Journal of Computational Physics **169,** 24–43 (2001)
doi:10.1006/jcph.2001.6705, available online at http://www.idealibrary.com on IDEAL®

An Edge-Based Method for the Incompressible Navier–Stokes Equations on Polygonal Meshes

Jeffrey A. Wright* and Richard W. Smith†

**Streamline Numerics Inc., Gainesville, Florida 32609; and †Coastal Systems Station, Naval Surface Warfare Center, Panama City, Florida 32407*
E-mail: smithr@atcf.ncsc.navy.mil

Received November 9, 1999; revised August 15, 2000

A pressure-based method is presented for discretizing the unsteady incompressible Navier–Stokes equations using hybrid unstructured meshes. The edge-based data structure and assembly procedure adopted lead naturally to a strictly conservative discretization, which is valid for meshes composed of *n*-sided polygons. Particular attention is given to the construction of a pressure–velocity coupling procedure which is supported by edge data, resulting in a relatively simple numerical method that is consistent with the boundary and initial conditions required by the incompressible Navier–Stokes equations. Edge formulas are presented for assembling the momentum equations, which are based on an upwind-biased linear reconstruction of the velocity field. Similar formulas are presented for assembling the pressure equation. The method is demonstrated to be second-order accurate in space and time for two Navier–Stokes problems admitting an exact solution. Results for several other well-known problems are also presented, including lid-driven cavity flow, impulsively started cylinder flow, and unsteady vortex shedding from a circular cylinder. Although the method is by construction minimalist, it is shown to be accurate and robust for the problems considered.

Key Words: unstructured grids; incompressible flows; unsteady flows; edge-based methods.

1. INTRODUCTION

Over the past decade or so edge-based finite volume methods have emerged as an interesting alternative to the conventional methods for discretizing conservation laws on so-called hybrid or mixed element meshes. Edge-based assembly methods are distinguished from both traditional finite element and finite difference methods in that the assembly of the discrete equations is performed by a sweep over edges, whereas with a traditional finite element or finite difference approach assembly is performed by a sweep over

0021-9991/01 $35.00

elements or nodes, respectively. Perhaps the most compelling prospect for such methods is that assembly can be simply performed for arbitrary polygonal (or polyhedral in three dimensions) meshes, since the only requirement is that the edges (or surfaces) of the tessellation form closed control volumes. Furthermore, edge-based assembly is efficient, requiring only one flux evaluation per edge and naturally leads to a strictly conservative discretization.

The majority of work to date on edge-based methods [1–4] has been associated with the compressible Euler and Navier–Stokes equations where a thermodynamic equation of state is available to link the pressure field to a conserved density field. As the incompressible limit is approached, the mass conservation equation changes from an evolution equation for density to a constraint equation imposing a divergence-free condition on the velocity field. This divergence-free constraint is independent of time and governs the evolution of a "special," purely elliptic pressure field which "ensures that the resulting acceleration field is divergence-free and thus that the velocity remains divergence-free" [5].

Recently, Thomadakis and Leschziner [6] proposed a first-order edge-based pressure-correction method for the steady incompressible Navier–Stokes equations in the spirit of the segregated implicit family of pressure-based methods (SIMPLE, SIMPLER, etc.) originally proposed by Patankar [7]. The present work is an extension of the method proposed by Thomadakis and Leschziner [6], but with the important difference that an equation for pressure rather than its correction is proposed to couple the pressure and velocity fields. It will be shown that this choice, when made in conjunction with a semistaggered variable storage arrangement, leads to a method that does not require *ad hoc* modifications to avoid a checkerboard pressure field nor does it require a boundary condition for pressure. Consequently, the method presented here exhibits the minimum level of complexity for linking the pressure and velocity fields in an edge-based incompressible Navier–Stokes algorithms. The minimalist viewpoint is evident in many other aspects of the algorithm as well, viz.,

- Fluxes are evaluated using linear reconstruction and the trapezoidal rule.
- No attempt is made to conserve mass to machine precision.
- No pressure or velocity correction step is undertaken.
- Jacobi iteration is adopted as the linear solution method.

The implicit two-stage time integrator does, unavoidably, introduce a measure of complexity to the overall unsteady algorithm. However, the first-order implicit time integrator and the steady-state form of the algorithm are special cases that can be easily recovered from the general formulation. Additionally, the initial and boundary conditions required by the method are identical to those required by the Navier–Stokes equations. Several related works addressing element-based finite volume methods have also appeared recently in the literature [8–10].

2. BASIC CONSERVATION LAWS

The equations governing fluid dynamics stated in weak coordinate-free form [11] for a finite volume Ω with boundary Γ are given by

$$\frac{\partial}{\partial t}\int_{\Omega} W\, d\Omega + \oint_{\Gamma} [F(W, \boldsymbol{n}) - G(W, \nabla W, \boldsymbol{n})]\, d\Gamma + \int_{\Omega} B\, d\Omega = 0, \tag{1}$$

where

$$W = [\rho, \rho \boldsymbol{v}]^T$$
$$F(W, \boldsymbol{n}) = W(\boldsymbol{v} \cdot \boldsymbol{n})$$
$$G(W, \nabla W, \boldsymbol{n}) = [0, \boldsymbol{t}]^T$$
$$\boldsymbol{t} = \boldsymbol{n} \cdot \boldsymbol{T}.$$

Adopting the incompressible form of the Navier–Poisson law as the constitutive relation for the fluid gives

$$\boldsymbol{T} = -p\boldsymbol{I} + \mu(\nabla \boldsymbol{v} + \boldsymbol{v}\nabla), \tag{2}$$

where $\boldsymbol{T}$ and $\boldsymbol{I}$ are the stress and identity tensors, respectively, and $\boldsymbol{v}\nabla$ is the transpose of $\nabla \boldsymbol{v}$. In Eq. (1), W is the vector of conserved variables, F and G are flux vectors, B is the vector of volume sources, $\boldsymbol{v}$ and $\boldsymbol{t}$ are the flow velocity and stress vectors, and $\boldsymbol{n}$ is the unit normal vector. The variables ρ, μ, and p denote the fluid density, viscosity, and flow pressure, respectively.

A number of methods for establishing the necessary algorithmic linkage between pressure and velocity for the incompressible Euler and Navier–Stokes equations have been developed and can be broadly classified as coupled direct discretization [12], artificial compressibility [13], or pressure based [7]. The first of these methods is seldom used because of prohibitive memory and CPU requirements when applied to large three-dimensional problems. The second of these methods has been widely used for several decades and relies on the proposition of a finite speed for the propagation of a pressure signal. As a consequence of a finite propagation speed for pressure, artificial compressibility methods have the same eigenstructure as the purely hyperbolic system associated with compressible fluid dynamics and consequently are obliged to retain the legacy of compressibility in the numerical implementation of the method. With the third method, a purely elliptic equation of the Poisson-type governing the pressure field (or its correction, p') can be derived in strong form by manipulation of the continuity and Navier–Stokes equations or in weak form by manipulating the finite volume form of the basic conservation laws in discrete form. The latter approach utilizing the discrete finite volume form of mass and momentum conservation is taken here.

