



HAL
open science

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

► **To cite this version:**

Arthur R. Ghigo, Stéphane Popinet, Anthony Wachs. A conservative finite volume cut-cell method on an adaptive Cartesian tree grid for moving rigid bodies in incompressible flows. 2021. hal-03948786

HAL Id: hal-03948786

<https://hal.science/hal-03948786>

Preprint submitted on 20 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

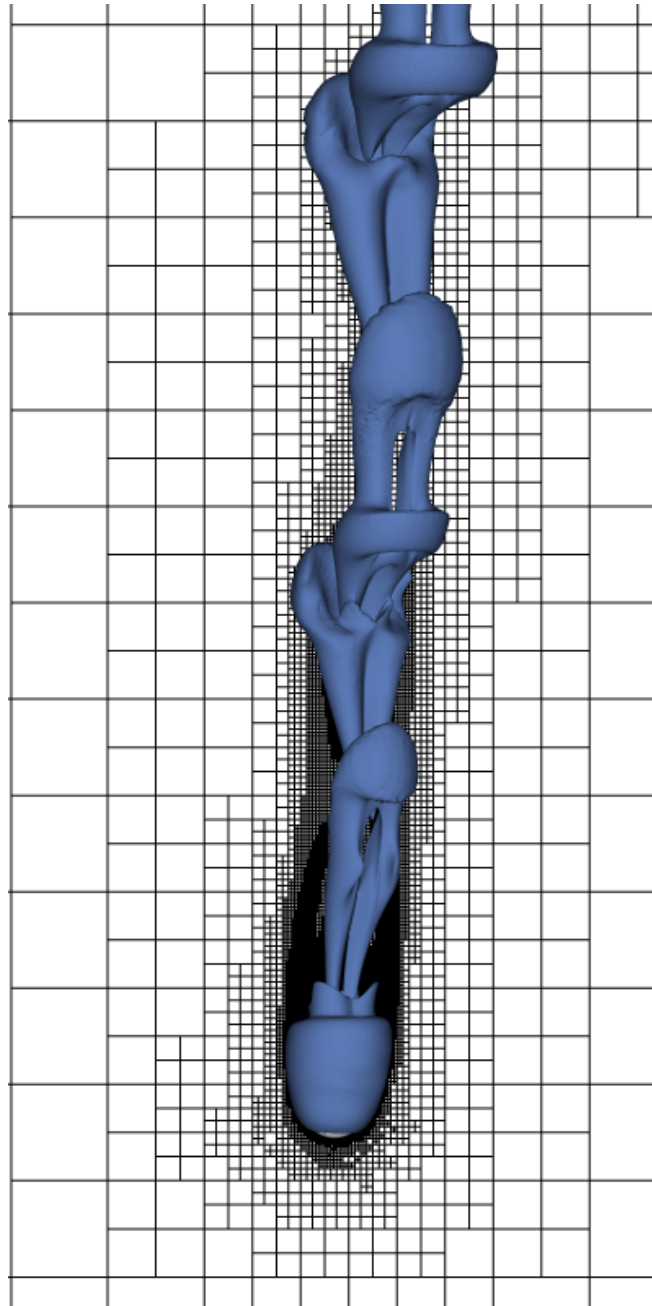


Distributed under a Creative Commons Attribution 4.0 International License

1 Graphical Abstract

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

4 Arthur R. Ghigo, Stéphane Popinet, Anthony Wachs



6 Highlights

7 **A conservative finite volume cut-cell method on an adaptive Cartesian tree grid** 8 **for moving rigid bodies in incompressible flows**

9 Arthur R. Ghigo, Stéphane Popinet, Anthony Wachs

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

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

20 Arthur R. Ghigo^{a,b}, Stéphane Popinet^d, Anthony Wachs^{a,c}

21 ^a*Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, BC V6T*
22 *1Z4, Canada*

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

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

26 ^d*Institut Jean Le Rond d'Alembert, Sorbonne Université, Centre National de la Recherche Scientifique, Paris*
27 *F-75005, France*

28 **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 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.

29 *Keywords:* Particle-laden flow; Embedded boundary; Cut-cell method; Conservative Finite
30 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

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

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

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

101 In non body-conforming methods, the complex numerical machinery required to keep track
102 of the deforming mesh is traded for a (relatively) simple grid management (in the case of
103 adaptive grids) or no grid management at all (in the case of fixed grids). However, this comes
104 at the expense of introducing either a velocity reconstruction at the rigid boundaries or an
105 additional body force distributed along the rigid boundaries in the momentum conservation
106 equation to impose the hydrodynamic coupling between the fluid and the rigid bodies. The IB
107 method based on discrete direct forcing (DF) using regularized Dirac delta functions [5, 30, 31,
108 32, 33, 34, 35] is probably the most popular PRS method in the literature but is also notoriously
109 non-sharp. Indeed, the regularized Dirac delta kernel used to distribute the additional body
110 force defined on the rigid boundaries is constructed with a compact support that symmetrically
111 spans multiple cells on both sides of the rigid boundaries and consequently smooths any gradient
112 discontinuity at the rigid boundaries. Note that in [7], the authors recently suggested an
113 improvement of the DF/IB method in the form of one-sided IB kernels that is equivalent to a
114 sharp treatment of the rigid boundaries. The standard DF/IB method is non-conservative in
115 the domain occupied by the fluid as cells cut by the rigid boundaries are not treated in any
116 specific manner in the fluid conservation equations (in particular the mass conservation does not
117 account at all for the presence of rigid boundaries). Note also that the standard DF/IB method
118 forces the no-slip boundary condition on the particle surface but often does not force any rigid
119 body motion in the particle volume, leading to nonphysical fictitious fluid motion inside rigid

120 particles. Recently, in [36], the authors added volume forcing to the DF/IB method proposed in
 121 [5], in the spirit of the DLM/FD method to improve the spatial accuracy of the DF/IB method
 122 applied to the problem of inertial settling of a single ellipsoid. The GC method, the BBS based
 123 LB method and the DLM/FD method all deliver the right rigid body motion in the particle
 124 volume. The GC and BBS-LB methods exclude the degrees of freedom and simply assign
 125 the right velocity values in the particle volume while the DLM/FD method employs a volume
 126 forcing. The GC method as well as the implementation of the DLM/FD method in [37] provide
 127 a sharp description of the rigid boundaries and properly capture the gradient discontinuity
 128 at the rigid boundaries but are again not strictly conservative in the domain occupied by the
 129 fluid. Other PRS methods are also sharp and pseudo-body conforming [38, 39] in the sense
 130 that they use a fixed Cartesian grid and adapt the finite difference approximation stencil of the
 131 gradient in the vicinity of the rigid boundaries to incorporate the Dirichlet no-slip condition
 132 at the rigid boundaries. Again, these methods are not strictly conservative in the domain
 133 occupied by the fluid. Finally, in most of the aforementioned PRS methods, the treatment of
 134 the immersed rigid boundaries disrupts the second-order accuracy of the conservation equation
 135 spatial discretization scheme in the absence of immersed rigid boundaries and the overall spatial
 136 accuracy is generally between first-order and second-order, depending on the flow problem.

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

165 implemented on a uniform Cartesian grid, PRS methods that describe the rigid boundaries by
 166 a set of Lagrangian discrete points require a quasi-homogeneous distribution of these points
 167 on the rigid boundaries [5, 16]. This quasi-homogeneous distribution is constructed such that
 168 Lagrangian points are approximately equidistant by $\alpha\Delta$ where $\alpha \in [1, 2]$ and Δ is the constant
 169 grid size. The linear relationship between the inter-point distance and the local grid size as
 170 well as the homogeneity of the distribution are key to the accuracy of the computed solution
 171 [5, 51, 16]. Various constructions can be adopted depending on the rigid body shape: (i) ex-
 172 plicit for a sphere [5, 16] and simple polyhedrons [16], (ii) solution of a time-dependent problem
 173 of charged particles constrained to the body surface for spheroids [5, 36], and (iii) vertices of
 174 an unstructured triangulation of the body surface. On a Cartesian tree grid with a local grid
 175 size that presumably varies along the body surface, the inter-point distance (or equivalently
 176 the surface density of Lagrangian points) would need to match the local grid size, leading to
 177 a non-homogeneous Lagrangian point distribution over the rigid boundary and thus rendering
 178 the Lagrangian point construction very challenging if not overly complicated. Conversely, PRS
 179 methods that describe rigid boundaries on the Eulerian grid only extend more naturally to
 180 variable cell size Cartesian grids like Cartesian tree grids. In our previous work, we adopted
 181 an intermediate solution when extending our DLM/FD method on Cartesian tree grids [52] by
 182 enforcing a uniform grid in a narrow band spanning 3 cells on both sides of each rigid boundary.
 183 While computationally efficient, this strategy does not take full advantage of the local mesh
 184 refinement capabilities of a genuine Cartesian tree grid.

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

199 The Cartesian grid embedded boundary method for incompressible flows with fixed and
 200 moving rigid boundaries we present here satisfies the four key properties listed previously.
 201 Indeed, we discretize the incompressible Navier-Stokes equations using a fractional-step projec-
 202 tion method and insure that at each time step, the finite volume discretization scheme remains
 203 spatially second-order and conservative in cut-cells by carefully imposing second-order accu-
 204 rate boundary conditions on both 2D and 3D embedded (rigid) boundaries. In particular, we
 205 follow and extend the work of [60, 61] to robustly compute gradients normal to the embedded
 206 boundaries, even in degenerated cases. To avoid stability issues due to the well-documented
 207 problem of small cut-cells, we use a simple and efficient flux redistribution technique, initially
 208 suggested in [62], to extend the range of influence of small cut-cells to their neighboring cells.
 209 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 $\mathbf{x} = [x, y, z]^\top$ by:

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

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

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 $D_{ij} \equiv (\partial_i u_j + \partial_j u_i)/2$.

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

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

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].

242 *2.2. Rigid body dynamics*

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

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

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}_Γ . These variables satisfy the following equations of motion [64]:

$$\frac{d\mathbf{x}_\Gamma}{dt} = \mathbf{u}_\Gamma \quad (4a)$$

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

$$\frac{d}{dt} (\mathbf{I}_\Gamma \omega_\Gamma) = \mathbf{T}_\Gamma, \quad (4c)$$

where V_Γ and \mathbf{I}_Γ 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}_Γ) exerted
 by the fluid on the rigid body Γ :

$$\mathbf{F}_\Gamma = - \int_{\delta\Gamma} (-p\mathbb{I} + 2\mu\mathbf{D}) \cdot \mathbf{n}_\Gamma dS \quad (5a)$$

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

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

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

- 251 • $\Gamma = \{\mathbf{x} \in \Omega \mid \Phi(\mathbf{x} - \mathbf{x}_\Gamma) \leq 0\}$;
- 252 • $\delta\Gamma = \{\mathbf{x} \in \Omega \mid \Phi(\mathbf{x} - \mathbf{x}_\Gamma) = 0\}$;
- 253 • $\Omega/\Gamma = \{\mathbf{x} \in \Omega \mid \Phi(\mathbf{x} - \mathbf{x}_\Gamma) > 0\}$.

254 **3. Cartesian grid embedded boundary method on a uniform grid**

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

262 *3.1. Uniform Cartesian grid in Basilisk*

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

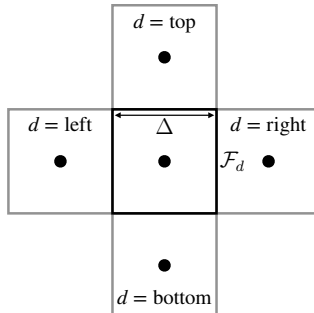


Figure 1: Representation of a 2D cell and its neighbors in *Basilisk*.

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

- 275 • $[0, 0] = []$ for the current cell, i.e. $s[0, 0]$;
- 276 • $[1, 0] = [1]$ for its *right* neighbor, i.e. $s[1, 0]$;
- 277 • $[-1, 0] = [-1]$ for its *left* neighbor, i.e. $s[-1, 0]$;
- 278 • $[0, 1]$ for its *top* neighbor, i.e. $s[0, 1]$;
- 279 • $[0, -1]$ for its *bottom* neighbor, i.e. $s[0, -1]$.

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

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

$$\nabla_x s [] = \frac{s[1] - s[-1]}{2\Delta}, \quad (6)$$

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

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

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

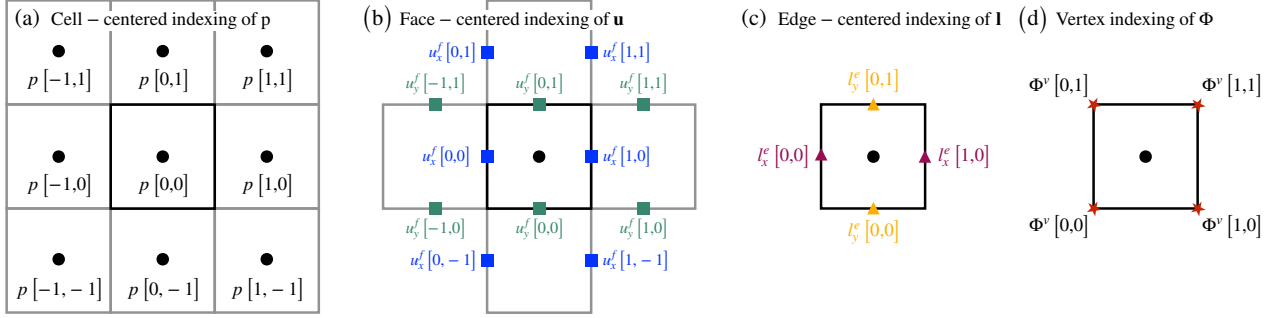


Figure 2: Stencils in *Basilisk* and their respective indexing on a 2D grid: (a) cell-centered; (b) face-centered \mathbf{f} ; (c) edge-centered \mathbf{e} (identical to face-centered in 2D); (d) vertex \mathbf{v} .

285 3.2. Integral description of rigid boundary $\delta\Gamma$

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

- 293 • the volume fraction $0 \leq c \leq 1$ of the cell, such that the effective volume occupied by the
 294 fluid in the cell is:

$$V = c\Delta^D, \quad (8)$$

295 where D is the number of space dimensions. A cell can therefore be: (i) a full cell with
 296 $c = 1$; (ii) a solid cell with $c = 0$ or (iii) a *cut-cell* with $0 < c < 1$.

- 297 • the area fraction $0 \leq f_d^f \leq 1$ of the face \mathcal{F}_d , such that the effective area occupied by the
 298 fluid on the face is:

$$A_d^f = f_d^f \Delta^{D-1}. \quad (9)$$

299 Note that we also refer to A_d^f as the *partial* face occupied by the fluid on the (*full*) face
 300 \mathcal{F}_d , i.e. we use the notation A_d^f for both the partial face and its surface area and denote
 301 \mathbf{m}_d^f its centroid (see Figure 3c).

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

308 In the following section, we detail the computation of the embedded fractions c and f_d^f start-
 309 ing from the distance function representation of the rigid boundary $\delta\Gamma$ introduced in Section
 310 2.2.

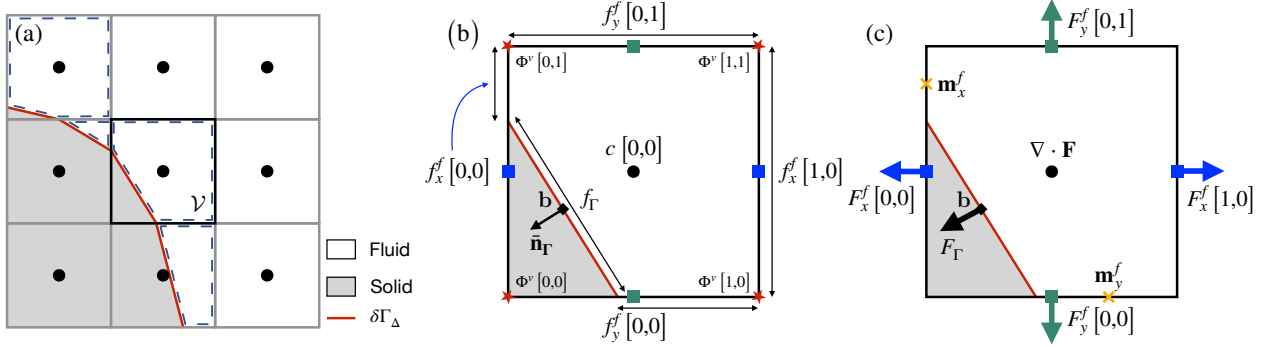


Figure 3: Graphical representation of the Cartesian grid embedded boundary method on a 2D grid: (a) Cartesian grid Ω_Δ cut by the discrete rigid boundary $\delta\Gamma_\Delta$ to form irregular control volumes \mathcal{V} ; (b) Geometric and VOF quantities in a *cut-cell*; (c) Conservative discretization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

$$\hat{\mathbf{n}}_\Gamma \cdot \mathbf{x} = \alpha, \quad (10)$$

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

Using the distance function $\Phi(\mathbf{x} - \mathbf{x}_\Gamma)$ sampled on the vertices of each cell of grid Ω_Δ , we then compute the geometric quantities $\hat{\mathbf{n}}_\Gamma$ 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[\]$, where $\Phi(\mathbf{x} - \mathbf{x}_\Gamma)$ 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[\] \cdot \Phi^v[1] < 0$, the face is cut by the rigid boundary $\delta\Gamma$ and we compute the corresponding area fraction $f_d^f = f_y^f[\]$ using a linear interpolation of the vertex values of Φ :

$$f_y^f[\] = \frac{1 - \text{sign}(\Phi^v[\])}{2} + \text{sign}(\Phi^v[\]) \frac{\Phi^v[\]}{\Phi^v[\] - \Phi^v[1]}. \quad (11)$$

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

$$\int_{\delta\mathcal{V}} \mathbf{n} \, dA = 0 \Leftrightarrow \sum_d f_d^f \mathbf{n}_d^f + \hat{\mathbf{n}}_\Gamma = 0, \quad (12)$$

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

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

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

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

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

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

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

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

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

341 5. Finally, we compute the volume fraction c using the well-established functions detailed in
 342 [67, 68], which account for the different ways a square (cubic in 3D) cell can be cut by a
 343 line (plane in 3D):

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

344 The generalisation of the previous algorithm to 3D is relatively straightforward. Indeed,
 345 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
 346 a 3D cut-cell as we would compute the volume fraction of a 2D cut-cell. We then apply steps 3
 347 to 5 of the previous algorithm to compute the volume fraction c of the 3D cut-cell, with small
 348 modifications to account for the third dimension. Complete details of the implementation of
 349 the previous algorithm and its extension to 3D can be found here: [http://basilisk.fr/src/
 350 fractions.h#computing-volume-fractions-from-a-levelset-function](http://basilisk.fr/src/fractions.h#computing-volume-fractions-from-a-levelset-function).

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

355 3.4. Discrete operators in a cut-cell

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

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

359 where \mathbf{n} is the outward unit normal vector to the boundary $\delta\mathcal{V}$ of the fluid control-volume \mathcal{V}
 360 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
 361 respectively represent the *finite volume* flux through the face \mathcal{F}_d and through the discrete rigid
 362 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
 363 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
 364 discrete rigid boundary $\delta\Gamma_\Delta$.

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

- 369 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
 370 centroid \mathbf{b} of the discrete rigid boundary $\delta\Gamma_\Delta$ in the cut-cell;
- 371 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
 372 of the partial face A_d^f in the cut-cell;
- 373 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
 374 A_d^f in the cut-cell.

