Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 373 (2021) 113498
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A semi-implicit direct forcing immersed boundary method for periodically moving immersed bodies: A Schur complement approach

Rafi Sela[a], Efi Zemach[b], Yuri Feldman[a,*]

[a] Department of Mechanical Engineering, Ben-Gurion University of the Negev, P.O. Box 653, Beer-Sheva 84105, Israel
[b] Department of Mechanical Engineering, Shamoon College of Engineering, Beer-Sheva 84100, Israel

Received 22 February 2019; received in revised form 5 October 2020; accepted 8 October 2020
Available online xxxx

Abstract

An extended immersed boundary methodology utilizing a semi-implicit direct forcing approach was formulated for the simulation of incompressible flows in the presence of periodically moving immersed bodies. The methodology utilizes a Schur complement approach to enforce no-slip kinematic constraints for immersed surfaces. The methodology is split into an "embarrassingly" parallel pre-computing stage and a time integration stage, both of which take advantage of the general parallel file system (GPFS) for efficient writing and reading of large amounts of data. The methodology can be embedded straight forwardly into the whole family of pressure–velocity segregated solvers of incompressible Navier–Stokes equations based on projection or fractional step approaches. The methodology accurately meets the no-slip kinematic constraints on the surfaces of immersed oscillating bodies. In this study, it was extensively verified by applying it for the simulation of a number of representative flows developing in the presence of an oscillating sphere. The capabilities of the methodology for the simulation of incompressible flow generated by a number of bodies whose motion is governed by general periodic kinematics were demonstrated by simulation of the flow developing in the presence of two out-of-phase oscillating spheres. The physical characteristics of the generated flows in terms of the time evolutions of the total drag coefficients were presented as a function of Reynolds values. The vortical structures inherent in the generated flows were visualized by presenting the isosurfaces of the λ_2 criterion.

Keywords: Immersed boundary method; Periodically moving bodies; Schur complement

1. Introduction

The periodic movement of bodies immersed in an incompressible Newtonian fluid is ubiquitous in various biological systems and in many engineering and industrial applications. Indeed, undulatory propulsion (i.e., the sequential periodic activation of different segments of a body) forms the basis of the locomotion of many elongated aquatic animals (see the extensive review of [1] for details) and also of other cellular organisms [2–5]. In engineering, the periodic external actuation of small rigid bodies is a commonly used mechanism for providing locomotion of

* Corresponding author.
E-mail address: yurifeld@bgu.ac.il (Y. Feldman).

https://doi.org/10.1016/j.cma.2020.113498

rigid synthetic micro- and nano-swimmers [6–8]. In this type of application, propulsion is typically provided by the symmetry-breaking excitation of the surrounding flow by the periodically moving body of the swimmer. Importantly, the periodic movement of externally activated immersed bodies in the form of rotating or oscillating stir-bars of various shapes is the basic mechanism driving the micro- and macromixers [9–12] that are widely used in the chemical and medical industries.

To simulate the fluid flow developing in the presence of periodically moving immersed bodies, it is necessary to perform moving boundary simulations. The idea underlying these simulations is to determine the kinematics of the immersed body and then to impose no-slip boundary conditions on the body surface at each computational time step. The major challenge in performing such simulations by utilizing a traditional body-conformal grid approach lies in the generation of a high-quality mesh that should, first, be capable of handling complex body geometries and, second, be rebuilt in each computational time step. Although the task can be successfully handled by using novel high-order sliding mesh methods [13–15], the ever-increasing demand in the analysis of the flow typical of realistic complex engineering systems can lead to a serious deterioration of the computational efficiency of the simulations. A promising alternative to this body-conformal grid approach is the immersed boundary method (IBM). Initially developed by Peskin [16], the method has become very popular over the years and is currently widely used for flow simulations in the presence of stationary and moving immersed bodies of complex geometry, as extensively reviewed in [17–21].

An additional rapidly developing field employing IBM for simulating two-way coupled fluid–structure interaction (FSI) problems is related to embedding the IBM within the framework of isogeometric analysis (IGA), originally formulated by Hughes et al. [22]. The key idea is to integrate the stand alone computational tools performing the finite element analysis (FEA) with available computer aided design (CAD) software to facilitate the design and testing of new models by using a common data set. Following the originally established IGA concept [22], new high order immersed methods using b-splines and other related interpolation functions, commonly used in computer aided design (CAD), have emerged over recent years. These formulations offer a powerful simulation strategy for FSI problems and a possible gateway for dealing with complex engineering parts and assemblies directly using CAD, as reported in [23–28].

Among the currently available methodologies for simulating the fluid flow developing in the presence of periodically moving immersed bodies, the present study focuses on the direct forcing approach, initially introduced in [29]. According to this approach, the impact of the immersed body – determined as a set of discrete Lagrangian points – on the surrounding flow is expressed by introducing additional unknowns in the form of volumetric forces, each associated with a corresponding Lagrangian point. The forces act as Lagrange multipliers that enforce a no-slip kinematic constraint on the surface of the immersed body without involving any dynamical process. In the most general case, the Lagrangian points on the body surface do not coincide with the underlying Eulerian Cartesian grid. Information is exchanged between the Eulerian grid and the Lagrangian points by two adjoint operators, namely, the interpolation operator, which interpolates the velocity values from the Eulerian grid to the Lagrangian points, and the regularization operator, which smears the volumetric forces from the Lagrangian points onto the underlying Eulerian grid. Chronologically, the implementation of the direct-forcing IBM was first based on an explicit calculation of the Lagrangian forces. The explicit formulation gained great popularity due to the fact that it did not require the introduction of any modifications to the original solver of the Navier–Stokes (NS) equations. In practice, to equip any existing NS solver with the immersed boundary functionality, the solver should be used twice: first, without taking into account the existence of the immersed body, and, second, by employing the same differential operator, but with its right hand side (RHS) modified by explicitly calculated Lagrangian forces reflecting the impact of the immersed body on the surrounding flow. The high flexibility of the explicit direct-forcing IBM is quite impressive. It has been successfully employed over the past two decades in a broad spectrum of fields, particularly in the simulation of particulate flows [30–35], thermally driven confined flows [36–39], and two-phase immiscible flows [40] and in the phenomenological modeling of the mobility and growth of cancerous tumors [41–43].

There are, however, two major drawbacks in the explicit calculation of the Lagrangian forces. The first of these is the requirement for running the simulations with very small time steps, as is required to satisfy the no-slip kinematic constraints with acceptable accuracy. A number of attempts have thus been made to relax the small time step limitation; these have included solving the coupled system of the boundary forces [37,38] and introducing an additional forcing loop for more accurate imposition of the interface velocity [32,33]. However, despite the evident success of the aforementioned studies in imposing no-slip kinematic constraints, none of the proposed strategies can

be seen as an ultimate remedy to the second – and most important – drawback related to the explicit calculation of the Lagrangian forces, namely, the violation of the elliptic character of the NS equations. The problem is encountered in the simulation of flows characterized by low and moderate Reynolds numbers (Re), but it can be solved by employing the fully coupled formulation. The key idea is to express the Lagrangian forces as additional unknowns in the form of distributed Lagrange multipliers (DLM) implicitly embedded into the corresponding NS equations. In this case, the kinematic constraints of no-slip are imposed with zero machine precision. Recent progress in this direction may be attributed to the studies of [44–49], which successfully established the DLM approach as a powerful tool for investigating incompressible flows for a wide range of Reynolds numbers, including linear stability analysis of pressure- and thermally driven flows [50] and analysis of two-phase immiscible flows [51]. Unfortunately, a purely implicit implementation of the IBM typically involves substantial modification of the original solvers, which are not initially equipped with the IBM capability. For this reason, a semi-implicit implementation of the IBM, in which the Lagrangian forces are implicitly coupled with a non-solenoidal velocity field subsequently projected onto a divergence free subspace has attracted increasing interest in recent years [52–54]. The semi-implicit formulation of the IBM can be straight forwardly embedded into the whole family of pressure–velocity segregated NS solvers based on projection or fractional step algorithms, while maintaining the accuracy of the imposed constraints of no-slip bounded by the discretization error of the numerical scheme [54].

The key idea underlying the semi-implicit implementation is to analytically decompose the operator coupling the NS equations with the constraints of no-slip and to pre-compute the contribution of the latter at the beginning of the computational process. It was shown in our previous study [54] that, after the pre-computing is completed, the efficiency of the time integration performed by the algorithm based on the semi-implicit implementation of the IBM is comparable with that of its explicit counterpart in the case of stationary immersed bodies. The present paper extends the previously developed methodology to flow simulations in the presence of periodically moving immersed bodies whose kinematics is governed by periodic functions and can therefore be split into a finite number of discrete states. Then, by taking advantage of the general parallel file system (GPFS), every pre-computed state can be saved, and, most importantly, read later very efficiently, making it possible to perform this kind of simulation in a reasonable amount of time. The current article presents the extended methodology, including a detailed description of the algorithm, a verification study of the obtained results, and the efficiency characteristics in terms of memory and time consumption. Last but not least, the most time-consuming pre-computing stage of the developed methodology is "embarrassingly" parallel, which allows for its efficient acceleration without introducing any significant modifications in the original solver. The developed method was extensively verified by comparison of the flow characteristics obtained for the simulation of confined flows developing in the presence of an oscillating sphere with the results available in the literature. The capability of the developed methodology to resolve the flows developing in the presence of bodies characterized by more complicated periodic kinematics is demonstrated by performing simulations of the flow generated by a pair of out-of-phase oscillating spheres for different values of the Reynolds number.

2. Theoretical background

The methodology presented here is a straight-forward extension of our previous study [54], incorporating the semi-implicit IBM into any generic solver of incompressible NS equations based on the projection approach. For the sake of completeness, we revisit the basic steps of the previously developed algorithm [54], with emphasis on the steps relevant to the efficient simulation of incompressible flows in the presence of periodically moving immersed bodies.

