Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 204 (2005) 157–192
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains

Mark N. Linnick *, Hermann F. Fasel [1]

Department of Aerospace and Mechanical Engineering, College of Engineering, University of Arizona, 1130 North Mountain, P.O. Box 210119, Tucson, AZ 85721-0119, USA

Received 4 July 2003; received in revised form 17 September 2004; accepted 26 September 2004
Available online 11 November 2004

Abstract

Immersed boundary methods and immersed interface methods are becoming increasingly popular for the computation of unsteady flows around complex geometries using a Cartesian grid. While good results, both qualitative and quantitative, have been obtained, most of the methods rely on low-order corrections to account for the immersed boundary. The objective of the present work is to present, as an alternative, a high-order modified immersed interface method for the 2D, unsteady, incompressible Navier–Stokes equations in stream function–vorticity formulation. The method employs an explicit fourth-order Runge–Kutta time integration scheme, fourth-order compact finite-differences for computation of spatial derivatives, and a nine-point, fourth-order compact discretization of the Poisson equation for computation of the stream function. Corrections to the finite difference schemes are used to maintain high formal accuracy at the immersed boundary, as confirmed by analytical tests. To validate the method in its application to incompressible flows, several physically relevant test cases are computed, including uniform flow past a circular cylinder and Tollmien–Schlichting waves in a boundary layer.

Keywords: Finite difference methods; Incompressible viscous fluids; Immersed boundary; Immersed interface; Cartesian grid method

* Corresponding author. Present address: Institut für Aerodynamik und Gasdynamik, Universität Stuttgart, Pfaffenwaldring 21, 70550 Stuttgart, Germany. Tel.: +49 711 685 3422/+1 520 621 4424; fax: +49 711 685 3438/+1 520 621 8191.
E-mail addresses: mark.linnick@iag.uni-stuttgart.de, linnick@iag.uni-stuttgart.de (M.N. Linnick), faselh@email.arizona.edu (H.F. Fasel).
[1] Tel.: +1 520 621 2771; fax: +1 520 621 8191.

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2004.09.017

1. Introduction

Immersed boundary methods (IBMs), and, more recently, immersed interface methods (IIMs) have been introduced as an alternative to traditional approaches for numerically solving initial/boundary-value problems on domains with complex geometric boundaries. One of the most significant differences between the IBM and the IIM is the use of a discrete delta function in the former. In both methods, however, the equations to be solved are discretized on a fixed Cartesian grid. As a result, the domain boundaries do not always conform to the computational domain boundaries. This gives rise to boundaries immersed inside the computational domain. With regard to Fig. 1, for example, one would typically like to solve a PDE defined on the open region Ω^+ with boundary conditions on $\partial\Omega_o$, the outer boundary which conforms to the computational boundary, and $\partial\Omega_i$, the immersed boundary which does not. The solution in the region Ω^- may or may not be of interest. In the present investigation, the solution in Ω^- is set identically equal to zero. In either case, the immersed boundary $\partial\Omega_i$ represents a singularity if one considers that a particular set of governing partial differential equations apply throughout the entire domain enclosed by $\partial\Omega_o$ (as in the analytical study of Sirovich [30], for example); field variables and/or their derivatives will be discontinuous across the immersed boundary.

The IBM and IIM determine a solution at every grid-point within the domain enclosed by $\partial\Omega_o$, both inside and outside the immersed boundary. The equations to be numerically solved are discretized on a rectangular Cartesian grid which is allowed to pass through $\partial\Omega_i$, as shown in Fig. 1. Several methods for handling the singularity at the immersed boundary have been proposed in the past.

In applications of the immersed boundary method to incompressible flow problems, this singularity is usually represented by a forcing term $\mathbf{F}$ in the Navier–Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = \mathbf{F}(\mathbf{x},t) - \nabla p + \frac{1}{Re}\nabla^2\mathbf{u}, \tag{1}$$

where $\mathbf{F}$ has support only on the immersed boundary and is determined from the surface integral

$$\mathbf{F}(\mathbf{x},t) = \int_{\partial\Omega_i(t)} \mathbf{f}(r,s,t)\delta(\mathbf{x}-\mathbf{X}(r,s,t))\,\mathrm{d}S. \tag{2}$$

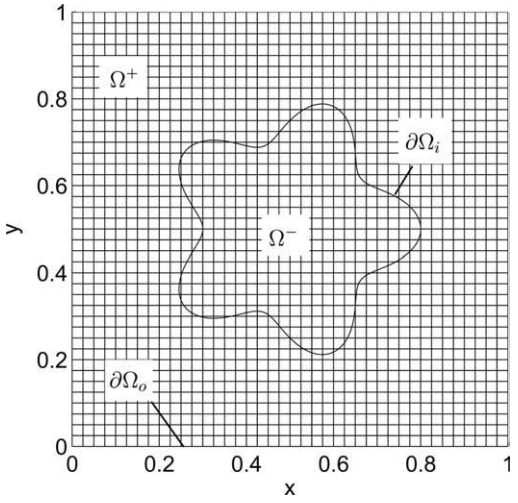

Fig. 1. An irregular immersed boundary $\partial\Omega_i$ and computational Cartesian grid.

The term $\mathbf{X}(r,s,t)$ is a parametric representation of the immersed boundary surface S, which is considered to be explicitly given, and $\delta(\mathbf{x})$ is the three-dimensional Dirac delta function. The term $\mathbf{f}(r,s,t)$ is the source strength, the value of which must be determined by imposing some additional constraint, usually

$$\mathbf{u}(\mathbf{x},t)|_{\mathbf{x}=\mathbf{X}} = \bar{\mathbf{u}}(\mathbf{X},t), \tag{3}$$

where $\bar{\mathbf{u}}(\mathbf{X},t)$ is the known velocity of the immersed boundary.

The earliest IBM approach is probably that of Viecelli [34,35], who proposed a numerical method for computing inviscid, incompressible flow with arbitrarily shaped curved boundaries using a Cartesian grid. In his calculations, the pressure at the immersed boundary is iteratively modified until fluid particles move along the tangent to the boundary. The original immersed boundary method, however, is usually attributed to the work of Peskin (early work includes [22–25]) who used the method to investigate flow patterns through heart valves and in a beating heart. In Peskin's approach, the geometry of the immersed boundary is determined as part of the overall solution, being dependent upon material properties and the surrounding viscous fluid flow. More recent investigations and applications of Peskin's immersed boundary method include [3,12,28,9].

In the numerical scheme, Eq. (2) is usually discretized as a linear sum of point forces

$$\mathbf{F}(\mathbf{x},t) = \sum_k \mathbf{f}_k \ \delta_h(\mathbf{x}-\mathbf{X}_k)\Delta S_k, \tag{4}$$

where $\mathbf{f}_k$ is the source strength, $\delta_h(\mathbf{x})$ is a numerical representation for the Dirac delta function, k denotes a finite distribution of points located on the surface of the immersed boundary, and ΔS_k is an element of surface area surrounding point k. A number of approaches have been suggested for determining the term $\mathbf{f}_k$. For example, with $\bar{\mathbf{u}}(\mathbf{X},t) \equiv \mathbf{0}$, Goldstein et al. [12] allow the force to adapt itself to the local flow field through the use of a feedback loop

$$\mathbf{f}_k = \alpha \int_0^t \mathbf{u}(\mathbf{X}_k,\tilde{t})\,\mathrm{d}\tilde{t} + \beta\mathbf{u}(\mathbf{X}_k,t) \tag{5}$$

where α and β are negative constants which determine the amount of control. A more direct method of determining $\mathbf{f}_k$ is discussed by Fadlun et al. [9].

The effect of the discretization represented by Eq. (4) on the accuracy of the immersed boundary scheme has not been discussed in the literature. With regard to the numerical approximation of the delta function $\delta(x)$, Waldén [36] proves that full convergence order of a numerical scheme involving singular source terms can be achieved away from the singularities, whereas poor convergence will be obtained in the vicinity of these. To illustrate the problem, and, simultaneously, to motivate the development of the IIM discussed in this paper, the use of a simple representation for $\delta(x)$ is investigated, the Gaussian function

$$\delta_h(x) = \frac{1}{\sqrt{\pi}\sigma}\mathrm{e}^{-\left(\frac{x}{\sigma}\right)^2}, \tag{6}$$

where σ is a constant determining the effective support of the function. This and similar representations have been used in several immersed boundary methods [3,12,28]. Results using this representation to numerically compute the solution to the ODE

$$u'' - 4u = 2\delta'(x-1/3) \tag{7}$$

are shown in Fig. 2. The effect of the selected singular source representation is to smear out what should be a sharp discontinuity in the solution in the vicinity of the singularity located at $x = 1/3$. As σ is reduced, the region that is affected becomes correspondingly smaller, although there is a limit to how small σ can be made.

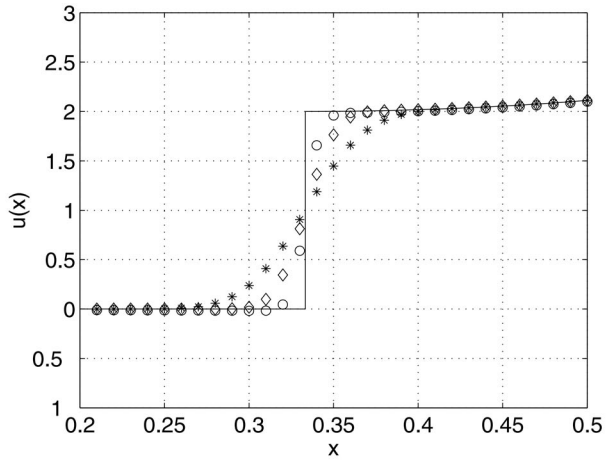

Fig. 2. Analytical (—) and numerical (symbols) solution of Eq. (7). Eq. (6) with $\sigma/h = 1$ (○), $\sigma/h = 2$ (◇), and $\sigma/h = 4$ (∗) was used as a discrete representation of the singular source term.

This smearing effect can be seen in an implementation [19,32] of the Goldstein immersed boundary method [12] used to compute an incompressible, zero-pressure gradient flat-plate boundary layer. The setup is shown in Fig. 3. The immersed flat plate runs from the inflow to the outflow, and is parallel to the computational, body-fitted wall. Vorticity boundary layer distributions which were obtained with a body-fitted code and also with the (Goldstein) immersed boundary code are shown in Fig. 4. As expected, a sharp interface at the immersed wall cannot be obtained.

Results obtained from the immersed boundary method can be improved with increased resolution in the vicinity of the immersed boundary. This increased resolution is usually only required locally, and is most efficient when coupled with adaptive mesh refinement (AMR). An investigation of an immersed boundary method coupled with AMR was carried out by Roma [27].

A significant advance in the immersed boundary method was made by LeVeque and Li [15] who introduced the idea of the immersed interface method mentioned earlier, and, most recently, by Wiegmann and Bube [37] who proposed the explicit-jump immersed interface method. These researchers made the simple, but important, observation that standard finite difference techniques fail when applied to non-smooth functions because the underlying Taylor expansions upon which they are based are invalid. Recent articles which make use of IIM elements for simulating incompressible flows include Li and Lai [17], Calhoun [5] and Li and Wang [16]. The key idea of the IIM is that the finite differences schemes at the interface

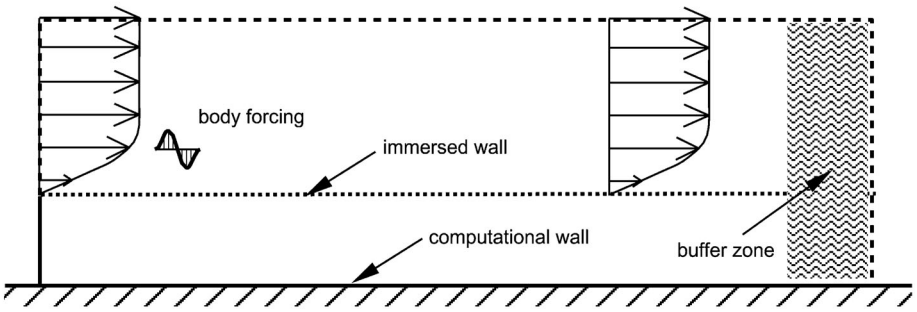

Fig. 3. A flat plate immersed in a rectangular computational domain.

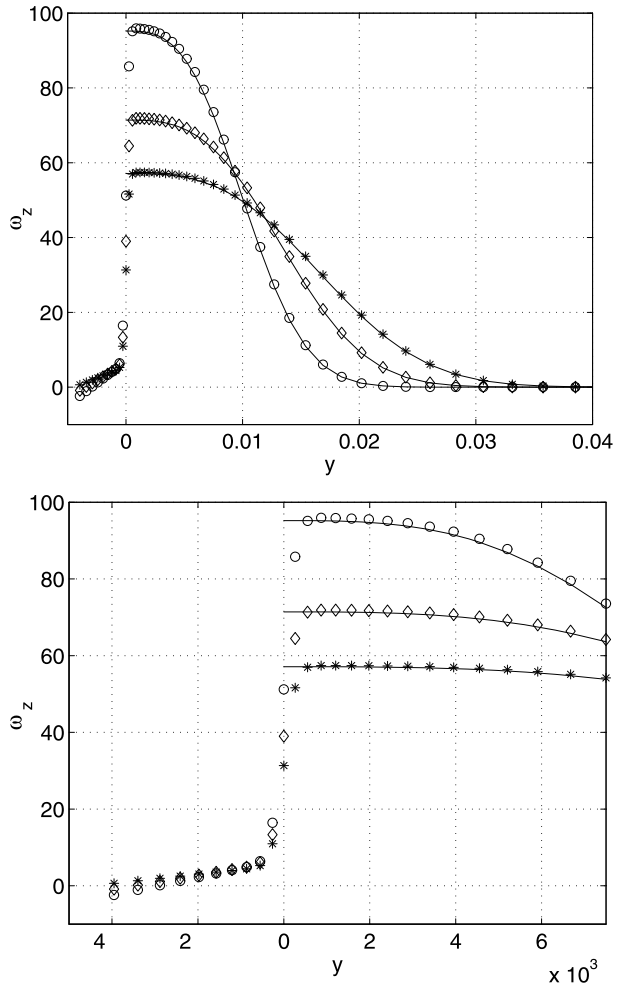

Fig. 4. Comparison of ω-vorticity profiles at several downstream locations: $x = 1.213$ ($R_{\delta_1} = 600$) (○), $x = 2.158$ ($R_{\delta_1} = 800$) (◇), and $x = 3.375$ ($R_{\delta_1} = 1000$) (∗). Symbols indicate values computed using an immersed wall (volume forcing) at $y = 0$, and solid lines using the body-fitted code. Lower figure is a zoom-in of the upper figure near $y = 0$.

of the immersed boundary must be corrected in order to maintain the formal accuracy of the underlying numerical scheme. This topic will be discussed below, following a brief introduction of the governing equations to which the immersed interface method will be applied.