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

381 3.4.1. Computation of the embedded face gradient of a cell-centered scalar

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

385 *Neumann boundary condition.* If a Neumann boundary condition g_Γ is imposed, then the em-
 386 bedded face gradient $\nabla_\Gamma s$ simply writes:

$$\nabla_\Gamma s = g_\Gamma. \quad (18)$$

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

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

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

393 direction of the normal vector $-\bar{\mathbf{n}}_\Gamma$ and at a positive distance d_1 and d_2 from the centroid \mathbf{b} .
 394 Without loss of generality, we consider the 2D configuration presented in Figure 4, where the
 395 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
 396 $d_2 = \frac{2-b_x}{-\bar{n}_{\Gamma,x}}$. We then compute s_1 and s_2 using values of s taken from the 3×2 ($3 \times 3 \times 2$ in 3D)
 397 stencil represented in Figure 4b, itself part of the 5×5 ($5 \times 5 \times 5$ in 3D) stencil of the cut cell:

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

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

402 To maintain a robust computation of the embedded face gradient $\nabla_\Gamma s$ in complex geomet-
 403 rical configurations that can naturally occur in particle-laden flows, we use equation (19) only
 404 if the cells required to compute the first and second interpolants s_1 and s_2 in equation (20) are
 405 *topologically connected*. This means that a line unbroken by a solid cell or face connects the
 406 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.
 407 cells $\{[1], [1, 1], [1, 2]\}$ and cells $\{[2], [2, 1], [2, 2]\}$ for the 2D configuration presented in Figure
 408 4. If this is not the case, we face a degenerated case. This situation was not considered in
 409 [60, 61] and we therefore propose the following extension of the computation of the embedded
 410 face gradient $\nabla_\Gamma s$ in degenerated cases:

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

$$\nabla_\Gamma s = \frac{s_\Gamma - s_1}{d_1}. \quad (21)$$

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

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

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

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

421 Complete details of the computation of the embedded face gradient $\nabla_\Gamma s$ and its extension to
 422 3D can be found here: <http://basilisk.fr/src/embed.h#dirichlet-boundary-condition>.

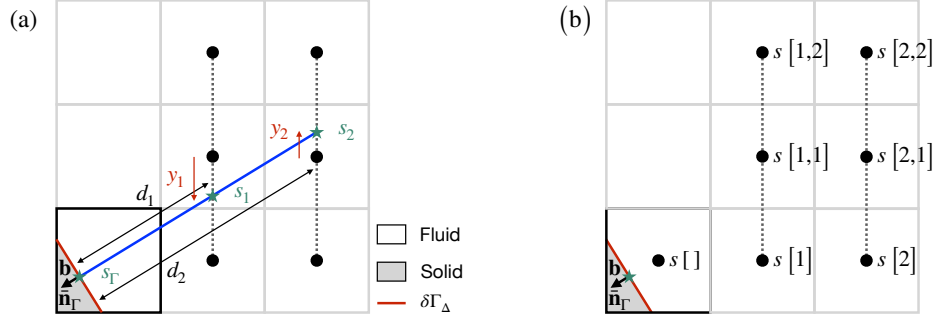


Figure 4: Graphical representation of the methodology proposed in [60, 61] and implemented in *Basilisk* to compute the second-order embedded face gradient $\nabla_{\Gamma}s$ on a 2D grid when the principal direction of the normal vector $-\bar{\mathbf{n}}_{\Gamma}$ is the positive x -direction.

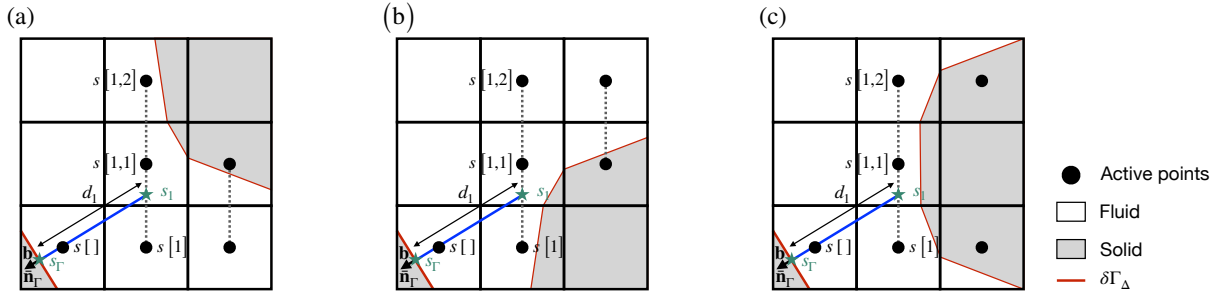


Figure 5: Graphical representation of the extension of the methodology proposed in [60, 61] and implemented in *Basilisk* to compute the embedded face gradient $\nabla_{\Gamma}s$ in a first type of degenerated case on a 2D grid when the principal direction of the normal vector $-\bar{\mathbf{n}}_{\Gamma}$ 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 $[0, 0]$.

423 3.4.2. Computation of the face gradient of a cell-centered scalar

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

$$\bar{\nabla}_x^f s[] = \frac{s[] - s[-1]}{\Delta}. \quad (23)$$

429 We then proceed as in [60] and define $\nabla_d^f s$ using a second-order linear (bilinear in 3D)
 430 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
 431 on neighboring faces located in the direction (two directions in 3D) orthogonal to the direction
 432 d . Without loss of generality, we consider here for instance the *left* face of the 2D cut-cell

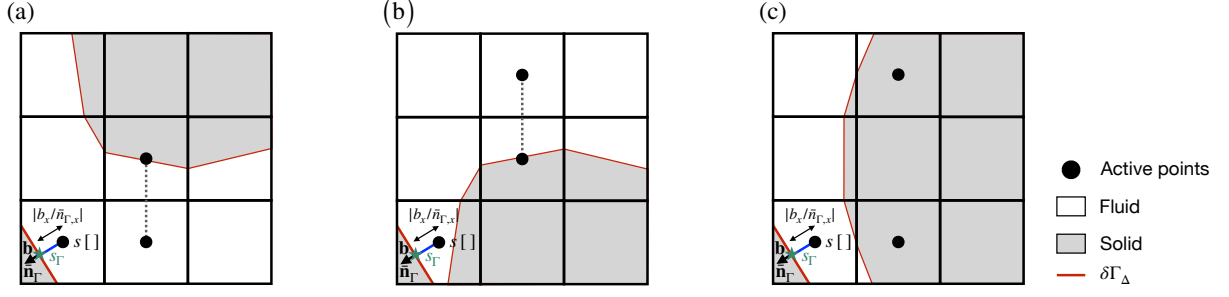


Figure 6: Graphical representation of the extension of the methodology proposed in [60, 61] and implemented in *Basilisk* to compute the embedded face gradient $\nabla_{\Gamma} s$ in a second type of degenerated case on a 2D grid when the principal direction of the normal vector $-\bar{\mathbf{n}}_{\Gamma}$ 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 $[0, 0]$.

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

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

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

436 As for the embedded face gradient $\nabla_{\Gamma} s$, we only use equation (24) if the cells required to
437 define the simple face gradients $\bar{\nabla}_d^f s$ used in equation (24) are *topologically connected* to the
438 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
439 not topologically connected could prevent the convergence of the multigrid Poisson-Helmholtz
440 solver [61].

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

443 3.4.3. Computation of the face value of a cell-centered scalar

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

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

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

452 when using an approximate projection scheme, due to some feedback between pressure and
 453 velocity modes, amplified by the face gradient (24) used to compute the viscous fluxes, which
 454 can behave as a third-order term in cut-cells when combined with Dirichlet boundary conditions
 455 [60, 61] (see Section 6.1).

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

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

460 4. Extension of the Cartesian grid embedded boundary method to quad/oct-tree 461 grids

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

468 4.1. Properties of tree grids in *Basilisk* in the absence of embedded boundaries

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

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

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

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

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

$$s [] = (9 \text{ coarse } (s []) + 3 (\text{coarse } (s [\text{child}.x]) + \text{coarse } (s [0, \text{child}.y])) + \text{coarse } (s [\text{child}.x, \text{child}.y])) / 16, \quad (26)$$

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

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

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

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

507 Complete details of the implementation of the *prolongation* and *restriction* functions in
 508 the absence of embedded boundaries and their extension to 3D can be found here: <http://basilisk.fr/src/grid/multigrid-common.h>.
 509

510 4.2. Properties of tree grids in Basilisk in the presence of embedded boundaries

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

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

516 4.2.1. Computation of embedded fractions in a halo cell

517 We use prolongation and restriction functions specific to the embedded fractions to recon-
 518 struct the volume and area fractions c and f_d^f in a halo cell.

519 The restriction functions defining the embedded fractions in a parent halo cell at level l
 520 use simple averaging of values in its children cells at level $l + 1$. The prolongation functions
 521 defining the embedded fractions in a halo cell at level $l + 1$ are based on a VOF reconstruction
 522 of the embedded boundary normal vector $\bar{\mathbf{n}}_{\Gamma,l}$ and intercept α_l in the corresponding parent cell
 523 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
 524 level l , we use equation (13) to compute the corresponding normal vector $\bar{\mathbf{n}}_{\Gamma,l}$ and a variation
 525 of equation (16) to compute the intercept α_l . Then, given $\bar{\mathbf{n}}_{\Gamma,l}$ and α_l defined in a coordinate

526 system with origin the center of the halo cell and in which the halo cell size is unity, we use
 527 function (16) to compute the volume fraction c_{l+1} in the quadrant of the parent cell matching
 528 the halo cell. We also use $\bar{\mathbf{n}}_{\Gamma,l}$, α_l and equation (10) to compute the area fractions $f_{d,l+1}^{f,\text{int}}$ of the
 529 faces of the halo cell contained within the parent cell. We finally compute the area fractions
 530 $f_{d,l+1}^{f,\text{ext}}$ of the faces of the halo cell that coincide with the faces of the parent cell using their
 531 corresponding area fractions $f_{d,l}^f$.

532 We refer the reader to [68] for more details on the prolongation and restriction of a vol-
 533 ume fraction and to the following for complete details on the implementation of the prolonga-
 534 tion and restriction functions specific to the embedded fractions: [http://basilisk.fr/src/
 535 embed-tree.h#volume-fraction-field-cs](http://basilisk.fr/src/embed-tree.h#volume-fraction-field-cs).

536 4.2.2. Prolongation and restriction functions in the presence of embedded boundaries

537 In the presence of embedded boundaries, we modify the *prolongation* and *restriction* func-
 538 tions (26) and (27) and define second-order operators using only values from available cells, i.e.
 539 cells not entirely contained within the embedded boundary and, if the discrete rigid boundary
 540 $\delta\Gamma_\Delta$ is moving, excluding *emerged* cells that have not yet been properly initialized.

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

- 547 1. if all four coarse cells are available, as in Figure 7a, we use the bilinear interpolation (26).
- 548 2. if three coarse cells are available, excluding the diagonal coarse cell [child.x, child.y], as
 549 in Figure 7b, we use the following triangular interpolation:

$$550 \quad s [] = (2 \text{ coarse } (s []) + \text{ coarse } (s [\text{child}.x]) + \text{ coarse } (s [0, \text{child}.y])) / 4. \quad (28)$$

- 551 3. if only three coarse cells are available, including the diagonal coarse cell [child.x, child.y],
 as in Figure 7c, we use the following diagonal interpolation:

$$552 \quad s [] = (3 \text{ coarse } (s []) + \text{ coarse } (s [\text{child}.x, \text{child}.y])) / 4. \quad (29)$$

- 553 4. if only one or two coarse cells are available, as in Figure 7d, we face a pathological
 554 situation. In this case, we compute the value of s in the halo cell using a Taylor expansion,
 555 where, if possible, one simple face gradient $\bar{\nabla}_d^f s$ at level l is computed per dimension,
 556 preferably along the faces of the parent cell that follow the natural orientation of the halo
 cell with respect to its parent cell:

$$557 \quad s [] = \text{coarse } (s []) + \frac{\Delta}{4} \text{coarse } (\bar{\nabla}_x^f s [1]) + \frac{\Delta}{4} \text{coarse } (\bar{\nabla}_y^f s []). \quad (30)$$

558 Note that in equation (30), the face gradient $\text{coarse } (\bar{\nabla}_y^f s [])$ does not follow the natural
 559 orientation of the halo cell with respect to its parent cell represented in Figure 7d.

560 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-centers>

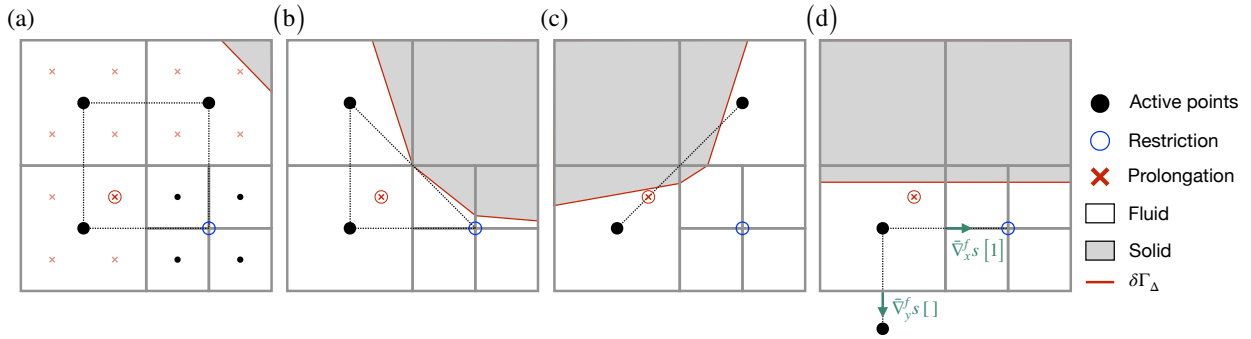


Figure 7: Graphical representation of the prolongation of a cell-centered scalar in the *halo* cell marked by a red circled cross, in the presence of embedded boundaries.

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

- 566 1. if four children cells are available, as in Figure 8a, we use the simple diagonal averaging
 567 (27).
- 568 2. if three children cells are available, as in Figure 8b, we also use diagonal averaging, but
 569 only along the available diagonal:

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

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

$$s[] = \bar{s} - \mathbf{b} \cdot \nabla s[] \quad (32)$$

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

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

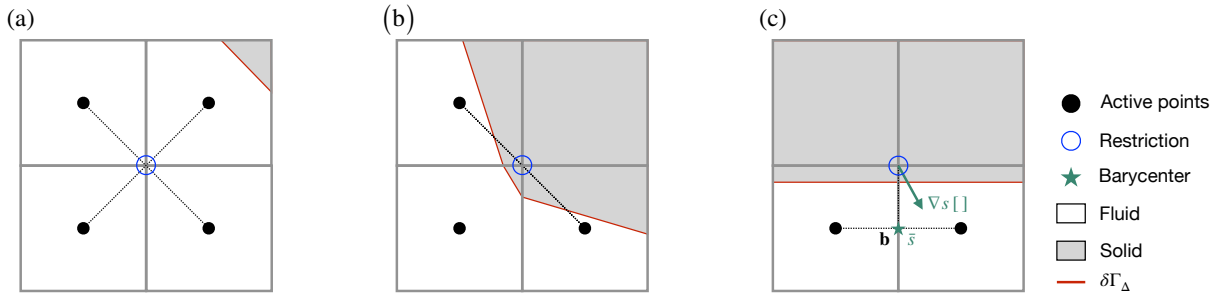


Figure 8: Graphical representation of the restriction of a cell-centered scalar in the *halo* cell marked by a large blue circle, in the presence of embedded boundaries.

4.2.3. Adaptive mesh refinement of quad/oct-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 $l - 1$:

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

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

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

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.

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

611 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:

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

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

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

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

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

618 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>.

621 5.2. Hydrodynamic force and torque

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

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

628 We then compute the viscous contribution to the force and torque using equation (19),
 629 assuming that the velocity \mathbf{u}^n is constant along the discrete rigid boundary $\delta\Gamma_\Delta^n$, i.e. $\nabla \mathbf{u}|_{\delta\Gamma_\Delta^n} \cdot$
 630 $\bar{\mathbf{t}}_\Gamma^n = 0$, where $\bar{\mathbf{t}}_\Gamma^n$ is the tangential vector to the discrete rigid boundary in the cut-cell.

631 Complete details can be found here: <http://basilisk.fr/src/embed.h#surface-force-and-vorticity>

632 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), $\forall x \in \Omega_\Delta/\Gamma_\Delta^{n+1}$:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + \mathbf{A}^{n+\frac{1}{2}} = 0 \quad (36a)$$

$$\rho \frac{\mathbf{u}^{**} - \mathbf{u}^*}{\Delta t} = \nabla \cdot [2\mu \mathbf{D}^{**}] \quad (36b)$$

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

$$\mathbf{u}^{n+1} = \mathbf{u}^{**} - \frac{\Delta t}{\rho} \nabla p^{n+1}, \quad (36d)$$

633 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
 634 we detail in Section 5.7.

635 Following the projection method described in [45], we enforce the incompressibility con-
 636 straint (1b) using equations (36c) and (36d). In practice, since both the velocity \mathbf{u} and the
 637 pressure p are collocated at center of each cell, we compute the divergence of the velocity
 638 $\nabla \cdot \mathbf{u}^{**}$ in equation (36c) using an auxiliary face velocity \mathbf{u}^f and the divergence operator (17).
 639 We therefore impose the incompressibility condition (1b) on the auxiliary face velocity \mathbf{u}^f
 640 and the cell-centered velocity \mathbf{u} , computed using equation (36d), is then only approximately
 641 incompressible.

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

644 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
 $\delta\Gamma_\Delta^{n+1}$:

$$\mathbf{u}^{n+1} = \mathbf{u}_\Gamma^{n+1} \quad (37a)$$

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

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

653 5.5. Submerged and emerged cells

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

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

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

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

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

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

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

681 Complete details of the treatment of emerged cells can be found here: [http://basilisk.
 682 fr/sandbox/ghigo/src/myembed-moving.h#prediction](http://basilisk.fr/sandbox/ghigo/src/myembed-moving.h#prediction).

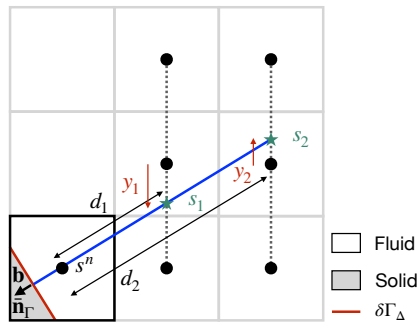


Figure 9: Graphical representation of the methodology implemented in *Basilisk* to extrapolate the value s^n at the center of an emerged cut-cell of a 2D grid when the principal direction of the normal vector $-\bar{\mathbf{n}}_\Gamma^{n+1}$ is the positive x -direction.

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

685 *5.6. Multigrid solver*

686 We solve each component of the velocity \mathbf{u} in equation (36b) and the pressure p in equation
 687 (36c) using a multigrid Poisson-Helmholtz solver. Indeed, multigrid solvers are known for being
 688 efficient at solving elliptic or parabolic problems such as equations (36b) and (36c), which can
 689 be written in a more general form as the following Poisson-Helmholtz equation:

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

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

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

$$\mathcal{L}(ds) = b - \mathcal{L}(\tilde{s}) = \text{res}, \quad (40)$$

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

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

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

708 The error ds is then efficiently reduced on each grid, starting from the coarsest grid at level
 709 $l = 1$, using only a few iterations of simple relaxation techniques such as Jacobi or Gauss-
 710 Seidel. These relaxation techniques rely on the discretization of the Laplacian operator $\nabla \cdot (\nabla)$
 711 in each cell of the grid Ω_Δ . In each full cell, we proceed as in [50]. In each cut-cell, we
 712 simply use equation (17) to discretize the divergence operator $\nabla \cdot$ and equations (19) and (24)
 713 to respectively compute the flux $F_\Gamma = \nabla_\Gamma s$ through the discrete embedded boundary and the
 714 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
 715 equation (40) by the volume fraction c so that the resulting linear system is well-conditioned
 716 for any value of the volume fraction [60].

