A multigrid ghost-point level-set method for incompressible Navier-Stokes equations on moving domains with curved boundaries

Armando Coco

Abstract

In this paper we present a numerical approach to solve the Navier-Stokes equations on moving domains with second-order accuracy. The space discretization is based on the ghost-point method, which falls under the category of unfitted boundary methods, since the mesh does not adapt to the moving boundary. The equations are advanced in time by using Crank-Nicholson. The momentum and continuity equations are solved simultaneously for the velocity and the pressure by adopting a proper multigrid approach. To avoid the checkerboard instability for the pressure, a staggered grid is adopted, where velocities are defined at the sides of the cell and the pressure is defined at the centre. The lack of uniqueness for the pressure is circumvented by the inclusion of an additional scalar unknown, representing the average divergence of the velocity, and an additional equation to set the average pressure to zero. Several tests are performed to simulate the motion of an incompressible fluid around a moving object, as well as the lid-driven cavity tests around steady objects. The object is implicitly defined by a level-set approach, that allows a natural computation of geometrical properties such as distance from the boundary, normal directions and curvature. Different shapes are tested: circle, ellipse and flower. Numerical results show the second order accuracy for the velocity and the divergence (that decays to zero with second order) and the efficiency of the multigrid, that is comparable with the tests available in literature for rectangular domains without objects, showing that the presence of a complex-shaped object does not degrade the performance.

1 Introduction

Navier-Stokes equations are a system of partial differential equations (PDEs) describing the motion of an incompressible viscous fluid subject to external forces. In the non-dimensional form, they read

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \frac{1}{\text{Re}} \nabla^2 \mathbf{u} + \mathbf{f}$$
$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

where $\mathbf{u}$ is the fluid velocity, p is the pressure, Re is the Reynolds number and $\mathbf{f}$ the external force. The first equation is obtained from the momentum equation, while the second one is the continuity equation and represents the incompressibility condition. Navier-Stokes equations have been central to innumerable scientific and engineering applications for several decades, as the necessity to model the motion of incompressible

fluids has a tremendous impact on the real-life world. Several numerical approaches have been developed so far to provide efficient and accurate methods in different contexts, both for time and space discretization. An overview on traditional methods can be found in [43].

One of the first approaches for time discretization was the *projection method*, proposed by Chorin in [14]. It consists of a fractional step, where the velocity $\mathbf{u}$ is advanced in time by discretising the momentum equation and ignoring the pressure term, obtaining an intermediate velocity $\mathbf{u}^*$. This velocity is not divergence-free, and therefore is corrected by the pressure term, that is determined by enforcing the divergence-free condition. Projection methods have been very popular for Navier-Stokes equations and are widely adopted nowadays, as they are computationally efficient and easy to implement (they require the solution of a sequence of elliptic equations for the intermediate velocity and for the pressure). However, the definition of boundary conditions for $\mathbf{u}^*$ and p is not obvious and it is challenging to achieve an accuracy higher than first order, even if the internal equations are discretized with higher order. There are existing approaches that suggest appropriate boundary conditions to improve the accuracy, as described in [11]. However, these boundary conditions are fully coupled each other and the resulting elliptic equations for $\mathbf{u}^*$ and for p cannot be solved in sequence, as the boundary condition on p depends on $\mathbf{u}^*$.

Another approach consists of discretising both PDEs of Eq. (1) and solve them simultaneously in the unknowns $(\mathbf{u}, p)$. This is the *monolithic* approach and has the advantage that no boundary conditions are required for the pressure, as described, for example, in [61, Sect. 8.1]. These methods can usually achieve higher order accuracy than projection methods, but they have the disadvantage that extra computational effort is needed to solve both PDEs simultaneously. This aspect can be alleviated if efficient approaches are adopted such as the monolithic solver for fluid/solid interactions proposed in [35], where the authors achieve second order accuracy by maintaining a symmetric positive definite discretization.

Projection and monolithic methods refer only to time discretization and then spatial discretization methods must be addressed regardless of the choice between the methods described above. Standard problems on rectangular domains are usually discretized in space by using a finite volume approach on a staggered grid (MAC grid, see for example [59]), where the velocities are defined at the edges of the cell and the pressure is defined at the centre. The MAC grid is used for example to avoid the checkerboard instability for the pressure term observed in non-staggered grids due to the fact that p appears in the equations only in the form of ∇p. The extension of MAC grids to non-rectangular domains (curved boundaries) is not straightforward, since a particular treatment is needed close to the boundary. Moreover, there is an increasing scientific interest to Navier-Stokes equations for complex-shaped (possibly moving) domains, for example to model the motion of an incompressible fluids around a deforming object, that may be central to more sophisticated applications such as fluid-structure interactions. Numerical methods for steady complex-shaped objects are mainly divided between fitted and unfitted boundary methods. In fitted boundary methods, the grid, either structured or unstructured (such as Finite Element Methods (FEM)), is adapted to the curved boundary [58, 38, 36, 47, 46, 25, 24]. Although they provide an extremely accurate representation of the domain, the generation of the mesh may become cumbersome and significantly expensive from a computational point of view, especially in moving domain problems where a new mesh generation is needed at each time step. In unfitted boundary methods the moving domain is instead embedded in a fixed grid. The methods of this class are based either on FEM, such as XFEM [40, 55] or GFEM [15], on meshless

methods [62], or Cartesian meshes, such as cut cell methods [37].

An alternative approach of finite–difference methods for curved boundaries is represented by the Ghost–Point method. A pioneering work of this class of methods is the Immersed Boundary Method [54] to model blood flows in the heart, where Peskin presented a first-order accurate method, later extended to higher order by the Immersed Interface Methods proposed by LeVeque and Li in [44]. More recent finite-difference methods to discretize PDEs on complex domains are the Ghost-Fluid Methods proposed in [30, 33, 34, 50], where the grid points external to the domain are called ghost points and a fictitious (ghost) value of the numerical solution is formally extrapolated onto these grid points in order to maintain a stable algorithm and achieve the desired accuracy order. The main feature of these methods is that the internal equations are firstly formally solved for the ghost values, resulting in a final linear system with eliminated boundary conditions. This class of methods may fail to obtain high accuracy for Neumann boundary condition, and anyway it remains first order accurate in the post-processing reconstruction of the gradient of the solution.

A new finite-difference method with an improved accuracy was proposed by the author and Russo in [20, 22, 19] for elliptic equations and extended to thermo-poroelastic Cauchy-Navier equations for elastic deformation [17], Euler equations for gas dynamics [13, 21], electron transport in semiconductors [48, 49], fluid flow in porous media for water-rock interaction [18, 16]. In this approach, the ghost point values are eventually coupled each other, resulting in a bigger linear system with non-eliminated boundary conditions, solved by a multigrid approach suitably designed for ghost-point methods. The domain is implicitly described by a level-set function.

Other numerical methods to solve sharp-edge boundary problems are the finite volume methods, the non-symmetric positive definite finite element method, the matched interface and boundary (MIB) method [63], the arbitrary Lagrangian Eulerian method (ALE) [31, 27], the penalization methods [42, 5, 12], and the class of Immersed Finite Volume Methods (IFVM) [29]. Other recent advances have been obtained in [41] to achieve higher accuracy in the presence of Neumann boundary conditions, in [9] to solve elliptic interface problems with discontinuous coefficients, in [28] to solve sharp interface conditions and in [8, 6, 53] to obtain higher accuracy in the gradient of the solution.

Time and space discretizations of Navier-Stokes equations lead to a linear system that must be solved efficiently. Multigrid method is one of the most efficient iterative approaches to solve the linear systems arising from the discretization of a class of PDEs. Initially designed for elliptic problems, multigrid methods have been widely adopted to solve countless problems [61, 39, 10]. Several approaches have been proposed for boundaries aligned with grid lines. As example, we mention [4, 26, 3, 45, 1]. These approaches fall under the class of geometric multigrid, since the operators are defined from the geometrical background of the problem. An extension of geometric multigrid is represented by the class of algebraic multigrid methods [60], where the operators are defined merely from the information gathered from the linear systems, and therefore it is more practical for complex problems such as curved boundaries. However, the inclusion of geometrical information to the definition of the operators usually results in a more efficient solver, as highlighted for example in [2], where a comparison between geometric and algebraic multigrid methods for curved boundaries is performed.

In this paper, we provide a finite-difference method to solve the Navier-Stokes equations on moving domains following a monolithic approach for the time discretization and a finite-difference ghost-point method for the space discretization. The singularity of the problem (due to the fact that p is not uniquely defined,

since it appears only in terms of ∇p) is circumvented by an additional equation for the pressure that requires that its average is zero. This additional equation is balanced with an additional unknown ξ that measures the average divergence of the velocity (that is therefore not necessarily zero, as in traditional methods, but decays with the same order of accuracy of the method). The linear system arising from the discretization is solved by a geometric multigrid approach, where the relaxation, restriction and interpolation operators are modified close to the boundary in order to maintain the same efficiency that standard multigrid techniques achieve away from the boundary.

The paper is structured as follows. In Sect. 2 and 3 we describe the time and space discretizations, respectively. The multigrid technique is detailed in Sect. 4, while numerical tests are provided in Sect. 5 and conclusions in Sect.6.

2 Discretization in time

The Navier-Stokes equations (1) are discretized in time using Crank-Nicholson and solved for $\mathbf{u}^{n+1}$ in the domain $\Omega^{(n+1)}$

$$\frac{\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)}}{\Delta t} + (\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)} + \nabla p^{(n+1/2)} = \frac{1}{2\,\mathrm{Re}} \left(\nabla^2 \mathbf{u}^{(n)} + \nabla^2 \mathbf{u}^{(n+1)} \right) + \mathbf{f}^{(n+1/2)} \text{ in } \Omega^{(n+1)}$$
$$\nabla \cdot \mathbf{u}^{(n+1)} = 0 \text{ in } \Omega^{(n+1)} \tag{2}$$
$$\mathbf{u}^{(n+1)}\Big|_{\partial\Omega^{(n+1)}} = \mathbf{u}_b$$

Taking the implicit terms on the left hand side and the explicit terms on the right hand side and multiplying by Δt, Eq. (2) reads:

$$\mathbf{u}^{(n+1)} - \frac{\Delta t}{2\,\mathrm{Re}} \nabla^2 \mathbf{u}^{(n+1)} + \Delta t \, \nabla p^{(n+1/2)} = \tilde{\mathbf{f}} - \Delta t \, (\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)}$$
$$\nabla \cdot \mathbf{u}^{(n+1)} = 0 \tag{3}$$
$$\mathbf{u}^{(n+1)}\Big|_{\partial\Omega^{(n+1)}} = \mathbf{u}_b$$

where $\tilde{\mathbf{f}} = \mathbf{u}^{(n)} + \dfrac{\Delta t}{2\,\mathrm{Re}} \nabla^2 \mathbf{u}^{(n)} + \Delta t \, \mathbf{f}^{(n+1/2)}$. The first question to address is the treatment of the convective term $(\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)}$. A natural approach is to approximate this term fully explicitly $(\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)} \approx \mathbf{u}^{(n)} \cdot \nabla \mathbf{u}^{(n)}$ or with an implicit-explicit technique $(\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)} \approx \mathbf{u}^{(n)} \cdot \nabla \mathbf{u}^{(n+1)}$, which however are only first order accurate. A higher order fully explicit approximation can be obtained by extrapolating the convective term from previous time steps at the desired accuracy. For example, for a second order approximation, the extrapolation is

$$(\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)} \approx (\mathbf{u} \cdot \nabla \mathbf{u})^{(n)} + \frac{1}{2} (\mathbf{u} \cdot \nabla \mathbf{u})^{(n-1)} - \frac{1}{2} (\mathbf{u} \cdot \nabla \mathbf{u})^{(n-2)} . \tag{4}$$

4

This approximation leads to a multistep method using the solution at the three previous time steps. For this reason, the solution at the first two time steps $\mathbf{u}^{(i)}$ with $i = 1, 2$ must be computed with an alternative method (observe that $\mathbf{u}^{(0)}$ is set up from the initial condition). Another second-order approximation is provided by the mid-point discretization

$$(\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)} \approx \frac{1}{2} \left((\mathbf{u} \cdot \nabla \mathbf{u})^{(n)} + (\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1)} \right).$$

With this approximation, Eqs. (3) becomes a nonlinear equation due to the presence of the nonlinear convective term $\mathbf{u}^{(n+1)} \cdot \nabla \mathbf{u}^{(n+1)}$. The nonlinearity can be circumvented by using, for example, a Newton-Raphson approach, or alternatively an iterative method (with an iterative parameter k)

$$\mathbf{u}^{(n+1,k+1)} - \frac{\Delta t}{2\,\mathrm{Re}} \nabla^2 \mathbf{u}^{(n+1,k+1)} + \Delta t \, \nabla p^{(n+1/2,k+1)} = \tilde{\mathbf{f}} - \Delta t \, (\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2,k)}$$

$$\nabla \cdot \mathbf{u}^{(n+1,k+1)} = 0 \qquad (5)$$

$$\left. \mathbf{u}^{(n+1,k+1)} \right|_{\partial \Omega^{(n+1)}} = \mathbf{u}_b$$

with

$$(\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2,k)} = \frac{(\mathbf{u} \cdot \nabla \mathbf{u})^{(n)} + (\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1,k)}}{2}.$$

The iteration algorithm is performed until $\mathbf{u}^{(n+1,k+1)}$ satisfies Eqs. (3) within a certain tolerance ε (say $\varepsilon = 10^{-6}$), i.e.

$$\left\| \tilde{\mathbf{f}} - \Delta t \, (\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2,k+1)} - \mathbf{u}^{(n+1,k+1)} + \frac{\Delta t}{2\,\mathrm{Re}} \nabla^2 \mathbf{u}^{(n+1,k+1)} - \Delta t \, \nabla p^{(n+1/2,k+1)} \right\| < \max \left\{ \varepsilon, \varepsilon \left\| \tilde{\mathbf{f}} \right\| \right\}.$$

Then, $\mathbf{u}^{(n+1)} := \mathbf{u}^{(n+1,k+1)}$ and the scheme is moved to the next time step.

Other approaches can be adopted, for example approximation techniques based on the Godunov method [7], or Runge-Kutta time discretization [51]. A detailed treatment is beyond the scope of this paper, as we mainly focus on the treatment of curved boundaries and the method proposed for that purpose do not rely on the approach adopted to approximate the nonlinear term.

In this paper, we adopt a hybrid approach: we use the iteration technique (5) for the first two time steps and the extrapolation (4) from the third time step.

Either if we are using the iteration technique (5) or the extrapolation approach (4), the problem that we have to solve at each iteration can be formulated as follows: find $(\mathbf{u}, p)$ such that

$$\mathbf{u} - \alpha \nabla^2 \mathbf{u} + \nabla p = \tilde{\mathbf{f}} \quad \text{in } \Omega$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \qquad (6)$$

$$\mathbf{u} = \mathbf{u}_b \quad \text{on } \partial \Omega$$

where $\Omega = \Omega^{(n+1)}$, $\alpha = \Delta t / (2\,\mathrm{Re})$, and p and $\tilde{\mathbf{f}}$ are redefined as $p := \Delta t\, p$ and $\tilde{\mathbf{f}} := \tilde{\mathbf{f}} - \Delta t \, (\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)}$.

