Computers and Fluids 148 (2017) 56–68

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

An adaptive numerical method for free surface flows passing rigidly mounted obstacles☆

Kirill D. Nikitin[a], Maxim A. Olshanskii[b,*], Kirill M. Terekhov[c], Yuri V. Vassilevski[a], Ruslan M. Yanbarisov[d]

[a] Institute of Numerical Mathematics of the Russian Academy of Sciences, Moscow, Russia and Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences, Moscow, Russia
[b] Department of Mathematics, University of Houston, Houston, TX, USA
[c] School of Earth, Energy & Environmental Sciences, Stanford University, Stanford, CA, USA
[d] Moscow Institute of Physics and Technology, Dolgoprudny, Russia

ARTICLE INFO

Article history:
Received 23 November 2016
Revised 7 February 2017
Accepted 9 February 2017
Available online 11 February 2017

Keywords:
Free surface
Incompressible flow
Mesh adaptation
Navier–Stokes
Octree meshes
Curvilinear boundaries
Sloshing container
3D Cylinder of circular cross-section
Flow around oil platform

ABSTRACT

The paper develops a method for the numerical simulation of a free-surface flow of incompressible viscous fluid around a streamlined body. The body is a rigid stationary construction partially submerged in the fluid. The application we are interested in the paper is a flow around a surface mounted offshore oil platform. The numerical method builds on a hybrid finite volume / finite difference discretization using adaptive octree cubic meshes. The mesh is dynamically refined towards the free surface and the construction. Special care is taken to devise a discretization for the case of curvilinear boundaries and interfaces immersed in the octree Cartesian background computational mesh. To demonstrate the accuracy of the method, we show the results for two benchmark problems: the sloshing 3D container and the channel laminar flow passing the 3D cylinder of circular cross-section. Further, we simulate numerically a flow with surface waves around an offshore oil platform for the realistic set of geophysical data.

1. Introduction

Free surface flows passing partially submerged objects are common in nature and engineering applications. The examples include water flows around bridge piers, ship bodies, water plants, or costal constructions. A mathematical model of such phenomena includes fluid dynamics equations and an evolution equation for the free surface. These equations can be posed in a domain of complex geometry. Handling the equations and the geometry numerically in an efficient and accurate way constitutes the major challenge for a CFD method applied to simulate free surface flows passing submerged obstacles. Depending on the applications, the fluid and free surface equations can be coupled to other mathematical models of transport, elasticity, etc. Thus, a reliable fast and accurate solver is desirable.

The previous studies of free surface flows passing submerged bodies include the simulation of Euler flows around hydrofoils [17], a boundary element method with the Lagrangian treatment of free surface evolution [21], a non-body conformal grid finite difference method for compressible flows [15], a stabilized finite element method for fluid equations in ALE form [34], and other FEM-based ALE techniques for fluid-structure interaction described in [2]. The variants of the immersed boundary method [30,37] for the free surface flows were discussed in [26,50]. Analytical and semi-analytical solutions of the free surface flows around specific submerged bodies were studied in [7,47].

The method developed in this paper is based on a hybrid discretization using octree Cartesian background meshes. Octree meshes enjoy a growing reliance in scientific computing community due to the simple Cartesian structure and embedded hierarchy, which makes mesh adaptation, reconstruction and data access fast and easy. In particular, octree meshes can be dynamically adapted towards the free surface. The adaptation can be also based on various error indicators. Fast remeshing with octree grids makes them a natural choice for the simulation of moving interfaces and free surface flows, see, e.g., [14,27,28,32,40,44], as well as more general non-Newtonian and high-speed Newtonian flows, see, e.g., [4,6,20,35,39,51]. The Cartesian structure of octree meshes requires,

☆ Supported by Russian Science Foundation through the grant 14-11-00434.
* Corresponding author.
E-mail address: molshan@math.uh.edu (M.A. Olshanskii).

http://dx.doi.org/10.1016/j.compfluid.2017.02.007
0045-7930/

however, a special technique for handling curvilinear boundaries and interfaces, since the mesh itself provides only the first order geometric accuracy in this case.

Using octree grids for the simulation of flows over partially submerged bodies gives the advantage of better local resolution of the free surface and fluid interaction with the body. For the more accurate treatment of the equations near the curvilinear boundary of the construction, we immerse the rigid object in the background mesh and construct the second order approximation of the fluid and free surface equations in the cut cells. The level-set method is used to recover the evolution of the free surface. Other important ingredients of our approach are the semi-Lagrangian characteristic method for the level-set equations on the dynamic octree meshes from [46], and the splitting method for the fluid equations on the octree meshes from [35] with filtering. In that paper, the method was studied for enclosed incompressible viscous flows in cavities and over bluff bodies.

Compared to well-studied higher-order finite volume and finite difference discretizations on uniform grids, the schemes that exploit adaptivity properties of octree meshes often pay the price of lower accuracy and higher numerical dissipation. This happens due to the presence of hanging nodes on irregular interfaces and non-uniform mesh size, which require interpolation of unknowns and make impossible certain cancellations of discretization errors. Such error cancellations take place for uniform grid due to the stencil symmetry. To overcome this loss of accuracy, we operate with a suitable sets of nodes and least-square minimizing interpolants. Further, we validate our approach by performing a series of numerical experiments. First, we compute a channel flow past a 3D circular cylinder. Second, we simulate the sloshing of water in a 3D tank subject to periodic horizontal excitation. The critical statistics, which are drag, lift coefficients for the first test and water levels for the second test, are compared against reference data found in the literature. The success of the numerical method for both benchmark problems demonstrates its ability to accurately simulate incompressible viscous free-surface flows and flows passing streamlined bodies with curvilinear boundaries. Therefore, we apply the method to simulate the water flow with surface waves around an offshore oil platform rigidly mounted in the Kara sea offshore. The platform is a reconstruction of a currently operating unit. The sea waves runup reproduces the realistic weather scenario in the region of the Kara sea offshore. The statistics of interest are water levels at the platform and forces experienced by the construction.

The rest of the paper is organized as follows. Section 2 reviews the mathematical model. Section 3.1 presents the splitting method for the numerical time integration. Section 3.2 discusses the details of the discretization on the gradely refined octree meshes. In Section 3.3 we devise the numerical treatment of the curvilinear boundaries embedded in the background mesh. Section 4 collects the results of numerical experiments.

2. Mathematical model

Consider a Newtonian incompressible fluid flow in a bounded time-dependent domain $\Omega(t) \in \mathbb{R}^3$ for $t \in (0, T]$. The fluid dynamics is governed by the incompressible Navier–Stokes equations

$$\begin{cases} \rho\left(\dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) - \operatorname{div}\boldsymbol{\sigma}(\mathbf{u}, p) = \mathbf{g} \\ \nabla\cdot\mathbf{u} = 0 \end{cases} \quad \text{in } \Omega(t),\ t \in (0, T], \tag{1}$$

where $\boldsymbol{\sigma}(\mathbf{u}, p) = \nu[\nabla\mathbf{u} + (\nabla\mathbf{u})^T] - p\mathbf{I}$ is the stress tensor of the fluid, $\mathbf{u}$ is the velocity vector field, p is the kinematic pressure, $\mathbf{g}$ is the external force (e.g., gravity), ρ is the density, and ν is the kinematic viscosity. At the initial time $t = 0$ the domain and the velocity field are known:

$$\Omega(0) = \Omega_0, \quad \mathbf{u}|_{t=0} = \mathbf{u}_0, \quad \nabla\cdot\mathbf{u}_0 = 0. \tag{2}$$

We assume that $\overline{\partial\Omega(t)} = \overline{\Gamma_D} \cup \overline{\Gamma(t)} \cup \overline{\Gamma}_{\text{out}} \cup \overline{\Gamma}_{\text{in}}$, where Γ_D is the static boundary(walls), $\Gamma(t)$ is the free surface of fluid, Γ_{in}, Γ_{out} are inflow and outflow parts of the boundary, respectively. Note, that Γ_D, Γ_{in}, Γ_{out} may vary in time, in general. We assume the free surface $\Gamma(t)$ passively evolves with the normal velocity of fluid, i.e., the following kinematic condition is valid

$$v_\Gamma = \mathbf{u}\cdot\mathbf{n} \quad \text{on } \Gamma(t), \tag{3}$$

where $\mathbf{n}$ is the normal vector for $\Gamma(t)$ and v_Γ is the normal velocity of $\Gamma(t)$. Since the free surface flows we interested in this paper have large Weber numbers, we ignore the capillary forces and the boundary condition on $\Gamma(t)$ reads

$$\boldsymbol{\sigma}(\mathbf{u}, p)\mathbf{n} = \mathbf{0} \quad \text{on } \Gamma(t). \tag{4}$$

On the static part of the flow boundary, we assume the velocity field satisfies either no-slip boundary condition

$$\mathbf{u} = \mathbf{0} \quad \text{on } \Gamma_D, \tag{5}$$

or no-penetration and free-slip boundary conditions:

$$\mathbf{u}\cdot\mathbf{n} = \mathbf{0} \quad \text{and} \quad \frac{\partial(\mathbf{u}\cdot\mathbf{t}_i)}{\partial\mathbf{n}} = 0,\ i = 1, 2, \quad \text{on } \Gamma_D, \tag{6}$$

where $\mathbf{t}_i$ and $\mathbf{n}$ are tangential and normal vectors on Γ_D. We shall use the generic notation $\mathcal{B}\mathbf{u}|_{\Gamma_D}$ to denote boundary conditions (5) or (6) on Γ_D. We assume that $\mathbf{u}$ is given on Γ_{in} and $\boldsymbol{\sigma}(\mathbf{u}, p)\mathbf{n} = \mathbf{0}$ on Γ_{out}.

For computational purposes, we shall employ the implicit definition of the free surface evolution with the help of an indicator function. Let $\Gamma(t)$ be given as the zero level of a globally defined Lipschitz continuous *level set* function $\varphi(t, \mathbf{x})$ such that

$$\varphi(t, \mathbf{x}) = \begin{cases} < 0 & \text{if } \mathbf{x} \in \Omega(t) \\ > 0 & \text{if } \mathbf{x} \in \mathbb{R}^3 \setminus \overline{\Omega(t)} \\ = 0 & \text{if } \mathbf{x} \in \Gamma(t) \end{cases} \qquad \text{for all } t \in [0, T].$$

The initial condition (2) defines $\varphi(0, \mathbf{x})$. The kinematic condition (3) implies that for $t > 0$ the level set function can be found as the solution to the transport equation [36]:

$$\frac{\partial\varphi}{\partial t} + \widetilde{\mathbf{u}}\cdot\nabla\varphi = 0 \quad \text{in } \mathbb{R}^3 \times (0, T], \tag{7}$$

where $\widetilde{\mathbf{u}}$ is any (divergence-free) smooth velocity field such that $\widetilde{\mathbf{u}} = \mathbf{u}$ on $\Gamma(t)$.

A numerical method studied in this paper solves the system of equations, boundary and initial conditions (1)–(7). The implicit definition of $\Gamma(t)$ as zero level of a globally defined function φ leads to numerical algorithms which can easily handle complex topological changes of the free surface. The level set function provides an easy access to useful geometric characteristics of $\Gamma(t)$. For instance, the unit outward normal to $\Gamma(t)$ is $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$, and the surface curvature is $\kappa = \nabla\cdot\mathbf{n}$. From the numerical point of view, it is often beneficial if the level set function possesses the signed distance property, i.e. it satisfies the Eikonal equation

$$|\nabla\varphi| = 1. \tag{8}$$

3. Numerical method

The section describes the key ingredients of our numerical approach.

3.1. Numerical time integration

We consider a semi-implicit spitting method based on the semi-Lagrangian approach for the level-set function evolution and a hybrid finite volume / finite difference solvers for the convection-diffusion equations and the Poisson equation for pressure. The algorithm is built on the well-known splitting procedure due to Chorin, Yanenko, Pironneau and others, see, for example, [9,38]. For the sake of presentation simplicity, in this section we ignore the spatial discretization. Important implementation details and the spatial discretization will be addressed in the next section.

