Coupling of OpenFOAM with a Lagrangian vortex particle method for external aerodynamic simulations

Pasolari, R.; Ferreira, Carlos; van Zuijlen, A.H.

Publication date
2023

Document Version
Final published version

Published in
Physics of Fluids

Citation (APA)
Pasolari, R., Ferreira, C., & van Zuijlen, A. H. (2023). Coupling of OpenFOAM with a Lagrangian vortex particle method for external aerodynamic simulations. *Physics of Fluids*, *35*(10). http://10.1063/5.0165878

RESEARCH ARTICLE | OCTOBER 10 2023

Coupling of OpenFOAM with a Lagrangian vortex particle method for external aerodynamic simulations

R. Pasolari ; C. Ferreira; A. van Zuijlen

Physics of Fluids 35, 107115 (2023)
https://doi.org/10.1063/5.0165878

Physics of Fluids ARTICLE pubs.aip.org/aip/pof

Coupling of OpenFOAM with a Lagrangian vortex particle method for external aerodynamic simulations

Cite as: Phys. Fluids **35**, 107115 (2023); doi: 10.1063/5.0165878
Submitted: 30 June 2023 · Accepted: 22 September 2023 ·
Published Online: 10 October 2023

R. Pasolari,[a] C. Ferreira, and A. van Zuijlen

AFFILIATIONS

Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands

[a]Author to whom correspondence should be addressed: r.pasolari@tudelft.nl

ABSTRACT

In the field of computational aerodynamics, it is vital to develop tools that can accurately, but also efficiently, simulate the flow around bluff objects and calculate the aerodynamic forces acting on them. When strong body–vortex interactions take place, the simulations become more demanding, since complex phenomena appear. To address this issue, hybrid Eulerian–Lagrangian solvers have been developed and are increasingly used in the field. In this paper, a Vortex Particle Method (VPM) is coupled with the OpenFOAM software. The Eulerian solver (OpenFOAM) resolves the regions close to the solid boundaries, while the vortex particles evolve the wake downstream, significantly reducing artificial diffusion. The coupling strategy and the validation results of a hybrid code based on the domain decomposition technique are presented. This work is the first to couple OpenFOAM with a Lagrangian solver in the framework of a hybrid solver. Our objective is twofold: to verify the capability of OpenFOAM to run with a VPM and to validate the hybrid solver using benchmark cases. We demonstrate the validation of the solver on the Lamb–Oseen vortex case, the dipole case in the unbounded domain, and the flow around a cylinder at Re = 550. Our results show that coupling OpenFOAM with a VPM can be achieved without complications and efficiently reproduces the results of pure Eulerian simulations.

I. INTRODUCTION

External aerodynamics is a broad field of engineering primarily governed by advection-dominated flows. Wind turbines, rotors, propellers, helicopters, airplanes, trains, and buildings are among the main applications in this field. The computational study of external aerodynamic problems is a challenging process, especially when strong body-vortex interactions take place, e.g., the case of the flow around vertical axis wind turbines,[1] where the turbine's blades are constantly interacting with the wake. The development of efficient and accurate tools for simulating such flows is essential.

Eulerian solvers are the most popular choice in Computational Fluid Dynamics (CFD). The Eulerian framework entails the computation of fluid properties at specific locations within a given space as the fluid flows through it. There are different methods used in the Eulerian description concerning the discretization technique. Finite differences,[2] finite volumes,[3–5] finite elements,[6,7] and spectral elements[8,9] are the main Eulerian methods used today. In general, these methods have shown great performance when they resolve regions close to solid boundaries including the boundary layer. They are capable of capturing the viscous phenomena and vorticity generation that takes place in the vicinity of the solid structure in a very efficient way, taking advantage of the anisotropy of their elements. However, their dissipative and dispersive nature, especially in regions where the mesh is sparse, is a bottleneck that cannot be easily overcome. Using higher-resolution meshes would increase the computational cost, making the method inefficient and the use of powerful hardware and parallel running essential. Moreover, Eulerian methods impose strict restrictions on the time step because of the advective terms (Courant–Friedrichs–Lewy condition[10]), and it is also very challenging to impose appropriate boundary conditions at the far-field region. This leads to using large computational domains, even when the region of importance is close to the bluff body. One potential approach to address this challenge involves the implementation of Adaptive Mesh Refinement (AMR) techniques, whereby the mesh undergoes dynamic adaptation based on the simulation requirements through localized mesh refinement or coarsening.[11]

Lagrangian solvers have gained in popularity because they can eliminate most of the restrictions that Eulerian solvers pose. In the Lagrangian framework, the observer follows a specific fluid parcel as it travels in space and time, i.e., in the inertial frame of reference. The flow is described by the quantities carried by the particles. One of the most known Lagrangian methods is the Vortex Particle Method (VPM), where in this situation, particles move across the flow field while carrying the vorticity. It has been used in many applications in the field of external aerodynamics, like the one by Pan *et al.*[12] A thorough analysis of VPM can be found in Refs. 13 and 14, while in Ref. 15 a detailed review of the vortex methods is presented. Lagrangian methods have shown many advantages that make them a very popular tool in the field of external aerodynamics. First, the numerical dissipation of this type of Lagrangian methods is much lower than widespread grid-based numerical methods, so that they can more accurately represent the dynamics of the wake. This enables a more efficient simulation of the wake effects in far-field regions, compared to grid-based methods. Second, vortex particles are free to move and thus are adapted to regions with high vorticity, eliminating the need to resolve regions without any physical importance. This adaptability is inherent to the method itself, without the necessity of employing techniques resembling AMR. Moreover, the boundary conditions at infinity are automatically satisfied overcoming the issue of the very large domains needed in Eulerian solvers. Finally, their linear nature allows for acceleration techniques[16–18] which can make the methods extremely fast. Pure VPMs have been used in many applications of external aerodynamics and have shown great performance.[19] However, VPMs have also their bottlenecks. Imposing boundary conditions close to solid boundaries is a challenging process, resulting in a supplementary solver, such as the vortex panel method[20] or the immersed body techniques.[21] Furthermore, they lack efficiency when they need to capture the flow structures close to solid boundaries. Their inability to use anisotropic elements makes the use of a vast number of particles necessary. Finally, due to the strains developed in the flow, the vortex particle configuration loses its structure and this distortion can cause computational inaccuracies.[22]

Considering the advantages and the drawbacks of the Eulerian and Lagrangian methods, numerous research teams have begun to develop solvers that integrate these approaches.[23–31] The application of hybrid Eulerian–Lagrangian methods is gaining traction in the field of external aerodynamics. The two solvers can be coupled in different ways. In the domain decomposition methods introduced by Cottet,[32] the computational domain is divided into Eulerian and Lagrangian parts, in which the Eulerian solver captures near-solid phenomena and the Lagrangian solver resolves the evolution of the wake. In the early stages of this method, an iterative process was employed at every time step for ensuring that the stream functions of the Eulerian and the Lagrangian domains match completely inside an overlap region. Daeninck[33] showed that the two solvers can be coupled without the use of this iterative process. In the vortex particle mesh methods introduced by Christiansen,[34] different equations can be solved on either the mesh or the particles, giving the hybrid solver flexibility. Hybrid solvers have been developed by different research teams and have been applied in many cases.[23–31,35] Shi *et al.*[29] coupled an Eulerian solver with a wake solver by transferring the Eulerian solution to the Lagrangian domain and calculating sectional lift coefficients on the body using the Kutta–Joukowski theorem. In the specific solver, Fluid Structure Interaction (FSI) methods are also present. Stock *et al.*[23] coupled a VPM with OVERFLOW, a fully compressible solver, and later[36] a high-order spectral finite difference method with an open-source VPM. Palha *et al.*[37] coupled the FEM FEniCS software[38] with a VPM. Billuart *et al.*[30] developed a weak coupling approach between a body-fitted velocity-pressure solver and a Vortex Particle-Mesh method in 2D and showed that the total circulation can be conserved without the use of any boundary conditions for the Lagrangian solver. Papadakis and Voutsinas[31] developed a strongly coupled compressible Eulerian–Lagrangian solver and used it for external compressible flows and flows including FSI.[27]

In this paper, a two-dimensional (2D) incompressible hybrid Eulerian–Lagrangian solver is presented, which couples a VPM with a finite volume solver, using the domain decomposition technique. The Eulerian method is implemented into the open-source software OpenFOAM.[39] This software is widely used in many applications including external aerodynamics,[40,41] which is a powerful and accessible tool for research and industry. The flexibility that OpenFOAM provides, allows the development of new tools[42–44] by either modifying existing solvers or coupling them with in-house solvers. The present study builds upon the work of Palha *et al.*,[37] with a primary focus on modifying the Eulerian component. Specifically, the explicit Finite Element Method (FEM) utilized in the work of Palha *et al.*[37] is replaced by an implicit Finite Volume Method (FVM) implementation using the OpenFOAM software. The principal objective of this paper is to verify the feasibility of employing the coupling technique between the Eulerian and Lagrangian solvers proposed by Palha *et al.*[37] within the context of an implicit Eulerian solver, while also validating the performance of the hybrid solver through the utilization of benchmark cases.

The paper is structured with Sec. II describing the component solvers. Subsection II A deals with the Lagrangian part, while Subsection II B describes the Eulerian part (OpenFOAM). Section III demonstrates the coupling techniques that are used here as well as the flow chart of the hybrid solver. Section IV deals with the software and the programming techniques that are employed in order to accelerate the code. Section V discusses the validation of the hybrid solver using two benchmark cases. Finally, Sec. VI presents the conclusions and a brief discussion of the potential and future of the solver.

II. COMPONENT SOLVERS

The hybrid solver developed and presented here consists of a Lagrangian Vortex Particle Method (VPM) and an Eulerian finite volume method, implemented in OpenFOAM. Before discussing the techniques that are used for the coupling (Sec. III), the basic concepts behind the two solvers are discussed. In addition to the discussion on the fundamentals, specific techniques that have been incorporated into the Lagrangian part are discussed here.

In this section, the explanation of the symbols employed is presented in the Nomenclature.

A. Vortex Particle Method (VPM)—Lagrangian solver

In this approach, the observer is in the particle's frame of reference. The equations of motion are solved for each particle separately, instead of considering the fluid as a whole. Depending on the physical quantity of the flow that the particles carry, many different methods arise. In VPMs particles carry the vorticity of the flow. Vorticity is a critical quantity in the field of aerodynamics, closely related to the

aerodynamic forces acting on the solid body that is examined, and this is the reason why VPMs are very often preferred in CFD over other Lagrangian methods.

Lagrangian methods in general, and VPM more specific, show many advantages in the field of external aerodynamics, and especially when strong body–vortex interactions take place. Some of their main advantages are listed as follows:

- They can achieve a significant reduction on the artificial diffusion that mesh-based solvers introduce, even with a lower computational effort.
- Due to the nature of the method, the particles can adapt to regions where vorticity structures exist, without resolving regions that do not contain any information for the flow.
- The boundary conditions in the far-field are automatically satisfied.
- The vorticity carried by the vortex particles stems from the linear solution of a Laplace equation. Hence, this allows to superimpose the solutions of the vortex elements. From a programming perspective, this allows acceleration techniques to be effectively used, such as parallel computations in GPU[18] and Fast Multipole Methods (FMM).[16,17]
- Vortex Particle Method (VPM) allows for the utilization of larger time steps, particularly when employing high-order integration schemes such as the fourth order Runge–Kutta scheme for the solution of the equations of motion.