2.1. Governing equations

Following the IBM formalism, the incompressible flow developing in the presence of an immersed body whose parameterized surface $\boldsymbol{X}(\xi,\eta)$ moves in accordance with a priori determined kinematics, giving the surface velocity $\boldsymbol{U}^{\Gamma}(\boldsymbol{X})$, is governed by non-dimensional incompressible NS equations:

$$\nabla\cdot\boldsymbol{u}=0, \tag{1}$$

$$\frac{\partial\boldsymbol{u}}{\partial t}+(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}=-\nabla p+\frac{1}{Re}\nabla^2\boldsymbol{u}+\boldsymbol{f}, \tag{2}$$

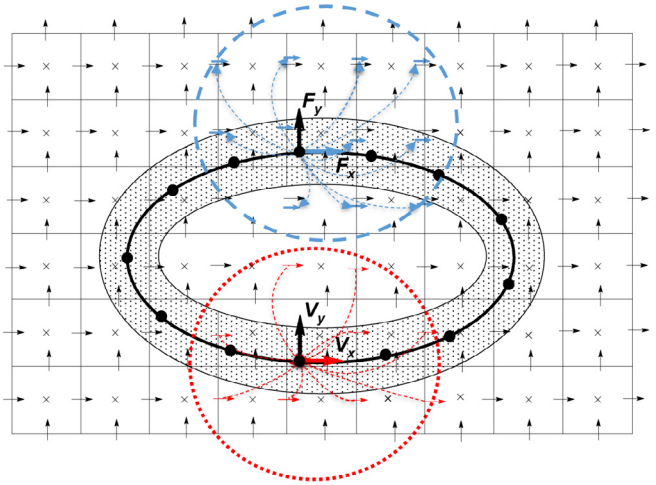


Fig. 1. Schematic representation of the major principles of the IBM. The immersed body is described by a set of Lagrangian points indicated by the full black circles. A dashed shell of thickness equal to one grid step attached to the immersed body corresponds to the set of discrete volumes surrounding each Lagrangian point. The dashed and dotted circles show the range of action of the regularized Dirac delta functions smearing the Lagrangian forces over the Eulerian grid and interpolating the Eulerian velocities on the Lagrangian points, respectively.

where the force field $\boldsymbol{f}(\boldsymbol{x})$ reflects the impact of the moving boundary of the immersed body on the surrounding flow. The value of $\boldsymbol{f}(\boldsymbol{x})$ is typically unknown, and it is therefore necessary to introduce a kinematic constraint of no-slip:

$$\boldsymbol{u}(\boldsymbol{X}) = \boldsymbol{U}^{\Gamma}(\boldsymbol{X}), \tag{3}$$

providing closure of the overall system of Eqs. (1)–(3). Note that Eqs. (1)–(2) are formulated on the whole Cartesian grid, while Eq. (3) holds only for the domain $(\boldsymbol{X})$, which includes the surface of the immersed body. We distinguish between the variables related to each of the grids by assigning them either lowercase or uppercase fonts, where all the lowercase variables are related to the Cartesian grid, and all the uppercase variables are related to the grid coinciding with the surface of the immersed body. In accordance with the IBM formalism, all the variables related to the Cartesian grid are called Eulerian variables, while the Cartesian grid is called an Eulerian grid. The surface of the immersed body is determined by a set of discrete points, which are called Lagrangian points, and all the variables related to these points are called Lagrangian variables.

An example of a uniform structured staggered 2D Eulerian grid underlying a set of Lagrangian points determining the surface of an immersed body is shown in Fig. 1. In general, the Lagrangian points do not coincide with the underlying Eulerian grid. As a result, two adjoint operators, namely, the interpolation operator, $\boldsymbol{I}$, interpolating the Eulerian velocities to the locations of the Lagrangian points, and the regularization operator $\boldsymbol{R}$, smearing the volumetric Lagrangian forces on the adjacent Eulerian grid, must now be introduced to allow exchange of information between the two grids:

$$\boldsymbol{I}(\boldsymbol{u}(\boldsymbol{x}_i)) = \int_{\Omega} (\boldsymbol{u}(\boldsymbol{x}_i)) \cdot \delta(\boldsymbol{X}_k - \boldsymbol{x}_i) dV_{\Omega_i}, \tag{4a}$$

$$\boldsymbol{R}(\boldsymbol{F}_k(\boldsymbol{X}_k)) = \int_{S} (\boldsymbol{F}_k(\boldsymbol{X}_k)) \cdot \delta(\boldsymbol{x}_i - \boldsymbol{X}_k) dV_{S_k}, \tag{4b}$$

where the indexes i and k run for the whole discrete range of Eulerian and Lagrangian coordinates, respectively, and the indexes Ω and S correspond to the Eulerian and Lagrangian grid cells, respectively, thus determining dV_{Ω_i} as the ith volume of the Eulerian flow domain and dV_{S_k} as the virtual volume confining the kth Lagrangian point. To achieve the best accuracy, a uniform grid in the vicinity of the immersed body surface is utilized. The surface of the immersed body is determined by a set of equi-spaced Lagrangian points, and the distance between the neighboring

Lagrangian points is approximately the same as the cell width of the underlying Eulerian grid, also yielding $dV_{\Omega_i} \approx dV_{S_k}$. Eqs. (4a) and (4b) utilize the regularized Dirac delta function δ of the form:

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

introduced by Roma et al. in [55]. Here, Δr is the cell width in the r direction, which means that the above delta function supports three grid cells in each spatial direction while interpolating Eulerian velocities and regularizing Lagrangian forces, since we utilize the same delta function in both the interpolation and regularization operators. The chosen delta function was specifically designed for performing calculations on staggered grids; this function has gained popularity in recent years [30,32,44,50,54] due to its compact kernel (only 3 cells in each direction of the computational domain). Interpolation of the discrete Eulerian velocities $\boldsymbol{u}_i$ and regularization of the discrete Lagrangian forces $\boldsymbol{F}_k$ for the 3D configuration is performed by employing the following formulas:

$$\boldsymbol{U}_k^{\Gamma} = \Delta x^3 \sum_i \boldsymbol{u}_i \delta(x_i - \epsilon_k)\delta(y_i - \eta_k)\delta(z_i - \zeta_k), \tag{6}$$

$$\boldsymbol{f}_i = \Delta x^3 \sum_k \boldsymbol{F}_k \delta(\epsilon_k - x_i)\delta(\eta_k - y_i)\delta(\zeta_k - z_i), \tag{7}$$

yielding the resultant discrete boundary velocity at the kth Lagrangian point, $\boldsymbol{U}_k^{\Gamma}$ and the discrete volumetric force at the ith point $(x_i,\ y_i,\ z_i)$ of the Eulerian staggered grid. According to the SIMPLE method [56], NS equations (Eqs. (1)–(3)), equipped with the IBM capability are transformed into:

$$\frac{1}{Re}\boldsymbol{L}(\boldsymbol{u}^*) - \frac{3\boldsymbol{u}^*}{2\Delta t} + \boldsymbol{R}(\boldsymbol{F}_k(\boldsymbol{X}_k)) = \frac{-4\boldsymbol{u}^n + \boldsymbol{u}^{n-1}}{2\Delta t} + \boldsymbol{N}(\boldsymbol{u}^n) + \nabla p^n, \tag{8}$$

$$\boldsymbol{I}(\boldsymbol{u}^*(\boldsymbol{x}_i)) = \boldsymbol{U}^{\Gamma}(\boldsymbol{X}_k), \tag{9}$$

$$\triangle(\delta p) = \frac{3}{2\Delta t}\nabla \cdot \boldsymbol{u}^*, \tag{10}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \frac{2\Delta t}{3}\nabla(\delta p), \qquad p^{n+1} = p^n + \delta p, \tag{11}$$

where Eqs. (8), (9) play the role of predictor, yielding a non-solenoidal velocity field $\boldsymbol{u}^*$ by taking the values of the pressure field p^n from the previous time step, and Eqs. (10), (11) are used to correct the pressure field and to project the predicted velocity onto a divergence-free subspace. The linear terms $\boldsymbol{L}$ entering the momentum equation (8) and corresponding to the Laplace operator are treated implicitly, while the nonlinear convective terms $\boldsymbol{N}$ are taken explicitly from the previous time step. A second-order backward finite difference scheme is used for time discretization, and a standard second-order finite volume method [56] is used for the discretization of all the spatial derivatives.

2.2. Domain decomposition

Explicit treatment of the nonlinear terms allows us to solve Eqs. (8)–(9) successively for each component of the predicted velocity vector $\boldsymbol{u}^*$ and then to continue with a standard projection-correction step determined by Eqs. (10)–(11). We then rewrite Eqs. (8)–(9) in a compact block-matrix form:

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{R} \\ \boldsymbol{I} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^* \\ \boldsymbol{F} \end{bmatrix} = \begin{bmatrix} RHS^{n-1,n} \\ \boldsymbol{U}^{\Gamma} \end{bmatrix}, \tag{12}$$

where $\boldsymbol{H} = \frac{1}{Re}\boldsymbol{L} - \frac{3}{2\Delta t}\mathbf{I}$ is the Helmholtz operator acting on each component of the predicted velocity vector $\boldsymbol{u}^*$; $\mathbf{I}$ is a unity matrix; $\boldsymbol{R}$ and $\boldsymbol{I}$ are rectangular matrices that contain terms resulting from applying the regularization and interpolation operators, respectively; and $RHS^{n-1,n}$ is the RHS vector containing the pressure gradient and nonlinear convective terms known from the previous time steps. We note in passing that the matrices $\boldsymbol{R}$ and $\boldsymbol{I}$ are transpose to one another if the same Dirac delta function is utilized in both the interpolation and regularization operators. This property is, however, not explicitly exploited in the present study.

