Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 225 (2007) 528–553
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A sharp interface immersed boundary method for compressible viscous flows

R. Ghias, R. Mittal *, H. Dong

Department of Mechanical and Aerospace Engineering, The George Washington University, Washington, DC 20052, United States

Received 6 February 2006; received in revised form 15 December 2006; accepted 15 December 2006
Available online 23 December 2006

Abstract

An immersed boundary method for computing viscous, subsonic compressible flows with complex shaped stationary immersed boundaries is presented. The method employs a ghost-cell technique for imposing the boundary conditions on the immersed boundaries. The current approach leads to a sharp representation of the immersed boundaries, a property that is especially useful for flow simulations at high Reynolds numbers. Another unique feature of the method is that it can be applied on Cartesian as well as generalized body non-conformal curvilinear meshes. A mixed second-order central difference-QUICK scheme is used which allows a high degree of control over the numerical damping. A bilinear interpolation scheme used in conjunction with the ghost-cell approach results in second-order global as well as local spatial accuracy. The solver is parallelized for distributed memory platforms using domain decomposition and message passing interface (MPI) and salient features of the parallel algorithm are presented. The accuracy, fidelity and efficiency of the solver are examined by simulating flow past circular cylinders and airfoils and comparing against experimental data and other established results. Finally, we present results from a simulation of wing-tip flow at a relatively high Reynolds number in order to demonstrate the ability of the solver to model complex, non-canonical three-dimensional flows.

Keywords: Computational fluid dynamics; Immersed boundary method; Ghost-cell; Non-conformal grid

1. Introduction

The conventional structured grid approach to simulating flows with complex immersed boundaries is to discretize the governing equations on a curvilinear grid that conforms to the boundaries. Since the boundary itself becomes a grid line, the imposition of boundary conditions is greatly simplified and the solver can be easily designed to maintain adequate accuracy and conservation properties. However, depending on the geometrical complexity of the immersed boundary, grid generation and grid quality can be major issues and one has to resort to multi-block or other such approaches in order to handle anything but the simplest geometries. For complex boundaries, unstructured grid methods offer greater flexibility and are being widely used.

* Corresponding author. Tel.: +1 202 994 9394.
E-mail address: mittal@gwu.edu (R. Mittal).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.12.007

However, due to the inapplicability of powerful line/block iterative and geometric multigrid techniques to unstructured grids, these methods are in general slower on a per-grid-point basis than structured grid methods. A different approach that retains most of the favorable properties of structured grids but also provides a high level of flexibility in handling highly complex geometry is the so-called immersed boundary method (IBM). This method, is usually employed in conjunction with a body non-conforming Cartesian grid. Thus, grid generation is greatly simplified and these methods can tackle flows with complex stationary or moving boundaries with relative ease [1,2]. However, since the immersed boundary can cut through the underlying mesh in an arbitrary manner, the main challenge is to treat the boundary in a way that does not adversely impact the accuracy and conservation property of the underlying solver [7]. This is especially critical for viscous flows where inadequate resolution of boundary layers, which form on the immersed boundaries, can reduce the fidelity of the numerical solution.

Immersed boundary methods can broadly be categorized under two categories [2]; first are methods that employ "continuous forcing" wherein a forcing term is added to the continuous Navier–Stokes equations before they are discretized. The original method of Peskin [1] as well as other methods such as those of Goldstein et al. [3,4] and Saiki and Biringen [5] fall in this category. The second category consists of methods that employ "discrete forcing" where the forcing is either explicitly or implicitly applied to the discretized Navier–Stokes equations. These include methods of Udaykumar et al. [6], Ye et al. [7], Fadlun et al. [8], Balaras [9], You et al. [10], Kim et al. [11], Gibou et al. [12], Almgren et al. [13] and others. The key advantage of the first category of methods is that they are formulated relatively independent of the spatial discretization and can therefore be implemented into an existing Navier–Stokes solver with relative ease. This is not the case with the methods in the second category since the forcing scheme is very much dependent on the spatial discretization scheme. The advantage of the second category of methods however is that for certain formulations, they allow for a "sharp" representation of the immersed boundary. In contrast, the first category of methods produce a "diffuse" boundary in that the boundary condition on the immersed boundary is not precisely satisfied at its actual location but within a localized region around the boundary.

One issue faced with Cartesian grid based immersed boundary methods is that the grid size can grow much more rapidly with Reynolds number than a corresponding structured curvilinear body-conformal mesh [2]. Generalization of the Cartesian grid approach to non-body-conformal curvilinear grids can therefore significantly enhance the power of the Cartesian grid approach. On the other hand, by not requiring the grid to conform precisely to the boundary, the grid generation requirements for complex geometries can be eased significantly, and the use of a curvilinear mesh can allow more control over the grid resolution in localized regions such as boundary layers. Consider for example the simulation of the tip-flow of a wing. Such flows are typically simulated on relatively complex multi-block C–H type of grids [14], which conform to the shape of the wing. However, the use of a body non-conformal curvilinear grid can greatly simplify the grid topology and lead to a viable simulation approach. This has been demonstrated by You et al. [10] who used an immersed boundary method in conjunction with an incompressible flow solver to simulate high Reynolds number tip-clearance flow of an axial turbomachine. Thus, for high Reynolds number flows, it is worthwhile to develop immersed boundary type methods that can be employed in conjunction with curvilinear structured grids. Such methods allow us to provide localized resolution to the boundary layers on the immersed boundary while still retaining the flexibility of a body non-conformal grid.

In the current paper, we describe a finite-difference based method that allows us to simulate viscous, subsonic compressible flows with complex immersed boundaries on Cartesian or curvilinear grids that do not conform to the immersed boundary. Although a number of different immersed boundary methods have been developed for incompressible flows (see Mittal and Iaccarino [2] for an extensive list of many methods developed to date), there exists, to the author's knowledge, no implementation of this approach for compressible, viscous flows. As will become apparent in the following sections, the differences in the boundary conditions between incompressible and compressible flows as well as the spatial discretization schemes used, requires some additional considerations when developing an IBM method for compressible flows and these are described in detail in the current paper. Furthermore, although the current solver is limited to subsonic flows, inclusion of appropriate shock capturing schemes would in principle allow for the extension of the current method to supersonic flows.

It should be pointed out that finite-volume based Cartesian grid methods have been developed for inviscid compressible flows by Collella and co-workers [15,16], and Legay et al. [17] have developed a method for simulating such flows with a body non-conformal finite-element method. A discussion of these methods, especially [15] provides a useful context for the current work. First, all of the above methods have been designed for *inviscid* flows whereas the current method is for viscous flows. This is significant not only because inclusion of viscosity changes the character of the governing equations and consequently has implications for the temporal and spatial discretization, but also because the presence of viscosity leads to the formation of boundary layers as well as phenomenon such as transition and turbulence. Boundary layers are oftentimes the key features in these flows and it is essential to ensure adequate spatial accuracy in these regions. Consequently we have developed here an immersed boundary treatment which is globally and *locally* second-order accurate. In contrast, the method of Collela [15] is globally second-order accurate but locally first-order accurate near the immersed boundary. This is most likely adequate for inviscid flows which do not have boundary layers. Furthermore, in the method of Collella and co-workers [15,16] one has to contend with the "small-cell" stability problem. This problem is due to a combination of two factors: first, in their method, the cells intersected by the immersed boundary can be arbitrarily small and second, the flow velocity in these cells can be quite high since the flow is inviscid. This can lead to very small convective time-scales for these cells and consequently to severe stability constraints, which have to be mitigated through appropriate methods [15]. In contrast, there is no such issue in the current method since, as will be shown in a later section, the size of the cell is unchanged by the presence of the immersed boundary and the interpolation scheme remains well behaved for all relative positions and distances between the immersed boundary and adjoining nodes. An attractive feature of the method of Collela et al. [15] is the use of adaptive mesh refinement (AMR) technique that allow for selective mesh refinement in local regions.

The current method is based on the calculation of the variables on "ghost-cells" inside the body such that the boundary conditions are satisfied precisely on the immersed boundary. This general approach has been described by Majumdar et al. [31] who have also investigated various interpolation schemes within the context of this method. The method also has some similarities to the "ghost-fluid method" of Gibou et al. [12] and this is discussed in detail in Section 2.5 of the paper. There are no *ad hoc* constants introduced in the current procedure and neither is any momentum forcing term employed in any of the fluid cells. Consequently, the method results in a sharp representation of the immersed boundary. Here we focus on describing the salient of the numerical methodology. The solver is validated by simulating two- and three-dimensional flow past circular cylinders and comparing the computed results with established experiments and other numerical simulations. We also verify the spatial and temporal accuracy of the solver through systematic refinement studies. The IBM solver has been parallelized using MPI, and the domain decomposition methodology and parallel performance of the solver are also discussed. Finally, in order to showcase the capability of the method for handling general immersed boundaries, we present some computed results of flow past airfoils and wings.

2. Numerical method

In this section, we begin by describing the governing equations, spatial and temporal discretization, boundary conditions and the method used to solve the discretized equations. This is followed by a detailed discussion of the method used for imposing the boundary condition on the immersed boundaries. Finally, we provide a concise description of the method used to parallelize the solver for distributed memory platforms.

2.1. Governing equations

The governing equations are unsteady, viscous, compressible Navier–Stokes equations written in terms of conservative variables. The continuity, momentum and energy equations are as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_k)}{\partial x_k} = 0 \tag{1}$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_k + p\delta_{ik} - \sigma_{ik})}{\partial x_k} = 0 \tag{2}$$

$$\frac{\partial E_t}{\partial t} + \frac{\partial[u_k(E_t + p) - \sigma_{ik}u_i + Q_k]}{\partial x_k} = 0 \tag{3}$$

where the total energy per unit volume E_t is related to the other variables through the equation-of-state for a perfect gas as follows:

$$E_t = \frac{p}{(\gamma - 1)} + \frac{1}{2}\rho(u_k u_k) \tag{4}$$

Furthermore, the heat flux and stress tensor are given by

$$Q_k = -\frac{\gamma}{(\gamma - 1)RePr}\frac{\partial T}{\partial x_k} \tag{5}$$

and

$$\sigma_{ik} = \frac{1}{Re}\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\right) - \frac{2}{3}\frac{1}{Re}\delta_{ik}\frac{\partial u_j}{\partial x_j} \tag{6}$$

respectively, where Pr and Re are the Prandtl and Reynolds numbers, respectively.

The above set of equations are transformed to a 2D generalized curvilinear coordinate system (y_1, y_2) by applying chain-rule differentiation and the equations are written in the strong conservation law form [18]. Thus a generic convective term for the variable ϕ of the form $\frac{\partial}{\partial x_k}(\rho u_k \phi)$ is transformed to $\frac{1}{J}\frac{\partial}{\partial y_k}(\rho U_k \phi)$ where J is the Jacobian of the transformation given by $|\frac{\partial x_i}{\partial y_k}|$. Furthermore, U_k is a component of the contravariant velocity which is equal to $U_k = u_j\beta^{jk}$ where $\beta^{jk} = \text{Cofactor}[\frac{\partial x_j}{\partial y_k}]$. Similarly, diffusive flux terms of the form $\frac{\partial}{\partial x_k}(\eta\frac{\partial \phi}{\partial x_j})$ where η is the diffusion coefficient, are transformed to

$$\frac{1}{J}\frac{\partial}{\partial y_k}\left(\eta\frac{B^{jk}}{J}\frac{\partial \phi}{\partial y_j}\right). \tag{7}$$

where in the above expression, $B^{jk} = \beta^{mk}\beta^{mj}$ is the metric tensor.

