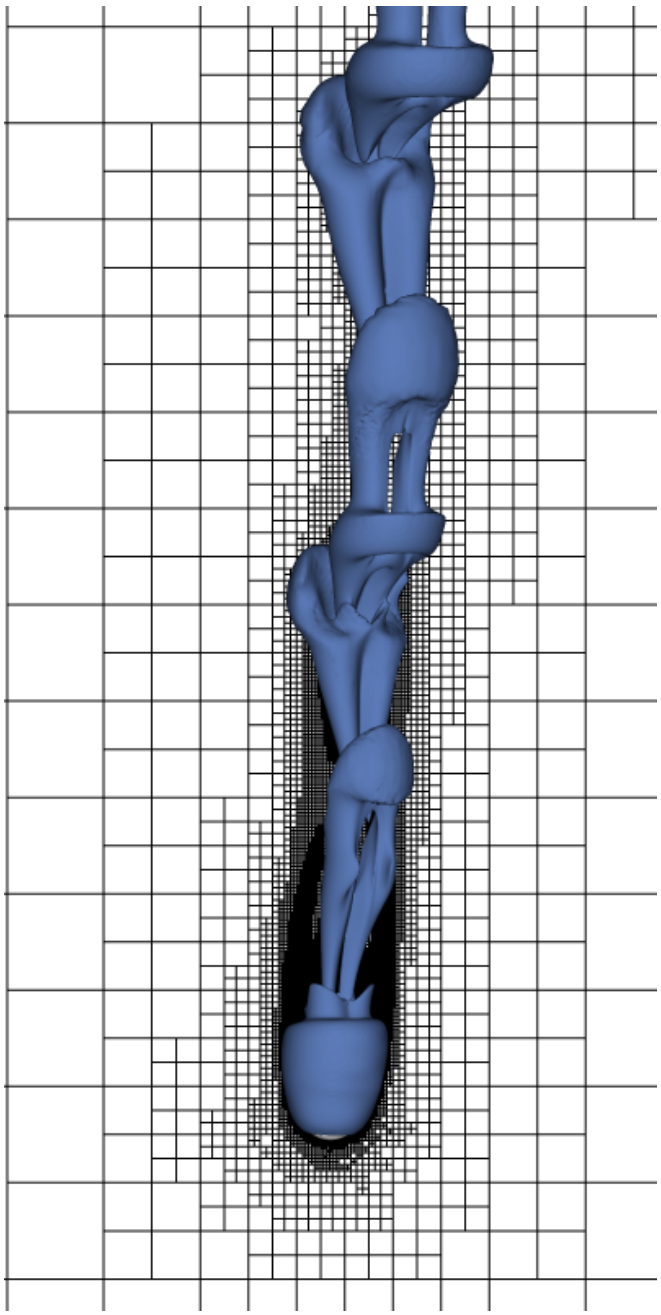
A conservative finite volume cut-cell method on an adaptive Cartesian tree grid for moving rigid bodies in incompressible flows

Arthur R. Ghigo, Stéphane Popinet, Anthony Wachs

Graphical Abstract

A conservative finite volume cut-cell method on an adaptive Cartesian tree grid for moving rigid bodies in incompressible flows

Arthur R. Ghigo, Stéphane Popinet, Anthony Wachs



Highlights

A conservative finite volume cut-cell method on an adaptive Cartesian tree grid for moving rigid bodies in incompressible flows

Arthur R. Ghigo, Stéphane Popinet, Anthony Wachs

- Development of a second-order accurate conservative finite volume cut-cell method for moving rigid bodies in an incompressible flow.
- Robust and second-order accurate computation of gradients normal to the embedded boundaries.
- Robust and efficient treatment of small cut-cells as well as submerged and emerged cells.
- Extension of the method to adaptive Cartesian quad/oc-tree grids.
- Implementation in the open-source software *Basilisk*.
- Validation test suite to verify the convergence and accuracy of the method.

A conservative finite volume cut-cell method on an adaptive Cartesian tree grid for moving rigid bodies in incompressible flows

Arthur R. Ghigo[a,b], Stéphane Popinet[d], Anthony Wachs[a,c]

[a] *Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, BC V6T 1Z4, Canada*

[b] *PIMS-CNRS, University of British Columbia, 4176-2207 Main Mall, Vancouver, BC, V6T 1Z4, Canada*

[c] *Department of Chemical and Biological Engineering, University of British Columbia, 2360 East Mall, Vancouver, BC V6T 1Z3, Canada*

[d] *Institut Jean Le Rond ∂'Alembert, Sorbonne Université, Centre National de la Recherche Scientifique, Paris F-75005, France*

Abstract

We present here a conservative finite volume cut-cell method for fixed and moving rigid bodies immersed in incompressible flows, implemented in the open-source software *Basilisk*. The method is constructed starting from a uniform Cartesian grid, in which we embed a discrete representation of a rigid body that intersects underlying cells to form irregular fluid control volumes, or cut-cells. In each cell, we then discretize the incompressible Navier-Stokes equations using a fractional-step projection method and insure that at each time step, the finite volume discretization scheme remains spatially second-order and conservative in cut-cells by carefully computing gradients normal to the embedded boundaries, even in degenerated cases. To avoid stability issues due to the well-documented problem of small cut-cells, we use a simple and efficient flux redistribution technique to extend the range of influence of small cut-cells to their neighboring cells. We also provide a time history to emerged cells through a field value reconstruction in the direction normal to the embedded boundaries. We then robustly extend our conservative finite volume cut-cell method for moving boundaries to adaptive Cartesian tree grids by constructing specific restriction and prolongation operators between two consecutive levels of a tree grid in the vicinity of a cut-cell. Finally, we a use a simple first-order in time explicit weak fluid-solid coupling strategy to describe the motion of freely moving particles. We successfully test the method on a series of validation test cases ranging from fixed, moving with a prescribed motion to freely moving 2D cylinders and spheres for a wide range of Reynolds ($0 \leq Re \leq 1000$) and Galileo numbers ($0 \leq Ga \leq 250$). In particular, by dynamically refining the mesh near the embedded boundaries, we are able to fully resolve boundary layers at a fraction of the computational cost of uniform grid computations. However, while the method is accurate, conservative, robust and efficient, we show that a low-amplitude pressure noise is generated when using mesh adaptation in the limit case of very high Reynolds numbers.

Keywords: Particle-laden flow; Embedded boundary; Cut-cell method; Conservative Finite Volume; Adaptive Cartesian tree grid; Validations

Email address: arthur.ghigo@math.ubc.ca (Arthur R. Ghigo)
URL: http://basilisk.fr/sandbox/ghigo/ (Arthur R. Ghigo)

1. Introduction

Incompressible particle-laden flows are ubiquitous in biological and geophysical phenomena as well as industrial processes. For example, the transport of sediments in rivers and the subsequent erosion of river beds are central to many problems as diverse as fish habitat preservation or operational strategy of hydroelectric dams. Fluidized beds are another example of a fluid-particle system widely used in the chemical industry to enhance heat and mass transfers. Unfortunately, analytical approaches to these problems are generally limited to asymptotic cases. There has therefore been in the past two decades a growing interest for numerical methods capable of solving fluid-structure interaction related problems such as particle-laden flows. While a large body of knowledge already exists on particle-laden flow dynamics, the complexity of the dominant fluid-particle momentum transfers is the primary reason why a complete understanding of this class of multiphase flow problems still escapes researchers and engineers. Indeed, particles exchange momentum through hydrodynamic interactions with the surrounding fluid and through collisions with neighbouring particles. When the suspension cannot be regarded as dilute anymore, particles also strongly disturb the flow field around neighbouring particles, leading to rich and complex multi-body and flow dynamics. The problem is further complexified by a distribution of particle sizes and/or a distribution of particle shapes (de facto including non-spherical particle shapes) and additional transfers such as heat or mass. A well-established way of modelling particle-laden flows involves recognizing their multiscale nature and lack of scale separation. Accordingly, specific numerical models are designed to describe fluid-particle dynamics at different scales: micro-scale particle-resolved simulation (PRS) at the particle scale, meso-scale Euler-Lagrange at the scale of a large group of particles, and macro-scale Euler-Euler at the scale of the flow. PRS constitutes the foundation of this multiscale analysis and the knowledge learnt from PRS is then transferred to the higher scale Euler-Lagrange and Euler-Euler models to improve their accuracy. Therefore, it is indispensable that PRS supplies high fidelity data that can be fully trusted.

In the quest for high fidelity, efficient and fast computational methods for incompressible flows laden with rigid particles, and more generally for incompressible flows with moving rigid boundaries, a wide variety of methods have been designed over the past three decades. They differ (i) in the way they spatially discretize the governing equations in the absence of rigid boundaries, ranging from conventional methods such as, e.g., finite difference, finite volume, finite element and spectral element, to non-conventional methods such as, e.g., lattice-Boltzmann (LB) and Smooth Particle Hydrodynamics, and (ii) in the way they treat the hydrodynamic coupling on the rigid boundaries, leading to well known techniques such as, e.g., immersed boundary (IB) [1, 2, 3, 4, 5, 6, 7], ghost cell (GC) [8, 9, 10, 11, 12, 13], distributed Lagrange multiplier/fictitious domain (DLM/FD) [14, 15, 16] and bounce-back scheme (BBS) [17, 18]. Some of the aforementioned techniques are specific to a given spatial discretization scheme (for example BBS in LB) while other techniques can be combined with different spatial discretization schemes (for example IB is combined to, e.g, finite volume, finite difference and LB), resulting in a very wide spectrum of PRS methods. We refer the interested reader to the following recent reviews on grid-based methods for PRS of particle-laden flows [6, 19, 20].

A common classification of grid-based PRS methods for particle-laden flows relies on distinguishing body-conforming methods from non body-conforming methods [20]. However, this classification does not provide much insight on the numerical characteristics of grid-based PRS

methods. It is therefore helpful to present them instead in light of their ability to satisfy the following properties:

1. the description of rigid boundaries is sharp, i.e., the method properly captures the gradient discontinuity at the boundary between the fluid and the rigid body, and additionally the correct rigid body motion is imposed on the grid points inside the rigid body;
2. the method is strictly mass, momentum and energy conservative in the sub-domain occupied by the fluid, a property essential for the prediction of discontinuous phenomena but also for stability in fluid–structure interaction methods [21, 22];
3. provided the spatial discretization scheme is at least second-order away from rigid boundaries, the treatment of moving rigid boundaries does not deteriorate the spatial accuracy in the vicinity of the moving rigid boundaries.
4. the method can be extended to adaptive grids without any major difficulty.

Body-conforming methods [23, 24, 25, 26, 27] may seem appealing at first as they naturally satisfy the four properties previously listed, provided that the spatial discretization scheme is intrinsically conservative and second-order accurate. Unfortunately, their performance is hindered by the computational overhead associated with the resolution of additional equations for the motion of the mesh, the complicated re-meshing process itself and the costly projection step of the solution onto the new grid. As problems of increasing complexity are now being tackled, e.g., problems involving a large collection of moving rigid bodies of arbitrary shape and/or a large computational domain, which require highly scalable computational methods on large supercomputers, large mesh deformations in body-conforming methods render these methods less competitive. Consequently, researchers have favored over the past 15-20 years non body-conforming methods based either on a fixed (often uniform) Cartesian grid or on an adaptive Cartesian grid. Note however the recent work of [28, 29] where the authors used an overset grid method, using a combination of both static grid and body-conforming grid methods, to perform PRS of particle-laden flows.

In non body-conforming methods, the complex numerical machinery required to keep track of the deforming mesh is traded for a (relatively) simple grid management (in the case of adaptive grids) or no grid management at all (in the case of fixed grids). However, this comes at the expense of introducing either a velocity reconstruction at the rigid boundaries or an additional body force distributed along the rigid boundaries in the momentum conservation equation to impose the hydrodynamic coupling between the fluid and the rigid bodies. The IB method based on discrete direct forcing (DF) using regularized Dirac delta functions [5, 30, 31, 32, 33, 34, 35] is probably the most popular PRS method in the literature but is also notoriously non-sharp. Indeed, the regularized Dirac delta kernel used to distribute the additional body force defined on the rigid boundaries is constructed with a compact support that symmetrically spans multiple cells on both sides of the rigid boundaries and consequently smooths any gradient discontinuity at the rigid boundaries. Note that in [7], the authors recently suggested an improvement of the DF/IB method in the form of one-sided IB kernels that is equivalent to a sharp treatment of the rigid boundaries. The standard DF/IB method is non-conservative in the domain occupied by the fluid as cells cut by the rigid boundaries are not treated in any specific manner in the fluid conservation equations (in particular the mass conservation does not account at all for the presence of rigid boundaries). Note also that the standard DF/IB method forces the no-slip boundary condition on the particle surface but often does not force any rigid body motion in the particle volume, leading to nonphysical fictitious fluid motion inside rigid

particles. Recently, in [36], the authors added volume forcing to the DF/IB method proposed in [5], in the spirit of the DLM/FD method to improve the spatial accuracy of the DF/IB method applied to the problem of inertial settling of a single ellipsoid. The GC method, the BBS based LB method and the DLM/FD method all deliver the right rigid body motion in the particle volume. The GC and BBS-LB methods exclude the degrees of freedom and simply assign the right velocity values in the particle volume while the DLM/FD method employs a volume forcing. The GC method as well as the implementation of the DLM/FD method in [37] provide a sharp description of the rigid boundaries and properly capture the gradient discontinuity at the rigid boundaries but are again not strictly conservative in the domain occupied by the fluid. Other PRS methods are also sharp and pseudo-body conforming [38, 39] in the sense that they use a fixed Cartesian grid and adapt the finite difference approximation stencil of the gradient in the vicinity of the rigid boundaries to incorporate the Dirichlet no-slip condition at the rigid boundaries. Again, these methods are not strictly conservative in the domain occupied by the fluid. Finally, in most of the aforementioned PRS methods, the treatment of the immersed rigid boundaries disrupts the second-order accuracy of the conservation equation spatial discretization scheme in the absence of immersed rigid boundaries and the overall spatial accuracy is generally between first-order and second-order, depending on the flow problem.

The most efficient PRS solvers using any of the computational methods discussed previously are implemented on a uniform Cartesian grid. LB methods are well known to be highly parallelizable and finite difference/finite volume based methods combined to efficient geometric or algebraic multigrid solvers are also very fast and scale very well on a large number of cores. Indeed, spectacular massively parallel computations of particle-laden flows have been reported in the recent literature, e.g., in [40] with a IB/LB method, $115,200$ spheres and 4.83 billion lattice nodes and in [41] with a DF/IB finite volume method, $1,053,648$ spheres and 3.6 billion cells. However, certain classes of particle-laden flow problems are not amenable to uniform fixed Cartesian grid PRS methods. Examples include but are not limited to flows laden with particles of complex shape, high Reynolds number flows with thin momentum boundary layers around particles, lubrication-dominated dense suspension flows with highly localized zones of spatially rapidly varying flow field, dilute to very dilute inertial particle-laden flows in large domains and particle-laden flows with a highly heterogeneous particle microstructure. These flow configurations all feature large sub-domains where the flow field spatially varies relatively slowly and locally requires less spatial resolution. Body-conforming methods that intrinsically use an unstructured grid and require constant re-meshing are naturally capable of performing local mesh refinement [25, 27], but at a large computing cost as discussed previously. Adaptive Cartesian grid methods offer a powerful alternative to dynamic unstructured re-meshing. There currently exists two types of adaptive Cartesian grid methods: (i) block-refined or patched adaptive Cartesian grid methods [42, 43, 44] in which entire cuboid sub-domains of uniform grid size are patched at different levels of refinement on the primary coarse grid and (ii) adaptive Cartesian tree grid methods [45, 46, 47, 48, 49, 50] in which individual cells live on different levels of refinement. Extending the implementation of any PRS methods from a uniform fixed Cartesian grid to a block refined adaptive Cartesian grid in which each rigid body is fully contained within a single block of cells at the same level of refinement is straightforward as within each block, the implementation of the numerical method to impose the no-slip boundary condition on the rigid boundaries is completely similar to the implementation on a uniform Cartesian grid. The extension to genuine adaptive Cartesian tree grids is much more challenging. When

implemented on a uniform Cartesian grid, PRS methods that describe the rigid boundaries by a set of Lagrangian discrete points require a quasi-homogeneous distribution of these points on the rigid boundaries [5, 16]. This quasi-homogeneous distribution is constructed such that Lagrangian points are approximately equidistant by $\alpha\Delta$ where $\alpha \in [1, 2]$ and Δ is the constant grid size. The linear relationship between the inter-point distance and the local grid size as well as the homogeneity of the distribution are key to the accuracy of the computed solution [5, 51, 16]. Various constructions can be adopted depending on the rigid body shape: (i) explicit for a sphere [5, 16] and simple polyhedrons [16], (ii) solution of a time-dependent problem of charged particles constrained to the body surface for spheroids [5, 36], and (iii) vertices of an unstructured triangulation of the body surface. On a Cartesian tree grid with a local grid size that presumably varies along the body surface, the inter-point distance (or equivalently the surface density of Lagrangian points) would need to match the local grid size, leading to a non-homogeneous Lagrangian point distribution over the rigid boundary and thus rendering the Lagrangian point construction very challenging if not overly complicated. Conversely, PRS methods that describe rigid boundaries on the Eulerian grid only extend more naturally to variable cell size Cartesian grids like Cartesian tree grids. In our previous work, we adopted an intermediate solution when extending our DLM/FD method on Cartesian tree grids [52] by enforcing a uniform grid in a narrow band spanning 3 cells on both sides of each rigid boundary. While computationally efficient, this strategy does not take full advantage of the local mesh refinement capabilities of a genuine Cartesian tree grid.

Our objective in this work is to propose a Cartesian grid embedded boundary method, also referred to in the literature as a *cut-cell* method, coupled to an adaptive mesh refinement technique (AMR) to compute efficiently and with second-order spatial accuracy the motion of particles of arbitrary shape in an incompressible flow. Indeed, cut-cell methods are a sub-class of non body-conforming methods [53] that satisfy the four properties listed previously. They are constructed starting from a uniform Cartesian grid, in which the embedded (rigid) boundaries intersect underlying cells to form irregular fluid control volumes. The regular structure of Cartesian grid allows for fast solution algorithms and a simple domain decomposition of the grid suitable for adaptation and parallel computing. To the best of our knowledge, cut-cell methods have mostly been applied to compressible flows with moving rigid bodies [49, 22, 54]. When the flow is incompressible or nearly incompressible, cut-cell methods have been mainly applied to fixed boundary problems [55, 56]. In the case of moving boundaries, we note the work of [57, 58] using a pressure-free projection method on both a uniform and an adaptive Cartesian grid and the work of [59, 58] to couple the cut-cell method to a two-fluid solver.

The Cartesian grid embedded boundary method for incompressible flows with fixed and moving rigid boundaries we present here satisfies the four key properties listed previously. Indeed, we discretize the incompressible Navier-Stokes equations using a fractional-step projection method and insure that at each time step, the finite volume discretization scheme remains spatially second-order and conservative in cut-cells by carefully imposing second-order accurate boundary conditions on both 2D and 3D embedded (rigid) boundaries. In particular, we follow and extend the work of [60, 61] to robustly compute gradients normal to the embedded boundaries, even in degenerated cases. To avoid stability issues due to the well-documented problem of small cut-cells, we use a simple and efficient flux redistribution technique, initially suggested in [62], to extend the range of influence of small cut-cells to their neighboring cells. We also provide a time history to emerged cells through a field value reconstruction in the

direction normal to the embedded boundaries. Finally, we take advantage of the conservative properties of the method, enhancing the stability and convergence properties of fluid-solid coupling strategies [21, 22], and use a simple first-order in time explicit weak fluid-solid coupling strategy to describe the motion of freely moving particles. We implement our method in the open-source software *Basilisk* [45, 50] and extend its use the adaptive Cartesian tree grids available in *Basilisk* by constructing specific restriction and prolongation operators between two consecutive levels of a tree grid in the vicinity of a cut-cell. We also benefit from the techniques already existing in *Basilisk* to dynamically manage tree grids and balance the computational load between processes in parallel computing [70]. Finally, the methodology proposed here and implemented in *Basilisk* is open-source and available to the entire research community.

This rest of the paper is organised as follows: in Section 2, we introduce the mathematical model describing the coupled motion of an incompressible flow and a rigid body; in Sections 3 and 4, we present the Cartesian grid embedded boundary method we implement in the software *Basilisk* on both uniform and tree grids; in Section 5 we describe the numerical method we use to solve the Navier-Stokes equations and the motion of a rigid body; in Section 6, we present several 2D and 3D validation test cases where we demonstrate the accuracy, robustness and efficiency of our Cartesian grid embedded boundary method for solving fixed and moving rigid body problems. Finally, we present our conclusions and perspectives in Section 7.

2. Mathematical model

We consider a Newtonian fluid of constant density ρ, constant dynamic viscosity μ, filling a domain Ω of boundary $\delta\Omega$. A moving rigid body $\Gamma \equiv \Gamma(t)$, of boundary $\delta\Gamma \equiv \delta\Gamma(t)$ and constant density ρ_Γ, is embedded in the domain. In the following, we briefly recall the governing equations for the coupled motion of the fluid and the rigid body.

2.1. Governing equations for the fluid

The motion of the fluid is governed by the incompressible Navier-Stokes equations, given in an Eulerian frame of reference $\mathbf{x} = [x, y, z]^\intercal$ by:

$$\rho\left(\partial_t \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \nabla\cdot\left(2\mu\mathbf{D}\right), \quad \forall \mathbf{x} \in \Omega/\Gamma \tag{1a}$$

$$\nabla\cdot\mathbf{u} = 0, \quad \forall \mathbf{x} \in \Omega/\Gamma, \tag{1b}$$

where $\mathbf{u} \equiv \mathbf{u}(\mathbf{x}, t)$ is the fluid velocity, $p \equiv p(\mathbf{x}, t)$ is the fluid pressure and $\mathbf{D}$ is the deformation tensor defined as $D_{ij} \equiv \left(\partial_i u_j + \partial_j u_i\right)/2$.

We supply system (1) with the following no-slip boundary condition for the velocity $\mathbf{u}$ on the rigid boundary $\delta\Gamma$:

$$\mathbf{u} = \mathbf{u}_\Gamma, \quad \forall \mathbf{x} \in \delta\Gamma, \tag{2}$$

where $\mathbf{u}_\Gamma \equiv \mathbf{u}_\Gamma(\mathbf{x}, t)$ is the velocity of the rigid body Γ. We specify boundary conditions for $\mathbf{u}$ on the boundary $\delta\Omega$ on a case-by-case basis. No boundary conditions for the pressure p are required at this point due to the orthogonality of the pressure gradient with the divergence-free velocity [63].

2.2. Rigid body dynamics

The motion of the rigid body Γ is characterized in an Eulerian frame of reference by the rigid body velocity $\mathbf{v}_\Gamma \equiv \mathbf{v}_\Gamma(\mathbf{x}, t)$:

$$\mathbf{v}_\Gamma = \mathbf{u}_\Gamma + \omega_\Gamma \times (\mathbf{x} - \mathbf{x}_\Gamma), \quad \forall \mathbf{x} \in \Gamma, \tag{3}$$

where $\mathbf{x}_\Gamma \equiv \mathbf{x}_\Gamma(t)$ is the position of the center of mass of the rigid body Γ and $\mathbf{u}_\Gamma \equiv \mathbf{u}_\Gamma(t)$ and $\omega_\Gamma \equiv \omega_\Gamma(t)$ are respectively the translation velocity of and the angular velocity about the center of mass $\mathbf{x}_\Gamma$. These variables satisfy the following equations of motion [64]:

$$\frac{\mathrm{d}\mathbf{x}_\Gamma}{\mathrm{d}t} = \mathbf{u}_\Gamma \tag{4a}$$

$$\frac{\mathrm{d}\mathbf{u}_\Gamma}{\mathrm{d}t} = \frac{\mathbf{F}_\Gamma}{\rho_\Gamma V_\Gamma} + \left(1 - \frac{\rho}{\rho_\Gamma}\right)\mathbf{g} \tag{4b}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{I}_\Gamma \omega_\Gamma) = \mathbf{T}_\Gamma, \tag{4c}$$

where V_Γ and $\mathbf{I}_\Gamma$ are respectively the volume and moment of inertia tensor of the rigid body Γ and $\mathbf{g}$ is the gravity acceleration vector. For the sake of simplicity, we do not include here the time evolution equation for the angular position θ_Γ of the rigid body as we consider only freely moving spherical particles in the following. The vectors $\mathbf{F}_\Gamma \equiv \mathbf{F}_\Gamma(t)$ and $\mathbf{T}_\Gamma \equiv \mathbf{T}_\Gamma(t)$ respectively represent the hydrodynamic force and torque (about the center of mass $\mathbf{x}_\Gamma$) exerted by the fluid on the rigid body Γ:

$$\mathbf{F}_\Gamma = -\int_{\delta\Gamma} (-p\mathbb{I} + 2\mu\mathbf{D}) \cdot \mathbf{n}_\Gamma \,\mathrm{d}S \tag{5a}$$

$$\mathbf{T}_\Gamma = -\int_{\delta\Gamma} (\mathbf{x} - \mathbf{x}_\Gamma) \times (-p\mathbb{I} + 2\mu\mathbf{D}) \cdot \mathbf{n}_\Gamma \,\mathrm{d}S, \tag{5b}$$

where $\mathbf{n}_\Gamma \equiv \mathbf{n}_\Gamma(\mathbf{x}, t)$ is the inward (pointing from the fluid towards the rigid body) unit normal vector to the rigid boundary $\delta\Gamma$.

The remaining unknown is the location and shape of the rigid boundary $\delta\Gamma$, which we describe using a user-defined distance function $\Phi(\mathbf{x} - \mathbf{x}_\Gamma)$ to guarantee a topologically and analytically coherent representation of the rigid boundary $\delta\Gamma$. This allows us to define the rigid body Γ, the rigid boundary $\delta\Gamma$ and the fluid domain Ω/Γ as follows:

- $\Gamma = \{\mathbf{x} \in \Omega \,|\, \Phi(\mathbf{x} - \mathbf{x}_\Gamma) \leq 0\}$;
- $\delta\Gamma = \{\mathbf{x} \in \Omega \,|\, \Phi(\mathbf{x} - \mathbf{x}_\Gamma) = 0\}$;
- $\Omega/\Gamma = \{\mathbf{x} \in \Omega \,|\, \Phi(\mathbf{x} - \mathbf{x}_\Gamma) > 0\}$.

3. Cartesian grid embedded boundary method on a uniform grid

In this section, we present in detail the Cartesian grid embedded boundary method, also known as *cut-cell* method, that we implement on a uniform Cartesian grid in *Basilisk*. In particular, we describe how we modify the *finite volume* discretization of the divergence and gradient operators, respectively $\nabla\cdot$ and ∇, to account for the presence of the rigid body Γ in a conservative, robust and accurate manner. This method is therefore not only designed to solve the Navier-Stokes equations (1) and can be more generally applied to any system of partial differential equations solved using a *finite volume* method.

3.1. *Uniform Cartesian grid in* Basilisk

We discretize the computational domain Ω using a uniform Cartesian grid, denoted Ω_Δ in the following. The grid is composed of square (cubic in 3D) *cells* and the length of a cell *edge* is denoted Δ. As illustrated in Figure 1, each cell has direct *neighbors* in each direction d (four in 2D, six in 3D) and each of these neighbors is accessed through a *face* of the cell, noted $\mathcal{F}_d$.

