Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 210 (2005) 75–108
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

# Incompressible Navier–Stokes method with general hybrid meshes

Y. Kallinderis, H.T. Ahn *

*Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, 210 East 24th Street, W.R. Woolrich Laboratories, 1 University Station, C0600, Austin, TX 78712-1085, USA*

Received 10 January 2005; received in revised form 6 April 2005; accepted 6 April 2005
Available online 23 May 2005

**Abstract**

A new incompressible Navier–Stokes numerical method is presented, capable of utilizing general hybrid meshes containing all four types of three-dimensional elements: hexahedra, prisms, tetrahedra, and pyramids. It is an artificial compressibility type of method using dual time stepping for time accuracy. The presented algorithms for (i) spatial discretization, (ii) time integration, and (iii) parallel implementation are transparent to the different types of elements. Further, the presence of grid interfaces between the multiple types of elements does not deteriorate accuracy of the solution. Efficient evaluation of the viscous terms is addressed via a special technique that avoids multiple spatial integration of the same edge of the mesh. An upwind spatial discretization, and a central scheme with two different formulations of the artificial dissipation operator are tested with the general hybrid meshes. Use of local blocks of hexahedra is evaluated in terms of accuracy and efficiency via simulations of high Reynolds number flows. Finally, the developed methods are implemented in parallel using partitioned general hybrid meshes and an efficient parallel communication scheme to minimize CPU time.


*Keywords:* Incompressible Navier–Stokes equations; General hybrid meshes; Artificial compressibility; Dual time-stepping; High Reynolds number flows

## 1. Introduction

Numerical solution of the incompressible Navier–Stokes equations is of a great interest due to its wide range of applications. The incompressible Navier–Stokes equations can be applied to low speed

* Corresponding author.
*E-mail addresses:* kallind@veltisto.net (Y. Kallinderis), ahn@mail.utexas.edu (H.T. Ahn).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.04.002

aerodynamics, bio-fluid flows, convective heat transfer problems, and hydrodynamics. A major obstacle to numerical solution of the incompressible Navier–Stokes equations is the enforcement of the incompressibility requirement. Since there is no time-evolution term in the continuity equation, the standard time marching schemes developed for compressible flow solvers cannot be applied. The continuity equation plays the role of a constraint that the momentum equation has to satisfy.

Two main solution approaches for the incompressible Navier–Stokes equations are the pressure correction method and the artificial compressibility method [1]. In the pressure correction method, introduced by Harlow and Welch [2], the pressure Poisson equation is being solved for a guessed velocity filed at each iteration until the velocity field satisfies the continuity equation. Even though this method is matured and successfully applied to a variety of applications [3,4], the solution accuracy and performance is highly dependent on the performance of the pressure Poisson equation solver, and this can be very expensive for time accurate simulation of complex turbulent flows. The method of artificial compressibility proposed by Chorin [5] introduces a pseudo time-derivative of pressure into the continuity equation. This pseudo term changes the mathematical character of the continuity equation from elliptic to hyperbolic by introducing the artificial compressibility, and lets the system of equations be solved with a variety of time-marching schemes developed for compressible flow solvers.

The square root of the artificial compressibility parameter $\sqrt{\beta}$ represents a speed of artificial pressure wave and also affects the overall convergence rate. If $\beta$ gets higher, then the artificial pressure wave travels faster, however for an extremely fast wave speed (ideally, the wave speed goes to the infinity in incompressible media), a large disparity in eigenvalues is introduced. This degrades the overall convergence and stability. Hence an optimal $\beta$ (wave speed) can be found for a given problem.

The original form of the artificial Compressibility method was developed for steady-state problems. It was extended to time accurate formulations first by Peyret [6]. Rogers and Kwak [7,8] applied the artificial compressibility method to unsteady problems with an implicit line-relaxation procedure using the finite difference method. Belov [9] applied Jameson's dual time-stepping scheme [10] to a time accurate formulation of the artificial compressibility method. The dual time-stepping scheme basically solves a sequence of steady-state problems in pseudo-time by using well established explicit multi-stage scheme. Hence, even if the formulation is implicit in true-time, the actual time advancement is driven by the explicit multi-stage scheme in pseudo-time. Lin [11] applied this method to adaptive unstructured meshes in two dimensions. Anderson et al. [12] applied the artificial compressibility method using the flux-difference splitting scheme on 2D unstructured meshes.

Even with the successful extension of the artificial compressibility method to the unsteady problems, most of the previous work is on structured meshes [6–9] and applications on the unstructured meshes are relatively recent and limited to two dimensions [11,12]. Furthermore, the applications of the time-accurate artificial compressibility method on unstructured meshes are only with simplicial elements (triangles in two dimensions, tetrahedra in three dimensions) and no result has been reported yet using general hybrid meshes with all four types of elements in three dimensions (hexahedra, prisms, pyramids, and tetrahedra), which is the main focus of the present work.

For viscous flow simulations, the superiority of the hybrid meshes over the conventional structured or unstructured meshes with simplexes is advocated by many researchers [13,14]. The hybrid meshes can combine good viscous layer resolving capability obtained from their structured elements, with the geometric flexibility of the simplicial unstructured meshes. The merits of the hybrid meshes can be further enhanced by introducing additional element types into the conventional hybrid meshes which typically consist of prisms and tetrahedrons. For example, placing local hexahedra on the viscous and wake regions can result in significant savings in the number of elements. The inclusion of hexahedra necessitates use of pyramids as buffer elements between hexahedra and tetrahedra which complicates application of the numerical solution methods.

The current work presents one of the first artificial compressibility methods on general hybrid meshes containing all four types of elements in three dimensions. The presented algorithms for (i) spatial discretization,

(ii) time integration, and (iii) parallel implementation are transparent to the different types of elements that are present in the general hybrid mesh. Further, the presence of the grid interfaces between the different types of elements does not deteriorate accuracy of the solution. Efficient evaluation of the viscous terms is addressed via a special technique which avoids multiple spatial integrations on the edges of the mesh.

In addition, three other numerical issues are examined in the present work. The first is accuracy of the simulation with high aspect ratio cells at the wall surfaces. Such elements are typical of high Reynolds number flows considered here. The second is maintaining a relative large allowable time step size when the mesh contains small-size elements in arbitrary regions of the domain. The third issue is comparison of solutions between upwind and central schemes using hybrid meshes.

Parallel implementation of the developed flow solution method focuses on two aspects: (i) development of a general grid data structure to handle partitioned meshes containing hexahedra, prisms, tetrahedral, and pyramids, (ii) use of an appropriate parallel communication scheme to minimize CPU time.

## 2. Time accurate formulation of the artificial compressibility method

The incompressible Navier–Stokes equations are written in terms of mass and momentum conservation. The time accurate formulation of the artificial compressibility method is introduced.

### 2.1. Governing equations

The conservation laws of mass and momentum for an arbitrarily closed control volume $V$ with the boundary $S$ in integral form can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho \,\mathrm{d}V + \int_S \rho \mathbf{V}\cdot\hat{\mathbf{n}}\,\mathrm{d}S = 0, \tag{1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho \mathbf{V}\,\mathrm{d}V + \int_S \rho \mathbf{V}\mathbf{V}\cdot\hat{\mathbf{n}}\,\mathrm{d}S = \int_S \boldsymbol{\sigma}\cdot\hat{\mathbf{n}}\,\mathrm{d}S, \tag{2}$$

where $\mathbf{V}^{\mathrm{T}} = (u, v, w)$ is the velocity vector of the fluid, $\rho$ is the density of the fluid, which is a constant for incompressible flows, and $\hat{\mathbf{n}}$ is the unit normal vector to the control volume. No mesh motion or volumetric force is considered for the current study. The stress tensor $\boldsymbol{\sigma}$ is composed of the normal stress representing hydrostatic pressure $p$ and the shear stress $\tau_{ij}$:

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij}.$$

For Newtonian fluid under incompressible flow condition, the shear stress tensor is expressed as

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right),$$

where $\mu$ is dynamic viscosity, and $u_i$ is the $i$th component of the velocity vector. The conservation laws are non-dimensionalized by using the relations with the reference quantities as below,

$$x^* = \frac{x}{D}, \quad u^* = \frac{u}{U_\infty}, \quad t^* = \frac{t}{D/U_\infty}, \quad p^* = \frac{p - p_\infty}{\rho U_\infty^2},$$

where $x^*$, $u^*$, $t^*$ and $p^*$ are the non-dimensionalized scales of length, velocity, time and pressure respectively, and $D$ is characteristic length scale which is diameter of the circular cylinder, and $U_\infty$ is magnitude of the

velocity of free stream and $p_\infty$ is the pressure of free stream. Substituting the above relations into the conservation laws and dropping $*$ yield the non-dimensionalized form of the governing equations. The non-dimensionalized incompressible Navier–Stokes equations, which consist of the continuity equation and three momentum equations for each coordinate direction, are expressed as a system of equations in the integral form as shown in the following equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{U}\,\mathrm{d}V + \int_S \left(\mathbf{F}_\mathrm{I}\hat{\mathbf{i}} + \mathbf{G}_\mathrm{I}\hat{\mathbf{j}} + \mathbf{H}_\mathrm{I}\hat{\mathbf{k}}\right)\cdot\hat{\mathbf{n}}\,\mathrm{d}S = \int_S \left(\mathbf{F}_\mathrm{V}\hat{\mathbf{i}} + \mathbf{G}_\mathrm{V}\hat{\mathbf{j}} + \mathbf{H}_\mathrm{V}\hat{\mathbf{k}}\right)\cdot\hat{\mathbf{n}}\,\mathrm{d}S, \tag{3}$$

where $\mathbf{U}$ is the vector of conserved flow properties, $\mathbf{F}_\mathrm{I}\hat{\mathbf{i}} + \mathbf{G}_\mathrm{I}\hat{\mathbf{j}} + \mathbf{H}_\mathrm{I}\hat{\mathbf{k}}$ is the convective flux vector, $\mathbf{F}_\mathrm{V}\hat{\mathbf{i}} + \mathbf{G}_\mathrm{V}\hat{\mathbf{j}} + \mathbf{H}_\mathrm{V}\hat{\mathbf{k}}$ is the viscous flux vector, and $\hat{\mathbf{n}}$ is the outward unit normal vector to the control volume $V$.

Components of the vectors are defined as follows:

$$\mathbf{U} = \begin{Bmatrix} 1 \\ u \\ v \\ w \end{Bmatrix}, \quad \mathbf{F}_\mathrm{I} = \begin{Bmatrix} u \\ uu + p \\ vu \\ wu \end{Bmatrix}, \quad \mathbf{G}_\mathrm{I} = \begin{Bmatrix} v \\ uv \\ vv + p \\ wv \end{Bmatrix}, \quad \mathbf{H}_\mathrm{I} = \begin{Bmatrix} w \\ uw \\ vw \\ ww + p \end{Bmatrix},$$

$$\mathbf{F}_\mathrm{V} = \frac{1}{Re}\begin{Bmatrix} 0 \\ 2\frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \end{Bmatrix}, \quad \mathbf{G}_\mathrm{V} = \frac{1}{Re}\begin{Bmatrix} 0 \\ \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \\ 2\frac{\partial v}{\partial y} \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \end{Bmatrix}, \quad \mathbf{H}_\mathrm{V} = \frac{1}{Re}\begin{Bmatrix} 0 \\ \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \\ \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \\ 2\frac{\partial w}{\partial z} \end{Bmatrix},$$

where $\mathbf{U}$ is the vector containing the conservation variables, $\mathbf{F}_\mathrm{I}$, $\mathbf{G}_\mathrm{I}$ and $\mathbf{H}_\mathrm{I}$ represent the convective flux vectors, and $\mathbf{F}_\mathrm{V}$, $\mathbf{G}_\mathrm{V}$ and $\mathbf{H}_\mathrm{V}$ are the viscous flux vectors, and $Re = \frac{U_\infty D}{\nu}$ is the Reynolds number and $\nu$ is the kinematic viscosity of the fluid defined as $\nu = \frac{\mu}{\rho}$.

### 2.2. Spalart–Allmaras turbulence model

Spalart and Allmaras [15] introduced a one-equation model originally developed for aerodynamic applications. A single model transport equation is solved for the turbulent viscosity ($\nu_\mathrm{t}$) and it has been shown that the model is quite successful especially for drag prediction in aerodynamic applications [16].

The Spalart–Allmaras eddy viscosity transport equation in differential form is expressed as follows:

$$\frac{\partial\tilde{\nu}}{\partial t} + \frac{\partial}{\partial x_j}(\tilde{\nu}u_j) = C_{b1}\tilde{S}\tilde{\nu} + \frac{1}{\sigma}\left\{\frac{\partial}{\partial x_j}\left[(\nu + \tilde{\nu})\frac{\partial\tilde{\nu}}{\partial x_j}\right] + C_{b2}\frac{\partial\tilde{\nu}}{\partial x_j}\frac{\partial\tilde{\nu}}{\partial x_j}\right\} - C_{w1}f_w\left(\frac{\tilde{\nu}}{d}\right)^2. \tag{4}$$

The complete model equation includes trip functions which are used for modeling laminar to turbulent transition at a pre-specified point, but the current turbulent simulation assumes that the entire flow field is affected by the turbulence model. Hence, no such trip functions are used. The terms on the left-hand side of Eq. (4) are the unsteady term and the convective flux term in conservative form. The terms on the right-hand side are the production term, the diffusive flux term, and the destruction term which includes the distance $d$ to the nearest viscous wall as a variable. In order to determine the eddy viscosity, first the transport equation has to be solved for the working variable $\tilde{\nu}$ and then it is converted to eddy viscosity $\nu_\mathrm{t}$ by the following formula:

$$\nu_\mathrm{t} = f_{v1}\tilde{\nu},$$

where $f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3}$, $C_{v1} = 7.1$, $\chi = \frac{\tilde{\nu}}{\nu_\mathrm{L}}$ and $\nu_\mathrm{L}$ is the laminar kinematic viscosity. For detailed definition of the constants and functions used in the model equation, one can refer to Spalart and Allmaras [15].

