A fictitious domain method with distributed Lagrange multipliers on adaptive quad/octrees for the direct numerical simulation of particle-laden flows

Can Selçuk, Arthur R. Ghigo, Stéphane Popinet, Anthony Wachs

A fictitious domain method with distributed Lagrange multipliers on adaptive quad/octrees for the direct numerical simulation of particle-laden flows

Can Selçuk[c,*], Arthur R. Ghigo[a,c,d], Stephane Popinet[b], Anthony Wachs[c,d]

[a] *PIMS-CNRS, University of British Columbia, 4176-2207 Main Mall, Vancouver, BC, V6T 1Z4, Canada*
[b] *Institut Jean le Rond d'Alembert, Sorbonne Université, CNRS, Paris, France*
[c] *Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, BC, Canada, V6T 1Z4.*
[d] *Departement of Chemical and Biological Engineering, University of British Columbia, 2360 East Mall, Vancouver, BC Canada, V6T 1Z3.*

Abstract

In this article, we extend our Distributed Lagrange Multiplier/Fictitious Domain method previously implemented on simple regular Cartesian grids to quadtree/octree adaptive grids. The objective is to improve both the accuracy and efficiency of our DLM/FD particle-resolved simulation method by extending its computing capabitilies through dynamic local mesh refinement. The main features of our numerical method, such as a first-order operator splitting time algorithm and a second-order reconstruction of the velocity field close to the boundary of the immersed rigid bodies (of arbitrary shape), are unchanged. We implemented our adaptive DLM/FD algorithm within Basilisk, a parallel platform to solve partial differential equations on dynamic quadtree/octree grids. The quadtree/octree structure of the grid and specific design rules of Basilisk impose a special treatment of some of the operations performed on the grid in the DLM/FD-Uzawa algorithm. The new computational method is then tested and validated on a set of flow configurations including the challenging problem of accurately computing lubrication interaction forces without resorting to using any ad hoc correction. Finally, we illustrate the potential of our code to compute complex particle-laden flow configurations that were not attainable in the past with a DLM/FD algorithm implemented on a simple regular Cartesian grid.

Keywords: Fictitious domain; Adaptive grid; Quadtree/octree; Particle-laden flow; Parallel computing;

1 Introduction

In this article, we combine a fictitious domain method (FD) with an adaptive mesh refinement (AMR) technique to compute the dynamics of particle-laden flows. We consider a Newtonian and incompressible fluid seeded with rigid and homogeneous particles of, possibly, complex shape. Such flows have many engineering and practical applications. For example, fluidized beds are a common process in the chemical engineering industry where the dynamics at the level of the particles including both short and long range hydrodynamic interactions controls the overall chemical conversion efficiency. Pollutant transport

*Corresponding author
Email address: cselcuk@math.ubc.ca (Can Selçuk)

with inertial particles is also an important category of application in the aerodynamic and environmental fields.

Particle-laden flows have been the subject of many investigations and efforts for the past decades and a broad range of numerical methods has been devised in the literature to compute them. In general, depending on the treatment of the underlying numerical mesh, these methods are classified into two main categories: body-conforming mesh methods and fixed mesh methods (Haeri and Shrimpton, 2012). The former category comprises (among others) the Arbitrary Lagrangian-Eulerian (ALE) (Choi and Kim, 2010), the Fictitious Boundary (Wan and Turek, 2006) and the Deforming-Spatial-Domain/Stabilized Space-Time (DSD/STT) (Tezduyar et al., 1992) methods while the latter category comprises the Immersed Boundary (IBM) (Peskin, 1972), the Lattice-Boltzmann (LBM) (Ladd, 1994) and the Distributed Lagrange Multiplier/Fictitious Domain (DLM/FD) (Glowinski et al., 1999) methods. For a comprehensive overview of existing particle-resolved computational methods, the reader is referred to the recent review articles by Haeri and Shrimpton (2012), Sotiropoulos and Yang (2014) or Wachs (2019).

Body-conforming mesh methods, while more appealing at first due to their theoretical higher accuracy are hindered by the numerical overhead associated with the resolution of additional equations for the motion of the mesh, the complicated re-meshing process itself and the costly projection steps between successive meshes when the rigid bodies position changes in time. Conversely, fixed-mesh methods are theoretically less accurate as they rely on a local reconstruction of the velocity field around the solid region (usually through an *ad hoc* interpolation strategy) but they are more convenient in practice and provide a better framework for parallel implementation and computational optimized speed-ups. Indeed, domain decomposition is straightforward and can be achieved with an almost perfect load balancing. The serial algorithms, once parallelized, also retain their convergence properties and scale well with the number of cores/CPUs when deployed on supercomputers, see e.g. Uhlmann (2005) or Wachs (2011).

Existing particle-resolved computational methods have shed some light on the hydrodynamics of highly complex particle-fluid systems. However, our comprehension of elementary mechanisms is still lacking for flow configurations with a high degree of scale separation throughout the spatial domain and/or the temporal domain such as turbulent flows where the smallest structures scale with the Reynolds number as $\sim Re^{-3/4}$ or highly dense suspensions where lubrication forces play a major role. For such flows, a fixed/static grid strategy and/or a constant time-stepping technique are not adequate to capture the relevant length and time scales (and thus the correct dynamics) of the flow. To overcome these difficulties, solvers with adaptive mesh refinement (AMR) have emerged and have been successfully employed in different fields: electrohydrodynamics (López-Herrera et al., 2011), study of atmospheric boundary layers (van Hooft et al., 2018), multiphase flows (Fuster et al., 2009; Losasso et al., 2004), flows in complex geometries (Popinet, 2003; Gibou et al., 2002), turbulence modeling (Schneider and Vasilyev, 2010). In general, AMR solvers aim to distribute available computational resources efficiently over a domain by dynamically refining and coarsening the computational grid in space and time. It is thus natural to use adaptive grids and adaptive time-steps with the aforementioned particle-resolved computational methods in order to simulate particle-laden flows.

Some fictitious domain/immersed boundary methods were designed from the start for a potential implementation on adaptive grids and some early contributions to the literature as, e.g., Johansen and Colella (1998), Calhoun and LeVeque (2000) or Popinet (2003) actually implemented them on adaptive grids and showed satisfactory results, even with the limited computing power available at that time. With the expansion of supercomputing, the contemporary literature features recent contributions to particle-laden flow computations on adaptive grids that are more advanced from a computational viewpoint and that

examine more challenging flow configurations: Mohaghegh and Udaykumar (2016) with an immersed-boundary/ghost cell method, Hartmann et al. (2011), Meinke et al. (2013) and Schneiders et al. (2013) with a cut-cell/embedded geometry method and Eitel-Amor et al. (2013) with a lattice-Boltzmann method. But the use of adaptive grids in particle-laden flow computations remains limited in the literature and, to the best of the authors' knowledge, there has been no attempt to port and implement a DLM/FD method on adaptive grids. Our primary objective in this paper is therefore to combine the powerful features of the DLM/FD method (unconditional stability with respect to the solid to fluid density ratio, enhanced fluid-solid coupling through the implicit computation of the hydrodynamic interactions and robust convergence properties of the Uzawa algorithm) to the improved spatial accuracy enabled by adaptive grids.
In practice, we implemented our DLM/FD method (Wachs, 2011; Wachs et al., 2015) in the Basilisk AMR framework (Popinet, 2015). Basilisk is an open-source software that provides a set of (multigrid) solvers based on a tree data-structure to solve a broad range of initial-boundary value problems.
It is coded in an "augmented" C programming language (the Basilisk C) and features high level of abstraction for all non-trivial operations such as grid traversal, mesh refinement and coarsening. The implementation of new models and/or new numerical schemes is also eased by the existence of local regular stencils for each cell of the computational domain (even when the cell of interest has neighboring cells of different sizes).
The paper is organized as follows. In section 2 we present the governing equations of the problem and the numerical schemes are presented in section 3. A set of specific rules for the implementation of the fictitious domain method are detailed in section 4 and the algorithm is given in appendix A. We then present a set of validation cases in section 5, including problems at high Reynolds numbers or with angular rigid particles. In section 6, we attempt to capture short-range lubrication forces in a direct fashion for the cases of a sphere approaching a wall or two spheres approaching each other in a simple shear flow. Finally, we discuss the potential of our DLM/FD algorithm on adaptive grids to compute a large range of challenging particle-laden flows that are hard or impossible to compute with an implementation on a regular Cartesian grid and future work in section 8.

2 Governing equations

To describe the behavior of a solid rigid body immersed in a fluid, the fictitious domain method represents the space occupied by this body as a fluid domain and enforces this "fictitious" fluid domain to behave as a rigid body. We consider a computational domain Ω filled with an incompressible Newtonian fluid of constant viscosity μ_f and constant density ρ_f and seeded with N rigid (homogeneous) particles of constant density ρ_s. The N rigid particles occupy the domain $P = \bigcup_{i=1}^{N} P_i$ while the fluid occupies the domain $\Omega \backslash P$ such that $\Omega \backslash P \bigcap P = 0$. We denote Γ the boundary of Ω and assume for simplicity (but without any loss of generality) that the fluid velocity $\boldsymbol{u}(\boldsymbol{x}, t)$ satisfies Dirichlet boundary conditions on Γ. We briefly recall the equations of motion for the fluid and the solid separately and then present the so-called *combined equation of motion* for the fluid-particle mixture, as originally introduced by Glowinski et al. (1999).

2.1 Equations for the fluid

The starting point of the method is to consider first the motion of the fluid that satisfies:

$$\rho_f \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} \right) = \rho_f \boldsymbol{g} + \nabla \cdot \boldsymbol{\sigma} \quad \text{in } \Omega \backslash P, \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \quad \text{in } \Omega \backslash P, \tag{2}$$

where $\boldsymbol{\sigma}$ is the (general) stress tensor and $\boldsymbol{g}$ the gravity acceleration. The boundary and initial conditions are

$$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\Gamma}} \quad \text{on } \Gamma, \tag{3}$$

$$\boldsymbol{u} = \boldsymbol{U_i} + \boldsymbol{\omega_i} \times \boldsymbol{r_i} \quad \text{on } \partial P_i \text{ for } i = 1, ..., N, \tag{4}$$

$$\boldsymbol{u}|_{t=0} = \boldsymbol{u_0} \quad \text{in } \Omega \backslash P. \tag{5}$$

Equation (4) and quantities $\boldsymbol{U_i}, \boldsymbol{\omega_i}$, $\boldsymbol{r_i}$ are directly related to the particles through the no-slip condition at the particles surface. Here, $\boldsymbol{U_i}$ refers to the ith-particle's translational velocity, $\boldsymbol{\omega_i}$ to the ith-particle's angular velocity and $\boldsymbol{r_i} \equiv \boldsymbol{x} - \boldsymbol{X_i}$ to the distance with respect to the ith-particle's center of mass position $\boldsymbol{X_i}$.

2.2 Equations for the particle

The equations of motion for the ith-particle are

$$M_i \frac{\mathrm{d}\boldsymbol{U_i}}{\mathrm{d}t} = M_i \boldsymbol{g} + \boldsymbol{F_i} + \boldsymbol{F_i'} \tag{6}$$

$$\frac{\mathrm{d}\boldsymbol{I_i}\boldsymbol{\omega_i}}{\mathrm{d}t} = \boldsymbol{T_i} + \boldsymbol{T_i'}, \tag{7}$$

where M_i and $\boldsymbol{I_i}$ denote the ith-particle's mass and inertia tensor, respectively. The vectors $\boldsymbol{F_i}$ and $\boldsymbol{T_i}$ are the hydrodynamic force and torque (about the center of mass) exerted on the ith-particle and read as follows:

$$\boldsymbol{F_i} = \int_{\partial P_i} \boldsymbol{\sigma} \cdot \hat{\boldsymbol{n}} \, dS, \quad \boldsymbol{T_i} = \int_{\partial P_i} \boldsymbol{r_i} \times (\boldsymbol{\sigma} \cdot \hat{\boldsymbol{n}}) \, dS, \tag{8}$$

where $\hat{\boldsymbol{n}}$ denotes the outward-oriented unit normal vector to ∂P_i. The force $\boldsymbol{F_i'}$ and torque $\boldsymbol{T_i'}$ result from potential collisions of the ith-particle with other particles and walls:

$$\boldsymbol{F_i'} = \sum_{j=1, i \neq j}^{N} \boldsymbol{F_{c,ij}} + \boldsymbol{F_{c,iw}}, \tag{9}$$

$$\boldsymbol{T_i'} = \sum_{j=1, i \neq j}^{N} \boldsymbol{r_j} \times \boldsymbol{F_{c,ij}} + \boldsymbol{r_w} \times \boldsymbol{F_{c,iw}}. \tag{10}$$

2.3 Equation for the fluid-particle mixture

The equation for the fluid-particle mixture, also called the *combined equation of motion*, is obtained by combining the weak formulations of the equations of motion of the fluid and of the rigid particles. It is obtained by first writing the weak formulation for the fluid domain $\Omega \backslash P$ where the rigid body constraint is enforced on the particles surface $\partial P = \bigcup_{i=1}^{N} \partial P_i$ only and then extending the weak formulation to the whole domain Ω by imposing the rigid body motion constraint in the whole region P occupied by the particles. We recall briefly the steps of the derivation below.

2.3.1 Weak formulation for the fluid domain $\Omega \backslash P$

To obtain the weak formulation in the fluid domain $\Omega \backslash P$, we first introduce the test functions $(\boldsymbol{v}, \boldsymbol{V}_i, \boldsymbol{\xi}_i)$ for $(\boldsymbol{u}, \boldsymbol{U}_i, \boldsymbol{\omega}_i)$ with $i = 1, \ldots, N$ belonging to the following function spaces:

$$(\boldsymbol{v}, \boldsymbol{V}_i, \boldsymbol{\xi}_i) \in \left\{ (\boldsymbol{v}, \boldsymbol{V}_i, \boldsymbol{\xi}_i) \mid \boldsymbol{v} \in H^1(\Omega \backslash P)^3, \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma, (\boldsymbol{V}_i, \boldsymbol{\xi}_i) \in R^3, \boldsymbol{v} = \boldsymbol{V}_i + \boldsymbol{\xi}_i \times \boldsymbol{r}_i \text{ on } \partial P_i \right\},$$

$$(\boldsymbol{u}, \boldsymbol{U}_i, \boldsymbol{\omega}_i) \in \left\{ (\boldsymbol{u}, \boldsymbol{U}_i, \boldsymbol{\omega}_i) \mid \boldsymbol{u} \in H^1(\Omega \backslash P)^3, \boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\Gamma}} \text{ on } \Gamma, (\boldsymbol{U}_i, \boldsymbol{\omega}_i) \in R^3, \boldsymbol{u} = \boldsymbol{U}_i + \boldsymbol{\omega}_i \times \boldsymbol{r}_i \text{ on } \partial P_i \right\}.$$

We then perform the following symbolic operation:

$$\int_{\Omega\backslash P} \text{Equation}(1)\cdot\boldsymbol{v}\,\mathrm{d}x + \sum_{i=1}^{N}\text{Equation}(6)\cdot\boldsymbol{V}_i + \sum_{i=1}^{N}\text{Equation}(7)\cdot\boldsymbol{\xi}_i. \tag{11}$$

After integrating by part, using the fact that $\boldsymbol{v}=\boldsymbol{V}_i+\boldsymbol{\xi}_i\times\boldsymbol{r}_i$ on ∂P_i, for $i=1,\ldots,N$, $\boldsymbol{v}=0$ on Γ and that the stress tensor $\boldsymbol{\sigma}$ is symmetric, we find:

$$\begin{aligned}&\int\limits_{\Omega\backslash P}\rho_f\left(\frac{\partial\boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}-\boldsymbol{g}\right)\cdot\boldsymbol{v}\,\mathrm{d}\boldsymbol{x}+\sum_{i=1}^{N}M_i\left(\frac{\mathrm{d}\boldsymbol{U}_i}{\mathrm{d}t}-\boldsymbol{g}\right)\cdot\boldsymbol{V}_i+\sum_{i=1}^{N}\frac{\mathrm{d}\boldsymbol{I}_i\boldsymbol{\omega}_i}{\mathrm{d}t}\cdot\boldsymbol{\xi}_i\\&-\sum_{i=1}^{N}\boldsymbol{F}_i'\cdot\boldsymbol{V}_i-\sum_{i=1}^{N}\boldsymbol{T}_i'\cdot\boldsymbol{\xi}_i=-\int\limits_{\Omega\backslash P}\boldsymbol{D}\left[\boldsymbol{v}\right]:\boldsymbol{\sigma}\,\mathrm{d}\boldsymbol{x},\quad\text{for all }\left(\boldsymbol{v},\boldsymbol{V}_i,\boldsymbol{\xi}_i\right),\end{aligned}\tag{12}$$

where $\boldsymbol{D}\left[\boldsymbol{v}\right]=\frac{1}{2}(\boldsymbol{\nabla}\boldsymbol{v}+(\boldsymbol{\nabla}\boldsymbol{v})^T)$ denotes the strain rate tensor. The weak incompressibility constraint is simply given by:

$$\int\limits_{\Omega\backslash P}q\boldsymbol{\nabla}\cdot\boldsymbol{u}\,\mathrm{d}\boldsymbol{x}=0,\ \text{for all } q\in L^2\left(\Omega\backslash P\right).\tag{13}$$

Note that the hydrodynamic forces $\boldsymbol{F}_i$ and torques $\boldsymbol{T}_i$ on the particle are completely eliminated in equation (12).

