Efficiency of Multiscale Hybrid Grid-Particle Vortex Methods

Mustapha El Ossmani, Philippe Poncet

MULTISCALE MODEL. SIMUL.
Vol. 8, No. 5, pp. 1671–1690

EFFICIENCY OF MULTISCALE HYBRID GRID-PARTICLE VORTEX METHODS[*]

M. EL OSSMANI[†] AND P. PONCET[‡]

Abstract. This article presents a study of computational cost for the vortex method, a Lagrangian numerical scheme using particles of fluids for simulation of three-dimensional flows. Its main features are to compute accurately transport effects and to be very robust, that is to say, often without prohibitive stability condition. Special attention is given to hybrid grid-particle vortex methods. They are shown to scale as $\mathcal{O}(n \log n)$, even when used in a multiscale context. Furthermore, a discussion is provided on the best strategies for simulations in complex geometry. Computational cost is shown to have the same efficiency when performing multiscale simulations of three-dimensional flows in complex geometry.

Key words. vortex methods, particle methods, three-dimensional flows, Navier–Stokes equations, multiscale simulation, wakes, penalization

AMS subject classifications. 65M06, 65M55, 76M06

DOI. 10.1137/090765006

1. Introduction.

The vortex-in-cell method is a numerical scheme introduced for computation of fluid dynamics, using fully the Lagrangian feature of the method in order to compute accurately transport and convection effects. It consists of a discretization of the fluid domains by cells of fluids defined by their position, volume, and amount of vorticity, expressed as a measure solution of the Navier–Stokes equation in their velocity-vorticity formulation.

While its early developments involved mainly two-dimensional (2D) flows, using random walks and raw Biot–Savart laws in order to compute, respectively, diffusion and transport on particles (see [17], then later [21] and [2]), these methods have since been dramatically improved. Nowadays, challenges on vortex methods deal with large three-dimensional (3D), multiphysics and/or multiscale flows.

The state of the art for 3D simulations using particle methods involves either very large simulations without multiscale (see [13], [37], and [15]) or multiscale simulations-based velocity-pressure formulation of Navier–Stokes equations (see [5], [29], and [40]). Problems involving complex geometry are not discussed in any of these studies.

In the present article, focus is given on multiscale 3D flow computation using vortex methods. Indeed, the Lagrangian feature of vortex methods, based on velocity-vorticity formulation of the Navier–Stokes equations, often allows the use of large time steps, thus allowing the accurate study of flow over large time scales (see [34], for example). This article brings new results on computational efficiency of multiscale vortex methods in both simple and complex geometries. Moreover, despite the fact that parallelization is an important aspect when performing large simulations, this article focuses on the numerical methods themselves rather than considering the impact or efficiency of parallel computing.

[*]Received by the editors July 14, 2009; accepted for publication (in revised form) July 2, 2010; published electronically September 23, 2010.
http://www.siam.org/journals/mms/8-5/76500.html

[†]ENSAM-Meknes, Marjane II, B.P. 4024 Beni Mhamed, Meknes 50000, Morocco (elossman@insa-toulouse.fr). The work of this author was supported by ANR grant 06BLAN0306-03.
[‡]Corresponding author. Toulouse Institute of Mathematics, CNRS, Team MIP, F-31077 Toulouse, France and Université de Toulouse, INSA, GMM 135 avenue de Rangueil, F-31077 Toulouse, France (philippe.poncet@math.univ-toulouse.fr).

Well-known results and historically Lagrangian techniques for vortex methods are described below in section 2. Section 2.1 presents the Biot–Savart laws. Section 2.2 describes the coupling between grids and particles and gives an estimation of the computational cost for such hybrid vortex method. The scalability obtained for the grid-particle coupling is extended to the multiscale context in section 3.

Section 4 contains a discussion on the most appropriate method used to perform the multi-scale vortex method in complex geometry among the boundary integral method, the implicit immersed boundary, Chorin's algorithm, and penalization. The penalization method is shown to be the most advantageous one, illustrated on the simulation of a three-dimensional wake generated behind a sphere for a Reynolds number $Re = 500$.

2. Conventional vortex-in-cell (VIC) methods.

2. Conventional vortex-in-cell (VIC) methods. One considers the Navier–Stokes equations, for an incompressible fluid in a domain Ω, of constant density ρ and constant dynamic viscosity μ:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u}\cdot\nabla\mathbf{u} - \mu\Delta\mathbf{u} = \mathbf{f} - \nabla p, \tag{2.1}$$

where $\mathbf{u}$ is the divergence-free velocity field satisfying the no-slip condition $\mathbf{u} = 0$ on $\Gamma = \partial\Omega$, p the pressure, and the external force, assumed to derive from a potential (that is to say, the gradient of a scalar function). Taking the curl of (2.1) and introducing the vorticity as $\boldsymbol{\omega} = \mathrm{curl}\mathbf{u}$, one gets the Navier–Stokes equation in its vorticity formulation:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \mathbf{u}\cdot\nabla\boldsymbol{\omega} - \boldsymbol{\omega}\cdot\nabla\mathbf{u} - \nu\Delta\boldsymbol{\omega} = 0 \tag{2.2}$$

with boundary conditions $\mathbf{u} = 0$, where $\nu = \mu/\rho$ is the kinematic viscosity. Two nonlinear terms are present in this equation: $\mathbf{u}\cdot\nabla\boldsymbol{\omega}$ is a transport of vorticity, usually leading to the most restrictive stability condition, and $\boldsymbol{\omega}\cdot\nabla\mathbf{u}$ is a stretching, responsible of evolution in vortex orientation.

Vortex-in-cell methods consist of discretizing the fluid domain Ω by a set of N particles (or cells) of volume v_p, position $\boldsymbol{\xi}_p$, and vorticity $\boldsymbol{\omega}_p$, p indexing the particles from 1 to N. The mathematical formulation of this particle description consists of writing the vorticity as a measure solution

$$\boldsymbol{\omega} = \sum_{p=1}^{N} \boldsymbol{\omega}_p \delta_{\boldsymbol{\xi}_p} v_p, \tag{2.3}$$

where $\delta_{\boldsymbol{\xi}_p}(\zeta) = \zeta(\boldsymbol{\xi}_p)$, for any test function ζ, is the Dirac function in $\boldsymbol{\xi}_p$.

The Lagrangian formulation of (2.2) can then be written as follows:

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{\omega}_p}{\mathrm{d}t} = [\boldsymbol{\omega}\cdot\nabla\mathbf{u} + \nu\Delta\boldsymbol{\omega}](\boldsymbol{\xi}_p(t), t), \\ \dfrac{\mathrm{d}\boldsymbol{\xi}_p}{\mathrm{d}t} = \mathbf{u}(\boldsymbol{\xi}_p(t), t), \\ \dfrac{\mathrm{d}v_p}{\mathrm{d}t} = v_p(t)\,\mathrm{div}\mathbf{u}(\boldsymbol{\xi}_p(t), t) \equiv 0 \end{cases} \tag{2.4}$$

for $p = 1, \dots, N$.

For problems involving a nonempty boundary, that is to say, $\Omega \neq \mathbb{R}^3$, the velocity field satisfies the no-slip boundary condition $\mathbf{u} = 0$ on the domain boundary Γ, and the tangential part of this boundary condition is responsible for vorticity creation at walls due to compatibility conditions between $\boldsymbol{\omega}$ and $\mathbf{u}$.

Among the main advantages of this Lagrangian formulation, the transport term $\mathbf{u} \cdot \nabla\boldsymbol{\omega}$ vanishes as well as its related stability condition. In most Eulerian methods, such a CFL reads $\|\mathbf{u}\|_\infty \delta t/\delta x \leqslant C$, where δt and δx denote the time and space discretization steps and potentially lead to prohibitive small time steps as δx gets small. Indeed, the convection part of the first line of dynamical system (2.4) induces a stability condition of the form $\|\boldsymbol{\omega}\|_\infty \delta t \leqslant C$, which does not depend on δx and makes Lagrangian methods especially interesting (for 2D simulations, the stability condition completely vanishes). Nevertheless, in some contexts involving very sharp vorticity gradients, such as homogeneous isotropic turbulence, combustion, or multi-fluids, the Eulerian and Lagrangian stability conditions can be equivalently restrictive when $\|\mathbf{u}\|_\infty/\delta x \simeq 1/\|\boldsymbol{\omega}\|_\infty$.

This Lagrangian feature is even more interesting for computation of very slightly viscous flows for which the (transport) CFL stability condition is dominant compared to the diffusion stability condition and for which computing a long time scale is interesting in order to characterize asymptotic behavior of the flow.

Furthermore, it is usual to split (2.2), over a time step, into a convective step and a diffusive step [4]. The resulting Lagrangian formulation of convective step reads

$$\frac{\mathrm{d}\boldsymbol{\omega}_p}{\mathrm{d}t} = [\boldsymbol{\omega} \cdot \nabla\mathbf{u}](\boldsymbol{\xi}_p(t), t), \quad \frac{\mathrm{d}\boldsymbol{\xi}_p}{\mathrm{d}t} = \mathbf{u}(\boldsymbol{\xi}_p(t), t), \tag{2.5}$$

with conservation of particle volume v_p for incompressible flows and subject to only normal boundary conditions $\mathbf{u}\cdot\mathbf{n} = 0$ on Γ, called usually a no-flow-through condition. Moreover, the diffusion step then reads

$$\begin{cases} \dfrac{\partial\boldsymbol{\omega}}{\partial t} - \nu\Delta\boldsymbol{\omega} = 0 \quad \text{in } \Omega\times]t, t+\delta t[, \\ \mathbf{u} = 0 \quad \text{on } \partial\Omega\times]t, t+\delta t[\end{cases} \tag{2.6}$$

for which the initial condition is the vorticity obtained at the end of the convective step. In this last equation the partial differential equation involves vorticity while the boundary condition involves the velocity. Section 4.2 will describe how to deal with such implicit boundary conditions.