3. NUMERICAL METHOD

3.1. *Connectivity and Assembly*

The conservation laws are discretized in space using an edge-based assembly procedure that results in a strictly conservative discrete form that is valid for n-sided polygonal control volumes. Figure 1 illustrates a typical mixed-element mesh comprising quadrilateral and triangular control volumes. In addition to the main mesh, a reciprocal or dual mesh is constructed by connecting the centroids of the main control volumes which share a common vertex. This system of main and dual control volumes forms a semistaggered grid arrangement where the velocity components are stored at the vertices of the main mesh and pressure is stored at the centroids of the main control volumes as indicated in the figure. It

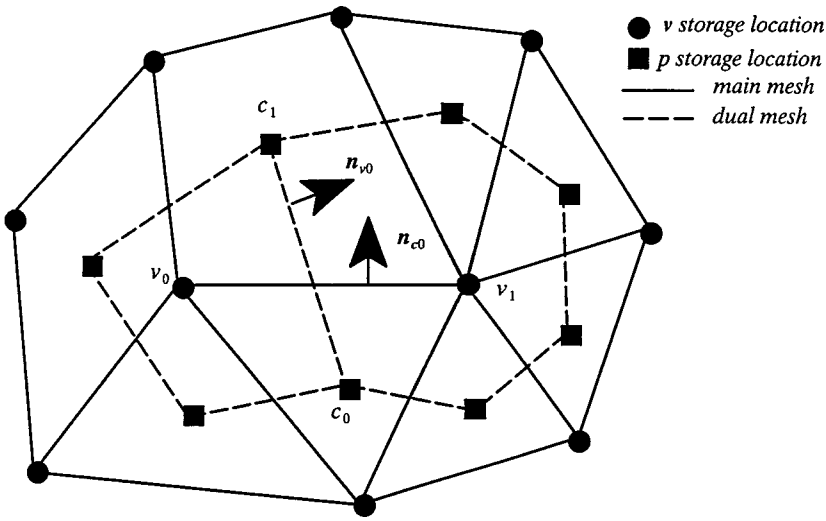

FIG. 1. Connectivity convention and storage locations for edge-based discretization.

will be shown that this arrangement provides sufficient algorithmic coupling of velocity and pressure, thus precluding the need for *ad hoc* modifications to the basic conservation laws such as artificial dissipation [14], momentum interpolation [15], or modified convection [8] to achieve a smooth pressure field.

The spatial residuals of mass and momentum are evaluated by projecting the dependent variables in Eq. (1) onto a Cartesian basis and performing a counterclockwise integration around the contour surrounding the main and dual control volumes, respectively. This integration is facilitated by forming a list of main edges comprising the tessellation with endpoints denoted $v0$ and $v1$ for each edge in the list. Each main edge in the tessellation shares exactly two main control volumes with centroids denoted by $c0$ and $c1$, which are arranged by construction such that $c0$ lies to the right of the directed line from $v0$ to $v1$ as shown in the figure. This connectivity convention allows the scaled outward normal vectors for both the main and dual control volumes to be unambiguously given by

$$\text{main edges} \begin{cases} \boldsymbol{n}_{c0} = -(y_{v1} - y_{v0})\boldsymbol{i} + (x_{v1} - x_{v0})\boldsymbol{j} \\ \boldsymbol{n}_{c1} = -\boldsymbol{n}_{c0} \end{cases} \tag{3a}$$

$$\text{dual edges} \begin{cases} \boldsymbol{n}_{v0} = (y_{c1} - y_{c0})\boldsymbol{i} - (x_{c1} - x_{c0})\boldsymbol{j} \\ \boldsymbol{n}_{v1} = -\boldsymbol{n}_{v0} \end{cases}, \tag{3b}$$

where $\boldsymbol{n}_{c0}$ and $\boldsymbol{n}_{c1}$ are the normal vectors outward from the main control volumes $c0$ and $c1$, respectively, and $\boldsymbol{n}_{v0}$ and $\boldsymbol{n}_{v1}$ are the normal vectors outward from the dual control volumes $v0$ and $v1$, respectively.

With the connectivity associating nodes, edges, and control volumes specified in this fashion the computation of fluxes and the assembly of the semidiscrete equations can be efficiently performed in a single sweep over all main edges in the tessellation. Furthermore, since the outward normal vectors differ only in sign for control volumes sharing a common edge, the per-edge flux components are computed only once and assigned concurrently to adjacent control volumes.

Jacobi point iteration is adopted here as the solver for the linearized discrete equations. Consequently, the form common to all discrete equation sets is simply

$$A^{\phi}\phi = \boldsymbol{b}^{\phi}, \tag{4}$$

where ϕ is the vector of generic unknowns, A^{ϕ} is the diagonal coefficient of ϕ, and $\boldsymbol{b}^{\phi}$ contains source terms as well as off-diagonal neighbor contributions to the conservation equations. Ultimately, all discrete equations are written in the form of Eq. (4). Consequently, the assembly procedure reduces to evaluating A^{ϕ} and $\boldsymbol{b}^{\phi}$ for each dependent variable (i.e., u, v, or p) at each grid point in the domain. During assembly each edge makes contributions to A^{ϕ} and $\boldsymbol{b}^{\phi}$ with valid expressions for the conservation laws available only after all edges in the tessellation have contributed and all control volumes have been closed.

3.2. *Momentum Equations*

Integrating the momentum equations over the dual control volumes leads to the semidiscrete form

$$\frac{d}{dt}(\rho \boldsymbol{v}\Omega) + \boldsymbol{R}^{\rho \boldsymbol{v}} = 0, \tag{5}$$

where the spatial momentum residuals are given by

$$\boldsymbol{R}^{\rho \boldsymbol{v}} = \sum [\rho \boldsymbol{v}(\boldsymbol{v}\cdot\boldsymbol{n}) + p\boldsymbol{n} - \boldsymbol{n}\cdot\boldsymbol{\tau}]_e + B\Omega, \tag{6}$$

where $\boldsymbol{\tau}$ is the viscous stress tensor and the summation is taken over all edges forming a closed dual control volume. Per-edge contributions to the diagonal coefficient and source term, denoted by A_e^{ϕ} and $\boldsymbol{b}_e^{\phi}$, respectively, are given below for the convective, viscous, and pressure fluxes forming the steady-state momentum balance. In the context of edge-based methods the per-edge flux contributions are often referred to as "edge formulas."

Convective flux. The momentum flux through a dual edge

$$[\rho \boldsymbol{v}(\boldsymbol{v}\cdot\boldsymbol{n})]_e = [\boldsymbol{v}f]_e \tag{7}$$

is evaluated using the trapezoidal rule in conjunction with a linear reconstruction [16] of the velocity based on the upwind value of $\boldsymbol{v}$ and its gradient $\boldsymbol{v}\nabla$. Sampling the edge mass flux, f_e, at the dual-edge midpoint yields

$$f_e = \frac{\rho}{2}(\boldsymbol{v}_{c0} + \boldsymbol{v}_{c1})\cdot\boldsymbol{n}_{v0}, \tag{8}$$

where the cell-centered velocities, $\boldsymbol{v}_{c0}$ and $\boldsymbol{v}_{c1}$, are taken to be the average of the main mesh vertex values surrounding $c0$ and $c1$.

Linearly reconstructing the edge velocity, $\boldsymbol{v}_e$, at the dual-edge midpoint using upwind data yields

$$\text{if } f_e > 0 \quad \boldsymbol{v}_e = \boldsymbol{v}_{v0} + (\boldsymbol{v}\nabla)_{v0}\cdot\boldsymbol{r}_{v0} \quad \text{else } \boldsymbol{v}_e = \boldsymbol{v}_{v1} + (\boldsymbol{v}\nabla)_{v1}\cdot\boldsymbol{r}_{v1}, \tag{9}$$

where $\boldsymbol{r}$ is the vector from the main mesh vertex to the mid-edge sample point shown in Fig. 2. Recalling that $\boldsymbol{n}_{v1} = -\boldsymbol{n}_{v0}$, the per-edge contributions to the convective flux are then

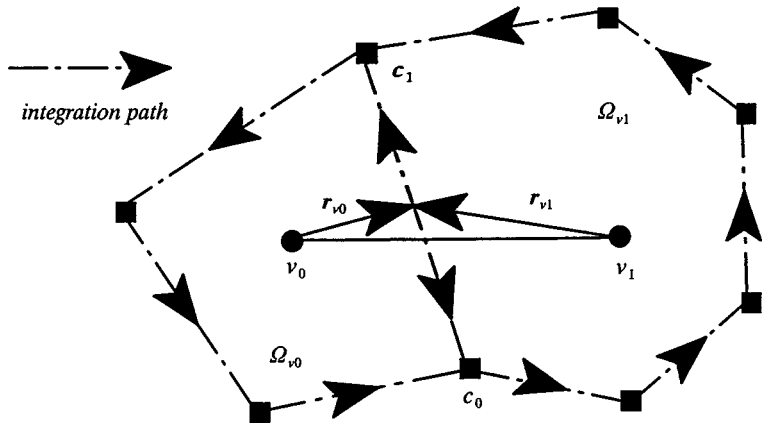

FIG. 2. Integration path used for evaluating $\boldsymbol{v}\nabla$ for convective fluxes.