717 Complete details of the multigrid Poisson-Helmholtz solver can be found here: [http://](http://basilisk.fr/src/poisson.h)
 718 basilisk.fr/src/poisson.h.

719 5.7. Nonlinear advection equation

720 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 722 the following *integral* form of the nonlinear advection equation for the velocity, written in conservative form using the incompressibility condition (1b): 724

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

725 where $\mathbf{F} = \mathbf{u} \otimes \mathbf{u}$. Using the Leibniz integration rule and the divergence theorem, we rewrite equation (41) as: 726

$$\Delta^D (c \mathbf{u})_{t^n}^{t^{n+\frac{1}{2}}} - \int_{t^n}^{t^{n+\frac{1}{2}}} dt \int_{\delta\mathcal{V}(t)} (\mathbf{u} (\mathbf{u}_{\delta\mathcal{V}} \cdot \mathbf{n})) dA + \int_{t^n}^{t^{n+\frac{1}{2}}} dt \int_{\mathcal{V}(t)} \nabla \cdot \mathbf{F} dV = 0, \quad (42)$$

727 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: 728

$$\Delta^D (c \mathbf{u})_{t^n}^{t^{n+\frac{1}{2}}} - \int_{t^n}^{t^{n+\frac{1}{2}}} dt \int_{\delta\Gamma_\Delta(t)} (\mathbf{u}_\Gamma (\mathbf{u}_\Gamma \cdot \mathbf{n})) dA + \int_{t^n}^{t^{n+\frac{1}{2}}} dt \int_{\mathcal{V}(t)} \nabla \cdot \mathbf{F} dV = 0. \quad (43)$$

729 Then, assuming that the velocity \mathbf{u}_Γ of the rigid body Γ is constant along the discrete boundary $\delta\Gamma_\Delta(t)$ in the cut-cell, we obtain: 730

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

731 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: 732 733

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

734 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. 735 736 737

738 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 740 741 742 743 evaluated at time t^{n+1} and not time $t^{n+\frac{1}{2}}$.

744 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 745 (45) here: <http://basilisk.fr/sandbox/ghigo/src/myembed.h#lifting-the-small-cell-cfl-restr> 746

747 *5.8. CFL condition and small cell restriction*

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

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

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

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

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

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

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

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

769 where the neighborhood \mathcal{N} corresponds to all full and cut-cells in the 3×3 ($3 \times 3 \times 3$ in
 770 3D) stencil of the cut-cell. We then update the velocity in each cut-cell using the following
 771 interpolation between $\mathbf{A}_c^{n+\frac{1}{2}}$ and $\mathbf{A}_{\text{nc}}^{n+\frac{1}{2}}$:

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

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

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

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

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

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

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

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

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

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

784 6. Numerical validation

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

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

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

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

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

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

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

806 6.1. Poisson-Helmholtz equation with Dirichlet boundary conditions in a domain defined by a
 807 2D rhodonea curve

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

$$\nabla \cdot \nabla s = 7r^2 \cos 3\theta, \quad (54)$$

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

$$r \leq 0.3 + 0.15 \cos 6\theta. \quad (55)$$

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

$$s(r, \theta) = r^4 \cos 3\theta. \quad (56)$$

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

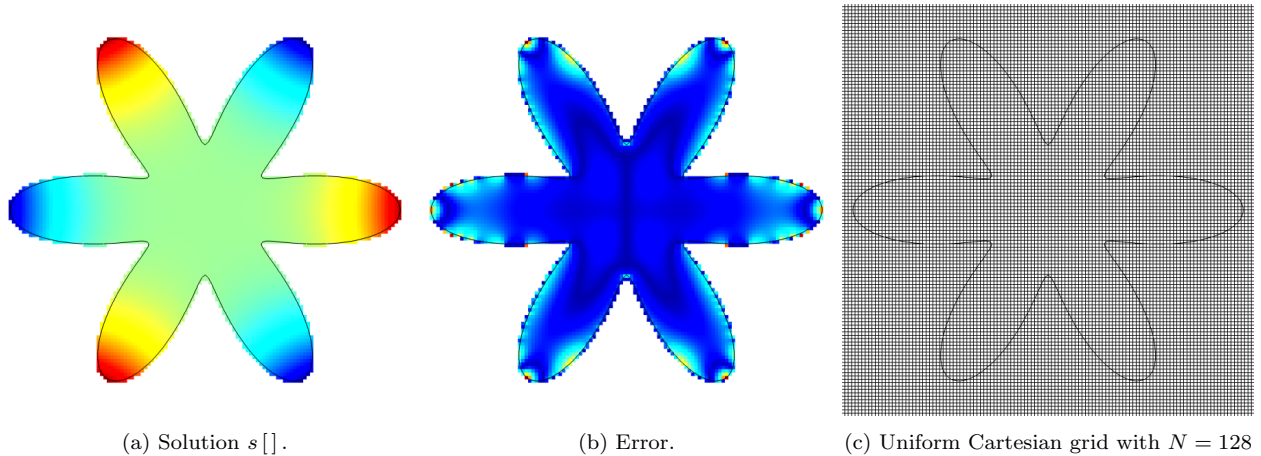


Figure 10: Solution to the Poisson-Helmholtz equation (54) computed on a uniform Cartesian grid with $N = 128$ cells and using Dirichlet boundary conditions: (a) cell-centered solution; (b) cell-centered error with the exact solution (56) and (c) uniform Cartesian grid. <http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c>.

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

822 To further assess the accuracy of the multigrid solver, we plot in Figure 11 the evolution
 823 with the number of cells N of the 1-norm (see Figures 11a and 11b) and ∞ -norm (see Figures
 824 11c and 11d) of the error between the computed and exact solutions. We observe that the

825 error in cut-cells converges for all values of N at the rate $n \approx 3$, higher than the expected
 826 second-order accuracy of the multigrid solver. This behavior was also observed in [60] and
 827 was explained by a *dipole* behavior of the solution in cut-cells when using Dirichlet boundary
 828 conditions. However, even though the error in full cells first converges at the same rate $n \approx 3$
 829 for coarser grids ($32 \leq N \leq 256$), it eventually converges at the expected convergence rate
 830 $n \approx 2$ for finer grids ($512 \leq N \leq 2048$). Indeed, as observed in Figures 11a and 11c, the error
 831 for coarser grids is larger in cut-cells and therefore affects the convergence rate of the error in
 832 full cells.

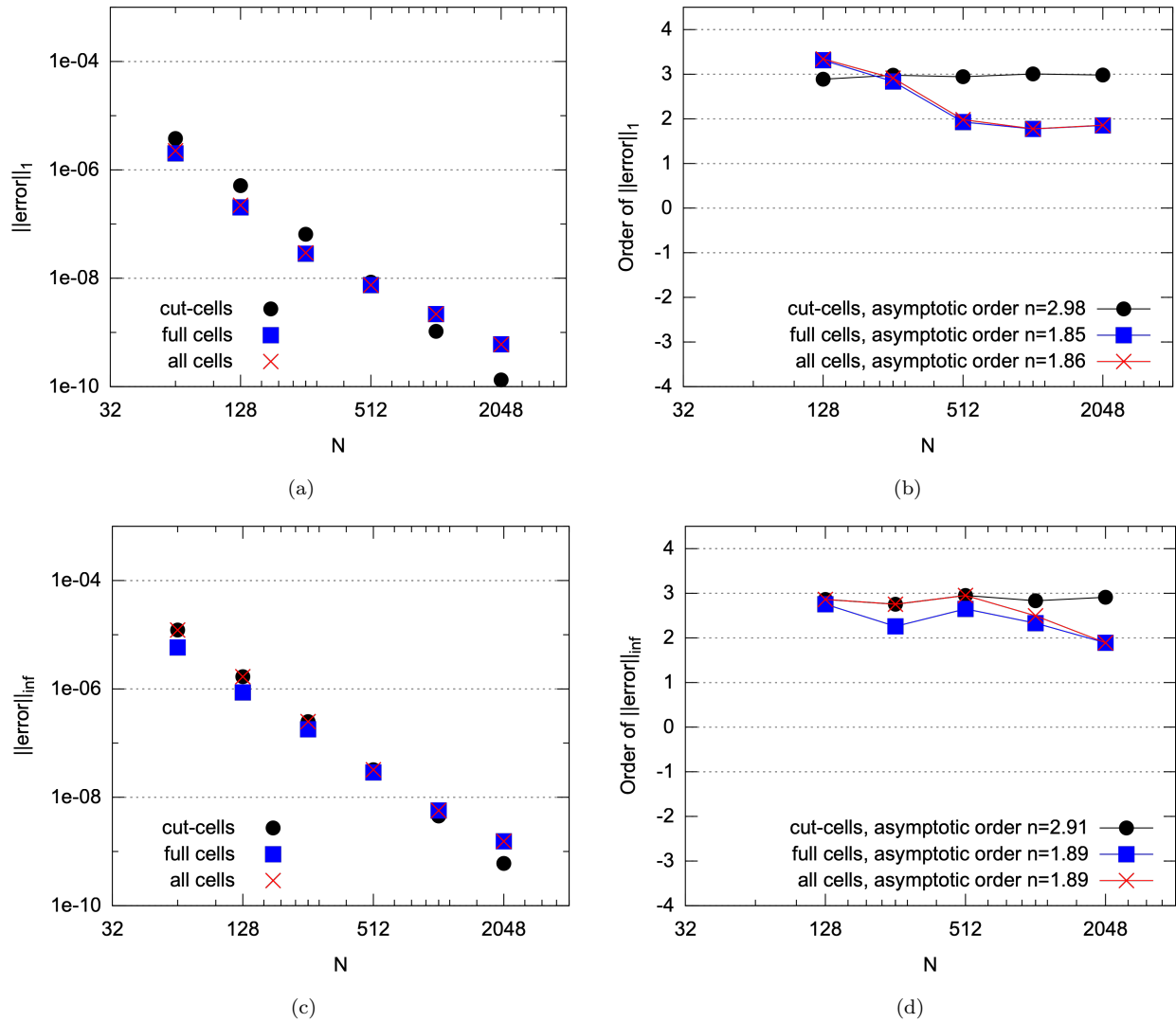


Figure 11: Evolution with the number of cells N of the error between the computed and exact solutions of the Poisson-Helmholtz equation (54) on a uniform Cartesian grid and using Dirichlet boundary conditions: (a) and (b): $\|error\|_1$ and the corresponding convergence rate; (c) and (d): $\|error\|_{\infty}$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

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

835 12a the solution, computed on the quadtree defined by $N_{\min} = 32$ and $N_{\max} = 128$ displayed
 836 in Figure 12c, and the corresponding error in Figure 12b. We observe that the solution closely
 837 resembles the one we obtain on a uniform grid in Figure 10a. We also notice in Figure 12b
 838 that, contrary to Figure 10b, the largest error now occurs in the core of the domain, where the
 839 grid is coarser.

840 In Figure 13, we then plot the evolution with the maximum number of cells N_{\max} of the
 841 1-norm (see Figures 13a and 13b) and ∞ -norm (see Figures 13c and 13d) of the error between
 842 the computed and exact solutions. We observe that the error in cut-cells and full cells converges
 843 at the rate $n \approx 2$ (the ∞ -norm of the error in cut-cells converges at an average rate $n = 1.77$).
 844 We therefore do not recover the convergence rate $n \approx 3$ observed previously in cut-cells of a
 845 uniform grid. Indeed, due to local mesh refinement near the embedded boundary, we use values
 846 in halo cells to compute face gradients in cut-cells. These halo cell values are computed using
 847 the second-order restriction and prolongation functions defined in Section 4.2.2, which break
 848 the dipole behavior of the solution. Furthermore, due to the complex shape of the embedded
 849 boundary defined by equation (55), pathological situations for the restriction function are likely
 850 to arise. As a reminder, these occur when two or less cells are available for the restriction
 851 function. In this case, we maintain the convergence rate $n \approx 2$ observed in Figure 13 by
 852 providing an *a priori* value for the gradient ∇s , derived from the exact solution (56) and used
 853 in equation (32).

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

860 Complete details of this test case can be found here: [http://basilisk.fr/sandbox/ghigo/src/test-poisson/](http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c)
 861 [neumann.c](http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c).

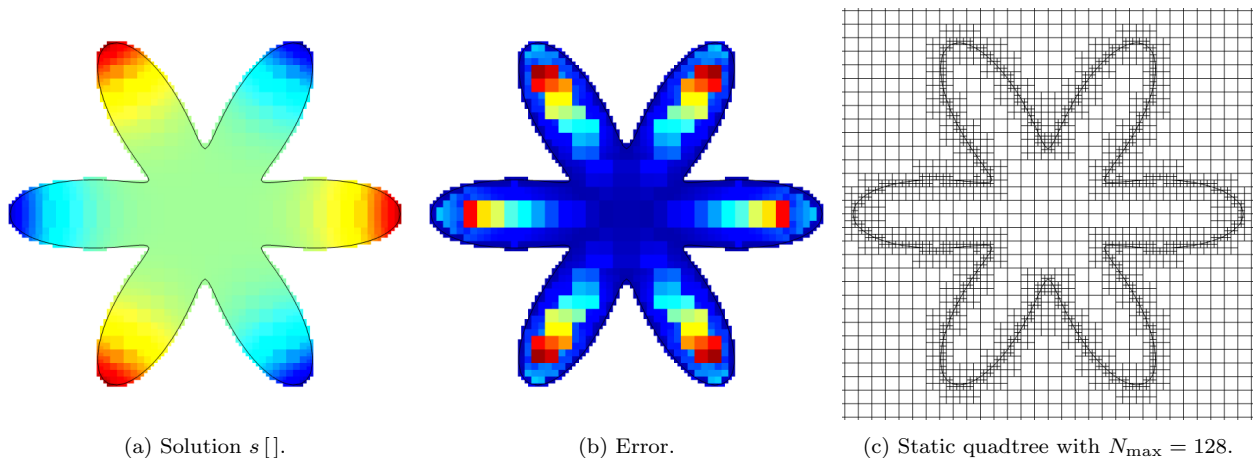


Figure 12: Solution to the Poisson-Helmholtz equation (54) computed on a static quadtree with $N_{\max} = 128$ and $N_{\min} = 32$ and using Dirichlet boundary conditions: (a) cell-centered solution; (b) cell-centered error with the exact solution (56) and (c) static quadtree. <http://basilisk.fr/sandbox/ghigo/src/test-poisson/neumann.c>.

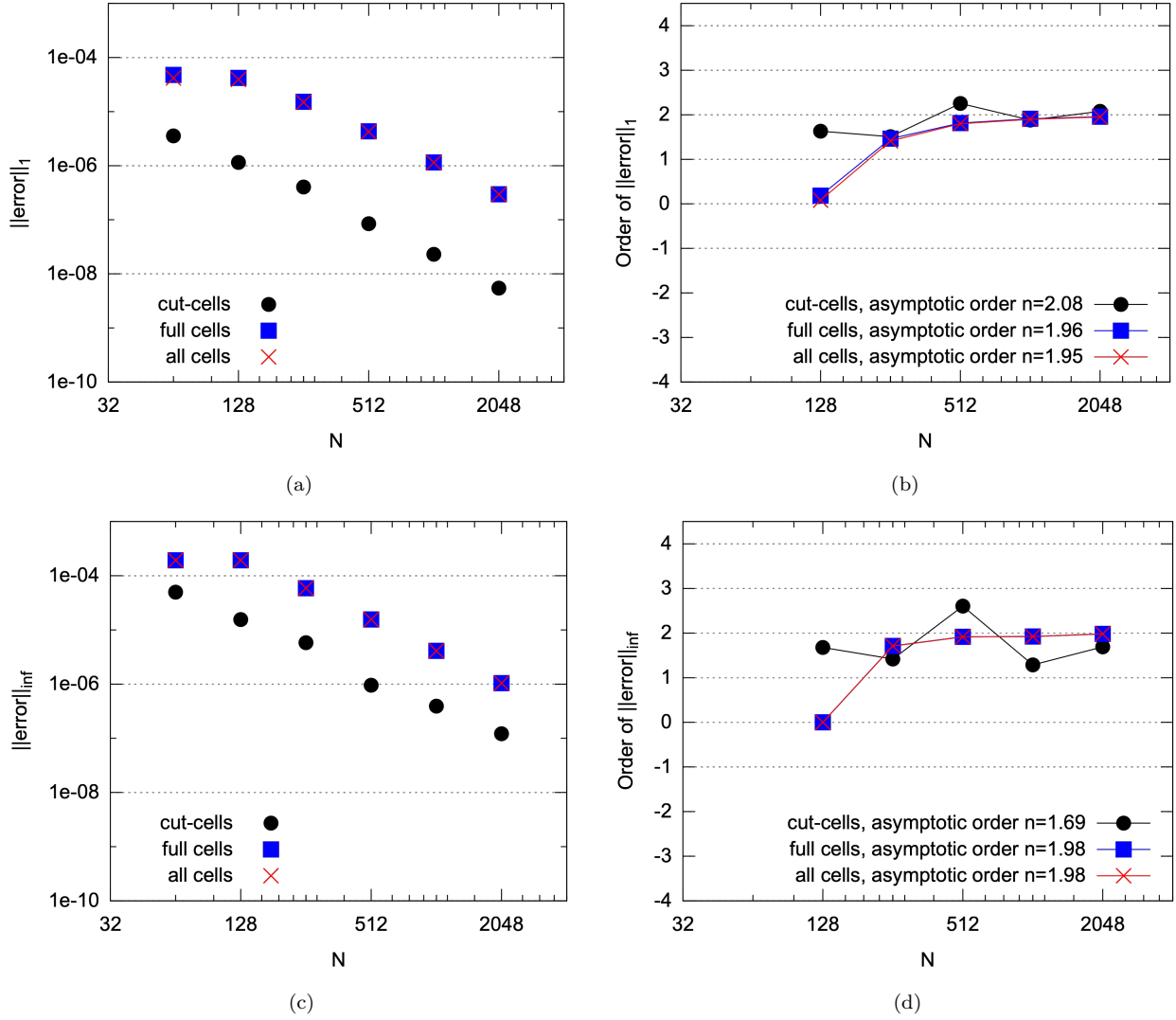


Figure 13: Evolution with the number of cells N_{\max} of the error between the computed and exact solutions of the Poisson-Helmholtz equation (54) on a quadtree and using Dirichlet boundary conditions: (a) and (b): $\|err\|_1$ and the corresponding convergence rate; (c) and (d): $\|err\|_{\infty}$ and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

862 6.2. Heat equation with Neumann boundary conditions in a 3D expanding sphere

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

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

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

$$r^2 \leq 0.392 + t, \quad t \leq 0.1. \quad (58)$$

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

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

870 We solve equation (57) from time $t = 0$ to time $t = 0.1$ using an implicit first-order time
 871 discretization with $\Delta t = 10^{-5}$ to minimize time discretization and splitting errors. As the
 872 sphere expands, we initialize the scalar s in emerged cells using the second-order extrapolation
 873 described in Section 5.5. At each time step, we therefore solve a discrete Poisson-Helmholtz
 874 equation using the multigrid solver with a tolerance set to $\xi_{\text{mg}} = 10^{-6}$. Note that, following
 875 [61], we evaluate the right-hand-side of equation (57) at the centroid of each cell, whereas we
 876 compute the exact solution (59) at the center of each cell.

