SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1655–1674

BLENDING FINITE-DIFFERENCE AND VORTEX METHODS FOR INCOMPRESSIBLE FLOW COMPUTATIONS[*]

M. L. OULD-SALIHI[†], G.-H. COTTET[†], AND M. EL HAMRAOUI[†]

Abstract. We describe and illustrate numerical procedures that combine grid and particle solvers for the solution of the incompressible Navier–Stokes equations. These procedures include vortex in cell (VIC) and domain decomposition schemes. Numerical comparisons with pure finite-difference methods demonstrate the effectiveness of these techniques for various flow geometries, bounded or unbounded.

Key words. vortex methods, finite-difference methods, Navier–Stokes equations, domain decomposition methods

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

PII. S1064827599350769

1. Introduction.

The idea of coupling finite-difference and particle methods is rather common in the field of Navier–Stokes and kinetic equations. The motivation for using particles is to reduce the dimensionality of the phase space and to deal in an elegant and robust way with the transport terms in the equations, while grid solvers can rely on fast Fourier transform (FFT) to evaluate the fields at a minimal numerical cost.

In plasma simulations, grid techniques are indeed the usual way to compute the fields that move the particles. For incompressible flow calculations, however, the common wisdom is that grid solvers not only go against the Lagrangian features of vortex methods, but they also cannot retain the subgrid information that particles carry with them.

The goal of this paper is to show that, when used with the appropriate tools, particle-grid methods combine accuracy, speed, and robustness and provide an efficient alternative to other high-order methods for a number of high Reynolds flows. This point will be made by direct comparisons with centered finite-difference, both from the accuracy and CPU time points of view. As a matter of fact, we believe that direct comparisons of vortex methods with other types of methods, although not very common in the literature, should be done in a systematic way in order to demonstrate their usefulness in computational fluid dynamics (CFD).

We will be concerned with two types of particle-grid techniques. In the first type, the particles and the grid are used in the same computational domain to deal with different aspects of the flow equations. Typically, particles solve the transport equations and the grid can be used to compute the velocity. The diffusion and the vorticity boundary conditions can be solved by either method. These methods are in the spirit of the vortex in cell (VIC) method pioneered by Christiansen [3]. They retain the key feature of particle methods; namely, the advection part of the equation is dealt with in a Lagrangian fashion and reduces to a set of differential equations. As a result, the stability of the vorticity transport and stretching equations is not constrained by classical CFL conditions. As for vortex-blob methods, the time step

[*]Received by the editors February 1, 1999; accepted for publication (in revised form) May 23, 2000; published electronically December 5, 2000.
http://www.siam.org/journals/sisc/22-5/35076.html

[†]LMC-IMAG, Université Joseph Fourier, B.P. 53, 38041 Grenoble Cédex 9, France (cottet@imag.fr).

is determined as a function of the maximum strain in the flow (typically a fraction of the inverse of the vorticity maximum). We will see that they can take advantage of high-order interpolation formulas for optimal particle-grid transfers.

In the second kind of method, the computational domain is split into several subdomains where different solvers are used. The coupling between the methods then has to be done within the framework of domain decomposition techniques and is most easily designed with overlapping subdomains. These methods have been proposed in [4]. The developments we propose here take advantage of high-order regridding of the particles in the overlapping zone. The gain that can be expected from this type of technique in the simulation of complex flows around obstacles is to combine the flexibility of grid and particle methods, respectively, near the solid boundaries and in the wake.

The outline of the paper is as follows. In section 2 we recall the principles of particle-grid assignment and interpolation, which are essential in all particle-grid techniques. Section 3 is devoted to VIC methods. We define these methods for 2D and 3D, ideal as well as viscous, flows and illustrate them on several classic 2D and 3D test cases with strong vortex–wall interactions. In section 4 we focus on particle-grid domain decomposition. We recall the key features of these methods and show that the principles of the method can be indifferently applied to the case when the finite-difference solver is used on a velocity-pressure or velocity–vorticity formulation of the Navier–Stokes equations. The method is then illustrated in the case of a flow over a backward-facing step and the wake of a cylinder. Finally, in section 5 we draw some conclusions on the suitability of particle-grid techniques for the simulation of complex 3D flows.

2. Particle-grid operators. All methods combining grids and particles rely heavily on the ability one has to transfer quantities such as vorticity and velocity between these two types of discretization. These transfers are most simply done by interpolation. Assume that we are given a distribution of particles located at x_p carrying circulations Γ_p, and thus a vorticity field is given by $\omega(x) = \sum_p \Gamma_p \delta(x - x_p)$. If ϕ_i is a basis of functions associated with the grid, a natural formula to assign vorticity values to the grid is

$$\omega_i = \frac{1}{V_i} \sum_p \Gamma_p \phi_i(x_p), \tag{1}$$

where V_i is the volume of the grid cell centered at x_i. Conversely, grid quantities, like velocities, are interpolated at particle locations with the formula

$$u_p = \sum_i u_i \phi_i(x_p). \tag{2}$$

In practice the grid often is deduced from a uniform Cartesian grid through a mapping given by explicit formulas (this is true, in particular, when one wishes to use finite-difference-type grid solvers, which is the case we have in mind). By using the same mapping formulas for the particles, one is led back to the case of a translation invariant grid, with grid size ε, and $V_i = \varepsilon^d$ (d is the dimension). The basis (ϕ_i) then reduces to a single kernel ϕ by the formula $\phi_i(x) = \phi(\frac{x - x_i}{\varepsilon})$, where ε is the grid size. Moreover, ϕ can be chosen as a tensor product of 1D kernels, and from now on we will focus on 1D kernels.

A class of kernels is given by successive convolutions of the top-hat filter χ with support in $[-1/2, 1/2]$. This gives kernels of increasing smoothness, which spread the circulation of one particle onto an increasing number of points. To evaluate the accuracy of a given kernel it is customary to consider its conservation properties. Conservation of the total circulation, as measured by the particles and the grid, reads

$$\sum_i \omega_i V_i = \sum_p \Gamma_p.$$

It is clearly requires that the kernel ϕ satisfies

$$\sum_i \phi\left(\frac{x - x_i}{\varepsilon}\right) \equiv 1. \tag{3}$$

Conservation of linear and angular impulse, respectively, then requires that

$$\sum_i x_i \phi\left(\frac{x - x_i}{\varepsilon}\right) \equiv x, \tag{4}$$

$$\sum_i x_i^2 \phi\left(\frac{x - x_i}{\varepsilon}\right) \equiv x^2. \tag{5}$$

It is readily seen that these properties imply that the interpolation formulas will be of orders one, two, and three, respectively. Fourier analysis [23] gives simple criteria to determine the accuracy of a given kernel. Let g be the Fourier transform of ϕ:

$$g(k) = \int_{-\infty}^{+\infty} \phi(x) \varepsilon^{-ikx} dx.$$

One can show that ϕ yields an interpolation formula of order m (that is, it conserves the moments up to $m - 1$) if the following two conditions hold simultaneously:

$$k \to g(k) - 1 \text{ has a zero of order } m \text{ at } k = 0, \tag{6}$$

$$k \to g(k) \text{ has zeros of order } m \text{ at all } k = 2\pi n \qquad (n \neq 0). \tag{7}$$

Since the Fourier transform of χ is given by

$$g(k) = \frac{\sin(\pi k)}{\pi k},$$

this criterion clearly shows that χ is of order 1, while $\chi \star \chi$ and all the following kernels are second order.

In practice, the requirement one must have on the kernel accuracy very much depends on what is to be done on the grid. In a VIC calculation, where vorticity values on the grid are used only to compute the velocity, one can tolerate a certain level of dissipation in the assignment, and second-order formulas can give sufficient accuracy. However, if vorticity on the grid is directly used in the vorticity equation to solve for the diffusion or in a vorticity boundary condition, for instance, the numerical dissipation induced by these formulas can affect the overall performance of the method. In this case it is advisable to use higher-order formulas. All our calculations use the following third-order kernel first proposed by Monaghan [20] in the context of smooth particle hydrodynamics (SPH) calculations:

$$\phi(x) = \begin{cases} 0 & \text{if } |x| > 2, \\ \frac{1}{2}(2-|x|)^2(1-|x|) & \text{if } 1 \le |x| \le 2, \\ 1 - \frac{5x^2}{2} + \frac{3|x|^3}{2} & \text{if } |x| \le 1. \end{cases} \tag{8}$$

The theoretical order of accuracy of a given kernel is actually obtained only insofar as the particles are able to represent a smooth function. Thus, besides the order of the interpolation kernel, another important factor in the accuracy of particle-grid schemes is the regularity of the particle distribution. Vortex-grid and vortex-blob methods are no different from this point of view. In the high Reynolds flows that we examine, the particles are likely to undergo severe distortions. Some cures can be proposed to maintain a minimal accuracy in the particle-grid transfers. One is to implement the formula (1) with a volume value deduced from the particles through the formula

$$V_i = \sum_p v_p \phi_i(x_p), \tag{9}$$

where v_p are the volumes of the particles. This formula has the effect of compensating for a local lack or excess of particles, because a constant vorticity field grid assignment with the formula (9) recovers the exact value, no matter how distorted the particles are. One may also process the weights of the particles in order to recover the correct vorticity values on the particles, along the same lines as in the iterative method proposed in the context of vortex-blob methods by Beale [1]. However, it is possible to maintain regularity in the particle distribution at almost no cost and with no discernible numerical dissipation (see [9]) by frequently regridding the particles. This has the effect of maintaining a good accuracy for the interpolation formulas. In all the vortex calculations shown in this paper, the same kernel (8) will be used for both the particle-grid transfers and for regridding the particles on regular locations.