A dynamical system, such as (2.4) or (2.5), can be integrated by any standard numerical method for ordinary differential equations (such as Runge–Kutta, for example), once one is able to compute $\mathbf{u}$ and $\nabla\mathbf{u}$ at location $\boldsymbol{\xi}_p$. The computation cost of building $\mathbf{u}$ and $\nabla\mathbf{u}$ from the set of vorticity $\boldsymbol{\omega}_p$ is of fundamental interest in the study of vortex-in-cell performance. The most-used techniques to do so are on the one hand the use of Biot–Savart, summarized in section 2.1, and on the other hand the use of grid-particle coupling, developed in section 2.2.

2.1. Biot–Savart laws. Historically, velocity computation for dynamical systems (2.4) and (2.5) has been performed by Biot–Savart law:

$$\mathbf{u}(\boldsymbol{\xi}_p) = \int_{\mathbb{R}^3} \nabla K(\boldsymbol{\xi}_p - \boldsymbol{x}) \times \boldsymbol{\omega}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \tag{2.7}$$

where $K(x) = (4\pi|x|)^{-1}$ is the Green kernel of the Laplacian operator. Given the particle description (2.3) of $\boldsymbol{\omega}$, formula (2.7) is equivalent to

$$\mathbf{u}(\boldsymbol{\xi}_p) = \sum_{q=1}^{N} \nabla K(\boldsymbol{\xi}_p - \boldsymbol{\xi}_q) \times \boldsymbol{\omega}_q v_q. \tag{2.8}$$

This method has been intensively and successfully used for decades for flow computations, with evaluations of (2.8) performed with a mollified kernel.

The cost of velocity computation $\mathbf{u}(\boldsymbol{\xi}_p)$ on the N particles scales as $\mathcal{O}(N^2)$, which is prohibitive for accurate simulations. This consequently requires the use of multipole expansions of kernel ∇K (or its mollification) and fast algorithms for the hierarchical structure of particle networks (for fast tree algorithms, see [25]). Such algorithms have been successfully developed and implemented for 2D vortex methods, making a consistent evaluation of formula (2.8) scale as $\mathcal{O}(N \log N)$.

Nevertheless, it has been shown in [20] that the 3D case does not allow such an improvement of computational cost, and this method based on grid-particle coupling is generally more efficient. Indeed, a balance between the number of pairwise interactions and the number of expansions has to be set. For example, a tree-code with computational cost scaling as $\mathcal{O}(N^{3/2})$ has been constructed in [10], and this exponent has been later dramatically reduced [39], down to $\mathcal{O}(N^{1.1})$ in [30], already mentioned in [19]. The major drawback of such method, even in the low exponent case, is the constant in front of this power law, which is in practice quite large. Moreover, pure 3D particle description requires evaluation of $\boldsymbol{\omega}_p \cdot \nabla \mathbf{u}(\boldsymbol{\xi}_p)$. This term vanishes in 2D context, but can be potentially difficult to compute accurately in a pure Lagrangian way.

Furthermore, in two dimensions as well as in three dimensions, computing diffusion of vorticity with a pure particle description has been historically performed by random walks. In order to improve the convergence rate, deterministic particle diffusion has been introduced by means of a convolution formula known as particle-strength-exchange (PSE), in the spirit of smoothed-particle-hydrodynamics methods. Let ϵ and h be, respectively, the characteristic lengths of the kernel support used in PSE methods and grid step. Convergence of PSE has been analyzed in [22] in the context of $\mathcal{O}((h/\epsilon)^2)$ convergence and has been later analyzed in the context of $\mathcal{O}\left(h^2\right)$ intrinsic convergence in [35], which is necessary, since in practice the kernel size is adapted to the grid step (which means h/ϵ is constant).

2.2. Hybrid grid-particle vortex methods. A hybrid grid-particle vortex method consists of computing the most possible quantities on a grid and at the same time using the Lagrangian feature of vortex methods in order to compute accurately transport without the restrictive stability condition on transport, such as for conventional Eulerian methods. The computational domain considered herein is a Cartesian box Ω, but such coupling has been successfully performed in the past in stretched geometries [20].

At the beginning of a time step, only the quantities defining the cells are available, that is to say, position $\boldsymbol{\xi}_p$, vorticity $\boldsymbol{\omega}_p$, and volume v_p. The coupling consists of interpolating the divergence-free vorticity field onto a grid, then computing the stream function $\boldsymbol{\psi}$ satisfying the following vectorial Poisson equation:

$$-\Delta \psi_x = \omega_x, \quad -\Delta \psi_y = \omega_y, \quad -\Delta \psi_z = \omega_z. \tag{2.9}$$

The boundary conditions of these Poisson equations are homogeneous Dirichlet conditions, except for the following components for which homogeneous Neumann conditions are set:

- on planes for which x is constant, $\partial \psi_x / \partial x = 0$,
- on planes for which y is constant, $\partial \psi_y / \partial y = 0$,
- on planes for which z is constant, $\partial \psi_z / \partial z = 0$.

These kinds of boundary conditions allow us to satisfy no-through-flow, that is to say, $\mathrm{curl}\boldsymbol{\psi} \cdot \mathbf{n} = 0$ on boundaries, as described thereafter.

$$\begin{tikzcd}
\boldsymbol{\omega} \text{ on particles} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \boldsymbol{\omega} \text{ on grid} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \begin{cases} -\Delta\boldsymbol{\psi} = \boldsymbol{\omega} \\ +\text{Boundary} \\ \text{conditions} \end{cases} \arrow[d, Rightarrow] \\
-\nu\Delta\boldsymbol{\omega} \text{ on particles} & \boldsymbol{\omega}\cdot\nabla\mathbf{u} \text{ on grid} \arrow[d, Rightarrow] & \mathbf{u} = \text{curl}\boldsymbol{\psi} \text{ on grid} \arrow[l, Rightarrow] \arrow[d, Rightarrow] \\
 & \begin{array}{c}\boldsymbol{\omega}\cdot\nabla\mathbf{u} \\ \text{on particles}\end{array} & \begin{array}{c}\mathbf{u} \text{ on} \\ \text{particles}\end{array}
\end{tikzcd}$$

FIG. 1. *Grid-particle coupling algorithm used to evaluate all terms of* (2.4), (2.5)*, and* (2.6).

Moreover, for any face of the Cartesian box Ω, the stream $\boldsymbol{\psi}$ satisfies $\text{div}\boldsymbol{\psi} = 0$ everywhere on boundary $\partial\Omega$. Since $\boldsymbol{\omega}$ is supposed divergence-free, the quantity $\text{div}\boldsymbol{\psi}$ satisfies the equation $-\Delta\text{div}\boldsymbol{\psi} = \text{div}\boldsymbol{\omega} = 0$, and consequently $\text{div}\boldsymbol{\psi} = 0$ everywhere in Ω. Several other kinds boundary conditions could also be applied, such as periodic, shearing-sheet, symmetric, or unbounded (see [15], for example).

The velocity is then provided by

$$\mathbf{u} = \text{curl}\boldsymbol{\psi} \tag{2.10}$$

and satisfies $\mathbf{u}\cdot\mathbf{n} = 0$ on boundaries, thanks to the choice of boundary conditions on stream $\boldsymbol{\psi}$ and the stretching by $\boldsymbol{\omega}\cdot\nabla\mathbf{u}$, by means of fourth order finite difference schemes on the grid. The relation $\mathbf{u} = \text{curl}\boldsymbol{\psi}$ gives that velocity satisfies $\text{div}\mathbf{u} = 0$ and

$$\text{curl}\mathbf{u} = \text{curlcurl}\boldsymbol{\psi} = -\Delta\boldsymbol{\psi} + \nabla\text{div}\boldsymbol{\psi} = \boldsymbol{\omega}.$$

The velocity and the stretching are then interpolated back to particles so that time integration of (2.5) can be performed. If diffusion is considered, for (2.4) or (2.6), the evaluation of the Laplace operator can be performed either on a grid followed by interpolation of $-\nu\Delta\boldsymbol{\omega}$ to particles or directly on a particle using a PSE scheme [35], [22]. This algorithm is summarized by the sketch on Figure 1.

The Runge–Kutta method and finite difference schemes used within the algorithm described above correspond to a computational cost linear with respect to the number of grid points. The overall method efficiency is consequently driven by the computational costs of grid-particle interpolations and the method used to solve the Poisson equation $-\Delta\boldsymbol{\psi} = \boldsymbol{\omega}$.

On the one hand, the Poisson solver used in Cartesian geometry is FISHPACK [42], [41], whose scalability scales as $\mathcal{O}(n\log n)$, n being the number of grid points. On the other hand, let $\tilde{\boldsymbol{\omega}}(\tilde{\boldsymbol{x}})$ denote the interpolation of the set of particle $\boldsymbol{\omega}_p$ on the grid point $\tilde{\boldsymbol{x}}$. The value on grid point is provided by the convolution formula

$$\tilde{\boldsymbol{\omega}}(\tilde{\boldsymbol{x}}) = \Lambda^\epsilon * \boldsymbol{\omega}(\tilde{\boldsymbol{x}}) = \sum_{p=1}^{N} \Lambda^\epsilon(\tilde{\boldsymbol{x}} - \boldsymbol{\xi}_p)\boldsymbol{\omega}_p v_p \tag{2.11}$$

with $\Lambda^\epsilon(\boldsymbol{x}) = \epsilon^{-3}\Lambda^{\otimes 3}(\boldsymbol{x}/\epsilon)$ and

$$\Lambda(x) = \begin{cases} \left(3x^3 - 5x^2 + 2\right)/2 & \text{if } 0 \leqslant x \leqslant 1, \\ (2-x)^2(1-x)/2 & \text{if } 1 \leqslant x \leqslant 2, \\ 0 & \text{if } x \geqslant 2. \end{cases} \tag{2.12}$$

This kernel is third order (in the sense that it preserves the 3 first moments of the distribution), twice continuously differentiable (except in 0), and symmetric, also known as M_4' Monaghan's kernel [32].