(assembling at $v0$)

$$\text{if } f_e > 0 \begin{cases} A_e = f_e \\ \boldsymbol{b}_e = -f_e((\boldsymbol{v}\nabla)_{v0} \cdot \boldsymbol{r}_{v0}) \end{cases} \quad \text{else } \begin{cases} A_e = 0 \\ \boldsymbol{b}_e = -f_e(\boldsymbol{v}_{v1} + (\boldsymbol{v}\nabla)_{v1} \cdot \boldsymbol{r}_{v1}) \end{cases} \tag{10a}$$

and (assembling at $v1$)

$$\text{if } f_e > 0 \begin{cases} A_e = 0 \\ \boldsymbol{b}_e = f_e(\boldsymbol{v}_{v0} + (\boldsymbol{v}\nabla)_{v0} \cdot \boldsymbol{r}_{v0}) \cdot \boldsymbol{r}_{v1}). \end{cases} \quad \text{else } \begin{cases} A_e = -f_e \\ \boldsymbol{b}_e = f_e((\boldsymbol{v}\nabla)_{v1} \end{cases} \tag{10b}$$

The velocity gradient tensor $\boldsymbol{v}\nabla$ is computed using Green's theorem along the contour of dual edges enclosing each velocity node as shown in Fig. 2. For a generic scalar variable ϕ, Green's theorem in the plane states

$$\int_\Omega \nabla\phi \, d\Omega = \oint_\Gamma \phi \boldsymbol{n} \, d\Gamma. \tag{11}$$

Assuming $\boldsymbol{v}\nabla$ is constant over dual control volumes and evaluating the contour integral in Eq. (11) using the trapezoidal rule, give the velocity gradient tensor,

$$\boldsymbol{v}\nabla = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix}, \tag{12}$$

where, for example, the derivatives at $v0$ are given by

$$\frac{\partial \boldsymbol{v}}{\partial x} = \frac{1}{2\Omega_{v0}} \sum (\boldsymbol{v}_{c0} + \boldsymbol{v}_{c1})(y_{c1} - y_{c0}) \tag{13a}$$

$$\frac{\partial \boldsymbol{v}}{\partial y} = -\frac{1}{2\Omega_{v0}} \sum (\boldsymbol{v}_{c0} + \boldsymbol{v}_{c1})(x_{c1} - x_{c0}). \tag{13b}$$

As with the assembly of the conservation laws, the components of $\boldsymbol{v}\nabla$ are evaluated by a single sweep over all edges in the tessellation with valid expressions being established only

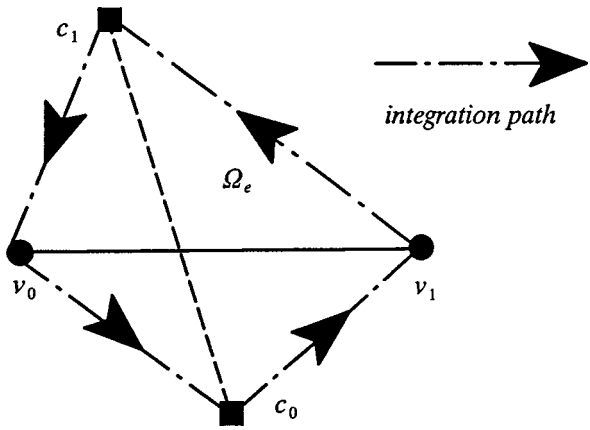

FIG. 3. Integration path used for evaluating $\boldsymbol{v}\nabla$ for viscous fluxes and ∇p for pressure equation.

after all contour paths have been closed. Finally, it may be observed that the reconstruction expressions, Eq. (10), recover the first-order upwind expressions when the velocity gradient is set to zero.

Viscous flux. The viscous stress vector at a dual edge

$$-[\boldsymbol{n}\cdot\boldsymbol{\tau}]_e \tag{14}$$

can be written for constant viscosity, divergence-free conditions as

$$-[\boldsymbol{n}\cdot\boldsymbol{\tau}]_e = -\mu[\boldsymbol{v}\nabla\cdot\boldsymbol{n}]_e, \tag{15}$$

where the Navier–Poisson law has been used to relate $\boldsymbol{\tau}$ to $\boldsymbol{v}\nabla$. Again, Green's theorem is used to evaluate the components of $\boldsymbol{v}\nabla$ over the edge area, Ω_e, shown in Fig. 3. After some manipulation, the per-edge contributions to the viscous flux become

(assembling at $v0$)

$$\begin{aligned} A_e &= \frac{\mu}{2\Omega_e}(\boldsymbol{n}_{v0}\cdot\boldsymbol{n}_{v0}) \\ \boldsymbol{b}_e &= \frac{\mu}{2\Omega_e}(\boldsymbol{n}_{v0}\cdot\boldsymbol{n}_{v0})\boldsymbol{v}_{v1} - \frac{\mu}{2\Omega_e}(\boldsymbol{v}_{c0}-\boldsymbol{v}_{c1})(\boldsymbol{n}_{c0}\cdot\boldsymbol{n}_{v0}) \end{aligned} \tag{16a}$$

and (assembling at $v1$)

$$\begin{aligned} A_e &= \frac{\mu}{2\Omega_e}(\boldsymbol{n}_{v0}\cdot\boldsymbol{n}_{v0}) \\ \boldsymbol{b}_e &= \frac{\mu}{2\Omega_e}(\boldsymbol{n}_{v0}\cdot\boldsymbol{n}_{v0})\boldsymbol{v}_{v0} + \frac{\mu}{2\Omega_e}(\boldsymbol{v}_{c0}-\boldsymbol{v}_{c1})(\boldsymbol{n}_{c0}\cdot\boldsymbol{n}_{v0}). \end{aligned} \tag{16b}$$

Pressure flux. The pressure flux at a dual edge

$$[p\boldsymbol{n}]_e \tag{17}$$

is evaluated using the trapezoidal rule giving the per-edge contributions

(assembling at $v0$)

$$\boldsymbol{b}_e = -\frac{1}{2}(p_{c0} + p_{c1})\boldsymbol{n}_{v0} \tag{18a}$$

and (assembling at $v1$)

$$\boldsymbol{b}_e = \frac{1}{2}(p_{c0} + p_{c1})\boldsymbol{n}_{v0}. \tag{18b}$$

3.3. *Time Integration*

An implicit Runge–Kutta method [17] is used to advance the velocity field governed by the semidiscrete system

$$\frac{d\boldsymbol{W}}{dt} + \boldsymbol{R} = 0 \tag{19a}$$

according to the two-stage formula

$$\boldsymbol{W}^{(1)} = \boldsymbol{W}^n - \Delta t\left(\beta_{10}\boldsymbol{R}^n + \beta_{11}\boldsymbol{R}^{(1)}\right) \tag{19b}$$

$$\boldsymbol{W}^{n+1} = \boldsymbol{W}^n - \Delta t\left(\beta_{20}\boldsymbol{R}^n + \beta_{21}\boldsymbol{R}^{(1)} + \beta_{22}\boldsymbol{R}^{n+1}\right), \tag{19c}$$

where the following set of weights yield a formally second-order-accurate method when Eq. (5) is a set of linear equations:

$$\beta_{10} = 0.0, \quad \beta_{11} = 0.2651, \quad \beta_{20} = -0.0545, \quad \beta_{21} = 0.7545, \quad \beta_{22} = 0.3.$$

Our experience has been that this two-stage method is stable and accurate for the incompressible Navier–Stokes set while having the additional benefits of being self-starting and requiring evaluation of spatial residuals at only two time levels. Furthermore, the first-order implicit Euler method can be easily recovered as a special case of the two-stage method by executing only the first stage with β_{11} set to unity.