We adopt the notation $\mathbf{u}^n$, p^n, φ^n for approximations to the velocity field, the pressure, and the level set function at $t = t_n$. Function φ^n implicitly defines an approximation to fluid domain at time $t = t_n$ through $\Omega_n := \{\mathbf{x} \in \mathbb{R}^3 : \varphi^n(\mathbf{x}) < 0\}$.

Initial conditions define $\mathbf{u}^0 = \mathbf{u}(t_0)$ and $\varphi^0 = \varphi(t_0)$. For $n = 0, 1, \dots$ and given $\mathbf{u}^n$, φ^n such that $\operatorname{div}\mathbf{u}^n = 0$, we find $\mathbf{u}^{n+1}$, p^{n+1}, φ^{n+1} in several steps:

The semi-Lagrangian step: $\Omega_n \to \Omega_{n+1}$. Consider the closest-point extension of the velocity at the boundary to the exterior of fluid domain: $\mathbf{u}^n|_{\Omega_n} \to \mathbf{u}^n|_{\mathbb{R}^3}$. In practice, the extension is performed to a bulk computational domain, rather than $\mathbb{R}^3$. For every $\mathbf{y} \in \mathbb{R}^3$, solve the characteristic equation backward in time

$$\frac{\partial \mathbf{x}(\tau)}{\partial \tau} = \widetilde{\mathbf{u}}^n(\mathbf{x}(\tau)), \quad \mathbf{x}(t_{n+1}) = \mathbf{y}, \quad \text{for } \tau \in [t_{n+1}, t_n]. \tag{9}$$

The mapping $\mathbf{X}$: $\mathbf{y} \to \mathbf{x}(t_n)$ defines an isomorphism on $\mathbb{R}^3$. Now, set

$$\varphi^{n+1}(\mathbf{y}) = \varphi^n(\mathbf{X}(\mathbf{y})). \tag{10}$$

For the numerical integration of (9) we apply the trapezoidal rule

$$\mathbf{x}(t_n + \frac{\Delta t}{2}) = \mathbf{x}_0 - \frac{\Delta t}{2}\mathbf{u}(\mathbf{x}_0, t_n), \quad \mathbf{x}(t_n) = \mathbf{x}_0 - \Delta t \widetilde{\mathbf{u}}^{n+\frac{1}{2}}, \tag{11}$$

with $\Delta t = t_n - t_{n+1}$. Since the velocity field is not given *a priori*, but recovered numerically at times t_k, $k = 0, \dots, n$, the linear extrapolation is used:

$$\widetilde{\mathbf{u}}^{n+\frac{1}{2}} = (1+\eta)\mathbf{u}(\mathbf{x}(t_n + \Delta t/2), t_n) - \eta\mathbf{u}(\mathbf{x}(t_n + \Delta t/2), t_{n-1}),$$
$$\eta = \frac{t_{n+1} - t_n}{t_n - t_{n-1}}.$$

To improve the accuracy of the semi-Lagrangian step, we apply the back-and-forth error compensation and correction (BFECC) technique from [11,12]: The same method is applied to integrate numerically the level-set equation forward in time to obtain an approximation to the error at time t_n. Further, the backward integration is performed one more time, but with the corrected level-set function values at time t_n. A tricubic interpolation is used to prescribe a value to φ^n at $\mathbf{X}(\mathbf{y})$. The interpolation is not monotone; therefore, a limiter is introduced to reduce oscillations. For smooth solutions, the method demonstrated second order of convergence for dynamically reconstructed meshes. Further details of the semi-Lagrangian BFECC method with a limiter on the octree grids can be found in [46].

After the completion of the semi-Lagrangian step, we perform the re-initialization of the level set function to satisfy equation (8). For this purpose, we use an algorithm from [31] based on the marching cubes method for free surface triangulation and a higher order closest point method. The numerical integration of (9) may also cause a divergence (loss or gain) of the fluid volume. So we perform the volume correction with the help of the procedure described in [31]. We note that the use of the BFECC method makes the re-initialization and volume correction steps less critical compared to the standard linear semi-Lagrangian method, but still they are necessary for long-time simulations.

Remeshing. Given the new fluid domain, we update and adapt the grid to account for the new position of the free surface. The adaptation is based on the information about the distance to the free surface provided by φ^{n+1}.

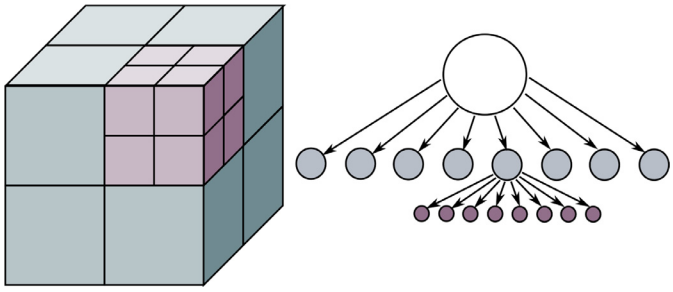


Fig. 1. An octree mesh (left) and its representation as a tree (right).

Re-interpolation. After remeshing we re-interpolate all discrete variables to the new grid. The re-interpolated velocity field is defined on the bulk computational domain (due to the extension procedure at the beginning of the level-set part).

Next we handle viscous and inertia terms and project the velocity into (discretely) divergence-free functions subspace and recover the new pressure. We denote $\Gamma_1 = \Gamma_D \cup \Gamma_{\text{in}}$, $\Gamma_2 = \Gamma(t_{n+1}) \cup \Gamma_{\text{out}}$.

The convection-diffusion step: Solve for $\widetilde{\mathbf{u}^{n+1}}$ in Ω_{n+1}:

$$\begin{cases} \frac{\alpha\widetilde{\mathbf{u}^{n+1}} + \beta\mathbf{u}^n + \gamma\mathbf{u}^{n-1}}{\Delta t_n} + (\mathbf{u}^n + \xi(\mathbf{u}^n - \mathbf{u}^{n-1}))\cdot\nabla\widetilde{\mathbf{u}^{n+1}} - \nu\Delta\widetilde{\mathbf{u}^{n+1}} = -\nabla p^n, \\ \widetilde{\mathbf{u}^{n+1}}|_{\Gamma_{\text{in}}} = \mathbf{u}_{\text{in}}, \quad \mathcal{B}\widetilde{\mathbf{u}^{n+1}}|_{\Gamma_D} = \mathbf{0}, \quad (\nabla\widetilde{\mathbf{u}^{n+1}} + \nabla\widetilde{\mathbf{u}^{n+1}}^T)\mathbf{n}\Big|_{\Gamma_2} = 0. \end{cases} \tag{12}$$

Here $\xi = \Delta t_n/\Delta t_{n-1}$, $\alpha = 1 + \xi/(\xi+1)$, $\beta = -(\xi+1)$, $\gamma = \xi^2/(\xi+1)$.

The projection step: Project $\widetilde{\mathbf{u}^{n+1}}$ on the divergence-free space to recover $\mathbf{u}^{n+1}$:

$$\begin{cases} \alpha(\mathbf{u}^{n+1} - \widetilde{\mathbf{u}^{n+1}})/\Delta t_n - \nabla q = 0, \\ \operatorname{div}\mathbf{u}^{n+1} = 0, \\ \mathbf{n}\cdot\mathbf{u}^{n+1}|_{\Gamma_1} = 0, \quad q|_{\Gamma_2} = 0. \end{cases} \tag{13}$$

The problem (13) is reduced to the Poisson problem for q:

$$\begin{cases} -\Delta q = \alpha/\Delta t_n \operatorname{div}\widetilde{\mathbf{u}^{n+1}}, \\ q|_{\Gamma_2} = 0, \quad \dfrac{\partial q}{\partial \mathbf{n}}\Big|_{\Gamma_1} = 0. \end{cases} \tag{14}$$

Finally, update the pressure:

$$p^{n+1} = p^n - q + \nu\operatorname{div}\widetilde{\mathbf{u}^{n+1}}. \tag{15}$$

The 'extra' divergence term in the pressure correction step (15) is used to reduce numerical boundary layers in the pressure, see, e.g., [16,41]. In this paper we do not address the problem of building a higher order accurate (with respect to the time step) stable pressure projection method for the case of open boundary conditions, cf. [16,24,35].

3.2. Spatial discretization

For the spatial discretization we use octree cubic meshes, which allow fast dynamic mesh adaptation based on geometric or error indicators.

Consider a graded octree mesh with cubic cells, see Fig. 1. An octree mesh is *graded* if the size of cells sharing (a part of) an edge or a face can differ in size only by the factor of two. This restriction simplifies support of mesh connectivity and the construction of discrete differential operators. We use the staggered location of velocity and pressure unknowns. The pressure degrees of freedom are assigned to cells centers and velocity variables are located at

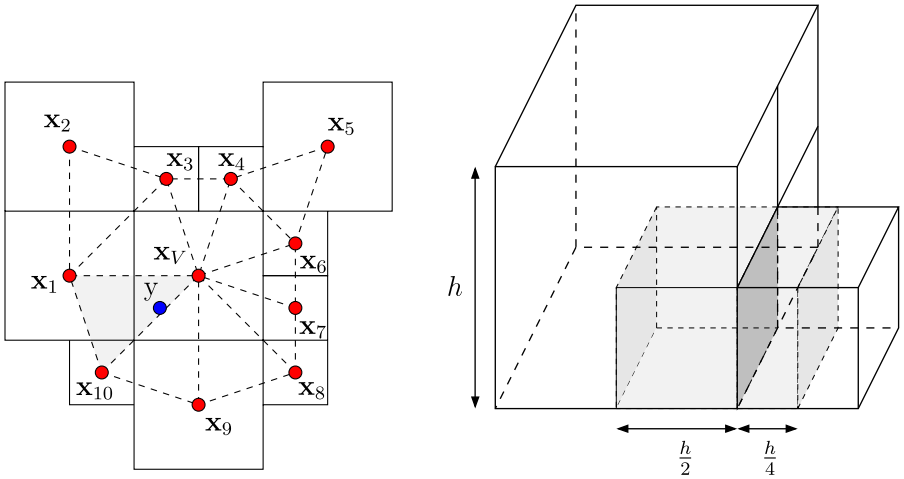


Fig. 2. Left: $u_h(\mathbf{y})$ is defined by a linear interpolation based on the fan triangulation with the center in $\mathbf{x}_V$, i.e. interpolation of $u_h(\mathbf{x}_V)$, $u_h(\mathbf{x}_1)$, $u_h(\mathbf{x}_{10})$ in this example. Right: the shaded region is the control volume V' associated with face F shared by cells of different sizes.

cells faces in such a way that every face stores normal velocity flux. If a face is shared by cells from different grid levels, then velocity degrees of freedom are assigned to the faces centers of fine grid cells (in the case of graded octree mesh, the corresponding face of the coarse grid cell holds 4 unknowns).

First, we describe how the advection and diffusion terms are treated in the interior of the computational domain. Several authors, e.g., [28,49], adopted semi-Lagrangian method to handle the time derivative and the inertia terms in finite difference discretizations of the momentum equations on the octree meshes. In [35] we found that semi-Lagrangian method on octree meshes can be either excessively diffusive or prone to instabilities for flows passing submerged objects. As an alternative, we consider a higher order upwind finite volume scheme on the graded octree meshes, which is both stable and accurate. Further details and the verification of the formal accuracy order of method can be found in thesis [45]. For the completeness of the presentation we describe the method below.