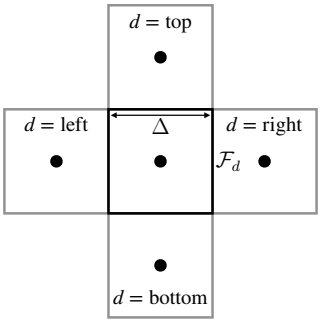


Figure 1: Representation of a 2D cell and its neighbors in *Basilisk*.

The principal variables, here the velocity $\mathbf{u}$ and the pressure p, are collocated at the center of each cell (see Figure 2a). Transported or diffused variables such as the velocity $\mathbf{u}$ are therefore interpreted as the volume-averaged values over the cell volume, whereas the pressure should be seen as a point-value estimate at the center the cell. Variables can also be staggered at the center of a face, the middle of an *edge* or on the *vertices* of a cell (see Figures 2b, 2c and 2d). When writing discrete operations on the grid Ω_Δ, we distinguish these last three discretization using the superscripts $^{\mathbf{f}}$, $^{\mathbf{e}}$ and $^{\mathbf{v}}$ and use the following *Basilisk* indexing, introduced in [50] and illustrated in Figure 2 for a 2D grid:

- $[0,0] = [\,]$ for the current cell, i.e. $s\,[0,0]$;
- $[1,0] = [1]$ for its *right* neighbor, i.e. $s\,[1,0]$;
- $[\text{-}1,0] = [\text{-}1]$ for its *left* neighbor, i.e. $s\,[\text{-}1,0]$;
- $[0,1]$ for its *top* neighbor, i.e. $s\,[0,1]$;
- $[0,\text{-}1]$ for its *bottom* neighbor, i.e. $s\,[0,\text{-}1]$.

As examples, the discretization of the gradient of the cell-centered scalar s:

- in the x-direction and defined at the center of the cell, denoted $\nabla_x s\,[\,]$, writes:

$$\nabla_x s\,[\,] = \frac{s[1] - s[\text{-}1]}{2\Delta}, \tag{6}$$

- in the y-direction and defined on the *top* face of the cell, denoted $\nabla_y^f s\,[0,1]$, writes:

$$\nabla_y^f s\,[0,1] = \frac{s[0,1] - s[\,]}{\Delta}. \tag{7}$$

We choose this notation system in order to match the notation system used in *Basilisk* and presented in detail here: `http://basilisk.fr/Basilisk%20C#stencils`.

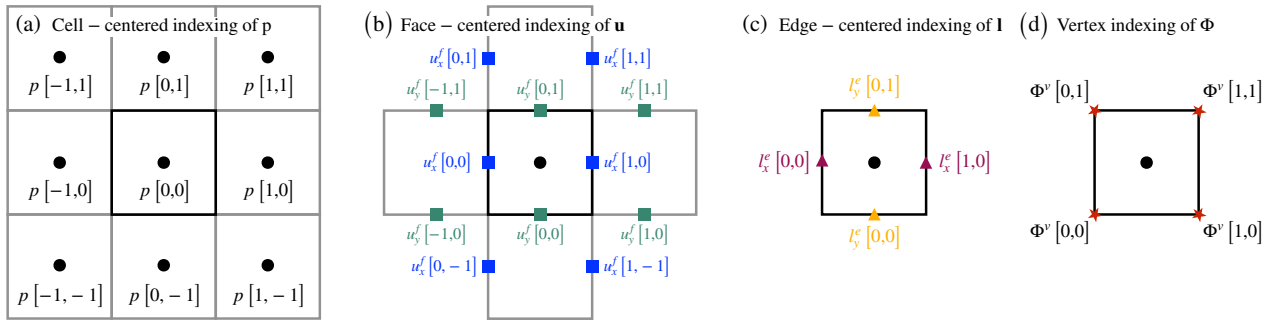


Figure 2: Stencils in *Basilisk* and their respective indexing on a 2D grid: (a) cell-centered; (b) face-centered $^{\mathbf{f}}$; (c) edge-centered $^{\mathbf{e}}$ (identical to face-centered in 2D); (d) vertex $^{\mathbf{v}}$.

3.2. Integral description of rigid boundary $\delta\Gamma$

The Cartesian grid embedded boundary method is constructed starting from the uniform Cartesian grid Ω_Δ described previously, in which we embed a discrete representation of the rigid body Γ. Therefore, as shown in Figure 3a, the discrete rigid boundary, denoted $\delta\Gamma_\Delta$, intersects underlying cells to form irregular fluid control volumes in each cell cut by the boundary. In such cells, the geometry of the irregular fluid control volume, denoted $\mathcal{V}$, is characterized by the following Volume-of-Fluid (VOF) quantities, also referred to as the embedded fractions and represented in Figure 3b:

- the volume fraction $0 \leq c \leq 1$ of the cell, such that the effective volume occupied by the fluid in the cell is:
$$V = c\Delta^D, \tag{8}$$
where D is the number of space dimensions. A cell can therefore be: (i) a full cell with $c = 1$; (ii) a solid cell with $c = 0$ or (iii) a *cut-cell* with $0 < c < 1$.

- the area fraction $0 \leq f_d^f \leq 1$ of the face $\mathcal{F}_d$, such that the effective area occupied by the fluid on the face is:
$$A_d^f = f_d^f \Delta^{D-1}. \tag{9}$$
Note that we also refer to A_d^f as the *partial* face occupied by the fluid on the (*full*) face $\mathcal{F}_d$, i.e. we use the notation A_d^f for both the partial face and its surface area and denote $\mathbf{m}_d^f$ its centroid (see Figure 3c).

This integral description of the geometry of the discrete rigid boundary $\delta\Gamma_\Delta$ allows us to represent rigid bodies of arbitrary shape. We cannot however represent thin bodies of width smaller than the cell size Δ or multiple bodies separated by less than the cell size. These restrictions can be lifted using for example the "multi-cut-cell" method presented in [65, 22]. However, the implementation of such a method for the incompressible Navier-Stokes equations (1) is beyond the scope of this study and will be addressed in future works.

In the following section, we detail the computation of the embedded fractions c and f_d^f starting from the distance function representation of the rigid boundary $\delta\Gamma$ introduced in Section 2.2.

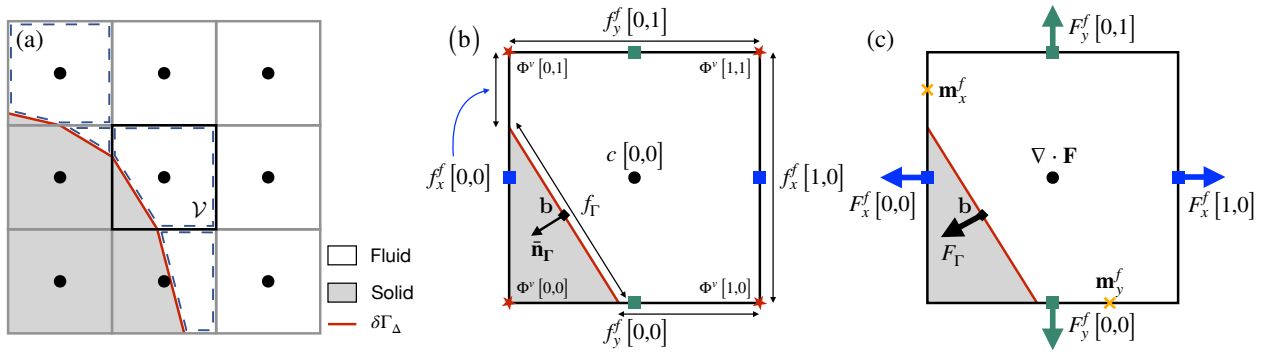


Figure 3: Graphical representation of the Cartesian grid embedded boundary method on a 2D grid: (a) Cartesian grid Ω_Δ cut by the discrete rigid boundary $\delta\Gamma_\Delta$ to form irregular control volumes $\mathcal{V}$; (b) Geometric and VOF quantities in a *cut-cell*; (c) Conservative discretization of $\nabla\cdot\mathbf{F}$ in a *cut-cell*.

3.3. Computation of the embedded fractions in a cut-cell

We first select a representation of the discrete rigid boundary $\delta\Gamma_\Delta$ in each cut-cell. As illustrated in Figure 3a, we simply choose a piecewise continuous contour, which satisfies in each cut-cell the following equation for a line (a plane in 3D):

$$\bar{\mathbf{n}}_\Gamma\cdot\mathbf{x}=\alpha, \tag{10}$$

where $\bar{\mathbf{n}}_\Gamma$ is the inward unit normal vector to the discrete rigid boundary and α is the intercept.

Using the distance function $\Phi\left(\mathbf{x}-\mathbf{x}_\Gamma\right)$ sampled on the vertices of each cell of grid Ω_Δ, we then compute the geometric quantities $\bar{\mathbf{n}}_\Gamma$ and α in each cut-cell, along with their corresponding volume fraction c and area fraction f_d^f of each face $\mathcal{F}_d$ of the cut-cell. We detail this procedure in the following, and consider first a 2D cut-cell such as the one presented in Figure 3b:

1. For each vertex of the cut-cell, here for instance the *bottom left* vertex:
   we evaluate $\Phi^v\left[\,\right]$, where $\Phi\left(\mathbf{x}-\mathbf{x}_\Gamma\right)$ is the distance function we use to represent the rigid boundary $\delta\Gamma$ in Section 2.2.

2. For each face $\mathcal{F}_d$ of the cut-cell, here for instance the *bottom* face:
   if $\Phi^v\left[\,\right]\cdot\Phi^v\left[1\right]<0$, the face is cut by the rigid boundary $\delta\Gamma$ and we compute the corresponding area fraction $f_d^f=f_y^f\left[\,\right]$ using a linear interpolation of the vertex values of Φ:

$$f_y^f\left[\,\right]=\frac{1-\operatorname{sign}\left(\Phi^v\left[\,\right]\right)}{2}+\operatorname{sign}\left(\Phi^v\left[\,\right]\right)\frac{\Phi^v\left[\,\right]}{\Phi^v\left[\,\right]-\Phi^v\left[1\right]}. \tag{11}$$

3. Following [66, 60], we express the circulation along the closed boundary of the fluid control volume $\mathcal{V}$ in the cut-cell:

$$\int_{\delta\mathcal{V}}\mathbf{n}\,\mathrm{d}A=0\Leftrightarrow\sum_d f_d^f\,\mathbf{n}_d^f+\hat{\mathbf{n}}_\Gamma=0, \tag{12}$$

   where $\mathbf{n}_d^f$ is the outward unit normal vector to the face $\mathcal{F}_d$ and $\hat{\mathbf{n}}_\Gamma$ is the inward (non-unit) normal vector to the discrete rigid boundary. Using equation (12), we derive the following

expression for $\hat{\mathbf{n}}_\Gamma$:

$$\hat{\mathbf{n}}_\Gamma = \begin{bmatrix} f_x^f\,[\,] - f_x^f\,[1] \\ f_y^f\,[\,] - f_y^f\,[0,1] \end{bmatrix}. \tag{13}$$

We then normalise $\hat{\mathbf{n}}_\Gamma$ to obtain $\bar{\mathbf{n}}_\Gamma$. Note that equation (13) is the exact expression for the volume-averaged normal vector to the discrete rigid boundary $\delta\Gamma_\Delta$ in the cut-cell.

4. For each face $\mathcal{F}_d$ of the cut-cell, here for instance the *bottom* face:
if $0 < f_d^f = f_y^f\,[\,] < 1$, we compute the value of the face intercept α_d as:

$$\alpha_d = \bar{\mathbf{n}}_\Gamma \cdot \mathbf{x}_d, \tag{14}$$

where $\mathbf{x}_d$ is the position of the intersection of the discrete boundary with the face $\mathcal{F}_d$. For the *bottom* face, we have:

$$\mathbf{x}_d = \begin{bmatrix} \text{sign}\left(\Phi^v\,[\,]\right)\left(f_y^f\,[\,] - 0.5\right) \\ -0.5 \end{bmatrix}. \tag{15}$$

We then compute α as the arithmetic average of all face intercepts α_d. Note here that the position vector $\mathbf{x}_d$ is defined in a coordinate system with origin the center of the cell and in which the cell size is unity.

5. Finally, we compute the volume fraction c using the well-established functions detailed in [67, 68], which account for the different ways a square (cubic in 3D) cell can be cut by a line (plane in 3D):

$$c = \mathcal{V}\left(\bar{\mathbf{n}}_\Gamma,\, \alpha\right). \tag{16}$$

The generalisation of the previous algorithm to 3D is relatively straightforward. Indeed, noticing that a 2D cell is a face of a 3D cell, we compute the area fraction f_d^f of each face $\mathcal{F}_d$ of a 3D cut-cell as we would compute the volume fraction of a 2D cut-cell. We then apply steps 3 to 5 of the previous algorithm to compute the volume fraction c of the 3D cut-cell, with small modifications to account for the third dimension. Complete details of the implementation of the previous algorithm and its extension to 3D can be found here: `http://basilisk.fr/src/fractions.h#computing-volume-fractions-from-a-levelset-function`.

Given the geometric quantities $\bar{\mathbf{n}}_\Gamma$ and α in a cut-cell, we also compute the area fraction f_Γ and the coordinates of the centroid $\mathbf{b}$ of the discrete rigid boundary $\delta\Gamma_\Delta$ in the cut-cell (see Figure 3b). Complete details can be found here: `http://basilisk.fr/src/embed.h#utility-functions-for-the-geometry-of-embedded-boundaries`.

3.4. Discrete operators in a cut-cell

Given the embedded fractions c, f_d^f and f_Γ in a cut-cell, we can now write the following conservative *finite volume* approximation of the divergence of a flux $\mathbf{F}$ in the cut-cell [60, 61], also illustrated in Figure 3c:

$$\begin{aligned} \nabla \cdot \mathbf{F} &\approx \frac{1}{c\Delta^D} \int_{\mathcal{V}} \nabla \cdot \mathbf{F}\, \mathrm{d}V = \frac{1}{c\Delta^D} \int_{\delta\mathcal{V}} \mathbf{F} \cdot \mathbf{n}\, \mathrm{d}A \\ &\approx \frac{1}{c\Delta} \left(\sum_d f_d^f F_d^f + f_\Gamma F_\Gamma \right), \end{aligned} \tag{17}$$

where $\mathbf{n}$ is the outward unit normal vector to the boundary $\delta\mathcal{V}$ of the fluid control-volume $\mathcal{V}$ in the cut-cell. Both $F_d^f = \mathbf{F}^f \cdot \mathbf{n}_d$ and $F_\Gamma = \mathbf{F}_\Gamma \cdot \bar{\mathbf{n}}_\Gamma$ are the discrete counterparts of $\mathbf{F} \cdot \mathbf{n}$ and respectively represent the *finite volume* flux through the face $\mathcal{F}_d$ and through the discrete rigid boundary $\delta\Gamma_\Delta$ in the cut-cell. As in [60, 61], the flux F_d^f is defined at the centroid $\mathbf{m}_d^f$ of the partial face A_d^f (of the full face $\mathcal{F}_d$). Similarly, the flux F_Γ is defined at the centroid $\mathbf{b}$ of the discrete rigid boundary $\delta\Gamma_\Delta$.

When solving the incompressible Navier-Stokes equations (1), the discrete fluxes F_d^f and F_Γ typically correspond to the nonlinear advection flux $\mathbf{u} \otimes \mathbf{u}$ or the viscous flux $2\mu\mathbf{D}$. In order to evaluate these fluxes in a cut-cell of the grid Ω_Δ, we therefore introduce the following three quantities:

1. $\nabla_\Gamma s = \nabla s|_{\delta\Gamma_\Delta} \cdot \bar{\mathbf{n}}_\Gamma$, the *embedded face gradient* of a cell-centered scalar s, defined at the centroid $\mathbf{b}$ of the discrete rigid boundary $\delta\Gamma_\Delta$ in the cut-cell;

2. $\nabla_d^f s = \nabla s|_{\mathcal{F}_d} \cdot \mathbf{n}_d$, the *face gradient* of a cell-centered scalar s, defined at the centroid $\mathbf{m}_d^f$ of the partial face A_d^f in the cut-cell;

3. s_d^f, the *face value* of a cell-centered scalar s, defined at the centroid $\mathbf{m}_d^f$ of the partial face A_d^f in the cut-cell.

In the following, we detail the computation of these three quantities using only values of the cell-centered scalar s in what we refer to as *available* cells. Available cells simply correspond to cells that are within the fluid domain, i.e. full cells ($c = 1$) and cut-cells ($0 < c < 1$). If the discrete rigid boundary $\delta\Gamma_\Delta$ is moving, available cells are cells that are within the fluid domain, excluding *emerged* cells that have not yet been properly initialized. We define emerged cells in Section 5.5.

3.4.1. Computation of the embedded face gradient of a cell-centered scalar

We detail here the computation in a cut-cell of the *embedded face gradient* $\nabla_\Gamma s$, defined at the centroid $\mathbf{b}$ of the discrete rigid boundary $\delta\Gamma_\Delta$ in a cut-cell. Two cases arise, depending on the nature of the boundary condition for s imposed at the centroid $\mathbf{b}$.

Neumann boundary condition. If a Neumann boundary condition g_Γ is imposed, then the embedded face gradient $\nabla_\Gamma s$ simply writes:

$$\nabla_\Gamma s = g_\Gamma. \tag{18}$$

Dirichlet boundary condition. If a Dirichlet boundary condition s_Γ is imposed, we compute the embedded face gradient $\nabla_\Gamma s$ following the methodology presented in [60, 61]. We use the following second-order discretization of the gradient in the direction of the normal vector $-\bar{\mathbf{n}}_\Gamma$, also illustrated in Figure 4a:

$$\nabla_\Gamma s = \frac{1}{d_2 - d_1}\left((s_\Gamma - s_1)\,\frac{d_2}{d_1} - (s_\Gamma - s_2)\,\frac{d_1}{d_2}\right). \tag{19}$$

To maintain second-order accuracy, we compute the quantities s_1 and s_2 using a third-order quadratic (biquadratic in 3D) interpolation along the direction orthogonal to the principal

direction of the normal vector $-\bar{\mathbf{n}}_\Gamma$ and at a positive distance d_1 and d_2 from the centroid $\mathbf{b}$. Without loss of generality, we consider the 2D configuration presented in Figure 4, where the principal direction of the normal vector $-\bar{\mathbf{n}}_\Gamma$ is the positive x-direction and $d_1 = \frac{1-b_x}{-\bar{n}_{\Gamma,x}}$ and $d_2 = \frac{2-b_x}{-\bar{n}_{\Gamma,x}}$. We then compute s_1 and s_2 using values of s taken from the 3×2 ($3 \times 3 \times 2$ in 3D) stencil represented in Figure 4b, itself part of the 5×5 ($5 \times 5 \times 5$ in 3D) stencil of the cut cell:

$$\begin{aligned} s_1 &= \left(s\left[1\right]\left(y_1 - 1\right) + s\left[1,2\right]\left(y_1 + 1.\right)\right)\frac{y_1}{2} - s\left[1,1\right]\left(y_1 - 1\right)\left(y_1 + 1\right) \\ s_2 &= \left(s\left[2\right]\left(y_2 - 1\right) + s\left[2,2\right]\left(y_2 + 1.\right)\right)\frac{y_2}{2} - s\left[2,1\right]\left(y_2 - 1\right)\left(y_2 + 1\right), \end{aligned} \tag{20}$$

where $y_1 = b_y + d_1\left[-\bar{n}_{\Gamma,y}\right] - 1$ and $y_2 = b_y + d_2\left[-\bar{n}_{\Gamma,y}\right] - 1$. Note here that the centroid $\mathbf{b}$ and the distances d_1, d_2, y_1 and y_2 are defined in a coordinate system with origin the center of the cell and in which the cell size is unity and that the values of s are interpreted as point-value estimates at the center of full cells.

To maintain a robust computation of the embedded face gradient $\nabla_\Gamma s$ in complex geometrical configurations that can naturally occur in particle-laden flows, we use equation (19) only if the cells required to compute the first and second interpolants s_1 and s_2 in equation (20) are *topologically connected*. This means that a line unbroken by a solid cell or face connects the center of the current cut-cell $[0,0]$ to the center of all cells in the stencils of both s_1 and s_2, i.e. cells $\{[1]\,,\,[1,1]\,,\,[1,2]\}$ and cells $\{[2]\,,\,[2,1]\,,\,[2,2]\}$ for the 2D configuration presented in Figure 4. If this is not the case, we face a degenerated case. This situation was not considered in [60, 61] and we therefore propose the following extension of the computation of the embedded face gradient $\nabla_\Gamma s$ in degenerated cases:

- if the stencil for s_2 is not topologically connected to the current cut-cell $[0,0]$, we simply use the following first-order discretization of $\nabla_\Gamma s$:

$$\nabla_\Gamma s = \frac{s_\Gamma - s_1}{d_1}. \tag{21}$$

  In Figure 5, we present the three possible configurations for which this degenerated case would occur for the 2D configuration presented in Figure 4.

- if the stencil for s_1 is not topologically connected to the current cut-cell $[0,0]$, we face a pathological situation. In this case, we resort to using the cell-centered value $s\left[\,\right]$ to compute $\nabla_\Gamma s$. In the configuration presented in Figure 4, the embedded face gradient would write:

$$\nabla_\Gamma s = \frac{s_\Gamma - s[\,]}{|b_x/\bar{n}_{\Gamma,x}|}. \tag{22}$$

  In Figure 6, we present the three possible configurations for which this degenerated case would occur for the 2D configuration presented in Figure 4.

Complete details of the computation of the embedded face gradient $\nabla_\Gamma s$ and its extension to 3D can be found here: `http://basilisk.fr/src/embed.h#dirichlet-boundary-condition`.

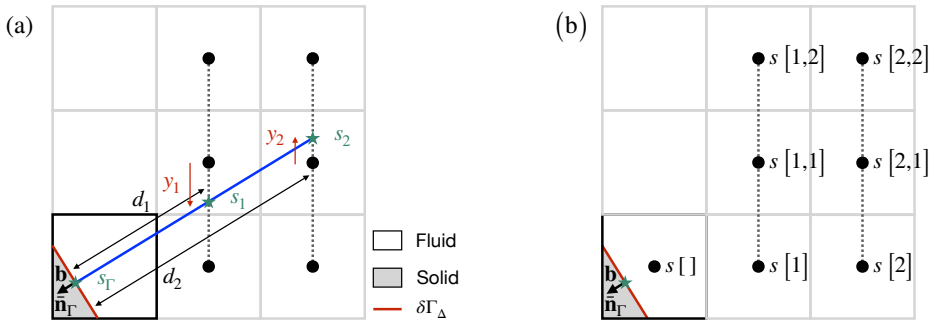


Figure 4: Graphical representation of the methodology proposed in [60, 61] and implemented in *Basilisk* to compute the second-order embedded face gradient $\nabla_\Gamma s$ on a 2D grid when the principal direction of the normal vector $-\bar{\mathbf{n}}_\Gamma$ is the positive x-direction.

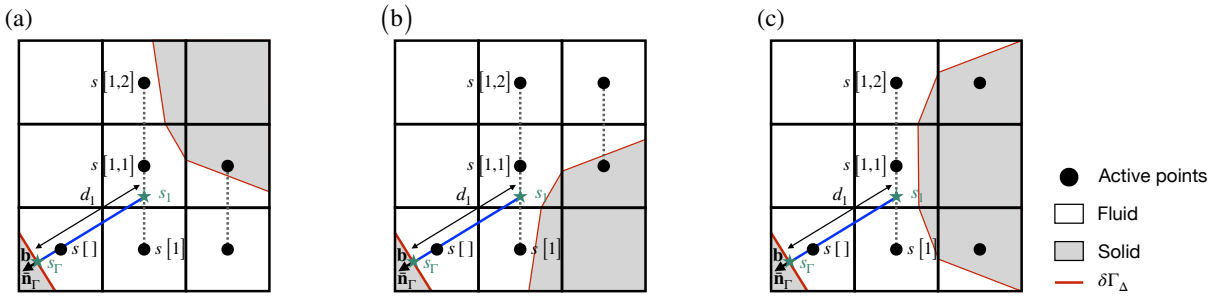


Figure 5: Graphical representation of the extension of the methodology proposed in [60, 61] and implemented in *Basilisk* to compute the embedded face gradient $\nabla_\Gamma s$ in a first type of degenerated case on a 2D grid when the principal direction of the normal vector $-\bar{\mathbf{n}}_\Gamma$ is the positive x-direction and the stencil for s_2 (cells $\{[2]\,,\,[2,1]\,,\,[2,2]\}$) is not topologically connected to the current cut-cell $[0,0]$.

3.4.2. Computation of the face gradient of a cell-centered scalar

We now describe the computation of the *face gradient* $\nabla_d^f s$, defined at the centroid $\mathbf{m}_d^f$ of the partial face A_d^f (of the full face $\mathcal{F}_d$) in a cut-cell. To match the second-order accuracy of the embedded face gradient $\nabla_\Gamma s$ described previously, we first define the following second-order *simple face gradient* on the face $\mathcal{F}_d$, denoted $\bar{\nabla}_d^f s$ and written here for instance on the *left* face of the cut-cell:

$$\bar{\nabla}_x^f s\,[\,] = \frac{s\,[\,] - s\,[\text{-}1]}{\Delta}. \tag{23}$$

We then proceed as in [60] and define $\nabla_d^f s$ using a second-order linear (bilinear in 3D) interpolation, at the centroid $\mathbf{m}_d^f$ of the partial face A_d^f, of simple face gradients $\bar{\nabla}_d^f s$ computed on neighboring faces located in the direction (two directions in 3D) orthogonal to the direction d. Without loss of generality, we consider here for instance the *left* face of the 2D cut-cell

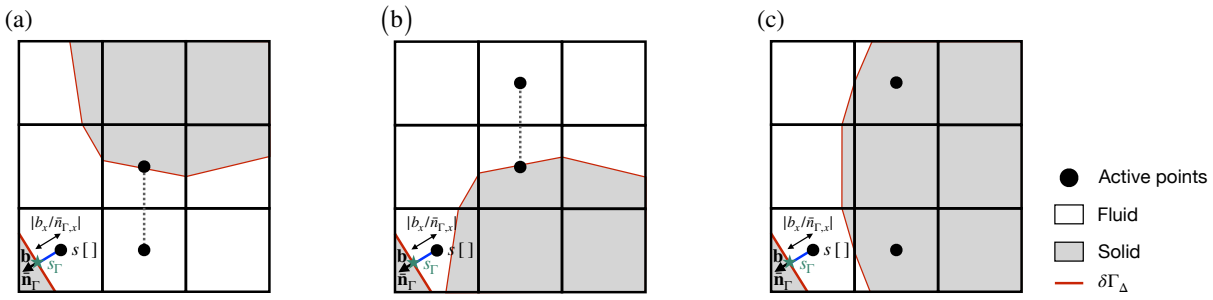


Figure 6: Graphical representation of the extension of the methodology proposed in [60, 61] and implemented in *Basilisk* to compute the embedded face gradient $\nabla_\Gamma s$ in a second type of degenerated case on a 2D grid when the principal direction of the normal vector $-\bar{\mathbf{n}}_\Gamma$ is the positive x-direction and the stencil for s_1 (cells $\{[1], [1,1], [1,2]\}$) is not topologically connected to the current cut-cell $[0,0]$.

presented in Figure 3b and compute $\nabla_d^f s = \nabla_x^f s\,[\,]$ as follows:

$$\nabla_x^f s\,[\,] = \begin{cases} \text{if } f_x^f\,[0,1] \geq f_x^f\,[0,\text{-}1]: \\ \qquad \left(1 + m_{x,y}^f\right) \bar{\nabla}_x^f s\,[\,] + m_{x,y}^f \bar{\nabla}_x^f s\,[0,1] \\ \text{else}: \\ \qquad \left(1 + m_{x,y}^f\right) \bar{\nabla}_x^f s\,[\,] + m_{x,y}^f \bar{\nabla}_x^f s\,[0,\text{-}1], \end{cases} \tag{24}$$

where $\mathbf{m}_x^f = \left[0, m_{x,y}^f\right]^\intercal$, with $m_{x,y}^f = \left(1 - f_x^f\,[\,]\right)/2$. Note here that the centroid $\mathbf{m}_d^f$ is defined in a coordinate system with origin the full face center and in which the face size is unity.