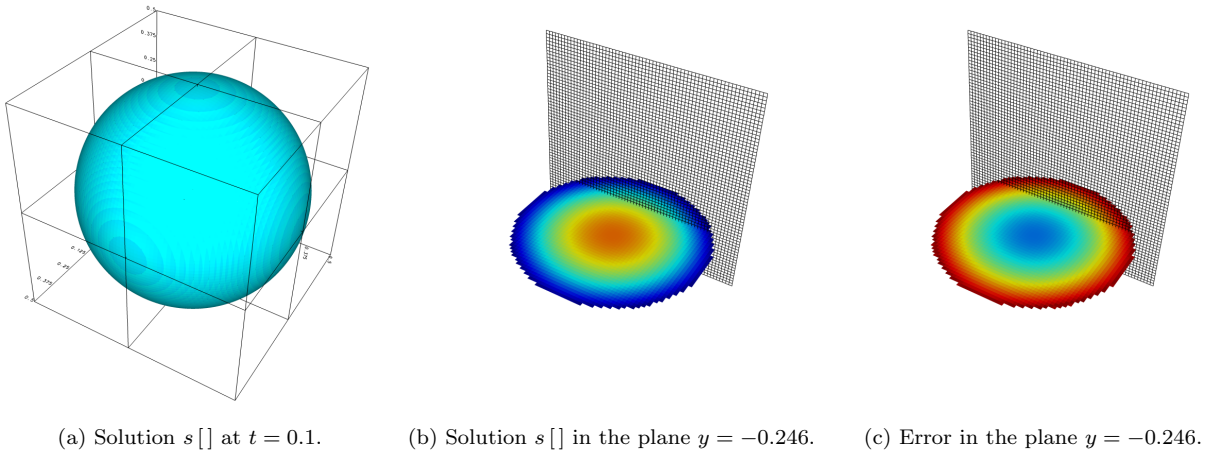


Figure 14: Solution to the heat equation (57) at time $t = 0.1$, computed on a uniform Cartesian grid with $N = 64$ cells and using Neumann boundary conditions: (a) cell-centered solution on the embedded boundary; (b) cell-centered solution in the $x-z$ plane defined by $y = -0.246$ and (c) cell-centered 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>.

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

881 In Figure 15, we then plot the evolution with the number of cells N of the 1-norm (see Figures
 882 15a and 15b) and ∞ -norm (see Figures 15c and 15d) of the error between the computed and
 883 exact solutions. As in [61], we observe in full cells that the 1-norm and the ∞ -norm of the
 884 error converge at the expected rate $n \approx 2$. In cut-cells, only the 1-norm of the error converges
 885 at a rate $n \approx 2$, whereas the ∞ -norm of the error converges at an average rate $n \approx 1.33$.
 886 This behavior is notably different from the one observed in Section 6.1, where we obtain a
 887 convergence rate $n \approx 3$ in cut-cells. The reason is twofold: (i) when using Neumann boundary
 888 conditions, the convergence rate of the multigrid Poisson-Helmholtz solver reduces to $n = 2$ in
 889 cut-cells [60, 45]; (ii) as the sphere expands isotropically, neighboring emerged cells are likely

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

895 Complete details of this test case can be found here: <http://basilisk.fr/sandbox/ghigo/src/test-heat/neumann3D.c>.
 896

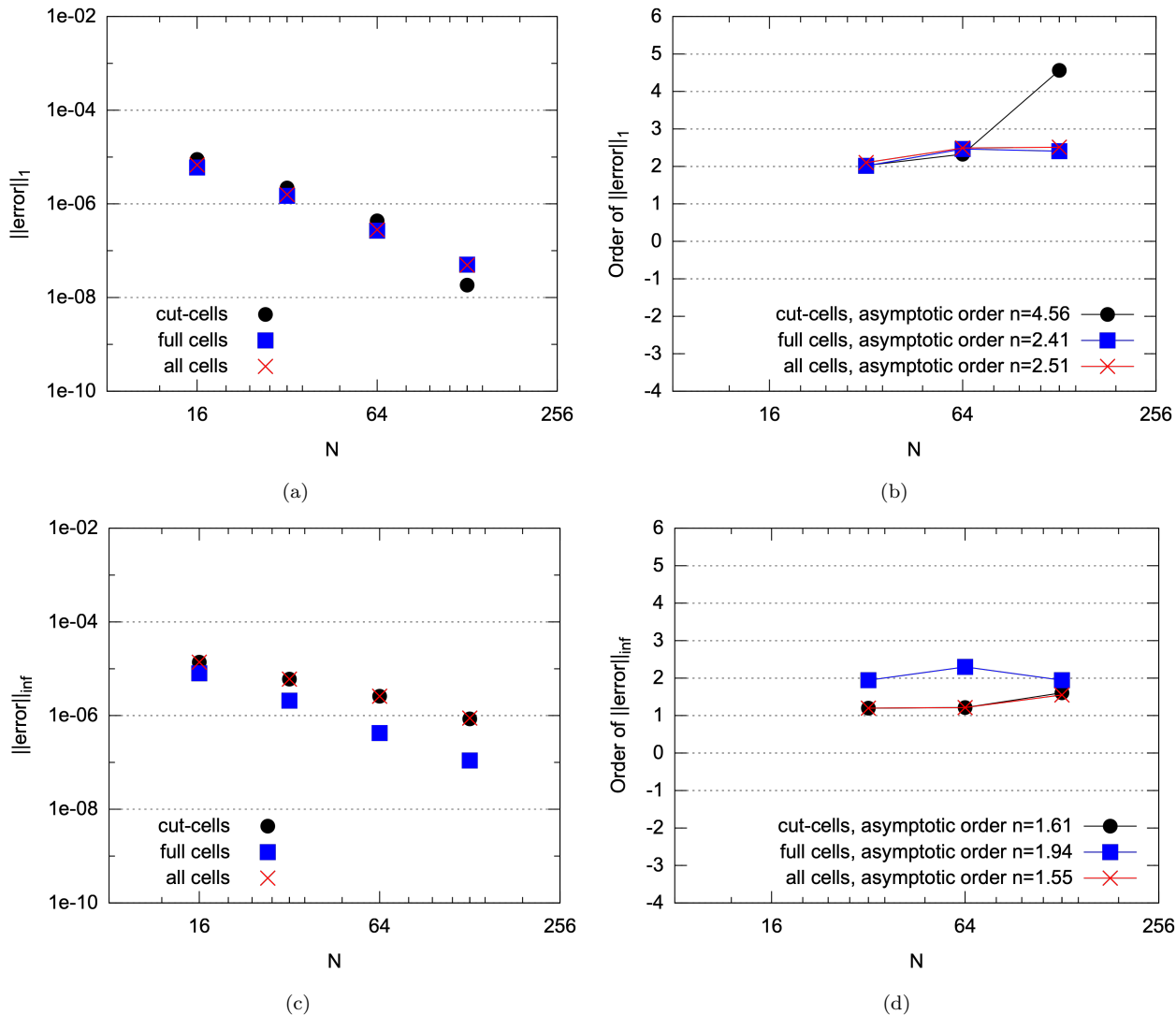


Figure 15: Evolution with the number of cells N of the error between the computed and exact solutions of the heat equation (57) on a uniform Cartesian grid and using Neumann boundary conditions. (a) and (b): $\|\text{error}\|_1$ and the corresponding convergence rate; (c) and (d): $\|\text{error}\|_{\infty}$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

897 6.3. Pressure-driven Stokes flow through a porous medium arbitrarily refined

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

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

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

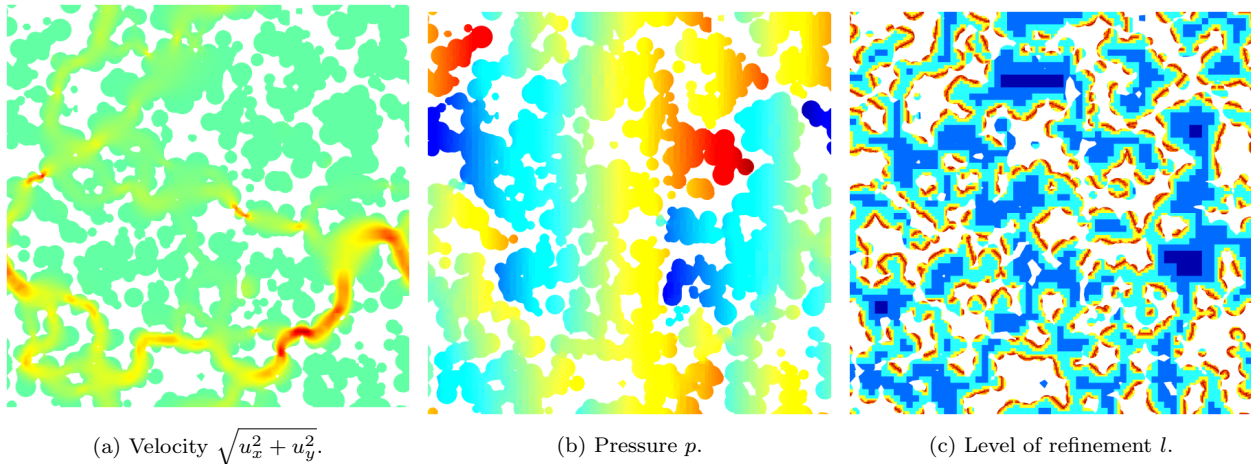


Figure 16: Pressure-driven Stokes flow in a 2D porous medium defined by the union of a random collection of disks and computed on a randomly refined quadtree defined by $N_{\min} = 2$ and $N_{\max} = 32$: (a) norm of the velocity $\sqrt{u_x^2 + u_y^2}$; (b) pressure p and (c) arbitrarily defined levels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

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

914 To further assess the performance of the multigrid solver, we plot in Figure 17 the evolution
 915 with the number of time steps n of several quantities related to the performance of the multigrid
 916 solver. In particular, we observe that the relative change in velocity $\|u_x^{n+1} - u_x^n\|_\infty / \|u_x^{n+1}\|_1$
 917 (black dots in Figure 17) rapidly converges towards zero, indicating that we obtain a steady
 918 solution. This is also highlighted by the near constant values of the 1-norm of the velocity $\|u_x\|_1$
 919 (green lozenge in Figure 17) and the ∞ -norm of the pressure $\|p\|_\infty$ (orange plus in Figure 17).
 920 Finally, we notice that the ∞ -norm of both the residual of the viscous Navier-Stokes equation
 921 (36b) $\|\text{res}_u\|_\infty$ (red cross in Figure 17) and of the pressure Poisson-Helmholtz equation (36c)
 922 $\Delta t \|\text{res}_p\|_\infty$ (blue square in Figure 17) rapidly converge below the tolerance ξ_{mg} of the multigrid
 923 solver, indicating its efficiency.

924 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
 925

926 similar performances (results not shown here).

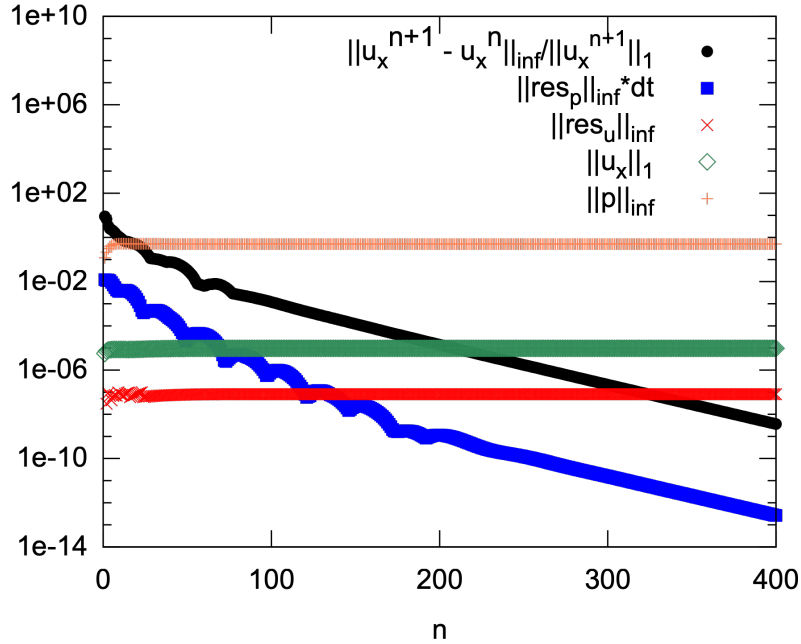


Figure 17: Evolution with the number of time steps n of several quantities related to the performance of the multigrid solver when solving a pressure-driven Stokes flow in a 2D porous medium. In particular, we plot the relative change in velocity $\|u_x^{n+1} - u_x^n\|_\infty / \|u_x^{n+1}\|_1$ (black 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>.

927 *6.4. Instabilities due to “third-order” Dirichlet boundary conditions at low Reynolds number*

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

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

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

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

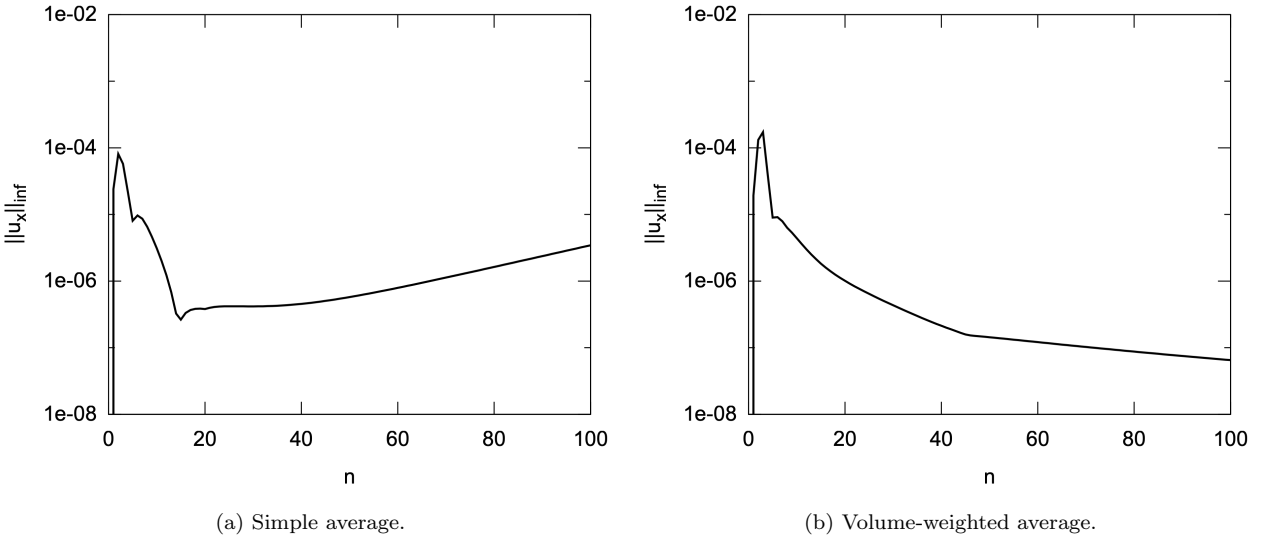


Figure 18: Evolution with the number of time steps n of the ∞ -norm of the horizontal velocity $\|u_x\|_\infty$ when solving a Stokes flow in a periodic channel: (a) unstable case where face values are computed using simple averaging and (b) stable case where face values are computed using the volume-weighted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

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

951 Complete details of this test case can be found here: [http://basilisk.fr/sandbox/](http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/uf.c)
 952 [ghigo/src/test-navier-stokes/uf.c](http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/uf.c).

953 6.5. Starting flow past a fixed and moving cylinder at $Re = 1000$

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

958 We compute here the highly inertial starting flow around a 2D cylinder at the Reynolds
 959 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
 960 cylinder. This is a canonical case of complex boundary layer separation [79, 80, 52], where a
 961 high spatial resolution is needed to properly resolve the boundary layers around the cylinder.

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

- 964 1. a starting flow passed a fixed cylinder, characterized by the constant far-field velocity
 965 $\mathbf{u}_\infty = [u_{\text{ref}}, 0]^\top$ and referred to as the *fixed cylinder case*;

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

968 In both cases, we consider a domain of size $L_0 = 18$ and represent only half of the cylinder,
 969 initially located in the middle of the bottom boundary of the domain. We therefore impose
 970 symmetry boundary conditions on the bottom boundary, a slip boundary condition on the top
 971 boundary and an outflow boundary condition on the right boundary. On the left boundary, we
 972 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
 973 $p = 0$, we then solve the Navier-Stokes equations (36) on a quadtree dynamically adapted at
 974 each time step based on the velocity \mathbf{u} with an adaptation criteria $\xi_{\text{adapt}}/u_{\text{ref}} = 10^{-3}$. Following
 975 [80], we choose the maximum level of refinement $l_{\text{max}} = 12$ (228 pt/d) to obtain a minimum
 976 resolution $d/\Delta_{\text{min}} \approx 10\sqrt{Re}$. We also set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-6}$.
 977 Finally, the time step is bounded by $\Delta t/(d/u_{\text{ref}}) \leq 10^{-3}$.

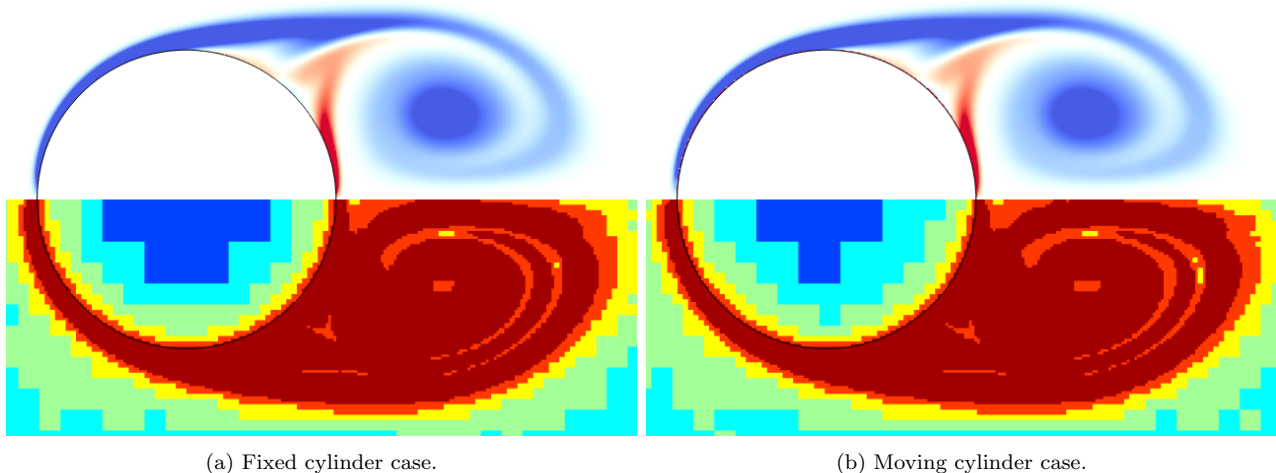


Figure 19: Vorticity ω_z (top half) and the corresponding spatial distribution of the level of refinement (bottom half) at $t/(d/u_{\text{ref}}) = 3$ for the impulsively started flow around a cylinder at $Re = 1000$. <http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c> and <http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c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

978 In Figure 19, we plot at time $t/(d/u_{\text{ref}}) = 3$ for both the fixed cylinder case (see Figure
 979 19a) and the moving cylinder case (see Figure 19b) the vorticity ω_z (top half) as well as the
 980 corresponding spatial distribution of the level of refinement l around the cylinder (bottom
 981 half). By visually comparing the vorticity structures in each case, we notice that they are
 982 almost identical and are in very good qualitative agreement with the vorticity plots displayed
 983 in Figure 16 of [79] and Figure 3 of [80]. Furthermore, we observe in both cases that the mesh
 984 is refined in the regions of strong vorticity while the core of the cylinder (which does not belong
 985 to the computational domain) and the far field regions are represented with coarser cells. This
 986 significantly reduces the number of cells required to compute an accurate solution (see note on
 987 performance below).