The approach suggested by the projection or gauge methods consists of solving an intermediate step

$$\mathbf{u}^* - \alpha \nabla^2 \mathbf{u}^* = \tilde{\mathbf{f}} \tag{7}$$

and then projecting the intermediate solution $\mathbf{u}^*$ onto the space of divergence-free functions. In detail, $\mathbf{u}^*$ is split as $\mathbf{u}^* = \mathbf{u} + \nabla \chi$, where $\mathbf{u}$ is divergence-free. The auxiliary function χ is determined by taking the divergence of both sides, leading to an elliptic problem $\nabla^2 \chi = \nabla \cdot \mathbf{u}^*$. Finally, the solution $\mathbf{u}$ is recovered from the splitting operation: $\mathbf{u} = \mathbf{u}^* - \nabla \chi$. Substituting this latter expression into Eq. (6), we obtain the expression to compute the pressure term p from the auxiliary function: $p = \chi - \alpha \nabla^2 \chi$.

One of the main aspects to consider in the projection method is the choice of boundary conditions for $\mathbf{u}^*$ in Eq. (7) and χ in the elliptic equation, in order to maintain the desired accuracy. The naïve choice $\mathbf{u}^*|_{\partial \Omega} = \mathbf{u}_b$ does degrade the overall method to first order accuracy, regardless of space/time discretization accuracy, because it is not guaranteed that $\nabla \chi|_{\partial \Omega} = 0$. By projecting $\mathbf{u}^* = \mathbf{u} + \nabla \chi$ to the boundary $\partial \Omega$ along the normal and tangential directions $\mathbf{n}$ and τ, respectively, we obtain the following boundary conditions for $\mathbf{u}^*$:

$$\mathbf{u}^* \cdot \mathbf{n} = \mathbf{u}_b \cdot \mathbf{n} + \frac{\partial \chi}{\partial \mathbf{n}}, \qquad \mathbf{u}^* \cdot \tau = \mathbf{u}_b \cdot \tau + \frac{\partial \chi}{\partial \tau}. \tag{8}$$

Although we can set an arbitrary value for $\dfrac{\partial \chi}{\partial \mathbf{n}}$ (because this value will then be prescribed as boundary condition in the elliptic equation for χ), the value $\dfrac{\partial \chi}{\partial \tau}$ is not known in (8) as χ is computed later. One possibility is to use an approximation for $\dfrac{\partial \chi}{\partial \tau}$ in (8) by extrapolating χ from previous time steps: $\chi^{(n+1)} = 2\chi^{(n)} - \chi^{(n-1)}$. A detailed discussion on projection and gauge methods can be found in [11].

In this paper we adopt a *monolithic* approach, i.e. we solve the simultaneous equations (6) as a single block of equations for both unknowns $\mathbf{u}$ and p. The pressure term p of Eq. (6) is not uniquely defined, as it appears as ∇p, leading to a singular linear system when discretising the equations in space. The issue is circumvented by augmenting the problem with an additional scalar unknown $\xi \in \mathbb{R}$ and an additional equation for p. The problem consists then of finding $(\mathbf{u}, p, \xi)$ such that

$$\begin{aligned}
\mathbf{u} - \alpha \nabla^2 \mathbf{u} + \nabla p &= \tilde{\mathbf{f}} \quad &&\text{in } \Omega \\
\nabla \cdot \mathbf{u} - \xi &= 0 \quad &&\text{in } \Omega \\
\int_\Omega p \, d\Omega &= 0 \\
\mathbf{u} &= \mathbf{u}_b \quad &&\text{on } \partial \Omega
\end{aligned} \tag{9}$$

3 Discretization in space

In this section we describe the space discretization of the equations, with particular attention to the treatment of the curved boundary using a fixed grid and a ghost-point approach. We describe the method in 2D for

simplicity, although the approach can be straightforwardly extended to 3D as it does not rely on two-dimensional aspects.

An arbitrary domain Ω can be represented in different ways. For example, it can be represented explicitly by the parametric equations that describe the boundary as a closed curve $\partial\Omega = \{(x(\gamma), y(\gamma)) \colon \gamma \in (\gamma_m, \gamma_M)\}$, or implicitly by a level-set function $\phi(x, y)$ that is negative inside the domain, positive outside, and vanished on the boundary. In this paper we adopt a level-set approach, which is more convenient to recover additional geometric information that are required from the numerical method proposed in this paper as we will see later, such as normal direction, curvature, or to easily detect whether a point is inside or outside the domain or to approximate its distance to the boundary. In addition, the level-set function can be used in the framework of level-set methods [52], that is widely adopted in the context of PDEs in moving domains as it allows the boundary of a domain to move according to a velocity field $\mathbf{u}$. Therefore, although the numerical tests provided in this paper model fluid dynamics around moving objects whose motion is assigned *a priori*, the numerical method proposed can be extended to the case of moving objects whose motion is implicitly generated by the fluid dynamics, as for example in fluid-structure interaction, bubble dynamics, wind turbines.

3.1 Level-set function

In this paper we simulate the motion of an incompressible viscous fluid around a (possibly moving) object $\mathcal{R}(t)$. The fluid domain is then $\Omega(t) = \mathcal{D}\backslash\mathcal{R}(t)$, where $\mathcal{D} = (-1, 1)^2$ is the computational domain. We simplify the notation with $\mathcal{R}$ and Ω instead of $\mathcal{R}(t)$ and $\Omega(t)$ when the object is steady or when it is not necessary to specify the domains as a function of time.

The object $\mathcal{R}$ is implicitly described by a level-set function $\phi \colon D \to \mathbb{R}$ such that:

$$\mathcal{R} = \left\{ \mathbf{x} \in \mathbb{R}^2 \colon \phi(\mathbf{x}) > 0 \right\}, \quad \Gamma = \partial\mathcal{R} = \left\{ \mathbf{x} \in \mathbb{R}^2 \colon \phi(\mathbf{x}) = 0 \right\}.$$

We observe that we have chosen ϕ to be positive inside the object so that it is negative inside the fluid domain Ω. From the level-set function it is possible to infer the geometric properties of the boundary Γ. For example, the outward unit normal vector $\mathbf{n}$ and the curvature κ can be computed as:

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}, \quad \kappa = \nabla \cdot \mathbf{n}.$$

Given an object $\mathcal{R}$, there are infinite level-set functions that can represent that object. A particular level-set function is the signed distance function $\phi_d(\mathbf{x}) = \pm\text{dist}(\mathbf{x}, \Gamma)$, where the sign is positive or negative depending on whether $\mathbf{x}$ is inside or outside $\mathcal{R}$, respectively. The signed distance function is convenient to avoid unstable issues caused by deep ($|\nabla\phi(\mathbf{x})| \gg 0$) or shallow ($|\nabla\phi(\mathbf{x})| \ll 0$) gradients that may develop when the level-set evolves over time according to a velocity field (the signed distance function does not present these issues since $|\nabla\phi_d(\mathbf{x})| = 1$). At each time step, the signed distance function ϕ_d can be recovered from a level-set function ϕ by the reinitialization process, that consists of finding the steady-state solution (in a fictitious

7

time μ) of the following equation

$$\frac{\partial \phi_d}{\partial \mu} = \text{sign}(\phi)\left(1 - |\nabla \phi_d|\right), \quad \phi_d = \phi \text{ when } \mu = 0.$$

In this paper, we focus for simplicity on predetermined rigid motion of the object $\mathcal{R}$, and therefore we do not need to perform the reinitialization process. Therefore, we design the numerical methods for a generic level-set function ϕ, that is not necessarily a signed-distance function. The extension of the proposed methods to deforming objects would be straightforward.

3.2   Internal grid points

The computational domain $\mathcal{D}$ is discretized through a uniform Cartesian mesh with spatial step $h = \Delta x = \Delta y$. The pressure p and the velocity components $\mathbf{u} = (u, v)$ are defined on a staggered grid (see Fig. 1): p is defined at the centre of each cell (*central grid*), u is defined at the mid points of the vertical side cells (*horizontally staggered grid*) and v at the mid points of the horizontal side cells (*vertically staggered grid*). The first two governing equations of (6), expanded component by component, are:

$$\begin{aligned}
u - \alpha \nabla^2 u + \frac{\partial p}{\partial x} &= \tilde{f}^u \quad \text{in } \Omega \\
v - \alpha \nabla^2 v + \frac{\partial p}{\partial y} &= \tilde{f}^v \quad \text{in } \Omega \\
\nabla \cdot \mathbf{u} &= \xi \quad \text{in } \Omega.
\end{aligned} \tag{10}$$

We discretize Eqs. (10) by central finite differences: $\dfrac{\partial p}{\partial x}, \dfrac{\partial u}{\partial x}, \dfrac{\partial u}{\partial y}$ and $\nabla^2 u$ are defined on the horizontally staggered grid, $\dfrac{\partial p}{\partial y}, \dfrac{\partial v}{\partial x}, \dfrac{\partial v}{\partial y}$ and $\nabla^2 v$ are defined on the vertically staggered grid, and $\nabla \cdot \mathbf{u}$ on the central grid.

$$\left.\frac{\partial p}{\partial x}\right|_{i+1/2,j} = \frac{p_{i+1,j} - p_{i,j}}{h}, \quad \left.\frac{\partial p}{\partial y}\right|_{i,j+1/2} = \frac{p_{i,j+1} - p_{i,j}}{h}$$

$$\left.\frac{\partial u}{\partial x}\right|_{i+1/2,j} = \frac{u_{i+3/2,j} - u_{i-1/2,j}}{2h}, \quad \left.\frac{\partial u}{\partial y}\right|_{i+1/2,j} = \frac{u_{i+1/2,j+1} - u_{i+1/2,j-1}}{2h},$$

$$\left.\frac{\partial v}{\partial x}\right|_{i,j+1/2} = \frac{v_{i+1,j+1/2} - v_{i-1,j+1/2}}{2h}, \quad \left.\frac{\partial v}{\partial y}\right|_{i,j+1/2} = \frac{v_{i,j+3/2} - v_{i,j-1/2}}{2h},$$

$$\left.\nabla^2 u\right|_{i+1/2,j} = \frac{u_{i+3/2,j} + u_{i-1/2,j} + u_{i+1/2,j+1} + u_{i+1/2,j-1} - 4u_{i+1/2,j}}{h^2},$$

$$\left.\nabla^2 v\right|_{i,j+1/2} = \frac{v_{i+1,j+1/2} + v_{i-1,j+1/2} + v_{i,j+3/2} + v_{i,j-1/2} - 4v_{i,j+1/2}}{h^2},$$

8

$$\nabla \cdot \mathbf{u}\big|_{i,j} = \frac{u_{i+1/2,j} - u_{i-1/2,j} + v_{i,j+1/2} - v_{i,j-1/2}}{h}.$$

The discretization of (10) occurs only on grid points inside of Ω (*internal grid points*). Each grid is treated separately for the identification of internal grid points. In detail, the first equation of (10) is discretized on internal grid points of the *horizontally staggered grid*, the second equation of (10) is discretized on internal grid points of the *vertically staggered grid*, and the third equation of (10) is discretized on internal grid points of the *central grid*.

The third equation of (9) is discretized by standard mid-point rule, leading to the linear equation

$$\sum_{(x_i,y_j) \text{ internal grid point}} p_{i,j} = 0. \tag{11}$$
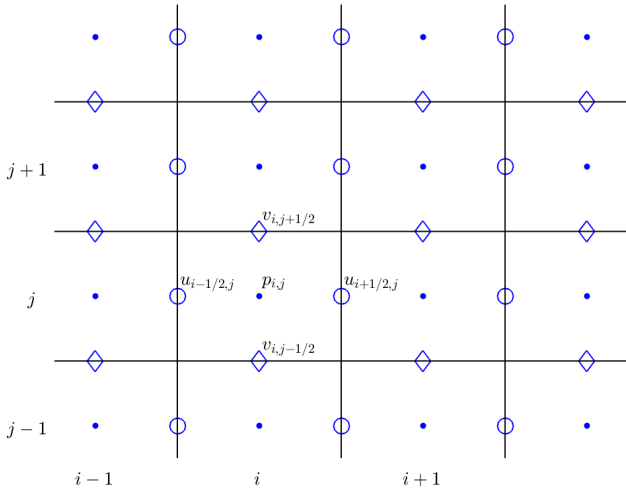

Fig. 1: Representation of the staggered grid. p is defined at the centre of each cell (*central grid* - small dots), u is defined at the mid points of vertical side cells (*horizontally staggered grid* - empty circles) and v at the mid points of horizontal side cells (*vertically staggered grid* - diamonds).
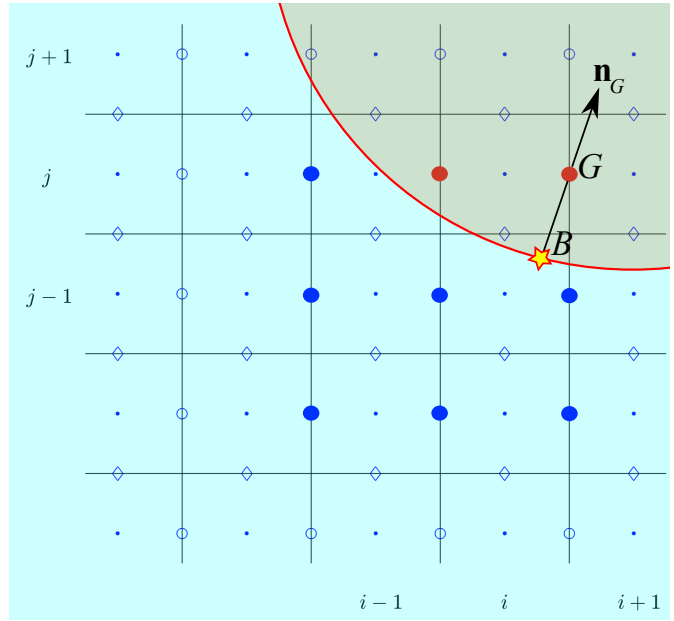

Fig. 2: Ghost-point extrapolation. The ghost value $u(G)$ is obtained by enforcing the boundary condition $\tilde{u}(B) = u_b(B)$, where $\tilde{u}(B)$ is the biquadratic interpolation of u on the nine-point stencil represented by the full circles (blue circles inside the fluid and red circles inside the object).

3.3   Ghost points

For internal grid points that are close to the boundary of Ω it may happen that some of the grid points needed in the discretization stencil may lie outside of the domain. For example, referring to Fig. 2, the

discretization of $\nabla^2 u$ on the internal grid point $(x_{i+1/2}, y_{j-1})$ requires the values of u on the external grid point $(x_{i+1/2}, y_j)$, and the discretization of $\nabla^2 v$ on the internal grid point $(x_{i-1}, y_{j-1/2})$ requires the values of v on external grid points $(x_{i-1}, y_{j+1/2})$ and $(x_i, y_{j-1/2})$. These external grid points are called *ghost points* and a fictitious value (ghost value) of u, v and p on these points is required by the discretization of the governing equation on internal grid points. This ghost value is assigned by enforcing the boundary conditions up to the desired accuracy, following a similar approach adopted in [20, 22] for elliptic equation and described below. Let $G = (x_{i+1/2}, y_j)$ be a ghost point of the horizontally staggered grid (the approach is similar for any grid), as depicted in Fig. 2. Firstly, we compute the boundary point $B = (x_B, y_B)$ from G by making use of the level-set function as follows. Starting from G, we move along the normal direction $\mathbf{n} = \nabla\phi/|\nabla\phi|$ by steps of $h/2$, until we obtain an internal point H. Let K be the last external point obtained from this process. Then, we perform a bisection method to solve the nonlinear equation $\phi(B) = 0$ in the unknown B along the segment $\overline{HK}$ up to a desired accuracy ϵ (we choose $\epsilon = 0.01h$). For the evaluation of ϕ and $\nabla\phi$ we use a bilinear interpolation on (2×2)-point stencils containing the point in which the evaluation is required.