2. Governing equations

The Navier–Stokes equations are solved in the stream function–vorticity formulation [26] in a Cartesian coordinate system (x,y). The primitive variables $\mathbf{u} = (u,v)$ and p are replaced by two scalar variables, the vorticity ω and the stream function ψ. The vorticity ω is defined here as

$$\omega = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \tag{8}$$

and the stream function ψ such that

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}. \tag{9}$$

The transport equation for the vorticity ω is

$$\frac{\partial \omega}{\partial t} + \frac{\partial (u\omega)}{\partial x} + \frac{\partial (v\omega)}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right). \tag{10}$$

From the definition of the vorticity and the stream function, the following Poisson equation for the stream function is obtained

$$\nabla^2 \psi = \omega. \tag{11}$$

Let $\mathbf{n}$ denote the outward-pointing unit vector normal to the body S, $\boldsymbol{\tau}$ the tangential unit vector, and s the curvilinear coordinate along the body (counter-clockwise orientation). Given the velocity of the boundary S as $\bar{\mathbf{u}}(s,t)$, the corresponding boundary conditions on ψ are

$$\left.\frac{\partial \psi}{\partial s}\right|_S = \mathbf{n} \cdot \bar{\mathbf{u}}(s,t), \tag{12}$$

$$\left.\frac{\partial \psi}{\partial n}\right|_S = -\boldsymbol{\tau} \cdot \bar{\mathbf{u}}(s,t). \tag{13}$$

Eq. (12) can be integrated over S to yield an equivalent Dirichlet boundary condition, so that one takes instead

$$\psi|_S = a(s,t), \tag{14}$$

$$\left.\frac{\partial \psi}{\partial n}\right|_S = -\boldsymbol{\tau} \cdot \bar{\mathbf{u}}(s,t). \tag{15}$$

In multiply connected domains, the constant of integration for Eq. (14) is determined by enforcing the condition that the pressure is single valued

$$\oint \nabla p \cdot \, \mathrm{d}\mathbf{s} = 0. \tag{16}$$

There are more boundary conditions for ψ than required by Eq. (11), so boundary conditions for ω must be specified such that both constraints on ψ can be satisfied. In the present case, this is accomplished by using the definition, Eq. (8), to compute the wall vorticity.

3. Numerical analysis

3.1. Taylor series of functions with jump-singularities

Fig. 5 shows a function $f(x)$ with a discontinuity at the point $x = x_\alpha$. One would like to write a Taylor series at point x_i to evaluate $f(x)$ at point x_{i+1}. Assume that $f(x)$ is analytic everywhere in the domain $D = \{x | x_{i-1} \leqslant x \leqslant x_{i+1}\}$ except at the point x_α (and only at this point, in all of what follows) where it has a jump discontinuity in the function value itself and/or higher derivatives. If $x_i < x_\alpha$, the standard Taylor series cannot proceed through x_α to correctly predict $f(x_{i+1})$ unless a correction term J_α is added:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + f''(x_i)\frac{h^2}{2!} + \cdots + J_\alpha, \tag{17}$$

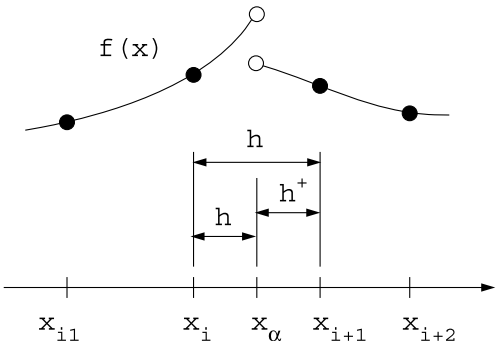

Fig. 5. A function $f(x)$ with discontinuity at $x = x_\alpha$.

where J_α is

$$J_\alpha = [f]_\alpha + [f']_\alpha h^+ + \frac{1}{2!}[f'']_\alpha (h^+)^2 + \cdots, \tag{18}$$

$h = x_{i+1} - x_i$, and $h^+ = x_{i+1} - x_\alpha$. The term $[\phi]_\alpha$ represents the jump in the value of ϕ at $x = x_\alpha$, that is

$$[\phi]_\alpha = \lim_{x \to x_\alpha^+} \phi(x) - \lim_{x \to x_\alpha^-} \phi(x) \tag{19}$$

so that $[f]_\alpha$ represents the jump in the function value at $x = x_\alpha$, $[f']_\alpha$ the jump in the value of the first derivative of the function, and so on. The case $x_i = x_\alpha$ requires one to decide whether the terms $f(x_i)$, $f'(x_i)$, etc. are to be defined as left or right limits, this determining in turn whether or not a correction to the Taylor series will be required. Finally, for the case $x_\alpha < x_i$, no correction is required to predict $f(x_{i+1})$.

Eq. (17) is termed the *corrected Taylor series*. Proof of the validity of the expansion is given by Wiegmann and Bube [37]. The corrected Taylor series will be used in the next section to correct finite-difference schemes that have been obtained using the standard Taylor series.

3.2. Derivation of jump corrected finite-differences

In this section, the case $x_i < x_\alpha$ discussed above is considered. Using the correction term J_α, one can now modify any type of standard finite difference scheme to obtain its jump-corrected counterpart. The jump-corrected scheme will maintain the order of accuracy of the original scheme when the stencil passes through a jump-singularity of the function to which it is applied.

Consider, for example, a finite difference scheme for numerically approximating a second derivative $f^{(2)}$

$$L_{i-1} f_{i-1}^{(2)} + L_i f_i^{(2)} + L_{i+1} f_{i+1}^{(2)} = R_{i-1} f_{i-1} + R_i f_i + R_{i+1} f_{i+1}, \tag{20}$$

where the L_i and R_i are functions of the computational grid. The existence of stretched grids is allowed for in the present discussion. However, all references to formal accuracy are stated for the equidistant-grid case. If the function $f(x)$ is analytic in the entire domain $x_{i-1} \leqslant x \leqslant x_{i+1}$, then the scheme given in Eq. (20) will be accurate up to the truncation error built into the approximation it represents. For Eq. (20), this error is proportional to h^4.

At this point, a note on the notation used here, and throughout this article, is appropriate. The term L_i refers to the finite-difference coefficients appearing on the left side (hence "L") of the finite difference scheme (containing derivatives of the function f), and R_i the coefficient on the right side (hence "R") of the finite difference scheme (containing values of the function f itself). For example, in the case of an equidistant grid of step-size h, Eq. (20) becomes

$$\frac{1}{12}\left(f_{i-1}^{(2)} + 10f_i^{(2)} + f_{i+1}^{(2)}\right) = \frac{1}{h^2}\left(f_{i-1} - 2f_i + f_{i+1}\right) \tag{21}$$

so that $L_{i-1} = L_{i+1} = 1/12$, $L_i = 10/12$, $R_{i-1} = R_{i+1} = 1/h^2$, and $R_i = -2/h^2$. Appendix A provides details on how these coefficients, and indeed all those appearing subsequently in this article, are computed.

Continuing, consider the function depicted in Fig. 5. If the scheme given in Eq. (20) is applied to this case, a large error, possibly as high as $\mathcal{O}(h^{-2})$, will result. This fact, and simultaneously, the remedy, can be seen by observing what would happen if the terms f_k and $f_k^{(2)}$ in Eq. (20) were to be expanded about some point $x^* < x_\alpha$ using the corrected Taylor series approximation given by Eq. (17). Clearly, because it was *designed* to do so, all derivatives $f^{(n)}(x^*)$, $n = 0,1,\ldots,5$ will drop out of Eq. (20), leaving the derivatives starting at $f^{(6)}(x^*)$. This is the truncation error in the approximation represented by this equation. Additionally, because of the singularity at x_α, the jump correction terms J_α will also remain, transforming Eq. (20) into

$$L_{i+1}J_{\alpha 2} = R_{i+1}J_{\alpha 0} + \mathcal{O}(h^4), \tag{22}$$

where

$$J_{\alpha 0} = [f^{(0)}]_\alpha + [f^{(1)}]_\alpha h^+ + \frac{1}{2!}[f^{(2)}]_\alpha (h^+)^2 + \cdots, \tag{23}$$

$$J_{\alpha 2} = [f^{(2)}]_\alpha + [f^{(3)}]_\alpha h^+ + \frac{1}{2!}[f^{(4)}]_\alpha (h^+)^2 + \cdots. \tag{24}$$

One can now see why the error using the uncorrected finite-difference scheme given in Eq. (20) will be $\mathcal{O}(h^{-2})$: the coefficient R_{i+1}, which is $\mathcal{O}(h^{-2})$, multiplies the $\mathcal{O}(1)$ term $[f^{(0)}]_\alpha$ in the remainder Eq. (22). If, instead, the function were continuous ($[f^{(0)}]_\alpha = 0$), but the first derivative were discontinuous ($[f^{(1)}]_\alpha \neq 0$), then this error would be reduced to $\mathcal{O}(h^{-1})$. Finally, if all derivatives up to and including $f^{(5)}$ were continuous, then the original $\mathcal{O}(h^4)$ accuracy of the finite-difference scheme would be maintained.

To avoid the above situation all-together, it is now clear how the finite-difference scheme in Eq. (20) must be modified: to the right-hand side of this equation, the term $L_{i+1}J_{\alpha 2} - R_{i+1}J_{\alpha 0}$ must be added

$$L_{i-1}f_{i-1}^{(2)} + L_i f_i^{(2)} + L_{i+1}f_{i+1}^{(2)} = R_{i-1}f_{i-1} + R_i f_i + R_{i+1}f_{i+1} + (L_{i+1}J_{\alpha 2} - R_{i+1}J_{\alpha 0}), \tag{25}$$

where $J_{\alpha 0}$ and $J_{\alpha 2}$ are truncated by taking only enough terms to maintain $\mathcal{O}(h^4)$.

Now, when the terms in Eq. (25) are expanded using the corrected Taylor series, the jump terms will cancel, leaving only an $\mathcal{O}(h^4)$ remainder term. The question remains: how does one obtain the jumps in the function and derivatives at x_α that are required for the correction scheme just outlined? An answer to this question will be given in the next section.

3.3. Obtaining jumps for jump corrections

In contrast to the case involving singular forcing terms, the jumps in the function and its derivatives cannot be found independently of the actual solution of the problem. Recognizing this, one is then forced to use the solution itself to obtain the jumps. Numerically, this can be accomplished through the use of one-sided finite differences. The accuracy and spatial configuration of the one-sided finite-difference schemes must be chosen with some care, as the following example will illustrate. The issue of accuracy is presented first, followed by a discussion of spatial configuration, that is, which points should and should not be used in the one-sided stencils.

With reference to Fig. 6, the second derivative of a function $f(x)$ at point x_i is computed using an explicit finite difference with jump correction

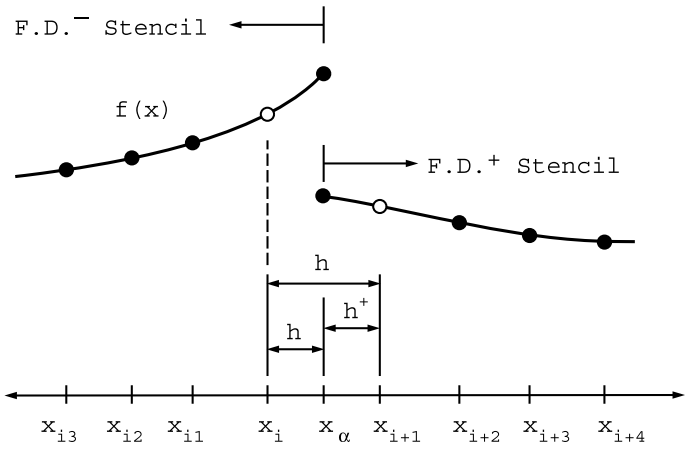

Fig. 6. Spatial configuration of one-sided finite difference stencils used to compute jumps in Eq. (27). The symbol ● indicates points used in the finite-difference scheme, whereas the symbol ○ indicates points not used in the scheme.

$$f_i^{(2)} = R_{i-1}f_{i-1} + R_i f_i + R_{i+1}f_{i+1} - R_{i+1}J_{\alpha 0} \tag{26}$$

and truncated jump correction term

$$J_{\alpha 0} = [f^{(0)}]_\alpha + [f^{(1)}]_\alpha h^+ + \frac{1}{2!}[f^{(2)}]_\alpha (h^+)^2 + \frac{1}{3!}[f^{(3)}]_\alpha (h^+)^3. \tag{27}$$

The expressions (26) and (27) together form an $\mathcal{O}(h^2)$ expression for $f_i^{(2)}$ in the presence of a jump singularity at the point $x = x_\alpha$. Starting with $[f^{(0)}]_\alpha$ and ending with $[f^{(3)}]_\alpha$, the jump terms must be known to $\mathcal{O}(h^4)$, $\mathcal{O}(h^3)$, $\mathcal{O}(h^2)$, and $\mathcal{O}(h)$, respectively, in order to ensure $\mathcal{O}(h^2)$ accuracy in Eq. (26). For example, the $[f^{(3)}]$ term is multiplied by R_{i+1}, an $\mathcal{O}(h^{-2})$ quantity, and by $(h^+)^3$ which together make a term $\mathcal{O}(h^1)$. If $[f^{(3)}]_\alpha$ is represented by a scheme with truncation error $\mathcal{O}(h)$, then the overall error contribution from the entire term $1/3!\ [f^{(3)}]_\alpha (h^+)^3$ to Eq. (26) will be $\mathcal{O}(h^2)$.

The objection might be raised, justifiably, that if the overall solution is $\mathcal{O}(h^2)$ accurate, then one will not be able to obtain the term $[f^{(1)}]_\alpha$ to the required $\mathcal{O}(h^3)$ (although not required, it will be assumed that $[f^{(0)}]_\alpha$ is always known). Indeed, in the present example, one order may well be lost locally. The effect of this phenomenon on the errors $\|\varepsilon\|_\infty$ and $\|\varepsilon\|_2$ is investigated through numerical experiments as discussed below. For now, the approach of discretizing the jump $[f^{(1)}]_\alpha$ using an $\mathcal{O}(h^3)$ scheme will be maintained.

In order to maintain the specified order of accuracy, each of the one-sided finite-difference schemes used to compute the jumps in Eq. (27) must contain four points. Again referring to Fig. 6, the spatial configuration of these four points is such that the jumps are computed as

$$[f^{(n)}]_\alpha = f^{(n)}_{\text{F.D.}^+} - f^{(n)}_{\text{F.D.}^-}, \tag{28}$$

where

$$f^{(n)}_{\text{F.D.}^+} = c_{n_{\alpha^+}} f_\alpha^+ + c_{n_{i+2}} f_{i+2} + c_{n_{i+3}} f_{i+3} + c_{n_{i+4}} f_{i+4}, \tag{29}$$

$$f^{(n)}_{\text{F.D.}^-} = c_{n_{\alpha^-}} f_\alpha^- + c_{n_{i-1}} f_{i-1} + c_{n_{i-2}} f_{i-2} + c_{n_{i-3}} f_{i-3}. \tag{30}$$

The coefficients c_{n_i} are used to determine a numerical approximation to the nth derivative of f by a linear combination of the f_i.

The + and − superscripts on f_α indicate, respectively, right and left limits at x_α. Note that the points x_i and x_{i+1} have intentionally not been used, and that because of this, many problems, numerical stability in

particular, have been avoided. Admitting arbitrarily shaped immersed boundaries, or further, moving immersed boundaries, it is clear that the singularities must be allowed to move arbitrarily close to, or even lie on top of, grid points. The spatial configuration of the finite-difference schemes described in this section allows just that.

4. Numerical method

4.1. The vorticity-transport equation

In this section, the numerical method for solving the vorticity transport equation, Eq. (10), with immersed boundaries is discussed. In this section, and all subsequent sections, it is assumed that the solution inside the immersed boundary is identically zero.

4.1.1. Temporal discretization

For time integration, either a second-order predictor–corrector or a fourth-order Runge–Kutta method is used. Both methods are explicit in time.