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

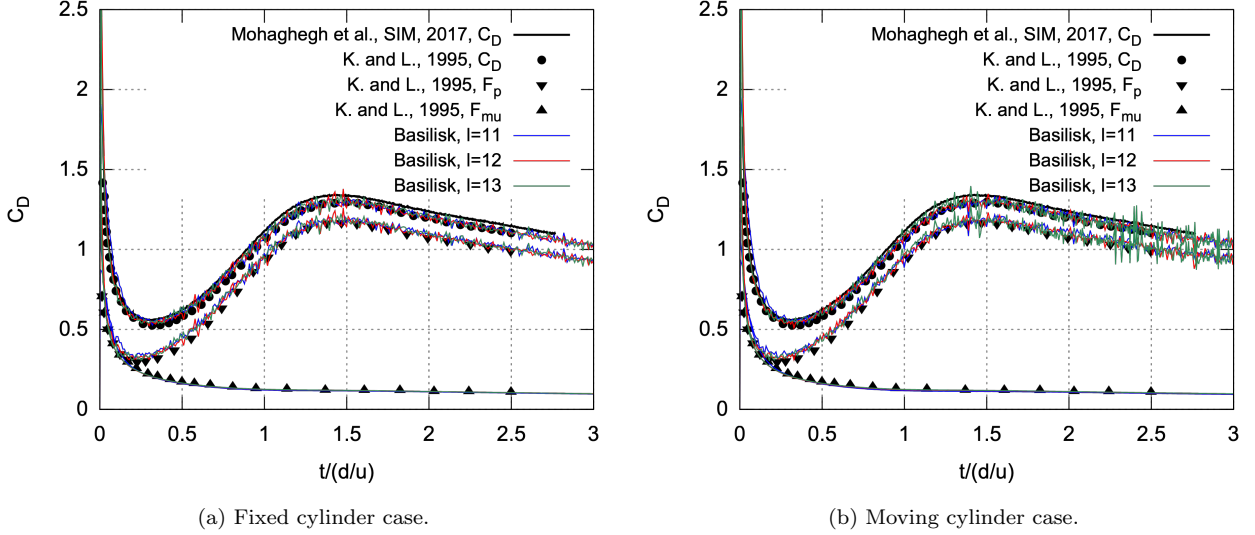


Figure 20: Time evolution of the hydrodynamic forces acting on the cylinder impulsively started at $Re = 1000$, computed on three different adaptive quadrees defined by the following values of the maximum level of refinement $l_{\max} = \{11 (114 \text{ pt/d}), 12 (228 \text{ pt/d}), 13 (455 \text{ pt/d})\}$.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

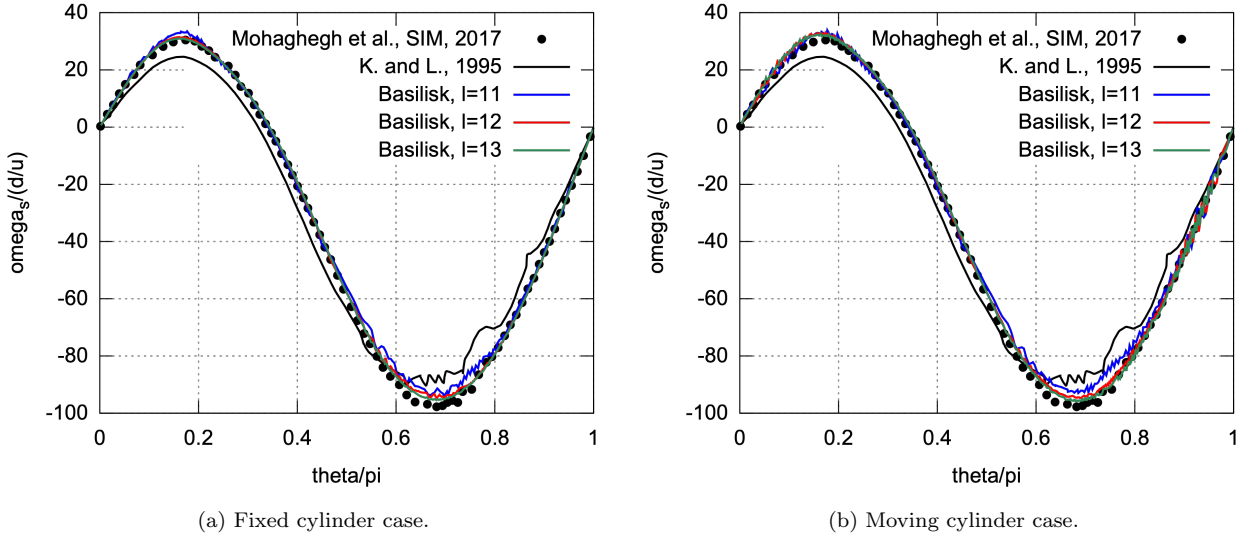


Figure 21: Distribution of the vorticity ω_z around the cylinder impulsively started at $Re = 1000$, computed at time $t/(d/u_{\text{ref}}) = 0.5$ on three different adaptive quadrees defined by the following values of the maximum level of refinement $l_{\max} = \{11 (114 \text{ pt/d}), 12 (228 \text{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

990 maximum level of refinement $l_{\max} = \{11 (114 \text{ pt/d}), 12 (228 \text{ pt/d}), 13 (455 \text{ pt/d})\}$. In Figures
 991 20a and 20b, we plot respectively for the fixed and moving cylinder cases the time evolution
 992 of the streamwise component of the pressure and viscous forces $F_{p,x}$ and $F_{\mu,x}$ acting on the

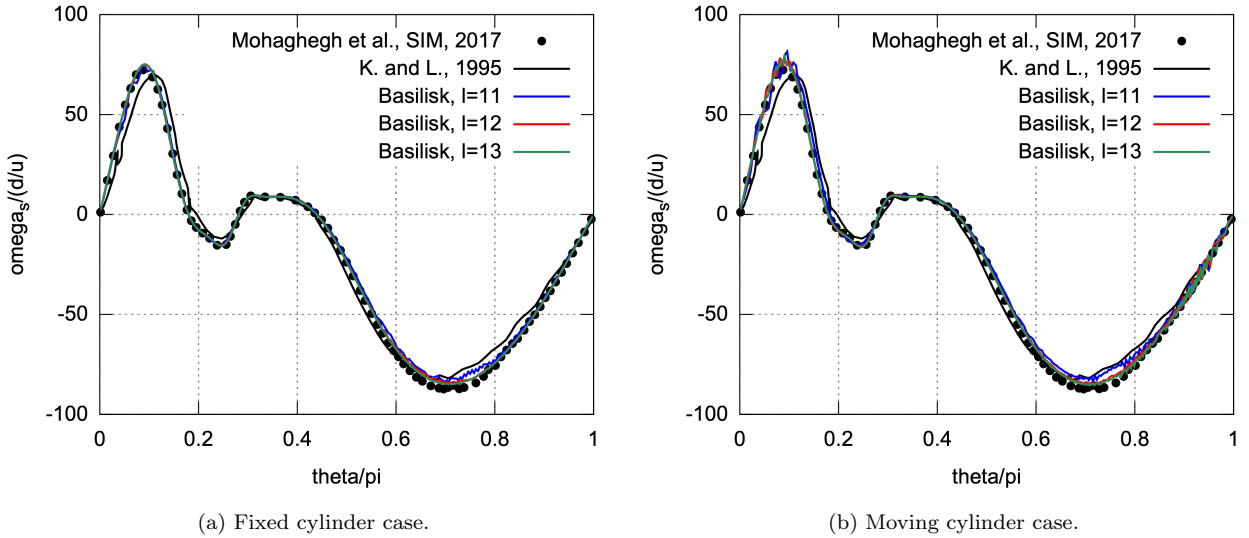


Figure 22: Distribution of the vorticity ω_z around the cylinder impulsively started at $Re = 1000$, computed at time $t/(d/u_{\text{ref}}) = 2.5$ on three different adaptive quadtrees defined by the following values of the maximum level of refinement $l_{\text{max}} = \{11 \text{ (114 pt/d)}, 12 \text{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

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

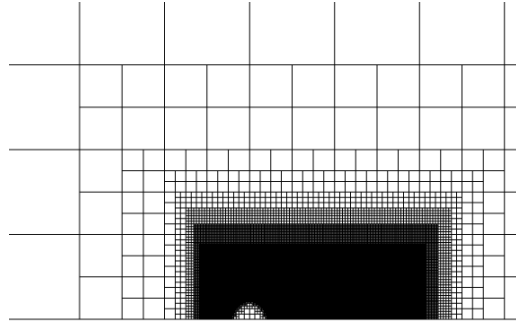
$$C_D = \frac{F_{p,x} + F_{\mu,x}}{\frac{1}{2}\rho u_{\text{ref}} S_{\text{ref}}}, \quad \text{with} \quad S_{\text{ref}} = \frac{d}{2}. \quad (60)$$

995 In Figures 21a and 21b and Figures 22a and 22b, we plot respectively for the fixed and moving
 996 cylinder cases the distribution of the vorticity ω_z on the surface of the cylinder obtained at times
 997 $t/(d/u_{\text{ref}}) = \{0.5, 2.5\}$. For each of these quantities, we compare our results to those presented
 998 in [79, 80] and observe that for both the fixed and moving cylinder cases the computed results
 999 converge towards the reference data as we increase the value of the maximum level of refinement
 1000 l_{max} . In particular, the time evolution of the viscous force $F_{\mu,x}$ in Figures 20a and 20b perfectly
 1001 matches the reference data for all considered values of l_{max} . We note however that the time
 1002 evolution of the pressure force $F_{p,x}$ in Figures 20a and 20b is noisier than the reference data
 1003 and that the noise is not significantly reduced when we increase the value of l_{max} .

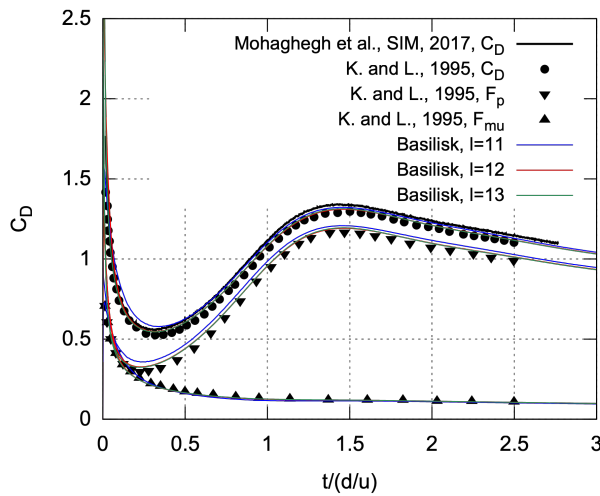
1004 Indeed, as the cell-centered velocity \mathbf{u} is not exactly incompressible, the pressure p feels
 1005 through the term $\nabla \cdot \mathbf{u}^{**}$ in equation (36c) the history of the divergence of the velocity. And in
 1006 this case, the history of the divergence contains noise induced by the dynamic mesh adaptation
 1007 at every time step.

1008 To corroborate this explanation, we plot in Figure 23 the time evolution of the hydrodynamic
 1009 forces obtained for parameters identical to those used to obtain the results in Figure 20 but
 1010 computed on the static quadtree presented in Figure 23a. We observe that in this case, for both
 1011 the fixed (see Figure 23b) and moving (see Figure 23c) cylinder cases, the time evolution of the
 1012 pressure force $F_{p,x}$ no longer contains any noise. This is a clear indication that the adaptive
 1013 mesh refinement along with the approximate projection method are responsible for the noise
 1014 observed in the pressure force $F_{p,x}$ in Figure 20.

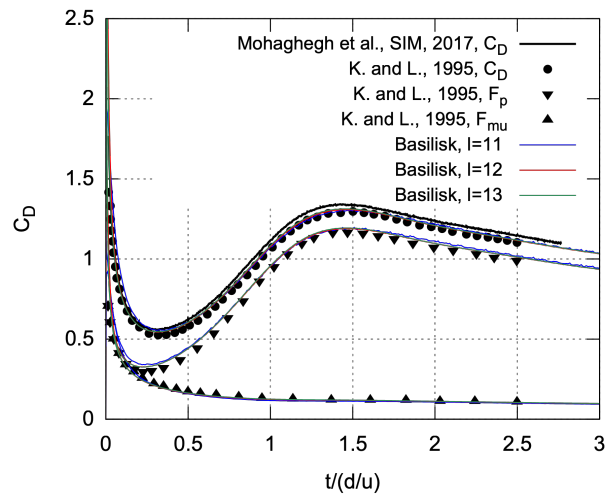
1015 Complete details of these two test cases can be found here: <http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting.c> and <http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/starting-moving.c>.
 1016
 1017



(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 $l_{\max} = \{11 (114 \text{ pt/d}), 12 (228 \text{ pt/d}), 13 (455 \text{ pt/d})\}$ and illustrated in (a) for $l_{\max} = 12 (228 \text{ pt/d})$: (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>.

1018 *A note on performance.* Performance, ideally measured by the wall-clock runtime for a given
 1019 accuracy, is a key parameter in the development and evaluation of numerical methods, and
 1020 this is especially true for adaptive mesh refinement techniques. We therefore compare here the
 1021 performances of the Cartesian grid embedded boundary method for both the fixed and moving
 1022 cylinder cases on three different grids: (i) a uniform Cartesian grid, (ii) an adaptive quadtree
 1023 and (iii) a static quadtree as described above.

1024 Indeed, a remarkable feature of *Basilisk* is that the definition of a cell on both a uniform
 1025 grid and a tree grid is identical. This is achieved in *Basilisk* by separating the low-level details

1026 of the different grid implementations (e.g. memory layout, grid traversal strategies) from the
 1027 numerical scheme itself. Therefore, *Basilisk* provides efficient Cartesian grid and quadtree im-
 1028 plementations and a comparison of the performances between both grids is therefore warranted.

Grid	Uniform Cartesian grid		Adaptive quadtree		Static quadtree	
	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 $l_{\text{max}} = 11$. 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>.

1029 In Table 1, we summarize the performances of the fixed and moving embedded boundary
 1030 algorithms. In both cases, we run the simulation until $t/(d/u_{\text{ref}}) = 2.5$ using a maximum
 1031 level of refinement $l_{\text{max}} = 11$ on a single Intel i5 processor. We notice that, as expected, the
 1032 fixed embedded boundary algorithm is faster than its moving counterpart by a factor of ~ 1.3 .
 1033 However, the computational time for the moving embedded boundary algorithm is twice that
 1034 of the fixed one due to additional constraints on the time step ($\text{CFL} = 0.5$) which increase the
 1035 total number of time steps. Finally, we observe that the gain in number of grid points (i.e.
 1036 memory) obtained with adaptivity is approximately a factor of 400, while the gain in computing
 1037 time is roughly a factor of 200.

1038 6.6. Oscillating sphere in a quiescent flow

1039 We now reproduce a 3D test case taken from [81, 82, 49] and compute the inertial flow
 1040 induced by the forced inline sinusoidal oscillation of a sphere in an otherwise quiescent fluid.
 1041 The oscillating motion of the sphere is characterized by a Reynolds number $Re = \frac{u_{\text{ref}} d}{\nu} = 40$
 1042 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,
 1043 $d = 1$ is the diameter of the sphere and ω is the oscillation frequency.

1044 We consider a domain of size $L_0 = 16$, where we impose slip boundary conditions on all
 1045 boundaries and embed the sphere in the center of the domain. Starting from the initial condition
 1046 $\mathbf{u}(t = 0) = \mathbf{0}$ and $p = 0$, we impose the following forced horizontal motion to the sphere:

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

1047 We then solve the Navier-Stokes equations (36) until a periodic state is reached. We use an
 1048 octree dynamically adapted at each time step based on the velocity \mathbf{u} with an adaptation criteria
 1049 $\xi_{\text{adapt}}/u_{\text{ref}} = 10^{-2}$. We also set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-6}$. Finally, the
 1050 time step is bounded by $\Delta t/(d/u_{\text{ref}}) \leq 10^{-2}$. We observe that in this case a periodic state is
 1051 reached after one oscillation period $T = \omega/(2\pi)$.

1052 In Figure 24, we display snapshots of the mesh and vorticity in the xy -plane, taken at the
 1053 following phase angles $\theta = \{0, 96, 192, 288\}$ during the second oscillation period and computed

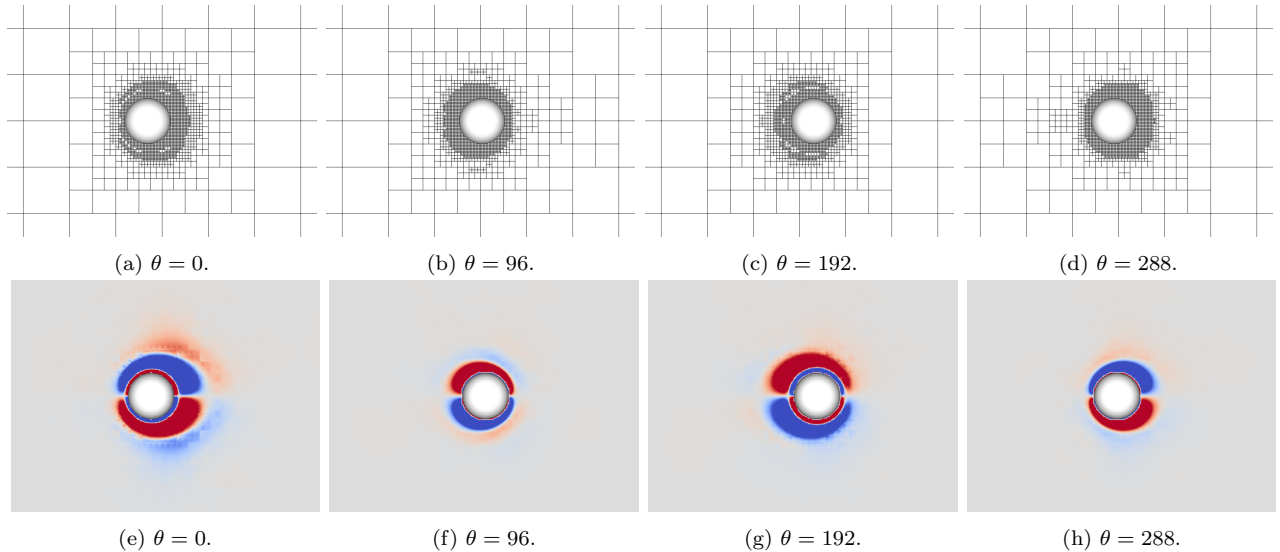


Figure 24: Snapshots of the mesh and the vorticity in the xy -plane, taken at the phase angles $\theta = \{0, 96, 192, 288\}$ during the second oscillation period of a sphere oscillating in a quiescent flow at $Re = 40$ and $St = 3.2$ on an adaptive octree defined by the following value of the maximum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

1054 on the octree defined by the following value of the maximum of level of refinement $l_{\max} =$
 1055 9 (32 pt/d). We observe that the mesh dynamically follows the movement of the sphere and
 1056 captures the resulting vorticity structures.