The algorithm is summarised in Alg. 1.

Algorithm 1 Bisection method to find the projection boundary point B from a ghost point G.

$H := G$
$\phi_H := \phi(G)$
while $\phi_H > 0$ **do**
$\quad K := H$
$\quad$ let $\tilde{\phi}$ be a bilinear interpolation of ϕ on a (2×2)-point stencil containing H
$\quad \mathbf{n}_H := \left(\nabla\tilde{\phi}/|\nabla\tilde{\phi}|\right)\big|_H$
$\quad H := H - 0.5\,h\,\mathbf{n}_H$
$\quad \phi_H := \tilde{\phi}(H),$
end while
for iter $= 1$ to $\lceil \log_2(\text{dist}(H, K)/\epsilon) \rceil$ **do**
$\quad B = \dfrac{H + K}{2}$
$\quad \phi_B := \tilde{\phi}(B)$, where $\tilde{\phi}$ is a bilinear interpolation of ϕ on a (2×2)-point stencil containing B
$\quad$ **if** $\phi_B \cdot \phi_H < 0$ **then**
$\quad\quad K := B$
$\quad\quad \phi_K := \phi_B$
$\quad$ **else**
$\quad\quad H := B$
$\quad\quad \phi_H := \phi_B$
$\quad$ **end if**
end for

Once the boundary point B is computed, the ghost value $u_{i+1/2,j}$ is implicitly defined by enforcing the

boundary condition $u(B) = u_b(B)$, where $u(B)$ is approximated by interpolating the grid values of u on a suitable stencil. Existing approaches require that the stencil is made only by internal grid points, as the numerical solution is not defined on external points [23]. However, we follow the approach of the author and Russo in [20, 22], where the stencil may contain external grid points, leading to a more flexible scheme. In detail, we identify the nine-point stencil in the opposite direction to the normal direction $\mathbf{n}_G$ (upwind direction):

$$\mathrm{ST}_9 = \left\{ (x_{i+1/2}, y_j) - h\left(s_x\, k_x, s_y\, k_y\right), (k_x, k_y) \in \{0,1,2\}^2 \right\},$$

where $s_x = \mathrm{SIGN}(n_x)$ and $s_y = \mathrm{SIGN}(n_y)$, with $\mathrm{SIGN}(z) = 1$ if $z \geq 0$ and $\mathrm{SIGN}(z) = -1$ if $z < 0$ (we take conventionally $\mathrm{SIGN}(0) = 1$). The value $u_{i+1/2,j}$ is implicitly defined by the equation $\tilde{u}(B) = u_b(B)$, where $\tilde{u}(B)$ is the biquadratic interpolation of u on the stencil ST_9 evaluated at the boundary point B. In detail, let $\vartheta_x = s_x \left(x_B - x_{i+1/2}\right)/h$ and $\vartheta_y = s_y \left(y_B - y_j\right)/h$. Then, the equation for $\tilde{u}(B) = u_b(B)$ can be represented using a convenient matrix notation:

$$\odot \left(\begin{bmatrix} \dfrac{(1-\vartheta_y)(2-\vartheta_y)}{2} \\[2mm] \vartheta_y(2-\vartheta_y) \\[2mm] \dfrac{\vartheta_y(\vartheta_y-1)}{2} \end{bmatrix} \times \begin{bmatrix} \dfrac{(1-\vartheta_x)(2-\vartheta_x)}{2}, & \vartheta_x(2-\vartheta_x), & \dfrac{\vartheta_x(\vartheta_x-1)}{2} \end{bmatrix} \right) = u_b(B),$$

$$\begin{bmatrix} u_{i+1/2,j} & u_{i+1/2-s_x,j} & u_{i+1/2-2s_x,j} \\ u_{i+1/2,j-s_y} & u_{i+1/2-s_x,j-s_y} & u_{i+1/2-2s_x,j-s_y} \\ u_{i+1/2,j-2s_y} & u_{i+1/2-s_x,j-2s_y} & u_{i+1/2-2s_x,j-2s_y} \end{bmatrix}$$

where $A \times B$ is the standard matrix multiplication, and $A \odot B = \sum_{r,s} A_{r,s} B_{r,s}$, or analogously:

$$\sum_{k_x,k_y=0}^{2} \mathcal{T}_{k_x}^{\vartheta_x}\, \mathcal{T}_{k_y}^{\vartheta_y}\, u_{i+1/2-k_x s_x, j-k_y s_y} = u_b(B), \quad \text{with } \mathcal{T}^\vartheta = \left[\frac{(1-\vartheta)(2-\vartheta)}{2},\ \vartheta(2-\vartheta),\ \frac{\vartheta(\vartheta-1)}{2} \right]^T. \quad (12)$$

The reason to take the stencil in the upwind direction is that we want to interpolate values of u from internal grid points and this approach ensures (for smooth boundaries and sufficiently fine grids) that the stencil contains as many internal grid points as possible. However, as in the example of Fig. 2, the stencil may contain external grid points other than G ($(x_{i-1/2}, y_j)$ is another ghost point). These grid points are marked as *ghost points* too, and a ghost value for them is defined repeating the entire procedure.

Analogously, we define ghost values for v on ghost points of the vertically staggered grid.

Remark 1 *The entire set of ghost points needed in the discretization is a subset of all external grid points that are within a predetermined distance $\delta = \sqrt{5}\,h$ from the boundary (referring to the notation of Fig. 2, the extreme case is when all grid points of the nine-point stencil except $(x_{i+1/2}, y_{j-1})$ are outside of the*

11

domain and the maximum distance between the external grid points of the stencil and the internal grid point $(x_{i+1/2}, y_{j-1})$ *is* $\delta = \sqrt{5}\,h$). *If the grid is sufficiently fine with respect to the curvature of* Γ, *this value can be decreased to* $\delta = \sqrt{2}\,h$, *as farther ghost points are actually not involved in the computation of the numerical solution inside* Ω. *For example, in Fig. 2, the ghost point* $(x_{i+1/2}, y_{j+1})$ *is not involved in the discretization of any internal equations, neither belongs to the nine-point stencil of any ghost values needed for internal discretizations. For simplicity, we mark as ghost points all external points within* δ *distance from the boundary. Another approach could be to scan all external points and mark as ghost points only those external points whose equation* (12) *is actually coupled with internal equation, following a similar approach of [20, 22]. We experienced that from a computational cost point of view the two approaches are practically similar.*

Remark 2 *Some existing ghost-point methods eliminate the ghost value from the internal equation by substituting its value with the extrapolated value from the boundary condition [33, 57]. This approach cannot be proposed here, as Eq.* (12) *may contain more than one ghost point and therefore the elimination of ghost values from the internal equation is not practical. This approach is called non-eliminated ghost values.*

Remark 3 *Existing ghost-point methods allow the definition of more ghost values on the same ghost point* G *if this is required from more than one internal equation [33, 23]. For example, looking at Fig. 2, the ghost value* $u_{i-1/2,j}$ *is required for the discretization of the internal equations at both* $(x_{i-1/2}, y_{j-1})$ *and* $(x_{i-3/2}, y_j)$. *The approach described in this paper, which is proposed by the author and Russo in [20, 22], define one single ghost value for each ghost point.*

Finally, internal equations and ghost interpolations (12) for u, v and p constitute a linear system in the unknowns $u_{i+1/2,j}$, $v_{i,j+1/2}$ and $p_{i,j}$, defined on all grid points of the horizontally staggered grid, vertically staggered grid and central grid, respectively. In summary, the linear system is obtained by writing a linear equation for each grid point P as described in the Algorithm 2. See Fig. 3 for an example of internal, ghost and inactive grid points. We notice that the distance $\mathrm{dist}(P, \Gamma)$ can be easily computed if the signed distance function is available, as $\mathrm{dist}(P, \Gamma) = \phi_d(P)$. Otherwise, $\mathrm{dist}(P, \Gamma) = d(P, B)$, where B is the boundary point obtained by Alg. 1.

3.4 Moving domains

The description of the numerical method proposed so far is designed for steady domains $\Omega^{(n)} = \Omega$, $\forall n \in \mathbb{N}$. In case of moving domain, there are some modifications to be considered. We observe that the explicit term $\tilde{\mathbf{f}}$ in Eq. (6)

$$\tilde{\mathbf{f}} = \mathbf{u}^{(n)} + \frac{\Delta t}{2\,\mathrm{Re}} \nabla^2 \mathbf{u}^{(n)} - \Delta t\,(\mathbf{u} \cdot \nabla \mathbf{u})^{(n+1/2)} + \Delta t\,\mathbf{f}^{(n+1/2)}$$

must be computed on internal grid points of $\Omega^{(n+1)}$. On the other hand, $\tilde{\mathbf{f}}$ contains values of $(\mathbf{u}, p)$ and of its derivatives evaluated at the previous time step, and therefore defined in $\Omega^{(n)}$, that may differ from $\Omega^{(n+1)}$. In particular, an issue arises when an internal grid point for $\Omega^{(n+1)}$ was an external point for $\Omega^{(n)}$,

Algorithm 2 In this algorithm we summarise the steps to be taken to write the linear system arising from the discretization of (9).

---

for P in the horizontally-staggered, vertically-staggered and central grids **do**

    if $\phi(P) < 0$ **then**

        mark P as internal grid point;

        the linear equation is obtained by discretizing the three PDEs (10) (first, second or third equation whether P is in the horizontally-staggered, vertically-staggered or central grid, respectively), using the discretization approach described in Sect. 3.2

    else if $\mathrm{dist}(P, \Gamma) < \delta$ **then**

        mark P as ghost point;

        the linear equation is obtained from (12) (boundary condition for u or for v whether P is in the horizontally-staggered or vertically-staggered grid, respectively)

    else

        mark P as inactive grid point;

        the linear equation is not needed (or it can be set up fictitiously as, for example, $u_{i+1,j} = 0$.)

    end if

end for

Write the linear equation (11).

---

as in Fig. 4, since central differences centred on ghost points of $\Omega^{(n)}$ may involve grid points that are too far from the boundary and for which there is no value of $(\mathbf{u}, p)$ available. This issue can be circumvented by enlarging the region of ghost points, i.e. by increasing the value of δ in the Algorithm 2 and in the Remark 1. In order to mitigate this effect, we ensure that the boundary does not move more than h by providing a suitable restriction for the time step Δt, that is $\Delta t < h / \max_{\partial\Omega^{(n)}} \left\| \mathbf{u}_b^{(n)} \right\|_2$, and then we set up $\delta = (\sqrt{5} + 1)h$.
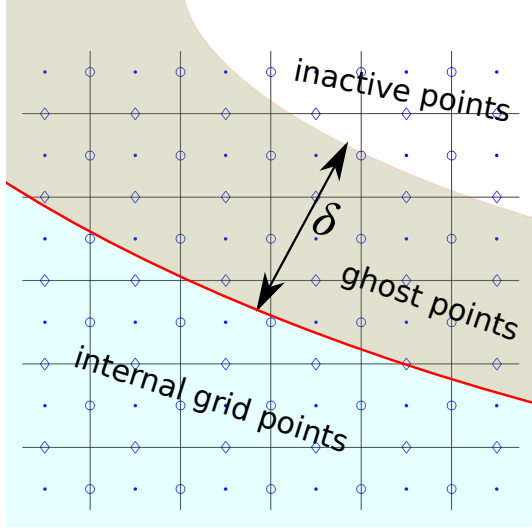
Fig. 3: The layer of ghost points (width δ) separate the internal grid points from the inactive grid points.
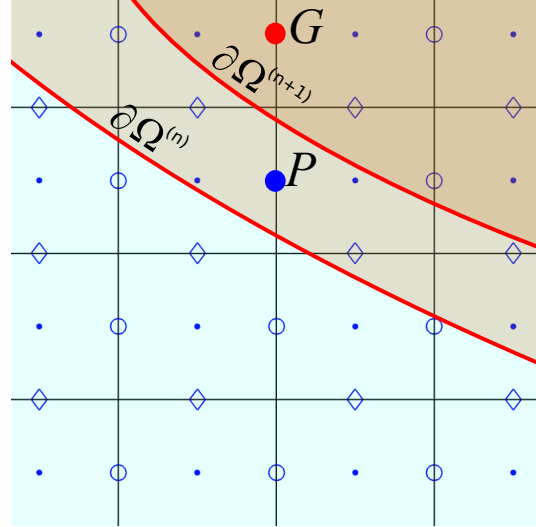

Fig. 4: The boundary $\partial\Omega$ moves from $\partial\Omega^{(n)}$ to $\partial\Omega^{(n+1)}$ in one time step. The grid point P is internal to the domain $\Omega^{(n+1)}$ and the discretization of the equations in P requires the value on G at the previous time step. In order to guarantee that a ghost value was computed in G at the previous time step, we have to increase (at each time step) the width of the narrow band of ghost points from $\delta = \sqrt{5}\,h$ to $\delta = (\sqrt{5}+1)h$ (or from $\delta = \sqrt{2}\,h$ to $\delta = (\sqrt{2}+1)h$ if the grid is sufficiently fine with respect to the curvature of the boundary).

Remark 4 *Following the Remark 1, if the grid is sufficiently fine with respect to the curvature of the boundary, the value of δ can be decreased to $\delta = (\sqrt{2}+1)\,h$.*

Remark 5 *The reverse case in which an internal grid point for $\Omega^{(n)}$ becomes a ghost point for $\Omega^{(n+1)}$ does not give any warnings, as the equation for that grid point is set up normally as described in Sect.3.3 enforcing boundary conditions on $\Omega^{(n+1)}$.*

14

4 Multigrid approach

The linear system obtained in Sect. 3 and described in Algorithm 2 can be summarised as

$$
\begin{pmatrix}
I_h^u - \alpha L_h^u & \mathbf{0} & (D_x)_h^u & \mathbf{0} \\
B_h^u & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & I_h^v - \alpha L_h^v & (D_y)_h^v & \mathbf{0} \\
\mathbf{0} & B_h^v & \mathbf{0} & \mathbf{0} \\
(D_x)_h^p & (D_y)_h^p & \mathbf{0} & -1 \\
\mathbf{0} & \mathbf{0} & \mathbf{1}_h & \mathbf{0}
\end{pmatrix}
\cdot
\begin{pmatrix}
u_h \\
v_h \\
p_h \\
\xi
\end{pmatrix}
=
\begin{pmatrix}
\tilde{f}_h^u \\
u_b \\
\tilde{f}_h^v \\
v_b \\
0 \\
0
\end{pmatrix},
\quad \text{or in a compact form:} \quad M_h \mathbf{u}_h = \mathbf{b}_h, \qquad (13)
$$

where $I_h^{u,v}$ is the identity matrix, $L_h^{u,v}$ is the discretization of the Laplacian operator, $B_h^{u,v}$ the discrete boundary condition operators, $(D_{x,y})_h^{u,v,p}$ the central difference operrartor for the first derivative, and $\mathbf{1}_h$ a vector of ones.

