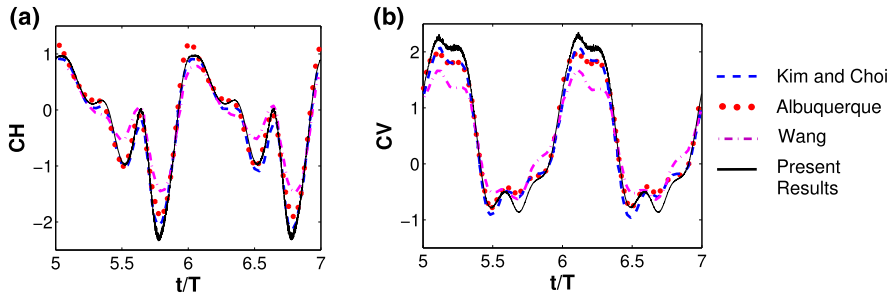

Fig. 12. Horizontal and vertical force coefficients of the last two periods before periodicity is achieved (- - Kim and Choi [36], · · · Albuquerque [38], -·- Wang [35], — proposed method).

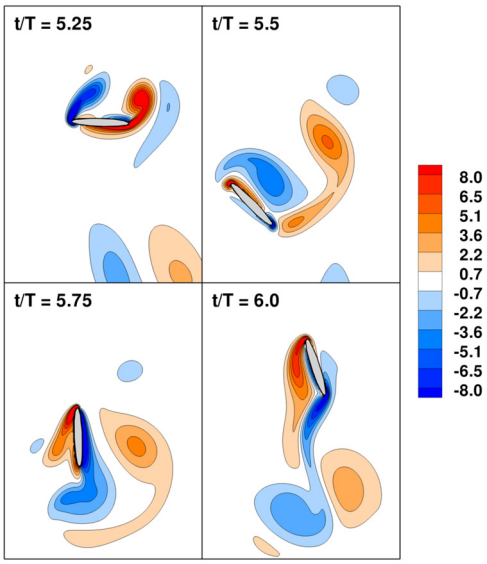

Fig. 13. Nondimensional vorticity contours for different instants for the first variation of the flapping wing movement.

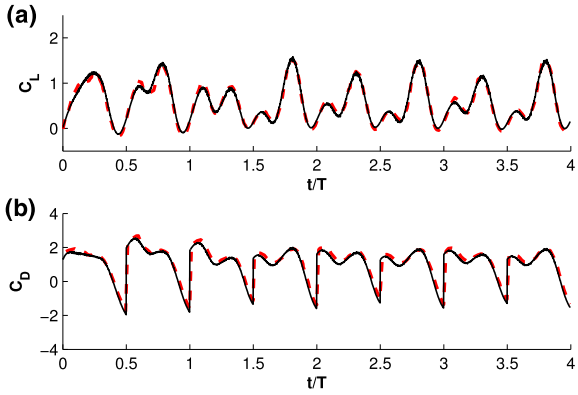

Fig. 14. Lift (a) and drag (b) force coefficients' time history for the horizontal movement case (— proposed method, - - Eldredge [37]).

test case corresponds to the one studied by Eldredge in [37]. Once again a domain of $20c \times 20c$ with pressure outlet in all outer boundaries is used. The grid contains 400×400 non-uniformly distributed cells, with $\Delta x = c/160$ in the finest region. The time step is $dt = T/800$, which results in a maximum CFL number of 0.88. As was proposed by Eldredge, C_D is defined as the horizontal force always opposing the movement direction:

$$C_D = sign(\dot{x}(t)) \frac{F_D}{1/2 \rho U_{max}^2 c} \tag{58}$$

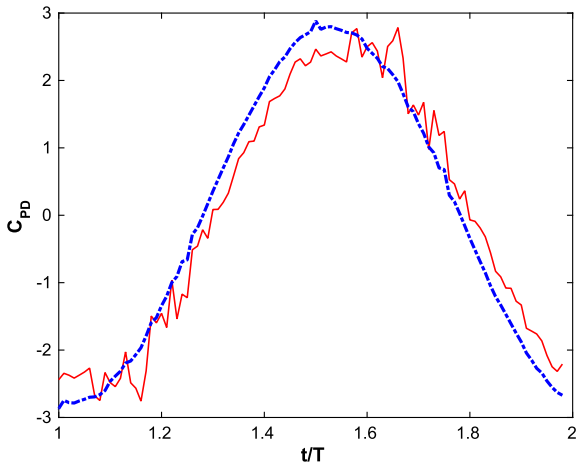

Fig. 15. Drag coefficient acting on the sphere body over time (— standard method, — · — proposed method).