2.2. Spatial discretization

The transformed equations are discretized in the computational domain with a non-staggered, cell-centered arrangement as shown in Fig. 1. A second-order, central-difference scheme is used for the diffusion terms. In order to minimize the effect of numerical dissipation, we would like to use the non-dissipative second-order central difference scheme for the convective term also. However, it has been found that such schemes are prone to accumulation of aliasing errors [19] and especially within the context of compressible flows, some small level of numerical dissipation is required in order to control the growth of these errors. In the past, various approaches to this have been tried including the use of high-order upwind biased scheme [19,20] as well as flux limiters [21]. In the current solver we employ a hybrid second-order central-difference-QUICK scheme [22] for the discretization of the convective terms in conjunction with a flux-splitting method [23] which is expected to provide a high degree of control over the numerical dissipation. In order to understand this discretization scheme consider for example the convective flux term in the y_1 direction of the form $\frac{\partial F}{\partial y_1}$ which has to be evaluated at the cell center location denoted by P in Fig. 1. Using a second-order central-difference scheme, the above derivative can be approximated as

$$\left.\frac{\partial F}{\partial y_1}\right|_P = \frac{1}{\Delta y_1}(F_e - F_w) \tag{8}$$

Following this, F_e and F_w, the fluxes on the east and west faces, respectively are estimated as follows:

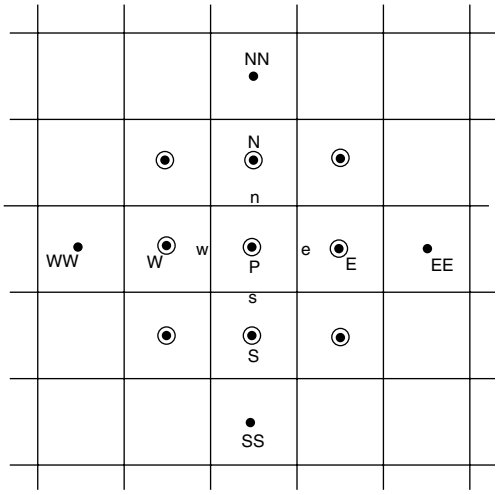

Fig. 1. Schematic shows the stencil is used in two-dimensional computational domain to discretize the equations.

$$F_e = (1-\xi)\underbrace{\frac{1}{2}(F_E + F_P)}_{\text{Central difference}} + \xi\underbrace{\left[\frac{1}{8}\left(-F_W^+ + 6F_P^+ + 3F_E^+\right) + \frac{1}{8}\left(3F_P^- + 6F_E^- - F_{EE}^-\right)\right]}_{\text{QUICK}} \tag{9}$$

$$F_w = (1-\xi)\underbrace{\frac{1}{2}(F_W + F_P)}_{\text{Central difference}} + \xi\underbrace{\left[\frac{1}{8}\left(-F_{WW}^+ + 6F_W^+ + 3F_P^+\right) + \frac{1}{8}\left(3F_W^- + 6F_P^- - F_E^-\right)\right]}_{\text{QUICK}} \tag{10}$$

In the above equations F^+ and F^- denote the downstream and upstream split fluxes which are obtained through a standard eigenvalue decomposition of the governing equations [23,24]. The key parameter in the above discretization is the adjustable weight factor ξ, which allows us to adjust the relative contributions of the central-difference and QUICK operators. Thus for $\xi = 0$ the scheme reverts to a pure second-order central difference scheme and for $\xi = 1.0$ we obtain the pure QUICK scheme which is itself a second-order scheme with a relatively low level of numerical dissipation [22]. Thus, by adjusting the weight factor, we get precise control over the numerical dissipation and can therefore adjust it appropriately for a particular simulation. There is no *a priori* method available for choosing the value of ξ for a given simulation. We essentially follow the principle that a simulation be performed with the minimum possible value of ξ that leads to an acceptable solution. For any given simulation, we start with a value of ξ equal to zero and successively increase it in increments of 0.03 after every thousand time-steps. Examination of the flow field at these stages allows us to determine a minimum value of ξ that leads to acceptable results. In all of the simulations presented in the current paper, the final value of ξ is at most equal to 0.1 and this severely constrains the magnitude of the numerical dissipation.

2.3. Temporal discretization

The diffusion terms can be broken up into diagonal terms (i.e. terms in Eq. (7) where $j = k$) which are analogous to terms that would appear on a Cartesian grid and cross-terms that appear due to grid non-orthogonality. Cross-terms (i.e. terms in Eq. (7) where $j \neq k$) contain cross-derivatives as well as velocity components in other directions. In the current approach, we employ an elliptic grid generator which produces meshes with fairly limited skewness and therefore the cross-terms in diffusion are very small compared to the diagonal terms. We treat the diagonal diffusion terms implicitly using a Crank–Nicolson scheme whereas all the other terms including the convective terms and cross-terms in diffusion are treated explicitly using a low-storage, third-order Runge–Kutta scheme [25,26]. The third-order Runge–Kutta scheme is attractive for time-accurate high Reynolds numbers simulations since this scheme has a larger stability region than the second-order

scheme and therefore allows larger time-steps. Equally important is the fact that the stability region of the third-order Runge–Kutta scheme includes the imaginary axis [27] and it therefore guarantees conditional stability even for very high Reynolds numbers up to the limit of purely inviscid flows. The resulting solver has overall second-order temporal accuracy and this will be demonstrated through a convergence study later in the paper.

The key advantage of explicit treatment of the cross-terms and convective terms is that this completely decouples the discretized momentum equations from each other thereby allowing us to solve them sequentially. Furthermore, implicit treatment of just the diagonal diffusion terms virtually eliminates the viscous stability constraint. The local CFL number in the current simulations is therefore estimated as

$$\mathrm{CFL} = \left[\frac{|u_1| + c}{\Delta x_1} + \frac{|u_2| + c}{\Delta x_2} + \frac{|u_3| + c}{\Delta x_3}\right]\Delta t \tag{11}$$

where c is speed of sound. The use of the Runge–Kutta scheme allows us to run these simulations with a variable time-step wherein the Δt in the simulations is chosen such that the maximum CFL number value in the domain is equal to about 1.0. No stability problems are encountered while running simulations under this scheme which indicates that the allowable time-step is not governed by the viscous time-scale thereby confirming that the viscous stability constraints have indeed been effectively eliminated by the temporal discretization adopted here. It should be noted that a similar semi-implicit approach has also been used successfully for direct and large-eddy simulation of incompressible turbulent flows on body-conformal curvilinear grids by Mittal and Moin [20] and Choi et al. [28].

With the above temporal discretization, each transport equation in the set Eqs. (1)–(3) is solved sequentially wherein we first solve the mass-conservation equation to get an updated density field. The inversion of the mass-conservation equation is straightforward since this equation does not contain any diffusion terms. This density is then used in the momentum equations to obtain updated momentum flux and velocity components in each direction. Subsequently, the updated density and velocity fields are used in the energy equation to update the temperature field. Finally, the pressure is updated by applying the equation-of-state. Thus, the entire set of equations is solved in a loosely coupled manner. The momentum and energy equations resulting from the discretization are solved by a Gauss–Siedel line successive over-relaxation (SOR) iterative method [18] which requires the solution of tridiagonal systems. Due to the diagonally dominant nature of the resulting discrete system of equations, the solver converges very rapidly and this results in a relatively fast solution procedure.

2.4. Boundary conditions

In this section we describe the typical boundary conditions used in the current simulations. The solver is currently designed for simulating external flows past immersed bodies and therefore we employ boundary conditions that are appropriate for such flows. At the regions of the outer boundary designated as inflow, all velocity components (and therefore the flow angle) as well as the temperature are specified, while the density is extrapolated from the flow domain and pressure calculated by using the equation-of-state. At the outflow, a non-reflecting Navier–Stokes characteristic boundary condition (NSCB) [29] is used which allows vortex structures to exit the computational domain with minimal spurious reflections. If the flow being simulated is unconfined then at the transverse boundaries we specify the free stream velocity and density, and temperature is determined by assuming adiabatic condition at this boundary. Pressure is again determined using the equation-of-state. Furthermore, lateral boundaries are located at a relatively large distance from the body in order to minimize confinement effect.

For solid boundaries we employ the no-slip, no-penetration boundary conditions along with an adiabatic boundary condition for temperature. Density is extrapolated from the interior of the flow onto the boundary and the pressure is determined by applying the equation-of-state at the wall. These boundary conditions are relatively easy to apply on a grid that conforms to the shape of the immersed body but the same is not true for a body non-conformal grid. Implementing this boundary condition for an immersed body in a consistent, efficient and accurate manner is the key issue that needs to be tackled and is the crux of any immersed boundary

method. In the following subsection we describe the method used to implement this boundary condition in the current solver.

2.5. Inclusion of the immersed boundary

The basic idea in this method is to compute the flow variables for a layer of cells inside and adjacent to the immersed boundary (the so called "ghost-cells") such that the boundary conditions on the immersed boundary in the vicinity of the ghost-cell are satisfied. The first step in this procedure is to define the immersed boundary and in the current solver we use closely spaced "body-markers" as shown in Fig. 2 to represent the boundary. Linear segments are extended between neighboring body markers to complete the specification of the immersed boundary. Next, a curvilinear body non-conformal grid is generated around this immersed boundary and in many of the cases presented here, we employ an elliptic grid generator available in GridGen® to create a smooth curvilinear grids. Following this, we identify "fluid cells" which are cells whose nodes lie outside the body and "solid cells" which as the name suggests, are cells whose nodes lie inside the solid body. There are a number of different ways of making this determination including ray-tracing [30] and we will not go into the details of this process here.

The next step is to identify the ghost-cells and a simple rule for this identification is that any cell is a ghost-cell if it lies within the computational stencil of a fluid cell. The stencil for the current spatial discretization scheme is shown in Fig. 1 and involves cells that are as far away as two cells from the central node. This relatively large stencil would lead to ghost-cells that, depending on the size and aspect-ratio, are embedded deep inside the immersed body. Our experience has shown that this situation can create both accuracy as well as convergence problems for the solver. Therefore we employ a simple remedy that alleviates this problem. We first identify "boundary cells" which are cells in the fluid that have at least one neighbor in the solid. For these cells we specify $\xi = 0$ which essentially forces the spatial discretization to revert to a pure central-difference scheme and eliminates the outlying nodes in the computational stencil. This modification of the discrete convective terms does not seem to have any deleterious effect on the solution since firstly the value of ξ in the rest of the domain is relatively small (typically less than 0.1) and secondly, the momentum and energy fluxes in these boundary cells are dominated by the diffusive components since they are located well inside the boundary layer. With this strategy we can now identify ghost-cells as all cells that have a one immediate neighbor in the fluid including even diagonal neighbors as shown in Fig. 2.

Following this, we need to devise a scheme that will allow us to compute the value of the variables at each of these ghost-cell nodes such that the boundary condition on the immersed boundary in the vicinity of the

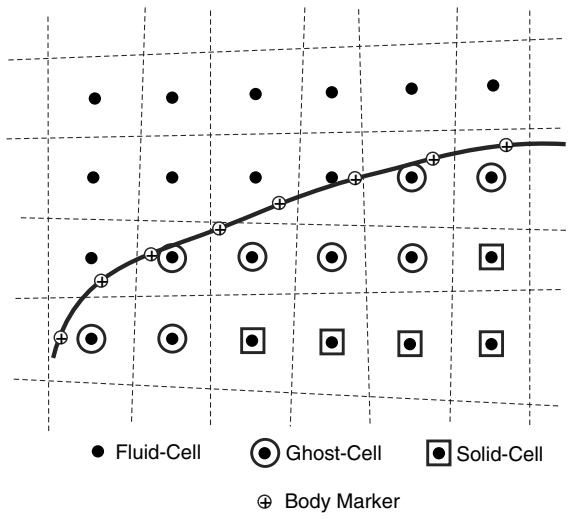

Fig. 2. Schematic showing an immersed boundary on a generalized curvilinear mesh along with grid cells identified as fluid, ghost or solid cells.