Moreover, let $\{\tilde{\boldsymbol{x}}_k\}_{k=1..n}$ denote the location of grid points. One proceeds to interpolation of vorticity on grid points onto particles as follows:

$$\boldsymbol{\omega}_p = \Lambda^\epsilon * \tilde{\boldsymbol{\omega}}(\boldsymbol{\xi}_p) = \sum_{k=1}^{n} \Lambda^\epsilon(\boldsymbol{\xi}_p - \tilde{\boldsymbol{x}}_k)\tilde{\boldsymbol{\omega}}(\tilde{\boldsymbol{x}}_k)\bar{v}, \tag{2.13}$$

where $\bar{v} = \delta x \delta y \delta z$ and δx, δy, and δz are the grid step sizes in the three dimensions of space.

The conservation of discrete integral is ensured by a spline-like property of this kernel:

$$\sum_{j\in\mathbb{Z}} \Lambda(x+j) = 1 \quad \forall x \in \mathbb{R}. \tag{2.14}$$

Besides, this property can be used more appropriately by considering an anisotropic interpolation kernel such as $\Lambda^\epsilon(\boldsymbol{x}) = \bar{v}^{-1}\Lambda\left(x/\delta x\right)\Lambda\left(y/\delta y\right)\Lambda\left(z/\delta z\right)$, where $\boldsymbol{x} = (x, y, z)$.

Since the support length of Λ^ϵ is 4ϵ in all directions, the sum above is restricted to the very neighborhood of $\boldsymbol{\xi}_p$: index of the required grid points with a contribution to this sum can be found by a simple integer division. It follows directly that the computational cost of this sum (2.13) is independent of n; thus the global grid-to-particle interpolation scales as $\mathcal{O}(N)$, N being the number of particles.

Reciprocally, the particle-to-grid interpolation (2.11) is performed by means of a loop over grid points. The sum being a loop over particles, the two loops can be swapped when implemented so that the particle-to-grid computational cost is the same as particle-to-grid interpolation, that is to say, scaling as $\mathcal{O}(N)$.

Furthermore, in order to avoid clustering and holes of particles, a frequent remesh of the particle lattice is performed: particles are interpolated to a new particle network collocated to the grid points (or staggered), using again the kernel displayed above. This remeshing procedure is nondiffusive and conservative since the kernel is third order. Its computational cost is also $\mathcal{O}(N)$, and the number of particle scales as the number of grid points, which means $\mathcal{O}(N/n) = 1$.

These steps are not algebraically conservative on vorticity divergence. Even if in practice vorticity divergence does not cumulate (see [20], [34]), some nonregular initial conditions (see [35], for example) may require periodically a projection on divergence-free fields. A solution is to replace $\boldsymbol{\omega}$ by $\boldsymbol{\omega} + \nabla\beta$ with a β solution of $-\Delta\beta = \mathrm{div}\boldsymbol{\omega}$ in the fluid domain and $\partial\beta/\partial n = -\boldsymbol{\omega}\cdot n$ on boundaries if $\boldsymbol{\omega}\cdot n$ has to be zero (on a flat body on which the condition $\mathbf{u} \equiv 0$ has to be satisfied). Another strategy simply consists of computing $\mathbf{u}$ from $\boldsymbol{\omega}$ and then setting $\boldsymbol{\omega} = \mathrm{curl}\mathbf{u}$ (see [16]). This first method has the same scaling as the Poisson solver, while the second is linear since $\mathbf{u}$ is already available in the algorithm. Both can be performed at the same time as the remeshing procedure.

The overall algorithm computational cost scales consequently as $\mathcal{O}(n\log n)$. The following sections will show that this hybrid grid-particle method can be generalized to multiscale computation (section 3) and complex geometry (section 4.4) with the same scalability.

3. Multiscale formulation and scalability of hybrid method. The present section shows how the vortex method using grid-particle coupling, described above, can be set up in the multiscale context and shows that it scales also as $\mathcal{O}(n \log n)$, that is to say, that there is no discrepancy in scalability due to the multiscale formulation.

3.1. Grid hierarchy and boundary condition inheritance. A hierarchical grid structure is created, defined by a recursive sequence of patches, that is to say, by the introduction of one or several child Cartesian grids embedded in their parent grid, starting from a root-grid which is the uniformly discretized domain Ω.

The scale coupling is a two-way strategy specific to stream-vorticity formulation: vorticity is interpolated from fine to coarse grids, and boundary conditions used for Poisson equations are interpolated from coarse to fine grids, as shown on Figure 2. This leads to the computation of stream $\boldsymbol{\psi}$ on a child grid Ω_{child} whose boundary conditions inherited from parent grid as follows:

$$\begin{cases} -\Delta \boldsymbol{\psi} = \boldsymbol{\omega} \quad \text{in } \Omega_{\mathsf{child}} \times]0, T[\\ \boldsymbol{\psi}(\boldsymbol{x}, t) = \boldsymbol{\psi}_{\mathsf{parent}}(\boldsymbol{x}, t) \quad \text{on } \partial\Omega_{\mathsf{child}} \times]0, T[. \end{cases} \tag{3.1}$$

Accuracy, convergence analysis, efficiency, and parallelization of this two-way coupling are discussed in [9].

Implementation of fine-to-coarse update and coarse-to-fine boundary condition transfer has been performed by the adaptive grid refinement in Fortran (AGRIF) library [23]. Syntax analysis software Lex and Yacc, part of AGRIF, allow us to replace float arrays declared (defined for a standard one-grid computation) by multigrid structured variables. In the grid tree, a grid is defined by its index and the index of its

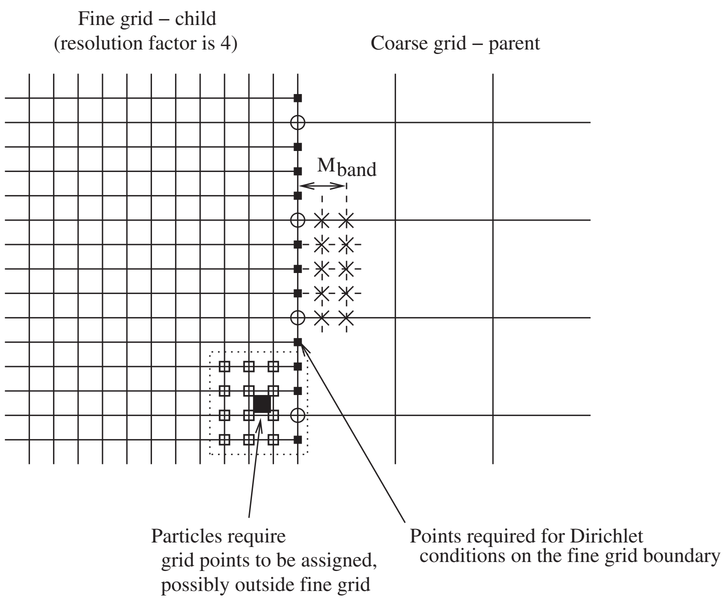
Fine grid – child
(resolution factor is 4)
Coarse grid – parent
M_band
Particles require grid points to be assigned, possibly outside fine grid
Points required for Dirichlet conditions on the fine grid boundary

FIG. 2. *Grid points at coarse level* (○) *are used to assign value of grid points* (■) *at boundaries of the fine grid. Particles are assigned from grid points* (□) *in the neighborhood of the particle by means of interpolation formula* 2.12, *which may involve values outside fine grid: a number of M_{band} grid point layers* (×) *are also interpolated from their coarser level in order to satisfy interpolation and transport computations. Dotted lines stand for the kernel support, used for grid-particle interpolations.*

parent grid, the node index in the parent grid of six box corners, and the three refinement coefficients (for the three direction, in order to be able to consider anisotropic refinement). Such Cartesian child grids also allow the use of fast Poisson solvers.

Since transfer between fine and coarse grids of boundary conditions scales as $\mathcal{O}(n^{2/3})$ and transfer of grid contents scales linearly, the overall multiscale vortex method computational cost is still $\mathcal{O}(n \log n)$. In order to provide sufficient data to subgrids for interpolation, two bands of grid points beyond boundaries are inherited from their parent grid.

Furthermore, particles close to a subgrid boundary may be transported substantially, thus creating an artificial and unwelcome hole of particles close to boundaries. In order to avoid this, vorticity is interpolated from parent on the M_{band} first bands inside the subgrid, after each transport/convection step (see Figure 2). This "feeds" automatically the subgrid with particles containing a consistent value of vorticity, whatever the direction of velocity. Nevertheless, this also introduces a sufficient condition of continuity, that is to say, a condition avoiding creation of a region free of particles, thus avoiding jumps of vorticity:

$$\frac{|u|_\infty \delta t}{h} \leqslant M_{\mathsf{band}}, \tag{3.2}$$

where h is the smallest grid step among the three dimensions of space.

This looks similar to a transport stability condition, but not satisfying it does not lead to any lack of stability. However, in practice, M_{band} is chosen from 4 to 8, which is much less restrictive than a transport CFL. Indeed, transport CFL usually leads to a constant less than 2: for example, second order Runge–Kutta with centered second order finite difference scheme leads to $|u|_\infty \delta t / H \leqslant \sqrt{3}$). Moreover, as one goes deeper in the grid tree, it is noticeable that this condition becomes more and more restrictive as h gets smaller. An alternative would be the use of time interpolation in subgrid boundaries, which has not been investigated in the present work.

3.2. Grid generation and accuracy. Such multigrid simulations can be run either with a stationary grid tree or with a periodic regridding procedure able to build a new set of a grid hierarchy, taking into account the flow evolution. In this last case, the new grid positions (and their number) is implemented in AGRIF and has been based on the success ratio of $\alpha|\boldsymbol{\omega}| + \beta|\mathrm{curl}\boldsymbol{\omega}|$ larger than a given value ($|\cdot|$ denotes the Euclidean norm of $\mathbb{R}^3$).

This is the algorithm developed in [6], [8], originally for hyperbolic equations. This clustering algorithm minimizes the number of subgrids and maximizes the number of grid points above a vorticity value. Consequently, the subgrid boundaries do not cut the vorticity field at its strongest values or strongest gradients, avoiding spurious high frequencies. This has been successfully used for computation of shocks in compressible fluid [7], a kind a simulation very sensitive to the quality of grid cutting, and this feature also applies to elliptic equations. We refer to [9] for further details.