In several places further in the text we need an approximation of the grid velocity function $\mathbf{a}$ in an arbitrary point of the computational domain. For a given point $\mathbf{y}$ in the computational domain we evaluate $\mathbf{a}(\mathbf{y})$ as follows. Assume $\mathbf{y}$ belongs to a cell V and we are interested in interpolating the x-component of velocity to $\mathbf{y}$, i.e. $a_x(\mathbf{y})$. Consider a plane $\mathcal{P}$ such that $\mathbf{y} \in \mathcal{P}$ and $\mathcal{P}$ is orthogonal to the Ox axis. Let $\mathbf{x}_V \in \mathcal{P}$ be the orthogonal projection of the center of V on $\mathcal{P}$ and $\mathbf{x}_k$, $k = 1, \dots, m$, $m \le 12$, are the projections of centers of all cells sharing a face with V. The values $a_x(\mathbf{x}_V)$ and $a_x(\mathbf{x}_k)$ can be defined by a linear interpolation of the velocity values at nodes where a_x is collocated. Once $a_x(\mathbf{x}_V)$ and $a_x(\mathbf{x}_k)$, $k = 1, \dots, m$, are computed, we consider the triangle fan based on $\mathbf{x}_V$ and $\mathbf{x}_k$, $k = 1, \dots, m$, as shown in Fig. 2 (left). Now $a_x(\mathbf{y})$ is defined by a linear interpolation between the values of a_x in the vertices of the triangle, which contains $\mathbf{y}$. The proposed interpolation procedure is faster and produces smaller stencil compared to a straightforward least squares fitting of a polynomial to velocity values in a set of nodes.

For the incompressible fluid we treat the inertia terms in the 'conservative' form $\mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{div}\,(\mathbf{u} \otimes \mathbf{u})$, where the vector $\mathbf{div}$ operator applies row-wise. Eq. (12) of the splitting method linearizes the nonlinear terms, so that we need to approximate $\mathbf{div}\,(\mathbf{u} \otimes \mathbf{a})$ for a given nodal velocity $\mathbf{a} = (a_x, a_y, a_z)^T$ and unknown nodal velocity $\mathbf{u} = (u, v, w)^T$. Below we discuss the FV discretization of $\mathrm{div}\,(u\mathbf{a})$. Other two components of $\mathbf{div}\,(\mathbf{u} \otimes \mathbf{a})$ are treated similarly.

Consider the velocity component u at the x-node $\mathbf{x}_F$, which is the barycenter of the face F. If F is shared by the cells of different sizes, we define the control volume V' as shown in Fig. 2 (right). If F is shared by the cells of the same size, then V' is defined in the obvious way by merging two half-cells. Let $\mathcal{F}(V')$ denote the set of all faces for V'. We have

$$\mathrm{div}\,(u\mathbf{a})(\mathbf{x}_F) \approx |V'|^{-1} \sum_{F' \in \mathcal{F}(V')} |F'|\,(\mathbf{a} \cdot \mathbf{n})(\mathbf{x}_{F'})\,u(\mathbf{x}_{F'}). \tag{16}$$

We need to define advective fluxes at the barycenters $\mathbf{x}_{F'}$ of faces $F' \in \mathcal{F}(V')$.

First, we discuss the approximation of the advective flux at $F' \in \mathcal{F}(V')$ orthogonal to F. Consider F' orthogonal to Oy so that $\mathbf{a} \cdot \mathbf{n} = a_y$. If two cells sharing F have the same size, then $(\mathbf{a} \cdot \mathbf{n})(\mathbf{x}_{F'})$ is the simple averaging of a_y values from the two neighboring nodes. Otherwise $a_y(\mathbf{x}_{F'})$ is computed by the interpolation procedure described above. To define $u(\mathbf{x}_{F'})$, we take four 'reference' points $(\mathbf{x}_{-1}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_0 := \mathbf{x}_F)$ as shown in Fig. 3 (left). Note that $\mathbf{x}_{-1}$, $\mathbf{x}_1$, and $\mathbf{x}_2$ are not necessarily grid nodes. Values u_{-1}, u_1, and u_2 in these nodes are then defined based on the following interpolation procedure.

If the reference point belongs to a cell *smaller* than the cell of $\mathbf{x}_0$ (points $\mathbf{x}_1$ and $\mathbf{x}_2$ in the figure), then the linear interpolation between the *two* barycenters of adjunct faces is used. If the node belongs to a cell *larger* than the cell of $\mathbf{x}_0$ (point $\mathbf{x}_{-1}$ in the figure), then one apply the same interpolation procedure as we used above to define the values of $\mathbf{a}$. The only difference is that instead of the linear interpolation using the fan triangulation for $\mathbf{x}_V$ we use the weighted least-square method to fit the velocity values $u(\mathbf{x}_V)$ and $u(\mathbf{x}_k)$ by the second order polynomial Q_2, and further set $u(\mathbf{x}_{-1}) := Q_2(\mathbf{x}_{-1})$.

If $a_y(\mathbf{x}_F) > 0$, the u-values in reference points $\mathbf{x}_{-1}$, $\mathbf{x}_0$, $\mathbf{x}_1$ are used to approximate the flux. Otherwise, the u-values in the reference points $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$ are needed. Assume $a_y(\mathbf{x}_F) < 0$, we set

$$u(\mathbf{x}_{F'}) = D^{-1}\big[u_0(hH^2 - h^2H) + u_1(rH^2 + r^2H) - u_2(hr^2 + h^2r) + \lambda \Delta x^2 (u_0(H-h) - u_1(H+r) + u_2(r+h))\big], \tag{17}$$

where $D = (r+h)(H-r)(H+r)$. A family of formally second order upwind discretization is parameterized by $\lambda \in \mathbb{R}$. We found that $\lambda = 0$ (defining the QUICK scheme [25] on uniform meshes) produces the most accurate results on octree meshes and we use this value for numerical experiments.

Now, consider the approximation of the advective flux at $F' \in \mathcal{F}(V')$ parallel to F, hence $\mathbf{a} \cdot \mathbf{n} = a_x$.

After prescribing $a_x(\mathbf{x}_{F'})$ value with the help of the linear interpolation at the corresponding faces of the control volume, we

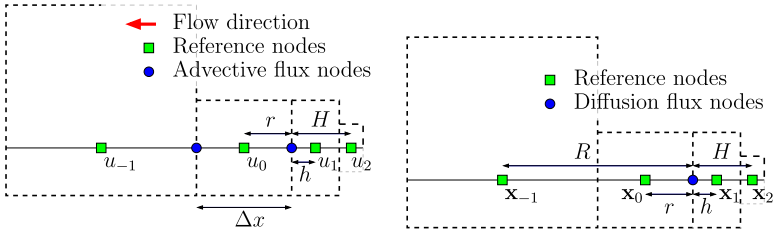


Fig. 3. Left: reference points for the upwind approximation of advection. This illustration is for the derivative tangential to a face F, where the velocity degree of freedom is located. Right: reference points for the diffusion flux approximation.

define $u(\mathbf{x}_{F'})$ using (17). The only differences with the treatment of the face F' orthogonal to Oy are the following: a_x is defined in $\mathbf{x}$ (no interpolation required), and the reference points $\mathbf{x}_{-1}$, $\mathbf{x}_1$, $\mathbf{x}_2$ are always lying on cells x-faces (although not necessarily in the centers and one has to do the interpolation).

Next, we explain how the discretization of viscous terms is computed. Consider a node $\mathbf{x}$ holding the velocity component u and lying on a face F and define a cubic control volume V' such that $\mathbf{x}$ is the center of V' and F is a middle cross section of V'. Note that the control volumes for x-nodes do not overlap, but for locally refined mesh they do not necessarily cover the whole bulk domain. Hence the dicretization of the viscous terms is a finite difference method, rather than a finite volume method. We have

$$(\Delta_h u)(\mathbf{x}) \approx |V'|^{-1} \sum_{F' \in \mathcal{F}(V')} |F'| (\nabla_h u \cdot \mathbf{n})(\mathbf{y}_{F'}). \tag{18}$$

To approximate the diffusion flux at the center $\mathbf{y}_{F'}$ of $F' \in \mathcal{F}(V')$, we take four reference points ($\mathbf{x}_{-1}$, $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$) as shown in Fig. 3 (right). Velocity values u_{-1}, u_0, u_1, and u_2 are assigned to reference points same way as for the advective terms described above. Using the notation from Fig. 3, the formal third order approximation of the diffusion flux density $(\nabla u \cdot \mathbf{n})$ can be written out as

$$\begin{aligned}(\nabla u \cdot \mathbf{n}) \approx D^{-1}\big[&(h^2H^3 + h^3R^2 - H^3R^2 + h^2R^3 - H^2R^3 - h^3H^2)u_0 \\ &+ (H^3R^2 + r^3R^2 + H^2R^3 - r^2R^3 - H^3r^2 - H^2r^3)u_1 \\ &+ (h^3r^2 + h^2r^3 - h^3R^2 - r^3R^2 - h^2R^3 + r^2R^3)u_{-1} \\ &+ (h^3H^2 - h^2H^3 - h^3r^2 + H^3r^2 - h^2r^3 + H^2r^3)u_2\big],\end{aligned} \tag{19}$$

with $D = (H-h)(h+r)(H+r)(h+R)(H+R)(R-r)$. If the reference point in $\mathbf{x}_2$ is not available, we use the point $\mathbf{x}_{-2}$.

To enforce incompressibility condition, we approximate $\operatorname{div}\mathbf{u}$ in the center $\mathbf{x}_V$ of a grid cell V. We define the grid divergence operator by

$$(\operatorname{div}_h \mathbf{u}_h)(\mathbf{x}_V) = |V|^{-1} \sum_{F \in \mathcal{F}(V)} |F| (\mathbf{u}_h \cdot \mathbf{n})(\mathbf{x}_F). \tag{20}$$

Thanks to the staggered location of velocity nodes, the fluxes $(\mathbf{u}_h \cdot \mathbf{n})(\mathbf{x}_F)$ are well-defined.

One way to introduce the discrete gradient is to define it as the adjoint of the discrete divergence. We found that an approximation of ∇_h based on the formal Taylor expansions gives more accurate results. For every internal face we assign the corresponding component of $\nabla_h p$ as follows. Since the octree mesh is graded, there can be only two geometric cases. If a face is shared by two equal-size cells, then the central difference approximation is used. Otherwise, for the approximation of p_x at the face center node $\mathbf{y}$ one considers the centers of five surrounding cells $\mathbf{x}_1, \ldots, \mathbf{x}_5$ and expand the pressure value $p(\mathbf{x}_i)$ with respect to $p(\mathbf{y})$:

$$p(\mathbf{x}_i) = p(\mathbf{y}) + \nabla p(\mathbf{y}) \cdot (\mathbf{x}_i - \mathbf{y}) + O(|\mathbf{x}_i - \mathbf{y}|^2).$$

Neglecting the second-order terms, we obtain the following overdetermined system:

$$\begin{pmatrix} 1 & -\Delta/2 & \Delta/4 & \Delta/4 \\ 1 & \Delta/4 & 0 & 0 \\ 1 & \Delta/4 & \Delta/2 & 0 \\ 1 & \Delta/4 & 0 & \Delta/2 \\ 1 & \Delta/4 & \Delta/2 & \Delta/2 \end{pmatrix} \begin{pmatrix} p(\mathbf{y}) \\ p_x(\mathbf{y}) \\ p_y(\mathbf{y}) \\ p_z(\mathbf{y}) \end{pmatrix} = \begin{pmatrix} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ p(\mathbf{x}_3) \\ p(\mathbf{x}_4) \\ p(\mathbf{x}_5) \end{pmatrix}, \tag{21}$$

where $\Delta \equiv \Delta x$. The least squares solution of (21) gives the stencil for the x-component of the gradient:

$$p_x(\mathbf{y}) \approx \frac{1}{3\Delta}(p_2 + p_3 + p_4 + p_5 - 4p_1). \tag{22}$$

The superposition of the discrete gradient and divergence operators generally leads to the non-symmetric matrix for the pressure problem. However, the corresponding linear algebraic systems are solved efficiently by a Krylov subspace method with a two-parameter threshold ILU preconditioner [22,23]. We note that in general non-symmetric FV approximations of diffusion equations may lead to the lack of coercivity and hence to stability issues, cf. [10], although symmetric and coercive approximations can produce unstable solutions as well, cf. Fig. 4 in [10]. The previous studies, e.g., [28,32,35,39], show that using the present non-symmetric approximations of the pressure Poisson equation does not disrupt the stability of projection methods.