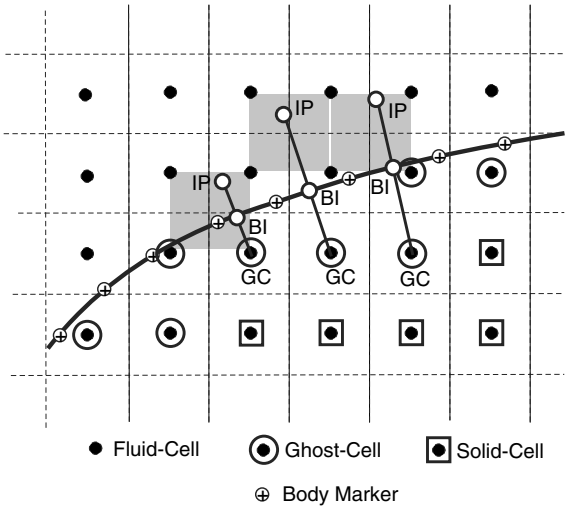

Fig. 3. Schematic in the computational domain corresponding to Fig. 2. Indicated on the figure are three types of image-points that can be encountered in the ghost-cell interpolation scheme.

ghost-cell is satisfied. The procedure starts by determining a point on the immersed boundary that is closest to the ghost-cell node in question. Obviously this point can be found by determining the normal "body-intercept" from the ghost node onto the immersed boundary. This is the location where the boundary condition will be satisfied and the key now is to determine a simple method for accomplishing this. It should be noted that the boundary conditions that we need to apply are either Dirichlet type, i.e.

$$\phi_{\mathrm{BI}} = \Phi \tag{12}$$

where ϕ is a generic variable and BI denotes body-intercept, or Neumann type of the form

$$(\hat{n} \cdot \nabla\phi)_{\mathrm{BI}} = \Psi \tag{13}$$

In the current solver, we first start by extending the body-intercept out into the fluid to a distance equal to the distance between the intercept point and the ghost-cell node. The tip of this segment is called the "image-point" for convenience. The overall strategy now is to express the value of the variable at the image-point in terms of the surrounding nodal values and then use this value and the boundary condition to extrapolate along the normal and obtain a value for the ghost-cell node. In conventional immersed boundary methods of this type that are implemented on Cartesian grids such as those of Majumdar et al. [31] and Bozkurttas et al. [32], this procedure is relatively straightforward. However, the use of a curvilinear grid here leads to some complexities that have to be addressed. In particular, the interpolation procedure can either be implemented in real space or in computational space. We have found that implementation in computational space is especially convenient and allows us to maintain the expected accuracy. The essential features of this approach are now explained. The discussion is based on Fig. 3, which is a representation in the computational domain of the grid schematic in Fig. 2.

The interpolation procedure is initiated by determining the coordinates of the image-point in computational space. For this we identify the grid-point closest to the image-point. Denoting the coordinates of this nearest grid point in the real and computational domains as (X_1, X_2) and (Y_1, Y_2), respectively, the coordinates of the image-point in the computational domain $(y_{1_{\mathrm{IP}}}, y_{2_{\mathrm{IP}}})$ are given by

$$\begin{Bmatrix} y_{1_{\mathrm{IP}}} \\ y_{2_{\mathrm{IP}}} \end{Bmatrix} = \begin{Bmatrix} Y_1 \\ Y_2 \end{Bmatrix} + \frac{1}{J} \begin{bmatrix} \beta^{11} & \beta^{12} \\ \beta^{21} & \beta^{22} \end{bmatrix} \begin{Bmatrix} x_{1_{\mathrm{IP}}} - X_1 \\ x_{2_{\mathrm{IP}}} - X_2 \end{Bmatrix} \tag{14}$$

where $(x_{1_{\mathrm{IP}}}, x_{2_{\mathrm{IP}}})$ are the coordinates of the image-point in real space and the matrix on the right-hand side contains the transformations metrics evaluated at (Y_1, Y_2). The above essentially amounts to a second-order

accurate interpolation procedure and a similar procedure can be used to obtain the coordinates of the body-intercept point in the computational domain.

Following this, the value of the variable at the image-point is expressed in computational domain in terms of a bi-linear interpolant of the form:

$$\phi = C_1 y_{1_{\mathrm{IP}}} y_{2_{\mathrm{IP}}} + C_2 y_{1_{\mathrm{IP}}} + C_3 y_{2_{\mathrm{IP}}} + C_4 \tag{15}$$

In the above function, the four unknown coefficients C_i can be expressed or determined in terms of the values of the variables at the four nodes surrounding the image-point. At this stage, three different situations shown in Fig. 3 can be encountered for a given image-point and these have to be handled in a well-posed and consistent manner. The simplest situation is when all four nodes surrounding the image-point are in the fluid. Of the three image-points identified in the figure, the middle image-point corresponds to this first situation. In this case, nothing special is needed and we use Eq. (15) to express the value at the image-point in terms of the values at these four surrounding nodes. Thus we can write the following equation:

$$\{\phi\} = [V]\{C\} \tag{16}$$

where

$$\{\phi\} = \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} \tag{17}$$

are the values of the variable at the four surrounding points and

$$V = \begin{bmatrix} y_1 y_2|_1 & y_1|_1 & y_2|_1 & 1 \\ y_1 y_2|_2 & y_1|_2 & y_2|_2 & 1 \\ y_1 y_2|_3 & y_1|_3 & y_2|_3 & 1 \\ y_1 y_2|_4 & y_1|_4 & y_2|_4 & 1 \end{bmatrix} \tag{18}$$

is the Vandermonde matrix corresponding to the bilinear interpolation scheme. The subscripts in the above equation are identifiers of the four surrounding nodes. Furthermore,

$$C = \begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} \tag{19}$$

is the matrix containing the four unknown constants. These constants can be determined by inverting Eq. (16) and finally by using Eq. (15), the value at the image-point can be expressed as

$$\phi_{\mathrm{IP}} = \sum_{i=1}^{4} \alpha_i \phi_i \tag{20}$$

In the above equation, αs depend on Cs as well as the coordinates of the image-point. Since all of these depend only on the geometry of the immersed boundary and the grid, the αs can be determined as soon as the immersed boundary and grid are specified.

The second situation is more complicated and occurs when one of the four surrounding nodes is the ghost-node itself. This situation is the one on the right in Fig. 3. Clearly use of the ghost-node value in the interpolation scheme itself would not be well-posed. Instead we use the boundary conditions at the body-intercept along with the nodal values at the three surrounding nodes to close the interpolation. In the case where we are applying a Dirichlet boundary condition at the body-intercept point, the row in Eq. (16) which would have corresponded to the ghost-node is simply replaced by the corresponding values at the body-intercept point. In the case where a Neumann boundary condition Eq. (13) is being applied, we first transform the boundary condition to the computational domain as follows

$$\left(\widehat{N}\cdot\nabla_y\phi\right)_{\mathrm{BI}} = \Psi \tag{21}$$

where ∇_y is the gradient operator in computational space and $N_i = \frac{1}{J}n_j\beta^{ij}$ is the body-normal transformed to computational space. Within the context of the current bilinear interpolation scheme, the above boundary condition can be expressed as

$$N_1\left(C_1 y_{2_{\mathrm{BI}}} + C_2\right) + N_2\left(C_1 y_{1_{\mathrm{BI}}} + C_3\right) = \Psi \tag{22}$$

and the corresponding Vandermonde matrix for this case then becomes

$$V = \begin{bmatrix} y_1y_2|_1 & y_1|_1 & y_2|_1 & 1 \\ y_1y_2|_2 & y_1|_2 & y_2|_2 & 1 \\ y_1y_2|_3 & y_1|_3 & y_2|_3 & 1 \\ N_1y_{2_{\mathrm{BI}}} + N_2y_{1_{\mathrm{BI}}} & N_1 & N_2 & 0 \end{bmatrix} \tag{23}$$

where we assume for the sake of discussion that it is the fourth node in the stencil that corresponds to the ghost-node. Note also that the $\{\phi\}$ matrix for this case would be modified to

$$\{\phi\} = \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \Psi \end{Bmatrix} \tag{24}$$

Following this the procedure for obtaining the corresponding discrete representation for ϕ_{IP} would be the same as described for the first case and would lead to an expression similar to Eq. (20).

The third and final situation is where the interpolation stencil for an image-point contains ghost-nodes other than those associated with the current image-point. This situation is also depicted in Fig. 3 on the left. In this particular situation we find that the use of the value of the variable at this other ghost-node is well-posed and consistent with the overall solution procedure and therefore no special treatment is needed for this node. It should also be pointed out that since the values of the fluid as well as the ghost-nodes are updated simultaneously during the SOR iterative process, there is no issue regarding the initialization of the ghost-node values or the order in which they are solved for.

With the value at the image-point expressed in terms of the surrounding nodes and boundary values, we can now evaluate the variable value at the ghost node. For this we employ linear interpolation along the normal and obtain the value at the ghost-cell node as

$$\phi_{\mathrm{GC}} = \zeta\phi_{\mathrm{IP}} + \Gamma \tag{25}$$

For Dirichlet boundary condition $\Gamma = 2\Phi$ and $\zeta = -1$. For the Neumann boundary condition $\Gamma = \Psi\cdot\Delta l$ and $\zeta = 1$ where Δl is the length of the normal segment in computational domain. It should be noted that the above is a second-order accurate interpolation scheme which is consistent with the overall accuracy of the flow solver. By combining Eq. (25) with Eq. (20) the final expression for the ghost-cell node can be written as

$$\phi_{\mathrm{GC}} = \zeta\sum_{i=1}^{4}\alpha_i\phi_i + \Gamma \tag{26}$$

Thus the value at the ghost-cell node is written completely in terms of the nodal values of the nodes surrounding the corresponding image-point and the boundary condition at the corresponding body-intercept point. This equation can be solved in a coupled manner with the discretized fluid equations for the fluid-nodes and the trivial $\phi = 0$ equation for the solid-nodes. As noted before, the entire system is solved in a loosely coupled manner using a Gauss–Siedel line-SOR iterative method and converges quite rapidly.

It is useful to compare and contrast the current approach with the "ghost-fluid method" of Fedkiw and coworkers [35,12], since this method also employs ghost-nodes in order to close the discretized system of equations. The work of Gibou et al. [12] is particularly relevant here since there the authors introduced a second-order accurate version of their method. In this method, ghost-nodes are identified in a manner similar to the

current work, however, the interpolation scheme adopted in order to determine the value at the ghost-nodes is quite different. In this earlier work, values at the ghost-nodes are computed through one-dimensional extrapolations schemes along the Cartesian grid lines. Thus considering for example the middle ghost-cell shown in Fig. 3, the linear interpolation scheme adopted by Gibou et al. [12] would employ the nodal value to the north of the ghost-cell as well as the boundary value on the interface at the location where the immersed boundary cuts the vertical line joining the ghost-cell and the north-cell. Thus for a Dirichlet boundary condition, the method of Gibou et al. [12] would construct the following extrapolation scheme for the ghost-cell:

$$\phi_{\mathrm{GC}} = \frac{\phi_{\mathrm{BI}} + (\theta - 1)\phi_{\mathrm{N}}}{\theta} \tag{27}$$

where subscript BI corresponds here to the point where the immersed boundary intersects the vertical line between the ghost-cell and north node, and subscript N corresponds to the node to the north of the ghost-cell. Furthermore, θ is the weight-factor of the linear interpolation scheme and would be given here by $\theta = (y_{2\mathrm{N}} - y_{2\mathrm{BI}})/(y_{2\mathrm{N}} - y_{2\mathrm{GC}})$. Gibou et al. [12] point out that the above interpolation scheme is poorly behaved for small θ and for such situations, they resort to imposing $\phi_{\mathrm{N}} = \phi_{\mathrm{BI}}$ which is locally first-order accurate. In contrast, in the current interpolation scheme, we ensure that the body-intercept is always exactly midway between the ghost and image-point and this guarantees that the interpolation scheme in Eq. (25) remains well behaved even in the limit of vanishing probe length. In fact, in this limiting case, Eqs. (20) and (25) automatically result in ϕ_{IP} and ϕ_{GC} both approaching ϕ_{BI} in a smooth manner for a Dirichlet boundary condition whereas for a Neumann boundary condition, ϕ_{GC} smoothly approaches ϕ_{IP}. Nothing special therefore needs to be done to handle such cases in the current approach. It should also be pointed out that Neumann boundary conditions are quite easy to implement in the current method since the interpolation is performed normal to the immersed boundary. In the ghost-fluid method, the interpolation is along the principle direction which would tend to complicate the imposition of Neumann boundary conditions. An advantage of the ghost-fluid method is that at least for the linear interpolation in Eq. (27), the stencil for the ghost-cell is limited at most to the usual five-point stencil (in 2D) for the Laplacian operator. In contrast in the current method, the stencil for the ghost-cell can contain points outside the usual five-point stencil. However, this has no significant deleterious effect on the convergence of the momentum equation since these equations are diagonally dominant.