### 2.3. Time-accurate formulation

The artificial compressibility method is presented in a time-accurate formulation. The original form of artificial compressibility method introduced by Chorin [5] was in the steady-state formulation where no true time-derivative term was included. Later the method was extended to time-accurate formulations applied to unsteady problems [6–9]. The present time accurate formulation was first presented by Belov [9] by using the dual time-stepping algorithm introduced by Jameson [10].

$$\mathbf{P}\frac{\mathrm{d}}{\mathrm{d}t^*}\int_V \mathbf{Q}\,\mathrm{d}V + \frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{U}\,\mathrm{d}V + \int_S \left(\mathbf{F}_\mathrm{I}\hat{\mathbf{i}} + \mathbf{G}_\mathrm{I}\hat{\mathbf{j}} + \mathbf{H}_\mathrm{I}\hat{\mathbf{k}}\right)\cdot\hat{\mathbf{n}}\,\mathrm{d}S = \int_S \left(\mathbf{F}_\mathrm{V}\hat{\mathbf{i}} + \mathbf{G}_\mathrm{V}\hat{\mathbf{j}} + \mathbf{H}_\mathrm{V}\hat{\mathbf{k}}\right)\cdot\hat{\mathbf{n}}\,\mathrm{d}S. \tag{5}$$

A pseudo time-derivative term (time derivative with respect to pseudo-time, $t^*$) is added not only to the continuity equation but also to the momentum equations. Hence, the continuity equation has a pseudo time-derivative of the pressure and the momentum equations have both true time-derivatives and the added pseudo time-derivatives of the velocity components.

The vector $\mathbf{Q}$ is containing primitive flow variables which are unknowns for the system of equations, and $\mathbf{P}$ is a diagonal matrix containing the artificial compressibility parameter $\beta$ acting as a pre-conditioner for the continuity equation.

$$\mathbf{P} = \begin{bmatrix} 1/\beta & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = \begin{Bmatrix} p \\ u \\ v \\ w \end{Bmatrix}. \tag{6}$$

The artificial compressibility parameter $\beta$ controls the speed of artificial pressure wave and also affects the overall convergence rate. Depending on the preconditioning method employed, more complex form of the preconditioning matrix can be used including variable $\beta$ which is scaled by the local flow velocity [17,18]. More recently, Malan et al. [19] included the viscous effect into the artificial compressibility parameter $\beta$. The main idea of the variable $\beta$ is minimizing the disparity of the eigenvalues by scaling the artificial speed of sound which is a function of $\beta$. However, numerous results have been reported that, in general, a constant $\beta$ is the best for the overall convergence rate [11,20]. For present study, a globally constant $\beta$ is used in the orders of $\mathrm{O}(1) \sim \mathrm{O}(100)$, and $\beta = 100$ is found to be the optimal for the overall convergence of the cases considered here.

In order to couple the turbulence model equation with the mean flow equations, the time-stepping scheme of the turbulence equation is presented in a consistent form with that of the mean flow equation. A dual time-stepping time-accurate formulation of the turbulence equation is given by

$$\frac{\partial}{\partial t^*}\int_V \tilde{\nu}\,\mathrm{d}V + R^*_{\mathrm{SA}}(\tilde{\nu}) = 0, \tag{7}$$

where $t^*$ refers to pseudo-time and $R^*_{\mathrm{SA}}$ refers to the integral form of the Spalart–Allmaras equation defined by

$$R^*_{\mathrm{SA}}(\tilde{\nu}) = \frac{\partial}{\partial t}\int_V \tilde{\nu}\,\mathrm{d}V + \oint_S \left(F^{\tilde{\nu}}_\mathrm{c} - F^{\tilde{\nu}}_\mathrm{v}\right)\mathrm{d}S - \int_V Q^{\tilde{\nu}}\,\mathrm{d}V, \tag{8}$$

where the first term in the right-hand side is the unsteady term, $F^{\tilde{\nu}}_\mathrm{c}$ and $F^{\tilde{\nu}}_\mathrm{v}$ are the convective and viscous flux vectors, and $Q^{\tilde{\nu}}$ is the source term which is including the production and destruction terms.

Once the unsteady residual is constructed by using the temporal and spatial discretization schemes, the steady-state problem in pseudo-time is solved for the eddy viscosity at the next time step. The pseudo transient problems for the turbulence model equation as expressed in Eq. (7) can be coupled with that of the

mean flow equation (5). The two sets of equations are coupled and solved simultaneously, hence the flow field and eddy viscosity are strongly coupled and converge concurrently.

Both the true time-derivatives and pseudo time-derivatives need their own time integrators. Hence the time advancement scheme of the current formulation requires the dual time-stepping scheme, which is solving a steady-state problem in pseudo-time ($t^*$) at each true time step. The discretization schemes of the true and pseudo time-derivatives are presented in Section 4.

## 3. Spatial discretization with general hybrid meshes

A conservative, finite-volume discretization scheme is used for solving the incompressible Navier–Stokes equations. A node-centered median dual volume is used for spatial discretization. An edge-based algorithm is used for the computation of the numerical fluxes [21]. For the viscous flux evaluation at an edge, another conceptual finite volume composed of its neighbor cells is used for the velocity gradient computation. A new computationally efficient algorithm is presented for the computation of the velocity gradients. This algorithm is composed of the first step of a face-wise loop to evaluate the surface integrals of edge-duals. The second step is an edge-wise operation for the final computation of velocity gradients and viscous fluxes.

The employed node-centered median dual control volume is shown in Fig. 1. The region indicated by dashed lines around node $i$ represents node-duals in two dimensions. The node-dual is constructed by connecting lines defined by edge midpoints and centroids of the cells sharing the center node $i$. The L and R in Fig. 1 represent left and right sides of the node-dual boundary assuming that the edge is directed outward with respect to the node-dual $i$. In three dimensions as depicted in Fig. 2, the node dual is constructed by connecting faces (instead of lines in two dimensions) defined by edge midpoints, cell centers and face centers sharing the common node $n$.

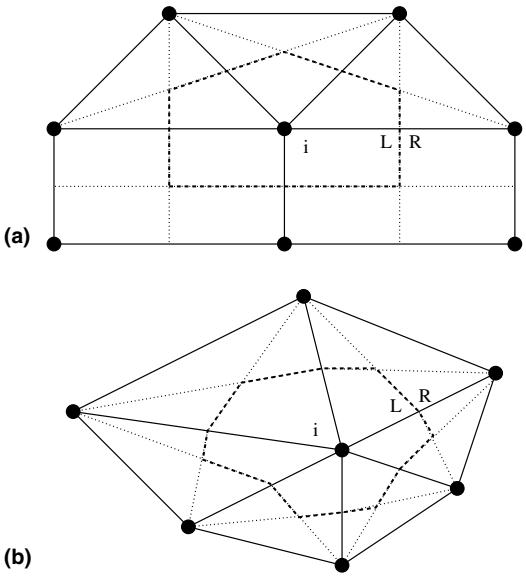


Fig. 1. Node-duals in two dimensions: (a) node-dual with mixed cells, (b) node-dual with triangular cells.

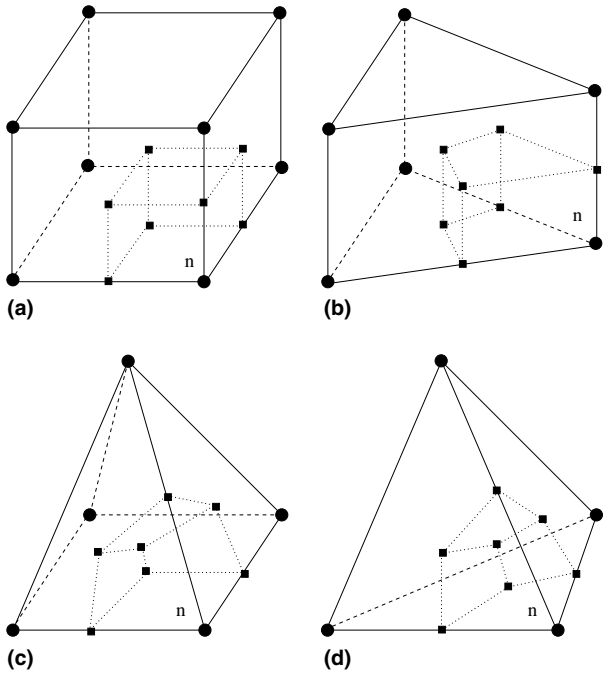


Fig. 2. Contributions to the dual volume of node $n$ from different types of elements in three dimensions: (a) hexahedron, (b) prism, (c) pyramid, and (d) tetrahedron.

### 3.1. Convective flux

The convective flux for the node-dual $i$ in discrete form can be expressed as

$$\oint_{S_i} (\mathbf{F}_\mathrm{I}\mathbf{i} + \mathbf{G}_\mathrm{I}\mathbf{j} + \mathbf{H}_\mathrm{I}\mathbf{k}) \cdot \mathbf{n}\, \mathrm{d}S \approx \sum_{j=1}^{J_i} (\mathscr{F})_j \Delta S_j = \mathbf{C}_i(\mathbf{Q}), \tag{9}$$

where $(\mathscr{F})_j = \mathbf{F}_\mathrm{I} n_x + \mathbf{G}_\mathrm{I} n_y + \mathbf{H}_\mathrm{I} n_z$ is the numerical flux evaluated at the mid-point of edge $j$, $J_i$ is the number of edges connected to node $i$ and $n_x$, $n_y$, and $n_z$ are the components of the outward unit normal vector of the node-dual boundary, and $\Delta S_j$ is the area of the node-dual boundary associated with edge $j$ . $\mathbf{C}_i(\mathbf{Q})$ is the summation of the numerical convective fluxes through the control volume boundaries.

Both the central difference and upwind schemes are used for the convective flux evaluation. For the central difference scheme, the numerical flux is evaluated at the node-dual boundary by the arithmetic averaging of the fluxes at the two end nodes as shown in Eq. (10).

$$(\mathscr{F}_{\mathrm{central}})_j = \frac{1}{2}(\mathscr{F}(\mathbf{Q}_i) + \mathscr{F}(\mathbf{Q}_j)). \tag{10}$$

Since the central difference scheme is susceptible for the odd–even mode decoupling, an additional artificial dissipation term is needed. Two different artificial dissipation models are presented and evaluated in Sections 3.3 and 3.4.

Instead of the central scheme with the supplementary artificial dissipation term, an upwind scheme can be used by evaluating the numerical fluxes using Roe's approximate Riemann solver [22] as below,

$$\mathscr{F}_{\text{upwind}} = \frac{1}{2}(\mathscr{F}(\mathbf{Q}_L) + \mathscr{F}(\mathbf{Q}_R)) + \frac{1}{2}|\hat{\mathbf{A}}(\mathbf{Q}_R, \mathbf{Q}_L)|(\mathbf{Q}_L - \mathbf{Q}_R), \tag{11}$$

where the $\mathscr{F}(\mathbf{Q}_L)$ and $\mathscr{F}(\mathbf{Q}_R)$ are the convective flux vectors from the solutions reconstructed on the left ($\mathbf{Q}_L$) and right ($\mathbf{Q}_R$) sides of control volume boundary. These solutions are reconstructed by using the Taylor series expansions about the two end nodes to the edge mid-point, which requires pre-computation of nodal gradients of the solutions. The Roe's matrix $|\hat{\mathbf{A}}|$ is defined as

$$|\hat{\mathbf{A}}| = \mathbf{R}|\hat{\Lambda}|\mathbf{R}^{-1},$$

where the $\mathbf{R}$ is the right eigenvector matrix of the flux Jacobian, $\mathbf{R}^{-1}$ is its inverse, and $|\hat{\Lambda}|$ is diagonal matrix whose components are absolute values of eigenvalues. As shown by Taylor and Whitfield [23], each component of the Roe's matrix is evaluated by arithmetic averaging of the left and right states, which satisfies the conditions for Roe's matrix. A non-singular eigensystem of the Roe's matrix was reported by Anderson et al. [12] in two dimensions, and by Kim [24] in three dimensions.

When the first order upwind scheme is employed, the solutions at the left and right sides of the control volume boundary can be chosen simply as the solutions at the two end nodes. However, for high order upwind schemes, the solutions at the control volume boundary must be reconstructed by using the Taylor series expansions about the two end nodes. The solution reconstruction from the two end nodes to the edge mid-point which represents the node-dual boundary can be expressed as

$$\mathbf{Q}_L = \mathbf{Q}_i + \frac{1}{2}(\nabla\mathbf{Q})_i \cdot \Delta\mathbf{r}_{ij}, \tag{12}$$

$$\mathbf{Q}_R = \mathbf{Q}_j - \frac{1}{2}(\nabla\mathbf{Q})_j \cdot \Delta\mathbf{r}_{ij}, \tag{13}$$

where the $\Delta\mathbf{r}_{ij}$ is the distance vector from node $i$ to $j$, and $\nabla\mathbf{Q}$ is the nodal gradient of the solution which is evaluated by least-square procedure [25]. Gram-Schmidt process is used for solving the least-square problem ($\mathbf{Ax} = \mathbf{b}$) by decomposing the coefficient matrix $\mathbf{A}$ into a product of an orthogonal matrix $\mathbf{Q}$ and an upper (right) triangular matrix $\mathbf{R}$. This Gram-Schmidt process allows pre-computation of all the weights only from the geometric information. Hence, the actual computation of the nodal gradients using the least-square procedure can be implemented just by a single loop over edges. For more details, readers can refer to Anderson and Bonhaus [25] for two dimensions and Haselbacher and Blazek [26] for three dimensions.

The surface integral in discrete form as expressed in Eq. (9) is presented in the node-wise fashion by visiting each node and accumulating the contributions from edges sharing the node. However, the actual implementation is in edge-wise manner by visiting each edge only once, computing the flux contribution, and sending positive contribution to the node inside and negative contribution to the node outside depending on the direction of the edge.

Since this convective flux computation is purely in edge-wise fashion, all the nodes get the complete contribution of the convective flux after a single edge-loop. This edge-wise algorithm does not require any information about cell topology, so the algorithm is suitable especially for the general hybrid meshes considered here.

