Chapter 6

Numerical Simulation of Vortex-Dominated Flows Using the Penalized VIC Method

Seung-Jae Lee

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/65371

Abstract

Vorticity plays a key role in determining fluid flow dynamics, especially in vortex-dominated flows. Vortex methods, which are based on the vorticity-based formulation of the Navier-Stokes equations, have provided deeper insight into physical reality in a variety of flows using vorticity as a primary variable. The penalized vortex-in-cell (VIC) method is a state-of-the-art variant of vortex methods. In the penalized VIC method, Lagrangian fluid particles are traced by continuously updating their position and strength from solutions at an Eulerian grid. This hybrid method retains beneficial features of pure Lagrangian and Eulerian methods. It offers an efficient and effective way to simulate unsteady viscous flows, thereby enabling application to a wider range of problems in flows. This article presents the fundamentals of the penalized VIC method and its implementations.

Keywords: vorticity, unsteady viscous flow, pressure, vortex-in-cell method, penalization method, vorticity transport equation

1. Introduction

Vorticity is an important derived variable playing both mathematical and physical roles in the solution and understanding of problems [1], irrespective of whether the flow is laminar or turbulent. Vortex methods (VMs) utilize such a vorticity as a main variable of fluid dynamics. VMs are essentially a grid-free approach in which Lagrangian fluid particles are used as basic computational elements. The computational domain is discretized into a set of N fluid particles, which carry time-evolving vorticity.

VMs are based on the vorticity-based formulation of the Navier-Stokes equations; namely, the vorticity transport equation (VTE). There are several advantages in using the vorticity-based formulation. Firstly, it allows a purely kinematical problem to be decoupled from the pressure term. Thereby, the need for solving the pressure is eliminated. It means that pressure computation is not part of the solution procedure. If the pressure field is desired, it can be obtained in an explicit manner using the resolved vorticity and velocity fields. Secondly, the vorticity-based formulation does not contain any frame-dependent terms (for detailed information, refer to [2]). In rotating frame, for example, VTE is only altered by the addition of a constant forcing function in the form of $2\ \Omega_\omega$ where Ω_ω is a time-dependent angular velocity. In the primitive-variable formulation, on the other hand, the mathematical character of the equations changes depending on whether or not the reference frame is inertial. Gresho [3] offers more comprehensive information on the fundamental formulations for incompressible flow simulations. Finally, the vorticity-based formulation is inherently able to better resolve finer scales because vorticity is one order higher than velocity. Also, computations are performed where vorticity-carrying particles exist, thus permitting a substantial reduction in the computational domain.

VMs have a history as old as finite differences. Many inherent benefits were discussed in comprehensive reviews of [4–8]. Admittedly, the primitive-variable and vorticity-based formulations, each have their own advantages and disadvantages. Their numerical characteristics cannot be a measure of whether one formulation is better than another. The choice of the numerical methodology is a matter of intended application. However, it is obvious that VMs have been an attractive alternative to conventional numerical methods based on the primitive-variable formulation, especially in terms of vortex-dominated flow simulations.

VMs track numerically the evolution of vortices in unsteady flow. Their most serious difficulty comes from a computational time of particle-particle interactions. Particles' velocity computed by the interaction can be quantified according to the familiar Biot-Savart law. This requires $O(N^2)$ operations, which is expensive for large N. A reduction in the computational complexity has been achieved using the fast multipole method (FMM) with hierarchical data structures, which have had a giant impact in computations using a Lagrangian framework. FMM, which was first introduced by Greengard and Rokhlin [9], is one of the ten algorithms with the greatest influence of the development and practice of science and engineering in the twentieth century [10]. FMM and its variants significantly reduce the cost of computing the interaction with N particles; namely, $O(N^2)$ to $O(NlogN)$ or $O(N)$ operations (refer to [11] for a short review).

Another approach for speed up in computation of particle velocities is the vortex-in-cell (VIC) method, which was originally developed by Christiansen [12]. This numerical method evaluates a velocity field, instead of the velocity of individual particles. It involves interpolation of physical quantities (vorticity and velocity) between Lagrangian particles and an Eulerian grid. Hence, the VIC method is often referred to as a hybrid method. In the VIC method, velocity field is computed by solving a Poisson equation, $\nabla^2\psi = -\omega$ or $\nabla^2 u = -\nabla \times \omega$. A fast Poisson solver accelerates the computation sufficiently. For example, a fast Fourier transform (FFT)-based solver reduces an operation count to $O(MlogM)$, where M is the number of grid

http://dx.doi.org/10.5772/65371

points. Although the amount of computation is problem dependent, the VIC method is known to be faster than FMM to simulate unsteady viscous flows (refer to [13]).

For treating vorticity diffusion problem in VMs, several different techniques were developed [14]; the core spreading method [15], the random-walk method [16], the particle strength exchange method [17], the deterministic method [18], and the vortex redistribution method [19]. Interestingly, Graham [20] successfully solved the diffusion term of VTE using an Eulerian grid instead of complicated Lagrangian operators (also refer to [21–23]). This means that in the VIC method both convection and diffusion parts can be evaluated on a regular grid. Another important point of concern about VMs is boundary conditions for wall-bounded flows. In recent years, the VIC method has been successfully combined with the penalization method [24], which is useful to enforce boundary conditions at a solid wall (for example, refer to [25–33]). The combination, hereafter called the penalized vortex-in-cell (pVIC) method, greatly simplifies an entire numerical procedure required to construct a solution algorithm, thus leading to a well-performing parallel program. As a consequence, the evolution of both velocity and vorticity fields are evaluated on an Eulerian grid, and then the values for Lagrangian fluid particles are determined from the grid solution using an interpolation scheme. This offers an efficient and effective way to simulate unsteady viscous flows.

The pVIC method to simulate unsteady viscous flows will be discussed further in the following sections. In Section 2, the basic formulations and numerical methods are described. Section 3 describes implementations of the numerical methods and post-procedures for obtaining pressure and force from numerical solutions. Section 4 introduces improved implementations to further efficiently simulate a flow. Section 5 presents numerical results and discussions to validate the presented numerical method. A summary and conclusion is included in Section 6.

2. Basic formulation and numerical methods

For an incompressible unsteady flow of a viscous fluid, conventional VTE is expressed as follows

$$\frac{\partial \omega}{\partial t} + u \cdot \nabla \omega = (\omega \cdot \nabla) u + \nu \nabla^2 \omega \tag{1}$$

where ν is the kinematic viscosity. The evolution of a flow can be evaluated in a fractional step manner. The algorithm of viscous splitting consists of sub-steps in which the convective and diffusive effects are considered. It is thus expressed in a Lagrangian frame as

$$\frac{Dx}{Dt} = u, \frac{D\omega}{Dt} = 0 \tag{2}$$

- Convection part
- Diffusion part

$$\frac{Dx}{Dt} = 0, \frac{D\omega}{Dt} = (\omega \cdot \nabla)u + \nu\nabla^2\omega \tag{3}$$

where $D/Dt = \partial/\partial t + u \cdot \nabla$ is the material derivative. N discrete fluid particles are linearly superposed to approximate the vorticity field as

$$\omega(x,t) = \sum_{p}^{N} \Gamma_p \zeta(x - x_p) \tag{4}$$

where ζ is a mollification of the Dirac-delta function. Each particle is characterized by its position x_p and strength Γ_p, *i.e.* a circulation $\Gamma_p = \int \omega dV \cong \omega_p V_p$ where V_p is a volume (an area in 2D) occupied by a fluid particle.

In the pVIC method, the velocity of the particles and vorticity evolution are evaluated on a uniform grid. For doing that, the particles' own vorticity is first transferred onto the grid by

$$\omega(x_g,t) = \frac{1}{h^3}\sum_{p}^{N} \Gamma_p M_4'\left(\frac{x_g - x_p}{h}\right) \tag{5}$$

where h is the grid spacing. The function M_4' is the third order interpolation kernel [34] defined as

$$M_4'(x) = \begin{cases} 0 & for\,|x| > 2 \\ \frac{1}{2}(2-|x|)^2(1-|x|) & for\,1 < |x| \le 2 \\ 1 - \frac{5}{2}|x|^2 + \frac{3}{2}|x|^3 & for\,|x| < 1 \end{cases} \tag{6}$$

in each coordinate direction. This kernel conserves the 0th, 1st, and 2nd order moments. Vorticity of a single fluid particle is transferred to the nearest 16 grid nodes in 2D and 64 grid nodes in 3D. The resultant vorticity at grid nodes is obtained by summing the contribution of all the particles.

http://dx.doi.org/10.5772/65371

2.1. Convection via vortex-in-cell (VIC) method

In the pVIC method, a velocity vector can be expressed as

$$u = U_\infty + u_\omega = U_\infty + \nabla \times \psi \tag{7}$$

where U_∞ is a free stream velocity and u_ω represents a rotational velocity. ψ is called as a vector potential in 3D and a stream function in 2D. Taking the curl of Eq. (7) yields the Poisson equation $\nabla^2\psi = -\omega$ linking vorticity to the vector potential and in turn to the velocity. ψ is computed on a uniform grid using a FFT-based solver. The grid for solving the Poisson equation can define a compact computational domain with non-homogeneous Dirichlet boundary conditions. The boundary condition of a domain Ω can be approximated using a Green's function approach [28], which is given by

$$\psi_b = \frac{1}{4\pi}\int_\Omega \frac{\omega_p}{\left|x_p - x_b\right|} dV = \frac{1}{4\pi}\sum_p^N \frac{\Gamma_p}{\left|x_p - x_b\right|} \tag{8}$$

where $x_p \in \Omega$, $x_b \in \partial\Omega$, and $x_p \neq x_b$. In 2D, it follows that

$$\psi_b = \frac{-1}{2\pi}\sum_p^N \Gamma_p ln\left(\left|x_p - x_b\right|\right) \tag{9}$$

Finally, rotational velocities on the nodes of the grid are computed from the definition $u_\omega = \nabla \times \psi$ using a finite difference scheme.

In 2D, two unknowns (two components of a velocity vector) are reduced to a single unknown ψ_z (the scalar stream function) without any loss of generality. There are no problems with the continuity equation; the velocity field is automatically divergence-free. In 3D, however, a total of six unknowns are required to be solved instead of the usual four of the primitive-variable formulation. Unlike the stream function in 2D, the vector potential in 3D is far from being uniquely defined. The 3D formulation requires that the vorticity, velocity, and vector potential are all divergence-free; $\nabla \cdot \omega = 0$, $\nabla \cdot u = 0$, and $\nabla \cdot \psi = 0$. This is a new difficulty for 3D flow simulation. Among these three conditions, the second is satisfied by letting $u = \nabla \times \psi$ and the third is a consequence of the first. To enforce the first, the well-known projection scheme can be applied. The divergence-free of a vorticity field is accomplished by $\omega - \nabla F$, where F is a scalar field which is obtained by solving $\nabla^2 F = \nabla \cdot \omega$.

2.2. Diffusion via penalization method

The penalization method has been used to take into account a solid body immersed in a flow. Among fictitious domain methods, the penalization method is very easy to implement, robust and efficient. This consists only in adding a penalty term in the momentum equations to

represent a solid body. For example, the L^2 penalization consists in adding a damping term on the velocity in the Navier-Stokes equations [24]. The penalty velocity satisfies Darcy's law in terms of a Neumann boundary condition on the pressure. Angot et al. [24] rigorously showed that the penalized equations can be used with confidence since penalty error is always negligible in the face of the error of approximation. In the penalization method, a solid body is regarded as a porous medium where the permeability is infinite in the fluid part and tends to zero in the solid part. Both the solid and fluid domains are solved directly without addition of any boundary condition at a solid wall. The penalization method can replace complicated algorithms to implement boundary conditions at a solid wall.