It should also be noted that the current method does not encounter any "small-cell" stability problems [15]. The effective size of a cell remains unaltered despite being cut by the immersed boundary and is not dependent in any way on the size of the intercept used for the interpolation or the relative distance between the points used in the interpolation scheme. On the other hand, the current method does not enforce strict conservation of fluxes for the cells that are cut by the immersed boundary, a property that is built into the embedded boundary methodology of [15]. It seems that all finite-difference based immersed/embedded boundary methods such as the current one as well as others [8,9,12], have to choose between higher (second in most cases) local accuracy or strict discrete conservation. In the method of [15], local accuracy is reduced to first-order while maintaining strict discrete conservation and this might be especially appropriate for high Mach number, inviscid flows where the flow is dominated by strong shocks and discontinuities, and resolution of boundary layers is not important. In the current study, we are primarily interested in subsonic, viscous flows at moderate to high Reynolds numbers where local accuracy in the vicinity of the immersed boundary is extremely important. As our results will confirm, the current method provides stable and accurate prediction of the global as well as local characteristics for a variety of such flows. It seems that only with a finite-volume based immersed-boundary method such as that of [7] which employs "cut-cells", is it possible to simultaneously maintain high local accuracy and strict discrete conservation. Note also that similar to the current method, the method of [7] does not suffer from the "small-cell" stability problem. However, extension of the cut-cell methodology to general 3D problems remains a highly challenging proposition [2].

2.6. Parallelization on distributed memory multiprocessing systems

The solver which was written in FORTRAN-90, has been modified so that it can run efficiently on distributed memory as well as shared memory multiprocessing systems. In the current solver we employed message

passing interface (MPI) in conjunction with a domain-decomposition strategy in order to implement the parallelization. As will be shown, the simple, single-block structured curvilinear grid makes the parallelization of the solver relatively straightforward and demonstrates yet another advantage of the immersed boundary approach.

Most of the CPU time in the numerical solution is consumed by the Gauss–Siedel line-SOR solution procedure that is used for the solution of the discretized mass, momentum and energy equations. Therefore, the parallelization strategy has to be designed around this module. The line-SOR scheme involves successive tri-diagonal matrix solutions along the y_1 direction followed by the y_2 direction and in the current solver we employ the well known Thomas algorithm [33] for this purpose.

In the current solver we employ a one-dimensional domain decomposition wherein the streamwise (y_1) direction is divided into M domains, where M is the number of processors. The streamwise direction is chosen for decomposition since in most of the current applications, the number of grid points along the streamwise direction is significantly larger than the other directions. It should be pointed out that although the current decomposition is one-dimensional, it can in principle be extended to multi-dimensional domain decomposition as well.

Consider now the solution of tri-diagonal systems along the streamwise (y_1) direction. Due to the domain decomposition along this direction, the tridiagonal matrix solution procedure has to be modified. Here we adopt a simple strategy of using an explicit update at the internal boundaries of the domain. The strategy is best demonstrated by considering an example. Consider a tri-diagonal matrix system given below:

$$\begin{pmatrix} a_1 & b_1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ c_2 & a_2 & b_2 & 0 & 0 & 0 & \cdots & 0 \\ 0 & c_3 & a_3 & b_3 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & c_p & a_p & b_p & 0 & \cdots & 0 \\ \cdots & 0 & 0 & c_{p+1} & a_{p+1} & b_{p+1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & 0 & 0 & 0 & 0 & c_{N_1} & a_{N_1} \end{pmatrix} \begin{Bmatrix} \phi_1^k \\ \phi_2^k \\ \phi_3^k \\ \vdots \\ \phi_p^k \\ \phi_{p+1}^k \\ \vdots \\ \phi_{N_1}^k \end{Bmatrix} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_p \\ r_{p+1} \\ \vdots \\ r_{N_1} \end{Bmatrix} \tag{28}$$

where k corresponds to the iteration index, N_1 is the number of grid points in the x_1 direction and r contains the right-hand side vector. Let us assume that the first domain contains 'p' number of grid points. Then, as per the current procedure adopted here, we obtain an approximate solution ϕ' of this tri-diagonal system by solving the following system of equations:

$$\begin{pmatrix} a_1 & b_1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ c_2 & a_2 & b_2 & 0 & 0 & 0 & \cdots & 0 \\ 0 & c_3 & a_3 & b_3 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & c_p & a_p & 0 & 0 & \cdots & 0 \\ \cdots & 0 & 0 & 0 & a_{p+1} & b_{p+1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & 0 & 0 & 0 & 0 & c_{N_1} & a_{N_1} \end{pmatrix} \begin{Bmatrix} \phi_1'^k \\ \phi_2'^k \\ \phi_3'^k \\ \vdots \\ \phi_p'^k \\ \phi_{p+1}'^k \\ \vdots \\ \phi_{N_1}'^k \end{Bmatrix} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_p - b_p \phi_{p+1}'^{k-1} \\ r_{p+1} - c_{p+1} \phi_p'^{k-1} \\ \vdots \\ r_{N_1} \end{Bmatrix} \tag{29}$$

where $k-1$ corresponds to the previous iteration. The above procedure temporarily decouples the solution in one domain from the solution in the adjacent domains and the solution procedure can therefore proceed independently in each domain.

At this point there are two strategies that can be adopted: first, one could perform sub-iterations for a given tridiagonal system and exchange information at the boundaries of the domains after each sub-iteration until the exact solution of Eq. (28) is obtained to certain accuracy. Subsequently, we would move to the next line-solve in the sequence. The second strategy would be to proceed to the next line-solve in the sequence immediately after obtaining the approximate solution ϕ'. Clearly the first strategy requires more CPU time per line-solve whereas the second strategy introduces additional error into the iterative solution and could potentially require a larger number of iterations for convergence. The strategy that leads to the lower overall CPU time is the one that should be adopted.

In order to better understand the CPU costs of each strategy, consider the extreme situation where each domain has only one point, i.e. M is equal to N_1. In this case, the second strategy effectively reduces the Gauss–Siedel line-SOR to a Jacobi point-SOR iteration. Now, theoretical estimates of asymptotic convergence rates indicate that the line Gauss–Siedel line method converges approximately four times faster than point-Jacobi [34]. This implies that even in this limiting case, the first strategy would be competitive with the second only if it took less than about four sub-iterations to solve the tri-diagonal system. However, in this limiting case, the first strategy also implies the solution of the tri-diagonal system using a point Jacobi iterative method. Since this is a slowly converging method, it is expected to take considerably larger number of iterations to converge. Thus even in this limiting case, the second strategy is expected to be more efficient than the first one. In realistic situations where $N_1 \gg M$ we would expect the second strategy to be even more effective than the first. Consequently, we employ the second strategy in our domain decomposition algorithm.

A key aspect of any distributed memory parallelization algorithm based on domain decomposition is the use of overlap regions at the boundaries of the domains. The size of the overlap region is usually determined by the size of the computational stencil used in the spatial discretization scheme. However, for the current solver, in addition to the discretization scheme which is shown in Fig. 1 we also need to consider the extent of the interpolation stencil for the ghost-cells in determining the size of the overlap region. For non-isotropic grids, nonuniform grids such as the ones used in the current study, the size of this interpolation stencil can vary over a large range. This is yet another feature that differentiates the current immersed boundary method from methods developed for Cartesian grids. In the current solver, we have allowed for a variable size overlap region and the overlap region for a given simulation is determined as a preprocessing step. For all the simulations presented in the current study, the overlap region is found to vary from two to five cells. Results on the parallel performance of the solver are presented in the following section.

3. Results and discussion

In order to assess the accuracy and validate our methodology, we simulate two- and three-dimensional flow past circular cylinders over a wide range of Reynolds numbers. The method is then applied to two-dimensional flow around an airfoil and three-dimensional wing-tip flow to demonstrate the ability and performance of the method for simulating flows with complex geometries.

3.1. Spatial and temporal accuracy study

In this section we describe the results of grid and time-step convergence studies carried out with the current solver. In addition to the second-order accurate spatial discretization used for the regular fluid cells, care has been taken to maintain a second-order accurate treatment in the imposition of the boundary condition on the immersed boundary. Thus, we expect the solver to exhibit second-order global and local accuracy. The second-order accuracy for the cells in the vicinity of the immersed boundary is especially important for the resolution of thin boundary layers that develop on the immersed boundary for moderate to high Reynolds number flows. Here we examine the spatial accuracy of the solver for flow past a circular cylinder at $Re_d = 45$ and $M = 0.2$. For this test, we employ a uniform Cartesian grid on a domain size of $4d \times 4d$. Since an exact solution for this case does not exist, we use the solution computed on a highly resolved 1260×1260 mesh as a baseline for computing the numerical error. A uniform time-step of $1 \times 10^{-4} d/U_\infty$ is used and we integrate the solution for 1×10^4 time-steps. The resulting solution is denoted by $\phi^{1260\times1260}$ and is shown in Fig. 4. The same flow is then computed on a 420×420, a 252×252, a 180×180, and a 84×84 grid with

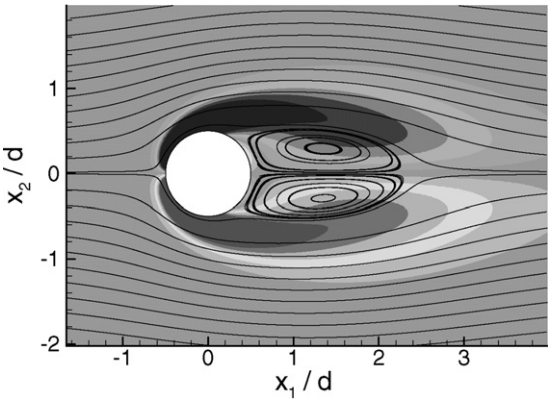

Fig. 4. Streamlines and spanwise vorticity contours for computed flow past a circular cylinder at $Re_d = 45$.

the same time-step size. The residual criterion for the SOR iteration is kept below 10^{-6} in order to ensure that iterative convergence does not contaminate the spatial errors. The L_2 and L_∞ norms of the error for a solution on a $N \times N$ grid can now be computed as

$$\varepsilon_2 = \left[\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\phi_{i,j}^{N\times N} - \phi_{i,j}^{1260\times 1260}\right)^2\right]^{1/2} \tag{30}$$

and

$$\varepsilon_\infty = \max\left|\phi_{i,j}^{N\times N} - \phi_{i,j}^{1260\times 1260}\right|; \quad i,j = 1,N \tag{31}$$

respectively. It should be pointed out that the L_2 error-norm is a good measure of the global error whereas the L_∞ error-norm effectively captures the local error around the immersed boundary.

Fig. 5 shows the variation of the L_2 and L_∞ error norms in the two velocity components plotted versus $1/N$ where N is the number of grid points in each direction. Also included in the log–log plot is a line denoting second-order convergence. Both error norms show nearly second-order convergence thereby confirming that the current immersed boundary solver is globally and locally second-order accurate.

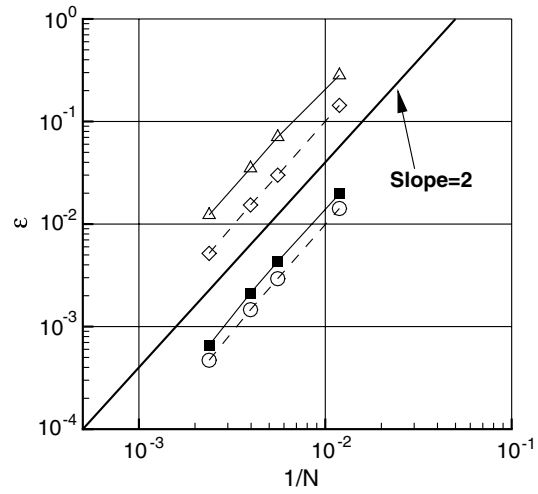

Fig. 5. L_2 and L_∞ norms of the error for the streamwise velocity (u_1) and transverse velocity (u_2) components versus the computational grid size. ■: $L_2 - u_1$, ○: $L_2 - u_2$, △: $L_\infty - u_1$, ◇: $L_\infty - u_2$.

We have used this same configuration to establish the temporal accuracy of the solver. Note that we are employing a semi-implicit temporal discretization scheme which combines a second-order Crank–Nicolson scheme with a third-order Runge–Kutta scheme. The overall temporal accuracy of the solver is therefore expected to be second-order. We confirm this by performing a time-step convergence study. Flow past the circular cylinder was simulated on a fixed grid with a very small time-step of $1 \times 10^{-4} d/U_\infty$ for 320 time-steps and this was designated as the baseline solution for computing the temporal error. Subsequently the flow was computed to the same time-instant on the same grid in four additional simulations that employed larger time-steps ranging from 4×10^{-4} to 3.2×10^{-3}. The L_2 norm of the error in the solution was then computed with respect to the baseline solution and this is plotted in Fig. 6. The plot shows a slope that very nearly matches the second-order line thereby confirming the expected temporal accuracy of the solver.