### 3.2. Viscous flux

In order to evaluate the viscous fluxes through the control volume boundaries, the gradients of velocity components are needed to be pre-computed at each edge. For this velocity gradient computation, another conceptual finite volume called edge-dual is constructed. The edge-dual is composed of the neighbor cells

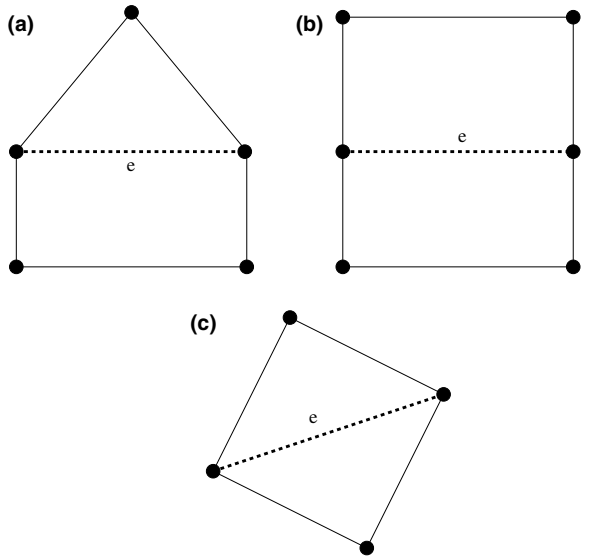


Fig. 3. Edge-duals in two dimensions for computation of the first order spatial derivatives.

sharing a common edge. Various kinds of edge-duals encountered in hybrid meshes are delineated in Fig. 3 for two dimensions and in Fig. 4 for three dimensions.

For the computation of the velocity gradients at each edge, the divergence theorem is used and the surface integrals along edge-dual boundaries are performed. For example, computing volume averaged value of $\frac{\partial u}{\partial x}$ using an edge-dual can be expressed as

$$\left(\frac{\partial u}{\partial x}\right)_e = \frac{1}{V_e} \oint_e u n_x \, dS, \tag{14}$$

where $V_e$ is the volume of an edge-dual .

This edge-dual approach has been advocated by other researchers [3,27] because it is less susceptible to the solution wiggles and yields more compact stencils than other methods. However, visiting edge-duals and performing surface integrals over various kinds of edge-duals introduces extra complexity.

This surface integral over the edge-duals as expressed in Eq. (14) (visiting an edge-dual and performing the surface integral) has two major drawbacks. First, it needs additional connectivity information of edge-to-faces for each edge-dual. In two dimensions, an edge-dual is composed of four to six edges (Fig. 3), so each edge has to store its edge-dual connectivity information of four to six integers of memory. In three dimensions, this edge-to-faces connectivity information reaches up to 18 integers (faces) for an edge in prismatic region and 16 integers (faces) for an edge in hexahedral region. Second, probably more serious than the first, it requires redundant visits of each element face regarding the evaluation of the surface integral, because a surface typically belongs to multiple edge-duals. In two dimensions, the number of visits of an edge is four if the edge is in triangular region, five if the edge is on the interface of triangular and quadrilateral regions, and six if the edge is in quadrilateral region. These redundant visits of a face become more severe in three dimensions. For a face in hexahedral region, the number of visits of the face is approximately 16, which can be very expensive.

A new computationally efficient algorithm is now presented for the velocity gradient computation using the edge-duals. In the algorithm, as depicted in Fig. 5 in two dimensions, the surface integrals over the edge-duals are performed in two steps. At the first step, cell-wise surface integrals are computed via a

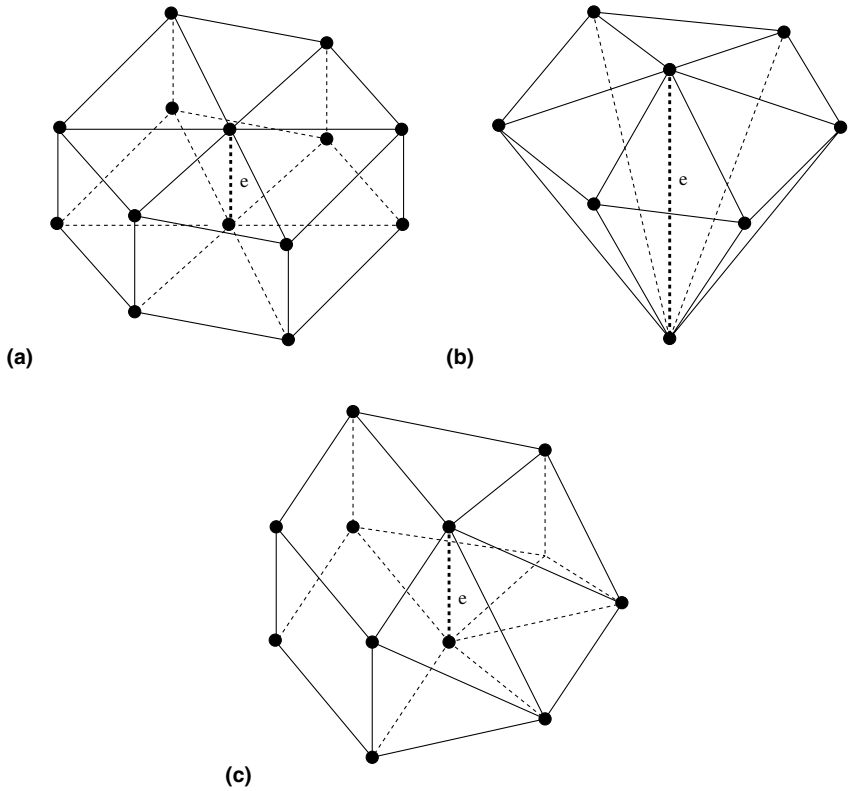


Fig. 4. Edge-duals in three dimensions for computation of the first order spatial derivatives: (a) edge-dual composed of prism, (b) edge-dual composed of tetrahedra, (c) edge-dual composed of mixed elements.

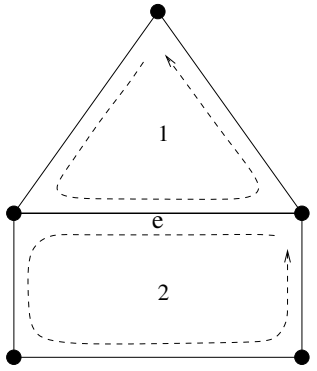


Fig. 5. Cell-wise surface integrals for velocity gradient computations using edge-duals.

single edge-loop. At the second step, the cell-wise surface integrals are gathered at the common edge and divided by the sum of the volumes of neighbor cells which compose the edge-dual volume. The cell-wise surface integrals are delineated in Fig. 5, and final computation of $\frac{\partial u}{\partial x}$ at the common edge is presented in Eq. (15).

$$\left(\frac{\partial u}{\partial x}\right)_e = \frac{\oint_1 un_x \, \mathrm{d}S + \oint_2 un_x \, \mathrm{d}S}{V_1 + V_2}, \tag{15}$$

where $V_1$ and $V_2$ are the volumes of triangular and quadrilateral cells in Fig. 5. This cell-wise surface integrals which are represented with dashed lines in Fig. 5 can be implemented via a single edge-loop. In three dimensions, this edge-loop is replaced with a face-loop because cells in three dimensions are delimited with faces. At the second loop of edges, the pre-computed surface integrals from the neighbor cells are gathered at the common edge. By the summation of the surface integrals from neighbor cells, the contributions from the common faces (edges in two dimensions) which are inside the edge-dual automatically cancel out, and the summation recovers the net surface integral of edge-dual boundary. Finally, the division of the surface integral by the edge-dual volume yields the edge-dual averaged velocity gradients.

The edge-dual approach can be interpreted as the computation of velocity gradients by volume weighted averaging of velocity gradients at the neighbor cells as shown in the following equation:

$$\left(\frac{\partial u}{\partial x}\right)_e = \left(\frac{1}{V_1}\oint_1 un_x \, \mathrm{d}S\right) w_1 + \left(\frac{1}{V_2}\oint_2 un_x \, \mathrm{d}S\right) w_2, \tag{16}$$

where $w_1 = \frac{V_1}{V_1+V_2}$ and $w_2 = \frac{V_2}{V_1+V_2}$ are the weights for the neighbor cells. In the end, all the velocity gradient computations expressed in Eqs. (14)–(16) give identical result.

Since each cell has to store these surface integrals from the first loop, an extra memory space is needed for each cell. This extra cell-wise memory space is for four real variables in two dimensions and nine real variables in three dimensions, and each of them corresponds to a velocity gradient. Additionally, for the second loop of edges, connectivity information of *edge-to-(neighbor) cells* should be pre-constructed for every edge. This *edge-to-cells* information requires the memory space of two integers in two dimensions (two neighbor cells for an interior edge in 2D). In three dimensions, the memory space requirement of *edge-to-cells* information is not fixed and varies largely depending on the types of neighbor cells. Overall, the major saving from the proposed algorithm comes from the CPU time, and the memory requirement of the new algorithm of Eq. (15) is comparable to the conventional method of Eq. (14).

Once the velocity gradients are computed at all edges, the same edge-wise operation is used for the viscous flux evaluation at node $i$ as shown in the following equation:

$$\oint_{S_i} (\mathbf{F}_\mathrm{V}\mathbf{i} + \mathbf{G}_\mathrm{V}\mathbf{j} + \mathbf{H}_\mathrm{V}\mathbf{k}) \cdot \mathbf{n} \, \mathrm{d}S \approx \sum_{j=1}^{J_i} (\mathscr{F}_\mathrm{V})_j \Delta S_j = \mathbf{V}_i(\mathbf{Q}), \tag{17}$$

where $(\mathscr{F}_\mathrm{V})_j = \mathbf{F}_\mathrm{V} n_x + \mathbf{G}_\mathrm{V} n_y + \mathbf{H}_\mathrm{V} n_z$ is the viscous flux evaluated at the edge $j$, $J_i$ is number of edges connected to the node $i$, $\Delta S_j$ is the area of node-dual boundary associated with the $j$th edge, and $\mathbf{V}_i(\mathbf{Q})$ is the final contribution of the viscous fluxes at node $i$. This edge-wise operation of viscous flux evaluation can be combined with the second step of the algorithm which evaluates the velocity gradients.

### 3.3. Artificial dissipation

Central difference schemes allow high frequency oscillations of the solution. In order to suppress this high frequency oscillation, a special dissipation term is added.

The conventional fourth-order difference scalar smoothing for a node $i$ can be expressed as below

$$\mathbf{D}_i(\mathbf{Q}) = -\sum_{j=1}^{J_i} \sigma_4 \rho_{ij} (\delta_{xx}\mathbf{Q}_j - \delta_{xx}\mathbf{Q}_i), \tag{18}$$

where $J_i$ is the number of nodes connected to the node $i$ by its neighbor edges, $\sigma_4$ is the coefficient for fourth-order scalar dissipation which is determined by empiricism, $\rho_{ij}$ is the estimate of spectral radius associated with the $j$th neighbor edge, and $\delta_{xx}$ is the undivided second order difference operator defined as below

$$\delta_{xx}\mathbf{Q}_i = \sum_{j=1}^{J_i}(\mathbf{Q}_j - \mathbf{Q}_i). \tag{19}$$

The estimate of spectral radius scaled with the node-dual boundary associated with the edge $j$ is defined as

$$\rho_{ij} = (|u_n| + c)\Delta S_j, \tag{20}$$

where the velocity normal to the node dual boundary is

$$u_n = un_x + vn_y + wn_z$$

and artificial speed of sound defined as below

$$c = \sqrt{u_n^2 + \beta}.$$

The spectral radius at each edge is scaled with the area of node-dual boundary $\Delta S_j$ associated with the edge $j$, and the normal velocity $u_n$ is associated with the direction normal to the node-dual boundary associated with the $j$th edge. This scaling is often called individual eigenvalue scaling [28] and has been reported that it is adequate for the meshes with aspect ratios up to 10 [29]. It is found that, for the present 2D simulations, the current individual eigenvalue scaling shown in Eq. (20) can be used with cells of aspect ratios at least up to 50. The artificial dissipation coefficient of $\sigma_4 = 0.005$ seems adequate for the 2D simulations.

By scaling the spectral radius with $\Delta S_j$, the actual contribution of artificial dissipation is dimensionally consistent with the surface integrals of convective and viscous flux terms. Ideally, on a uniform mesh, the final contribution of artificial dissipation when it is divided by the node-dual volume is

$$\frac{1}{V_i}\mathbf{D}_i(\mathbf{Q}) \sim \sigma_4 \mathrm{O}(\Delta x)^3 \nabla^4 \mathbf{Q}.$$

Therefore, the added artificial dissipation which is a third-order term should not affect the order of spatial accuracy of the central difference scheme which is second. However, on meshes with high aspect ratios, the artificial dissipation may not be scaled properly [28–31]. Furthermore, a severe oscillation on hybrid meshes has been reported by Haselbacher and Blazek [26] when the explicit Runge–Kutta scheme is used in conjunction with the central scheme. Haselbacher and Blazek proposed a modified artificial dissipation scheme as expressed below

$$\mathbf{D}_i(\mathbf{Q}) = -\sum_{j=1}^{J_i}\sigma_4\rho_{ij}(\mathbf{Q}_\mathrm{L} - \mathbf{Q}_\mathrm{R}) \tag{21}$$

based on the difference of the reconstructed solutions on the left and the right side of control volume boundary ($\mathbf{Q}_\mathrm{L}$ and $\mathbf{Q}_\mathrm{R}$). A Taylor series expansion is used for the solution reconstruction at the node-dual boundary as presented in Eqs. (12) and (13). Note that the artificial dissipation model expressed in Eq. (21) results in a second-order accurate dissipation scheme. If the left and right state is simply chosen as the nodal state on either side of the boundary, instead of using the Taylor series expansions, the above dissipation scheme gives a first-order accurate scheme.

### 3.4. Comparisons of dissipation models with a general hybrid mesh

Central differences with the two different artificial dissipation models as well as Roe's upwind scheme are tested for the investigation of their applicability to the general hybrid meshes.

#### 3.4.1. Generation of general hybrid meshes

The hybrid grid generator consists of two major parts: (1) the prisms/hexahedra generator, which is an algebraic, marching-type technique, and (2) the tetrahedra generator which is an advancing front type of method. An unstructured triangular/quadrilateral grid is employed as the starting surface to generate a prismatic/hexahedral mesh. This grid, covering the body surface, is marched away from the body in distinct steps, resulting in generation of semi-structured prismatic layers as well as hexahedra in the marching direction. The rest of the domain is covered with tetrahedra. Finally, pyramids are employed at the interfaces between the hexahedral and tetrahedral regions. Details of the two technique can be found in [13].

#### 3.4.2. Performance comparison of dissipation schemes

The general hybrid mesh used for the current comparison of artificial dissipation models is displayed in Fig. 6. It includes local hexahedra on the frontal viscous region of the cylinder, prisms in its rear half, prisms at the interface between the hexahedral and tetrahedral regions, and tetrahedrons for the rest.

The comparison between the central difference schemes with the conventional artificial dissipation and the modified artificial dissipation based on the solution reconstructions is presented. The second order upwind scheme result is also included. The pressure contours obtained by using the aforementioned schemes are displayed in Fig. 7.