This is a sparse non-symmetric linear system, whose solution is obtained efficiently by adopting a geometric multigrid approach. Multigrid methods are efficient methods to solve sparse linear systems arising from the discretization of a class of PDEs. The aim of a multigrd method is to solve the high-frequency components of the solution on a fine grid and the low-frequency components on coarser grids [61, 39, 10]. Initially designed for elliptic equations, their implementation has been extended to PDEs that have similarities with elliptic equations, in particular those (systems of) equations whose discretization presents a high *ellipticity* factor, defined as a quantity to establish the qualitative resemblance of the discrete operator to the elliptic operator (see [61, Ch. 4]. In the recent decades, they have also been designed to several other classes of PDEs, including parabolic [32] and hyperbolic equations [56], as well as to general sparse linear systems whose derivation is not (or is partially) known, leading to the class of *Algebraic Multigrid Methods* [60] (in contrast with geometric multigrid methods), where geometric information on the grid or on the relation between the linear equations is not available.

In this paper we adopt a geometric multigrid method specifically designed for ghost-point methods, extending the idea proposed in [20, 22] and combining the approach with traditional multigrid methods developed for Navier-Stokes equations for rectangular domains. We confine our explanation to the main components of the multigrid approach, i.e. relaxation operator, restriction operation and interpolation operator, from which the entire multigrid algorithm is uniquely defined. For a comprehensive explanation of multigrid methods see, for example, [61, 39, 10].

4.1 Relaxation operator

The most important property that a relaxation operator must satisfy for a multigrid performance is the smoothing property, defined as the ability of the relaxation scheme to smooth (i.e. quickly dump high-frequency components of) the residual $\mathbf{r}_h = \mathbf{b}_h - M_h \tilde{\mathbf{u}}_h$, where $\tilde{\mathbf{u}}_h$ is the approximation of the solution obtained after few relaxation steps. The relaxation operator is also called smoother. For elliptic equations discretized by standard central-differences, the Gauss-Seidel scheme is an appropriate smoother, as well as weighted-Jacobi scheme, with weight $\omega_{\text{Jac}} = 4/5$ in 2D, while non-weighted Jacobi does not have the

smoothing property. Several types of Gauss-Seidel schemes can be defined, depending on the order chosen to swap the grid points, i.e. the equations of the linear systems. Common choices are lexicographic order and red-black. Although the latter is more efficient and suitable for parallelization, we focus in this paper on the Gauss-Seidel lexicographic order (GS-LEX) for simplicity, keeping in mind that the ideas can be straightforwardly extended to other Gauss-Seidel schemes.

The linear system (13) is made by internal equations and ghost equations. The smoother will swap onto all grid nodes, acting in a different way whether it is on internal grid points or on ghost grid nodes, as described in the next sections.

4.1.1 Internal smoother

For the internal equations, we follow the approach described in cite [61, Sect. 8.7] and briefly summarised here. First of all, a standard Gauss-Seidel scheme does not act as an efficient smoother for systems of PDEs. A proper adjustment to retrieve the smoothing property is that for every single node we update simultaneously all variables defined at that node. This idea leads to the solution of small $n \times n$ systems, where n is the number of variables defined in that grid node. This approach is called *collective* relaxation. For the linear system (13), a collective relaxation is not properly defined, as for example the variables defined in the nodes of the central grid, i.e. $p_{i,j}$, do not appear in the linear equations associated with those nodes, that are

$$\frac{u_{i+1/2,j} - u_{i-1/2,j} + v_{i,j+1/2} - v_{i,j-1/2}}{h} = \xi,$$

or, in other terms, the linear system has zero diagonal entries. A modification to overcome this issue is the *box relaxation*: for each cell we update simultaneously all variables defined in that cell, including cell borders. This leads to the solution of 5×5 systems for each cell, as the variables defined in the cell with centre (x_i, y_j) are $p_{i,j}, u_{i\pm1/2,j}, v_{i,j\pm1/2}$. We observe that variables defined on the border cells are updated twice during one relaxation step, as for example $v_{i,j+1/2}$ is updated for the cells having centres (x_i, y_j) and (x_i, y_{j+1}).

4.1.2 Ghost smoother

The relaxation scheme in the ghost points is defined using a similar approach to the one adopted in [20, 22] for scalar equations. Since a naïve Gauss-Seidel approach does not converge for ghost equations, we adopt a fictitious-time relaxation on the boundary conditions. In detail, let $B(w) = g$ represent a generic boundary condition for the variable w (e.g. $B(w) = w$ for Dirichlet boundary conditions and $B(w) = \partial w/\partial n$ for Neumann boundary conditions). The relaxation scheme is obtained by discretising the associated fictitious-time boundary condition:

$$\frac{\partial w}{\partial \tau} + Bw = g, \tag{14}$$

where τ is a fictitious time. Observe that the steady-state solution of (14) is the original boundary condition $B(w) = g$: therefore (provided that we have convergence) the solution of (14) will approach asymptotically a solution w that satisfies $B(w) = g$ on the boundary. Since the accuracy of the method is not affected by

the (fictitious) time discretization, we can discretize the time derivative with first order accuracy, and in particular we use forward Euler method. In addition, we can discretize the time derivative of Eq. (14) in a convenient point rather than on a boundary point, for example on the grid points in which w is defined, obtaining:

$$w_{r,s}^{(m+1)} = w_{r,s}^{(m)} + \Delta\tau_w \left(g - Bw^{(m)} \right), \tag{15}$$

where $Bw^{(m)}$ is the discretization in space described in Sect. 3.3. The fictitious time step $\Delta\tau_w$ is an additional parameter that must be defined in order to guarantee the convergence of the relaxation scheme. It is observed numerically that the following conditions will guarantee convergence (see [20, 22] for a theoretical explanation):

$0 < \Delta\tau_w < 1$ for Dirichlet boundary conditions, $\quad 0 < \Delta\tau_w < h$ for Neumann boundary conditions,

where h is the spatial step. In practice, we choose $\Delta\tau_w = 0.9$ and $\Delta\tau_w = 0.9\,h$, respectively.

In order to guarantee the smoothing property and cancel possible boundary effects that would affect the multigrid performance, the relaxation over the ghost points is performed a certain number of times, say λ, against each step of the internal iterations.

In summary, Algorithm 3 describes how ν relaxation steps are performed.

Algorithm 3 Summary of ν iteration steps of the relaxation scheme.

---
for iter1 from 1 to ν **do**
   for $(x_i, y_j) \in \Omega$ **do**
      Box-relaxation over the cell centred in (x_i, y_j)
   end for
   for iter2 from 1 to λ **do**
      for each ghost point of the central, horizontally staggered and vertically staggered grid **do**
         ghost relaxation (15)
      end for
   end for
end for

4.2 Restriction operator

After few relaxations (say ν_1) on the fine grid (with a spatial step h), an approximation $\tilde{\mathbf{u}}_h$ of the final solution is obtained. Then, the residual $\mathbf{r}_h = \mathbf{b}_h - M_h\tilde{\mathbf{u}}_h$ is computed and transferred (restricted) to a coarser grid, i.e. a staggered mesh with a spatial step $H = 2h$, where the residual equation is solved: $M_H\mathbf{e}_H = \mathbf{r}_H$. Therefore, proper restriction operators must be defined. Since the residual $\mathbf{r}_h = (\mathbf{r}_h^u, \mathbf{r}_h^v, \mathbf{r}_h^p)$ is defined on the horizontally-staggered, vertically-staggered and central grids (for the residuals $\mathbf{r}_h^u$ of the horizontal velocity, $\mathbf{r}_h^v$ of the vertical velocity, and $\mathbf{r}_h^p$ of the continuity equation, respectively), then we have to define three restriction operators $\mathbf{r}_H^u = \mathcal{I}_H^u(\mathbf{r}_h^u)$, $\mathbf{r}_H^v = \mathcal{I}_H^v(\mathbf{r}_h^v)$ and $\mathbf{r}_H^p = \mathcal{I}_H^p(\mathbf{r}_h^p)$.

The standard restriction operators for staggered grids are (see [61, Sect. 8.7.1]):

$$\mathcal{I}_H^u = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ & \cdot & \\ 1 & 2 & 1 \end{bmatrix}, \quad \mathcal{I}_H^v = \frac{1}{8}\begin{bmatrix} 1 & & 1 \\ 2 & \cdot & 2 \\ 1 & & 1 \end{bmatrix}, \quad \mathcal{I}_H^p = \frac{1}{4}\begin{bmatrix} 1 & & 1 \\ & \cdot & \\ 1 & & 1 \end{bmatrix}. \tag{16}$$

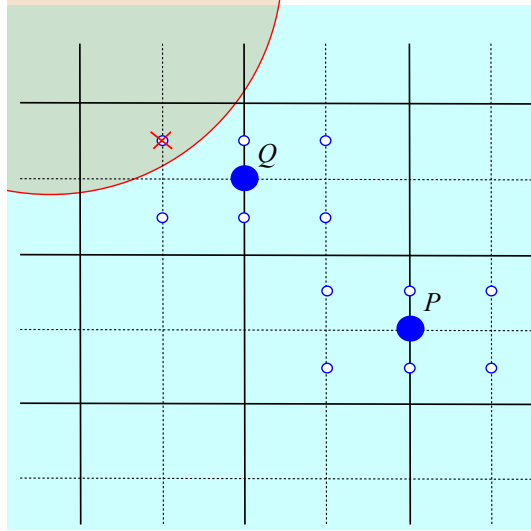An example on the horizontally staggered grid is represented in Fig. 5.

Fig. 5: Restriction operation $\mathbf{r}_H^u = \mathcal{I}_H^u(\mathbf{r}_h^u)$ for the horizontally staggered grid. The grid with spatial step h is represented by dashed and bold solid lines, while the grid with spatial step $H = 2h$ represented by bold solid lines. The restriction value $\mathbf{r}_H^u(P)$ is obtained by using the standard restriction operator (first stencil of Eq. 16), since the grid points of the stencil are all internal (represented by the six empty small circles surrounding P). The restriction value $\mathbf{r}_H^u(Q)$ is obtained by using the restriction operator of Eq. (17), since the grid points of the stencil are all internal except the one at the top-left.

Since the smoother acting on the internal equations (Sect. 4.1.1) is different from the one acting on the ghost points (Sect. 4.1.2), the residual that is obtained after few relaxations, although is smooth either internally or along the boundary, might appear discontinuous from inside grid points to ghost points. For this reason, following the idea proposed in [20, 22], the restriction of the residual for internal grid points is performed without involving the values obtained in the ghost points.

In practice, for each internal grid point of the coarser grid with spatial step H, if the grid points of the finer grid with spatial step h that are involved in the restriction operator are all internal grid points, than the usual restriction (16) is performed, otherwise the restriction is the average of the residual on the internal grid points of the stencil. For example, assume that a grid point Q of the coarser horizontally-staggered grid is close to the boundary and that the grid points of the finer grid involved in the restriction are all inside Ω

except the point at the top-left (ghost point), then the restriction is (see Fig. 5)

$$\mathcal{I}_H^u = \frac{1}{5} \begin{bmatrix} 0 & 1 & 1 \\ & \cdot & \\ 1 & 1 & 1 \end{bmatrix}. \tag{17}$$

The restriction of the residual of the ghost equations is performed in the same way, i.e. only involving ghost values and ignoring the values of the residual of internal equations.

4.3 Interpolation operator

Once the restriction of the residual $\mathbf{r}_H$ is computed and the residual equation $M_H \mathbf{e}_H = \mathbf{r}_H$ is solved on the coarser grid, the error $\mathbf{e}_H$ is interpolated back from the coarser grid to the finer grid $\mathbf{e}_h$. Since the error is the solution of the residual equation, we do not expect any discontinuities from the internal values to the ghost values. Therefore, the standard interpolation operator is adopted, either for internal or ghost values. In practice, for each grid point on the fine grid, we compute the bilinear interpolation on the coarser grid points that surround it (see, for example, [61, Sect. 8.7.1]). Therefore, the approximation $\tilde{\mathbf{u}}_h$ is updated as $\tilde{\mathbf{u}}_h := \tilde{\mathbf{u}}_h + \mathbf{e}_h$, and additional ν_2 relaxations are performed on the fine grid. The final approximation is the result of the Two-Grid Correction Scheme (TGCS).

When solving the residual equation in the grid with spatial step H, one can recursively apply a multigrid iteration again, moving to a coarser grid with spatial step $2H$, and so on until a very coarse grid is reached, on which the (small) linear system is solved exactly by a standard solver. Examples of multigrid approaches based on this strategy are $V-$, $W-$ and $F-$cycles (we refer to [61] for a more comprehensive description). In this paper, we adopt the $W-$cycle approach, although the technique can be easily extended to other approaches.

5 Numerical tests

In this section we perform several numerical tests to assess the accuracy of the discretization and the efficiency of the multigrid. The computational domain is $\mathcal{D} = (-1, 1)^2$ and the fluid domain is $\Omega = \mathcal{D} \backslash \mathcal{R}(t)$, where $\mathcal{R}(t)$ represents the object at time t, described by a level-set function (Sect. 3.1). Different shapes will be tested: circle, ellipse and flower-shaped, defined at time $t = 0$ as follows (see Fig. 6):

$$\text{CIRCLE:} \qquad \phi(x, y) = R^2 - (x^2 + y^2), \quad R = \frac{1}{\sqrt{15}},$$

$$\text{ELLIPSE:} \quad \phi(x, y) = 1 - \left(\frac{(\cos(\vartheta)x - \sin(\vartheta)y)^2}{A^2} + \frac{(\sin(\vartheta)x + \cos(\vartheta)y)^2}{B^2} \right), \quad A = \frac{1}{\sqrt{14}}, \ B = \frac{1}{\sqrt{2}}, \ \vartheta = \frac{\pi}{6},$$

$$\text{FLOWER:} \qquad \phi(x, y) = \sqrt{x^2 + y^2} - A - \frac{B(y^5 + 5x^4 y - 10x^2 y^3)}{\sqrt{x^2 + y^2}^5}, \quad A = 0.5, \ B = 0.15.$$

$$\tag{18}$$

19

For steady domains, the object will remain steady for the entire simulation, i.e. $\mathcal{R}(t) = \mathcal{R}(0)$, while for moving domains we rotate the initial object $\mathcal{R}(0)$ around the origin with a constant angular velocity $\omega \in \mathbb{R}$, i.e. $\mathcal{R}(t)$ is the rotation of $\mathcal{R}(0)$ by an angle $\omega\,t$. The level-set function at time t is computed as $\phi(x,y,t) = \phi(\bar{x},\bar{y},0)$, where $(\bar{x},\bar{y})$ is the point obtained rotating (x,y) by an angle $-\omega\,t$ around the origin, i.e.

$$\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} \cos(\omega t) & -\sin(\omega t) \\ \sin(\omega t) & \cos(\omega t) \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}.$$
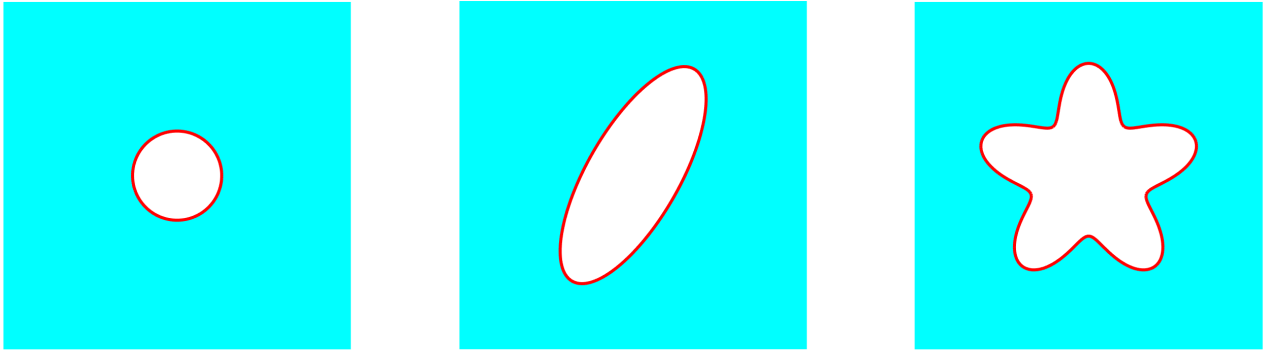
Fig. 6: Different shapes of the object $\mathcal{R}(0)$ adopted in the numerical tests: CIRCLE (left), ELLIPSE (middle) and FLOWER (right).