3.2. Parallel performance

The parallel code has been successfully tested on our in-house parallel computers. The test case chosen corresponds to a 3D simulation of flow past a circular cylinder with a $548 \times 385 \times 10$ grid. The first test was carried out on a 24-CPU Beowulf cluster which consists of 12, dual 2.8 GHz Pentium-4 processors and employs a gigabit ethernet between the nodes. Each dual processor node has four Gigabytes of core memory. The computer employs RedHat 9.2 Linux operating system and PGI Fortran compiler. It should be pointed out that the gigabit ethernet interconnect used here has a documented 80 MBps bandwidth and 170 ms latency. In contrast, Myrinet, which is a popular interconnect used in high performance Beowulf clusters, has a 225 MBps bandwidth and 7 ms latency. Thus the network employed here has almost a three times lower bandwidth and 24 times higher latency than Myrinet. This difference in network performance needs to be kept in mind while assessing the parallel performance of the solver on this platform. It should also be noted that the grid chosen for the parallel speedup study is practically the largest grid on which a simulation can be performed on a single node of this computer. Larger grids exhaust the available core memory of the node leading to excessive disk swapping and such cases do not provide a realistic assessment of the parallel performance.

Simulations have been performed on up to 22 processors and the parallel speedup of the solver estimated by comparing the execution time for each simulation with the corresponding simulation on one processor. Fig. 7 shows the parallel speedup obtained from these tests. It is noted that the solver exhibits reasonably good speedup over the range of processors tested with a speedup factor of about 64% obtained for 22 processors. Also worth noting is the fact that that at least up to 22 processors, the speedup curve does not seem to show an asymptotic limiting behavior which indicates that even for the relatively small mesh chosen, the parallel performance would be maintained even on larger number of processors.

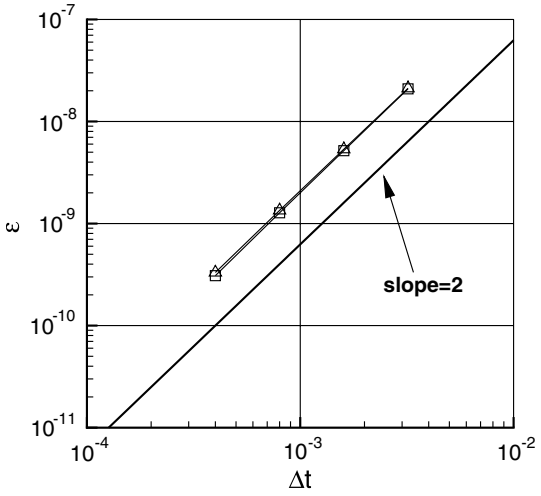

Fig. 6. L_2 norm of the temporal error versus the Δt. □: u_1, △: u_2.

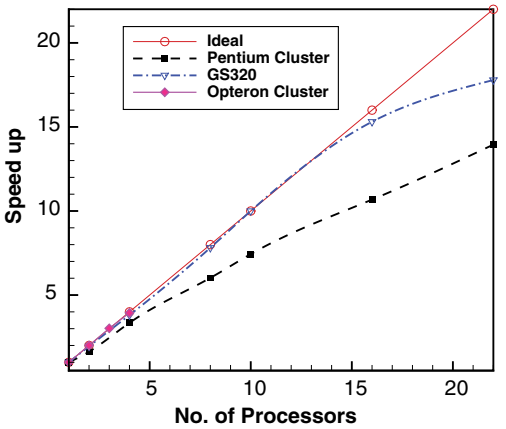

Fig. 7. Speed up of the solver versus number of the processors.

The second set of tests have been carried out on a 24-processor HP Alphaserver GS320 computer which is a ccNUMA (cache-coherent Non-Uniform Memory Access) platform with 731 MHz Alpha EV67 processors and 128 gigabytes of total memory. The computer is partitioned into a 16 and an 8 CPU partition and each partition essentially shares memory. Communication across the partition occurs over a high bandwidth switch. The computer uses the TRU64 UNIX operating system and COMPAQ FORTRAN-90 compiler. Since this is a shared memory platform we expect better speedup performance on this platform and as can be seen in Fig. 7, this is indeed found to be the case. The plot shows that up to about 16-processors we get almost perfect speedup. Beyond that, there is some drop in the parallel performance but despite that a reasonably good speedup factor of about 82% is obtained on 22 processors. The nearly perfect speedup up to 16 processor also provides indirect proof that the domain decomposition strategy and the associated modifications in the line-SOR iterative procedure do not increase the computational cost of the solution process to any appreciable extent.

The final set of tests have been carried out on a 4-CPU, 844 MHz AMD Opteron cluster which has a 64-bit operating system. The four CPUs share a total of 16 gigabytes of memory and therefore this is also a shared-memory system. The computer uses the SuSe Linux operating system and a 64-bit PGI Fortran-90 compiler. The current test case is run on one, two and four processors and the results shown in Fig. 7 again indicate an almost perfect speedup. This is inline with the speedup observed on the shared memory HP Alphaserver GS320 system and confirms the effectiveness of the current domain decomposition strategy.

3.3. Circular cylinder flow simulations

The flow past a circular cylinder depends on the Reynolds number which is defined as $Re_d = U_\infty d/v$ where d is the cylinder diameter, U_∞ is the free stream velocity and v is the kinematic viscosity. At Reynolds numbers up to around $Re_d = 47$, the flow is steady and symmetrical about the wake-centerline. At Reynolds numbers higher than this value, the flow become unstable to perturbations and leads to periodic shedding of vortices and the formation of the Karman vortex street. However, the flow remains two-dimensional up to a Reynolds number of about 180 [36]. Beyond this the flow becomes susceptible to a secondary instability mechanism that leads to intrinsic three-dimensionality in the wake [37,36]. Due to extensive numerical and experimental investigations of this flow, it is an excellent case for validating a numerical solver. We have performed 2D and 3D simulations for Reynolds numbers ranging from 20 to 3900 and compared the computed results with available numerical and experimental results. The free stream Mach number for all these simulations is kept at a relatively low value of 0.2. However, one case at $Re_d = 80$ has been performed with free stream Mach number of 0.4 and this is intended to demonstrate the performance of the solver for a case where compressibility effects are not negligible.

A curvilinear grid with 459×261 grid points shown in Fig. 8 was employed for all cases except the $Re_d = 3900$ for which a 549×385 curvilinear grid of similar topology was used. As can been seen in Fig. 8, the grid around the cylinder although not exactly body-conformal, does provide enhanced resolution to the boundary layer on the cylinder surface. Fig. 9 shows computed spanwise vorticity contour plots over a wide range of Reynolds numbers. As expected, all plots show distinct vortices associated with the Karman vortex street which extend up to 10 diameters downstream and beyond in the wake. It should further be noted that for simulations up to $Re_d = 590$ we employed a weight factor (ξ) of 0.05 whereas for the higher Reynolds numbers, a value of 0.1 was used. Attention needs to be drawn to the absence of excessive dissipation or dispersion errors in these vorticity contour plots which is indicative of the favorable resolution characteristics of the current hybrid spatial discretization scheme.

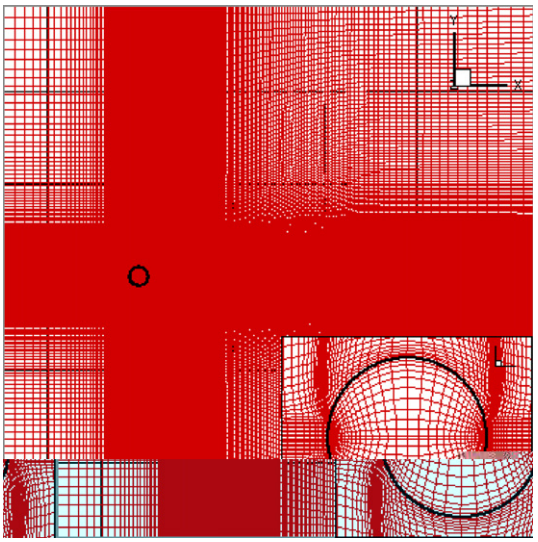

Fig. 8. The 459×261 curvilinear grid used for the circular cylinder simulations. Inset shows a close-up view of the grid around the cylinder.

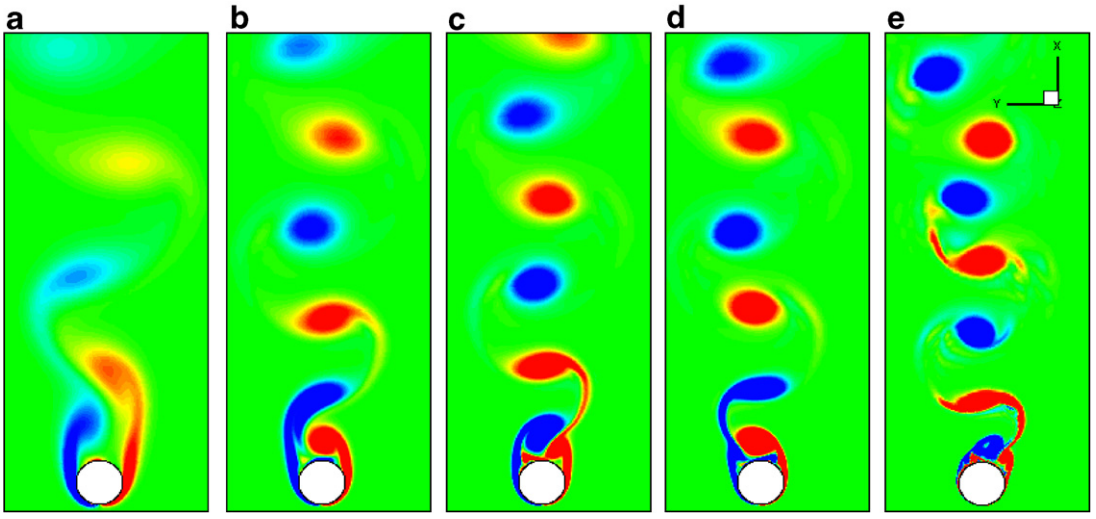

Fig. 9. Instantaneous vorticity contour plots in the wake of a circular cylinder at (a) $Re_d = 100$, (b) $Re_d = 300$, (c) $Re_d = 590$, (d) $Re_d = 1000$ and (e) $Re_d = 3900$.

Next we examine the computed surface pressure, drag and lift coefficients and compare them to established data. These quantities are based on the evaluation of surface pressure and/or shear stress and the procedure used to compute these surface quantities needs some explanation. In the current solver, we identify four nodes surrounding a body-marker (see Fig. 3) and then use a bilinear interpolation to estimate the pressure on this body marker. Shear stress at the body-intercept points (τ_{BI}) is also computed in a straightforward manner by using the values at body-intercept point and corresponding image-points via

$$\tau_{\mathrm{BI}} \approx \frac{\mu}{\Delta L}\left[\vec{u}_{IP} - \vec{u}_{\mathrm{BI}}\right] \cdot \hat{t}_{\mathrm{BI}} \tag{32}$$

where $\vec{u}$ is the flow velocity, μ is the fluid viscosity, ΔL is the length of the normal segment in real space and $\hat{t}$ is the surface tangent at the body-intercept point. Fig. 10 shows the mean pressure coefficient on the surface of the cylinder the instantaneous value of which at a point is defined as $C_{\mathrm{p}} = (p - p_\infty)/(\frac{1}{2}\rho_\infty U_\infty^2)$. The key point to note is the smoothness of the surface pressure over the entire range of Reynolds numbers which indicates that the flow and pressure field in the vicinity of the immersed boundary are well resolved and do not suffer from any spurious effects. The computed mean based pressure coefficient (C_{pb}) is compared to the well established experimental data of Williamson and Roshko [38]. Note that for Reynolds number greater than about 180, the flow past a circular cylinder is intrinsically three-dimensional and therefore 2D simulations at these Reynolds are not expected to produce flow fields and pressure distributions that would match with corresponding experiments. As shown by Mittal and Balachandar [37], base suction pressure and drag are typically over predicted in 2D simulations. Thus the current 2D simulation results match the experimental data of Williamson and Roshko [38] quite well up to about $Re_d = 180$ although beyond that there is a significant deviation between the two data sets. However, the current data does match well with the results of other 2D simulations in the entire range [19,37,39] thereby validating the accuracy of the current 2D simulations.