First of all, the conventional dissipation model shows severe oscillation in the pressure contours. Increasing the coefficient ($\sigma_4$) for the dissipation model cannot be a remedy of this problem, because a larger $\sigma_4$

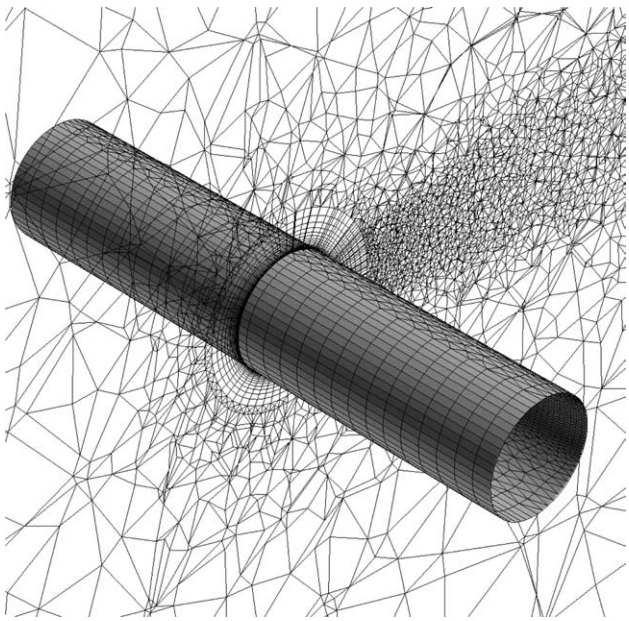

Fig. 6. General hybrid mesh-1 (GHM-1) containing hexahedra in the frontal region of the cylinder.

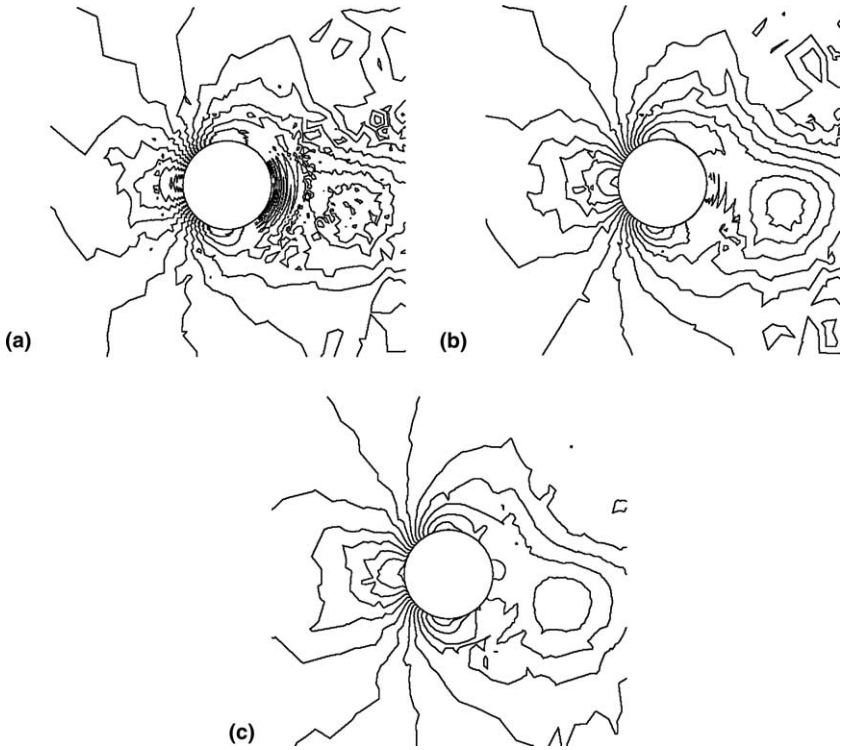


Fig. 7. Comparison of pressure contours obtained with different dissipation models: (a) corresponds to the conventional scalar dissipation model ($\sigma_4 = 0.002$), (b) to the modified dissipation model of Haselbacher and Blazek, and (c) to the implicit dissipation by upwind scheme. The pressure contours on the plane cuts are obtained at the same time within a shedding cycle. $Re = 150$.

only resulted in the decay of vortex shedding without suppressing of the oscillations in pressure field. This indicates that the conventional dissipation scheme is incapable of suppressing the solution oscillations on general hybrid meshes. Furthermore, as pointed by Haselbacher and Blazek [26], this smoothing model shows a severe stability problem on hybrid meshes when it is driven by the explicit Runge–Kutta time-stepping scheme which is used in the current study.

The modified smoothing model of Haselbacher and Blazek [26] based on the differences of the reconstructed solutions (Eq. (21)) is tested, and then compared with the second order upwind scheme by using Roe's flux-difference splitting. The Haselbacher and Blazek's scheme shows superior ability of suppressing the oscillations to the conventional dissipation scheme. The pressure contours are similar to the ones yielded by the upwind scheme.

The effect of artificial dissipation models on skin friction distribution on the surface of the cylinder is presented in Fig. 8. In a similar fashion as the pressure distribution, the conventional smoothing shows oscillations in $C_f$ in the rear half of the cylinder, where Haselbacher and Blazek's scheme and upwind scheme show smooth variation of it.

The computational cost of the modified smoothing is slightly higher than the conventional smoothing due to the solution reconstruction step for each edge. However, the modified smoothing is still less expensive than the upwind scheme, because the upwind scheme requires the evaluation of Roe's matrix at each edge.

In general, in the regions of highly stretched cells and anisotropic support of the edges (e.g., along the interfaces of prisms/hexahedra and tetrahedrons/pyramids, where only a single edge resides inside of the

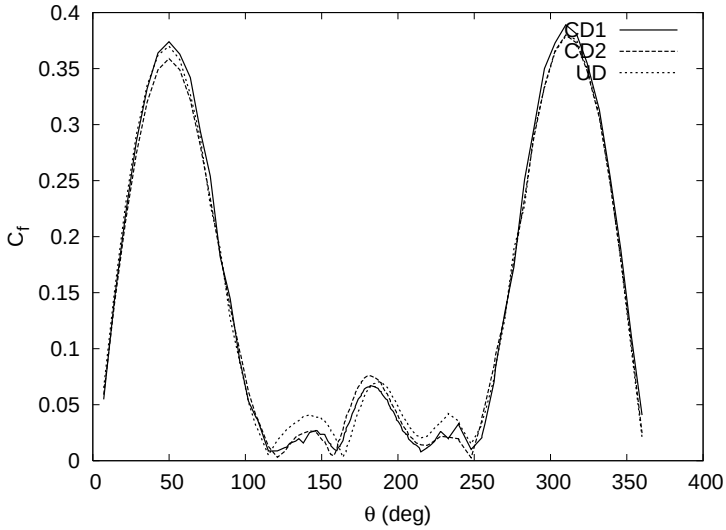


Fig. 8. Effects of dissipation schemes on the skin friction coefficient ($C_f$) distribution along the circumferential direction of the cylinder surface. $\theta$ goes from the stagnation point along the clock-wise direction. CD1 represents central difference scheme with conventional scalar smoothing, CD2 is for central difference with modified smoothing, and UD is for the second order upwind by Roe's flux-difference splitting. CD1 shows oscillations in $C_f$ in the rear half of the cylinder, where the CD2 and UD show smooth variation of it.

prismatic/hexahedral region), the Haselbacher and Balzek dissipation model and upwind scheme yield more smooth pressure contours than the conventional scalar dissipation model. Hence, for general hybrid meshes, the central difference scheme with the modified dissipation model of Haselbacher and Blazek or upwind schemes are preferred to the conventional scalar dissipation model.

### 3.5. Boundary conditions

On the viscous wall, which is the surface of the cylinder, no slip condition is specified and pressure is extrapolated from the domain such that the gradient of pressure along the normal direction is zero.

There is only one out-going characteristic artificial wave at the inflow. Typically, pressure is chosen to be extrapolated for accounting this out-going wave. All the velocity components are specified with the free stream values for the rest of the in-coming characteristic artificial waves. At the outlet, all the velocity components are extrapolated for the three out-going waves and only the pressure is specified as the free stream quantity for the one in-coming artificial wave.

All the boundary condition application routines can be implemented either by Neumann type or by Dirichlet type. The Dirichlet type boundary just needs node-wise specification of appropriate values. For the Neumann type boundary condition, the actual implementation can be either node-wise or face-wise. For the current hybrid mesh, both of the methods (node-wise and face-wise) are used in order to take advantage of the geometric characteristics of the hybrid mesh. On the viscous wall, typically very thin layers of hexahedral or prismatic cells are located with good normality condition. Therefore, node-wise extrapolation scheme is the best on the viscous wall. On the boundaries other than the viscous wall, no such cells are located with good normality condition. Hence, the face-wise extrapolation is used. The face-wise application is composed of two steps. First, a cell-averaged value is computed, and then at the second step the cell averaged value is extrapolated to each boundary nodes by using the inverse-distance weighting.

## 4. Temporal discretization of the time accurate artificial compressibility method

### 4.1. Dual time-stepping scheme

The present time-accurate formulation of the artificial compressibility method includes two different time evolution terms: one in true (physical) time $t$ and the other in pseudo-time $t^*$ as indicated in Eq. (5). The true time-derivative term is discretized by using a second order backward difference formula, while a Runge–Kutta multistage scheme is employed for the pseudo time-derivative term. For each marching step in true time, the pseudo time marching problem is solved to convergence.

Evaluating all the flux contributions to the finite control volume $V_i$ and adding the supplementary artificial dissipation term, if a central difference scheme is employed, yields the dual time-stepping time accurate formulation as

For node $i$, Eq. (5) is written as follows:

$$\mathbf{P}\frac{\mathrm{d}}{\mathrm{d}t^*}(\mathbf{Q}_i V_i) + \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{U}_i V_i) + \mathbf{C}_i(\mathbf{Q}) = \mathbf{V}_i(\mathbf{Q}) + \mathbf{D}_i(\mathbf{Q}), \tag{22}$$

where $\mathbf{P}$ is the preconditioning matrix previously defined in Eq. (6), $\mathbf{C}_i(\mathbf{Q})$ is the convective flux, $\mathbf{V}_i(\mathbf{Q})$ is the viscous flux, and $\mathbf{D}_i(\mathbf{Q})$ is the added artificial dissipation for the case of the central spatial discretization. The system of equations can be further written as

$$\frac{\mathrm{d}}{\mathrm{d}t^*}(\mathbf{Q}_i V_i) + \mathbf{P}^{-1}\mathbf{R}_i^*(\mathbf{Q}) = 0 \tag{23}$$

by introducing the unsteady residual $\mathbf{R}_i^*(\mathbf{Q})$ as defined below

$$\mathbf{R}_i^*(\mathbf{Q}) = \frac{3}{2\Delta t}\left(\mathbf{U}_i^{n+1}V_i^{n+1}\right) - \frac{2}{\Delta t}\left(\mathbf{U}_i^{n}V_i^{n}\right) + \frac{1}{2\Delta t}\left(\mathbf{U}_i^{n-1}V_i^{n-1}\right) + \mathbf{R}_i(\mathbf{Q}).$$

The unsteady residual $\mathbf{R}_i^*(\mathbf{Q})$ is the sum of the true time-derivative term discretized by using the second order backward difference formula, and the steady residual which includes all the flux terms, as shown below

$$\mathbf{R}_i(\mathbf{Q}) = \mathbf{C}_i(\mathbf{Q}) - \mathbf{V}_i(\mathbf{Q}) - \mathbf{D}_i(\mathbf{Q}).$$

In order to advance a time step from the current time $t_n$ to the next time $t_{n+1}$, the unsteady residual $\mathbf{R}^*(\mathbf{Q})$ is first constructed by discretizing the true time-derivative with the implicit backward difference formula, and then the steady-state problem shown in Eq. (23) is solved in pseudo-time. Once the steady-state in pseudo-time is reached, the solution has been advanced to the next time step.

The dual time-stepping scheme is driven by the time integration scheme for the pseudo steady-state problem. Therefore, the overall performance of the dual time-stepping scheme is highly dependent on the efficiency of the steady-state solver in pseudo-time.

A 5-stage Runge–Kutta scheme is used for solving the steady-state problem in pseudo-time

$$\begin{aligned}
\mathbf{Q}^{(0)} &= \mathbf{Q}^{k} \\
&\vdots \\
\mathbf{Q}^{(l)} &= \mathbf{Q}^{(0)} - \alpha_l \Delta t_i^* \tfrac{1}{V_i}\mathbf{P}^{-1}\mathbf{R}_i^*(\mathbf{Q}^{(l-1)}) \\
&\vdots \\
\mathbf{Q}^{k+1} &= \mathbf{Q}^{(5)}
\end{aligned} \tag{24}$$

where $\alpha_1, \ldots, \alpha_5$ are optimized coefficients for accelerating the convergence to the steady state, and $\Delta t_i^*$ is local pseudo-time step for node $i$. The marching from stage 1 to stage $k+1$ will be called subiteration in

the following. If the central scheme is employed, the diffusive fluxes (viscous flux and artificial dissipation) are evaluated only at the odd stages, and combinations of diffusive terms at previous stages are used at even stages. This multistage scheme is referred to as hybrid multi-stage compared to the scheme of evaluating all the fluxes at every stage. A more detailed description of the hybrid multistage scheme for central differences is presented in [29]. For the upwind scheme, all the fluxes are evaluated at every stage.

The dual-time stepping scheme has two different time steps: true time step and pseudo-time step. The true time-stepping is discretized by using the $A$-stable second order backward difference scheme. This scheme is stable regardless of the time step size. For pseudo time-stepping scheme, which is driven by an explicit multistage scheme, needs a time step calculation formula. Current local-pseudo time step is calculated by using both convective limit and diffusion limit which is originally proposed by Kallinderis [32].

The local-pseudo time step for the node $i$ is presented as

$$\Delta t_i^* = \omega \frac{V_i}{A_x + A_y + A_z + D}, \tag{25}$$

where

$$A_x = (|u| + c_x)S_x, \quad A_y = (|v| + c_y)S_y, \quad A_z = (|w| + c_z)S_z$$

and

$$D = 2\frac{1}{Re}\frac{V_i}{S_x + S_y + S_z}.$$

The artificial speeds of sound in each coordinate direction are

$$c_x = \sqrt{u^2 + \beta}, \quad c_y = \sqrt{v^2 + \beta}, \quad c_z = \sqrt{w^2 + \beta}$$

and the projected areas of node-dual volume are given as

$$S_x = \frac{1}{2}\sum_e |S_x|_e, \quad S_y = \frac{1}{2}\sum_e |S_y|_e, \quad S_z = \frac{1}{2}\sum_e |S_z|_e,$$

where $S_x$, $S_y$, and $S_z$ are the components of area normal vector.