3. VIC methods. Let us first recall the velocity–vorticity formulation of the incompressible Navier–Stokes equations

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega - (\omega \cdot \nabla)u - \sigma \Delta \omega = 0, \tag{10}$$

$$u = \nabla \times \psi, \tag{11}$$

$$-\Delta \psi = \omega. \tag{12}$$

This system has to be supplemented with initial and boundary conditions. Here we are interested in methods where grid and particle solvers coexist in the same computational domain. One important motivation for using Navier–Stokes vortex solvers lies in their robust and accurate treatment of the advection terms in (10): The vortex solution to (10) consists of moving vortices with their local velocities and updating their circulations to account for stretching and diffusion, and no explicit computation of the convection term $(u \cdot \nabla)\omega$ is required. On the other hand, grid solves can take advantage of FFT and fast Poisson solvers to compute velocity fields in (11)–(12).

In the early 1970s, grid Poisson solvers were actually the only way to do vortex simulations with more than a thousand particles. The advent of fast summation techniques has allowed completely grid-free simulations using up to a million particles, but in regular bounded geometries grid-based velocity solvers still offer a significant speed-up, unless particles occupy a very small part of the domain.

A typical VIC algorithm in two dimensions consists of the following sequence:

- Assign circulations to the grid with formula (1),
- solve (12) for the stream function on the grid with the desired boundary condition,
- compute grid velocities by finite differences,
- interpolate velocities on particles and move particles.

In this algorithm, most of the CPU time is devoted in the interpolation formulas involved in the particle-grid operators. A typical CPU time for the computation of the velocities on two million particles with the same number of grid points is a few seconds on a DEC workstation in three dimensions. To give an idea of the savings over a fast multipole algorithm in a vortex-blob method, from the figures given in [24] we can extrapolate that the same number of particles would require with these methods more than an hour of CPU time.

In a viscous code, it is also possible to solve for the diffusion on the grid. After the assignment step one computes $\Delta\omega$ on the grid by finite differences and then interpolates this quantity on the particles before updating their circulation. One must be aware that this scheme is not conservative. A conservative variant can be obtained with a finite-element formulation (see [6]). However, our simulations show that, when regridding, we maintain enough regularity in the particle distribution so the lack of conservation is not noticeable.

In the case where the grid is used for the diffusion, also it must handle a vorticity boundary condition to supplement (10) and translate the desired velocity boundary condition. One may either use an influence matrix technique to derive exact, at the grid level, vorticity boundary conditions, or rely on Taylor expansions to derive approximate boundary conditions, accurate to a given order. There has been in the last years extensive work in this direction. We refer in particular to [22, 11] and the references therein. We now illustrate these techniques on three flow examples: the 2D driven cavity, vortex dipoles, and vortex rings.

3.1. Driven cavity flow. In the VIC method implemented here, the diffusion is solved on the grid. It is compared with a finite-difference scheme using a centered second-order approximation of the advection and diffusion terms. When used together with fourth-order Runge–Kutta time stepping, this scheme does not suffer any cell Reynolds number limitations and is linearly stable under a convection CFL condition. We have implemented several other finite-difference schemes, in particular Arakawa and a fourth-order compact scheme [12]. We have observed that, when used with sufficient resolution, these schemes give very close results. When insufficiently resolved, all centered schemes ultimately lead to instabilities. The second-order scheme has been chosen in view of its simplicity and low CPU cost.

Concerning the vorticity boundary condition, we have implemented Thom's first-order formula, various second-order formulas, and Briley's fourth-order formulas. Here again we have observed that the difference between second- and fourth-order formulas was very marginal. (These formulas, however, clearly outperform Thom's formulas.) We have finally chosen the following second-order scheme: velocities at the interior grid points are computed from the stream function and at the boundary using the

Table 1
Run parameters and CPU times with VIC and finite-difference methods for various Reynolds numbers and resolutions.

Reynolds number	100	2000	10,000
M_{fd}	64	128	256
M_{vic}	64	128	256
δt_{fd}	0.01	0.008	0.004
δt_{vic}	0.01	0.02	0.04
CPU time for finite-difference scheme	3	24	225
CPU time for VIC scheme	5	16	32

velocity boundary conditions; then second-order one-sided finite differences are used to compute the vorticity at the boundary, which yields the Dirichlet boundary conditions used in the diffusion solver. The same vorticity boundary condition is used in the VIC method.

The finite-difference method used a fourth-order Runge–Kutta time stepping. This is actually necessary for the centered scheme to be stable [12] under a CFL condition. As for VIC method, it does not impose on the time step any stability constraint from the treatment of the convection term. In our simulations with the VIC method, we used second-order Runge–Kutta time stepping.

Due to the explicit treatment of the diffusion, for both methods the time step also must satisfy a diffusion CFL. In practice, for high Reynolds number flows, this last stability condition is far less drastic than the advective CFL condition. As a result, in most cases of interest the VIC code could be run with bigger time steps than the finite-difference code. Table 1 summarizes some typical run resolution and CPU times for a given simulation time on a Sun Sparc station. Due to the interpolation steps, each iteration of the VIC code requires slightly more computational time than the finite-difference code, but for Reynolds numbers beyond 10,000, the bigger time steps used in the VIC code lead to substantial CPU savings. Another nice feature of the VIC method is that it is insensitive to the lack of smoothness of the flow, whereas centered finite difference may become unstable when used without sufficient resolution if sharp vorticity gradients develop.

Figure 1 shows the vorticity contours obtained with the VIC and the finite-difference method for 256^2 and 512^2 resolutions.

The Reynolds number is 10,000, and the velocity profile at the top boundary has a parabolic profile. At the lower resolution, the two methods diverge from each other in the transient stage before reaching the same quasi-steady-state regime. As we refine the resolution, both methods converge toward the same solution. This is confirmed by looking at the vorticity profile on the first diagonal (Figure 2). The 256^2 resolution is enough in the VIC code to obtain converged results for a large time behavior. At this resolution it yields a sevenfold speed-up over the finite-difference method.

This CPU gain makes affordable large time simulations at higher Reynolds numbers. Figure 3 shows successive stages of the vorticity field in a driven cavity at a Reynolds number of 10^5, with 1024^2 grid points. (In this calculation, as in all our VIC simulations, we used the same mesh size for the grid and the particles.) The same resolution would have required a CPU time 17 times larger with the finite-difference method. (However, it is fair to mention that, due to the proliferation of small scales at this Reynolds number, it is virtually impossible to obtain converged