The vorticity-transport equation is written generically as

$$\frac{\partial \phi}{\partial t} = g(\phi). \tag{31}$$

The time integration scheme advances the solution ϕ of Eq. (31) from time t_n to $t_{n+1} = t_n + \Delta t$. For the predictor–corrector

$$\phi_1 = \phi_n + \Delta t g(\phi_n), \tag{32}$$

$$\phi_{n+1} = \phi_n + \frac{\Delta t}{2}\left(g(\phi_n) + g(\phi_1)\right) \tag{33}$$

and for the fourth-order Runge–Kutta

$$\phi_1 = \phi_n + \frac{\Delta t}{2} g(\phi_n), \tag{34}$$

$$\phi_2 = \phi_n + \frac{\Delta t}{2} g(\phi_1), \tag{35}$$

$$\phi_3 = \phi_n + \Delta t g(\phi_2), \tag{36}$$

$$\phi_{n+1} = \phi_n + \frac{\Delta t}{6}\left(g(\phi_n) + 2g(\phi_1) + 2g(\phi_2) + g(\phi_3)\right). \tag{37}$$

One intermediate variable can be eliminated by rewriting the scheme as

$$\phi_1 = \phi_n + \frac{\Delta t}{2} g(\phi_n), \tag{38}$$

$$\phi_2 = \phi_n + \frac{\Delta t}{2} g(\phi_1), \quad \phi_1 = \phi_1 + 2\phi_2, \tag{39}$$

$$\phi_2 = \phi_n + \Delta t g(\phi_2), \quad \phi_1 = \frac{1}{3}(-\phi_n + \phi_1 + \phi_2), \tag{40}$$

$$\phi_{n+1} = \phi_1 + \frac{\Delta t}{6} g(\phi_2). \tag{41}$$

This latter form of the time integration scheme was used in the present investigation.

4.1.2. Spatial discretization

To compute numerical approximations to the spatial derivatives appearing in Eq. (10), a three-point, fourth-order compact finite-difference scheme [14,13] is used. For the spatial first derivative, $f^{(1)}$, representing the terms $\partial(u\omega)/\partial x$ and $\partial(v\omega)/\partial y$, the scheme is

$$L^1_{i-1}f^{(1)}_{i-1} + L^1_i f^{(1)}_i + L^1_{i+1}f^{(1)}_{i+1} = R^1_{i-1}f_{i-1} + R^1_i f_i + R^1_{i+1}f_{i+1} + (L^1_I J_{\alpha 1} - R^1_I J_{\alpha 0}). \tag{42}$$

The second derivative, $f^{(2)}$, representing the terms $\partial^2\omega/\partial x^2$ and $\partial^2\omega/\partial y^2$ is computed as

$$L^2_{i-1}f^{(2)}_{i-1} + L^2_i f^{(2)}_i + L^2_{i+1}f^{(2)}_{i+1} = R^2_{i-1}f_{i-1} + R^2_i f_i + R^2_{i+1}f_{i+1} + (L^2_I J_{\alpha 2} - R^2_I J_{\alpha 0}). \tag{43}$$

In these two schemes, $I = i + 1$ if the jump singularity occurs for $x_i < x_\alpha < x_{i+1}$, in which case one has $h^+ = x_{i+1} - x_\alpha$ and

$$J_{\alpha 0} = [f^{(0)}]_\alpha + [f^{(1)}]_\alpha h^+ + \frac{1}{2!}[f^{(2)}]_\alpha (h^+)^2 + \frac{1}{3!}[f^{(3)}]_\alpha (h^+)^3 + \frac{1}{4!}[f^{(4)}]_\alpha (h^+)^4 + \frac{1}{5!}[f^{(5)}]_\alpha (h^+)^5, \tag{44}$$

$$J_{\alpha 1} = [f^{(1)}]_\alpha + [f^{(2)}]_\alpha h^+ + \frac{1}{2!}[f^{(3)}]_\alpha (h^+)^2 + \frac{1}{3!}[f^{(4)}]_\alpha (h^+)^3 + \frac{1}{4!}[f^{(5)}]_\alpha (h^+)^4, \tag{45}$$

$$J_{\alpha 2} = [f^{(2)}]_\alpha + [f^{(3)}]_\alpha h^+ + \frac{1}{2!}[f^{(4)}]_\alpha (h^+)^2 + \frac{1}{3!}[f^{(5)}]_\alpha (h^+)^3. \tag{46}$$

If the jump singularity occurs for $x_{i-1} < x_\alpha < x_i$, then $I = i-1$. In this case one has $h^- = x_\alpha - x_{i-1}$ and

$$J_{\alpha 0} = -[f^{(0)}]_\alpha + [f^{(1)}]_\alpha h^- - \frac{1}{2!}[f^{(2)}]_\alpha (h^-)^2 + \frac{1}{3!}[f^{(3)}]_\alpha (h^-)^3 - \frac{1}{4!}[f^{(4)}]_\alpha (h^-)^4 + \frac{1}{5!}[f^{(5)}]_\alpha (h^-)^5, \tag{47}$$

$$J_{\alpha 1} = -[f^{(1)}]_\alpha + [f^{(2)}]_\alpha h^- - \frac{1}{2!}[f^{(3)}]_\alpha (h^-)^2 + \frac{1}{3!}[f^{(4)}]_\alpha (h^-)^3 - \frac{1}{4!}[f^{(5)}]_\alpha (h^-)^4, \tag{48}$$

$$J_{\alpha 2} = -[f^{(2)}]_\alpha + [f^{(3)}]_\alpha h^- - \frac{1}{2!}[f^{(4)}]_\alpha (h^-)^2 + \frac{1}{3!}[f^{(5)}]_\alpha (h^-)^3. \tag{49}$$

When x_α falls exactly on a grid-point, the decision must be made as to whether the function value here is set to the left or right limit. The scheme is then corrected or not corrected accordingly. For example, if $x_\alpha = x_i$, and $f(x_i) = \lim^+_{x\to x_i} f(x)$, then $I = i-1$. However, if $x_\alpha = x_{i-1}$ and $f(x_{i-1}) = \lim_{x\to x^+_{i-1}} f(x)$, then no correction will be required. The other cases can be handled similarly, and if done consistently, pose no particular problems. Note that if no correction is required, then the terms $(L^1_I J_{\alpha 1} - R^1_I J_{\alpha 0})$ and $(L^2_I J_{\alpha 2} - R^2_I J_{\alpha 0})$ are not included in the finite difference scheme.

All jumps are discretized at the immersed boundary using one-sided finite-differences of an order such that the schemes in Eqs. (42) and (43) maintain their formal (fourth-order) accuracy if f were known to one order higher (fifth). The term $[f^{(1)}]_\alpha$, for example, is discretized to $\mathcal{O}(h^5)$, $[f^{(2)}]_\alpha$ to $\mathcal{O}(h^4)$, etc., requiring a total of six points in each case.

4.1.3. Solution algorithm

Using the temporal and spatial discretization schemes described above, advancement of the vorticity-transport equation, Eq. (10), from time t^n to time t^{n+1} proceeds as follows. At time t^n, the vorticity ω and velocities u and v are known both inside the domain and on the immersed boundary, hence the first and second partial spatial derivatives, e.g. $\partial(u\omega)/\partial x$, $\partial^2\omega/\partial x^2$, etc., appearing in Eq. (10) can be numerically approximated using the schemes presented in Section 4.1.2.

Along lines x = const. or y = const. which do not intersect the immersed boundary, numerical derivatives are computed by solving tridiagonal systems of equations as is usually done when using compact finite differences [14] in the absence of immersed boundaries. On the other hand, lines which do intersect the immersed boundary require the extra correction terms seen in Eqs. (42) and (43). These correction terms are required each time, but *only if*, the three-point stencil passes through the immersed boundary. These

correction terms are explicit functions of ω, u and v, and can be computed at time t_n and added to the right-hand side of the compact system before solving for derivatives. So that there is no confusion, a specific example is given next.

Consider that Fig. 5 represents a line $y = \text{const.}$ along which one would like to numerically approximate the first derivative $\partial f/\partial x$, with e.g. $f(x) = u(x)\omega(x)$. The immersed boundary is intersected between points x_i and x_{i+1} at $x = x_\alpha$. Points $x_{i+1}, x_{i+2}, \ldots, x_{i+M}$ to the right of x_α are inside the body, hence here $f(x) \equiv 0$ (despite the fact that it has not been so depicted in the figure). At grid point x_{i-1} the equation is

$$L^1_{i-2} f^{(1)}_{i-2} + L^1_{i-1} f^{(1)}_{i-1} + L^1_i f^{(1)}_i = R^1_{i-2} f_{i-2} + R^1_{i-1} f_{i-1} + R^1_i f_i \tag{50}$$

No correction term appears because the stencil does not intersect the immersed boundary. In contrast, at grid point x_i, the stencil does intersect the immersed boundary, and the correction $(L^1_{i+1} J_{\alpha 1} - R^1_{i+1} J_{\alpha 0})$ is required

$$L^1_{i-1} f^{(1)}_{i-1} + L^1_i f^{(1)}_i + L^1_{i+1} f^{(1)}_{i+1} = R^1_{i-1} f_{i-1} + R^1_i f_i + R^1_{i+1} f_{i+1} + (L^1_{i+1} J_{\alpha 1} - R^1_{i+1} J_{\alpha 0}) \tag{51}$$

The terms $J_{\alpha 1}$ and $J_{\alpha 0}$ are given in Eqs. (44) and (45), where they themselves are seen to involve additional defined terms such as $[f^{(1)}]_\alpha$. Taking $[f^{(1)}]_\alpha$ as an example, this term is computed as

$$[f^{(1)}]_\alpha = f^{(1)}_{\text{F.D.}^+} - f^{(1)}_{\text{F.D.}^-}, \tag{52}$$

where $f^{(1)}_{\text{F.D.}^+} \equiv 0$ because it is inside the immersed boundary, and

$$f^{(1)}_{\text{F.D.}^-} = c_{1_{\alpha^-}} f^-_\alpha + c_{1_{i-1}} f_{i-1} + c_{1_{i-2}} f_{i-2} + c_{1_{i-3}} f_{i-3} + c_{1_{i-4}} f_{i-4} + + c_{1_{i-5}} f_{i-5} \tag{53}$$

is an explicit finite difference scheme for numerically approximating the (right-hand limit) derivative $f^{(1)}_{\text{F.D.}^-}$. Note that it is through the term f^-_α appearing in the above equation that variables u, v and ω assert their influence as immersed-boundary boundary conditions. Finally, at point x_{i+1}, the equation is simply

$$f^{(1)}_{i+1} = 0. \tag{54}$$

This equation, together with Eq. (50), (51), and similar equations for each of the other discrete points $i = 1,2,\ldots,N_x$ along the line $y = \text{const.}$ form a tridiagonal system of equations which can be solved with any standard numerical linear algebra technique, such as the Thomas algorithm, to obtain the numerical approximation to the derivative $f^{(1)}(x)$ at grid points x_i. Naturally, the scheme is identical for y derivatives along lines $x = \text{const.}$

One additional note on boundary conditions must be made here. Near the immersed boundary $\partial\Omega_i$, the value of the function f has been seen to directly influence the value of the computed numerical derivative through the term f^-_α (or f^+_α, as the case may be). On the computational boundary $\partial\Omega_o$, i.e. at $i = 1$ and $i = N_x$ in the present example, the compact schemes require that a value be specified for the derivative being computed. For the validation case described in the next section, exact analytical derivatives were provided from the known analytical solution, Eq. (57). If the derivatives are not known, one-sided schemes may, for example, be used as an alternative – the exact choice is not dependent on the immersed interface scheme discussed in this article. Where of interest, specific boundary conditions used for the solution of the Navier–Stokes equations will be discussed in Section 5.

With the spatial derivatives known at every point in the computational domain, the solution can be advanced in time using an explicit Runge–Kutta time integration scheme in the usual manner.

4.1.4. Validation

In this section, Eq. (10) is solved with u and v specified

$$u(x,y,t) = e^{-\alpha t}\sin(\omega x)\sin(\omega y), \tag{55}$$

$$v(x,y,t) = e^{-\alpha t}\cos(\omega x)\cos(\omega y). \tag{56}$$

A forcing term is added to the right-hand side of Eq. (10), and boundary conditions are imposed such that the analytical solution is known to be

$$\omega(x,y,t) = 2\omega e^{-\alpha t}\sin(\omega x)\cos(\omega y). \tag{57}$$

The constants are selected to be $\omega = 2\pi$ and $\alpha = 1$. By comparing the numerical solution of Eq. (10) with the above analytical solution, the spatial and temporal convergence properties of the immersed boundary scheme, as applied to the convection–diffusion equation, can be numerically determined.

The numerical solution is computed on the square domain $x, y \in [0,1]$, and the immersed boundary is a circle of radius $r = 0.1$ centered at $(x,y) = (0.5,0.5)$. The fourth-order scheme described above is used for the spatial discretization, with fourth-order corrections on the immersed boundary, and a second-order predictor–corrector method is used for time integration.

Table 1 presents the results of the spatial convergence study. The numerical solution to Eq. (10) is computed on several grids with the time-step held fixed. The resulting numerical solution is compared with the analytical solution, Eq. (57), and the maximum absolute error is found as a function of h: $\|\varepsilon\|_\infty = Ah^n + B$. From the data in Table 1, the exponent n is found to be 4.0; for $\|\varepsilon\|_2$, the exponent n is also 4.0. It can be seen that the loss of accuracy, about which had been speculated in the discussion above, has not materialized. There it was pointed out that the immersed boundary scheme appeared to require that the numerical solution be known to one order higher than was available.

The temporal convergence study is carried out in a similar fashion. The numerical solution to Eq. (10) is computed for several different time-steps, this time with the spatial grid held fixed. Table 2 presents the results of the temporal convergence study. The resulting numerical solutions are again compared with the analytical solution, and the maximum absolute error is found as a function of Δt: $\|\varepsilon\|_\infty = A\Delta t^n + B$. From the data in Table 2, the exponent n is found to be 2.0; for $\|\varepsilon\|_2$, $n = 2.0$ is also found. Thus, second-order temporal accuracy is maintained.

It was empirically determined that, for numerical stability, a safe CFL criterion was approximately $\mathrm{CFL} \leqslant 0.3$, with the diffusion number satisfying $\mathrm{DFL} \leqslant 0.25$, where $\mathrm{CFL} = c\Delta t/\Delta x$ and $\mathrm{DFL} = \alpha\Delta t/\Delta x^2$ (c is a convection velocity, α a diffusivity).

4.2. The stream function equation

In this section, the numerical solution of the equation for the stream function ψ, Eq. (11), with immersed boundaries is discussed. The equation for the stream function is a Poisson equation of the form

$$\nabla^2 f(x,y) = \rho(x,y). \tag{58}$$

Table 1
Error ($\varepsilon = f_a - f_n$) in the numerical solution of Eq. (10) at $t = 0.4$ on grids of size $N \times N$ (or step-size $h = 1/(N-1)$) with fixed Δt

N	$\|\varepsilon\|_\infty$	$\|\varepsilon\|_2$
65	3.113409×10^{-6}	1.101375×10^{-6}
129	1.949443×10^{-7}	6.972817×10^{-8}
257	1.210198×10^{-8}	4.368400×10^{-9}

The error is $\|\varepsilon\|_\infty = Ah^n + B$, with $A = 51.49$, $n = 4.0$, and $B = -1.1875 \times 10^{-10}$; $\|\varepsilon\|_2 = Ah^n + B$, with $A = 17.03$, and $n = 4.0$, and $B = -5.2543 \times 10^{-11}$.

Table 2
Error ($\varepsilon = f_a - f_n$) in the numerical solution of Eq. (10) at $t = 0.2$ on grids of size 61×61 for various time-steps Δt

Δt	$\|\varepsilon\|_\infty$	$\|\varepsilon\|_2$
4.0×10^{-5}	4.810835×10^{-6}	1.695369×10^{-6}
2.0×10^{-5}	4.807736×10^{-6}	1.694273×10^{-6}
1.0×10^{-5}	4.806962×10^{-6}	1.693999×10^{-6}
5.0×10^{-6}	4.806768×10^{-6}	1.693930×10^{-6}

The error is $\|\varepsilon\|_\infty = A\Delta t^n + B$, with $A = 2.61$, and $n = 2.0$, and $B = 4.81 \times 10^{-6}$; $\|\varepsilon\|_2 = A\Delta t^n + B$, with $A = 0.93$, and $n = 2.0$, and $B = 1.69 \times 10^{-6}$.