It was noted in [35] for octree staggered grids, that the discrete Helmholtz decomposition, which essentially constitutes the projection step of the splitting scheme, is unstable due to oscillatory spurious *velocity* modes tailored to course-to-fine grid interfaces. If the viscosity is sufficiently large, then such modes are suppressed, otherwise they propagate and destroy the accuracy of numerical solution. Following that paper we apply a technique, which eliminates the spurious modes and improves the accuracy of numerical solution significantly.

The constructed spatial discretization is hybrid: a finite volume method was used to handle the incompressibility constraint and inertia terms, while a finite difference method was applied to diffusion terms and pressure gradient. To solve the velocity equation on each time step, we use BiCGStab(2) [43] iteration with a two-parameter threshold ILU preconditioner [22,23]. This combination of the Krylov subspace method and the preconditioner resulted in a robust and efficient solver.

3.3. Boundary conditions and curvilinear boundaries

The discretization method in Section 3.2 assumes that velocity values in all nodes forming flux stencils are given. When all the cubic volumes in the stencil are internal, then all corresponding velocity values are treated as active degrees of freedom. Close-to-boundary cells require special treatment. Below we introduce such a treatment when a curvilinear boundary is immersed in the background octree mesh.

For the computational purposes, the static boundary is defined with the help of a signed distance function φ_s. We assume that

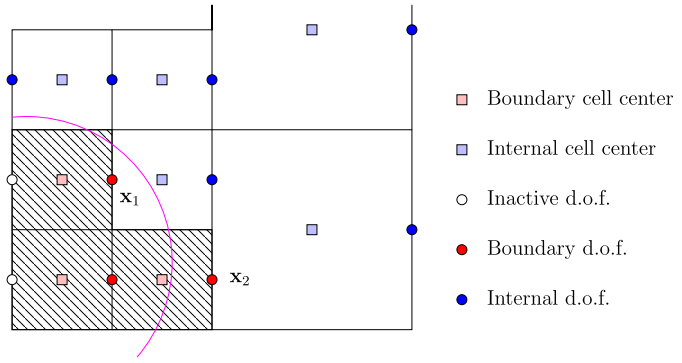


Fig. 4. Internal (blue), boundary (red) and inactive (white) nodes near the curvilinear boundary. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the static boundary consists of several smooth components. Each component is described by its own φ_s (domains of definition of the level set functions may overlap). This is similar to the description of the free surface, but φ_s is defined by the domain geometry and does not vary in time. We assume that $\varphi_s < 0$ in the fluid domain Ω, and $\varphi_s > 0$ in the exterior, so the boundary is given as the zero isosurface of function φ_s. Denote by $\mathcal{T}_h$ the background octree mesh, the collection of cubic volumes forming the tessellation of the bulk computational domain. For each $V \in \mathcal{T}_h$, $\mathbf{p}_V$ denotes the barycenter of V. We divide $\mathcal{T}_h$ into the sets of internal, boundary and external cells:

$$\mathcal{T}_{int} := \{V \in \mathcal{T}_h : \varphi_s(\mathbf{p}_V) \le -h_{thr}\},$$
$$\mathcal{T}_{bdr} := \{V \in \mathcal{T}_h : \exists K \in \mathcal{T}_{int},\ s.t.\ \overline{V} \cap \overline{K} \neq \emptyset\},$$
$$\mathcal{T}_{ext} := \mathcal{T}_h \setminus (\mathcal{T}_{int} \cup \mathcal{T}_{bdr}),$$

where $h_{thr} = h_V/10$ is a threshold parameter. Based on this splitting we also divide all velocity nodes on $\mathcal{T}_h$ into three groups. Denote by $\mathcal{N}_h$ the collection of all velocity nodes from the bulk computational mesh. The nodes on the boundary of the bulk domain are not active. Any other node $\mathbf{x} \in \mathcal{N}_h$ has exactly two cells $V_{\mathbf{x}}^1$ and $V_{\mathbf{x}}^2$ such that $\mathbf{x} \in \bar{V}_{\mathbf{x}}^1 \cap \bar{V}_{\mathbf{x}}^2$. Now we divide $\mathcal{N}_h$ into the sets of internal, boundary and external nodes:

$$\mathcal{N}_{int} := \{\mathbf{x} \in \mathcal{N}_h : V_{\mathbf{x}}^1 \in \mathcal{T}_{int} \text{ and } V_{\mathbf{x}}^2 \in \mathcal{T}_{int}\},$$
$$\mathcal{N}_{bdr} := \{\mathbf{x} \in \mathcal{N}_h : V_{\mathbf{x}}^1 \in \mathcal{T}_{bdr} \text{ or } V_{\mathbf{x}}^2 \in \mathcal{T}_{bdr}\},$$
$$\mathcal{N}_{ext} := \mathcal{N}_h \setminus (\mathcal{N}_{int} \cup \mathcal{N}_{bdr}).$$

The velocity degrees of freedom are assigned to the internal nodes and boundary nodes, i.e. those from $\mathcal{N}_{int} \cup \mathcal{N}_{bdr}$. There is a difference, however, how the method works for the nodes from $\mathcal{N}_{int}$ and $\mathcal{N}_{bdr}$: For each node from $\mathcal{N}_{int}$ we have a set of algebraic equations derived in the previous section, while each node from $\mathcal{N}_{bdr}$ receives an auxiliary equation based on boundary conditions. The nodes from $\mathcal{N}_{ext}$ are not active. This subdivision of velocity nodes into three groups based on the position of the immersed boundary is illustrated in Fig. 4 (the figure shows a 2D mesh and only nodes for the horizontal velocity component).

Now we derive equations for the nodes from $\mathcal{N}_{bdr}$. For the Dirichlet boundary condition $\mathbf{u} = \mathbf{u}_b$ on the immersed boundary, this is done componentwise as follows. For each boundary node $\mathbf{x}$ either interpolation or extrapolation procedure is performed depending on the sign of $\varphi_s(\mathbf{x})$.

For $\varphi_1 = \varphi_s(\mathbf{x}_1) > 0$ (the node $\mathbf{x}_1$ is outside the domain Ω) we apply extrapolation, cf. Fig. 5 (left):

$$u(\mathbf{x}_1) = \frac{d_1 + \varphi_1}{d_1} u_b(\mathbf{x}_1^b) - \frac{\varphi_1}{d_1} u(\mathbf{x}_1^v), \tag{23}$$

where $\mathbf{x}_1^b$ is the closest boundary point to $\mathbf{x}_1$, $d_1 = \max(\varphi_1, h_{V_1})$ is an outstep to the internal domain, and $\mathbf{x}_1^v$ is a virtual node belonging to line passing through $\mathbf{x}_1$ and $\mathbf{x}_1^b$, and $|\mathbf{x}_1 - \mathbf{x}_1^v| = d_1$. The velocity value is interpolated to $\mathbf{x}_1^v$ from internal velocity degrees of freedom.

For $\varphi_2 = \varphi_s(\mathbf{x}_2) < 0$ (the node $\mathbf{x}_2$ is inside the domain Ω) we set

$$u(\mathbf{x}_2) = \frac{d_2}{d_2 - \varphi_2} u_b(\mathbf{x}_2^b) - \frac{\varphi_2}{d_2 - \varphi_2} u(\mathbf{x}_2^v), \tag{24}$$

where $\mathbf{x}_2^b$ is the closest boundary point to $\mathbf{x}_2$, $d_2 = \max(-\varphi_2, h_{V_2})$ is an outstep to the external domain and $\mathbf{x}_2^v$ is a virtual node belonging to line $(\mathbf{x}_2, \mathbf{x}_2^b)$, $|\mathbf{x}_2 - \mathbf{x}_2^v| = d_2$. Again the velocity value is interpolated to $\mathbf{x}_2^v$ from internal velocity degrees of freedom.

For the free-slip boundary condition we use the approach similar to the no-slip condition. Consider the boundary node $\mathbf{x}_1$ and the virtual point $\mathbf{x}_1^v$ with all velocity components $\mathbf{u}(\mathbf{x}_1^v)$ interpolated in it, see Fig. 5 (right). First, we write down the set of equations for $\mathbf{x}_1$ assuming for a moment that all three components of $\mathbf{u}$ are defined in $\mathbf{x}_1$. Thus, we seek for $\mathbf{u}(\mathbf{x}_1)$ such that interpolated (or extrapolated) boundary value $\mathbf{u}(\mathbf{x}_1^b)$ has the normal component vanishing and tangential components equal to those in the internal virtual node. This yields the following equations

$$\begin{cases} \mathbf{u}(\mathbf{x}_1^b) = \frac{d_1}{\varphi_1 + d_1}\mathbf{u}(\mathbf{x}_1) + \frac{\varphi_1}{\varphi_1 + d_1}\mathbf{u}(\mathbf{x}_1^v), \\ \mathbf{u}(\mathbf{x}_1^b) \cdot \mathbf{n} = 0, \\ \mathbf{u}(\mathbf{x}_1^b) - (\mathbf{u}(\mathbf{x}_1^b) \cdot \mathbf{n})\,\mathbf{n} = \mathbf{u}(\mathbf{x}_1^v) - (\mathbf{u}(\mathbf{x}_1^v) \cdot \mathbf{n})\,\mathbf{n}, \end{cases}$$

where $\mathbf{n}$ is the unit normal vector for the boundary in point $\mathbf{x}_1^b$.

Substituting the first and the second equations in the third one, we get the equation for $\mathbf{u}(\mathbf{x}_1)$:

$$\mathbf{u}(\mathbf{x}_1) = \mathbf{u}(\mathbf{x}_1^v) - \frac{\varphi_1 + d_1}{\varphi_1}(\mathbf{u}(\mathbf{x}_1^v) \cdot \mathbf{n})\,\mathbf{n}. \tag{25}$$

The final equation tailored to the node $\mathbf{x}_1$ follows by extracting only one equality from (25). This equality corresponds to the component of $\mathbf{u}$ located at $\mathbf{x}_1$.

The boundary condition (4) on the free surface and Γ_{out} is decomposed into the homogeneous Neumann boundary condition for the auxiliary velocity in the convection-diffusion step (12) and the homogeneous Dirichlet boundary condition for the pressure correction variable q in (15). For the pressure Dirichlet condition the missing values at the barycenters of boundary cells are recovered by the same technique as Dirichlet velocity values for the boundary with the no-slip condition. Therefore, the pressure field is known in all close-to-free-boundary cells and the pressure update (15) is well defined in cells from $\mathcal{T}_{int}$, which did not belong to $\mathcal{T}_{int}$ at time t_n. The Neumann velocity boundary condition is enforced in the same way as the slip-condition on Γ_D. Of course, no-penetration condition does not apply in this case.

Note that boundary nodes receive velocity values *implicitly* through Eqs. (23), (24), or (25). These equations are added to the global system of algebraic equations. To obtain a complete system, we need to discretize the momentum and continuity equations in all cut cells. To this end, we first extend the density and viscosity coefficients by the same constant values from the cut cells to the whole cubic cells. Next, we apply the "full-cell" expressions in (16), (18), and (20) to define discrete operators for the cut cells. Due to the linear extrapolation of boundary conditions, the resulting differences approximate the required differential operators.