1057 We then plot in Figure 25 the time evolution during the second oscillation period of the
 1058 drag coefficient C_D defined by equation (60), with $S_{\text{ref}} = \pi \frac{d^2}{4}$. The results are computed
 1059 on three different octrees defined by the following values of the maximum level of refinement
 1060 $l_{\max} = \{8 \text{ (16 pt/d)}, 9 \text{ (32 pt/d)}, 10 \text{ (64 pt/d)}\}$. For all values of l_{\max} , we obtain a very good
 1061 agreement with the reference data from [49], computed using 50 pt/d, confirming the high
 1062 accuracy of our method.

1063 Complete details of this test case can be found here: [http://basilisk.fr/sandbox/](http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/sphere-oscillating.c)
 1064 [ghigo/src/test-navier-stokes/sphere-oscillating.c](http://basilisk.fr/sandbox/ghigo/src/test-navier-stokes/sphere-oscillating.c).

1065 6.7. Sphere of near-unity density ratio settling in a closed box

1066 Finally, we conclude the validation of the Cartesian grid embedded boundary method im-
 1067 plemented in *Basilisk* by solving the coupled system of equations (1)–(2) and (4)–(5) describing
 1068 the motion of a freely moving rigid body.

1069 We first investigate the settling at small to moderate Reynolds numbers of a sphere of
 1070 near-unity density ratio $\rho_{\Gamma}/\rho \approx 1$ in a closed box under the action of the gravity $\mathbf{g} =$
 1071 $[0, -9.81, 0]^{\top}$ m/s². This test case is inspired by the experimental and numerical work of [83],
 1072 where the authors characterized the flow using the Reynolds number $1.5 \leq Re = \frac{u_{\text{ref}} d}{\nu} \leq 31.9$
 1073 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
 1074 and $d = 0.015$ m the sphere diameter. This test case has since then been used on numerous
 1075 occasions as a validation case for fluid-particle coupling methods [84, 85, 86, 16] at small to
 1076 moderate Reynolds numbers.

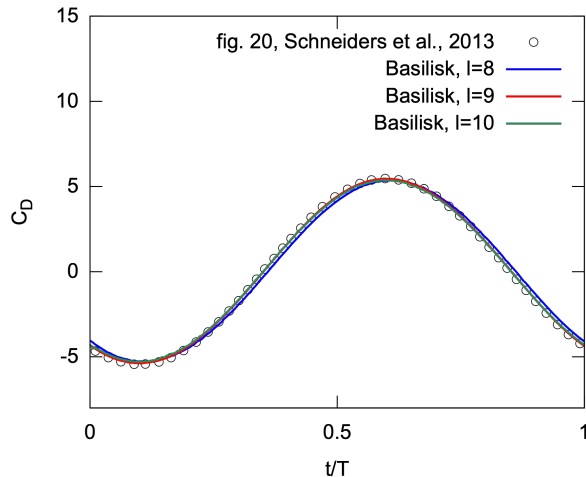


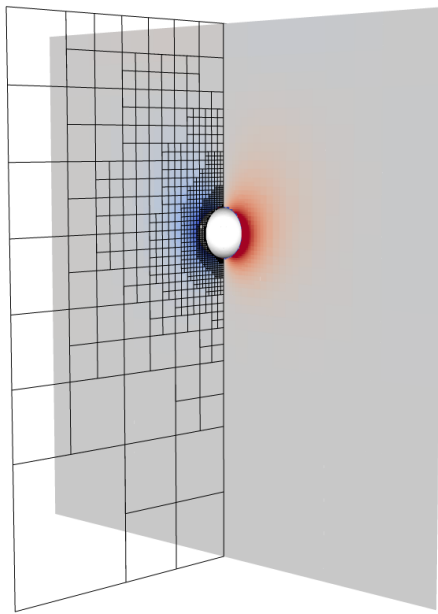
Figure 25: Time evolution during the second oscillation period of the drag coefficient C_D computed around a sphere oscillating in quiescent flow at $Re = 40$ and $St = 3.2$ on three different adaptive octrees defined by the following values of the maximum level of refinement $l_{\max} = \{8 \text{ (16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

1077 We consider a domain of size $L_0 = 0.16$ m, where we impose no-slip boundary conditions on
 1078 all boundaries and embed the sphere at a distance of 0.04 m between the bottom of the sphere
 1079 and the top boundary. Then, starting from the initial condition $\mathbf{u}(t = 0) = \mathbf{0}$ and $p = 0$, we
 1080 solve the coupled system of equations (1)–(2) and (4)–(5) on an octree, dynamically adapted
 1081 at each time step based on the velocity \mathbf{u} with an adaptation criteria $\xi_{\text{adapt}}/u_{\text{ref}} = 10^{-2}$. We
 1082 also set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-4}$. Finally, the time step is bounded
 1083 by $\Delta t / (d/u_{\text{ref}}) \leq 10^{-3}$.

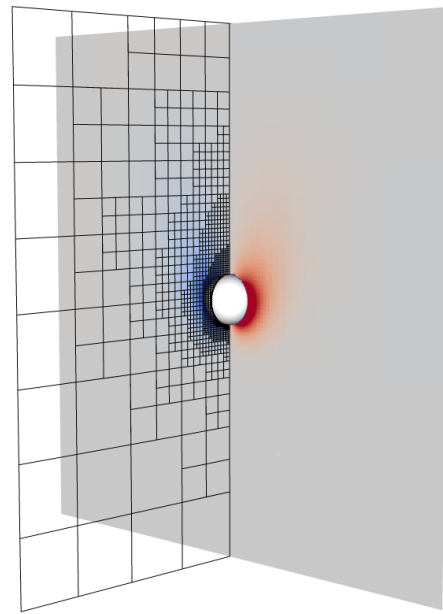
1084 The range of Reynolds numbers considered is $Re \in [1.5, 31.9]$, which we obtain by varying
 1085 the reference velocity u_{ref} and the viscosity ν . Similarly, the range of Stokes numbers considered
 1086 is $St \in [0.19, 4.13]$, which we obtain by varying the density ratio ρ_{Γ}/ρ . For these low to
 1087 moderate Reynolds number flows, the particle experiences a period of nearly steady fall after
 1088 an initial acceleration. As it approaches the bottom of the box, the particle decelerates towards
 1089 zero and we stop the simulation when one cell is left between the sphere and the bottom wall.
 1090 As an illustration, we display in Figure 26 snapshots at $t / (d/u_{\text{ref}}) = 1$ of the vorticity in the
 1091 xy -plane obtained for different combinations of the Reynolds and Stokes numbers.

1092 In Figure 27, we plot the time evolution of the position $x_{\Gamma,y}/d$ (see Figure 27a) and the set-
 1093 tling velocity $u_{\Gamma,y}/u_{\text{ref}}$ (see Figure 27b) of the sphere for different combinations of the Reynolds
 1094 and Stokes numbers: $[Re, St] = \{[1.5, 0.19], [4.1, 0.53], [11.6, 1.5], [31.9, 4.13]\}$. To assess the
 1095 accuracy of the method, the results are obtained on three different octrees defined by the follow-
 1096 ing values of the maximum level of refinement $l_{\max} = \{7 \text{ (12 pt/d)}, 8 \text{ (24 pt/d)}, 9 \text{ (48 pt/d)}\}$.
 1097 For each case, we compare our results to those of [83] and observe a very good agreement
 1098 with the reference experimental data, even when using $l_{\max} = 7$ (12 pt/d). However, values of
 1099 $l_{\max} > 7$ are required to obtain converged results for moderate Reynolds number flows.

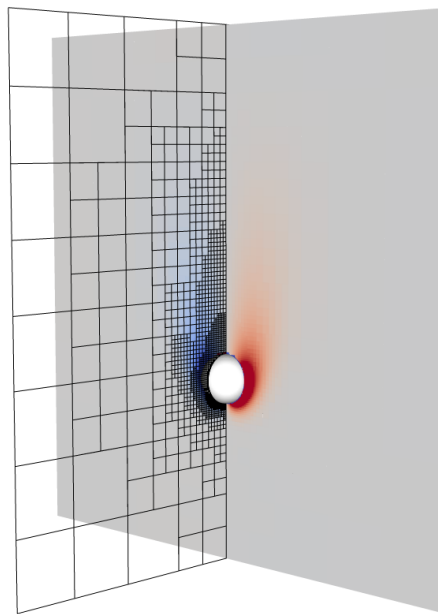
1100 The Cartesian grid embedded boundary method along with the first-order in time explicit
 1101 weak fluid-solid coupling strategy are therefore able to accurately capture the two-way interac-
 1102 tions between the fluid and the particle with a limited number of grid points.



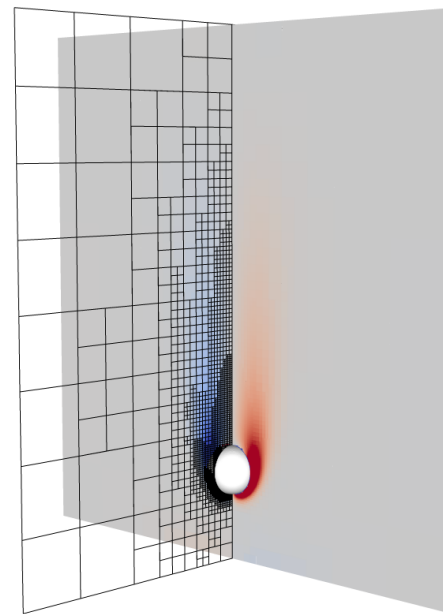
(a) $[Re, St] = [1.5, 0.19]$.



(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 $t/(d/u_{\text{ref}}) = 1$ 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 $l_{\text{max}} = 9$ (48 pt/d). <http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling.c>.

1103 Complete details of this test case can be found here: [http://basilisk.fr/sandbox/](http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling.c)
 1104 [ghigo/src/test-particle/sphere-settling.c](http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling.c).

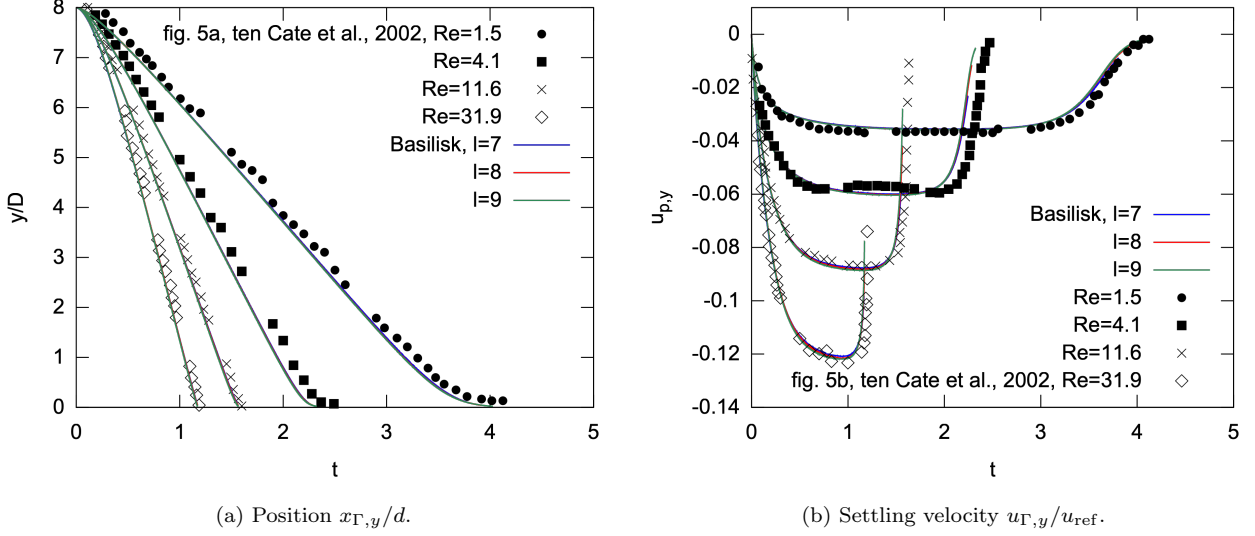


Figure 27: Time evolution of (a) the position $x_{\Gamma,y}/d$ and (b) the settling velocity $u_{\Gamma,y}/u_{ref}$ of a sphere of near-unity density ratio settling in a closed box, for different combinations of the Reynolds and Stokes numbers: $[Re, St] = \{[1.5, 0.19], [4.1, 0.53], [11.6, 1.5], [31.9,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

1105 6.8. Heavy sphere settling in a large closed box

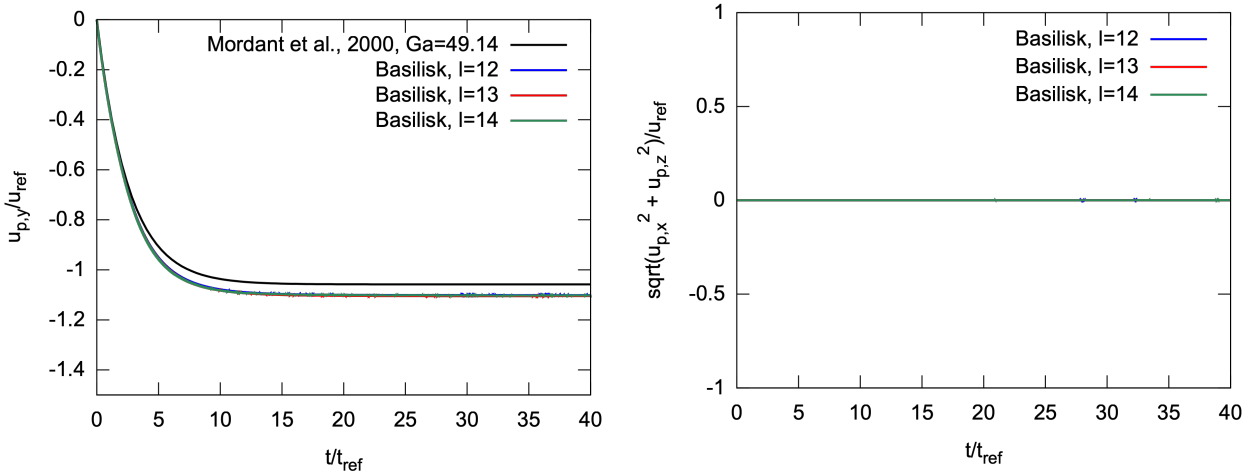
1106 In this final validation test case, we investigate the settling at large Reynolds or Galileo
 1107 numbers of a heavy sphere under the action of the gravity $\mathbf{g} = [0, -9.81, 0]^T \text{ m/s}^2$. This test
 1108 case is inspired by the experimental work of [87], where the authors studied the motion of
 1109 spherical beads in a large container filled with water. More particularly, we reproduce here
 1110 cases 1, 2 and 4 from [87].

1111 Following [5], we use a sphere of diameter $d = 2/12 \text{ m}$, define a reference velocity $u_{ref} =$
 1112 $\sqrt{|g_y|d}$ and consider the following three combinations of density ratio ρ_{Γ}/ρ and Galileo
 1113 $Ga = \sqrt{|1 - \frac{\rho_{\Gamma}}{\rho}|d^3|g_y|/\nu}$: $[\rho_{\Gamma}/\rho, Ga] = \{[2.56, 49.14]$ (case 1), $[2.56, 255.35]$ (case 2), $[7.71, 206.27]$ (case
 1114 which we obtain by varying the viscosity ν . Unfortunately, it is not possible to deduce from
 1115 the experimental results of [87] to which sphere trajectory regime these three cases belong to.
 1116 Indeed, the settling velocities presented in [87] have been averaged over multiple realizations of
 1117 the same experiment. However, according to the map of regimes of sphere trajectories in the
 1118 $[\rho_{\Gamma}/\rho, Ga]$ plane obtained numerically in [88] and experimentally in [89], we expect that cases
 1119 1, 2 and 4 should display different sphere trajectory regimes.

1120 In case 1, the Galileo number is low enough that the sphere should settle vertically with a
 1121 steady axisymmetric wake. In case 4, as the Galileo number increases, the sphere should shed
 1122 a single-sided chain of vortex loops, producing an unsteady side force that causes the sphere
 1123 to follow an unsteady oblique trajectory. In [89], the authors qualify this sphere trajectory
 1124 regime as the oblique trajectory regime. However, in [88], the authors further distinguish
 1125 between the steady, low frequency oscillating and high frequency oscillating oblique trajectory
 1126 regimes. For the combination of density ratio and Galileo number considered in case 4, the
 1127 sphere should follow, according to [88], an oblique and high frequency oscillating trajectory.

1128 Finally, in case 2, the Galileo number is even larger and the sphere should follow, according to
 1129 [88], a chaotic trajectory. Note however that for the Galileo number considered in case 2, the
 1130 sphere trajectory regime is not described in [89] as a chaotic trajectory regime but rather as an
 1131 intermittent oblique trajectory regime. Indeed, in this regime the authors still observe in the
 1132 wake structure some vortex loops consistent with an oblique trajectory.

1133 Instead of defining a tri-periodic domain as in [5], we take full advantage of the mesh
 1134 adaptation capabilities of the Cartesian grid embedded boundary method and define a very
 1135 large box of size $L_0 = 32$ m as in the experiments of [87]. We therefore do not have to worry
 1136 about the particle being affected by the remnants of its periodic wake. We impose no-slip
 1137 boundary conditions on all boundaries and embed the sphere at a distance $5d$ from the top
 1138 boundary. Then, starting from an initial condition $\mathbf{u}(t=0) = \mathbf{0}$ and $p = 0$, we solve the
 1139 coupled system of equations (1)–(2) and (4)–(5) on an octree, dynamically adapted at each
 1140 time step based on the velocity \mathbf{u} with an adaptation criteria $\xi_{\text{adapt}}/u_{\text{ref}} = 5 \times 10^{-3}$. We also
 1141 set the tolerance of the multigrid solver to $\xi_{\text{mg}} = 10^{-4}$. Finally, the time step is bounded by
 1142 $\Delta t/(d/u_{\text{ref}}) \leq 10^{-3}$.



(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 28: 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 $[\rho_{\Gamma}/\rho, Ga] = [2.56, 49.14]$ (case 1 of [87]), obtained on three different octrees defined by the following values of the maximum level of refinement $l_{\text{max}} = \{12 (21 \text{ pt/d}), 13 (42 \text{ pt/d}), 14 (85 \text{ pt/d})\}$. <http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling-large-domain.c>.

1143 In Figure 28, we first present the results we obtain for case 1 of [87] using the following
 1144 combination of density ratio and Galileo number $[\rho_{\Gamma}/\rho, Ga] = [2.56, 49.14]$. We plot both the
 1145 time evolution of the vertical settling velocity $u_{\Gamma,y}/u_{\text{ref}}$ of the sphere as well as the norm of
 1146 its lateral velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{\text{ref}}$. To assess the accuracy of the method, the results are
 1147 obtained on three different octrees defined by the following values of the maximum level of
 1148 refinement $l_{\text{max}} = \{12 (21 \text{ pt/d}), 13 (42 \text{ pt/d}), 14 (85 \text{ pt/d})\}$. After a transient acceleration
 1149 phase, we observe that we recover the expected steady vertical trajectory [89], characterized by

1150 a constant vertical settling velocity and a zero lateral velocity. Furthermore, the final steady
 1151 vertical settling velocity differs by less than 5% from the experimental results of [87].