In the pVIC method, a penalty term is added into VTE as follows

$$\frac{D\omega}{Dt} = (\omega \cdot \nabla)u + \nu\nabla^2\omega + \nabla \times \left[\lambda\chi(u_s - u)\right] \tag{10}$$

where u_s is the velocity of the solid body and χ denotes a mask function that yields 0 in the fluid and 1 in the solid. The penalty parameter λ is equivalent to an inverse permeability. In the case of explicit Euler time discretization, λ must satisfy $\lambda\Delta t < O(1)$ to ensure stability. Alternatively, an implicit penalty term can expressed as

$$\frac{\Delta\omega}{\Delta t} = \nabla \times \left[\lambda\chi\left(u_s^n - \tilde{u}^{n+1}\right)\right] \tag{11}$$

where

$$\tilde{u}^{n+1} = \frac{u^n + \lambda\Delta t\chi u_s^n}{1 + \lambda\Delta t\chi} \tag{12}$$

The implicit scheme is unconditionally stable [27]. This allows to use large λ regardless of Δt for accurate solutions near a solid body. Based on implicit penalization scheme, the VTE is rewritten as

$$\frac{D\omega}{Dt} = (\omega \cdot \nabla)u + \nu\nabla^2\omega + \nabla \times \left[\frac{\lambda\chi}{1 + \lambda\Delta t\chi}(u_s - u)\right]. \tag{13}$$

Each term can be solved using a finite difference scheme. For example, the penalty term can be discretized by a first-order centred difference scheme. To reduce an error, one can use a smoothed mask function [35]. The stretching term, taking its alternative form $(\omega \cdot \nabla^T)u$ (the so-called transpose scheme), can be computed with the fourth-order central difference scheme. The transpose scheme is advantageous due to its conservative and accurate properties [36].

http://dx.doi.org/10.5772/65371

The diffusion term, $\nu\nabla^2\omega$, can be discretized by a 27-point isotropic Laplacian scheme which is less sensitive to the grid orientation. In 2D, the diffusion term can be discretized by a classical 9-point finite difference scheme.

2.3. Numerical procedure

For high performance computing, the message passing interface (MPI) can be used on a distributed-memory parallel system. For parallel computing, a computational domain Ω is decomposed by splitting it into several subdomains, with one subdomain per processor. Fluid particles are distributed among the subdomains depending on their positions.

In the pVIC method, convection, penalization, diffusion, and stretching (only in 3D) parts are sequentially evaluated on each subdomain. A numerical procedure follows that

a. Each processor interpolates the vorticity of its own particles into grid nodes through the M'_4 interpolation kernel.

b. Each processor computes ψ_b with its own particles' vorticity using the Green's function approach. Then, ψ_b for solving the Poisson equation is determined by a summation of the values obtained by each processor.

c. The Poisson equation $\nabla^2\psi = -\omega$ is solved using the FFT-based solver. Velocities on the grid nodes are computed from the resultant ψ, using a finite difference scheme.

d. Each processor evaluates the penalization term on the nodes using a finite difference scheme, and then the diffusion and stretching terms are computed. Divergence-free condition of vorticity field is enforced using the projection method.

e. Each processor interpolates the velocity and vorticity on the grid back to its own particle positions through the M'_4 interpolation kernel. Finally, the particles' position and strength are updated using an appropriate time integrator.

Some particles that have left a subdomain are assigned to an adjacent processor. If needed, the computational domain is increased depending on particle distribution. The whole process is repeated for the next time-step.

Particle redistribution is conducted every few time steps to ensure particle overlap. Concurrently, with this step, particles with negligibly small vorticity are discarded to prevent unnecessary increase in particle population. The resultant particles are filtered with a relative criterion; $\Gamma_i < C_\omega |\Gamma|_{max}$ where C_ω is a cut-off criterion. The discarded strengths were shared to the particles remaining in order to guarantee the conservation of circulation in the flow. It is found that the cut-off criterion has a significant influence on global force evaluation, rather than on vorticity and velocity fields. This will be further discussed in Section 3.4.

2.4. Numerical stability

In the pVIC method, fluid particles are traced by continuously updating their position and strength (vorticity or circulation) from the grid solution. The method retains the key feature

of pure Lagrangian vortex methods; namely, the convection part disappears from VTE. As noted in [13], the particle convection is linearly unconditionally stable. The convection part does not impose any stability constraint but relies on the Lagrangian particle convection. The nonlinear stability condition that particles do not collide during one time-step (Δt) based on the rate of strain of a flow reads $\Delta t \leq C_1 |\nabla u|^{-1} \approx C_1 |\omega|^{-1}$. Also, the maximum allowable Δt is constrained by the diffusion Courant-Friedrichs-Lewy (CFL) condition, $\Delta t \leq C_2 h^2/\nu$, because the vorticity field is computed on a grid using a finite-difference scheme. This is less restrictive than the convective CFL condition for pure finite-difference methods, $\Delta t \leq Ch|u|^{-1}$. In practice, the nonlinear stability condition for the convection is much less demanding since the redistribution of particles to regular positions is periodically carried out. Δt for the diffusion is regarded as the physical time-step for an entire flow simulation. This permits the use of a relative large time-step compared to pure finite-difference methods.

3. Implementations

3.1. Domain decomposition for parallel computing

To decompose a computational domain in parallel, there are two typical strategies. One is that a computational domain is equidistantly decomposed. The other is that a computational domain is decomposed so as to assign the same number of particles to each processor; consequently, the subdomains have different sizes (for detailed information, refer to [37]). In our 2D flow simulations, the latter was typically faster than the former, whereas in 3D the former outperformed the latter. This is related to both balancing of computation load for Dirichlet boundary conditions and communication load of grid assembly to solve a Poisson equation.

In our pVIC code, each processor has all the boundary nodes to minimize communications among processors and boundary conditions at are determined by summing the contribution of all the particles. The same number of particles among processors ensures near-perfect balanced computation and communication loads, even if it causes unbalanced communication for grid because of different domain sizes. Since local computation of boundary conditions in 2D cases was typically more expensive than inter-processor communication, we found the second approach to be much faster. In 3D cases, however, balancing of communication for grid assembly becomes more important than that of computation of boundary conditions. This is due to a significant reduction in computing time by the spline approximation method (which will be introduced in Section 4.2). As a result, the equidistant domain decomposition can be a better choice for 3D flow simulations.

3.2. Time integrator

To move fluid particles with their own velocity, predictor-corrector methods are usually used. **Table 1** shows a summary of time integration schemes used in the literature, but the detailed scheme was not explained in most of papers.

http://dx.doi.org/10.5772/65371

Author(s)	Convection part	Diffusion part	Remarks
Ould-Salihi et al. [38]	RK2	EE	2D and 3D VIC
Coquerelle and Cottet [39]			2D and 3D pVIC
Rasmussen et al. [33]			2D pVIC
Lee at al. [32]			2D pVIC
Mimeau et al. [30]			2D pVIC
Morency et al. [40]	RK4	EE	2D pVIC
Cottet and Poncet [13]			3D VIC
Kosior and Kudela [41]			3D VIC
Cocle et al. [23]	RK2 and LF2	RK2 and AB2	3D VIC
Lonfils and Winckelmans [42]			3D VIC

The second-order Runge-Kutta method (RK2), the fourth-order Runge-Kutta method (RK4), the second-order Leapfrog method (LF2), the second-order Adams-Bashforth method (AB2), the first-order explicit Euler method (EE)].

Table 1. Comparison of numerical time-integration schemes reported in the available literature.

A popular Runge-Kutta method is known as a single stepping scheme with multiple stages per step. For example, the method can be expressed as

$$\boldsymbol{x}_p^* = \boldsymbol{x}_p^n + \boldsymbol{u}_p^n\left(\boldsymbol{x}_p^n, \boldsymbol{\omega}_p^n\right)\Delta t \tag{14}$$

$$\omega_p^{n+1} = \omega_p^n + \frac{d\omega_p^n}{dt}\Delta t \tag{15}$$

$$\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + 0.5\left[\boldsymbol{u}_p^n\left(\boldsymbol{x}_p^n, \boldsymbol{\omega}_p^n\right) + \boldsymbol{u}_p^*\left(\boldsymbol{x}_p^*, \boldsymbol{\omega}_p^{n+1}\right)\right]\Delta t \tag{16}$$

where the superscript * indicates an intermediate stage [32, 43]. Note that the velocity field is evaluated twice in finding the new position of x_p^{n+1}. If a higher time integration scheme is used, more involved computations are needed. It is obvious that a higher-order integrator helps minimize error in particle tracking, compared to one evaluation. However, more evaluations of velocity field cause an increase in computational time.

Interestingly, Rossinelli et al. [44] evaluated particles' velocity using a modified explicit scheme as follows

$$\begin{aligned} \boldsymbol{x}_p^* &= \boldsymbol{x}_p^n + 0.5\Delta t\boldsymbol{u}_p^n\left(\boldsymbol{x}_p^n, \boldsymbol{\omega}_p^n\right) \\ \boldsymbol{x}_p^{n+1} &= \boldsymbol{x}_p^n + \Delta t\boldsymbol{u}_p^n\left(\boldsymbol{x}_p^*, \boldsymbol{\omega}_p^n\right) \end{aligned} \tag{17}$$

and a notable feature is that velocity field is evaluated only once although this scheme can be still considered as a kind of the Runge-Kutta method. Particles' velocity at the intermediate stage is computed using interpolation. This provides a considerable saving in computational time and memory consumption. Our research group investigated the feasibility of different integrators such as the Euler scheme with several stages, the midpoint rule, and the Simpson rule, as follows

$$-\text{Euler method} \quad x_p^{n+1} = \sum_{i=0}^{n-1}\left[x_p\left(t + i\delta t\right) + \delta t x_p u_p^n\left(t + i\delta t\right)\right] \tag{18}$$

where $\delta t = \Delta t/n'$

$$-\text{Midpoint rule} \quad x_p^{n+1} = x_p^n + \frac{\Delta t}{2}[u_p^n(x(t)) + u_p^n(x(t+\Delta t))] \tag{19}$$

$$-\text{Simpson rule} \quad x_p^{n+1} = x_p^n + \frac{\Delta t}{6}[u_p^n(x(t)) + u_p^n(x(t+\Delta t/2)) + u_p^n(x(t+\Delta t))] \tag{20}$$

Akin to Rossinelli et al. [44], particles' velocity at intermediate position was computed using interpolation from the values at neighbouring grid nodes. A benchmark problem was a flow around a sphere at Reynolds numbers of 50, 100, and 200. Overall computational time was about two times faster in the midpoint and Simpson rules, compared with the typical RK2 method in which velocity field is computed twice. The midpoint and Simpson rules provided comparable accuracy, and the difference in drag force was typically <0.3%. This approach seems to be useful. However, an essential prerequisite is that velocity field is little or not changing during Δt. This can require a smaller Δt, especially in high Reynolds number flows. Since the accuracy of a time integrator for tracing particles is dependent upon the change of a flow field in both time and space, a time integration scheme should be determined carefully.

3.3. Fast Poisson solvers

To solve a Poisson equation, the fast Fourier transform (FFT) and multigrid (MG) are utilized most widely. For a uniform grid, the behaviour of both solvers is well-known. In serial computation, a FFT solver requires $O(MlogM)$ for M grid points and a MG solver requires $O(M)$ operations. In an ideal parallel computation with zero cost communication, the FFT and MG solvers require $O(logM)$ and $O(log^2M)$, respectively (for detailed information, refer to [45]). Here, the performance of two solvers is compared under exactly the same condition; a rectangular computational domain with equidistant nodes. Both solvers were developed in our research group. The FFT solver is based on an open-source library called Fastest Fourier Transform in the West (FFTW) [46], and the MG solver is based on full multigrid algorithms provided in Section 19.6 of the Numerical Recipes [47] (also refer to [41]). Here, the algorithm

http://dx.doi.org/10.5772/65371

is briefly introduced. The full multigrid algorithm starts with the coarsest possible grid and then proceeds to finer grids. It performs one or more V-cycles at each level before proceeding down to the next finer grid. This means that the full multigrid algorithm produces a solution for each level. We used the red-black Gauss-Seidel scheme as a smoothing operator, bilinear interpolation for a prolongation operator, and half-weighting for a restriction operator. The full multigrid cycle can be seen in **Figure 1**.

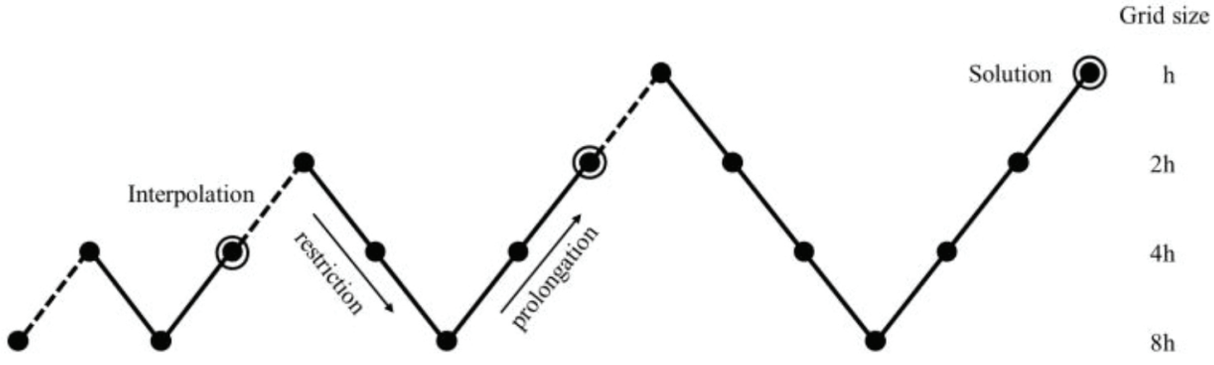


Figure 1. A computational cycle used in the MG solver.