We next focus on the strategy for the solution of the system of Eqs. (12) by utilizing the domain decomposition technique. Suppose that a generic package capable of solving the Helmholtz equation is available:

$$[\boldsymbol{H}][\boldsymbol{u}^*] = [RHS^{n-1,n}]. \tag{13}$$

Assume also that the above package can be used as a black box driver (i.e., without introducing any modifications into the package itself) to obtain $\boldsymbol{u}^*$. In general, such a driver can be one of two types: either a standalone solver capable of solving the Helmholtz equation (e.g., FISHPACK [57]) or a whole CFD package for simulation of incompressible flows from the family of projection or fractional step solvers (e.g., OpenFoam [58]). When utilizing a driver of the first type, one should explicitly build and provide the driver with the matrix $\boldsymbol{H}$ and the vector $RHS^{n-1,n}$ with appropriate boundary conditions, while utilizing a driver of the second type will require only proper determination of the boundary conditions. Furthermore, drivers of both types can also be used for calculating the product of $\boldsymbol{H}^{-1}$ and any generic vector $\boldsymbol{Z}$. For a driver of the first type, the product is obtained by providing the package with an already existing matrix $\boldsymbol{H}$ and with a modified RHS vector whose values are now equal to the values of the $\boldsymbol{Z}$ vector. For a driver of the second type, the only requirement is modification of the RHS vector. To demonstrate the capabilities of the developed methodology, the solver of the second type developed in [59] is used as the driver. Keeping in mind the above capabilities, we perform an analytic transformation of the system of Eqs. (12) known as a Schur complement decomposition:

$$\boldsymbol{F} = [\boldsymbol{I}\boldsymbol{H}^{-1}\boldsymbol{R}]^{-1}[\boldsymbol{I}\boldsymbol{H}^{-1}RHS^{n-1,n} - \boldsymbol{U}^{\Gamma}], \tag{14a}$$

$$\boldsymbol{u}^* = \boldsymbol{H}^{-1}[RHS^{n-1,n} - \boldsymbol{R}\boldsymbol{F}]. \tag{14b}$$

The idea is to first find the values of the DLM, $\boldsymbol{F}$, providing the kinematic constraints of no-slip, and thereafter to use them to find the values of the predicted velocity vector, $\boldsymbol{u}^*$. Note that despite the fact that the process is separated into two stages, the obtained $\boldsymbol{F}$ and $\boldsymbol{u}^*$ values are fully coupled, as the solution procedure is implicit. Obtaining the solution of the whole problem by a straight-forward application of Eqs. (14a) and (14b) would require us to invert the matrices $[\boldsymbol{H}]$ and $[\boldsymbol{I}\boldsymbol{H}^{-1}\boldsymbol{R}]$, which is computationally prohibitive. Instead, the solution can be obtained by calculation of a series of matrix–vector products for the matrix $[\boldsymbol{H}^{-1}]$ and subsequent lower-upper (LU) decomposition of the small matrix $[\boldsymbol{I}\boldsymbol{H}^{-1}\boldsymbol{R}]$. Moreover, most calculations required to complete the stages described in Eqs. (14a) and (14b) can be pre-computed once at the beginning of the computational process and then reused throughout the simulation. The details of implementation of the developed methodology are given in the next section.

3. Implementation details

Prior to describing the numerical procedure developed for the solution of Eqs. (14a) and (14b), we make a number of observations regarding the structure of matrices $\boldsymbol{I}$ and $\boldsymbol{R}$. We recall that the matrices contain the terms that were obtained by employing the interpolation and regularization operators, respectively, to enforce the kinematic constraints of no-slip on the surface of the immersed body. The characteristic property of these two matrices is their extreme sparseness, resulting from the compact kernel of the utilized discrete delta function. In fact, considering a typical 3D problem characterized by $[10^6 - 10^7]$ degrees of freedom for each velocity component, a row of matrix $\boldsymbol{I}$ (or alternatively a column of matrix $\boldsymbol{R}$) contains only order of 10^2 non-zero values. As a result, both the $\boldsymbol{I}$ and $\boldsymbol{R}$ matrices can be stored in compressed sparse row (CSR) format, while their matrix–vector product can be efficiently calculated by employing standard routines from the Intel Math Kernel Library (MKL). Next, we give a detailed description of both the pre-computing and time integration stages of the developed methodology.

3.1. Pre-computing stage

Recalling that the present methodology was developed for flow simulation in the presence of periodically moving immersed bodies, we divide a single period into an integer number of cyclically repeated time steps. All the procedures described in this section are thus performed for each time step entering into a single period.

3.1.1. Calculation of matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$

The matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$ is a square matrix of dimensions $(m \times m)$, where m is the total number of Lagrangian points determining the surfaces of all the immersed bodies involved in the simulation. For a typical 3D problem, m lies within the range of $m \in [10^3 - 10^4]$. Note also that according to [54] an absolute value of the sparsing threshold, s, is set to $s = 10^{-21}$. Consequently, only entries with an absolute value higher than the value of s are stored in the matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$. The matrix itself is stored in a row-column-value (RCV) format, as follows:

- For each column $[\boldsymbol{R}]_k$, $1 \leq k \leq m$ of the matrix $\boldsymbol{R}$:
  - Employ a generic driver of either the first or second type, with a modified right hand side $[RHS^{n-1,n}] = [\boldsymbol{R}]_k$ to calculate the product $\boldsymbol{H}^{-1}[\boldsymbol{R}]_k$;
  - Multiply the matrix $\boldsymbol{I}$ stored in the CSR format by the obtained vector, employing standard routines from the Intel MKL;
  - Fill the kth column $[\boldsymbol{IH}^{-1}\boldsymbol{R}]_k$ of the matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$ in RCV format while using the sparsing threshold, s.

Note that the above process allows us to avoid the creation of intermediate matrices of large dimensions and to directly build the matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$ of dimensions $(m \times m)$. An additional important observation is that all three stages of the above algorithm are independent of both the column number k and the time step Δt, which makes the whole pre-computing stage "embarrassingly" parallel. This parallelism can be exploited on two levels: first, while building the matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$, i.e., the columns of the matrix can be calculated separately and then collected into the whole matrix for performing the calculation stage,[1] and second, while calculating the matrices $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$ corresponding to the different time steps in a single period.[2]

3.1.2. Factorization of matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$

Instead of direct calculation of the inverse of the matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$, we perform LU factorization, which then allows us to obtain the product of $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$ and any generic vector of appropriate length. LU factorization was performed by utilizing the open-source MUMPS solver [60,61], and the LU factors calculated for each time step were stored on a hard disk.[3]

3.2. Time integration stage

After completing the pre-computing stage, the time integration stage can be initiated. Implementation details of all the steps required to complete the solution of Eqs. (14a) and (14b) for one time step are as follows:

- Calculate $[\boldsymbol{IH}^{-1}RHS^{n-1,n}]$:
  - Employ a generic driver, of either the first or the second type, with its original RHS to calculate the product $[\boldsymbol{H}^{-1}RHS^{n-1,n}]$;
  - Multiply the matrix $\boldsymbol{I}$ stored in CSR format by the obtained vector, employing standard routines from the Intel MKL;
- Calculate $[\boldsymbol{IH}^{-1}RHS^{n-1,n} - \boldsymbol{U}^{\Gamma}]$ simply by subtracting two vectors;
- Calculate $\boldsymbol{F}$[4] (See Eq. (14a)):
  - Retrieve the LU factors of the matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$, which had been calculated in the pre-computing stage and stored on a hard disk for every time step;
  - Assign $[\boldsymbol{IH}^{-1}RHS^{n-1,n} - \boldsymbol{U}^{\Gamma}]$ to the RHS of Eq. (14a);
  - Perform standard backward and forward substitutions of the factorized matrix $[\boldsymbol{IH}^{-1}\boldsymbol{R}]$ with respect to the RHS built in the previous step;

[1] This strategy is preferable for stationary immersed bodies.

[2] This strategy is preferable for periodically moving immersed bodies and was utilized in the present study.

[3] A still not released version of MUMPS allowing for the storage and reconstruction of LU factors was provided by the MUMPS developers, who can be contacted through the software website http://mumps.enseeiht.fr/.

[4] All the stages of this bullet are automated in MUMPS.

- Calculate $\boldsymbol{u}^*$ (See Eq. (14b)):
  - Multiply the matrix $\boldsymbol{R}$ stored in CSR format by the calculated DLM vector $\boldsymbol{F}$, employing standard routines from the Intel MKL;
  - Calculate $[RHS^{n-1,n} - \boldsymbol{RF}]$ simply by subtracting two vectors;
  - Employ a generic driver, of either the first or the second type, with a modified RHS equal to $[RHS^{n-1,n} - \boldsymbol{RF}]$.

Note that after completing the pre-computing stage, the time integration of the current methodology is based on a double implementation of the original generic driver (of either the first or the second type), namely, for the first time when calculating $\boldsymbol{H}^{-1}RHS^{n-1,n}$ and for the second time when calculating $\boldsymbol{H}^{-1}[RHS^{n-1,n} - \boldsymbol{RF}]$ products exactly the same procedure as that for any fully explicit formulation of the direct forcing IBM. As a result, the developed semi-implicit methodology not only provides a more accurate imposition of the kinematic constraints of no-slip on the surfaces of the periodically oscillating immersed bodies, but it is also as time efficient as its fully explicit counterpart.

4. Results and discussion

4.1. Flow around a transversely oscillating sphere

An oscillating sphere of diameter D in an otherwise quiescent fluid confined by a rectangular prism of dimensions $4D \times 4D \times 6D$ is considered (see Fig. 2). The sphere oscillates in the z direction with periodic velocity U_z given by:

$$U_z = U_{max} sin(\omega T), \tag{15}$$

where ω is the angular oscillating frequency, and T is dimensional time. By setting the location of the center of the coordinates in the middle of the prism, the vertical coordinate of the center of the sphere can be obtained directly by integrating Eq. (15) over time to yield:

$$Z = -\frac{U_{max}}{\omega} cos(\omega T). \tag{16}$$

No-slip boundary conditions are applied to the surface of the sphere and all the walls of the computational domain. After the values of D, U_{max}, U_{max}/D and ρU_{max}^2 have been used for scaling the length, velocity, time and pressure fields, respectively, the kinematics of the oscillating sphere is governed by the following non-dimensional equations:

$$u_z = sin\left(\frac{D}{A}t\right), \tag{17}$$

$$z = -\frac{A}{D} cos\left(\frac{D}{A}t\right), \tag{18}$$

where $A = U_{max}/\omega$ corresponds to the oscillation amplitude of the sphere. The flow is governed by two non-dimensional parameters, namely, $Re = U_{max}D/\nu$ and the ratio of the oscillation amplitude to the sphere diameter, A/D. We now focus on the force balance equation of an accelerating sphere in an otherwise quiescent fluid. Note that in accordance with the IBM formalism the sphere is filled with the same fluid as that outside the sphere; the force balance equation may thus be expressed as [33]:

$$\frac{d}{dt}\int_{V_{sph}} \boldsymbol{u} dV = \boldsymbol{f}_D + \int_{V_{sph}} \boldsymbol{f} dV. \tag{19}$$

The first term of the RHS of Eq. (19) represents the instantaneous drag force exerted on the sphere by the fluid, while the second term corresponds to an instantaneous external force that should be exerted on the body to provide its prescribed kinematics. This term can be directly calculated by summing all the IBM forces in accordance with:

$$\int_{V_{sph}} \boldsymbol{f} dV = \sum_{ijk} \boldsymbol{f}_{ijk} \Delta x \Delta y \Delta z. \tag{20}$$

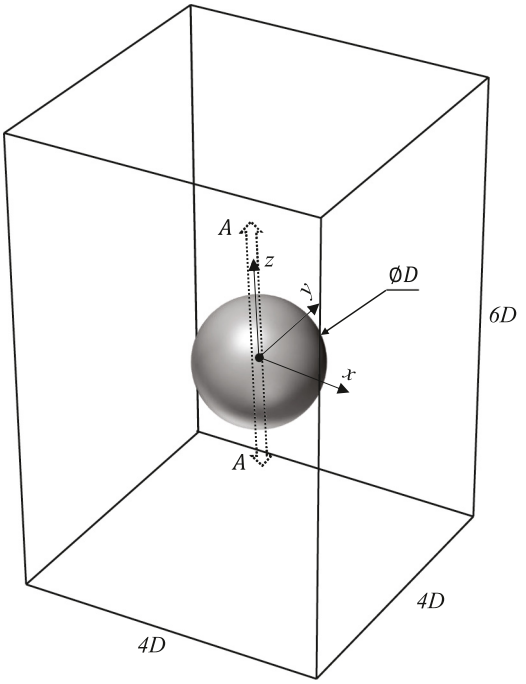


Fig. 2. Schematic representation of the physical model. A sphere of diameter D is confined by a prismatic enclosure of dimensions $4D \times 4D \times 6D$. The sphere oscillates in the z direction with amplitude A.

