Journal of Computational Physics **175,** 764–791 (2002)
doi:10.1006/jcph.2001.6973, available online at http://www.idealibrary.com on IDEAL®

Accuracy and Conservation Properties of a Three-Dimensional Unstructured Staggered Mesh Scheme for Fluid Dynamics

Xing Zhang, David Schmidt, and Blair Perot

University of Massachusetts, Amherst, Massachusetts 01003-2210
E-mail: perot@ecs.umass.edu

Received April 25, 2001; revised November 9, 2001

A three-dimensional unstructured mesh discretization of the rotational from of the incompressible Navier–Stokes is presented. The method uses novel and highly efficient algorithms for interpolating the velocity vector and constructing the convention term. The resulting discretization is shown to conserve mass, kinetic energy, and vorticity to machine precision both locally and globally. The spatial accuracy of the method is analyzed and found to be second order on regular meshes and first order on irregular meshes. The numerical efficiency, accuracy, and conservation properties of the method are tested on three-dimensional meshes and found to be in agreement with theory. © 2002 Elsevier Science (USA)

Key Words: Navier–Stokes; staggered mesh; conservation; accuracy; unstructured; three-dimensional.

1. INTRODUCTION

The staggered mesh discretization referred to in this work is a discretization scheme where pressure and other scalar quantities, such as thermodynamic variables, are stored at cell centers but the velocity vector is distributed to cell faces. Each face stores only the normal component of velocity at that face. Because every three-dimensional cell contains at least four nonparallel faces, the velocity vector can always be recovered from face normal components. This recovery or interpolation of the velocity vector from the face normal components is not unique and it is a defining characteristic of various staggered mesh schemes. As will be shown in this text, particular interpolations can lead to important properties for the overall solution method. The primary difficulty of constructing conservation proofs of vector derived quantities, such as vorticity or kinetic energy, is in determining the correct definition (or interpolation) which defines the velocity vector from the primary face normal components.

Harlow and Welch [1] first described a Cartesian mesh implementation of the staggered mesh discretization in 1965. Their motivation was primarily the simplicity with which the incompressibility constraint can be implemented from within this framework. An important property of the method, which makes it popular even to this day, is the fact that discretizations of the incompressible Navier–Stokes equations that are based on this staggering scheme do not display spurious pressure modes. No particular treatment of the pressure is required to remove the red/black uncoupling found in colocated finite volume methods or the pressure locking found in some finite element and Lagrangian methods. Subsequently, it was realized that the staggered mesh approach had attractive conservation properties. While mass and momentum conservation are typically imposed explicitly via the discrete equations, Lilly [2] showed that Cartesian staggered mesh methods also conserve vorticity (or circulation in three dimensions) and conserve kinetic energy in the absence of viscosity. Perot [3] has recently shown in a two-dimensional setting that unstructured staggered mesh methods can also obtain such conservation properties. The conservation properties of a numerical method can be an indication of the methods underlying physical accuracy. As a result, staggered mesh methods are also popular in physically demanding situations such as the direct numerical simulation (DNS) of turbulence [4–6] and the large eddy simultation (LES) of turbulence [7]. The dynamic subgrid scale models used in LES appear to be particularly sensitive to the kinetic energy conserving properties of a numerical method [8], and as such, are a primary motivation for this particular research investigation.

The generalization of staggered mesh methods to unstructured meshes was proposed by Nicolaides [9] and separately by Porshing [10]. The enhanced utility to be gained by using unstructured meshes in complex geometries is obvious. However, the extension of structured staggered mesh methods to an unstructured mesh setting is more complex than it might first appear. On a Cartesian mesh, cell and face centers are relatively unambiguous. On a three-dimensional unstructured mesh, which is typically a collection of irregular tetrahedra, numerous possibilities for cell and face centers exist, resulting in a host of different staggering possibilities. This work will describe a method that uses circumcenters as the defining cell and face locations.

The attractive properties of both the Cartesian and unstructured staggered mesh methods can be traced back to the fundamental mathematical properties of the discrete difference and averaging operators. The operators have symmetry properties (such as the discrete divergence is the transpose of the discrete gradient), have operator orthogonality properties (such as the discrete curl of a discrete gradient is always zero), and satisfy discrete chain rule identities (such as discrete integration by parts analogs). These mathematical properties are relatively easy to show in a Cartesian setting. Nicolaides [11] demonstrated how the first two types of properties can be obtained for unstructured meshes through the use of cell circumcenters. Perot [3] has recently shown how chain rule identities for unstructured staggered mesh methods can be obtained using appropriately defined interpolation schemes.