Furthermore, since the boundary conditions are inherited from the coarser grid values, the order of stream computation cannot be higher than the order of convergence at the coarse level. Indeed, accuracy of the fine solution is driven by the estimation of its boundary condition, which is the value of the coarse solution followed by interpolation: if the coarse solution is computed at order p and interpolation is performed at order q, then boundary conditions for the fine grids are set with an error scaling as $\mathcal{O}\left(h^{\mathsf{min}(p,q)}\right)$. In the present configuration, computations at the coarse level are second order (the FishPack solver), and interpolation are third order (using the kernel given on (2.12)). There is consequently no gain of order due to refinement, as in most, if not all, multiresolution methods.

If a refinement criterion like $\alpha|\boldsymbol{\psi}| + \beta|\text{curl}\boldsymbol{\psi}|$ is used instead of the one based on vorticity field $\boldsymbol{\omega}$, a multigrid method very similar to [11] would be obtained, except the fact that they use cell-centered discretization, thus leading to a more complex error analysis (that is to say, requiring a specific formulation to handle the truncation error).

An advantage of using a criterion based on stream such as $\alpha|\boldsymbol{\psi}| + \beta|\text{curl}\boldsymbol{\psi}|$ is that classical theory applies; that is to say, refining on a Sobolev norm $\mathbb{H}^1$ on the solution allows the same convergence as if the grid step of the finest grid was chosen everywhere. Advantages of using a criterion based on vorticity are that there is refinement where $\boldsymbol{\omega}$ and curl$\boldsymbol{\omega}$ are strong. First, this allows us to refine the grid in boundary layers (close to walls). Second, regions where $\boldsymbol{\omega}$ is strong or exhibits strong variations need to be refined in order to compute accurately diffusion $-\nu\Delta\boldsymbol{\omega}$ and stretching $\boldsymbol{\omega}\cdot\nabla\mathbf{u}$.

These two quantities are specific to vortex methods. It has been shown that even with a stream kept at its coarsest level, refining the quantity transported (in our case the vorticity) improves the solution by means of a better conservation of momentum and an improvement of energy spectrum [18]. Furthermore, one has to be clever on fine/coarse interfaces effects, which are able to generate spurious sources. While a solution can be to cancel it with as appropriate single layer potential (see [26]), we have not observed such effects, which could appear only through stretching $\boldsymbol{\omega}\cdot\nabla\mathbf{u}$.

As a first validation, one considers the one-dimensional problem involving the logistic function $\bar{\psi}(x) = (1 + e^{-\alpha x})^{-1}$ over interval $[-1, 1]$, a solution of

$$\begin{cases} -\Delta\psi = f & \text{in } [-1, 1], \\ \psi(x) = \bar{\psi}(x) & \text{for } x = -1 \text{ or } x = 1, \end{cases} \tag{3.3}$$

where $f(x) = -\Delta\bar{\psi}(x) = \alpha^2(1 - e^{-\alpha x})(1 + e^{-\alpha x})^{-3}$. We set $\alpha = 80$ and introduce a subgrid over interval $[-1/8, 1/8]$ with a refinement factor of 8. The global $\mathbb{L}^2$ error on the solution is displayed on Figure 3 with respect to the number of grid points involved in the computation, for the two-level grid tree, the uniformly coarse and uniformly fine grids. Refinement in regions of large variations or values of source f in (3.3) leads to a better convergence than if grids are uniform, whether they are coarse or fine everywhere.

For validation using grid trees deeper than two levels or in dimensions of space larger than one, we refer to the convergence of no-slip condition in the case of the 3D wake developed behind a sphere, described in section 4.3.

3.3. Scalability. The algorithm has been firstly validated on two- and three-dimensional flows using a 3-level grid hierarchy (displayed on Figure 4), with the elliptic vortex used in [28] as an initial condition for 2D computation. The computational box is $[-2.5, 2.5]^d$, where d is the number of spatial dimension, and the viscosity is chosen as $\nu = 10^{-2}$. In the 3D case, the initial condition is a column vortex, that is to say, $\boldsymbol{\omega}(x, y, z) = \omega_{2\text{D}}(x, y)\mathbf{e}_z$, whose support is restricted in the region $-2 < z < 2$.

In order to check the scalability within a wide range of grid points, one considers a 3D ellipsoid divergence-free vortex defined from the stream by

$$\boldsymbol{\omega} = \text{curl}^2\boldsymbol{\psi} \quad \text{with} \quad \boldsymbol{\psi}(\boldsymbol{x}) = \exp\left(\frac{1}{2}\boldsymbol{x}^T\mathbf{S}^{-2}\boldsymbol{x}\right)\mathbf{e}_z, \quad \boldsymbol{x} \in \mathbb{R}^3, \tag{3.4}$$

where $\mathbf{S}$ is a diagonal matrix of coefficients $(0.4, 0.8, 0.8)$. In this case, the Euler equation is considered (i.e., $\nu = 0$), in order to have a flow generating small vorticity

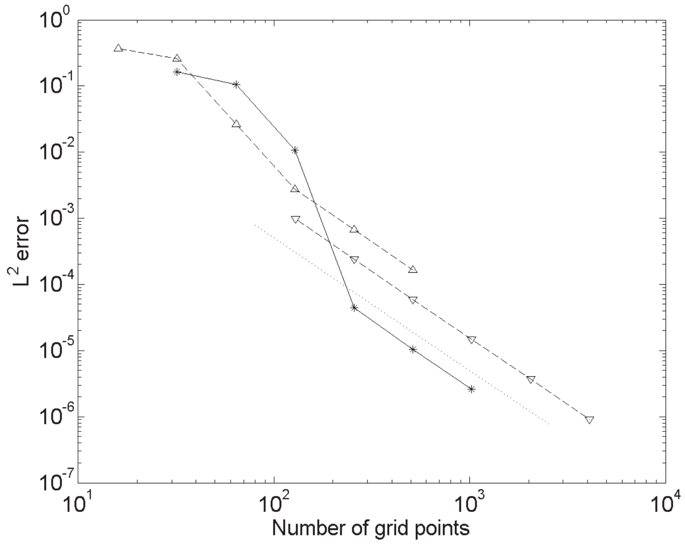

FIG. 3. *Global $\mathbb{L}^2$ error on logistic function, solution of* (3.3) *with respect to the number of grid points involved in the computation, for the two-level grid tree* ($*$)*, and the uniformly coarse* ($\triangle$) *and uniformly fine grids* ($\triangledown$)*. Refinement factor is* 8*, and curves for uniform fine and coarse grids have a ratio of* $\sqrt{8}$ *due to the* $\mathbb{L}^2$ *norm expression. Dotted line is second order convergence.*

structures and thus a complex grid hierarchy, able to exhibit the algorithm behavior for large number of grid points. The computational box is $[-5, 5]^3$, with a resolution of 64^3 and 96^3 for the root-grid, the time step set at $\delta t = 0.05$, and the grid refinement factor between child and parent grids set to 4.

The resulting CPU time with respect to the number of grid points, not constant since the subgrid tree is refreshed every 5 iterations, is displayed on Figure 5.

The 3-level multigrid hierarchy and isosurfaces of the solution are displayed in Figure 6.

First, this figure shows that scalability is linear and validates the theoretical efficiency $\mathcal{O}(n \log n)$ established in the previous section. The linear scalability is established in the range of 10^6 to $3\,10^8$ grid points (or particles).

Second, when small vortex structures are generated as time increases, the simulation based on a 64^3 root-grid resolution requires us to create more subgrids in order to capture the dynamics of small vortex. The number of grid points in this case consequently reaches higher values than for the simulation based of 96^3 points of root-grid. It follows from this fact that despite the strong adaptivity of multiscale simulation, the root-grid must nevertheless be chosen carefully in order to avoid useless computational time.

4. Multiscale vortex method in complex geometry.

When complex geometries are involved, the no-slip condition $\mathbf{u} = 0$ at walls (physical boundaries) $\Gamma \subset \partial\Omega$ can be declined into two kinds of boundary conditions:

- The no-flow-though condition $\mathbf{u} \cdot \mathbf{n} = 0$ (the fluid does not flow through the boundary, whether it is viscous or not);
- The tangential no-slip conditions $\mathbf{u} \cdot \boldsymbol{\tau}_i = 0$, $i = 1, 2$, indexing the two vectors tangent to Γ (the fluid, if viscous, is adherent to the wall).

While in velocity-pressure, these two boundary conditions are usually treated at the same time as one; they are of very different nature in velocity-vorticity formulation.