The weighting factor $\omega$ may be considered as the local Courant number of the CFL (Courant, Friedrichs, and Lewy) condition, and a value of 1–3 is used.

Small size cells can be encountered not only in the viscous region but also in the rest of the domain. These small cells may inhibit the overall convergence of the pseudo-time marching, as well as of the true time procedure for cases of explicit scheme. The current time-marching scheme is tested on a mesh containing small size cells randomly located. Fig. 9 shows the comparison of the original mesh and locally redistributed mesh, where small cells appear in the wake region.

A comparison of the two different mesh simulations, which are obtained by using exactly the same number of sub-iteration, is presented in Fig. 10. As indicated in Fig. 10, these small cells in the wake region deteriorate the convergence at each time step, and this results in the growing phase error in the drag and lift responses. This tells that the current dual-time stepping scheme is very sensitive to the pseudo-time stepping, and the temporal accuracy can be assured only if the pseudo-transient problem is converged under adequate tolerance.

## 5. Time step and mesh convergence

Temporal and spatial accuracies are verified by using time step and mesh refinement. Formal second order accuracy of the three point backward difference formula is verified by time step refinement study. Due

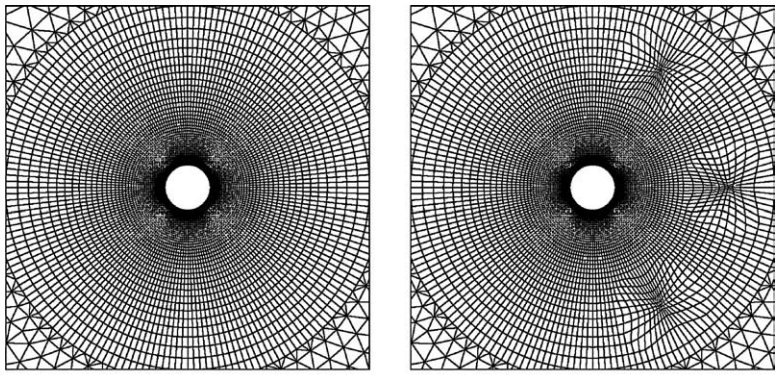


Fig. 9. Two-dimensional hybrid meshes with and without small cells in the wake region: (a) for the mesh with regular cells and (b) for the mesh containing locally in three places.

to the computational cost of the three-dimensional simulations, the time step refinement study is performed in two dimensions with the same time integration scheme as in three dimensions. The mesh convergence study is performed in three dimensions by using a set of three successively refined meshes for flows around a sphere.

### 5.1. Time step convergence study

Three successively halved time steps ($\Delta t = 0.4, 0.2, 0.1$) are employed and the resulting shedding periods are compared. The two-dimensional hybrid mesh used for this time step refinement study is displayed in Fig. 9(a). The $C_D$ and $C_L$ histories with $Re = 150$ corresponding to the three different time steps are displayed in Fig. 11.

The shedding periods averaged over the last three shedding cycles are listed in Table 1 along with the corresponding time steps used. The order of convergence for the shedding period is estimated as shown in Eq. (26).

$$\log_2\left(\frac{(T)_{\Delta t=0.4} - (T)_{\Delta t=0.2}}{(T)_{\Delta t=0.2} - (T)_{\Delta t=0.1}}\right) = 2.0. \tag{26}$$

The result of the current simulation is compared with other reported computational and experimental results. The hybrid mesh shown in Fig. 9(a) is used for this comparison study with time step of $\Delta t = 0.1$. The $C_D$, $C_L$, and $St$ are chosen for the parameters of comparison and the details are presented in Table 2.

The results of Belov [9] and Lin [11] are obtained with a computation based on the artificial compressibility method in two dimensions. Belov used a structured polar mesh ($257 \times 257$ nodes), and Lin used an unstructured mesh only with triangles (128 nodes on wall, and 42,200 nodes total). Overall, the current simulation result agrees well with the compared computational and experimental results, and small variances in the results may be attributed to several factors, such as the different types of mesh and/or the amount of artificial dissipation introduced.

### 5.2. Mesh convergence study

A set of three successively refined meshes around a sphere are constructed for the mesh convergence study. A triangular surface mesh over the sphere is first generated and the surface mesh is extruded along

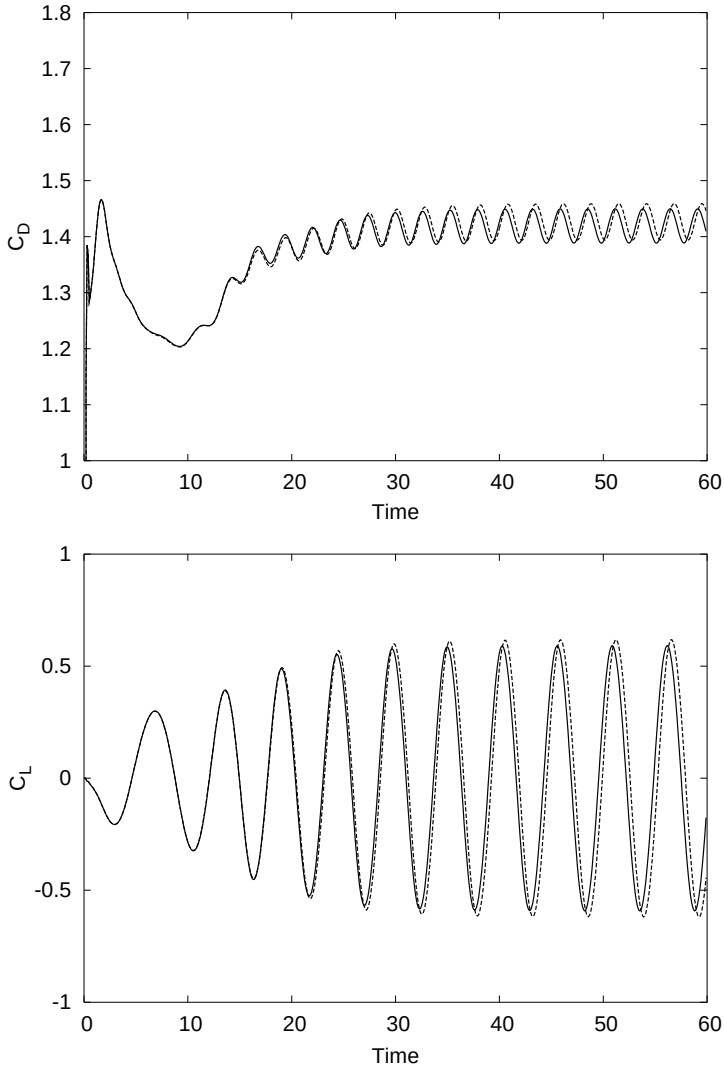


Fig. 10. Small cells effect on the global solution behavior. The comparison of $C_D$ and $C_L$ histories for the meshes with and without small cells. Solid lines are for the mesh without small cells, and dashed lines are for the mesh with small cells in the wake region. The locally small size cells do not reduce the global time step size. The growing phase difference is due to the tolerance used for the convergence of the pseudo-transient problem.

the radial directions with a uniform stretching of 20% creating prismatic cells. The outer boundary is located about 17 diameters away from the center of the sphere. In order to get a one-step refined mesh, each prismatic cell is subdivided into eight prisms (each surface triangle divided into four surface triangles and each radial edge is divided into two edges). This procedure is repeated twice to construct three levels of sphere mesh, as display in Fig. 12. The detailed characteristics of the resulting meshes are summarized in Table 3.

Mesh convergence tests are performed in two different ways; first we compute two representative differential operations using a prescribed analytic velocity field and second we measure the drag coefficients from the flow simulations around the sphere using a Reynolds number of 100.

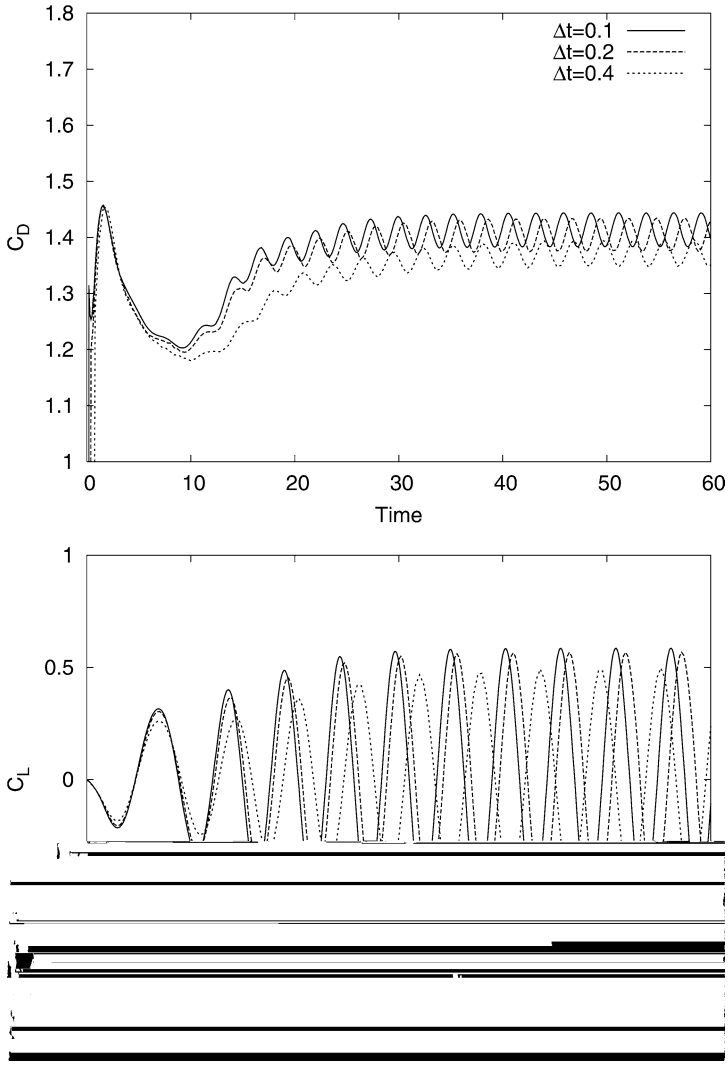


Fig. 11. Time step refinement study in two dimensions. Three-point backward difference formula is used for the true time-stepping. Flow around a circular cylinder with $Re = 150$.

Table 1
Shedding periods averaged over the last three cycles

| True time step $\Delta t$ | Shedding period, $T$ |
|---|---|
| 0.1 | 5.30 |
| 0.2 | 5.40 |
| 0.4 | 5.80 |

Flow around a circular cylinder with $Re = 150$.

### 5.2.1. Analytic velocity function test

The divergence operator is chosen to mimic the convective flux computation and the laplacian operator is chosen for emulating the viscous flux evaluation. The numerical and exact solutions are compared to show the accuracy of the present spatial discretization scheme. The analytic velocity field prescribed is

Table 2
Comparisons with other computational and experimental results

| | $C_D$ | $C_L$ | $St$ |
|---|---|---|---|
| Hybrid mesh | 1.413 ± 0.03 | ±0.586 | 0.188 |
| Belov et al. [9] | 1.168 ± 0.025 | ±0.486 | 0.182 |
| Lin [11] | 1.166 ± 0.023 | ±0.477 | 0.182 |
| Experiment-1 [33] | N.A. | N.A. | 0.183 |
| Experiment-2 [34] | 1.328 | N.A. | N.A. |

Hybrid mesh result is from the current 2D simulation with the time step of $\Delta t = 0.1$. Belov's result is obtained with a 2D structured polar mesh, and Lin's result is obtained with 2D unstructured mesh with triangles. Experiment-2 is for $Re = 152$, and the rest are for $Re = 150$.

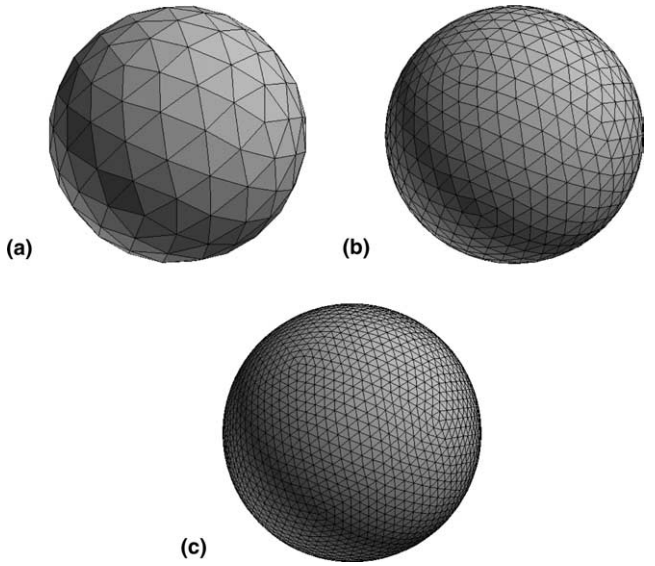


Fig. 12. Three levels of sphere mesh resolution used for the mesh convergence study: (a) coarse ($162 \times 33$ nodes, $320 \times 32$ elements), (b) medium ($642 \times 65$ nodes, $1280 \times 64$ elements), and (c) fine ($2562 \times 129$ nodes, $5120 \times 128$ elements).