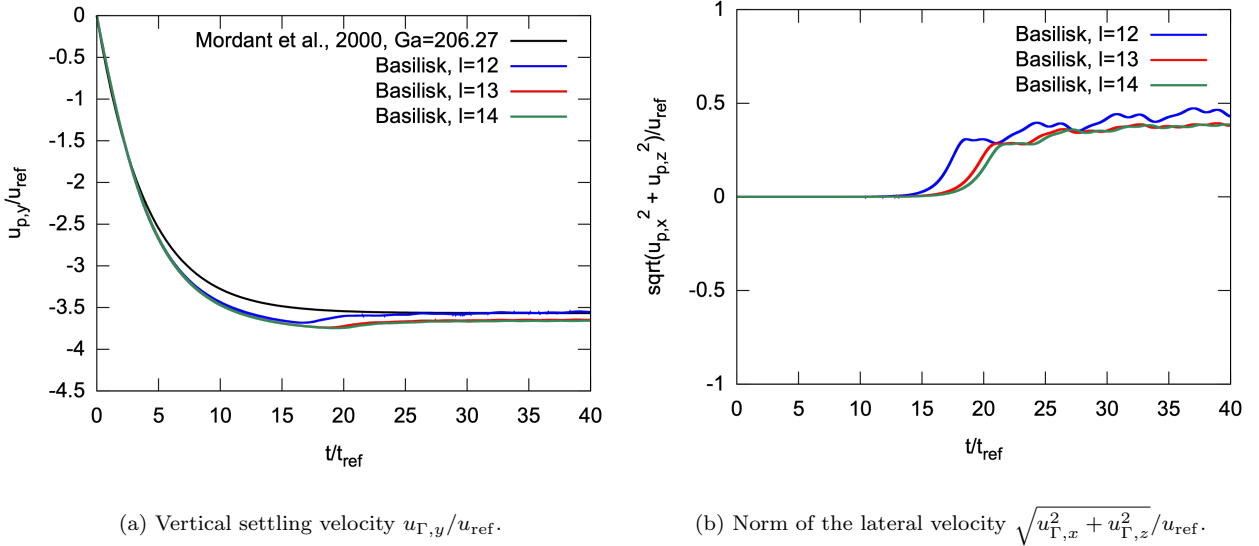


Figure 29: Time evolution of (a) the vertical settling velocity $u_{\Gamma,y}/u_{\text{ref}}$ and (b) the norm of the lateral velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{\text{ref}}$ of a heavy sphere settling in a large domain for the following combination of density ratio and Galileo number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

1152 In Figure 29, we present the time evolution of the same quantities obtained for case 4 of [87]
 1153 using the following combination of density ratio and Galileo number $[\rho_{\Gamma}/\rho, Ga] = [7.71, 206.27]$.
 1154 We observe that the sphere follows the expected oblique and high frequency oscillating trajec-
 1155 tory [88]. Indeed, the computed lateral velocity in Figure 29b oscillates with a high frequency
 1156 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 pre-
 1157 dicted, using a Lattice-Boltzmann method, a steady lateral velocity $u_{pH}/u_g \approx 0.1$ for the
 1158 following combination of density ratio and Galileo number $[\rho_{\Gamma}/\rho, Ga] = [1.5, 178.46]$, with
 1159 $u_g = u_{\text{ref}}\sqrt{|\rho_{\Gamma}/\rho - 1|}$. Therefore, rescaling our results by $\sqrt{|\rho_{\Gamma}/\rho - 1|}$, we obtain the average
 1160 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
 1161 we use a higher Galileo number. Furthermore, the final value of the vertical settling velocity
 1162 in Figure 29a matches the one obtained in [87]. Note however that since the Galileo number is
 1163 larger than in case 1, the results are converged only for $l_{\text{max}} \geq 13$ (42 pt/d). Finally, we confirm
 1164 that the sphere follows an oblique and high frequency oscillating trajectory by displaying in
 1165 Figure 30 the iso-surfaces $\lambda_2 = -0.05$ at times $t/(d/u_{\text{ref}}) = \{20, 40\}$. Indeed, we observe the
 1166 vortex structures characteristic of this regime [89, 88], with a double threaded vortical structure
 1167 near the sphere and a train of one-sided vortex loops further downstream.

1168 Finally, we plot in Figure 31 similar results obtained for case 2 of [87] using the follow-
 1169 ing combination of density ratio and Galileo number $[\rho_{\Gamma}/\rho, Ga] = [2.56, 255.35]$. We first
 1170 observe that the final value of the vertical settling velocity matches the one obtained in [87]
 1171 for $l_{\text{max}} \geq 13$ (42 pt/d). Furthermore, we notice that sphere no longer follows an oblique and

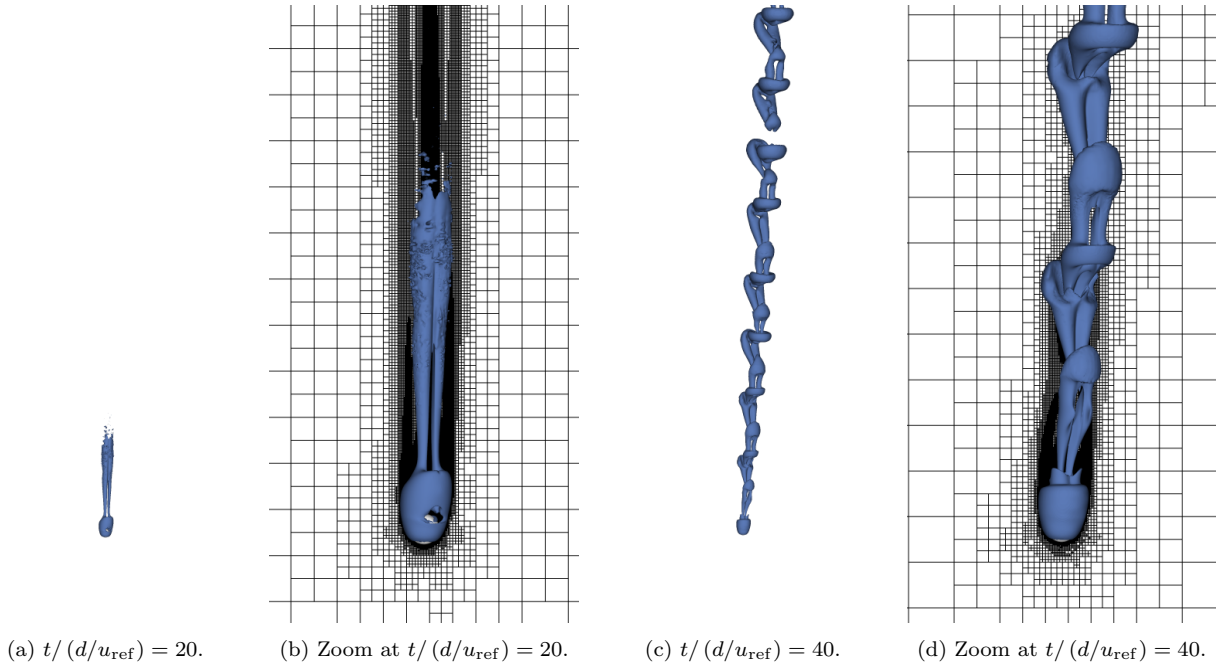


Figure 30: Snapshots at times $t/(d/u_{\text{ref}}) = \{20, 40\}$ of the iso-surface $\lambda_2 = -0.05$ around of a heavy sphere settling in a large domain for the following combination of density ratio and Galileo number $[\rho_{\Gamma}/\rho, Ga] = [7.71, 206.27]$ (case 4 of [87]), obtained on an octree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

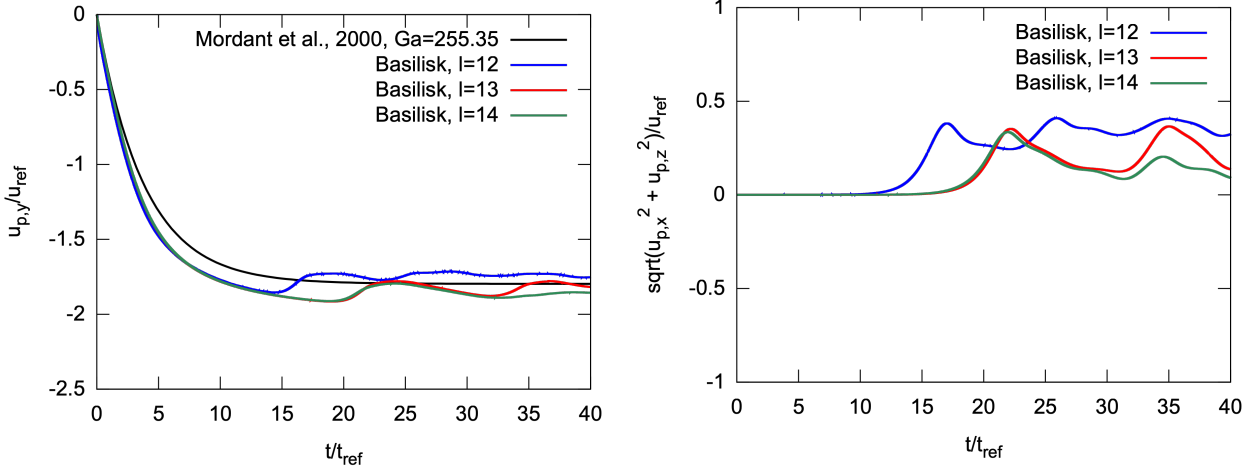
1172 high frequency oscillating trajectory as in case 4 but rather an intermittent oblique trajectory
 1173 as described in [89]. Indeed, in Figure 32, we display the iso-surfaces $\lambda_2 = -0.05$ at times
 1174 $t/(d/u_{\text{ref}}) = \{30, 40\}$ and observe again a double threaded vortical structure near the sphere
 1175 and one-sided vortex loops downstream of the sphere. However, contrary to the vortex struc-
 1176 tures presented for case 4 in Figure 32, the vortex loops evolve here into vortex rings that are
 1177 responsible for the intermittency in the sphere’s oblique trajectory.

1178 These results therefore confirm the ability of the Cartesian grid embedded boundary method
 1179 and the first-order in time explicit weak coupling strategy to reproduce for a considerable range
 1180 of Galileo numbers the dynamic behavior of a settling particle under the action of gravity in a
 1181 very large domain.

1182 Complete details of this test case can be found here: <http://basilisk.fr/sandbox/ghigo/src/test-particle/sphere-settling-large-domain.c>.
 1183

1184 *A note on performance.* We compare here the performances of the Cartesian grid embedded
 1185 boundary method and the first-order in time explicit weak fluid-solid coupling strategy we use
 1186 to compute cases 1, 2 and 4 from [87] and summarize the results in Table 2.

1187 In all three cases, we run the simulation, without having to restart, until $t/(d/u_{\text{ref}}) = 40$
 1188 using a maximum level of refinement $l_{\text{max}} = 12$ on 48 Intel Platinum processors. Note here that
 1189 number of processors chosen is not optimal in terms of memory management and could have
 1190 been roughly reduced by a factor of 5. We notice that, as expected, the computational speeds
 1191 are similar in all three cases. However, we observe that the dynamic mesh adaptation of the



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

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

Figure 31: Time evolution of (a) the vertical settling velocity $u_{\Gamma,y}/u_{ref}$ and (b) the norm of the lateral velocity $\sqrt{u_{\Gamma,x}^2 + u_{\Gamma,z}^2}/u_{ref}$ 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 $t/(d/u_{ref}) = 40$ 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>.

1192 unsteady wake behind the sphere in cases 2 and 4 (see Figures 30 and 32) increases the number
 1193 of cells and therefore the wall-clock time compared to case 1. This is especially true for case
 1194 4, where the density ratio $\rho_{\Gamma}/\rho = 7.71$ is larger than in cases 1 and 2 and therefore the sphere
 1195 settles faster, resulting in a longer wake.

1196 7. Conclusions and perspectives

1197 We have presented here an adaptive Cartesian grid embedded boundary method, or cut-
 1198 cell method, for fixed and moving rigid bodies in an incompressible flow. To the best of our
 1199 knowledge, this is one of the first attempts to use a cut-cell method for moving embedded
 1200 boundaries in an incompressible flow. Building on and extending the work of [60, 61], [45, 50]
 1201 and [49, 22], in particular for the treatment of degenerated cases when computing the embedded
 1202 face gradient $\nabla_{\Gamma}s$ (see Section 3.4.1), we have constructed a method that is conservative, second-

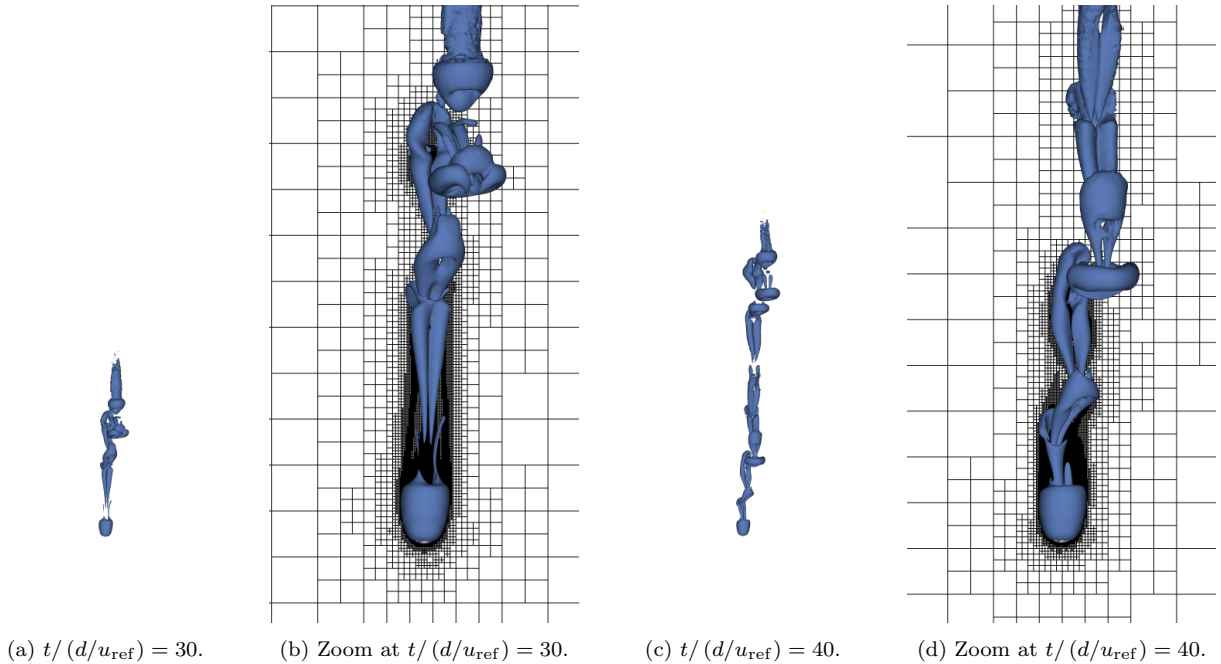


Figure 32: Snapshots at times $t/(d/u_{\text{ref}}) = \{30, 40\}$ of the iso-surface $\lambda_2 = -0.05$ around of a heavy sphere settling in a large domain for the following combination of density ratio and Galileo number $[\rho_{\Gamma}/\rho, Ga] = [2.56, 255.35]$ (case 2 of [87]), obtained on an octree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

1203 order in space, robust and efficient. For the sake of simplicity, we have considered in this study
 1204 only single moving particles and chosen a simple first-order in time explicit integration scheme
 1205 to weakly couple the motion of a particle and the fluid. Finally, we have implemented the
 1206 method along with an extensive validation test suite for canonical particle-laden flow cases in
 1207 the open source software *Basilisk* and extended its use to adaptive tree grids in a robust and
 1208 efficient manner.

1209 Our Cartesian grid embedded boundary method on quad/oc-tree adaptive grids therefore
 1210 opens up unprecedented opportunities for particle-resolved particle-laden flow simulations. In-
 1211 deed, the numerical simulation of moving rigid bodies in an incompressible flow presents many
 1212 numerical challenges due to the wide variety of particle shapes and sizes and to the complexity
 1213 of the fluid-particle momentum transfers. We can now overcome these difficulties by taking
 1214 full advantage of the simplicity of the Cartesian grid approach and cut-cell representation of
 1215 the geometry to describe moving particles of arbitrary shape. The high-order accuracy and
 1216 mesh adaptivity properties of the method also allow us to properly capture boundary layers
 1217 around moving rigid bodies while maintaining computational costs to a minimum. We were
 1218 therefore able to perform fluid-particle simulations at spatial resolution that were previously
 1219 unattainable using simple regular Cartesian grid and predict expected particle dynamics for a
 1220 wide range of Reynolds ($0 \leq Re \leq 1000$) and Galileo numbers ($0 \leq Ga \leq 250$) (see Sections
 1221 6.7 and 6.8). Furthermore, the conservative properties of the method enhance the stability and
 1222 convergence properties of the fluid-solid coupling [21, 22], allowing us to use an explicit weak
 1223 fluid-solid coupling strategy. In future works, we nevertheless plan to couple the method to the

1224 granular solver Grains3D [64] to compute particle-particle collisions for any particle shape with
1225 second-order time accuracy. As demonstrated in Section 6.2, our method can also be straight-
1226 forwardly extended to heat transfers and could therefore be used to compute high Reynolds
1227 and high Prandtl numbers particle-laden flows.

1228 In future works, we will improve the method to remove the small oscillations observed in
1229 the pressure signal in Section 6.5 in the limit case of high Reynolds numbers ($Re \sim 1000$).
1230 As a reminder, these oscillations are caused by the dynamic adaptation of the mesh coupled
1231 to the approximate projection of the cell-centered velocity field. Inspired by the work of [91],
1232 these pressure oscillations can be removed by introducing an additional projection step and
1233 defining two pressures: a standard pressure used to project the velocity as in equation (36c)
1234 and an auxiliary diagnostic pressure used to compute the pressure force on the embedded
1235 boundaries. This auxiliary pressure is obtained by projecting only the update (i.e. its evolution
1236 in time) of the centered velocity field $\mathbf{u}^{n+1} - \mathbf{u}^n$ and therefore does not feel the history of
1237 divergence of the centered velocity field, which includes the noise induced by adaptive mesh
1238 refinement. Preliminary work and results for fixed embedded boundaries can be found here:
1239 <http://basilisk.fr/src/navier-stokes/double-projection.h>.

1240 References

- 1241 [1] C. Peskin, Numerical analysis of blood flow in the heart, *Journal of Computational Physics*
1242 25 (3) (1977) 220–252.
- 1243 [2] E. A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary
1244 finite-difference methods for three-dimensional complex flow simulations, *Journal of Com-
1245 putational Physics* 161 (1) (2000) 35–60.
- 1246 [3] C. Peskin, The immersed boundary method, *Acta Numerica* 11 (1) (2002) 479–517.
- 1247 [4] Z. Feng, E. Michaelides, The immersed boundary-lattice Boltzmann method for solving
1248 fluid-particles interaction problems, *Journal of Computational Physics* 195 (2) (2004) 602–
1249 628.
- 1250 [5] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of
1251 particulate flows, *Journal of Computational Physics* 209 (2) (2005) 448–476.
- 1252 [6] R. Mittal, G. Iaccarino, Immersed boundary methods, *Annu. Rev. Fluid Mech.* 37 (2005)
1253 239–261.
- 1254 [7] R. Bale, A. P. S. Bhalla, B. E. Griffith, M. Tsubokura, A one-sided direct forcing immersed
1255 boundary method using moving least squares, *Journal of Computational Physics* 440 (2021)
1256 110359.
- 1257 [8] R. Ghias, R. Mittal, T. Lund, A non-body conformal grid method for simulation of com-
1258 pressible flows with complex immersed boundaries, in: 42nd AIAA Aerospace Sciences
1259 Meeting and Exhibit, 2004, p. 80.
- 1260 [9] S. Majumdar, G. Iaccarino, P. Durbin, RANS solvers with adaptive structured boundary
1261 non-conforming grids, Tech. rep., CTR Annual Research Brief (2001).