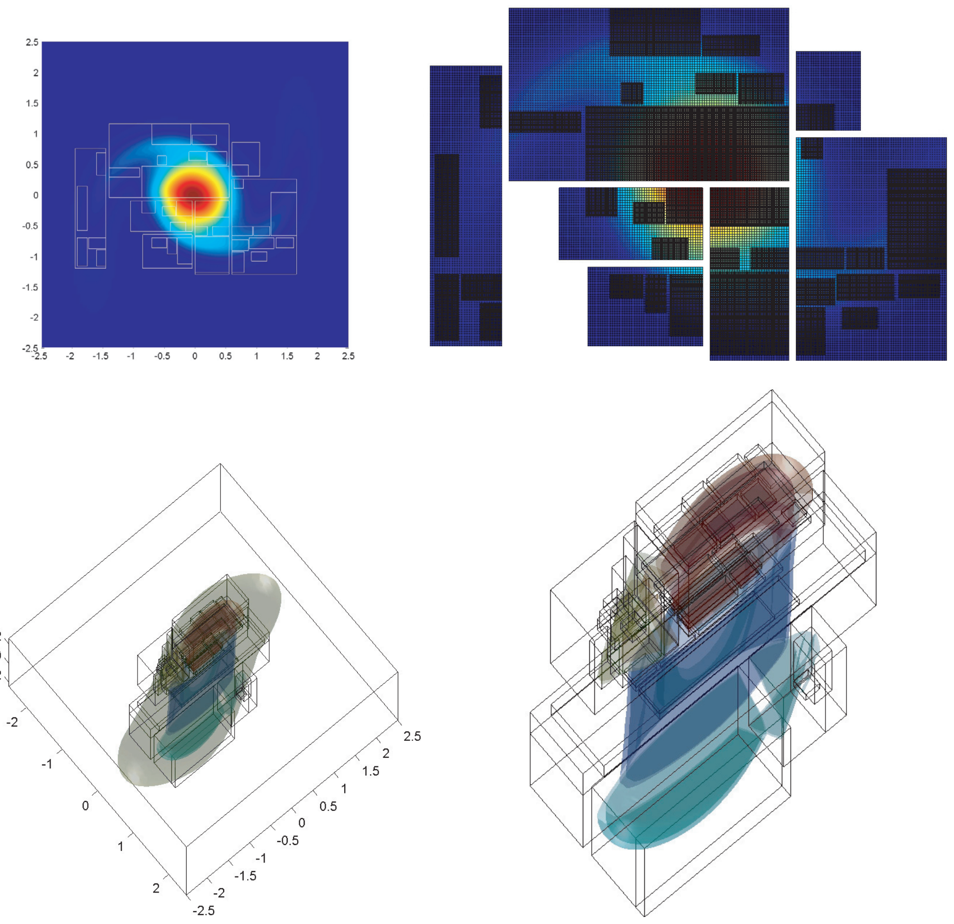

FIG. 4. *2D and 3D (respectively, top and bottom pictures), 3-level multiscale computation of the dynamic of an elliptic vortex, with refinement ratio of 4 and refinement criterion based on $\|\nabla\boldsymbol{\omega}\|$. For 2D simulation, color is the vorticity level. Iso-surfaces of vorticity are displayed for the 3D simulation, and color is the grid index (in order to identify easily different patches). Left pictures show vorticity in the whole computational domain, and right pictures show level 2 and 3 subgrids.*

Let Ω denote a Cartesian box that includes a body $\mathcal{B}$ of boundary $\Gamma = \partial\mathcal{B}$ with the fluid domain $\Omega^* = \Omega\backslash\overline{\mathcal{B}}$ around, in which the flow is computed. The goal, in order to handle complex geometry, is to find appropriate boundary conditions on stream, find an alternative formula providing velocity, or add an adequate source of vorticity (right-hand side of the Navier–Stokes equations).

The few sections thereafter give a description of such methods, for vortex methods, in order to satisfy these boundary conditions. A discussion on the best strategy for multiscale computing is also provided.

4.1. Boundary integral methods and implicit immersed boundaries. In this section, focus is made on the computation of a velocity field, subject only to no-flow-through condition $\mathbf{u}\cdot\mathbf{n} = 0$ on body boundary $\Gamma = \partial\mathcal{B}$.

Getting the velocity from the stream by means of $\mathbf{u} = \text{curl}\boldsymbol{\psi}$ and the Poisson equation (2.9) requires us to be able to satisfy both $\text{div}\boldsymbol{\psi} = 0$ and $\text{curl}\boldsymbol{\psi}\cdot\mathbf{n} = 0$ at wall Γ.

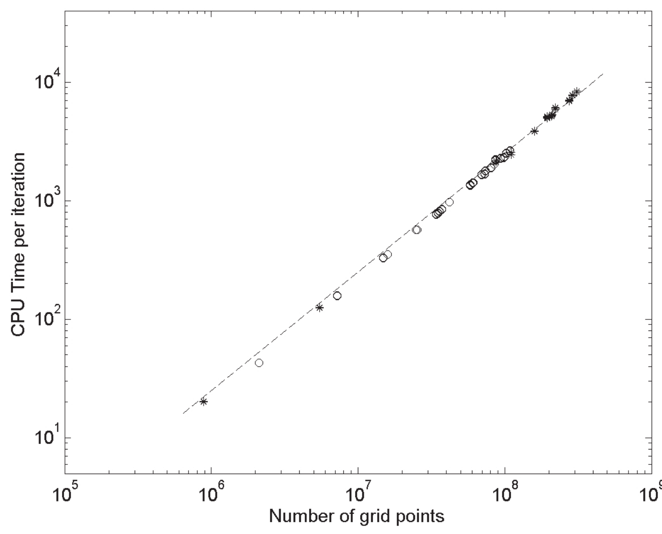

FIG. 5. *CPU time (in seconds) with respect to the total number of grid points.* $*$ *and* $\circ$ *correspond, respectively, to root-grids of resolutions* 64^3 *and* 96^3*.* $--$ *is* $\mathcal{O}(n)$ *scalability.*

Nevertheless, these two conditions are coupled as soon as geometry is more complex than a flat wall or cylindrical body. It follows that in arbitrary geometry, the vectorial Poisson equation (2.9) has its components coupled through its boundary conditions: this leads to an elliptic problem whose discretization is a matrix of large dimension and nonstandard kind (that is to say, forbidding the use of fast PDE solvers). The problem consequently reaches the computational complexity obtained if the domain was meshed around the body and the matrix assembled, thanks to a finite element discretization. Such a strategy would ruin the efficiency of the vortex method described.

In order to avoid this coupling, it has been shown (see [20] for instance) that it is useful to extend formula (2.10) to

$$\mathbf{u} = \mathrm{curl}\boldsymbol{\psi} - \nabla\phi, \tag{4.1}$$

where $\boldsymbol{\psi}$ is solution of (2.9) on the Cartesian box Ω, i.e., with boundary conditions only at the box faces as if the body $\mathcal{B}$ did not exist. The geometry is taken into account by means of the potential stream ϕ, solution of

$$\begin{cases} -\Delta\phi = 0 \quad \text{in } \Omega^*, \\ \dfrac{\partial\phi}{\partial\mathbf{n}} = \mathrm{curl}\boldsymbol{\psi}\cdot\mathbf{n} \quad \text{on } \Gamma. \end{cases} \tag{4.2}$$

When the computation deals with unbounded domains, that is to say, $\Omega^* = \mathbb{R}^3\backslash\mathcal{B}$, the usual boundary integral method consists of expressing the solution of the equation above under its fundamental form

$$\phi(\boldsymbol{x}) = \int_\Gamma q(\boldsymbol{\xi})K(\boldsymbol{x}-\boldsymbol{\xi})\mathrm{d}\sigma(\boldsymbol{\xi}), \tag{4.3}$$

where $K(\boldsymbol{x}) = (4\pi|\boldsymbol{x}|)^{-1}$ is the Green kernel and σ is the Lebesgue measure induced on Γ. Its integral density $q : \Gamma \to \mathbb{R}$ is then solution of the integral equation

$$-\frac{q(\boldsymbol{x})}{2} + \int_\Gamma \mathbf{n}\cdot\nabla K(\boldsymbol{x}-\boldsymbol{\xi})\,q(\boldsymbol{\xi})\,\mathrm{d}\sigma(\boldsymbol{\xi}) = \mathrm{curl}\boldsymbol{\psi}(\boldsymbol{x})\cdot\mathbf{n}(\boldsymbol{x}). \tag{4.4}$$

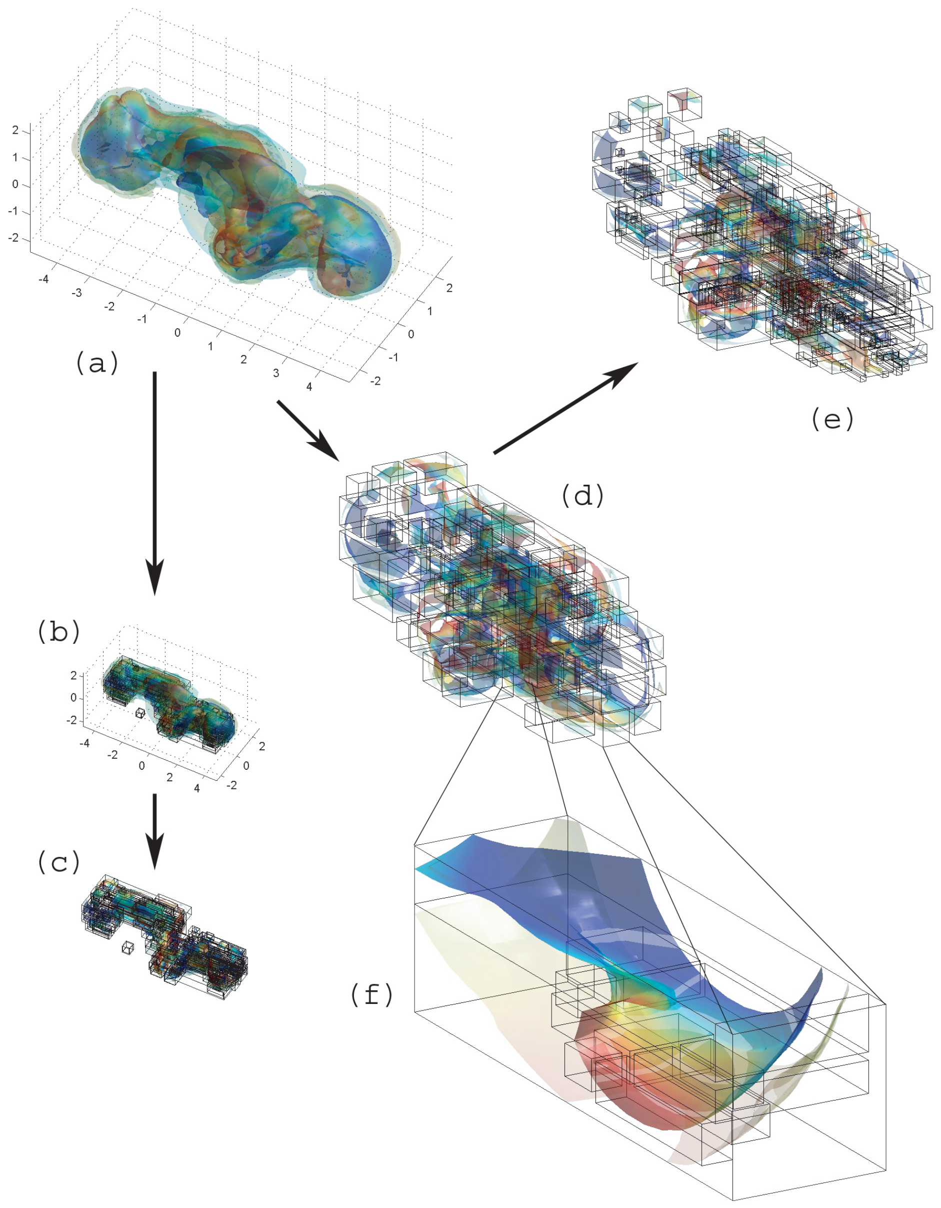

FIG. 6. *Multiscale grid hierarchy for ellipsoid vortex dynamics* (a) *Snapshot of iso-vorticity surface at time* $t = 12$ *(root-grid, level* 1*);* (b) *Same as* (a) *with all subgrid positions (all levels);* (c) *All subgrids;* (d) *All level* 2 *subgrids (children grids of root-grid);* (e) *All level* 3 *subgrids (children grids of level* 2 *grids);* (f) *Zoom on one particular level* 2 *grid with its level* 3 *children.*

This integral equation can be solved, for example, by using a boundary element method (using a mesh of Γ) leading to a full linear system, sizing as the mesh of Γ, and thus quite efficient. Nevertheless, the main drawbacks of this method are the computational cost for building the potential ϕ and the nonconformity of meshes between the surface neighborhood and particle lattice.