Poisson equation for the pressure correction q of the projection step involves degrees of freedom at pressure nodes, i.e. at barycenters of cells from $\mathcal{T}_h$. We solve for the pressure degrees of freedom only for cells from $\mathcal{T}_{int}$. Thus, the discrete gradient is well defined at all velocity nodes from $\mathcal{N}_{int}$ with the help of the pressure correction values at $\mathcal{T}_{int}$. The discrete gradient at the nodes from $\mathcal{N}_{bdr}$ for Γ_2 is also well defined with the help of internal degrees of freedom and zero Dirichlet values for the pressure correction in the free-boundary cells. To assign the gradient of the pressure correction to the nodes from $\mathcal{N}_{bdr}$ for Γ_1, we proceed as follows: From

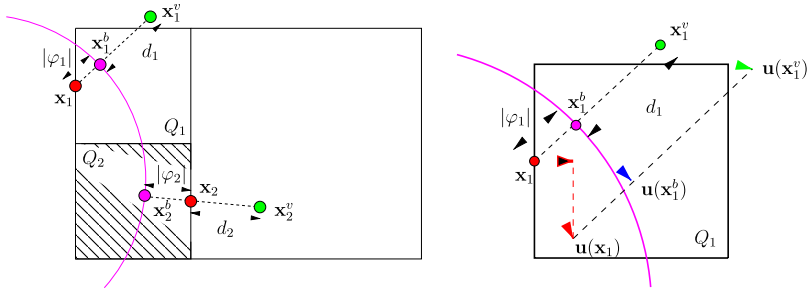


Fig. 5. Left: extrapolation of velocity values at the boundary node $\mathbf{x}_1$ and interapolation at the boundary node $\mathbf{x}_2$ near the curvilinear boundary; Right: extrapolation of the velocity values at the boundary node $\mathbf{x}_1$ for the free-slip boundary condition.

(13) we get $\mathbf{u}^{n+1} = \widetilde{\mathbf{u}^{n+1}} + \triangle t_n/\alpha \nabla q$. The variable $\widetilde{\mathbf{u}^{n+1}}$ receives its values in all nodes from $\mathcal{N}_{bdr}$ during the predictor step (12) of the splitting algorithm. Further we substitute the equality in the corresponding equations from (23)–(25) for $\mathbf{u}^{n+1}$ and this yields the equation for ∇q in the boundary nodes. Further we build the pressure Laplace operator as the superposition of the gradient (22) and divergence (20) grid operators.

Remark 1. A rigorous stability analysis of the hybrid method is an open question. We note that stability of the semi-discrete scheme from Section 3.1 (only discretization in time) for free-surface flows was studied in [31]. The scheme was shown to conserve global momentum and angular momentum, and based on that an energy inequality was shown to hold. Thorough numerical studies of the stability and numerical dissipation of the method for the case of enclosed flows (no free boundary) and fitted boundary conditions (no curvilinear boundaries) was done in [35]. In that paper, the method was shown stable for a vast range of flows (from laminar to developed turbulent); it was shown to have lower numerical diffusion compared to some alternative approaches on octree meshes. The numerical results of the present paper suggest that this stability property extends to flow problems with free boundaries and streamlined bodies.

4. Numerical experiments

Our first series of numerical experiments aims to assess the stability of the presented method, its lower dissipation and ability to handle free surface evolution accurately. To this end, we consider several standard benchmark problems.

The first two benchmark tests deal with laminar flows around a 3D cylinder of circular cross-section at Re = 20 and varying Reynolds number. This problem does not require a dynamic adaptation of the octree mesh. Our goal here is to check the accuracy of the scheme in a domain with curved boundary by comparing computed drag and lift coefficients with those found in the literature. These statistics are known to be sensitive to excessive numerical dissipation of a numerical method.

The lateral sloshing tank benchmark verifies the ability of the scheme to reproduce complex dynamics of fluid free surface. The correctly recovered free surface evolution after the termination of excitation forces is another indicator of the scheme reliability and low numerical dissipation. Dynamic mesh adaptation is very helpful in this problem.

After validation of the numerical scheme, we apply it to simulate a water flow with surface waves around an oil platform rigidly mounted in the Kara sea offshore.

4.1. Flow around cylinder of circular cross-section

The first numerical test is the laminar 3D channel flow around a cylinder of circular cross-section. The problem was suggested as a benchmark by Schäfer and Turek in [42] and further studied in, e.g., [5,6,19].

The flow domain is shown in Fig. 6. The no-slip and no-penetration boundary condition $\mathbf{u} = 0$ is prescribed on the channel walls and the cylinder surface. For the outflow boundary conditions we put the normal component of the stress tensor equal zero on Γ_{out}. The parabolic velocity profile is set on the inflow boundary:

$$\mathbf{u} = (0, 0, 16\widetilde{U}xy(H-x)(H-y)/H^4)^T \quad \text{on } \Gamma_{\text{in}},$$

with $H = 0.41$ and a peak velocity $\widetilde{U}$. The Reynolds number, $Re = \nu^{-1}D\widetilde{U}$, is defined based on the cylinder width $D = 0.1$. The viscosity coefficient ν is set to 10^{-3}. We consider two benchmark tests from [42]:

- Problem Z1: Steady flow with $Re = 20$ ($\widetilde{U} = 0.45$);
- Problem Z3: Unsteady flow with varying Reynolds number for $\widetilde{U} = 2.25\sin(\pi t/8)$.

The initial condition for both problems is $\mathbf{u} = 0$ for $t = 0$.
The following statistics are of interest:

- The difference $\Delta p = p(\mathbf{x}_2) - p(\mathbf{x}_1)$ between the pressure values in points $\mathbf{x}_1 = \{0.2, 0.205, 0.55\}$ and $\mathbf{x}_2 = \{0.2, 0.205, 0.45\}$.
- The drag coefficient given by an integral over the surface of the cylinder S:

$$C_{\text{drag}} = \frac{2}{DH\widetilde{U}^2}\int_S \left(\nu\frac{\partial(\mathbf{u}\cdot\mathbf{t})}{\partial\mathbf{n}}n_x - pn_z\right)ds. \tag{26}$$

  Here $\mathbf{n} = (n_x, n_y, n_z)^T$ is the normal vector to the cylinder surface pointing to Ω and $\mathbf{t} = (-n_z, 0, n_x)^T$ is a tangent vector.
- The lift coefficient given by an integral over the surface of the cylinder:

$$C_{\text{lift}} = -\frac{2}{DH\widetilde{U}^2}\int_S \left(\nu\frac{\partial(\mathbf{u}\cdot\mathbf{t})}{\partial\mathbf{n}}n_z + pn_x\right)ds. \tag{27}$$

The octree mesh is refined locally towards the channel walls (we set $h_{wall} = \ell/256$ except the coarsest mesh where $h_{wall} = \ell/128$, $\ell = 2.5m$ is the length of the computational domain) and the circular cylinder ($h_{\min}$ in this experiment denotes the mesh size near the cylinder). The cutaway of the mesh with $h_{min} = \ell/64$ and $h_{max} = \ell/1024$ is shown in Fig. 7.

To compute the drag and lift coefficients, we replace the surface integrals in (26) and (27) by integration over the whole domain [6,19]: Assume $\mathbf{u} = (u, v, w)^T$ and p is the Navier–Stokes solution in a fixed domain Ω, then applying the integration by parts one

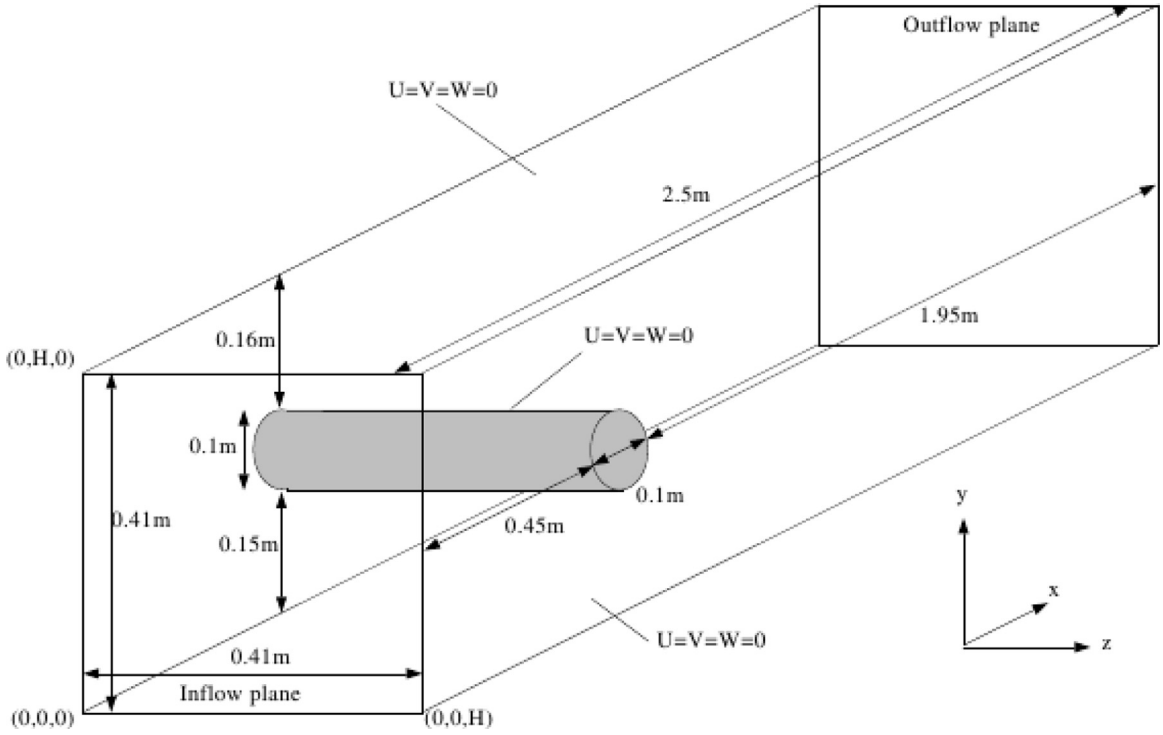


Fig. 6. Computational domain for flow around cylinder of circular cross-section.

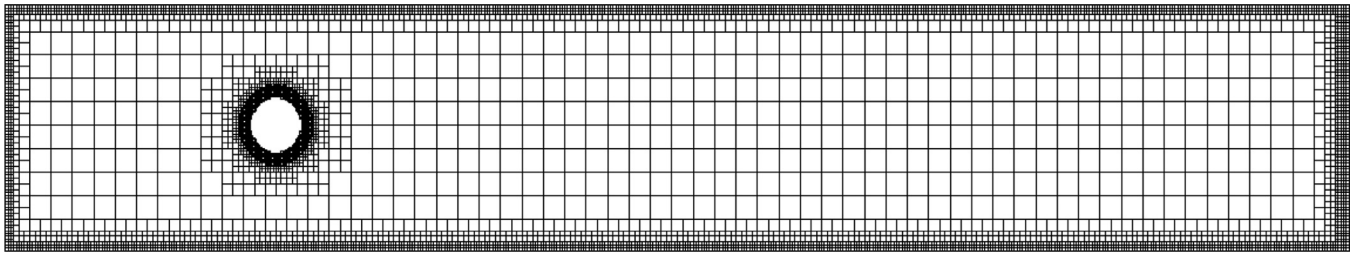


Fig. 7. The cutaway of the grid at $y = 0.205$ for $h_{max} = \ell/64$ and $h_{min} = \ell/1024$.

checks the following identities:

$$C_{\text{drag}} = \widetilde{C} \int_\Omega \left[\left(\frac{\partial w}{\partial t} + (\mathbf{u} \cdot \nabla) w \right) \varphi + \nu \nabla w \cdot \nabla \varphi - p \partial_z \varphi \right] \mathrm{d}\mathbf{x}$$
$$C_{\text{lift}} = \widetilde{C} \int_\Omega \left[\left(\frac{\partial u}{\partial t} + (\mathbf{u} \cdot \nabla) u \right) \varphi + \nu \nabla u \cdot \nabla \varphi - p \partial_x \varphi \right] \mathrm{d}\mathbf{x}, \tag{28}$$

$\widetilde{C} = \frac{2}{DH\bar{U}^2}$, for any $\varphi \in H^1(\Omega)$ such that $\varphi|_S = 1$ and $\varphi|_{\partial\Omega/S} = 0$. The accuracy of evaluation of (28) for a numerical solution depends on the regularity of φ. In our numerical scheme φ is defined in pressure nodes as the discrete harmonic function solving $\mathrm{div}_h \nabla_h \varphi = 0$. The derivatives in (28) are approximated with the second order of accuracy. Using the volume based formulas (28) gives more accurate values of drag and lift coefficients compared to (26) and (27), if the Navier–Stokes solution is sufficiently smooth, see [6].