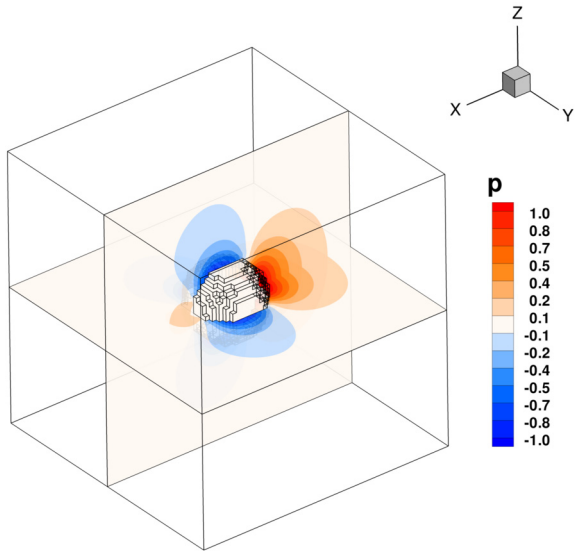

Fig. 16. Pressure contours in y and z slices at $t = 1.3T$, using the proposed CCLS method.

The results obtained, as well as those in [37] are shown in Fig. 14. An excellent agreement is seen in both force components with the literature results. It shows also that the proposed CCLS method provides very smooth results of the force over time, unlike the standard unconstrained method.

4.5. Oscillating sphere

To demonstrate the proposed method's extension to 3D, the flow over an oscillating sphere is simulated, as seen in [13]. The required modifications to the proposed CCLS method are explained in Appendix A. A computational domain of $4D^3$ is used and the boundary conditions are static walls for the four boundaries parallel to the X axis and pressure outlet is imposed for the other two. The sphere oscillates according to equation (47) with $A_m = 0.25D$. The Reynolds number is $Re = 78.54$ and a frequency of $f = 1$ Hz was chosen. The problem is solved in a grid of 64^3 cells with a time step $\Delta t = T/100$, resulting in a CFL number of 0.125, with both the standard method and the proposed CCLS method.

Fig. 15 shows the drag coefficient over time using the unconstrained and the CCLS method. It is immediately apparent the suppression of SFOs obtained with the proposed CCLS method, that is capable of computing a stable pressure force, unlike the original method. The suppression of SFOs achieved with this new method is therefore demonstrated to be easily extended to 3D which increases its applicability and usefulness.

Fig. 16 shows pressure contours, in slices perpendicular to the y and z axis. This figure shows also the conservative cut around the sphere and it is noticeable that the grid is very coarse and the conservative cut is crude, still the CCLS method is capable of achieving good results when it comes to SFO suppression due to the use of solid points located at the sphere's surface which are not represented in the figure.

5. Conclusions

A new discrete forcing, flow reconstruction immersed boundary method, in a finite volume Cartesian grid framework was presented. This new technique enforces local continuity through the use of a continuity constraint in the interpolation least-squares polynomials, as well as a polynomial compatibility through an iterative algorithm. A constraint for the solid body velocity is also used, which has been showed to provide benefits.

The proposed CCLS method was then verified and validated with several test cases, including steady-state and moving body problems. The method was first shown to have the same convergence rate as the bulk flow solver, and was then validated using both experimental data and literature computational results from body-fitted methods. The SFO suppression properties of the new proposed method were demonstrated in all simulated moving body test cases. The new method was demonstrated to improve IB methods by eliminating the problem of pressure oscillations, including bodies with rotation, situations featuring significant vortex structure interaction and even a three dimensional case where its effectiveness has been showed.

The new interpolation technique was studied for a specific discrete forcing IB method, however this technique is applicable to other IB methods which make use of least-squares interpolation technique.