Indeed, the computational cost for building the potential ϕ from the integral density q over the whole domain has the same complexity as Biot–Savart laws (2.7), (2.8) when evaluated on a surface instead of a volume. Despite its heaviness, this method has been successfully used in a lightly different context for computation of three-dimensional flows in moderate domain size [15]: formula (2.8) has been used to carry out values of outer boundary conditions on a computational box.

Moreover, when using nonconformal meshes, that is to say, when the mesh of Γ does not coincide to the grid points on which ϕ is defined, a major discrepancy is observed on the accuracy of ϕ. A high order boundary element method is then required in order to resolve the singularity correctly. This implies that if a method accelerating the computation of sum (2.8) is used, using an integral boundary method in order to compute the velocity field $\mathbf{u}$ can be possible (though hardly possible), but the non-conformal context between mesh of Γ and grid points of Ω remains a major drawback of using the integral technique.

An alternative to the boundary integral method, formally equivalent but numerically more appropriate to the present configuration, is the implicit immersed boundary method described in [38]: one considers (4.2) with $\Omega^* = \Omega \setminus \mathcal{B}$ and Ω a Cartesian box. The integral density q, defined on Γ, induces the generalized function

$$T_q(\zeta) = \int_\Gamma \zeta(\boldsymbol{\xi}) q(\boldsymbol{\xi}) \mathrm{d}\sigma(\boldsymbol{\xi}),$$

where $\zeta \in \mathcal{D}(\Omega)$ is a test function, so that solving (4.2) is equivalent to solve

$$\begin{cases} -\Delta\phi = T_q & \text{in } \Omega, \\ \dfrac{\partial\phi}{\partial\mathbf{n}} = \operatorname{curl}\boldsymbol{\psi}\cdot\mathbf{n} & \text{on } \partial\Omega. \end{cases} \tag{4.5}$$

Let Ξ be the immersion integro-differential operator that takes integral density q and gives the normal derivative of ϕ on the fluid side Ω^* of Γ. This operator is defined on $\mathbb{H}^{1/2}(\Gamma)$ onto itself (when body $\mathcal{B}$ is convex and with a light discrepancy on regularity otherwise, see [38], for instance).

Solving $\Xi(q) = \operatorname{curl}\boldsymbol{\psi}\cdot\mathbf{n}$ is formally equivalent to solving the integral equation (4.4), but it can easily be defined in a tubular neighborhood of Γ by $\widetilde{\Xi}$:

$$\widetilde{\Xi}(f) = \Lambda^\epsilon * (\Xi \circ \varrho \circ (\Lambda^\epsilon * f)), \tag{4.6}$$

where ϱ is the trace on Γ (that is to say, the restriction) and Λ^ϵ is the local interpolation kernel (2.12). This allows the definition of immersion operator Ξ on grid points close to Γ, natural since (4.5) is numerically solved on the grid discretizing the Cartesian box Ω.

This leads to a very fast algorithm to compute the velocity from the vorticity, satisfying the no-through-flow boundary condition $\mathbf{u}\cdot\mathbf{n} = 0$: one gets 4 Poisson equations (2.9), (4.5) in Cartesian box Ω:

$$-\Delta\psi_x = \omega_x, \quad -\Delta\psi_y = \omega_y, \quad -\Delta\psi_z = \omega_z, \quad -\Delta\phi = T_{\Xi^{-1}(\operatorname{curl}\boldsymbol{\psi}\cdot\mathbf{n})} \tag{4.7}$$

with homogeneous boundary conditions on faces of the box Ω, which can be solved by a last solver in Cartesian coordinates, such as FishPack or MudPack.

Nevertheless, this method requires us to build the matrix which is the discretization of operator Ξ, whose size scales as the number of points discretizing the surface Γ. On the multi-scale point of view, the method is consequently very efficient for a fixed grid hierarchy (i.e., still scaling as $\mathcal{O}(n \log n)$), but in practice requires too much time to build the matrix of Ξ when the grid tree is time dependent. Indeed, the number of points of Γ scales as $m = n^{2/3}$ (n being the total number of grid points in box Ω), while the matrix of Ξ inversion scales as $\mathcal{O}(m^3) = \mathcal{O}(n^2)$: it is interesting when done once a run, since Ξ^{-1} is reused every time step, but the method is not suitable if Ξ^{-1} has to be computed often, at every change of grid tree.

4.2. Chorin–Lighthill method. To the opposite of the last section, one focuses on how to enforce the full no-slip condition $\mathbf{u} = 0$ on Γ, that is to say, tangential boundary conditions on velocity. The idea is to split the Navier–Stokes equation into a convection time step (2.5), performed in complex geometry by one of the methods described in the last section, and a diffusion time step (2.6). The diffusion step computes viscous effects, responsible of adherence condition and creation of a boundary layer, generating a flux of vorticity at walls. Chorin's [14] and Lighthill's [31] methods consist of setting appropriate singular sources of vorticity on the wall and diffusing it in the fluid.

Chorin's algorithm consists of replacing, over a time step $[t_0, t_0 + \delta t]$, the implicit equation (2.6) into a standard heat equation with Robin–Fourier boundary (adjusted Neumann) condition

$$\begin{cases} \dfrac{\partial \boldsymbol{\omega}}{\partial t} - \nu \Delta \boldsymbol{\omega} = 0 \quad \text{in } \Omega \times]t_0, t_0 + \delta t[, \\ \mathcal{L}\boldsymbol{\omega} = \mathbf{g} \quad \text{on } \partial\Omega \times]t_0, t_0 + \delta t[, \\ \boldsymbol{\omega}(\boldsymbol{x}, t_0) = \boldsymbol{\omega}^c(\boldsymbol{x}) \quad \text{on } \Omega, \end{cases} \tag{4.8}$$

where $\mathcal{L}$ is a first order vectorial differential operator whose components on the tangent bundle can be locally written as $\mathcal{L}_i = \kappa_i \mathrm{Id} + \partial/\partial \mathbf{n}$, κ_i being the curvature of Γ in direction $\boldsymbol{\tau}_i$. The initial condition $\boldsymbol{\omega}^c$ is the value obtained at the end of the convection step.

In practice, (4.8) is split by linearity into

$$\begin{cases} \dfrac{\partial \boldsymbol{\omega}}{\partial t} - \nu \Delta \boldsymbol{\omega} = 0 \quad \text{in } \Omega \times]t_0, t_0 + \delta t[, \\ \boldsymbol{\omega}(\boldsymbol{x}, t_0) = \boldsymbol{\omega}^c(\boldsymbol{x}) \quad \text{on } \Omega, \\ + \text{ arbitrary boundary condition at walls}, \end{cases} \tag{4.9}$$

with a posteriori setting $\bar{\mathbf{g}} = \mathcal{L}\boldsymbol{\omega}$, $\bar{\mathbf{u}}$ the consequent spurious velocity [19], and

$$\begin{cases} \dfrac{\partial \boldsymbol{\omega}}{\partial t} - \nu \Delta \boldsymbol{\omega} = 0 \quad \text{in } \Omega \times]t_0, t_0 + \delta t[, \\ \mathcal{L}\boldsymbol{\omega} = \mathbf{g} - \bar{\mathbf{g}} \quad \text{on } \partial\Omega \times]t_0, t_0 + \delta t[, \\ \boldsymbol{\omega}(\boldsymbol{x}, t_0) = 0 \quad \text{on } \Omega. \end{cases} \tag{4.10}$$

The sum of solutions of (4.9), (4.10) is a solution of (4.8).

The only difficulty so far is to have an accurate estimation of $\mathbf{g}$ that makes the velocity field satisfy the no-slip boundary condition. While aspects of the 2D problem, which does not involve curvature, have been investigated in the 1990s [27], consistent estimations of $\mathbf{g}$ for the 3D problem and accuracy analysis (in both two dimensions and three dimensions) have been proposed only recently [36].