Next, we compare computed lift and drag with established results. The drag and lift coefficients are defined as $C_{\mathrm{D}} = F_{\mathrm{D}}/\left(\frac{1}{2}\rho_\infty U_\infty^2 d\right)$ and $C_{\mathrm{L}} = F_{\mathrm{L}}/\left(\frac{1}{2}\rho_\infty U_\infty^2 d\right)$. In these expressions, F_{D} and F_{L} are the drag and lift forces, respectively that are computed by integrating the pressure and shear stress on the cylinder surface. In Fig. 11 is plotted the variation of drag and lift coefficients with time for $Re_d = 300$, and 3900 and the plots show a quasi-periodic variation in these quantities which is inline with the simulations of Mittal and Balachandar [20] and Beaudan [19]. Past simulations of compressible bluff-body wake flows have shown that spurious reflections from the outflow boundary can lead to the appearance of distinct and strong low-frequencies in the wake [41] even at Reynolds numbers as low as 80. No such low frequency phenomenon is observed in the current simulations even for relatively high Reynolds numbers and this is indicative of the effectiveness of the non-reflective boundary condition used in the current simulations.

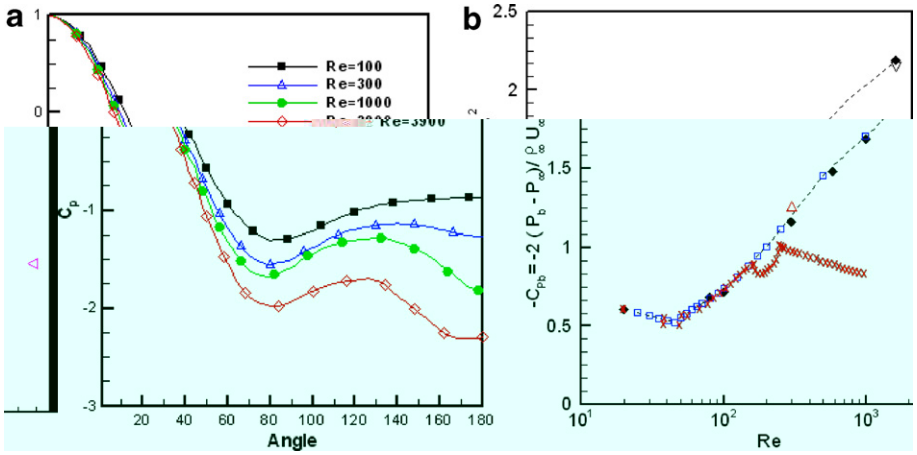

Fig. 10. (a) Pressure coefficient distribution on the surface of the circular cylinder at different Reynolds numbers. (b) Comparison of computed base pressure coefficients with other data; ◆: present work, *X*: Willamson and Roshko [38], □: Henderson [39], △: Mittal and Balachandar [37], ▽: Beaudan [19], ◁: Norberg [40].

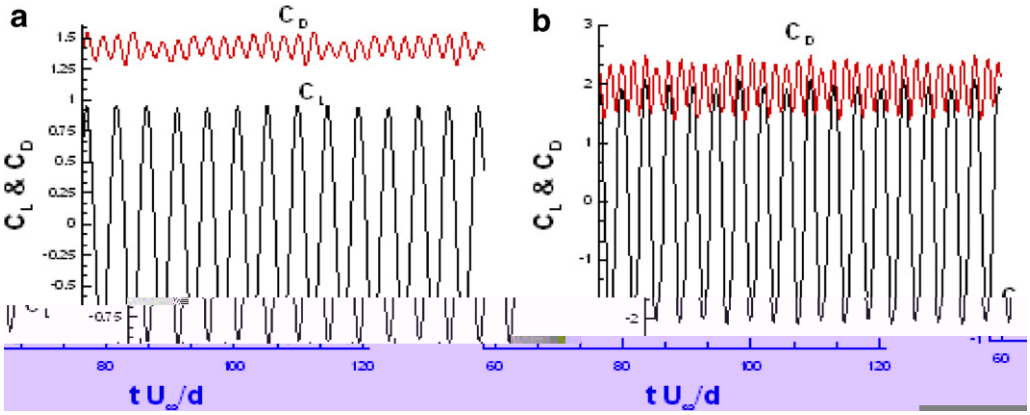

Fig. 11. Temporal variation of drag and lift coefficients: (a) $Re_d = 300$ and (b) $Re_d = 3900$.

The pressure and shear components of the drag force on the cylinder has been time-averaged over a number of shedding cycles for all the cases simulated here and Fig. 12(a) shows a comparison of these two components with the 2D simulations of Henderson [39] which were carried out using a spectral element method. It can be seen that over the entire range of Reynolds numbers simulated here, the results are in excellent agreement with the results of Henderson. This provides further verification of the accuracy and fidelity of the solver. Finally, the lift variation is used to estimate the vortex shedding Strouhal number and this is compared to established experimental and numerical data in Fig. 12(b). Here too, the match up to a Reynolds number of about 200 is quite good. Beyond that the 2D simulations deviate from the experimental results which are intrinsically three-dimensional. However the match at $Re_d = 3900$ with the 2D simulations of Beaudan [19] is reasonably good.

All of the previous cases have been simulated at low Mach numbers corresponding to essentially incompressible flow. This was done deliberately since all experimental and most numerical data available for comparison is in the incompressible regime. However, in order to demonstrate that the current solver can adequately handle compressibility effects we have performed one circular cylinder simulation at a higher free-stream Mach number of 0.4. The Reynolds number for this simulation is 80. Fig. 13 shows the instantaneous Mach number and normalized temperature contours computed for this flow. As the flow goes over the cylinder it accelerates and consequently the local Mach number goes up to about 0.55 which is clearly in the

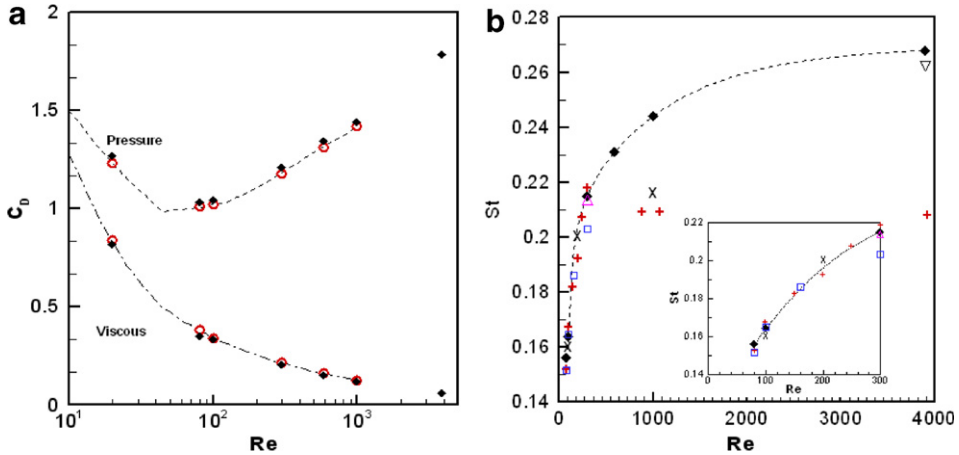

Fig. 12. Comparison of computed 2D results with established experimental and numerical data for $20 \leqslant Re_d \leqslant 3900$: (a) pressure and viscous drag coefficients versus Reynolds number for two-dimensional flow past a circular cylinder. ◆: present results, ○: Henderson [39]. (b) Strouhal number versus Reynolds number. ◆: present work, ×: Braza [42], □: Willamson [43], △: Mittal and Balachandar [37], ▽: Beaudan [19], +: Roshko [44]. Inset figure shows the details of the curve for the range $100 \leqslant Re_d \leqslant 300$.

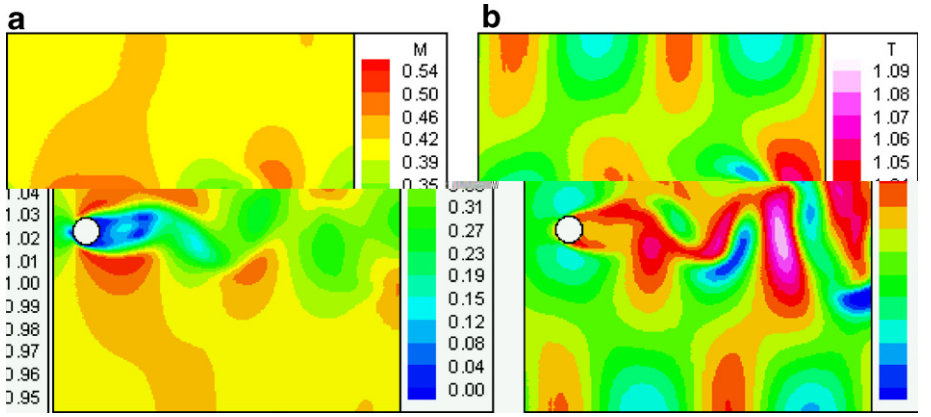

Fig. 13. (a) Instantaneous iso-Mach contour lines and (b) instantaneous iso-temperature lines at $Re_d = 80$ and $M_\infty = 0.4$.

compressible regime. Also as indicated in the figure, temperature variations in the flow are limited to about 10% of the freestream value. It should be noted that the solver in its current form is not designed for supersonic compressible flow. However, it could in principle be modified for such flows by including appropriate shock capturing schemes since the immersed boundary treatment employed here does not preclude the use of such schemes.

The final cylinder simulation presented here is a 3D simulation at $Re_d = 300$ which has been carried out on a $143 \times 97 \times 49$ Cartesian grid. The spanwise domain size was chosen to be $3d$ for this simulation and periodic boundary conditions employed in the span. The 3D simulation was initiated by using a corresponding 2D flow field and introducing a small amplitude spanwise perturbation in the flow field for a very short time. The 3D perturbations evolved naturally until a saturated three-dimensional state was reached. Fig. 14 shows isosurfaces of transverse vorticity magnitude in the wake at one time instant and flow structures similar to those in the DNS by Mittal and Balachandar [45] are observed. As was mentioned before, flow at $Re_d = 300$ is intrinsically three-dimensional, and the results from 3D simulation are expected to match the corresponding experimental values. In Table 1, the mean drag coefficient, vortex shedding Strouhal number and base pressure coefficient are compared with other numerical and experimental results and the match is found to be reasonably good.

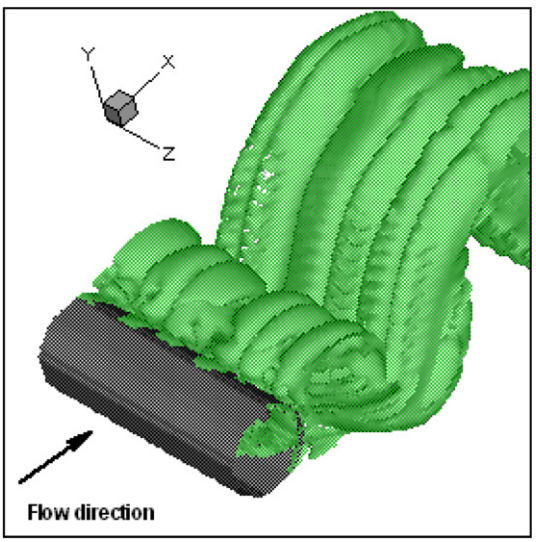

Fig. 14. Isosurface of transverse vorticity magnitude at one time instant for $Re_d = 300$ circular cylinder 3D simulation with periodic boundary conditions in the span.

Table 1
Comparison of computed C_D, St, and C_{pb} for 3D cylinder flow at $Re_d = 300$ with available numerical and experimental data

Study	$\overline{C}_D$	St	$\overline{C}_{pb}$
Current 3D simulation	1.23	0.21	−1.00
Simulation [45]	1.27	0.21	−1.04
Experiment [46]	1.22	0.20	−0.96

3.4. Airfoil and wing-tip flow simulations

One of the primary applications to be targeted with this solver is wing and rotor-tip flows. It is therefore important to assess the performance of this method for airfoil-type geometries. Simulations of such flows also allows us to demonstrate the capabilities of the solver for non-canonical geometries. We have performed 2D simulations at different angles-of-attack over the Eppler-211 airfoil at a chord-based Reynolds number (Re_c) of 60,000 and freestream Mach number of 0.2. Numerical simulations of a similar flow configuration have been carried out previously by Monttinen et al. [47] and experimental data of Althaus [48] is also available for comparison. Comparison with these data sets is used to evaluate the accuracy of the current simulations. It is useful to point out that the simulations of Mottinen et al. [47] were carried out using conventional, body-conformal structured grids.