Adopting the notation introduced above for the point-iterative solution strategy the spatial momentum residuals in Eq. (6) are given by

$$\boldsymbol{R}^{\rho\boldsymbol{v}} = A\boldsymbol{v} - \boldsymbol{b}, \tag{20}$$

and the system of nonlinear algebraic equations used to advance the velocity field from t^n to $t^{(1)}$ is then

$$\left[\boldsymbol{v}\left(\frac{\rho\Omega}{\Delta t\beta_{11}} + A\right) - \boldsymbol{b}\right]^{(1)} = \left[\frac{\rho\boldsymbol{v}\Omega}{\Delta t\beta_{11}}\right]^n. \tag{21a}$$

Similarly, the second stage of the Runge–Kutta method, which advances the velocity from $t^{(1)}$ to t^{n+1}, is given by the system

$$\left[\boldsymbol{v}\left(\frac{\rho\Omega}{\Delta t\beta_{22}} + A\right) - \boldsymbol{b}\right]^{n+1} = \left[\frac{\rho\boldsymbol{v}\Omega}{\Delta t\beta_{22}} - \frac{\beta_{20}}{\beta_{22}}(A\boldsymbol{v} - \boldsymbol{b})\right]^n - \left[\frac{\beta_{21}}{\beta_{22}}(A\boldsymbol{v} - \boldsymbol{b})\right]^{(1)}. \tag{21b}$$

In the following section Eq. (21) will be used in conjunction with mass conservation to derive a discrete equation governing the evolution of the pressure field.

3.4. *Pressure Equation*

The mass conservation equation in semidiscrete form is

$$\frac{d}{dt}(\rho\Omega) + R^{\rho} = 0 \tag{22a}$$

with the spatial mass residual given by

$$R^{\rho} = \sum [\rho \boldsymbol{v} \cdot \boldsymbol{n}]_e \tag{22b}$$

and the summation taken over all edges forming a closed main control volume. For an incompressible flow, conservation of mass reduces to

$$\sum [\boldsymbol{v} \cdot \boldsymbol{n}]_e = 0. \tag{23}$$

Since the divergence-free constraint expressed by Eq. (23) is instantaneous for incompressible flows, the velocity appearing in Eq. (23) may be taken to be either $\boldsymbol{v}^{(1)}$ or $\boldsymbol{v}^{n+1}$ during the first and second stages of time integration, respectively.

In order to facilitate the construction of the pressure–velocity coupling relation, it is convenient to rewrite the discrete form of the momentum equations for the first and second stages of time integration, Eq. (21), as

$$\boldsymbol{v}^{(1)} = \hat{\boldsymbol{v}}^{(1)} - \sum \left(\frac{\frac{1}{2}(p_{c0} + p_{c1})\boldsymbol{n}}{\frac{\rho\Omega}{\Delta t \beta_{11}} + A} \right)_e^{(1)} \tag{24a}$$

$$\boldsymbol{v}^{n+1} = \hat{\boldsymbol{v}}^{n+1} - \sum \left(\frac{\frac{1}{2}(p_{c0} + p_{c1})\boldsymbol{n}}{\frac{\rho\Omega}{\Delta t \beta_{22}} + A} \right)_e^{n+1}, \tag{24b}$$

where the pseudo-velocity vectors, $\hat{\boldsymbol{v}}^{(1)}$ and $\hat{\boldsymbol{v}}^{n+1}$, for the first and second stages, respectively, are given by

$$\hat{\boldsymbol{v}}^{(1)} = \frac{\left(\frac{\rho \boldsymbol{v} \Omega}{\Delta t \beta_{11}}\right)^n + \bar{\boldsymbol{b}}^{(1)}}{\left(\frac{\rho\Omega}{\Delta t \beta_{11}} + A\right)^{(1)}} \tag{25a}$$

$$\hat{\boldsymbol{v}}^{n+1} = \frac{\left(\frac{\rho \boldsymbol{v} \Omega}{\Delta t \beta_{22}} - \frac{\beta_{20}}{\beta_{22}}(A\boldsymbol{v} - \boldsymbol{b})\right)^n - \frac{\beta_{21}}{\beta_{22}}(A\boldsymbol{v} - \boldsymbol{b})^{(1)} + \bar{\boldsymbol{b}}^{n+1}}{\left(\frac{\rho\Omega}{\Delta t \beta_{22}} + A\right)^{n+1}}, \tag{25b}$$

where $\bar{\boldsymbol{b}}^{(1)}$ and $\bar{\boldsymbol{b}}^{n+1}$ are the spatial source terms excluding the pressure flux. These definitions are similar in spirit of the pseudo-velocities used in the SIMPLER algorithm [7].

Recalling that the mass conservation equation, Eq. (23), is enforced as a sum over edges for a main control volume, a main-edge velocity vector, $\boldsymbol{v}_e$, is proposed for facilitating the coupling of the pressure and velocity fields. Appealing to the definition of the nodal pseudo-velocity given in Eq. (25), $\boldsymbol{v}_e$ is taken at each stage of time integration to be of the form

$$\boldsymbol{v}_e = \hat{\boldsymbol{v}}_e - \left(\frac{1}{\frac{\rho\Omega}{\Delta t \beta} + A} \right)_e \int_{\Omega_e} (\nabla p)_e \, d\Omega_e. \tag{26}$$

Taking ∇p to be a constant over Ω_e allows the integral in Eq. (26) to be evaluated using Green's theorem over the edge area, Ω_e, shown in Fig. 3, with the path integral evaluated by the trapezoidal rule. After some manipulation, the integral in Eq. (26) can be written succinctly in terms of main and dual control volume normal vectors, $\boldsymbol{n}_{c0}$ and $\boldsymbol{n}_{v0}$, respectively. The resulting relation is

$$\int_{\Omega_e} (\nabla p)_e \, d\Omega_e = \frac{1}{2}((p_{c1} - p_{c0})\boldsymbol{n}_{c0} + (p_{v1} - p_{v0})\boldsymbol{n}_{v0}). \tag{27}$$

Examination of Eq. (27) reveals that ∇p has a projection normal to the main edge and a projection normal to the dual edge. When the main and dual edges are orthogonal, the second term on the right-hand side of Eq. (27) vanishes identically when Eq. (26) is substituted in Eq. (23). This is simply because of the mass conservation involves only the projection of the velocity vector and pressure gradient normal to a main control volume edge. More generally, the second term will persist when the main and dual edges of a mesh are nonorthogonal. However, in the interest of establishing the simplest pressure–velocity coupling relationship that is supported by edge data the second term on the right-hand side of Eq. (27) is neglected here and the following directed pressure difference is adopted as the integral of the pressure gradient over Ω_e:

$$\int_{\Omega_e} (\nabla p)_e \, d\Omega_e = \frac{1}{2}(p_{c1} - p_{c0})\boldsymbol{n}_{c0}. \tag{28}$$

Neglecting the second term on the right-hand side of Eq. (27) may be interpreted as a reduction in order from a piecewise linear representation of the pressure field to a piecewise constant representation. Equation (28) is consistent with the SIMPLER algorithm when implemented on structured grids [7], as well as unstructured grids using element-based finite volume methods [8, 9]. Substituting Eq. (27) into Eq. (26) yields the following pressure–velocity coupling on a per-edge basis at any instant of time:

$$\boldsymbol{v}_e = \hat{\boldsymbol{v}}_e - \frac{1}{2}\left(\frac{(p_{c1} - p_{c0})\boldsymbol{n}_{c0}}{\frac{\rho\Omega}{\Delta t\beta} + A} \right)_e. \tag{29}$$

Substituting this edge velocity into the mass conservation equation, Eq. (23), the pressure equation for the first stage of time integration is given by

$$\sum \left((\hat{\boldsymbol{v}}_e \cdot \boldsymbol{n}_{c0}) - \frac{(p_{c1} - p_{c0})\boldsymbol{n}_{c0} \cdot \boldsymbol{n}_{c0}}{2\left(\frac{\rho\Omega}{\Delta t\beta_{11}} + A\right)_e} \right)^{(1)} = 0. \tag{30}$$

Following the notation used to assemble the discrete form of the momentum equations, the per-edge contributions to the pressure equation can now be stated for the first stage of time integration. They are