A wide range of numerical methods are available to solve (4.9) since no boundary condition is required. Moreover, the solution of (4.10) can be written under its fundamental formulation [24] involving the Gaussian heat kernel (with only boundary source due to the zero initial condition)

$$\boldsymbol{\omega}(\boldsymbol{x}, t) = \int_{t_0}^{t} \int_{\Gamma} \boldsymbol{\mu}(\boldsymbol{\xi}, s) G_{\boldsymbol{x},t}(\boldsymbol{\xi}, s) \mathrm{d}\sigma(\boldsymbol{\xi}) \mathrm{d}s, \tag{4.11}$$

where

$$G_{\boldsymbol{x},t}(\boldsymbol{\xi}, s) = \exp\left(-\frac{|\boldsymbol{x} - \xi|_2^2}{4\nu(t - s)}\right) \tag{4.12}$$

is the Gaussian heat kernel and the density $\boldsymbol{\mu}$ is solution of an integro-differential equation

$$-\frac{1}{2}\boldsymbol{\mu}(\boldsymbol{x}, t) + \nu \int_{t_0}^{t} \int_{\Gamma} \mathbf{n}(\boldsymbol{x}) \cdot \nabla G_{\boldsymbol{x},t}(\boldsymbol{\xi}, s)\boldsymbol{\mu}(\boldsymbol{\xi}, s)\mathrm{d}\sigma(\boldsymbol{\xi})\mathrm{d}s = \mathbf{g}(\boldsymbol{x}, t) - \bar{\mathbf{g}}(\boldsymbol{x}, t). \tag{4.13}$$

On the one hand, the flux of vorticity at wall is

$$\mathbf{g}(\boldsymbol{x}, t) = \mathbf{n} \times \frac{\partial \bar{\mathbf{u}}}{\partial t}(\boldsymbol{x}, t), \tag{4.14}$$

which is a natural extension from [14] and [19]. On the other hand, it has been shown in [36] that a consistent approximation (up to order 5/2) of the integral density $\boldsymbol{\mu}$ is given directly by

$$\boldsymbol{\mu}(\boldsymbol{x}, t) \cdot \boldsymbol{\tau}_i(\boldsymbol{x}) \simeq \frac{-2(\mathbf{g}(\boldsymbol{x}, t) - \bar{\mathbf{g}}(\boldsymbol{x}, t)) \cdot \boldsymbol{\tau}_i(\boldsymbol{x})}{1 - 2(\bar{\kappa}(\boldsymbol{x}) - \kappa_i(\boldsymbol{x}))\sqrt{\nu(t - t_0)/\pi}}, \tag{4.15}$$

where $\bar{\kappa}$ is the mean arithmetic curvature.

Such a formula, when compared to the vortex method without boundary, requires only the computation of $\bar{\mathbf{u}}$ and the evaluation of formula (4.11), which scales as $\mathcal{O}(n^{2/3})$, due to the numerical compactness of the Gaussian heat kernel support. Such an efficiency has been successfully used in the past for 3D flow control (see [34] and [37], for instance). One can notice that formula (4.15) can be dramatically improved by performing a symbolic integration on panels (see the part of [33] related to no-slip conditions), with or without the use of an adjacency matrix in order to compute the density leading to algebraic no-slip flow.

Consequently, when coupling together the implicit immersed boundary method described above in section 4.1 and the vortex flux at wall described in the present section, the vortex method in complex geometry scales with the same efficiency as without geometry, that is to say, as $\mathcal{O}(n \log n)$, whether it is multiscale or not. Nevertheless, in this case, the limitations exhibited in section 4.1 still apply, which means that this method is efficient for a fixed hierarchy of grid. For a time-dependent grid tree, one may prefer the penalization described thereafter.

4.3. Penalization. An alternative to the methods presented above, in order to deal with complex geometry, is the penalization introduced and analyzed in [3] and [12] and revisited recently in [16]. This consists of penalizing the Navier–Stokes equations (2.2) by a term hold with the characteristic function $\chi_{\mathcal{B}}$ of body $\mathcal{B}$:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \Delta \mathbf{u} + \lambda \chi_{\mathcal{B}} \mathbf{u} = \frac{\mathbf{f} - \nabla p}{\rho}, \tag{4.16}$$

which gives, by taking the curl of this equation

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \mathbf{u} \cdot \nabla \boldsymbol{\omega} - \boldsymbol{\omega} \cdot \nabla \mathbf{u} - \nu \Delta \boldsymbol{\omega} = -\lambda \chi_{\mathcal{B}} \boldsymbol{\omega} - \lambda \delta_{\Gamma} \mathbf{n} \times \mathbf{u}, \tag{4.17}$$

where δ_Γ is the Dirac function of surface Γ, that is to say, the derivation of the characteristic function $\chi_{\mathcal{B}}$.

In practice, a three-step splitting convection/diffusion/penalization is performed. Surface Γ is discretized by a P0 mesh, that is to say, a set of points $\boldsymbol{\gamma}_k$ associated to a surface element s_k, so that the discretization of term $\delta_\Gamma \mathbf{n} \times \mathbf{u}$ is $\sum_k \mathbf{n} \times \mathbf{u}(\boldsymbol{\gamma}_k) s_k$, itself interpolated on grid points, thanks to transfer formula (2.12). Furthermore, it has been observed that a good choice for the penalization parameter is $\lambda = 2/\delta t$.

The main advantage of this method is that it is not a grid-dependent partial differential equation to solve, and hence a natural feature for multiscale computing. As for its drawbacks, there is eventually a penalization parameter λ to tune and some sensibility of solution accuracy to this parameter, but the choice $\lambda = 2/\delta t$ has been satisfactorily used in [16] and in our computations.

4.4. Scalability of multiscale VIC using penalization. Given advantages and drawbacks of the methods listed in the previous section, the penalization method has been chosen to perform 3D flow computation around a sphere with a time-dependent grid hierarchy.

Indeed, the penalization algorithm has been used to compute the wake behind a sphere of radius $R = 1$, at a Reynolds number $Re_R = U_\infty R/\nu = 500$. The far field velocity is chosen as $U_\infty = 1$, and the root-grid as

$$\Omega = [-5, 25] \times [-5, 5] \times [-5, 5]$$

discretized by $192 \times 64 \times 64$ grid points, such as when the grid step is constant in all directions $h = \delta x = \delta y = \delta z$. The time step $\delta t = 0.1$, and the grid hierarchy is updated every five time steps in order to follow the vorticity evolution. Two kinds of multiscale configurations have been used:

1. level 2 grid hierarchy with refinement coefficient 4;
2. level 3 grid hierarchy with refinement coefficient 2,

so that the maximum refinement factor between root-grid and finest child grids is four.

The underresolution nondimensional parameter $\delta^+ = \sqrt{4\nu\delta t}/h$ is slightly larger than 1 so that the simulation is fully resolved on fine grids. Snapshots for both the configurations are displayed on Figure 7. As expected, refinement is provided close to the body were vorticity reaches its highest values and gradients.

Since the singular terms of (4.17) require only an additional computation of velocity field, the overall algorithm still scales as $\mathcal{O}(n \log n)$, as shown on Figure 8 for both the configurations.

Furthermore, accuracy of penalization is not well established, to the opposite of boundary integral method, immersed boundaries, and vorticity flux at walls. In order to provide a fair comparison to the other methods listed above, Figure 9 shows that penalization is second order in space by plotting the residual velocity on the body, that is to say, the $\mathbb{L}^2(\Gamma)$ norm of velocity as $t \to \infty$.

5. Conclusion. In this article, the computational cost of the vortex method, coupling particles and grids, has been checked to scale as $\mathcal{O}(n \log n)$, n being the number of grid points and/or particles involved in the simulation. The method has been shown to keep this scalability for a multiscale formulation of such hybrid vortex methods.

Furthermore, a discussion has been made on the methodology in order to handle complex geometries. The discussion has compared boundary integral methods and implicit immersed boundary methods in order to satisfy a no-flow-through condition, Chorin's algorithm for the full no-slip condition related to viscous flows, and the penalization method for both kinematic boundary conditions. It appears that coupling

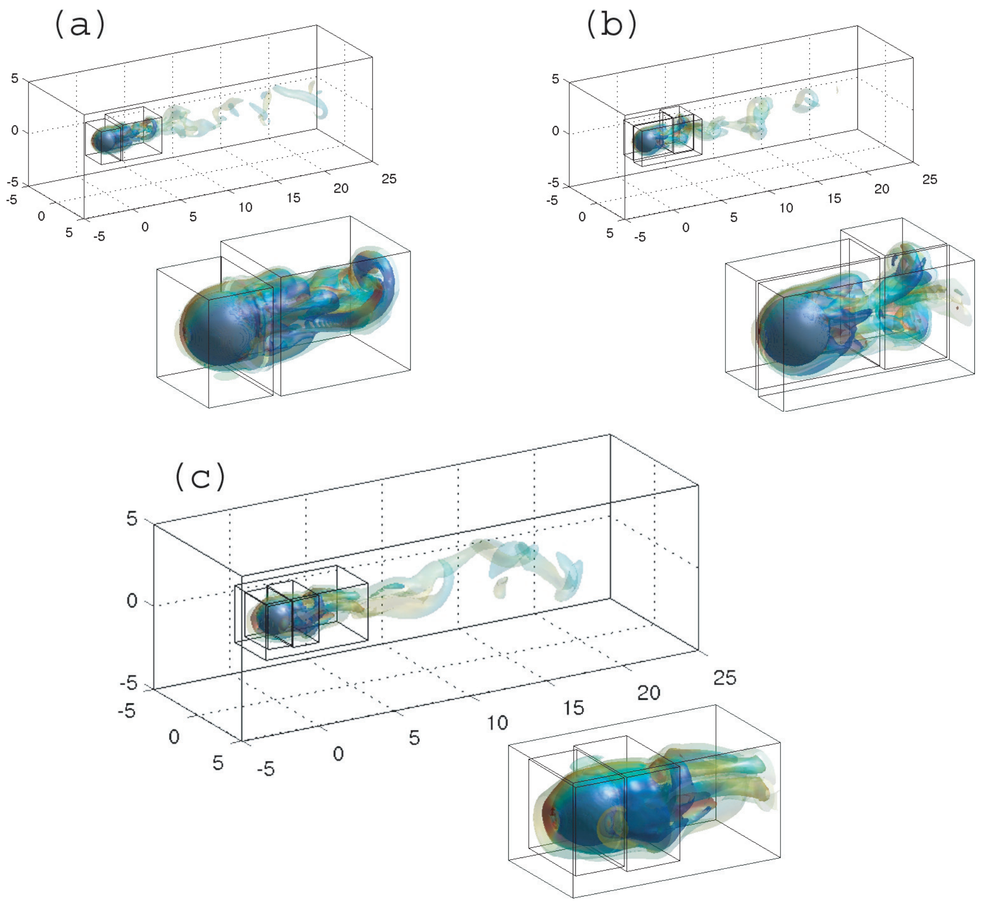

FIG. 7. *Surfaces of iso-vorticity for the wake developed behind a sphere of radius* 1*: For level* 2 *grid hierarchy with refinement coefficient* 4 *(at time* $t = 110$ (a) *and* $t = 126$ (b)*), and for level* 3 *grid hierarchy with refinement coefficient* 2 (c).