5.1 Accuracy of the discretization

In this section we test the accuracy of the method for the different domains presented in (18). We choose an exact solution $\mathbf{u}_{\mathrm{exa}} = (u_{\mathrm{exa}}, v_{\mathrm{exa}}, p_{\mathrm{exa}})$ and adjust the right-hand side of (9) accordingly. Since the exact solution is not necessary a physical solution, we do not expect in general that it satisfies the continuity equation. To this purpose, the continuity equation is generalised to $\nabla \cdot \mathbf{u} - \xi = \mathbf{g}$, where $\mathbf{g} = \nabla \cdot \mathbf{u}_{\mathrm{exa}}$. Source term $\mathbf{f}$ and boundary conditions are defined from the exact solution.

The exact solution that we use in the tests is:

$$u_{\mathrm{exa}} = \cos(5x)\cos(6y\log(t+2)),\ v_{\mathrm{exa}} = \sin(4t)\sin(3x^2 + 4y^2 + 2),\ p_{\mathrm{exa}} = \cos(6xt)\sin(2yt)\log(3t+1) + \sin(5t). \tag{19}$$

Reynolds number is Re $= 100$. The computational domain $\mathcal{D} = (-1,1)^2$ is divided in N^2 cells. As described in Sect. 2, discretization in time is performed using a Crank-Nicholson approach, which is unconditionally stable. Moreover, for moving domains the time step is chosen in such a way that the object does not move more than the spatial step h at each time step (see the restriction on the time step provided in Sect. 3.4). Therefore, the time step chosen for the numerical tests is

$$\Delta t \leq \min\left\{ h, \frac{h}{\max \|\mathbf{u}\|_2} \right\}.$$

20

For each test, we choose several values of N and for each of them we compute the error as the difference between the exact solution and the numerical solution at time $t = 10$. We plot the errors against N in logarithmic scale, together with the best fit line to compute an approximation of the accuracy order. The values are also reported in tables. We perform two tests for each geometry of Eq. (18): one test for the steady case and one test for the moving geometry, with an angular velocity of $\omega = 2\pi/5$. We use the notation STEADY/MOVING GEOMETRY, where GEOMETRY is CIRCLE, ELLIPSE or FLOWER.

In Figs. 7, 8 and 9 we plot the best fit line for the tests STEADY CIRCLE, STEADY ELLIPSE and STEADY FLOWER, respectively. The error values are reported in Tables 1, 2 and 3. The tests with moving domains, i.e. MOVING CIRCLE, MOVING ELLIPSE and MOVING FLOWER, are plotted in Figs. 10, 11 and 12, respectively, with the error values reported in Tables 4, 5 and 6.

The best fit line shows second order accuracy for all tests in the L^1 norm, and for most tests in the L^∞ norm. The only test that does not show a clear second order accuracy in the L^∞ norm is the MOVING FLOWER test, although we expect to obtain a second order accuracy asymptotically. In fact, we observe (not shown here) that the numerical error is mainly concentrated close to the boundary of the object and we speculate that a more stable second order decay is observed when the ratio κ/N is sufficiently small, where κ is the curvature of the boundary. Since the curvature of the flower is in some regions higher than the one of the circle, we expect to observe a clear decay for the flower domain with a higher value of N. In general, the norms of the numerical errors fluctuate in all tests around the best fit line, especially in the L^∞ norm. This is mainly due to interpolation errors of ghost values. In fact, from one value of N to the other, the structure of the grid and then of the ghost points around the boundary can significantly change. The effect is more evident for more complex geometries, as the flower-shaped object, and for moving domains. The error of $\nabla \cdot \mathbf{u}$ is computed as $\|\mathbf{g} - \nabla \cdot \mathbf{u}\|$. Due to the presence of ξ in the continuity equation (see Eq. 9), we do not expect to have a divergence-free solution (or, in this case, a solution that satisfies the equation $\nabla \cdot \mathbf{u} = \mathbf{g}$). What we observe is however a second order decay of the error. We observe that the L^1 and L^∞ norms of the error of $\nabla \cdot \mathbf{u}$ are the same, since $\nabla \cdot \mathbf{u} - \mathbf{g} = \xi$ is constant in Ω.
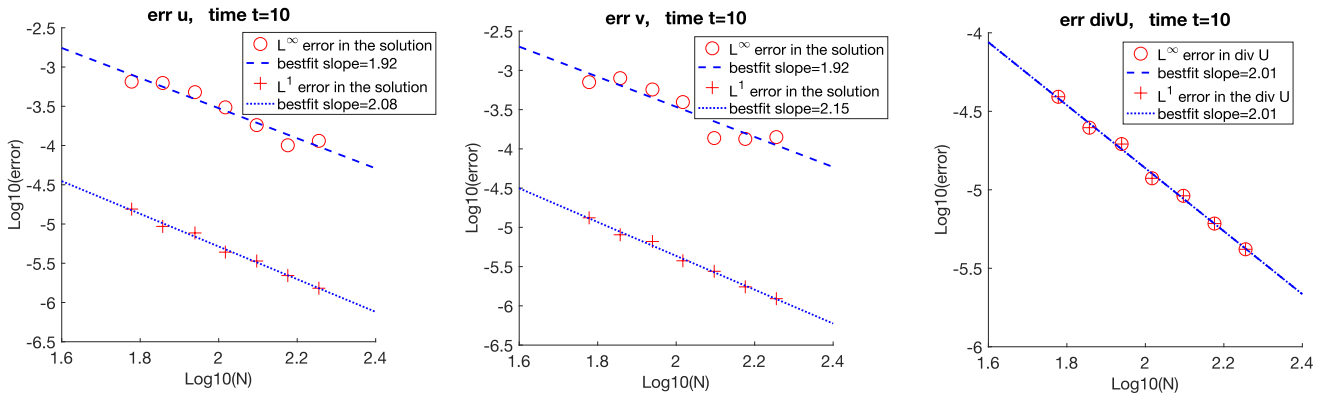

Fig. 7: Test STEADY CIRCLE: best fit accuracy order at time $t = 10$. The exact solution is (19).

No. of points	L^1 error of u	order	L^1 error of v	order	L^1 error of $\nabla \cdot \mathbf{u}$	order
60×60	$1.55 \cdot 10^{-5}$	-	$1.32 \cdot 10^{-5}$	-	$3.91 \cdot 10^{-5}$	-
72×72	$9.33 \cdot 10^{-6}$	2.79	$8.05 \cdot 10^{-6}$	2.73	$2.49 \cdot 10^{-5}$	2.49
87×87	$7.69 \cdot 10^{-6}$	1.02	$6.60 \cdot 10^{-6}$	1.05	$1.96 \cdot 10^{-5}$	1.27
104×104	$4.38 \cdot 10^{-6}$	3.15	$3.75 \cdot 10^{-6}$	3.16	$1.18 \cdot 10^{-5}$	2.82
125×125	$3.38 \cdot 10^{-6}$	1.42	$2.76 \cdot 10^{-6}$	1.68	$9.16 \cdot 10^{-6}$	1.40
150×150	$2.20 \cdot 10^{-6}$	2.34	$1.75 \cdot 10^{-6}$	2.51	$6.09 \cdot 10^{-6}$	2.23
180×180	$1.52 \cdot 10^{-6}$	2.03	$1.23 \cdot 10^{-6}$	1.90	$4.18 \cdot 10^{-6}$	2.07
No. of points	L^∞ error of u	order	L^∞ error of v	order	L^∞ error of $\nabla \cdot \mathbf{u}$	order
60×60	$6.52 \cdot 10^{-4}$	-	$7.08 \cdot 10^{-4}$	-	$3.91 \cdot 10^{-5}$	-
72×72	$6.26 \cdot 10^{-4}$	0.22	$7.96 \cdot 10^{-4}$	-0.64	$2.49 \cdot 10^{-5}$	2.49
87×87	$4.79 \cdot 10^{-4}$	1.42	$5.71 \cdot 10^{-4}$	1.75	$1.96 \cdot 10^{-5}$	1.27
104×104	$3.06 \cdot 10^{-4}$	2.50	$3.95 \cdot 10^{-4}$	2.07	$1.18 \cdot 10^{-5}$	2.82
125×125	$1.82 \cdot 10^{-4}$	2.83	$1.37 \cdot 10^{-4}$	5.74	$9.16 \cdot 10^{-6}$	1.40
150×150	$1.01 \cdot 10^{-4}$	3.24	$1.34 \cdot 10^{-4}$	0.14	$6.09 \cdot 10^{-6}$	2.23
180×180	$1.15 \cdot 10^{-4}$	-0.72	$1.41 \cdot 10^{-4}$	-0.29	$4.18 \cdot 10^{-6}$	2.07

Table 1: Test Steady Circle: errors and accuracy orders in the L^1 (top) and L^∞ (bottom) norms, for u (left), v (middle) and $\nabla \cdot \mathbf{u}$ (right). The exact solution is (19).
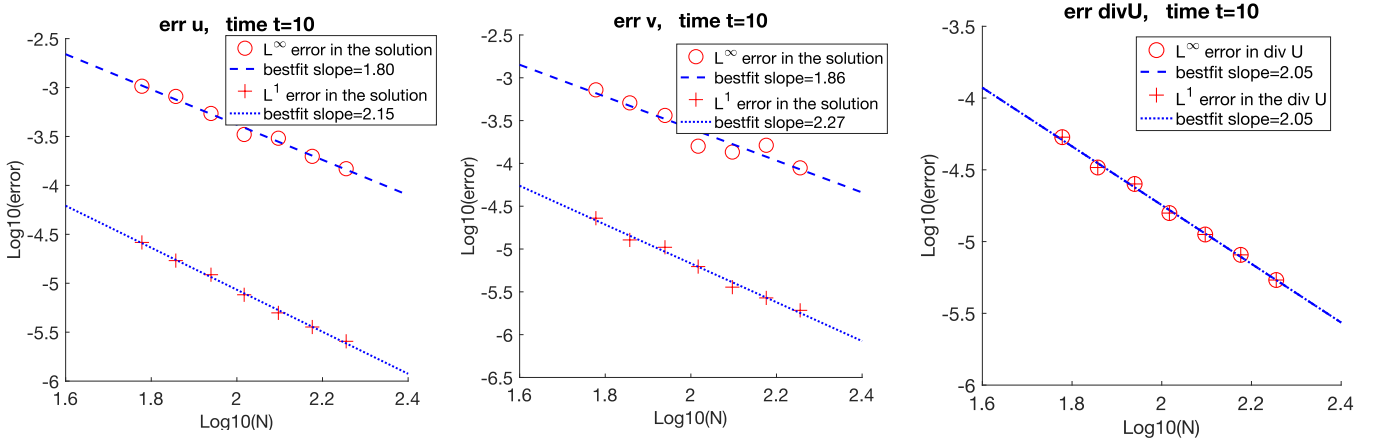

Fig. 8: Test Steady Ellipse: best fit accuracy order at time $t = 10$. The exact solution is (19).

No. of points	L^1 error of u	order	L^1 error of v	order	L^1 error of $\nabla \cdot \mathbf{u}$	order
60×60	$2.61 \cdot 10^{-5}$	-	$2.29 \cdot 10^{-5}$	-	$5.34 \cdot 10^{-5}$	-
72×72	$1.70 \cdot 10^{-5}$	2.35	$1.28 \cdot 10^{-5}$	3.21	$3.28 \cdot 10^{-5}$	2.67
87×87	$1.22 \cdot 10^{-5}$	1.75	$1.05 \cdot 10^{-5}$	1.04	$2.52 \cdot 10^{-5}$	1.39
104×104	$7.63 \cdot 10^{-6}$	2.65	$6.24 \cdot 10^{-6}$	2.91	$1.58 \cdot 10^{-5}$	2.62
125×125	$4.98 \cdot 10^{-6}$	2.31	$3.59 \cdot 10^{-6}$	3.01	$1.12 \cdot 10^{-5}$	1.87
150×150	$3.57 \cdot 10^{-6}$	1.83	$2.69 \cdot 10^{-6}$	1.59	$8.08 \cdot 10^{-6}$	1.79
180×180	$2.55 \cdot 10^{-6}$	1.85	$1.92 \cdot 10^{-6}$	1.83	$5.40 \cdot 10^{-6}$	2.21
No. of points	L^∞ error of u	order	L^∞ error of v	order	L^∞ error of $\nabla \cdot \mathbf{u}$	order
60×60	$1.03 \cdot 10^{-3}$	-	$7.23 \cdot 10^{-4}$	-	$5.34 \cdot 10^{-5}$	-
72×72	$8.12 \cdot 10^{-4}$	1.32	$5.09 \cdot 10^{-4}$	1.92	$3.28 \cdot 10^{-5}$	2.67
87×87	$5.44 \cdot 10^{-4}$	2.12	$3.63 \cdot 10^{-4}$	1.79	$2.52 \cdot 10^{-5}$	1.39
104×104	$3.31 \cdot 10^{-4}$	2.77	$1.59 \cdot 10^{-4}$	4.63	$1.58 \cdot 10^{-5}$	2.62
125×125	$3.04 \cdot 10^{-4}$	0.48	$1.36 \cdot 10^{-4}$	0.86	$1.12 \cdot 10^{-5}$	1.87
150×150	$1.98 \cdot 10^{-4}$	2.34	$1.63 \cdot 10^{-4}$	-1.00	$8.08 \cdot 10^{-6}$	1.79
180×180	$1.48 \cdot 10^{-4}$	1.59	$8.87 \cdot 10^{-5}$	3.33	$5.40 \cdot 10^{-6}$	2.21

Table 2: Test Steady Ellipse: errors and accuracy orders in the L^1 (top) and L^∞ (bottom) norms, for u (left), v (middle) and $\nabla \cdot \mathbf{u}$ (right). The exact solution is (19).

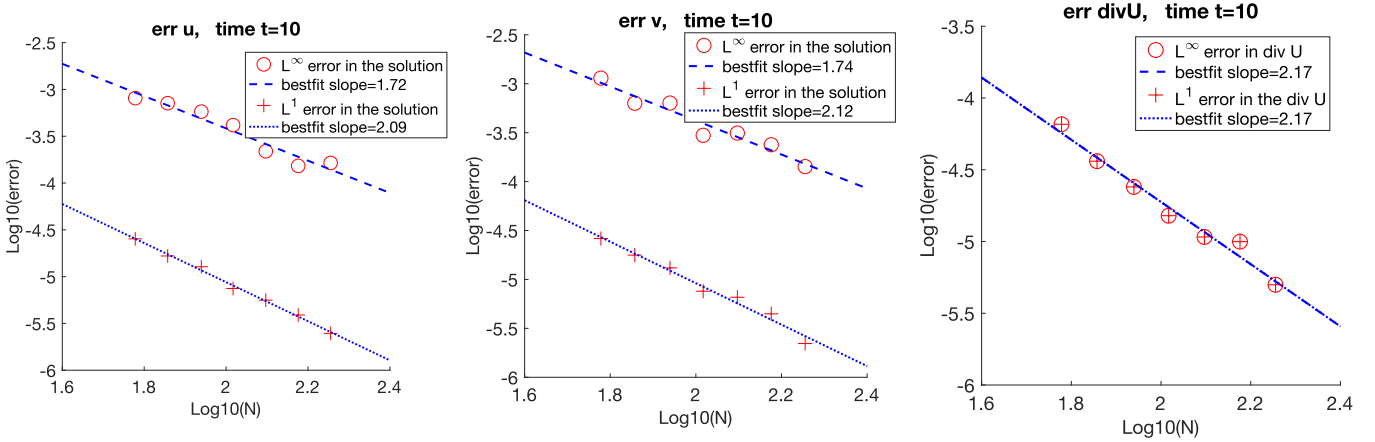Fig. 9: Test Steady Flower: best fit accuracy order at time $t = 10$. The exact solution is (19).