To compare the two solvers, a 2D flow around a circular cylinder at a Reynolds number of 550 was considered as a benchmark problem. The MG solver with four levels was tuned by choosing one pre-smoothing sweep and two post-smoothing sweeps. **Figure 2a** shows a comparison of numerical results between the FFT and MG solvers. The MG solver performed comparably to the FFT solver in terms of accuracy. As expected, however, the MG solver was slower than the FFT solver as shown in **Figure 2b**. This is due that the MG solver is a kind of iterative method and thus it is an order magnitude more expensive compared to the FFT solver as a non-iterative method. Since, so far, most of VIC codes solve the Poisson equation in a computational domain of simple shape such as a square or cube, the FFT solver as a non-iterative method is thought to be the best choice. However, it is worth noting that the MG solver can be applied to locally refined grids in a straightforward way. Adaptive mesh refinement is useful both to increase accuracy and to decrease computational cost in comparison with a uniform grid.

3.4. Force calculations

From the fact that the integral of vorticity moments becomes the change of the momentum, fluid force exerted on the solid body (F_s) is expressed as

$$F_s = -\rho \frac{d}{dt}\left(\frac{1}{N-1} \int_V x \times \omega dV \right) \tag{21}$$

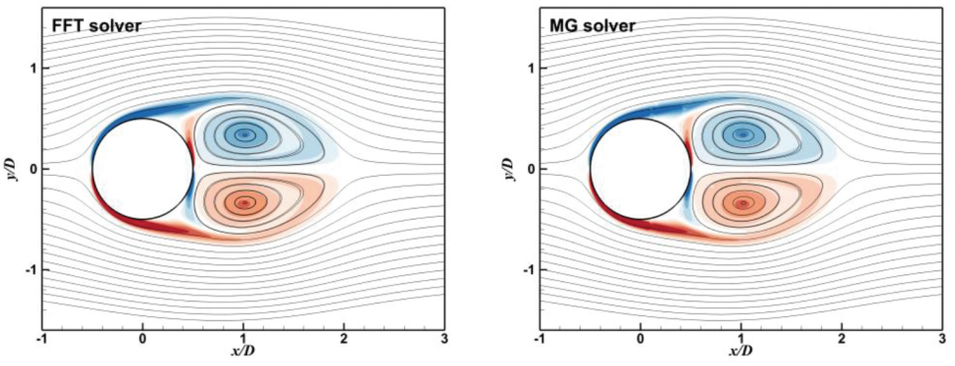
where N is the dimension of the space. The volume integral can be replaced by the summation of the moments for all the particles. Also, the adoption of the penalization method introduces another approach for the force evaluation. One can obtain the force with the velocity field inside the body as follows:

$$F_s = -\rho \int_V \lambda \chi \left(u_s - u \right) dV \tag{22}$$

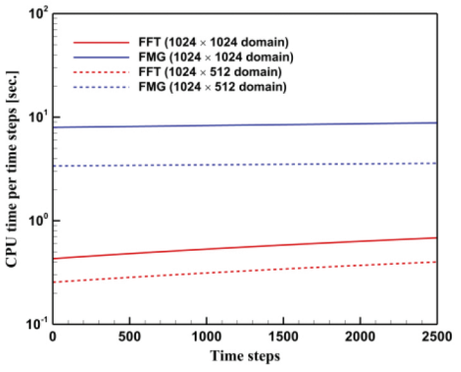
where the penalization term itself is included in the integral as the time-change of the momentum. For the implicit penalization scheme, it can be rewritten as

$$F_s = -\rho \int_V \frac{\lambda \chi}{1 + \lambda \chi \Delta t} \left(u_s - u \right) dV \tag{23}$$

The integral over the whole domain is actually confined to the interior of the body due to the definition of χ.



(a) vorticity contours and streamlines

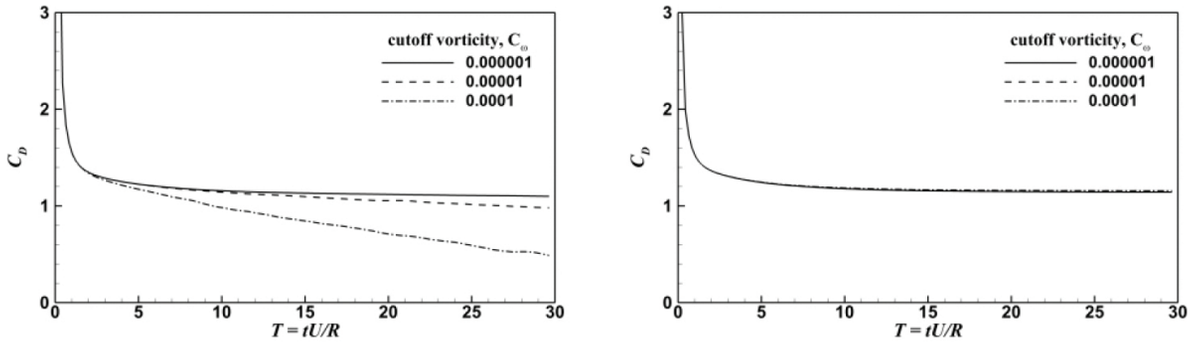


(b) CPU times per time steps

Figure 2. Comparison of flow fields and CPU times between the FFT and MG solvers. (a) Vorticity contours and streamlines and (b) CPU times per time steps.

http://dx.doi.org/10.5772/65371

In 2D flow simulations, the drag coefficients have been reported in the literatures [30, 37, 44, 48]. They have evaluated the drag coefficient using one or both of the approaches. Rasmussen et al. [33] and Gazzola et al. [31] addressed that the discrepancy in the drag coefficient between the two approaches were not significant. In 3D, however, it was found that the method based on the vorticity moment is unstable occasionally; that is, a large C_ω can make it incorrect even if there is little difference both in vorticity and velocity fields. **Figure 3** shows the drag coefficient (C_D) of a sphere at a Reynolds number of 100 computed by the two different approaches employing three different cut-off criteria; $C_\omega = 10^{-4}$, 10^{-5}, and 10^{-6}. C_D is defined by $F_x/(0.5\rho U_\infty A)$ with a sectional area $A = \pi R^2$. The drag coefficient computed with vorticity moments were converged at $C_\omega = 10^{-6}$, whereas the drags computed with the penalized velocity were nearly the same in all the tested cases. This describes that force evaluation using the vorticity moment should be taken with special care. As a result, the force evaluation with the penalized velocity can be a better option, especially in 3D simulations. This enables one to avoid an excessive increase in the number of particles, thus reducing the computational time and memory. In the cases with $C_\omega = 10^{-4}$, 10^{-5}, and 10^{-6}, there were 1.0, 1.9, and 3.5 million particles, respectively.



(a) with the vorticity moment (b) with the penalized velocity

Figure 3. Drag coefficients calculated with two different approaches: (a) vorticity moment and (b) penalized velocity.

3.5. Pressure field

As discussed previously, pressure field is independently computed from the entire solution procedure and explicitly obtained by solving a pressure Poisson equation, $\nabla^2 H = \nabla \cdot (u \times \omega)$. Lee [49] introduced an integral approach to obtain the pressure field at any fixed time as follows:

$$H = \int_S \left(\frac{\partial H}{\partial n} G - H \frac{\partial G}{\partial n} \right) dS + \int_V \nabla \cdot \left(u \times \omega \right) G dV \tag{24}$$

where G is the Green function solution. H is the Bernoulli function defined as

$$H = \frac{p - P_\infty}{\rho} + \frac{1}{2}\left(u^2 - U_\infty^2\right) \tag{25}$$

where p is the static pressure above the reference pressure P_∞ and ρ is fluid density. The value of H on the body surface is computed using a boundary integral approximation. A mathematical identity for a vector or scalar field is used to define field values of a quantity of interest, which involves an integral of singularities distributed over the surface and throughout the field. The boundary integral approach has been successfully established. However, it has disadvantages such as the presence of singular Green's kernels. Special attention is needed to accurately compute boundary integrals around a singular point. The higher mathematical complexity is needed to get a usable computational formulation. The matrices that result from the integral method are asymmetrical, and they are not easy to solve. Furthermore, this approach takes long time of computation.

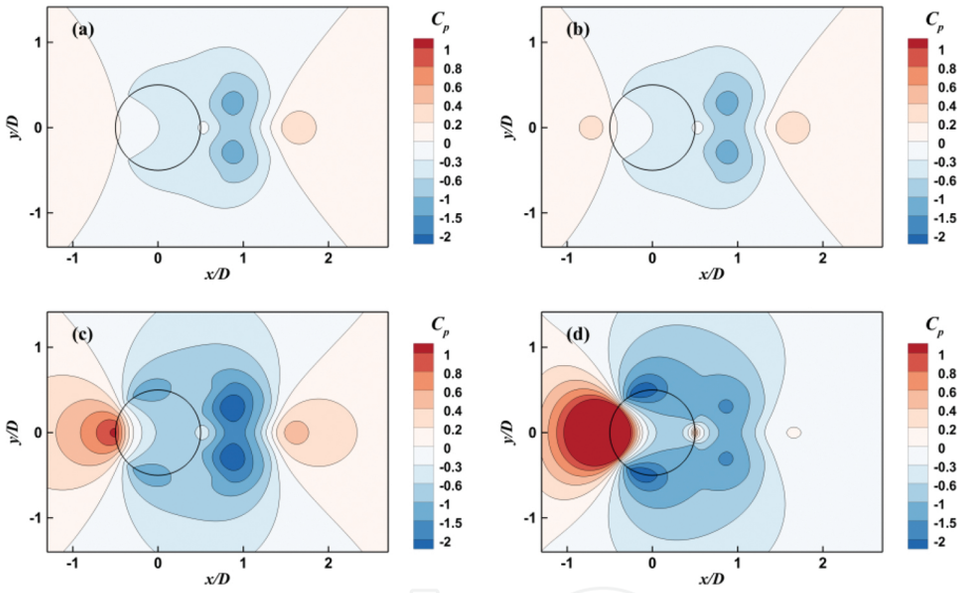


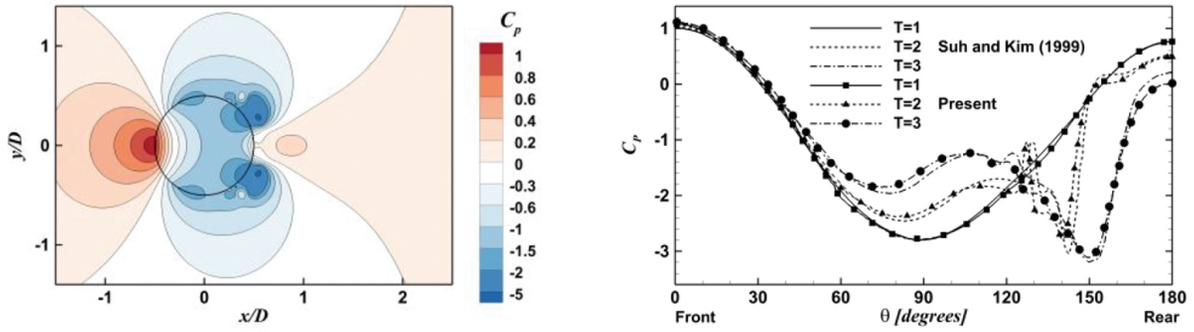
Figure 4. Pressure distributions computed with different penalty parameters for pressure; (a) $\lambda' = 0$, (b) $\lambda' = 0.1/\Delta t$, (c) $\lambda' = 1/\Delta t$, and (d) $\lambda' = 10/\Delta t$. Note that the figures are reprinted with permission from [28].