As for the embedded face gradient $\nabla_\Gamma s$, we only use equation (24) if the cells required to define the simple face gradients $\bar{\nabla}_d^f s$ used in equation (24) are *topologically connected* to the face $\mathcal{F}_d$. Otherwise, we simply set $\nabla_d^f s = \bar{\nabla}_d^f s$. Indeed, using faces of the grid Ω_Δ that are not topologically connected could prevent the convergence of the multigrid Poisson-Helmholtz solver [61].

Complete details of the computation of the face gradient $\nabla_d^f s$ and its extension to 3D can be found here: `http://basilisk.fr/src/embed.h#operator-overloading`.

3.4.3. Computation of the face value of a cell-centered scalar

We finally detail the computation of the *face value* s_d^f, defined at the centroid $\mathbf{m}_d^f$ of the partial face A_d^f in a cut-cell. We compute s_d^f using equation (24), in which we substitute the simple face gradients $\bar{\nabla}_d^f s$ by a volume-weighted average of s in the direction d, denoted $\bar{s}_d^f$. Without loss of generality, we consider here for instance the *left* face of the 2D cut-cell presented in Figure 3b and compute the volume-weighted average $\bar{s}_d^f = \bar{s}_x^f\,[\,]$ as:

$$\bar{s}_x^f\,[\,] = \frac{\left(\frac{3}{2} + c\,[\,]\right) s[\,] + \left(\frac{3}{2} + c\,[\text{-}1]\right) s[\text{-}1]}{c\,[\,] + c\,[\text{-}1] + 3}. \tag{25}$$

We use the weighted average (25) instead of a simple average, i.e. $\left(s_x\,[\,] + s_x\,[\text{-}1]\right)/2$, to prevent the occurrence of instabilities when solving the Stokes equations (1), a phenomenon that has also been noticed in [69] for low Reynolds number flows. These instabilities appear

when using an approximate projection scheme, due to some feedback between pressure and velocity modes, amplified by the face gradient (24) used to compute the viscous fluxes, which can behave as a third-order term in cut-cells when combined with Dirichlet boundary conditions [60, 61] (see Section 6.1).

More details are provided in Section 6.4 and complete details of the computation of the face value s_d^f can be found here: `http://basilisk.fr/src/embed.h#operator-overloading`.

In the following section, we present the extension to quad/oc-tree grids of the Cartesian grid embedded boundary method previously described on the uniform Cartesian Ω_Δ.

4. Extension of the Cartesian grid embedded boundary method to quad/oc-tree grids

Besides the uniform Cartesian grid Ω_Δ, *Basilisk* also enables the use of *tree* grids, referred to as *quadtree* in 2D and *octree* in 3D [45, 50]. A tree grid allows for a variable resolution of the computational domain Ω and provides a convenient and efficient framework for adaptive mesh refinement (AMR). We refer the reader to [50, 70, 71, 52] for a more detailed presentation of tree grids in *Basilisk* and describe in the following only the necessary steps to extend their use in the presence of embedded boundaries.

4.1. Properties of tree grids in Basilisk *in the absence of embedded boundaries*

In a tree grid in *Basilisk*, cells are organized hierarchically starting from the *root* cell located at the base of the tree, also referred to as *level* $l = 0$. A cell of size Δ at level l can be *parent* to up to 4 *children* cells (8 in 3D) of size $\Delta/2$ located at level $l + 1$. Finally, a cell with no *children* is a *leaf cell*.

A remarkable feature of *Basilisk* is that the definition of a cell on both a uniform grid Ω_Δ and a tree grid is identical. In other words, discrete operations on both grid types are implemented in an identical manner. In particular, the Cartesian grid embedded boundary method presented previously on a uniform Cartesian grid Ω_Δ can therefore be identically applied to a tree grid. This is achieved by guaranteeing that each cell, whether of a uniform grid Ω_Δ or of a tree grid, has access to a 5×5 ($5 \times 5 \times 5$ in 3D) regular stencil (i.e. $\{[2], [1], [-1], \dots\}$) [50]. *Basilisk* therefore extends the regular stencil of a cell if the cell is located near a *domain boundary* or a *resolution boundary*, using respectively *ghost* and *halo* cells.

For all grid types, the regular stencil of a cell near the *domain boundary* $\delta\Omega$ is extended beyond the limits of the computational domain Ω using two layers of *ghost cells*. Cell-centered values in these boundary ghost cells are updated using some discrete approximation of the boundary conditions imposed on the boundary $\delta\Omega$. The presence of embedded boundaries does not modify how boundary ghost cells are updated and we refer the reader to [50] for more details.

For tree grids only, the regular stencil of a leaf cell inside the computational domain and near a *resolution boundary* is extended using two layers of *halo* cells. We call *resolution boundary* any region of a tree grid where two neighboring leaf cells are located at different levels (i.e. have a different size). Note that in *Basilisk*, the level of neighboring leaf cells cannot vary by more than one. As described in [50] and illustrated in Figures 7a and 8a, cell-centered values in resolution boundary halo cells (blue circle and red crosses in Figures 7a and 8a) are updated using the following second-order functions, described here for a quadtree:

- a *prolongation* function which uses a bilinear interpolation of values defined in cells located at level l (large black dots and large blue circle in Figure 7a) to update the value in a halo cell located at level $l+1$ (red crosses in Figure 7a). For a quadtree without embedded boundaries, the prolongation of a cell-centered scalar s from level l to level $l+1$ writes:

$$\begin{aligned} s\left[\,\right] = &\left(9\,\mathrm{coarse}\left(s\left[\,\right]\right) + \right. \\ &3\left(\mathrm{coarse}\left(s\left[\mathrm{child}.x\right]\right) + \mathrm{coarse}\left(s\left[0, \mathrm{child}.y\right]\right)\right) + \\ &\left.\mathrm{coarse}\left(s\left[\mathrm{child}.x, \mathrm{child}.y\right]\right)\right)/16, \end{aligned} \tag{26}$$

  where the *coarse* operator accesses the value in a cell located on the coarser level l and $child.\,(x|y)$ are the coordinates of the grid point at level $l+1$ relative to its parent at level l, i.e. $child.\,(x|y) = \pm 1$.

- a *restriction* function which uses simple diagonal averaging of values defined in children cells located at level $l+1$ (small black dots in Figures 7a and 8a) to update the value in the parent halo cell located at level l (large blue circle in Figures 7a and 8a). For a quadtree without embedded boundaries, the restriction of a cell-centered scalar s from level $l+1$ to level l writes:

$$s\left[\,\right] = \frac{1}{2}\left(\frac{\mathrm{fine}\left(s\left[\,\right]\right) + \mathrm{fine}\left(s\left[1,1\right]\right)}{2} + \frac{\mathrm{fine}\left(s\left[1\right]\right) + \mathrm{fine}\left(s\left[0,1\right]\right)}{2}\right), \tag{27}$$

  where the *fine* operator accesses the value in a child cell located on the finer level $l+1$.

Complete details of the implementation of the *prolongation* and *restriction* functions in the absence of embedded boundaries and their extension to 3D can be found here: `http://basilisk.fr/src/grid/multigrid-common.h`.

4.2. Properties of tree grids in Basilisk *in the presence of embedded boundaries*

To use the Cartesian grid embedded boundary method presented in Section 3 on a tree grid, we extend the definition of:

- the embedded fractions c and f_d^f in resolution boundary halo cells;
- the prolongation and restriction functions (26) and (27), such that they use only values taken from *available* fluid cells while maintaining their second-order accuracy.

4.2.1. Computation of embedded fractions in a halo cell

We use prolongation and restriction functions specific to the embedded fractions to reconstruct the volume and area fractions c and f_d^f in a halo cell.

The restriction functions defining the embedded fractions in a parent halo cell at level l use simple averaging of values in its children cells at level $l+1$. The prolongation functions defining the embedded fractions in a halo cell at level $l+1$ are based on a VOF reconstruction of the embedded boundary normal vector $\bar{\mathbf{n}}_{\Gamma,l}$ and intercept α_l in the corresponding parent cell at level l. Indeed, using the area fractions $f_{d,l}^f$ defined on the faces $\mathcal{F}_{d,l}$ of the parent cell at level l, we use equation (13) to compute the corresponding normal vector $\bar{\mathbf{n}}_{\Gamma,l}$ and a variation of equation (16) to compute the intercept α_l. Then, given $\bar{\mathbf{n}}_{\Gamma,l}$ and α_l defined in a coordinate

system with origin the center of the halo cell and in which the halo cell size is unity, we use function (16) to compute the volume fraction c_{l+1} in the quadrant of the parent cell matching the halo cell. We also use $\bar{\mathbf{n}}_{\Gamma,l}$, α_l and equation (10) to compute the area fractions $f^{f,\text{int}}_{d,l+1}$ of the faces of the halo cell contained within the parent cell. We finally compute the area fractions $f^{f,\text{ext}}_{d,l+1}$ of the faces of the halo cell that coincide with the faces of the parent cell using their corresponding area fractions $f^f_{d,l}$.

We refer the reader to [68] for more details on the prolongation and restriction of a volume fraction and to the following for complete details on the implementation of the prolongation and restriction functions specific to the embedded fractions: `http://basilisk.fr/src/embed-tree.h#volume-fraction-field-cs`.

4.2.2. Prolongation and restriction functions in the presence of embedded boundaries

In the presence of embedded boundaries, we modify the *prolongation* and *restriction* functions (26) and (27) and define second-order operators using only values from available cells, i.e. cells not entirely contained within the embedded boundary and, if the discrete rigid boundary $\delta\Gamma_\Delta$ is moving, excluding *emerged* cells that have not yet been properly initialized.

Prolongation function in the presence of embedded boundaries. We detail here the extension of the prolongation function (26) in the presence of embedded boundaries. Without loss of generality, we consider the quadtree presented in Figure 7. To update the value of the cell-centered scalar s in the halo cell marked by a red circled cross in Figure 7, we distinguish four cases, depending on the number of coarse cells available (large black dots and large blue circle in Figure 7):

1. if all four coarse cells are available, as in Figure 7a, we use the bilinear interpolation (26).
2. if three coarse cells are available, excluding the diagonal coarse cell $[\text{child}.x, \text{child}.y]$, as in Figure 7b, we use the following triangular interpolation:
$$\begin{aligned} s\left[\,\right] = (&2\,\text{coarse}\left(s\left[\,\right]\right) + \\ &\text{coarse}\left(s\left[\text{child}.x\right]\right) + \text{coarse}\left(s\left[0, \text{child}.y\right]\right))/4. \end{aligned} \tag{28}$$
3. if only three coarse cells are available, including the diagonal coarse cell $[\text{child}.x, \text{child}.y]$, as in Figure 7c, we use the following diagonal interpolation:
$$s\left[\,\right] = \left(3\,\text{coarse}\left(s\left[\,\right]\right) + \text{coarse}\left(s\left[\text{child}.x, \text{child}.y\right]\right)\right)/4. \tag{29}$$
4. if only one or two coarse cells are available, as in Figure 7d, we face a pathological situation. In this case, we compute the value of s in the halo cell using a Taylor expansion, where, if possible, one simple face gradient $\bar{\nabla}^f_d s$ at level l is computed per dimension, preferably along the faces of the parent cell that follow the natural orientation of the halo cell with respect to its parent cell:
$$s\left[\,\right] = \text{coarse}\left(s\left[\,\right]\right) + \frac{\Delta}{4}\text{coarse}\left(\bar{\nabla}^f_x s\left[1\right]\right) + \frac{\Delta}{4}\text{coarse}\left(\bar{\nabla}^f_y s\left[\,\right]\right). \tag{30}$$
Note that in equation (30), the face gradient $\text{coarse}\left(\bar{\nabla}^f_y s\left[\,\right]\right)$ does not follow the natural orientation of the halo cell with respect to its parent cell represented in Figure 7d.

Complete details of the implementation of the prolongation function and its extension to 3D are presented here: `http://basilisk.fr/src/embed-tree.h#refinementprolongation-of-cell-cente`

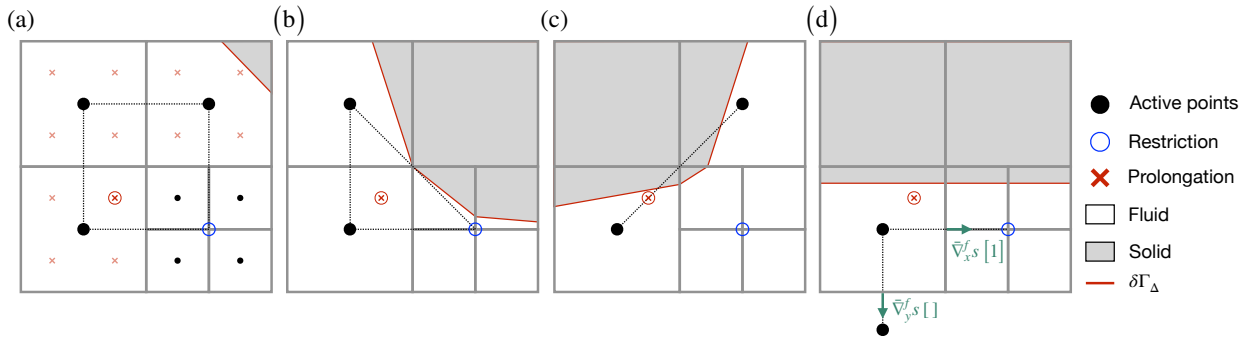


Figure 7: Graphical representation of the prolongation of a cell-centered scalar in the *halo* cell marked by a red circled cross, in the presence of embedded boundaries.

Restriction function in the presence of embedded boundaries. Next, we describe the extension of the restriction function (27) in the presence of embedded boundaries. Without loss of generality, we consider the quadtree presented in Figure 8. To update the value of the cell-centered scalar s in the halo cell marked by a blue circle in Figure 8, we distinguish three cases, depending on the number of children cells available (black dots in Figure 8):

1. if four children cells are available, as in Figure 8a, we use the simple diagonal averaging (27).

2. if three children cells are available, as in Figure 8b, we also use diagonal averaging, but only along the available diagonal:

$$s\,[\,] = \begin{cases} \text{if fine}\,(c\,[\,]) > 0 \text{ and fine}\,(c\,[1,1]) > 0: \\ \qquad \dfrac{\text{fine}\,(s\,[\,]) + \text{fine}\,(s\,[1,1])}{2} \\ \text{if fine}\,(c\,[1]) > 0 \text{ and fine}\,(c\,[0,1]) > 0: \\ \qquad \dfrac{\text{fine}\,(s\,[1]) + \text{fine}\,(s\,[0,1])}{2}. \end{cases} \tag{31}$$

3. if less than three children cells are available, as in Figure 8c, we face a pathological situation. In this case, we first compute the average value of s, denoted $\bar{s}$, over all available children cells as well as the barycenter $\mathbf{b}$ of the available children cells (green star in Figure 8c). We then use, if possible, a user-provided value of the gradient of s defined at the center of the cell (blue circle in Figure 8c), denoted $\nabla s\,[\,]$, and a Taylor expansion to improve our approximation of s in the halo cell:

$$s[] = \bar{s} - \mathbf{b} \cdot \nabla s\,[\,]. \tag{32}$$

   If an *a priori* value of $\nabla s\,[\,]$ is not available, the restriction function in this pathological situation is first-order only.

Complete details of the implementation of the restriction function and its extension to 3D are presented here: `http://basilisk.fr/src/embed-tree.h#restriction-of-cell-centered-fields`.

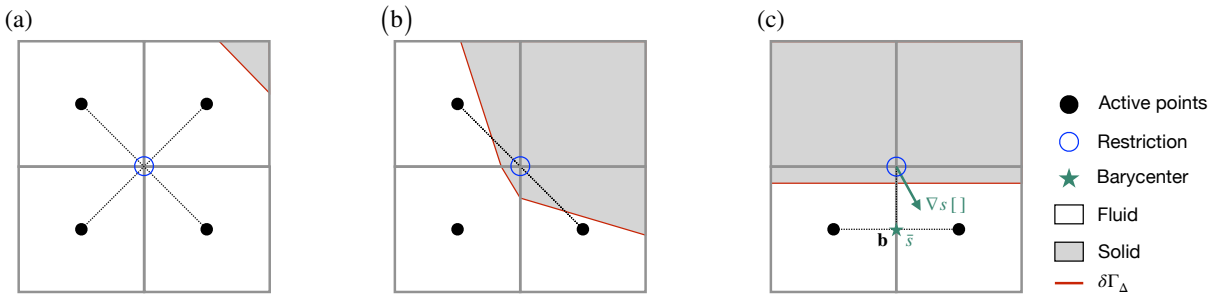


Figure 8: Graphical representation of the restriction of a cell-centered scalar in the *halo* cell marked by a large blue circle, in the presence of embedded boundaries.

4.2.3. Adaptive mesh refinement of quad/oc-trees in Basilisk

Finally, we describe the adaptive mesh refinement of tree grids available in *Basilisk*. Indeed, tree grids can be dynamically updated and cells refined or coarsened based on a multi-resolution analysis of selected scalar fields [50, 70, 71]. In particular, mesh adaptation allows us to minimize the use of halo cells in regions of strong Hessian by dynamically refining the grid in these regions and therefore maintain overall second-order accuracy when computing gradients. Indeed, values in halo cells are computed with second-order accuracy and therefore lead to first-order accuracy for gradients.

As an example, consider a cell-centered scalar s discretized at the grid level l and denoted s_l. To determine if a cell must be refined, coarsened or left unchanged, we first use the restriction function 4.2.2 to downsample s_l to the coarser level $l-1$:

$$s_{l-1} = \text{restriction}\left(s_l\right), \tag{33}$$

We then use the prolongation function 4.2.2 to upsample s_{l-1} to the current level l:

$$g_l = \text{prolongation}\left(s_{l-1}\right). \tag{34}$$

The *prolongated* value g_l is then compared to the original value s_l, and the current cell is either refined, coarsened or remains unchanged based on a user-defined threshold value ξ_{adapt} for the absolute (not relative) error $\|g_l - s_l\|$.

In the following section, we apply the Cartesian grid embedded boundary method presented previously to solve the coupled system of equations (1)–(2) and (4)–(5) describing the motion of a rigid body Γ in an incompressible fluid.

5. Temporal discretization

We describe here the temporal discretization of the coupled fluid-solid system of equations (1)–(2) and (4)–(5) on a uniform Cartesian grid Ω_Δ, for simplicity. The extension to a tree grid is straightforward following the methodology detailed in Section 4. Taking advantage of the conservative properties of the Cartesian embedded boundary method, known to enhance the stability and convergence properties of fluid-solid coupling strategies [21, 22], we implement here an first-order in time explicit weak coupling strategy, also for simplicity reasons.

At any given discrete time t^n, denoted with the superscript ^{n} and with Δt the time step, we assume that the velocity $\mathbf{u}^n$ and the fractional step pressure p^n are known in each cell. We also assume that the position $\mathbf{x}_\Gamma^n$ of the rigid body Γ as well as its translation and rotational velocities $\mathbf{u}_\Gamma^n$ and ω_Γ^n are known. Finally, we assume that the location of the discrete rigid boundary $\delta\Gamma_\Delta^n$ is known and that in each cut-cell the corresponding volume fraction c^n and area fraction $f_d^{f,n}$ of each face $\mathcal{F}_d$ of the cut-cell are also known.

5.1. Temporal discretization of the motion of the rigid body Γ

We first integrate system (4), describing the motion of the rigid body Γ, from time t^n to time t^{n+1} using the following first-order explicit time discretization:

$$\frac{\mathbf{u}_\Gamma^{n+1} - \mathbf{u}_\Gamma^n}{\Delta t} = \frac{\mathbf{F}_\Gamma^n}{\rho_\Gamma V_\Gamma} + \left(1 - \frac{\rho}{\rho_\Gamma}\right)\mathbf{g} \tag{35a}$$

$$\frac{\mathbf{I}_\Gamma^{n+1}\omega_\Gamma^{n+1} - \mathbf{I}_\Gamma^n\omega_\Gamma^n}{\Delta t} = \mathbf{T}_\Gamma^n \tag{35b}$$

$$\frac{\mathbf{x}_\Gamma^{n+1} - \mathbf{x}_\Gamma^n}{\Delta t} = \frac{\mathbf{u}_\Gamma^n + \mathbf{u}_\Gamma^{n+1}}{2}. \tag{35c}$$

Note that if we prescribe the motion of the rigid body analytically, we simply compute $\mathbf{x}_\Gamma^{n+1}$, $\mathbf{u}_\Gamma^{n+1}$ and ω_Γ^{n+1} using user-provided functions.

Given the new position of the rigid body $\mathbf{x}_\Gamma^{n+1}$, we then update the location $\delta\Gamma_\Delta^{n+1}$ of the discrete rigid boundary using the distance function $\Phi\left(\mathbf{x} - \mathbf{x}_\Gamma^{n+1}\right)$. In each cut-cell, we finally compute the corresponding volume fraction c^{n+1} and area fraction $f_d^{f,n+1}$ of each face $\mathcal{F}_d$ of the cut-cell using the methodology described in Section 3.3.

Complete details of the temporal discretization of system (4) can be found here: `http://basilisk.fr/sandbox/ghigo/src/myembed-particle.h`, and here: `http://basilisk.fr/sandbox/ghigo/src/myembed-moving.h#prediction`.

5.2. Hydrodynamic force and torque

In order to update the position and velocities of the rigid body Γ using system (35), we compute the hydrodynamic force $\mathbf{F}_\Gamma^n$ and torque $\mathbf{T}_\Gamma^n$ exerted by the fluid on the discrete rigid boundary $\delta\Gamma_\Delta^n$ using equations (5) discretized with second-order accuracy.

In each cut-cell, we compute the pressure contribution to the force and torque by linearly interpolating the pressure p^n from the center of the cell to the centroid $\mathbf{b}^n$ of the discrete rigid boundary $\delta\Gamma_\Delta^n$ in the cut-cell.

We then compute the viscous contribution to the force and torque using equation (19), assuming that the velocity $\mathbf{u}^n$ is constant along the discrete rigid boundary $\delta\Gamma_\Delta^n$, i.e. $\nabla\mathbf{u}|_{\delta\Gamma_\Delta^n} \cdot \bar{\mathbf{t}}_\Gamma^n = 0$, where $\bar{\mathbf{t}}_\Gamma^n$ is the tangential vector to the discrete rigid boundary in the cut-cell.

Complete details can be found here: `http://basilisk.fr/src/embed.h#surface-force-and-vortic`

5.3. Temporal discretization of the Navier-Stokes equations

Next, we integrate the incompressible Navier-Stokes equations (1) from time t^n to time t^{n+1} using a fractional-step projection method [72, 68]. As in [22], we assume that the discrete rigid boundary has been updated to its location $\delta\Gamma_\Delta^{n+1}$ and that the values of cell-centered fields in

full and cut-cells of the domain $\Omega_\Delta/\Gamma_\Delta^{n+1}$ are known at time t^n. We then obtain the following first-order time discretization of system (1), $\forall x \in \Omega_\Delta/\Gamma_\Delta^{n+1}$:

$$\frac{\mathbf{u}^\star - \mathbf{u}^n}{\Delta t} + \mathbf{A}^{n+\frac{1}{2}} = 0 \tag{36a}$$

$$\rho\frac{\mathbf{u}^{\star\star} - \mathbf{u}^\star}{\Delta t} = \nabla \cdot [2\mu\mathbf{D}^{\star\star}] \tag{36b}$$

$$\nabla \cdot \left[\frac{\Delta t}{\rho}\nabla p^{n+1}\right] = \nabla \cdot \mathbf{u}^{\star\star} \tag{36c}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^{\star\star} - \frac{\Delta t}{\rho}\nabla p^{n+1}, \tag{36d}$$

where the term $\mathbf{A}^{n+\frac{1}{2}}$ is an approximation of the nonlinear advection term $[\mathbf{u}\cdot\nabla\mathbf{u}]^{n+\frac{1}{2}}$, which we detail in Section 5.7.

Following the projection method described in [45], we enforce the incompressibility constraint (1b) using equations (36c) and (36d). In practice, since both the velocity $\mathbf{u}$ and the pressure p are collocated at center of each cell, we compute the divergence of the velocity $\nabla \cdot \mathbf{u}^{\star\star}$ in equation (36c) using an auxiliary face velocity $\mathbf{u}^f$ and the divergence operator (17). We therefore impose the incompressibility condition (1b) on the auxiliary face velocity $\mathbf{u}^f$ and the cell-centered velocity $\mathbf{u}$, computed using equation (36d), is then only approximately incompressible.

Complete details of the temporal discretization of the Navier-Stokes equations (36) can be found here: `http://basilisk.fr/src/navier-stokes/centered.h#time-integration`.

5.4. Discrete boundary conditions

We supply system (36) with the following no-slip Dirichlet boundary condition for velocity and Neumann boundary condition for pressure at time t^{n+1} on the discrete rigid boundary $\delta\Gamma_\Delta^{n+1}$:

$$\mathbf{u}^{n+1} = \mathbf{u}_\Gamma^{n+1} \tag{37a}$$

$$\nabla_\Gamma p^{n+1} = 0. \tag{37b}$$

Equation (37a) is the discrete equivalent of the no-slip boundary condition (2). The homogeneous Neumann boundary condition for pressure (37b), used when solving equation (36c), is suitable for a fixed rigid body [73, 74]. When considering a moving rigid body, a suitable boundary condition for pressure can be obtained by projecting the Navier-Stokes equation 1a along the normal to the discrete rigid boundary $\delta\Gamma_\Delta$ [73, 22]. Nevertheless, we have found that using the homogeneous Neumann boundary condition for pressure (37b) with a moving rigid body does not significantly affect the computed solution and we therefore use equation (37b) for simplicity.

5.5. Submerged and emerged cells

As mentioned previously, the motion on a grid Ω_Δ of the rigid body Γ represented by embedded boundaries results in the disappearance or appearance of fluid cells, respectively called *submerged* and *emerged* cells (also referred to as dead and fresh cells in [74, 75]). We characterize these cells using the time evolution of the volume fraction c in each cell:

- *submerged* cell: $c^n > 0$ and $c^{n+1} = 0$;
- *emerged* cell: $c^n = 0$ and $0 < c^{n+1} < 1$.

Note that we adjust the time step Δt such that a solid cell at time t^n should not become an emerged cell at time t^{n+1} with a volume fraction $c^{n+1} = 1$.