2.3.2 Extension to the entire domain Ω: fictitious domain formulation

Equation (12) enforces the rigid body motion on the particles surface ∂P only. To extend this constraint to the whole particle domain P, we modify the function spaces as follows:

$$\begin{aligned}&\left(\boldsymbol{v},\boldsymbol{V}_i,\boldsymbol{\xi}_i\right)\ \in\left\{\left(\boldsymbol{v},\boldsymbol{V}_i,\boldsymbol{\xi}_i\right)\ |\ \boldsymbol{v}\in H^1\left(\Omega\right)^3,\boldsymbol{v}=\boldsymbol{0}\text{ on }\Gamma,\left(\boldsymbol{V}_i,\boldsymbol{\xi}_i\right)\in R^3,\boldsymbol{v}=\boldsymbol{V}_i+\boldsymbol{\xi}_i\times\boldsymbol{r}_i\text{ in }P_i\right\},\\&\left(\boldsymbol{u},\boldsymbol{U}_i,\boldsymbol{\omega}_i\right)\in\left\{\left(\boldsymbol{u},\boldsymbol{U}_i,\boldsymbol{\omega}_i\right)\ |\ \boldsymbol{u}\in H^1\left(\Omega\right)^3,\boldsymbol{u}=\boldsymbol{u}_{\boldsymbol{\Gamma}}\text{ on }\Gamma,\left(\boldsymbol{U}_i,\boldsymbol{\omega}_i\right)\in R^3,\boldsymbol{u}=\boldsymbol{U}_i+\boldsymbol{\omega}_i\times\boldsymbol{r}_i\text{ in }P_i\right\}.\end{aligned}$$

Deriving $\boldsymbol{u}=\boldsymbol{U}_i+\boldsymbol{\omega}_i\times\boldsymbol{r}_i$ in P_i for $i=1,\ldots,N$ with respect to time leads to the following relationship:

$$\frac{\partial\boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}=\frac{D\boldsymbol{u}}{Dt}=\frac{\mathrm{d}\boldsymbol{U}_i}{\mathrm{d}t}+\frac{\mathrm{d}\boldsymbol{\omega}_i}{\mathrm{d}t}\times\boldsymbol{r}_i+\boldsymbol{\omega}_i\times\left(\boldsymbol{\omega}_i\times\boldsymbol{r}_i\right)\ \text{ in } P_i,\ i=1,\ldots,N.\tag{14}$$

Next, to derive the weak formulation for the particle domain $P=\bigcup_{i=1}^{N}P_i$, we multiply equation (14) by ρ_{s_i} and integrate it for each particle P_i, $i=1,\ldots,N$. Then we use the facts that $\int\limits_{P_i}\boldsymbol{r}_i\mathrm{d}\boldsymbol{x}=\boldsymbol{0}$, $\int\limits_{P}\boldsymbol{\sigma}:\boldsymbol{D}\left[\boldsymbol{v}\right]\mathrm{d}\boldsymbol{x}=0$, $\int\limits_{P_i}\rho_{s_i}\boldsymbol{g}\boldsymbol{v}\mathrm{d}\boldsymbol{x}=M_i\boldsymbol{g}\boldsymbol{V}_i$ and sum all weak formulations of each particle P_i to obtain a formulation similar to (12) as follows:

$$\begin{aligned}&\int\limits_{P}\rho_f\left(\frac{\partial\boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}-\boldsymbol{g}\right)\cdot\boldsymbol{v}\,\mathrm{d}\boldsymbol{x}-\sum_{i=1}^{N}\frac{\rho_f}{\rho_{s_i}}M_i\left(\frac{\mathrm{d}\boldsymbol{U}_i}{\mathrm{d}t}-\boldsymbol{g}\right)\cdot\boldsymbol{V}_i\\&-\sum_{i=1}^{N}\frac{\rho_f}{\rho_{s_i}}\frac{\mathrm{d}\boldsymbol{I}_i\boldsymbol{\omega}_i}{\mathrm{d}t}\cdot\boldsymbol{\xi}_i=-\int\limits_{P}\boldsymbol{D}\left[\boldsymbol{v}\right]:\boldsymbol{\sigma}\,\mathrm{d}\boldsymbol{x},\quad\text{for all }\left(\boldsymbol{v},\boldsymbol{V}_i,\boldsymbol{\xi}_i\right).\end{aligned}\tag{15}$$

Summing equations (15) and (12) yields the so-called *combined weak equation of motion for the entire domain* Ω.

$$\begin{aligned}&\int\limits_{\Omega}\rho_f\left(\frac{\partial\boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}-\boldsymbol{g}\right)\cdot\boldsymbol{v}\,\mathrm{d}\boldsymbol{x}+\int\limits_{\Omega}\boldsymbol{\sigma}:\boldsymbol{D}\left[\boldsymbol{v}\right]\,\mathrm{d}\boldsymbol{x}+\sum_{i=1}^{N}\left(\left(1-\frac{\rho_f}{\rho_{s_i}}\right)M_i\left(\frac{\mathrm{d}\boldsymbol{U}_i}{\mathrm{d}t}-\boldsymbol{g}\right)-\boldsymbol{F}_i'\right)\cdot\boldsymbol{V}_i\\&+\sum_{i=1}^{N}\left(\left(1-\frac{\rho_f}{\rho_{s_i}}\right)\frac{\mathrm{d}\boldsymbol{I}_i\boldsymbol{\omega}_i}{\mathrm{d}t}-\boldsymbol{T}_i'\right)\cdot\boldsymbol{\xi}_i=\boldsymbol{0},\quad\text{for all }\left(\boldsymbol{v},\boldsymbol{V}_i,\boldsymbol{\xi}_i\right).\end{aligned}\tag{16}$$

2.3.3 Relaxing the rigid-body constraint: distributed Lagrange multipliers

We now relax the constraint of rigid-body motion through the introduction of distributed Lagrange multipliers $\boldsymbol{\lambda}$. We look for solutions in the function spaces defined as:

$$\mathcal{W}_\Gamma = \{\boldsymbol{v} \in H^1(\Omega)^3 \mid \boldsymbol{v} = \boldsymbol{u_\Gamma} \text{ on } \Gamma\}, \quad \mathcal{W}_0 = \{\boldsymbol{v} \in H^1(\Omega)^3 \mid \boldsymbol{v} = 0 \text{ on } \Gamma\},$$
$$\mathcal{L}^2_0 = \left\{ q \in \mathcal{L}^2(\Omega)^3 \mid \int q \mathrm{d}\boldsymbol{x} = 0 \right\}, \ \Lambda = H^1(P)^3.$$

and solve a constrained optimization problem which reads, for a Newtonian fluid with $\boldsymbol{\sigma} = -p\boldsymbol{I}_d + 2\mu_f \boldsymbol{D}[\boldsymbol{u}]$, as follows: find $\boldsymbol{u} \in \mathcal{W}_\Gamma$, $p \in \mathcal{L}^2_0$, $\boldsymbol{\lambda} \in \Lambda$, $(\boldsymbol{U}_i, \boldsymbol{\omega}_i) \in \boldsymbol{R}^3 \times \boldsymbol{R}^3$ such that

$$\int_\Omega \rho_f \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla}) \boldsymbol{u} \right) \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} - \int_\Omega p \boldsymbol{\nabla} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \int_\Omega \mu_f \boldsymbol{\nabla} \boldsymbol{u} : \boldsymbol{\nabla} \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} = -\sum_{i=1}^{N} \int_{P_i} \boldsymbol{\lambda} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} \tag{17}$$

$$\int_\Omega -q \boldsymbol{\nabla} \cdot \boldsymbol{u} \, \mathrm{d}\boldsymbol{x} = 0, \tag{18}$$

for particle $i = 1, \ldots, N$

$$\left(1 - \frac{\rho_f}{\rho_{s_i}}\right) M_i \left(\frac{\mathrm{d}\boldsymbol{U_i}}{\mathrm{d}t} - \boldsymbol{g} \right) \cdot \boldsymbol{V_i} - \boldsymbol{F_i'} \cdot \boldsymbol{V_i} = \int_{P_i} \boldsymbol{\lambda} \cdot \boldsymbol{V_i} \, \mathrm{d}\boldsymbol{x}, \tag{19}$$

$$\left(1 - \frac{\rho_f}{\rho_{s_i}}\right) \frac{\mathrm{d}\boldsymbol{I_i}\boldsymbol{\omega_i}}{\mathrm{d}t} \cdot \boldsymbol{\xi_i} - \boldsymbol{T_i'} \cdot \boldsymbol{\xi_i} = \int_{P_i} \boldsymbol{\lambda} \cdot (\boldsymbol{\xi_i} \times \boldsymbol{r}) \, \mathrm{d}\boldsymbol{x}, \tag{20}$$

$$\int_{P_i} \boldsymbol{\nu} \cdot (\boldsymbol{u} - (\boldsymbol{U_i} + \boldsymbol{\omega_i} \times \boldsymbol{r}_i)) \, \mathrm{d}\boldsymbol{x} = 0. \tag{21}$$

for all $\boldsymbol{v} \in \mathcal{W}_0$, $(\boldsymbol{V_i}, \boldsymbol{\xi_i}) \in \boldsymbol{R}^3 \times \boldsymbol{R}^3$, $\boldsymbol{\nu} \in \Lambda$, $q \in \mathcal{L}^2(\Omega)$

Note that the term $-\int_\Omega \rho_f \boldsymbol{g} \, \mathrm{d}\boldsymbol{x}$ is now included in the pressure term. Also, the test functions $\boldsymbol{V_i}$ and $\boldsymbol{\xi_i}$ are simply chosen such that $\boldsymbol{V_i} = \boldsymbol{\xi_i} = \boldsymbol{1}$ in $P_i \cup \partial P_i$ for $i = 1, \ldots, N$.

3 Numerical method

In this section, we present the numerical method we use to solve the set of equations (17)-(21).

3.1 Time discretization: 1st order operator splitting

The set of equations (17) - (21) forms a fully coupled problem that is difficult to solve directly as one has to deal with multiple sources of difficulty (Glowinski et al., 1999):

- an advection-diffusion equation,
- the constraint of incompressibility with the pressure p as unknown,
- the detection and computation of the contact forces $\boldsymbol{F_i'}$ and torques $\boldsymbol{T_i'}$ for particle $i = 1, \ldots, N$,
- the constraint of rigid body motion with the Lagrange multipliers $\boldsymbol{\lambda}$ as unknown.

Following Glowinski et al. (1999), we adopt a first-order accurate operator-splitting method (or fractional step method *a la* Marchuk-Yanenko) and split the whole problem into multiple sub-problems that we solve successively. The process can be seen as an initial value problem such that:

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} + \mathbf{A}_1(\phi) + \mathbf{A}_2(\phi) + \mathbf{A}_3(\phi) = f$$
$$\phi(t = 0) = \phi_0.$$

The definition of operators $\boldsymbol{A}_i$ is not unique and leads to different splitting strategies (see, e.g., Glowinski et al. (2001) and Yu (2005)). We choose here a 3-step splitting strategy defined as follows:

$$\begin{aligned}
&\frac{\phi^{n+1/3}-\phi^{n}}{\Delta t}+\mathbf{A}_1\left(\phi^{n+1/3}\right)=f_1^{n+1},\\
&\frac{\phi^{n+2/3}-\phi^{n+1/3}}{\Delta t}+\mathbf{A}_2\left(\phi^{n+2/3}\right)=f_2^{n+1},\\
&\frac{\phi^{n+1}-\phi^{n+2/3}}{\Delta t}+\mathbf{A}_3\left(\phi^{n+1}\right)=f_3^{n+1},\\
&f_1^{n+1}+f_2^{n+1}+f_3^{n+1}=f\left((n+1)\Delta t\right).
\end{aligned}$$

The operator $\boldsymbol{A}_1$ refers (symbolically) to the Navier-Stokes problem where the incompresibility condition and the advection-diffusion equation are treated all together. The operator $\boldsymbol{A}_2$ represents a purely granular problem (i.e without the action of the fluid) where the particle-positions are updated and their intermediate velocities are found by computing the contact forces $\boldsymbol{F}_i'$, contact torques $\boldsymbol{T}_i'$ and by solving Newton's equations of motion. Finally, the operator $\boldsymbol{A}_3$ refers to the fictitious domain problem where the constraint of rigid body motion is enforced by solving a saddle-point problem for the particles velocities $(\boldsymbol{U}_i, \boldsymbol{\omega}_i)$, fluid velocity $\boldsymbol{u}$ and Lagrange multipliers $\boldsymbol{\lambda}$. Note that while this splitting algorithm is first-order accurate only, it is proven to be robust, stable and also to preserve stationary solutions (MacNamara and Strang, 2016). From a practical point of view, it offers flexibility to select an efficient solver for each sub-problem and is rather simple to implement. Here, we choose to use the (open-source) Basilisk code for the solution of the Navier-Stokes problem, and the Grains3D code for the granular problem. Basilisk is chosen for its adaptive mesh refinement (AMR) capabilities along with the good convergence properties of its geometric multi-grid solver while Grains3D is chosen for its appealing capability to handle rigid particles with a complex shape. For more detail on these codes, see Popinet (2015) about Basilisk and Wachs et al. (2012); Rakotonirina et al. (2019) about Grains3D. For the fluid-particle interaction problem, we implemented the iterative Uzawa/Conjugate gradient algorithm in Basilisk and coupled it with Grains3D. In the next sub-sections, we present the temporal discretization scheme of each sub-problem (in their strong formulation for the sake of simplicity). The details and practical aspects of the Uzawa algorithm are presented in the appendix A.

3.1.1 First sub-problem: Navier-Stokes

The temporal discretization of the Navier-Stokes equations implemented in Basilisk uses an operator-splitting method similar to the one proposed by (Bell et al., 1989), and is

composed of the following intermediate steps (or *events* in Basilisk's terminology) :

$$\rho_f \left(\frac{\boldsymbol{u_a} - \boldsymbol{u}^n}{\Delta t} + [\boldsymbol{u} \cdot \boldsymbol{\nabla u}]^{n+1/4} \right) = \boldsymbol{0}, \tag{22}$$

$$\rho_f \left(\frac{\boldsymbol{u_r} - \boldsymbol{u_a}}{\Delta t} \right) = + \left(-\boldsymbol{\nabla} p^n + \rho \boldsymbol{a}^n \right), \tag{23}$$

$$\rho_f \left(\frac{\boldsymbol{u_\lambda} - \boldsymbol{u_r}}{\Delta t} \right) = -\boldsymbol{\lambda}^{\boldsymbol{n}} \tag{24}$$

$$\rho_f \left(\frac{\boldsymbol{u_v} - \boldsymbol{u_\lambda}}{\Delta t} \right) = \boldsymbol{\nabla} \cdot \left(2\mu_f \boldsymbol{D} \left[\boldsymbol{u_v} \right] \right), \tag{25}$$

$$\rho_f \left(\frac{\boldsymbol{u_*} - \boldsymbol{u_v}}{\Delta t} \right) = - \left(-\boldsymbol{\nabla} p^n + \rho \boldsymbol{a}^n \right) \tag{26}$$

$$\rho_f \left(\frac{\boldsymbol{u}^{n+1/2} - \boldsymbol{u_*}}{\Delta t} \right) = -\boldsymbol{\nabla} p^{n+1/2} + \rho \boldsymbol{a}^{n+1/2}, \tag{27}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{u}^{n+1/2} = \boldsymbol{0}. \tag{28}$$

Equation (22) describes a pure advection problem. The convective term $[\boldsymbol{u} \cdot \boldsymbol{\nabla u}]^{n+1/4}$ is computed with an explicit and second-order accurate Godunov prediction procedure detailed in Bell et al. (1989). Equation (25) is a purely viscous problem which is solved implicitly for $\boldsymbol{u_v}$ and equations (27) to (28) are the projection of the velocity field onto a divergence-free space. These two steps involve solving Helmholtz problems by inverting a Laplacian operator. The term $\boldsymbol{\nabla} p^n + \boldsymbol{a}^n$ in equation (23) is a coupling term from the previous time step that improves the accuracy of the intermediate solutions. It is subtracted in equation (26) and therefore vanishes when all the equations are summed. Equation (24) is added here as an explicit forcing term to improve the coupling between the Navier-Stokes and fictitious domain problems (Yu et al., 2006; Wachs et al., 2015). It is also subtracted when solving the fictitious domain problem in equation (35). At the end of the sequence (when all the equations are summed), the global temporal scheme for the Navier-Stokes problem is:

$$\frac{\boldsymbol{u}^{n+1/2} - \boldsymbol{u}^n}{\Delta t} = - \left[\boldsymbol{u} \cdot \boldsymbol{\nabla u} \right]^{n+1/4} + \frac{1}{\rho_f} \left[\boldsymbol{\nabla} \cdot \left(2\mu_f \boldsymbol{\mathcal{D}_v} \left[\boldsymbol{u}^{n+1/2} \right] \right) - \boldsymbol{\nabla} p^{n+1/2} + \boldsymbol{a}^{n+1/2} - \boldsymbol{\lambda}^n \right], \tag{29}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{u}^{n+1/2} = \boldsymbol{0}. \tag{30}$$

3.1.2 Second sub-problem: granular problem

The second sub-problem is a pure granular problem which reads as follows: given particles velocity at time t^n, $\boldsymbol{U}_i^n, \boldsymbol{\omega}_i^n$, find $\boldsymbol{U}_i^{n+1/2}, \boldsymbol{\omega}_i^{n+1/2}$ for particle $i = 1, \ldots, N$ such that

$$\left(1 - \frac{\rho_f}{\rho_s} \right) M_i \left(\frac{\boldsymbol{U}_i^{n+1/2} - \boldsymbol{U}_i^n}{\Delta t} \right) = \left(1 - \frac{\rho_f}{\rho_s} \right) M_i \boldsymbol{g} + \boldsymbol{F_i'} \tag{31}$$

$$\left(1 - \frac{\rho_f}{\rho_s} \right) \frac{\boldsymbol{I}_i^{n+1/2} \boldsymbol{\omega}_i^{n+1/2} - \boldsymbol{I}_i^n \boldsymbol{\omega}_i^n}{\Delta t} = \boldsymbol{T_i'}. \tag{32}$$

and update particles center of mass position and particles angular position accordingly. Note that in practice we solve equation (32) in the body-fixed frame of reference and hence replace $\frac{\mathrm{d}\boldsymbol{I}_{i,b}\boldsymbol{\omega}_{i,b}}{\mathrm{d}t}$ by $\boldsymbol{I}_{i,b}\frac{\mathrm{d}\boldsymbol{\omega}_{i,b}}{\mathrm{d}t} + \boldsymbol{\omega}_{i,b} \times \boldsymbol{I}_{i,b}\boldsymbol{\omega}_{i,b}$ where the subscript b means "in the body-fixed frame of reference". In the body-fixed frame of reference and with an explicit treatment of the non-linear term $\boldsymbol{\omega}_{i,b} \times \boldsymbol{I}_{i,b}\boldsymbol{\omega}_{i,b}$, equation (32) becomes:

$$\left(1 - \frac{\rho_f}{\rho_{s_i}} \right) \boldsymbol{I}_{i,b} \frac{\boldsymbol{\omega}_{i,b}^{n+1/2} - \boldsymbol{\omega}_{i,b}^n}{\Delta t} = - \left(1 - \frac{\rho_f}{\rho_{s_i}} \right) \boldsymbol{\omega}_{i,b}^n \times \boldsymbol{I}_{i,b} \boldsymbol{\omega}_{i,b}^n + \boldsymbol{T}_{i,b}'. \tag{33}$$

These equations are integrated in time with a second-order accurate leap-frog Verlet scheme. In the equations above, the contact terms (force $\boldsymbol{F}_i'$ and torque $\boldsymbol{T}_i'$) usually involve time scales much shorter than the fluid time scale. Thus, the time-step Δt is not appropriate for the granular problem when multiple particles are colliding and does not allow the resolution of this problem in a single iteration from $t^n = n\Delta t$ to $t^{n+1} = (n+1)\Delta t$. Instead, we divide the interval Δt into N_g sub-intervals and uses a smaller time-step Δt_g such that $\Delta t_g = \Delta t / N_g$ with N_g chosen to be a reasonably small integer. The choice of Δt_g is not trivial and depends mainly on the technique that one uses to detect the contacts. With the so-called soft-sphere model that Grains3D relies upon, the contact time and Δt_g are directly linked to the stiffness of the material. While realistic stiffness values lead to too small and impracticable time-steps, one common practice is to artificially soften the particles as described in Wachs et al. (2012).
Please note that in the test cases presented in this paper, the collision detection and contact force calculation are not used as either (i) the test case pertains to the flow past an array of fixed obstacles, (ii) the test case involves a single particle or (iii) in section 6.2 discussing the case of two spheres approaching each other in a simple shear flow, we purposely turn off the calculation of the contact force and torque to assess the ability of the directly computed (i.e. as a result of solving the mass and momentum conservation equations on the adaptive grid in the narrow gap between the two particles) hydrodynamic force and torque to keep the two particles apart.

3.1.3 Third sub-problem: Fictitious domain problem

The fictitious domain problem reads: given $\boldsymbol{u}^{n+1/2}$, $\boldsymbol{\lambda}^n$, $\boldsymbol{U}_i^{n+1/2}$, $\boldsymbol{\omega}_i^{n+1/2}$ find $\boldsymbol{u}^{n+1}$, $\boldsymbol{\lambda}^{n+1}$, $\boldsymbol{U}_i^{n+1}$, $\boldsymbol{\omega}_i^{n+1}$ for particle $i = 1, \ldots, N$ such that

$$\rho_f \left(\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^{n+1/2}}{\Delta t} \right) + \boldsymbol{\lambda}^{n+1} = \boldsymbol{\lambda}^n \text{ over } \Omega \tag{34}$$

$$\left(1 - \frac{\rho_f}{\rho_s}\right) \left(M_i \left[\frac{\boldsymbol{U}_i^{n+1} - \boldsymbol{U}_i^{n+1/2}}{\Delta t} \right] \right) - \int_{P_i} \boldsymbol{\lambda}^{n+1} \, \mathrm{d}\boldsymbol{x} = \boldsymbol{0} \text{ over } P_i \tag{35}$$

$$\left(1 - \frac{\rho_f}{\rho_s}\right) \left(\boldsymbol{I_i} \left(\frac{\boldsymbol{\omega}_i^{n+1} - \boldsymbol{\omega}_i^{n+1/2}}{\Delta t} \right) \right) - \int_{P_i} \boldsymbol{r_i} \times \boldsymbol{\lambda}^{n+1} \, \mathrm{d}\boldsymbol{x} = \boldsymbol{0} \text{ over } P_i \tag{36}$$

$$\int_{P_i} \boldsymbol{u}^{n+1} - \left(\boldsymbol{U}_i^{n+1} + \boldsymbol{\omega}_i^{n+1} \times \boldsymbol{r_i} \right) \mathrm{d}\boldsymbol{x} = \boldsymbol{0} \text{ over } P_i. \tag{37}$$

This is the final step of the algorithm where the effect of the Lagrange multipliers $\boldsymbol{\lambda}$ are accounted for. The Lagrange multipliers $\boldsymbol{\lambda}$ are computed together with the particles velocity $(\boldsymbol{U}_i, \boldsymbol{\omega}_i)$ and the constraint of rigid body motion (37) within the particle domain $P = \bigcup_{i=1}^{N} P_i$. Problem (35)-(37) is a saddle-point problem solved with an iterative Uzawa/Conjugate gradient method detailed in appendix A.