This work will analyze the conservation properties and accuracy of a three-dimensional unstructured staggered mesh scheme. The scheme is described in Section 2. Its conservation properties are analyzed in Section 3, and accuracy is evaluated in Section 4. Numerical tests of the accuracy and conservation are presented in Section 5, and a 3D driven cavity problem is used to test this scheme in Section 6. A discussion of the method and its relationship to other schemes is found in Section 7.

2. NUMERICAL DISCRETIZATION

This work focuses on a discretization of the rotational form of the Navier–Stokes equations. The unstructured staggered mesh discretization can also be applied to other forms of the Navier–Stokes equations, including the much more common divergence form. As shown in Perot [3], the specific form of the Navier–Stokes equations appears to determine the conservation properties that can be obtained by the unstructured discretization. Interestingly, this is not true of Cartesian staggered mesh methods, where discretizations of the rotational and divergence forms of the Navier–Stokes equations can be shown to be equivalent. The rotational form of the incompressible Navier–Stokes equations is

$$\frac{\partial \mathbf{u}}{\partial \mathbf{t}} + (\boldsymbol{\omega} \times \mathbf{u}) = -\nabla p^d - \nabla \times (v\boldsymbol{\omega}) \tag{1a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{1b}$$

where $\mathbf{u}$ is the velocity vector, $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \mathbf{u}$ is the vorticity, $p^d = p + \frac{1}{2}\mathbf{u} \cdot \mathbf{u}$ is the specific dynamic pressure, and v is the kinematic viscosity. This particular equation assumes that viscosity is constant. Variable viscosity can be represented in the rotational form but the extra term involving second derivatives of viscosity complicates the analysis unnecessarily and will not be treated in this analysis. The rotational form of the Navier–Stokes equations is a mathematically elegant way to view the equations. The convection term acts perpendicularly to the velocity $[\mathbf{u} \cdot (\boldsymbol{\omega} \times \mathbf{u}) = 0]$, and the fluid accelerations are now explicitly decomposed into dilatational (pressure) and solenoidal (viscous) parts.

The primary quantity in staggered mesh methods is not the velocity vector, but the component of velocity normal to a face. Integrating Eq. (1a) over a cell face and taking a dot product with the face normal vector (see Fig. 1) produces an exact equation for the evolution of the mass flux, $U = \int_{A_f} \mathbf{u} \cdot \mathbf{n}\, dA$

$$\frac{\partial U}{\partial t} + \int\limits_{A_f} (\boldsymbol{\omega} \times \mathbf{u}) \cdot \mathbf{n}\, dA = -\int\limits_{A_f} \frac{\partial p^d}{\partial n}\, dA - \oint\limits_{\partial A_f} v\boldsymbol{\omega} \cdot d\mathbf{L}, \tag{2}$$

where Stokes theorem has been used to transform the viscous term from an area integral to a counterclockwise integration around the edges of the face. Counterclockwise in three dimensions is defined with respect to the normal vector and the right-hand rule. The actual orientation of the normal vector (two directions are possible) is arbitrary.

Only the vorticity component along each edge of the face is required for the viscous term. If we denote $\omega_e = \omega \cdot \mathbf{t}_e$ to be the component of vorticity along an edge (where $\mathbf{t}_e$ is a unit vector pointing along the edge), and $\hat{\omega}_e$ to be the component of vorticity along an edge that is oriented *counterclockwise* with respect to a particular face, then the viscous term can be written as $\sum^{\text{face edges}} v\hat{\omega}_e L_e$, where L_e is the length of an edge. Note that in three dimensions, each edge has two possible orientations. It is impossible to force the edge orientation to be counterclockwise with respect to all faces that are formed using that edge. For that reason, the edge orientations are taken to be arbitrary with respect to the faces and any directional dependence is accounted for with the hat (circumflex) notation, which translates into a plus or minus sign in the actual code implementation. In two dimensions, all edges point out of the plane of interest and have unit length.