The numerical solutions to problem Z1 were computed on a sequence of locally refined meshes, see Table 1 for the information of the corresponding discrete space dimensions. Note that we refine the mesh sequence towards the cylinder and keep it coarser in the wake. Such refinement is known to be crucial for accurate computation of the statistics of interest, see, for example [6,35].

The reference [42] collects several DNS results based on various finite element, finite volume discretizations of the Navier–Stokes equations and the Lattice Boltzmann method. One can find there reference intervals where the statistics of interest should converge. Using a higher order finite element method and locally refined adaptive meshes, more accurate reference values of C_{drag} and C_{lift} are found in [6] for problem Z1. For a sequence of locally refined octree meshes, Table 2 demonstrates the convergence of computed drag and lift coefficients, and pressure drop value to reference intervals.

For problem Z3 less accurate reference data is available. Table 3 summarizes the results computed by the present method and those available in the literature [1,42]. The values of $\max C_{\text{drag}}$, $\max C_{\text{lift}}$ and $\min C_{\text{lift}}$ are the maximum drag and maximum/minimum lift coefficients over the whole time interval $t \in [0, 8]$, the pressure drop Δp is computed at $t = 8$. The most sensitive statistics are C_{lift} and Δp. Table 3 shows their convergence to the

Table 1
The number of velocity and pressure d.o.f. for different meshes for problem Z1.

h_{min}	h_{max}	u d.o.f.	p d.o.f.
$\ell/128$	$\ell/64$	175,126	65,002
$\ell/256$	$\ell/64$	855,529	304,395
$\ell/512$	$\ell/64$	925,177	338,997
$\ell/1024$	$\ell/64$	1346577	524,983

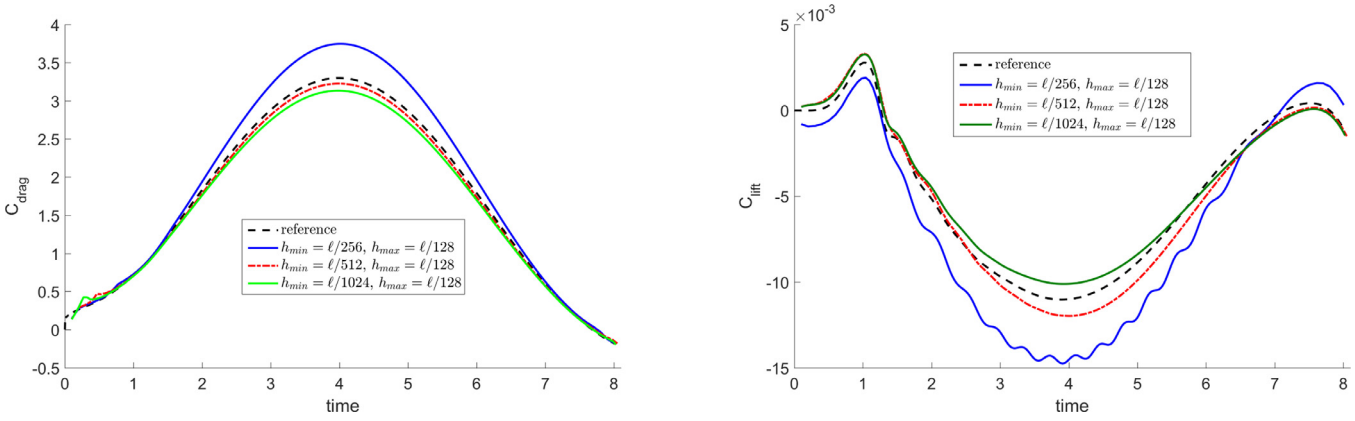


Fig. 8. Computed drag (left) and lift (right) coefficients dependence on time versus reference data from [1].

Table 2
Problem Z1: convergence of drag, lift, and pressure drop to reference intervals.

h_{min}	h_{max}	C_{drag}	C_{lift}	Δp
$\ell/128$	$\ell/64$	3.07235	−0.019821	0.13840
$\ell/256$	$\ell/64$	6.20151	0.00778	0.15961
$\ell/512$	$\ell/64$	6.15078	0.00962	0.16298
$\ell/1024$	$\ell/64$	6.14193	0.00990	0.16636
Braack & Richter		6.18533	0.009401	
Schäfer & Turek		6.05–6.25	0.008–0.01	0.165–0.175

Table 4
Dimensional and non-dimensional parameters in the sloshing benchmark.

Value	Dimensional	Non-dimensional
Lengths	$D = 0.3$ m	$\tilde{D} = 1.0$
	$H = 0.1$ m	$\tilde{H} = 0.3333$
	$W = 0.8$ m	$\tilde{W} = 2.6667$
Frequency	$f = 0.89$ s^{-1}	$\tilde{f} = 1.0$
Acceleration	$g = 9.81$ ms^{-2}	$\tilde{g} = 1.0$
Viscosity	$\nu = 1.0 \times 10^{-6}$ m^{2}s^{-1}	$\tilde{\nu} = 1.943 \times 10^{-6}$

reference intervals. The value of the maximum drag coefficient on the finest mesh is slightly (3%) less than the reference one.

In Fig. 8 we compare the computed curves $C_{\text{drag}}(t)$, $C_{\text{lift}}(t)$ with reference data from [1]. The computed coefficients fit the reference data reasonably well. For a better fitting, a stronger mesh refinement is needed which exceeds our computing capabilities.

4.2. Sloshing tank

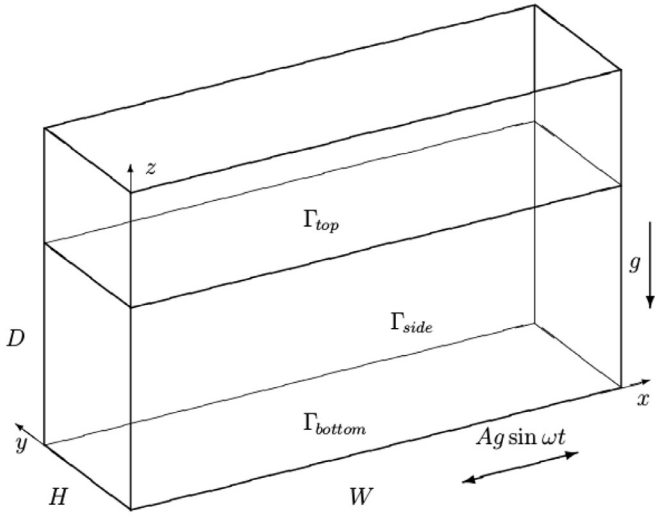
The sloshing of fluid in a tank is a benchmark problem for numerical free surface flow solvers and a problem of independent interest, see, e.g., [3,8,13,18,33,48]. The setup of the sloshing tank problem is given in [3,18]. A volume of water fills a rectangular tank as illustrated in Fig. 9. The initial bulk dimensions are $W = 0.8$ m, $H = 0.1$ m and $D = 0.3$ m. The container walls Γ_{bottom} and Γ_{side} impose slip and no-penetration conditions for the fluid. The fluid is exposed to external forces: a constant gravitational acceleration of magnitude $g = 9.81$ ms^{-2} and a sinusoidal horizontal excitation $Ag\sin\omega t$ with $A = 0.01$ and $\omega = 2\pi f$, $f = 0.89$ Hz. The problem is non-dimensionalized following [18]. The full set of dimensional and non-dimensional parameters is summarized in Table 4.

The sloshing motion is initiated as soon as the horizontal excitation is applied. After the initial ten periods the excitation is terminated. The excitation frequency is designed to induce the first mode of wave motion in the x direction, i.e., the motion with a wavelength approximately equal the doubled width of the tank W. The time histories for the height of the wave at the two opposite tank walls orthogonal to the x-axis are shown in Fig. 9 (right). These data were computed for the 2D setting of the problem in [18]. These results are believed to correspond well to physical observations [33].

The octree FV method recovers correctly time dependence of the water level at the midline of the left wall ($x = -D/2$), see Fig. 10 (left). For the first ten periods of excitation the measured wave height matches the heights reported in [18] with the deviations less than 4%. Numerical dissipation is low enough to avoid amplitude dumping after termination of the excitations even on relatively coarse meshes. The mesh convergence of the free surface contact line evolution on the wall at $x = -D/2$ is demonstrated in Fig. 10 (right). The meshes are refined dynamically to the tank walls up to the meshsize h_{wall} and to the free surface up to the meshsize $h_{\min}$, the coarsest cell size is fixed $h_{\max} = \ell/8$, here $\ell = W$. At the "Remeshing" step of the splitting method we refine all cubic cells intersected by the zero level set of $\varphi(t^{n+1})$ so that all these cells have the width $h_{\min}$. All other cells except boundary cells are marked for coarsening. The coarsening is performed in such a way that the octree remains balanced (two neighboring cells may differ in size at most by a factor of two) and the maximum cell width in the fluid domain is $h_{\max}$. The following combinations of the mesh refinements were used: $h_{wall} = h_{\min} = \ell/64$,

Table 3
Problem Z3: maximum drag, maximum/minimum lift, and pressure drop (at $t = 8$) and reference intervals. Time steps subject to $\Delta t_k = 5h_{\min}/\max_{\mathbf{x}}|\mathbf{u}(\mathbf{x}, t_k)|$.

h_{min}	h_{max}	$\max C_{\text{drag}}$	$\max C_{\text{lift}}$	$\min C_{\text{lift}}$	$\Delta p(t = 8)$
$\ell/256$	$\ell/128$	3.74685	0.00190	−0.01474	−0.09740
$\ell/512$	$\ell/128$	3.22627	0.00329	−0.01197	−0.12083
$\ell/1024$	$\ell/128$	3.13382	0.00325	−0.01011	−0.11933
Bayraktar & Mierka & Turek		3.29–3.33	0.0027–0.0033	−0.010– −0.012	
Schäfer & Turek		3.2–3.3	0.002–0.004		−0.14– −0.12



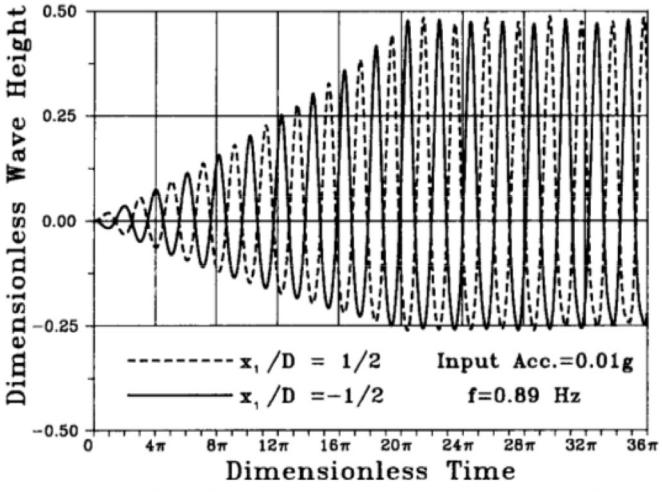
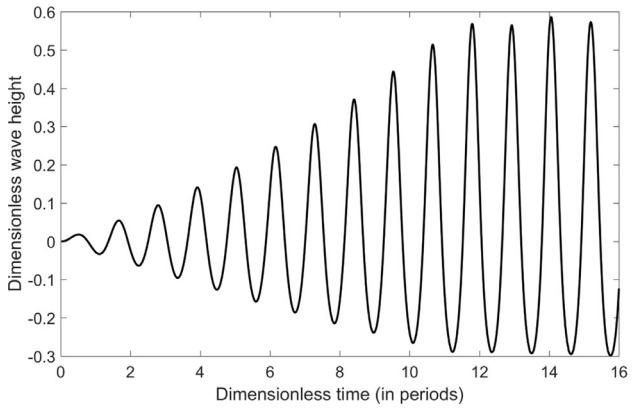


Fig. 27. Time histories of the wave heights at the wall.