1. Mathematical description—Governing equations

The most common formulation of the incompressible Navier–Stokes (N–S) equations with constant viscosity in CFD is in terms of velocity and pressure,

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u}. \tag{1}$$

In order to get the relation between velocity and vorticity, we can take the curl of Eq. (1) and end up with the following formulation:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{\omega} = (\boldsymbol{\omega}\cdot\nabla)\boldsymbol{u} + \nu\nabla^2\boldsymbol{\omega} \quad \text{in 3D}, \tag{2a}$$

$$\frac{\partial \omega}{\partial t} + (\boldsymbol{u}\cdot\nabla)\omega = \nu\nabla^2\omega \quad \text{in 2D}. \tag{2b}$$

In the formulation of Eq. (2), the pressure field is absent (the curl of the grad of a scalar field is always zero). This means that there is no need for velocity–pressure coupling techniques like the SIMPLE, PISO, and PIMPLE algorithms in FVM, which often increase the computational cost of the simulations. As we can see in Eq. (2), in two dimensions the vortex stretching term $(\boldsymbol{\omega}\cdot\nabla)\boldsymbol{u}$ is eliminated (all terms are zero), while the vorticity field has only one non-zero component and so it is a scalar. Consequently, the set of 2D incompressible equations can be summarized as

$$\frac{D\omega}{Dt} = \nu\nabla^2\omega \quad \text{N-S equations in 2D}, \tag{3a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{incompressibility constraint}, \tag{3b}$$

$$\nabla\times\boldsymbol{u} = \omega \quad \text{velocity-vorticity relation}, \tag{3c}$$

$$\omega(\boldsymbol{x},t) = \omega_0(\boldsymbol{x}) \quad \text{initial vorticity}, \tag{3d}$$

while their boundary conditions are

$$\lim_{|\boldsymbol{x}|\to\infty} \boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{u}_\infty \quad \text{velocity at infinity}, \tag{4a}$$

$$\lim_{|\boldsymbol{x}|\to\infty} \omega(\boldsymbol{x},t) = 0 \quad \text{vorticity at infinity}. \tag{4b}$$

2. Discretization into vortex elements

The fluid can be discretized into vortex particles, and the set of these particles describes the vorticity field. Given this vorticity field, the induced velocity field can be retrieved by solving the Poisson equation that relates the two fields,

$$-\nabla^2\boldsymbol{u} = \nabla\times\omega. \tag{5}$$

The solution of the Poisson equation is the Green's function,

$$\boldsymbol{u} = -\frac{1}{2\pi}\frac{\boldsymbol{x}-\boldsymbol{x}_p}{|\boldsymbol{x}-\boldsymbol{x}_p|^2}\times\boldsymbol{e}_z. \tag{6}$$

The velocity and vorticity fields are linear solutions, so the corresponding total fields can be written as the linear combination of the contribution of all the particles. The total velocity field is obtained by summing the free-stream velocity with the induced velocity field. Hence, we conclude with the following equation:

$$\boldsymbol{u}(\boldsymbol{x}) = -\sum_p \frac{G_p^\sigma(|\boldsymbol{x}-\boldsymbol{x}_p|)}{2\pi}\frac{\boldsymbol{x}-\boldsymbol{x}_p}{|\boldsymbol{x}-\boldsymbol{x}_p|^2}\times\boldsymbol{e}_z + \boldsymbol{u}_\infty, \tag{7}$$

where G_p^σ is the kernel of the particle (e.g., a Diral delta distribution). In order to get a smooth representation of the vorticity field, instead of a spurious one that the Dirac distributions produces, and to avoid the singularity that comes from the particles, we can use mollified kernels giving the particles a finite core to end up with smooth induced velocity and vorticity fields. The smoothing function is selected to ensure the conservation of the total circulation. The total fields can be written as

$$\boldsymbol{u}_p(\boldsymbol{x}) = -\sum_p \frac{g_\sigma(|\boldsymbol{x}-\boldsymbol{x}_p|)}{|\boldsymbol{x}-\boldsymbol{x}_p|^2}(\boldsymbol{x}-\boldsymbol{x}_p)\times\boldsymbol{e}_z\Gamma_p + \boldsymbol{u}_\infty, \tag{8a}$$

$$\omega_p(\boldsymbol{x}) = \sum_p \zeta_\sigma(|\boldsymbol{x}-\boldsymbol{x}_p|)\Gamma_p. \tag{8b}$$

Here, we use the Gaussian kernel as the smoothing function and so, the participating functions in Eq. (8) are

$$g_\sigma(r) = \frac{1}{2\pi\sigma^2}\left(1 - e^{-\frac{r^2}{2\sigma^2}}\right), \tag{9a}$$

$$\zeta_\sigma(r) = \frac{1}{2\pi\sigma^2}e^{-\frac{r^2}{2\sigma^2}}. \tag{9b}$$

3. Evolution of the vortex particles

The VPM is mainly used for the description of incompressible, inviscid flows. Nevertheless, in order to make the model more realistic and capable of solving problems with greater accuracy, it is vital to take the viscous effect into consideration. This means that the processes that take place during the evolution of the flow are convection and diffusion. Chorin[45] proposed a viscous splitting algorithm where the two processes are decoupled and solved sequentially (first the

particles are convected and then are diffused). Equation (10) summarizes the equations that are going to be solved for each process,

$$\frac{\partial \omega}{\partial t} + \boldsymbol{u} \cdot \nabla \omega = 0 \quad \text{convection step}, \tag{10a}$$

$$\frac{\partial \omega}{\partial t} - \nu \nabla^2 \omega = 0 \quad \text{diffusion step}. \tag{10b}$$

a. Convection step. The first step of the splitting algorithm is the convection, which can be modeled by the following system of ordinary differential equations (ODEs):

$$\frac{d\boldsymbol{x_p}}{dt} = \boldsymbol{u}(\boldsymbol{x_p}), \tag{11a}$$

$$\frac{d\Gamma_p}{dt} = 0. \tag{11b}$$

Equation (11) shows that in the convection step, the blobs are moving due to the entire flow field, while their strengths remain unchanged. We solve Eq. (11a) with the fourth order Runge–Kutta integration scheme to introduce stability to the scheme and allow greater time steps. In this way, most of the artificial diffusion introduced by the inaccurate computation of the particle's trajectory can be eliminated.

b. Diffusion step. The second step of the splitting algorithm is diffusion. The diffusion problem can also be written in the form of two ODEs,

$$\frac{d\boldsymbol{x_p}}{dt} = 0, \tag{12a}$$

$$\frac{d\omega_p}{dt} = \nu \Delta \omega_p. \tag{12b}$$

There are different techniques that one can employ to treat this step. A brief description of these techniques can be found in the review of Mimeau and Mortazavi.[15] In our solver, we have implemented the following:

- **Core spreading model**
  In this model, we use the solution of the ODE [Eq. 12(b)]. The vorticity field of the diffused particle evolving in time can be written as

$$\boldsymbol{\omega}(\boldsymbol{x}, t) = -\frac{\Gamma_p}{4\pi\nu t} \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{x}_p|^2}{4\nu t}\right). \tag{13}$$

  This equation is the same with the vorticity field of a finite core size particle if we define the core radius as $\sigma^2 = 2\nu t$. This means that the particle expands its core as time passes simulating the diffusion process (Fig. 1). This method is computationally efficient as it does not introduce new particles in the simulation. However, in order to get a smooth and accurate representation of the vorticity field, we need particles with a small core size. In the specific method, the existence of particles with wide core is inevitable. That is the reason why we do not rely solely on this diffusion model; instead, we combine it with other diffusion methods, such as the vortex redistribution method described below. To be more specific, the core spreading model comes into play in regions where an extremely precise flow description is not necessary, particularly in distant areas downstream from the solid body when simulating the aerodynamics of bluff bodies.
- **Vortex particle redistribution**
  In the redistribution method, the participating particles exchange strengths (circulation) with their neighbors, simulating the diffusion process in this way. Here, we have employed the method proposed by Tutty.[46] During this process, the particles are distributed in a fixed grid, having the same core size, and a third-order accurate B-spline kernel is used to redistribute their strengths.

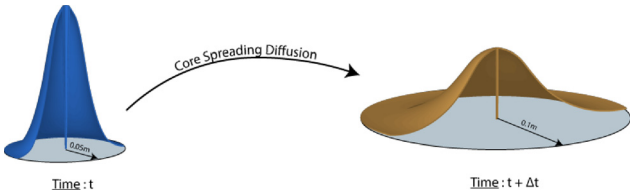

FIG. 1. Core spreading diffusion mechanism. The particle expands its core as time passes diffusing the vorticity field in this way.

B. OpenFOAM—Eulerian solver

In the hybrid framework of this project, we employ OpenFOAM[39] as the Eulerian part. OpenFOAM is a finite-volume cell-centered open-source software. The open-source nature of the software allows the implementation of new solvers as well as modifications of existing ones, making OpenFOAM very popular in research and academia. The implementation of the finite volume method in OpenFOAM has been extensively covered in bibliography.[47]

As the basis for our solver, pimpleFOAM is used, an inherent OpenFOAM solver which is designed to solve transient, incompressible, and turbulent (can also run for laminar) flows. It uses the PIMPLE loop for the correction of the velocity and pressure fields and has been extensively used in many applications.[48,49]

III. HYBRID SOLVER

Hybrid solvers exploit the advantages of the Eulerian and Lagrangian solvers, using the former for capturing the near-solid phenomena and the latter for the evolution of the wake. Here, we present the way that we coupled the Eulerian solver implemented in OpenFOAM, with the VPM solver developed by our research team.[37]

A. Decomposition technique

As it was already mentioned in Sec. I, there is no unique way to couple the solvers. Here, the hybrid solver we present, employs the domain decomposition technique in the way proposed by Daeninck[33] and applied by Palha *et al.*[37] The key difference between the code we present here and the code in Ref. 37 is the Eulerian part. Palha *et al.*[37] uses an explicit Finite Element solver as the Eulerian part of the hybrid solver, while we use the implicit Finite Volume Method implemented in the open-source OpenFOAM software.