Acknowledgements

This work was supported by FCT, through IDMEC, under LAETA, project UID/EMS/50022/2013.

Appendix A. 3D methodology

The proposed method is very easily extended to three dimensions (3D) with only minor modifications. Firstly, the conservative cut is performed identically to the process described in section 3.1. The cells with no vertices inside the solid body are categorized as *fluid cells*, all non-*fluid cells* that are face-neighbors with a *fluid cell* are categorized as *ib cells* with the respective face being a *ib face* where Dirichlet conditions of velocity are applied. The fluid equations are not solved in the remaining cells, the *solid cells*.

The stencil of the least-squares problem for each *ib face* is assembled similarly to the 2D algorithm. The main *fluid cell* containing the respective *ib face* and all *fluid cells* sharing vertices with this cell are added to the stencil. If any of the stencil's cells has an *ib face*, then the respective *solid point* is also added to the stencil. The compatibility points are then created between the main *ib face* and any *ib face* that shares an edge with it. Notice that unlike the 2D case where the compatibility points were always two, in 3D the number of compatibility points is equal to the number of edges in the face, which in a Cartesian grid results in four compatibility points. The compatibility point's position is computed from an equation similar to eq. (26), where instead of just using the unique vertex of both faces, the mean position of the two vertices belonging to the common edge must be used:

$$\mathbf{c} = 0.5\underbrace{(0.5\mathbf{v}_0 + 0.5\mathbf{v}_1)}_{\textbf{mean vertex}} + 0.5\underbrace{(0.5\mathbf{sp}_0 + 0.5\mathbf{sp}_1)}_{\textbf{pseudo solid point}} \tag{59}$$

where $\mathbf{c}$ is the compatibility point's position, $\mathbf{v}_0$ and $\mathbf{v}_1$ are the two common vertices shared between both *ib faces*, and $\mathbf{sp}_0$ and $\mathbf{sp}_1$ the solid points of each *ib face*. The iterative compatibility algorithm is identical to the 2D case.

The form of the polynomial for the least-squares problem is a quadratic 3D one, with the main *solid point* velocity and continuity constraints applied to it, these results in the following polynomials for each velocity component:

$$u - u_s = a_1x + a_2y + a_3z + a_4x^2 + a_5xy + a_6xz + a_7y^2 + a_8yz + a_9z^2 \tag{60}$$

$$v - v_s = b_1x + b_2y + b_3z + b_4x^2 + b_5xy + b_6xz + b_7y^2 + b_8yz + b_9z^2 \tag{61}$$

$$\begin{aligned} w - w_s = {} & c_1x + c_2y + (-a_1 - b_2)z + c_4x^2 + c_5xy + (-2a_4 - b_5)xz + \\ & + c_7y^2 + (-a_5 - 2b_7)yz + (-a_6/2 - b_8/2)z^2 \end{aligned} \tag{62}$$

where x_n, y_n, z_n, u_n, v_n and w_n are the coordinates and velocity component values for each of the stencil's points and u_s, v_s and w_s the velocity components of the main *solid point* of the stencil.

As in the 2D case all the velocity components are solved simultaneously in a single coupled least-squares problem which can be shown in following matrix form:

$$
\begin{bmatrix}
x_1 & y_1 & z_1 & x_1^2 & x_1y_1 & x_1z_1 & y_1^2 & y_1z_1 & z_1^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & x_1^2 & x_1y_1 & x_1z_1 & y_1^2 & y_1z_1 & z_1^2 & 0 & 0 & 0 & 0 \\
-z_1 & 0 & 0 & -2x_1z_1 & -y_1z_1 & -z_1^2/2 & 0 & 0 & 0 & 0 & -z_1 & 0 & 0 & -x_1z_1 & 0 & -2y_1z_1 & -z_1^2/2 & 0 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 \\
x_2 & y_2 & z_2 & x_2^2 & x_2y_2 & x_2z_2 & y_2^2 & y_2z_2 & z_2^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & x_2^2 & x_2y_2 & x_2z_2 & y_2^2 & y_2z_2 & z_2^2 & 0 & 0 & 0 & 0 \\
-z_2 & 0 & 0 & -2x_2z_2 & -y_2z_2 & -z_2^2/2 & 0 & 0 & 0 & 0 & -z_2 & 0 & 0 & -x_2z_2 & 0 & -2y_2z_2 & -z_2^2/2 & 0 & x_2 & y_2 & x_2^2 & x_2y_2 & y_2^2 \\
 & & & & & & & & & & & \vdots & & & & & & & & & & \\
x_n & y_n & z_n & x_n^2 & x_ny_n & x_nz_n & y_n^2 & y_nz_n & z_n^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_n & y_n & z_n & x_n^2 & x_ny_n & x_nz_n & y_n^2 & y_nz_n & z_n^2 & 0 & 0 & 0 & 0 \\
-z_n & 0 & 0 & -2x_nz_n & -y_nz_n & -z_n^2/2 & 0 & 0 & 0 & 0 & -z_n & 0 & 0 & -x_nz_n & 0 & -2y_nz_n & -z_n^2/2 & 0 & x_n & y_n & x_n^2 & x_ny_n & y_n^2
\end{bmatrix}
\begin{bmatrix}
a_1 \\ \vdots \\ a_9 \\ b_1 \\ \vdots \\ b_9 \\ c_2 \\ c_3 \\ c_7 \\ c_8 \\ c_9
\end{bmatrix}
=
$$

$$
=
\begin{bmatrix}
u_1 - u_s \\ v_1 - v_s \\ w_1 - w_s \\ u_2 - u_s \\ v_2 - v_s \\ w_2 - w_s \\ \vdots \\ u_n - u_s \\ v_n - v_s \\ w_n - w_s
\end{bmatrix}
\tag{63}
$$

This matrix results from the assembling of previous quadratic polynomials. Since the matrix is solved implicitly there is no bias behavior towards the velocity component that has a modified polynomial (w velocity component), which is evident from the symmetry between the y and z directions at the pressure field in Fig. 16.

References

[1] Rajat Mittal, Gianluca Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[2] Charles S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (2) (1972) 252–271.
[3] Charles S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (3) (1977) 220–252.
[4] Charles S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[5] Ming-Chih Lai, Charles S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comput. Phys. 160 (2) (2000) 705–719.
[6] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, J. Comput. Phys. 105 (2) (1993) 354–366.
[7] Zhilin Li, Ming-Chih Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2) (2001) 822–842.
[8] Felipe Pamplona Mariano, L. de Q. Moreira, A. de Silveira-Neto, Carlos Bettencourt da Silva, José C.F. Pereira, A new incompressible Navier–Stokes solver combining Fourier pseudo-spectral and immersed boundary methods, Comput. Model. Eng. Sci. 59 (2) (2010) 181–216.
[9] Paulo J.S.A. Ferreira de Sousa, José C.F. Pereira, James J. Allen, Two-dimensional compact finite difference immersed boundary method, Int. J. Numer. Methods Fluids 65 (6) (2011) 609–624.
[10] Jinmo Lee, Donghyun You, An implicit ghost-cell immersed boundary method for simulations of moving body problems with control of spurious force oscillations, J. Comput. Phys. 233 (2013) 295–314.
[11] Markus Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2) (2005) 448–476.
[12] Jianming Yang, Elias Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, J. Comput. Phys. 215 (1) (2006) 12–40.
[13] Jung Hee Seo, Rajat Mittal, A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations, J. Comput. Phys. 230 (19) (2011) 7347–7363.
[14] Jongho Lee, Jungwoo Kim, Haecheon Choi, Kyung-Soo Yang, Sources of spurious force oscillations from an immersed boundary method for moving-body problems, J. Comput. Phys. 230 (7) (2011) 2677–2695.
[15] Haoxiang Luo, Hu Dai, Paulo J.S.A. Ferreira de Sousa, Bo Yin, On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries, Comput. Fluids 56 (2012) 61–76.
[16] Matthias Meinke, Lennart Schneiders, Claudia Günther, Wolfgang Schröder, A cut-cell method for sharp moving boundaries in cartesian grids, Comput. Fluids 85 (2013) 135–142.
[17] Y. Zhang, C.H. Zhou, J.Q. Ai, Improvement of mass conservation and reduction of spurious oscillations for the local DFD method, Int. J. Comput. Fluid Dyn. 29 (3–5) (2015) 257–271.
[18] Xiaolei Yang, Xing Zhang, Zhilin Li, Guo-Wei He, A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations, J. Comput. Phys. 228 (20) (2009) 7821–7836.
[19] Jungwoo Kim, Dongjoo Kim, Haecheon Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, J. Comput. Phys. 171 (1) (2001) 132–150.
[20] Marcos Vanella, Elias Balaras, A moving-least-squares reconstruction for embedded-boundary formulations, J. Comput. Phys. 228 (18) (2009) 6617–6628.
[21] Marcos Vanella, Patrick Rabenold, Elias Balaras, A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid structure interaction problems, J. Comput. Phys. 229 (18) (2010) 6427–6449.