Eqs. (19), (20) enable computation of the drag force, $\boldsymbol{f}_D$. To perform this calculation, we assume rigid-body motion of the whole sphere interior [30], which allows us to approximate the LHS term of Eq. (19) as:

$$\frac{d}{dt}\int_{V_{sph}} \boldsymbol{u} dV = \frac{d\boldsymbol{u}_c}{dt} V_{sph}, \tag{21}$$

where $d\boldsymbol{u}_c/dt$ is the acceleration of the center of mass of the sphere, which can be obtained analytically by taking the time derivative of Eq. (17). For purposes of comparison, the drag force, $\boldsymbol{f}_D$, can also be expressed in terms of the drag coefficient determined for a spherical geometry as:

$$C_{D_i} = \frac{8F_{D_i}}{\rho U_{max}^2 \pi D^2}, \tag{22}$$

where F_{D_i} is the value of the ith component of the dimensional drag force $\boldsymbol{F}_D$. Utilizing the scaling of the current study, the drag coefficient may be expressed in terms of the non-dimensional drag force $\boldsymbol{F}_D$ as $C_{D_i} = 8F_{d_i}/\pi$.

4.1.1. Verification study

The developed methodology was verified by comparison of the results obtained for a transversely oscillating sphere with the corresponding data available in the literature. We started with a comparison of the presently calculated time evolutions of the non-dimensional drag forces in the z direction with the corresponding values reported in [62] for two sets of Re and A/D values, namely, $(Re, A/D) = (40, 5)$ and $(Re, A/D) = (40, 0.3125)$, as shown in Fig. 3[5]. The calculations were performed on $200 \times 200 \times 300$ uniform grid with a time step Δt equal to 10^{-3} of the oscillation period. An acceptable agreement between the present and previously reported F_D values was obtained for the entire length of the oscillation period, while the existing insignificant deviation between the F_D values can be attributed to the assumptions of an infinite computational domain, and an axi-symmetric flow regime made in [62].

[5] The non-dimensional time t and drag force F_D were multiplied by the factors $\frac{A}{D}$ and $\frac{3\pi}{Re}$, respectively, to match the scaling used in [62].

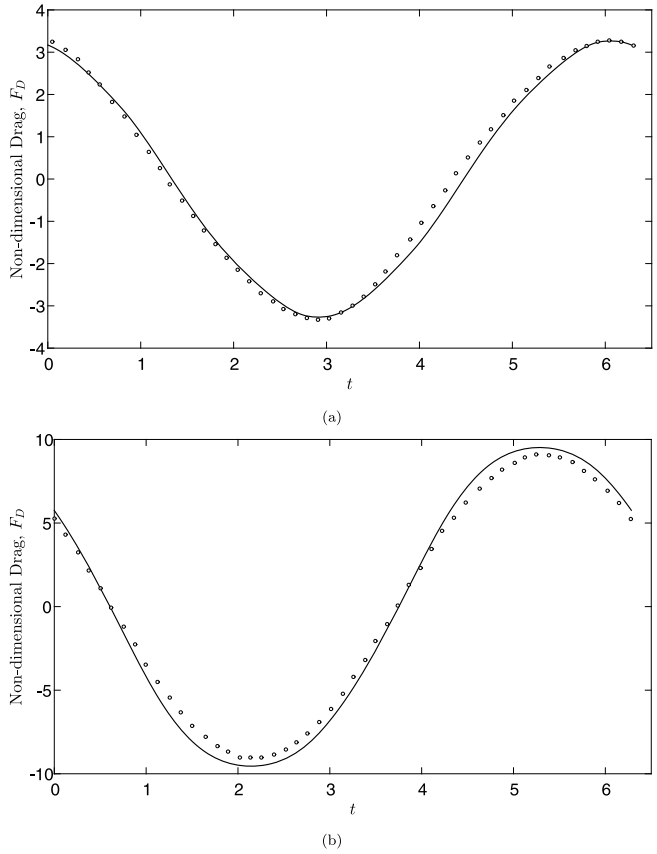


Fig. 3. Comparison between the time evolutions of the non-dimensional drag forces F_D obtained for: (a) $(Re, A/D) = (40, 5)$, and, (b) $(Re, A/D) = (40, 0.3125)$. Solid lines correspond to the currently obtained results, while open circles correspond to the values that were digitally scanned from [62].

We then further verified the developed methodology by comparison of the values of the peak drag coefficient, C_{Dmax}, with the corresponding values reported in [63] as shown in Fig. 4. The C_{Dmax} values were calculated on the basis of the drag force in the z direction exerted by the flow on the vertically oscillating sphere. The characteristics of the computational set up in terms of the grid resolution and the time step value were the same as those used in the previous verification test. In agreement with the available data [63], our current results successfully reproduce two general trends: first, for a given value of the A/D ratio, the C_{Dmax} values decreased with increasing values of the Re number; and second, for a given value of the Re number, the C_{Dmax} values decreased with increasing A/D. The first trend can be attributed to the fact that the value of the drag coefficient is inverse to Re, while the second trend is related to the inertia effects of the flow, which are inverse to the A/D ratio.[6]

It can be seen that there is acceptable agreement between the current and previously reported C_{Dmax} values for the entire range of Re and A/D values. As a general trend, the discrepancy between the results increased with increasing Re and decreasing A/D values. The maximal discrepancy (not exceeding 7%) was observed for $Re = 100$ and $A/D = 0.5$. The discrepancies between the results of the current study and those reported in [63] can be explained by differences in the size of the computational domains and the basic assumptions made when performing the numerical simulations. In particular, the results reported in [63] were obtained on the assumption of axi-symmetric flow in the computational domain extending 50 diameters in both the radial and axial directions, while the present simulations were performed for a much smaller $(4D \times 4D \times 6D)$ non-axi-symmetric domain. The higher values of the peak drag coefficient obtained in the current study can be attributed to the increased

[6] This statement becomes obvious after taking the time derivative of Eq. (17).

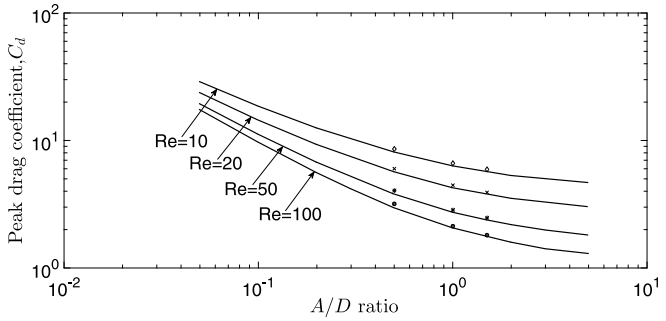


Fig. 4. Comparison of the peak drag coefficient, C_{Dmax}, obtained for $Re = 10$ ($\diamond$), $Re = 20$ ($\times$), $Re = 50$ ($*$) and $Re = 100$ ($\bullet$) as a function of the A/D ratio with the corresponding C_{Dmax} values reported in [63].

impact of the no-slip boundaries of the prism. Another possible reason for the observed discrepancies could be the assumption, in the current study, of rigid-body motion of the sphere interior in the calculation of the instantaneous drag force F_D. This assumption is a good approximation when the flow inertia is low, i.e., for high values of the A/D ratio, but it may not be entirely true for low A/D values, a notion that is also consistent with the slightly higher value of the presently obtained peak drag compared with the corresponding value from [62] obtained for $(Re, A/D) = (40, 0.3125)$ (see Fig. 3-b). However, this deviation is evident only in the post-processing stage and has no impact on the accuracy of the calculated flow field characteristics.

4.1.2. Grid and time step convergence

The sensitivity of the developed methodology to the grid size and time step values was also investigated. For this purpose, the convergence of the values of the peak drag coefficients obtained the coarse ($100 \times 100 \times 150$) and fine ($200 \times 200 \times 300$) grids was studied. The results obtained on the two grids were then further extrapolated to their zero-grid-size asymptotic values, by employing the Richardson extrapolation, and thereafter compared with the numerical results reported in [63]. A detailed comparison of all the obtained results is summarized in Table 1. It can be seen that the C_{Dmax} values decreased monotonically with mesh refinement, while the deviation between the fine grid and the zero-grid-size asymptotic values did not exceed 3%. An additional important observation is related to the time evolution of the C_{Dmax} obtained on different grids. Spurious high-frequency oscillations (in the form of saw teeth) were observed for the time evolution of C_{Dmax} values calculated on the coarse grid. These oscillations were, however, completely smoothed out with mesh refinement. For this reason, all further results obtained in the course of the present study were obtained on a $200 \times 200 \times 300$ grid. An extensive investigation of the sensitivity of the obtained results to the time step value revealed that sustainable numerical stability of the developed method was achieved for Courant number values $C \leq 0.2$, while a time step independence of the obtained results (including smoothing out of the spurious saw teeth oscillations) was achieved for the values of $\Delta X \leq 2 \times 10^{-2}$, $\Delta t \leq 10^{-3}$ and $Re \leq 200$. In addition, we were able to show that throughout the entire oscillation period the maximal absolute discrepancy between the sphere velocity determined by its kinematics and the velocities obtained by interpolation from the Eulerian grid to the sphere surface was within a truncation error of the numerical scheme (i.e., $O(\Delta X^2)$), which verifies accuracy of the imposition of no-slip kinematic constraints on the surface of the sphere.