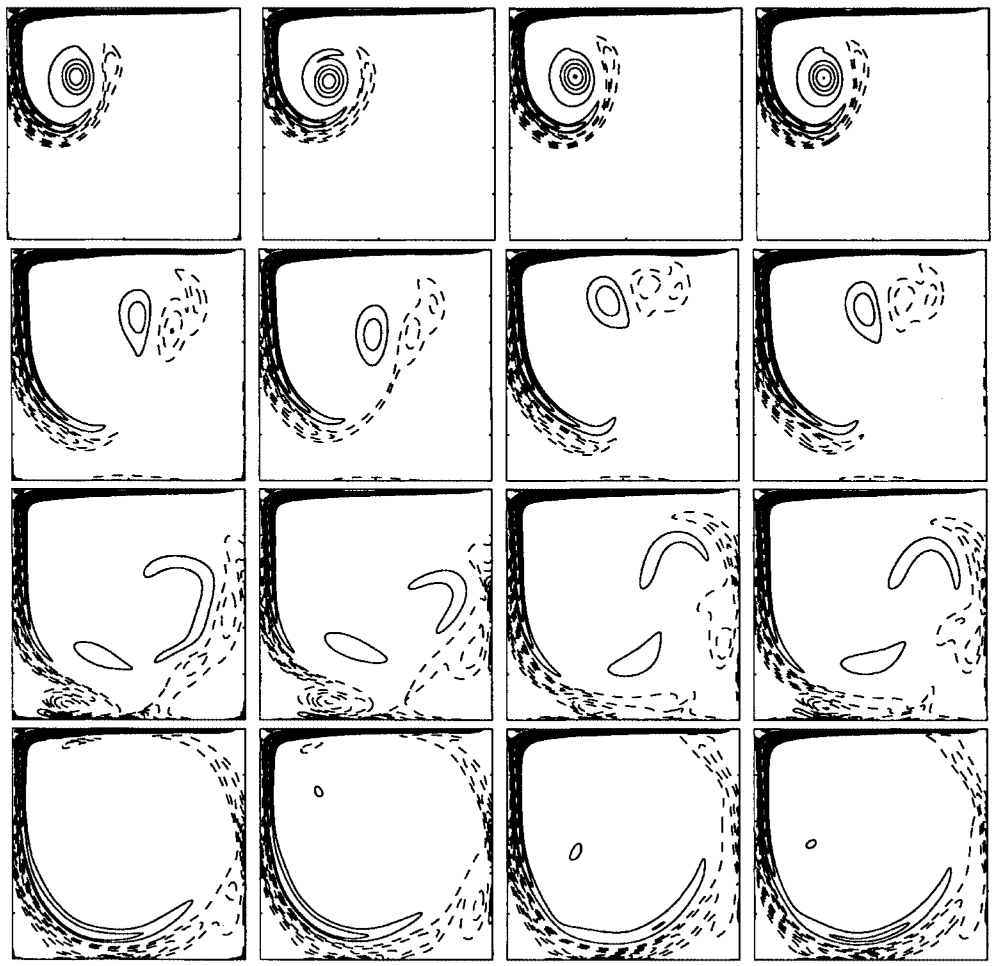

FIG. 1. *Sequence of vorticity profiles in a driven cavity at a Reynolds number of* 10^4*. From left to right: VIC method with* 256^2 *resolution, finite-difference method with* 256^2 *resolution, VIC method with* 512^2*, and finite-difference method with* 512^2*. From top to bottom: times* 10, 20, 30, 40.

results; these simulations only convey qualitative information, such as the number of eddies.) Demonstrations of this simulation can be downloaded from the World Wide Web site www-lmc.imag.fr/lmc-edp/Georges-Henri.Cottet/vortex.html.

3.2. Dipole-wall interaction. In the calculations we now show, the diffusion and vorticity boundary conditions are computed on the particles and the grid is used only to compute the velocity. The particle diffusion solver is a particle strength exchange (PSE) scheme [10]; that is, particles exchange vorticity among them through

$$\frac{d\omega_p}{dt} = \sigma\varepsilon^{-2}\sum_q v_q(\omega_q - \omega_p)\Lambda_\varepsilon(x_p - x_q).$$

In the above formula, v_q are the volumes of the particles (typically h^d when particles are initialized on a uniform mesh with mesh size h) and Λ is a kernel satisfying appropriate second-order moment conditions.

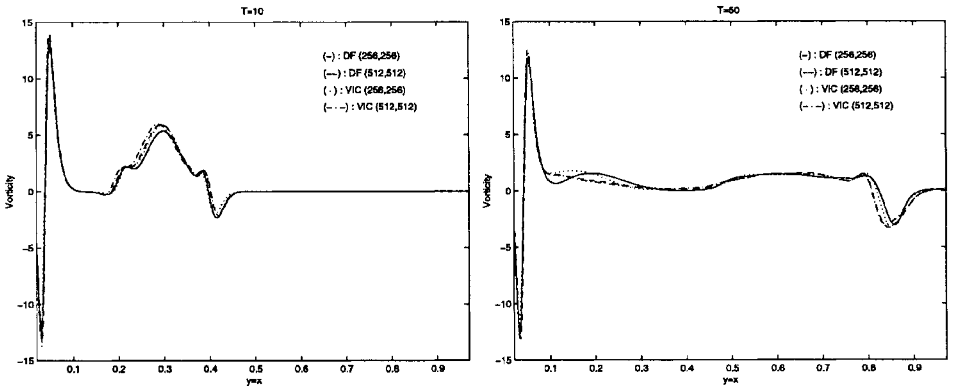

FIG. 2. *Vorticity profiles along the line $x_1 = x_2$ in a driven cavity at a Reynolds number of* 10^4.

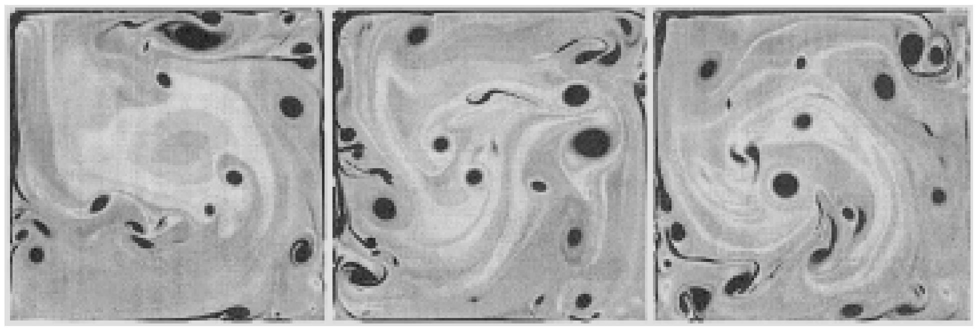

FIG. 3. *Successive stages of the evolution of the driven cavity at a Reynolds number of* 10^5. *VIC method using* 1024^2 *particles.*

This diffusion scheme is used in a viscous splitting algorithm with a Neumann vorticity boundary condition translating the no-slip condition at the wall. This algorithm is described in [5, 16]. It consists of first doing a time step of the PSE scheme, then evaluating the slip $u \cdot \tau$ at the wall. Finally, the diffusion step is repeated with a vorticity flux equal to $-u{\cdot}\tau/\delta t$. This last step is solved by means of integral equations.

This scheme has been applied successfully in various geometries [15, 17]. However, to our knowledge there has not been any direct comparison in terms of CPU time and accuracy of this method with grid-based schemes. On the basis of the results for the driven cavity, we have chosen to compare the VIC method with a second-order centered finite-difference scheme with fourth-order Runge–Kutta time stepping and the second-order vorticity boundary condition described above for the driven cavity problem.

Figure 4 shows a comparison of the results given by the VIC and finite-difference codes at a Reynolds number of 800. The initial condition is a Lamb dipole with circulation $\Gamma = 6.83$. The boundary conditions are periodic in the direction parallel to the wall. At the top boundary we assume no through flow and zero vorticity. The resolution is $\Delta x = 0.027$ for both simulations. Due to the advection CFL, the time step for the finite-difference scheme is five times smaller than for the VIC code. The results are in good agreement, although as time goes on the VIC solution seems to go slightly faster than the finite-difference solution.