Alternatively, pressure field can be evaluated based on the penalized Navier-Stokes equation [28]. Satisfying the continuity, the Poisson equation for pressure can be expressed as

$$\nabla^2 H = \nabla \cdot \left(u \times \omega\right) + \lambda' \left(u_s - u\right) \cdot \nabla \chi \tag{26}$$

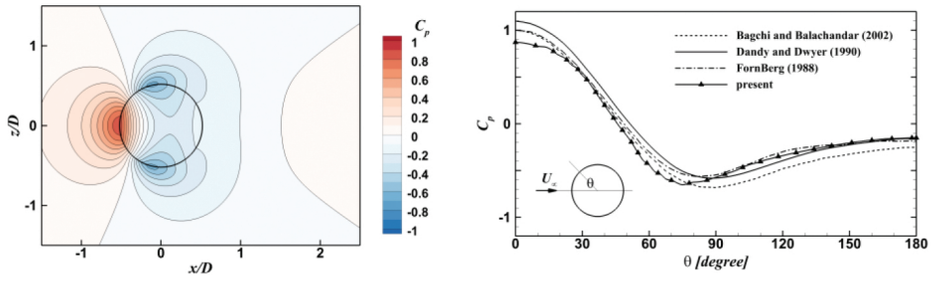
where λ' is distinguished from the λ by its order of magnitude. This equation is solved on the grid using a fast Poisson solver. The boundary condition for H can be given as homogeneous Dirichlet type. **Figure 4** shows pressure distributions around a circular cylinder for a Reynolds number of 550, which are computed using different orders of magnitude for λ'; $\lambda' = 0$, $0.1/\Delta t$,

$1/\Delta t$, and $10/\Delta t$. The assumption of $\lambda^{'} \cong 1/\Delta t$ is thought to be reasonable. With the assumption, Lee et al. [28] successfully computed pressure field around a 2D cylinder as shown in **Figure 5**. It is still valid in 3D flow simulations as shown in **Figure 6**. This approach is quite efficient, but an accurate approximation of H at domain boundaries still remains a problem.



(a) pressure contour at $T = t\,U_\infty/R = 3$ (b) pressure coefficient at a solid surface

Figure 5. Pressure distribution around a circular cylinder at a Reynolds number of 9500. Note that the figures are reprinted with permission from [28]. (a) Pressure contour at $T = tU_\infty/R = 3$ and (b) pressure coefficient at a solid surface.



(a) pressure contour (b) pressure coefficient at a solid surface

Figure 6. Pressure distribution around a sphere for a Reynolds number of 100 at $T = tU_\infty/R = 30$. (a) Pressure contour and (b) pressure coefficient at a solid surface.

4. More efficient implementations

4.1. Multiple domains

In the pVIC method, the size of a computational domain relies on the distribution of fluid particles. A large domain leads to an increase in computational memory required for a grid. A multiple domain approach can be considered to handle much more particles with a limited computational memory. The entire domain Ω can be defined as the union of physical subdomains, covering all the fluid particles; that is, $x_p \in \Omega$ where $\Omega = \Omega_1 \cup \cdots \cup \Omega_{N_D}$. The number of small domains, N_D, is not constant and depends on a spatially evolving flow. When the domain

size exceeds a certain limit, a new domain is created. For example, the first domain Ω_1 includes a solid body and the others are sequentially located downstream from the body as shown in **Figure 7**. Note that this approach differs from the domain decomposition for parallel computations. That is, each small domain Ω_i is again decomposed into subdomains for parallel computations; $\Omega_i = \Omega_{i_1} \cup \cdots \cup \Omega_{i_{N_P}}$ where N_P is the number of processors

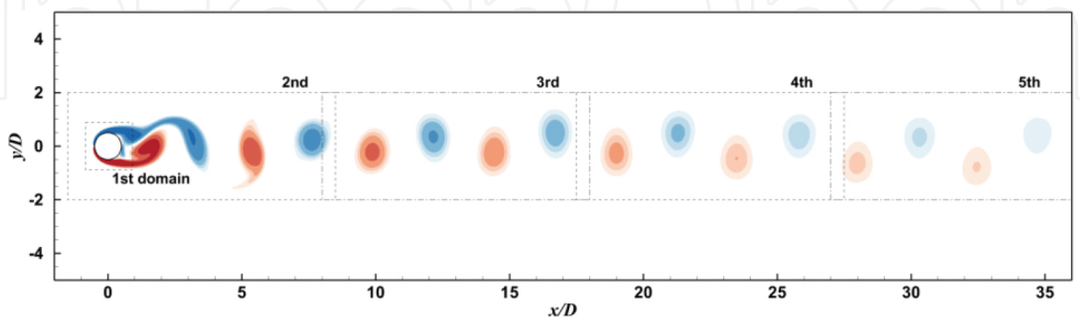


Figure 7. Vorticity contour around a circular cylinder for a Reynolds number of 185.

In Ω_i, vorticity and velocity fields are computed according to the numerical procedure given in Section 2.3. There is no dependency between the small domains since domain boundary conditions of Ω_i are computed by summing the contribution of all the particles. Hence, independent domains ensure a relatively small use of computational memory during a flow simulation, compared to an approach based on parent and child grids [23]. Also, the multiple domain approach enables to have different resolutions among small domains. For example, the grid spacing of Ω_i can be defined as either $h_i = \epsilon$ (single-resolution) or $h_i = 2^n\epsilon$ (multi-resolution) where ϵ denotes the particle size. The particle strength can be transferred into the nodes as follows:

$$\omega\left(x_g, t\right) = \frac{1}{h_i^3}\sum_{p}^{N}\Gamma_p M_4'\left(\frac{x_g - x_p}{h_i}\right) \tag{27}$$

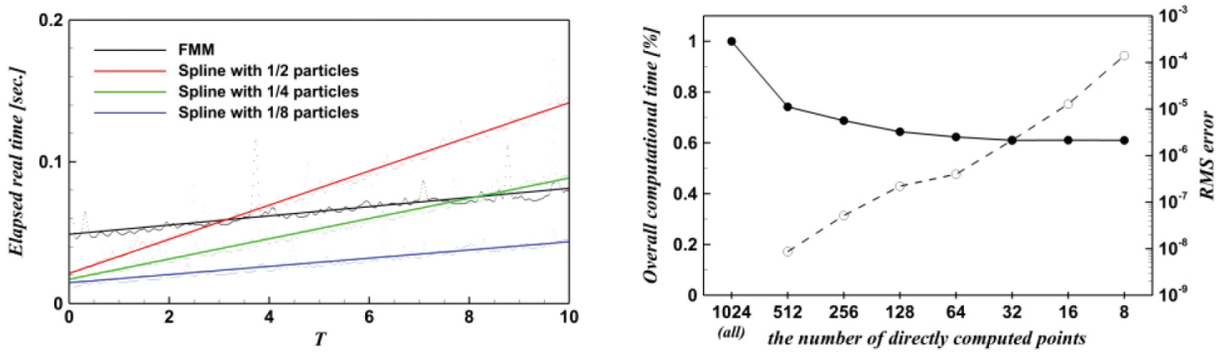
Although this approach ensures the small consumption of computational memory, it causes an increase in computational time because of an increase in the number of domain boundaries. Fast computations of ψ at domain boundaries will be discussed in Section 4.2.

4.2. Approximation of far-field conditions

For simplicity, consider a single domain Ω. The resultant values at $\partial\Omega$ are determined by summing the values computed by all processors. Each processor requires $O(M_bN/P)$, where M_b and N/P denote the number of boundary nodes and its owned particles, respectively. Note that N/P can be reduced by an increase in the number of processors (P), whereas M_b is a constant depending on the size of Ω. M_b can be critical to determine the computational time. In practice,

http://dx.doi.org/10.5772/65371

a substantial part of the overall computational time is spent on the calculation of ψ_b at domain boundaries for solving Poisson's equation. Lee et al. [37] attempted a fast computation of ψ_b using splines. They found that ψ_b varies very smoothly at boundaries of a domain. This feature permits the use of spline interpolation. In 2D, for example, one spline curve enables to represent one side of the computational domain. ψ_b at the part of the boundary nodes (M_{b_D}) is computed by pure direct summations, and then values at the other nodes (M_{b_S}) can be approximated using a spline curve. Here, $M_b = M_{b_D} + M_{b_S}$. Using spline, approximation leads to a reduction in M_{b_D}, which is directly linked to the reduction in the computational time.



(a) computational time for boundary values (b) overall computational time and error

Figure 8. CPU time in numerical simulation using the spline approximation. Note that the FMM is based on a tree level of 3 and an expansion degree of 10. The figures are reprinted from [50]. (a) Computational time for boundary values and (b) overall computational time and error.

Figure 8 shows CPU times elapsed in computations of boundary values using the spline approximation and the FMM [50]. The numerical simulation for a flow past a circular cylinder at Reynolds number of 550 was carried out using 16 CPUs with 4 GB of memory per processor. Once one-eighth of all the boundary points are equidistantly chosen as points for the direct summation, the spline approximation approach becomes faster than the FMM that is modified to compute only boundary values. In this case, the spline approximation approach has accuracy comparable to the FFM. A weak point of the FMM is that it has a quad-tree (oct-tree in 3D) data structure to hierarchically subdivide the computational domain. Each processor in a distributed memory parallel system must have a sufficient amount of memory for tree data structures. A required memory in the FFM depends on both the number of particles and grid nodes.

In 3D, the bi-cubic interpolation can be used instead of the bi-cubic spline (for detailed information on interpolation methods, refer to [47]). According to prior tests to assess this approach, the former was more accurate for our purpose. For example, for computing the stream functions at one side of the cube with 256 × 256 nodes from 3 million randomly distributed particles, the trial using the bi-cubic interpolation method took approximately 1 s when 16 × 16 nodes were used for direct summations, whereas the fully direct evaluations took 194 s in average. The error was typically <0.01%.

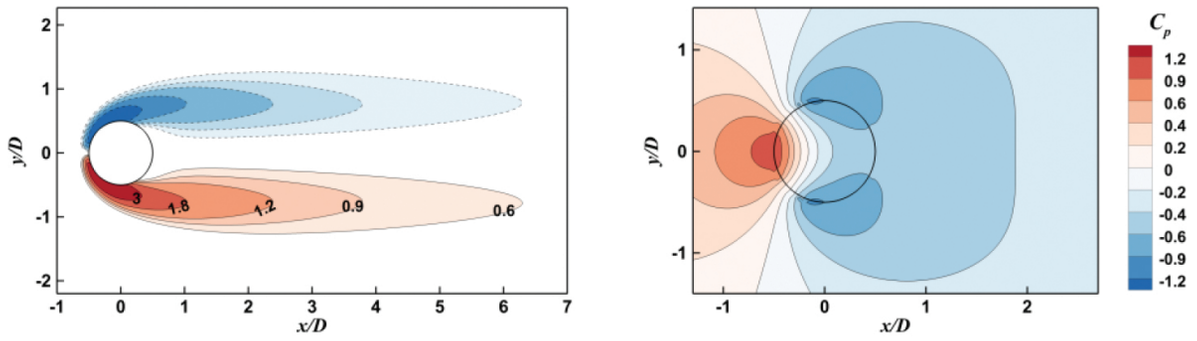
5. Numerical validations

5.1. Flows past an impulsive started circular cylinder

The impulsively started circular cylinder problem is a good prototype to validate a numerical method. The Reynolds number, $Re = U_\infty D/\nu$, based on the free stream velocity, U_∞ and the diameter of the cylinder, D, is selected to be 40, 550, 3000, or 9500. In the simulations, no symmetry constraint is imposed. The dimensionless time is based on the radius ($R = 0.5D$) of the cylinder; $T = tU_\infty/R$. The penalization parameter λ is fixed to 10^8.

5.1.1. Reynolds number of 40

The numerical parameters are $h = 0.04$ and $\Delta t = 0.016$, which are determined through the stability condition $\Delta t = h^2/\nu$. At a Reynolds number of 40, it is well-known that the flow reaches a steady state. A pair of stationary recirculating wakes develops behind the cylinder. The wake length, L/D, is 4.16 and separation angle, θ_s, is 51.7°, respectively. Fornberg [51] made a similar remark and gave $L/D = 4.48$ and $\theta_s = 51.5°$ experimentally. The drag coefficient is computed as $C_D = 1.483$. This is close to the experimentally measured value of 1.498 in [52]. **Figure 9** shows pressure coefficient (C_P) distribution around the cylinder body. The pressure is continuous through the cylinder body, and there is no Gibb's oscillation associated with the discontinuity at the body surface.



(a) vorticity contour (b) pressure contour

Figure 9. Vorticity and pressure contours around a circular cylinder at a Reynolds number of 40. (a) Vorticity contour and (b) pressure contour.