It is well known that the disappearance (respectively appearance) of submerged (respectively emerged) cells can lead to spurious mass sources or sinks [75, 22]. We therefore handle these events carefully to avoid nonphysical oscillations of the velocity and pressure. The incompressibility condition (1b), combined with the conservative discretization of the divergence operator (17), allows us to simply let submerged cells "disappear" and emerged cells "appear" from the *available* cells, i.e. cells that are in the domain $\Omega_\Delta/\Gamma_\Delta^{n+1}$. This is not the case for the compressible Navier-Stokes equations for example, for which a conservative handling of submerged and emerged cells is necessary [22, 54].

However, emerged cells of the domain $\Omega_\Delta/\Gamma_\Delta^{n+1}$ have no history at time t^n, which violates the assumption made in Section 5.3 that the values of cell-centered fields in full and cut-cells of the domain $\Omega/\Gamma_\Delta^{n+1}$ are known at time t^n. Therefore, for each cell-centered scalar s, we estimate the value s^n at the center of all emerged cells of the domain $\Omega_\Delta/\Gamma_\Delta^{n+1}$. Following an approach similar to the one described in [74] and illustrated in Figure 9, we employ the following second-order linear extrapolation in the direction of the normal vector $\bar{\mathbf{n}}_\Gamma^{n+1}$ to the discrete rigid boundary $\delta\Gamma_\Delta^{n+1}$ to estimate the value of s^n:

$$s^n = \frac{s_1^n\, d_2 - s_2^n\, d_1}{d_2 - d_1}, \tag{38}$$

where the interpolants s_1^n, s_2^n are computed as in Section 3.4.1 using values of s taken only from *available* cells and at the positive distances d_1 and d_2 from the extrapolation point. If the stencils for s_1^n or s_2^n are not topologically connected to the emerged cell, we use a similar approach as in Section 3.4.1 to treat degenerated cases.

Complete details of the treatment of emerged cells can be found here: `http://basilisk.fr/sandbox/ghigo/src/myembed-moving.h#prediction`.

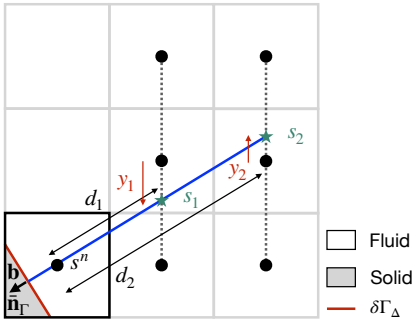


Figure 9: Graphical representation of the methodology implemented in *Basilisk* to extrapolate the value s^n at the center of an emerged cut-cell of a 2D grid when the principal direction of the normal vector $-\bar{\mathbf{n}}_\Gamma^{n+1}$ is the positive x-direction.

In the following, we describe the numerical schemes we use to solve equations (36a), (36b) and (36c). We follow closely the methodology detailed in [45, 68, 50].

5.6. Multigrid solver

We solve each component of the velocity $\mathbf{u}$ in equation (36b) and the pressure p in equation (36c) using a multigrid Poisson-Helmholtz solver. Indeed, multigrid solvers are known for being efficient at solving elliptic or parabolic problems such as equations (36b) and (36c), which can be written in a more general form as the following Poisson-Helmholtz equation:

$$\mathcal{L}(s) = b, \quad \mathcal{L}(s) = \nabla \cdot (\nabla s) + \lambda s. \tag{39}$$

The multigrid solver implemented in *Basilisk* to solve equation (39) has been described in detail in [50] and we recall here its essential features only.

Given an initial guess $\tilde{s}$, we define the error $\mathrm{d}s = s - \tilde{s}$, where s denotes the unknown exact solution to equation (39). We then use the linearity of the operator $\mathcal{L}$ to rewrite equation (39) as the following equivalent Poisson-Helmholtz equation:

$$\mathcal{L}(\mathrm{d}s) = b - \mathcal{L}(\tilde{s}) = \mathrm{res}, \tag{40}$$

where res is called the *residual*. We then proceed as follows:

1. Given an initial guess $\tilde{s}$, we compute the residual $\mathrm{res} = b - \mathcal{L}(\tilde{s})$;
2. If $\|\mathrm{res}\| \leq \xi_{\mathrm{mg}}$, where ξ_{mg} denotes the *tolerance* of the multigrid solver, the initial guess $\tilde{s}$ is good enough;
3. Otherwise, we solve equation (40);
4. We compute an updated initial guess $\tilde{s} = \tilde{s} + \mathrm{d}s$ and go back to step 1.

We use the multigrid solver itself only in step 3 and proceed as in [50]. We first decompose the error $\mathrm{d}s$ on successively coarser grids, using the restriction function (27), even in the presence of embedded boundaries. We then use a simplified prolongation function to define the error on successively finer grids. This simplified prolongation function is equivalent to function (26) if all 4 coarse cells (8 in 3D) are *available*. Otherwise, we use simple injection from the coarser level. Complete details of the simplified prolongation function can be found here: `http://basilisk.fr/src/embed.h#prolongation-for-the-multigrid-solver`.

The error $\mathrm{d}s$ is then efficiently reduced on each grid, starting from the coarsest grid at level $l = 1$, using only a few iterations of simple relaxation techniques such as Jacobi or Gauss-Seidel. These relaxation techniques rely on the discretization of the Laplacian operator $\nabla \cdot (\nabla)$ in each cell of the grid Ω_Δ. In each full cell, we proceed as in [50]. In each cut-cell, we simply use equation (17) to discretize the divergence operator $\nabla\cdot$ and equations (19) and (24) to respectively compute the flux $F_\Gamma = \nabla_\Gamma s$ through the discrete embedded boundary and the flux $F_d^f = \nabla_d^f s$ through each face $\mathcal{F}_d$ of the cut-cell. Finally, note that in practice we multiply equation (40) by the volume fraction c so that the resulting linear system is well-conditioned for any value of the volume fraction [60].

Complete details of the multigrid Poisson-Helmholtz solver can be found here: `http://basilisk.fr/src/poisson.h`.

5.7. Nonlinear advection equation

We detail here the computation of the term $\mathbf{A}^{n+\frac{1}{2}}$ used in the nonlinear advection equation (36a). Without loss of generality, we consider here a cut-cell of the grid Ω_Δ, defining a time-dependent control volume $\mathcal{V}(t)$, of boundary $\delta\mathcal{V}(t)$. We then compute in this cut-cell the following *integral* form of the nonlinear advection equation for the velocity, written in conservative form using the incompressibility condition (1b):

$$\int_{t^n}^{t^{n+\frac{1}{2}}} \mathrm{d}t \int_{\mathcal{V}(t)} \left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F} \right) \mathrm{d}V = 0, \tag{41}$$

where $\mathbf{F} = \mathbf{u} \otimes \mathbf{u}$. Using the Leibniz integration rule and the divergence theorem, we rewrite equation (41) as:

$$\Delta^D \left(c\,\mathbf{u} \right)_{t^n}^{t^{n+\frac{1}{2}}} - \int_{t^n}^{t^{n+\frac{1}{2}}} \mathrm{d}t \int_{\delta\mathcal{V}(t)} \left(\mathbf{u} \left(\mathbf{u}_{\delta V} \cdot \mathbf{n} \right) \right) \mathrm{d}A + \int_{t^n}^{t^{n+\frac{1}{2}}} \mathrm{d}t \int_{\mathcal{V}(t)} \nabla \cdot \mathbf{F} \mathrm{d}V = 0, \tag{42}$$

where $\mathbf{u}_{\delta\mathcal{V}}$ is the velocity of the boundary $\delta\mathcal{V}(t)$. As the only moving boundary of the control volume $\mathcal{V}(t)$ is the discrete rigid boundary $\delta\Gamma_\Delta(t)$, we rewrite equation (42) as:

$$\Delta^D \left(c\,\mathbf{u} \right)_{t^n}^{t^{n+\frac{1}{2}}} - \int_{t^n}^{t^{n+\frac{1}{2}}} \mathrm{d}t \int_{\delta\Gamma_\Delta(t)} \left(\mathbf{u}_\Gamma \left(\mathbf{u}_\Gamma \cdot \mathbf{n} \right) \right) \mathrm{d}A + \int_{t^n}^{t^{n+\frac{1}{2}}} \mathrm{d}t \int_{\mathcal{V}(t)} \nabla \cdot \mathbf{F} \mathrm{d}V = 0. \tag{43}$$

Then, assuming that the velocity $\mathbf{u}_\Gamma$ of the rigid body Γ is constant along the discrete boundary $\delta\Gamma_\Delta(t)$ in the cut-cell, we obtain:

$$\Delta^D \left(c\,\mathbf{u} \right)_{t^n}^{t^{n+\frac{1}{2}}} - \Delta^D \int_{t^n}^{t^{n+\frac{1}{2}}} \left(\mathbf{u}_\Gamma \frac{\mathrm{d}c}{\mathrm{d}t} \right) \mathrm{d}t + \int_{t^n}^{t^{n+\frac{1}{2}}} \mathrm{d}t \int_{\mathcal{V}(t)} \nabla \cdot \mathbf{F} \mathrm{d}V = 0. \tag{44}$$

Finally, by approximating the second and third terms in equation (44) at time $t^{n+\frac{1}{2}}$ and using equation (17) to discretize the divergence operator $\nabla \cdot \mathbf{F}$, we rewrite equation (44) as equation (36a) with:

$$\mathbf{A}^{n+\frac{1}{2}} = \frac{c^{n+\frac{1}{2}} - c^n}{c^{n+\frac{1}{2}}} \frac{\mathbf{u}^n - \mathbf{u}_\Gamma^{n+\frac{1}{2}}}{\Delta t} + \frac{1}{c^{n+\frac{1}{2}} \Delta} \left(\sum_d f_d^f F_d^f + f_\Gamma F_\Gamma \right)^{n+\frac{1}{2}}. \tag{45}$$

The first term on the right-hand side of equation (45) accounts for the possible change of the volume of a cut-cell in the presence of moving embedded boundaries, while the second term on the right-hand side of equation (45) is simply the discretization (17) of the divergence operator $\nabla \cdot \mathbf{F}$ in a cut-cell.

As in [45, 68], we compute the flux $F_d^{f,n+\frac{1}{2}}$ through each face $\mathcal{F}_d$ of full and cut-cells using the explicit and conservative Bell-Colella-Glaz second-order unsplit upwind scheme [63]. We then compute the flux $F_\Gamma^{n+\frac{1}{2}}$ through the embedded boundary $\delta\Gamma_\Delta^{n+\frac{1}{2}}$ simply using the no-slip Dirichlet boundary condition for velocity (37a). Finally, note that in practice we solve the nonlinear advection equation (36a) using equation (45) with values of the embedded fractions evaluated at time t^{n+1} and not time $t^{n+\frac{1}{2}}$.

Complete details of the implementation of the Bell-Colella-Glaz numerical scheme can be found here: `http://basilisk.fr/src/bcg.h`, and of the volume correction term in equation (45) here: `http://basilisk.fr/sandbox/ghigo/src/myembed.h#lifting-the-small-cell-cfl-restr`

5.8. CFL condition and small cell restriction

The principal limitation of Cartesian grid embedded boundary methods is the well-known small cell problem [62, 49, 22]. Indeed, the volume fraction $c^{n+\frac{1}{2}}$, that appears at the denominator of the right-hand side of equation (45), can become arbitrarily small depending on the intersection of the discrete rigid boundary $\delta\Gamma_\Delta$ with the grid Ω_Δ. This translates for the explicit nonlinear advection equation (36a) in the following small cell CFL condition:

$$\Delta t_{\text{sc}} < \frac{c}{f_d^f} \frac{\Delta}{|u_d^f|}, \quad \forall \text{ faces } \mathcal{F}_d, \tag{46}$$

where the time step Δt_{sc} may become arbitrarily small if the ratio c/f_d^f goes to zero, rendering any time-dependent simulation impossible.

Numerous strategies have been proposed to avoid this problem, including cell merging techniques, where small cells are merged with neighboring larger cells [76, 45], or special difference schemes that properly balance the volume fraction $c^{n+\frac{1}{2}}$ in equation (45) [77]. We choose here to use the simple and efficient flux redistribution technique [62, 49] that algebraically expands the range of influence of small cells to neighboring cells to obtain a stable method.

In each cut-cell, we first compute two approximations of the nonlinear advection term $\mathbf{A}^{n+\frac{1}{2}}$ in equation (36a):

- a conservative but unstable term $\mathbf{A}_{\text{c}}^{n+\frac{1}{2}}$ using equation (45), responsible for updating the velocity $\mathbf{u}$ in large cut-cells and full cells;
- a non-conservative but stable term $\mathbf{A}_{\text{nc}}^{n+\frac{1}{2}}$, responsible for updating the velocity $\mathbf{u}$ in small cut-cells.

Following [55], we compute in each cut-cell the non-conservative term $\mathbf{A}_{\text{nc}}^{n+\frac{1}{2}}$ using the following weighted average of the conservative term $\mathbf{A}_{\text{c}}^{n+\frac{1}{2}}$, in which we do not include the volume correction term:

$$\mathbf{A}_{\text{nc}}^{n+\frac{1}{2}} = \frac{\sum_{\text{cell}\in\mathcal{N}} \left(c^{n+\frac{1}{2}}\right)^2 \mathbf{A}_{\text{c}}^{n+\frac{1}{2}}}{\sum_{\text{cell}\in\mathcal{N}} \left(c^{n+\frac{1}{2}}\right)^2}, \tag{47}$$

where the neighborhood $\mathcal{N}$ corresponds to all full and cut-cells in the 3×3 ($3 \times 3 \times 3$ in 3D) stencil of the cut-cell. We then update the velocity in each cut-cell using the following interpolation between $\mathbf{A}_{\text{c}}^{n+\frac{1}{2}}$ and $\mathbf{A}_{\text{nc}}^{n+\frac{1}{2}}$:

$$\frac{\mathbf{u}^\star - \mathbf{u}^n}{\Delta t} + \lambda\left(c^{n+\frac{1}{2}}\right) \mathbf{A}_{\text{c}}^{n+\frac{1}{2}} + \left(1 - \lambda\left(c^{n+\frac{1}{2}}\right)\right) \mathbf{A}_{\text{nc}}^{n+\frac{1}{2}} = 0. \tag{48}$$

Following [49, 78], we compute the interpolation factor $\lambda(c)$ as:

$$\lambda(c) = \begin{cases} 0 & \text{if} \quad \dfrac{\Delta t_{\text{sc}}}{\Delta t} < 0 \\ 3\left(\dfrac{\Delta t_{\text{sc}}}{\Delta t}\right)^2 - 2\left(\dfrac{\Delta t_{\text{sc}}}{\Delta t}\right)^3 & \text{if} \quad 0 \leq \dfrac{\Delta t_{\text{sc}}}{\Delta t} \leq 1 \\ 1 & \text{if} \quad \dfrac{\Delta t_{\text{sc}}}{\Delta t} > 1, \end{cases} \tag{49}$$

where Δt is the time step limited by the standard CFL condition:

$$\Delta t < \frac{\Delta}{|u_d^f|}, \quad \forall \text{ faces } \mathcal{F}_d. \tag{50}$$

Using the interpolation factor (49) allows us to delay the apparition of small-cells while removing the small cell limitation as $\lambda(c)$ is proportional to the volume fraction c.

Finally, we maintain overall conservation in each cut-cell by redistributing in a conservative manner the following defect in momentum e:

$$e^{n+\frac{1}{2}} = c^{n+\frac{1}{2}} \left(1 - \lambda\left(c^{n+\frac{1}{2}}\right)\right) \left(\mathbf{A}_{\mathrm{c}}^{n+\frac{1}{2}} - \mathbf{A}_{\mathrm{nc}}^{n+\frac{1}{2}}\right), \tag{51}$$

in the 3×3 ($3 \times 3 \times 3$ in 3D) stencil of the cut-cell, proportionally to the square of the volume fraction $\left(c^{n+\frac{1}{2}}\right)^2$. We therefore obtain a locally conservative and stable scheme for the nonlinear advection equation (36a), which verifies the standard CFL condition (50). In practice, we use a CFL number of 0.5 similar to that of a VOF scheme.

Complete details of the treatment of small cells can be found here: `http://basilisk.fr/sandbox/ghigo/src/myembed.h#lifting-the-small-cell-cfl-restriction`.

6. Numerical validation

We now present several validation cases for the Poisson-Helmholtz, heat, Stokes and Navier-Stokes equations in the presence of fixed and moving rigid bodies. These test cases are part of a larger test suite (accessible here: `http://basilisk.fr/sandbox/ghigo/src`) and are selected to highlight specific features of the Cartesian grid embedded boundary method presented in the previous sections. We analyze in particular the accuracy and robustness of the method by comparing our results to analytical, numerical and experimental solutions from the literature.

In each test case, we consider a square (cubic in 3D) computational domain Ω of length L_0 in which we embed a fixed or moving rigid body Γ. We characterize the spatial discretization using the level of refinement l, or equivalently the number of cells N in each direction. A uniform Cartesian grid is therefore defined by one level l and contains $N = 2^l$ cells in each direction, whereas a tree grid is defined by a maximum level $l_{\max}$ and a minimum level $l_{\min}$. The number of leaf cells in each directions then varies between $N_{\min} = 2^{l_{\min}}$ and $N_{\max} = 2^{l_{\max}}$, depending on the selected scalar fields that govern the dynamic adaptation of the tree grid.

To determine the accuracy of a computed solution, we define the p-norm of a cell-centered scalar s, with $p = \{1, 2\}$, as:

$$\|s\|_p = \frac{\sum_{\text{leaf}} |s[\,]^p| c[\,] \Delta^D}{\sum_{\text{leaf}} c[\,] \Delta^D}, \tag{52}$$

where $\sum_{\text{leaf}}$ denotes the sum over all the leaf cells of the domain. We also define the ∞-norm of s, denote $\|s\|_\infty$, as the maximum over all leaf cells of the absolute value of s. Finally, knowing two solutions computed on grids respectively characterised by a maximum level l_1 and l_2, the convergence rate in any given p-norm can be estimated as:

$$O_p = \frac{\log\left(\|s_1\|_p / \|s_2\|_p\right)}{(l_2 - l_1)\log(2)}. \tag{53}$$

If the convergence rate $O_p = n$, this indicates n^{th}-order accuracy, i.e., the leading term in the truncation error scales as $O(\Delta^n)$.

6.1. Poisson-Helmholtz equation with Dirichlet boundary conditions in a domain defined by a 2D rhodonea curve

To establish the second-order accuracy of the multigrid Poisson-Helmholtz solver in the presence of fixed embedded boundaries, we solve the following Poisson-Helmholtz equation:

$$\nabla \cdot \nabla s = 7r^2 \cos 3\theta, \tag{54}$$

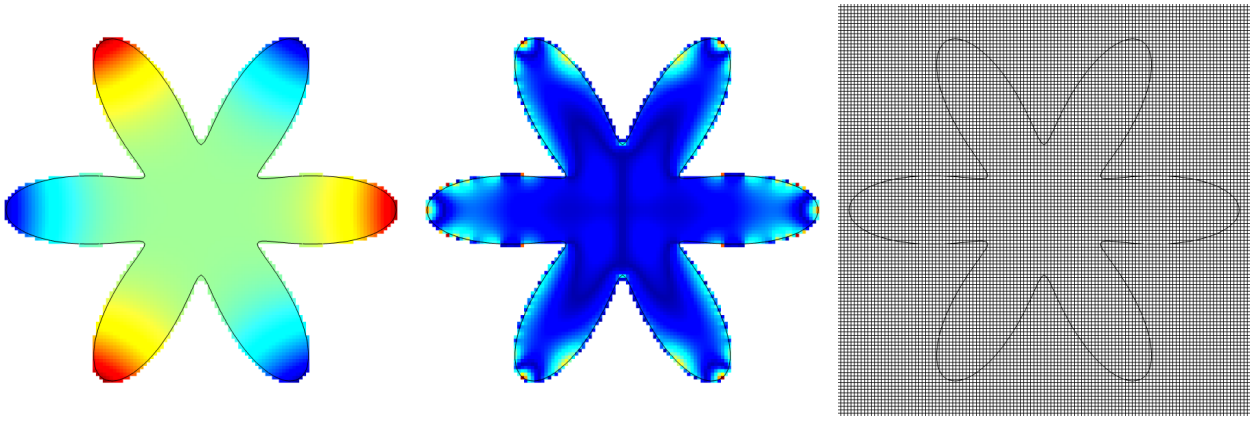
using Dirichlet boundary conditions applied on the 2D embedded boundary defined by the following rhodonea curve, also illustrated in Figure 10:

$$r \leq 0.3 + 0.15 \cos 6\theta. \tag{55}$$

This test case was originally proposed in [60], where the authors compared their results, computed using a second-order Cartesian grid embedded boundary methodology similar to the one described in Section (3.4.1), to the exact solution to equation (54):

$$s\,(r,\,\theta) = r^4 \cos 3\theta. \tag{56}$$

We solve equation (54) using the multigrid solver with a tolerance set to $\xi_{\text{mg}} = 10^{-6}$. Following [60, 45], we evaluate the right-hand side of equation (54) at the centroid of each cell, whereas we evaluate the exact solution (56) at the center of each cell.



(a) Solution $s\,[\,]$. (b) Error. (c) Uniform Cartesian grid with $N = 128$

Figure 10: Solution to the Poisson-Helmholtz equation (54) computed on a uniform Cartesian grid with $N = 128$ cells and using Dirichlet boundary conditions: (a) cell-centered solution; (b) cell-centered error with the exact solution (56) and (c) uniform Cartesian grid. `http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c`.

We first solve equation (54) on a uniform grid. We represent in Figure 10a the solution, computed on the uniform Cartesian grid with $N = 128$ cells displayed in Figure 10c, and the corresponding error in Figure 10b. We observe that the computed solution matches perfectly the contour plot of the exact solution (of Problem 1) displayed in Figure 7 of [60].

To further assess the accuracy of the multigrid solver, we plot in Figure 11 the evolution with the number of cells N of the 1-norm (see Figures 11a and 11b) and ∞-norm (see Figures 11c and 11d) of the error between the computed and exact solutions. We observe that the

error in cut-cells converges for all values of N at the rate $n \approx 3$, higher than the expected second-order accuracy of the multigrid solver. This behavior was also observed in [60] and was explained by a *dipole* behavior of the solution in cut-cells when using Dirichlet boundary conditions. However, even though the error in full cells first converges at the same rate $n \approx 3$ for coarser grids ($32 \leq N \leq 256$), it eventually converges at the expected convergence rate $n \approx 2$ for finer grids ($512 \leq N \leq 2048$). Indeed, as observed in Figures 11a and 11c, the error for coarser grids is larger in cut-cells and therefore affects the convergence rate of the error in full cells.

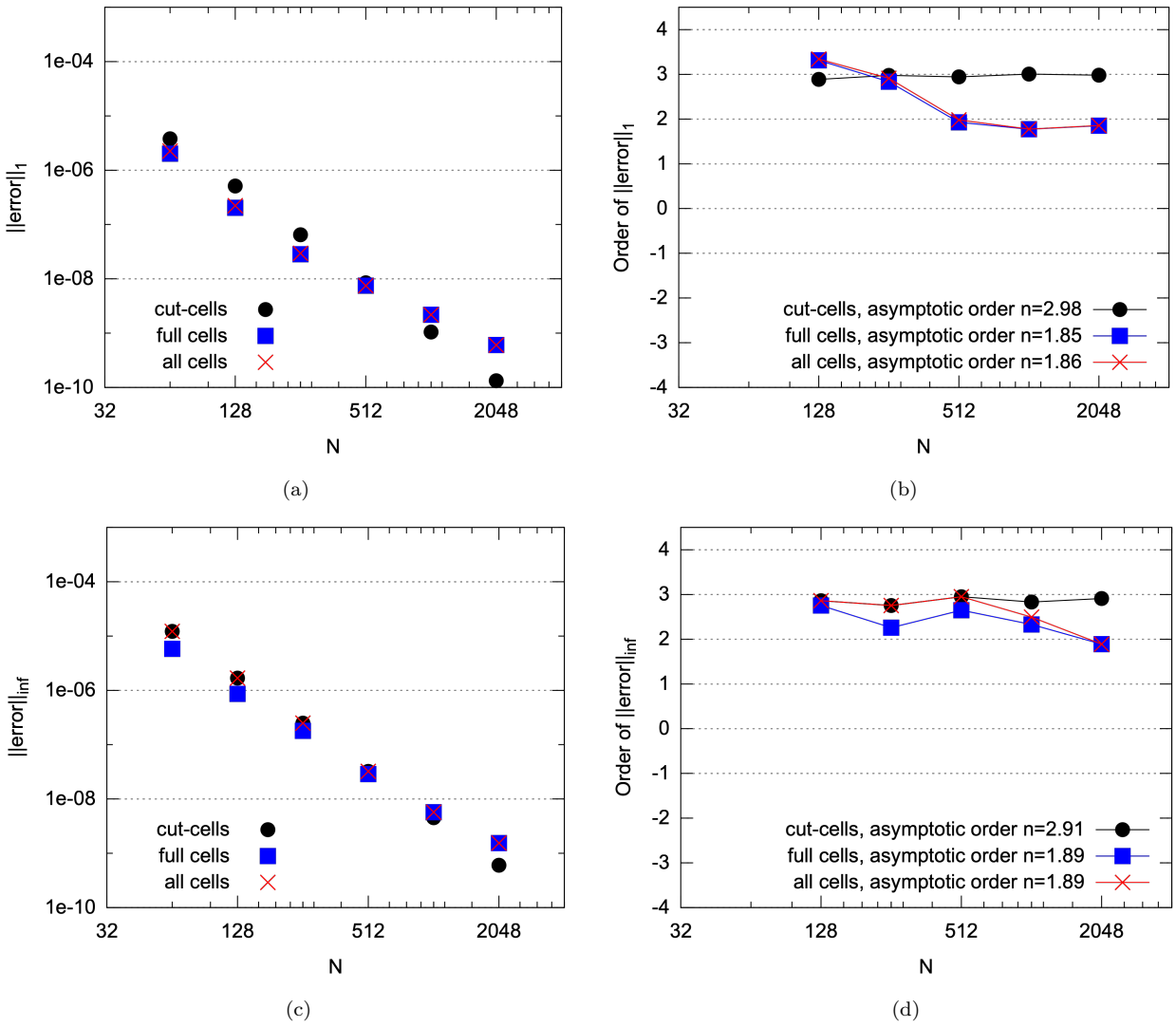


Figure 11: Evolution with the number of cells N of the error between the computed and exact solutions of the Poisson-Helmholtz equation (54) on a uniform Cartesian grid and using Dirichlet boundary conditions: (a) and (b): $\|\text{error}\|_1$ and the corresponding convergence rate; (c) and (d): $\|\text{error}\|_\infty$ and the corresponding convergence rate. `http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c`.