- 1262 [10] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas, A. von Loebbecke, A ver-
1263 satile sharp interface immersed boundary method for incompressible flows with complex
1264 boundaries, *Journal of computational physics* 227 (10) (2008) 4825–4852.
- 1265 [11] J. H. Seo, R. Mittal, A high-order immersed boundary method for acoustic wave scattering
1266 and low-Mach number flow-induced sound in complex geometries, *Journal of Computa-
1267 tional Physics* 230 (4) (2011) 1000–1019.
- 1268 [12] S. Tenneti, R. Garg, S. Subramaniam, Drag law for monodisperse gas–solid systems us-
1269 ing particle-resolved direct numerical simulation of flow past fixed assemblies of spheres,
1270 *International Journal of Multiphase Flow* 37 (9) (2011) 1072–1092.
- 1271 [13] J. Xia, K. Luo, J. Fan, A ghost-cell based high-order immersed boundary method for inter-
1272 phase heat transfer simulation, *International Journal of Heat and Mass Transfer* 75 (2014)
1273 302–312.
- 1274 [14] R. Glowinski, T. Pan, T. Hesla, D. Joseph, A distributed Lagrange multiplier/fictitious
1275 domain method for particulate flows, *International Journal of Multiphase Flow* 25 (5)
1276 (1999) 755–794.
- 1277 [15] Z. Yu, X. Shao, A. Wachs, A fictitious domain method for particulate flows with heat
1278 transfer, *Journal of Computational Physics* 217 (2) (2006) 424–452.
- 1279 [16] A. Wachs, A. Hammouti, G. Vinay, M. Rahmani, Accuracy of finite volume/staggered
1280 grid distributed Lagrange multiplier/fictitious domain simulations of particulate flows,
1281 *Computers & Fluids* 115 (2015) 154–172.
- 1282 [17] A. Ladd, R. Verberg, Lattice-Boltzmann simulations of particle-fluid suspensions, *Journal
1283 of Statistical Physics* 104 (5) (2001) 1191–1251.
- 1284 [18] C. Peng, O. M. Ayala, L.-P. Wang, A comparative study of immersed boundary method
1285 and interpolated bounce-back scheme for no-slip boundary treatment in the lattice Boltz-
1286 mann method: Part I, laminar flows, *Computers & Fluids* 192 (2019) 104233.
- 1287 [19] C. K. Aidun, J. R. Clausen, Lattice-Boltzmann method for complex flows, *Annual Review
1288 of Fluid Mechanics* 42 (2010) 439–472.
- 1289 [20] A. Wachs, Particle-scale computational approaches to model dry and saturated granular
1290 flows of non-Brownian, non-cohesive, and non-spherical rigid bodies, *Acta Mechanica* 230
1291 (2019) 1919–1980.
- 1292 [21] C. Michler, E. Van Brummelen, S. Hulshoff, R. De Borst, The relevance of conservation
1293 for stability and accuracy of numerical methods for fluid–structure interaction, *Computer
1294 methods in applied mechanics and engineering* 192 (37-38) (2003) 4195–4215.
- 1295 [22] L. Schneiders, C. Günther, M. Meinke, W. Schröder, An efficient conservative cut-cell
1296 method for rigid bodies interacting with viscous compressible flows, *Journal of Computa-
1297 tional Physics* 311 (2016) 62–86.

- 1298 [23] J. Feng, H. Hu, D. Joseph, Direct simulation of initial value problems for the motion of
1299 solid bodies in a Newtonian fluid. Part 1. Sedimentation, *Journal of Fluid Mechanics* 261
1300 (1994) 95–134.
- 1301 [24] J. Feng, H. Hu, D. Joseph, Direct simulation of initial value problems for the motion of
1302 solid bodies in a Newtonian fluid. Part 2. Couette and Poiseuille flows, *Journal of Fluid*
1303 *Mechanics* 277 (271) (1994) 271–301.
- 1304 [25] A. A. Johnson, T. E. Tezduyar, Advanced mesh generation and update methods for 3D
1305 flow simulations, *Computational Mechanics* 23 (2) (1999) 130–143.
- 1306 [26] C. R. Choi, C. N. Kim, Direct numerical simulations of the dynamics of particles with
1307 arbitrary shapes in shear flows, *Journal of Hydrodynamics, Ser. B* 22 (4) (2010) 456–465.
- 1308 [27] N. O. Jaensson, M. A. Hulsen, P. D. Anderson, Direct numerical simulation of particle
1309 alignment in viscoelastic fluids, *Journal of Non-Newtonian Fluid Mechanics* 235 (2016)
1310 125–142.
- 1311 [28] A. Koblitz, S. Lovett, N. Nikiforakis, W. D. Henshaw, Direct numerical simulation of
1312 particulate flows with an overset grid method, *Journal of computational physics* 343 (2017)
1313 414–431.
- 1314 [29] W. J. Horne, K. Mahesh, A massively-parallel, unstructured overset method to simulate
1315 moving bodies in turbulent flows, *Journal of Computational Physics* 397 (2019) 108790.
- 1316 [30] K. Luo, Z. Wang, J. Fan, K. Cen, Full-scale solutions to particle-laden flows: Multidirect
1317 forcing and immersed boundary method, *Physical Review E* 76 (6) (2007) 066709.
- 1318 [31] Z. Wang, J. Fan, K. Luo, Combined multi-direct forcing and immersed boundary method
1319 for simulating flows with moving particles, *International Journal of Multiphase Flow* 34 (3)
1320 (2008) 283–302.
- 1321 [32] T. Kempe, J. Fröhlich, Collision modelling for the interface-resolved simulation of spherical
1322 particles in viscous fluids, *Journal of Fluid Mechanics* 709 (2012) 445–489.
- 1323 [33] Y. Tang, S. Kriebitzsch, E. Peters, M. van der Hoef, J. Kuipers, A methodology for highly
1324 accurate results of direct numerical simulations: drag force in dense gas–solid flows at
1325 intermediate Reynolds number, *International Journal of Multiphase Flow* 62 (2014) 73–
1326 86.
- 1327 [34] S. Schwarz, T. Kempe, J. Fröhlich, A temporal discretization scheme to compute the
1328 motion of light particles in viscous flows by an immersed boundary method, *Journal of*
1329 *Computational Physics* 281 (2015) 591–613.
- 1330 [35] G. Akiki, S. Balachandar, Immersed boundary method with non-uniform distribution of
1331 Lagrangian markers for a non-uniform Eulerian mesh, *Journal of Computational Physics*
1332 307 (2016) 34–59.

- 1333 [36] M. Moriche, M. Uhlmann, J. Dušek, A single oblate spheroid settling in unbounded ambi-
1334 ent fluid: A benchmark for simulations in steady and unsteady wake regimes, *International*
1335 *Journal of Multiphase Flow* 136 (2021) 103519.
- 1336 [37] A. Wachs, A. Hammouti, G. Vinay, M. Rahmani, Accuracy of Finite Volume/Staggered
1337 Grid Distributed Lagrange Multiplier/Fictitious Domain simulations of particulate flows,
1338 *Computers & Fluids* 115 (2015) 154–172.
- 1339 [38] J. Keating, P. Mineev, A fast algorithm for direct simulation of particulate flows using
1340 conforming grids, *Journal of Computational Physics* 255 (2013) 486–501.
- 1341 [39] N. Deen, S. Kriebitzsch, M. van der Hoef, J. Kuipers, Direct numerical simulation of flow
1342 and heat transfer in dense fluid-particle systems, *Chemical Engineering Science* 81 (2012)
1343 329–344.
- 1344 [40] X. Liu, W. Ge, L. Wang, Scale and structure dependent drag in gas–solid flows, *AICHE*
1345 *Journal* 66 (4) (2020) e16883.
- 1346 [41] A. G. Kidanemariam, M. Uhlmann, Formation of sediment patterns in channel flow: mini-
1347 mal unstable systems and their temporal evolution, *Journal of Fluid Mechanics* 818 (2017)
1348 716–743.
- 1349 [42] A. Roma, C. Peskin, M. Berger, An adaptive version of the immersed boundary method,
1350 *Journal of Computational Physics* 153 (2) (1999) 509–534.
- 1351 [43] B. E. Griffith, R. D. Hornung, D. M. McQueen, C. S. Peskin, An adaptive, formally sec-
1352 ond order accurate version of the immersed boundary method, *Journal of Computational*
1353 *Physics* 223 (1) (2007) 10–49.
- 1354 [44] M. Bauer, S. Eibl, C. Godenschwager, N. Kohl, M. Kuron, C. Rettinger, F. Schornbaum,
1355 C. Schwarzmeier, D. Thönnies, H. Köstler, U. Rude, waLBerla: A block-structured high-
1356 performance framework for multiphysics simulations, *Computers & Mathematics with Ap-*
1357 *plications* 81 (2020) 478–501.
- 1358 [45] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in
1359 complex geometries, *Journal of Computational Physics* 190 (2) (2003) 572–600.
- 1360 [46] C. Min, F. Gibou, A second order accurate projection method for the incompressible
1361 Navier-Stokes equations on non-graded adaptive grids, *Journal of Computational Physics*
1362 219 (2) (2006) 912–929.
- 1363 [47] G. Eitel-Amor, M. Meinke, W. Schröder, A lattice-Boltzmann method with hierarchically
1364 refined meshes, *Computers & Fluids* 75 (2013) 127–139.
- 1365 [48] M. Meinke, L. Schneiders, C. Günther, W. Schröder, A cut-cell method for sharp moving
1366 boundaries in cartesian grids, *Computers & Fluids* 85 (2013) 135–142.
- 1367 [49] L. Schneiders, D. Hartmann, M. Meinke, W. Schröder, An accurate moving boundary
1368 formulation in cut-cell methods, *Journal of Computational Physics* 235 (2013) 786–809.

- 1369 [50] S. Popinet, A quadtree-adaptive multigrid solver for the Serre–Green–Naghdi equations,
1370 *Journal of Computational Physics* 302 (2015) 336–358.
- 1371 [51] G. D’Avino, M. Hulsen, A comparison between a collocation and weak implementation of
1372 the rigid-body motion constraint on a particle surface, *International Journal for Numerical*
1373 *Methods in Fluids* 64 (9) (2010) 1014–1040.
- 1374 [52] C. Selcuk, A. R. Ghigo, S. Popinet, A. Wachs, A fictitious domain method with dis-
1375 tributed lagrange multipliers on adaptive quad/octrees for the direct numerical simulation
1376 of particle-laden flows, *Journal of Computational Physics* 430 (2021) 109954.
- 1377 [53] A. Wachs, Particle-scale computational approaches to model dry and saturated granu-
1378 lar flows of non-brownian, non-cohesive, and non-spherical rigid bodies, *Acta Mechanica*
1379 230 (6) (2019) 1919–1980.
- 1380 [54] W. Bennett, N. Nikiforakis, R. Klein, A moving boundary flux stabilization method for
1381 Cartesian cut-cell grids using directional operator splitting, *Journal of Computational*
1382 *Physics* 368 (2018) 333–358.
- 1383 [55] K. Sverdrup, A. Almgren, N. Nikiforakis, An embedded boundary approach for efficient
1384 simulations of viscoplastic fluids in three dimensions, *Physics of Fluids* 31 (9) (2019)
1385 093102.
- 1386 [56] K. Fröhlich, M. Meinke, W. Schröder, Correlations for inclined prolates based on highly
1387 resolved simulations, *Journal of Fluid Mechanics* 901 (2020) A5.
- 1388 [57] M.-H. Chung, Cartesian cut cell approach for simulating incompressible flows with rigid
1389 bodies of arbitrary shape, *Computers & Fluids* 35 (6) (2006) 607–623.
- 1390 [58] M.-H. Chung, An adaptive Cartesian cut-cell/level-set method to simulate incompressible
1391 two-phase flows with embedded moving solid boundaries, *Computers & Fluids* 71 (2013)
1392 469–486.
- 1393 [59] Z. Xie, T. Stoesser, A three-dimensional Cartesian cut-cell/volume-of-fluid method for two-
1394 phase flows with moving bodies, *Journal of Computational Physics* 416 (2020) 109536.
- 1395 [60] H. Johansen, P. Colella, A cartesian grid embedded boundary method for Poisson’s equa-
1396 tion on irregular domains, *Journal of Computational Physics* 147 (1) (1998) 60–85.
- 1397 [61] P. Schwartz, M. Barad, P. Colella, T. Ligocki, A cartesian grid embedded boundary method
1398 for the heat equation and Poisson’s equation in three dimensions, *Journal of Computational*
1399 *Physics* 211 (2) (2006) 531–550.
- 1400 [62] P. Colella, D. T. Graves, B. J. Keen, D. Modiano, A cartesian grid embedded boundary
1401 method for hyperbolic conservation laws, *Journal of Computational Physics* 211 (1) (2006)
1402 347–366.
- 1403 [63] J. B. Bell, P. Colella, H. M. Glaz, A second-order projection method for the incompressible
1404 navier-stokes equations, *Journal of Computational Physics* 85 (2) (1989) 257–283.

- 1405 [64] A. Wachs, L. Girolami, G. Vinay, G. Ferrer, Grains3d, a flexible dem approach for particles
1406 of arbitrary convex shape — Part I: Numerical model and validations, *Powder Technology*
1407 224 (2012) 374–389.
- 1408 [65] L. Schneiders, C. Guenther, J. H. Grimmer, M. H. Meinke, W. Schroeder, Sharp resolution
1409 of complex moving geometries using a multi-cut-cell viscous flow solver, in: *22nd AIAA*
1410 *Computational Fluid Dynamics Conference*, 2015, p. 3427.
- 1411 [66] R. B. Pember, J. B. Bell, P. Colella, W. Y. Curtchfield, M. L. Welcome, An adaptive
1412 Cartesian grid method for unsteady compressible flow in irregular regions, *Journal of*
1413 *computational Physics* 120 (2) (1995) 278–304.
- 1414 [67] R. Scardovelli, S. Zaleski, Analytical relations connecting linear interfaces and volume
1415 fractions in rectangular grids, *Journal of Computational Physics* 164 (1) (2000) 228–237.
- 1416 [68] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, *Journal*
1417 *of Computational Physics* 228 (16) (2009) 5838–5866.
- 1418 [69] D. Trebotich, D. Graves, An adaptive finite volume method for the incompressible Navier–
1419 Stokes equations in complex geometries, *Communications in Applied Mathematics and*
1420 *Computational Science* 10 (1) (2015) 43–82.
- 1421 [70] J. A. van Hooft, S. Popinet, C. C. van Heerwaarden, S. J. van der Linden, S. R. de Roode,
1422 B. J. van de Wiel, Towards adaptive grids for atmospheric boundary–layer simulations,
1423 *Boundary-layer meteorology* 167 (3) (2018) 421–443.
- 1424 [71] J. López-Herrera, S. Popinet, A. Castrejón-Pita, An adaptive solver for viscoelastic incom-
1425 pressible two–phase problems applied to the study of the splashing of weakly viscoelastic
1426 droplets, *Journal of Non-Newtonian Fluid Mechanics* 264 (2019) 144–158.
- 1427 [72] A. J. Chorin, On the convergence of discrete approximations to the Navier-Stokes equa-
1428 tions, *Mathematics of computation* 23 (106) (1969) 341–353.
- 1429 [73] P. M. Gresho, R. L. Sani, On pressure boundary conditions for the incompressible Navier–
1430 Stokes equations, *International Journal for Numerical Methods in Fluids* 7 (10) (1987)
1431 1111–1145.
- 1432 [74] H. Udaykumar, R. Mittal, P. Rampungoon, A. Khanna, A sharp interface cartesian grid
1433 method for simulating flows with complex moving boundaries, *Journal of computational*
1434 *physics* 174 (1) (2001) 345–380.
- 1435 [75] J. H. Seo, R. Mittal, A sharp–interface immersed boundary method with improved mass
1436 conservation and reduced spurious pressure oscillations, *Journal of computational physics*
1437 230 (19) (2011) 7347–7363.
- 1438 [76] J. J. Quirk, An alternative to unstructured grids for computing gas dynamic flows around
1439 arbitrarily complex two-dimensional bodies, *Computers & fluids* 23 (1) (1994) 125–142.

- 1440 [77] C. Helzel, M. J. Berger, R. J. LeVeque, A high-resolution rotated grid method for con-
1441 servation laws with embedded geometries, *SIAM Journal on Scientific Computing* 26 (3)
1442 (2005) 785–809.
- 1443 [78] N. Gokhale, N. Nikiforakis, R. Klein, A dimensionally split Cartesian cut cell method for
1444 hyperbolic conservation laws, *Journal of Computational Physics* 364 (2018) 186–208.
- 1445 [79] P. Koumoutsakos, A. Leonard, High-resolution simulations of the flow around an impul-
1446 sively started cylinder using vortex methods, *Journal of Fluid Mechanics* 296 (1995) 1–38.
- 1447 [80] F. Mohaghegh, H. Udaykumar, Comparison of sharp and smoothed interface methods for
1448 simulation of particulate flows II: Inertial and added mass effects, *Computers & Fluids* 143
1449 (2017) 103–119.
- 1450 [81] R. Mei, Flow due to an oscillating sphere and an expression for unsteady drag on the
1451 sphere at finite Reynolds number, *Journal of Fluid Mechanics* 270 (1994) 133–174.
- 1452 [82] X. Wang, P. Yu, K. Yeo, B. Khoo, SVD–GFD scheme to simulate complex moving body
1453 problems in 3D space, *Journal of Computational Physics* 229 (6) (2010) 2314–2338.
- 1454 [83] A. Ten Cate, C. Nieuwstad, J. Derksen, H. Van den Akker, Particle imaging velocimetry
1455 experiments and Lattice-Boltzmann simulations on a single sphere settling under gravity,
1456 *Physics of Fluids* 14 (11) (2002) 4012–4025.
- 1457 [84] C. Veeramani, P. D. Mineev, K. Nandakumar, A fictitious domain formulation for flows
1458 with rigid particles: A non-lagrange multiplier version, *Journal of Computational Physics*
1459 224 (2) (2007) 867–879.
- 1460 [85] Z.-G. Feng, E. E. Michaelides, Robust treatment of no-slip boundary condition and velocity
1461 updating for the lattice-Boltzmann simulation of particulate flows, *Computers & Fluids*
1462 38 (2) (2009) 370–381.
- 1463 [86] A. Wachs, PeliGRIFF, a parallel DEM-DLM/FD direct numerical simulation tool for 3D
1464 particulate flows, *Journal of Engineering Mathematics* 71 (1) (2011) 131–155.
- 1465 [87] N. Mordant, J.-F. Pinton, Velocity measurement of a settling sphere, *The European Phys-
1466 ical Journal B-Condensed Matter and Complex Systems* 18 (2) (2000) 343–352.
- 1467 [88] M. Jenny, J. Dušek, G. Bouchet, Instabilities and transition of a sphere falling or ascending
1468 freely in a Newtonian fluid, *Journal of Fluid Mechanics* 508 (2004) 201–239.
- 1469 [89] M. Horowitz, C. Williamson, The effect of reynolds number on the dynamics and wakes of
1470 freely rising and falling spheres, *Journal of Fluid Mechanics* 651 (2010) 251–294.
- 1471 [90] C. Rettinger, U. Rüde, A comparative study of fluid-particle coupling methods for fully
1472 resolved lattice Boltzmann simulations, *Computers & Fluids* 154 (2017) 74–89.
- 1473 [91] A. S. Almgren, J. B. Bell, W. Y. Crutchfield, Approximate projection methods: Part I.
1474 Inviscid analysis, *SIAM Journal on Scientific Computing* 22 (4) (2000) 1139–1159.