4.1.3. Efficiency characteristics

We now focus on the efficiency characteristics of the developed methodology. In the first stage, we focus on obtaining trends for memory consumption and averaged wall clock time taken to compute a time step in the pre-computing and time integration stages, both utilizing the MUMPS solver. The trends were obtained by performing all the simulations on a standard Unix server equipped with 2 Intel Xeon 12C Processors (Model E5-2697v2, 24 cores in total), 128 GB RAM. We used the default MUMPS settings, which activate OpenMP parallelization on all the cores for performing LU-decomposition. It should be noted that the current study used only the shared memory (i.e., OpenMP) parallelization version of the driver for the solution of the NS equations [59]. During this stage, the number of cores used was kept constant and equal to 24, which means that all the available cores of the server were used. The purpose of the current stage was to predict the RAM consumption and the averaged wall clock time

Table 1
Comparison between current and previously published C_{Dmax} values.

Peak drag coefficient									
		$Re = 10$					$Re = 20$		
A/D	$100 \times 100 \times 150$	$200 \times 200 \times 300$	Richardson	Ref. [63]	A/D	$100 \times 100 \times 150$	$200 \times 200 \times 300$	Richardson	Ref. [63]
0.5	8.88	8.59	8.49	8.14	0.5	6.26	6.01	5.93	5.7
1	6.77	6.62	6.57	6.32	1	4.58	4.47	4.43	4.27
1.5	6.02	5.92	5.89	5.8	1.5	3.98	3.93	3.91	3.83
		$Re = 50$					$Re = 100$		
A/D	$100 \times 100 \times 150$	$200 \times 200 \times 300$	Richardson	Ref. [63]	A/D	$100 \times 100 \times 150$	$200 \times 200 \times 300$	Richardson	Ref. [63]
0.5	4.27	4.04	3.96	3.82	0.5	3.44	3.18	3.09	2.97
1	2.93	2.85	2.82	2.74	1	2.33	2.13	2.06	2.06
1.5	2.51	2.47	2.46	2.39	1.5	1.89	1.82	1.8	1.77

at both the pre-computing and time integration steps, which will be required for solving problems with an even denser grid resolution than those presently utilized, given that the same hardware and the OpenMP parallelization are used.

The trends for both efficiency characteristics were obtained by utilizing a power law best fit of the corresponding measurements of the efficiency characteristics made for three grid resolutions and four values of the Re number, as shown in Fig. 5(a) – (c). It is noteworthy that the pre-computing stage was characterized by higher absolute values of consumed RAM than the time integration stage. This difference can be explained by the extra memory consumed by the MUMPS solver while performing LU factorization. Once the factorization stage is completed, only the LU factors are stored on the hard disk,[7] while all the dynamically allocated auxiliary memory is automatically released. The finding that the configurations characterized by lower Re numbers typically consumed more memory can be explained by the more pronounced elliptical character of the low-Reynolds flows, which led to a higher number of non-zero entries in the $[\mathbf{I}\mathbf{H}^{-1}\mathbf{R}]$ matrix, thereby satisfying the pre-determined sparsing threshold $s = 10^{-21}$. We note that the exponent values of the memory consumption best fits built for both the pre-computing and time integration stages are significantly lower than unity, indicating efficient exploitation of the sparseness of the matrices involved in both stages.

The efficiency of the developed methodology in terms of the averaged wall clock time taken to compute a time step in both pre-computing and time integration stages can be assessed by examining the corresponding best fits shown in Fig. 5(c). It can be seen that the exponent value corresponding to the best fit built for the time integration stage is very close to unity, indicating the high efficiency of the developed methodology, whose time consumption grows almost linearly with increasing grid resolution. We note here that the best fit corresponding to the time consumption in the pre-computing stage is characterized by a much higher slope than that corresponding to the time integration stage. This observation can be explained by the fact that an increasing grid resolution actually has a dual effect on the number of calculations to be performed in the pre-computing stage. First, it increases the dimensions of the matrix $[\mathbf{H}]$, corresponding to the Helmholtz operator of original solver, which is not equipped with the immersed boundary functionality and second, it increases the number of Lagrangian points so as to meet the requirement of approximately the same distance between the neighboring Lagrangian points and the cell width of the underlying Eulerian grid, which is prerequisite for achieving high accuracy of the obtained results.

In the next stage we focus on the strong scaling characteristics of the developed methodology. The speed up S was obtained for pre-computing and time integration steps on $100 \times 100 \times 150$ and $200 \times 200 \times 300$ grids by employing 1, 2, 6, 12 and 24 processors on a single server. The speed up values for both stages averaged over the entire set of Re numbers are shown if Fig. 6. Remarkably, that the speed up values obtained for the pre-computing stage on both grids and approximated by the power fit law are characterized by about the same (≈ 0.6) value of exponent. The superiority of the speed up values obtained for the denser grids is due to the coefficient multiplying the exponent. The coefficient value is apparently correlated with the ratio between the time spent on the operations run in parallel and the total time spent also on sequential parts of the algorithm and including overheads. As expected, the ratio value increases with the grid resolution resulting in about 1.5 higher speed up obtained for $200 \times 200 \times 300$ grid compared to that obtained for $100 \times 100 \times 150$ when running the simulation of 24 cores.

[7] The stored factors are later retrieved cyclically during the time integration stage.

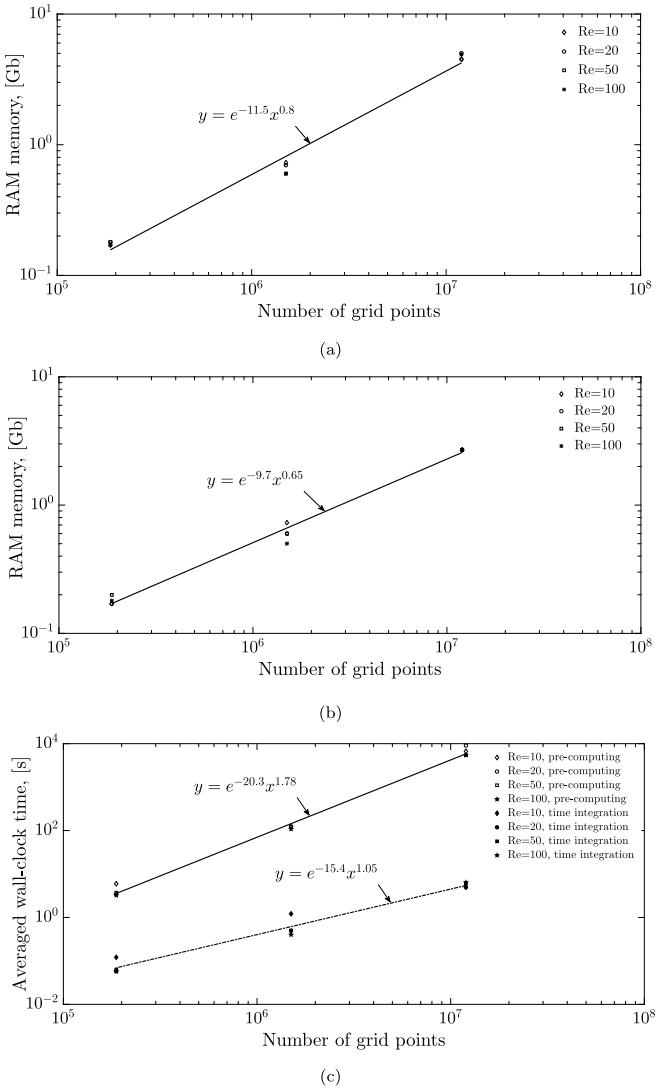


Fig. 5. Efficiency characteristics: (a) memory consumption in the pre-computing stage; (b) memory consumption in the time integration stage; (c) averaged wall-clock time taken to compute a time step in the pre-computing and time integration stages.

The time integration step is characterized by a much lower speed up values compared to the pre-computing step which is a consequence of intrinsically sequential character of backward and forward substitution procedures constituting the major part of calculations performed at this step. Despite significant scattering of the speed up values, the same trend as that observed for the pre-computing stage, i.e. the higher speed up values typical of the denser grids is evident. After approximating $S - Np$ functionality by the power law best fit it can be seen that the approximation made for the denser grid is characterized by both higher coefficient and exponent values, compared to its coarse grid counterpart.

4.1.4. Flow around a transversely oscillating sphere

Fig. 7 presents the time evolution of the drag coefficient, C_D, obtained for the values of $Re = 100, 200, 150$ and 200 and $A/D = 1$. The trend observed here was already evident in Fig. 4, i.e., the inverse growth of the peak drag coefficient with Re is also preserved for the higher values of Re number. Noteworthy also is the presence of non-negligible inertia effects of the flow, which are expressed in a clearly visible phase lag between the time evolution of the position of the sphere $z - z_0$ and the C_D curves, indicating that the fluid driven by an oscillating sphere

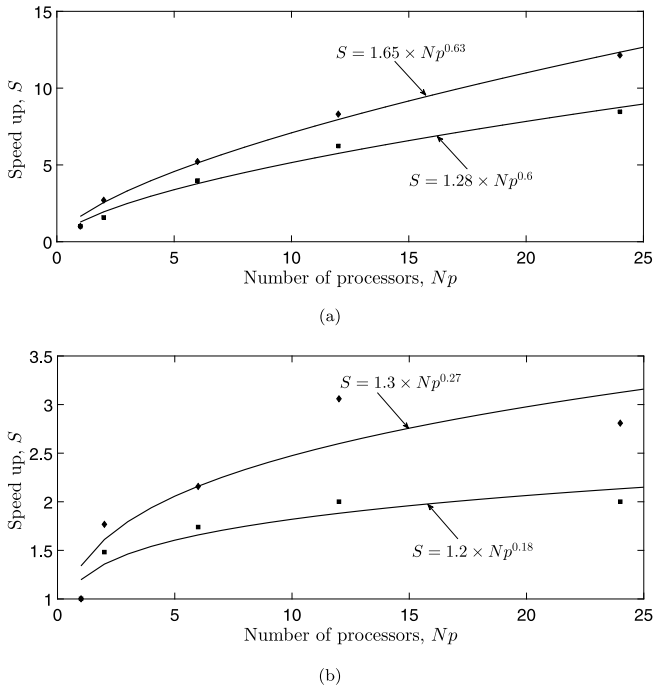


Fig. 6. Speed up values S calculated for the problem simulated on $100 \times 100 \times 150$ grid (marked by black squares ■) and on $200 \times 200 \times 300$ grid (marked by black diamonds ♦) as function of employed cores: (a) pre-computing step; (b) time integration step.