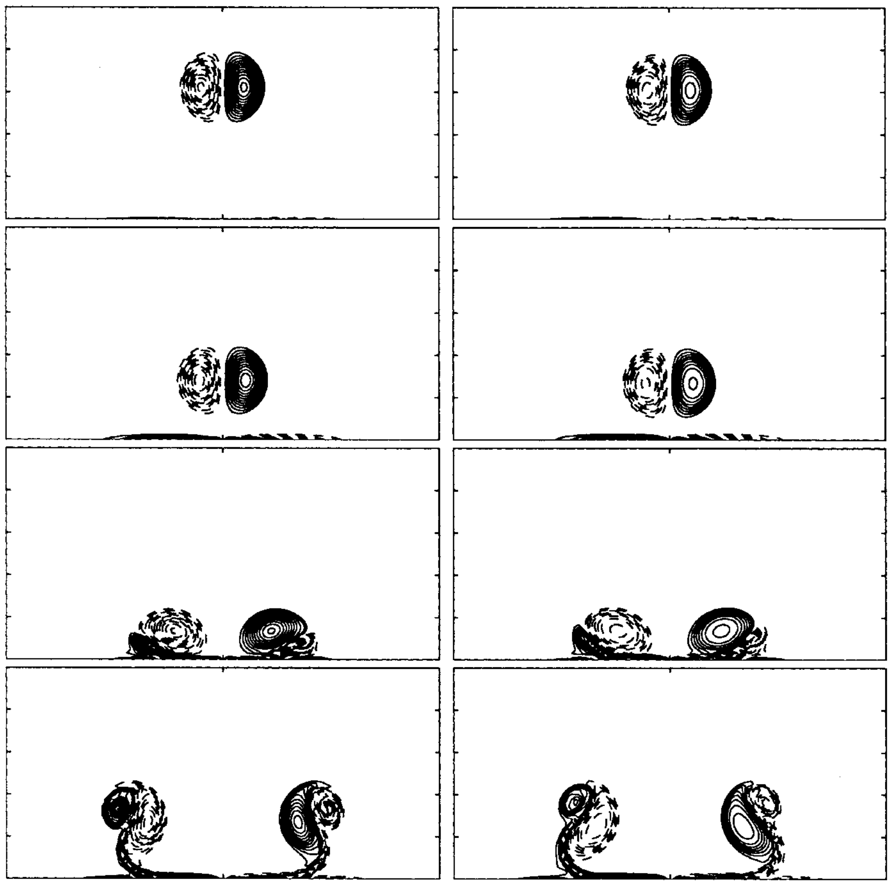

FIG. 4. *Contours of vorticity for a vortex dipole impinging a wall at times* 2, 4, 6, 8. *Left column: finite-difference method,* $\Delta t = .02$*; right column: vortex method,* $\Delta t = 0.1$.

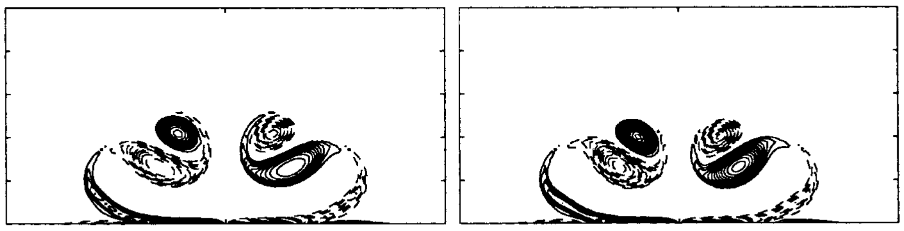

FIG. 5. *Contours of vorticity for a vortex dipole impinging a wall at time* 10. *Reynolds number* 800*, vortex method. Left picture:* $\Delta x = .027$*,* $\Delta t = .1$*; right picture:* $\Delta x = 0.0135$*,* $\Delta t = .05$.

Figure 5 is a refinement study at a later time for the VIC method (the resolution and time steps have been divided by a factor of 2), which shows that the VIC results in Figure 4 can be considered as nearly converged results.

Finally, Figure 6 is a VIC simulation at a Reynolds number of 1600. The resolution has been increased to $\Delta x = 0.0195$, but the time step has been kept equal to 0.1. For the same resolution, the finite-difference method would have required a time step 10 times smaller. Compared with the former case, the lower viscous dis-

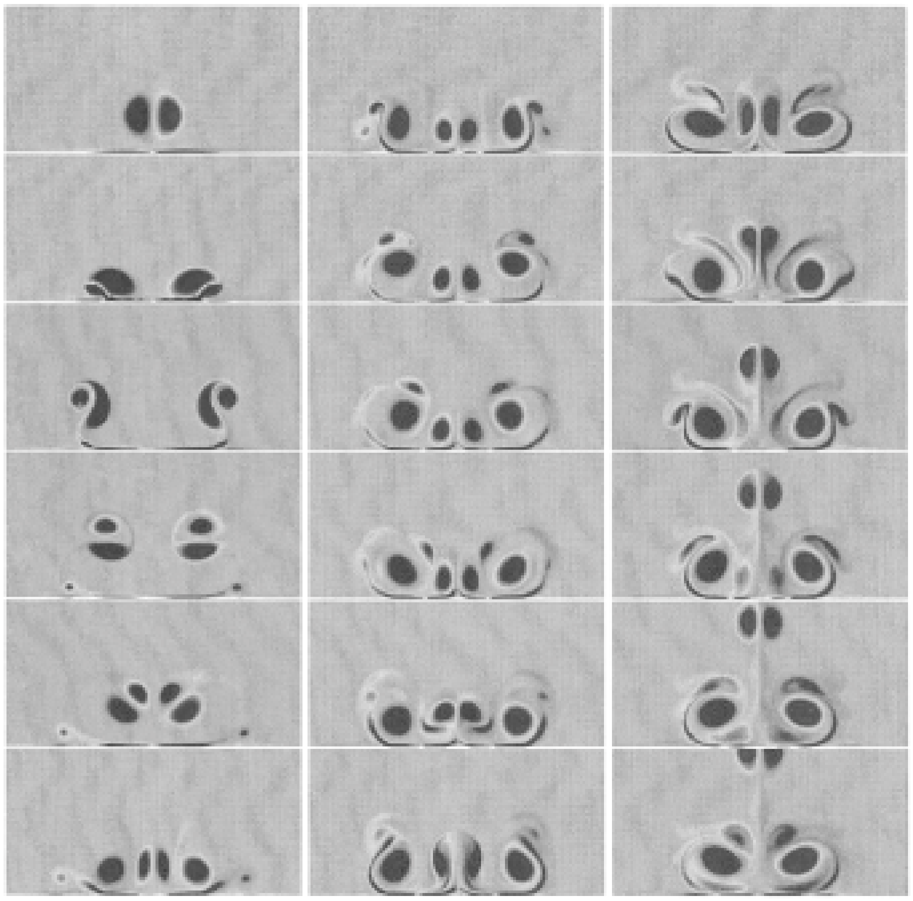

FIG. 6. *Vortex dipole impinging a wall by a VIC method with* $\Delta x = .0195$*,* $\Delta t = .1$*. Reynolds number of* 1600*.*

sipation yields larger circulations in the secondary vorticity produced at the wall. The resulting new dipole therefore gets enough circulation to eject itself from the wall.

These results are in good agreement with the results of Orlandi [21], however, with the same delay as noticed for the Reynolds 800 case.

It is interesting to note that, although first order in time, the vorticity flux boundary conditions give a satisfactory accuracy even when used with large time steps. By contrast, we have found that the vorticity boundary conditions given by Thom's formula, which is first-order accurate in space, failed to give acceptable results for this type of flow where it is crucial to capture accurately the sharp vorticity gradients produced at the boundary.

3.3. Ring-wall interactions. Here we consider the 3D counterpart of the dipole calculations just seen. Let us first describe how to extend the VIC schemes described in section 3.2 to three dimensions. First we must discretize the stretching term $(\omega \cdot \nabla)u$ appearing in the vorticity equation. One way is to obtain velocity derivatives by finite difference on the grid, then to interpolate these values at the particle locations. Particle vorticities are then updated, as in a grid-free vortex method, by solving

$$\frac{d\omega_p}{dt} = [\nabla u(x_p)]\omega_p.$$

Another possibility, which we have elected in our calculations, is based on the conservative form $\mathrm{div}(\omega : u)$ of the stretching term. The idea is to multiply grid values of vorticity and velocity. The tensor $\omega_i u_j$ is then differentiated on the grid, and its divergence is finally interpolated on the particles to update their circulations. This last option has the advantage of being conservative at the grid level, even when the vorticity field is not exactly divergence-free. To avoid numerical dissipation in this procedure, we have used a fourth-order finite-difference formula for the divergence to compute the stretching term on the grid.

The second point that needs to be clarified is the extension to three dimensions of the vorticity flux boundary condition used in the 2D vortex schemes. One needs boundary conditions for each vorticity component. For no-slip velocity, the normal component of the vorticity must clearly vanish. For the components parallel to the wall, we adopt the same viscous-splitting point of view as in the 2D case and write for them a Neumann vorticity boundary condition to cancel the slip in the orthogonal direction of the wall. We refer to [6] for a more detailed account of this technique and a proof that it does not violate the divergence-free constraints for the vorticity.

As explained in [21], the features of the vorticity equation in three dimensions imply some important differences between the ring-wall and the dipole-wall interactions. These differences are noticeable particularly when the ring hits the wall with an angle. In this case the lower part of the ring is subject, as it approaches the wall, to intense stretching, which has the effect of pumping its circulation, which then feeds the upper part of the ring, causing it to rebound. We have chosen for these simulations the same parameter as in [25]: The angle is 38.5 degrees, the computational box is $[0,1]^2 \times [0,1/2]$ with periodic boundary conditions in the directions parallel to the wall $z = 0$, no slip at the wall, and no through flow and zero vorticity at the top boundary. The ring has radius $R = 1/8$ with a Gaussian core of radius equal to $0.4R$. At time zero it is located at a distance 0.25 of the bottom wall. The Reynolds number, based on the ring circulation, is 1400.