4.2.1. Discretization

The discretization of Eq. (58) is based on two 1D, fourth-order compact finite-difference schemes [2]

$$L_{xx_i} f_{xx_{ij}} = R_{xx_i} f_{ij}, \tag{59}$$

$$L_{yy_j} f_{yy_{ji}} = R_{yy_j} f_{ij}. \tag{60}$$

Here, L_{xx}, L_{yy}, R_{xx} and R_{yy} denote the coefficients in the schemes, whereas the similarly subscripted f_{xx} and f_{yy}, on the other hand, represent numerical approximations to the second partial derivatives in the x and y directions, respectively. When no jump corrections are needed, the 2D, nine-point compact scheme centered at point (i,j) inside the computational domain is

$$L_{ij} f_{ij} = R_{ij} \rho_{ij}, \tag{61}$$

where

$$L_{ij} = R_{xx_i} L_{yy_j} + L_{xx_i} R_{yy_j}, \tag{62}$$

$$R_{ij} = L_{xx_i} L_{yy_j} \tag{63}$$

The above expressions for L_{ij} and R_{ij} are derived using the Poisson equation, Eq. (58), and the schemes in Eqs. (59) and (60) as follows:

$$f_{xx_{ij}} + f_{yy_{ij}} = \rho_{ij}, \tag{64}$$

$$L_{xx_i} L_{yy_j} (f_{xx_{ij}} + f_{yy_{ij}}) = L_{xx_i} L_{yy_j} \rho_{ij}, \tag{65}$$

$$L_{yy_j} (L_{xx_i} f_{xx_{ij}}) + L_{xx_i} (L_{yy_j} f_{yy_{ij}}) = L_{xx_i} L_{yy_j} \rho_{ij}, \tag{66}$$

$$L_{yy_j} (R_{xx_i} f_{ij}) + L_{xx_i} (R_{yy_j} f_{ij}) = L_{xx_i} L_{yy_j} \rho_{ij}, \tag{67}$$

$$(R_{xx_i} L_{yy_j} + L_{xx_i} R_{yy_j}) f_{ij} = L_{xx_i} L_{yy_j} \rho_{ij}, \tag{68}$$

$$L_{ij} f_{ij} = R_{ij} \rho_{ij}. \tag{69}$$

Note that the 2D scheme shown on the last line of this derivation has been obtained by combining two 1D schemes, one in x and one in y. A consequence of this is that when a 2D stencil intersects the immersed boundary, the corresponding 2D scheme can be corrected by using the method employed for 1D schemes. For example, when, as shown in Fig. 7, the nine-point stencil intersects an immersed boundary, jump corrections to the 1D finite-difference schemes in Eqs. (59) and (60) must be made, as described in Section 4.1.2

$$L_{xx_i} f_{xx_{ij}} = R_{xx_i} f_{ij} - J_{\alpha x I}, \tag{70}$$

$$L_{yy_j} f_{yy_{ij}} = R_{yy_j} f_{ij} - J_{\alpha y J}, \tag{71}$$

[2] Index notation, with the standard summation convention, is used here.

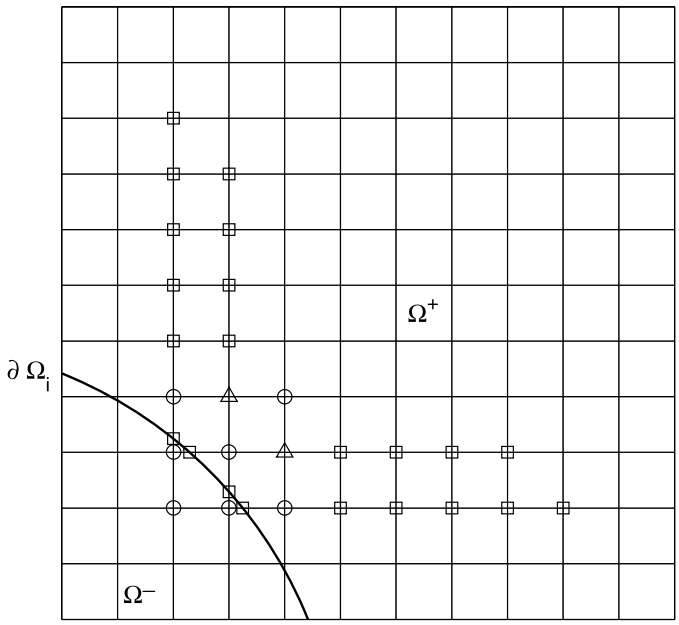

Fig. 7. The intersection of the nine-point finite-difference stencil (○) given in Eq. (61) with an immersed boundary, and jump-correction terms (□). The points where (○) and (□) overlap are represented by (△).

where, for convenience, the following definitions are made

$$J_{\alpha x I} = -(L_{xx_I} J_{\alpha 2x} - R_{xx_I} J_{\alpha 0x}), \tag{72}$$

$$J_{\alpha y J} = -(L_{yy_J} J_{\alpha 2y} - R_{yy_J} J_{\alpha 0y}) \tag{73}$$

and

$$I = \begin{cases} i-1, & x_{i-1} < x_\alpha < x_i, \\ i+1, & x_i < x_\alpha < x_{i+1}, \end{cases} \tag{74}$$

and similarly for J (J the subscript, not to be confused with the jump J). Note that $I = I(j)$, and $J = J(i)$, and that the corrections are being made as if the schemes are being used one-dimensionally.

The resulting 2D discretization of the Poisson equation with jump corrections becomes

$$L_{ij} f_{ij} = R_{ij}\rho_{ij} + (L_{yy_j} J_{\alpha x I(j)} + L_{xx_i} J_{\alpha y J(i)}). \tag{75}$$

Again, so that the notation used does not lead to confusion, the correction term in the finite-difference scheme given in Eq. (75) is written out, with explicit summation, for the stencil shown in Fig. 7, assuming for convenience that the stencil is centered at $(i,j) = (2,2)$

$$\sum_{i=1}^{3}\sum_{j=1}^{3} L_{ij} f_{ij} = \sum_{i=1}^{3}\sum_{j=1}^{3} R_{ij}\rho_{ij} + L_{xx_1} J_{\alpha y 3} + L_{xx_2} J_{\alpha y 1} + L_{yy_1} J_{\alpha x 3} + L_{yy_2} J_{\alpha x 1}. \tag{76}$$

For further illustration, the term $J_{\alpha y 3}$ can be expanded using Eq. (73) as

$$J_{\alpha y 3} = -(L_{yy_3} J_{\alpha 2y} - R_{yy_3} J_{\alpha 0y}) \tag{77}$$

where, again, the terms $J_{\alpha 2y}$ and $J_{\alpha 0y}$ have been defined in Section 4.1.2. Note that these terms are linear in the yet unknown function values f_{ij}, as well as in the known values $f(x_\alpha, y_\alpha)$ on the immersed boundary at the

locations where the immersed boundary is intersected by the grid. These known function values are the boundary conditions on the immersed boundary (only Dirichlet boundary conditions are considered here), and are represented by the terms f_α^+ and f_α^- in Eqs. (29) and (30).

4.2.2. Solution algorithm

The discretization of the Poisson equation without immersed boundaries leads to a system of equations $\mathbf{Ax} = \mathbf{b}$ where the matrix $\mathbf{A}$ has the following structure:

$$\mathbf{A} = \begin{bmatrix} B_1 & U_1 & 0 & 0 & 0 & 0 \\ L_2 & B_2 & U_2 & 0 & 0 & 0 \\ 0 & L_3 & B_3 & U_3 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & L_{N_x-1} & B_{N_x-1} & U_{N_x-1} \\ 0 & 0 & 0 & 0 & L_{N_x} & B_{N_x} \end{bmatrix} \tag{78}$$

with tridiagonal blocks L_i, B_i, and U_i of size $N_y \times N_y$, and where $i = 1, \ldots, N_x$, sweeping in j. This system $\mathbf{Ax} = \mathbf{b}$ is solved using a multigrid technique together with an ILLU (incomplete line LU decomposition) relaxation method [31]. The ILLU has very good convergence properties, even for highly stretched grids. Standard relaxation methods, for example successive over relaxation (SOR), successive line over relaxation (SLOR), or Gauss–Seidel, which perform well on equidistant grids will perform poorly on stretched grids [38,4].

As the solution procedure without immersed boundaries requires only minor modifications to take them into account, the discussion of the ILLU procedure will begin assuming that no immersed boundaries are present. Later, the extensions required to include immersed boundaries will be discussed. In the ILLU decomposition, one attempts to find a matrix $\mathbf{D}$ such that

$$\mathbf{A} = (\mathbf{L} + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}), \tag{79}$$

where $\mathbf{L}$ is lower block-tridiagonal, $\mathbf{U}$ is upper block-tridiagonal, and $\mathbf{D}$ is block-tridiagonal. Eq. (79) can be expanded as

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U} + \mathbf{L}\mathbf{D}^{-1}\mathbf{U}, \tag{80}$$

where $\mathbf{L}\mathbf{D}^{-1}\mathbf{U}$ is the block-diagonal matrix

$$\mathbf{L}\mathbf{D}^{-1}\mathbf{U} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & L_2 D_1^{-1} U_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & L_3 D_2^{-1} U_2 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & L_{N_x} D_{N_x-1}^{-1} U_{N_x-1} \end{bmatrix}. \tag{81}$$

From Eqs. (80) and (81), it can be seen that $\mathbf{D}$ can be computed as

$$D_1 = B_1, \quad D_i = B_i - L_i D_{i-1}^{-1} U_{i-1}, \quad i = 2, 3, \ldots, N_x. \tag{82}$$

The matrix D_i^{-1} is in general full, so the approximation of taking only the tridiagonal part is made, resulting in algorithm 82 being modified as

$$\tilde{D}_1 = B_1, \quad \tilde{D}_i = B_i - \mathrm{tridiag}(L_i \tilde{D}_{i-1}^{-1} U_{i-1}), \ i = 2, 3, \ldots, N_x. \tag{83}$$

The ILLU decomposition of **A** is now defined to be

$$\mathbf{A} \approx (\mathbf{L} + \tilde{\mathbf{D}})\tilde{\mathbf{D}}^{-1}(\tilde{\mathbf{D}} + \mathbf{U}). \tag{84}$$

With this decomposition, the single-grid iterative method for solving $\mathbf{Ax} = \mathbf{b}$ with starting guess $\mathbf{x}^n$ becomes:

(1) $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}^n$,
(2) $(\mathbf{L} + \tilde{\mathbf{D}})\mathbf{c} = \mathbf{r}$,
(3) $(\tilde{\mathbf{D}} + \mathbf{U})\mathbf{e} = \tilde{\mathbf{D}}\mathbf{c}$,
(4) $\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{e}$.

A considerable amount of detail has been left out of the present description of the ILLU algorithm. An efficient implementation this algorithm requires this additional information, and the interested reader is referred to [31].

The extensions required to included immersed boundaries will now be given. As discussed in Section 4.2.1, the jump-corrections to the nine-point compact discretization are solution dependent. Normally, this would require one to introduce terms into locations (row m, column n) in the matrix **A** which, unlike the tridiagonal matrices L_i, B_i, and U_i above, are not at all regular. Here, these terms are not introduced directly, so that the matrix **A** always denotes a discretization of the Poisson equation *without* jump-corrections. Specifically, this means that **A** only contains the coefficients L_{ij} appearing on the left-hand side of Eq. (75). The irregularly located right-hand side terms of this equation which are linear in f_{ij} are not moved to the left-hand side. Instead, the irregular entries are handled using the following strategy.

A is ILLU decomposed, and `ILLUSolve`(**r**) is defined as a function returning the solution **e** of

$$(\mathbf{L} + \tilde{\mathbf{D}})\tilde{\mathbf{D}}^{-1}(\tilde{\mathbf{D}} + \mathbf{U})\mathbf{e} = \mathbf{r}. \tag{85}$$

The relaxation procedure then becomes

(1) $\mathbf{r} = \mathbf{b} - (\mathbf{A} + \mathbf{A}')\mathbf{x}^n$,
(2) $\mathbf{e} \leftarrow$ `ILLUSolve`(**r**),
(3) $\mathbf{r}^* \leftarrow c_1\mathbf{r} + c_2$`JumpCorrect`(**e**),
(4) $\mathbf{e}^* \leftarrow$ `ILLUSolve`(**r***),
(5) $\mathbf{x}^{n+1} \leftarrow \mathbf{x}^n + \mathbf{e}^*$,

where $c_1 = 0.8$ and $c_2 = 0.2$, and the function `JumpCorrect` adds all jump-correction terms to the right-hand side of $\mathbf{Ax} = \mathbf{b}$ given a guess solution $\mathbf{x}^n$. Specifically, in the presence of immersed boundaries, to **A** is added a matrix $\mathbf{A}'$ and $(\mathbf{A} + \mathbf{A}')\mathbf{x} = \mathbf{b}$ is solved to obtain the jump-corrected solution **x** to the Poisson equation. The function `JumpCorrect`($\mathbf{x}^n$), then, returns the column vector $-\mathbf{A}'\mathbf{x}^n$.

The procedure just outlined was empirically determined, and was found to converge in all cases in which it was employed to solve the Poisson equation with immersed boundaries. It is a convenient algorithm that allows the ILLU decomposition to be performed on the regular (block tridiagonal) matrix **A**, yet it easily accommodates the irregularly located entries introduced by immersed boundaries.

4.2.3. Validation

The Poisson equation (58) is solved numerically on the square domain $x, y \in [0,1]$, and the immersed boundary is a circle of radius $r = 0.1$ centered at $(x,y) = (0.5,0.5)$. The right-hand side source term $\rho(x,y)$ and boundary conditions are chosen such that the analytical solution is

$$f(x,y) = (-1/\omega)\sin(\omega x)\cos(\omega y) \tag{86}$$

with $\omega = 2\pi$.

A computed solution and the corresponding error are shown in Fig. 8. The reader's attention is again drawn to the sharp interface in the solution obtained, and the relative smoothness of the error distribution in the vicinity of the immersed boundary. Table 3 presents the results of the numerical experiment. The solution and the corresponding error is computed on several grids, and the errors $\|\varepsilon\|_\infty$ and $\|\varepsilon\|_2$ are tabulated as a function of grid size h. The error is $\|\varepsilon\|_\infty = Ah^n$, with $n = 4.05$, and $\|\varepsilon\|_2 = Ah^n$, with $n = 4.07$. Both measures show fourth-order convergence.