3.2 Spatial discretization

3.2.1 Navier-Stokes

The Navier-Stokes equations are discretized in space with a 2nd order accurate finite-volume scheme. The velocity field $\boldsymbol{u}$ and pressure field p are defined at the center of the cells while pressure-gradient terms, acceleration terms and fluxes are defined (and computed) at the center of the faces. The discretization of the Laplacian operator is performed using a standard cell-centered finite difference approximation (five-points stencil in $2D$). The computation of the convective term $\boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u}$ in (22) involves a 3 steps procedure: (i) a reconstruction step (via slope limiters) followed by (ii) a characteristic extrapolation (prediction) of $\boldsymbol{u}^n$ to $\boldsymbol{u}^{n+1/4}$ on the cell-faces and (iii) the evaluation of the fluxes. Finally,

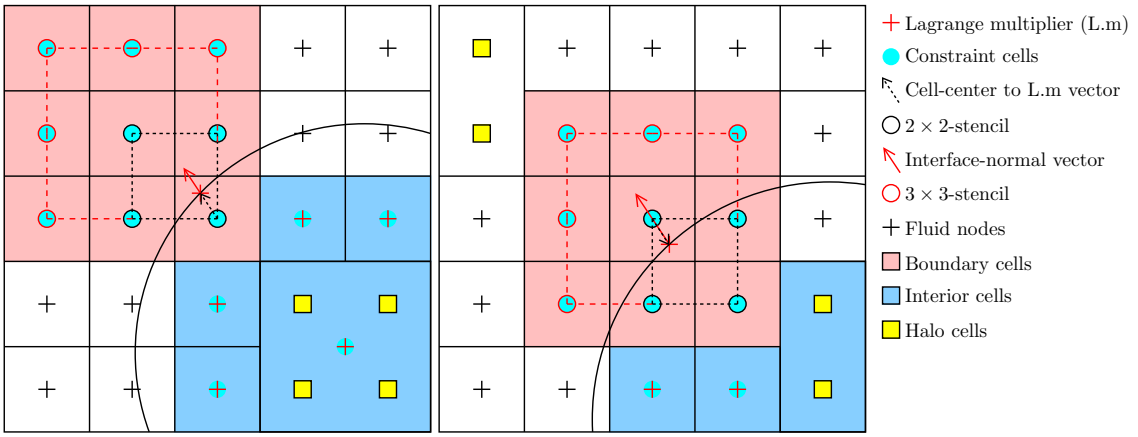


Figure 1: Constrained fluid cells by the Lagrange multipliers (shown with red + symbol) with a collocation point method. Red colored cells: cells constrained by the multiplier when located on the particle's surface. The stencil is oriented in the direction of the particle's (exterior) normal in order to encapsulate (and constraint) a maximum number of fluid cells. Two different configurations are shown, the left case is the optimum with 8 cells in the fluid region. Blue colored cells: each multiplier constraints one fluid cell as their positions coincide exactly with the fluid cells.

the projection step of (27)-(28) ensures that the velocity field defined at the face centers of a given cell is divergence-free at the discrete level. The centered velocity field components of $\boldsymbol{u}$ are then obtained via an arithmetic average of the face-centered values. One can argue that $\boldsymbol{u}$ is not strictly divergence-free at the discrete level. Because the DLM/FD problem is solved after the projection step in the global temporal splitting algorithm, our final velocity field at the end of the sequence is never strictly divergence-free. The resolution of the linear problems (the Helmholtz/Poisson equation of the projection step and the purely viscous problem that both require to invert a Laplacian operator) are handled by a geometric multi-grid solver implemented on adaptive quadtree/octree grids (Popinet, 2003, 2015).

3.2.2 Fictitious domain

The spatial discretization of the fictitious domain problem (35) leads to a saddle point problem for velocities $\boldsymbol{u}^{n+1}$, $\boldsymbol{U}^{n+1}$, $\boldsymbol{\omega}^{n+1}$ and Lagrange multipliers $\boldsymbol{\lambda}^{n+1}$. It is solved implicitly with an iterative Uzawa/Conjugate-gradient method described in appendix A. In practice, we choose the so-called collocation point method (Glowinski et al., 1999; Wachs et al., 2015) that revolves around distributing a set of discrete points as uniformly as possible on the surface and in the volume of the particles (such that they materialize the fictitious domain) and constraining the flow field locally in order to enforce the rigid-body motion as accurately as possible at each discrete point associated with a multiplier. It results that the functional space for the Lagrange multipliers $\boldsymbol{\lambda}$ is constructed by covering the ith-particle domain P_i with a set of L_i points $\hat{P}_i = \{\boldsymbol{x_{il}}\}_{l=1}^{L_i}$ and associating a Dirac delta function to each point $\boldsymbol{x_{il}}$ as its corresponding basis function:

$$\boldsymbol{\lambda}(\boldsymbol{x}) = \sum_{i=1}^{N}\sum_{l=1}^{L} \boldsymbol{\lambda_{il}}\delta(\boldsymbol{x}-\boldsymbol{x_{il}}). \tag{38}$$

The computation of the scalar products in the weak-formulation (17)-(21) $<\boldsymbol{\lambda},\boldsymbol{v}>_{P_i}= \int_{P_i}\boldsymbol{\lambda}\cdot\boldsymbol{v}\mathrm{d}\boldsymbol{x}$ and $<\boldsymbol{\nu},\boldsymbol{u}>_{P_i}= \int_{P_i}\boldsymbol{\nu}\cdot\boldsymbol{u}\mathrm{d}\boldsymbol{x}$ is then significantly eased. For instance, the former product (same applies to the latter product) becomes:

$$<\boldsymbol{\lambda},\boldsymbol{v}>_{P_i}= \sum_{l=1}^{L_i}\boldsymbol{\lambda_{il}}\,\boldsymbol{v}(\boldsymbol{x_{il}}). \tag{39}$$

The test functions $\boldsymbol{v}(\boldsymbol{x_{il}})$ associated to each Lagrange multiplier at point $\boldsymbol{x_{il}}$ define the velocity reconstruction used to enforce the rigid-body motion. Following (Wachs et al., 2015), we choose quadratic *shape* functions constructed on a 3×3 in 2D and $3 \times 3 \times 3$ in 3D regular Cartesian stencil. In the two-dimensional case, for a given particle i (we drop the index i for ease of notations), we then have:

$$\boldsymbol{v}(\boldsymbol{x_l}) = \sum_{m=1}^{3} \sum_{n=1}^{3} \boldsymbol{v_{mn}} \phi_{mn}(x_l, y_l),$$

where the components of the vector field $\boldsymbol{v_{mn}}$ are equal to 1 as $\boldsymbol{v}(\boldsymbol{x_l})$ is a test function. The shape functions $\phi_{mn}(x_l, y_l)$ are the standard shape functions of a quadrilateral/hexahedral finite element. They are equal to 1 at (x_m, y_n) and zero at all other points of the stencil. The shape functions written in terms of re-scaled variables $0 \leq \zeta \leq 1$ and $0 \leq \eta \leq 1$ are as follows:

$$\phi_{mn}(x_l, y_l) = \hat{\phi_{mn}}(\zeta, \eta) = \phi'_m(\zeta)\, \phi'_n(\eta) \quad \text{for } m, n = 1, 2, 3 \tag{40}$$

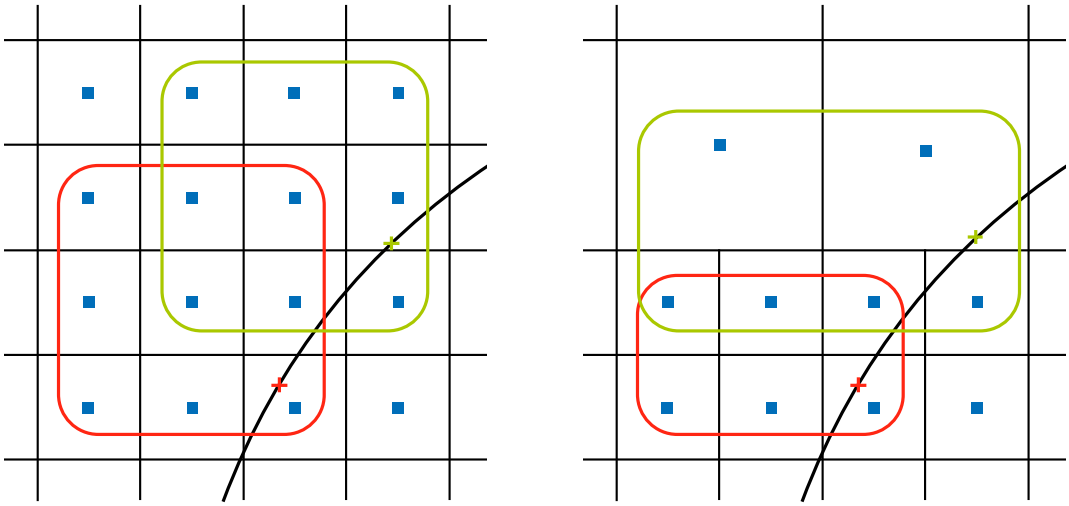
where

$$\begin{aligned} \phi'_1(\zeta) &= (1-\zeta)(1-2\zeta), \\ \phi'_2(\zeta) &= (2\zeta - 1)\zeta, \\ \phi'_3(\zeta) &= 4\zeta(1-\zeta) \end{aligned}$$

are the one-dimensional Lagrange quadratic polynomials.
For a multiplier (with index l) distributed in the *interior* of the fictitious domain (blue cells on figure 1), its position $\boldsymbol{x_l}$ matches the position of a fluid cell center $(x_l, y_l) = (x_m, y_n)$ for a combination of (m, n). It results that only one shape function has a non-zero term equal to 1 and the scalar product (39) reduces to the simple sum of all multipliers that lie in the *interior* part of the fictitious domain.
For a Lagrange multiplier distributed on the *boundary* (surface) of the fictitious domain, the 3×3 stencil used for the reconstruction is shown in figure 1 with red colored cells. This stencil is chosen such that a maximum number of constrained cells lie in the fluid region. Depending on the location of the multiplier within a given fluid cell and the direction of the particle's outwards normal vector, different scenarios are possible. Two such scenarios are depicted in figure 1 for the case of a circular fictitious domain. The optimum configuration is the case 1-(a) with a minimum number of constrained cells associated with the *boundary* Lagrange multiplier in the *interior* domain of the fictitious domain. An additional condition for a fluid cell to be considered as part of the *interior* of the fictitious domain is that it must not be part of a stencil associated with a *boundary* Lagrange multiplier.
The extension to the three-dimensional case is straightforward and involves adding a third one-dimensional Lagrange quadratic polynomial to the product of equation (40). For each multiplier lying on the *boundary* of the particle, we hence use a $3 \times 3 \times 3$ stencil with 27 associated shape functions.

3.2.3 A discussion on mesh adaptation, simple reconstruction stencil, homogeneous surface point distribution and computing efficiency

The principle of local mesh refinement is to densify the grid in regions of strong spatial variations of the discrete fields of interest. Basilisk uses the interpolation error between a field value at a grid point belonging to a grid level n and its interpolated value from a coarser grid level $n-1$ as a criterion to coarse or refine the grid locally by merging 8 cubes into a parent cube or slicing a cube into 8 subcubes, respectively (the same applies to squares in 2D). Given a user-defined threshold ϵ, the grid is refined or coarsened depending on whether this interpolation error is larger or lower than ϵ (van Hooft et al.,



(a) Constant grid size over the rigid body surface (b) Varying grid size over the rigid body surface

Figure 2: Problems caused by a varying grid size over the rigid body surface shown in (b) compared to a regular grid size where velocity interpolation is straightforward in (a). In (b), the two stencils for the red and green boundary points and the location of the green point with respect to the red point are illustrative only.

2018). In particle-laden flows, one preferential region of strong velocity variations is the boundary layer around each individual rigid body. As a result, by simply relying on the velocity field and the standard grid refining/coarsening algorithm in Basilisk, the grid would automatically be very fine in boundary layers around rigid bodies. However, velocity variations in boundary layers are often not homogeneous and this would lead in the general case to a distribution of sizes over the set of grid cells that host a Lagrange multiplier boundary point. As a result, the regular 27 cells (9 cells in 2D) quadratic reconstruction described in the previous section would not be applicable anymore.
The problem of the grid being of different size close to the rigid body surface results in two sub-problems of different complexity:

1. how to determine the interpolation stencil associated to a boundary point when the boundary point is not part of a $3 \times 3 \times 3$ (3×3 in 2D) group of cells of equal size (i.e. belonging to the same grid level) ?
2. how to distribute boundary points on the rigid body surface in a non-homogeneous way (non-homogeneous since the grid size varies along the rigid body surface) such that the velocity is spatially properly imposed but the system of equations is locally not over-constrained ?

This problem is illustrated in figure 2 where the groups of cells supporting the interpolation stencils of the red and green boundary points are illustrative only, although they are valid stencils. Sub-problem 1 above is relatively easy to solve and well documented in the literature on simple regular Cartesian grids (see Mittal et al. (2008) and Lu et al. (2018) among others) but computationally very expensive in the case of moving rigid bodies. In fact, the construction of the velocity interpolation stencil, while introduced in the context of simple regular Cartesian grids of constant grid size in, e.g., Mittal et al. (2008) and Lu et al. (2018), is general and applicable to any set of grid nodes. The principle of construction of the interpolation stencil is to consider a set of 10 grid nodes (6 grid nodes in 2D) and the following basis functions $1, x, y, z, xy, xz, yz, x^2, y^2, z^2$ ($1, x, y, xy, x^2, y^2$ in 2D) where x, y and z are the relative coordinates with respect to the location of the boundary point. This set of basis functions constitutes an adequate basis for a second-order interpolation of any field ϕ. Indeed, the second-order interpolation of ϕ in the

vicinity of the boundary point then reads:

$$\phi(x,y,z) = \sum_{i=0}^{i=2}\sum_{j=0}^{j=2}\sum_{k=0}^{k=2} A_{ijk}x^i y^j z^k, \quad i+j+k \leq 2 \tag{41}$$

and the coefficients A_{ijk} are the solution of the following Vandermonde linear system:

$$\begin{bmatrix} 1 & x_0 & y_0 & z_0 & x_0y_0 & x_0z_0 & y_0z_0 & x_0^2 & y_0^2 & z_0^2 \\ 1 & x_1 & y_1 & z_1 & x_1y_1 & x_1z_1 & y_1z_1 & x_1^2 & y_1^2 & z_1^2 \\ 1 & x_2 & y_2 & z_2 & x_2y_2 & x_2z_2 & y_2z_2 & x_2^2 & y_2^2 & z_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_9 & y_9 & z_9 & x_9y_9 & x_9z_9 & y_9z_9 & x_9^2 & y_9^2 & z_9^2 \end{bmatrix} \begin{bmatrix} A_{000} \\ A_{010} \\ A_{001} \\ \vdots \\ A_{002} \end{bmatrix} = \begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \vdots \\ \phi_9 \end{bmatrix} \tag{42}$$

where ϕ_n , $n = 0, 1, \ldots, 9$, and (x_n, y_n, z_n) , $n = 0, 1, \ldots, 9$, stand for the values of field ϕ at the grid nodes of coordinates (x, y, z). So in theory we can construct a second-order interpolation around each boundary point with any set of 10 points by inverting (42), but this operation is computationally very expensive and has been implemented primarily in the context of fixed rigid bodies (Mittal et al., 2008; Lu et al., 2018) and rarely in the context of moving rigid bodies with adaptive grids (Mohaghegh, Fazlolah and Udaykumar, HS, 2017). In fact, in the case of moving rigid bodies or when the mesh changes over times to adapt to the flow field as with an adaptive grid, the pseudo-code is provided in Alg 1. When rigid bodies are fixed and the mesh does not change with time, the two inner loops in the above pseudo-code are only performed at initialization and the sets of points and corresponding coefficients are computed and stored once and for all. Thus, the corresponding computing cost, even if substantial, becomes negligible compared to the total cost of computing, e.g., thousands of time steps to simulate the time evolution of the solution.

Algorithm 1 A pseudo-code to compute the interpolation polynomial for velocity reconstruction at the particle surface for moving particles or a grid non-constant in time as an adaptive grid.

for all discrete times **do**
  for all rigid bodies **do**
    for all boundary points **do**
      1. determine the set of 10 grid nodes around the boundary point
      2. invert the Vandermonde system to compute the coefficients A_{ijk}
      3. store the coefficients A_{ijk} for subsequent computations when running the Uzawa algorithm
    end for
  end for
end for

However, even if we would be able to come up with a fast implementation of the above solution to sub-problem 1, we would still need to supply an appropriate solution to sub-problem 2. In fact, it has been shown multiple times in the literature that the proper distribution of points of the rigid body surface in the context of fictitious domain/immersed boundary methods that represent rigid bodies on the fluid grid as a set of points is crucial to the overall accuracy of the computed solution (Uhlmann, 2005; d'Avino and Hulsen, 2010; Wachs et al., 2015). The common practice is to distribute points homogeneously with an inter-point distance of $\alpha.h$, where α is a coefficient between 1 and 2 and h is the fluid grid size. When h is constant over the rigid body surface, various efficient methods to construct the set of boundary points have been suggested in the literature for spheres

(Uhlmann, 2005) and for some specific non-spherical shapes (Wachs et al., 2015; Pierson et al., 2019). These construction methods all lead to reasonably homogeneous distributions of points that are all mostly distant from each other by the specified $\alpha.h$. The idea to separate boundary points by $\alpha.h$ is related to the size of the support of the regularized delta functions in the context of direct forcing immersed boundary methods or the size of the support of the stencil for the velocity reconstruction at the boundary, which here is $2h$. In the context of the solution of the DLM/FD saddle-point problem by a Uzawa algorithm, it also guarantees that the system is not locally overconstrained (see the original paper on DLM/FD by Glowinski et al. (1999)). Now, if the grid size varies over the rigid body surface, the determination of the set of boundary points that both guarantees (i) that the rigid body motion constraint is spatially properly enforced on the fluid grid and (ii) that the problem is locally not overconstrained is a daunting task for which there is no straightforward solution, to the best of our knowledge, even for a spherical rigid body, and hence even less for a non-spherical rigid body. As a simple illustration, this problem can be posed as follows in figure 2b: where should the green boundary point be located with respect to the red boundary point to satisfy both condition (i) and condition (ii) ?
While a varying grid size over the rigid body surface does not cause any particular problem to methods that do not represent rigid bodies on the fluid grid as a set of discrete points as, e.g., embedded geometry/cut-cell methods (Udaykumar et al., 1996; Johansen and Colella, 1998; Chung, 2013), we do not have at that point a satisfactory solution to the problem of varying grid size over the rigid body surface and even if we would, this would be computationally very expensive and potentially detrimental to the efficiency of the whole algorithm. To circumvent this issue and provide an efficient solution, we force the grid to be constant and equal to the smallest grid size over all rigid body surfaces and in a narrow region the width of which is equal to three cells around all rigid bodies. This is achieved by defining a color function or flag function on the fluid grid and artificially imposing large variations in the normal direction to the rigid body surface, i.e., large variations in the 3-cell narrow band. The outcome is twofold:

1. we can use the regular quadratic interpolation presented in section 3.2.2,
2. we can use the same boundary points construction rules established for a constant inter-point distance introduced in Wachs et al. (2015).

To summarize, we locally treat the problem of local velocity reconstruction on an adaptive quadtree-octree grid as if the grid was a simple regular Cartesian grid. As a result, we may refine the mesh more than required over some part of the rigid body surfaces, but we postulate that this potential overhead of grid cells is favorably balanced by a simpler implementation, the ease to distribute boundary points and a local velocity reconstruction relying on a classical 27-point quadratic interpolation whose coefficients can be readily computed.