Particles were initialized and remeshed on a regular 128^3 grid. The same grid resolution was used for the computation of the velocity field. This resolution is roughly the same as in [25] (these authors used $129 \times 91 \times 97$ grid points, with grid refinement in the normal direction of about a factor of 2 near the wall). At the last stage of the computation considered here, there were about 900,000 particles (which means that the vorticity occupied roughly half of the computational domain).

Figure 7 shows the contours of the y-component of the vorticity in the symmetry (x, z) plane together with isosurfaces of the vorticity magnitude (for clarity, only half of the surfaces are shown) at times 24, 40, and 64. These results are in excellent agreement with the results of [25]. At this resolution the time steps used in the finite-difference and vortex methods are of the same order, and the goal of this example mostly was to demonstrate the ability of vortex methods to perform accurate direct numerical simulations of complex vortex-wall interactions. Quantitative comparisons with spectral methods for homogeneous turbulent flows and Crow instabilities are presented elsewhere [8].

4. Particle-grid domain decomposition. We now describe numerical techniques in which grid and particles are used in different subdomains of the computational domain. These techniques are rather natural: Vortex methods are attractive

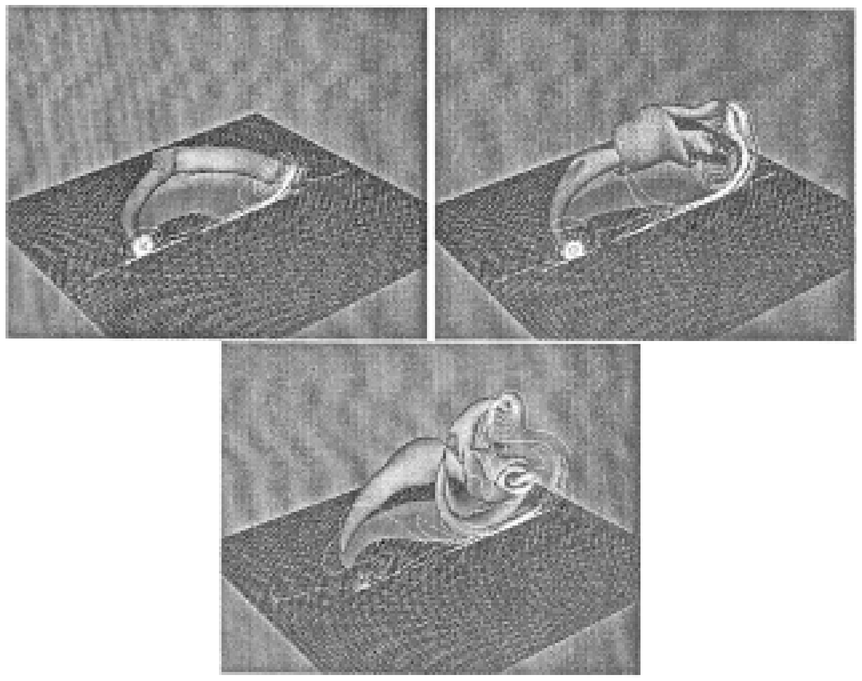

Fig. 7. *Vortex ring hitting a wall at an angle of* 38.5 *degrees. Contours of the vorticity component perpendicular to the symmetry plane, and isosurfaces of vorticity magnitude at times* 24, 40*, and* 64 *(from left to right, top to bottom).*

particularly in flow regions where the vorticity is confined to areas of small dimension and do not need to model far-field conditions, whereas grid-based methods offer more flexibility in dealing with boundary conditions at solid walls, in particular because using the velocity-pressure formulation is allowed.

4.1. Design. Particle-grid domains have been formalized for the first time in [4]. Note that at this time there was no clear understanding of how to accurately implement no-slip boundary conditions in vortex methods, which was a strong motivation for coupling vortex methods with grid techniques near obstacles. The method described in this reference is based on the velocity–vorticity formulation of the equations in the whole domain, but as we will see it extends readily to the case when a velocity-pressure formulation is chosen in the finite-difference subdomain.

As stressed in [4], for a clear-cut particle-grid domain decomposition algorithm, it is important to separate the kinematic and kinetic parts of the flow equations. They correspond to equations of a different nature and thus require different interface conditions.

Without loss of generality, we will assume no-slip boundary conditions and a domain decomposition as sketched in Figure 8. Ω_1 and Ω_2 are, respectively, the finite-difference and vortex domains. Important ingredients in the method will be the overlapping of Ω_1 and Ω_2 and the fact that there exists a domain Ω_2' including Ω_2 such that particles and grid coexist in $\Omega_1 \cap \Omega_2'$. The precise role played by Ω_2' and the geometrical constraints imposed on the width of the overlapping will appear in the discussion below.

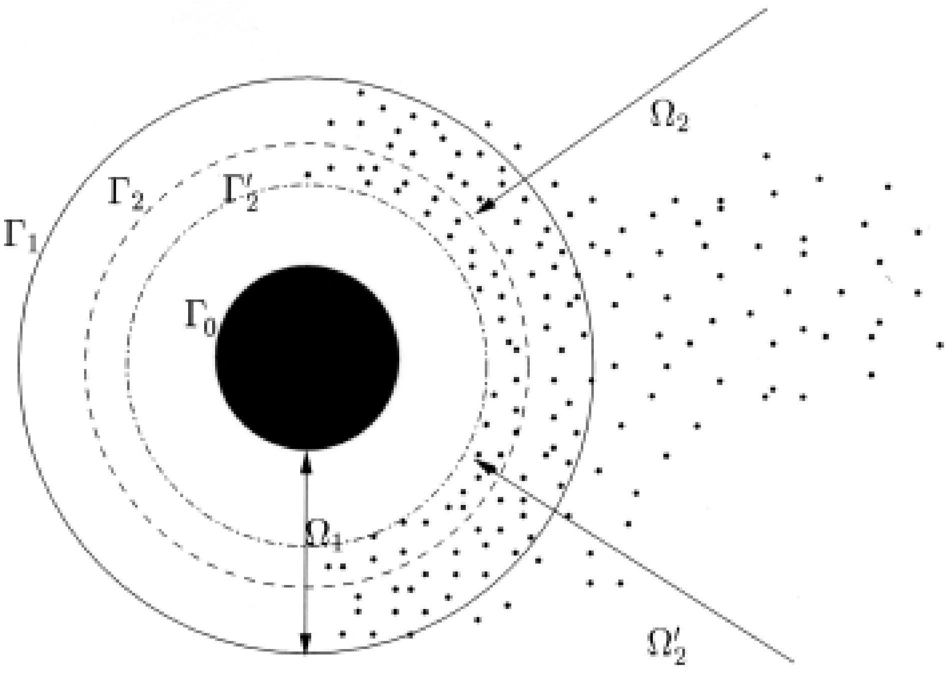

FIG. 8. *Particle-grid domain decomposition with overlapping.*

Let us mention that an alternative approach, without overlapping, has been proposed in [13]. This method, however, does not clearly indicate the interface conditions on which it relies and its consistency is not obvious.

Let us first consider the case when the velocity–vorticity formulation is used in both domains. At each time step—or substep in a Runge–Kutta scheme—one has to solve the elliptic equation $-\Delta\psi = \omega$ in each subdomain with the appropriate boundary conditions. While on Γ_0 the no-through-flow condition gives a homogeneous Dirichlet boundary condition, on Γ_1 and Γ_2 these boundary conditions are part of the calculations. A popular and efficient way to determine them is the Schwarz alternating method. In our case, it is slightly modified to take into account that different solvers are used in the two subdomains: a finite-difference solver and an integral one, given by the Poincaré identity, in the vortex zone. Note that, unlike for the bounded flows considered in the previous section, a VIC approach would not be appropriate in Ω_2, as it would not retain the ability of vortex methods to exactly satisfy the far-field condition. One iteration of the Schwarz algorithm proceeds as follows.

1. Solve $-\Delta\psi = \omega$ in Ω_1 by finite differences with $\psi = 0$ on Γ_0 and $\psi = a$ on Γ_1.
2. Compute $b = \psi$ and $c = \partial\psi/\partial n$ on source points on Γ_2.
3. Evaluate

$$\psi = \int_{\Omega_2} G(x-y)\omega(y)\,dy + \int_{\Gamma_2} G(x-y)c(y)\,dy + \int_{\Gamma_2} \frac{\partial G}{\partial \nu}(x-y)b(y)\,dy$$

on grid points of Γ_1, where ω is the vortex-blob solution in Ω_2 and G is the Green function $G(x) = \log|x|/2\pi$, to get new values of a on Γ_1.