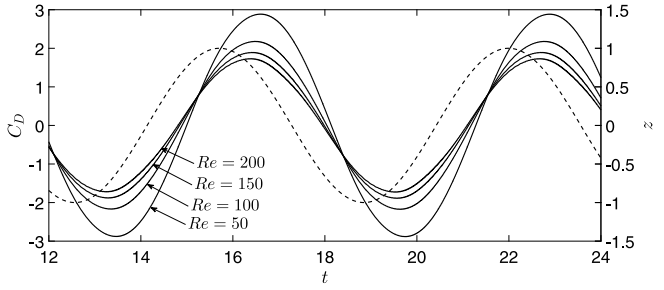
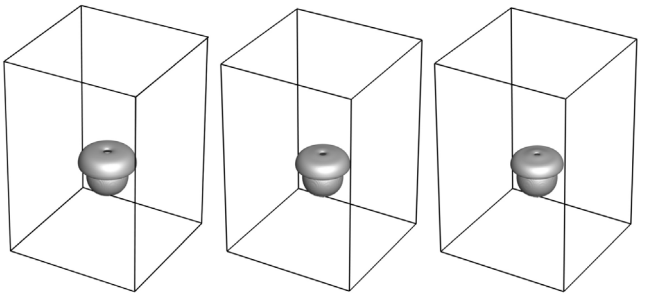


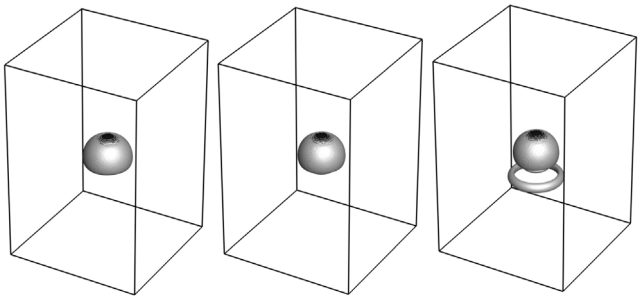
Fig. 7. Time evolution of the drag coefficient, C_D, obtained for $Re = 50, 100, 150$, and 200 (solid line) and $A/D = 1$ superimposed on the time evolution of the position of the sphere, $z - z_0$ (dashed line).

continues to move even after the sphere has completely stopped moving. The flow patterns developing in different parts of the sphere trajectory may be visualized by examining the isosurfaces of the λ_2 criterion corresponding to the value of $\lambda_2 = -0.1$, as shown in Fig. 8. According to [64], the isosurfaces characterize the vortical structures of the flow. As a result of the flow inertia, annular vortical structures form at the lowest and highest points of the sphere trajectories and are then shed from the sphere surface. It is also remarkable that the lifetime of the annular vortices increases with increasing values of the Re number. In fact, for $Re = 100$, the vortical structure dissipates almost immediately after separation from the surface of the sphere, while for $Re = 200$, the structure continues to evolve within about half a period of the sphere oscillation.

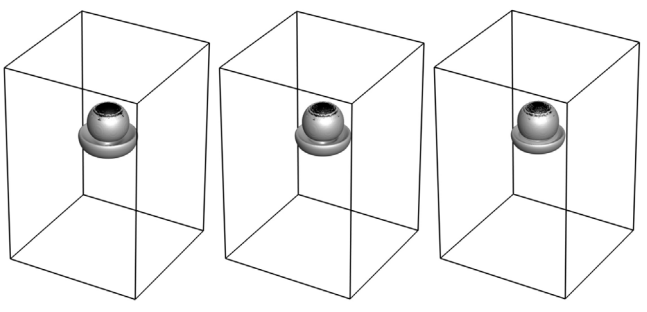
It should be noted here that attempts were made to simulate the flow developing at an even higher value of Re, namely, $Re = 300$. Unfortunately, however, when the simulations were performed on a $200 \times 200 \times 300$ grid, the time evolution of C_D was again polluted by the spurious high-frequency oscillations similar to those previously observed for simulations on a $100 \times 100 \times 150$ grid. For stationary setups, it would be possible to eliminate this numerical artifact by performing simulations on even finer grids, but for configurations containing oscillatory



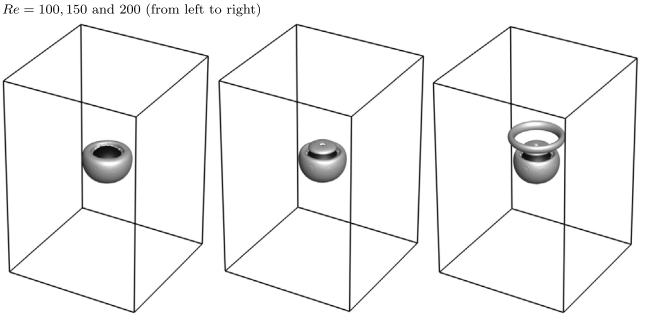
(a) Isosurfaces of the $\lambda_2 = -0.1$ criterion calculated at the lowest point of the sphere trajectory for $Re = 100, 150$ and 200 (from left to right)



(b) Isosurfaces of the $\lambda_2 = -0.1$ criterion calculated at the mid-point of the sphere trajectory on its way up for $Re = 100, 150$ and 200 (from left to right)



(c) Isosurfaces of the $\lambda_2 = -0.1$ criterion calculated at the highest point of the sphere trajectory for $Re = 100, 150$ and 200 (from left to right)



(d) Isosurfaces of the $\lambda_2 = -0.1$ criterion calculated at the mid-point of the sphere trajectory on its way down for $Re = 100, 150$ and 200 (from left to right)

Fig. 8. Visualization of the vortical structures generated by a transversely oscillating sphere over a single oscillation period, calculated for $Re = 100, 150$ and 200 and $A/D = 1$.

moving immersed bodies it becomes prohibitively expensive. Motivated by this constraint, our future research will be focused on developing a second-order sharp interface formulation of the developed methodology that will allow us to implicitly impose no-slip kinematic constraints with the second-order accuracy while keeping the method's

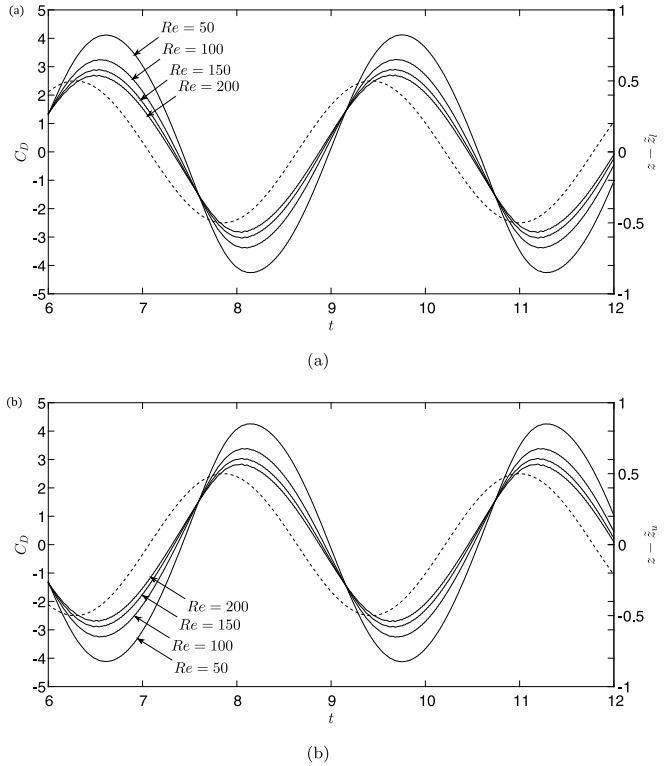


Fig. 9. Time evolution of the drag coefficient, C_D, obtained for $Re = 50, 100, 150$ and 200 (solid line) and $A/D = 0.5$ for: (a) the lower sphere superimposed on the time evolution of the position of the sphere, $z - \tilde{z}_l$ (dashed line); and (b) the upper sphere superimposed on the time evolution of the position of the sphere, $z - \tilde{z}_u$ (dashed line).

intrinsic portability, i.e., without introducing any modifications into the existing efficient time steppers of the NS equations.

4.2. Flow around a pair of transversely out-of-phase oscillating spheres

To further demonstrate the capabilities of the developed methodology, the configuration of a pair of out-of-phase oscillating spheres was considered. Similarly to the previous configuration, the flow was calculated for the values of $Re = 100, 200, 150$ and 200. Both spheres oscillate with a value of $A/D = 0.5$, each around its own oscillation center placed at distance of $1.25D$ from the top and the bottom boundaries of the prism for the upper and lower spheres, respectively. As a result, the minimal distance between the spheres of $0.5D$ is reached when the velocity of the two spheres is zero. This setup was chosen carefully with the aim to use the same prismatic enclosure of dimensions $4D \times 4D \times 6D$ as that utilized in previous simulations. The time evolution of the drag coefficient C_D calculated for the upper and lower spheres as a function of the Re number is presented in Fig. 9(a) and (b), respectively. The time evolution of C_D is superimposed on the time evolution of the position of each sphere relative to its oscillation center, as indicated by dashed lines in the Figures.

Similarly to the previous configuration, there is a clearly distinguishable phase lag between the time evolutions of C_D and the corresponding position of the sphere, which again confirms the existence of the non-negligible inertia effects of the surrounding flow. By virtue of the symmetry of the initial and boundary conditions, it can be seen that the time evolutions of C_D of both spheres are symmetric relative to the corresponding oscillation centers for the entire range of Re numbers up to a slight bias reflected in a difference between the maximal and minimal absolute values for C_D of each sphere. In particular, the absolute C_D values are consistently higher when the spheres are close to each other for the entire range of Re values, as follows from the data acquired over a single oscillating period for

Table 2
Absolute values of the minimum and maximum C_D acquired for the upper sphere as a function of the Re number.