4 Basilisk-specific rules associated with parallelism and stencil-operations

While the DLM/FD algorithm was originally derived and implemented with a fixed grid in mind, we can implement it in Basilisk's quadtree/octree framework provided that we adapt it to satisfy Basilisk's specific set of rules.
In the quadtree/octree framework, cells are not labeled with indices i, j, k as it is usually the case with constant Cartesian grids but instead, cells and neighbors are accessed with predefined operators. The traversal of the grid is done via an operator called *foreach()* that loops over all the computational cells (or leaves of the tree). The (regular) stencil associated to a cell (its direct neighbors) can be accessed via the operator *foreach_neighbors()*

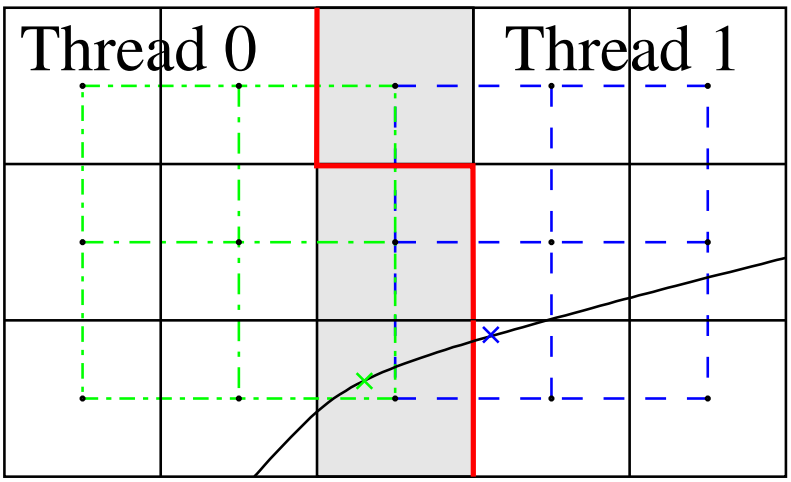


Figure 3: Example of cells (colored in grey) being part of two different stencils which are split by the domain-decomposition (represented by the red line). The green and blue crosses are the position of the Lagrange multipliers and the dash-dotted lines are the associated stencil used for the local reconstruction of the velocity field. In this configuration, each stencil has cells that belongs to the other thread's domain (one for the green, and two for the blue).

when nested within the *foreach()* loop. With such cell-traversal operators, accessing the stencils associated to a Lagrange multiplier is rather straight-forward but the modification of the fields, defined on these cells, is more complicated and is subject to a fundamental constraint/rule in Basilisk. Indeed, in order to maintain the consistency of the stencils, Basilisk only allows writing within cells which are *local* to a sub-domain (i.e., a thread) and accessed directly (i.e., accessed with a *foreach()* loop).
This rule also guarantees the consistency of the domain decomposition technique used during parallel computations as cells which belong to a given stencil can be part of another thread's domain. An example of such a scenario is depicted on figure 3 where the stencil associated to the green Lagrange multiplier has one cell in the other thread's computational domain. This particular cell (the top middle grey-colored one) is a *ghost* cell for thread 0 and it is a *local* one for thread 1. With the above mentioned rule, this means that thread 0 can access (read) the value stored in that cell (with the combination of operators *foreach()* and *foreach_neighbors()*) but cannot modify its value, only thread 1 can modify it within a direct *foreach()* loop. Thus, the modification of a cell accessed indirectly or non-locally is not allowed. This slightly complicates the implementation of the algorithm as the effect of the Lagrange multipliers (through the computation of scalar products and residuals involved at different stages of the algorithm) has to be spread to non-local cells as well. It also forbids the straightforward and natural approach where one would first locate the cell containing a multiplier, sweep its neighborhood and distribute the associated weights (given by the shape-functions in equation (40)).
The solution is to reverse the algorithm and to consider local operations only. We can see in figure 3 that the middle top cell is actually part of two different stencils as it is affected by two multipliers. Thus, instead of using the straight-forward (and probably optimal) approach described above, we check the neighborhood of each (local) cell if a Lagrange multiplier is present and compute the weights and scalar products when it is the case. This comes with an additional number of operations of about $\sim 5^2 2^{2M}$ in 2D and $\sim 5^3 2^{3M}$ in 3D for the worst case scenario (i.e when the grid is fully refined and becomes a constant Cartesian grid) with M the level of refinement (the number of time that a cell is sub-divided). While the algorithm is iterative, this operation is performed only once per time-step (as the grid does not change over the iteration of the Uzawa algorithm) and its load is perfectly distributed among the different sub-domains (i.e., threads).

5 Validation cases on quadtrees/octrees

We present now a set of test cases for which either analytical or reference solutions are available and to which we compare our numerical results. The test cases are chosen to cover a broad range of applications: from Stokes flow to highly inertial regime, in 2D and in 3D, with spherical and non-spherical particles. Special attention is given to the spatial (obtained with and without adaptive meshes) and temporal convergence of our method.

5.1 Dimensionless form of the equations and dimensionless parameters

The set of governing equations (17)-(21) can be easily written in a dimensionless form using the following scales: L_c for length, V_c for velocity, L_c/V_c for time, $\rho_f V_c^2$ for pressure, V_c^2/L_c for gravity acceleration, $\rho_f V_c^2/L_c$ for distributed Lagrange multiplier and $\rho_f V_c^2 L_c^2$ for contact force (see (Wachs et al., 2015) among many others). Using * superscripts to distinguish dimensionless quantities, the dimensionless and strong form of (17)-(21) then reads as follows:

$$\frac{\partial \boldsymbol{u}^*}{\partial t^*} + \boldsymbol{u}^* \cdot \boldsymbol{\nabla}^* \boldsymbol{u}^* + \boldsymbol{\nabla}^* p^* - \frac{1}{Re^*} \Delta^* \boldsymbol{u}^* = -\boldsymbol{\lambda}^* \text{ in } \Omega^*, \tag{43}$$

$$-\boldsymbol{\nabla}^* \cdot \boldsymbol{u}^* = 0 \text{ in } \Omega^*, \tag{44}$$

for particle $i = 1, \ldots, N$

$$(\rho_r^* - 1)\, V_i^* \left(\frac{\mathrm{d}\boldsymbol{U}_i^*}{\mathrm{d}t^*} - Fr^* \frac{\boldsymbol{g}}{g} \right) - \boldsymbol{F}_i'^* = \int_{P_i^*} \boldsymbol{\lambda}^* \, \mathrm{d}\boldsymbol{x}^*, \tag{45}$$

$$(\rho_r^* - 1)\, \frac{\mathrm{d}\boldsymbol{I}_i^* \boldsymbol{\omega}_i^*}{\mathrm{d}t} - \boldsymbol{T}_i'^* = \int_{P_i^*} \boldsymbol{r} \times \boldsymbol{\lambda}^* \, \mathrm{d}\boldsymbol{x}^*, \tag{46}$$

$$\boldsymbol{u}^* - (\boldsymbol{U}_i^* + \boldsymbol{\omega}_i^* \times \boldsymbol{r}^*) = \boldsymbol{0} \text{ in } P_i^*. \tag{47}$$

where $V_i^* = M_i/(\rho_s L_c^d)$ denotes the dimensionless particle volume, $\boldsymbol{I}_i^* = \boldsymbol{I}_i/(\rho_s L_c^{d+2})$ the dimensionless particule inertia tensor, g the gravity acceleration modulus and d the space dimension. In (43)-(47), we have also introduced the three following dimensionless numbers:

$$\text{Reynolds number } Re^* = \frac{\rho_f V_c L_c}{\mu_f} = \frac{T_v}{T_a} \tag{48}$$

$$\text{Inverse Froude number } Fr^* = \frac{g L_c}{V_c^2} \tag{49}$$

$$\text{Density ratio } \rho_r^* = \frac{\rho_s}{\rho_f} \tag{50}$$

where $T_v = \rho_f L_c^2/\mu_f$ and $T_a = L_c/V_c$ denote the viscous time scale and advective time scale, respectively. In the following, we take $L_c = D$ where D is the particle diameter or particle equivalent diameter in the case of a non-spherical particle. Please note that depending on the choice of the characteristic velocity V_c, the inverse Froude number is not necessarily a parameter independent of the density ratio ρ_r^* (Yu et al., 2004).

5.2 Stokes flow through a periodic array of spheres

Pure Stokes flows are known to be challenging problems for operator-splitting methods. The problem of a creeping flow through an infinite array of spheres was solved analytically by Zick and Homsy (1982) who reformulated the initial tri-dimensional problem as a set of two-dimensional integral equations. With numerical integration, they obtained the drag coefficient K^* as a function of the solid volume fraction c^* and packing characteristics that agrees well with respect to experiments and theoretical asymptotic analysis. Our goal is to verify here that our octree DLM/FD method can properly compute the flow field, especially at high c^*, and yield the correct drag coefficient K^*.

5.2.1 Numerical setup

We compute the motion of a creeping flow through an infinite array of spheres for various solid volume fractions c^* and for different numerical parameters. We use a cubic computational domain of edge length L with periodic boundary conditions in all directions and we place the particle of diameter $D = (6c^*/\pi)^{1/3}L$ at the center of the box. We use the diffusive time T_v as characteristic time scale and set μ_f and ρ_f such that $Re^* = 0.01$, i.e., the corresponding Reynolds number is sufficiently small to represent a quasi-pure Stokes flow. The streamwise direction of the flow is set to the x direction. The flow field is initially at rest and is driven by an imposed constant pressure gradient $\boldsymbol{\nabla} p_{imp} = \frac{\Delta p_{imp}}{L}\boldsymbol{e_x}$ where Δp_{imp} is the pressure difference between $x = 0$ and $x = L$. We choose the dimensionless time-step such that $\Delta t^* = \Delta t/T_v = 6.3 \cdot \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$ and the spatial resolution is varied by changing the adaptive grid maximum level of refinement M. This ensures a maximum number of 2^M cells per direction or, equivalently, $D/\Delta = 2^M D/L$ cells per diameter where Δ denotes the grid size of the highest level of refinement. We vary M such that the inverse of the minimum dimensionless grid size $1/\Delta^* = D/\Delta$ varies within the range $D/\Delta = \{20, \ldots, 200\}$ for all solid volume fractions c^* considered.
According to Zick and Homsy (1982), the drag coefficient K^* is related to the force exerted on each sphere of the array by

$$\boldsymbol{F} = 3\pi\mu_f D K^* \boldsymbol{V_s}, \tag{51}$$

where $\boldsymbol{V_s}$ is the superficial velocity defined as

$$\boldsymbol{V_s} = \frac{1}{L^3}\int_0^L\int_0^L\int_0^L \boldsymbol{u}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \tag{52}$$

In practice, using mass conservation, we measure $\boldsymbol{V_s}$ as $\boldsymbol{V_s} = \boldsymbol{Q}/L^2$ and compute the flow rate $\boldsymbol{Q}$ through the plane normal to the streamwise direction x located at $x = 0$ as:

$$\boldsymbol{Q} = \int_0^L\int_0^L \boldsymbol{u}(0, y, z)\mathrm{d}y\mathrm{d}z. \tag{53}$$

5.2.2 Temporal and spatial convergence

To assess the temporal accuracy of the method we set the spatial resolution to $D/\Delta = 100$ and we consider a dense case of $c^* = 0.45$. The time evolution of the dimensionless superficial velocity $|\boldsymbol{V_s^*}| = |\boldsymbol{V_s}|D\rho_f/\mu_f$ and the dimensionless force $|\boldsymbol{F^*}| = |\boldsymbol{F}|/|\boldsymbol{\nabla} p_{imp}|$ acting on the sphere are plotted in figure 4 for all time-steps considered. In all cases, we observe a similar trend: after a short transient, a steady state is reached and $|\boldsymbol{V_s^*}|$ and $|\boldsymbol{F^*}|$ tend towards a converged value as Δt^* is reduced. Interestingly, and as demonstrated by Wachs et al. (2015) and the references therein, the dependency with respect to the time-step observed in figures 4-(a) and 4-(b) is significantly reduced by introducing the Lagrange multipliers $\boldsymbol{\lambda}^n$ (solution of the fictitious domain problem of the previous time-step) as an explicit forcing term in the Navier-Stokes equations (29). This can be seen in figures 4-(c) and 4-(d) where all the curves collapse onto a single curve as well as in figure 5-(a) where the drag coefficient K^* is plotted for various Δt^*.
In order to measure the temporal convergence rate of our computed solutions, we compute the relative error $\epsilon = |K^*_{ref} - K^*|/K^*_{ref}$, where the reference drag coefficient K^*_{ref} is obtained by fitting the curve $K^*(\Delta t^*)$ with a first-order polynomial $P(\Delta t^*)$ within a range of chosen points. We typically use 3 points as shown in figure 5-(a) and get $K^*_{ref} = P(\Delta t^* = 0)$. The error ϵ as a function of Δt^* is plotted in figure 5-(b) with their associated fits. We obtain a temporal convergence rate which strongly depends on the presence of the explicit Lagrange multipliers as an additional coupling term. When not present, we measure an order of convergence of 0.94 which is very close to 1, the theoretical value

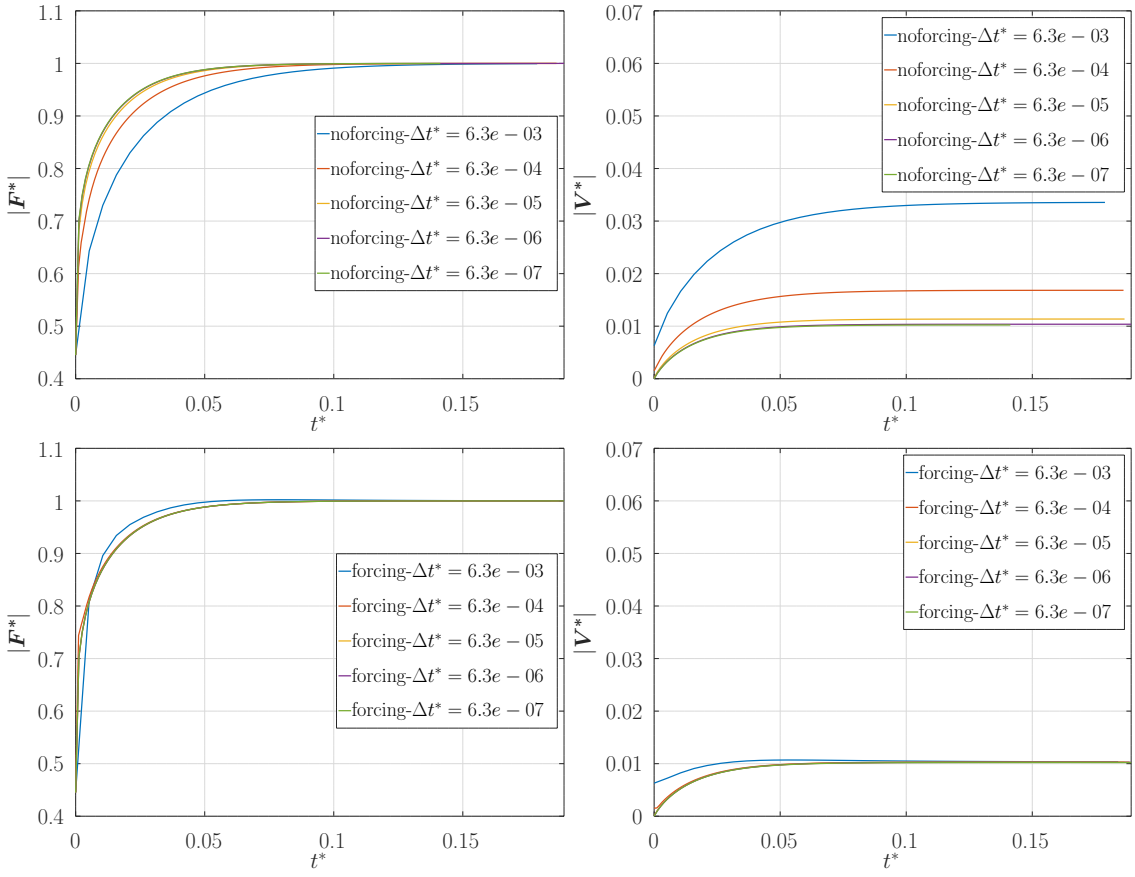


Figure 4: Time evolution of dimensionless force acting on the sphere $|\boldsymbol{F^*}| = |\boldsymbol{F}|/|\boldsymbol{\nabla} p_{imp}|$ and magnitude of the dimensionless superficial velocity $|\boldsymbol{V^*}| = |\boldsymbol{V}|D\rho_f/\mu_f$ for concentration $c^* = 0.45$ and various Δt^*. The effect of the explicit Lagrange multipliers as an additional coupling term in equations (29) and (35) is illustrated on the bottom/top row figures. Top row: results obtained without the forcing term.

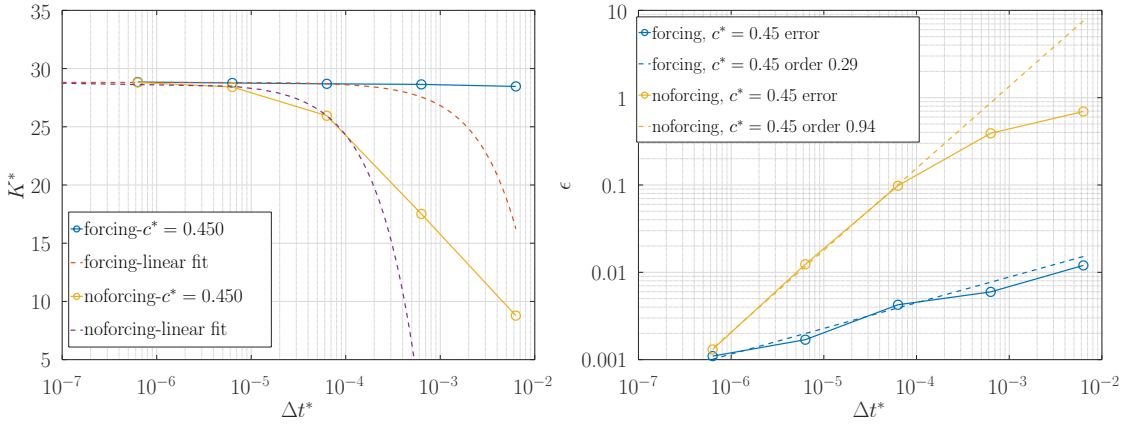


Figure 5: (a) Drag coefficient $K^*(\Delta t^*)$ for $c^* = 0.45$ with and without the use of the explicit coupling term $\boldsymbol{\lambda}^n$ in the sub-problems (29) (more precisely (24)) and (35). (b) plot of the relative error $\epsilon(\Delta t^*)$ in solid-lines and the fitted curves $a\Delta t^{*\alpha}$ (with α the fitted convergence rate) shown in dotted lines as visual reference.

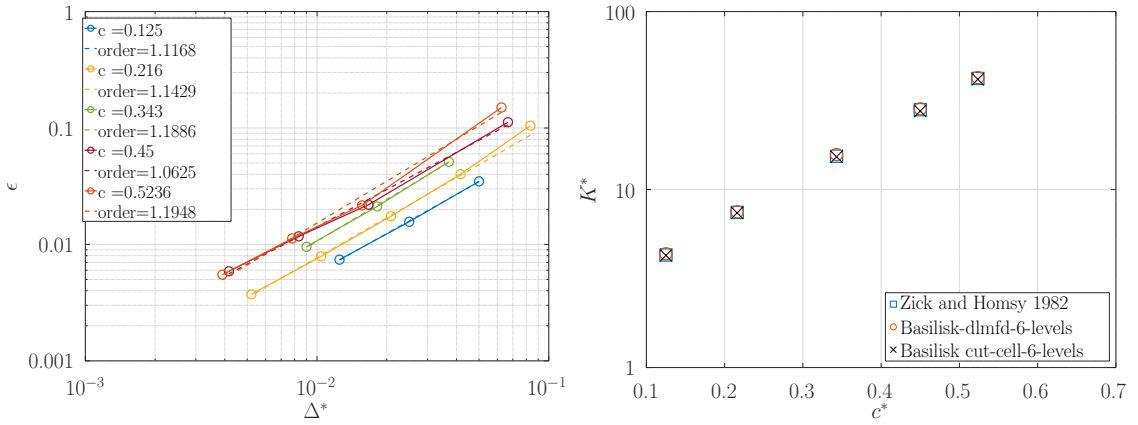


Figure 6: (a) Relative error ϵ versus the dimensionless grid size $\Delta^* = \Delta/D$. The fitted curves $a\Delta^{*\alpha}$ (with α the convergence rate) are shown in dotted lines as visual reference. (b) Drag coefficient K^* versus the concentration c^*, squares: present study, crosses: semi analytical results of Zick and Homsy (1982) and crosses: Basilisk's cut-cell method.

for our splitting algorithm. When present, it reduces to ~ 0.3 but the curve is globally shifted downwards by almost two orders of magnitude for the largest values of Δt^*. This emphasizes the importance of the explicit Lagrange multiplier coupling term in Stokes flows in order to obtain accurate solutions with larger time-steps. These results are in line with what we observed with our simple regular Cartesian grid DLM/FD method in Wachs et al. (2015). Without the explicit forcing term, the convergence rate of the solution is linear while with the explicit forcing term activated, the convergence rate drops but the magnitude of the error is significantly reduced for large time steps.