A curvilinear 389×145 grid is used in the current simulations and we employ a weight factor (ξ) of 0.05. A relatively large domain size of $7c \times 6c$, where c is the chord length of the airfoil, is employed in order to minimize the domain confinement effects. Simulations are carried out for angles-of-attack ranging from 2° to 6° and Fig. 15(a) shows the computed flow for the 6° angle-of-attack case. The contours indicate the presence of Karman vortex shedding in the wake in addition to an incipient unsteady separation on the suction surface of the airfoil. The computed lift and drag coefficients are compared with available data of Althaus [48] and Mottinen et al. [47] in Fig. 15(b). As the plot shows there is a very good match with these two separate data sets. This favorable comparison for a non-canonical geometry at a relatively high Reynolds number further bolsters our confidence in the fidelity of the solution technique.

Finally we present results from a simulation of a wing-tip flow. The simulations have been performed for a rectangular NACA 2415 wing at 4.5° angle-of-attack, a chord-based Reynolds number of 10^5 and freestream Mach number of 0.26. We have used a QUICK weight factor (ξ) of 0.1 to control the dispersion errors at this

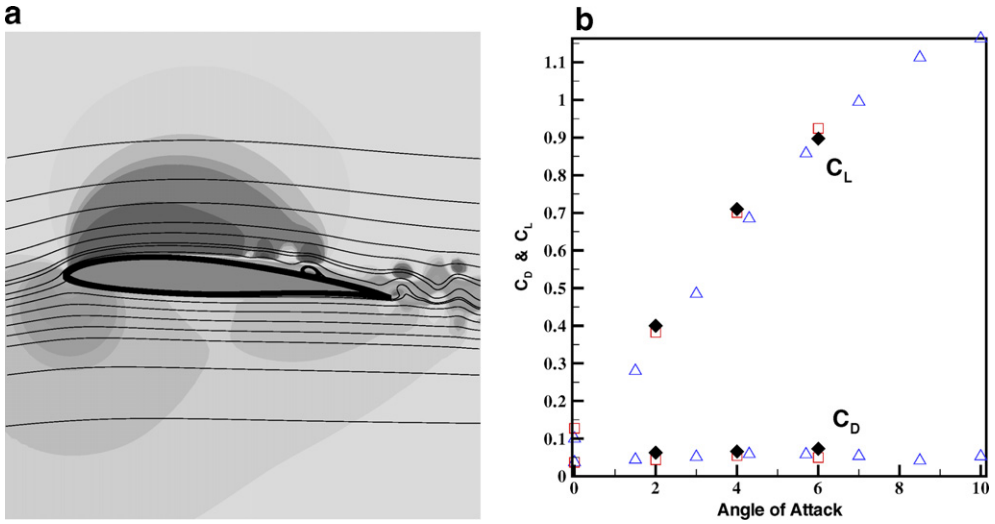

Fig. 15. Computed results for flow past Eppler-211 airfoil at $Re_c = 60,000$ and $M = 0.2$: (a) Streamwise velocity contours and streamlines for angle-of-attack of 6°. (b) The lift and drag coefficients at different angles-of-attack. ◆: present results, △: Althaus [48], □: Monttinen et al. [47].

relatively high Reynolds number. Overall the configuration chosen is based on the experiments of Martin [49] who have examined the tip-flow of a rotor in hover using particle image velocimetry (PIV). The tip Reynolds number in these experiments was 2.72×10^5 which is significantly higher than the current simulation. Furthermore, the current simulation also does not include rotational effects. The intention here is to qualitatively demonstrate the capability of the solver to simulate a realistic tip-flow and to assess its ability to resolve the vortical features of the flow. At the lower Reynolds number of 10^5, the flow in the tip-region and near wake is essentially laminar and we therefore do not need to account for turbulence effects. In an ongoing effort we are carrying out a large-eddy simulation that precisely matches the experimental conditions and we expect to validate the simulations by comparing against the experiments.

Since the airfoil does not vary in shape across the span, we use a grid that is curvilinear in the $x_1 - x_2$ plane and planar in the spanwise (x_3) direction. The overall grid employed is $460 \times 179 \times 152$ and a 2D view of the grid is shown in Fig. 16. The use of a curvilinear mesh allows better control over the grid resolution in localized regions such as boundary layers. It should be noted for instance that as shown in Fig. 16, the surface of the airfoil is nearly parallel to one set of grid lines and this allows us to provide a higher resolution selectively in the boundary layer region.

The grid in the spanwise direction although planar, is highly non-uniform with high resolution provided to the wing-tip region. In order to handle the boundary conditions over the spanwise wing-tip surface this surface is made to coincide exactly with a spanwise grid line. Subsequently we employ a simple one-dimensional ghost-cell methodology to impose the boundary conditions in the spanwise direction. This procedure is shown schematically in Fig. 17.

The simulations have been carried out on ten processors of the Pentium-4 Beowulf cluster and the results presented correspond to a non-dimensional time C/U_∞ of 1.5. The integration of the flow to this time-instant takes about 300 h on this platform. Fig. 18(a) shows an iso-surface of streamwise vorticity at one time instant and this gives a clear view of the three-dimensional vortex topology in the formation region. The plot clearly shows the presence of at least two strong vortex systems, one associated with the suction side of the wing and the other with the pressure side of the wing-tip. The tip-vortex is highly compact and extends over one chord length downstream of the trailing edge.

In order to examine the vortex structure in more detail, in Fig. 18(b) we have plotted contours of streamwise vorticity at a number of streamwise stations along the wingtip. Near the leading edge we observe the formation of two counter rotating vortices which are formed due to the leakage of flow from the wing sur-

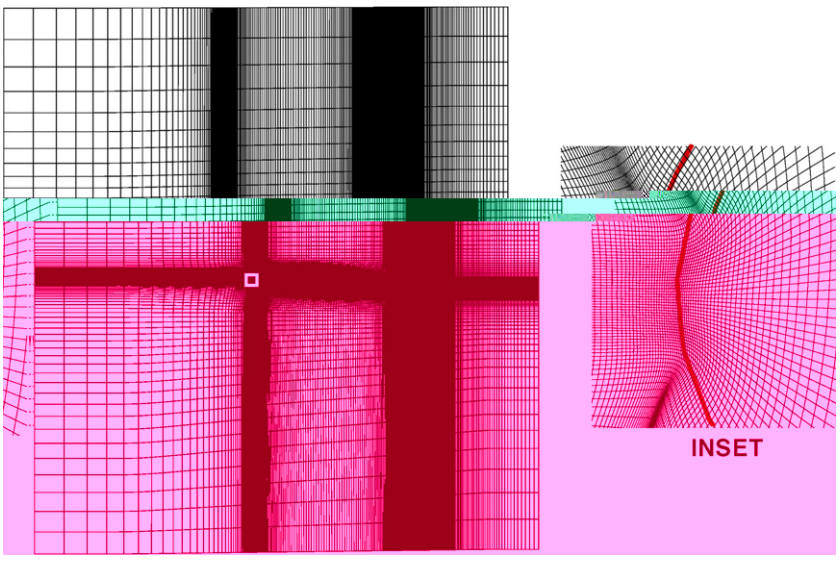

Fig. 16. 2D view of the grid employed in the NACA 2415 wing-tip simulations. The inset figure shows the grid in the vicinity of the airfoil leading-edge.

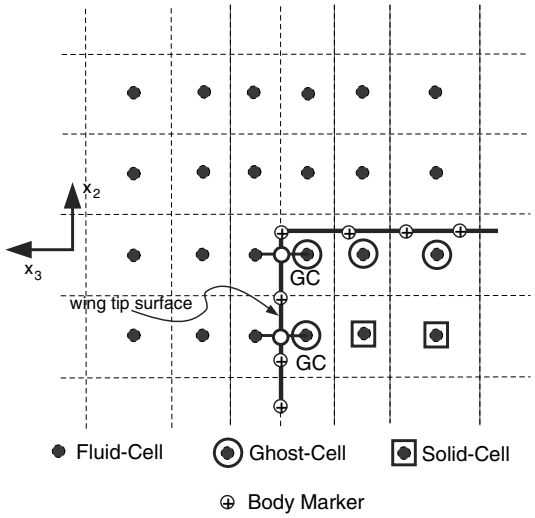

Fig. 17. Schematic showing the ghost-cell methodology employed in order to impose the boundary conditions on the spanwise wing-tip surface.

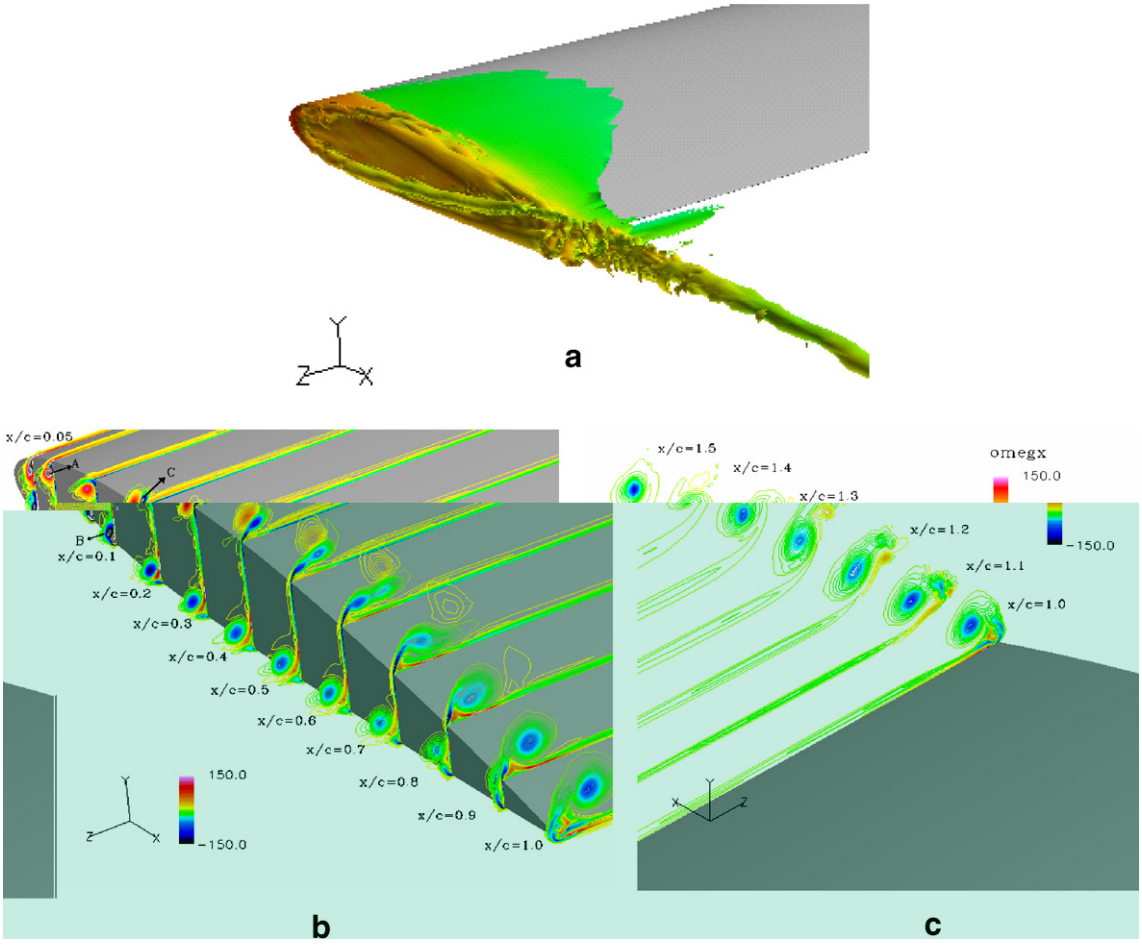

Fig. 18. Axial component of vorticity at various x/c for $Re = 100000$, $M = 0.26$, and AOA $= 4.5^{\circ}$. (a) Isosurfaces. (b) Contours in the wing-tip region and (c) near-wake region.

face to the tip region. This is due to the strong spanwise pressure gradient that is known to be present in this region. [50–53]. We denote the vortex from the suction surface as vortex-A and that from the pressure surface as vortex-B. At $x/c = 0.1$–0.2, these two vortices are observed to be of almost equal strength and have nearly circular vortex cores. At $x/c = 0.3$ we observe the development of a new vortex feature on the suction surface. This station is located where the pressure on the suction side wing surface is lower than that in the tip region. This results in the flow turning from the wing tip region back onto the suction surface. This has two consequences; first vortex-A also convects toward the suction surface and second, a new vortex (vortex-C) is created due to the roll up of the shear layer that forms as a result of the flow moving from the tip to the suction surface. Vortex-C has a rotation opposite to that of vortex-A and in fact, vortex-C is the incipient, primary wing-tip vortex. At $x/c = 0.4$, as vortex-C grows, it tends to wrap vortex-A around itself. At the same time, vortex-A starts to lose strength due to cross diffusion of vorticity with vortex-C. At this station we also see that due to the bulk flow from the wing-tip to the suction surface, vortex-B also starts to convect upwards. By the time the vortices reach at $x/c = 0.9$, vortex-C has gained significantly in strength whereas vortex-A has all but disappeared. Furthermore, vortex-B has reached the suction surface and is beginning to interact with the wing-tip vortex-C. In fact, we observe that on this plane, vortex-B creates a set of small tertiary vortex structures on the suction surface. At $x/c = 1.0$ which is at the trailing edge, the wing tip vortex-C is the dominant feature in the flow and moved significantly inwards away from the tip region. Thus, at this relatively low Reynolds number, secondary vortices play a significantly role in the formation of the wing-tip vortex. It is known that the effect of these secondary vortices diminishes at higher Reynolds numbers [54].