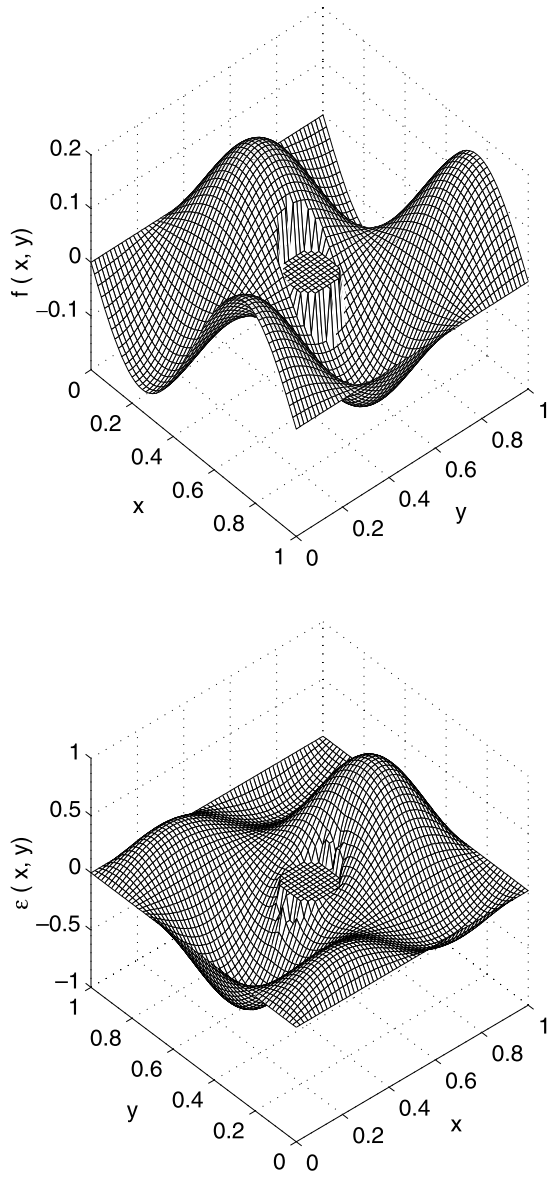

Fig. 8. Numerical solution, top, and corresponding error, $\varepsilon = f_a - f_n$, bottom, of Eq. (58).

Table 3
Error ($\varepsilon = f_a - f_n$) in the numerical solution of Eq. (58) on grids of size $N \times N$ (or step-size $h = 1/(N-1)$)

N	$\|\varepsilon\|_\infty$	$\|\varepsilon\|_2$
41	3.851207×10^{-7}	1.411451×10^{-7}
81	2.317782×10^{-8}	8.371542×10^{-9}
161	1.428926×10^{-9}	5.175142×10^{-10}

The error is $\|\varepsilon\|_\infty = Ah^n$, with $A = 1.126$, and $n = 4.04$; $\|\varepsilon\|_2 = Ah^n$, with $A = 0.425$, and $n = 4.05$.

4.3. Computing wall vorticity

In the present work, the wall vorticity ω is computed from its definition, Eq. (8). Where allowed by the intersection, explicit finite-differences are used to compute u_y and v_x. For example, referring to Fig. 9, one can see that at the intersection denoted by the symbol □, one can easily compute v_x, but the computation of u_y would require a 2D finite-difference scheme. For the intersection denoted by ○, the reverse is true.

The following strategy was developed to avoid the use of 2D stencils in the cases mentioned above. The case of computing v_x at the intersection denoted by the symbol ○ is considered. If the tangent vector makes an angle ϕ with the x axis such that $\pi/4 \leqslant |\phi| \leqslant 3\pi/4$, then v_x at ○-intersections are computed by interpolating nearest neighbor □-intersection values along the arc defining the immersed boundary (all angles are given assuming a $\pm\pi$ branch-cut in the arg function). For the angles ϕ not satisfying this inequality, the known derivative v_s computed along the arc length s, and v_y computed using an explicit finite-difference scheme are used to obtain

$$v_x = \frac{v_s - v_y y_s}{x_s}, \tag{87}$$

where the tangent vector $\boldsymbol{\tau} = (x_s, y_s)$. A similar strategy is used to obtain u_y at □-intersections.

To numerically validate this strategy for computing wall vorticity, the values of u and v are set to analytical functions

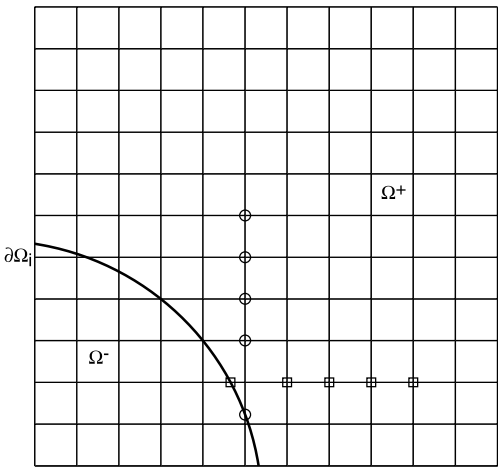

Fig. 9. Stencils used for computing wall vorticity at an immersed boundary: first derivative in y (○), and first derivative in x (□).

$$u(x,y) = \sin(\omega x)\sin(\omega y), \tag{88}$$

$$v(x,y) = \cos(\omega x)\cos(\omega y) \tag{89}$$

on an equidistant grid $x,y \in [0,1]$ with a circular immersed boundary of radius $r = 0.1$ centered at $(x,y) = (0.5,0.5)$. Fourth-order finite-difference stencils are used, where allowed by the grid/immersed boundary intersection, to compute u_y and v_x; interpolation along the immersed boundary is also done to fourth order. As seen in Table 4, the resulting wall vorticities are computed to fourth-order accuracy.

For flows around bluff bodies, an inviscid flow solution was often specified as an initial condition. However, special care is needed here, as suddenly imposing no-slip boundaries on the inviscid solution may cause the numerical solution to blow-up. To circumvent this problem, first-order differences are used to compute wall vorticity during the initial start-up phase. Shortly thereafter, however, higher-order differences are switched on for the remainder of the calculation. For the Navier–Stokes calculations using stretched grids near the immersed boundary, a third-order finite-difference scheme (consisting of four points) was used.

4.4. Complete solution algorithm for Navier–Stokes equations

The solution of the Navier–Stokes equations in stream function–vorticity formulation consists mainly of the three computational components discussed above, specifically, the time integration of a convection–diffusion equation, the vorticity-transport equation, Eq. (10), the solution of a Poisson equation for the stream function, Eq. (11), and the computation of the wall vorticity.

The algorithm starts at time t^n with discrete variables u, v and ω known everywhere in the domain $\Omega^- \cup \Omega^+$, and, additionally, on both the computational boundary $\partial\Omega_o$ and the immersed boundary $\partial\Omega_i$. The vorticity-transport equation, Eq. (10), can be used to advance to vorticity in time from t^n to t^{n+1}, as discussed in Section 4.1.

Next, the stream function equation, Eq. (11), is solved using the scheme discussed in Section 4.2 to obtain the stream function ψ at time t^{n+1}. The source term on the right-hand side of Eq. (11) is equal to the vorticity ω at t^{n+1}. At this stage, the wall vorticity at the immersed boundary is not known. However, for the cases considered in the present investigation, it is also not needed. Specifically, the computational boundary $\partial\Omega_o$ is either:

(1) An inflow boundary where ω is known: for flow past a bluff body, $\omega = 0$; for a boundary layer flow, $\omega = \omega_B$, where ω_B is a solution of Prandtl's boundary layer equations.
(2) A free stream boundary assumed to be irrotational, so that $\omega = 0$.
(3) An outflow boundary where ω is ramped to a specified distribution, e.g. zero in the case of flow past a bluff body. See [21] for a discussion of outflow treatment using buffer domain techniques.

Were the computational boundary to consist entirely, or even partly, of a no-slip wall, then the wall vorticity would be required. In such a case, an influence matrix method could be used [7]. On the other hand, at grid points inside the domain Ω^+, the vorticity is known at time t^{n+1} from time integration of its transport equation. At grid points inside the immersed boundary (in Ω^-), $\omega = 0$. A grid point whose location (x,y)

Table 4
Error ($\varepsilon = \omega_a - \omega_n$) in the numerical computation of wall vorticity on grids of size $N \times N$ (or step-size $h = 1/(N-1)$)

N	$\|\varepsilon\|_\infty$	$\|\varepsilon\|_2$
129	1.3946×10^{-5}	3.9883×10^{-6}
257	8.6255×10^{-7}	2.0887×10^{-7}
513	5.4499×10^{-8}	1.6603×10^{-8}

The error is $\varepsilon = Ah^n$, with A=3.7251 × 10^3, $n = 4.0$ for $\|\varepsilon\|_\infty$, and A=7.9925 × 10^2, $n = 4.0$ for $\|\varepsilon\|_2$.

Table 5
Parameters used to compute the flow around a circular cylinder

Parameter	Value
Domain size ($\lambda = 0.056$, $\lambda = 0.023$)	
Inflow x_1	0.0, 0.0
Outflow x_2	46.5795, 46.5795
Free stream y_1	−8.9401, −21.3278
Free stream y_2	8.9401, 21.3278
Grid size ($\lambda = 0.056$, $\lambda = 0.023$)	
N_x	641, 641
N_y	281, 321
$\Delta x_{\min}$	1.386×10^{-2} ($\approx D/72$)
$\Delta y_{\min}$	1.148×10^{-2} ($\approx D/87$)
Miscellaneous	
Cylinder center	(10.0,0.0)
Δt	5.0×10^{-4} to 1.0×10^{-3}
Time integration	second order P-C
Buffer start x_B	35.2033

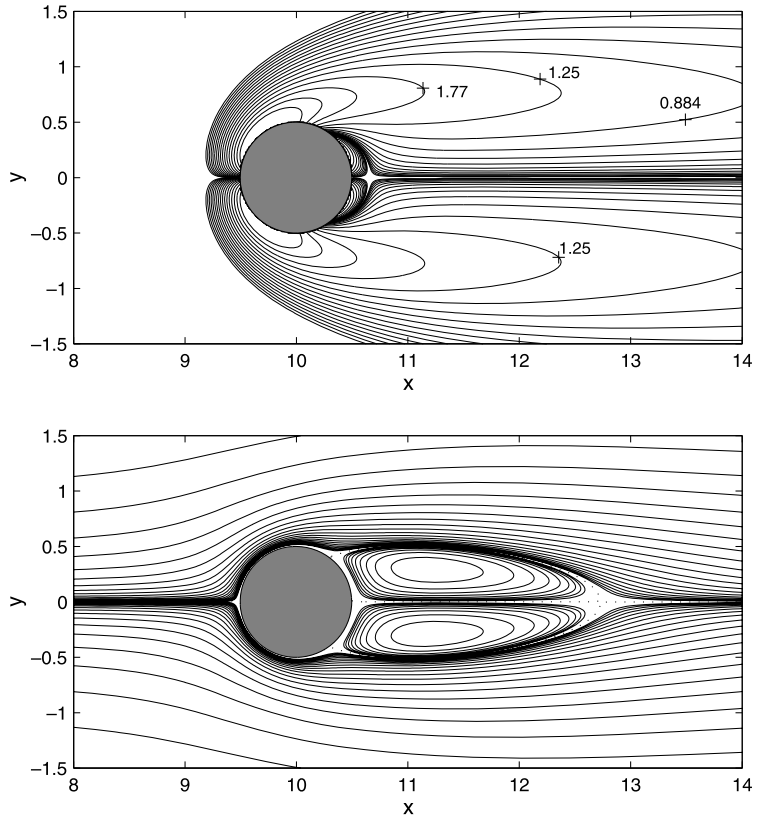

Fig. 10. Uniform flow past a circular cylinder, $Re_D = 40$, $\lambda = 0.056$. Exponentially spaced contours of vorticity, top, and stream function, bottom.

exactly coincides with the immersed boundary can also be treated as if it were inside the immersed boundary using the convention that $f(x_\alpha) = \lim_{x \to x_\alpha^{\text{in}}} f(x)$. Here, the superscript "in" is meant to indicate that the limit is taken from inside the body, and that the finite difference scheme for the Poisson equation should be jump-corrected accordingly.

The validity of the above strategy was confirmed by allowing the immersed flat plate described in Section 5.2 to come arbitrarily close to a set $i = 1,2,\ldots,N_x$ of grid points $j = \text{const}$, and monitoring the resulting solution during its approach. To within the order of the numerical scheme, no difference was detected between solutions where the immersed plate was placed half-way between grid points j and $j + 1$, and solutions where the plate was placed arbitrarily close to one of the grid points.

Once the stream function has been computed, the velocities u and v can be determined from Eq. (9). With the velocity field determined, the wall vorticity can be computed as a final step, using the scheme discussed in Section 4.3.

5. Results

5.1. Uniform flow past a circular cylinder

In this section, the developed immersed boundary method is used to compute the 2D flow around a circular cylinder placed in a uniform free-stream. The salient features of the computed flow fields are compared with results, both experimental and computational, available in the literature. Reynolds numbers ($Re_D = U_\infty D/\nu$) in the range $Re_D = 20$ to $Re_D = 200$ are considered, spanning the range of steady, steady-to-unsteady transitional, and unsteady flow regimes. Experimentally, it has been found that the well-known phenomenon of periodic vortex shedding first appears for Reynolds numbers around 40–50; for smaller Reynolds numbers, the flow is found to be steady [33,1].

The circular cylinder has served as a validation case for several past immersed boundary implementations including those of Goldstein et al. [12], Saiki and Biringen [28], and Calhoun [5]. The former two studies make use of a singular forcing term to represent the immersed boundary, the lower accuracy of which

Table 6
Steady flow past a circular cylinder: length L of standing eddy behind cylinder, locations a and b of the vortex centers, separation angle θ, and drag coefficient C_D for $Re_D = 20$ and $Re_D = 40$

	L	a	b	θ	C_D
$Re_D = 20$					
Fornberg [11]	0.91	–	–	45.7°	2.00
Dennis and Chang [8]	0.94	–	–	43.7°	2.05
Coutanceau and Bouard [6]*	0.93	0.33	0.46	45.0°	–
Tritton [33]*	–	–	–	–	2.09
present, $\lambda = 0.056$	0.93	0.36	0.43	43.9°	2.16
present, $\lambda = 0.023$	0.93	0.36	0.43	43.5°	2.06
$Re_D = 40$					
Fornberg [11]	2.24	–	–	55.6°	1.50
Dennis and Chang [8]	2.35	–	–	53.8°	1.52
Coutanceau and Bouard [6]*	2.13	0.76	0.59	53.8°	–
Tritton [33]*	–	–	–	–	1.59
present, $\lambda = 0.056$	2.23	0.71	0.59	53.4°	1.61
present, $\lambda = 0.023$	2.28	0.72	0.60	53.6°	1.54

See Fig. 12 for dimension nomenclature. Note: (1) (*) denotes experimental results. (2) The results of Coutanceau [6] cited here are for $\lambda = 0$, obtained via extrapolation.

leads to the "noise" seen in plots of their results near the immersed boundary. Comparisons, similar to those to be presented in this section, with published literature made by the authors show relatively good agreement, despite the lower accuracy. Like the backward-facing step case in Terzi et al. [32], the parameters selected for study in the circular cylinder validation may not require high near-wall accuracy to be faithfully reproduced.