The decomposition of the computational domain is illustrated in Fig. 2. OpenFOAM resolves the region close to the solid body up to the numerical boundary (Fig. 2), while VPM solves the equations of

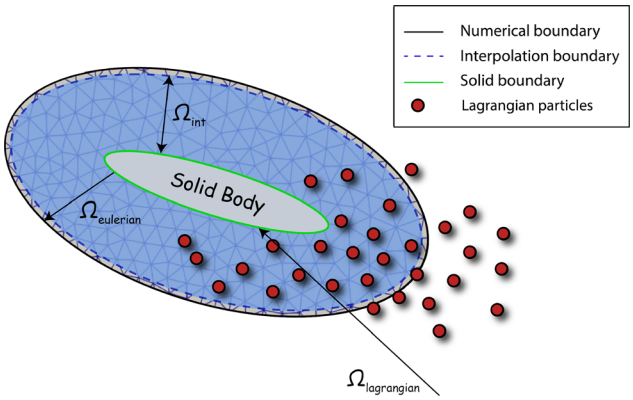

FIG. 2. Decomposition of the computational domain. The Eulerian mesh extends up to a short distance (numerical boundary) away from the solid boundary, whereas the Lagrangian solver covers the entire computational domain.

motion for the entire domain. It is obvious that the Lagrangian domain completely overlaps the Eulerian subdomain.

B. Coupling strategy

A two-way coupling between VPM and OpenFOAM is employed (Fig. 3). As one can see in Fig. 2, the Eulerian domain is very narrow compared to the domain that would be used in a pure Eulerian simulation. In pure Eulerian simulations, the computational domain extends up to a large distance away from the solid body to impose boundary conditions assuming a fully developed flow. However, in this case, it is obvious that the flow is not developed when it reaches the outer Eulerian boundary (numerical boundary). Hence, it is vital to find a way to correctly impose boundary conditions. This is done by using the evolved Lagrangian particles, which will be covered in the next paragraph. This is the first step of the two-way coupling. The second step of the coupling concerns the Lagrangian solution. At a given time step t_n, we have the solution for the Eulerian and the Lagrangian fields. Nevertheless, we know that the Lagrangian solution close to the solid body is not accurate, because of the weakness of the Lagrangian solver to resolve the boundary layer effects. So, it would be wrong to evolve the particles using this solution. To overcome this problem, we correct the Lagrangian solution using the more accurate Eulerian solution in the interpolation region $\mathbf{\Omega_{int}}$ (Fig. 2). In this way, a more accurate circulation is assigned to the particles before progressing to the next time step. It should be noted that these coupling steps are performed once in each time step (weak coupling); no iterative process between the two solvers is present.

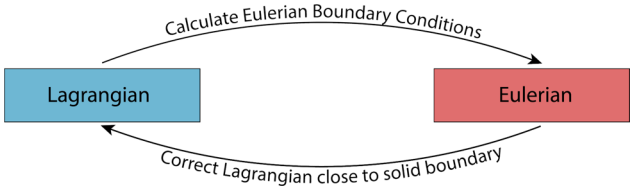

FIG. 3. The two steps of the coupling between the Eulerian and the Lagrangian solver. First, the evolved Lagrangian solution defines the boundary conditions for the numerical boundary of the Eulerian, and second, the more accurate Eulerian solution is used for updating the Lagrangian one in the interpolation region. The coupling steps are performed once in each time step (weak coupling).

1. Calculate Eulerian boundary conditions

In OpenFOAM, we need to specify velocity and pressure boundary conditions when we use pimpleFOAM without turbulence. Hence, the evolved Lagrangian particles need to calculate boundary conditions at the outer Eulerian domain for the velocity and the pressure field. The boundary conditions used here are as follows:

- Dirichlet boundary conditions for the velocity across the boundary ($u_{n,f}$).
- Neumann boundary conditions for the pressure across the boundary ($\partial p/\partial n$).

The velocity at the faces of the numerical boundary is calculated using Eq. (7), while the pressure gradient is obtained from the unsteady Bernoulli equation [Eq. (14)]. Knowing the induced velocity from all the particles, all the terms in the right-hand side (RHS) can be calculated in order to obtain the pressure gradient. These values are assigned at the center of the outer faces as it is shown in Fig. 4,

$$\nabla \bar{p} = -\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nu\nabla^2\boldsymbol{u}\right), \quad \bar{p} = p/\rho. \tag{14}$$

As we have already discussed, the Lagrangian solver has no strict limit for the time step. Typically, different time steps are used for the Eulerian and the Lagrangian solvers. Specifically, a Lagrangian time step can be performed in k_E number of sub-steps using $\Delta t_E = \frac{\Delta t_L}{k_E}$, where Δt_E and Δt_L are the Eulerian and the Lagrangian time steps, respectively. In this case, the boundary conditions for every sub-step (t_{int}) can be obtained by interpolating between the times t_n and t_{n+1},

$$u_{\text{boundary}}(t_{\text{int}}) = u_{\text{boundary}}(t_n) + \frac{t_{int} - t_n}{\Delta t_L}\left[u_{\text{boundary}}(t_{n+1}) - u_{\text{boundary}}(t_n)\right]. \tag{15}$$

2. Correction of the Lagrangian field

The second step of the coupling between the Lagrangian and the Eulerian solver is the correction of the Lagrangian solution close to the solid body. The Lagrangian solver under-resolves the regions close to

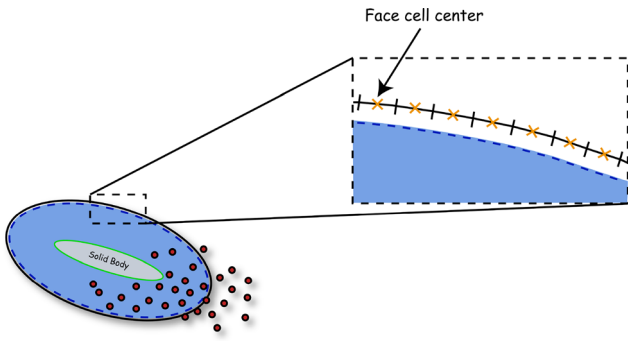

FIG. 4. Calculation of boundary conditions for the numerical Eulerian boundary, using the evolved Lagrangian solution.

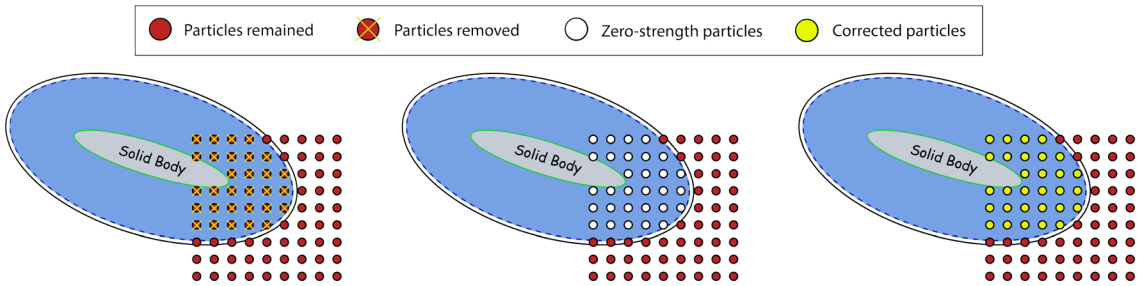

FIG. 5. Correction of the Lagrangian field inside the interpolation region. First, the particles that are located inside this region are identified and removed. New particles with zero strength are generated on the grid (the same grid that particles are redistributed to), and their corrected strength is computed by projecting the Eulerian vorticity field.

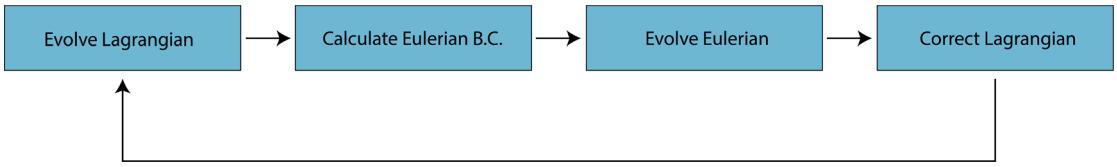

FIG. 6. The flow chart of the hybrid solver.

the aerodynamic body, therefore introducing errors in the calculation of the Eulerian boundary conditions at the external (numerical) boundary. In order to overcome this issue, we use the solution of the Eulerian field in this region at the same time level to correct the particles' strengths. The correction takes place only when the Eulerian solver time coincides with the Lagrangian time instant. At the end of this correction step, the solutions of the two solvers match. The region where we correct the particles ($\boldsymbol{\Omega_{int}}$ in Fig. 2) is the Eulerian subdomain, but excluding a thin layer of cells (around 2–3 cells) close to the numerical boundary (Fig. 2). This area is excluded because an accurate representation of the velocity gradients would not be possible, since there is a boundary there, and so any errors in the computation of the vorticity would be magnified.[31] The idea behind this method is that we remove all the particles located in the interpolation region, and we create new particles that can reproduce the Eulerian vorticity field. The steps of this process can be seen in Fig. 5 and are described as follows:

1. The particles residing within the interpolation region are identified through the utilization of Python's *matplotlib.path* module *contains_points* function.
2. These particles are removed.
3. New particles are created on the background Lagrangian grid. This is the same grid that particles are redistributed to.
4. The vorticity field from the Eulerian solution is exported.
5. The vorticity is interpolated from the cell centers to the particles using the interpolation method of SciPy.[50] The order of the interpolation method is used-defined, and so different methods can be used in each case. In the cases that are presented in this paper, the linear method is used.
6. The strength of the new particles is calculated using the formula $\alpha_p = \omega_p \cdot h^2$, where h is the nominal separation between the particles, ω_p is the vorticity at the point where the particle is created, and α_p is the assigned circulation to the particle.

C. Flowchart

The hybrid solver runs in a loop, where each iteration includes the two coupling steps as well as the evolution of the Lagrangian and Eulerian solvers. Given the Lagrangian and Eulerian solutions at an arbitrary time t_n, the algorithm to proceed to time t_{n+1} is presented in the following, summarized in the flow chart of Fig. 6.

1. Evolve the Lagrangian solution for a Lagrangian time step Δt_L.
2. Calculate the Eulerian boundary conditions: Knowing the evolved solution of the Lagrangian field, we can calculate and impose the boundary conditions for the numerical boundary.
3. Evolve the Eulerian solution for Δt_L: The Eulerian part being much more sensitive to artificial diffusion and instabilities is evolved in k_E time steps, where

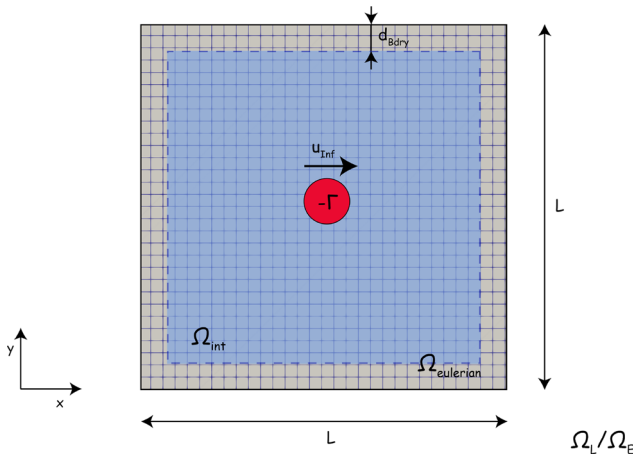

FIG. 7. The geometry of a Lamb–Oseen vortex, initially located at the center of a square domain and convected by a velocity u_{inf}.