Re	Max	Min
50	4.1	4.25
100	3.24	3.37
150	2.89	3
200	2.69	2.8

the upper sphere,[8] as detailed in Table 2. It is noteworthy that the C_D time evolutions acquired for the two spheres are equal to each other up to multiplication by minus unity i.e., the mutual effect of the spheres is independent of the direction of their motion and is a function only of their acceleration and the distance between them. As the spheres move away from each other, their mutual effect decreases and eventually tends to zero, resulting in smaller absolute values of the C_D extrema. This observation is supported by a comparison of the current extremum values of C_D with the corresponding values of peak drag coefficients calculated for a single oscillating sphere (at $A/D = 0.5$) and for $Re = 50$ and 100, as detailed in Table 1. It can be seen that for the case when the two spheres are at the maximal distance from one another the current extremum values of C_D are fairly close (1.5% deviation) to the peak drag coefficient values acquired for a single oscillating sphere.

It should be mentioned, that, although significantly alleviated, the insignificant non-physical high-frequency oscillations can again be recognized in the time evolution of the C_D values of the two spheres for $Re \geq 100$. These high-frequency oscillations are a numerical artifact, reappearing as a result of the acceleration of the two spheres[9] being twice as high as that characterizing the previous configuration. As has already been mentioned, the observed high-frequency oscillations can be eliminated either by utilizing grids that are even more dense or by employing a second-order sharp interface formulation of the developed methodology, which will be the focus of our future work.

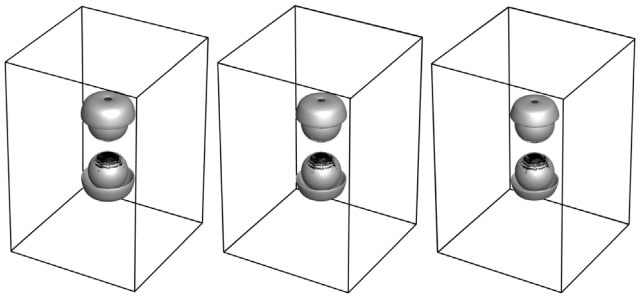
The obtained results were visualized by utilizing the same technique as that used for the previous configuration, i.e., by presenting vortical structures recognized by isosurfaces of the λ_2 criterion corresponding to the value of $\lambda_2 = -0.1$. The isosurfaces are presented for four representative time instances taken over the oscillation period, as shown in Fig. 10. Surprisingly, the inertia of the surrounding flow does not have a strong effect on the intensity of the shedding phenomenon. In fact, despite the higher acceleration characterizing the flow regime of the flow under consideration, the intensity of shedding of the annular vortex structures from the sphere surface is significantly less pronounced, as can be seen in Fig. 10. This is apparently a consequence of the half length of the trajectory of the spheres, which leads to lower peak values of the local fluid flow rate generated by the spheres.

5. Summary and conclusions

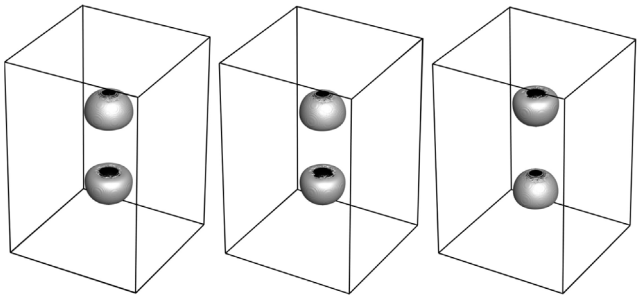
A novel semi-implicit immersed boundary formulation based on the Schur complement approach was developed for the simulation of incompressible flows in the presence of periodically moving immersed bodies. The developed formulation was successfully verified for the flow generated by an oscillating sphere for the entire range of Re values and for the values of $A/D = 0.5$, 1, and 1.5. Being a continuation of our previous study [54], the developed approach comprises a generic methodology that allows us to equip any existing time marching solver of NS equations based on a segregated pressure–velocity coupling (e.g., SIMPLE, fractional step, projection methods and their derivatives) with the IBM functionality. The developed method was applied for simulation of the flows generated by a pair of out-of-phase oscillating spheres. The results of the analysis were presented in terms of the major flow characteristics, including the time evolution of the total forces exerted on the oscillating bodies by the surrounding flow and the qualitative structure of the vortical structures generated by the bodies.

A distinctive feature of the developed methodology is that it is composed of two stages, namely, a pre-computing stage and a time-integration stage. The pre-computing stage of the algorithm is “embarrassingly” parallel, which allows us to boost the computational efficiency without investing any significant additional effort. The parallelism can be intelligently adopted for configurations with stationary or periodically moving immersed bodies. The

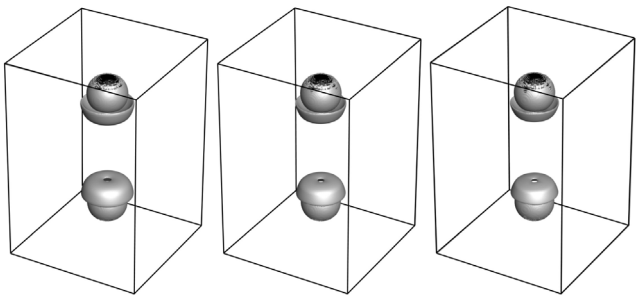
[8] There is no difference in the maximal and minimal absolute values of C_D acquired for the two spheres.
[9] A direct consequence of a double reduction of the A/D value.



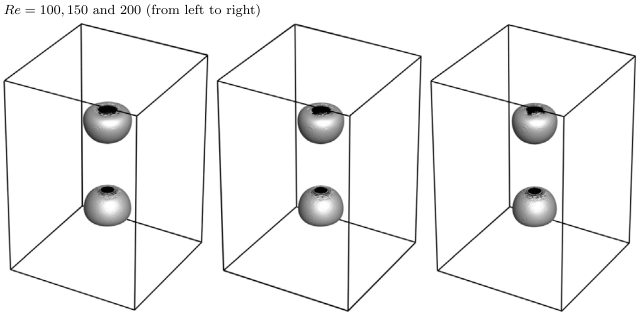
(a) Isosurfaces of the $\lambda_2 = -0.1$ criterion calculated at the lowest point of the sphere trajectory for $Re = 100, 150$ and 200 (from left to right)



(b) Isosurfaces of the $\lambda_2 = -0.1$ criterion calculated at the mid-point of the sphere trajectory on its way up for $Re = 100, 150$ and 200 (from left to right)



(c) Isosurfaces of the $\lambda_2 = -0.1$ criterion calculated at the highest point of the sphere trajectory for $Re = 100, 150$ and 200 (from left to right)



(d) Isosurfaces of the $\lambda_2 = -0.1$ criterion calculated at the mid-point of the sphere trajectory on its way down for $Re = 100, 150$ and 200 (from left to right)

Fig. 10. Typical pattern of vortical structures characterized by the isosurfaces of $\lambda_2 = -0.1$, generated by a pair of out-of-phase transversely oscillating spheres over a single oscillation period calculated for $Re = 100, 150$ and 200 and $A/D = 0.5$.

efficiency of the time integration stage can be further increased taking advantage of the GPFS allowing for extremely fast reading of LU factors pre-computed and saved on a hard disk for each time step over a single oscillation period. Overall, the performance characteristics of the developed methodology for all the configurations simulated in the

framework of the present study can be summarized in terms of the following quantitative data: RAM consumption – no more than 10 GB; hard disk space required for storing LU factors for all the flow fields for 10^3 time steps over a single period – no more than 1.5 TB; typical wall-clock time required for pre-computing a single time step on a single standard Linux server containing 24 cores – order of one hour on $200 \times 200 \times 300$ grids; and typical wall-clock time required for performing time integration of a single time step on a single standard Linux server containing 24 cores – less than 2 s. The strong scaling study was performed and revealed that the pre-computing and time integration stages of the currently developed methodology scale as $\sim Np^{0.63}$ and $\sim Np^{0.27}$, respectively, when the simulations are run on $200 \times 200 \times 300$ grid.

Although the numerical simulations performed in the framework of the current study were restricted to configurations containing a single sphere and a pair of out-of-phase oscillating spheres, it should be stressed that the methodology can be used to simulate a wide spectrum of incompressible flows in the presence of immersed bodies with any periodic kinematics. For example, the method can be straight forwardly applied for the simulation of flows in the presence of bodies characterized by any kind of periodic (including rotating) or undulatory kinematics. This capability is of significant importance for high-fidelity simulations of flows driven by various kinds of rotor machinery, for the undulatory motion of tiny sea creatures and miniaturized robots, and for various biomedical applications in which the flow is driven by peristaltic contraction of the surrounding tissues. All the above applications will be the focus of our future studies, which will include (i) development of a second-order sharp interface formulation of the methodology to enable relaxation of the high grid resolution requirement, and (ii) implementing more efficient parallelizing of both pre-computing and time marching stages by employing hybrid and distributed memory paradigms.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The study was partially funded by GIF-German Israel Foundation, under contract number: I-2471-405.10/2017.