Fig. 18(c) shows streamwise vorticity at a number of streamwise planes in the near wake. It can be observed that in very near wake, secondary and tertiary vortices continue to interact with the primary wing-tip vortex and modify its structure. The current solver therefore allows us to examine the details of the tip-vortex formation and evolution and as mentioned before, we are currently using this solver to examine the tip-flow of a rotor in hover at a higher Reynolds number.

4. Conclusions

We have described a new finite-difference based method that allows us to simulate compressible, viscous flows with complex *stationary* immersed boundaries on body non-conformal grids. The method is based on the calculation of the variables on ghost-cells inside the body such that the boundary conditions are satisfied on the immersed boundary. There were no *ad hoc* constants introduced in this procedure and neither is any momentum forcing employed in any of the fluid cells. Consequently the method leads to a sharp representation of the immersed boundary. This method is also not subject to any stability problems associated with the interpolation scheme employed at the immersed boundary. The method is implemented on 2D, generalized curvilinear grids which provides more flexibility than Cartesian grids and allows us to apply this method to relatively high Reynolds number flows. A hybrid implicit-explicit scheme is used for temporal discretization and we also employ a novel hybrid, second-order central difference-QUICK scheme which allows us to precisely control the numerical damping.

A flexible MPI based parallel version of the solver has been developed for performing flow simulations on shared as well as distributed memory computers. The simple mesh topology and the structured nature of the underlying grid is used to develop a simple, yet effective domain decomposition strategy. The parallel performance of the solver is tested on available platforms and found to be quite good. Details of the domain decomposition strategy and some issues specific to the immersed boundary technique are described in detail.

Two and three-dimensional flow past a circular cylinder has been simulated over a range of Reynolds numbers in order to validate the approach and numerical procedure. Computed parameters such as drag coefficients, vortex shedding Strouhal numbers and base pressure coefficients are found to be in good agreement with previous experiments and simulations. A grid refinement study confirms that the current approach yields second-order global and local spatial accuracy. Also, in order to show the capability of the method for non-canonical shapes, we present some results of flow past airfoils and wings at relatively high Reynolds numbers.

Simulations of the wing-tip flow yield a number of interesting insights into the wing-tip vortex topology and these are being examined in detail in the context of a rotor-tip flow in hover.

Acknowledgment

This research was supported by the US Army Research Office under Grant No. DAAD 19-01-1-0704 monitored by Dr. T. Doligalski.

References

[1] C.S. Peskin, Flow patterns around the heart valves, J. Comput. Phys. 10 (1972) 252–271.
[2] R. Mittal, G. Iaccarino, Immersed boundary methods, Ann. Rev. Fluid Mech. 37 (2005) 239–261.
[3] D. Goldstein, R. Handler, L. Sirovich, Modeling an no-slip flow boundary with an external force field, J. Comput. Phys. 105 (1993) 354–366.
[4] D. Goldstein, R. Handler, L. Sirovich, Direct numerical simulation of turbulent flow over a modelled riblet covered surface, J. Comput. Phys. 302 (1995) 333.
[5] 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.
[6] H.S. Udaykumar, R. Mittal, W. Shyy, Computational of solid–liquid phase fronts in the sharp interface limit on fixed grids, J. Comput. Phys. 153 (1999) 535–574.
[7] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comput. Phys. 156 (1999) 209–240.
[8] 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) 30–60.
[9] E. Balaras, Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations, J. Comput. Fluids 133 (2004) 375–404.
[10] D. You, R. Mittal, M. Wang, P. Moin, Study of rotor tip-clearance flow using large eddy simulation, AIAA J. (2003) 0838.
[11] J. Kim, D. Kim, H. Choi, An immersed boundary finite-volume method for simulations of flows in complex geometries, J. Comput. Fluids 171 (2001) 132–150.
[12] F. Gibou, R.J. Fedkiw, L-T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the Poisson equation on irregular domains, J. Comput. Phys. 176 (2002) 205–227.
[13] A.S. Almgren, J.B. Bell, P. Colella, T. Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, SIAM J. Sci. Comput. 18 (5) (1997) 1243–1532.
[14] R.C. Strawn, E.P. Duque, J. Ahmad, Rotorcraft aeroacoustics computations with overset-grid CFD methods, J. Am. Helicopter Soc. 44 (2) (1999) 132–140.
[15] P. Colella, D.T. Graves, B.J. Keen, D. Modiano, A Cartesian grid embedded boundary method for hyperbolic conservation laws, J. Comput. Phys. 211 (2006) 347366.
[16] G.H. Miller, P. Colella, A conservative three-dimensional Eulerian method for coupled solid-fluid shock capturing, J. Comput. Phys. 183 (2002) 2682.
[17] A. Legay, J. Chessa, T. Belytschko, An Eulerian–Lagrangian method for fluid–structure interaction based on level sets, Comput. Methods Appl. Mech. Eng. 195 (2006) 20702087.
[18] A.D. Anderson, C.J. Tannehill, R.H. Pletcher, Computational Fluid Mechanics and Heat Transfer, Hemisphere Publishing Corporation, New York, 1984.
[19] P. Beaudan, P. Moin, Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number. Thermo-sciences Division, Department of Mechanical Engineering. TF-62, Stanford University, 1994.
[20] R. Mittal, P. Moin, Suitability of upwind biased schemes for large-eddy simulation of turbulent flows, AIAA J. 35 (8) (1997) 1415–1417.
[21] P. Tamamidis, D.N. Assanis, Evaluation of various high-order accuracy schemes with and without flux limiters, Int. J. Numer. Methods Fluids 16 (1993) 931–948.
[22] B.P. Leonard, A stable and accurate on convection modeling procedure based on quadratic upstream interpolation, Comput. Meth. Appl. Mech. Eng. 19 (1979) 59–98.
[23] B. Van Leer, Flux-vector splitting for the Euler equations, in: Proceedings of the 8th International Conference on Numerical Methods in Fluid Dynamics, Springer, Verlag, 1982.
[24] J. Steger, R.F. Warming, Flux vector splitting of inviscid gasdynamic equations with application to finite-difference methods, J. Comput. Phys. 40 (1981) 263–293.
[25] P.R. Spalart, R.D. Moser, M.M. Rogers, Spectral method for the Navier–Stokes equation with one infinite and two periodic directions, J. Comput. Phys. 96 (1991) 297–324.
[26] C.A. Kennedy, M.A. Carpenter, R.M. Lewis, Low-storage, explicit Runge–Kutta scheme for the compressible Navier–Stokes equations, ICASE Report No. 99-22, 1999.
[27] C. Canuto, A. Quarteroni, T.A. Zang, Spectral Methods in Fluid Dynamics, Springer-Verlag, 1987.

[28] H. Choi, P. Moin, J. Kim, Direct numerical simulation of turbulent flow over riblets, J. Fluid Mech. 255 (1993) 503–539.
[29] T.J. Poinsot, S.K. Lele, Boundary conditions for direct simulations of compressible viscous flows, J. Comput. Phys. 101 (1992) 104–129.
[30] Z. Li, An overview of the immersed interface method and its applications, Taiwanese J. Math. 7 (2003) 1–49.
[31] S. Majumdar, G. Iaccarino, PA. Durbin, RANS solver with adaptive structured boundary non-conforming grids, Cent. Turbul. Res. Annu. Res. Briefs (2001) 353–364.
[32] M. Bozkurttas, H. Dong, V. Seshadri, R. Mittal, F. Najjar, Towards numerical simulation of flapping foils on fixed Cartesian grids, AIAA (2005) 0081.
[33] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in Fortran, Cambridge University Press, 1992.
[34] W. Shyy, H.S. Udaykumar, M.M. Rao, R.W. Smith, Computational Fluid Dynamics with Moving Boundaries, Taylor & Francis, London, 1996.
[35] R.J. Fedkiw, Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the Ghost fluid method, J. Comput. Phys. 175 (2002) 200–224.
[36] C.H.K. Williamson, Vortex dynamics in the cylinder wake, Ann. Rev. Fluid Mech. 8 (1996) 477–539.
[37] R. Mittal, S. Balachandar, Effect of three-dimensionality on the lift and drag of nominally two-dimensional cylinder, Phys. Fluids 7 (8) (1995) 1841–1865.
[38] C.H.K. Williamson, A. Roshko, Measurement of base pressure in the wake of a cylinder at low Reynolds numbers, Z. Flugwiss. Weltraumforsch 14 (1990) 38.
[39] R.D. Henderson, Details of the drag curve near the onset of vortex shedding, J. Phys. Fluids 7 (9) (1995) 2102–2104.
[40] C. Norberg, Effects of Reynolds number and a low-intensity free-stream turbulence on the flow around a circular cylinder. Department of Applied Thermodynamics and Fluid Mechanics, Report No. 87/2, Chalmer University of Technology, Gothenburg, Sweden, 1987.
[41] S.S. Abarbanel, W.S. Don, D. Gottlieb, D.H. Rudy, J.C. Townsend, Secondary frequencies in the wake of a circular cylinder with vortex shedding, J. Fluid Mech. 225 (1991) 557–569.
[42] M. Braza, P. Chassaing, H.H. Minh, Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, J. Fluid Mech. 165 (1986) 79–130.
[43] C.H.K. Williamson, The natural and forced formation of spot-like ' vortex dislocations ' in the transition of a wake, J. Fluid Mech. 243 (1992) 393–441.
[44] A. Roshko. On the development of turbulent wakes from vortex streets. NACA Tech. Note, No. 1191, 1954.
[45] R. Mittal, S. Balachandar, On the inclusion of three-dimensional effects in simulations of two-dimensional bluff-body wake flows. In: Symposium on separated and complex flows. ASME Summer Meeting, Vancouver, 1997.
[46] H.Q. Zhang, Y. Fey, B.R. Noack, On the transition of the cylinder wake, Phys. Fluids 7 (4) (1995) 779–794.
[47] J. Monttinen, H.L. Reed, K.D. Squires, W.S. Saric, On the effect of winglets on the performance of micro-aerial-vehicles. In: Proceedings of the First ASU Symposium on Research in Engineering and Applied Sciences, Temp, AZ, 2003.
[48] D. Althaus, Profilpolaren fur den modellflug Band 2, Necar-Verlag, Villingen-Schwenningen, Germany, 1985, pp. 37–43.
[49] B.P. Martin, Measurements of the trailing vortex formation, structure, and evolution in the wake of hovering rotor. Doctoral Dissertation, University of Maryland, 2001.
[50] M.S. Francis, D.A. Kennedy, of a Trailing Vortex, J. Aircraft 16 (3) (1979) 148–154.
[51] J.S. Chow, G.G. Zilliac, P. Bradshaw, Mean and Turbulence Measurements in the Near Field of a Wingtip Vortex, AIAA J. 35 (10) (1997) 1562–1567.
[52] D.L. Bacon, Mean and turbulence measurements in the near field of a wingtip vortex, NACA Report, No. 161, 1924.
[53] M. M. Munk, Note on Vortices and Their Relation to the Lift of Airfoil, NACA TN, No. 184, 1924.
[54] D. Birch, T. Lee, F. Mokhtarian, F. Kafyeke, Rollup and near-field behavior of a tip vortex, J. Aircraft 40 (3) (2003) 603–607.