TABLE I. Simulation parameters for the case of a traveling Lamb–Oseen vortex in the unbounded domain.

Parameter	Symbol	Value	Dimension
Particle strength	Γ_c	−0.5	m^2/s
Domain edge length	L	1.0	m
Initial position	$\boldsymbol{x}$	$[0.0\,0.0]^T$	m
Freestream velocity	$\boldsymbol{u}_{\text{inf}}$	$[1.0\,0.0]^T$	m/s
Lamb–Oseen time constant	τ	4.0	s
Kinematic viscosity	ν	5×10^{-4}	kg/(m·s)
Diffusion and convection time step	$\Delta t_c = \Delta t_d$	0.01	s
Overlap ratio	λ	1	–
Interpolation domain offset from Eulerian boundary	d_{bdry}	$3 \cdot h$	m
Vortex particles spacing	h	0.008	m
Gaussian kernel width spreading	k	2	–
Population control thresholds	$(\Gamma_{loc}, \Gamma_{glob})$	$(10^{-14}, 10^{-14})$	–
Eulerian mesh density	N_{cells}	320×320	–

$$\Delta t_L = k_E \cdot \Delta t_E.$$

4. Correct the Lagrangian solution in the interpolation region: The Eulerian solution is more accurate close to the solid boundary and so it is used for updating the Lagrangian inside the interpolation region.

IV. SOFTWARE AND ACCELERATION TECHNIQUES

The aim for the hybrid solver is to efficiently simulate flows in the field of external aerodynamics. In order to make the solver more efficient, we can employ techniques both on the programming and the modeling level.

A. Modeling techniques

One of the main problems of particle methods is that during the simulation time, the number of the computational elements increases rapidly, slowing down the Lagrangian part of the hybrid solver. In the simulations of flows around solid bodies, there is a high concentration of elements in the wake, where most of them carry a low amount of circulation.

1. Population control

The population control is a process that is performed under a frequency f_{pc}, and it removes the particles that carry a negligible amount

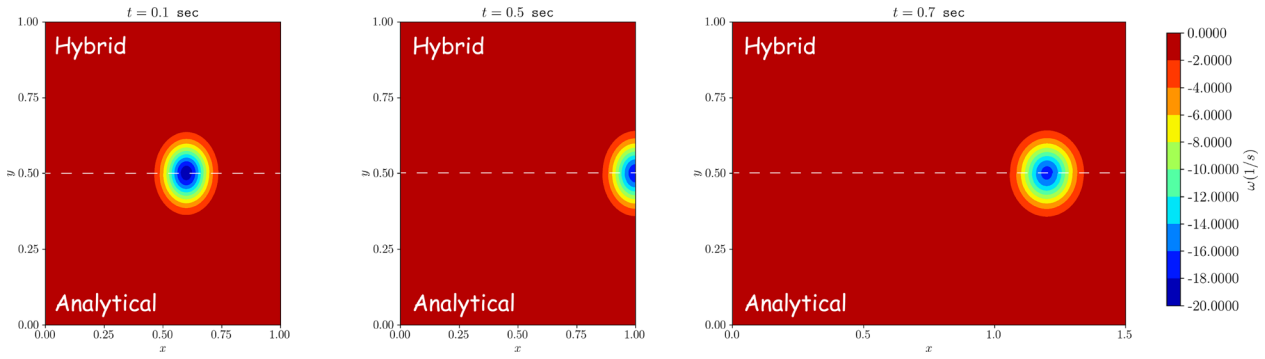

FIG. 8. The vorticity field for the hybrid (upper part) and the analytical (lower part) solutions in three different time instances ($t = 0.1$, $t = 0.5$, and $t = 0.7$ s), for the case of a Lamb–Oseen vortex, initially located at the center of a square domain, and convected by a freestream velocity u_{inf}.

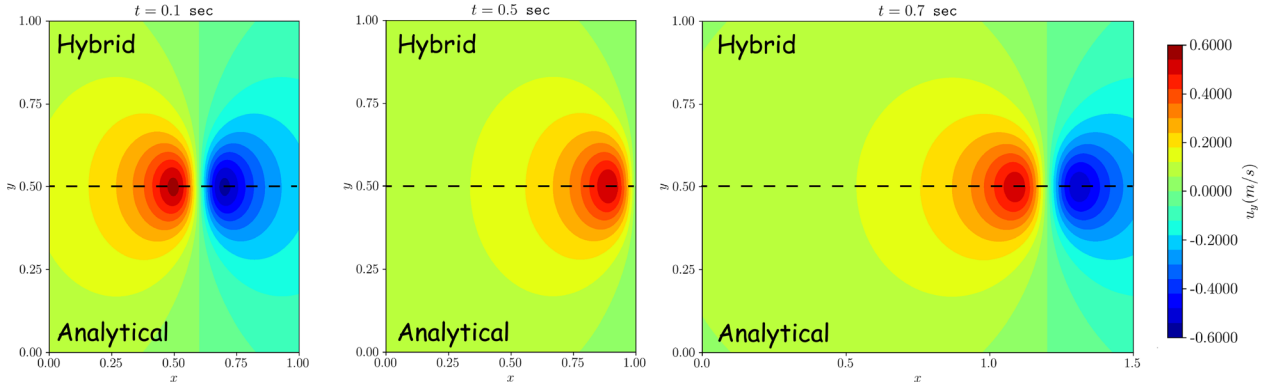

FIG. 9. The y-component of the velocity field for the hybrid (upper part) and the analytical (lower part) solutions in three different time instances ($t = 0.1$, $t = 0.5$, and $t = 0.7$ s), for the case of a Lamb–Oseen vortex, initially located at the center of a square domain, and convected by a freestream velocity u_{inf}.

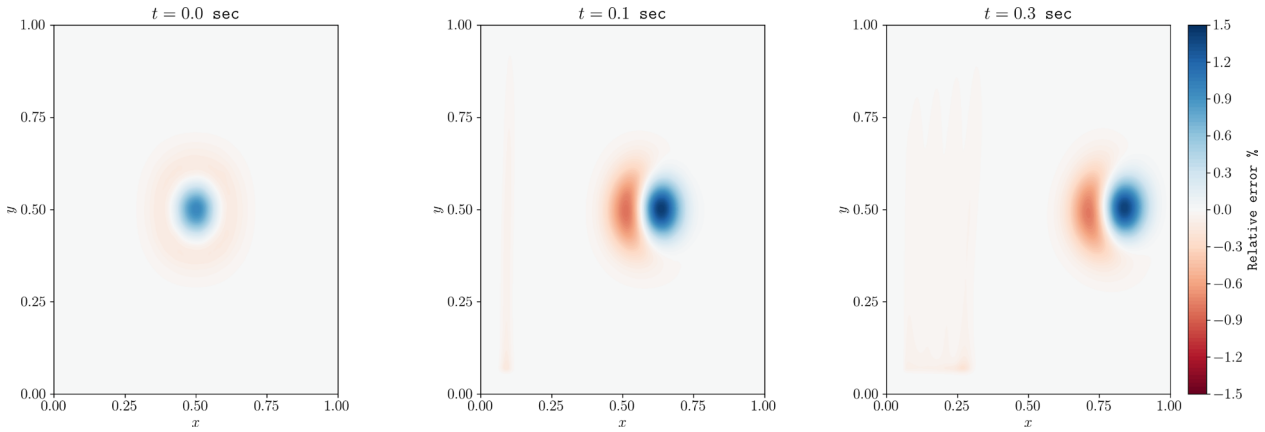

FIG. 10. The relative vorticity error in three different time instances ($t = 0.0$, $t = 0.1$, and $t = 0.3$ s), for the case of a Lamb–Oseen vortex, initially located at the center of a square domain, and convected by a freestream velocity u_{inf}.

of circulation. There are two parameters to be defined here—the local (Γ_{loc}) and the global (Γ_{glob}) circulation thresholds. The particles with strength lower than (Γ_{loc}) are denoted as flagged particles. If their sum is lower that the global threshold, these particles are removed. Otherwise, the local threshold decreases to 10% of the initial value and we repeat the process.

2. Confine the wake

The Lagrangian particles are evolved in the wake extending continuously in the computational domain. In most applications, the information is important up to a specific distance downstream, and so the particles that exceed this point can be removed. However, Stock and Gharakhani[36] mention that this arbitrary removal of particles can cause a bounce of the vorticity field at the interface. Therefore, it is proposed to gradually weaken these particles until they reach a specified circulation threshold so we can remove them. Here, this recommendation has not been employed yet, and so the confinement distance is kept far from the examined body, in order to ensure that this arbitrary removal does not affect the aerodynamic behavior.

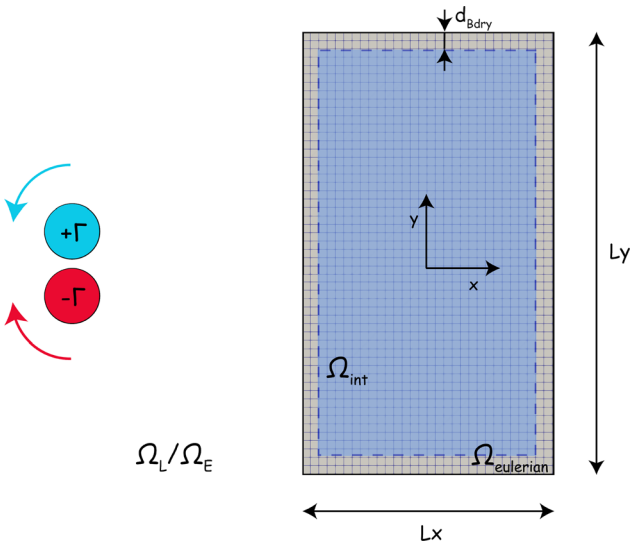

FIG. 11. The geometry of a vortex dipole, initially located out of the Eulerian subdomain. The Eulerian and the Lagrangian domains, as well as the interpolation region, are also illustrated.

B. Software

The main part of the solver has been developed in Python 3.9.[51] However, due to the fact that Python is a high-level language, we

TABLE II. Simulation parameters for the case of a dipole in the unbounded domain.