No. of points	L^1 error of u	order	L^1 error of v	order	L^1 error of $\nabla \cdot \mathbf{u}$	order
60×60	$2.53 \cdot 10^{-5}$	-	$2.62 \cdot 10^{-5}$	-	$6.56 \cdot 10^{-5}$	-
72×72	$1.66 \cdot 10^{-5}$	2.31	$1.77 \cdot 10^{-5}$	2.14	$3.63 \cdot 10^{-5}$	3.25
87×87	$1.28 \cdot 10^{-5}$	1.40	$1.31 \cdot 10^{-5}$	1.59	$2.40 \cdot 10^{-5}$	2.18
104×104	$7.47 \cdot 10^{-6}$	3.00	$7.58 \cdot 10^{-6}$	3.08	$1.51 \cdot 10^{-5}$	2.59
125×125	$5.60 \cdot 10^{-6}$	1.56	$6.57 \cdot 10^{-6}$	0.77	$1.08 \cdot 10^{-5}$	1.86
150×150	$3.89 \cdot 10^{-6}$	2.00	$4.45 \cdot 10^{-6}$	2.14	$9.98 \cdot 10^{-6}$	0.41
180×180	$2.47 \cdot 10^{-6}$	2.48	$2.21 \cdot 10^{-6}$	3.83	$5.00 \cdot 10^{-6}$	3.79
No. of points	L^∞ error of u	order	L^∞ error of v	order	L^∞ error of $\nabla \cdot \mathbf{u}$	order
60×60	$8.07 \cdot 10^{-4}$	-	$1.14 \cdot 10^{-3}$	-	$6.56 \cdot 10^{-5}$	-
72×72	$7.13 \cdot 10^{-4}$	0.68	$6.32 \cdot 10^{-4}$	3.24	$3.63 \cdot 10^{-5}$	3.25
87×87	$5.79 \cdot 10^{-4}$	1.10	$6.35 \cdot 10^{-4}$	-0.02	$2.40 \cdot 10^{-5}$	2.18
104×104	$4.16 \cdot 10^{-4}$	1.85	$2.97 \cdot 10^{-4}$	4.26	$1.51 \cdot 10^{-5}$	2.59
125×125	$2.18 \cdot 10^{-4}$	3.50	$3.13 \cdot 10^{-4}$	-0.29	$1.08 \cdot 10^{-5}$	1.86
150×150	$1.53 \cdot 10^{-4}$	1.97	$2.38 \cdot 10^{-4}$	1.51	$9.98 \cdot 10^{-6}$	0.41
180×180	$1.64 \cdot 10^{-4}$	-0.39	$1.42 \cdot 10^{-4}$	2.82	$5.00 \cdot 10^{-6}$	3.79

Table 3: Test STEADY FLOWER: errors and accuracy orders in the L^1 (top) and L^∞ (bottom) norms, for u (left), v (middle) and $\nabla \cdot \mathbf{u}$ (right). The exact solution is (19).
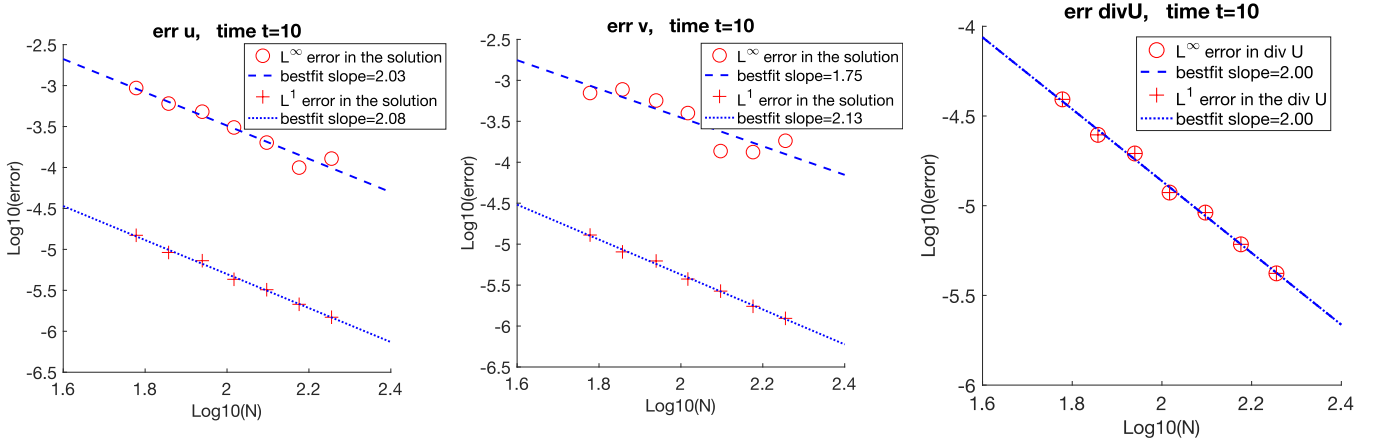

Fig. 10: Test MOVING CIRCLE: best fit accuracy order at time $t = 10$. The exact solution is (19); the angular velocity is $\omega = 2\pi/5$.

No. of points	L^1 error of u	order	L^1 error of v	order	L^1 error of $\nabla \cdot \mathbf{u}$	order
60×60	$1.49 \cdot 10^{-5}$	-	$1.29 \cdot 10^{-5}$	-	$3.91 \cdot 10^{-5}$	-
72×72	$9.17 \cdot 10^{-6}$	2.65	$8.05 \cdot 10^{-6}$	2.60	$2.48 \cdot 10^{-5}$	2.50
87×87	$7.28 \cdot 10^{-6}$	1.22	$6.23 \cdot 10^{-6}$	1.35	$1.95 \cdot 10^{-5}$	1.26
104×104	$4.31 \cdot 10^{-6}$	2.93	$3.75 \cdot 10^{-6}$	2.85	$1.18 \cdot 10^{-5}$	2.81
125×125	$3.22 \cdot 10^{-6}$	1.59	$2.67 \cdot 10^{-6}$	1.85	$9.16 \cdot 10^{-6}$	1.40
150×150	$2.14 \cdot 10^{-6}$	2.25	$1.74 \cdot 10^{-6}$	2.34	$6.09 \cdot 10^{-6}$	2.24
180×180	$1.48 \cdot 10^{-6}$	1.99	$1.24 \cdot 10^{-6}$	1.86	$4.20 \cdot 10^{-6}$	2.04
No. of points	L^∞ error of u	order	L^∞ error of v	order	L^∞ error of $\nabla \cdot \mathbf{u}$	order
60×60	$9.34 \cdot 10^{-4}$	-	$7.01 \cdot 10^{-4}$	-	$3.91 \cdot 10^{-5}$	-
72×72	$6.03 \cdot 10^{-4}$	2.40	$7.75 \cdot 10^{-4}$	-0.55	$2.48 \cdot 10^{-5}$	2.50
87×87	$4.80 \cdot 10^{-4}$	1.20	$5.67 \cdot 10^{-4}$	1.65	$1.95 \cdot 10^{-5}$	1.26
104×104	$3.08 \cdot 10^{-4}$	2.48	$3.99 \cdot 10^{-4}$	1.98	$1.18 \cdot 10^{-5}$	2.81
125×125	$2.01 \cdot 10^{-4}$	2.31	$1.37 \cdot 10^{-4}$	5.81	$9.16 \cdot 10^{-6}$	1.40
150×150	$9.97 \cdot 10^{-5}$	3.86	$1.33 \cdot 10^{-4}$	0.16	$6.09 \cdot 10^{-6}$	2.24
180×180	$1.28 \cdot 10^{-4}$	-1.39	$1.83 \cdot 10^{-4}$	-1.75	$4.20 \cdot 10^{-6}$	2.04

Table 4: Test Moving Circle: errors and accuracy orders in the L^1 (top) and L^∞ (bottom) norms, for u (left), v (middle) and $\nabla \cdot \mathbf{u}$ (right). The exact solution is (19); the angular velocity is $\omega = 2\pi/5$.
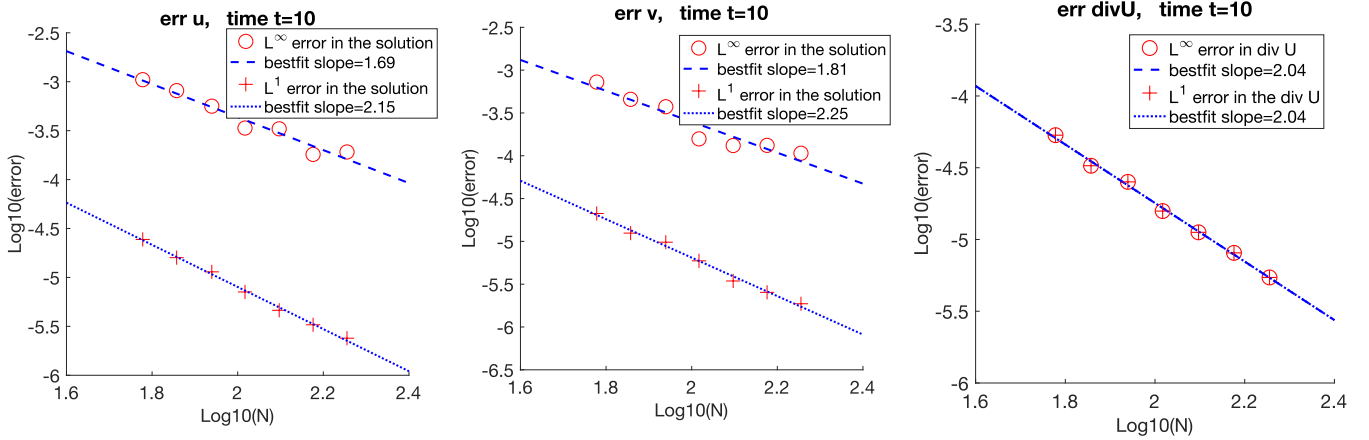

Fig. 11: Test Moving Ellipse: best fit accuracy order at time $t = 10$. The exact solution is (19); the angular velocity is $\omega = 2\pi/5$.

No. of points	L^1 error of u	order	L^1 error of v	order	L^1 error of $\nabla \cdot \mathbf{u}$	order
60×60	$2.44 \cdot 10^{-5}$	-	$2.12 \cdot 10^{-5}$	-	$5.33 \cdot 10^{-5}$	-
72×72	$1.59 \cdot 10^{-5}$	2.35	$1.25 \cdot 10^{-5}$	2.90	$3.27 \cdot 10^{-5}$	2.69
87×87	$1.14 \cdot 10^{-5}$	1.77	$9.78 \cdot 10^{-6}$	1.29	$2.52 \cdot 10^{-5}$	1.37
104×104	$7.11 \cdot 10^{-6}$	2.65	$5.93 \cdot 10^{-6}$	2.81	$1.58 \cdot 10^{-5}$	2.63
125×125	$4.61 \cdot 10^{-6}$	2.35	$3.45 \cdot 10^{-6}$	2.95	$1.12 \cdot 10^{-5}$	1.85
150×150	$3.28 \cdot 10^{-6}$	1.87	$2.54 \cdot 10^{-6}$	1.68	$8.07 \cdot 10^{-6}$	1.81
180×180	$2.39 \cdot 10^{-6}$	1.73	$1.87 \cdot 10^{-6}$	1.68	$5.44 \cdot 10^{-6}$	2.16
No. of points	L^∞ error of u	order	L^∞ error of v	order	L^∞ error of $\nabla \cdot \mathbf{u}$	order
60×60	$1.05 \cdot 10^{-3}$	-	$7.25 \cdot 10^{-4}$	-	$5.33 \cdot 10^{-5}$	-
72×72	$8.11 \cdot 10^{-4}$	1.41	$4.56 \cdot 10^{-4}$	2.54	$3.27 \cdot 10^{-5}$	2.69
87×87	$5.63 \cdot 10^{-4}$	1.93	$3.74 \cdot 10^{-4}$	1.05	$2.52 \cdot 10^{-5}$	1.37
104×104	$3.36 \cdot 10^{-4}$	2.89	$1.57 \cdot 10^{-4}$	4.84	$1.58 \cdot 10^{-5}$	2.63
125×125	$3.30 \cdot 10^{-4}$	0.10	$1.32 \cdot 10^{-4}$	0.97	$1.12 \cdot 10^{-5}$	1.85
150×150	$1.81 \cdot 10^{-4}$	3.30	$1.32 \cdot 10^{-4}$	-0.03	$8.07 \cdot 10^{-6}$	1.81
180×180	$1.91 \cdot 10^{-4}$	-0.32	$1.07 \cdot 10^{-4}$	1.16	$5.44 \cdot 10^{-6}$	2.16

Table 5: Test Moving Ellipse: errors and accuracy orders in the L^1 (top) and L^∞ (bottom) norms, for u (left), v (middle) and $\nabla \cdot \mathbf{u}$ (right). The exact solution is (19); the angular velocity is $\omega = 2\pi/5$.
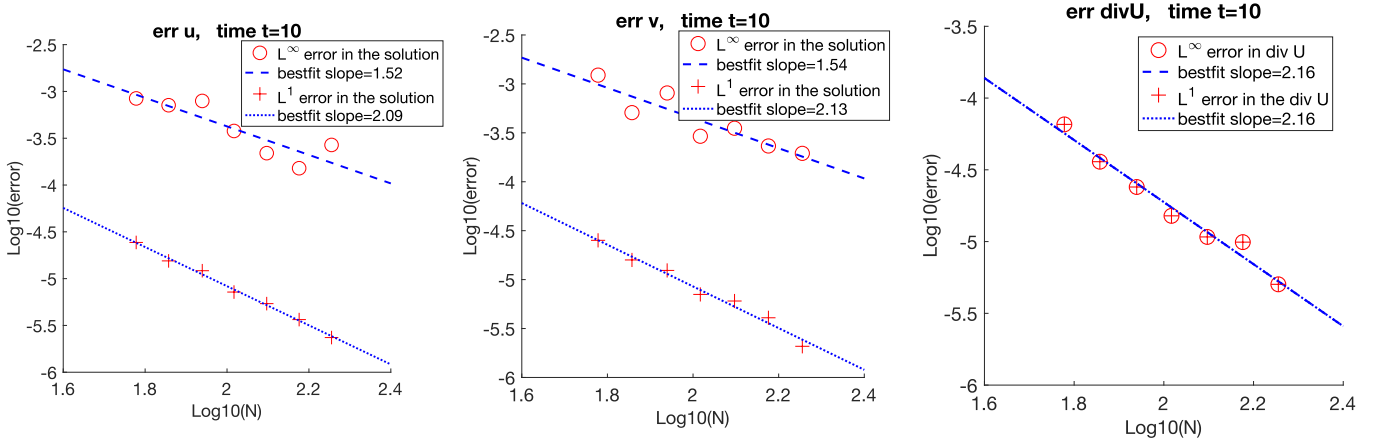

Fig. 12: Test Moving Flower: best fit accuracy order at time $t = 10$. The exact solution is (19); the angular velocity is $\omega = 2\pi/5$.