Numerical parameters used in the computations described in this section are given in Table 5. At the inflow, a uniform flow of $\psi(x_1,y) = y$ with $\omega = 0$ is specified. The upper and lower free-stream boundaries

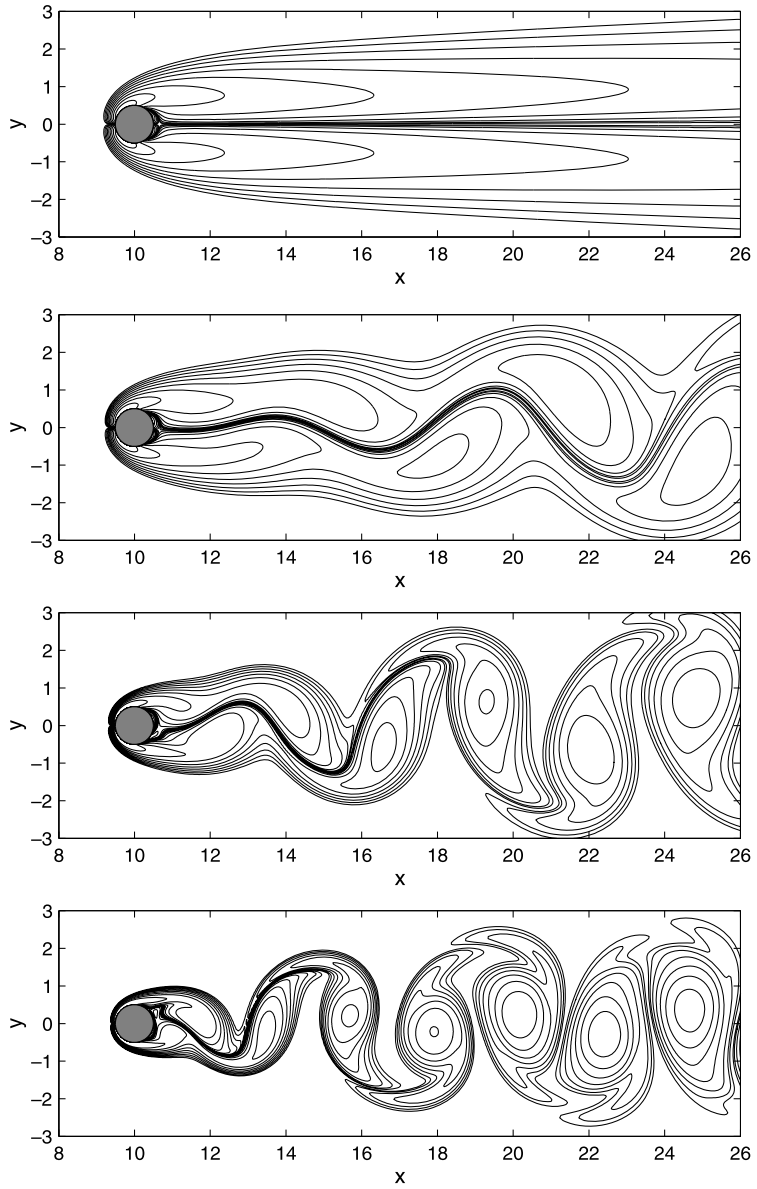

Fig. 11. Uniform flow past circular cylinders for, top to bottom, $Re_D = 40$, 50 (unsteady), 100, and 200, $\lambda = 0.056$. Exponentially spaced contours of vorticity ω.

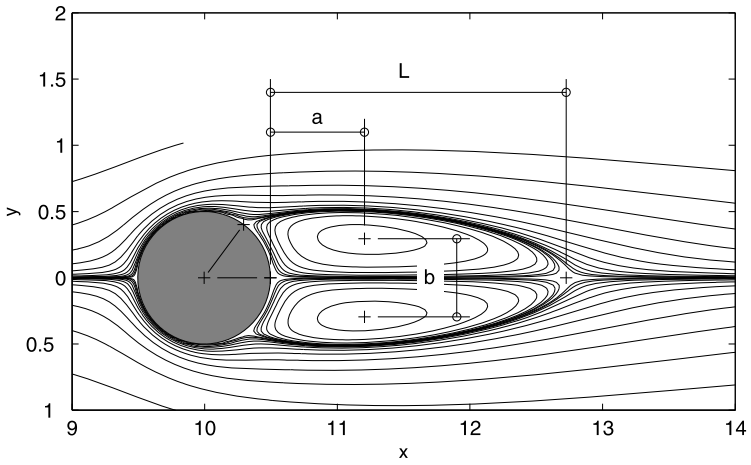

Fig. 12. Nomenclature used in Table 6.

are specified as shear-free walls, $\psi(x,y_1) = y_1$, $\psi(x,y_2) = y_2$, and $\omega = 0$. The location of these boundaries can have a significant impact on the calculation if not placed far enough away. The parameter λ, the ratio of the cylinder diameter D to the domain height $H = y_2 - y_1$, is introduced as a measure and made small enough to minimize the influence of the free stream boundaries. Grid stretching in the y-direction allows this to be accomplished with relatively few points. At the outflow, a buffer domain technique [21] is used to ramp the vorticity down to zero. A uniform flow at the outflow, $\psi(x_2,y) = y$, can then be specified.

Results for the steady regime of Reynolds numbers $Re_D = 20$, 40, and 50 were obtained, a typical example being given in Fig. 10. The pair of attached, steady, symmetric vorticies behind the cylinder are found to grow in length as Re_D increases, and the vorticity generated at the cylinder surface is less able to penetrate the oncoming free-stream. Table 6 presents a comparison of the present results with those published in the literature. For all quantities of interest, excellent agreement is found within the scatter of the data (see Fig. 12).

Table 7
Unsteady flow past a circular cylinder: period τ, Strouhal number St, drag coefficient C_D, and lift coefficient C_L

	$St = 1/\tau$	C_D	C_L
$Re_D = 100$			
Berger and Wille [2]*	0.16–0.17	–	–
Liu et al. [18]	0.165	1.35 ± 0.012	± 0.339
present, $\lambda = 0.056$‡	0.169	1.38 ± 0.010	± 0.337
present, $\lambda = 0.023$	0.166	1.34 ± 0.009	± 0.333
$Re_D = 200$			
Berger and Wille [2]*	0.18–0.19	–	–
Belov [1]	0.193	1.19 ± 0.042	± 0.64
Rogers, Kwak†	0.185	1.23 ± 0.050	± 0.65
Miyake et al.†	0.196	1.34 ± 0.043	± 0.67
Liu et al. [18]	0.192	1.31 ± 0.049	± 0.69
present, $\lambda = 0.056$ ‡	0.199	1.37 ± 0.046	± 0.70
present, $\lambda = 0.023$	0.197	1.34 ± 0.044	± 0.69

Comparison of present results with results published in the literature. Note: (1) (*) denotes experimental results. (2) † in Belov [1]. (3) ‡ approximate (see text). (4) St determined from time variation of C_L.

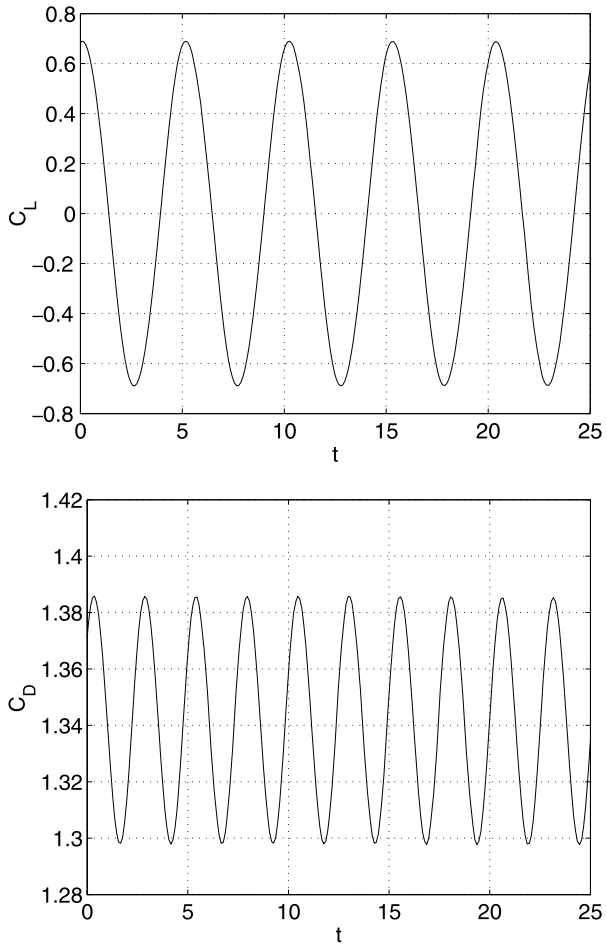

Fig. 13. Lift coefficient C_L, top, and drag coefficient C_D, bottom, versus time t for uniform flow past a circular cylinder, $Re_D = 200$, $\lambda = 0.023$.

As noted above, the case $Re_D = 50$ lies at the upper range of the transition regime between steady and unsteady flow. This case was allowed to run for a considerably long time, but remained steady for the course of the calculation. Given enough time, the calculation may have eventually become unsteady on its own due to round-off or other sources of numerical noise. Instead, a small pulse was introduced into the recirculation zone behind the cylinder, and the calculation continued. As seen in Fig. 11, the flow becomes unsteady with the formation of a vortex street behind the cylinder. By predicting a steady/unsteady transition near $Re_D = 50$, further proof has been obtained that the immersed boundary code is capable of accurately modelling the physics of the flow.

Flows with higher Reynolds numbers are also shown in Fig. 11. Here the flow becomes unsteady on its own, that is, without requiring any external disturbance to trigger the instability. The computations are started with an inviscid solution as initial condition, which is equivalent to impulsively starting the cylinder from rest. A symmetric recirculating eddy then forms behind the cylinder, and grows in length until the point in time at which unsteady vortex shedding of frequency f (or period τ) sets in. Table 7 compares the Strouhal number $St = f D/U_\infty$, lift coefficient C_L, and drag coefficient C_D obtained in the present study with results

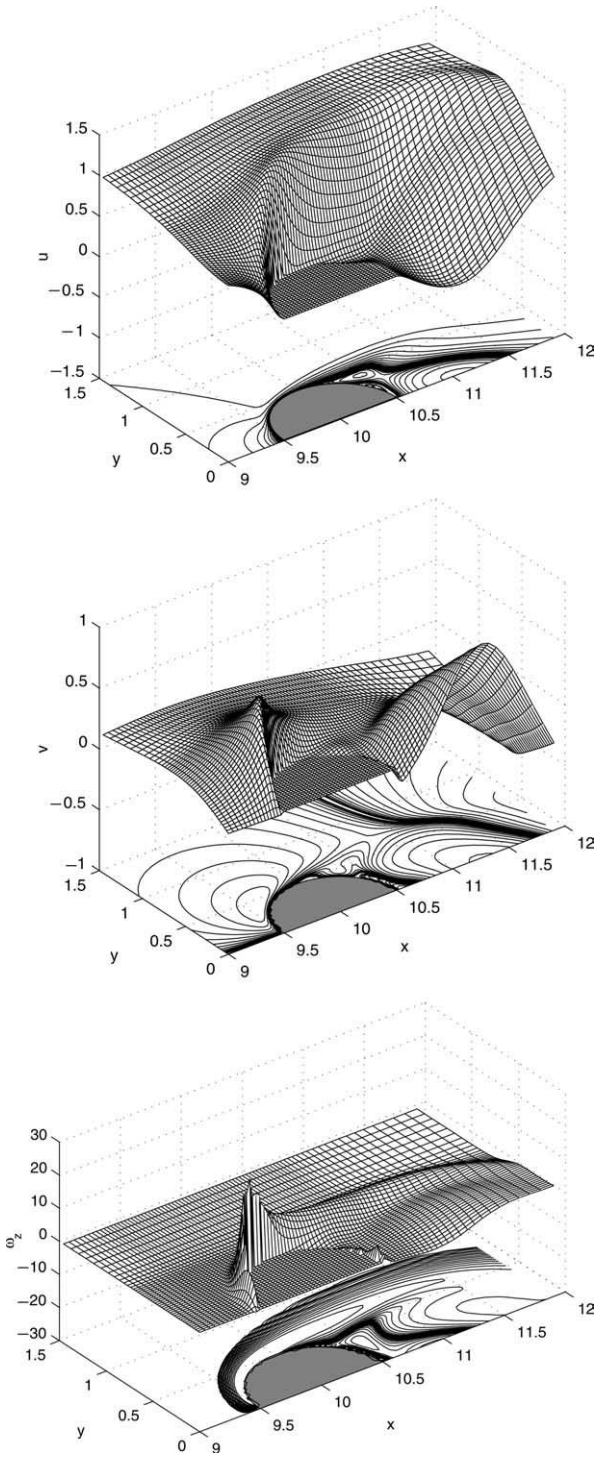

Fig. 14. Close-up view of, top to bottom, u, v, and ω in the vicinity of the cylinder surface for $Re_D = 200$, $\lambda = 0.056$. Exponentially spaced contour levels are given below the mesh plot, for which every second grid-point is shown.

Table 8
Parameters for the TS-wave computation

Parameter	Value
Physical parameters	
Re_L	10^5
Domain size	
Inflow x_1	0.4 ($Re_{\delta_1} = 344$)
Outflow x_2	5.326 ($Re_{\delta_1} = 1256$)
Free stream y_2	0.4 ($\approx 11\delta$)
Grid size	
N_x	541
N_y	117
Δx_{min}	7.992×10^{-3} ($\lambda_{TS} \approx 21\Delta x_{min}$)
Δy_{min}	1.566×10^{-4}
Forcing	
Range in x	$0.8375 \rightarrow 0.9225$
Frequency f	1336.8 Hz ($F = 1.4 \times 10^{-4}$)
Miscellaneous	
Δt	1.122×10^{-3}
Time integration	fourth order R–K
Buffer start x_B	4.554 ($Re_{\delta_1} = 1161$)

published in the literature. As for the steady flows, good agreement is found. For domains with $\lambda = 0.056$, the drag coefficient was found to exhibit behavior that was somewhat irregular. The deviation was more pronounced for lower Reynolds numbers. The lift coefficient, however, was almost unaffected. Where irregularities occurred, data values cited have been labelled approximate, and were obtained as the mean and first harmonic of a Fourier analysis of the time series. For domains with $\lambda = 0.023$, this irregularity disappeared, and both the lift and drag coefficients exhibited regular sinusoidal behavior, as in Fig. 13.

Close-up views of the solution in the vicinity of the immersed boundary are shown in Fig. 14 for $Re_D = 200$. The solutions are presented as mesh plots so that their behavior near the immersed boundaries can be seen. The sharp interface near the immersed boundary is evident, especially for the vorticity, which is discontinuous across the immersed boundary.

5.2. Tollmien–Schlichting waves in a Blasius boundary layer

In this section, the application of the immersed boundary method to computing Tollmien–Schlichting waves in a zero pressure-gradient, flat-plate boundary-layer is investigated. For comparison, the results are obtained for both a body-fitted code (the temporally and spatially fourth-order accurate code `nst2d` of Meitz and Fasel [21]) and the present immersed boundary scheme. This test case was selected because, in contrast to the circular cylinder case described above, near-wall accuracy, in particular, the phase relation between wave velocity components close to the wall, is critically important if correct results are to be obtained [32].