Fig. 9. Left: problem setup for sloshing tank test. Right: wave height at the mid-lines of two opposite tank walls from [18].



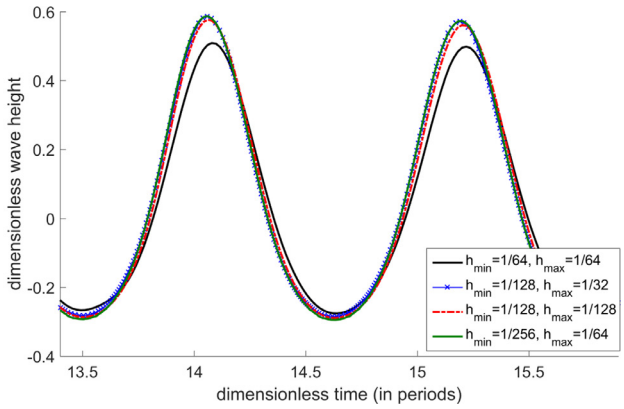


Fig. 10. Computed water level at the midline of the left wall ($x = -D/2$). Left plot shows the evolution computed with the $h_{wall} = \ell/64$, $h_{\min} = \ell/256$. Right plot demonstrates the mesh-convergence of the results.

$h_{wall} = h_{\min} = \ell/128$, $h_{wall} = \ell/32$, $h_{\min} = \ell/128$, and $h_{wall} = \ell/64$, $h_{\min} = \ell/256$. For this problem we use adaptive time step, $\Delta t_k = \min\{0.0187, h_{\min}/\max_{\mathbf{x}}|\mathbf{u}(\mathbf{x}, t_k)|\}$.

Fig. 11 demonstrates the same pattern of the free surface evolution computed by the octree 3D code and the reference 2D results.

4.3. Free surface flow passing rigidly mounted offshore oil platform

To define the initial and boundary conditions for the simulation of sea waves passing a rigidly mounted obstacle, we consider simple, yet efficient, model of open sea waves introduced in [29] for the purpose of breaking waves animation. The model is based on the third order Stokes wave which is defined as follows.

One starts by defining the first order Stokes wave in terms of x- and z-components of the free surface velocity u, w and the water level η:

$$\begin{aligned}\eta(x,t) &= A\cos(kx-\omega t)\\ u(x,z,t) &= A\omega e^{-kz}\cos(kx-\omega t)\\ w(x,z,t) &= A\omega e^{-kz}\sin(kx-\omega t).\end{aligned} \tag{29}$$

Here $z = 0$ is the mean water level, $\omega = \frac{2\pi}{T}$ is the wave frequency, T is the wave period, $k = \frac{2\pi}{\lambda}$ is the wave number, λ is the wave length.

Further one introduces the third order Stokes wave by the superposition of several first order Stokes waves (29):

$$\begin{aligned}\eta(x,t) &= \frac{1}{k}(\epsilon\cos(kx-\omega t) + \frac{1}{2}\epsilon^2\cos(2kx-\omega t)\\ &\quad + \frac{3}{8}\epsilon^3\cos(3kx-\omega t))\\ u(x,z,t) &= \frac{\omega}{k}(e^{-kz}\epsilon\cos(kx-\omega t) + \frac{1}{2}e^{-2kz}\epsilon^2\cos(2kx-\omega t)\\ &\quad + \frac{3}{8}e^{-3kz}\epsilon^3\cos(3kx-\omega t))\\ w(x,z,t) &= \frac{\omega}{k}(e^{-kz}\epsilon\sin(kx-\omega t) + \frac{1}{2}e^{-2kz}\epsilon^2\sin(2kx-\omega t)\\ &\quad + \frac{3}{8}e^{-3kz}\epsilon^3\sin(3kx-\omega t)).\end{aligned} \tag{30}$$

With the help of (30) we define the water level $\eta_{2D}(x,y,t) = \eta(x,t)$ and the bulk velocity

$$\mathbf{u}_{\text{wave}}(x,y,z,t) = (u(x,z,t), 0, w(x,z,t))^T \quad \text{for} \quad z \le \eta_{2D}(x,y,t).$$

We use $\mathbf{u}_{\text{wave}}(x, y, z, 0)$ to prescribe the initial condition of our simulation.

The bulk computational domain is the 440 m × 110 m × 110 m box. Box walls are orthogonal to the coordinate axes. The sea depth is 55 m. The inlet boundary is orthogonal to x-axis and has the minimal x-coordinate. The outlet boundary is opposite to the inlet boundary. On the inlet and outlet boundaries we impose the

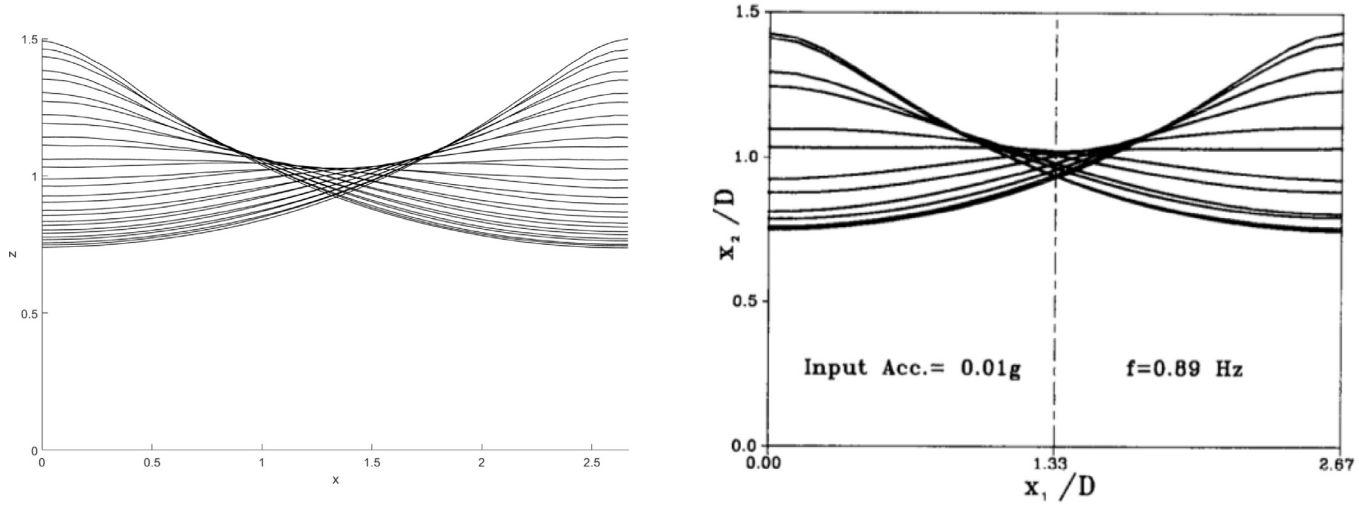


Fig. 11. Free surface mid-lines evolution : the computed results for 28 equally distributed time moments over one period (left) and for the purpose of comparison the reference results from [18] (right).

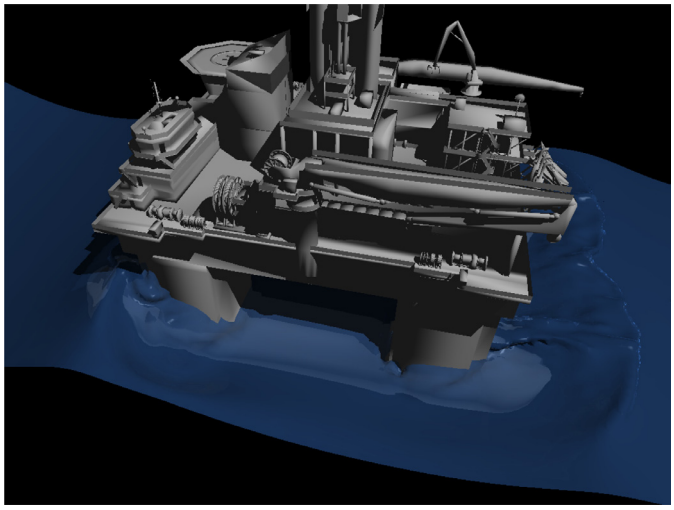


Fig. 12. A reconstruction of an operating offshore unit.

Dirichlet boundary condition using the Stokes wave, $\mathbf{u}_{\text{wave}}(\mathbf{x}, t)$, $\mathbf{x} \in \Gamma_{in} \cup \Gamma_{out}$. On other sides of the virtual box (except the top one) and the obstacle boundary we prescribe the no-penetration and free slip boundary condition.

The partially submerged object of interest is a rigidly mounted offshore oil platform. The platform shape is given by the reconstruction (with the help of a surface triangulation) of a currently operating unit, see Fig. 12.

The sea waves runup models the realistic weather scenario in the Kara sea offshore region. In particular, $A = 3$ m and $T = 4$ s correspond to a moderate storm, whereas $A = 11.5$ m and $T = 8.4$ s define the largest waves recorded in this region over the time of observations. In this paper we study the case of the largest sea waves with wave length $\lambda = 110$ m. The practical statistics of interest are the highest water levels at the platform and forces experienced by the construction.

In Fig. 13 we show the computational octree mesh, where different colors mark different type of cells: interior fluid, free surface, and solid boundary. We use the same dynamic adaptation strategy as in the previous experiment with the sloshing tank. In

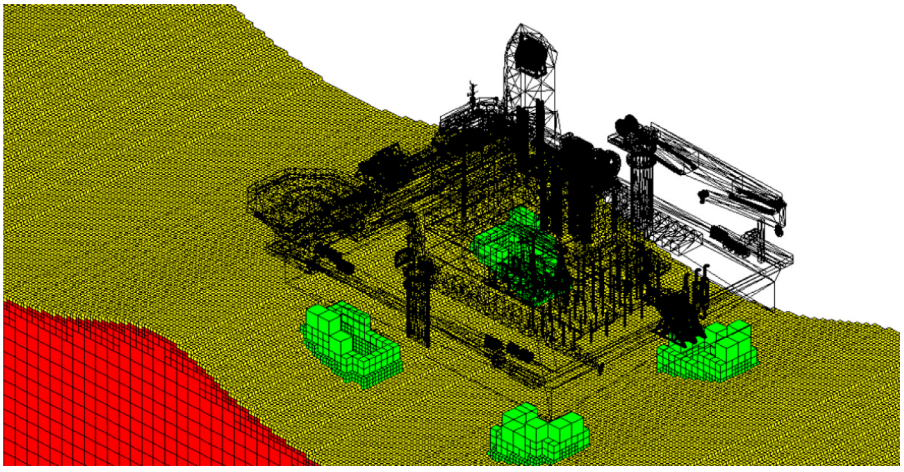


Fig. 13. Octree mesh for wave runup simulation: interior fluid cells (red), free surface cells (yellow), and immersed boundary cells (green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

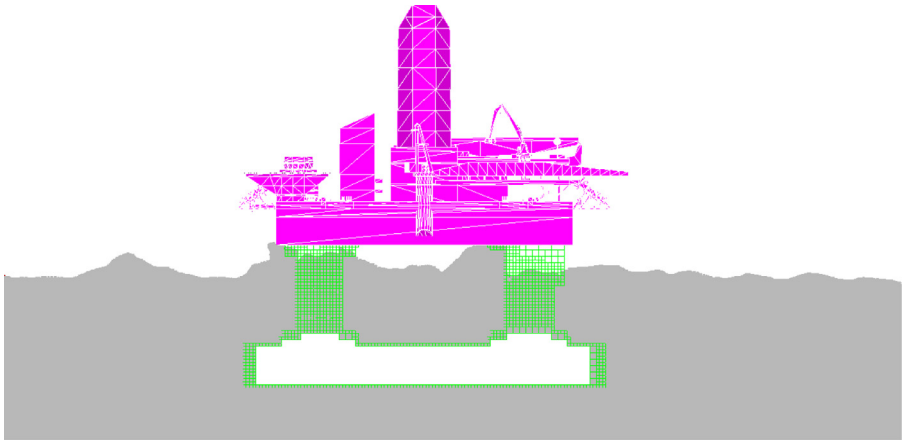


Fig. 14. Maximum observed water level, central cross-section of the computational domain.