5.1.2. Reynolds number of 550

The simulation parameters are $h = 0.005$ and $\Delta t = 0.002$. Simulation was carried out until $T = 7$ to validate the present formulation in the early time stage after the impulsive start. The number of fluid particles ranged from approximately 30,000 to 150,000, and the total computation time is approximately 4 h on 8 CPUs (Intel Xeon64 3.3 GHz). Numerical results are presented for vorticity contours and streamlines in **Figure 10**. As time increases, a pair of secondary symmetric vortices appears and become stronger. The so-called bulge phenomena observed

http://dx.doi.org/10.5772/65371

experimentally by Bouard and Coutanceau [53] is well captured by our numerical simulation. The vortex core position indicated by the coordinates a/D and b/D is investigated. In this flow simulation, the abscissa $a/D = 0.34$ and ordinate $b/D = 0.27$, and the wake length $L/D = 0.82$. In [53], $a/D = 0.36$, $b/D = 0.28$, and $L/D = 0.85$.

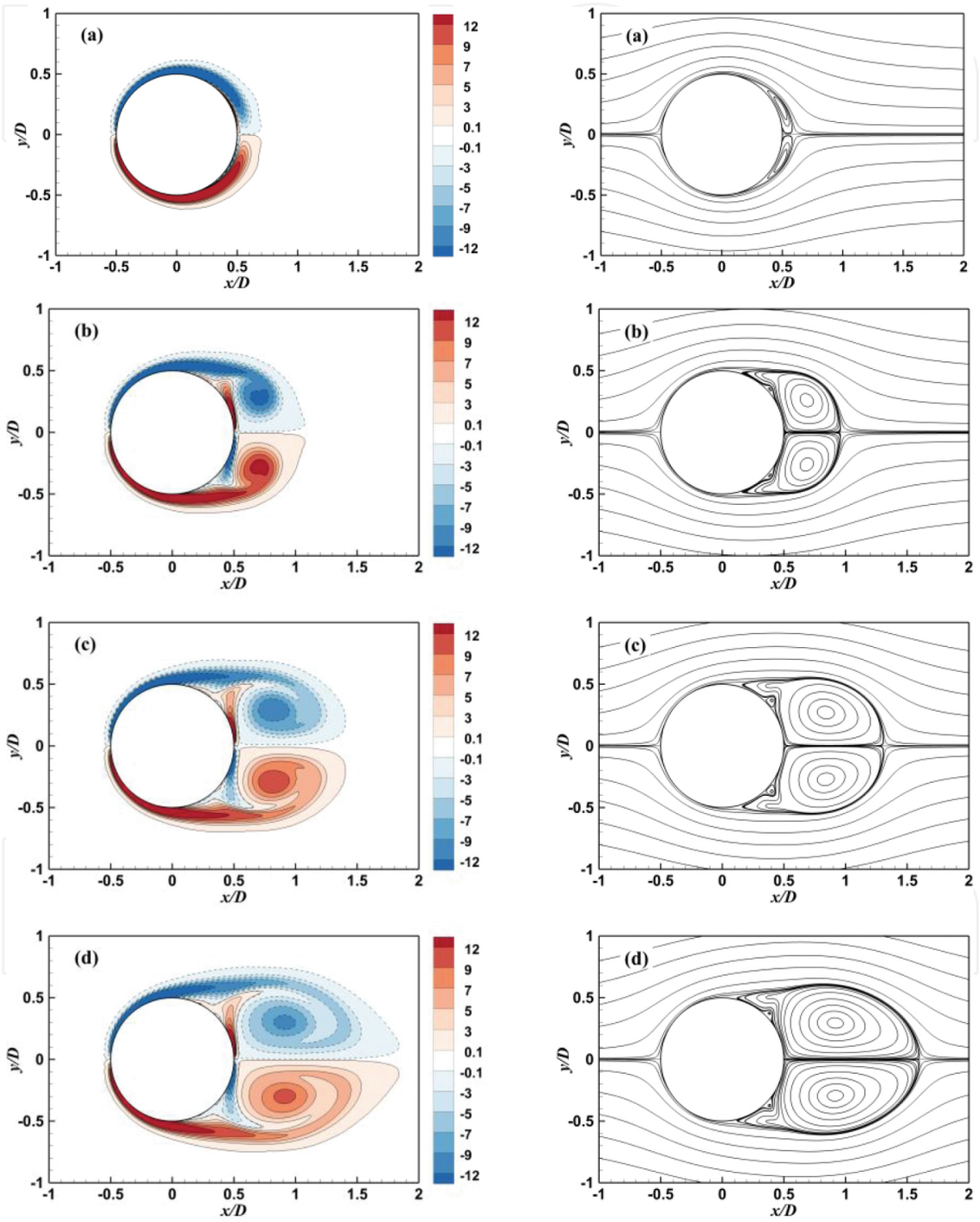


Figure 10. Vorticity contours (*left*) and streamlines (*right*) at a Reynolds number of 550; (a) $T = 1$, (b) $T = 3$, (c) $T = 5$, and (d) $T = 7$.

Figure 11 shows the time evolution of the drag coefficient for an impulsively started flow around a two-dimensional cylinder for Re = 550 calculated with the pVIC method, compared with results from the short time asymptotic solution of Bar-Lev and Yang [54], the vortex method result of Koumoutsakos and Leonard [55], and the VIC method result of Kudela and Kozlowski [56].

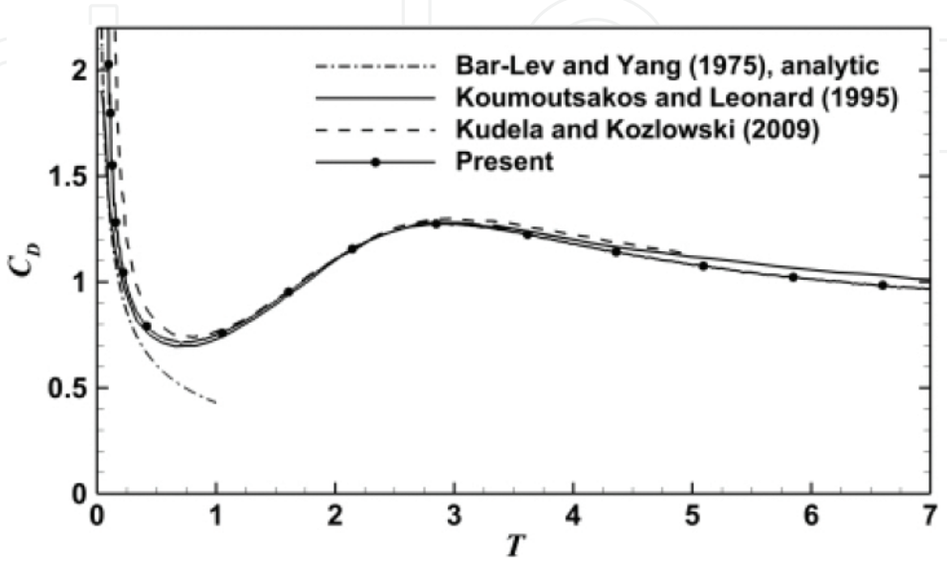


Figure 11. Time evolution of the drag coefficient for a cylinder at a Reynolds number of 550. Note that the figure is reprinted with permission from [28].

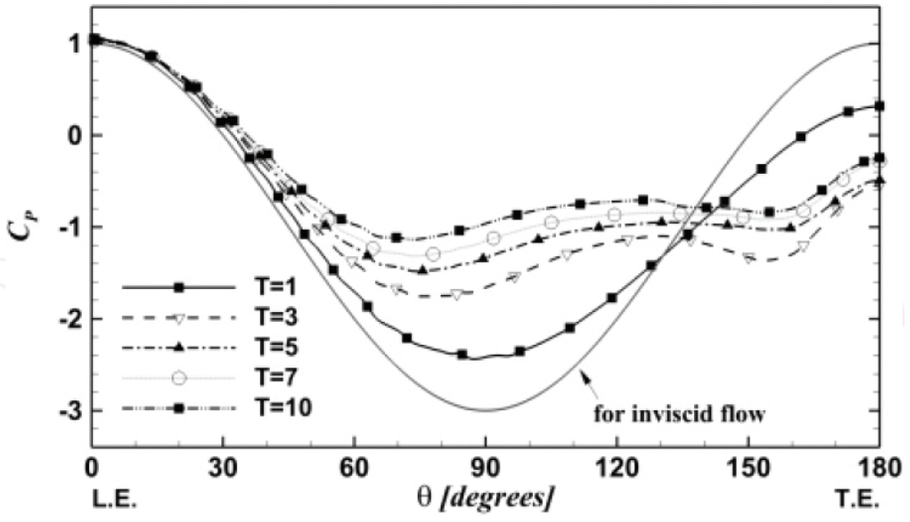


Figure 12. Surface pressure coefficients for a cylinder at a Reynolds number of 550. Note that the figure is reproduced with permission from [28].

The pressure distribution at the cylinder surface is shown in **Figure 12**. At very early stages in the numerical simulation, the surface pressure distribution is quite close to that for an ideal inviscid flow. As flow evolves, the minimum pressure coefficient gradually moves upstream.

http://dx.doi.org/10.5772/65371

The corresponding locations of $C_P = 0$ have the same upstream moving trends and is at the angular position $\theta = 35°$ as experimentally measured by Norberg [57].

5.1.3. Reynolds number of 3000

The simulation parameters are $h = 0.0025$ and $\Delta t = 0.002$. Simulation was carried out until T = 10 and the number of fluid particles ranged from approximately 130,000 to 420,000. The total run time is about 9 h on 16 CPUs (Intel Xeon64 3.3 GHz). **Figure 13** shows numerical results. Compared with a flow at Re = 550, the secondary vortices appear at an earlier time and grow larger. In the case of Re = 3000, the two secondary vortices formed are equivalent in size and in strength. It is the so-called α phenomena [53]. Also, the streamline computed for Re = 3000 compares with the flow visualization result of Loc and Bouard [58]. The present simulation correctly captures the expected physics of the flow at Re = 3000.

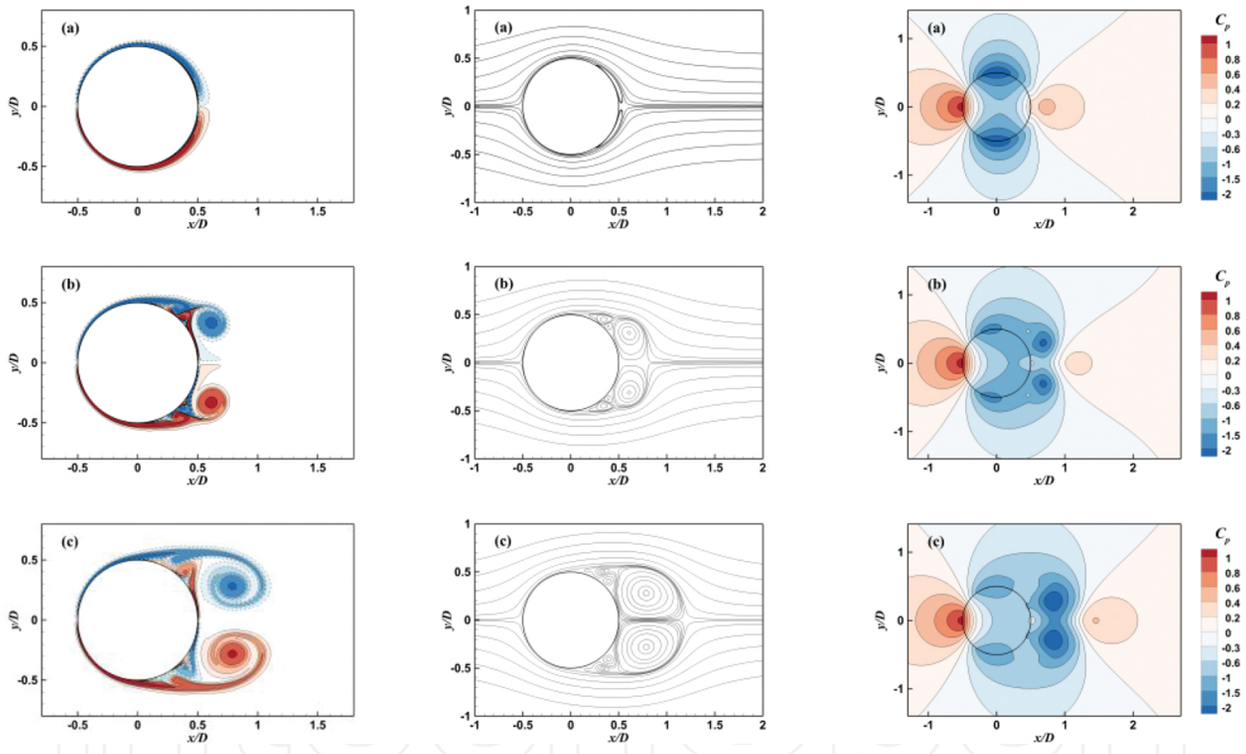


Figure 13. Numerical results for a circular cylinder at Re = 3000 at (a) $T = 1$, (b) $T = 3$, and (c) $T = 5$. Note that the figures are reprinted and reproduced with permission from [28]; vorticity contour (*left*), streamline (*middle*), pressure contour (*right*).