To assess the spatial convergence of our computed solutions, we compute the drag coefficient K^* for various levels of refinement. The time step is set to $\Delta t^* = 10^{-3}$ and we use the explicit Lagrange multipliers as an additional coupling term. The relative error ϵ is plotted in figure 6-(a) for various solid volume fractions c^*. We measure a clean convergence rate of $\sim 1.1 - 1.2$ for all the cases. A direct comparison with Zick and Homsy (1982)'s analytical results and Basilisk's built-in $2nd$ order accurate in space cut-cell/embedded geometry method with the same level of refinement is provided in figure 6-(b). We obtain a very satisfactory agreement.

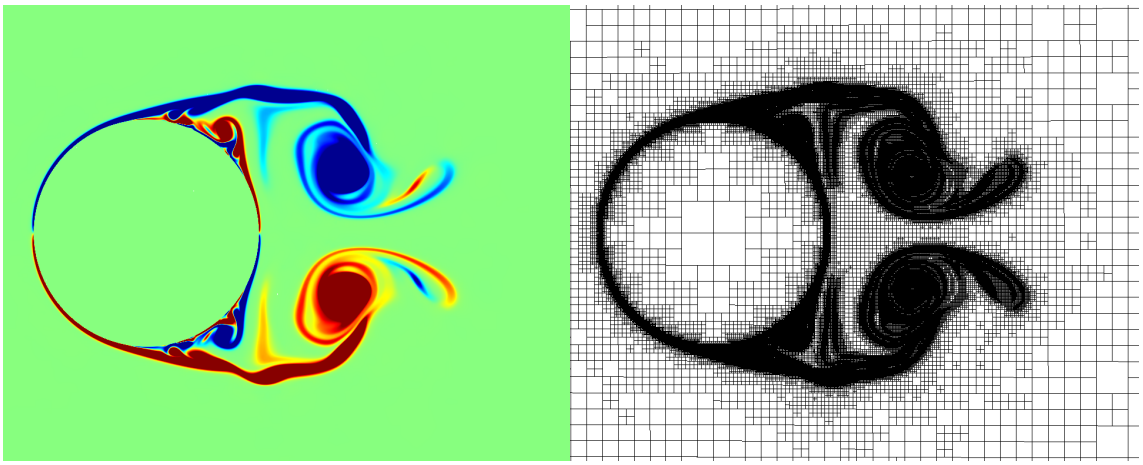


Figure 7: Snapshot of a flow past a cylinder at $Re^* = 9500$ at $t^* = 3$: (a) Axial vorticity field ω_z. (b) Adaptive quadtree grid.

5.3 Flow past a circular cylinder at $Re^* = 9500$

The impulsively started flow past a circular cylinder problem is a canonical problem of complex boundary layer separation. Inspired by the experiments of Bouard and Coutanceau (1980), notable early numerical simulations include the results of Koumoutsakos and Leonard (1995), hereafter referred to as K & L and used in the comparisons below. Our goal here is to test our implementation with adaptive quadtrees on a two-dimensional flow past a circular cylinder in a highly inertial regime.

High resolution is needed to resolve the boundary layers properly. Mohaghegh, Fazlolah and Udaykumar, HS (2017) propose to use a maximum resolution of order $D/10/Re^*$ with Re^* defined with the inflow far field velocity U_{in} as characteristic velocity V_c. The computational domain is a square of edge length $L = 18D$. The maximum level of refinement is chosen to be 16 such that the inverse of the minimum dimensionless grid size $1/\Delta^* = D/\Delta$ is 3600. On a constant regular Cartesian grid, this would correspond to a total number of cells of $2^{32} \sim 4 \cdot 10^9$ for the same resolution. The dimensionless time step Δt^* is dynamically adapted during the simulation with the velocity U^* such that a $CFL \equiv max(U^*)\Delta t^*/\Delta^*$ restriction of 0.8 is satisfied and is also bounded by 10^{-3} to make the splitting error tolerable. The streamwise direction of the flow is set to the x direction, hence we impose a Dirichlet boundary condition $(1, 0)$ on the dimensionless velocity over the entry located at $x^* = 0$ and a zero Dirichlet boundary condition on the dimensionless pressure combined to homogeneous Neumann boundary conditions on the dimensionless velocity over the exit located at $x^* = 18$. We also impose a Dirichlet boundary condition $(1, 0)$ on the dimensionless velocity over lateral boundaries $y^* = 0$ and $y^* = 18$. The fluid is initially at rest.

A snapshot at $t^* = tU_{in}/D = 3$ of the axial vorticity field, zoomed closely around the cylinder, is depicted in figure 7-(a) and the corresponding mesh is shown in figure 7-(b). It can be seen that the (adaptive) mesh is refined on the regions where thin layers of vorticity are present and where the velocity gradients are strong. This results in a high number of cells used around the cylinder and in the wake while the core of the cylinder (which is a fictitious domain filled with fluid that is constrained to be at rest) and the remaining regions of the flow field are represented with coarser grid cells. The vortex structures can be visually compared to the results obtained by Koumoutsakos and Leonard (1995) in figure 26 in their paper and to those obtained by Mohaghegh, Fazlolah and Udaykumar, HS (2017) in figure 3 in their paper. We observe a good quantitative agreement. To go one step further, we plot in figure 8-(a) the time history of the drag coefficient C_d^* defined

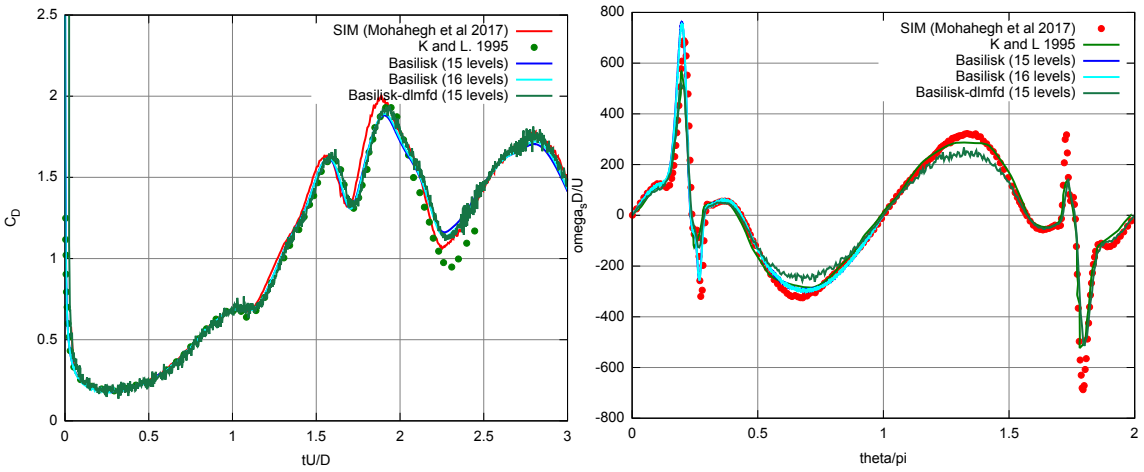


Figure 8: (a) Comparison of the temporal evolution of the drag coefficient C_d with various models. (b) Position of the zeros of the axial vorticity ω_z on the cylinder's surface at $t^* = 3$.

as:

$$C_d^* = \frac{|F_x|}{\frac{1}{8}\rho_f U_{in}^2 \pi D^2} \tag{54}$$

along with the results of Mohaghegh, Fazlolah and Udaykumar, HS (2017) obtained with an immersed boundary method, Koumoutsakos and Leonard (1995) obtained with a vortex method and those obtained with Basilisk's built-in $2nd-$order accurate in space cut-cell/embedded geometry method. We see that our DLM/FD method is capable of properly capturing the correct dynamics of this highly complex flow field and compares well with other numerical methods. The visible noisy aspect of our signal can have multiple sources:

- A flow field at such high Reynolds number might be subject to strong fluctuations that can manifest through high-frequency oscillations.
- Our reconstruction technique is locally non-conservative (for fixed and moving particles) as it uses the shape-functions given in equation (40). There is no flux balance considered in such approach which is common to DLM/FD and immersed-boundary methods.
- The adaptive mesh can be a source of noise by itself. In our formulation, the force is computed by integrating the Lagrange multipliers over the domain of the particle. As we use Dirac functions as basis functions for each Lagrange multiplier (see equation 38), the force is directly obtained by computing their sum. As the mesh is adaptive inside the particle (we force it to be constant on a shell on the surface but not inside), this change of topology and spatial resolution might induce oscilations on the computed force.

Note also that our signal is plotted for every time-step without any treatement of post-process operations.
In figure 8-(b), we plot the evolution of the axial vorticity on the surface of the cylinder and compare it to the other authors and techniques. We obtain a similar profile as other authors, although the peaks are not as sharp as for the other methods. We argue that this is due to the post processing techniques we employed. Indeed, the vorticity field is not solution of the fictitious domain problem and it has to be computed from the velocity field and then interpolated to the cylinder surface. For now, this interpolation does not use the fictitious domain specific reconstruction strategy but rather the standard interpolation scheme available in Basilisk, thus introducing interpolation error close to the particle boundary. Nonetheless we find the agreement rather satisfactory.

5.4 Flow past a sphere

The flow past a fixed spherical obstacle is also a classical and well documented flow configuration. We hence compare our computed results both to published numerical/experimental works and existing correlations for the drag and lift force coefficients. Our objective is to compute the flow at Reynolds numbers between 10 and 300, i.e., to capture the onset of vortex shedding manifesting for $Re^* > 270$. We define Re^* with the inflow far field velocity U_{in} as characteristic velocity V_c. Assuming the streamwise direction of the flow is set to the x direction, we also define the lift coefficient C_l^* and the Strouhal number St^* as follows:

$$C_l^* = \frac{\sqrt{F_y^2 + F_z^2}}{\frac{1}{8}\rho_f U_{in}\pi D^2} \tag{55}$$

$$St^* = \frac{f_v D}{U_{in}} \tag{56}$$

where f_v denotes the vortex shedding frequency for $Re^* > 270$, while the drag coefficient C_d^* is already defined in (54). We place the sphere at the center of a cubic computational domain of edge length $L = 30D$. The dimensionless time step Δt^* is dynamically adapted during the simulation such that a CFL restriction of 0.8 is satisfied and is also bounded by 10^{-3} to make the splitting error tolerable. We impose a Dirichlet boundary condition $(1, 0, 0)$ on the dimensionless velocity over the entry located at $x^* = 0$ and a zero Dirichlet boundary condition on the dimensionless pressure combined to homogeneous Neumann boundary conditions on the dimensionless velocity over the exit located at $x^* = 30$. We also impose a Dirichlet boundary condition $(1, 0, 0)$ on the dimensionless velocity over the four lateral boundaries. The fluid is initially at rest.

For Reynolds numbers within the range $150 \leq Re^* \leq 250$, we perform four computations with different spatial resolutions by varying the maximum level of refinement of the adaptive mesh from 10 to 13 such that the inverse of the minimum dimensionless grid size $1/\Delta^* = D/\Delta$ is 34, 68, 136 and 272, respectively. For $Re^* = 50$ and $Re^* = 100$ we consider the cases $1/\Delta^* = 17, 34, 68$ and 136.

Group	C_d^*	C_l^*	St^*
Johnson and Patel (1999)	0.656	0.069	0.137
Kim et al. (2001)	0.657	0.067	0.134
Hartmann et al. (2011)	0.657	0.069	0.135
Eitel-Amor et al. (2013)	0.660	0.065	0.132
Our results	0.652	0.068	0.133

Table 1: Comparison of our computed drag coefficient, lift coefficient and Strouhal number to other published numerical works in the case of the flow past a fixed sphere at $Re^* = 300$

For $Re^* = 300$, a case for which vortices are shed periodically from the sphere, we set the maximum level of refinement such that the inverse of the minimum dimensionless grid size $1/\Delta^* = D/\Delta$ is 48. Although the wake is only moderately well resolved as shown in figure 9, our computed C_d^*, C_l^* and St^* are already in very satisfactory agreement with the literature, as supported by data presented in table 1. The comparison of our results with the experimental work of Roos and Willmarth (1971), numerical data of Johnson and Patel (1999) and the classical Schiller-Naumann correlation defined for $Re^* < 1000$ as $C_d^* = 24(1 + 0.15Re^{*0.687})/Re^*$ is presented in figure 10. Our results match very well the results obtained by these authors.

The spatial convergence of the computed solution with mesh refinement is presented in figure 11. We plot in figure 11-(a) the drag coefficient C_d^* as a function of the minimum

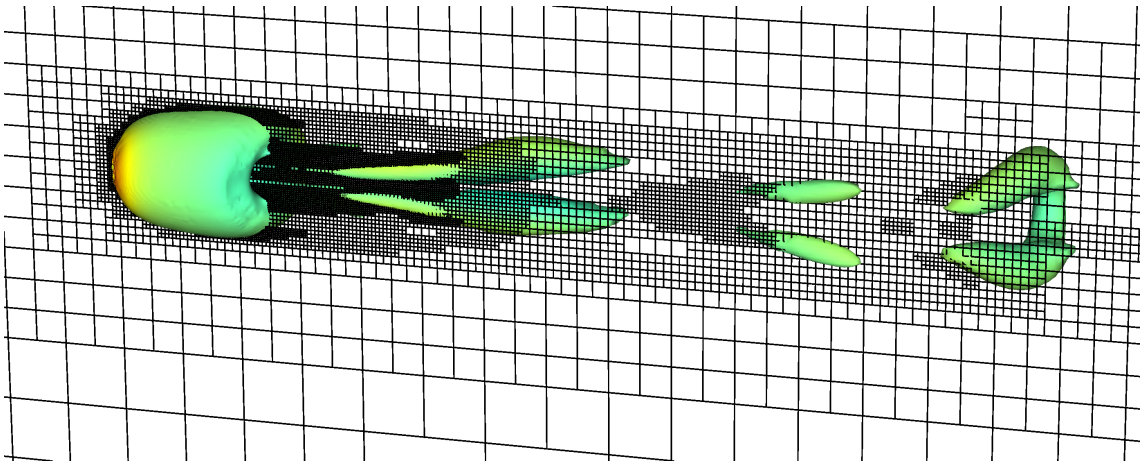


Figure 9: Vortex shedding at $Re^* = 300$ visualized with the Q criterion together with the adaptive grid in a xy cut plane containing the sphere center of mass

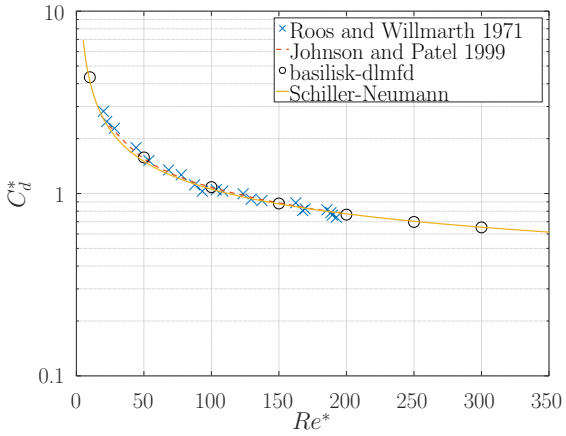


Figure 10: Comparison of the drag coefficient C_d^* as a function of the Reynolds number Re^* against the experimental results of Roos and Willmarth (1971), the numerical work of Johnson and Patel (1999) and the Schiller-Neumann's correlation.

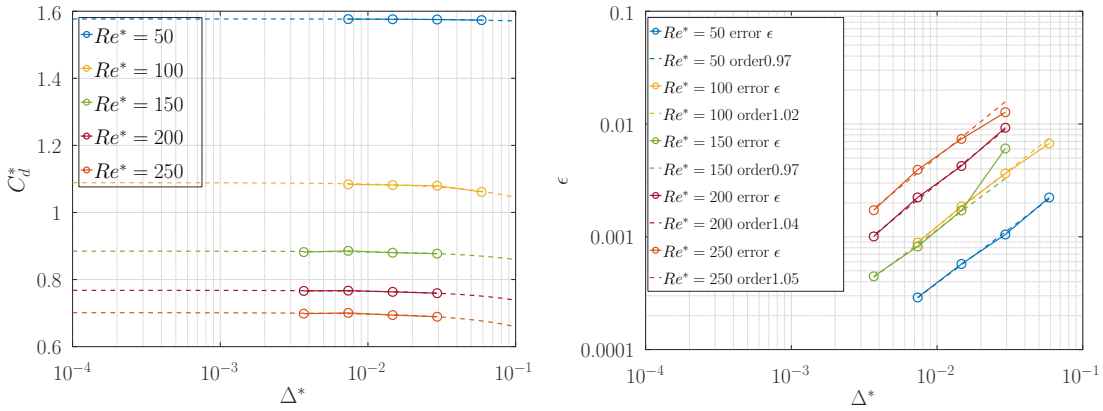


Figure 11: Flow past a sphere with adaptive grids: (a) Drag coefficient C_d^* as a function of the smallest dimensionless grid size Δ^* in full lines and their corresponding fit in dashed-lines. For $Re^* > 50$ we used time-averaged quantities. (b) relative error versus Δ^*.

dimensionless grid size Δ^* with their associated (1st order) polynomial fits $P(\Delta^*)$ for various Re^*. Note that, for $Re^* > 50$, we time-average our results and the reference value for C_d^* is obtained by evaluating $P(\Delta^* = 0)$. The evolution of the relative error ϵ as a function of Δ^* is shown in figure 11-(b). For all the Reynolds considered, the error ϵ decreases with Δ^* and we measure an order of convergence that remains very close to ~ 1. In terms of number of cells, we measure during computations an average number of grid cells of $\sim 680,000$ for the case of 68 points per diameter at $Re^* = 250$. In order to achieve the same resolution in a box of size $L = 30D$ with a fixed Cartesian mesh, we would require $2^{3(11)}$ cells, a computation at least (in theory) ~ 12500 times more expensive.