Table 3
Characteristics of the initial (coarse), once refined (medium) and twice refined (fine) sphere meshes

| | Nodes | Elements | Initial spacing normal to wall, $\Delta r_0$ |
|---|---|---|---|
| Coarse | $162 \times 33$ | $320 \times 32$ | 0.01 |
| Medium | $642 \times 65$ | $1280 \times 64$ | 0.005 |
| Fine | $2562 \times 129$ | $5120 \times 128$ | 0.0025 |

$$\mathbf{U} = \sin(x)\mathbf{i} + \cos(y)\mathbf{j} + \sin(z)\mathbf{k}. \tag{27}$$

Since the exact values of the divergence and laplacian are known for the analytic velocity filed, the errors are computed by

$$\text{divergence Error}_{\Delta r} = \sqrt{\frac{\sum_{i=1}^{NI} |(\nabla \cdot \mathbf{U})_{\Delta r} - (\nabla \cdot \mathbf{U})_{exact}|^2}{NI}},$$

$$\text{laplacian Error}_{\Delta r} = \sqrt{\frac{\sum_{i=1}^{NI} \|(\nabla^2 \mathbf{U})_{\Delta r} - (\nabla^2 \mathbf{U})_{exact}\|^2}{NI}},$$

where $NI$ is the number of interior nodes. The order of convergence is approximated as follows:

$$\log_2\left(\frac{\text{divergence Error}_{\Delta r=0.005}}{\text{divergence Error}_{\Delta r=0.0025}}\right) = 1.751,$$

$$\log_2\left(\frac{\text{laplacian Error}_{\Delta r=0.005}}{\text{laplacian Error}_{\Delta r=0.0025}}\right) = 1.746.$$

The actual values of error for the three meshes considered are summarized in Table 4 and plotted in Fig. 13. The deviation from the second order accuracy, the theoretical order of accuracy of the central difference scheme which can be shown for uniformly spaced meshes seems to be due to the stretching of the spherical meshes. Further, a third refinement would be a more precise indicator, but was not performed due to the very large number of elements.

*5.2.2. Flows around a sphere*

Flows around a sphere for $Re = 100$ are tested using the three meshes described in the previous section.

The $C_D$'s for the three cases are compared in Table 5 and the order of convergence is estimated using the same technique as follows:

$$\log_2\left(\frac{(C_D)_{\text{fine}} - (C_D)_{\text{medium}}}{(C_D)_{\text{medium}} - (C_D)_{\text{coarse}}}\right) = 2.0.$$

Table 4
Errors of derivatives computations by using prescribed analytic velocity field

| | $\Delta r_0$ | divergence Error | laplacian Error |
|---|---|---|---|
| Coarse | 0.01 | $3.401 \times 10^{-1}$ | $3.285 \times 10^{-1}$ |
| Medium | 0.005 | $1.276 \times 10^{-1}$ | $1.271 \times 10^{-1}$ |
| Fine | 0.0025 | $3.789 \times 10^{-2}$ | $3.789 \times 10^{-2}$ |

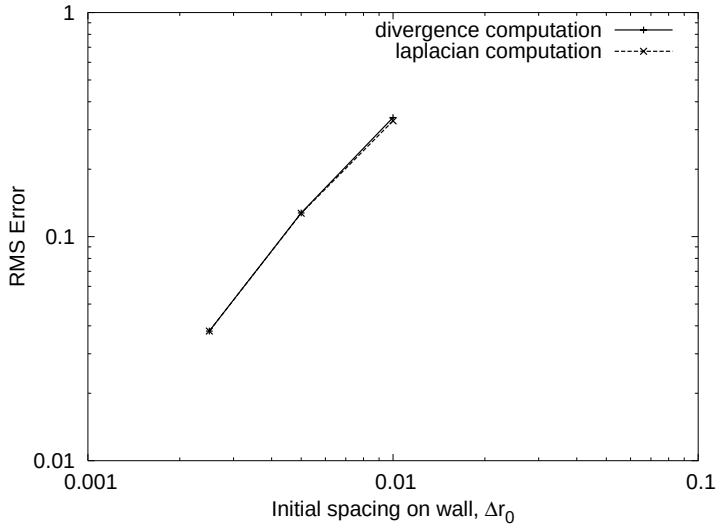


Fig. 13. Mesh convergence test using analytic velocity fields.

Table 5
Drag coefficients using the initial (coarse), once (medium) and twice (fine) refined sphere meshes, $Re = 100$

| | $C_D$ |
|---|---|
| Coarse | 1.069 |
| Medium | 1.081 |
| Fine | 1.084 |

As shown in the above extrapolation, the current discretization scheme yields second order accuracy for the drag coefficient $C_D$.

## 6. Utilization of local hexahedra within hybrid meshes

The major distinction of the general hybrid meshes from the conventional hybrid meshes is the inclusion of local hexahedra, possibly unstructured. Inclusion of hexahedra yields certain advantages. The hexahedron can express multi-directional anisotropy. Besides clustering along the normal direction to the wall, it can also cluster along the lateral directions. Hence these hexahedral layers are suitable for the *viscous wall with severe curvature*, such as the leading edge of a wing or the frontal area of a cylinder. Second, the hexahedrons can also be used in the regions where relatively isotropic cells are required, such as the wake region of a bluff body. In either case, the inclusion of the local hexahedra can yield savings in the number of elements of the meshes.

To take advantage of the merits of local hexahedra, the general hybrid mesh 1 as shown in Fig. 6 is modified with inclusion of local hexahedra in the wake region of the cylinder, as well. The resulting mesh is shown in Fig. 14. This mesh will be termed general hybrid mesh-2 (GHM-2) in the following. The first mesh of Fig. 6 contains local hexahedra only in the frontal area of the cylinder.

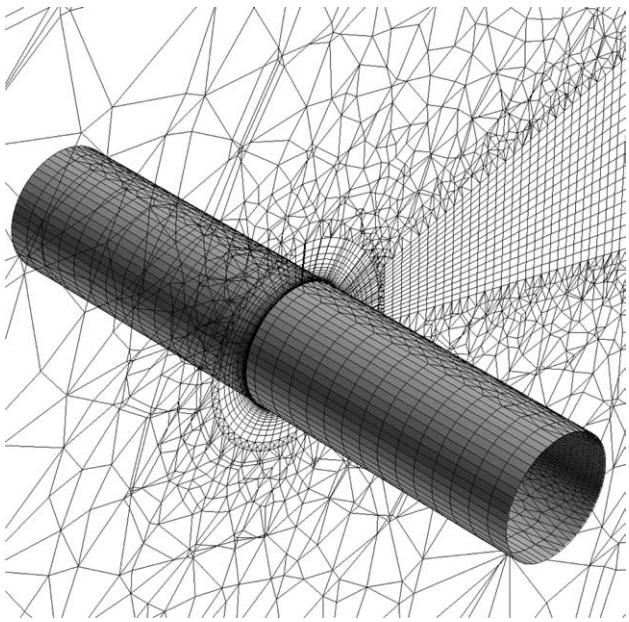

Fig. 14. General hybrid mesh-2 (GHM-2) containing hexahedra in the wake region as well as in the frontal region of the cylinder. $L/D = 5$.

The purpose of this simulation is (i) to demonstrate the generality of the developed numerical scheme to treat meshes with four different types of elements and (ii) to explore potential savings in computing resources by using local hexahedra in the wake.

### 6.1. Local hexahedra evaluation

One of the major advantages of using hybrid meshes is their lower connectivity as compared to the meshes with simplexes (triangles in two dimensions and tetrahedrons in three dimensions). Asymptotically, neglecting the boundary effect, for a given set of nodes, the mesh with tetrahedrons contains about seven times as many edges as the nodes, while the mesh with hexahedra contains only three times as many edges and the mesh with prisms contains only four times as many edges as the nodes [35].

For the present node-centered scheme based on edge-wise flux computations, the unknowns are stored at the nodes while the computational cost is directly proportional to the number of edges. If the hexahedra are introduced locally in the regions where the majority of nodes are located, the total number of edges can be reduced, thus resulting in CPU time and memory storage savings.

The statistics of the two general hybrid meshes employed are presented in Table 6. The number of edges is the parameter indicating the overall cost of flux computation based on edge-wise operations, and the number of faces is an indicator for the velocity gradient computation which involves the face-wise surface integrals of edge-duals. It can be expected that the computational cost of GHM-2, which contains fewer edges and faces than the GHM-1, will be less expensive than the GHM-1 even though the GHM-2 has more nodes.

The $C_D$ and $C_L$ histories from the flows around a circular cylinder by using the two general hybrid meshes are presented in Fig. 15. The results using the two hybrid meshes are almost identical in terms of the hydrodynamics forces exerted on the cylinder. This is important as it provides an indication that the accuracy of the developed scheme is not affected by the presence of extra interfaces between hexahedra, pyramids and tetrahedra, as well as extra pyramids in mesh GHM-2.

The performance metrics for the two hybrid meshes are compared in Table 7. The performance is measured in terms of total computational time and in the maximum memory requirement. The central scheme is used with modified smoothing based on the solution reconstruction, and the second order upwind scheme is employed by using Roe's flux-different splitting scheme. Regardless of the scheme employed and regardless of the performance metric chosen (CPU time or memory requirement), the GHM-2 shows about 10% savings over the GHM-1. This is the direct effect of local hexahedra in the wake region, which results in about 9% of savings in the total number of edges and 18% of savings in the total number of faces. The savings of computational cost is not only in the CPU time but also in the amount of memory required.

Since both of the tested meshes are already taking advantage of the local hexahedra, this savings from local hexahedra in the general hybrid meshes should be much larger when they are compared to the meshes only with tetrahedrons or conventional hybrid meshes of prisms and tetrahedrons.

Table 6
Characteristics of employed general hybrid meshes

| | GHM-1 | GHM-2 |
|---|---|---|
| Number of nodes | 148,719 | 158,293 |
| Number of cells | 509,269 | 385,115 |
| Number of edges | 749,664 | 683,563 |
| Number of faces | 1,110,214 | 910,385 |

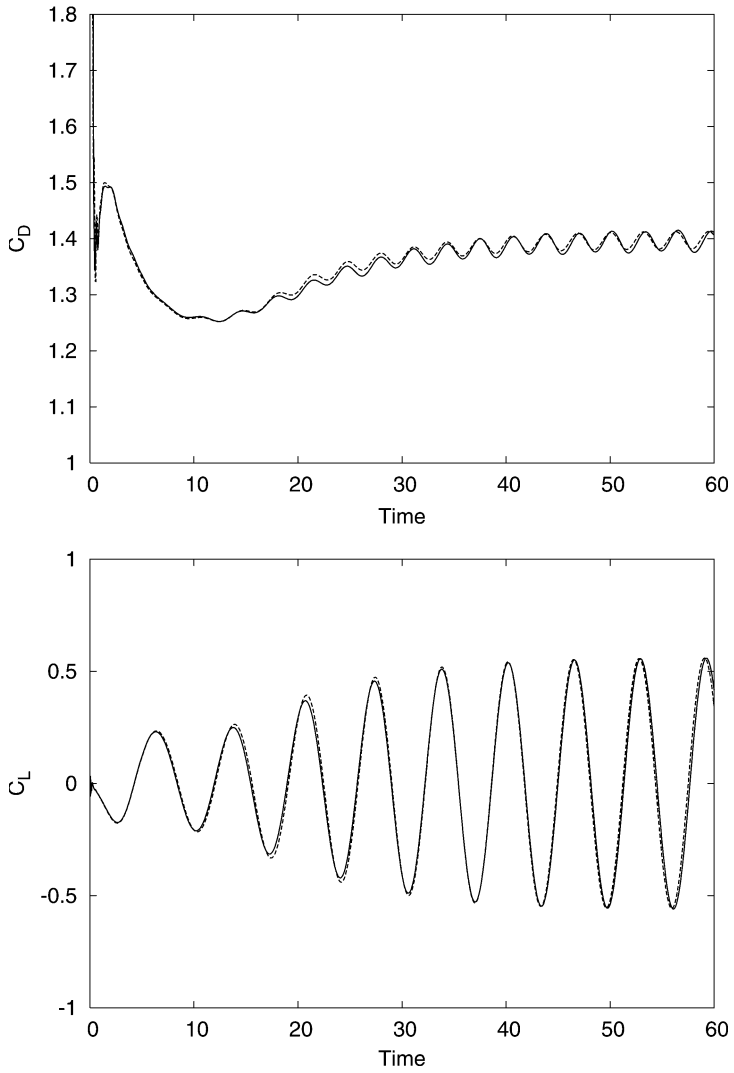


Fig. 15. Local hexahedra effect on $C_D$ and $C_L$ histories. The central difference scheme is used with modified smoothing of $\sigma_4 = 0.1$. Solid lines are for the GHM-1 with tetrahedra in the wake region, and the dashed lines are for the GHM-2 with hexahedra in the wake region ($Re = 150$). Central difference scheme with artificial dissipation of Haselbacher and Blazek, shown in Eq. (21), is used.

Table 7
Performance metrics for the hybrid meshes with and without local hexahedra in the wake region

| | | GHM-1 | GHM-2 |
|---|---|---|---|
| Central | CPU time (s)/$N_{total}$ | 0.0328 | 0.0297 |
| | Max. memory (MB)/$N_{total}$ | 0.00850 | 0.00766 |
| Upwind | CPU time (s)/$N_{total}$ | 0.0420 | 0.0383 |
| | Max. memory (MB)/$N_{total}$ | 0.00852 | 0.00768 |

CPU time is measured when the simulation reaches *Time* = 1.0 on 16 processors. Metrics are normalized by the total number of nodes ($N_{total}$).

## 7. High Reynolds number flow simulations

The validity of the proposed numerical method combined with general hybrid meshes is examined further with simulations of high Reynolds number flows over a sphere and a cylinder. The Spalart–Allmaras one equation turbulence model [15] is used for the current Reynolds-averaged Navier–Stokes (RANS) computations without a trip function for transition. The free stream value of $\tilde{\nu}$ was set to 0.1, and Roe's scheme was used for the discretization of the S–A equation.

RANS approach is one of the least expensive strategies for turbulent flow simulations, and its validity for unsteady separated flows may be arguable. However, the main objective of this section is to provide the applicability of the presented hybrid mesh methods for high Reynolds number flows. The approaches of detached eddy simulations (DES) [36] and large eddy simulations (LES) are currently investigated by the community for bluff body flows.

A series of simulations are performed for various orders of Reynolds numbers in the range of 10–1,000,000, and they are compared to the available experimental results. For critical Reynolds numbers ($\geqslant$100,000), the delayed separation of the boundary layer over the surface of the bluff body which is a strong indication of the boundary layer transition to turbulent was predicted.

### 7.1. Turbulent flows around a sphere

The medium resolution mesh is used for the turbulent flow simulation over a sphere. The drag coefficient ($C_D$) values obtained by the simulations are compared with experimental results for various values of surface roughness, and those are presented in Fig. 16. Overall, the computation results follow the experiments quite well. Drag reduction induced by boundary layer transition from laminar to turbulent is also observed for $Re \geqslant 100{,}000$. The delayed separation of the boundary layer on the surface due to the boundary layer transition is also observed and displayed in Fig. 17. The narrowed wake region behind the sphere is evident by the comparison between two representative Reynolds numbers, $Re = 1000$ and $Re = 100{,}000$.