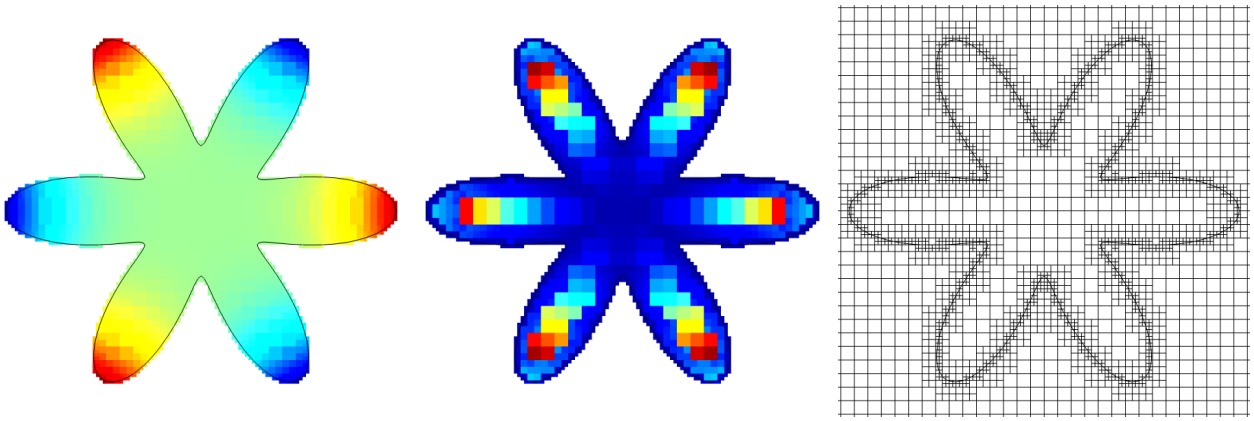
We now consider a *static* quadtree where the mesh is locally refined up to level l only around the embedded boundary and coarsened up to level $l-2$ everywhere else. We represent in Figure

12a the solution, computed on the quadtree defined by $N_{\min} = 32$ and $N_{\max} = 128$ displayed in Figure 12c, and the corresponding error in Figure 12b. We observe that the solution closely resembles the one we obtain on a uniform grid in Figure 10a. We also notice in Figure 12b that, contrary to Figure 10b, the largest error now occurs in the core of the domain, where the grid is coarser.

In Figure 13, we then plot the evolution with the maximum number of cells $N_{\max}$ of the 1-norm (see Figures 13a and 13b) and ∞-norm (see Figures 13c and 13d) of the error between the computed and exact solutions. We observe that the error in cut-cells and full cells converges at the rate $n \approx 2$ (the ∞-norm of the error in cut-cells converges at an average rate $n = 1.77$). We therefore do not recover the convergence rate $n \approx 3$ observed previously in cut-cells of a uniform grid. Indeed, due to local mesh refinement near the embedded boundary, we use values in halo cells to compute face gradients in cut-cells. These halo cell values are computed using the second-order restriction and prolongation functions defined in Section 4.2.2, which break the dipole behavior of the solution. Furthermore, due to the complex shape of the embedded boundary defined by equation (55), pathological situations for the restriction function are likely to arise. As a reminder, these occur when two or less cells are available for the restriction function. In this case, we maintain the convergence rate $n \approx 2$ observed in Figure 13 by providing an *a priori* value for the gradient ∇s, derived from the exact solution (56) and used in equation (32).

This test case therefore highlights that manually adapting the mesh, irrespective of the values of the scalar s in the computational domain, is not the correct way of refining the mesh on a tree grid. Indeed, in most flow configurations we cannot provide an *a priori* value of the gradient ∇s necessary here to maintain second-order accuracy. Nevertheless, we expect that by dynamically refining the mesh based on the multi-resolution analysis of selected scalar fields available in *Basilisk*, we will recover a convergence rate $n = 2$ nonetheless.

Complete details of this test case can be found here: `http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c`.



(a) Solution s [].

(b) Error.

(c) Static quadtree with $N_{\max} = 128$.

Figure 12: Solution to the Poisson-Helmholtz equation (54) computed on a static quadtree with $N_{\max} = 128$ and $N_{\min} = 32$ and using Dirichlet boundary conditions: (a) cell-centered solution; (b) cell-centered error with the exact solution (56) and (c) static quadtree. `http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c`.

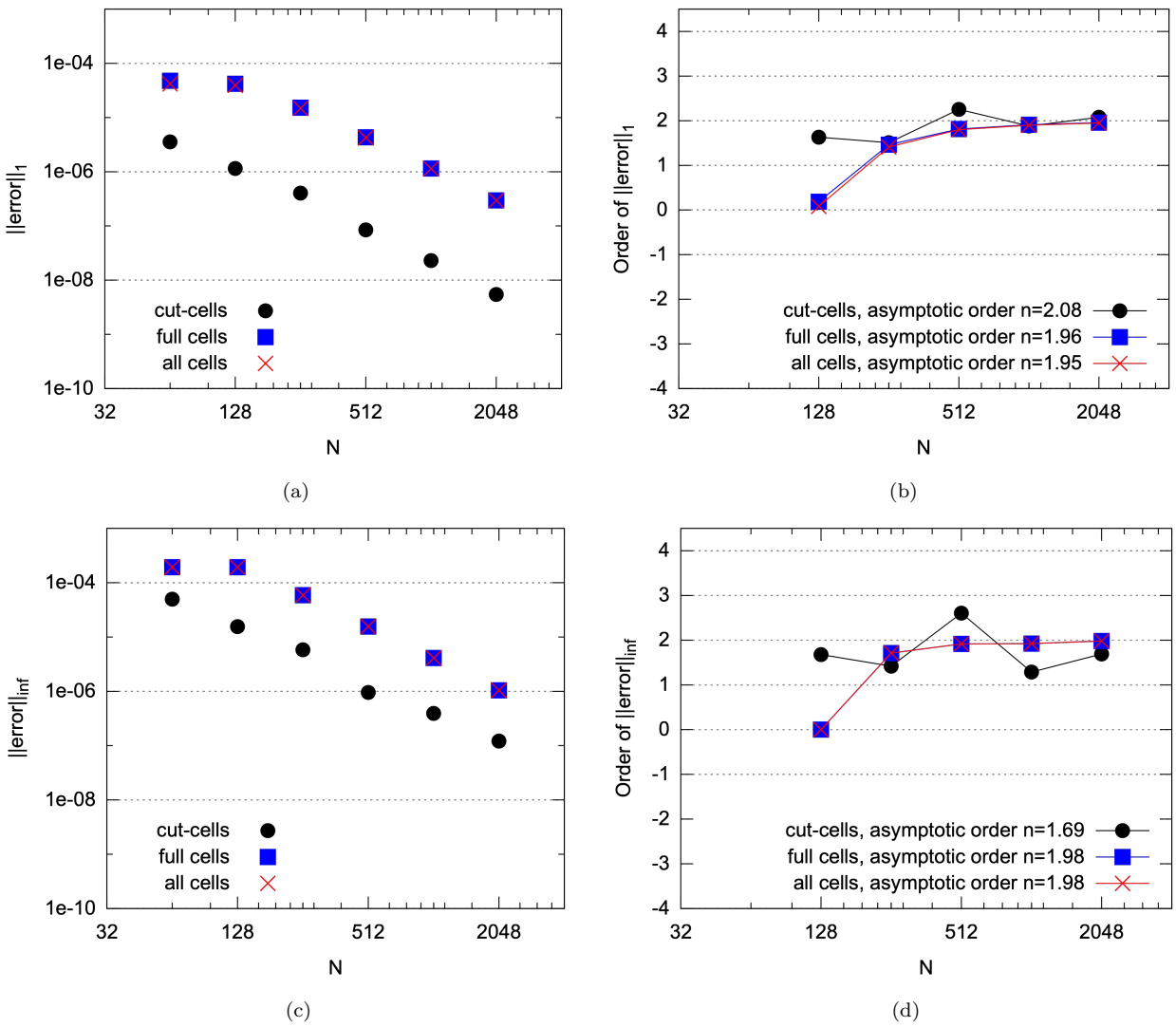


Figure 13: Evolution with the number of cells $N_{\max}$ of the error between the computed and exact solutions of the Poisson-Helmholtz equation (54) on a quadtree and using Dirichlet boundary conditions: (a) and (b): $\|\text{err}\|_1$ and the corresponding convergence rate; (c) and (d): $\|\text{err}\|_\infty$ and the corresponding convergence rate. `http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c`.

6.2. Heat equation with Neumann boundary conditions in a 3D expanding sphere

We now establish the second-order accuracy of the multigrid Poisson-Helmholtz solver in the presence of 3D and moving rigid embedded boundaries. We solve the following heat equation:

$$\partial_t s + \nabla \cdot \nabla s = \frac{4}{125\pi} \frac{r^2 + 5(t+1)}{(t+1)^3} \exp\left(-\frac{r^2}{5(t+1)}\right), \tag{57}$$

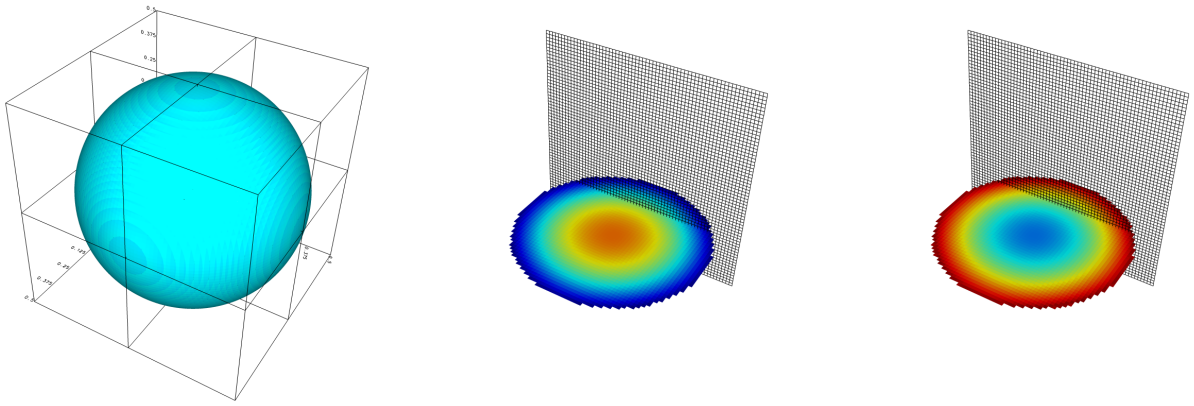
using Neumann boundary conditions applied on the 3D moving embedded boundary defined by the following equation for an expanding sphere, also illustrated in Figure 14:

$$r^2 \leq 0.392 + t, \quad t \leq 0.1. \tag{58}$$

This test case was originally proposed in [61], where the authors compared their results, computed using a second-order Cartesian grid embedded boundary method similar to the one described in Section 3.4.1, to the exact solution to equation (57):

$$s\left(r,\,\theta\right) = \frac{4}{5\pi(t+1)} \exp\left(-\frac{r^2}{5(t+1)}\right). \tag{59}$$

We solve equation (57) from time $t = 0$ to time $t = 0.1$ using an implicit first-order time discretization with $\Delta t = 10^{-5}$ to minimize time discretization and splitting errors. As the sphere expands, we initialize the scalar s in emerged cells using the second-order extrapolation described in Section 5.5. At each time step, we therefore solve a discrete Poisson-Helmholtz equation using the multigrid solver with a tolerance set to $\xi_{\text{mg}} = 10^{-6}$. Note that, following [61], we evaluate the right-hand-side of equation (57) at the centroid of each cell, whereas we compute the exact solution (59) at the center of each cell.



(a) Solution $s\,[\,]$ at $t = 0.1$. (b) Solution $s\,[\,]$ in the plane $y = -0.246$. (c) Error in the plane $y = -0.246$.

Figure 14: Solution to the heat equation (57) at time $t = 0.1$, computed on a uniform Cartesian grid with $N = 64$ cells and using Neumann boundary conditions: (a) cell-centered solution on the embedded boundary; (b) cell-centered solution in the $x-z$ plane defined by $y = -0.246$ and (c) cell-centered error with the exact solution (59) in the $x-z$ plane defined by $y = -0.246$. `http://basilisk.fr/sandbox/ghigo/src/test-heat/neumann3D.c`.

We solve equation (57) on a uniform grid. As an example, we represent in Figures 14a and 14b the solution, computed on a uniform Cartesian grid with $N = 64$ cells, and the corresponding error in Figure 14c. We observe that in this case the largest error occurs near the embedded boundary.

In Figure 15, we then plot the evolution with the number of cells N of the 1-norm (see Figures 15a and 15b) and ∞-norm (see Figures 15c and 15d) of the error between the computed and exact solutions. As in [61], we observe in full cells that the 1-norm and the ∞-norm of the error converge at the expected rate $n \approx 2$. In cut-cells, only the 1-norm of the error converges at a rate $n \approx 2$, whereas the ∞-norm of the error converges at an average rate $n \approx 1.33$. This behavior is notably different from the one observed in Section 6.1, where we obtain a convergence rate $n \approx 3$ in cut-cells. The reason is twofold: (i) when using Neumann boundary conditions, the convergence rate of the multigrid Poisson-Helmholtz solver reduces to $n = 2$ in cut-cells [60, 45]; (ii) as the sphere expands isotropically, neighboring emerged cells are likely

to appear simultaneously. In this case, the number of available cells in the stencil of these emerged cells is too small to use the second-order extrapolation presented in Section 5.5 and we use a first-order extrapolation instead. Nevertheless, we expect to recover a convergence rate $n = 2$ in both cut-cells and full cells in most flow configurations involving moving rigid embedded boundaries as neighboring emerged cells should not appear simultaneously.

Complete details of this test case can be found here: `http://basilisk.fr/sandbox/ghigo/src/test-heat/neumann3D.c`.

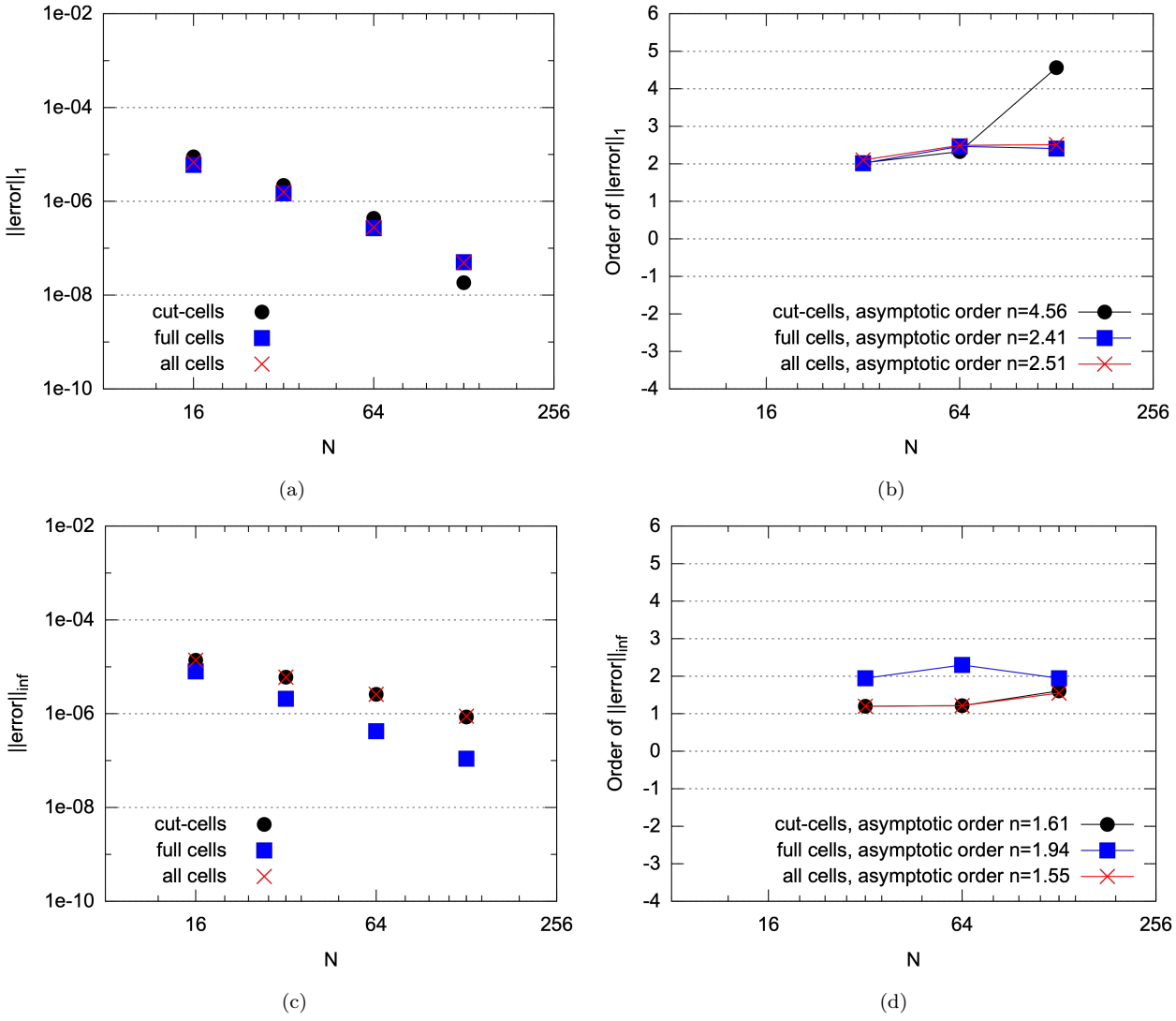


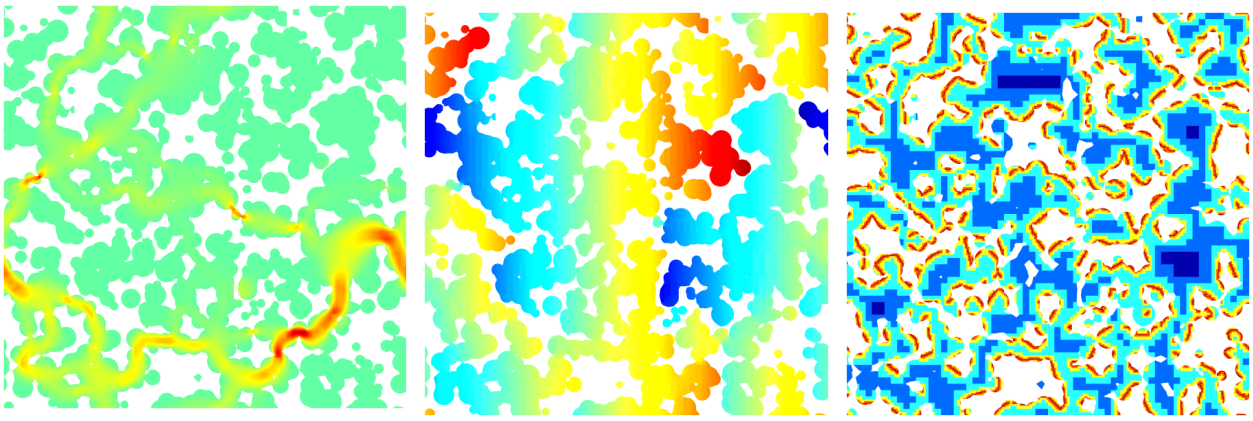
Figure 15: Evolution with the number of cells N of the error between the computed and exact solutions of the heat equation (57) on a uniform Cartesian grid and using Neumann boundary conditions. (a) and (b): $\|\text{error}\|_1$ and the corresponding convergence rate; (c) and (d): $\|\text{error}\|_\infty$ and the corresponding convergence rate. `http://basilisk.fr/sandbox/ghigo/src/test-heat/neumann3D.c`.

6.3. Pressure-driven Stokes flow through a porous medium arbitrarily refined

We now assess the robustness of the multigrid Poisson-Helmholtz solver when dealing with complex embedded boundaries. We consider the 2D periodic porous medium illustrated in

Figure 16 and defined by the union of a random collection of disks. In this porous medium, we solve a Stokes flow driven by the pressure gradient $\mathbf{g} = [1, 0]^\intercal$.

To test the robustness of the treatment of arbitrary embedded boundaries with arbitrary levels of refinement, we discretize the computational domain using the randomly refined quadtree represented in Figure 16c and defined by $N_{\min} = 2$ and $N_{\max} = 32$. Starting from the initial solution $\mathbf{u} = 0$ and $p = 0$, we solve the Stokes equations for $n = 400$ time steps, using the time step $\Delta t = 2 \times 10^{-5}$ to minimize splitting errors. We also set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-3}$.



(a) Velocity $\sqrt{u_x^2 + u_y^2}$. (b) Pressure p. (c) Level of refinement l.

Figure 16: Pressure-driven Stokes flow in a 2D porous medium defined by the union of a random collection of disks and computed on a randomly refined quadtree defined by $N_{\min} = 2$ and $N_{\max} = 32$: (a) norm of the velocity $\sqrt{u_x^2 + u_y^2}$; (b) pressure p and (c) arbitrarily defined levels of refinement. `http://basilisk.fr/sandbox/ghigo/src/test-stokes/porous1.c`.

In Figures 16a and 16b, we observe that the Cartesian grid embedded boundary method handles without difficulty the complex geometry of the porous medium and that the multigrid solver is able to compute in each pore, whether open or closed, a solution for the norm of the velocity $\sqrt{u_x^2 + u_y^2}$ (see Figure 16a) and the pressure p (see Figure 16b). We also note that, as expected, the flow follows a preferred path aligned with the direction of the pressure gradient $\mathbf{g}$.

To further assess the performance of the multigrid solver, we plot in Figure 17 the evolution with the number of time steps n of several quantities related to the performance of the multigrid solver. In particular, we observe that the relative change in velocity $\|u_x^{n+1} - u_x^n\|_\infty / \|u_x^{n+1}\|_1$ (black dots in Figure 17) rapidly converges towards zero, indicating that we obtain a steady solution. This is also highlighted by the near constant values of the 1-norm of the velocity $\|u_x\|_1$ (green lozenge in Figure 17) and the ∞-norm of the pressure $\|p\|_\infty$ (orange plus in Figure 17). Finally, we notice that the ∞-norm of both the residual of the viscous Navier-Stokes equation (36b) $\|\text{res}_u\|_\infty$ (red cross in Figure 17) and of the pressure Poisson-Helmholtz equation (36c) $\Delta t \|\text{res}_p\|_\infty$ (blue square in Figure 17) rapidly converge bellow the tolerance ξ_{mg} of the multigrid solver, indicating its efficiency.

Complete details of this test case can be found here: `http://basilisk.fr/sandbox/ghigo/src/test-stokes/porous1.c`. We have also extended this test case to 3D and obtain

similar performances (results not shown here).

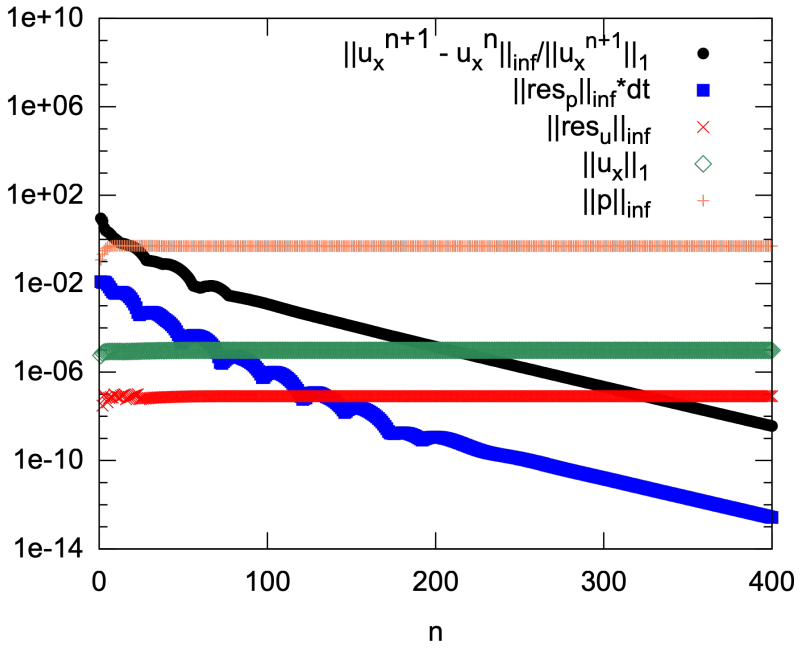


Figure 17: Evolution with the number of time steps n of several quantities related to the performance of the multigrid solver when solving a pressure-driven Stokes flow in a 2D porous medium. In particular, we plot the relative change in velocity $\|u_x^{n+1} - u_x^n\|_\infty / \|u_x^{n+1}\|_1$ (black dot) and the ∞-norm of the residual of the viscous Navier-Stokes equation (36b) $\|\text{res}_u\|_\infty$ (red cross) and of the pressure Poisson-Helmholtz equation (36c) $\Delta t \|\text{res}_p\|_\infty$ (blue square), solved using the multigrid solver. `http://basilisk.fr/sandbox/ghigo/src/test-stokes/porous1.c`.

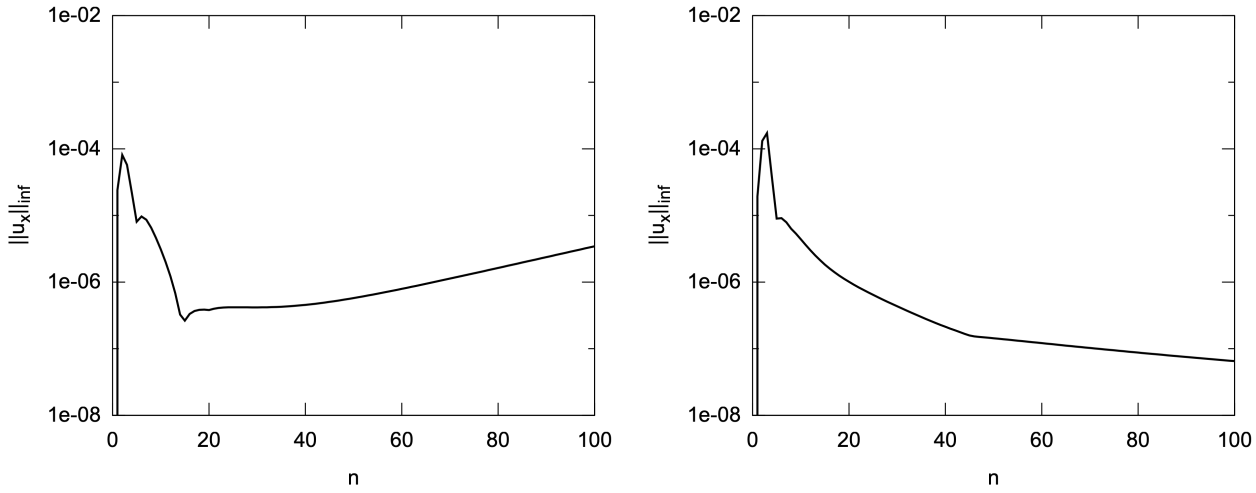
6.4. Instabilities due to "third-order" Dirichlet boundary conditions at low Reynolds number

In Section 3.4.3, we discussed the importance of using the volume-weighted average $\bar{s}_d^f$ (25) when computing the face value s_d^f of a scalar s to prevent the occurrence of an instability when solving the Stokes equations. We highlight this instability here by solving the Stokes equations in a periodic straight channel of length $L_0 = 1$ and width $h = \frac{1}{2}$ in which we let a flow, initialized with a transverse velocity $u_y = 1$ everywhere, return to a rest equilibrium state.

We use a uniform Cartesian grid with $N = 32$ and a time step $\Delta t = 4 \times 10^{-5}$. We set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-7}$. Note that the occurrence of the instability is sensitive to the previously defined parameters, especially the time step Δt.

In Figure 18, we plot the evolution with the number of time steps n of the ∞-norm of the horizontal velocity u_x and consider two cases: (i) in Figure 18a, we use simple averaging to compute face values and observe that an instability appears and that the velocity in the channel diverges after an initial transient decay; (ii) in Figure 18b, we use the volume-weighted average (25) to compute face values and observe that the velocity in the channel remains stable and decays towards zero.

As mentioned in Section 3.4.3, this instability occurs when using an approximate projection method such as system (36), due to feedback between the pressure p and the approximately



(a) Simple average.