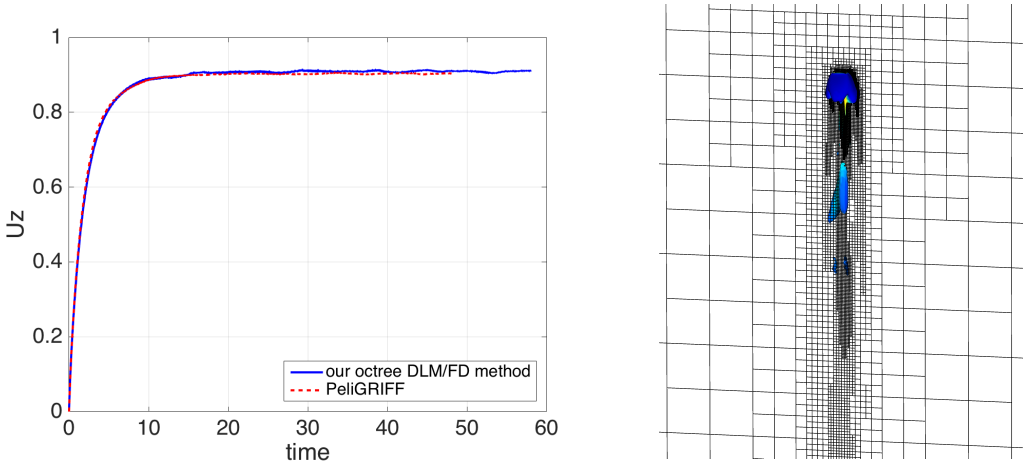
5.5 Coupling with Grains3D: freely-moving particles of arbitrary shape

As in our previous implementation of the DLM/FD algorithm on simple regular Cartesian grids (Wachs et al., 2015; Dorai et al., 2015), we have coupled our quadtree-octree DLM/FD algorithm implemented in Basilisk to our granular solver Grains3D (Wachs et al., 2012; Rakotonirina et al., 2019) to compute the motion of rigid particles of arbitrary shape immersed in a fluid. Here our objective is to validate our quadtree/octree DLM/FD algorithm for non-spherical particles, i.e., to validate the computation of the hydrodynamic interaction between the fluid and the rigid non-spherical particles, not to consider immersed particle/particle collisions. Hence we focus on a single particle case.
In a series of recent papers (Rahmani and Wachs, 2014; Seyed-Ahmadi and Wachs, 2019), we examined the dynamics of a single settling or rising non-spherical particle in an unbounded domain in which the fluid is quiescent far away from the moving particle with a simple regular Cartesian grid implementation of the DLM/FD method in our legacy code PeliGRIFF. The motion is driven by the density difference between the particle and the fluid. We established a flow map for the case of a cubic particle as a function of the solid to fluid density ratio $\rho_r^* = \rho_s/\rho_f$ and the Galileo number Ga^* defined as:

$$Ga^* = \frac{\rho_f \sqrt{|1 - \rho_r^*| g D^3}}{\mu_f} \tag{57}$$

In the case of a cube, D is given by $(6/\pi)^{1/3} e$ where e denotes the cube edge length such that a sphere of diameter D has the same volume as the cube of edge length e. Rahmani and Wachs (2014) and Seyed-Ahmadi and Wachs (2019) revealed some striking features of the dynamics of a settling or rising cube and in particular the existence of a spiralling (helical) motion at all ρ_r^* for Ga^* approximately between 120 and 170. The regularity of the helical trajectory was more marked for $\rho_r^* < 1$, i.e., for lighter rising cubes. The helical motion is a signature of the complex hydrodynamic interaction between the cube and the surrounding fluid and hence represents a challenging test case for any numerical method designed to compute particle-laden flows.
The flow configuration is as follows. The gravity acceleration vector points downwards in the z direction. The computational domain is a cube of edge length $700D$. The cubic particle is initially placed at the center of the box in the horizontal xy plane and $10D$ above the bottom wall of the box and rises in the domain as time evolves. The dimensionless time step Δt^* is dynamically adapted during the simulation such that a CFL restriction of 0.8 is satisfied and is also bounded by 10^{-3} to make the splitting error tolerable. We impose a zero Dirichlet boundary condition on the dimensionless velocity on all six boundary walls of the computational domain. We consider two sets of (ρ_r^*, Ga^*). In the first set $(\rho_r^*, Ga^*) = (0.2, 100)$, the trajectory as predicted by PeliGRIFF in Seyed-Ahmadi and Wachs (2019) is vertical. We run the same computation with our octree DLM/FD method with a smallest dimensionless grid size of $1/24$. We also find a vertical motion and perfectly reproduce the time evolution of the vertical velocity computed with PeliGRIFF. Indeed, in figure 12a, the two curves are almost superposed and the final rising

velocity predicted by our octree DLM/FD method differs by less than 2% from the final rising velocity predicted with PeliGRIFF.



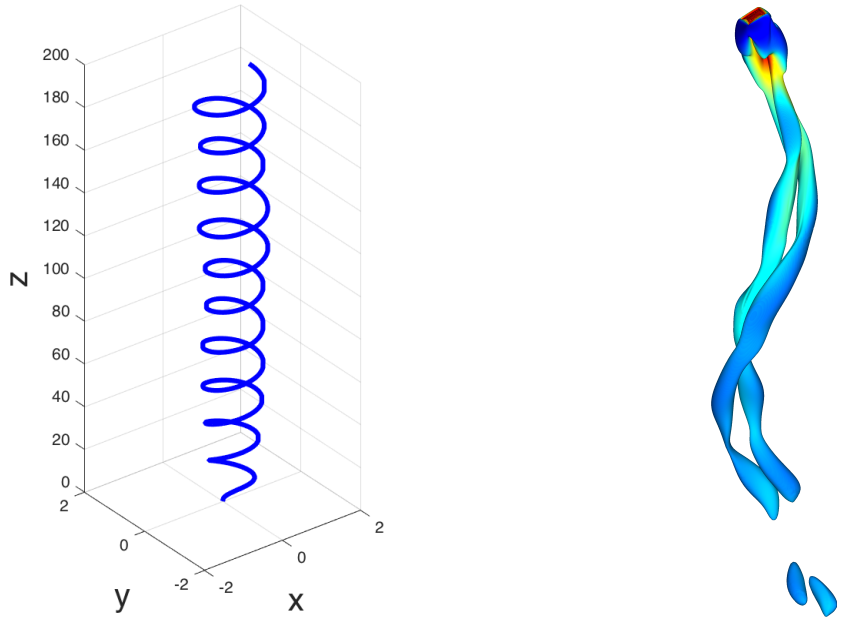
(a) Vertical velocity (b) Wake

Figure 12: Vertical rising motion of the cube at $(\rho_r^*, Ga^*) = (0.2, 100)$: (a) dimensionless rising velocity as a function of dimensionless time computed by our octree DLM/FD method and PeliGRIFF and (b) vortex thread in the wake visualized with the Q criterion and colored by the vertical velocity (red is max and blue is min)

In the second set $(\rho_r^*, Ga^*) = (0.5, 140)$, the trajectory as predicted with PeliGRIFF in Seyed-Ahmadi and Wachs (2019) is helical. We run the same computation with our octree DLM/FD method with a smallest dimensionless grid size of 1/48 and resolve the wake accurately as Seyed-Ahmadi and Wachs (2019) pointed out that the dynamics is primarily controlled by the vortex thread in the wake of the cube. To speed up the onset of the instability that leads to the helical motion, we initially tilt the cube by 20^o in each direction. Again we predict very well the helical regime with the same dimensionless amplitude of ≈ 1.4 found in Seyed-Ahmadi and Wachs (2019). We show in figure 13 the helical motion of the rising cube and the intertwined vortex thread in the wake that attests of the helical motion.
After about eight complete revolutions in the xy horizontal plane along its helical trajectory, the cube has risen by about 200. The total number of grid cells has reached a pseudo-stationary number of about $5,000,000$. Given the large size of the box of 700 and the requested smallest dimensionless grid size of 1/48, the grid comprises 15 levels and enables to model very large domains without any particular problem. The octree grid is shown in figure 14. In comparison, the simple regular Cartesian grid in PeliGRIFF imposes to adopt a specific strategy to model a single particle in a very large domain. Since the whole computational domain cannot be arbitrarily large without adaptive mesh refinement, the strategy implemented in PeliGRIFF is based on selecting a computational domain of rather limited size and translating it to follow the particle motion (Rahmani and Wachs, 2014; Seyed-Ahmadi and Wachs, 2019). Our octree DLM/FD method avoids such complications.

6 Attempt to capture lubrication forces with adaptive grids

In this section, we attempt to capture the lubrication forces acting on one or two spheres in a Stokes flow (for which analytical solutions are available) by fully resolving the short length scale relevant to lubrication with our octree DLM/FD method.



(a) Trajectory (b) Wake

Figure 13: Helical rising motion of the cube at $(\rho_r^*, Ga^*) = (0.5, 140)$: (a) cube trajectory and (b) intertwined vortex thread in the wake visualized with the Q criterion and colored by the vertical velocity (rex is max and blue is min)

6.1 One sphere close to a plane wall

We consider the ideal case of a semi-infinite domain bounded by a rigid plane wall at the bottom. A rigid sphere of radius $a = D/2$ moves toward the bottom wall in the normal direction to the bottom wall with an imposed constant velocity U_s. As shown by Brenner (1961) and Cooley and O'Neill (1969), the normal force F_n acting on the sphere is given by the following expression:

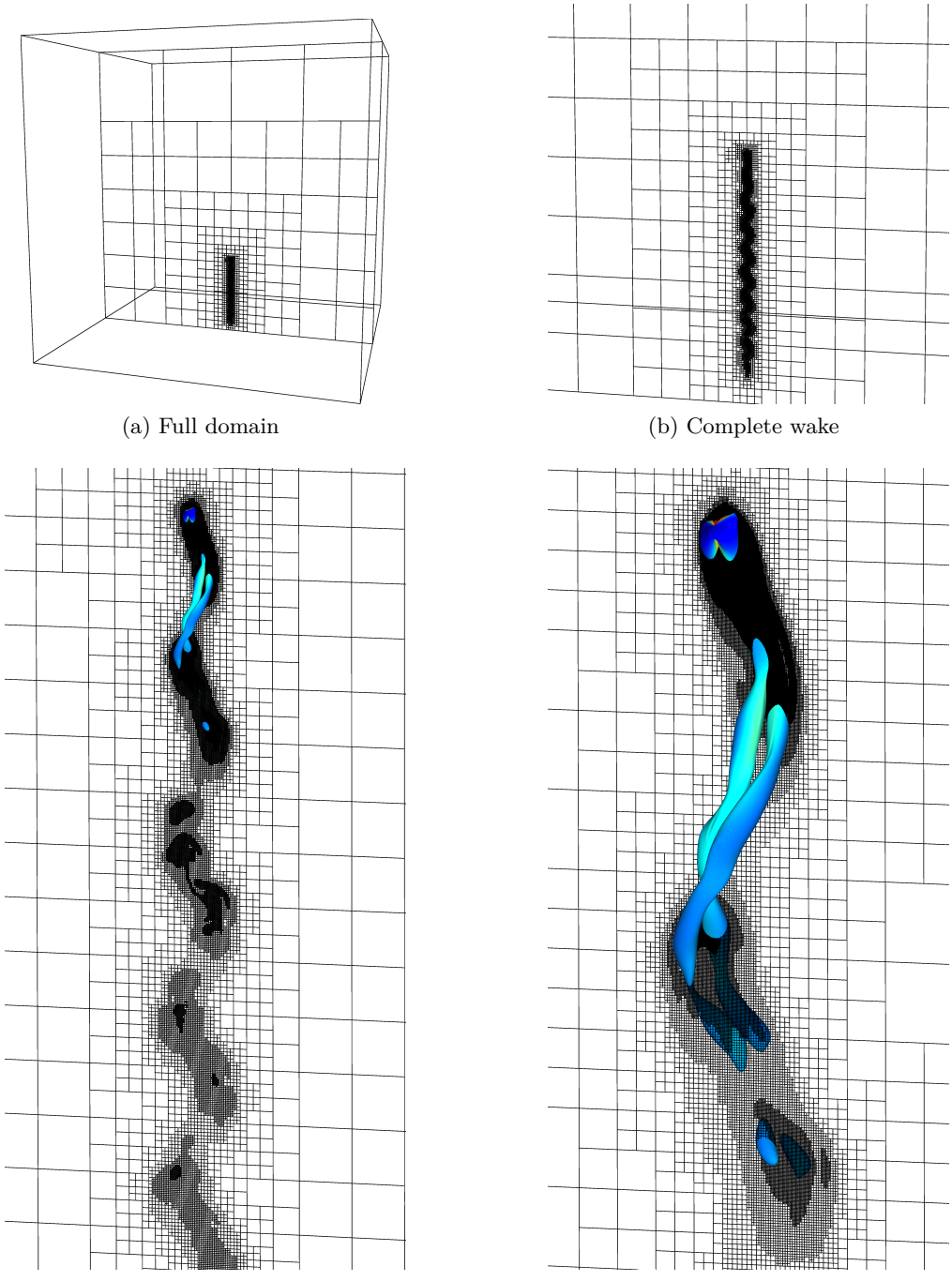
$$F_n/F_{st} = F_n^* = (\delta/a)^{-1} - \frac{1}{5}\log(\delta/a) + 0.97128, \tag{58}$$

where $F_{st} = 6\pi\mu_f a U_s$ and δ are the Stokes force in an unbounded domain and the gap distance, respectively. The arising difficulty is that the force increases fast when δ/a decreases and eventually diverges as δ/a tends toward zero. We compute F_n with our octree DLM/FD method and compare our results to equation (58). We also pay a particular attention to the temporal and spatial convergence of our computed solutions.

6.1.1 Numerical setup and results

The flow configuration is depicted in figure 15. We use the sphere radius a, the imposed velocity U_s and the viscous time $\rho_f a^2/\mu_f$ as characteristic length, characteristic velocity and characteristic time, respectively. The flow domain is a cubic box of dimensionless edge length $L^* = 60$. We impose a zero Dirichlet boundary condition on the dimensionless velocity over the top and bottom walls and periodic boundary conditions on the left/right and front/back walls to mimic the semi-infinite domain. Such a large box size ensures that the images of the periodic sphere are sufficiently far away not to affect the global dynamics. The fluid is initially at rest. The dimensionless velocity of the sphere is imposed constant to $(\boldsymbol{U}^*, \boldsymbol{\omega}^*) = ((0,-1,0),(0,0,0))$ through time while maintaining the sphere at a constant position in order to establish the steady Stokes flow for a given $\delta^* = \delta/a$. We set $Re^* = 0.005$ to model a quasi-pure Stokes flow.

In order to assess the temporal and spatial convergence of our computed solutions, we run simulations with various time steps and grid sizes independently. The time evolution of



(a) Full domain

(b) Complete wake

(c) Zoom in the wake

(d) Zoom in the close wake

Figure 14: Different views of the 15-level grid in the case of the rising cube at $(\rho_r^*, Ga^*) = (0.5, 140)$ after 8 full revolutions in the xy plane along its helical trajectory

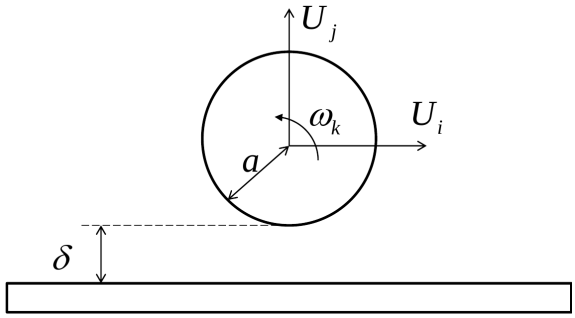


Figure 15: Sphere-wall interaction: flow configuration and notations.

the normal force F_n^* for two different gap distances $\delta^* = 0.1$ and $\delta^* = 0.3$ with different spatial and temporal resolutions is plotted in figures 16-a and 16-b. In all cases, we observe that after a short transient evolution, the flow reaches a steady state as expected. The sensitivity to the numerical parameters is more visible for the more challenging case $\delta^* = 0.1$ but both spatial and temporal refinements lead to values of F_n^* close to the analytical prediction (58). The better agreement with respect to the analytical solution for $\delta^* = 0.3$ results from the larger number of fluid cells within the gap than for $\delta^* = 0.1$ for the same level of refinement. The minimum dimensionless grid size we consider is $\Delta^* = \Delta/a = 1/136$ corresponding to 13 levels of refinement. It amounts to $\delta^*/\Delta^* \sim 40$ cells within the fluid gap for $\delta^* = 0.3$ and to $\delta^*/\Delta^* \sim 13$ for $\delta^* = 0.1$. We plot all results in figure 17 together with the analytical solution (58). We obtain a very good agreement for all values of δ^* considered, especially for the smallest $\delta^* = 0.025$ corresponding to the most challenging case.

6.2 Two spheres moving towards each other in a creeping shear flow

We consider the case of two neutrally buoyant and identical spheres in a simple creeping shear flow moving in opposite direction. The analytical solution was derived by Batchelor and Green (1972) and Lin et al. (1970) for the particle separation vector $\boldsymbol{r} = \boldsymbol{r_1} - \boldsymbol{r_2}$ where $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the position vectors of the first sphere and the second sphere, respectively. Using the sphere radius a as a characteristic length and the inverse shear rate $\dot{\gamma}^{-1}$ as a characteristic time, the analytical solution for $\boldsymbol{r}^*$ is obtained by solving the following set of ordinary differential equations:

$$\frac{\mathrm{d}r_x^*}{\mathrm{d}t} = r_y^* + e r_x^* - \frac{B\left(\boldsymbol{r}^*\right)}{2} r_y^*, \tag{59}$$

$$\frac{\mathrm{d}r_y^*}{\mathrm{d}t} = e r_y^* - \frac{B\left(\boldsymbol{r}^*\right)}{2} r_x^*, \tag{60}$$

$$\frac{\mathrm{d}r_z^*}{\mathrm{d}t} = e r_z^*, \tag{61}$$

where

$$e = \frac{r_x^* r_y^* \left(B\left(\boldsymbol{r}^*\right) - A\left(\boldsymbol{r}^*\right)\right)}{\boldsymbol{r}^{*2}}, \tag{62}$$

and $A\left(\boldsymbol{r}^*\right), B\left(\boldsymbol{r}^*\right)$ are dimensionless scalar functions of $\boldsymbol{r}^*$. Their values are tabulated in Batchelor and Green (1972) and they are reported here in table 2 for the sake of completeness. Our goal is to compute the motion of the two spheres with our octree DLM/FD method and to compare our result for $\boldsymbol{r}^*$ with the analytical solution obtained by solving (59)-(61). We consider an initial separating distance of $\boldsymbol{r_0^*} = (-10, 1, 0)$ and we solve (59)-(61) with a fourth order Runge-Kutta integration scheme (with the open source software octave and its built-in "ode45" function). The analytical solution provides insight into the length scale that we need to resolve accurately in our numerical simulation. The

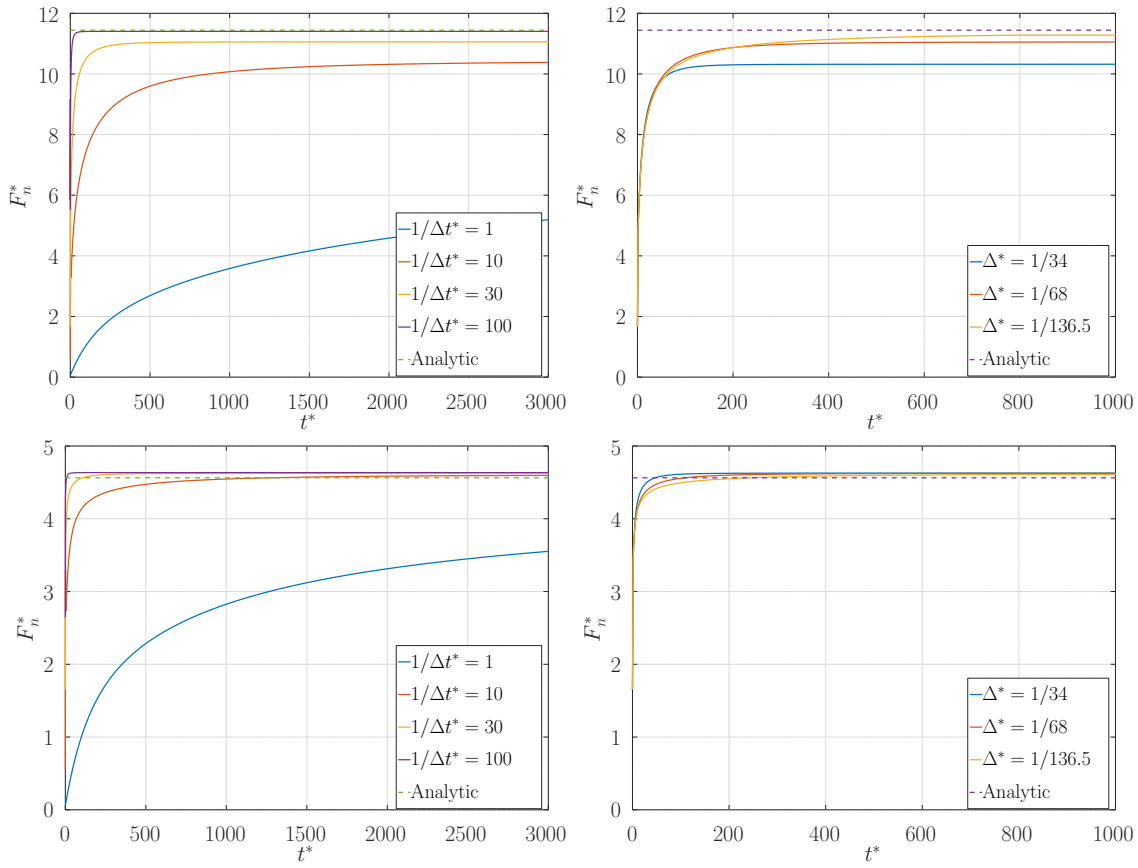


Figure 16: Time evolution of the normal force F_n^* acting on the sphere for two different gap distances: $\delta^* = 0.1$ (top row) and $\delta^* = 0.3$ (bottom row). Left column: effect of the time-step with fixed $\Delta^* = 1/136$. Right column: effect of the spatial resolution with time-step fixed at $\Delta t^* = 1/30$. In all plots, the dashed lines represent the analytical prediction of Brenner (1961) and Cooley and O'Neill (1969).