Parameter	Symbol	Value	Dimension
Kinematic viscosity	ν	1.6×10^{-3}	kg/(m·s)
Diffusion and convection time step	$\Delta t_c = \Delta t_d$	2.5×10^{-4}	s
Overlap ratio	λ	1	···
Interpolation domain offset from Eulerian boundary	d_{bdry}	$5 \cdot h$	m
Vortex particles spacing	h	5×10^{-3}	m
Gaussian kernel width spreading	k	2	···
Population control thresholds	$(\Gamma_{loc}, \Gamma_{glob})$	$(10^{-14}, 10^{-14})$	···
Domain edge x length	L_x	0.5	m
Domain edge y length	L_y	1.0	m
Eulerian domain	Ω_E	$[-0.25, 0.25] \times [-0.5, 0.5]$	···
Eulerian mesh density	N_{cells}	150×300	···
Eulerian time step	Δt_E	2.5×10^{-4}	s

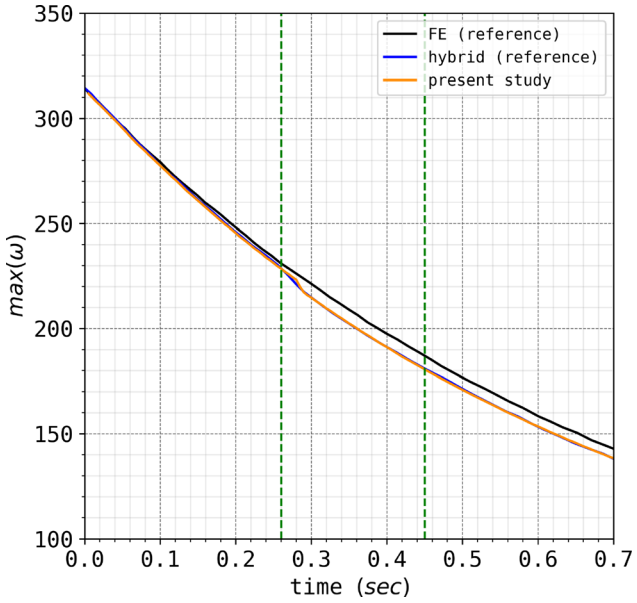

FIG. 12. The maximum vorticity in the case of the traveling dipole in the unbounded domain, in the time interval $t = 0$ to $t = 0.7$ s. The result of the present study is compared to the finite element simulation and the hybrid simulation provided by Palha *et al.*[37]

combined it with lower-level programming languages to improve the performance of specific parts of the code. All the calculations of the induced fields for the particles can run in serial and in parallel in CPU and GPU. For the CPU calculations, we have developed codes in C,[52] C++,[53] and Cython[54] and parallelized the codes using OpenMP.[55] For the GPU calculations, cuda[56] codes have been developed.

Moreover, a Fast Multipole Method (FMM) code, developed by Goude and Engblom,[16] has been incorporated into the solver to obtain induced velocities. FMM is a numerical algorithm based on a hierarchical tree structure which efficiently computes particle interactions. It can reduce the number of calculations from N^2 to $N \log(N)$, where N is the number of particles.

V. VALIDATION

The performance of the solver is validated through the use of three benchmark cases:

1. Traveling Lamb–Oseen vortex in the unbounded domain: This case allows for the verification of the coupling procedure between the Eulerian and Lagrangian solvers and also tests the ability of the solver to accurately propagate information out of the Eulerian subdomain. Additionally, it serves as a validation for the pure Lagrangian solver in a mixed convection–diffusion problem.
2. Dipole in the unbounded domain: In this scenario, another case in an unbounded domain is encountered where the dipole traverses the Eulerian domain without the influence of a freestream velocity. This particular test case offers valuable insight since it allows for a comparison with the research conducted by Palha *et al.*[37]
3. Flow around a cylinder at low Reynolds number (Re = **550**): This case is a commonly used benchmark for new solvers and allowed for a more demanding real-life case validation. The presence of a solid boundary can validate that the vorticity captured by the Eulerian solver is transferred correctly to the Lagrangian solver. The focus of this case is primarily on the quantitative validation of the aerodynamic coefficients and the Strouhal number.

It is essential to note that all the cases presented in this context were executed on GPUs and employed the FMM algorithms.

A. Traveling Lamb–Oseen vortex in the unbounded domain

The Lamb–Oseen vortex[57] is the case of a vortex with a finite core that is diffused in space and time. The analytical solutions for the velocity and the vorticity field induced by the vortex are presented in the following equation:

$$u_\theta = \frac{\Gamma_c}{2\pi r}\left[1.0 - \exp\left(-\frac{r^2}{4\nu(t+\tau)}\right)\right], \quad u_r = 0, \tag{16a}$$

$$\omega = \frac{\Gamma_c}{4\pi\nu(t+\tau)}\exp\left(-\frac{r^2}{4\nu(t+\tau)}\right). \tag{16b}$$

u_θ is the circumferential velocity, u_r is the radial velocity, and ω is the vorticity. Γ_c is the strength of the vortex, t is the simulation time, τ is the time constant (for smooth distribution of the vorticity field), ν is the kinematic viscosity, and r is the distance from the core center.

Here, a Lamb–Oseen vortex with strength $-\Gamma$ is initially located at the center of a square domain with length L. It is convected with the freestream velocity U_{inf} as it is illustrated in Fig. 7.

The simulation was executed until the vortex traversed totally out of the Eulerian boundary, which is depicted as the numerical boundary in Fig. 2. Table I provides a summary of the problem parameters, including geometry, and the simulation parameters.

Figures 8 and 9 display the vorticity and y-component of the velocity field, respectively, for the hybrid and analytical solutions at three distinct time points. The upper portion of the contour plots represents the hybrid solution, while the analytical solution is depicted in the lower portion. The two solutions exhibit perfect agreement at their interface, and the transition between them occurs smoothly without any irregularities. This is evident even when the vortex approaches and finally exits the Eulerian domain. Figure 10 illustrates the relative error of the vorticity field in the three different time instances. The error presented in the figure is calculated as

$$\omega_{\text{error,rel}} = \frac{\omega_{\text{hybrid}} - \omega_{\text{analytical}}}{|\omega_{\text{analytical,max}}|}.$$

The initial observation reveals a discretization error at the center of the initial vortex, approximately around 1%. As the vortex moves, the primary error remains concentrated at its core, gradually increasing to about 1.5%. Additionally, a minor error, less than 0.3%, emerges at the left edge of the Eulerian domain, but it remains stable throughout the simulation without magnifying further.

Provided these results, it can be stated that the coupling process is accurate. The Lagrangian particles are capable of computing the boundary conditions for the Eulerian subdomain, and the Eulerian solution

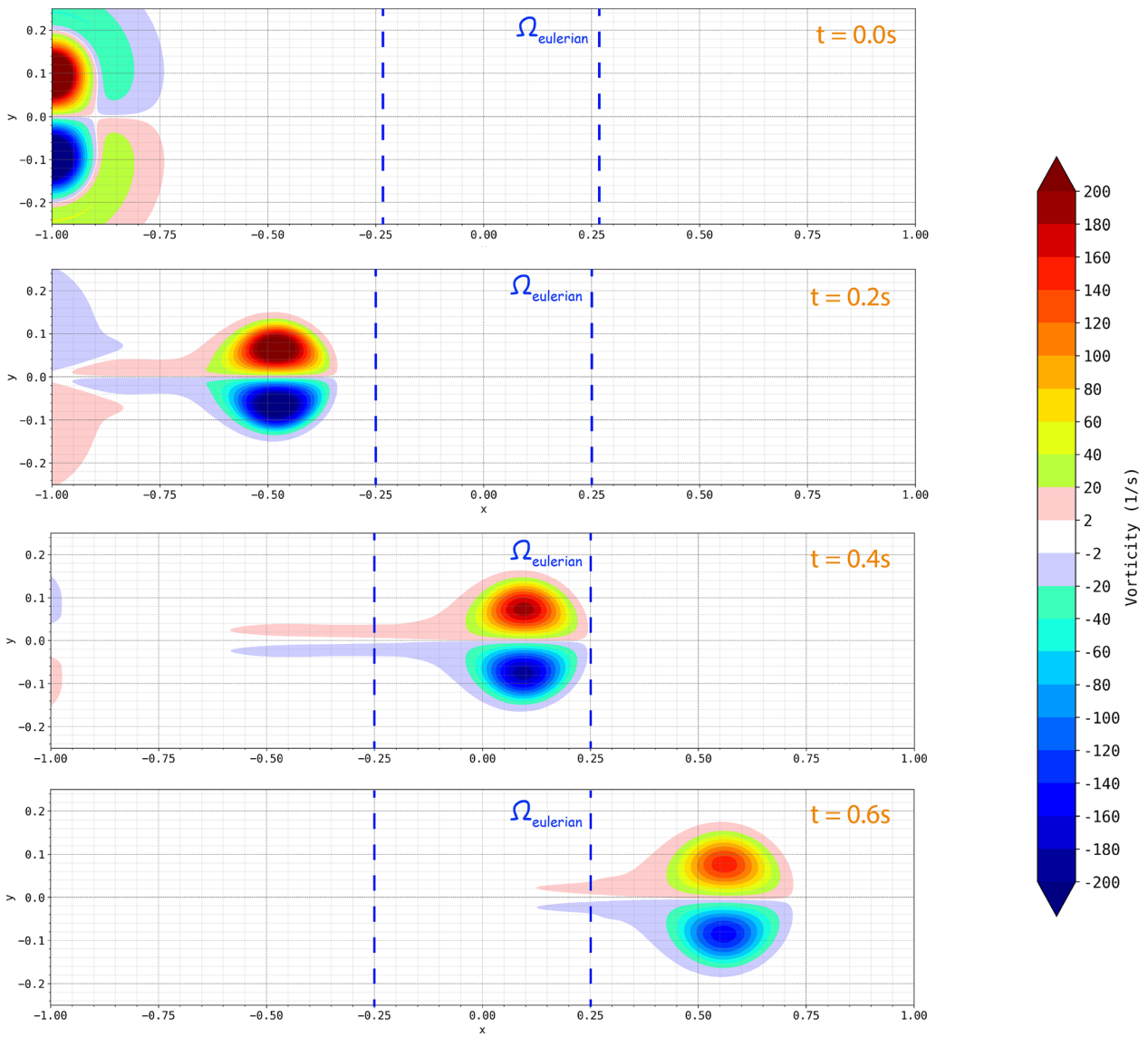

FIG. 13. The evolution of the vorticity field of the dipole in the unbounded domain at $t = [0.0, 0.2, 0.4, 0.6]$s. The contours are in accordance with the corresponding contours in the work of Palha *et al.*[37]

can be transmitted back to the particles without inducing significant error. While a sensitivity analysis for the simulation parameters, such as particle resolution, Eulerian mesh resolution, and time step, could be conducted, such an analysis falls beyond the scope of this paper.

B. Dipole in the unbounded domain

This case deals with the evolution of a vortex dipole in an unbounded domain (there are no solid boundaries). The specific cases were also examined by Palha *et al.*[37] and so it would be interesting to show a comparison between the two solvers. All the parameters are set to be the same, and specifically, the simulation is initialized with a Clercx–Bruneau dipole,[58] with the positive monopole located at $(x_1, y_1) = (-1.0, 0.1)$ and the negative monopole at $(x_2, y_2) = (-1.0, -0.1)$. Both the monopoles have a radius $R = 0.1$, and the induced vorticity field is calculated as

$$\omega(x, y, 0) = \omega_0 \left(1 - \frac{{r_1}^2}{R^2}\right) e^{-(r_1^2/R^2)} - \omega_0 \left(1 - \frac{{r_2}^2}{R^2}\right) e^{-(r_2^2/R^2)}. \quad (17)$$

where $\omega_0 = 299.528\,385\,375$ is the characteristic vorticity, and $r_i^2 = (x - x_i)^2 + (y - y_1)^2$. The configuration and the geometry of the case can be seen in Fig. 11, while the simulation parameters are summarized in Table II.