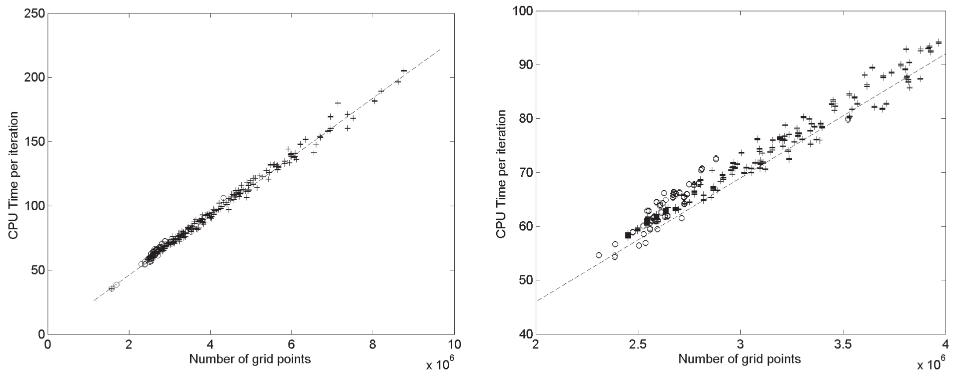

FIG. 8. *CPU time (seconds) versus the number of grid points for the sphere wake simulation;* (+)*: Level* 2 *grid hierarchy with refinement coefficient* 4*;* (∘)*: Level* 3 *grid hierarchy with refinement coefficient* 2*; (- -):* $\mathcal{O}(n)$ *scalability.*

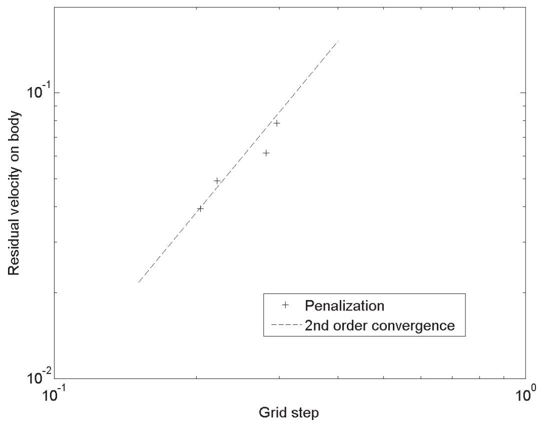

FIG. 9. *Residual velocity, plotted with respect to grid step, shows that the penalization method is second order.*

implicit immersed boundary and Chorin's algorithm may be satisfactory when the hierarchical grid structure is not time dependent, but the penalization method is better (that is to say, more affordable computationally) when time adaptivity of grids is required.

Finally, the scalability as $\mathcal{O}(n \log n)$ is shown to be still valid, theoretically and practically, for a multiscale hybrid grid-particle vortex method in complex geometry.

REFERENCES

[1] J. ADAMS, *MUDPACK: Multigrid Fortran software for the efficient solution of linear elliptic partial differential equations*, Appl. Math. Comput., 34 (1989), pp. 113–146.

[2] C. ANDERSON AND C. GREENGARD, *On vortex methods*, SIAM J. Numer. Anal., 22 (1985), pp. 413–440.

[3] P. ANGOT, C.-H. BRUNEAU, AND P. FABRIE, *A penalization method to take into account obstacles in incompressible viscous flows*, Numer. Math., 81 (1999), pp. 497–520.

[4] J. T. BEALE AND A. MAJDA, *Rates of convergence for viscous splitting of the Navier-Stokes equations*, Math. Comp., 37 (1981), pp. 243–259.

[5] M. BERGDORF, G.-H. COTTET, AND P. KOUMOUTSAKOS, *Multilevel adaptive particle methods for convection-diffusion equations*, Multiscale Model. Simul., 4 (2005), pp. 328–357.

[6] M. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[7] M. BERGER AND P. COLELLA, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.

[8] M. BERGER AND I. RIGOUTSOS, *An algorithm for point clustering and grid generation*, IEEE Trans. Syst. Man. Cybernet., 21 (1992), pp. 1278–1286.

[9] E. BLAYO AND L. DEBREU, *Nesting ocean models*, in Ocean Weather Forecasting—An Integrated View of Oceanography, E. P. Chassignet and J. Verron, eds, Springer, Netherlands, 2006.

[10] M. BRADY, Ph.D. dissertation, Caltech, Pasadena, CA, 1998.

[11] J. D. BROWN AND L. L. LOWE, *Multigrid elliptic equation solver with adaptive mesh refinement*, J. Comput. Phys., 209 (2005), pp. 582–598.

[12] C-H. BRUNEAU, *Boundary conditions on artificial frontiers for incompressible and compressible Navier-Stokes equations*, M2AN Math. Model. Numer. Anal., 34 (2000), pp. 303–314.

[13] 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. Engrg., 197 (2008), pp. 1296–1304.

[14] A. J. CHORIN, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785–796.

[15] R. COCLE, G. WINCKELMANS, AND G. DAENINCK, *Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations*, J. Comput. Phys., 227 (2007), pp. 2263–2292.

[16] M. Coquerelle and G-H. Cottet, *A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies*, J. Comput. Phys., 227 (2008), pp. 9121–9137.

[17] J. P. Christiansen, *Numerical solution of hydrodynamics by the method of point vortices*, J. Comput. Phys., 13 (1973), pp. 363–379.

[18] G.-H. Cottet, G. Balarac, and M. Coquerelle, *Sub-grid particle resolution for the turbulent transport of a passive scalar*, in Proceedings of the 12th Euromech Conference, Marburg, 2009.

[19] G.-H. Cottet and P. D. Koumoutsakos, *Vortex Methods, Theory and Practice*, Cambridge University Press, London, 2000.

[20] G.-H. Cottet and P. Poncet, *Advances in direct numerical simulations of three-dimensional wall-bounded flows by particle in cell methods*, J. Comput. Phys., 193 (2003), pp. 136–158.

[21] B. Couët, O. Buneman, and A. Leonard, *Simulation of three-dimensional flows with vortex-in-cell methods*, J. Comput. Phys., 39 (1981), pp. 305–308.

[22] P. Degond and S. Mas-Gallic, *The weighted particle method for convection-diffusion equations*, Math. Comp., 53 (1989), pp. 485–526.

[23] L. Debreu, C. Vouland, and E. Blayo, *AGRIF: Adaptive grid refinement in Fortran*, Comput. Geosci., 34 (2008), pp. 8–13.

[24] A. Friedman, *Partial Differential Equations of Parabolic Type*, Prenctice-Hall, Englewood Cliffs, NJ, 1964.

[25] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulation*, J. Comput. Phys., 73 (1987), pp. 325–348.

[26] J. Huang and L. Greengard, *A fast direct solver for elliptic partial differential equations on adaptively refined meshes*, SIAM J. Sci. Comput., 21 (1999), pp. 1551–1566.

[27] P. D. Koumoutsakos, A. Leonard, and F. Pepin, *Boundary conditions for viscous vortex methods*, J. Comput. Phys., 113 (1994), pp. 52–61.

[28] P. Koumoutsakos, *Inviscid axisymmetrization of an elliptical vortex*, J. Comput. Phys., 138 (1997), pp. 821–857.

[29] P. Koumoutsakos, *Multiscale flow simulations using particles*, Annu. Rev. Fluid Mech., 37 (2005), pp. 457–487.

[30] P. Li, H. Johnston, and R. Krasny, *A Cartesian treecode for screened Coulomb interactions*, J. Comput. Phys., 228 (2009), pp. 38–58.

[31] M. J. Lighthill, *Boundary Layer Theory*, Oxford University Press, London, 1963.

[32] J. J. Monaghan, *Extrapolating B-splines for interpolation*, J. Comput. Phys., 60 (1985), pp. 253–262.

[33] P. Ploumhans and G. S. Winckelmans, *Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry*, J. Comput. Phys., 165 (2000), pp. 354–406.

[34] P. Poncet, *Topological aspects of the three-dimensional wake behind rotary oscillating circular cylinders*, J. Fluid Mech., 517 (2004), pp. 27–53.

[35] P. Poncet, *Finite difference stencils based on particle strength exchange schemes for improvement of vortex methods*, J. Turbul., 7 (2006), pp. 1–24.

[36] P. Poncet, *Analysis of direct three-dimensional parabolic panel methods*, SIAM J. Numer. Anal., 45 (2007), pp. 2259–2297.

[37] P. Poncet, R. Hildebrand, G-H. Cottet, and P. Koumoutsakos, *Spatially distributed control for optimal drag reduction in cylinder wakes*, J. Fluid Mech., 599 (2008), pp. 111–120.

[38] P. Poncet, *Analysis of an immersed boundary method for three-dimensional flows in vorticity formulation*, J. Comput. Phys., 228 (2009), pp. 7268–7288.

[39] J. K. Salmon and M. S. Warren, *Skeletons from the treecode closet*, J. Comput. Phys., 111 (1994), pp. 136–155.

[40] I. F. Sbalzarini, J. H. Walther, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, and P. Koumoutsakos, *PPM—A highly efficient parallel particle-mesh library for the simulation of continuum systems*, J. Comput. Phys., 215 (2006), pp. 566–588.

[41] P. Swarztrauber and R. Sweet, *Efficient FORTRAN Subprograms for the Solution of Elliptic Partial Differential Equations*, National Center for Atmospheric Research, Technical Note TN/IA-109, Boulder, CO, 1975.

[42] R. Sweet, *A parallel and vector variant of the cyclic reduction algorithm*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 761–766.