[22] Lin Sun, Sanjay R. Mathur, Jayathi Y. Murthy, An unstructured finite-volume method for incompressible flows with complex immersed boundaries, Numer. Heat Transf., Part B, Fundam. 58 (4) (2010) 217–241.
[23] B. Yildirim, Sun Lin, Sanjay Mathur, Jayathi Y. Murthy, A parallel implementation of fluid–solid interaction solver using an immersed boundary method, Comput. Fluids 86 (2013) 251–274.
[24] Stephan Schwarz, Tobias Kempe, Jochen Fröhlich, An immersed boundary method for the simulation of bubbles with varying shape, J. Comput. Phys. 315 (2016) 124–149.
[25] João P.P. Magalhães, Duarte M.S. Albuquerque, José M.C. Pereira, José C.F. Pereira, Adaptive mesh finite-volume calculation of 2d lid-cavity corner vortices, J. Comput. Phys. 243 (2013) 365–381.
[26] Duarte M.S. Albuquerque, José M.C. Pereira, José C.F. Pereira, Residual least-squares error estimate for unstructured h-adaptive meshes, Numer. Heat Transf., Part B, Fundam. 67 (3) (2015) 187–210.
[27] Raad I. Issa, Solution of the implicitly discretised fluid flow equations by operator-splitting, J. Comput. Phys. 62 (1) (1986) 40–65.
[28] Suhas V. Patankar, D. Brian Spalding, A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, Int. J. Heat Mass Transf. 15 (10) (1972) 1787–1806.
[29] C.M. Rhie, W.L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, AIAA J. 21 (11) (1983) 1525–1532.
[30] Manish Kumar, Somnath Roy, M. Sujaat Ali, An efficient immersed boundary algorithm for simulation of flows in curved and moving geometries, Comput. Fluids 129 (2016) 159–178.
[31] T. Shih, C. Tan, B. Hwang, Effects of grid staggering on numerical schemes, Int. J. Numer. Methods Fluids 9 (2) (1989) 193–212.
[32] M. Kobayashi, José M.C. Pereira, José C.F. Pereira, A conservative Finite-Volume second-order accurate projection method on hybrid unstructured grids, J. Comput. Phys. 150 (1999) 40–75.
[33] José M.C. Pereira, M. Kobayashi, José C.F. Pereira, A fourth-order-accurate finite volume compact method for the incompressible Navier–Stokes solutions, J. Comput. Phys. 167 (2001) 217–243.
[34] H. Dütsch, F. Durst, S. Becker, H. Lienhart, Low-Reynolds-number flow around an oscillating circular cylinder at low Keulegan–Carpenter numbers, J. Fluid Mech. 360 (1998) 249–271.
[35] Z. Jane Wang, Two dimensional mechanism for insect hovering, Phys. Rev. Lett. 85 (10) (2000) 2216.
[36] Dokyun Kim, Haecheon Choi, Two-dimensional mechanism of hovering flight by single flapping wing, J. Mech. Sci. Technol. 21 (1) (2007) 207–221.
[37] Jeff D. Eldredge, Numerical simulation of the fluid dynamics of 2d rigid body motion with the vortex particle method, J. Comput. Phys. 221 (2) (2007) 626–648.
[38] Duarte M.S. Albuquerque, José M.C. Pereira, José C.F. Pereira, Calculation of a deformable membrane airfoil in hovering flight, Comput. Model. Eng. Sci. 72 (4) (2011) 337–366.