Once the stream function, and thus the velocity, is known on both subdomains, we can describe how to update the vorticity. While the vortex method is always based on explicit time discretization of the equations, one may consider finite-difference solvers using either an explicit or implicit treatment of the diffusion. However, for the Reynolds numbers we are interested in, the time step is always limited by the advective CFL condition, and explicit diffusion solvers do not yield any additional stability constraints, so we will discard the option of implicit schemes.

To simplify the discussion we will consider the case of a forward Euler time discretization in both domains. To advance from time t_n to time t_{n+1}, explicit finite-difference solvers require only the knowledge at time t_n of the vorticity at the grid points on the boundary, assuming a second-order stencil, or a higher-order method with one-sided finite differences near the boundaries. We have already indicated in section 3 how to deal with the no-slip boundary on Γ_0. Since $\Gamma_1 \subset \Omega_2$, it is natural to obtain boundary values there by assigning vorticity values available at that time on particles of Ω_2, provided Ω_2 contains all the particles contributing to this assignment: If ε is the grid size and the interpolation kernel is given by (8), this means that $d(\Gamma_1, \Gamma_2) \geq 3\varepsilon$. Conversely, to update the circulations of the particles in Ω_2 one needs the circulations at time t_n of the particles in this domain surrounded by a layer of size at least $u_{\max}\Delta t + \rho$, where ρ is the width of the kernel used in the PSE scheme (of the order of the grid size ε). This dictates the geometrical constraints that have to be fulfilled by Ω_2 and Ω_2': One must have $d(\Gamma_2, \Gamma_2') \geq u_{\max}\Delta t + \rho$. In practice the various boundaries are separated by distances that are fractions of the body size, and these constraints are satisfied easily. To update the circulations of the particles in Ω_2, one finally interpolates grid values on the particles in $\Omega_2' - \Omega_2$ and includes these particles into the vortex scheme.

Note that the scheme just described is nothing but an extension of what would be done if a finite-difference scheme was used in both domains Ω_1 and Ω_2. In some sense the particles in $\Omega_2' - \Omega_2$ play the role of the outer grid points that would be needed in a finite-difference solver.

In summary, the algorithm proceeds as follows, assuming a forward Euler time stepping in both subdomains: Given a vorticity field ω^n known on the grid in Ω_1 and on particles in Ω_2,

- compute the stream function on Γ_1 and Γ_2 and its normal derivative on Γ_1, using the Schwarz alternating method;
- deduce velocity values on the grid and on particles in Ω_2;
- interpolate grid values of ω^n on particles in $\Omega_2' - \Omega_2$, and assign particle circulations to grid points on Γ_1;
- update vorticity in both domains to obtain ω^{n+1}.

In our calculations, this scheme is extended to a fourth-order Runge–Kutta time discretization in both domains.

The consistency of the interface conditions just described clearly relies on the ability of the particles in $\Omega_2' \cap \Omega_1$ to transfer vorticity between the two subdomains. This is achieved by periodically regridding particles in Ω_2' and by using high-order interpolation formulas like those described in section 2.

Let us mention that very similar ingredients can be used to design particle–particle domain decomposition methods. In this case, vortex methods are used in all subdomains, but they correspond to different mesh resolutions. We refer to [6, 7] for details.

We now turn to the case when a velocity-pressure formulation is used in the finite-difference domain. We have chosen in our simulations a second-order projection technique that is a straight-forward extension, based on a second-order Runge–Kutta time advancing scheme, of Chorin's classical projection method [2]. For the sake of simplicity, as in the velocity–vorticity case, we will describe the domain decomposition method assuming a forward Euler discretization. In this case, starting from a divergence-free velocity u^n the projection method consists in computing

$$R^n = -(u^n \cdot \nabla)u^n + \sigma\Delta u^n \tag{13}$$

then

$$u^{n+1} = u^n + \Delta t(R^n + \nabla p^n). \tag{14}$$

The pressure p^n is adjusted to make u^{n+1} divergence-free and to satisfy the appropriate boundary condition for the normal component:

$$\Delta p^n = -\text{div } R^n \text{ in } \Omega_1, \tag{15}$$

$$\frac{\partial p^n}{\partial \nu} = -R^n \cdot \nu + \frac{u^{n+1} - u^n}{\Delta t} \cdot \nu \text{ on } \partial\Omega_1. \tag{16}$$

This system requires the knowledge of $u^{n+1} \cdot \nu$ on Γ_1. It may be obtained again by a Schwarz alternating method: Each Schwarz iteration consists of solving (15), (16), then evaluating the velocity on Γ_2 with (14), and finally using the Poincaré identity in Ω_2 to obtain new values for $u^{n+1} \cdot \nu$ on Γ_1.

One step of the complete algorithm, still in the case of a forward Euler scheme, proceeds as follows, starting from u^n in Ω_1 and ω^n in Ω_2:

- compute the particle velocities in Ω_2 with the Poincaré formula;
- differentiate u^n on the grid in Ω_1 and interpolate grid vorticity values on particles in $\Omega_2' - \Omega_2$;
- update particles in Ω_2 to get ω^{n+1};
- compute G^n, then $u^{n+1} \cdot \nu$ on Γ_1 through (14)–(16) and the Schwarz method, and finally u^{n+1}.

For a Runge–Kutta scheme, these steps have to be repeated at each substep.

Another type of particle-grid coupling method with overlapping has been proposed in [14]. The main difference with the method of [4] used here is that particle velocities are computed using a Biot–Savart law involving both particles and grid points. We believe that this approach is more expensive, since the Biot–Savart solver, even in the most efficient implementations of fast solvers, is significantly slower than a finite-difference one in simple geometries (in our calculations the number of grid points was always larger than the number of particles, even for a wake extending far downstream). Moreover, mixing grid points and particles in the same quadrature formula requires adjusting their volumes in the interface zone, which introduces some technical complications.

4.2. Numerical results. We have selected here a few results that illustrate the main features of the method. A more systematic study of particle-grid domain decomposition can be found in [18].

Our first example is to demonstrate that the interface conditions, although combining solvers of a different nature, maintain a smooth transfer of vorticity. Figure 9 shows the vorticity in a flow over a backward-facing step at a Reynolds number of 355. In the bottom picture, the finite-difference zone surrounds the top and bottom limits of the channel. The interface Γ_2 with particle is shown by a dotted line.

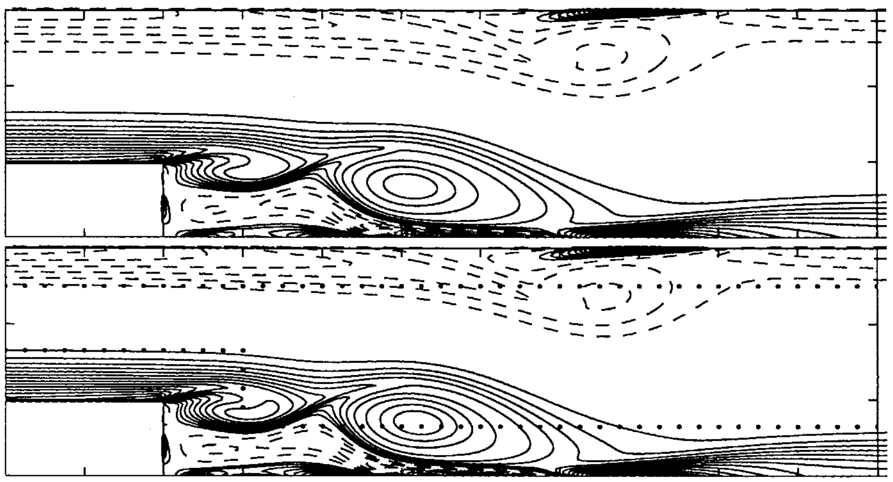

FIG. 9. *Vorticity contours for a flow over a backward-facing step at a Reynolds number of* 355. *Top picture: finite-difference calculation; bottom picture: particle-grid domain decomposition. The bullets materialize the interface.*

As a comparison, the top picture shows the results of a calculation using the finite-difference solver everywhere. The results are almost identical. Although the interface cuts through several recirculation zones with sharp vorticity gradients, no oscillations in the vorticity are discernible. Particles and grid have the same mesh size equal to 3% of the height of the step. This results in about 30,000 discretization points.