The setup is shown in Fig. 3 where the immersed flat plate wall runs from the inflow to the outflow boundary. The immersed wall does not lie on the computational grid, but rather half-way between grid points $j = 6$ and $j = 7$ in the y-direction. Computational parameters are given in Table 8. The free-stream is placed roughly 11 boundary-layer thicknesses away from the wall, and the boundary condition here is taken as $\psi_y = 1$. The outflow condition is set, somewhat less than optimally, to $\psi_{xx} = 0$, so that one may

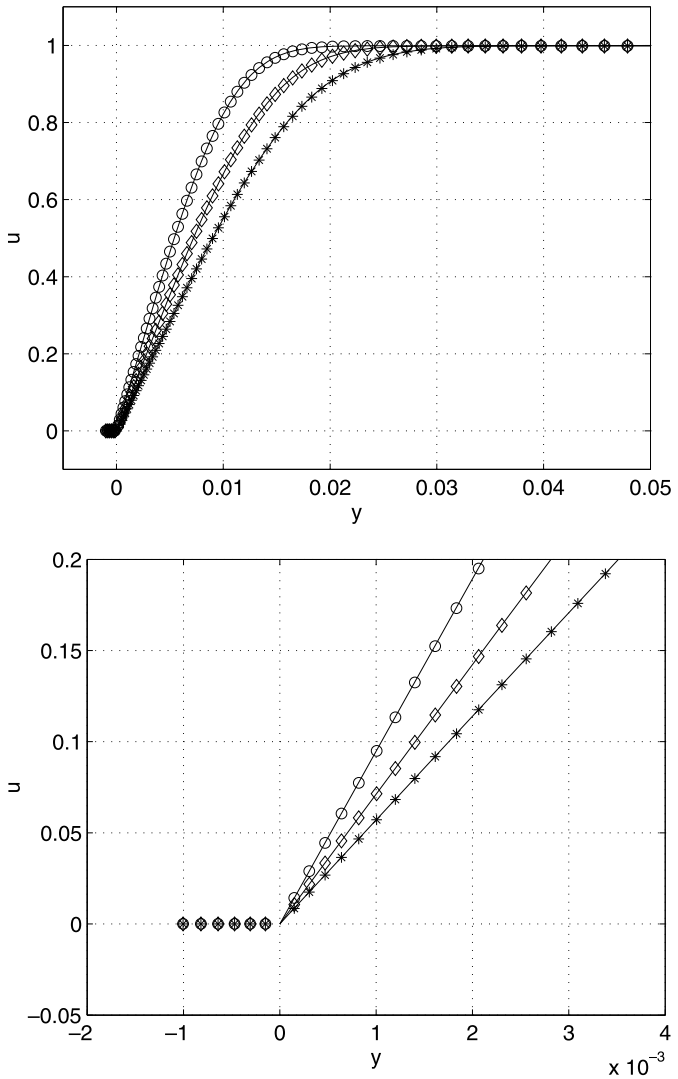

Fig. 15. Comparison of u-velocity profiles at several downstream locations: $x = 1.214$ ($R_{\delta_1} = 600$) (○), $x = 2.157$ ($R_{\delta_1} = 800$) (◇), and $x = 3.372$ ($R_{\delta_1} = 1000$) (∗). Symbols indicate values computed using an immersed wall located at $y = 0$, and solid lines results from the code `nst2d` of Meitz and Fasel [21]. Lower figure is a zoom-in of the upper figure near $y = 0$.

obtain ψ by solving the ODE $\psi_{yy} = \omega$. This is equivalent to setting $v_x = 0$, an approximation that is only valid for $Re_x \gg 1$. However, the outflow is placed far enough away so that the adverse upstream-influence is minimal.

As a first step, the undisturbed flat-plate boundary layer is computed, starting with the Blasius similarity solution as initial condition and converging to a steady-state. Figs. 15 and 16 compare the resulting u and ω profiles at various x-locations along the immersed wall with those computed with the body-fitted code. Excellent agreement between the body-fitted and immersed boundary code is found. In particular, the sharp interface in ω at the wall is captured.

Next, small, time-harmonic disturbances are introduced into the steady boundary layer at a given upstream location near the inflow. If the disturbances introduced are small enough, their behavior is, to a

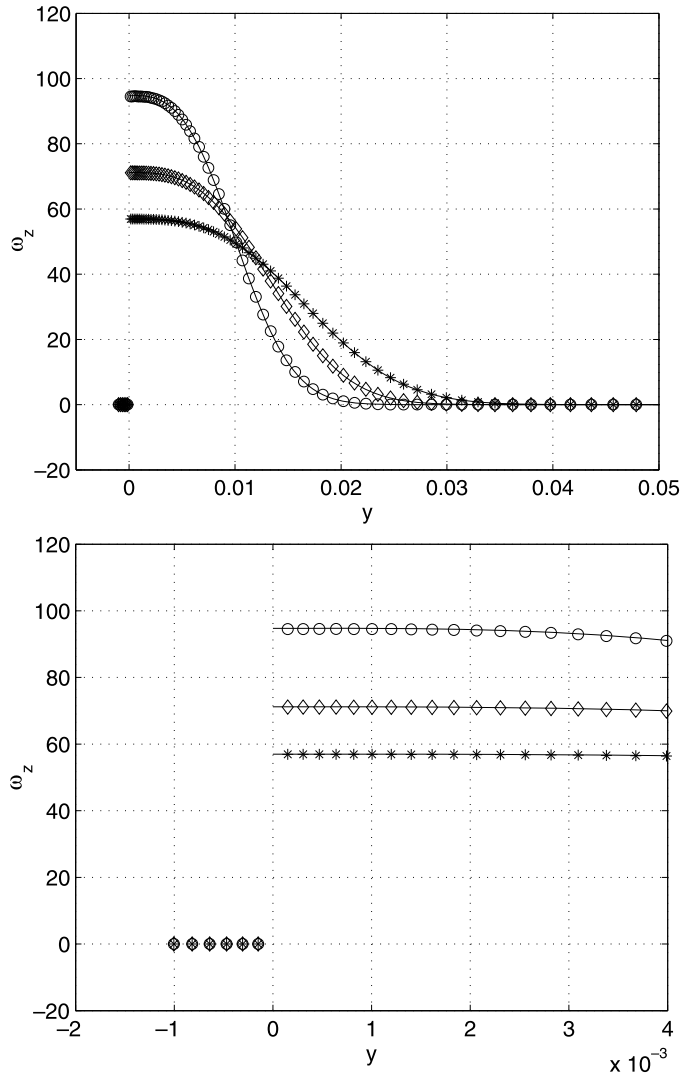

Fig. 16. Comparison of ω-vorticity profiles at several downstream locations: $x = 1.214$ ($R_{\delta_1} = 600$) (○), $x = 2.157$ ($R_{\delta_1} = 800$) (◇), and $x = 3.372$ ($R_{\delta_1} = 1000$) (∗). Symbols indicate values computed using an immersed wall located at $y = 0$, and solid lines results from the code `nst2d` of Meitz and Fasel [21]. Lower figure is a zoom-in of the upper figure near $y = 0$.

good approximation, "linear" (exponential growth or decay), and the resulting velocity and vorticity perturbations can be compared with the results from linear stability theory (LST) [29,20]. LST predicts a range of unstable frequencies for which wave-like perturbations of a laminar boundary-layer will exponentially grow as they are convected downstream. These wave-like perturbations are known as Tollmien–Schlichting waves (TS-waves).

The results of the time harmonic forcing are shown in Figs. 17–20. Figs. 17 and 18 depict the amplitude envelopes of the u and ω disturbance quantities. Assuming linear behavior, these quantities are obtained as the first harmonic of a Fourier series with the fundamental frequency equal to the forcing frequency. Numerically, these values are computed with an FFT applied to the data time series. The agreement

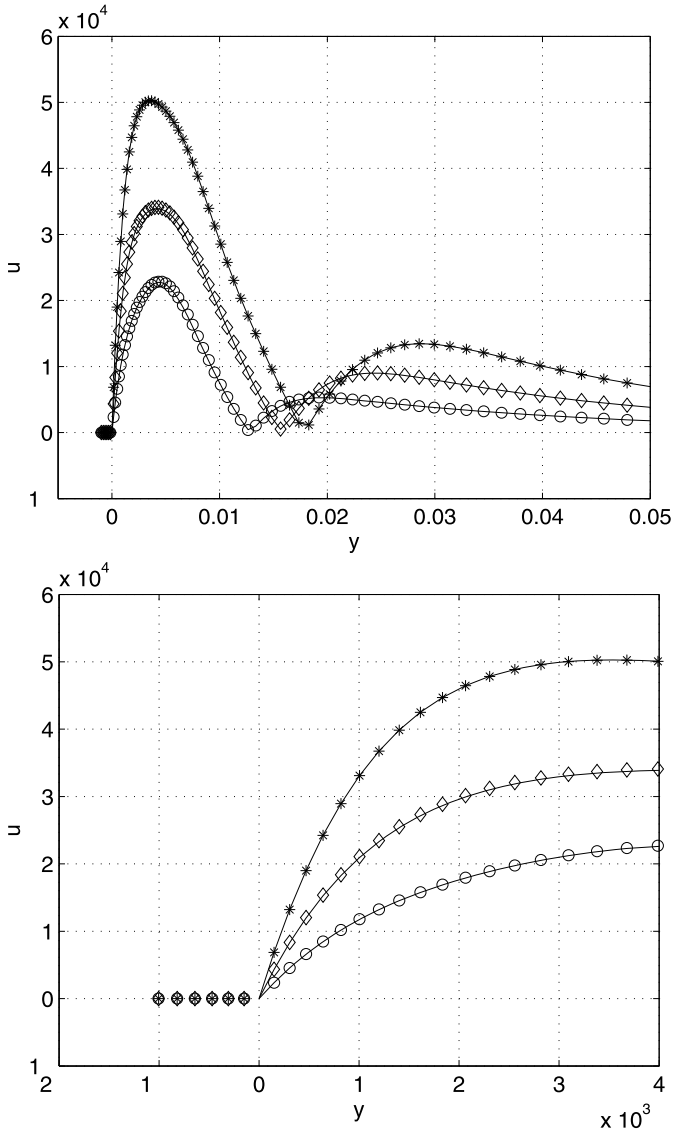

Fig. 17. Comparison of disturbance u-velocity profiles (fundamental) at several downstream locations: $x = 1.214$ ($R_{\delta_1} = 600$) (○), $x = 2.157$ ($R_{\delta_1} = 800$) (◇), and $x = 3.372$ ($R_{\delta_1} = 1000$) (∗). Symbols indicate values computed using an immersed wall located at $y = 0$, and solid lines results from the code `nst2d` of Meitz and Fasel [21]. Lower figure is a zoom-in of the upper figure near $y = 0$.

between the body-fitted results and the immersed boundary results is again excellent. Once more it is noted that the sharp interface in ω is captured by the immersed boundary scheme. A comparison of the phase distribution for u is shown in Fig. 19 for reference, where excellent agreement is also found. Finally, Fig. 20 shows the spatial development of the inner maximum of the disturbance u-velocity amplitude envelope, along with the prediction of LST. The results agree favorably, and in addition, agree well with those of Fasel and Konzelmann [10]. The deviation between the numerical solutions and LST is due to simplifying assumptions used in the derivation of LST (specifically, non-parallel effects [10]).

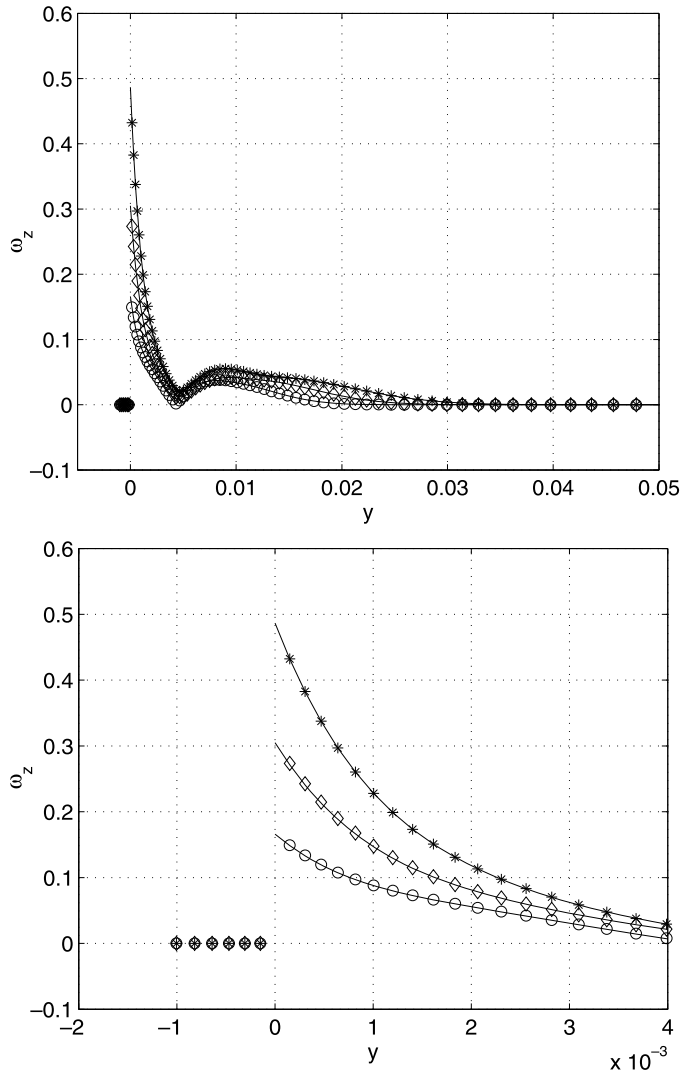

Fig. 18. Comparison of disturbance ω-vorticity profiles (fundamental) at several downstream locations: $x = 1.214$ ($R_{\delta_1} = 600$) (○), $x = 2.157$ ($R_{\delta_1} = 800$) (◇), and $x = 3.372$ ($R_{\delta_1} = 1000$) (∗). Symbols indicate values computed using an immersed wall located at $y = 0$, and solid lines results from the code `nst2d` of Meitz and Fasel [21]. Lower figure is a zoom-in of the upper figure near $y = 0$.

5.3. Additional examples

Two additional examples are shown in Figs. 21 and 22 to illustrate the flexibility of the immersed boundary code with respect to geometry. In the first figure, flow over a curved wall is shown. A zero pressure-gradient boundary layer solution and a no slip wall are specified at the inflow and free stream boundaries, respectively. The immersed boundary runs from the inflow to the outflow, similar to the Tollmien–Schlichting case described above. In the second figure, the flow past a 6:1 ellipse is shown. The boundary conditions used are the same as for the circular cylinder case.

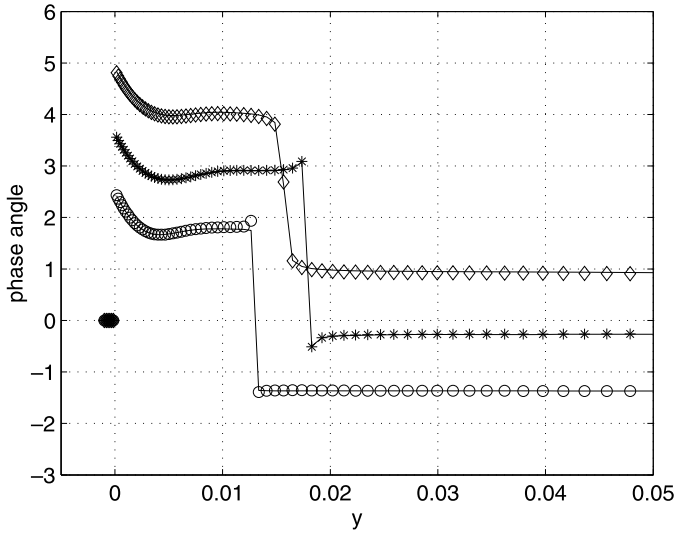

Fig. 19. Comparison of disturbance u-velocity phase profiles (fundamental) at several downstream locations: $x = 1.214$ ($R_{\delta_1} = 600$) (○), $x = 2.157$ ($R_{\delta_1} = 800$) (◇), and $x = 3.372$ ($R_{\delta_1} = 1000$) (∗). Symbols indicate values computed using an immersed wall located at $y = 0$, and solid lines results from the code `nst2d` of Meitz and Fasel [21].

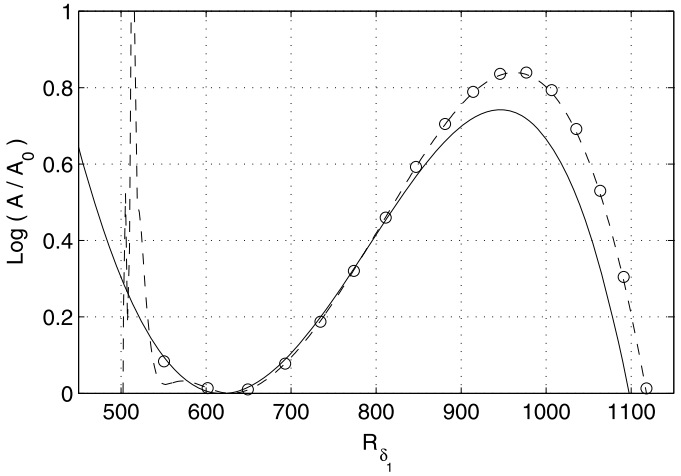

Fig. 20. Comparison of the growth rates of a 2D TS-wave ($A =$ inner maximum of $u_{(1,0)}$) for frequency $F = 1.4 \times 10^{-4}$: LST (—), an immersed wall (– – –), and results from the code `nst2d` of Meitz and Fasel [21] (○).