(assembling at $c0$)

$$\begin{aligned} A_e &= \frac{1}{2}\left(\frac{\boldsymbol{n}_{c0} \cdot \boldsymbol{n}_{c0}}{\left(\frac{\rho\Omega}{\Delta t\beta_{11}} + A\right)_e} \right)^{(1)} \\ b_e &= \frac{1}{2}\left(\frac{\boldsymbol{n}_{c0} \cdot \boldsymbol{n}_{c0}}{\left(\frac{\rho\Omega}{\Delta t\beta_{11}} + A\right)_e} \right)^{(1)} p_{c1} - (\hat{\boldsymbol{v}}_e \cdot \boldsymbol{n}_{c0})^{(1)} \end{aligned} \tag{31a}$$

and (assembling at $c1$)

$$A_e = \frac{1}{2}\left(\frac{\boldsymbol{n}_{c0} \cdot \boldsymbol{n}_{c0}}{\left(\frac{\rho\Omega}{\Delta t \beta_{11}} + A\right)_e} \right)^{(1)}$$
$$b_e = \frac{1}{2}\left(\frac{\boldsymbol{n}_{c0} \cdot \boldsymbol{n}_{c0}}{\left(\frac{\rho\Omega}{\Delta t \beta_{11}} + A\right)_e} \right)^{(1)} p_{c0} + (\hat{\boldsymbol{v}}_e \cdot \boldsymbol{n}_{c0})^{(1)}. \tag{31b}$$

Similarly, the per-edge contributions to the pressure equation for the second stage of time integration are obtained by replacing β_{11} with β_{22} and $t^{(1)}$ with t^{n+1} in Eq. (31).

The edge values appearing in Eq. (31) are interpolated from main mesh vertex data as

$$\frac{\left(\frac{\rho\Omega}{\Delta t \beta} + A\right)_e}{\Omega_e} = \frac{1}{2}\left(\frac{\left(\frac{\rho\Omega}{\Delta t \beta} + A\right)_{n0}}{\Omega_{n0}} + \frac{\left(\frac{\rho\Omega}{\Delta t \beta} + A\right)_{n1}}{\Omega_{n1}} \right) \tag{32a}$$

$$\hat{\boldsymbol{v}}_e = \frac{1}{2}(\hat{\boldsymbol{v}}_{n0} + \hat{\boldsymbol{v}}_{n1}), \tag{32b}$$

where the area-weighted interpolation appearing in Eq. (32) properly accounts for the different control volume areas involved in the interpolation [6].

3.5. *Boundary Conditions*

Because of the semistaggered grid arrangement and the pressure–velocity coupling procedure described above, no boundary or initial conditions for pressure are required by the present algorithm. This is consistent with the basic set of conservation laws, Eq. (1), which require only the specification of initial and boundary conditions on the velocity field [5]. Furthermore, the decomposition implied by Eq. (24) is not performed for an edge on the domain boundary since no pressure–velocity coupling is sought across boundary edges. Consequently, when the pressure equation is assembled at control volumes adjacent to the domain boundary, the physical velocity flux leaving the domain through a boundary edge, $(\boldsymbol{v} \cdot \boldsymbol{n})_e$, appears naturally in place of the pseudo-velocity flux $(\hat{\boldsymbol{v}} \cdot \boldsymbol{n})_e$ appearing in Eq. (31). Therefore, the global mass flux

$$\oint_\Gamma \boldsymbol{v} \cdot \boldsymbol{n}\, d\Gamma \tag{33}$$

naturally appears in the discrete equation for pressure. Again, this is consistent with the basic conservation laws, where the global mass flux must be zero to satisfy the solvability constraint associated with the incompressible Navier–Stokes equations [5].

Although no boundary condition is required by the pressure equation presented here, inspection of Eq. (32) shows that values of A, $\hat{\boldsymbol{v}}$, and Ω are required at boundary nodes to evaluate the interpolation formulas. To this end, dual control volumes are constructed at the domain boundary to evaluate the necessary fluxes, source terms, and time fragments for each boundary node as shown in Fig. 4. The correspondence of main edges, dual edges, and normal vectors is maintained at the boundary by the edge-based data structure, allowing the boundary control volume fluxes to be evaluated and coefficients to be assembled in the same edge sweep used to assemble interior nodes. Finally, Dirichlet boundary conditions are imposed on the velocity vector at the domain boundary.

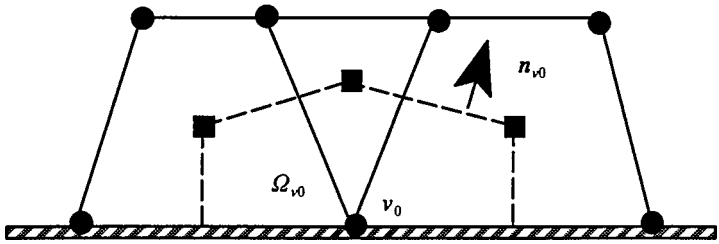

FIG. 4. Dual control volume on domain boundary.

3.6. *Overall Algorithm*

The overall solution procedure for an unsteady flow problem can now be summarized.

1. Initialize fluid velocity state at t^n, i.e., $\boldsymbol{v}^n$.
2. Compute temporal source terms for first and second stages of time integration.
3. Solve first stage of time integrator.
   - 3.1 Compute spatial momentum residuals.
   - 3.2 Compute $\hat{\boldsymbol{v}}^{(1)}$.
   - 3.3 Compute $p^{(1)}$ using Jacobi point iteration.
   - 3.4 Compute $\boldsymbol{v}^{(1)}$ using Jacobi point iteration.
   - 3.5 Return to 3.1 until convergence criteria are met.
4. State is now $(\boldsymbol{v}, p,)^{(1)}$; begin next stage.
5. Add temporal source term at $t^{(1)}$.
6. Solve second stage of time integrator.
   - 6.1 Compute spatial momentum residuals.
   - 6.2 Compute $\hat{\boldsymbol{v}}^{n+1}$.
   - 6.3 Compute p^{n+1} using Jacobi point iteration.
   - 6.4 Compute $\boldsymbol{v}^{n+1}$ using Jacobi point iteration.
   - 6.5 Return to 6.1 until convergence criteria are met.
7. State is now $(\boldsymbol{v},p,)^{n+1}$; go to next time step.

The following special cases of the unsteady algorithm may be noted.

1. For first-order time integration execute only the first stage with β_{11} set to unity.
2. For steady-state problems execute only the first stage for one time step with Δt set to infinity.

Since the momentum and pressure equations are solved iteratively in a sequential fashion for each stage, underrelaxation is incorporated into the algorithm to retard changes in the solution (principally the velocity field) from iteration to iteration in the outer loop. Relaxation is implemented according to

$$\frac{A}{\alpha}\boldsymbol{v} = b + (1-\alpha)\frac{A}{\alpha}\boldsymbol{v}^* \qquad \frac{A}{\alpha}p = b + (1-\alpha)\frac{A}{\alpha}p^*, \tag{34}$$

where $\boldsymbol{v}^*$ and p^* are the stored values of velocity and pressure from the previous outer iteration and a is the relaxation coefficient [9]. When $\alpha < 1$, the diagonal coefficient of the linear system is enhanced, resulting in a more diagonally dominant and well-conditioned system of linear equations. The preconditioning implied by Eq. (34) essentially results in a trade-off between inner loop convergence, where a low relaxation factor strongly preconditions

the linear systems of equations, and outer loop convergence, where a low relaxation factor retards the convergence of the coupled system of conservation laws. Since a rudimentary linear equation solver (Jacobi iteration) is used in the present method, the preconditioning implied by Eq. (34) significantly enhances the performance of the algorithm. Of course, other more sophisticated linear solvers could be implemented.

4. NUMERICAL RESULTS

In order to determine the accuracy and performance of the present method, a number of well-known Navier–Stokes problems are solved. For problems admitting exact solutions, the L_1 measure of the error norm is adopted,

$$e_1 = \frac{1}{N} \sum_{i=1}^{N} |\phi_{\text{computed}} - \phi_{\text{exact}}|, \tag{35}$$

where ϕ is a generic dependent variable (i.e., u, v, or p) and N is the number of unknowns.