(b) Volume-weighted average.

Figure 18: Evolution with the number of time steps n of the ∞-norm of the horizontal velocity $\|u_x\|_\infty$ when solving a Stokes flow in a periodic channel: (a) unstable case where face values are computed using simple averaging and (b) stable case where face values are computed using the volume-weighted average (25). `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/uf.c`.

incompressible cell-centered velocity $\mathbf{u}$. However, the instability is only triggered here because the face gradient (24) used to compute the velocity gradient in equation (36b) behaves in cut-cells as a third-order term when combined with Dirichlet boundary conditions (see Section 6.1). We therefore use the volume-weighted average (25) to reduce the contribution of small cut-cells when computing face values and to dampen velocity perturbations. Note that the instability also disappears if we use Neumann boundary conditions on the embedded boundary or Dirichlet boundary conditions with first- or second-order face gradients (see equations (21) and (22)).

Complete details of this test case can be found here: `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/uf.c`.

6.5. Starting flow past a fixed and moving cylinder at $Re = 1000$

The results presented in Sections 6.1, 6.2, 6.3 and 6.4 show that the multigrid Poisson-Helmholtz solver implemented in *Basilisk* is second-order accurate, robust and stable. We now assess the accuracy of the Bell-Colella-Glaz scheme and the flux redistribution technique presented in Sections 5.7 and 5.8 and used to solve the non-linear advection equation (36a).

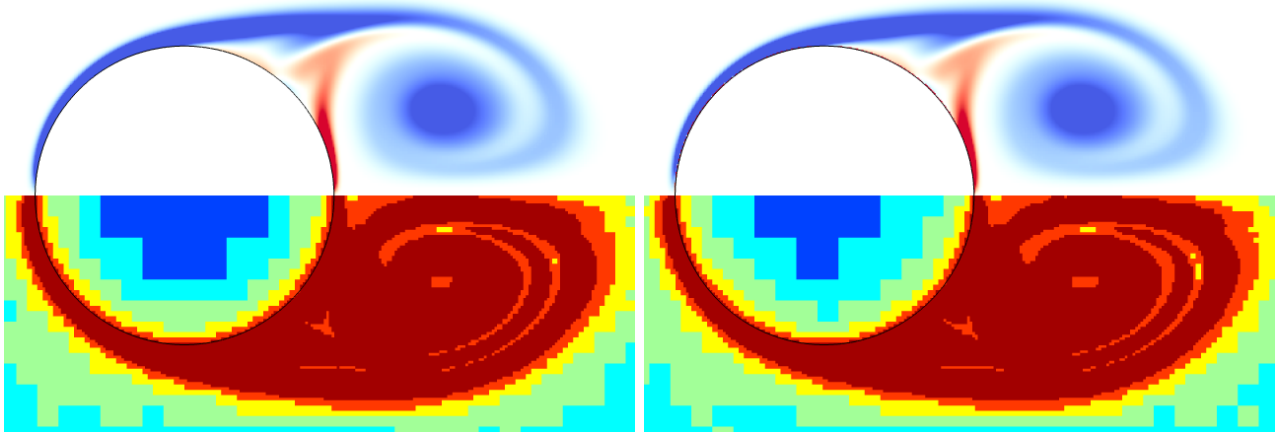
We compute here the highly inertial starting flow around a 2D cylinder at the Reynolds number $Re = \frac{u_{\text{ref}} d}{\nu} = 1000$, where $u_{\text{ref}} = 1$ is the reference velocity and $d = 1$ is diameter of the cylinder. This is a canonical case of complex boundary layer separation [79, 80, 52], where a high spatial resolution is needed to properly resolve the boundary layers around the cylinder.

In the following, we compare two cases which should be equivalent assuming a Galilean invariance of our method:

1. a starting flow passed a fixed cylinder, characterized by the constant far-field velocity $\mathbf{u}_\infty = [u_{\text{ref}}, 0]^\intercal$ and referred to as the *fixed cylinder case*;

2. a cylinder impulsively started in an otherwise quiescent flow, moving at the imposed constant rigid body velocity $\mathbf{u}_\Gamma = [-u_{\text{ref}}, 0]^\intercal$ and referred to as the *moving cylinder case*.

In both cases, we consider a domain of size $L_0 = 18$ and represent only half of the cylinder, initially located in the middle of the bottom boundary of the domain. We therefore impose symmetry boundary conditions on the bottom boundary, a slip boundary condition on the top boundary and an outflow boundary condition on the right boundary. On the left boundary, we impose the inlet velocity $\mathbf{u}_\infty + \mathbf{u}_\Gamma$. Starting from the initial condition $\mathbf{u}(t=0) = \mathbf{u}_\infty + \mathbf{u}_\Gamma$ and $p = 0$, we then solve the Navier-Stokes equations (36) on a quadtree dynamically adapted at each time step based on the velocity $\mathbf{u}$ with an adaptation criteria $\xi_{\text{adapt}}/u_{\text{ref}} = 10^{-3}$. Following [80], we choose the maximum level of refinement $l_{\max} = 12$ (228 pt/d) to obtain a minimum resolution $d/\Delta_{\min} \approx 10\sqrt{Re}$. We also set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-6}$. Finally, the time step is bounded by $\Delta t/(d/u_{\text{ref}}) \leq 10^{-3}$.



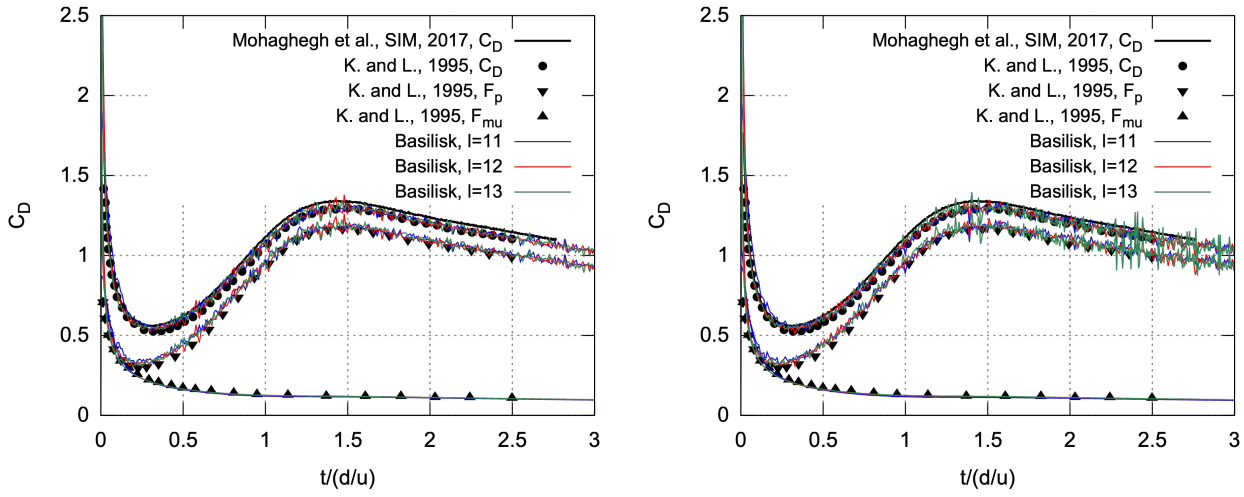
(a) Fixed cylinder case.

(b) Moving cylinder case.

Figure 19: Vorticity ω_z (top half) and the corresponding spatial distribution of the level of refinement (bottom half) at $t/(d/u_{\text{ref}}) = 3$ for the impulsively started flow around a cylinder at $Re = 1000$. `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c` and `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c`.

In Figure 19, we plot at time $t/(d/u_{\text{ref}}) = 3$ for both the fixed cylinder case (see Figure 19a) and the moving cylinder case (see Figure 19b) the vorticity ω_z (top half) as well as the corresponding spatial distribution of the level of refinement l around the cylinder (bottom half). By visually comparing the vorticity structures in each case, we notice that they are almost identical and are in very good qualitative agreement with the vorticity plots displayed in Figure 16 of [79] and Figure 3 of [80]. Furthermore, we observe in both cases that the mesh is refined in the regions of strong vorticity while the core of the cylinder (which does not belong to the computational domain) and the far field regions are represented with coarser cells. This significantly reduces the number of cells required to compute an accurate solution (see note on performance bellow).

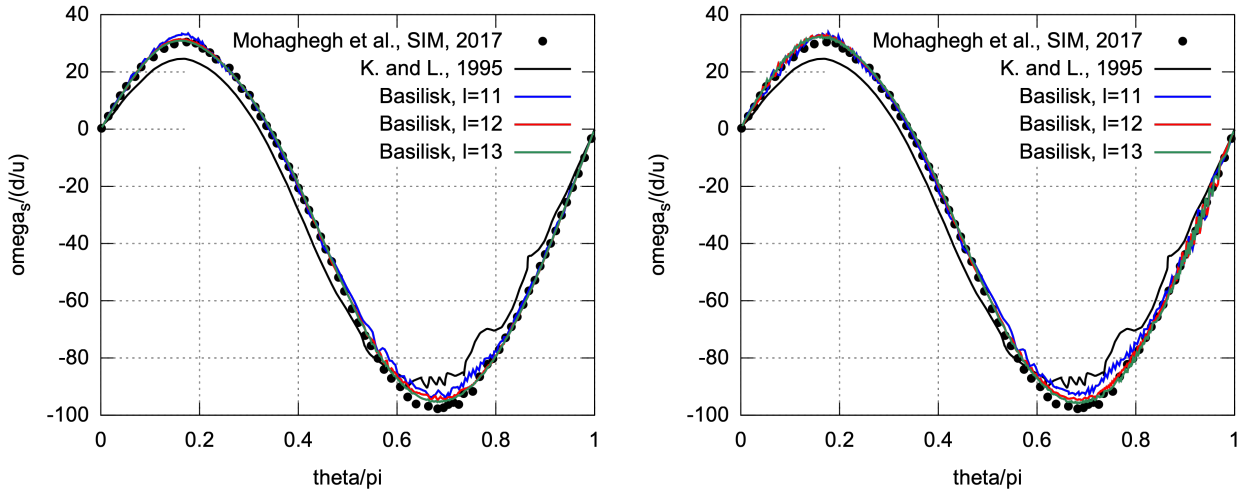
In Figures 20 to 22, we perform a more quantitative comparison with [79, 80] by displaying results computed on three different adaptive quadtrees defined by the following values of the



(a) Fixed cylinder case.

(b) Moving cylinder case.

Figure 20: Time evolution of the hydrodynamic forces acting on the cylinder impulsively started at $Re = 1000$, computed on three different adaptive quadtrees defined by the following values of the maximum level of refinement $l_{\max} = \{11\ (114\,\mathrm{pt/d})\,,\ 12\ (228\,\mathrm{pt/d})\,,\ 13\ (455\,\mathrm{pt/d})\}$. `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c` and `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c`.

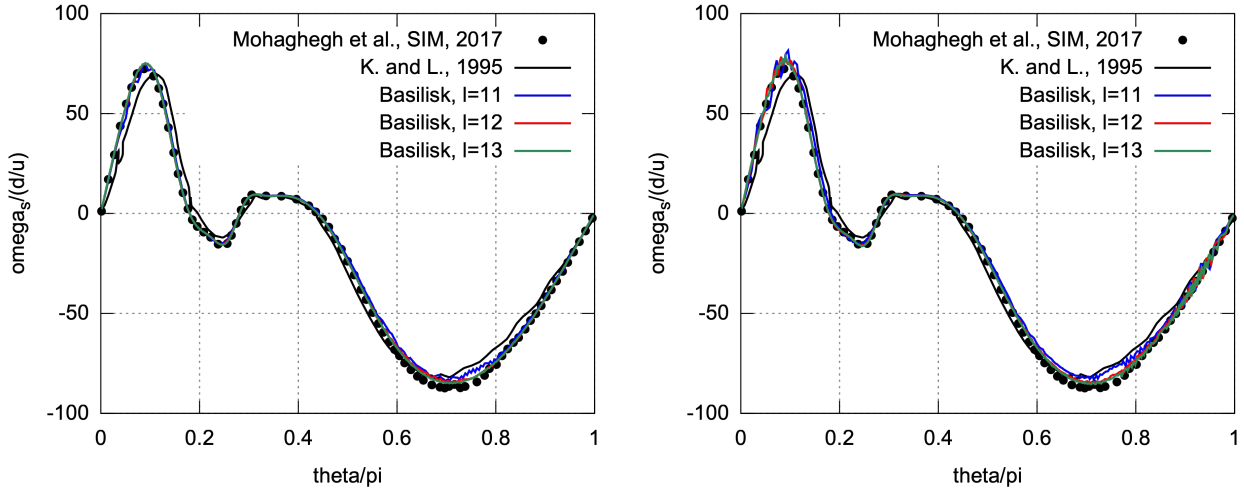


(a) Fixed cylinder case.

(b) Moving cylinder case.

Figure 21: Distribution of the vorticity ω_z around the cylinder impulsively started at $Re = 1000$, computed at time $t/\left(d/u_{\mathrm{ref}}\right) = 0.5$ on three different adaptive quadtrees defined by the following values of the maximum level of refinement $l_{\max} = \{11\ (114\,\mathrm{pt/d})\,,\ 12\ (228\,\mathrm{pt/d})\,,\ 13\ (455\,\mathrm{pt/d})\}$. `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c` and `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c`.

maximum level of refinement $l_{\max} = \{11\ (114\,\mathrm{pt/d})\,,\ 12\ (228\,\mathrm{pt/d})\,,\ 13\ (455\,\mathrm{pt/d})\}$. In Figures 20a and 20b, we plot respectively for the fixed and moving cylinder cases the time evolution of the streamwise component of the pressure and viscous forces $F_{p,x}$ and $F_{\mu,x}$ acting on the



(a) Fixed cylinder case.

(b) Moving cylinder case.

Figure 22: Distribution of the vorticity ω_z around the cylinder impulsively started at $Re = 1000$, computed at time $t/(d/u_{\text{ref}}) = 2.5$ on three different adaptive quadtrees defined by the following values of the maximum level of refinement $l_{\max} = \{11\ (114\,\text{pt/d}),\ 12\ (228\,\text{pt/d}),\ 13\ (455\,\text{pt/d})\}$. `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c` and `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c`.

cylinder. We also plot in the same figure the time evolution of the drag coefficient C_D defined as:

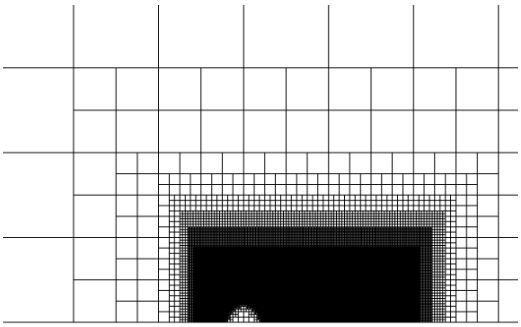
$$C_D = \frac{F_{p,x} + F_{\mu.x}}{\frac{1}{2}\rho u_{\text{ref}} S_{\text{ref}}}, \qquad \text{with} \qquad S_{\text{ref}} = \frac{d}{2}. \tag{60}$$

In Figures 21a and 21b and Figures 22a and 22b, we plot respectively for the fixed and moving cylinder cases the distribution of the vorticity ω_z on the surface of the cylinder obtained at times $t/(d/u_{\text{ref}}) = \{0.5,\ 2.5\}$. For each of these quantities, we compare our results to those presented in [79, 80] and observe that for both the fixed and moving cylinder cases the computed results converge towards the reference data as we increase the value of the maximum level of refinement $l_{\max}$. In particular, the time evolution of the viscous force $F_{\mu,x}$ in Figures 20a and 20b perfectly matches the reference data for all considered values of $l_{\max}$. We note however that the time evolution of the pressure force $F_{p,x}$ in Figures 20a and 20b is noisier than the reference data and that the noise is not significantly reduced when we increase the value of $l_{\max}$.

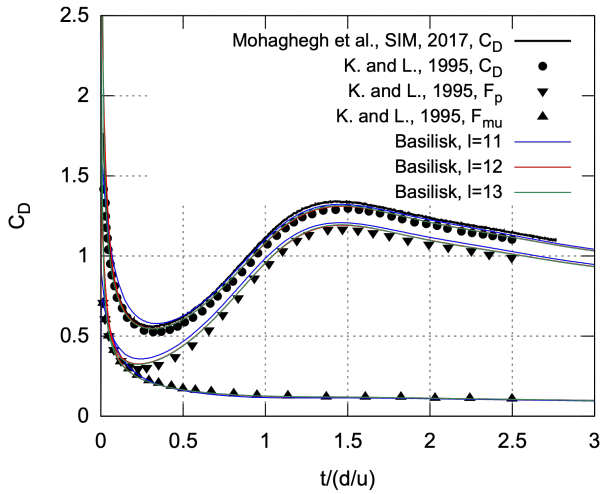
Indeed, as the cell-centered velocity $\mathbf{u}$ is not exactly incompressible, the pressure p feels through the term $\nabla \cdot \mathbf{u}^{\star\star}$ in equation (36c) the history of the divergence of the velocity. And in this case, the history of the divergence contains noise induced by the dynamic mesh adaptation at every time step.

To corroborate this explanation, we plot in Figure 23 the time evolution of the hydrodynamic forces obtained for parameters identical to those used to obtain the results in Figure 20 but computed on the static quadtree presented in Figure 23a. We observe that in this case, for both the fixed (see Figure 23b) and moving (see Figure 23c) cylinder cases, the time evolution of the pressure force $F_{p,x}$ no longer contains any noise. This is a clear indication that the adaptive mesh refinement along with the approximate projection method are responsible for the noise observed in the pressure force $F_{p,x}$ in Figure 20.

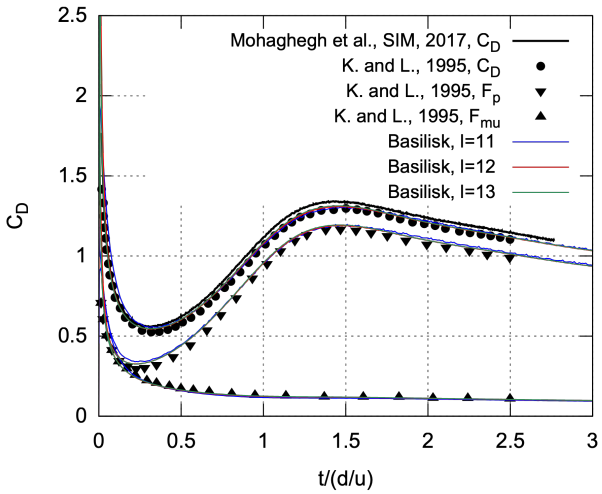
Complete details of these two test cases can be found here: `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c` and `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c`.



(a) Static quadtree with $l_{\max} = 12$.



(b) Fixed cylinder case.



(c) Moving cylinder case.

Figure 23: Time evolution of the hydrodynamic forces acting on the cylinder impulsively started at $Re = 1000$, computed on three different static quadtrees defined by the following values of the maximum level of refinement $l_{\max} = \{11\ (114\,\mathrm{pt/d})\,,\ 12\ (228\,\mathrm{pt/d})\,,\ 13\ (455\,\mathrm{pt/d})\}$ and illustrated in (a) for $l_{\max} = 12\ (228\,\mathrm{pt/d})$: (b) fixed cylinder case, (c) moving cylinder case. `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c` and `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c`.

A note on performance. Performance, ideally measured by the wall-clock runtime for a given accuracy, is a key parameter in the development and evaluation of numerical methods, and this is especially true for adaptive mesh refinement techniques. We therefore compare here the performances of the Cartesian grid embedded boundary method for both the fixed and moving cylinder cases on three different grids: (i) a uniform Cartesian grid, (ii) an adaptive quadtree and (iii) a static quadtree as described above.

Indeed, a remarkable feature of *Basilisk* is that the definition of a cell on both a uniform grid and a tree grid is identical. This is achieved in *Basilisk* by separating the low-level details

of the different grid implementations (e.g. memory layout, grid traversal strategies) from the numerical scheme itself. Therefore, *Basilisk* provides efficient Cartesian grid and quadtree implementations and a comparison of the performances between both grids is therefore warranted.

Grid	Uniform Cartesian grid		Adaptive quadtree		Static quadtree	
Case	Fixed	Moving	Fixed	Moving	Fixed	Moving
Wall-clock time	7685.79	14005.9	30.58	66.17	320.97	620.2
Time steps	616	818	616	907	617	899
Cells	4194304	4194304	10576	11077	78430	83194
Speed	3.36×10^5	2.69×10^5	1.33×10^5	1.03×10^5	1.51×10^5	1.21×10^5

Table 1: Computing times and speeds for the starting flow past a cylinder at $Re = 1000$, computed on one Intel i5 processor until $t/(d/u_{\text{ref}}) = 2.5$ using a maximum level of refinement $l_{\max} = 11$. The wall-clock time is given in seconds (s) and the speed in points·steps/s. `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c` and `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c`.

In Table 1, we summarize the performances of the fixed and moving embedded boundary algorithms. In both cases, we run the simulation until $t/(d/u_{\text{ref}}) = 2.5$ using a maximum level of refinement $l_{\max} = 11$ on a single Intel i5 processor. We notice that, as expected, the fixed embedded boundary algorithm is faster than its moving counterpart by a factor of ~ 1.3. However, the computational time for the moving embedded boundary algorithm is twice that of the fixed one due to additional constraints on the time step (CFL $= 0.5$) which increase the total number of time steps. Finally, we observe that the gain in number of grid points (i.e. memory) obtained with adaptivity is approximately a factor of 400, while the gain in computing time is roughly a factor of 200.

6.6. Oscillating sphere in a quiescent flow

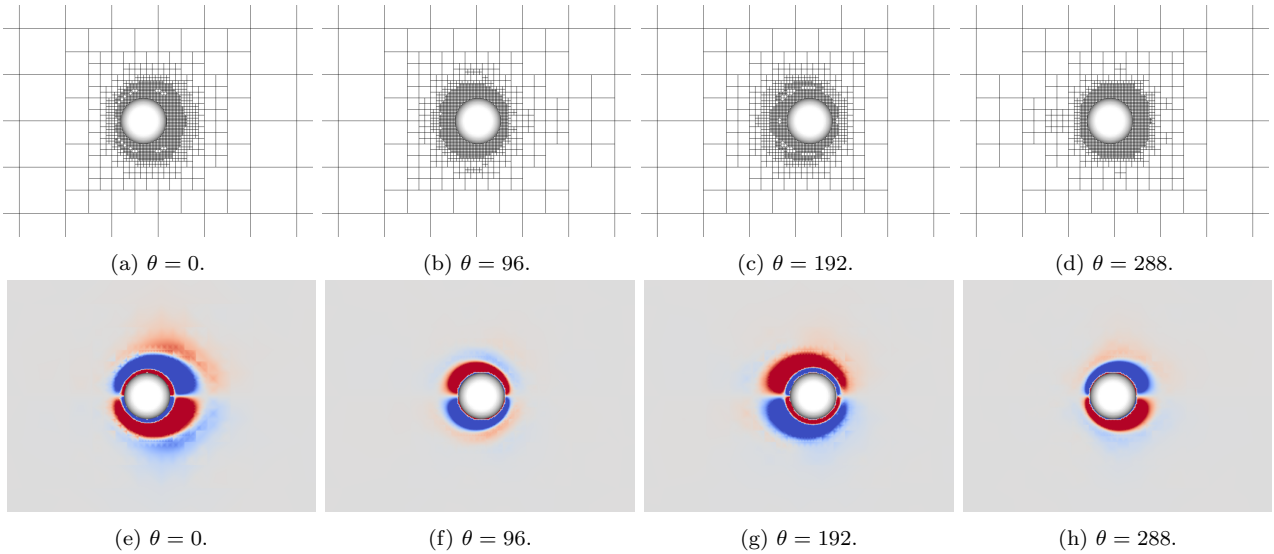
We now reproduce a 3D test case taken from [81, 82, 49] and compute the inertial flow induced by the forced inline sinusoidal oscillation of a sphere in an otherwise quiescent fluid. The oscillating motion of the sphere is characterized by a Reynolds number $Re = \frac{u_{\text{ref}} d}{\nu} = 40$ and a Strouhal number $St = \frac{\omega d}{u_{\text{ref}}} = 3.2$, where $u_{\text{ref}} = 1$ is the maximum velocity of the sphere, $d = 1$ is the diameter of the sphere and ω is the oscillation frequency.

We consider a domain of size $L_0 = 16$, where we impose slip boundary conditions on all boundaries and embed the sphere in the center of the domain. Starting from the initial condition $\mathbf{u}(t = 0) = \mathbf{0}$ and $p = 0$, we impose the following forced horizontal motion to the sphere:

$$\mathbf{u}_\Gamma = u_{\text{ref}} \sin(\omega t)\, \mathbf{e}_{\mathbf{x}}. \tag{61}$$

We then solve the Navier-Stokes equations (36) until a periodic state is reached. We use an octree dynamically adapted at each time step based on the velocity $\mathbf{u}$ with an adaptation criteria $\xi_{\text{adapt}}/u_{\text{ref}} = 10^{-2}$. We also set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-6}$. Finally, the time step is bounded by $\Delta t/(d/u_{\text{ref}}) \leq 10^{-2}$. We observe that in this case a periodic state is reached after one oscillation period $T = \omega/(2\pi)$.

In Figure 24, we display snapshots of the mesh and vorticity in the xy-plane, taken at the following phase angles $\theta = \{0, 96, 192, 288\}$ during the second oscillation period and computed



(a) $\theta = 0$. (b) $\theta = 96$. (c) $\theta = 192$. (d) $\theta = 288$.

(e) $\theta = 0$. (f) $\theta = 96$. (g) $\theta = 192$. (h) $\theta = 288$.

Figure 24: Snapshots of the mesh and the vorticity in the xy-plane, taken at the phase angles $\theta = \{0, 96, 192, 288\}$ during the second oscillation period of a sphere oscillating in a quiescent flow at $Re = 40$ and $St = 3.2$ on an adaptive octree defined by the following value of the maximum level of refinement $l_{\max} = 9$ ($32\,\mathrm{pt/d}$). `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/sphere-oscillating.c`.

on the octree defined by the following value of the maximum of level of refinement $l_{\max} = 9$ ($32\,\mathrm{pt/d}$). We observe that the mesh dynamically follows the movement of the sphere and captures the resulting vorticity structures.

We then plot in Figure 25 the time evolution during the second oscillation period of the drag coefficient C_D defined by equation (60), with $S_{\mathrm{ref}} = \pi\frac{d^2}{4}$. The results are computed on three different octrees defined by the following values of the maximum level of refinement $l_{\max} = \{8\ (16\,\mathrm{pt/d})\,,\ 9\ (32\,\mathrm{pt/d})\,,\ 10\ (64\,\mathrm{pt/d})\}$. For all values of $l_{\max}$, we obtain a very good agreement with the reference data from [49], computed using $50\,\mathrm{pt/d}$, confirming the high accuracy of our method.

Complete details of this test case can be found here: `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/sphere-oscillating.c`.

6.7. Sphere of near-unity density ratio settling in a closed box

Finally, we conclude the validation of the Cartesian grid embedded boundary method implemented in *Basilisk* by solving the coupled system of equations (1)–(2) and (4)–(5) describing the motion of a freely moving rigid body.