The present immersed boundary technique allows for a wide range of complex geometries to be studied within a Cartesian grid framework. There are some geometries, however, that will require the immersed boundary method, as described here, to be modified. For example, geometries that possess cusps will have jump corrections that require more points in Ω^+ than are available. In this case, one would have to resort to increasing the resolution, perhaps using AMR, or to using a 2D finite-difference scheme to compute jump corrections.

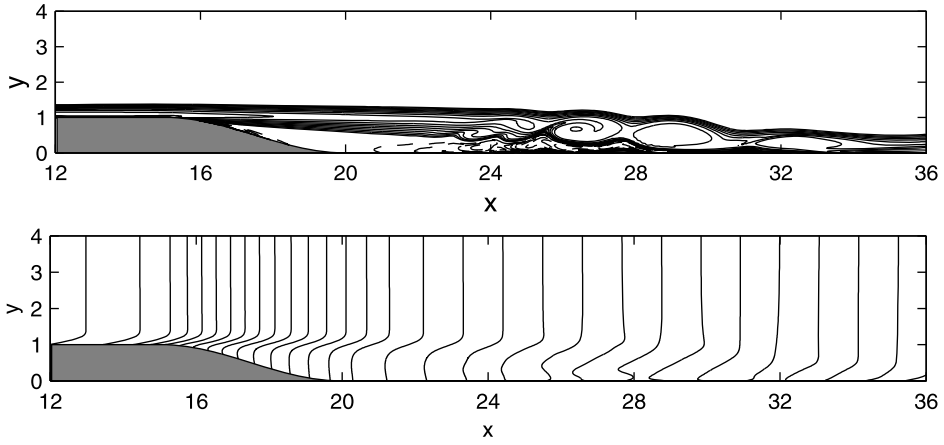

Fig. 21. Flow over a curved wall, $Re_H = 2500$. Shown are instantaneous values of spanwise vorticity ω and streamwise velocity profiles u.

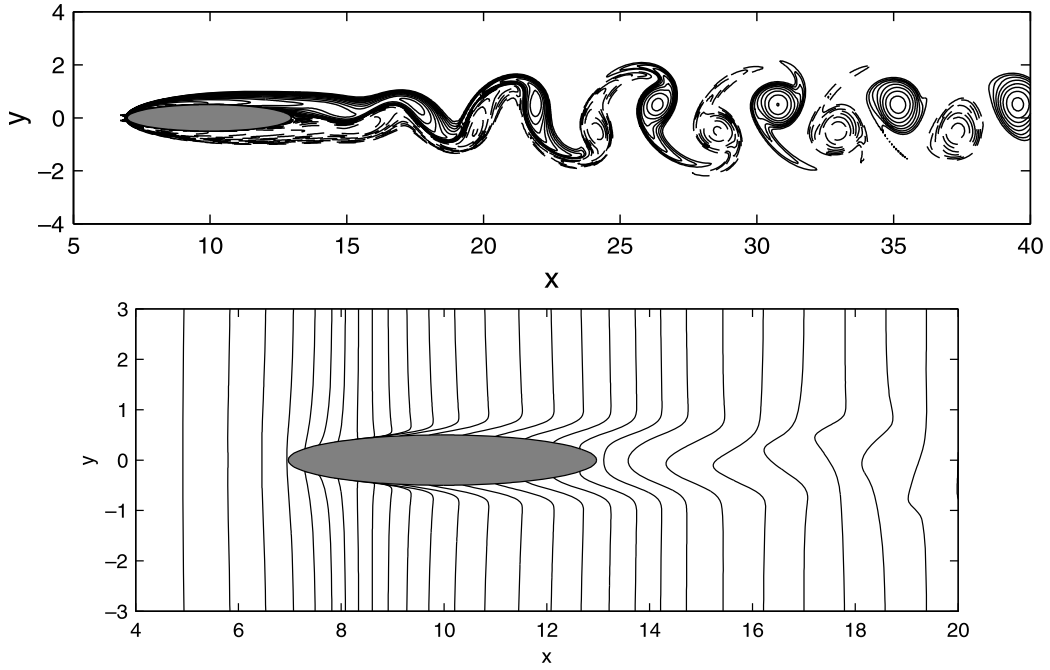

Fig. 22. Flow over a 6:1 ellipse at zero angle of attack, Re_C=2400 based on chord. Shown are instantaneous contours of spanwise vorticity ω and streamwise velocity profiles u.

6. Conclusions

An immersed interface method for accurately solving incompressible flow problems has been presented. By separately studying the convection–diffusion equation and Poisson equation solutions as model equations, the accuracy of the numerical method has been shown to be fourth order. For the 2D, high-Reynolds

number, incompressible flow test-cases presented, very good agreement was found with results published in the literature. The extension of the method to 3D should present no particular difficulties beyond the increased complexity required to determine intersections of the immersed boundary with the underlying Cartesian grid, and the corresponding jump corrected finite-difference stencils. In addition, a suitable formulation of the Navier–Stokes equations will have to be selected. The stream function–vorticity formulation used in the present investigation is only valid in two dimensions.

Acknowledgment

Funding for this research from the Office of Naval Research (ONR) under Grant No. N00014-01-1-09 (Dr. Ronald Joslin, program manager) is gratefully acknowledged.

Appendix A. Derivation of finite difference schemes

All of the coefficients, e.g. L_i and R_i, in the finite difference schemes used have been derived by performing a Taylor series analysis. Each term in the scheme, e.g. $f^{(2)}_{i+1}, f_{i-1}$, is expanded about some point x^*, and the coefficients are used maximize the formal accuracy of the approximation, instead of, say, sacrificing formal accuracy and improving wave resolution accuracy (as discussed in [14]).

Most of the grids used in the present investigation were non-equidistant. This allowed for the more efficient clustering of points in the domain where higher resolution was required, and courser gridding where gradients in the solution were mild. The expressions for the coefficients appearing in the finite difference schemes used are, in general, much more complex functions of the grid points x_i than are their equidistant counterparts. As a result, the systems of equations satisfied by these coefficients, rather than their explicit forms, are presented here.

In fact, these systems are actually solved in the current implementation of the numerical scheme discussed above. Rather than deriving these coefficients analytically, using perhaps symbolic manipulation software, and hard-coding them into the computer program, a much more flexible alternative was used. The finite difference scheme could be defined as a linear combination of the function and its derivatives, i.e. $L(f_i, f'_i, f''_i, \ldots) = 0$, and the stencil specified, and a subroutine would return the coefficients maximizing the formal accuracy of the stencil. Problems due to ill-conditioning of the resulting system of equations were not encountered.

(1) Compact first-order derivatives, $f^{(1)}$: Eqs. (42), (50), (51).

$$L_{i-1}f^{(1)}_{i-1} + L_i f^{(1)}_i + L_{i+1}f^{(1)}_{i+1} = R_{i-1}f_{i-1} + R_i f_i + R_{i+1}f_{i+1}, \tag{90}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & h_{i-1} & 0 & h_{i+1} \\ h_{i-1} & 0 & h_{i+1} & h^2_{i-1}/2! & 0 & h^2_{i+1}/2! \\ h^2_{i-1}/2! & 0 & h^2_{i+1}/2! & h^3_{i-1}/3! & 0 & h^3_{i+1}/3! \\ h^3_{i-1}/3! & 0 & h^3_{i+1}/3! & h^4_{i-1}/4! & 0 & h^4_{i+1}/4! \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} L_{i-1} \\ L_i \\ L_{i+1} \\ -R_{i-1} \\ -R_i \\ -R_{i+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \tag{91}$$

where $h_{i-1} = x_{i-1} - x_i$, and $h_{i+1} = x_{i+1} - x_i$.

(2) Compact second-order derivatives, $f^{(2)}$: Eqs. (20), (25), (43), (59), (60), (70), (71).

$$L_{i-1}f^{(2)}_{i-1} + L_i f^{(2)}_i + L_{i+1}f^{(2)}_{i+1} = R_{i-1}f_{i-1} + R_i f_i + R_{i+1}f_{i+1}, \tag{92}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & h_{i-1} & 0 & h_{i+1} \\ 1 & 1 & 1 & h^2_{i-1}/2! & 0 & h^2_{i+1}/2! \\ h_{i-1} & 0 & h_{i+1} & h^3_{i-1}/3! & 0 & h^3_{i+1}/3! \\ h^2_{i-1}/2! & 0 & h^2_{i+1}/2! & h^4_{i-1}/4! & 0 & h^4_{i+1}/4! \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} L_{i-1} \\ L_i \\ L_{i+1} \\ -R_{i-1} \\ -R_i \\ -R_{i+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \tag{93}$$

where $h_{i-1} = x_{i-1} - x_i$, and $h_{i+1} = x_{i+1} - x_i$.

(3) Explicit finite differences for nth derivative $f^{(n)}$ at $x = x_\alpha$: Eqs. (26), (29), (30), (53).

$$f^{(n)}_\alpha = c_\alpha f_\alpha + c_i f_i + c_{i+1}f_{i+1} + c_{i+2}f_{i+2} + c_{i+3}f_{i+3} + c_{i+4}f_{i+4}, \tag{94}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & h_i & h_{i+1} & h_{i+2} & h_{i+3} & h_{i+4} \\ 0 & h^2_i & h^2_{i+1} & h^2_{i+2} & h^2_{i+3} & h^2_{i+4} \\ 0 & h^3_i & h^3_{i+1} & h^3_{i+2} & h^3_{i+3} & h^3_{i+4} \\ 0 & h^4_i & h^4_{i+1} & h^4_{i+2} & h^4_{i+3} & h^4_{i+4} \\ 0 & h^5_i & h^5_{i+1} & h^5_{i+2} & h^5_{i+3} & h^5_{i+4} \end{bmatrix} \begin{bmatrix} c_\alpha \\ c_i \\ c_{i+1} \\ c_{i+2} \\ c_{i+3} \\ c_{i+4} \end{bmatrix} = \begin{bmatrix} 1\,\delta_{n0} \\ 1\,\delta_{n1} \\ 2!\,\delta_{n2} \\ 3!\,\delta_{n3} \\ 4!\,\delta_{n4} \\ 5!\,\delta_{n5} \end{bmatrix}, \tag{95}$$

where $h_i = x_i - x_\alpha$, and δ_{ij} is the Kronecker delta function

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{96}$$

References

[1] A. Belov, L. Martinelli, A. Jameson, A new implicit algorithm with multigrid for unsteady incompressible flow calculations, AIAA Paper 95-0049.

[2] E. Berger, R. Wille, Periodic flow phenomena, Ann. Rev. Fluid Mech. 4 (1972) 313–340.

[3] R.P. Beyer, R.J. LeVeque, Analysis of a one-dimensional model for the immersed boundary method, SIAM J. Numer. Anal. 29 (1992) 332–364.

[4] E.F.F. Botta, F.W. Wubs, The convergence behaviour of iterative methods on severly stretched grids, Int. J. Numer. Meth. Eng. 36 (1993) 3333–3350.

[5] D. Calhoun, A cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions, J. Comput. Phys. 176 (2) (2002) 231–275.

[6] M. Coutanceau, R. Bouard, Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. steady flow, J. Fluid Mech. 79 (1977) 231–256.

[7] O. Daube, Resolution of the 2D Navier–Stokes equations in velocity-vorticity form by means of an influence matrix technique, J. Comput. Phys. 103 (1992) 402–414.

[8] S.C.R. Dennis, G.-Z. Chang, Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100, J. Fluid Mech. 42 (1970) 471–489.

[9] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed boundary finite-difference methods for three-dimensional complex flow simulations, J. Comput. Phys. 161 (2000) 35–60.

[10] H.F. Fasel, U. Konzelmann, Non-parallel stability of a flat-plate boundary layer using the complete Navier–Stokes equations, J. Fluid Mech. 221 (1990) 311–347.
[11] B. Fornberg, A numerical study of steady viscous .ow past a circular cylinder, J. Fluid Mech. 98 (1980) 819–855.
[12] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, J. Comput. Phys. 105 (1993) 354–366.
[13] M. Kloker, A robust high-resolution split-type compact FD scheme for spatial direct numerical simulation of boundary-layer transition, Appl. Sci. Res. 59 (1998) 353–377.
[14] S.K. Lele, Compact finite difference schemes with spectral-like resolution, J. Comput. Phys. 103 (1992) 16–42.
[15] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coeffcients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019–1044.
[16] Z. Li, C. Wang, A fast finite difference method for solving Navier–Stokes equations on irregular domains, Commun. Math. Sci. 1 (1) (2003) 180–196.
[17] Z. Li, M.-C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2001) 822–842.
[18] C. Liu, X. Zheng, C.H. Sung, Preconditioned multigrid methods for unsteady incompressible flows, J. Comput. Phys. 139 (1998) 35–57.
[19] M.N. Linnick, Investigation of actuators for use in active flow control, Master's Thesis, The University of Arizona, 1999.
[20] L.M. Mack, Boundary-layer linear stability theory, AGARD Report 709, 1984.
[21] H. Meitz, H.F. Fasel, A compact-difference scheme for the Navier–Stokes equations in vorticity-velocity formulation, J. Comput. Phys. 157 (2000) 371–403.
[22] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (1972) 252–271.
[23] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[24] C.S. Peskin, D.M. McQueen, Modeling prosthetic heart valves for numerical analysis of blood flow in the heart, J. Comput. Phys. 37 (1980) 113–132.
[25] C.S. Peskin, D.M. McQueen, A three-dimensional computational method for blood flow in the heart I. Immersed elastic fibers in a viscous incompressible fluid, J. Comput. Phys. 81 (1989) 372–405.
[26] L. Quartapelle, Numerical Solution of the Incompressible Navier–Stokes Equations, Birkhäuser, 1993.
[27] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (1999) 509–534.
[28] E.M. Saiki, S. Biringen, Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method, J. Comput. Phys. 123 (1996) 450–465.
[29] H. Schlichting, Boundary-Layer Theory, seventh ed., McGraw-Hill, New York, 1979.
[30] L. Sirovich, Steady gasdynamic flows, Phys. Fluids 11 (7) (1968) 1424–1439.
[31] P. Sonneveld, P. Wesseling, P.M. de Zeeuw, Multigrid and Conjugate Gradient Methods as Convergence Acceleration Techniques, Clarendon Press, Oxford, 1985, pp. 117–167.
[32] D.A.v. Terzi, M.N. Linnick, J. Seidel, H.F. Fasel, Immersed boundary techniques for high-order finite-difference methods, AIAA Paper 2001-2918.
[33] D.J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds numbers, J. Fluid Mech. 6 (1959) 547–567.
[34] J.A. Viecelli, A method for including arbitrary external boundaries in the MAC incompressible fluid computing technique, J. Comput. Phys. 4 (1969) 543–551.
[35] J.A. Viecelli, A computing method for incompressible flow bounded by moving walls, J. Comput. Phys. 8 (1971) 119–143.
[36] J. Waldén, On the approximation of singular source terms in differential equations, Numer. Meth. Partial Differential Equations 15 (4) (1999) 503–520.
[37] A. Wiegmann, K. Bube, The explicit-jump immersed interface method: finite difference methods for pdes with piecewise smooth solutions, SIAM J. Numer. Anal. 37 (3) (2000) 827–862.
[38] P.M. de Zeeuw, Acceleration of iterative methods by coarse grid corrections, Ph.D. Thesis, University of Amsterdam, 1997.