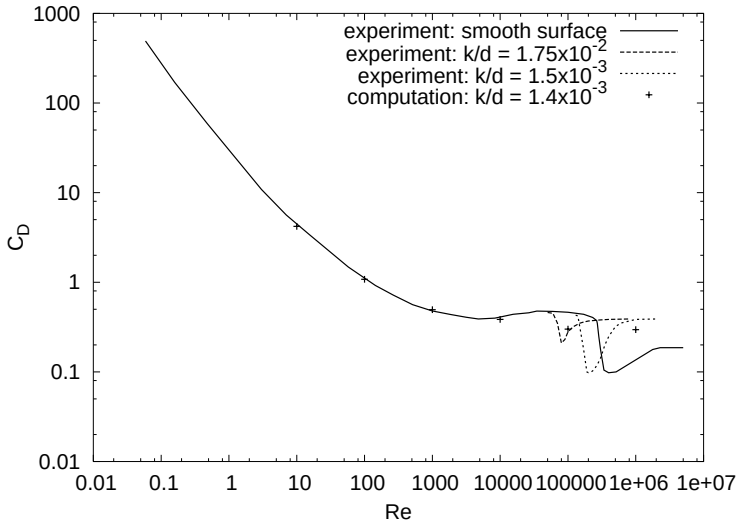


Fig. 16. $C_D$ vs. $Re$ curve for flows around a sphere. Experimental results from [37,38] and computational results obtained from the current simulation using the medium resolution mesh are depicted.

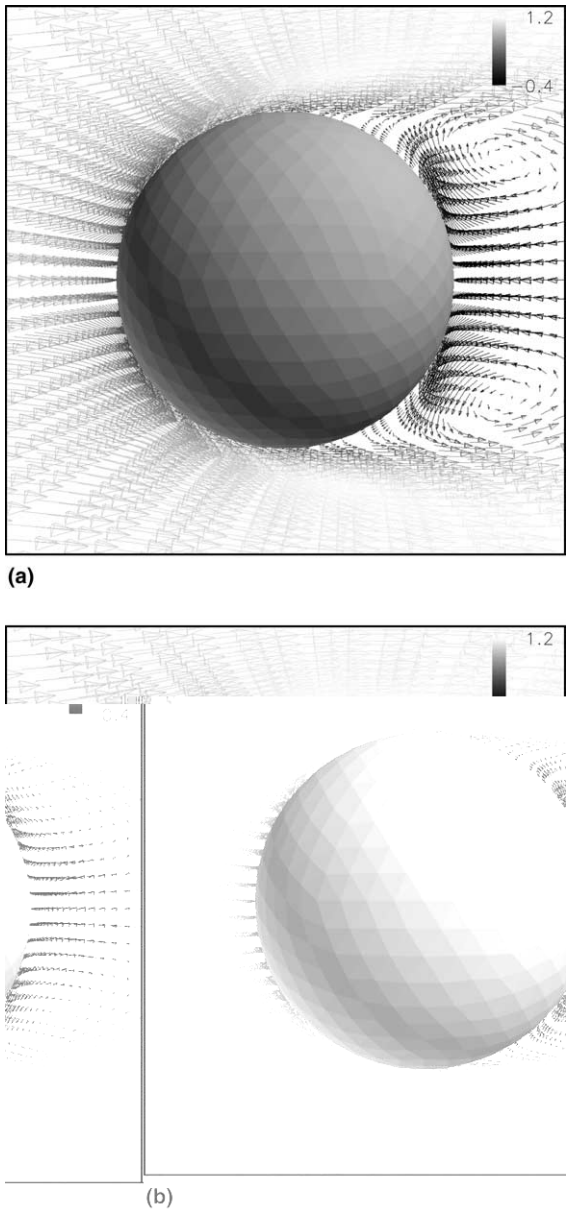


Fig. 17. Delayed separation and narrowed wake region for flows around a sphere for the higher Reynolds number. Velocity is scaled by $u$-velocity magnitude. The medium sphere mesh as shown in Fig. 12 is used. (a) $Re = 1000$ and (b) $Re = 100{,}000$.

One can observe a deviation of numerical result from the experiments for Reynolds numbers in the supercritical regime ($Re \geqslant 100{,}000$). This discrepancy at the supercritical regime is attributed to many factors, such as surface roughness of the current computation model and mis-prediction of boundary layer transition by using a relatively simple turbulent model equation. However, it should be noticed that even the experimental results are very sensitive to surface roughness in the supercritical regime.

### 7.2. Turbulent flows around a cylinder

The next class of cases is turbulent flows around a cylinder. The hybrid mesh GHM-1, shown in Fig. 6, is used for all of the cases.

For a range of Reynolds numbers, $C_D$ is compared with experimental results. As shown in Fig. 18, the simulations agree well with the experimental results. Especially for the $Re = 100{,}000$ case, a sharp drop of $C_D$ is predicted which is indicating the delayed separation induced by the boundary layer transition from laminar to turbulent. This delayed separation and narrowed wake region at $Re = 100{,}000$ is compared to the $Re = 1000$ case and those are displayed in Fig. 19.

## 8. Hybrid mesh data structure for parallel execution

Parallel implementation of the developed solution algorithms for general hybrid meshes is presented for a distributed memory machine by using MPI (message passing interface) library functions. The original mesh is partitioned using METIS, an open source graph/mesh partitioner [42]. The interface elements, whose nodes are assigned to multiple parts, are stored redundantly in the neighbor parts in order to reduce the communication time. Hybrid mesh data structure for parallel execution is presented for the inter-processor communication. Finally, the scalability of the parallel implementation is measured on a distributed memory machine. The communication overhead is estimated by counting the number of nodes participating in the inter-processor communication.

### 8.1. Overlapping of the interface cells for efficient inter-processor communication

Graph partitioning is transparent to the cell topologies and the node-connectivity is the only information required by the partitioner. Hence, graph partitioning can be applied directly to general hybrid meshes.

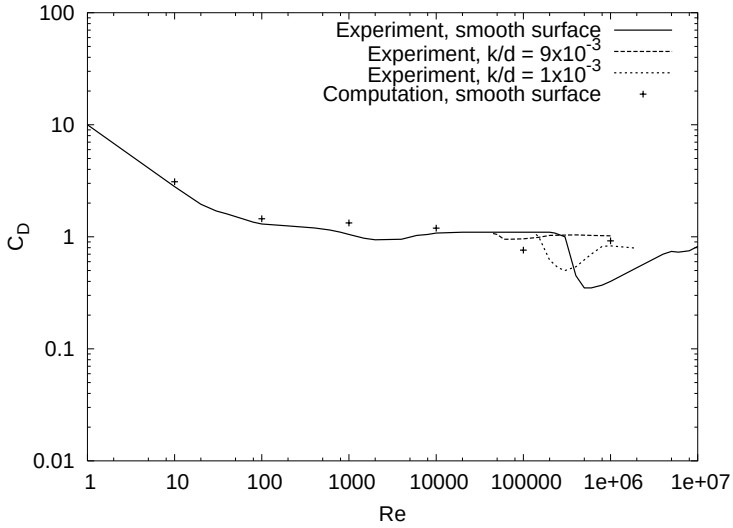


Fig. 18. $C_D$ vs. $Re$ curve for cylinder using the general hybrid mesh 1. Comparison between experimental result [37,39–41] and numerics.

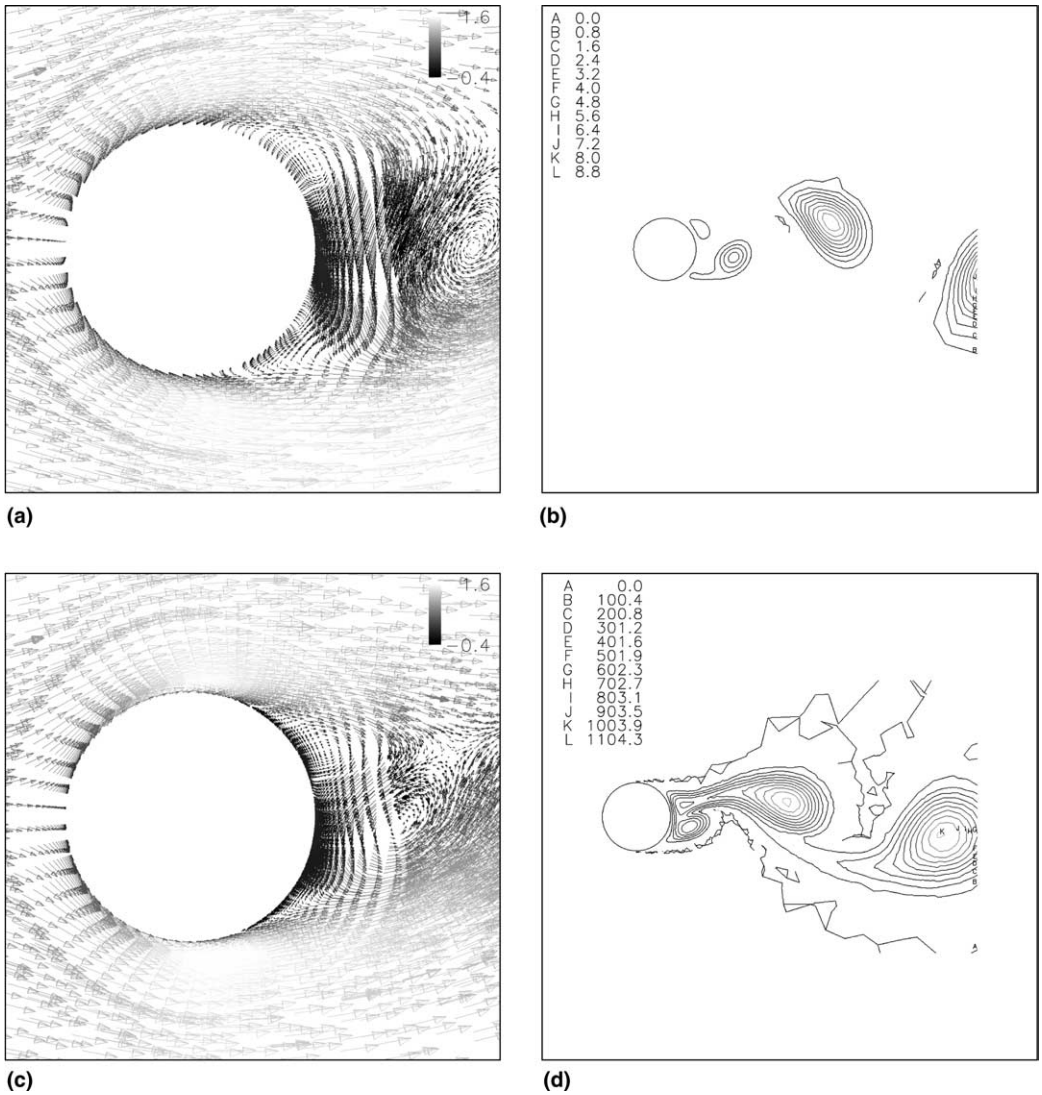


Fig. 19. Prediction of delayed separation (accompanied by a smaller wake region) for the higher Reynolds number. Velocity (scaled by $u$-velocity) and $\tilde{\nu}$ (the working variable of Spalart–Allmaras turbulence model) plots are taken approximately at the same time step within a shedding cycle. The turbulent eddy viscosity is defined as $\tilde{\nu} = \frac{\chi^3}{\chi^3+C_{v1}^3}$, where $\chi = \frac{\tilde{\nu}}{\nu_L}$, $C_{v1} = 7.1$ and $\nu_L$ is the laminar kinematic viscosity. The hybrid mesh GHM-1 is used. (a) Velocity ($Re = 1000$), (b) $\tilde{\nu}$ ($Re = 1000$), (c) velocity ($Re = 100{,}000$) and (d) $\tilde{\nu}$ ($Re = 100{,}000$).

The parallel communication strategy of the present work is similar to the approach of Tai and Zhao [43] who used two-dimensional triangular meshes. The main idea of their implementation is overlapping the interface cells, whose nodes are assigned to multiple partitions. This may look as an overhead of memory space. However, once an explicit scheme is parallelized on a distributed memory machine, the amount of memory required for each processor is not the critical issue, but the amount of communication is.

By overlapping the interface cells, the communication amount can be reduced considerably. For the current scheme, only two steps of node-wise inter-processor communication can be achieved per solution

update. The first communication is for exchanging the nodal gradients of the solution, which is required for the solution reconstructions for the artificial dissipation or high order upwind scheme. The second communication step is for the boundary condition application which is performed after an intra-processor solution update. Once all the nodal solutions are updated, the solutions at the ghost nodes, which reside at the inter-processor boundaries, are updated by node-wise inter-processor communication. This overlapping strategy of general hybrid mesh partitioning is delineated in Fig. 20 in two dimensions. An example of the information required for the inter-processor communication is listed in Table 8. Processors store their own communication table which contains lists of nodes for inter-processor communications.

If the interface cells are not overlapped, the inter-processor communication may introduce more steps of communication as well as extra complexity in the communication strategy. For example, in the overlapping approach there is no extra communication needed for the viscous flux computation, while in the non-overlapping strategy there should be extra communications regarding the computation using edge-duals which lie on the inter-partition boundaries.

According to the partitioning strategy presented in Fig. 20, there is no assumption that each part has to be contiguous. A completely separate part of the original mesh can be assigned to the same processor, like an island can belong to a country which is separated by the ocean. This is provides flexibility to a graph partitioner, which may produce a non-contiguous partitioning.

Each partition has its own local node numbering, and the translation table of the local node number to the global node number is required for the communications between the neighbors. Once the communication table is built for all processors, the inter-processor communication can be implemented via a single loop over the processors.