The maximum vorticity of the dipole during the simulation is depicted in Fig. 12. In this plot also, the FE and hybrid results from

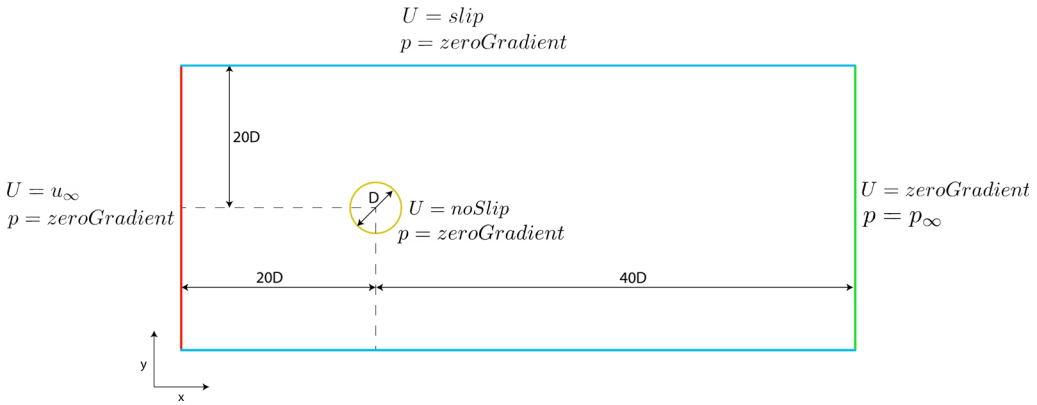

FIG. 14. The geometry and the boundary conditions for the pure Eulerian case of the flow around a cylinder at $Re = 550$.

TABLE III. Grid Convergence Study (GCS) for the pure Eulerian case of a flow around a cylinder at $Re = 550$.

Case	No of cells	mean cd	max cl	Strouhal
Coarse	12 820	1.429	1.1067	0.2333
First refinement level	28 212	1.441	1.197	0.2325
Second refinement level	63 366	1.441	1.208	0.2272
Third refinement level	140 540	1.434	1.206	0.2272
Fourth refinement level	310 500	1.437	1.204	0.2272
Fifth refinement level	512 080	1.431	1.199	0.2272

Palha *et al.*[37] are presented. It can be observed that the present solver predicts the maximum vorticity in the flow in exactly the same way as the reference. The same decrease in the vorticity when the dipole inserts the Eulerian domain is present here. In addition to this, Fig. 13 illustrates the vorticity contours in four different time instances, and specifically at the times $t = [0.0, 0.2, 0.4, 0.6]$s. These results are in total accordance with the corresponding results of the reference case. An important comment that must be added here is that the trailing vorticity of the dipole undergoes a noticeable reduction upon traversing the Eulerian subdomain. This reduction is a direct consequence of the increased artificial diffusion introduced by the Eulerian solver.

C. Flow around a cylinder at low Reynolds number ($Re = 550$)

The last case to validate the hybrid solver is the flow around a cylinder at $Re = 550$. This is a test case extensively studied in the past,[59] and it can be considered as a simplified case of the flow around an airfoil. This allows us to assess the performance of the solver when simulating real-world problems. The presence of the solid body facilitates benchmarking of the process of capturing vorticity generation and transferring the vorticity field from the Eulerian mesh to the Lagrangian particles.

The vorticity generation and the viscous phenomena take place in a narrow region close to the solid body, and they are responsible for its aerodynamic behavior. For the validation of our solver, the aerodynamic coefficients c_l and c_d and the Strouhal number are

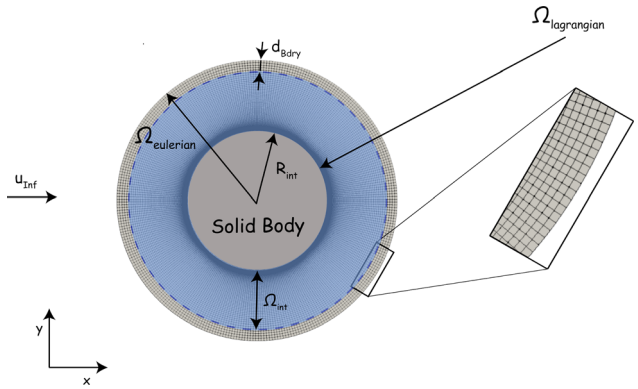

FIG. 15. The geometry of the flow around a cylinder case at $Re = 550$ for the hybrid, with a detailed view of the cells close to the numerical boundary. The Eulerian mesh is a narrow region close to the solid boundary, while the Lagrangian extends indefinitely.

TABLE IV. Problem and simulation parameters for the case of the flow around a cylinder at Re = 550.

Parameter	Symbol	Value	Dimension
Simulation time	t	0.0–180.0	s
Eulerian time step	Δt_E	0.002	s
Eulerian mesh density	N_{cells}	51 804	⋯
Diffusion and convection time step	$\Delta t_c = \Delta t_d$	0.004	s
Overlap ratio	λ	1.0	⋯
Vortex blob spacing	h	0.006	m
Interpolation domain offset from Eulerian boundary	d_{bdry}	0.1	m
Gaussian kernel width spreading	k	2	⋯
Population control thresholds	$(\Gamma_{loc}, \Gamma_{glob})$	$(1 \times 10^{-6}, 1 \times 10^{-8})$	⋯

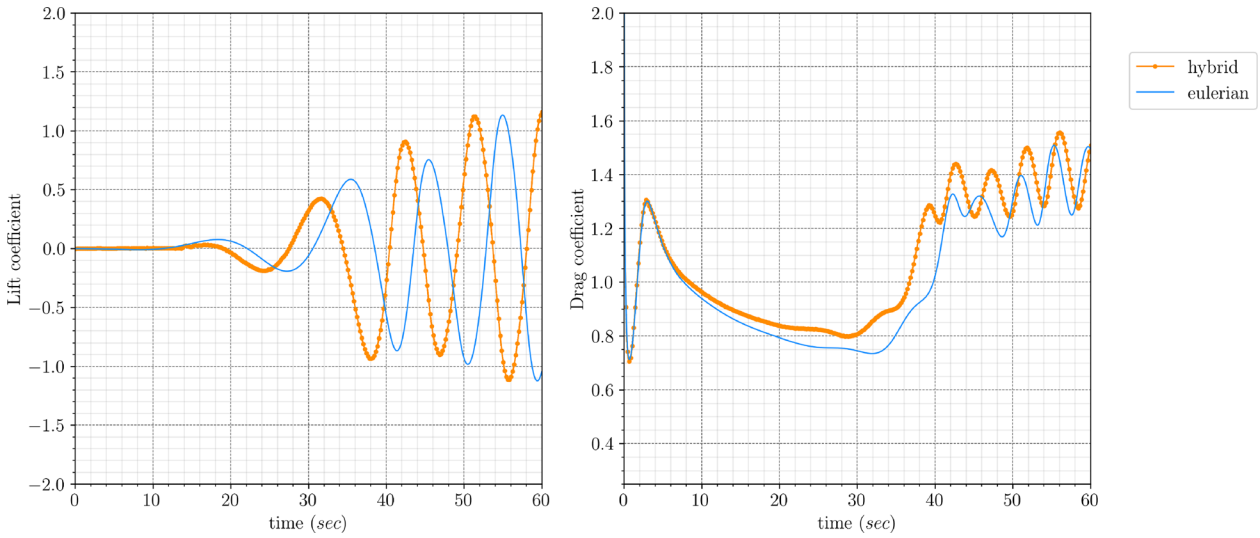

FIG. 16. Aerodynamic coefficients for the flow around a cylinder at $Re = 550$ for the hybrid and the pure Eulerian case, at the initial phase.

compared with the corresponding results of pure Eulerian simulations performed in OpenFOAM, and the results from the hybrid solver developed by Billuart *et al.*[30] The computational domain for the pure Eulerian case and the boundary conditions are illustrated in Fig. 14. A Grid Convergence Study (GCS) analysis of the pure Eulerian case was performed to establish the reference case for comparison with the new solver, and it can be seen in Table III. The initial mesh case employed in this study was obtained from the study conducted by Alletto.[3]

Figure 15 depicts the domain decomposition utilized for the hybrid simulation, where the Eulerian subdomain is confined to a narrow region in proximity to the body extending from the surface of the cylinder to a radius $R_{\text{eulerian}} = 2 \times R_{\text{cylinder}}$. The parameters employed for this simulation are outlined in Table IV. The value of the particle spacing h comes from the findings presented by Billuart *et al.*,[30] where it is mentioned that the particle spacing should match the cell size of the Eulerian domain close to the outer boundary, in order to minimize the interpolation errors at that region.

For the Eulerian subdomain, a structured grid (constructed in Gmsh[60]) is used, as it can be seen in Fig. 15. The cells close to the numerical boundary have aspect ratio close to 1.0, and their size is similar to that of the core size of the Lagrangian particles, in order to reduce the interpolation errors close to the boundary during the correction step.[30]

The primary goal of this case study is to examine the periodic phase of the flow around a cylinder, rather than focusing on the

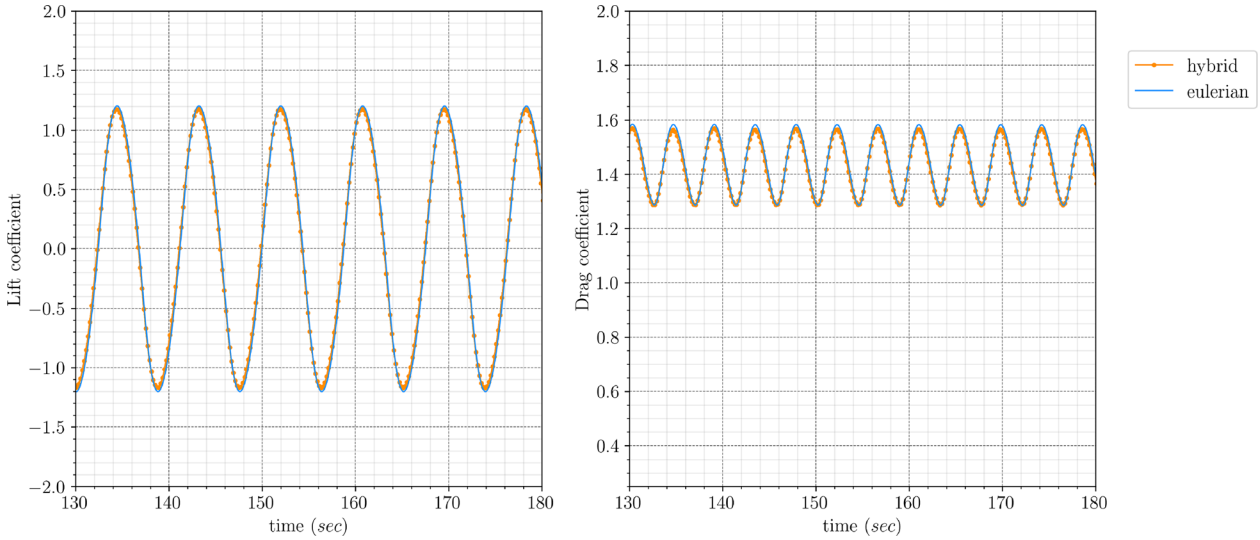

FIG. 17. Aerodynamic coefficients for the flow around a cylinder at $Re = 550$ for the hybrid and the pure Eulerian case, at the periodic phase.