No. of points	L^1 error of u	order	L^1 error of v	order	L^1 error of $\nabla \cdot \mathbf{u}$	order
60×60	$2.44 \cdot 10^{-5}$	-	$2.52 \cdot 10^{-5}$	-	$6.55 \cdot 10^{-5}$	-
72×72	$1.55 \cdot 10^{-5}$	2.49	$1.59 \cdot 10^{-5}$	2.53	$3.60 \cdot 10^{-5}$	3.28
87×87	$1.22 \cdot 10^{-5}$	1.28	$1.24 \cdot 10^{-5}$	1.30	$2.40 \cdot 10^{-5}$	2.15
104×104	$7.19 \cdot 10^{-6}$	2.94	$7.06 \cdot 10^{-6}$	3.16	$1.51 \cdot 10^{-5}$	2.59
125×125	$5.41 \cdot 10^{-6}$	1.55	$6.04 \cdot 10^{-6}$	0.84	$1.08 \cdot 10^{-5}$	1.84
150×150	$3.65 \cdot 10^{-6}$	2.15	$4.07 \cdot 10^{-6}$	2.16	$9.91 \cdot 10^{-6}$	0.46
180×180	$2.35 \cdot 10^{-6}$	2.43	$2.08 \cdot 10^{-6}$	3.68	$5.04 \cdot 10^{-6}$	3.71
No. of points	L^∞ error of u	order	L^∞ error of v	order	L^∞ error of $\nabla \cdot \mathbf{u}$	order
60×60	$8.42 \cdot 10^{-4}$	-	$1.23 \cdot 10^{-3}$	-	$6.55 \cdot 10^{-5}$	-
72×72	$7.12 \cdot 10^{-4}$	0.92	$5.09 \cdot 10^{-4}$	4.84	$3.60 \cdot 10^{-5}$	3.28
87×87	$7.92 \cdot 10^{-4}$	-0.56	$8.08 \cdot 10^{-4}$	-2.45	$2.40 \cdot 10^{-5}$	2.15
104×104	$3.78 \cdot 10^{-4}$	4.14	$2.92 \cdot 10^{-4}$	5.70	$1.51 \cdot 10^{-5}$	2.59
125×125	$2.19 \cdot 10^{-4}$	2.97	$3.52 \cdot 10^{-4}$	-1.03	$1.08 \cdot 10^{-5}$	1.84
150×150	$1.51 \cdot 10^{-4}$	2.03	$2.32 \cdot 10^{-4}$	2.29	$9.91 \cdot 10^{-6}$	0.46
180×180	$2.68 \cdot 10^{-4}$	-3.14	$1.95 \cdot 10^{-4}$	0.95	$5.04 \cdot 10^{-6}$	3.71

Table 6: Test MOVING FLOWER: errors and accuracy orders in the L^1 (top) and L^∞ (bottom) norms, for u (left), v (middle) and $\nabla \cdot \mathbf{u}$ (right). The exact solution is (19); the angular velocity is $\omega = 2\pi/5$.

5.2 Driven cavity

In this section we perform the driven cavity test around a steady flower-shaped object. No-slip boundary conditions $\mathbf{u} = (0,0)$ are prescribed on the boundary of the object and on the walls of the computational domain $(-1,1)^2$, except for the top wall, that models a lid moving horizontally and at a constant velocity, i.e. $\mathbf{u} = (1,0)$. External forces, such as gravity, are neglected ($\mathbf{f} = 0$).

In Fig. 13 we plot the velocity field at time $t = 10$, $N = 60$ and with four different values of the Reynolds number: Re $= 1, 10, 100$ and 1000. The velocity vectors are scaled in order to have the same length for graphical purposes. Their magnitude can be inferred from the color map.
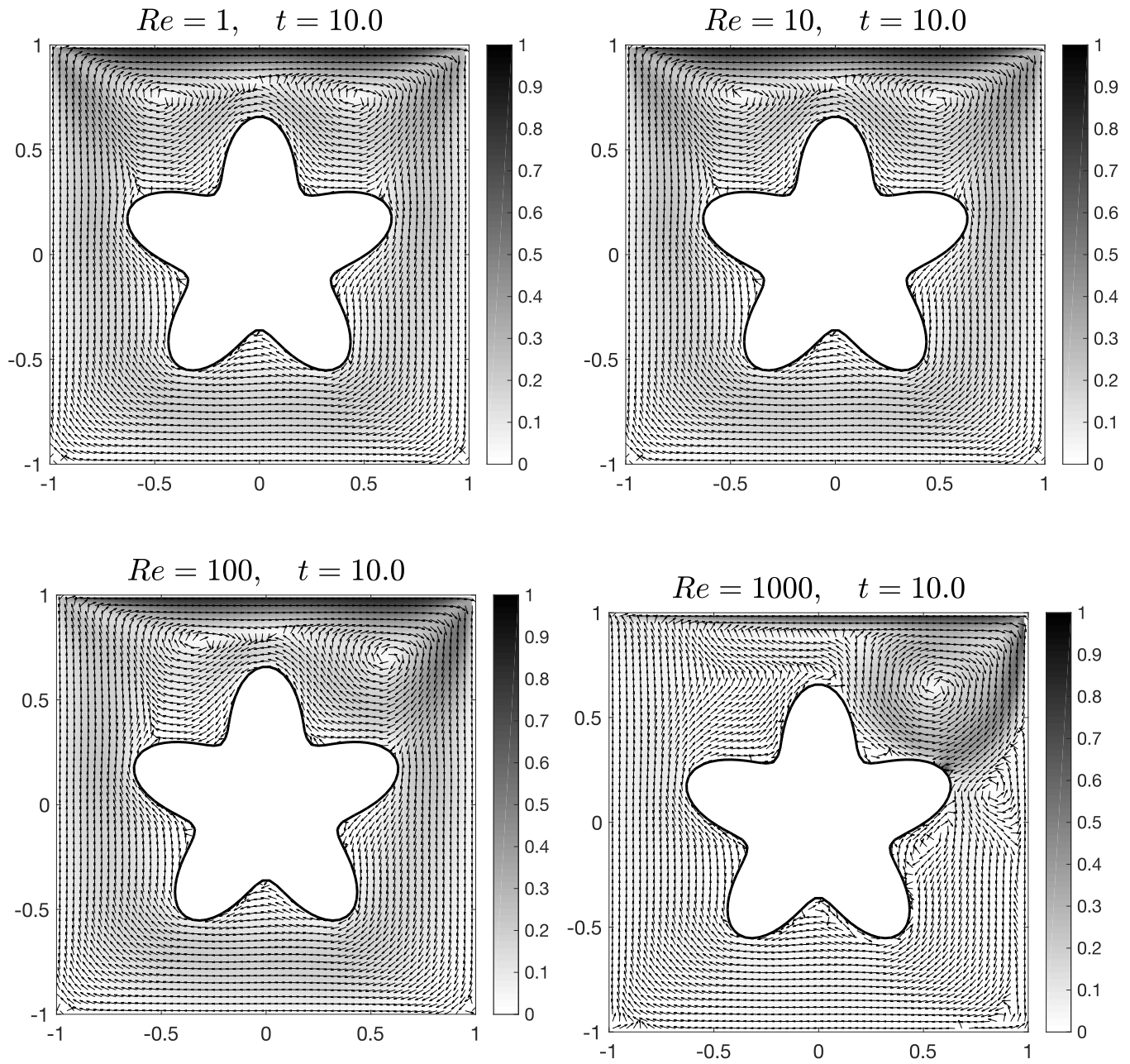
Fig. 13: Velocity field for the driven cavity test with $N = 60$. The velocity vectors are scaled in order to have constant length. Their magnitude can be inferred from the color map.

The presence of the object prevents the formation of anticlockwise vortices in the bottom corners, which are instead observed in classical driven cavity tests (i.e. without objects immersed in the fluid). Moreover, the central large clockwise vortex observed in classical tests is here split into two vortices for low Reynolds numbers Re = 1 and 10, mainly driven by the reentrant boundaries of the flower-shaped object. The x-coordinates of the centres of the vortices are about $x = -0.5$ and $x = 0.5$. For Re = 100, the vortex on the top-left is more eccentric and slightly moved to the right. For Re = 1000 the vortex on the top-left is not observed, while another vortex is forming in the middle-right region.

5.3 Rotating flower-shaped object

The flower shaped object is rotated around the origin at a constant anticlockwise rotation velocity of $\omega = 2\pi/5$. The fluid is initially steady and no-slip boundary conditions on the walls ($\mathbf{u} = \mathbf{0}$) and on the boundary of the object ($(u(x, y), v(x, y), 0) = (0, 0, \omega) \times (x, y, 0)$) are implemented for the entire simulation. Reynolds number is Re = 100. Fig.14 shows the velocity field at nine time steps and $N = 60$. The black dot is plotted on the same petal and it is meant to help the visualisation of the rotation. At time $t = 1.25$ two clockwise vortices occupy a large region of the top-right and bottom-right corners. At $t = 2.50$ another clockwise vortex forms at the top-left corner. At $t = 3.75$ the vortex on the bottom-right corner almost disappears, while at $t = 5.00$ (i.e. after a complete revolution) there is a clockwise vortex in each of the four corners. We observe that at time $t = 5.00$ the vortices on the right occupy a smaller region than at $t = 1.25$. From $t = 5.00$ to $t = 10.00$, i.e. during the second revolution, the velocity field is almost constant in time away from the object, showing that it is approaching a steady-state solution.
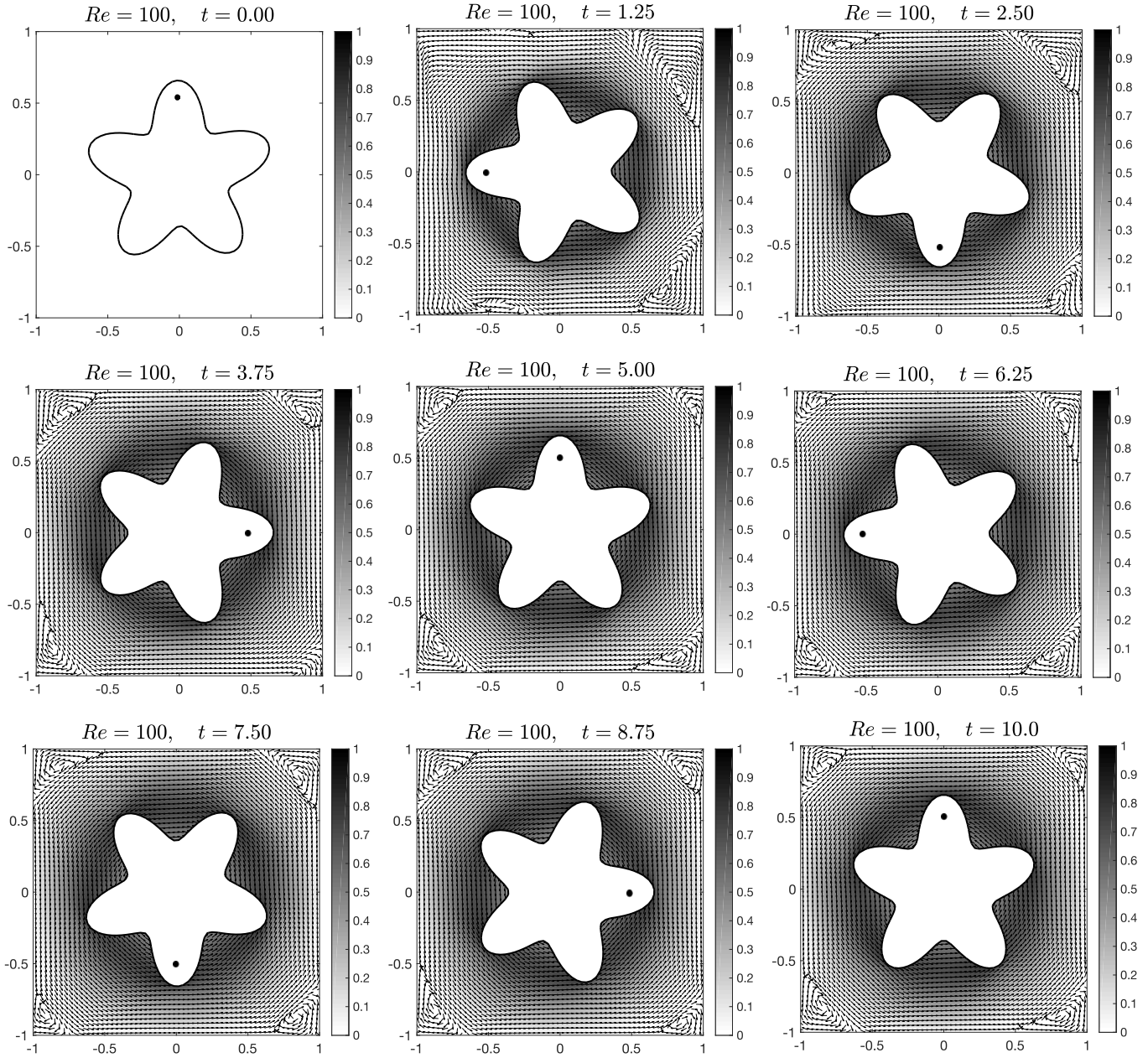
Fig. 14: Velocity field at nine time steps and $N = 60$. The black dot is plotted on the same petal and it is meant to help the visualisation of the rotation. At time $t = 1.25$ two clockwise vortices occupy a large region of the top-right and bottom-right corners. At $t = 2.50$ another clockwise vortex forms at the top-left corner. At $t = 3.75$ the vortex on the bottom-right corner almost disappears, while at $t = 5.00$ (i.e. after a complete revolution) there is a clockwise vortex in each of the four corners. We observe that at time $t = 5.00$ the vortices on the right occupy a smaller region than at $t = 1.25$. From $t = 5.00$ to $t = 10.00$, i.e. during the second revolution, the velocity field is almost constant in time away from the object, showing that it is approaching a steady-state solution.