5.1.4. Reynolds number of 9500

The simulation parameters are $h = 0.001$ and $\Delta t = 0.002$. Flows were numerically simulated until T = 4, and the number of fluid particles ranged from approximately 800,000 to 1,200,000. The total run time is about 25 h on 24 CPUs (Intel Xeon64 3.3 GHz). **Figure 14** shows vorticity contours. They are in good agreement with the numerical results in [29].

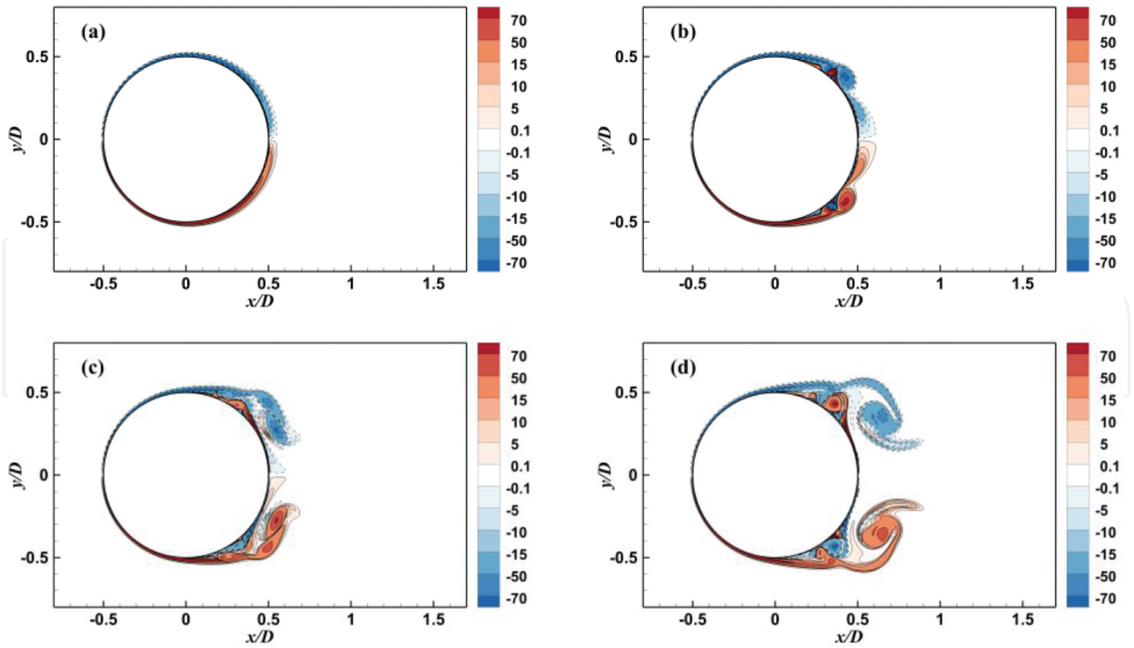


Figure 14. Vorticity contours at a Reynolds number of 9500; (a) T = 1, (b) T = 2, (c) T = 3, and (d) T = 4. Note that the figures are reprinted and reproduced with permission from [28].

As shown in **Figure 15**, the streamline patterns computed for Re = 9500 compare quite well with the flow visualization results in [58]. The α and β phenomena, which are observed experimentally by Bouard and Coutanceau [53], are well captured by the numerical simulation.

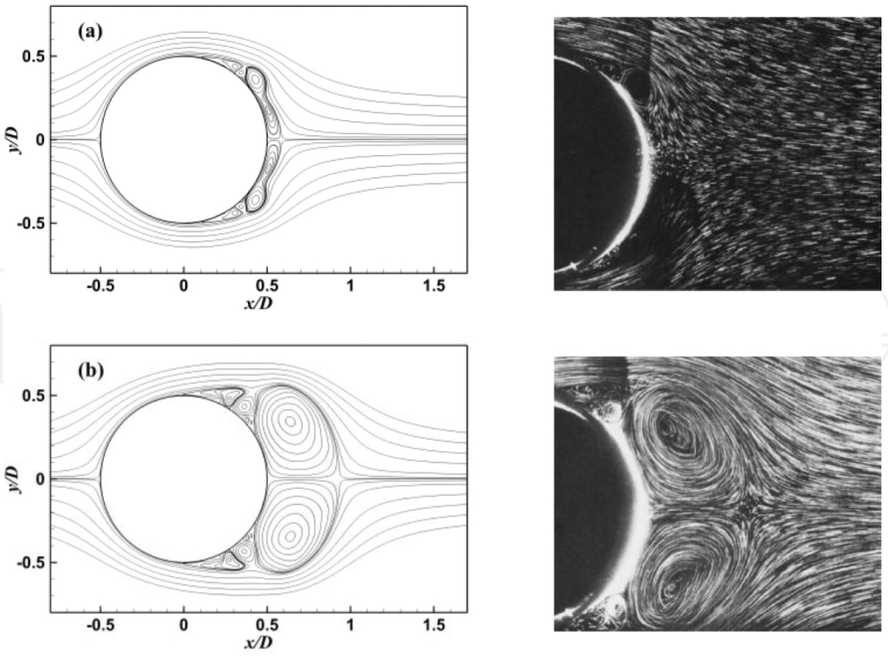


Figure 15. Streamlines at Reynolds number of 9500; (a) T = 2 and (b) T = 4 compared with flow visualization results of Loc and Bouard [58]. Note that the figures are reprinted with permission from [28].

http://dx.doi.org/10.5772/65371

5.2. Flows past an impulsive started sphere

When the pVIC method comes to a 3D implementation, a little attention should been paid to the quantitative validation as mentioned in Sections 2.1 and 3.4. An incompressible viscous flow past a sphere has extensively been studied by many researchers in theoretical, experimental, and numerical ways. It is well-known that the wake behind a sphere depends on the Reynolds number. Especially, the flows at $20 < Re < 210$ are often investigated as a validation case by virtue of their steady and axisymmetric features in wake structure including separation. To demonstrate the feasibility of the pVIC method, we carried out numerical simulations of the impulsively started flow past a sphere at a Reynolds number of 100. **Figure 16** illustrates the configuration of the multiple domains. The sphere is immersed in a Cartesian grid that does not conform to its surface.

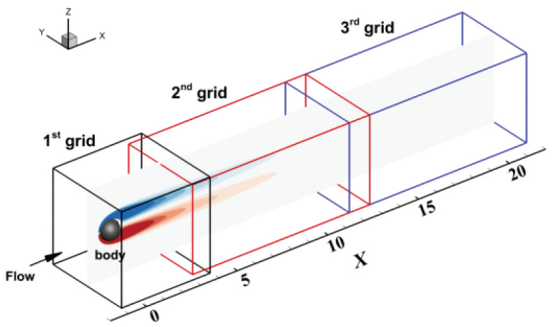


Figure 16. Configuration of computational domains.

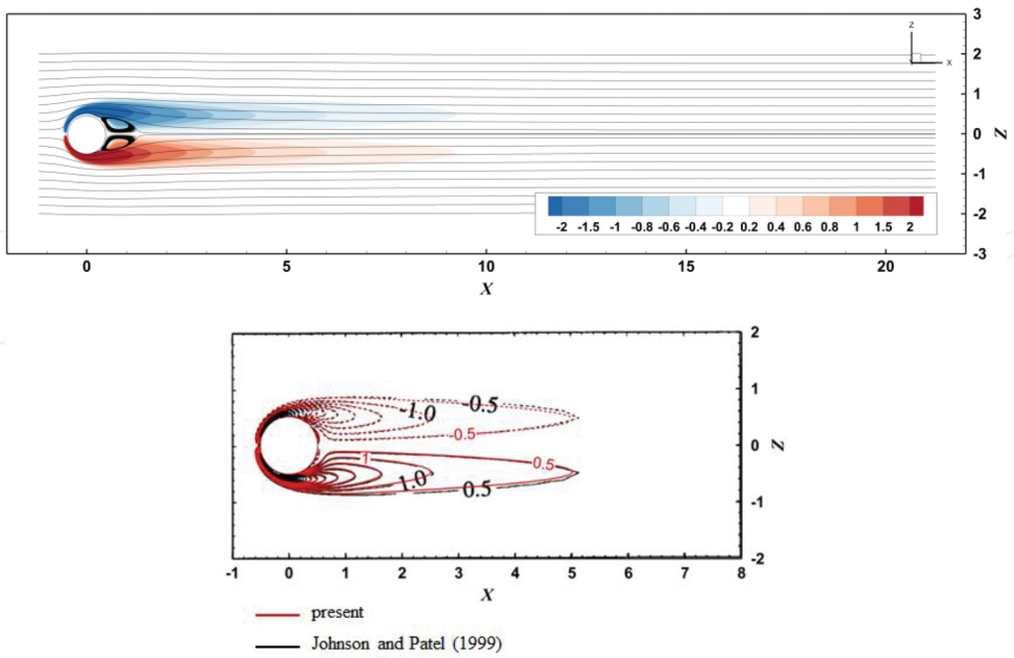


Figure 17. Contour of vorticity magnitude and streamline on *xz*-plane at a steady state.

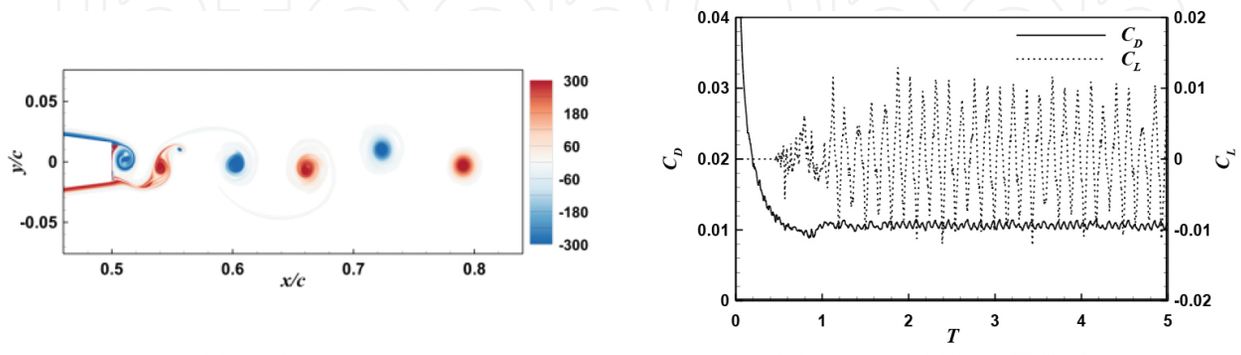
The simulations were carried out until T = 30 to demonstrate the convergence of the present method and test its capability of long-time simulation. Numerical parameters were determined through the stability condition, $\nu\Delta t/h^2 < 1/6$, and the grid convergence was achieved with $h = 0.02$ and $\Delta t = 0.005$. It was concluded that giving $C_\omega = 10^{-5}$ is sufficient in terms of the force calculation. The number of fluid particles ranges from 2 to 5 million during the simulation, and the total computation time is roughly 67 h on 16 CPUs (Intel Xeon64 3.3 GHz). **Figure 17** shows the vorticity contour at the steady state. The vorticity contour is in excellent agreement with the numerical result in [59]. The wake shape and drag coefficient at the steady state are compared to the references in **Table 2**. The numerical results are good agreements with the reference results.

Author(s)	Wake centre (a/D, b/D)	Wake length (L/D)	Separation angle	Drag coefficient
Taneda [60], *exp.*	(0.75, 0.28)	0.89	52.3°	–
Johnson and Patel [59], *cal.*	(0.76, 0.29)	0.89	53.3°	–
Bagchi and Balachandar [61], *cal.*	–	0.87	53.2°	1.09
The presented pVIC method	(0.764, 0.282)	0.877	53.1°	1.09

Table 2. Comparisons of wake shape behind a sphere and drag coefficient with previous results.