TABLE V. Maximum values and relative errors of the aerodynamic coefficients and Strouhal number for the case of the flow around a cylinder at $Re = 550$.

Case	Mean cd	Max cl	Strouhal
Pure Eulerian (after GCS)	1.431	1.199	0.2272
Present hybrid	1.427	1.190	0.2272
Error	0.279%	0.750%	...
Billuart *et al.*[30] (reference hybrid)	1.427	1.177	0.2272

starting-transition phase. As a result, the validation process emphasizes the fully developed flow regime, where a periodic Von Karman vortex street can be observed in the cylinder's wake. Nonetheless, the aerodynamic coefficients during the initial phase are also provided to offer a comprehensive overview of the simulation. The comparison on the aerodynamic coefficients with the pure Eulerian case after GCS is depicted in Figs. 16 and 17. The aerodynamic coefficients and the Strouhal number are compared with the pure Eulerian simulation in OpenFOAM, but also with the results of the hybrid solver presented by Billuart *et al.*[30] This comparison in summarized in Table V.

Figure 16 demonstrates the initial phase of the flow. It is important to note that this study primarily focuses on investigating the periodic phase. Due to this emphasis, there is a discrepancy between the two solvers during the initial phase. Specifically, no perturbation was introduced to initiate instabilities from the beginning, resulting in different starting phases between the two solvers. The curves depicted in Fig. 17 represent the periodic phase, and they exhibit remarkable similarity. Both solvers yield the same Strouhal number, with the difference in aerodynamic forces being less than 1.0%. The two lines here have been adjusted to align with each other at the same phase.

The outcomes obtained using the current hybrid solver demonstrate excellent concurrence with the corresponding results from the hybrid solver developed by Billuart *et al.*[30] The maximum disparity in the lift coefficient is around 1%, while the Strouhal number and the mean value of the drag coefficient are identical up to the third decimal place, as presented in Table V.

Figure 18 illustrates the vorticity field for the hybrid and for the pure Eulerian case at four different time instances. In order to have an agreement of the vortical structures at the wake for the two cases, different time instances will be presented under the condition that the two simulations are at similar stage. The first contour is at the beginning of the simulation where the two vortices are being created at top and bottom part of the cylinder. During the transition phase, two

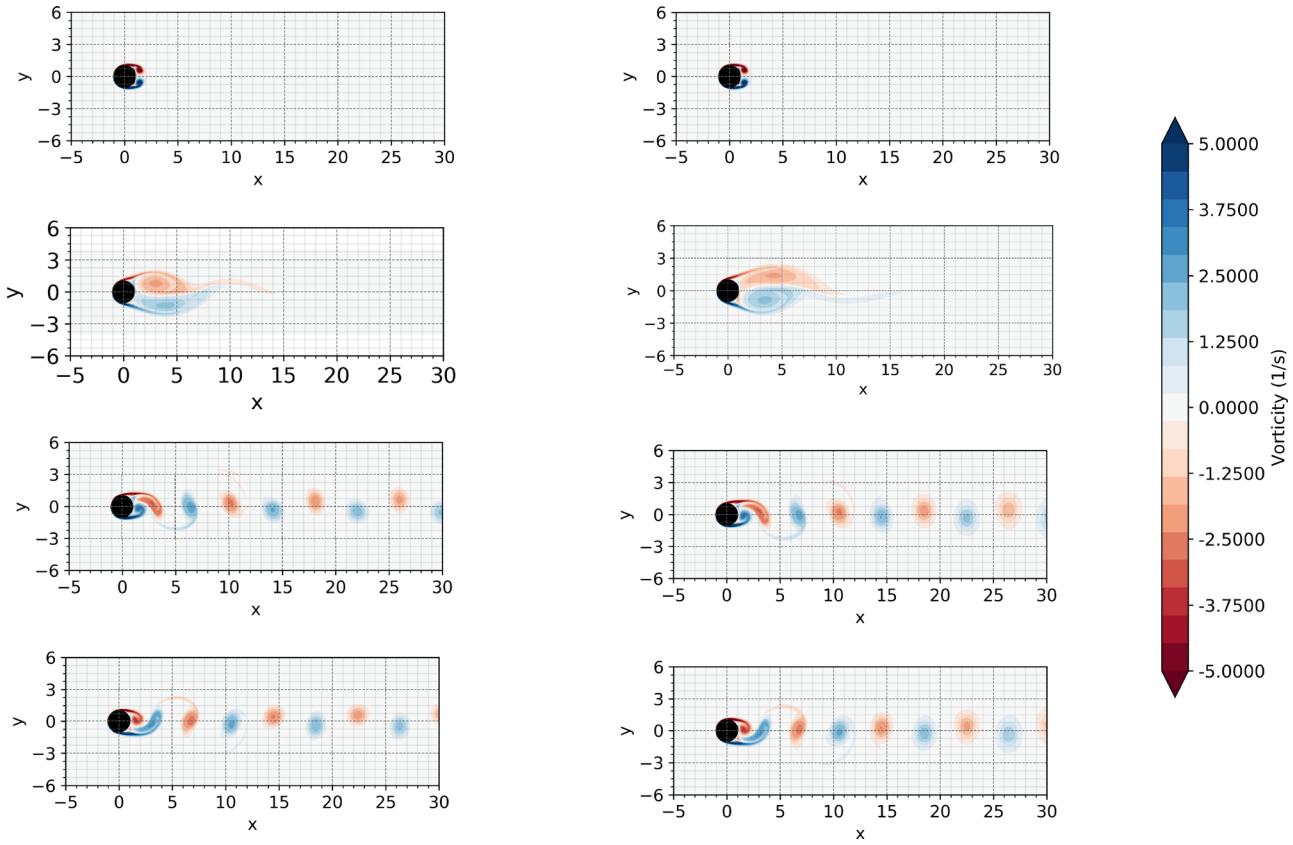

FIG. 18. The vorticity field contours at four different time instances (starting phase, transition phase, minimum lift, and maximum lift) for the hybrid (left) and the pure Eulerian (right) case, for the case of the flow around a circular cylinder at $Re = 550$.

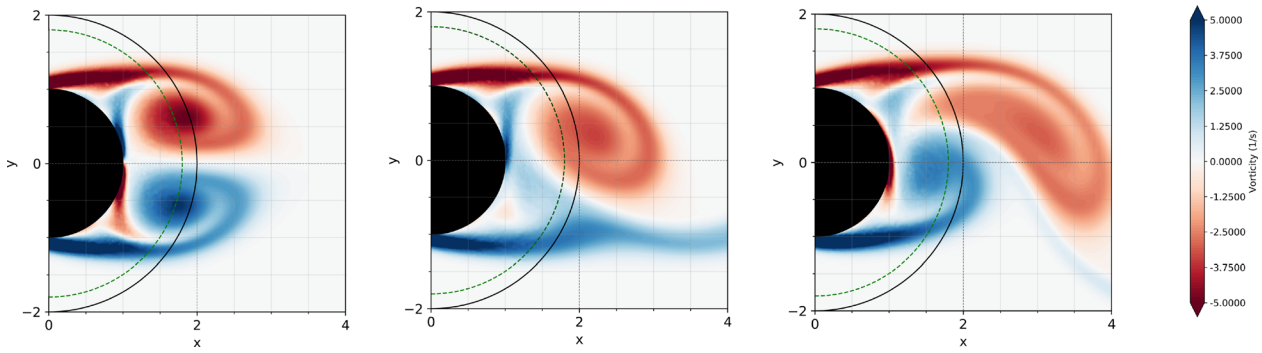

FIG. 19. The vorticity field contours at three different time instances (starting phase, minimum lift, and maximum lift) for the hybrid simulation, for the case of the flow around a circular cylinder at $Re = 550$. The contours depict the region close the solid boundary to focus on the transition between the Eulerian and the Lagrangian field. Inside the interpolation region, the Eulerian solution is depicted, while the rest is the Lagrangian solution.

similar instances are presented, but there is a notable difference between them. As mentioned earlier, this distinction arises due to the fact that the transition in the two solvers does not occur in the same manner. Specifically, the most significant contrast between these instances lies in the direction where the instability initiates. The third and the fourth contours are at a time instance where the lift coefficient is minimum and maximum. It can be seen that the hybrid solver can reproduce the vorticity field in great agreement with the pure Eulerian solver. It is worth mentioning that in this figure, it can be observed that the vortical structures that are in the far-field, appear more diffused in the pure Eulerian simulation, which demonstrates the diffusive nature of Eulerian solvers.

Indeed, the transition region between the Eulerian and Lagrangian subdomains is a critical aspect of hybrid simulations and can introduce errors if not handled properly. The smooth transition observed in Fig. 19 indicates that the coupling between the two subdomains is well implemented and the vorticity field can be accurately transferred between them. This is important for the overall accuracy of the simulation, as errors in this region can propagate throughout the domain and affect the entire solution.

VI. CONCLUSIONS AND DISCUSSION

It has been shown that OpenFOAM can work along a Lagrangian VPM in the framework of a hybrid Eulerian–Lagrangian solver without any crucial complexity. The two solvers can work together without any irregularities introduced by their coupling. The hybrid solver is capable of simulating unbounded and bounded flows, showing a great agreement with existing CFD solvers. In all cases that the solver was tested, the transition of the solution from the Eulerian to the Lagrangian domain was very smooth. The hybrid solver is capable of predicting the aerodynamic coefficients as well as the Strouhal number with variations less than 1.0%. The present hybrid solver produces similar results with the existing hybrid solver presented in Ref. 30. The hybrid solver seems to reduce the artificial diffusion that is present in pure Eulerian simulation. This effect is particularly pronounced in the case of the dipole, where even a minor Eulerian domain can significantly diffuse the trailing vorticity. Furthermore, when considering the flow around the cylinder, a subtle reduction in artificial diffusion becomes apparent in the downstream-traveling vortices.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Delft Blue supercomputer at the Technical University of Delft for providing the necessary computing resources to carry out the simulations presented in this manuscript. The authors also thank the support staff at the Delft High Performance Computing Centre (DHPC) for their technical assistance during the simulations.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Rention Pasolari: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Project administration (equal); Resources (equal); Software (equal); Validation (equal); Visualization (equal); Writing – original draft (equal). **Carlos J. Ferreira:** Conceptualization (equal); Formal analysis (equal); Methodology (equal); Project administration (equal); Supervision (equal); Validation (equal); Writing – review & editing (equal). **Alexander van Zuijlen:** Conceptualization (equal); Formal analysis (equal); Methodology (equal); Project administration (equal); Supervision (equal); Validation (equal); Writing – review & editing (equal).