Buoyancy-Driven Cavity Flow

Figure 5 shows the computed solution for the buoyancy-driven cavity flow problem [8, 10, 18] at $Re = 10$ using a hybrid quadrilateral–triangular mesh where the interior of the domain was triangulated using a Delaunay method [19]. The pressure field is shown to be smooth in the figure. Figure 6 shows a comparison of the computed and exact solutions at $Re = 10$ using a 10×10 uniform quadrilateral grid. The agreement is seen to be quite good. Figure 7 shows the convergence of the method with mesh refinement for a family of uniform quadrilateral and hybrid meshes at $Re = 1$ and 1000. Tables I and II give the order of the accuracy exponent computed from the data of Fig. 7. The method is seen to be second-order accurate in space. Although not shown in the figure, the L_1 measure of mass conservation error has an order of accuracy exponent greater than 3 for the cases shown.

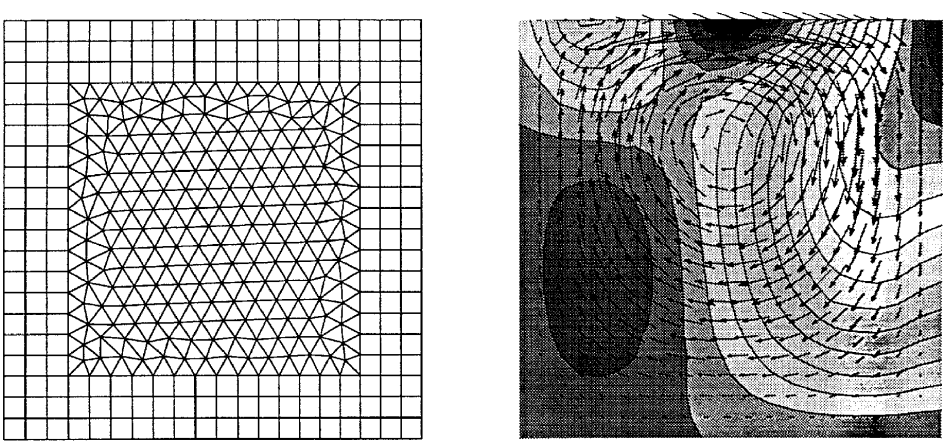

FIG. 5. Hybrid mesh and computed solution showing velocity vectors and pressure contours for buoyancy-driven cavity flow at $Re = 10$.

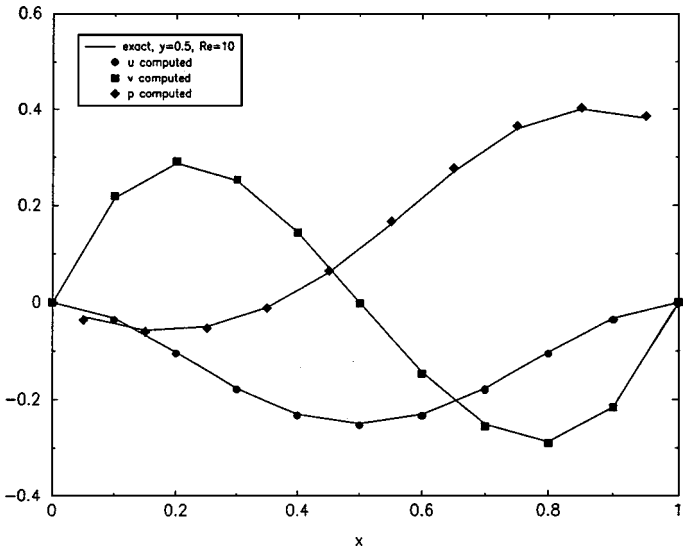

FIG. 6. Computed and exact solutions for buoyancy-driven cavity flow at $Re = 10$ on a 10×10 uniform quadrilateral mesh.

Decaying Vortex Flow

Figure 8 shows the computed solution for the decaying vortex problem [20, 21] at $Re = 100$ using a 10×10 uniform quadrilateral mesh. The pressure field is shown to be smooth in the figure. The solution shown in the figure corresponds to $t = 32$, when the velocity field has decayed to half its initial value. The figure also shows a comparison of the computed and exact solutions at $Re = 100$ using the same mesh. The agreement is seen to be quite good. Figure 9 shows the convergence of the method with mesh and

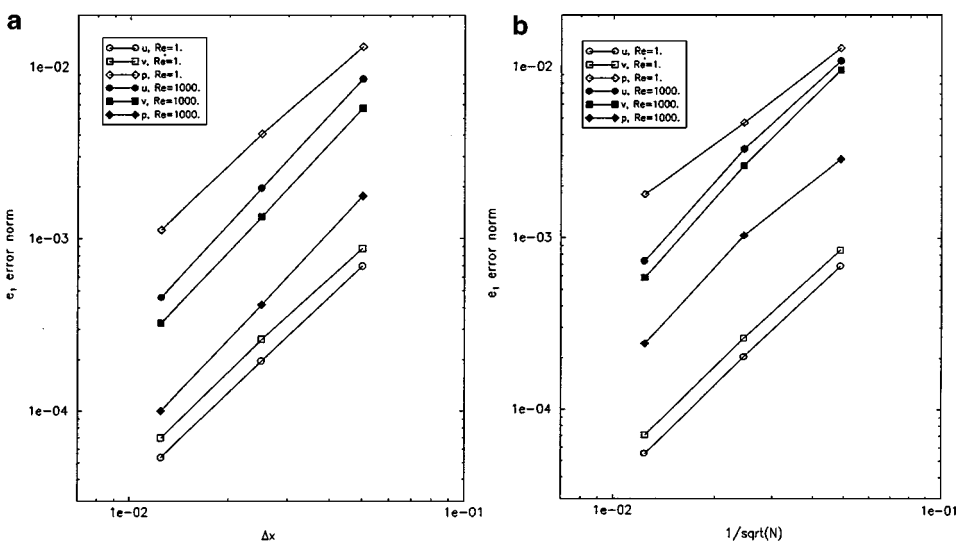

FIG. 7. Error norms versus mesh spacing for buoyancy-driven cavity flow (a) for a family of uniform quadrilateral meshes and (b) for a family of hybrid meshes.

TABLE I
Spatial Order of Accuracy for Buoyancy-Driven Cavity Problem on Uniform Quadrilateral Meshes

Variable	$Re = 1$	$Re = 1000$
u	1.9	2.1
v	1.9	2.1
p	1.9	2.0

time-step refinement for a family of uniform quadrilateral meshes and a series of time steps at $Re = 100$. Table III gives the order of accuracy exponents computed from the data of Fig. 9. The method is seen to be second-order accurate in time and space for this problem. Again, although not shown in the figure, the method is higher order in mass error.

Lid-Driven Cavity Flow

The computed velocity field for lid-driven cavity flow at $Re = 1000$ using an 80×80 uniform quadrilateral grid is shown in Fig. 10. Benchmark results [22] for the same problem are shown for comparison. Agreement with the benchmark data is quite good. The steady-state convergence performance of the method is also shown in the figure. The relaxation values used for this problem are 0.80 and 0.95 for velocity and pressure, respectively. The ability to use essentially unrelaxed values for pressure is a significant advantage offered by methods presented here in contrast to the pressure-correction methods, which typically require much lower relaxation values for convergence [10]. Timing results for this problem indicated that 0.33 CPU second per outer iteration was required on a single processor Alpha EV6 workstation rated at 1.0 GFLOPS requiring a total CPU time of 330 seconds to generate the convergence history shown in Fig. 10.

Impulsively Started Circular Cylinder Flow

The computed length of the recirculation region in the wake of an impulsively started circular cylinder at $Re = 40$ is shown in Fig. 11. Computations were done using a 100×60 quadrilateral grid with appropriate boundary conditions imposed at the symmetry boundary [17]. The outer domain boundary was located 20 diameters away from the cylinder where free-stream conditions were imposed on the velocity field.