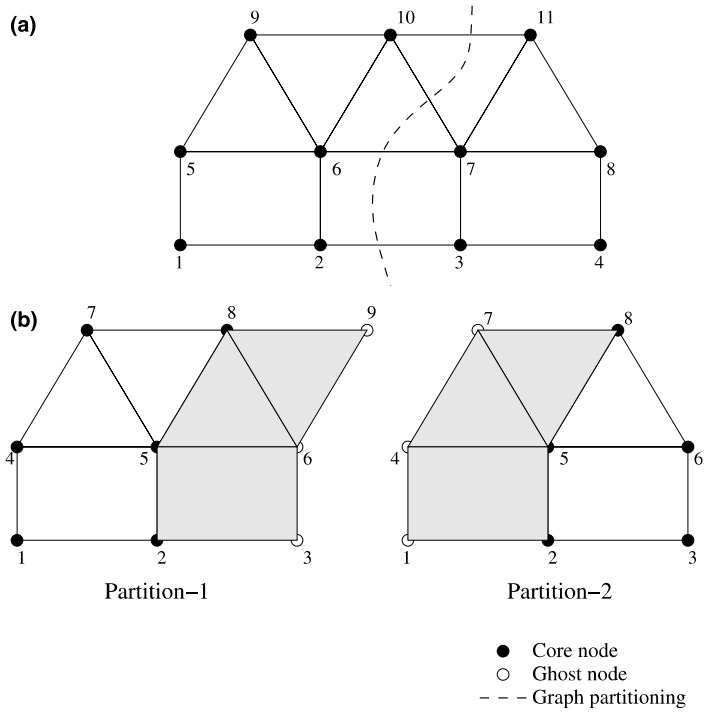


Fig. 20. Graph partitioning of a two-dimensional hybrid mesh with overlapping interface cells (cells in gray color): (a) original hybrid mesh with global node numbering, and (b) partitioned hybrid meshes with local node numbering.

Table 8
Communication tables for the node-wise inter-processor communications

| Send to | *Local*(*global*) | Recv. from | *Local*(*global*) |
|---|---|---|---|
| *(a) Communication for partition-1* | | | |
| 2 | 2(2) | 2 | 3(3) |
| | 5(6) | | 6(7) |
| | 8(10) | | 9(11) |
| *(b) Communication table for partition-2* | | | |
| 1 | 2(3) | 1 | 1(2) |
| | 5(7) | | 4(6) |
| | 8(11) | | 7(10) |

The *local* is local node numbering and *global* is global node numbering. Each partition has its own local node numbering, and the translation table of the local node number to the global node number is required for the communications between the neighbors.

This strategy follows the SPMD (single-program multiple-data) paradigm. All processors simultaneously get involved into the communication with their neighbors, and this enables the parallel data-exchange. This single-loop parallel data-exchange does not require any a priori optimal scheduling of communication order, which simplifies the parallel implementation.

### 8.2. Scalability

The scalability of a given algorithm (program) on a specific machine depends on the problem size. Typically the larger the problem size, the better scalability is. For the parallel speed-up check of the present parallel implementation scheme, the general hybrid mesh of 148,719 nodes is used.

The performance of the parallel implementation is measured on a Linux cluster. As shown in Fig. 21, the speed-up of the parallel execution is almost linear up to 16 processors and then it starts to deviate from the ideal and becomes sub-linear. This deviation of the actual computation from the ideal is due to the increase of communication load.

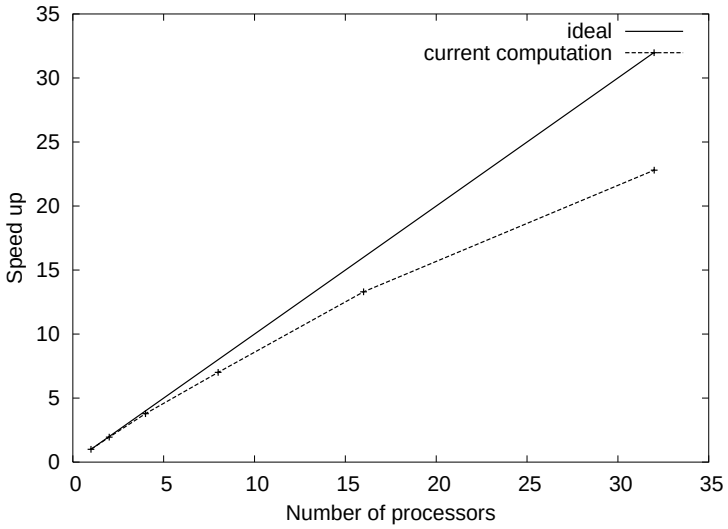


Fig. 21. Scalability of the parallel implementation with a Linux cluster using a general hybrid mesh of 148,719 nodes.

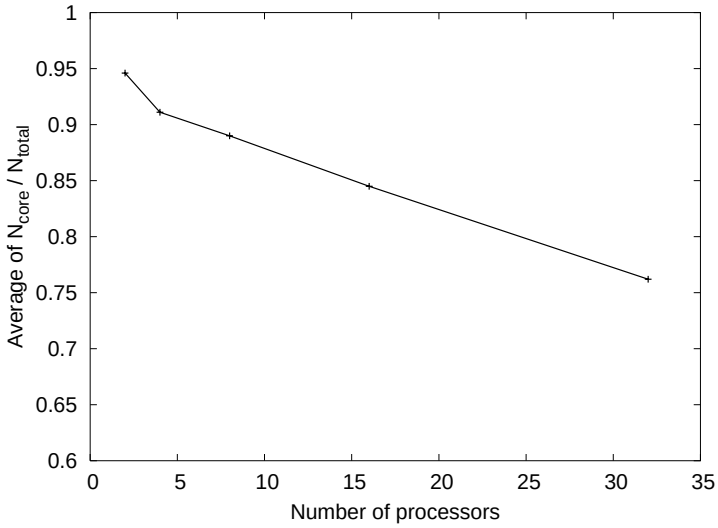


Fig. 22. Average of the ratios of core nodes with respect to the total nodes assigned to each partition. The total number of nodes for each part is sum of core nodes and ghost nodes. The general hybrid mesh of 148,719 nodes is used.

According to the present parallel implementation strategy, there are core nodes and ghost nodes at each processor. The total number of nodes at each partition is composed of the core nodes and ghost nodes. Solution at the core nodes can be updated without inter-processor communication, but the solution at the ghost nodes needs to receive information from its neighbor processor which contains the ghost as a core node. Hence, as the number of partitions increase, the number of ghost nodes also increases as shown in Fig. 22. This is communication overhead, and it is one of the major reasons for the deviation of speed-up from the ideal. Even with the rapid increase of the ghost nodes among the total nodes at each part, the parallel performance is fairly ideal, and this is because of the fewer number of communications required by the current parallel algorithm.

## 9. Conclusions

A new incompressible Navier–Stokes method for general hybrid meshes was presented. A special formulation using dual time stepping was employed in order to have a time accurate artificial compressibility method. Data structures for partitioned mashes containing hexahedra, prisms, tetrahedra, and pyramids were developed along with an appropriate communication strategy for parallel implementation of the numerical method.

The presented algorithms for (i) spatial discretization, (ii) time integration, and (iii) parallel implementation are transparent to the type of computational element due to their special data structures and proposed integration techniques. Further, accuracy of the simulations did not deteriorate with the presence of the interfaces between hexahedra, prisms, and tetrahedral or with inclusion of pyramids. The proposed special evaluation of the velocity gradients is less CPU time consuming compared to previous such algorithms. Temporal and spatial accuracies were checked via practical definitions of the error for cases lacking analytical solution. The developed numerical method is tolerant to high aspect ratio cells and to spurious small-size elements in the domain. The modified artificial dissipation operator gave similar results to the more expansive upwind method. Use of local blocks of hexahedra yielded reduced CPU time and memory

storage without sacrificing accuracy for the wide range of Reynolds numbers ($10$–$10^6$) considered here. Finally, the developed partitioned mesh data structures for general hybrid grids along with the proposed parallel communication technique yielded quite scalable computations.

## References


[1] P.M. Gresho, Some current CFD issues relevant to the incompressible Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 87 (1991) 201–252.
[2] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Physics of Fluids 8 (1965) 2182.
[3] A.J. Chen, Y. Kallinderis, Adaptive hybrid (prismatic-tetrahedral) grids for incompressible flows, International Journal for Numerical Methods in Fluids 26 (1998) 1085–1105.
[4] K.W. Schulz, Y. Kallinderis, Unsteady flow structure interaction for incompressible flows using deformable hybrid grids, Journal of Computational Physics 143 (1998) 569–597.
[5] A.J. Chorin, A numerical method for solving incompressible viscous flow problems, Journal of Computational Physics 2 (1967) 12–26.
[6] R. Peyret, Unsteady evolution of a horizontal jet in a stratified fluid, Journal of Fluid Mechanics 78 (1976) 49–63.
[7] S.E. Rogers, D. Kwak, Upwind differencing scheme for the time-accurate incompressible Navier–Stokes equations, AIAA Journal 28 (1990) 253–262.
[8] S.E. Rogers, D. Kwak, C. Kiris, Steady and unsteady solutions of the incompressible Navier–Stokes equations, AIAA Journal 29 (1991) 603–610.
[9] A. Belov, A. Jameson, L. Martinelli, A new implicit algorithm with multigrid for unsteady incompressible flow calculations, AIAA Paper 95-0049.
[10] A. Jameson, Time-dependent calculations using with applications to unsteady flows past airfoils and wings, AIAA Paper 91-1596.
[11] P. Lin, Two-dimensional implicit time dependent calculation for incompressible flows on adaptive unstructured meshes, Ph.D. Thesis, Princeton University, 2001.
[12] W.K. Anderson, R.D. Rausch, D.L. Bonhaus, Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids, Journal of Computational Physics 128 (1996) 391–408.
[13] Y. Kallinderis, Hybrid grids and their applications, in: J.F. Thompson, B.K. Soni, N.P. Weatherill (Eds.), Handbook of Grid Generation, CRC Press, Boca Raton, FL, 1999.
[14] Y. Kallinderis, Parallel adaptive Navier–Stokes method and load balancing with hybrid prismatic/tetrahedral grids, in: M. Hafez, K. Oshima (Eds.), Computational Fluid Dynamics Review 1995, Wiley, New York, 1996.
[15] P.R. Spalart, S.R. Allmaras, A one-equation turbulence model for aerodynamic flows, AIAA Paper 92-0439.
[16] P. Godin, D.W. Zingg, T.E. Nelson, High-lift aerodynamic computations with one and two-equation turbulence models, AIAA Journal 35 (1997) 237–243.
[17] E. Turkel, Preconditioned method for solving the incompressible and low speed compressible equations, Journal of Computational Physics 72 (1987) 277–298.
[18] E. Turkel, A review of preconditioning methods for fluid dynamics, Applied Numerical Mathematics 12 (1993) 257–284.
[19] A.G. Malan, R.W. Lewis, P. Nithiarasu, An improved unsteady, unstructured, artificial compressibility, finite volume scheme for viscous incompressible flows: Part i. theory and implementation, Internal Journal for Numerical Methods in Engineering 54 (2002) 697–714.
[20] E. Turkel, Frontiers of the Computational Fluid Dynamics 1994, Wiley, New York, 1998, Chapter 13: Preconditioning and the limit of the compressible to the incompressible flow equations for finite difference schemes.
[21] T.J. Barth, Aspects of unstructured grids and finite-volume solvers for the Euler and Navier–Stokes equations, Technical Report AGARD Report 787 on unstructured grid methods for advection dominated flows, AGARD, 1992, pp. 6.1–6.60.
[22] P.L. Roe, Approximate Riemann solvers, parameter vectors and difference schemes, Journal of Computational Physics 43 (1981) 357–372.
[23] L.K. Taylor, D.L. Whitfield, Unsteady three-dimensional incompressible Euler and Navier–Stokes solver for stationary and dynamic grids, AIAA 92 (1991) 1650.
[24] S. Kim, Reynolds-averaged Navier–Stokes computation of tip clearance flow in a compressor cascade using an unstructured grid, Ph.D. Thesis, Virginia Polytechnic Institute and State University, 2001.
[25] W.K. Anderson, D.L. Bonhaus, Algorithm for computing turbulent flows on unstructured grids, Computers & Fluids 23 (1994) 1–21.
[26] A. Haselbacher, J. Blazek, On the accurate and efficient discretization of the Navier–Stokes equations on mixed grids, AIAA Journal 38 (2000) 2094–2102.

[27] M.E. Braaten, S.D. Connell, Three-dimensional unstructured adaptive multigrid scheme for the Navier–Stokes equations, AIAA Journal 34 (1996) 281–290.
[28] R.C. Swanson, R. Radespiel, E. Turkel, On some numerical dissipation schemes, Journal of Computational Physics 147 (1998) 518–544.
[29] L. Martinelli, Calculation of viscous flows with a multigrid method, Ph.D. Thesis, Princeton University, 1987.
[30] R.C. Swanson, E. Turkel, On central-difference and upwind schemes, Journal of Computational Physics 101 (1992) 292–306.
[31] Y. Kallinderis, H. McMorris, Magnitude of artificial dissipation for numerical simulations, AIAA Journal 33 (1995) 1526–1529.
[32] Y. Kallinderis, A 3-D finite-volume method for the Navier–Stokes equations with adaptive hybrid grids, Applied Numerical Mathematics 20 (1996) 387–406.
[33] C.H.K. Williamson, Defining a universal and continuous Strouhal–Reynolds number relationship for the laminar vortex shedding of a circular cylinder, Physics of Fluids 31 (1988) 2742–2744.
[34] E. Relf, Discussion of results of measurements of the resistance of wires, with some additional tests on the resistance of wires of small diameter, Technical Report 102, The Advisory Committee for Aeronautics, 1915.
[35] D.J. Mavriplis, Unstructured grid techniques, Annual Review of Fluid Mechanics 29 (1997) 473–514.
[36] P. Spalart, W. Jou, M. Strelets, S. Allmaras, Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach, in: C. Liu, Z. Liu (Eds.), Advances in DNS/LES, Greyden Press, 1997.
[37] R.H. Sabersky, A.J. Acousta, E. Hauptman, Fluid Flow: a First Course in Fluid Mechanics, Prentice-Hall, Englewood Cliffs, NJ, 1998.
[38] E. Achenbach, Experiments on flow past spheres at very high Reynolds numbers, Journal of Fluid Mechanics 54 (1972) 565–575.
[39] Y. Nakamura, Y. Tomonari, The effects of surface roughness on the flow past circular cylinders at high Reynolds numbers, Journal of Fluid Mechanics 123 (1982) 363–378.
[40] E. Achenbach, Influence of surface roughness on the cross-flow around a circular cylinder, Journal of Fluid Mechanics 46 (1971) 321–335.
[41] A. Roshko, Experiments on the flow past a circular cylinder at very high Reynolds number, Journal of Fluid Mechanics 10 (1961) 345–356.
[42] G. Karypis, V. Kumar, METIS: A software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 4.0, Department of Computer Science, University of Minnesota, 1998.
[43] C.H. Tai, Y. Zhao, Parallel unsteady incompressible viscous flow computations using an unstructured multigrid method, Journal of Computational Physics 192 (2003) 277–311.