We now focus on flows past a cylinder. This is a good prototype of an unbounded flow producing complex dynamics near the boundary and it is well documented. In all our calculations the cylinder had a unit radius, and the velocity at infinity was equal to 1. The finite-difference domain extended to $r = 2$. The finite-difference method was a second-order centered scheme written in polar coordinates. The vortex and finite-difference schemes used fourth-order Runge–Kutta time stepping. In all cases the grid resolution in the finite-difference and vortex domains were identical.

We have first compared the particle-grid domain decomposition with a finite-difference method using an artificial boundary condition at the outer boundary. Both methods are based on a velocity–vorticity formulation of the flow equations. Figure 10 shows the vorticity on the cylinder at time 5, and the velocity profile on the symmetry axis behind the cylinder, for a Reynolds number of 550. The finite-difference domain extended to $r = 7$. One can observe that the vorticity values are almost identical, while the velocity profiles for late times differ in the wake due to the artificial boundary condition used in the finite-difference solver.

Figure 11 is concerned with a Reynolds number of 3000. It shows the vorticity contours obtained by the particle-grid domain decomposition method at times 3.2 and 8.4. Figure 12 shows the drag history. These results are in excellent agreement with the results of Koumoutsakos and Leonard [15]. The finite-difference method used 120×620 grid points on a polar grid extending to two times the radius of a cylinder. The number of particles at time 8 was about 36,000. At that time, the vorticity support extends to about $r = 7.1$, which means that a pure finite-difference method with the same resolution would have needed more than 720 grid points in the radial direction, and thus about four times more computational elements than the domain decomposition method. This example, although in a case where the vorticity is fairly

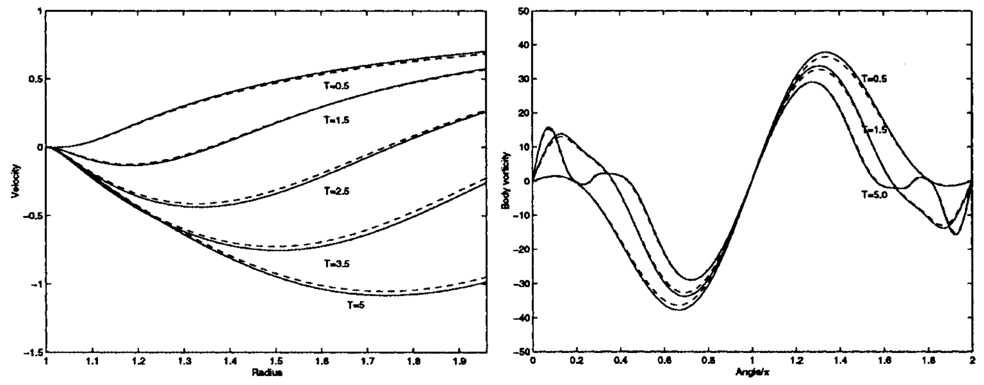

FIG. 10. *Flow past a cylinder at Reynolds* 550. *Vorticity values on the cylinder (right picture) and velocity in the back of the cylinder (left picture). Solid line: particle-grid domain decomposition; dashed line: finite-difference method.*

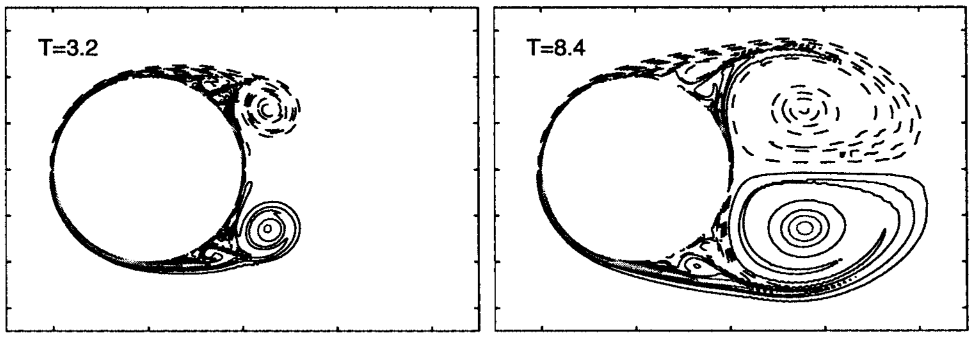

FIG. 11. *Flow past a cylinder at Reynolds number* 3000 *by a particle-grid domain decomposition. Vorticity contours.*

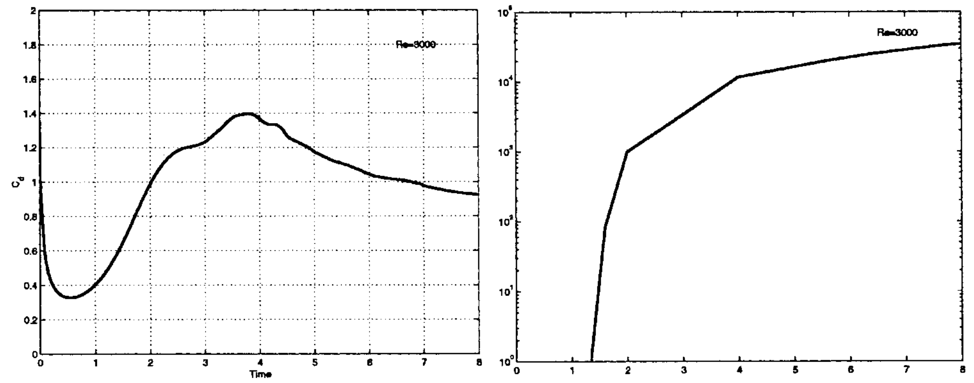

FIG. 12. *Flow past a cylinder at Reynolds number* 3000 *by a particle-grid domain decomposition. Drag history and number of particles.*

confined, illustrates the potential savings that the particle-grid method offers.

Our last examples are concerned with a particle-grid domain decomposition using a velocity-pressure formulation in the finite-difference domain. Figure 13 is a compar-

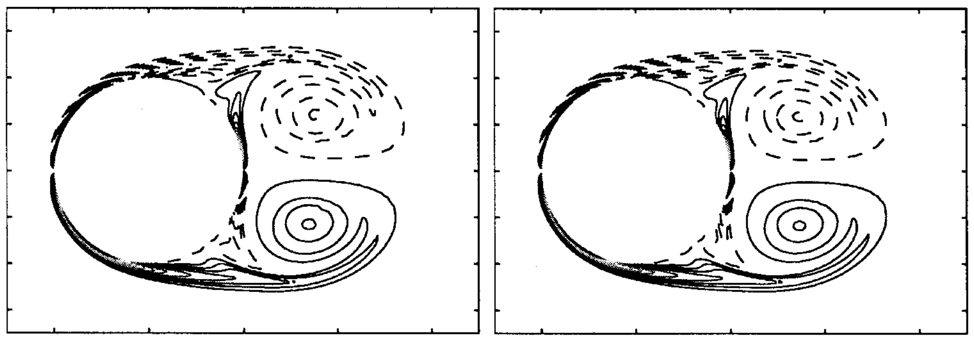

FIG. 13. *Vorticity contours for flow past a cylinder at Reynolds number* 1000. *Particle-grid domain decomposition based on a velocity–pressure formulation (left picture) and a velocity–vorticity formulation (right picture) in the finite-difference at time $T = 6$.*

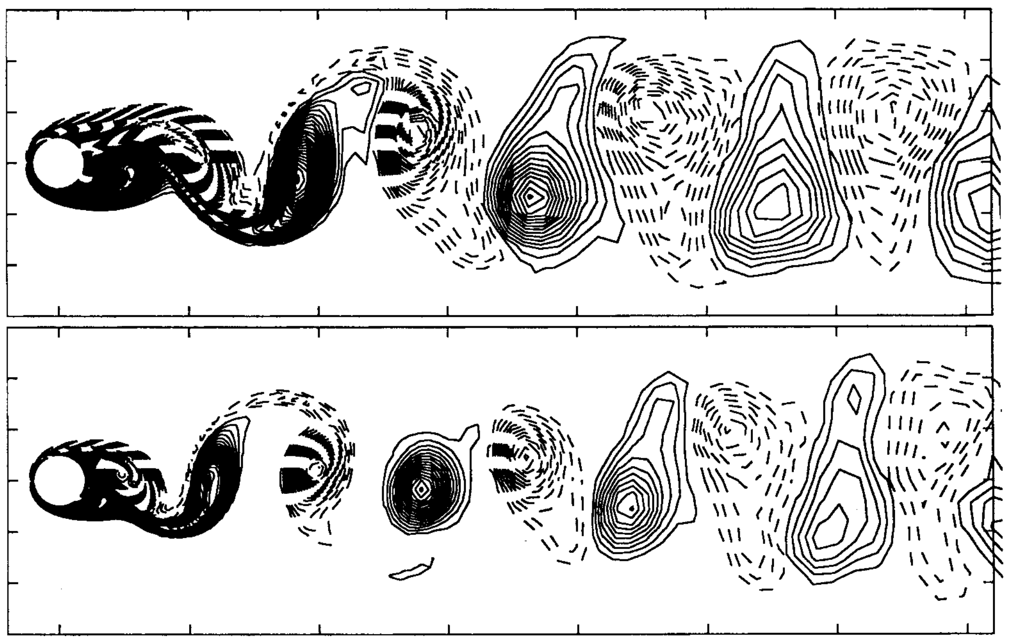

FIG. 14. *Karman streets behind a cylinder at Reynolds number of* 150 *(top picture) and* 300 *(bottom picture).*

ison of this technique with the domain-decomposition method based on a velocity–vorticity formulation in both domains at a Reynolds number of 1000. The vorticity contours are almost identical, which shows that combining different solvers based on different formulations of the equations does not introduce new difficulties, provided the interface conditions are written in a consistent way.