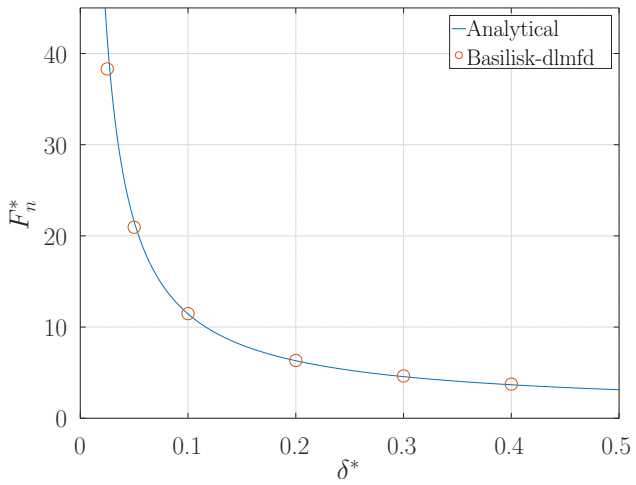


Figure 17: Comparison of our computed lubrication force for various gap distances against the analytical prediction of Brenner (1961) and Cooley and O'Neill (1969).

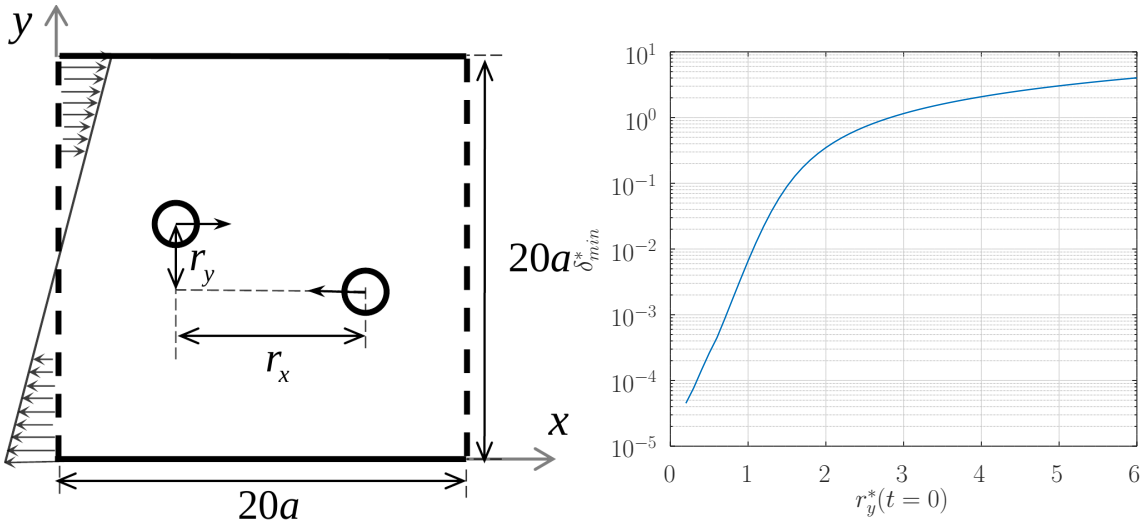


Figure 18: (a) Two spheres in a creeping shear flow: numerical setup and notations. (b) Minimum gap distance $\delta^*_{min} = \delta_{min}/a$ as a function of the initial position vector $\boldsymbol{r}^*_0 = (-10, r_y(t=0)^*, 0)$ for various $r_y(t=0)^*$ obtained by solving (59)-(61).

minimum dimensionless gap distance $\delta^*_{min} = \delta_{min}/a$ within which the fluid film is squeezed between the two spheres is plotted in figure 18-(b) as a function of the initial dimensionless separation distance in the cross-flow direction $r_y(t=0)^* = r_y(t=0)/a$. For the initial $r_y(t=0)^* = 1$ considered, the minimum dimensionless gap distance (and length scale) that we need to resolve is $\delta^*_{min} \sim 5 \cdot 10^{-3}$. This underscores how challenging this flow configuration is in terms of required grid resolution. Besides, we purposely do not use here any collision model in our simulations so that equations (31) and (32) have no force and torque term due to the contact (the granular solver is only used to predict particles' positions and velocities). As a consequence, the repelling force that the two spheres experience only results from the thin fluid film squeezed between them as they approach towards each other. This implies that for the cases where the lubrication force is not fully captured, the two spheres' domains end up overlapping each other. This scenario, although not realistic, is carefully handled at the numerical level. For the Lagrange multipliers lying on the surface, we remove the ones for which the associated stencil extends to the interior of the other sphere. For the interior part of the domains, a fluid cell is set to be constrained by the Lagrange-multiplier of only one sphere even if the given cell is located inside both of them (otherwise the problem would be over-constrained). This procedure is equivalent to locally "pixelating" the spheres around the regions of the contact and has already been used successfully in our previous works (Wachs, 2009).

6.2.1 Numerical setup and results

The flow configuration is shown in figure 18. The computational domain is a cube of dimensionless edge length $L^* = 20$. The flow and the particles are initially at rest and the initial separation vector between the two spheres is $\boldsymbol{r}^*_{\boldsymbol{0}} = (-10, 1, 0)$. The streamwise direction of the flow is set to the x direction. The boundary conditions are as follows. We impose periodicity in the x and z directions and Dirichlet boundary conditions on the dimensionless velocity of the form $(-10, 0, 0)$ and $(10, 0, 0)$ on the bottom $y^* = -0.5L^*$ wall and the top $y^* = 0.5L^*$ wall, respectively, in order to impose a dimensionless linear shear rate $\dot{\gamma}^* = 1$. We set the particle Reynolds number defined as $Re^*_p = \rho_f a \dot{\gamma} L/(2\mu_f)$ to 0.001 and the dimensionless time step $\Delta t^* = \Delta t \gamma$ to 10^{-4}. The solution computed by our octree DLM/FD method in terms of $r^*_y = r_y/a$ as a function of $r^*_x = r_x/a$ for various spatial resolutions $\Delta^* = \Delta/a$ is plotted in figure 19.

We observe that even for our coarsest spatial resolution $\Delta^* = 1/12.5$ a large portion of

$\lvert\boldsymbol{r}^*\rvert$	$A\left(\lvert\boldsymbol{r}\rvert^*\right)$	$B\left(\lvert\boldsymbol{r}\rvert^*\right)$
20.1353	0.0006	0.0000
11.1139	0.0036	0.0000
6.2149	0.0204	0.0006
4.7048	0.0468	0.0023
3.6213	0.1033	0.0086
3.3370	0.1331	0.0130
3.0862	0.1704	0.0193
2.8662	0.2167	0.0281
2.6749	0.2735	0.0399
2.5103	0.3424	0.0553
2.3709	0.4248	0.0748
2.2553	0.5214	0.0988
2.1621	0.6313	0.1275
2.0907	0.7505	0.1608
2.0401	0.8679	0.1996
2.0100	0.9619	0.2461
2.0025	0.9900	0.2762
2.0006	0.9975	0.2968

Table 2: Values for the function A and B in equations (59)-(61) describing the velocities of two identical spheres derived from the numerical data of (Lin et al., 1970).

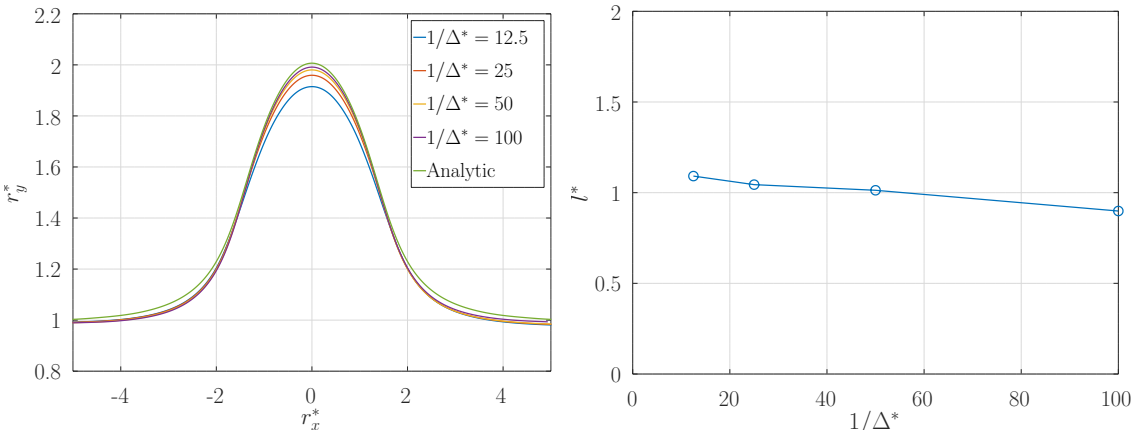
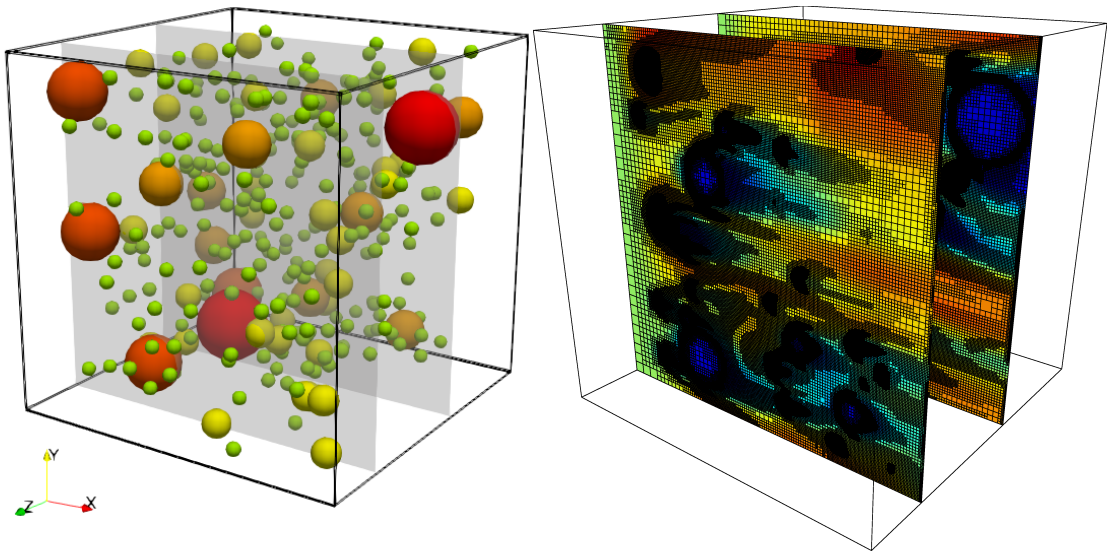


Figure 19: (a) Comparison of inter-particle trajectory $r_y^* = r_y/a$ vs $r_x^* r_x/a$ for various spatial resolution $\Delta^* = \Delta/a$ against the analytical solution to equations (59)-(61). (b) Overlapping length l/a between the two particles as a functions of the inverse spatial resolution a/Δ.

the lubrication force is already accurately captured. As we increase the level of refinement further, we obtain a better agreement with the analytical solution. The missing portion of the lubrication force can be directly related to the overlapping distance l of the two spheres, i.e., the difference between the analytical r_y^* for $r_x^* = 0$ and the r_y^* for $r_x^* = 0$ computed with the various spatial resolutions. We report this overlapping distance in figure 19-(b) as a function of $1/\Delta^*$ and we observe that it is always close to 1, i.e., one smallest grid size indicating a (relative) lack of spatial resolution. Indeed, assuming that we need at (the very) least one fluid cell within the fluid gap between the two spheres to properly compute the flow, i.e. $\delta_{min}/\Delta \geq 1$, it would require a spatial resolution of $\Delta^* \leq 1/200$.

7 Polydisperse suspensions with adaptive DLM/FD



(a) Array of polydisperse particles (b) AMR grid and flow field

Figure 20: Flow through an array of polydisperse spheres of size ratio 5: (a) the five classes of particles of dimensionless diameter 1, 0.8, 0.6, 0.4 and 0.2 and (b) adaptive grid and x (streamwise) velocity countours (blue is min and red is max) in the two xy cut planes shown in grey in (a).

Another possible application of interest for our adaptive DLM/FD method is polydisperse suspensions that have been largely unexplored in the literature with particle-resolved simulation tools, in particular when the particle size distribution is broad. In fact, with a simple regular Cartesian grid and assuming that we intend to resolve all particles properly, the constant grid size in the domain would be dictated by the smallest particles, resulting in a very large number of cells, presumably larger than what available computing resources allow. This would restrict the largest particle to smallest particle size ratio to low values in order to render the computation feasible (as, e.g., a size ratio of 2 as in (Van der Hoef et al., 2005; Beetstra et al., 2007). With our adaptive grid method, we can consider any size ratio and still limit the number of cells to a very tractable number. As an illustration below, we compute the flow past a fixed array of polydisperse spheres of size ratio 5. The Reynolds number based on the largest sphere is 50. We consider 5 classes of sphere of dimensionless diameter 1, 0.8, 0.6, 0.4 and 0.2. The number of spheres in each class is 2, 4, 10, 32 and 250, respectively, such that the total volume of all spheres in each class is approximately the same. The resulting solid volume fraction is 0.032. The computational domain is a

cube of dimensionless edge length 5. We set the number of levels in our adaptive grid to 10 such that the lowest level corresponds to a dimensionless grid size of $\approx 0.2/40$, i.e., the smallest spheres are resolved with a grid size equivalent to ≈ 40 points per diameter. This is generally considered as a very fine resolution, even an excessively fine resolution in such a flow regime and we could have easily accommodated even smaller particles of, e.g., dimensionless diameter 0.1 and hence considered a size ratio of 10. Anyhow, a simple regular Cartesian grid method would hence require a 1024^3 grid for a similar resolution of small particles, thus comprising $\approx 1,000,000,000$ grid cells. The corresponding computation would likely run on about $1,000$ cores, assuming a $1,000,000$ grid cell load per process/core. Here the 10 levels of the adaptive grid result in $\approx 31,000,000$ grid cells only, hence a substantial reduction of a factor of 32 of the required number of grid cells with respect to a simple regular Cartesian grid method, and correspondingly of the required computing resources. The computation ran on 96 cores only, and could have easily run on fewer cores if needed. Figure 20a shows the array of polydisperse particles and figure 20b illustrates the adaptive grid together with contours of the streamwise velocity in two xy cut planes.

8 Discussion and Perspectives

We extended our DLM/FD method for the particle-resolved simulation of particle-laden flows to quadtree-octree adaptive grids. The method retains the robust convergence properties, the inconditional stability as a function of the solid to fluid density ratio thanks to the implicit computation of the hydrodynamic force and torque and the temporal/spatial convergence properties observed when implemented on a simple regular Cartesian grid, i.e., order 1 in time and order 1 in space (Wachs et al., 2015). In practice, we implemented a DLM/FD method in Basilisk and addressed specific issues related to the parallelization of the Uzawa algorithm in a quadtree-octree framework. The key point of the implementation is to impose an uniform grid size in a narrow band of the boundary layer around the particles in order to use a standard quadratic interpolation defined on a regular $3 \times 3 \times 3$ stencil for the velocity reconstruction. We tested and successfully validated our new quadtree-octree DLM/FD method on a set of flow configurations, including challenging problems dominated by lubrication forces or involving particle of non-spherical shape.

Our new DLM/FD implementation on quadtree-octree adaptive grids opens up unprecedented opportunities for the particle-resolved computation of particle-laden flows that were unattainable with an implementation on a simple regular Cartesian grid. As any other immersed boundary/fictitious domain method, our method can be straightforwardly extended to heat or mass transfer in the case of small Biot numbers, i.e., when temperature or chemical species gradients inside the rigid body are negligible and the assumption of uniform temperature or chemical species over the entire rigid body volume is acceptable. In the case of small Biot numbers, there are many heat and mass transfer problems at large Prandtl Pr or Schmidt Sc numbers that cannot be computed with a simple regular Cartesian grid method. Most of the published literature focuses on $Pr \approx 1$ and $Sc \approx 1$ cases to guarantee that thermal/mass boundary layers around rigid bodies have approximately the same thickness as momentum boundary layers. As reported by (Deen et al., 2014), in order to tackle systems with high high values of Sc and/or Pr efficient grid refinement techniques are clearly mandatory. Recently, the same authors have suggested an approach to address moderate Reynolds but high Pr particle-laden flows that combines solving the flow field on a constant, regular and relatively coarse Cartesian grid with an immersed boundary/ghost cell method and solving the heat equation on an adaptive octree grid with a staircase (i.e., pixelated) representation the particle surface (Panda et al., 2019).

Our proposed octree adaptive grid implementation of the DLM/FD method is another contribution in the direction of computing high Reynolds, Sc and/or Pr particle-laden flows.
Restricting opportunities to momentum transfer only, our method is also well suited to rigid bodies of complex shape, in particular with angular edges and corners as shown in Section 5.5. In fact, the local mesh refinement capability of our method enables us to refine the grid in the vicinity of geometric singularities as sharp edges and corners of non-spherical particles. Coupled to our granular solver Grains3D to compute particle-particle collisions for any particle shape, our method shows a large potential for the accurate computation of flows laden with particles of complex shape.
Another application of interest that can now be examined with significantly fewer computing resources thanks to local mesh refinement is inertial particle-laden flows and particle-induced turbulence in the dilute regime, i.e., at low solid volume fraction, when the background flow itself is not yet turbulent (and hence does not require a fine resolution everywhere in the flow domain).
As usual with the DLM/FD method, its main strength is also its main drawback. In fact, by formulating the hydrodynamic interaction problem as a saddle-point problem, the DLM/FD method computes the hydrodynamic force and torque implicitly, unlike weak coupling/direct-forcing immersed boundary methods, and in particular computes the added mass contribution to the hydrodynamic force and torque implicitly. This enhances the coupling of the fluid-solid equations and gives unconditional stability to the method with respect to the solid to fluid density ratio. However, the solution of the DLM/FD saddle point problem requires the use of the iterative Uzawa algorithm, and we may postulate that the DLM/FD method is as many times more expensive than a direct forcing method as the number of iterations required by the Uzawa algorithm to converge. In practice, the Uzawa algorithm applied to the DLM/FD problem requires between 10 and 30 iterations to converge. This can represent a substantial computing overhead with respect to weak coupling/direct-forcing methods. However, this postulation may be flawed as the iterative process of the Uzawa algorithm represents a part only of the total overhead related to the consideration of immersed rigid particles. In fact, there are many, mostly geometric, operations that our code performs before the Uzawa algorithm iterates such as, e.g., the determination of the set of points representing immersed bodies, the determination of their location of the grid or the computation of the coefficients of the reconstruction polynomial for each boundary point. All these operations are common to a direct forcing method, such that the overhead of the iterative Uzawa algorithm may not be that signficant in the overall computing time when compared to a direct forcing method. This requires a deeper analysis.
For now, the computational performance of the method is roughly as follow: in most computations, the Navier-Stokes solution costs about 60 to 70% of the total computing time and the DLM/FD problem about 30 to 40%. There is potentially room for improvement. Also, we have so far tested our implementation on a moderate number of cores only. In fact, most of the computations presented in this paper are performed on up to 256 cores. We have not noticed any significant loss of performance with the increase of the number of cores up to 256 cores. This seems to indicate that our implementation of the DLM/FD method in Basilisk scales well. However, the Navier-Stokes solver of Basilisk has been shown to scale well on up to multiple thousands of cores, showing promises for very large-scale computations. To run jobs on $O(10^4)$ cores, the DLM/FD method is thus required to scale accordingly. Future work hence involves examining the scaling and the optimization of our quadtree-octree DLM/FD method on a larger number of cores.