TABLE II
Spatial Order of Accuracy for Buoyancy-Driven Cavity Problem on Hybrid Meshes

Variable	$Re = 1$	$Re = 1000$
u	1.9	2.2
v	1.9	2.2
p	1.4	2.1

TABLE III
Spatial and Temporal Order of Accuracy for Decaying Vortex Problem at $Re = 100$

Variable	Spatial	Temporal
u	3.3	1.9
v	3.3	1.9
p	2.8	1.3

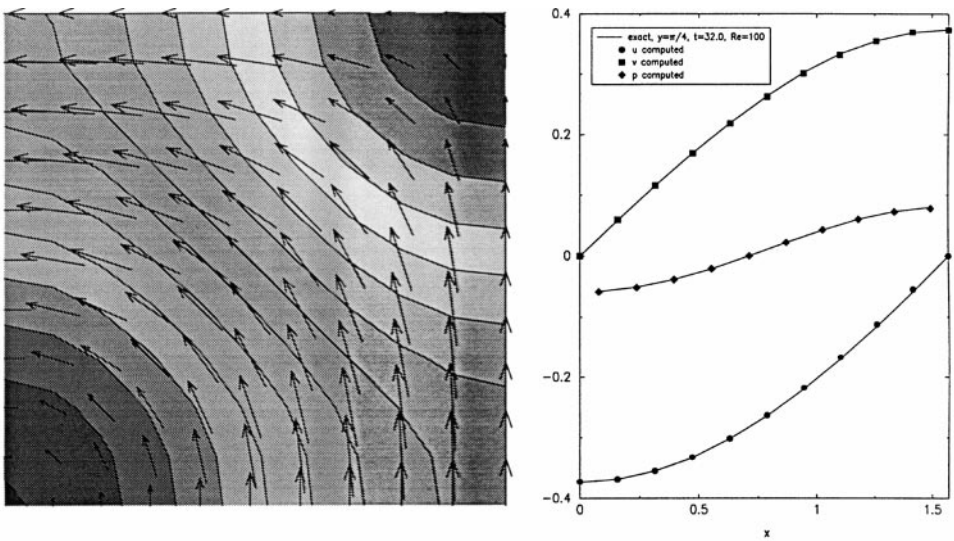

FIG. 8. Computed velocity vectors, pressure contours, and comparison with exact solution for decaying vortex flow at $Re = 100$ and $t = 32.0$ on a 10×10 uniform quadrilateral mesh using Runge–Kutta time integration, with $\Delta t = 16.0$.

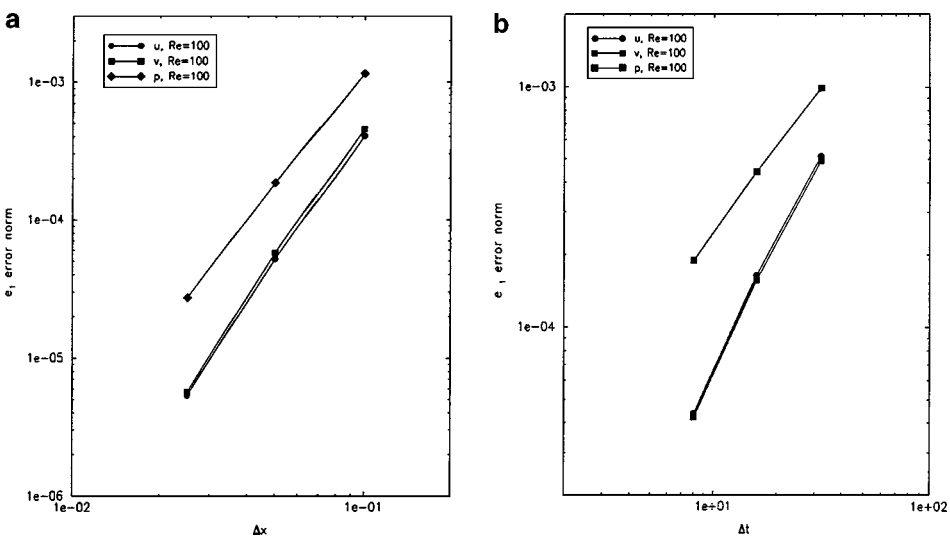

FIG. 9. Error norms for decaying vortex flow at $Re = 100$ and $t = 32.0$ (a) for a family of uniform quadrilateral meshes using Runge–Kutta time integration, with $\Delta t = 1.0$, and (b) for a series of time steps using Runge–Kutta time integration on an 80×80 uniform quadrilateral mesh.

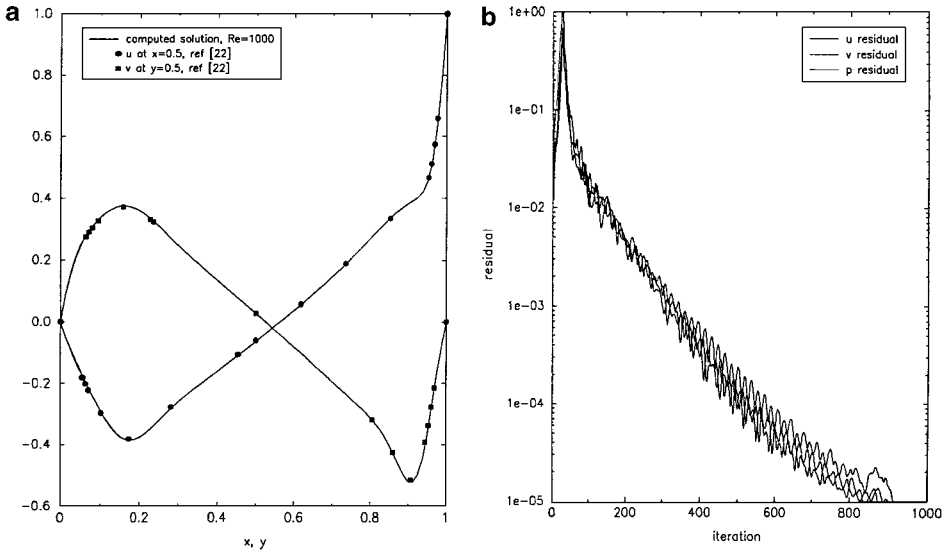

FIG. 10. (a) Computed and benchmark solutions for lid-driven flow at $Re = 1000$ on an 80×80 uniform quadrilateral mesh; (b) convergence history.

The results obtained using both the first-order Euler implicit and the second-order implicit Runge–Kutta method are shown in the figure along with experimental data [23]. Agreement with the experimental data is reasonably good using either the first- or second-order method once a time-step-independent solution is obtained. Time-step independence is achieved with $\Delta t = 0.2$ and 0.01 for the second- and first-order methods, respectively. Timing results for

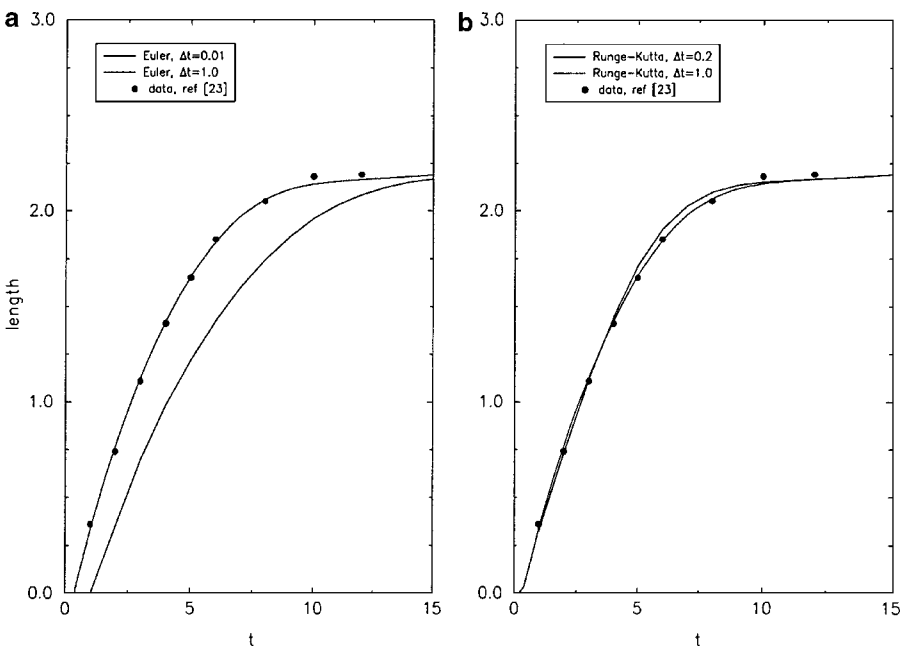

FIG. 11. Wake recirculation length for an impulsively started circular cylinder at $Re = 40$ on a 100×60 quadrilateral mesh: (a) Euler time integration; (b) Runge–Kutta time integration.