We first investigate the settling at small to moderate Reynolds numbers of a sphere of near-unity density ratio $\rho_\Gamma/\rho \approx 1$ in a closed box under the action of the gravity $\mathbf{g} = [0, -9.81, 0]^\intercal\ \mathrm{m/s^2}$. This test case is inspired by the experimental and numerical work of [83], where the authors characterized the flow using the Reynolds number $1.5 \leq Re = \frac{u_{\mathrm{ref}} d}{\nu} \leq 31.9$ and the particle Stokes number $0.2 \leq St = \frac{1}{9} Re \frac{\rho_\Gamma}{\rho} \leq 4$, with u_{ref} a reference settling velocity and $d = 0.015\,\mathrm{m}$ the sphere diameter. This test case has since then been used on numerous occasions as a validation case for fluid-particle coupling methods [84, 85, 86, 16] at small to moderate Reynolds numbers.

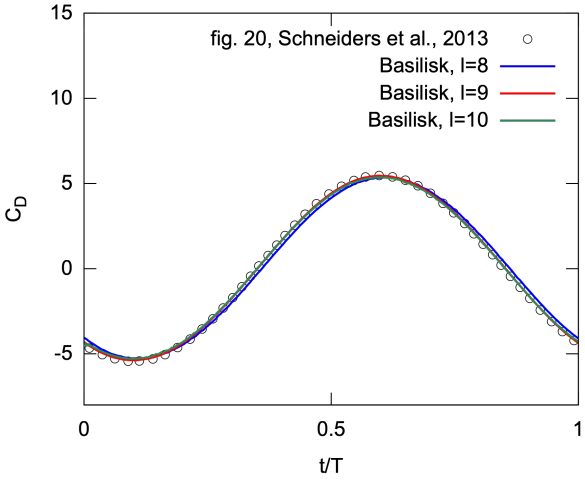


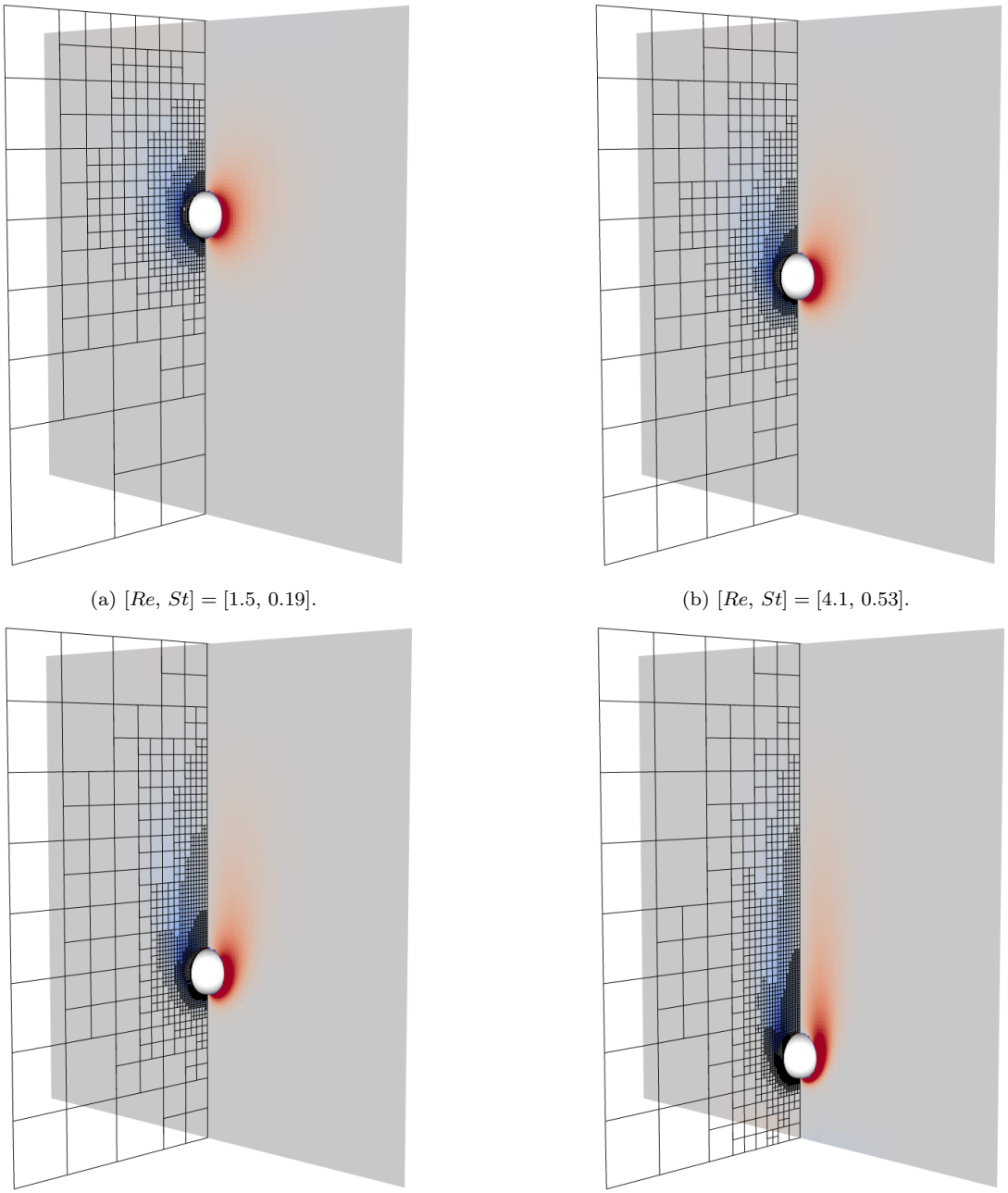
Figure 25: Time evolution during the second oscillation period of the drag coefficient C_D computed around a sphere oscillating in quiescent flow at $Re = 40$ and $St = 3.2$ on three different adaptive octrees defined by the following values of the maximum level of refinement $l_{\max} = \{8\ (16\,\text{pt/d})\,,\ 9\ (32\,\text{pt/d})\,,\ 10\ (64\,\text{pt/d})\}$. `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/sphere-oscillating.c`.

We consider a domain of size $L_0 = 0.16\,\text{m}$, where we impose no-slip boundary conditions on all boundaries and embed the sphere at a distance of $0.04\,\text{m}$ between the bottom of the sphere and the top boundary. Then, starting from the initial condition $\mathbf{u}\,(t=0) = \mathbf{0}$ and $p = 0$, we solve the coupled system of equations (1)–(2) and (4)–(5) on an octree, dynamically adapted at each time step based on the velocity $\mathbf{u}$ with an adaptation criteria $\xi_{\text{adapt}}/u_{\text{ref}} = 10^{-2}$. We also set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-4}$. Finally, the time step is bounded by $\Delta t/\left(d/u_{\text{ref}}\right) \leq 10^{-3}$.

The range of Reynolds numbers considered is $Re \in [1.5,\ 31.9]$, which we obtain by varying the reference velocity u_{ref} and the viscosity ν. Similarly, the range of Stokes numbers considered is $St \in [0.19,\ 4.13]$, which we obtain by varying the density ratio ρ_Γ/ρ. For these low to moderate Reynolds number flows, the particle experiences a period of nearly steady fall after an initial acceleration. As it approaches the bottom of the box, the particle decelerates towards zero and we stop the simulation when one cell is left between the sphere and the bottom wall. As an illustration, we display in Figure 26 snapshots at $t/\left(d/u_{\text{ref}}\right) = 1$ of the vorticity in the xy-plane obtained for different combinations of the Reynolds and Stokes numbers.

In Figure 27, we plot the time evolution of the position $x_{\Gamma,y}/d$ (see Figure 27a) and the settling velocity $u_{\Gamma,y}/u_{\text{ref}}$ (see Figure 27b) of the sphere for different combinations of the Reynolds and Stokes numbers: $[Re,\ St] = \{[1.5,\ 0.19]\,,\ [4.1,\ 0.53]\,,\ [11.6,\ 1.5]\,,\ [31.9,\ 4.13]\}$. To assess the accuracy of the method, the results are obtained on three different octrees defined by the following values of the maximum level of refinement $l_{\max} = \{7\ (12\,\text{pt/d})\,,\ 8\ (24\,\text{pt/d})\,,\ 9\ (48\,\text{pt/d})\}$. For each case, we compare our results to those of [83] and observe a very good agreement with the reference experimental data, even when using $l_{\max} = 7\ (12\,\text{pt/d})$. However, values of $l_{\max} > 7$ are required to obtained converged results for moderate Reynolds number flows.

The Cartesian grid embedded boundary method along with the first-order in time explicit weak fluid-solid coupling strategy are therefore able to accurately capture the two-way interactions between the fluid and the particle with a limited number of grid points.



(a) $[Re,\, St] = [1.5,\, 0.19]$.

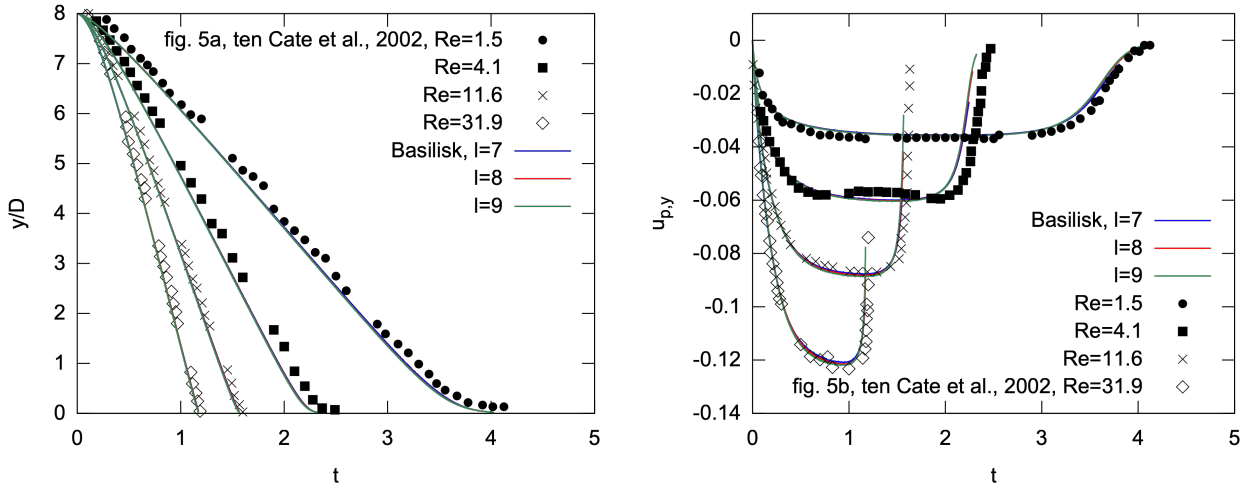
(b) $[Re,\, St] = [4.1,\, 0.53]$.

(c) $[Re,\, St] = [11.6,\, 1.5]$.

(d) $[Re,\, St] = [31.9,\, 4.13]$.

Figure 26: Snapshots at time $t/\left(d/u_{\mathrm{ref}}\right) = 1$ of the vorticity in the xy-plane around a sphere of near-unity density ratio settling in a closed box for different combinations of the Reynolds and Stokes numbers $[Re,\, St]$, computed on an octree grid defined by the following value of the maximum level of refinement $l_{\mathrm{max}} = 9$ $(48\,\mathrm{pt/d})$. `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling.c`.

Complete details of this test case can be found here: `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling.c`.



(a) Position $x_{\Gamma,y}/d$.

(b) Settling velocity $u_{\Gamma,y}/u_{\mathrm{ref}}$.

Figure 27: Time evolution of (a) the position $x_{\Gamma,y}/d$ and (b) the settling velocity $u_{\Gamma,y}/u_{\mathrm{ref}}$ of a sphere of near-unity density ratio settling in a closed box, for different combinations of the Reynolds and Stokes numbers: $[Re,\, St] = \{[1.5,\, 0.19]\,,\, [4.1,\, 0.53]\,,\, [11.6,\, 1.5]\,,\, [31.9,\, 4.13]\}$, obtained on three different octrees defined by the following values of the maximum level of refinement $l_{\mathrm{max}} = \{7\ (12\,\mathrm{pt/d})\,,\, 8\ (24\,\mathrm{pt/d})\,,\, 9\ (48\,\mathrm{pt/d})\}$. `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling.c`.

6.8. Heavy sphere settling in a large closed box

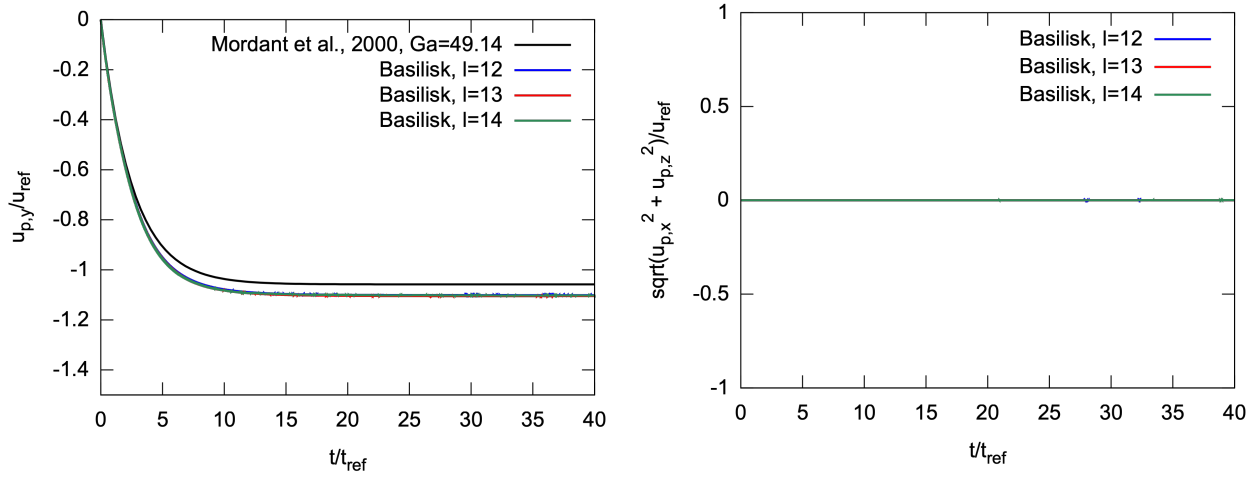
In this final validation test case, we investigate the settling at large Reynolds or Galileo numbers of a heavy sphere under the action of the gravity $\mathbf{g} = [0,\, -9.81,\, 0]^{\intercal}\ \mathrm{m/s^2}$. This test case is inspired by the experimental work of [87], where the authors studied the motion of spherical beads in a large container filled with water. More particularly, we reproduce here cases 1, 2 and 4 from [87].

Following [5], we use a sphere of diameter $d = 2/12\,\mathrm{m}$, define a reference velocity $u_{\mathrm{ref}} = \sqrt{|g_y|d}$ and consider the following three combinations of density ratio ρ_Γ/ρ and Galileo number $Ga = \sqrt{|1 - \frac{\rho_\Gamma}{\rho}|d^3|g_y|}/\nu$: $[\rho_\Gamma/\rho,\, Ga] = \{[2.56,\, 49.14]$ (case 1)$\,,\, [2.56,\, 255.35]$ (case 2)$\,,\, [7.71,\, 206.27]$ (case 4$]\}$ which we obtain by varying the viscosity ν. Unfortunately, it is not possible to deduce from the experimental results of [87] to which sphere trajectory regime these three cases belong to. Indeed, the settling velocities presented in [87] have been averaged over multiple realizations of the same experiment. However, according to the map of regimes of sphere trajectories in the $[\rho_\Gamma/\rho,\, Ga]$ plane obtained numerically in [88] and experimentally in [89], we expect that cases 1, 2 and 4 should display different sphere trajectory regimes.

In case 1, the Galileo number is low enough that the sphere should settle vertically with a steady axisymmetric wake. In case 4, as the Galileo number increases, the sphere should shed a single-sided chain of vortex loops, producing an unsteady side force that causes the sphere to follow an unsteady oblique trajectory. In [89], the authors qualify this sphere trajectory regime as the oblique trajectory regime. However, in [88], the authors further distinguish between the steady, low frequency oscillating and high frequency oscillating oblique trajectory regimes. For the combination of density ratio and Galileo number considered in case 4, the sphere should follow, according to [88], an oblique and high frequency oscillating trajectory.

Finally, in case 2, the Galileo number is even larger and the sphere should follow, according to [88], a chaotic trajectory. Note however that for the Galileo number considered in case 2, the sphere trajectory regime is not described in [89] as a chaotic trajectory regime but rather as an intermittent oblique trajectory regime. Indeed, in this regime the authors still observe in the wake structure some vortex loops consistent with an oblique trajectory.

Instead of defining a tri-periodic domain as in [5], we take full advantage of the mesh adaptation capabilities of the Cartesian grid embedded boundary method and define a very large box of size $L_0 = 32\,\mathrm{m}$ as in the experiments of [87]. We therefore do not have to worry about the particle being affected by the remnants of its periodic wake. We impose no-slip boundary conditions on all boundaries and embed the sphere at a distance $5d$ from the top boundary. Then, starting from an initial condition $\mathbf{u}(t=0) = \mathbf{0}$ and $p = 0$, we solve the coupled system of equations (1)–(2) and (4)–(5) on an octree, dynamically adapted at each time step based on the velocity $\mathbf{u}$ with an adaptation criteria $\xi_{\mathrm{adapt}}/u_{\mathrm{ref}} = 5 \times 10^{-3}$. We also set the tolerance of the multigrid solver to $\xi_{\mathrm{mg}} = 10^{-4}$. Finally, the time step is bounded by $\Delta t/\left(d/u_{\mathrm{ref}}\right) \leq 10^{-3}$.



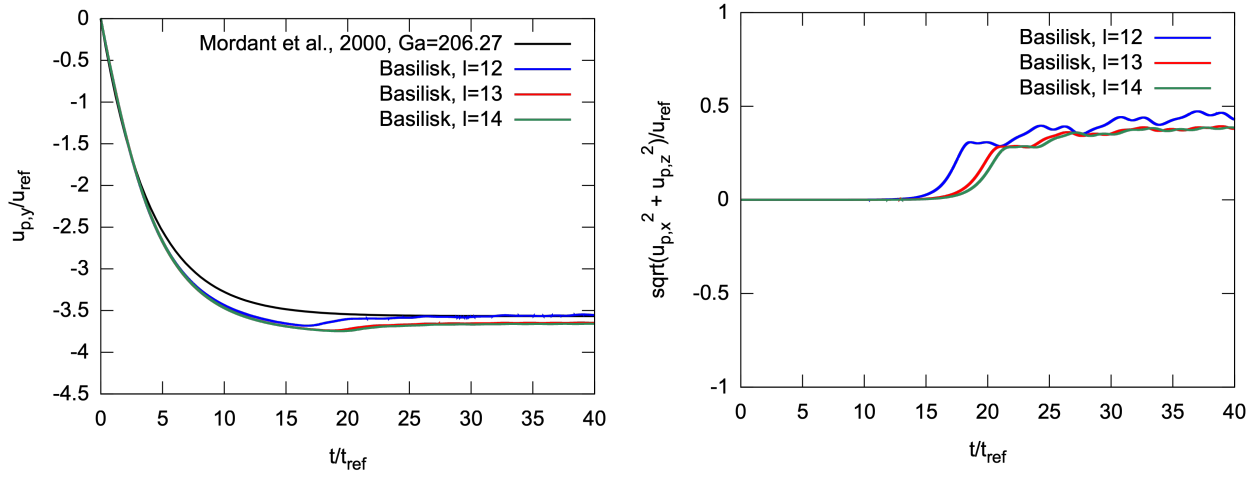
(a) Vertical settling velocity $u_{\Gamma,y}/u_{\mathrm{ref}}$.

(b) Norm of the lateral velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{\mathrm{ref}}$.

Figure 28: Time evolution of (a) the vertical settling velocity $u_{\Gamma,y}/u_{\mathrm{ref}}$ and (b) the norm of the lateral velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{\mathrm{ref}}$ of a heavy sphere settling in a large domain for the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho,\, Ga] = [2.56,\, 49.14]$ (case 1 of [87]), obtained on three different octrees defined by the following values of the maximum level of refinement $l_{\max} = \{12\ (21\,\mathrm{pt/d})\,,\ 13\ (42\,\mathrm{pt/d})\,,\ 14\ (85\,\mathrm{pt/d})\}$. `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling-large-domain.c`.

In Figure 28, we first present the results we obtain for case 1 of [87] using the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho,\, Ga] = [2.56,\, 49.14]$. We plot both the time evolution of the vertical settling velocity $u_{\Gamma,y}/u_{\mathrm{ref}}$ of the sphere as well as the norm of its lateral velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{\mathrm{ref}}$. To assess the accuracy of the method, the results are obtained on three different octrees defined by the following values of the maximum level of refinement $l_{\max} = \{12\ (21\,\mathrm{pt/d})\,,\ 13\ (42\,\mathrm{pt/d})\,,\ 14\ (85\,\mathrm{pt/d})\}$. After a transient acceleration phase, we observe that we recover the expected steady vertical trajectory [89], characterized by

a constant vertical settling velocity and a zero lateral velocity. Furthermore, the final steady vertical settling velocity differs by less than 5% from the experimental results of [87].



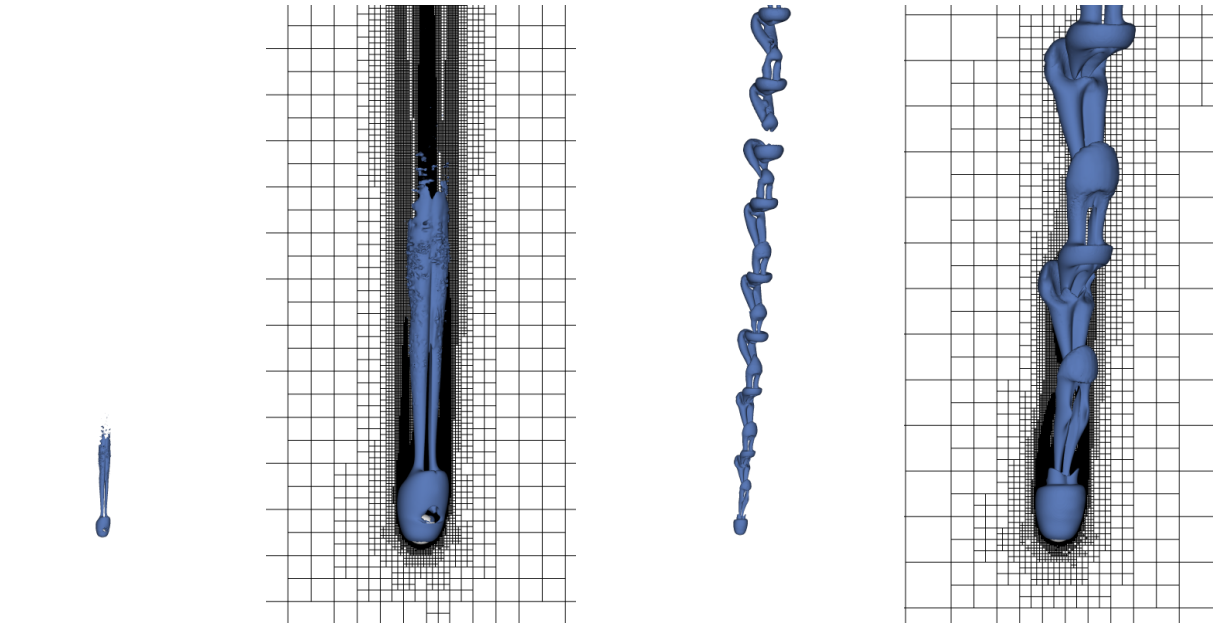
(a) Vertical settling velocity $u_{\Gamma,y}/u_{\text{ref}}$.

(b) Norm of the lateral velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{\text{ref}}$.

Figure 29: Time evolution of (a) the vertical settling velocity $u_{\Gamma,y}/u_{\text{ref}}$ and (b) the norm of the lateral velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{\text{ref}}$ of a heavy sphere settling in a large domain for the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho,\, Ga] = [7.71,\, 206.27]$ (case 4 of [87]), obtained on three different octrees defined by the following values of the maximum level of refinement $l_{\max} = \{12\ (21\,\text{pt/d})\,,\ 13\ (42\,\text{pt/d})\,,\ 14\ (85\,\text{pt/d})\}$. `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling-large-domain.c`.

In Figure 29, we present the time evolution of the same quantities obtained for case 4 of [87] using the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho,\, Ga] = [7.71,\, 206.27]$. We observe that the sphere follows the expected oblique and high frequency oscillating trajectory [88]. Indeed, the computed lateral velocity in Figure 29b oscillates with a high frequency around an average value $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{\text{ref}} \approx 0.38$. For comparison, in [90], the authors predicted, using a Lattice-Boltzmann method, a steady lateral velocity $u_{pH}/u_g \approx 0.1$ for the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho,\, Ga] = [1.5,\, 178.46]$, with $u_g = u_{\text{ref}}\sqrt{|\rho_\Gamma/\rho - 1|}$. Therefore, rescaling our results by $\sqrt{|\rho_\Gamma/\rho - 1|}$, we obtain the average horizontal velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_g \approx 0.14$, which is higher than the value obtained in [90] as we use a higher Galileo number. Furthermore, the final value of the vertical settling velocity in Figure 29a matches the one obtained in [87]. Note however that since the Galileo number is larger than in case 1, the results are converged only for $l_{\max} \geq 13$ (42 pt/d). Finally, we confirm that the sphere follows an oblique and high frequency oscillating trajectory by displaying in Figure 30 the iso-surfaces $\lambda_2 = -0.05$ at times $t/(d/u_{\text{ref}}) = \{20,\, 40\}$. Indeed, we observe the vortex structures characteristic of this regime [89, 88], with a double threaded vortical structure near the sphere and a train of one-sided vortex loops further downstream.

Finally, we plot in Figure 31 similar results obtained for case 2 of [87] using the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho,\, Ga] = [2.56,\, 255.35]$. We first observe that the final value of the vertical settling velocity matches the one obtained in [87] for $l_{\max} \geq 13$ (42 pt/d). Furthermore, we notice that sphere no longer follows an oblique and



(a) $t/\left(d/u_{\mathrm{ref}}\right) = 20$. (b) Zoom at $t/\left(d/u_{\mathrm{ref}}\right) = 20$. (c) $t/\left(d/u_{\mathrm{ref}}\right) = 40$. (d) Zoom at $t/\left(d/u_{\mathrm{ref}}\right) = 40$.

Figure 30: Snapshots at times $t/\left(d/u_{\mathrm{ref}}\right) = \{20, 40\}$ of the iso-surface $\lambda_2 = -0.05$ around of a heavy sphere settling in a large domain for the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho,\, Ga] = [7.71,\, 206.27]$ (case 4 of [87]), obtained on an octree defined by the following value of the maximum level of refinement $l_{\max} = 13$ (42 pt/d). `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling-large-domain.c`.