Finally, we now consider the case of a developed wake behind a cylinder. This is clearly the case where the vortex method is able to yield the most savings, as it uses computational elements only in the support of the vorticity and does not require any far-field artificial boundary condition. The finite-difference domain extended in this case to a six times the radius of a cylinder and the vortex domain started at three times the radius of a cylinder. We have used a grid resolution with exponential stretching in the radial direction in both domains. The finite-difference domain extended to six times the radius of a cylinder and the vortex domain started at three times the radius of a cylinder. The vortex method used variable-size blobs with an exponential grid stretching in the radial direction (see [6, 7]). Figure 14 shows the Karman street obtained for Reynolds numbers of 150 and 300 at some late times. In the latter case the finite-difference grid used in the domain was 17,000 points, and the number of particles at the end of the simulation was slightly less than 12,000. The Strouhal numbers measured from these computations were, respectively, 0.183 and 0.208, in excellent agreement with the experimental values found by Williamson [26]. Note that, due to the loose resolution away from the cylinder, the far wake is obviously not accurately captured beyond about 10 diameters. However, this does not affect the computations of the forces on the body. In this case the particle-grid domain decomposition can be considered as a way to provide a good artificial boundary condition for the finite-difference solver for a limited cost.

5. Conclusion and future plans. We have presented a class of methods that combine grid-based and vortex schemes. High-order interpolation formulas are the key ingredients to transfer information between the two solvers at low cost and with minimal numerical dissipation.

VIC methods, where the grid essentially is used to compute the velocity fields and to take advantage of FFT-based fast Poisson solvers, provide robust and accurate solvers for flows in simple geometries. These features make them an attractive alternative to high-order grid-based methods, whenever it is crucial to compute complex dynamics on large time scales without numerical dissipation.

In two dimensions, stratified geophysical flows are good prototypes of such flows that could benefit from vortex methods. In three dimensions there are a number of flows in simple geometries that still lack accurate descriptions at high Reynolds numbers, in particular due to the time-step limitations that are in general implied by fine grids. Concerning 3D flows, work is underway to gain from classical homogeneous turbulence problems further insight into the effects at a subgrid level of the interpolations involved in grid-particle interpolations.

From another perspective, although vortex methods have proved that they are efficient in the simulation of complex vortex-wall interactions in two and three dimensions, particle-grid domain decomposition methods may be viewed as a more flexible tool, in particular in three dimensions when the grid solver near the obstacle is based on a velocity-pressure formulation. On the other hand, for 3D flows, it seems that pure finite-difference solvers still suffer from a dilemma between accuracy and stability. The numerical dissipation produced by upwind high-order schemes makes them inappropriate for large eddy simulations. Centered schemes do not have this difficulty, but they are very sensitive to grid regularity and to outflow boundary conditions [19]. Numerical schemes using vortex methods in the wake may be able to improve these aspects of current finite-difference solvers.

REFERENCES

[1] J. T. Beale, *On the accuracy of vortex methods at large time*, in Computational Fluid Dynamics and Reacting Gas Flows, B. Engquist, M. Luskin, and A. Majda, eds., Springer-Verlag, New York, 1988, pp. 19–32.

[2] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[3] J. P. Christiansen, *Vortex methods for flow simulation*, J. Comput. Phys., 13 (1973), p. 363–379.

[4] G.-H. Cottet, *Particle-grid domain decomposition methods for the Navier-Stokes equations in exterior domains*, in Vortex Dynamics and Vortex Methods, Lectures in Appl. Math. 28, AMS, Providence, RI, 1991, pp. 100–118.

[5] G.-H. Cottet, *A vorticity creation algorithm for the Navier-Stokes equations in arbitrary domain*, in Navier-Stokes Equations and Related Nonlinear Problems, Plenum, New York, 1995, pp. 335–349.

[6] G.-H. Cottet and P. Koumoutsakos, *Vortex Methods*, Cambridge University Press, Cambridge, 2000.

[7] G.-H. Cottet, P. Koumoutsakos, and M. L. Ould-Salihi, *Vortex methods with spatially varying cores*, J. Comput. Phys., 7 (1999), pp. 164–185.

[8] G.-H. Cottet, B. Michaux, S. Ossia, and G. VanderLinden, *A comparison of spectral and vortex methods in three dimensional incompressible flows*, J. Comput. Phys., submitted.

[9] G.-H. Cottet, M.-L. Ould-Salihi, and M. El-Hamraoui, *Multi-purpose regridding in vortex methods*, Vortex Flows and Related Numerical Methods, ESAIM Proc., 7 (1999), pp. 94–103; also available online from http://www.emath.fr/proc/vol.7.

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

[11] Weinan E and J.-G. Liu, *Vorticity boundary conditions and related issues for finite difference schemes*, J. Comput. Phys., 124 (1996), pp. 368–382.

[12] Weinan E and J.-G. Liu, *Essentially compact schemes for unsteady viscous incompressible flows*, J. Comput. Phys., 126 (1996), pp. 122–138.

[13] J. L. Guermond, S. Huberson, and W. S. Shen, *Simulation of* 2D *external viscous flows by means of a domain decomposition method*, J. Comput. Phys., 108 (1993), pp. 343–352.

[14] J. L. Guermond and H. Z. Lu, *A Domain Decomposition Method for Simulating Advection Dominated, External Viscous Flows*, preprint, 1997.

[15] P. Koumoutsakos and A. Leonard, *High resolution simulations of the flow around an impulsively started cylinder using vortex methods*, J. Fluid Mech., 296 (1995), pp. 1–38.

[16] P. Koumoutsakos, A. Leonard, and F. Pepin, *Viscous boundary conditions for vortex methods*, J. Comput. Phys., 113 (1994), p. 52–61.

[17] P. Koumoutsakos and D. Shiels, *Simulations of the viscous flow normal to an impulsively started and uniformly accelerated flat plate*, J. Fluid Mech., 328 (1996), p. 177–227.

[18] M.-L. Ould-Salihi, *Couplage des méthodes numériques en simulation directe d'écoulements incompressibles*, Thèse de doctorat, Université Joseph Fourier, Grenoble, 1998.

[19] R. Mittal, *Large-eddy simulation of flow past a circular cylinder*, in CTR Annual Research Briefs, 1995, pp. 107–116.

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

[21] P. Orlandi, *Vortex dipole rebound from a wall*, Phys. Fluids A, 2 (1990), pp. 1429–1436.

[22] L. Quartapelle, *Vorticity conditioning in the computation of two-dimensional viscous flows*, J. Comput. Phys., 40 (1981), pp. 453–477.

[23] I. J. Schoenberg, *Cardinal Spline Interpolation*, SIAM, Philadelphia, 1973.

[24] J. H. Strickland, L. A. Gritzo, R. S. Baty, G. F. Homicz, and S. F. Burns, *Fast multipole solvers for three-dimensional radiation and fluid problems*, ESAIM Proc., 7 (1999), pp. 408–417; also available online from http://www.emath.fr/proc/Vol.7.

[25] R. Verzicco and P. Orlandi, *Normal and oblique collision of a vortex ring with a wall*, Meccanica, 29 (1994), pp. 383–391.

[26] C. H. K. Williamson, 2D *and* 3D *aspects of the wake of a cylinder and their relation with wake computation*, in Vortex Dynamics and Vortex Methods, Lectures in Appl. Math. 28, AMS, Providence, RI, 1991, pp. 719–751.