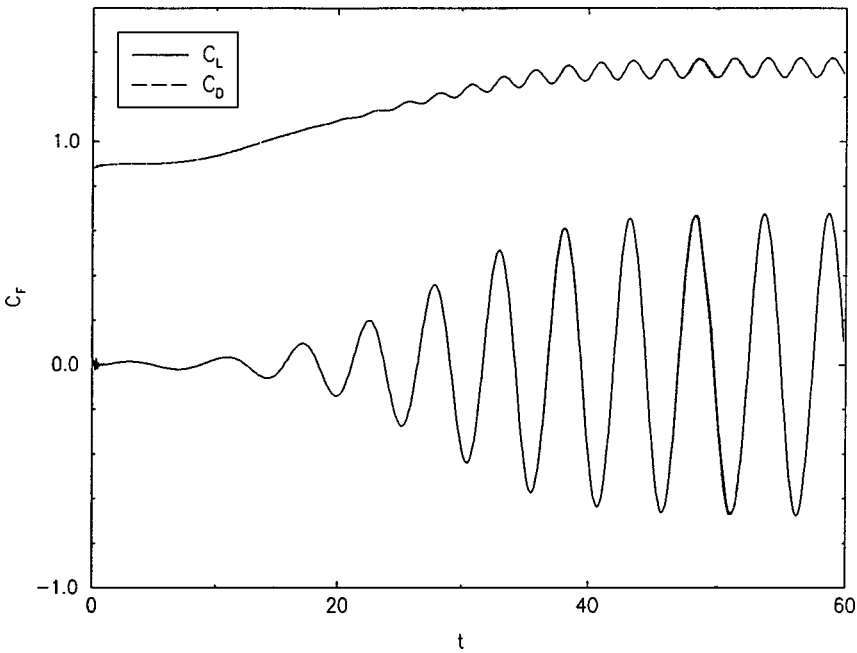

FIG. 12. Lift and drag time histories for a circular cylinder with vortex shedding at $Re = 200$ on a 200×120 quadrilateral mesh using Runge–Kutta time integration, with $\Delta t = 0.1$.

this problem indicated that the second-order implicit Runge–Kutta method required approximately one-third the total CPU time to produce time-step-independent results compared to the first-order implicit Euler method.

Vortex Shedding from a Circular Cylinder

The time history of lift and drag on a circular cylinder at $Re = 200$ is shown in Fig. 12. Computations were done using a 200×120 quadrilateral grid with the outer domain boundary located 30 diameters away from the cylinder where free-stream conditions are imposed on the velocity field [17, 21, 24]. A small asymmetric perturbation was incorporated into the initial conditions to initiate vortex shedding. Table IV summarizes and compares the present results with previously reported computations at $Re = 200$. Agreement with previous studies is quite good. The close agreement with Rogers [25] is noteworthy since a fifth-order method was used in that study.

TABLE IV
Vortex Shedding Parameters for a Circular Cylinder at $Re = 200$

Source	C_L	C_D	St
Rosenfeld *et al.* [24]	± 0.65	1.31 ± 0.04	0.20
Marx [17]	± 0.62	1.35 ± 0.04	0.195
Rogers [25]	± 0.68	1.33 ± 0.05	0.19
Present	± 0.68	1.33 ± 0.04	0.196

5. CONCLUSION

The method presented here has been demonstrated to be accurate and robust for the incompressible Navier–Stokes problems considered. The development of a pressure–velocity coupling procedure which is supported by edge data is at the heart of the method and is largely responsible for its success. The simplicity of the overall algorithm, its consistency with the boundary and initial conditions required by the Navier–Stokes equations, and its ability to accommodate arbitrary meshes are perhaps its most appealing features. Finally, the test cases reported here offer a firm foundation for extending the method to include time-dependent meshes, multigrid acceleration, and solution-dependent grid refinement.

ACKNOWLEDGMENT

The authors acknowledge the support provided by the Independent Laboratory Independent Research (ILIR) Program at the Coastal System Station, Naval Surface Warfare Center, Dahlgren Division, which is sponsored by the Office of Naval Research.

REFERENCES

1. V. Venkatakrishnan, Perspective on unstructured grid flow solvers, *AIAA J.* **34**, 533 (1996).
2. D. J. Mavriplis, Unstructured grid techniques, *Annu. Rev. Fluid Mech.* **29**, 473 (1997).
3. T. J. Barth, *Recent Developments in High Order k-Exact Reconstruction on Unstructured Meshes*, Technical Paper 93-0668 (AIAA Press, Washington, DC, 1993).
4. H. Luo, J. D. Baum, and R. Lohner, Edge-based finite element scheme for the Euler equations, *AIAA J.* **32**, 1183 (1994).
5. P. M. Gresho and R. L. Sani, On pressure boundary conditions for the incompressible Navier–Stokes equations, *Int. J. Numer. Meth. Fluids* **7**, 1111 (1987).
6. M. Thomadakis and M. Leschziner, A pressure-correction method for the solution of incompressible viscous flows on unstructured grids, *Int. J. Numer. Meth. Fluids* **22**, 581 (1996).
7. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow* (Hemisphere, Washington, DC, 1980).
8. S. Rida, F. McKenty, F. L. Meng, and M. Reggio, A staggered control volume scheme for unstructured triangular grids, *Int. J. Numer. Meth. Fluids* **25**, 697 (1997).
9. R. Jyotsna and S. P. Vanka, Multigrid calculation of steady, viscous flow in a triangular cavity, *J. Comput. Phys.* **122**, 107 (1995).
10. M. H. Kobayashi, J. M. C. Pereira, and J. C. F. Pereira, A conservative finite-volume second-order-accurate projection method on hybrid unstructured grids, *J. Comput. Phys.* **150**, 40 (1999).
11. C. Hirsch, *Numerical Computation of Internal and External Flows* (Wiley, New York, 1988), Vol. 1, pp. 24–25.
12. J. Strikwerda, Finite difference methods for the Stokes and Navier–Stokes equations, *SIAM J. Sci. Stat. Comput.* **5**, 56 (1984).
13. Y. H. Choi and C. L. Merkle, The application of preconditioning in viscous flows, *J. Comput. Phys.* **105**, 207 (1993).
14. F. Sotiropoulos and S. Abdallah, The discrete continuity equation in primitive variable solutions of incompressible flow, *J. Comput. Phys.* **95**, 212 (1991).
15. C. M. Rhie and W. L. Chow, A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation, *AIAA J.* **21**, 1525 (1983).
16. T. J. Barth and D. C. Jespersen, *The Design and Application of Upwind Schemes on Unstructured Meshes*, Technical Paper 89-0366 (AIAA Press, Washington, DC, 1989).

17. Y. P. Marx, Time integration schemes for the unsteady incompressible Navier–Stokes equations, *J. Comput. Phys.* **112**, 182 (1994).
18. T. M. Shih, C. H. Tan, and B. C. Hwang, Effects of grid staggering on numerical schemes, *Int. J. Numer. Meth. Fluids* **9**, 193 (1989).
19. S. Rebay, Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer–Watson algorithm, *J. Comput. Phys.* **106**, 125 (1993).
20. G. I. Taylor, On the decay of vortices in a viscous fluid, *Philos. Mag.* **46**, 671 (1923).
21. M. Braza, P. Chassaing, and H. Ha Minh, Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, *J. Fluid Mech.* **165**, 79 (1986).
22. U. Ghia, K. N. Ghia, and C. T. Shin, High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method, *J. Comput. Phys.* **48**, 387 (1982).
23. M. Coutanceau and R. Bouard, Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 2. Unsteady flow, *J. Fluid Mech.* **79**, 257 (1977).
24. M. Rosenfeld, D. Kwak, and M. Vinokur, A fractional step solution method for the unsteady incompressible Navier–Stokes equations in generalized coordinate systems, *J. Comput. Phys.* **94**, 102 (1991).
25. S. E. Rogers, NASA Ames Research Center, Moffett Field, CA, private communication (1999).