5.3. Vortex shedding from a hydrofoil

We selected a National Advisory Committee for Aeronautics (NACA) 0009 cross section with a truncated trailing edge for the numerical simulations. This hydrofoil has the same as the experimental model used in [62–64]. Ten percent of the original chord c_o was removed from the trailing-edge region of the NACA 0009 hydrofoil. The hydrofoil geometry is further detailed in [63]. The maximum thickness, as normalized by the chord length c, is $t_{max}/c = 0.1$ and the thickness at the trailing edge is 0.0322. The numerical parameters $h = 0.0003$ and $\Delta t = 0.00015$ were chosen to simulate flow past the 2D hydrofoil at a Reynolds number of 2×10^6.



(a) vorticity contour (b) drag and lift coefficients

Figure 18. (a) Instantaneous vorticity contour and (b) a time history of drag and lift coefficients at a Reynolds number of 2×10^6.

http://dx.doi.org/10.5772/65371

The instantaneous contour of vorticity magnitude is plotted in **Figure 18a**. Vortices are regularly shed from the symmetric trailing-edge in the form of two trains of opposite-sign but equal-strength vortices. The Strouhal number, $St = fc/U_\infty$ = is about 7.0 from our numerical simulation, and Zobeiri et al. [64] made a similar remark experimentally gave 7.2 (also, refer to [63]). Regular vortex shedding causes periodic loading on the structure. **Figure 18b** shows the evolution of drag and lift coefficients. It has been well-known that the drag force oscillation during vortex shedding is much smaller than the lift force. Oscillations in drag force occur at twice the vortex shedding frequency owing to the fact that two vortices are shed from alternate sides during one full period of wake oscillation. Such features are well captured by our numerical simulation as shown in **Figure 18b**. The mean drag coefficient C_D is 0.0107 and the root-mean-square lift coefficient is 0.0056.

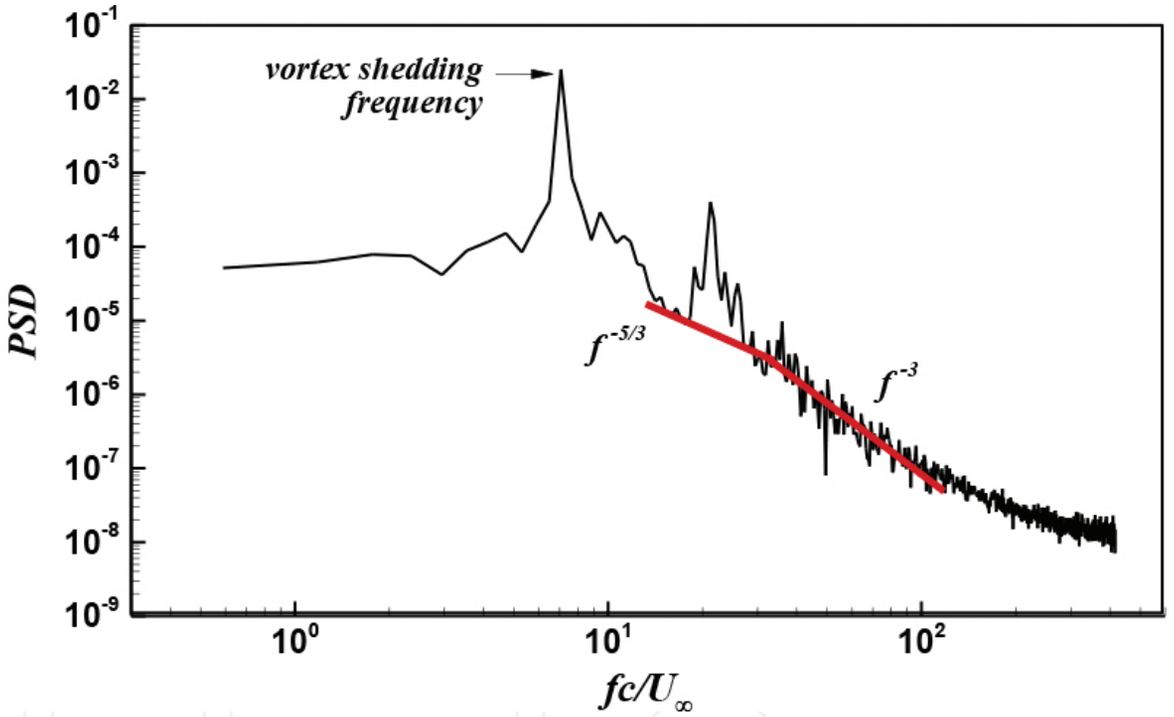


Figure 19. Temporal power spectral density (PSD) of vertical velocity fluctuations. Note that this figure is reprinted with permission from [32].

Figure 19 shows the temporal power spectrum of vertical velocity fluctuations measured in the near wake. The maximum peak is identical to the vortex shedding frequency, and the second peak area is found in the third harmonic of the shedding frequency. For $U_\infty < 5$, spectral levels are almost flat as f decreases. This trend is similar to experimental data of Bourgoyne et al. [65]. In their measurements, an important characteristic of the spectra is the presence of a clear region with a −5/3 slope; spectral power-law for high Reynolds number turbulent fluctuations. From our numerical simulation, however, the spectral density of the velocity fluctuations shows a decay of the form. A similar remark was made by Singh and Mittal [66]. This is due that the vortex stretching mechanism is absent in two-dimensional flows. More detail on fluid turbulence confined two spatial dimensions can be found in Boffetta and Ecke [67].

Figure 20 shows vortex shedding from a hydrofoil with different bevel angles in degrees. At β = 0°, the vortex shedding is regular and periodic. As the bevel angle β increases, vortex shedding becomes increasingly disorganized. The periodicity of both forces is almost lost at β = 60°.

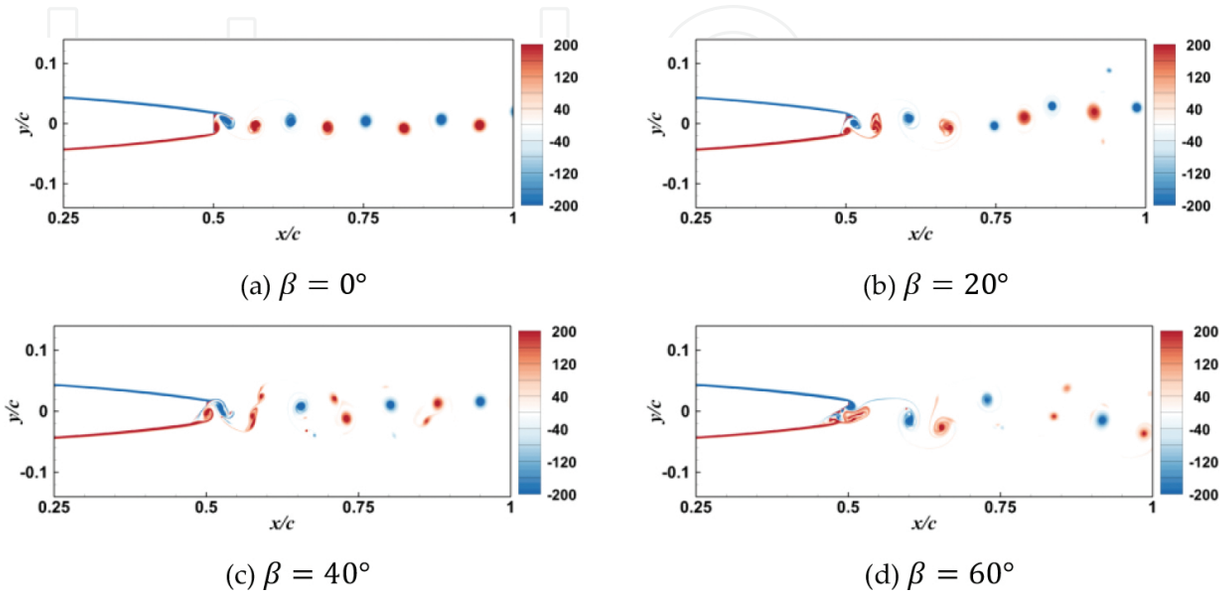


Figure 20. Vortex shedding from a hydrofoil with different bevel angles. Note that the figures are reproduced with permission from [37]. (a) β = 0°; (b) β = 20°; (c) β = 40°; (d) β = 60°.

Also, we conducted numerical simulations to investigate vortex shedding with respect to the sinusoidal motions of the free stream flow. The angle of attack varied periodically with time t; $\alpha = \alpha_o sin(2\pi f_\infty t)$ where f_∞ was the frequency of the free-stream flow oscillation and its oscillation amplitude was restricted at α_o = 2 in degrees. The magnitude of the free-stream velocity was kept constant. The tested model is the NACA 0009 hydrofoil with β = 60°. Interestingly, at f_∞ = 20 vortices are regularly shed from the trailing edge, as shown in **Figure 21**. In the power spectral density (PSD) functions, drag oscillation induced by vortex shedding is clearly observed. This means that a particular oscillation frequency in the free stream velocity can cause regular and periodic vortex shedding.

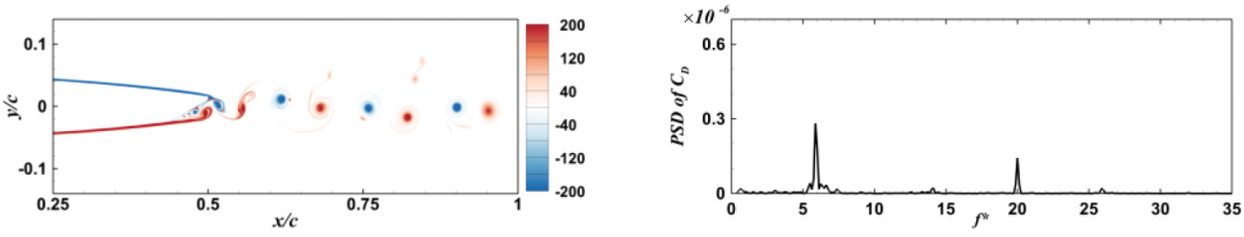


Figure 21. Vortex shedding at f_∞ = 20. Note that the figures reproduced and reprinted with permission from [37].

http://dx.doi.org/10.5772/65371

6. Concluding remarks

The pVIC method was applied to the simulation of an incompressible viscous flow past a solid body in 2D and 3D. The obtained results showed good agreement with experimental and numerical data from the published literature. This demonstrates the feasibility of the pVIC method. Obviously, the pVIC method offers a simple, efficient, and effective way to simulate unsteady flows. The fact that vortex-dominated flows are well-characterized by vortices makes the pVIC method more attractive. However, some hard problems still remain; for example, rigorous boundary conditions at solid walls and an efficient approximation of pressure boundary conditions at far field.

Author details

Seung-Jae Lee

Address all correspondence to: hul94@snu.ac.kr

Research Institute of Marine Systems Engineering, Seoul National University, South Korea

References

[1] B. R. Morton. The generation and decay of vorticity. Geophysical and Astrophysical Fluid Dynamics. 1984;28:277–308. doi:10.1080/03091928408230368

[2] C. G. Speziable. On the advantages of the vorticity-velocity formulation of the equations of fluid dynamics. Journal of Computational Physics. 1987;73(2):476–480. doi: 10.1016/0021-9991(87)90149-5

[3] P. M. Gresho. Incompressible fluid dynamics: some fundamental formulation issues. Annual Review of Fluid Mechanics. 1991;23:413–453. doi:10.1146/annurev.fl. 23.010191.002213

[4] A. Leonard. Vortex methods for flow simulation. Journal of Computational Physics. 1980;37(3):289–335. doi:10.1016/0021-9991(80)90040-6

[5] A. Leonard. Computing three-dimensional incompressible flows with vortex elements. Annual Review of Fluid Mechanics. 1985;17:523–559. doi:10.1146/annurev.fl.17.010185. 002515

[6] T. Sarpkaya. Computational methods with vortices-the 1988 freeman scholar lecture. Journal of Fluids Engineering. 1989;111(1):5–52. doi:10.1115/1.3243601

[7] K. Kamemoto. On attractive features of the vortex methods. In: M. Hafez and K. Oshima, editors. Computational Fluid Dynamics Review 1995. Wiley; 1995. p. 334–353.