30

5.3.1 Multigrid efficiency

To investigate the efficiency of the multigrid, we compute the convergence factor

$$\rho = \frac{\left\| \mathbf{r}_h^{(s+1)} \right\|_\infty}{\left\| \mathbf{r}_h^{(s)} \right\|_\infty},$$

where $\mathbf{r}_h^{(s)}$ is the residual after s W-cycles. We use $\nu_1 = 2$ pre-relaxations and $\nu + 1$ post-relaxations. For these values of ν_1 and ν_2, the Local Fourier Analysis predicts an asymptotic convergence factor of $\rho = 0.119$ for standard multigrid applied to elliptic equations in rectangular domains (see [61, Sect. 4]). In Fig. 15 we plot residuals obtained in the most complex problem of this paper, namely the rotating flower-shaped object of Sect. 5.3. We observe that the convergence factor for the tests proposed in this paper is not degraded by the curved boundary and it is in agreement with the expected one $\rho = 0.119$ (dashed line).
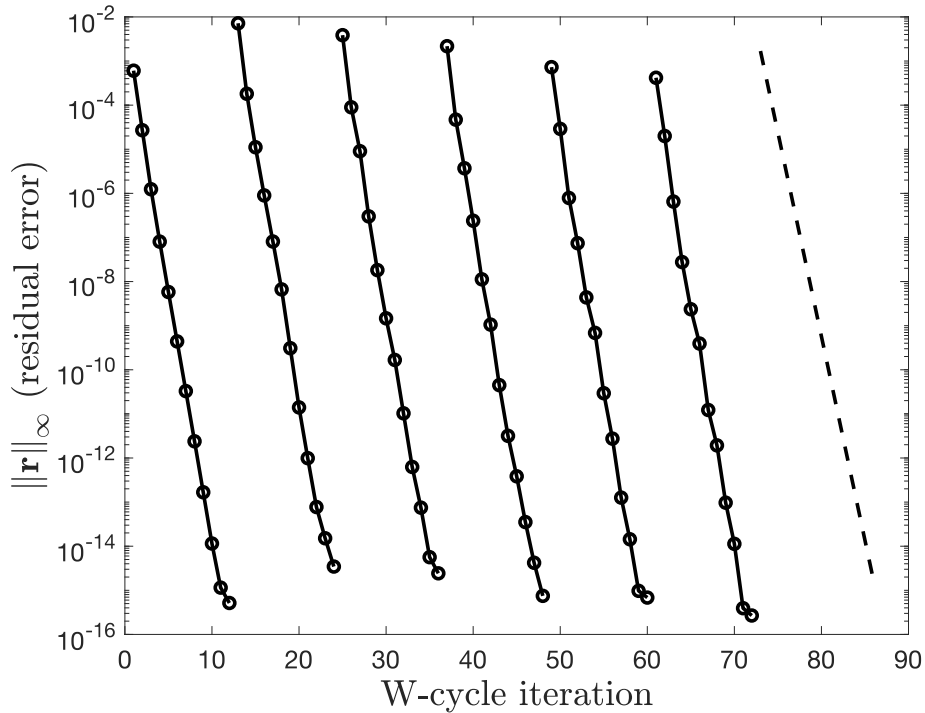

Fig. 15: residual error for each $W-$cycle. Each line refers to a single time step. The dashed line is the reference decay of $\rho = 0.119$ (see Eq. 5.3.1), as expected by the Local Fourier Analysis for elliptic equations in rectangular domains.

6   Conclusion

We have presented a numerical method to solve the Navier-Stokes equations in a moving domain. The equations are discretized by Crank-Nicholson in time and finite differences in space. The moving domain is described by a level-set function and the curved boundaries are discretized by a ghost-point method. The resulting linear system is solved by a multigrid approach properly developed for curved boundaries. The method is second order accurate in space and time. Several tests are provided, varying the Reynolds number and/or the shape of the domain. The method falls under the class of unfitted boundary methods, since the curved boundaries are embedded in a fixed grid. These methods are preferred for moving domain problems, since fitted boundary methods require an additional computational cost for the re-meshing procedures (needed at each time step as the domain moves).

In fact, one of the bottlenecks of computer simulation is the computational time needed to represent complex-shaped objects in 3D. Most existing commercial software used by research centres and industries are based on fitted boundary method approaches, such as FEM. On the other hand, numerical approaches embedded in fixed grids have recently gained an increased interest in scientific computing, as simulation time is drastically reduced, because computationally expensive algorithms to generate the mesh are not required. This aspect is more exasperated if the domain moves and changes shape repetitively. In addition, the increase of computational power of modern massively parallel supercomputers such as exascale computing demands for more efficient numerical methods to solve complex problems with higher accuracy.

The numerical approaches adopted in this paper, namely the ghost-point method on Cartesian meshes and the multigrid solver, are also designed to allow a more natural implementation in parallel computer architectures.

Finally, the method is suitable to model non turbulent viscous flows of incompressible fluids around moving objects, but can be applied in several other contexts where a moving/deforming domain is present, such as fluid-structure interaction or free-boundary problems. The latter is part of the author ongoing efforts.

References

[1] L. Adams and T. P. Chartier. New geometric immersed interface multigrid solvers. *SIAM J. on Sci. Comput.*, 25(5):1516–1533, 2004.

[2] L. Adams and T. P. Chartier. A comparison of algebraic multigrid and geometric immersed interface multigrid methods for interface problems. *SIAM J. on Sci. Comput.*, 26(3):762–784, 2005.

[3] L. Adams and Z. Li. The immersed interface/multigrid methods for interface problems. *SIAM J. on Sci. Comput.*, 24(2):463–479, 2002.

[4] R. E. Alcouffe, A. Brandt, J. E. Dendy, Jr, and J. W. Painter. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. on Sci. Stat. Comput.*, 2(4):430–454, 1981.

[5] P. Angot, C.-H. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numer. Math.*, 81, 1999.

[6] V. Arias, D. Bochkov, and F. Gibou. Poisson equations in irregular domains with Robin boundary conditions—Solver with second-order accurate gradients. *J. Comput. Phys.*, 365:1–6, 2018.

[7] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier–Stokes equations. *J. Comput. Phys.*, 85(2):257–283, 1989.

[8] D. Bochkov and F. Gibou. Solving Poisson-type equations with Robin boundary conditions on piecewise smooth interfaces. *J. Comput. Phys.*, 376:1156–1198, 2019.

[9] D. Bochkov and F. Gibou. Solving Elliptic Interface Problems with Jump Conditions on Cartesian Grids. *J. Comput. Phys.*, page 109269, 2020.

[10] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*, volume 72. Siam, 2000.

[11] D. L. Brown, R. Cortez, and M. L. Minion. Accurate projection methods for the incompressible Navier–Stokes equations. *J. computational physics*, 168(2):464–499, 2001.

[12] F. Chantalat, C.-H. Bruneau, C. Galusinski, and A. Iollo. Level-set, penalization and cartesian meshes: A paradigm for inverse problems and optimal design. *J. Comput. Phys.*, 228:6291–6315, 2009.

[13] A. Chertock, A. Coco, A. Kurganov, and G. Russo. A Second-Order Finite-Difference Method for Compressible Fluids in Domains with Moving Boundaries. *Commun. Comput. Phys.*, 23(1):230–263, 2018.

[14] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *J. computational physics*, 135(2):118–125, 1997.

[15] S. H. Christiansen and K. Hu. Generalized finite element systems for smooth differential forms and Stokes' problem. *Numer. Math.*, 140(2):327–371, 2018.

[16] A. Coco, G. Currenti, J. Gottsmann, G. Russo, and C. Del Negro. A hydro-geophysical simulator for fluid and mechanical processes in volcanic areas. *J. Math. Ind.*, 6(1):6, 2016.

[17] A. Coco, G. Currenti, C. D. Negro, and G. Russo. A Second Order Finite-Difference Ghost-Point Method for Elasticity Problems on unbounded domains with applications to Volcanology. *Commun. Comput. Phys.*, 16:983–1009, 2014.

[18] A. Coco, J. Gottsmann, F. Whitaker, A. Rust, G. Currenti, A. Jasim, and S. Bunney. Numerical models for ground deformation and gravity changes during volcanic unrest: simulating the hydrothermal system dynamics of a restless caldera. *Solid Earth*, 2016.

[19] A. Coco and G. Russo. Second order multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface, I: one dimensional problems. *Numer. Math. Theory, Methods Appl.*, 5(1):19–42, 2012.

[20] A. Coco and G. Russo. Finite-Difference Ghost-Point Multigrid Methods on Cartesian Grids for Elliptic Problems in Arbitrary Domains. *J. Comput. Phys.*, 241:464–501, 2013.

[21] A. Coco and G. Russo. Boundary Treatment in Ghost Point Finite Difference Methods for Compressible Gas Dynamics in Domain with Moving Boundaries. In *Hyperbolic Problems: Theory, Numerics, Applications, Proceedings to HYP2012*, pages 455–465, 2014.

[22] A. Coco and G. Russo. Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface. *J. Comput. Phys.*, 361:299–330, 2018.

[23] R. Costa, J. M. Nóbrega, S. Clain, G. J. Machado, and R. Loubère. Very high-order accurate finite volume scheme for the convection-diffusion equation with general boundary conditions on arbitrary curved boundaries. *Int. J. for Numer. Methods Eng.*, 117(2):188–220, 2019.

[24] S. Court and M. Fournié. A fictitious domain finite element method for simulations of fluid–structure interactions: The Navier–Stokes equations coupled with a moving solid. *J. Fluids Struct.*, 55:398–408, 2015.

[25] A. Danilov, A. Lozovskiy, M. Olshanskii, and Y. Vassilevski. A finite element method for the Navier–Stokes equations in moving domain with application to hemodynamics of the left ventricle. *Russ. J. Numer. Analysis Math. Model.*, 32(4):225–236, 2017.

[26] J. Dendy. Black box multigrid. *J. Comput. Phys.*, 48(3):366–386, 1982.

[27] J. Donea. An arbitrary Lagrangian-Eulerian finite element method for transient fluid-structure interactions. *Comput. Methods Appl. Mech. Eng.*, 33:689–723, 1982.

[28] R. Egan and F. Gibou. xGFM: Recovering convergence of fluxes in the ghost fluid method. *J. Comput. Phys.*, page 109351, 2020.

[29] R. E. Ewing, Z. Li, T. Lin, and Y. Lin. The immersed finite volume element methods for the elliptic interface problems. *Math. Comput. Simul.*, 50(1):63–76, 1999.

[30] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492, 1999.

[31] L. Formaggia and F. Nobile. Stability analysis of second-order time accurate schemes for ALE-FEM. *Comput. Methods Appl. Mech. Eng.*, 193:4097–4116, 2004.

[32] M. J. Gander and M. Neumuller. Analysis of a new space-time parallel multigrid algorithm for parabolic problems. *SIAM J. on Sci. Comput.*, 38(4):A2173–A2208, 2016.

[33] F. Gibou and R. Fedkiw. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comput. Phys.*, 176:205–227, 2002.

[34] F. Gibou and R. Fedkiw. A fourth order accurate discretization for the Laplace and heat equations on arbitary domains, with applications to the Stefan problem. *J. Comput. Phys.*, 202:577–601, 2005.

[35] F. Gibou and C. Min. Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions. *J. Comput. Phys.*, 231(8):3246–3263, 2012.

[36] V. Girault and P.-A. Raviart. *Finite element methods for Navier–Stokes equations: theory and algorithms*, volume 5. Springer Science & Business Media, 2012.

[37] N. Gokhale, N. Nikiforakis, and R. Klein. A dimensionally split Cartesian cut cell method for the compressible Navier–Stokes equations. *J. Comput. Phys.*, 375:1205–1219, 2018.

[38] S. Groí and A. Reusken. An extended pressure finite element space for two-phase incompressible flows with surface tension. *J. Comput. Phys.*, 224(40–58), 2007.

[39] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[40] F. Heimann, C. Engwer, O. Ippisch, and P. Bastian. An unfitted interior penalty discontinuous Galerkin method for incompressible Navier–Stokes two-phase flow. *Int. J. for Numer. Methods Fluids*, 71(3):269–293, 2013.

[41] Á. Helgadóttir, Y. T. Ng, C. Min, and F. Gibou. Imposing mixed Dirichlet–Neumann–Robin boundary conditions in a level-set framework. *Comput. Fluids*, 121:68–80, 2015.

[42] D. Lacanette, S. Vincent, A. Sarthou, P. Malaurent, and J.-P. Caltagirone. An Eulerian/Lagrangian method for the numerical simulation of incompressible convection flows interacting with complex obstacles: Application to the natural convection in the Lascaux cave. *Int. J. Heat Mass Transf.*, 52(11-12):2528–2542, 2009.

[43] H. P. Langtangen, K.-A. Mardal, and R. Winther. Numerical methods for incompressible viscous flow. *Adv. water Resour.*, 25(8-12):1125–1146, 2002.

[44] R. J. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *J. Numer. Analysis*, 31:1019–1044, 1994.

[45] Z. Li and K. Ito. Maximum principle preserving schemes for interface problems with discontinuous coefficients. *SIAM J. on Sci. Comput.*, 23(1):339–361, 2001.

[46] A. Lozovskiy, M. A. Olshanskii, and Y. V. Vassilevski. A quasi-Lagrangian finite element method for the Navier–Stokes equations in a time-dependent domain. *Comput. Methods Appl. Mech. Eng.*, 333:55–73, 2018.

[47] A. Masud and T. J. Hughes. A space-time Galerkin/least-squares finite element formulation of the Navier–Stokes equations for moving domain problems. *Comput. Methods Appl. Mech. Eng.*, 146(1-2):91–126, 1997.

[48] O. Muscato, T. Castiglione, and A. Coco. Hydrodynamic modeling of electron transport in gated silicon nanowires transistors. *Atti della Accademia Peloritana dei Pericolanti-Classe di Scienze Fisiche, Matematiche e Nat.*, 97(S1):18, 2019.

[49] O. Muscato, T. Castiglione, V. Di Stefano, and A. Coco. Low-field electron mobility evaluation in silicon nanowire transistors using an extended hydrodynamic model. *J. Math. Ind.*, 8(1):14, 2018.

[50] Y. T. Ng, C. Min, and F. Gibou. An efficient fluid-solid coupling algorithm for single-phase flows. *J. Comput. Phys.*, 228:8807–8829, 2009.

[51] N. Nikitin. Third-order-accurate semi-implicit Runge–Kutta scheme for incompressible Navier–Stokes equations. *Int. journal for numerical methods fluids*, 51(2):221–233, 2006.

[52] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006.

[53] J. Papac, A. Helgadottir, C. Ratsch, and F. Gibou. A level set approach for diffusion and Stefan-type problems with Robin boundary conditions on quadtree/octree adaptive Cartesian grids. *J. Comput. Phys.*, 233:241–261, 2013.

[54] C. S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[55] D. Peterseim and S. A. Sauter. Finite element methods for the Stokes problem on complicated domains. *Comput. Methods Appl. Mech. Eng.*, 200(33-36):2611–2623, 2011.

[56] A. A. Ruggiu and J. Nordstrom. Multigrid schemes for high order discretizations of hyperbolic problems. In *AIAA Scitech 2019 Forum*, page 0103, 2019.

[57] G. H. Shortley and R. Weller. The numerical solution of Laplace's equation. *J. Appl. Phys.*, 9:334–348, 1938.

[58] R. K. Shukla, M. Tatineni, and X. Zhong. Very high-order compact finite difference schemes on non-uniform grids for incompressible Navier–Stokes equations. *J. Comput. Phys.*, 224(2):1064–1094, 2007.

[59] K. Srinivas and C. Fletcher. *Computational techniques for fluid dynamics: a solutions manual.* Springer Science & Business Media, 2002.

[60] K. Stüben. A review of algebraic multigrid. In *Numerical Analysis: Historical Developments in the 20th Century*, pages 331–359. Elsevier, 2001.

[61] U.Trottemberg, C. Oosterlee, and A. Schuller. *Multigrid.* Academic Press, 2000.

[62] J. Yang, M. Potier-Ferry, and H. Hu. Solving the stationary Navier–Stokes equations by using Taylor meshless method. *Eng. Analysis with Boundary Elem.*, 98:8–16, 2019.

[63] S. Yu, Y. Zhou, and G. Wei. Matched Interface and Boundary (MIB) method for elliptic problems with sharp-edged interfaces. *J. Comput. Phys.*, 224:729–756, 2007.