References

[1] S. Lauder, E. Tytell, Hydrodynamics of undulatory propulsion, Fish Biomech. (Fish Physiol.) 23 (2006) 425–468.
[2] D. Bray, Cell Movements: From Molecules to Motility, Garland Sciencem, New York, 1998.
[3] J. Lighthill, Mathematical Biofluid Dynamics, SIAM, Philadelphia, 1975.
[4] C. Brennen, H. Winet, Fluid mechanics of propulsion by cilia and flagella, Ann. Rev. Fluid Mech. 9 (1) (1977) 339–398.
[5] E. Lauga, T. Powers, The hydrodynamics of swimming microorganisms, Rep. Progr. Phys. 72 (2009) 096601.
[6] S. Ebbens, T. Howse, In pursuit of propulsion at the nanoscale, Soft Matter 6 (2010) 726–738.
[7] W. Wang, S. Duan, S. Ahmed, T. Mallouk, A. Sen, Small power: Autonomous nano- and micromotors propelled by self-generated gradients, Nano Today 8 (2013) 531–554.
[8] F. Nadal, E. Lauga, Asymetric steady streaming as a mechanism for acoustic propulsion of rigid bodies, Phys. Fluids 26 (2014) 082001.
[9] L. Lu, K. Ryu, C. Liu, A magnetic microstirrer and array for microfluidic mixing, J. Electromech. Syst. 11 (5) (2002) 462–469.
[10] Y. Huh, T. Park, E. Lee, W. Hong, S. Lee, Development of a fully integrated microfluidic system for sensing infectious viral disease, Electrophoresis 29 (14) (2008) 2960–2969.
[11] L. Capretto, W. Cheng, M. Hill, X. Zhang, Micromixing within microfluidic devices, Top. Curr. Chem. 304 (2011) 27–68.
[12] R. Shamsoddini, M. Sefid, R. Fatehi, ISPH modelling and analysis of fluid mixing in a microchannel with an oscillating or a rotating stirrer, Eng. Appl. Comput. Fluid Mech. 8 (2) (2014) 289–298.
[13] E. Ferrer, A. Willden, A high order discontinuous Galerkin–Fourier incompressible 3D Navier–Stokes solver with rotating sliding meshes, J. Comput. Phys. 231 (2012) 7037–7056.
[14] L. Ramíres, C. Foulquié, X. Nogueira, S. Khelladi, J.-C. Chassaing, I. Colominas, New high-resolution-preserving sliding mesh techniques for higher-order finite volume schemes, J. Comput. Phys. 118 (2015) 114–130.
[15] B. Zhang, C. Liang, A simple, efficient, and high-order accurate curved sliding-mesh interface approach to spectral difference method on coupled rotating and stationary domains, J. Comput. Phys. 295 (2015) 147–160.
[16] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (2) (1972) 252–271.
[17] R. Mittal, G. Iaccarino, The immersed boundary method, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[18] G. Hou, J. Wang, A. Layton, Numerical method for fluid-structure interaction - A review, Commun. Comput. Phys. 12 (2012) 337–377.
[19] W. Huang, F.G. Tian, Recent trends and progress in the immersed boudary method, Proc. Inst. Mech. Eng. C 223 (2019) 7617–7636.
[20] F. Sotiropoulos, X. Yang, Immersed boundary methods for simulating fluid–structure interaction, Prog. Aerosp. Sci. 65 (2014) 1–21.

[21] W. Kim, H. Choi, Immersed boundary methods for fluid-structure interaction: A review, Int. J. Heat Fluid Flow 75 (2019) 301–309.
[22] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: Cad, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (2005) 4135–4195.
[23] R. Sanches, P. Bornemann, F. Cirak, Immersed b-spline (i-spline) finite element method for geometrically complex domains, Comput. Methods Appl. Mech. Engrg. 200 (2011) 1432–1445.
[24] E. Rank, M. Ruess, S. Kollmannsberger, D. Shiillinger, A. Düster, Geometric modeling, isogeometric analysis and the finite cell method, Comput. Methods Appl. Mech. Engrg. 249 (2012) 104–115.
[25] D. Schillinger, L. Dede, M. Scott, J. Evans, M. Borden, E. Rank, T. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Comput. Methods Appl. Mech. Engrg. 249 (2012) 116–150.
[26] D. Kamensky, M.-C. Hsu, D. Shillinger, J. Evans, A. Aggarwal, Y. Basilevs, M. Sacks, T. Hughes, An immersogeometric variational framework for fluidstructure interaction: Application to bioprosthetic heart valves, Comput. Methods Appl. Mech. Engrg. 284 (2015) 1005–1053.
[27] C. Kadapa, W. Dettmer, D. Perić, A fictitious domain/distributed lagrange multiplier based fluid–structure interaction scheme with hierarchical B-spline grids, Comput. Methods Appl. Mech. Engrg. 301 (2016) 1–27.
[28] A. Nitti, J. Kiendl, A. Reali, M. Tullio, An immersed-boundary/isogeometric method for fluid–structure interaction involving thin shells, Comput. Methods Appl. Mech. Engrg. 364 (2020) 112977.
[29] J. Mohd-Yusof, Combined immersed-boundary/b-spline methods for simulations of flow in complex geometries, in: Center for Turbulence Research, Annual Research Briefs, 1997, pp. 317–327.
[30] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2005) 448–476.
[31] S.-Y. Lin, Y.-H. Chin, J.-J. Hu, Y.-C. Chen, A pressure correction method for fluid-particle interraction flow: Direct-forcing method and sedimentation flow, Int. J. Numer. Methods Fluids 67 (2011) 1771–1798.
[32] T. Kempe, J. Fröhlich, An improved immersed boundary method with direct forcing for the simulation of particle laden flows, J. Comput. Phys. 231 (2012) 3663–3684.
[33] W.-P. Breugem, A second-order accurate immersed boundry method for fully resolved siuations of particle-laden flows, J. Comput. Phys. 231 (2012) 4469–4498.
[34] T. Kempe, M. Lennartz, S. Schwarz, J. Fröhlich, Imposing the free-slip condition with a continuous forcing immersed boundary method, J. Comput. Phys. 282 (2015) 183–209.
[35] D. Lo, C.-P. Lee, I.-F. Lin, An efficient immersed boundary method for rluif flow simulations with moving boundaries, Appl. Math. Comput. 328 (2018) 312–337.
[36] H.S. Yoon, D.H. Yu, M.Y. Ha, Y.G. Park, Three-dimensional natural convection in an enclosure with a sphere at different vertical locations, Int. J. Heat Mass Transfer 53 (2010) 3143–3155.
[37] W.W. Ren, C. Shu, W.M. Yang, Boundary condition-enforced immersed boundary method for thermal flow problems with Dirichlet temperature condition and its applications, Comput. Fluid 57 (2012) 40–51.
[38] W. Ren, C. Shu, W. Y., An efficient immersed boundary method for thermal flow problems with heat flux boundary conditions, Int. J. Heat Mass Transfer 64 (64) (2013) 694–705.
[39] Y. Gulberg, Y. Feldman, On laminar natural convection inside multi-layered spherical shells, Int. J. Heat Mass Transfer 91 (2015) 908–921.
[40] Y. Li, E. Jung, W. Lee, H. Lee, K. J., Volume preserving immersed boundary methods for two-phase fluid flows, Internat. J. Numer. Methods Fluids 69 (2012) 842–858.
[41] R. Dillon, M. Owen, K. Painter, A single-cell-based model of multicellular growth using the immersed boundary method, AMS Contemp. Math. 466 (2008) 1–15.
[42] K. Rejniak, An immersed boundary framework for modelling the growth of individual cells: An application to the early tumour development, J. Theoret. Biol. 936 (2007) 186–204.
[43] K. Rejniak, Circulating tumor cells: When a solid tumor meets a fluid microenvironment, Adv. Exp. Med. Biol. 936 (2016) 93–106.
[44] K. Taira, T. Colonius, The immersed boundary method: A projection approach, J. Comput. Phys. 225 (2007) 3121–3133.
[45] B. Kallemov, A. Bhalla, B. Griffith, A. Donev, An immersed boundary method for rigid bodies, Commun. Appl. Math. Comput. Sci. 11 (1) (2016) 79–141.
[46] D. Stein, R. Guy, B. Thomases, Immersed boundary smooth extension: A high-order method for solving PDE on arbitrary smooth domains using Fourier spectral methods, J. Comput. Phys. 304 (2016) 252–274.
[47] S. Liska, T. Colonius, A fast immersed boundary method for external incompressible viscous flows using lattice Green's functions, J. Comput. Phys. 331 (2016) 257–279.
[48] Y. Bao, A. Donev, B. Griffith, D. McQueen, C. Peskin, An immersed boundary method with divergence-free velocity interpolation and force spreading, J. Comput. Phys. 347 (2017) 183–206.
[49] D. Stein, R. Guy, B. Thomases, Immersed boundary smooth extension (IBSE): A high-order method for solving incompressible flows in arbitrary smooth domains, J. Comput. Phys. 335 (2017) 155–178.
[50] Y. Feldman, Y. Gulbeg, An extension of the immersed boundary method based on the distributed Lagrange multiplier approach, J. Comput. Phys. 322 (2016) 248–266.
[51] A. Spizzichino, S. Goldring, Y. Feldman, The immersed boundary method: application to two-phase immiscible flows, Commun. Comput. Phys. 25 (1) (2019) 107–134.
[52] Y. Park, M. Ha, C. Choi, J. Park, Natural convection in a square enclosure with two inner circular cylinders positioned at different vertical locations, Int. J. Heat Mass Transfer 77 (2014) 501–518.

[53] D. Le, B. Khoo, K. Lim, An implicit-forcing immersed boundary method for simulating viscous flows in irregular domains, Comput. Methods Appl. Mech. Engrg. 197 (2008) 2119–2130.
[54] Y. Feldman, Semi-implicit direct forcing immersed boundary method for incompressible viscous thermal flow problems: a Schur complement approach, Int. J. Heat Mass Transfer 127 (2018) 1267–1283.
[55] A. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (1999) 509–534.
[56] S.V. Patankar, D.B. Spalding, A calculation procedure for heat, mass and momentum in three-dimensional parabolic fows, Int. J. Heat Mass Transfer 15 (1972) 1787–1806.
[57] J. Adams, P. Swarztrauber, R. Sweet, FISHPACK: Efficient FORTRAN subprograms for the solution of separable elliptic partial differential equations, 1999, http://adsabs.harvard.edu/abs/2016ascl.soft09004A.
[58] H.G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, Comput. Phys. 12 (6) (1998) 620–631.
[59] H. Vitoshkin, A.Y. Gelfgat, On direct inverse of Stokes, Helmholtz and Laplacian operators in view of time-stepper-based Newton and arnoldi solvers in incompressible CFD, Commun. Comput. Phys. 14 (2013) 1103–1119.
[60] P. Amestoy, I. Duff, J. LÉxcellent, J. Koster, Multifrontal parallel distributed symmetric and unsymmetric solvers, Comput. Methods Appl. Mech. Engrg. 184 (1998) 501–520.
[61] P. Amestoy, I. Duff, J. LÉxcellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM J. Matrix Anal. Appl. 23 (2001) 15–41.
[62] R. Mei, Flow due to an oscillating sphere and an expression for unsteady frag on the sphere at finite Reynolds number, J. Fluid Mech. 270 (1994) 133–174.
[63] H. Blckburn, Mass and momentum transport from a sphere in steady and oscillatory flows, Phys. Fluid 14 (11) (2002) 3997–4011.
[64] J. Jeong, F. Hussain, On the identification of a vortex, J. Fluid Mech. 285 (1995) 69–94.