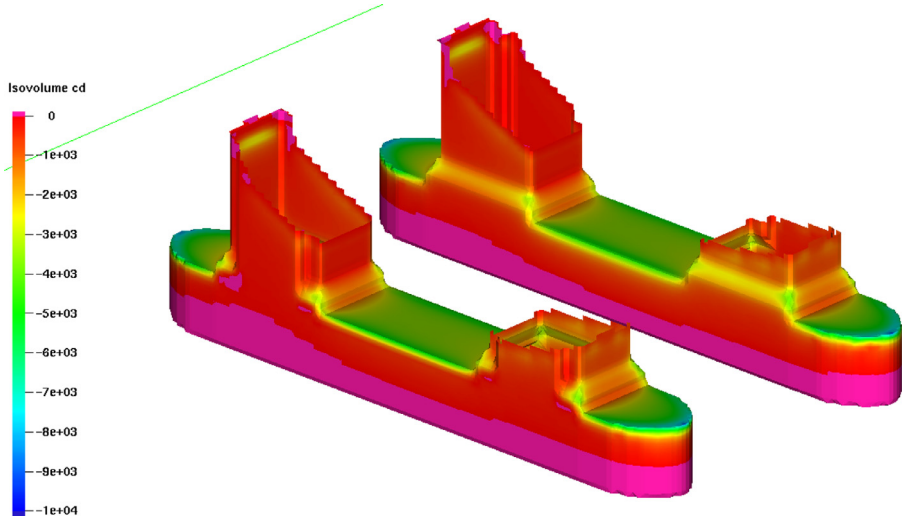


Fig. 15. A field of normal stresses projection at *x*-direction.

Fig. 14 we show the maximum water level observed in the simulation at the central cross-section of the computational domain.

In Fig. 15 we present the maximum dynamic load experienced by the oilrig piers in the x-direction. The dynamic load is computed by taking the x-projection of the normal stresses at the surface, i.e. the subintegral expression in (26).

5. Conclusions

We built a hybrid finite volume / finite difference scheme for the simulation of free-surface flows in complex geometries. The computational efficiency was achieved by using octree Cartesian meshes, while geometry was handled through the immersing of both free and static boundaries in the background mesh. The major challenges were to construct compact stencil discretizations on the gradely refined meshes with low numerical dissipation and to enforce various velocity and pressure boundary conditions on curvilinear parts of Γ. For a number of test examples, we demonstrated that the developed methods are particular suitable for the simulation of viscous free-surface flows over submerged or partially submerged objects.

Acknowledgment

The authors are grateful to N. Dianskiy and I. Kabatchenko for providing geophysical data for the Kara sea offshore.

References

[1] Bayraktar E, Mierka O, Turek S. Benchmark computations of 3d laminar flow around a cylinder with cfx, openfoam and featflow. Int J Comput Sci Eng 2012;7(3):253–66.
[2] Bazilevs Y, Takizawa K, Tezduyar TE. Computational fluid-structure interaction. Wiley-Blackwell; 2013.
[3] Behr M. Stabilized finite element methods for incompressible flows with emphasis on moving boundaries and interfaces. Doctoral dissertation, University of Minnesota; 1992.
[4] Bonito A, Guermond J-L, Lee S. Numerical simulations of bouncing jets. Int J Numer Methods Fluids 2016;80(1):53–75.
[5] Bowers AL, Rebholz LG, Takhirov A, Trenchea C. Improved accuracy in regularization models of incompressible flow via adaptive nonlinear filtering. Int J Numer Methods Fluids 2012;70:805–28.
[6] Braack M, Richter T. Solutions of 3d Navier–Stokes benchmark problems with adaptive finite elements. Comput Fluids 2006;35:372–92.
[7] Chatjigeorgiou IK, Miloh T. Free-surface hydrodynamics of a submerged prolate spheroid in finite water depth based on the method of multipole expansions. Q J Mech ApplMath 2014;67(4):525–52.
[8] Chen B-F, Nokes R. Time-independent finite difference analysis of fully non–linear and viscous fluid sloshing in a rectangular tank. J Comput Phys 2005;209(1):47–81.
[9] Chorin A. Numerical solution of the Navier–Stokes equations. Math Comput 1968;22:745–62.
[10] Droniou J. Finite volume schemes for diffusion equations: introduction to and review of modern methods. Math Models Methods Appl Sci 2014;24(08):1575–619.
[11] Dupont TF, Liu Y. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. J Comput Phys 2003;190(1):311–24.
[12] Dupont TF, Liu Y. Back and forth error compensation and correction methods for semi-Lagrangian schemes with application to level set interface computations. Math Comput 2007:647–68.

[13] Frandsen JB. Sloshing motions in excited tanks. J Comput Phys 2004;196(1):53–87.
[14] Fuster D, Agbaglah G, Josserand C, Popinet S, Zaleski S. Numerical simulation of droplets, bubbles and waves: state of the art. Fluid Dyn Res 2006;41:065001.
[15] Ghias R, Mittal R, Lund T. A non-body conformal grid method for simulation of compressible flows with complex immersed boundaries. 42nd AIAA aerospace sciences meeting and exhibit. American Institute of Aeronautics and Astronautics (AIAA); 2004.
[16] Guermond JL, Minev P, Shen J. An overview of projection methods for incompressible flows. Comput Methods Appl Mech Eng 2006;195:6011–45.
[17] Hino T. Numerical computation of a free surface flow around a submerged hydrofoil by the Euler/Navier–Stokes equations. J Soc Nav Archit Jpn 1988;164:9–17.
[18] Huerta A, Liu W. Viscous flow with large free surface motion. Comput Methods Appl Mech Eng 1988;69(3):277–324.
[19] John V. Higher order finite element methods and multigrid solvers in a benchmark problem for 3D Navier–Stokes equations. Int J Numer Methods Fluids 2002;40:775–98.
[20] Mahady LKK, Afkhami S. On the influence of initial geometry on the evolution of fluid filaments. Phys Fluids 2015;27(9):092104.
[21] Kang R, Shili S. Free surface flow generated by submerged twin-cylinders in forward motion using a fully nonlinear method. J Mar Sci Appl 2015;14(2):146–55.
[22] Kaporin IE. High quality preconditioning of a general symmetric positive definite matrix based on its $u^tu + u^tr + r^tu$-decomposition. NumerLinear Algebra Appl 1998;5(6):483–509.
[23] Konshin IN, Olshanskii MA, Vassilevski YV. ILU preconditioners for nonsymmetric saddle-point matrices with application to the incompressible Navier–Stokes equations. SIAM J Sci Comput 2015;37(5):A2171–97.
[24] Lee S, Salgado AJ. Stability analysis of pressure correction schemes for the Navier–Stokes equations with traction boundary conditions. Comput Methods Appl Mech Eng 2016;309:307–24.
[25] Leonard B. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. Comput Methods Appl Mech Eng 1979;19(1):59–98.
[26] Liu X. A new immersed boundary method for simulating free-surface flows around arbitrary objects. In: River flow 2014. Informa UK Limited; 2014. p. 141–6.
[27] Losasso F, Fedkiw R, Osher S. Spatially adaptive techniques for level set methods and incompressible flow. Comput Fluids 2006;35:995–1010.
[28] Losasso F, Gibou F, Fedkiw R. Simulating water and smoke with an octree data structure. ACM Trans Graph (TOG) 2004;23.
[29] Mihalef V, Metaxas D, Sussman M. Animation and control of breaking waves. In: Proceedings of the 2004 ACM SIGGRAPH/eurographics symposium on computer animation. Eurographics Association; 2004. p. 315–24.
[30] Mittal R, Iaccarino G. Immersed boundary methods. Annu Rev Fluid Mech 2005;37(1):239–61.
[31] Nikitin K, Olshanskii M, Terekhov K, Vassilevski Y. A splitting method for numerical simulation of free surface flows of incompressible fluids with surface tension. Comput Methods Appl Math 2015;15(1):59–78.
[32] Nikitin K, Vassilevski YV. Free surface flow modelling on dynamically refined hexahedral meshes. Rus J Numer Anal Math Model 2008;23:469–85.
[33] Noh W. CEL: a time dependent two-space-dimensional coupled Eulerian–Lagrangian code. Methods in computational physics. Alder B, Fernbach S, Rotenberg M, editors. Academic Press, New York; 1964.
[34] Oate E, Garc J. A finite element method for fluid–structure interaction with surface waves using a finite calculus formulation. Comput Methods Appl Mech Eng 2001;191(67):635–60. Minisymposium on Methods for Flow Simulation and Modeling
[35] Olshanskii MA, Terekhov KM, Vassilevski YV. An octree-based solver for the incompressible Navier–Stokes equations with enhanced stability and low dissipation. Comput Fluids 2013;84:231–46.
[36] Osher S, Fedkiw R. Level set methods and dynamic implicit surfaces. Springer-Verlag; 2002.
[37] Peskin CS. Flow patterns around heart valves: a numerical method. J Comput Phys 1972;10(2):252–71.
[38] Pironneau O. On the transport-diffusion algorithm and its applications to the Navier–Stokes equations. Numerische Mathematik 1982;28:309–32.
[39] Popinet S. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. J Comput Phys 2003;190(2):572–600.
[40] Popinet S. An accurate adaptive solver for surface-tension-driven interfacial flows. J Comput Phys 2009;228:5838–66.
[41] Prohl A. Projection and quasi-compressibility methods for solving the incompressible Navier–Stokes equations. B.G. Teubner (Stuttgart); 1997.
[42] Schäfer M, Turek S. Benchmark computations of laminar flow around a cylinder. In: Flow Simulation with High-Performance Computers II, Volume 48 of the series Notes on Numerical Fluid Mechanics (NNFM), 52; 1996. p. 547–66.
[43] Sleijpen GL, Fokkema DR. Bicgstab (l) for linear equations involving unsymmetric matrices with complex spectrum. Electron Trans Numer Anal 1993;1(11):2000.
[44] Sochnikov V, Efrima S. Level set calculations of the evolution of boundaries on a dynamically adaptive grid. Int J Numer Methods Eng 2003;56:1913–29.
[45] Terekhov K. Solution of filtration and hydrodynamic problems on adaptive octree meshes. Institute of Numerical Mathmatics RAS, Moscow (in Russian); 2013. Phd thesis. http://dodo.inm.ras.ru/~terekhov/thesis.pdf.
[46] Terekhov KM, Nikitin KD, Olshanskii MA, Vassilevski YV. A semi-Lagrangian method on dynamically adapted octree meshes. Russ J Numer AnalMath Model 2015;30(6):363–80.
[47] Tyvand PA, Miloh T. Free-surface flow due to impulsive motion of a submerged circular cylinder. J Fluid Mech 1995;286(-1):67.
[48] Virella JC, Prato CA, Godoy LA. Linear and nonlinear 2d finite element analysis of sloshing modes and pressures in rectangular tanks subject to horizontal harmonic motions. J Sound Vib 2008;312(3):442–60.
[49] Xiu D, Karniadakis GE. A semi-Lagrangian high-order method for Navier–Stokes equations. J Comput Phys 2001;172(2):658–84.
[50] Zhang Y, Zou Q, Greaves D, Reeve D, Hunt-Raby A, Graham D, et al. A level set immersed boundary method for water entry and exit. Commun Comput Phys 2010;8(2):265–88.
[51] Zingan V, Guermond J-L, Morel J, Popov B. Implementation of the entropy viscosity method with the discontinuous Galerkin method. Comput Methods Appl Mech Eng 2013;253:479–90.