DATA AVAILABILITY

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

NOMENCLATURE

e_z	Unit vector in z-direction
p	Pressure
r	Distance
Subscript p	Particle
Subscript ∞	Free-stream condition
t	Time

$\boldsymbol{u}$	Velocity
$\boldsymbol{x}$	Position
Γ	Strength/circulation
ν	Kinematic viscosity
ρ	Density
σ	Core radius
ω	Vorticity

REFERENCES

[1]C. Simão Ferreira, G. Van Kuik, G. Van Bussel, and F. Scarano, "Visualization by PIV of dynamic stall on a vertical axis wind turbine," Exp. Fluids **46**, 97–108 (2009).
[2]M. Colera and M. Pérez-Saborid, "An efficient finite differences method for the computation of compressible, subsonic, unsteady flows past airfoils and panels," J. Comput. Phys. **345**, 596–617 (2017).
[3]M. Alletto, "Comparison of overset mesh with morphing mesh: Flow over a forced oscillating and freely oscillating 2D cylinder," OpenFOAM® J. **2**, 13–30 (2022).
[4]E. Daniele, "Wind turbine control in computational fluid dynamics with OpenFOAM," Wind Eng. **41**, 213–225 (2017).
[5]Z. Cheng, F.-S. Lien, E. Yee, and J. H. Zhang, "Mode transformation and interaction in vortex-induced vibration of laminar flow past a circular cylinder," Phys. Fluids **34**, 033607 (2022).
[6]Q. Zhu and J. Yan, "A moving-domain CFD solver in FEniCS with applications to tidal turbine simulations in turbulent flows," Comput. Math. Appl. **81**, 532–546 (2021).
[7]C. J. Ruiz-Sánchez, A. Martínez-Cava, M. Chávez-Módena, and E. Valero, "An adjoint-based drag reduction technique for unsteady flows," Phys. Fluids **35**, 073603 (2023).
[8]L. C. Hsu, J. Z. Ye, and C. H. Hsu, "Simulation of flow past a cylinder with adaptive spectral element method," J. Mech. **33**, 235–247 (2017).
[9]A. Palha and M. Gerritsma, "A mass, energy, enstrophy and vorticity conserving (MEEVC) mimetic spectral element discretization for the 2D incompressible Navier–Stokes equations," J. Comput. Phys. **328**, 200–220 (2017).
[10]R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen Differenzengleichungen der mathematischen Physik," Math. Ann. **100**, 32–74 (1928).
[11]D. Rettenmaier, D. Deising, Y. Ouedraogo, E. Gjonaj, H. De Gersem, D. Bothe, C. Tropea, and H. Marschall, "Load balanced 2D and 3D adaptive mesh refinement in OpenFOAM," SoftwareX **10**, 100317 (2019).
[12]J. Pan, C. Ferreira, and A. van Zuijlen, "Estimation of power performances and flow characteristics for a Savonius rotor by vortex particle method," Wind Energy **26**, 76–97 (2023).
[13]G. Cottet and P. Koumoutsakos, *Vortex Methods: Theory and Practice* (Cambridge University Press, Cambridge, 2000).
[14]G. S. Winckelmans, "Vortex methods," in *Encyclopedia of Computational Mechanics*, 2nd ed. (John Wiley & Sons, Ltd, 2017), pp. 1–24.
[15]C. Mimeau and I. Mortazavi, "A review of vortex methods and their applications: From creation to recent advances," Fluids **6**, 68 (2021).
[16]A. Goude and S. Engblom, "Adaptive fast multipole methods on the GPU," J. Supercomput. **63**, 897–918 (2013).
[17]S. Engblom, "On well-separated sets and fast multipole methods," Appl. Numer. Math. **61**, 1096–1102 (2011).
[18]Q. Hu, N. A. Gumerov, and R. Duraiswami, "GPU accelerated fast multipole methods for vortex particle simulation," Comput. Fluids **88**, 857–865 (2013).
[19]P. Chatelain, A. Curioni, M. Bergdorf, D. Rossinelli, W. Andreoni, and P. Koumoutsakos, "Billion vortex particle direct numerical simulations of aircraft wakes," Comput. Methods Appl. Mech. Eng. **197**, 1296–1304 (2008).
[20]J. S. Calabretta and R. A. McDonald, "A three dimensional vortex particle-panel method for modeling propulsion-airframe interaction," AIAA Paper No. AIAA 2010-679, 2010.
[21]P. Poncet, "Analysis of an immersed boundary method for three-dimensional flows in vorticity formulation," J. Comput. Phys. **228**, 7268–7288 (2009).
[22]D. Rossinelli and P. Koumoutsakos, "Vortex methods for incompressible flow simulations on the GPU," Visual Comput. **24**, 699–708 (2008).
[23]M. J. Stock, A. Gharakhani, and C. P. Stone, "Modeling rotor wakes with a hybrid OVERFLOW-vortex method on a GPU cluster," AIAA Paper No. AIAA 2010-4553, 2010.
[24]J. L. Guermond and H. Z. Lu, "A domain decomposition method for simulating advection dominated, external incompressible viscous flows," Comput. Fluids **29**, 525–546 (2000).
[25]S. G. Huberson and S. G. Voutsinas, "Particles and grid," Comput. Fluids **31**, 607–625 (2002).
[26]A. Golas, R. Narain, J. Sewall, P. Krajcevski, P. Dubey, and M. Lin, "Large-scale fluid simulation using velocity-vorticity domain decomposition," ACM Trans. Graphics **31**, 1–10 (2012).
[27]G. Papadakis, V. A. Riziotis, and S. G. Voutsinas, "A hybrid Lagrangian–Eulerian flow solver applied to elastically mounted cylinders in tandem arrangement," J. Fluids Struct. **113**, 103686 (2022).
[28]G. Papadakis and S. G. Voutsinas, "In view of accelerating CFD simulations through coupling with vortex particle approximations," J. Phys. **524**, 012126 (2014).
[29]Y. Shi, G. Xu, and P. Wei, "Rotor wake and flow analysis using a coupled Eulerian–Lagrangian method," Eng. Appl. Comput. Fluid Mech. **10**, 384–402 (2016).
[30]P. Billuart, M. Duponcheel, G. Winckelmans, and P. Chatelain, "A weak coupling between a near-wall Eulerian solver and a Vortex Particle-Mesh method for the efficient simulation of 2D external flows," J. Comput. Phys. **473**, 111726 (2023).
[31]G. Papadakis and S. G. Voutsinas, "A strongly coupled Eulerian Lagrangian method verified in 2D external compressible flows," Comput. Fluids **195**, 104325 (2019).
[32]G. H. Cottet, "A particle-grid superposition method for the Navier-Stokes equations," J. Comput. Phys. **89**(2), 301–318 (1990).
[33]G. Daeninck, "Developments in hybrid approaches: Vortex method with known separation location; vortex method with near-wall Eulerian solver; RANS-LES coupling," Ph.D. thesis, Université Catholique de Louvain (2006).
[34]I. P. Christiansen, "Numerical simulation of hydrodynamics by the method of point vortices," J. Comput. Phys. **13**, 363–379 (1973).
[35]D. G. Caprace, P. Chatelain, and G. Winckelmans, "Wakes of rotorcraft in advancing flight: A large-eddy simulation study," Phys. Fluids **32**, 087107 (2020).
[36]M. J. Stock and A. Gharakhani, "A hybrid high-order vorticity-based Eulerian and Lagrangian vortex particle method, the 2D case," in Fluids Engineering Division Summer Meeting, Volume 1: Aerospace Engineering Division Joint Track; Computational Fluid, 2021.
[37]A. Palha, L. Manickathan, C. S. Ferreira, and G. van Bussel, "A hybrid Eulerian–Lagrangian flow solver," arXiv:1505.03368 (2015).
[38]M. S. Alnaes, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, "The FEniCS project version 1.5," Archive Numer. Software **3**, 9–23 (2015).
[39]H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A tensorial approach to computational continuum mechanics using object orientated techniques," Comput. Phys. **12**, 620–631 (1998).
[40]G. Ferrari, D. Federici, P. Schito, F. Inzoli, and R. Mereu, "CFD study of Savonius wind turbine: 3D model validation and parametric analysis," Renewable Energy **105**, 722–734 (2017).
[41]D. Wang, F. Xie, T. Ji, X. Zhang, Y. Lu, and Y. Zheng, "Prediction of wind shear layer for dynamic soaring by using proper orthogonal decomposition and long short term memory network," Phys. Fluids **35**, 085103 (2023).
[42]P. S. D. Robinson, "A coupled actuator line and finite element and analysis tool," OpenFOAM® J. **2**, 81–93 (2022).
[43]N. Godinaud, P. Boivin, P. Freton, J. J. Gonzalez, and F. Camy-Peyret, "Development of a new OpenFOAM solver for plasma cutting modeling," Comput. Fluids **241**, 105479 (2022).
[44]M. V. Kraposhin, E. V. Smirnova, T. G. Elizarova, and M. A. Istomina, "Development of a new OpenFOAM solver using regularized gas dynamic equations," Comput. Fluids **166**, 163–175 (2018).
[45]A. J. Chorin, "Numerical study of slightly viscous flow," J. Fluid Mech. **57**, 785–796 (1973).
[46]O. R. Tutty, "A simple redistribution vortex method (with accurate body forces)," arXiv:1009.0166 (2010).
[47]F. Moukalled, L. Mangani, and M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics* (Springer, Cham, 2016).

[48]K. Anirudh and S. Dhinakaran, "On the onset of vortex shedding past a two-dimensional porous square cylinder," J. Wind Eng. Ind. Aerodyn. **179**, 200–214 (2018).
[49]G. Gaurava and P. Dhatrak, "Performance analysis of vertical axis wind turbines by varying tip-speed ratio using open source CFD solver," AIP Conf. Proc. **2358**, 110005 (2021).
[50]P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nat. Methods **17**, 261–272 (2020).
[51]G. Van Rossum and F. L. Drake, *Python 3 Reference Manual* (CreateSpace, Scotts Valley, CA, 2009).
[52]ISO, *Programming languages — C — Amendment 1: C integrity (ISO/IEC 9899:1990/AMD 1:1995)* (ISO, 1996).
[53]B. Stroustrup, *The C++ Programming Language* (Pearson Education India, 2000).
[54]S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: The best of both worlds," Comput. Sci. Eng. **13**, 31–39 (2011).
[55]R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel Programming in OpenMP* (Morgan Kaufmann, 2001).
[56]P. Vingelmann and F H. P. Fitzek, CUDA toolkit, release 10.2.89, https://developer.nvidia.com/cuda-toolkit (2020).
[57]H. Lamb, *Hydrodynamics*, 6th ed. (Cambridge University Press, Cambridge, 1932), pp. 1–8.
[58]H. J. H. Clercx and C.-H. Bruneau, "The normal and oblique collision of a dipole with a no-slip boundary," Comput. Fluids **35**, 245–279 (2006).
[59]P. Koumoutsakos and A. Leonard, "High-resolution simulations of the flow around an impulsively started cylinder using vortex methods," J. Fluid Mech. **296**, 1–38 (1995).
[60]C. Geuzaine and J.-F. Remacle, Jean-Francois, Gmsh, version 4.6.0, http://gmsh.info/ (2022).