[8] L. A. Barba, A. Leonard and C. B. Allen. Advances in viscous vortex methods—meshless spatial adaption based on radial basis function interpolation. International Journal for Numerical Methods in Fluids. 2005;47:387–421. doi:10.1002/fld.811

[9] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. Journal of Computational Physics. 1987;73(2):325–348. doi:10.1016/0021-9991(87)90140-9

[10] VJ. Dongarra and F. Sullivan. Guest editors introduction to the top 10 algorithms. Computing in Science and Engineering. 2000;2(1):22–23. doi:10.1109/MCISE.2000.814652

[11] R. Beatson and L. Greengard. A short course on fast multipole methods. In: Wavelets, Multilevel Methods and Elliptic PDEs. Oxford University Press; 1997. p. 1–37. doi:10.1.1.129.7826

[12] J. P. Christiansen. Numerical simulation of hydrodynamics by the method of point vortices. Journal of Computational Physics. 1973;13(3):363–379. doi:10.1016/0021-9991(73)90042-9

[13] G.-H. Cottet and P. Poncet. Advances in direct numerical simulations of 3D wall-bounded flows by Vortex-in-Cell methods. Journal of Computational Physics. 2004;193(1):136–158. doi:10.1016/j.jcp.2003.08.025

[14] T. K. Sheel. Development of a fast Vortex method for fluid flow simulation using special-purpose computers [thesis]. Japan: Keio University; 2008. 169

[15] HaK. Kuwahara and H. Takami. Numerical studies of two-dimensional vortex motion by a system of point vortices. Journal of the Physical Society of Japan. 1973;34(1):247–253. doi:10.1143/JPSJ.34.247

[16] A. J. Chorin. Numerical study of slightly viscous flow. Journal of Fluid Mechanics. 1973;57(4):785–796. doi:10.1017/S0022112073002016

[17] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. Part 1: the case of an isotropic viscosity. Mathematics of Computation. 1989;53(188):485–507. doi:10.2307/2008716

[18] D. Fishelov. A new vortex scheme for viscous flows. Journal of Computational Physics. 1990;86(1):211–224. doi:10.1016/0021-9991(90)90098-L

[19] S. Shankar and L. van Dommelen. A new diffusion procedure for vortex methods. Journal of Computational Physics. 1996;127(1):88–109. doi:10.1006/jcph.1996.0160

[20] J. M. R. Graham. Computation of viscous separated flow using a particle method. In: K. W. Morton and M. J. Baines, editors. Numerical Methods for Fluid Dynamics III. Oxford: Oxford University Press; 1988. p. 310–317.

[21] J. T. Beale and A. Majda. Vortex methods. II: higher order accuracy in two and three dimensions. Mathematics of Computation. 1982;39(159):29–52. doi:10.2307/2007618

[22] C. H. Liu and D. J. Doorly. Vortex particle-in-cell method for three-dimensional viscous unbounded flow computations. International Journal for Numerical Methods in Fluids. 2000;32:29–50. doi:10.1002/(SICI)1097-0363(20000115)32:1%3C23::AID-FLD922%3E3.0.CO;2-O

[23] R. Cocle, G. Winckelmans, and G. Daeninck. Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations. Journal of Computational Physics. 2008;227(21):9091–9120. doi:10.1016/j.jcp.2007.10.010

[24] P. Angot, C.-H. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in viscous flows. Numerische Mathematik. 1999;81(4):497–520. doi:10.1007/s002110050401

[25] LiM. El Ossmani and P. Poncet. Efficiency of multiscale hybrid grid-particle vortex methods. Multiscale Modeling and Simulation. 2010;8(5):1671–1690. doi:10.1137/090765006

[26] R. Chatelin and P. Poncet. Hybrid grid–particle methods and penalization: a Sherman-Morrison-Woodbury approach to compute 3D viscous flows using FFT. Journal of Computational Physics. 2014;269(15):314–328. doi:10.1016/j.jcp.2014.03.023

[27] H. Beaugendre, F. Morency, F. Gallizio, and S. Laurens. Computation of ice shedding trajectories using Cartesian grids, penalization, and level sets. Modelling and Simulation in Engineering. 2011:274947. doi:10.1155/2011/274947

[28] S.-J. Lee, J.-H. Lee, and J.-C. Suh. Computation of pressure fields around a two-dimensional circular cylinder using the vortex-in-cell and penalization methods. Modelling and Simulation in Engineering. 2014:708372. doi:10.1155/2014/708372

[29] G.-H. Cottet, F. Gallizio, A. Magni, and I. Mortazavi. A vortex immersed boundary method for bluff body flows. In: Proceedings of 3rd Joint US-European ASME Fluids Engineering Summer Meeting; August 1-5; Montreal, Canada. 2010.

[30] C. Mimeau, F. Gallizio, G.-H. Cottet, and I. Mortazavi. Votex penalization method for bluff body flows. International Journal for Numerical Methods in Fluids. 2015;79:55–83. doi:10.1002/fld.4038

[31] M. Gazzola, C. Mimeau, A. A. Tchieu, and P. Koumoutsakos. Flow mediated interactions between two cylinders at finite Re numbers. Physics of Fluids. 2012;24:043103. doi:10.1063/1.4704195

[32] S.-J. Lee, J.-H. Lee, and J.-C. Suh. Further validation of the hybrid particle-mesh method for vortex shedding flow simulations. International Journal of Naval Architecture and Ocean Engineering. 2015;7:1034–1043. doi:10.1515/ijnaoe-2015-0072

[33] J. T. Rasmussen, G.-H. Cottet, and J. H. Walther. A multiresolution remeshed vortex-in-cell algorithm using patches. Journal of Computational Physics. 2011;230(19):6742–6755. doi:10.1016/j.jcp.2011.05.006

[34] J. J. Monaghan. Extrapolating B splines for interpolation. Journal of Computational Physics. 1985;60(2):253–262. doi:10.1016/0021-9991(85)90006-3

[35] W. Iwakami, Y. Yatagai, N. Hatakeyama, and Y. Hattori. A new approach for error reduction in the volume penalization method. Communications in Computational Physics. 2014;16(5):1181–1200. doi:10.4208/cicp.220513.070514a

[36] G. S. Winckelmans and A. Leonard. Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows. Journal of Computational Physics. 1993;109(2):247–273. doi:10.1006/jcph.1993.1216

[37] S.-J. Lee, J.-H. Lee, and J.-C. Suh. Numerical investigation on vortex shedding from a hydrofoil with a beveled trailing edge. Modelling and Simulation in Engineering. 2015:565417. doi:10.1155/2015/565417

[38] M. L. Ould-Salihi, G.-H. Cottet, and M. El Hamraoui. Bending finite-difference and vortex methods for incompressible flow computations. SIAM Journal on Scientific Computing. 2000;22(5):1655–1674. doi:10.1137/S1064827599350769

[39] M. Coquerelle and G.-H. Cottet. A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies. Journal of Computational Physics. 2008;227(21):9121–9137. doi:10.1016/j.jcp.2008.03.041

[40] F. Morency, H. Beaugendre, and F. Gallizio. Aerodynamic for evaluation for ice shedding phenomenon using vortex in cell scheme, penalization and level set approaches. International Journal of Computational Fluid Dynamics. 2012;26(9–10):435–450. doi:10.1080/10618562.2012.739683

[41] A. Kosior and H. Kudela. Parallel computations on GPU in 3D using the vortex particle method. Computers and Fluids. 2013;80(10):423–428. doi:10.1016/j.compfluid.2012.01.014

[42] T. Lonfils and G. Winckelmans. Development of an immersed boundary method using boundary elements within a vortex-in-cell/parallel fast multipole method. In: J. C. F. Pereira and A. Sequeira, editors. European Conference on Computational Fluid Dynamics, ECCOMAS CFD; June 14–17; Lisbon, Portugal. 2010.

[43] P. Ploumhans and G. S. Winckelmans. Vortex methods for high-resolution simulations of viscous flow pat bluff bodies of general geometry. Journal of Computational Physics. 2000;165(2):354–406. doi:10.1006/jcph.2000.6614

[44] D. Rossinelli, M. Bergdorf, G.-H. Cottet, and P. Koumoutsakos. GPU accelerated simulations of bluff body flows using vortex particle methods. Journal of Computational Physics. 2010;229(9):3316–3333. doi:10.1016/j.jcp.2010.01.004

[45] U. Trottenberg, C. Oosterlee, and A. Schuller. Multigrid. Oxford, UK: Elsevier; 2001. P. 631.

[46] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. Proceedings of the IEEE. 2005;93(2):216–231. doi:10.1109/JPROC.2004.840301

[47] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical Recipes in C++. Cambridge, UK: Cambridge University Press; 2002. p. 1002.

[48] M. M. Hejlesen, P. Koumoutsakos, A. Leonard, and J. H. Walther. Iterative Brinkman penalization for remeshed vortex methods. Journal of Computational Physics. 2015;280(1):547–562. doi:10.1016/j.jcp.2014.09.029

[49] S.-J. Lee. Numerical simulation of single-bubble dynamics with two-way coupling uing the Lagrangian vortex method [thesis]. Seoul, South Korea: Seoul National University; 2005. 193 p.

[50] S.-J. Lee and J.-C. Suh. Fast computation of domain boundary conditions using a cubic Spline for the penalized VIC method. International Journal of Computational Methods. Forthcoming.

[51] B. Fornberg. A numerical study of steady viscous flow past a circular cylinder. Journal of Fluid Mechanics. 1980;98(4):819–855. doi:10.1017/S0022112080000419

[52] A. S. Grove, F. H. Shair, and E. E. Petersen. An experimental investigation of the steady separated flow past a circular cylinder. Journal of Fluid Mechanics. 1964;19(1):60–80. doi:10.1017/S0022112064000544

[53] R. Bouard and M. Coutanceau. The early stage of development of the wake behind an impulsively started cylinder for $40 < Re < 10^4$. Journal of Fluid Mechanics. 1980;101(3): 583–607. doi:10.1017/S0022112080001814

[54] M. Bar-Lev and H. T. Yang. Initial flow field over an impulsively started circular cylinder. Journal of Fluid Mechanics. 1975;72(4):625–647. doi:10.1017/S0022112075003199

[55] P. Koumoutsakos and A. Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. Journal of Fluid Mechanics. 1995;296:1–38. doi:10.1017/S0022112095002059

[56] H. Kudela and T. Kozlowski. Vortex in cell method for exterior problems. Journal of Theoretical and Appiled Mechanics. 2009;47(4):779–796.

[57] C. Norberg. Pressure distribution around a circular cylinder in cross-flow. In: The Symposium on Bluff Body Wakes and Vortex-Induced Vibration (BBVIC); Queensland, Australia. 2002.

[58] T. P. Loc and R. Bouard. Numerical solution of the early stage of the unsteady viscous flow around a circular cylinder: a comparison with experimental visualization and

measurements. Journal of Fluid Dynamics. 1985;160:93–117. doi:10.1017/S0022112085003408

[59] T. A. Johnson and V. C. Patel. Flow past a sphere up to a Reynolds number of 300. Journal of Fluid Mechanics. 1999;378:19–70. doi:10.1017/S0022112098003206

[60] S. Taneda. Experimental investigation of the wake behind a sphere at low Reynolds numbers. Journal of the Physical Society of Japan. 1956;11:1104–1108. doi:10.1143/JPSJ.11.1104

[61] P. Bagchi and S. Balachandar. Shear versus vortex-induced lift force on a rigid sphere at moderate Re. Journal of Fluid Mechanics. 2002;473:379–388. doi:10.1017/S0022112002002628

[62] P. Ausoni, M. Farhat, X. Escaler, E. Egusquiza, and F. Avellan. Cavitation influence on Karman vortex shedding and induced hydrofoil vibrations. Journal of Fluid Engineering. 2007;129(8):966–973. doi:10.1115/1.2746907

[63] Ausoni. Turbulent vortex shedding from a blunt trailing edge hydrofoil [thesis]. École polytechnique fédérale de Lausanne, EPFL; 2009. 169 p. Available from: https://infoscience.epfl.ch/record/138935

[64] A. Zobeiri, P. Ausoni, F. Avellan, and M. Farhat. How oblique trailing edge of a hydrofoil reduces the vortex-induced vibration. Journal of Fluids and Structures. 2012;32:78–89. doi:10.1016/j.jfluidstructs.2011.12.003

[65] D. A. Bourgoyne, S. L. Ceccio, and D. R. Dowling. Vortex shedding from a hydrofoil at high Reynolds number. Journal of Fluid Mechanics. 2005;531:293–324. doi:10.1017/S0022112005004076

[66] S. P. Singh and S. Mittal. Flow past a cylinder: shear layer instability and drag crisis. International Journal for Numerical Methods in Fluids. 2005;47:75–98. doi:10.1002/fld.807

[67] G. Boffetta and R. E. Ecke. Two-dimensional turbulence. Annual Review of Fluid Mechanics. 2012;44:427–451. doi:10.1146/annurev-fluid-120710-101240