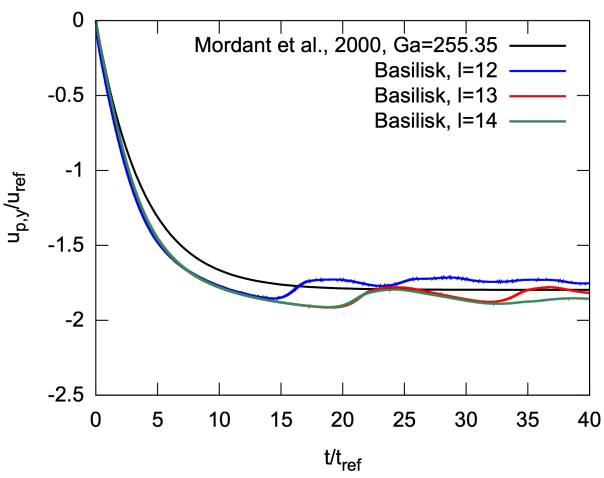
high frequency oscillating trajectory as in case 4 but rather an intermittent oblique trajectory as described in [89]. Indeed, in Figure 32, we display the iso-surfaces $\lambda_2 = -0.05$ at times $t/(d/u_{\mathrm{ref}}) = \{30, 40\}$ and observe again a double threaded vortical structure near the sphere and one-sided vortex loops downstream of the sphere. However, contrary to the vortex structures presented for case 4 in Figure 32, the vortex loops evolve here into vortex rings that are responsible for the intermittency in the sphere's oblique trajectory.

These results therefore confirm the ability of the Cartesian grid embedded boundary method and the first-order in time explicit weak coupling strategy to reproduce for a considerable range of Galileo numbers the dynamic behavior of a settling particle under the action of gravity in a very large domain.

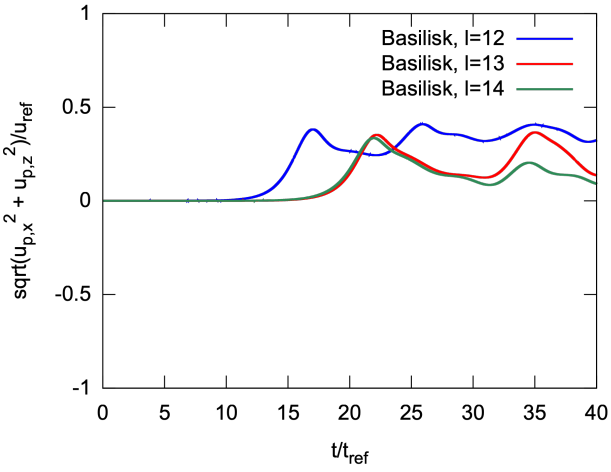
Complete details of this test case can be found here: `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling-large-domain.c`.

A note on performance. We compare here the performances of the Cartesian grid embedded boundary method and the first-order in time explicit weak fluid-solid coupling strategy we use to compute cases 1, 2 and 4 from [87] and summarize the results in Table 2.

In all three cases, we run the simulation, without having to restart, until $t/(d/u_{\mathrm{ref}}) = 40$ using a maximum level of refinement $l_{\max} = 12$ on 48 Intel Platinum processors. Note here that number of processors chosen is not optimal in terms of memory management and could have been roughly reduced by a factor of 5. We notice that, as expected, the computational speeds are similar in all three cases. However, we observe that the dynamic mesh adaptation of the



(a) Vertical settling velocity $u_{\Gamma,y}/u_{\text{ref}}$.



(b) Norm of the lateral velocity $\sqrt{u^2_{\Gamma,x}+u^2_{\Gamma,z}}/u_{\text{ref}}$.

Figure 31: Time evolution of (a) the vertical settling velocity $u_{\Gamma,y}/u_{\text{ref}}$ and (b) the norm of the lateral velocity $\sqrt{u^2_{\Gamma,x}+u^2_{\Gamma,z}}/u_{\text{ref}}$ of a heavy sphere settling in a large domain for the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho,\, Ga] = [2.56,\, 255.35]$ (case 2 of [87]), obtained on three different octrees defined by the following values of the maximum level of refinement $l_{\max} = \{12\ (21\,\text{pt/d})\,,\ 13\ (42\,\text{pt/d})\,,\ 14\ (85\,\text{pt/d})\}$. `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling-large-domain.c`.

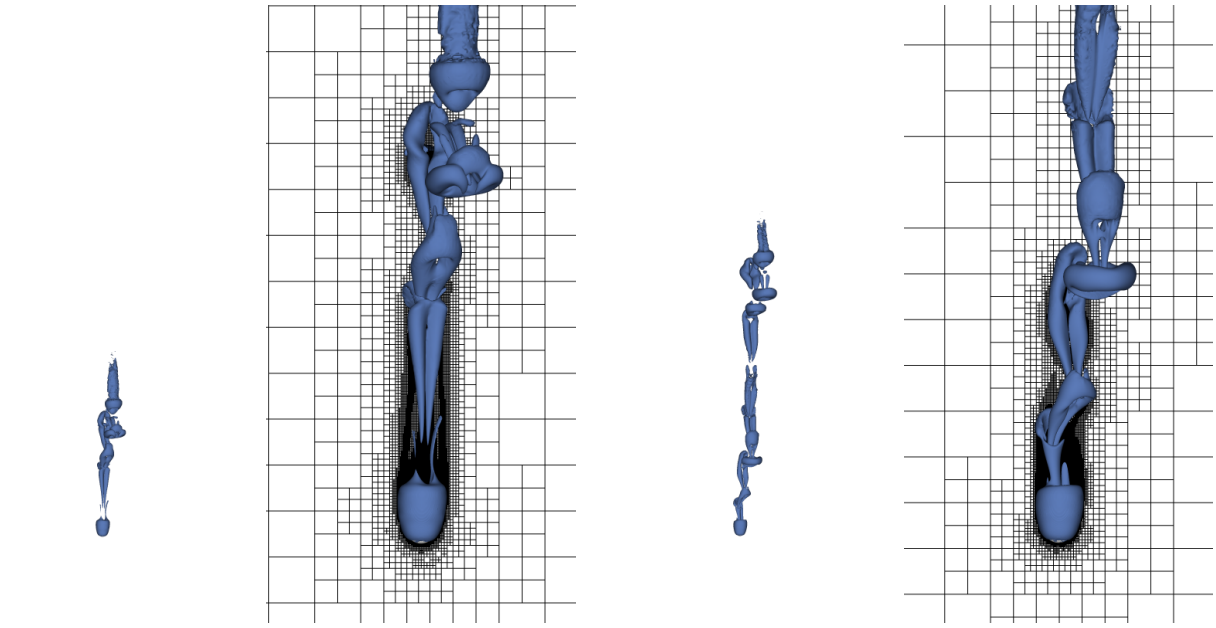
Case from [87]	Case 1: $[\rho_\Gamma/\rho,\, Ga] = [2.56,\, 49.14]$	Case 2: $[\rho_\Gamma/\rho,\, Ga] = [2.56,\, 255.35]$	Case 4: $[\rho_\Gamma/\rho,\, Ga] = [7.71,\, 206.27]$
Wall-clock time	2.40×10^4	6.45×10^4	15.80×10^4
Time steps	40036	43325	46438
Cells	2.15×10^5	9.23×10^5	18.07×10^5
Speed	5.91×10^3	6.89×10^3	6.58×10^3

Table 2: Computing times and speeds for a sphere settling in a large domain at different Galileo numbers Ga, computed on 48 Intel Platinum processors until $t/(d/u_{\text{ref}}) = 40$ using a maximum level of refinement $l_{\max} = 12$. The wall-clock time is given in seconds (s) and the speed in points·steps/s/cores. `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c` and `http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c`.

unsteady wake behind the sphere in cases 2 and 4 (see Figures 30 and 32) increases the number of cells and therefore the wall-clock time compared to case 1. This is especially true for case 4, where the density ratio $\rho_\Gamma/\rho = 7.71$ is larger than in cases 1 and 2 and therefore the sphere settles faster, resulting in a longer wake.

7. Conclusions and perspectives

We have presented here an adaptive Cartesian grid embedded boundary method, or cut-cell method, for fixed and moving rigid bodies in an incompressible flow. To the best of our knowledge, this is one of the first attempts to use a cut-cell method for moving embedded boundaries in an incompressible flow. Building on and extending the work of [60, 61], [45, 50] and [49, 22], in particular for the treatment of degenerated cases when computing the embedded face gradient $\nabla_\Gamma s$ (see Section 3.4.1), we have constructed a method that is conservative, second-



(a) $t/\left(d/u_{\mathrm{ref}}\right) = 30$. (b) Zoom at $t/\left(d/u_{\mathrm{ref}}\right) = 30$. (c) $t/\left(d/u_{\mathrm{ref}}\right) = 40$. (d) Zoom at $t/\left(d/u_{\mathrm{ref}}\right) = 40$.

Figure 32: Snapshots at times $t/\left(d/u_{\mathrm{ref}}\right) = \{30, 40\}$ of the iso-surface $\lambda_2 = -0.05$ around of a heavy sphere settling in a large domain for the following combination of density ratio and Galileo number $[\rho_\Gamma/\rho, Ga] = [2.56, 255.35]$ (case 2 of [87]), obtained on an octree defined by the following value of the maximum level of refinement $l_{\mathrm{max}} = 13$ (42 pt/d). `http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling-large-domain.c`.

order in space, robust and efficient. For the sake of simplicity, we have considered in this study only single moving particles and chosen a simple first-order in time explicit integration scheme to weakly couple the motion of a particle and the fluid. Finally, we have implemented the method along with an extensive validation test suite for canonical particle-laden flow cases in the open source software *Basilisk* and extended its use to adaptive tree grids in a robust and efficient manner.

Our Cartesian grid embedded boundary method on quad/oc-tree adaptive grids therefore opens up unprecedented opportunities for particle-resolved particle-laden flow simulations. Indeed, the numerical simulation of moving rigid bodies in an incompressible flow presents many numerical challenges due to the wide variety of particle shapes and sizes and to the complexity of the fluid-particle momentum transfers. We can now overcome these difficulties by taking full advantage of the simplicity of the Cartesian grid approach and cut-cell representation of the geometry to describe moving particles of arbitrary shape. The high-order accuracy and mesh adaptivity properties of the method also allow us to properly capture boundary layers around moving rigid bodies while maintaining computational costs to a minimum. We were therefore able to perform fluid-particle simulations at spatial resolution that were previously unattainable using simple regular Cartesian grid and predict expected particle dynamics for a wide range of Reynolds ($0 \leq Re \leq 1000$) and Galileo numbers ($0 \leq Ga \leq 250$) (see Sections 6.7 and 6.8). Furthermore, the conservative properties of the method enhance the stability and convergence properties of the fluid-solid coupling [21, 22], allowing us to use an explicit weak fluid-solid coupling strategy. In future works, we nevertheless plan to couple the method to the

granular solver Grains3D [64] to compute particle-particle collisions for any particle shape with second-order time accuracy. As demonstrated in Section 6.2, our method can also be straightforwardly extended to heat transfers and could therefore be used to compute high Reynolds and high Prandtl numbers particle-laden flows.

In future works, we will improve the method to remove the small oscillations observed in the pressure signal in Section 6.5 in the limit case of high Reynolds numbers ($Re \sim 1000$). As a reminder, these oscillations are caused by the dynamic adaptation of the mesh coupled to the approximate projection of the cell-centered velocity field. Inspired by the work of [91], these pressure oscillations can be removed by introducing an additional projection step and defining two pressures: a standard pressure used to project the velocity as in equation (36c) and an auxiliary diagnostic pressure used to compute the pressure force on the embedded boundaries. This auxiliary pressure is obtained by projecting only the update (i.e. its evolution in time) of the centered velocity field $\mathbf{u}^{n+1} - \mathbf{u}^n$ and therefore does not feel the history of divergence of the centered velocity field, which includes the noise induced by adaptive mesh refinement. Preliminary work and results for fixed embedded boundaries can be found here: `http://basilisk.fr/src/navier-stokes/double-projection.h`.

References

[1] C. Peskin, Numerical analysis of blood flow in the heart, Journal of Computational Physics 25 (3) (1977) 220–252.

[2] E. A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, Journal of Computational Physics 161 (1) (2000) 35–60.

[3] C. Peskin, The immersed boundary method, Acta Numerica 11 (1) (2002) 479–517.

[4] Z. Feng, E. Michaelides, The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems, Journal of Computational Physics 195 (2) (2004) 602–628.

[5] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, Journal of Computational Physics 209 (2) (2005) 448–476.

[6] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.

[7] R. Bale, A. P. S. Bhalla, B. E. Griffith, M. Tsubokura, A one-sided direct forcing immersed boundary method using moving least squares, Journal of Computational Physics 440 (2021) 110359.

[8] R. Ghias, R. Mittal, T. Lund, A non-body conformal grid method for simulation of compressible flows with complex immersed boundaries, in: 42nd AIAA Aerospace Sciences Meeting and Exhibit, 2004, p. 80.

[9] S. Majumdar, G. Iaccarino, P. Durbin, RANS solvers with adaptive structured boundary non-conforming grids, Tech. rep., CTR Annual Research Brief (2001).

[10] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, Journal of computational physics 227 (10) (2008) 4825–4852.

[11] J. H. Seo, R. Mittal, A high-order immersed boundary method for acoustic wave scattering and low-Mach number flow-induced sound in complex geometries, Journal of Computational Physics 230 (4) (2011) 1000–1019.

[12] S. Tenneti, R. Garg, S. Subramaniam, Drag law for monodisperse gas–solid systems using particle-resolved direct numerical simulation of flow past fixed assemblies of spheres, International Journal of Multiphase Flow 37 (9) (2011) 1072–1092.

[13] J. Xia, K. Luo, J. Fan, A ghost-cell based high-order immersed boundary method for interphase heat transfer simulation, International Journal of Heat and Mass Transfer 75 (2014) 302–312.

[14] R. Glowinski, T. Pan, T. Hesla, D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, International Journal of Multiphase Flow 25 (5) (1999) 755–794.

[15] Z. Yu, X. Shao, A. Wachs, A fictitious domain method for particulate flows with heat transfer, Journal of Computational Physics 217 (2) (2006) 424–452.

[16] A. Wachs, A. Hammouti, G. Vinay, M. Rahmani, Accuracy of finite volume/staggered grid distributed Lagrange multiplier/fictitious domain simulations of particulate flows, Computers & Fluids 115 (2015) 154–172.

[17] A. Ladd, R. Verberg, Lattice-Boltzmann simulations of particle-fluid suspensions, Journal of Statistical Physics 104 (5) (2001) 1191–1251.

[18] C. Peng, O. M. Ayala, L.-P. Wang, A comparative study of immersed boundary method and interpolated bounce-back scheme for no-slip boundary treatment in the lattice Boltzmann method: Part I, laminar flows, Computers & Fluids 192 (2019) 104233.

[19] C. K. Aidun, J. R. Clausen, Lattice-Boltzmann method for complex flows, Annual Review of Fluid Mechanics 42 (2010) 439–472.

[20] 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 (2019) 1919–1980.

[21] C. Michler, E. Van Brummelen, S. Hulshoff, R. De Borst, The relevance of conservation for stability and accuracy of numerical methods for fluid–structure interaction, Computer methods in applied mechanics and engineering 192 (37-38) (2003) 4195–4215.

[22] L. Schneiders, C. Günther, M. Meinke, W. Schröder, An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows, Journal of Computational Physics 311 (2016) 62–86.

[23] J. Feng, H. Hu, D. Joseph, Direct simulation of initial value problems for the motion of solid bodies in a Newtonian fluid. Part 1. Sedimentation, Journal of Fluid Mechanics 261 (1994) 95–134.

[24] J. Feng, H. Hu, D. Joseph, Direct simulation of initial value problems for the motion of solid bodies in a Newtonian fluid. Part 2. Couette and Poiseuille flows, Journal of Fluid Mechanics 277 (271) (1994) 271–301.

[25] A. A. Johnson, T. E. Tezduyar, Advanced mesh generation and update methods for 3D flow simulations, Computational Mechanics 23 (2) (1999) 130–143.

[26] C. R. Choi, C. N. Kim, Direct numerical simulations of the dynamics of particles with arbitrary shapes in shear flows, Journal of Hydrodynamics, Ser. B 22 (4) (2010) 456–465.

[27] N. O. Jaensson, M. A. Hulsen, P. D. Anderson, Direct numerical simulation of particle alignment in viscoelastic fluids, Journal of Non-Newtonian Fluid Mechanics 235 (2016) 125–142.

[28] A. Koblitz, S. Lovett, N. Nikiforakis, W. D. Henshaw, Direct numerical simulation of particulate flows with an overset grid method, Journal of computational physics 343 (2017) 414–431.

[29] W. J. Horne, K. Mahesh, A massively-parallel, unstructured overset method to simulate moving bodies in turbulent flows, Journal of Computational Physics 397 (2019) 108790.

[30] K. Luo, Z. Wang, J. Fan, K. Cen, Full-scale solutions to particle-laden flows: Multidirect forcing and immersed boundary method, Physical Review E 76 (6) (2007) 066709.

[31] Z. Wang, J. Fan, K. Luo, Combined multi-direct forcing and immersed boundary method for simulating flows with moving particles, International Journal of Multiphase Flow 34 (3) (2008) 283–302.

[32] T. Kempe, J. Fröhlich, Collision modelling for the interface-resolved simulation of spherical particles in viscous fluids, Journal of Fluid Mechanics 709 (2012) 445–489.

[33] Y. Tang, S. Kriebitzsch, E. Peters, M. van der Hoef, J. Kuipers, A methodology for highly accurate results of direct numerical simulations: drag force in dense gas–solid flows at intermediate Reynolds number, International Journal of Multiphase Flow 62 (2014) 73–86.

[34] S. Schwarz, T. Kempe, J. Fröhlich, A temporal discretization scheme to compute the motion of light particles in viscous flows by an immersed boundary method, Journal of Computational Physics 281 (2015) 591–613.

[35] G. Akiki, S. Balachandar, Immersed boundary method with non-uniform distribution of Lagrangian markers for a non-uniform Eulerian mesh, Journal of Computational Physics 307 (2016) 34–59.

[36] M. Moriche, M. Uhlmann, J. Dušek, A single oblate spheroid settling in unbounded ambient fluid: A benchmark for simulations in steady and unsteady wake regimes, International Journal of Multiphase Flow 136 (2021) 103519.

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

[38] J. Keating, P. Minev, A fast algorithm for direct simulation of particulate flows using conforming grids, Journal of Computational Physics 255 (2013) 486–501.

[39] N. Deen, S. Kriebitzsch, M. van der Hoef, J. Kuipers, Direct numerical simulation of flow and heat transfer in dense fluid-particle systems, Chemical Engineering Science 81 (2012) 329–344.

[40] X. Liu, W. Ge, L. Wang, Scale and structure dependent drag in gas–solid flows, AIChE Journal 66 (4) (2020) e16883.

[41] A. G. Kidanemariam, M. Uhlmann, Formation of sediment patterns in channel flow: minimal unstable systems and their temporal evolution, Journal of Fluid Mechanics 818 (2017) 716–743.

[42] A. Roma, C. Peskin, M. Berger, An adaptive version of the immersed boundary method, Journal of Computational Physics 153 (2) (1999) 509–534.

[43] B. E. Griffith, R. D. Hornung, D. M. McQueen, C. S. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, Journal of Computational Physics 223 (1) (2007) 10–49.

[44] M. Bauer, S. Eibl, C. Godenschwager, N. Kohl, M. Kuron, C. Rettinger, F. Schornbaum, C. Schwarzmeier, D. Thönnes, H. Köstler, U. Rude, waLBerla: A block-structured high-performance framework for multiphysics simulations, Computers & Mathematics with Applications 81 (2020) 478–501.

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

[46] C. Min, F. Gibou, A second order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids, Journal of Computational Physics 219 (2) (2006) 912–929.

[47] G. Eitel-Amor, M. Meinke, W. Schröder, A lattice-Boltzmann method with hierarchically refined meshes, Computers & Fluids 75 (2013) 127–139.

[48] M. Meinke, L. Schneiders, C. Günther, W. Schröder, A cut-cell method for sharp moving boundaries in cartesian grids, Computers & Fluids 85 (2013) 135–142.

[49] L. Schneiders, D. Hartmann, M. Meinke, W. Schröder, An accurate moving boundary formulation in cut-cell methods, Journal of Computational Physics 235 (2013) 786–809.

[50] S. Popinet, A quadtree–adaptive multigrid solver for the Serre–Green–Naghdi equations, Journal of Computational Physics 302 (2015) 336–358.

[51] G. D'Avino, M. 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) (2010) 1014–1040.

[52] C. Selcuk, A. R. Ghigo, S. Popinet, A. Wachs, A fictitious domain method with distributed lagrange multipliers on adaptive quad/octrees for the direct numerical simulation of particle-laden flows, Journal of Computational Physics 430 (2021) 109954.

[53] 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 (6) (2019) 1919–1980.

[54] W. Bennett, N. Nikiforakis, R. Klein, A moving boundary flux stabilization method for Cartesian cut-cell grids using directional operator splitting, Journal of Computational Physics 368 (2018) 333–358.

[55] K. Sverdrup, A. Almgren, N. Nikiforakis, An embedded boundary approach for efficient simulations of viscoplastic fluids in three dimensions, Physics of Fluids 31 (9) (2019) 093102.

[56] K. Fröhlich, M. Meinke, W. Schröder, Correlations for inclined prolates based on highly resolved simulations, Journal of Fluid Mechanics 901 (2020) A5.

[57] M.-H. Chung, Cartesian cut cell approach for simulating incompressible flows with rigid bodies of arbitrary shape, Computers & Fluids 35 (6) (2006) 607–623.

[58] M.-H. Chung, An adaptive Cartesian cut-cell/level-set method to simulate incompressible two-phase flows with embedded moving solid boundaries, Computers & Fluids 71 (2013) 469–486.

[59] Z. Xie, T. Stoesser, A three-dimensional Cartesian cut-cell/volume-of-fluid method for two-phase flows with moving bodies, Journal of Computational Physics 416 (2020) 109536.

[60] H. Johansen, P. Colella, A cartesian grid embedded boundary method for Poisson's equation on irregular domains, Journal of Computational Physics 147 (1) (1998) 60–85.

[61] P. Schwartz, M. Barad, P. Colella, T. Ligocki, A cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions, Journal of Computational Physics 211 (2) (2006) 531–550.

[62] P. Colella, D. T. Graves, B. J. Keen, D. Modiano, A cartesian grid embedded boundary method for hyperbolic conservation laws, Journal of Computational Physics 211 (1) (2006) 347–366.

[63] J. B. Bell, P. Colella, H. M. Glaz, A second-order projection method for the incompressible navier-stokes equations, Journal of Computational Physics 85 (2) (1989) 257–283.

[64] A. Wachs, L. Girolami, G. Vinay, G. Ferrer, Grains3d, a flexible dem approach for particles of arbitrary convex shape — Part I: Numerical model and validations, Powder Technology 224 (2012) 374–389.

[65] L. Schneiders, C. Guenther, J. H. Grimmen, M. H. Meinke, W. Schroeder, Sharp resolution of complex moving geometries using a multi-cut-cell viscous flow solver, in: 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 3427.

[66] R. B. Pember, J. B. Bell, P. Colella, W. Y. Curtchfield, M. L. Welcome, An adaptive Cartesian grid method for unsteady compressible flow in irregular regions, Journal of computational Physics 120 (2) (1995) 278–304.

[67] R. Scardovelli, S. Zaleski, Analytical relations connecting linear interfaces and volume fractions in rectangular grids, Journal of Computational Physics 164 (1) (2000) 228–237.

[68] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, Journal of Computational Physics 228 (16) (2009) 5838–5866.

[69] D. Trebotich, D. Graves, An adaptive finite volume method for the incompressible Navier–Stokes equations in complex geometries, Communications in Applied Mathematics and Computational Science 10 (1) (2015) 43–82.

[70] J. A. van Hooft, S. Popinet, C. C. van Heerwaarden, S. J. van der Linden, S. R. de Roode, B. J. van de Wiel, Towards adaptive grids for atmospheric boundary–layer simulations, Boundary-layer meteorology 167 (3) (2018) 421–443.

[71] J. López-Herrera, S. Popinet, A. Castrejón-Pita, An adaptive solver for viscoelastic incompressible two–phase problems applied to the study of the splashing of weakly viscoelastic droplets, Journal of Non-Newtonian Fluid Mechanics 264 (2019) 144–158.

[72] A. J. Chorin, On the convergence of discrete approximations to the Navier-Stokes equations, Mathematics of computation 23 (106) (1969) 341–353.

[73] P. M. Gresho, R. L. Sani, On pressure boundary conditions for the incompressible Navier–Stokes equations, International Journal for Numerical Methods in Fluids 7 (10) (1987) 1111–1145.

[74] H. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface cartesian grid method for simulating flows with complex moving boundaries, Journal of computational physics 174 (1) (2001) 345–380.

[75] J. H. Seo, R. Mittal, A sharp–interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations, Journal of computational physics 230 (19) (2011) 7347–7363.

[76] J. J. Quirk, An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies, Computers & fluids 23 (1) (1994) 125–142.

[77] C. Helzel, M. J. Berger, R. J. LeVeque, A high-resolution rotated grid method for conservation laws with embedded geometries, SIAM Journal on Scientific Computing 26 (3) (2005) 785–809.

[78] N. Gokhale, N. Nikiforakis, R. Klein, A dimensionally split Cartesian cut cell method for hyperbolic conservation laws, Journal of Computational Physics 364 (2018) 186–208.

[79] P. Koumoutsakos, A. Leonard, High–resolution simulations of the flow around an impulsively started cylinder using vortex methods, Journal of Fluid Mechanics 296 (1995) 1–38.

[80] F. Mohaghegh, H. Udaykumar, Comparison of sharp and smoothed interface methods for simulation of particulate flows II: Inertial and added mass effects, Computers & Fluids 143 (2017) 103–119.

[81] R. Mei, Flow due to an oscillating sphere and an expression for unsteady drag on the sphere at finite Reynolds number, Journal of Fluid Mechanics 270 (1994) 133–174.

[82] X. Wang, P. Yu, K. Yeo, B. Khoo, SVD–GFD scheme to simulate complex moving body problems in 3D space, Journal of Computational Physics 229 (6) (2010) 2314–2338.

[83] A. Ten Cate, C. Nieuwstad, J. Derksen, H. Van den Akker, Particle imaging velocimetry experiments and Lattice-Boltzmann simulations on a single sphere settling under gravity, Physics of Fluids 14 (11) (2002) 4012–4025.

[84] C. Veeramani, P. D. Minev, K. Nandakumar, A fictitious domain formulation for flows with rigid particles: A non-lagrange multiplier version, Journal of Computational Physics 224 (2) (2007) 867–879.

[85] Z.-G. Feng, E. E. Michaelides, Robust treatment of no-slip boundary condition and velocity updating for the lattice-Boltzmann simulation of particulate flows, Computers & Fluids 38 (2) (2009) 370–381.

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

[87] N. Mordant, J.-F. Pinton, Velocity measurement of a settling sphere, The European Physical Journal B-Condensed Matter and Complex Systems 18 (2) (2000) 343–352.

[88] M. Jenny, J. Dušek, G. Bouchet, Instabilities and transition of a sphere falling or ascending freely in a Newtonian fluid, Journal of Fluid Mechanics 508 (2004) 201–239.

[89] M. Horowitz, C. Williamson, The effect of reynolds number on the dynamics and wakes of freely rising and falling spheres, Journal of Fluid Mechanics 651 (2010) 251–294.

[90] C. Rettinger, U. Rüde, A comparative study of fluid-particle coupling methods for fully resolved lattice Boltzmann simulations, Computers & Fluids 154 (2017) 74–89.

[91] A. S. Almgren, J. B. Bell, W. Y. Crutchfield, Approximate projection methods: Part I. Inviscid analysis, SIAM Journal on Scientific Computing 22 (4) (2000) 1139–1159.