8.1 Acknowledgement

We greatly appreciate the financial support of the Natural Sciences and Engineering Research Council of Canada via their Discovery Grant program. This research was enabled by support provided by Compute Canada (www.computecanada.ca) through Prof. Wachs' computing ressource allocation qpf-764-ab.

A Uzawa/conjugate-gradient algorithm in Basilisk

We present the implementation of the iterative Uzawa/Conjugate-gradient algorithm we use to solve the saddle-point problem of the fictitious domain step described in section 3.1.3. We follow closely the algorithm given in the annex of Wachs (2011) (minus some sign errors that are present in the original paper) transposed to Basilisk's matrix-free and Finite Volume framework.
In the following, Δv denotes the volume of a grid cell. The Uzawa/conjugate-gradient algorithm proceeds as follows:

- Initialisation $k = 0$:
  - Given an initial guess $\boldsymbol{\lambda}^0$ (we use either $\boldsymbol{\lambda}^0 = \mathbf{0}$ or the Lagrange multipliers of the previous time-step $\boldsymbol{\lambda}^0 = \boldsymbol{\lambda}^n = \boldsymbol{\lambda}(t^n = n\Delta t)$) and solutions from the previous problems $\boldsymbol{U}_i^{n+1/2}$, $\boldsymbol{\omega}_i^{n+1/2}$, $\boldsymbol{u}^{n+1/2}$ compute vectors $\boldsymbol{q_u}$ and $(\boldsymbol{q_{U_i}}, \boldsymbol{q_{\omega_i}})$, $i = 1, \dots, N$ as follows:

$$\boldsymbol{q_u} = \frac{\rho_f \boldsymbol{u}^{n+1/2}}{\Delta t}\Delta v + \sum_{i=1}^{N} < \boldsymbol{\lambda}^n - \boldsymbol{\lambda}_{\boldsymbol{il}}^0, \boldsymbol{v} >_{P_i} = \frac{\rho_f \boldsymbol{u}^{n+1/2}}{\Delta t}\Delta v + \sum_{i=1}^{N}\sum_{l=1}^{L_i}(\boldsymbol{\lambda}^n - \boldsymbol{\lambda}_{\boldsymbol{il}}^0)\,\boldsymbol{v}\,(\boldsymbol{x_{il}}),$$

$$\boldsymbol{q_{U_i}} = M_i\left(1 - \frac{\rho_f}{\rho_{s_i}}\right)\left(\frac{\boldsymbol{U}_i^{n+1/2}}{\Delta t} + \boldsymbol{g}\right) + \sum_{l=1}^{L_i}\boldsymbol{\lambda}_{\boldsymbol{il}}^0 \;,\; i = 1, \dots, N,$$

$$\boldsymbol{q_{\omega_i}} = \left(1 - \frac{\rho_f}{\rho_{s_i}}\right)\boldsymbol{I_i}\frac{\boldsymbol{\omega}_i^{n+1/2}}{\Delta t} + \sum_{l=1}^{L_i}\boldsymbol{r_{il}} \times \boldsymbol{\lambda}_{\boldsymbol{il}}^0 \;,\; i = 1, \dots, N.$$

  - Compute:

$$\boldsymbol{u} = \frac{\boldsymbol{q_u}\Delta t}{\rho_f \Delta v};\quad \boldsymbol{U_i} = \frac{\boldsymbol{q_{U_i}}\Delta t}{(1 - \rho_f/\rho_{s_i})\,M_i} \;,\; \boldsymbol{\omega_i} = \left[\frac{1 - \rho_f/\rho_{s_i}}{\Delta t}\boldsymbol{I_i}\right]^{-1}\boldsymbol{q_{\omega_i}} \;,\; i = 1, \dots, N.$$

  - Compute the initial residual $\boldsymbol{r}^0$:

$$\boldsymbol{r}^0 = -\sum_{i=1}^{N} < \boldsymbol{\alpha}, \boldsymbol{u} - (\boldsymbol{U_i} + \boldsymbol{\omega_i} \times \boldsymbol{r_i}) >_{P_i} = -\sum_{i=1}^{N}\sum_{l=1}^{L_i}(\boldsymbol{u}\,(\boldsymbol{x_{il}}) - (\boldsymbol{U_i} + \boldsymbol{\omega_i} \times \boldsymbol{r_{il}})).$$

  - Set $\boldsymbol{s}^0 = \boldsymbol{r}^0$

- Iterative procedure: for $k > 0$ do while $\| \boldsymbol{r} \| < \epsilon$

  (1) Compute vectors $\boldsymbol{q_u}$ and $(\boldsymbol{q_{U_i}}, \boldsymbol{q_{\omega_i}})$, $i = 1, \dots, N$ as follows:

$$\boldsymbol{q_u} = \sum_{i=1}^{N} < \boldsymbol{s}^{k-1}, \boldsymbol{v} >_{P_i} = \sum_{i=1}^{N}\sum_{l=1}^{L_i}\boldsymbol{s}_{\boldsymbol{il}}^{\boldsymbol{k-1}}\,\boldsymbol{v}\,(\boldsymbol{x_{il}}),$$

$$\boldsymbol{q_{U_i}} = -\sum_{l=1}^{L_i}\boldsymbol{s}_{\boldsymbol{il}}^{k-1} \;,\; i = 1, \dots, N,$$

$$\boldsymbol{q_{\omega_i}} = -\sum_{l=1}^{L_i}\boldsymbol{r_{il}} \times \boldsymbol{s}_{\boldsymbol{il}}^{k-1} \;,\; i = 1, \dots, N.$$

(2) Compute:

$$\boldsymbol{t_u} = \frac{\boldsymbol{q_u}\Delta t}{\rho_f \Delta v}; \quad \boldsymbol{t_{U_i}} = \frac{\boldsymbol{q_{U_i}}\Delta t}{(1-\rho_f/\rho_{s_i})\, M_i} \ , \ \boldsymbol{t_{\omega_i}} = \left[\frac{1-\rho_f/\rho_{s_i}}{\Delta t}\boldsymbol{I_i}\right]^{-1} \boldsymbol{q_{\omega_i}} \ , \ i = 1, \ldots, N.$$

(3) Compute the residual $\boldsymbol{y}$:

$$\boldsymbol{y} = \sum_{i=1}^{N} < \boldsymbol{\alpha}, \boldsymbol{t_u} - (\boldsymbol{t_{U_i}} + \boldsymbol{t_{\omega_i}} \times \boldsymbol{r_i}) >_{P_i} = \sum_{i=1}^{N}\sum_{l=1}^{L_i} (\boldsymbol{t_u}\,(\boldsymbol{x_{il}}) - (\boldsymbol{t_{U_i}} + \boldsymbol{t_{\omega_i}} \times \boldsymbol{r_{il}})).$$

(4) Compute scalar γ:

$$\gamma = \frac{\boldsymbol{r^{k-1}} \cdot \boldsymbol{r^{k-1}}}{\boldsymbol{s}^{k-1} \cdot \boldsymbol{y}}.$$

(5) Compute new Lagrange multiplicators $\boldsymbol{\lambda}^k$ and residual $\boldsymbol{r}^k$ as follows:

$$\boldsymbol{\lambda}^k = \boldsymbol{\lambda}^{k-1} - \gamma\, \boldsymbol{s}^{k-1},$$
$$\boldsymbol{r}^k = \boldsymbol{r}^{k-1} - \gamma\, \boldsymbol{y}.$$

(6) Update solutions:

$$\boldsymbol{u}^k = \boldsymbol{u}^{k-1} + \gamma \boldsymbol{t_u}; \quad \boldsymbol{U_i^k} = \boldsymbol{U_i^{k-1}} + \gamma \boldsymbol{t_{U_i}} \ , \ \boldsymbol{\omega_i^k} = \boldsymbol{\omega_i^{k-1}} + \gamma \boldsymbol{t_{\omega_i}} \ , \ i = 1, \ldots, N.$$

(7) Compute $\| \boldsymbol{r}^k \|$.

(8) Compute scalar β:

$$\beta = \frac{\| \boldsymbol{r}^k \|^2}{\| \boldsymbol{r}^{k-1} \|^2}.$$

(9) Update $\boldsymbol{s}^k$:

$$\boldsymbol{s}^k = \boldsymbol{r}^k + \beta \boldsymbol{s}^{k-1}.$$

Note that the whole implementation of the method along with various examples and test cases are freely available on Basilisk's website on the following link http://basilisk.fr/sandbox/cselcuk/.

References

GK Batchelor and J-T_ Green. The hydrodynamic interaction of two small freely-moving spheres in a linear flow field. *Journal of Fluid Mechanics*, 56(2):375–400, 1972.

R. Beetstra, M.A. Van der Hoef, and J.A.M. Kuipers. Drag force of intermediate Reynolds number flow past mono-and bidisperse arrays of spheres. *AIChE journal*, 53(2):489–501, 2007.

John B Bell, Phillip Colella, and Harland M Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 85(2):257–283, 1989.

Roger Bouard and Madeleine Coutanceau. The early stage of development of the wake behind an impulsively started cylinder for $40 < \mathrm{Re} < 10^4$. *Journal of Fluid Mechanics*, 101(3):583–607, 1980.

Howard Brenner. The slow motion of a sphere through a viscous fluid towards a plane surface. *Chemical Engineering Science*, 16(3):242 – 251, 1961. ISSN 0009-2509. doi: https://doi.org/10.1016/0009-2509(61)80035-3. URL http://www.sciencedirect.com/science/article/pii/0009250961800353.

D. Calhoun and R. J. LeVeque. A Cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries. *Journal of Computational Physics*, 157(1):143–180, 2000.

Choeng Ryul Choi and Chang Nyung Kim. Direct numerical simulations of the dynamics of particles with arbitrary shapes in shear flows. *Journal of Hydrodynamics, Ser. B*, 22(4):456–465, 2010.

Meng-Hsuan Chung. An adaptive Cartesian cut-cell/level-set method to simulate incompressible two-phase flows with embedded moving solid boundaries. *Computers & Fluids*, 71:469–486, 2013.

M. D. A. Cooley and M. E. O'Neill. On the slow motion generated in a viscous fluid by the approach of a sphere to a plane wall or stationary sphere. *Mathematika*, 16(1):37–49, 1969. doi: 10.1112/S0025579300004599.

G d'Avino and MA Hulsen. A comparison between a collocation and weak implementation of the rigid-body motion constraint on a particle surface. *International Journal for Numerical Methods in Fluids*, 64(9):1014–1040, 2010.

N.G. Deen, E.A.J.F Peters, J.T. Padding, and J.A.M. Kuipers. Review of direct numerical simulation of fluid–particle mass, momentum and heat transfer in dense gas–solid flows. *Chemical Engineering Journal*, 116:710–724, 2014.

Ferdaous Dorai, Carlos Moura Teixeira, Matthieu Rolland, Eric Climent, Manuel Marcoux, and Anthony Wachs. Fully resolved simulations of the flow through a packed bed of cylinders: Effect of size distribution. *Chemical Engineering Science*, 129:180–192, 2015.

G Eitel-Amor, M Meinke, and W Schröder. A lattice-Boltzmann method with hierarchically refined meshes. *Computers & Fluids*, 75:127–139, 2013.

Daniel Fuster, Anne Bagué, Thomas Boeck, Luis Le Moyne, Anthony Leboissetier, Stéphane Popinet, Pascal Ray, Ruben Scardovelli, and Stéphane Zaleski. Simulation of primary atomization with an octree adaptive mesh refinement and VOF method. *International Journal of Multiphase Flow*, 35(6):550–565, 2009.

Frederic Gibou, Ronald P Fedkiw, Li-Tien Cheng, and Myungjoo Kang. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *Journal of Computational Physics*, 176(1):205–227, 2002.

R. Glowinski, T.W. Pan, T.I. Hesla, and D.D. Joseph. A distributed Lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25(5):755–794, 1999. ISSN 0301-9322.

Roland Glowinski, Tsorng-Whay Pan, Todd I Hesla, Daniel D Joseph, and Jácques Periaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow. *Journal of Computational Physics*, 169(2):363–426, 2001.

S Haeri and JS Shrimpton. On the application of immersed boundary, fictitious domain and body-conformal mesh methods to many particle multiphase flows. *International Journal of Multiphase Flow*, 40:38–55, 2012.

Daniel Hartmann, Matthias Meinke, and Wolfgang Schröder. A strictly conservative cartesian cut-cell method for compressible viscous flows on adaptive grids. *Computer Methods in Applied Mechanics and Engineering*, 200(9-12):1038–1052, 2011.

Hans Johansen and Phillip Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *Journal of Computational Physics*, 147(1):60–85, 1998.

TA Johnson and VC Patel. Flow past a sphere up to a Reynolds number of 300. *Journal of Fluid Mechanics*, 378:19–70, 1999.

Jungwoo Kim, Dongjoo Kim, and Haecheon Choi. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of computational physics*, 171(1):132–150, 2001.

Petros Koumoutsakos and A Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *Journal of Fluid Mechanics*, 296:1–38, 1995.

Anthony JC Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of fluid mechanics*, 271:285–309, 1994.

Jongho Lee, Jungwoo Kim, Haecheon Choi, and Kyung-Soo Yang. Sources of spurious force oscillations from an immersed boundary method for moving-body problems. *Journal of computational physics*, 230 (7):2677–2695, 2011.

CJ Lin, KJ Lee, and NF Sather. Slow motion of two spheres in a shear field. *Journal of Fluid Mechanics*, 43(1):35–47, 1970.

JM López-Herrera, Stéphane Popinet, and MA Herrada. A charge-conservative approach for simulating electrohydrodynamic two-phase flows using volume-of-fluid. *Journal of Computational Physics*, 230(5): 1939–1955, 2011.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. In *ACM transactions on graphics (TOG)*, volume 23, pages 457–462. ACM, 2004.

Jiangtao Lu, Saurish Das, EAJF Peters, and JAM Kuipers. Direct numerical simulation of fluid flow and mass transfer in dense fluid-particle systems with surface reactions. *Chemical Engineering Science*, 176: 1–18, 2018.

Shev MacNamara and Gilbert Strang. Operator splitting. In *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 95–114. Springer, 2016.

M. Meinke, L. Schneiders, C. Günther, and W. Schröder. A cut-cell method for sharp moving boundaries in Cartesian grids. *Computers & Fluids*, 85:135–142, 2013.

Rajat Mittal, Haibo Dong, Meliha Bozkurttas, FM Najjar, Abel Vargas, and Alfred Von Loebbecke. A

versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of computational physics*, 227(10):4825–4852, 2008.

F. Mohaghegh and H.S. Udaykumar. Comparison of sharp and smoothed interface methods for simulation of particulate flows I: Fluid structure interaction for moderate reynolds numbers. *Computers & Fluids*, 140:39–58, 2016.

Mohaghegh, Fazlolah and Udaykumar, HS. Comparison of sharp and smoothed interface methods for simulation of particulate flows II: Inertial and added mass effects. *Computers & Fluids*, 143:103–119, 2017.

A. Panda, E.A.J.F. Peters, M.W. Baltussen, and J.A.M. Kuipers. Fully Resolved Scalar Transport for High Prandtl Number Flows using Adaptive Mesh Refinement. *Chemical Engineering Science: X*, 4: 100047, 2019.

Charles S Peskin. Flow patterns around heart valves: a numerical method. *Journal of computational physics*, 10(2):252–271, 1972.

Jean-Lou Pierson, Franck Auguste, Abdelkader Hammouti, and Anthony Wachs. Inertial flow past a finite-length axisymmetric cylinder of aspect ratio 3: Effect of the yaw angle. *Physical Review Fluids*, 4 (4):044802, 2019.

Stéphane Popinet. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. *Journal of Computational Physics*, 190(2):572–600, 2003.

Stéphane Popinet. A quadtree-adaptive multigrid solver for the Serre–Green–Naghdi equations. *Journal of Computational Physics*, 302:336–358, 2015.

Mona Rahmani and Anthony Wachs. Free falling and rising of spherical and angular particles. *Physics of Fluids*, 26(8):083301, 2014.

Andriarimina Daniel Rakotonirina, Jean-Yves Delenne, Farhang Radjai, and Anthony Wachs. Grains3D, a flexible DEM approach for particles of arbitrary convex shape-Part III: extension to non-convex particles modelled as glued convex particles. *Computational Particle Mechanics*, 6(1):55–84, 2019.

Frederick W Roos and William W Willmarth. Some experimental results on sphere and disk drag. *AIAA journal*, 9(2):285–291, 1971.

Kai Schneider and Oleg V Vasilyev. Wavelet methods in computational fluid dynamics. *Annual review of fluid mechanics*, 42:473–503, 2010.

L. Schneiders, D. Hartmann, M. Meinke, and W. SchröDer. An accurate moving boundary formulation in cut-cell methods. *Journal of Computational Physics*, 235:786–809, 2013.

Arman Seyed-Ahmadi and Anthony Wachs. Dynamics and wakes of freely settling and rising cubes. *Physical Review Fluids*, 4(7):074304, 2019.

F. Sotiropoulos and X. Yang. Immersed boundary methods for simulating fluid–structure interaction. *Progress in Aerospace Sciences*, 65:1–21, 2014.

Tayfun E Tezduyar, Mittal Behr, S Mittal, and J Liou. A new strategy for finite element computations involving moving boundaries and interfaces—the deforming-spatial-domain/space-time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Computer methods in applied mechanics and engineering*, 94(3):353–371, 1992.

HS Udaykumar, Wei Shyy, and MM Rao. Elafint: a mixed Eulerian–Lagrangian method for fluid flows with complex and moving boundaries. *International journal for numerical methods in fluids*, 22(8): 691–712, 1996.

Markus Uhlmann. An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, 209(2):448–476, 2005.

M.A. Van der Hoef, R Beetstra, and J.A.M. Kuipers. Lattice-Boltzmann simulations of low-Reynolds-number flow past mono-and bidisperse arrays of spheres: results for the permeability and drag force. *Journal of Fluid Mechanics*, 528:233–254, 2005.

J Antoon van Hooft, Stéphane Popinet, Chiel C van Heerwaarden, Steven JA van der Linden, Stephan R de Roode, and Bas JH van de Wiel. Towards adaptive grids for atmospheric boundary-layer simulations. *Boundary-layer meteorology*, 167(3):421–443, 2018.

A. Wachs. A DEM-DLM/FD method for direct numerical simulation of particulate flows: Sedimentation of polygonal isometric particles in a Newtonian fluid with collisions. *Computers & Fluids*, 38(8):1608–1628, 2009.

A. Wachs. Particle-scale computational approaches to model dry and saturated granular flows of non-Brownian, non-cohesive, and non-spherical rigid bodies. *Acta Mechanica*, 230:1919–1980, 2019.

A. Wachs, A. Hammouti, G. Vinay, and M. Rahmani. Accuracy of Finite Volume/Staggered Grid Distributed Lagrange Multiplier/Fictitious Domain simulations of particulate flows. *Computers & Fluids*, 115:154–172, 2015.

Anthony Wachs. PeliGRIFF, a parallel DEM-DLM/FD direct numerical simulation tool for 3D particulate flows. *Journal of Engineering Mathematics*, 71(1):131–155, 2011.

Anthony Wachs, Laurence Girolami, Guillaume Vinay, and Gilles Ferrer. Grains3D, a flexible DEM approach for particles of arbitrary convex shape-Part I: Numerical model and validations. *Powder Technology*, 224:374–389, 2012.

Decheng Wan and Stefan Turek. Direct numerical simulation of particulate flow via multigrid FEM techniques and the fictitious boundary method. *International Journal for Numerical Methods in Fluids*, 51(5):531–566, 2006.

Z. Yu, N. Phan-Thien, and R.I. Tanner. Dynamic simulation of sphere motion in a vertical tube. *Journal of Fluid Mechanics*, 518:61–93, 2004.

Z. Yu, A. Wachs, and Y. Peysson. Numerical simulation of particle sedimentation in shear-thinning fluids with a fictitious domain method. *Journal of Non-Newtonian Fluid Mechanics*, 136(2):126–139, 2006.

Zhaosheng Yu. A DLM/FD method for fluid/flexible-body interactions. *Journal of computational physics*, 207(1):1–27, 2005.

A.A. Zick and G.M. Homsy. Stokes flow through periodic arrays of spheres. *Journal of Fluid Mechanics*, 115(1):13–26, 1982.