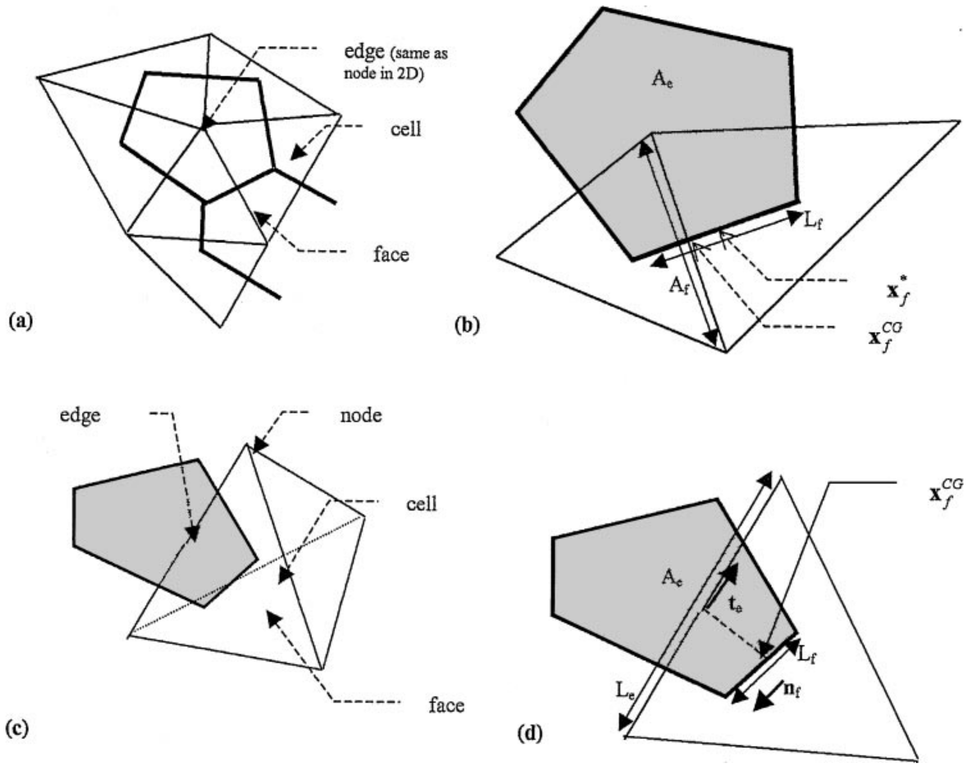

FIG. 1. (a) Two-dimensional unstructured mesh (thin lines) and its circumcenter dual mesh (bold lines). (b) Geometric parameters associated with a 2D face. (c) Three-dimensional mesh cell (thin lines), and a dual mesh polygon which is perpendicular to one of the edges (bold lines and shaded area). (d) Geometric parameters and their locations for the 3D unstructured mesh.

If pressure is assumed to be located at cell circumcenters, then the pressure term can be approximated by $-\frac{A_f}{L_f}(p^d_{c2} - p^d_{c1})$ where L_f is the distance between the two neighboring circumcenters and the normal vector points from cell c1 and toward cell c2. The fact that the pressure term can be written in this very simple form results from one of the many interesting properties of circumcenters. In particular, the property that a line connecting two neighboring cell circumcenters is always normal to the face it passes through, and that it passes through the circumcenter of that face.

The lines connecting circumcenters form an entirely new mesh, called the circumcenter dual mesh. Many of the important properties of staggered mesh discretization and averaging operators are a direct result of the local orthogonality of the primary mesh and the circumcenter dual mesh. Note that staggering places different variables on each of these mutually orthogonal meshes. Cartesian staggered meshes are trivially orthogonal to their dual mesh. In fact they are not just locally orthogonal, they are also globally orthogonal. It is this global orthogonality which allows one to prove equivalence of the rotational and divergence discretizations for Cartesian staggered meshes.

The local orthogonality property of the circumcenter dual mesh has some even more remarkable consequences than forcing cell centers to be normal to cell faces. It forces cell centers to be orthogonal to edges as well. That is, the circumcenters of all the cells with a common edge (and there can be an arbitrary number) will always lie in a plane, and that plane will be perpendicular to the edge. One remarkable consequence of this property is that the vorticity along an edge (required for the viscous term) can be obtained directly from the normal velocity components at the faces. Even more remarkable is the fact that the convection term can be approximated as

$$\int_{A_f} (\boldsymbol{\omega} \times \mathbf{u}) \cdot \mathbf{n}\, dA = -\sum^{\substack{\text{face}\\\text{edges}}} \hat{\omega}_e L_e \mathbf{u}_e^{\perp} \cdot (\mathbf{x}_e - \mathbf{x}_f^*), \tag{3a}$$

where $\mathbf{u}_e^{\perp}$ is the velocity vector in the plane perpendicular to the edge, $\mathbf{x}_e$ is the edge midpoint position, $\mathbf{x}_f^*$ is the midpoint between neighboring cell circumcenters, and $\hat{\omega}_e$ is the vorticity along the edge and oriented counterclockwise to the face normal. Note that the point $\mathbf{x}_f^*$ lies on a line perpendicular to the face circumcenter, but not necessarily on the face itself. Equation (3a) is not an intuitive approximation. It implies that an average of the product involving only one component of the vorticity and one component of the velocity at the edges is able to account for the full effects of the three-dimensional vector cross product. Locally at each edge this expression represents only half of the cross product. Somehow, the large error due to neglecting half the cross product at each edge cancels out when the average over all faces edges is taken. The derivation of Eq. (3a) is given in Section 4. Note that only the vorticity component along the edge is required in this expression as well, so if an efficient method for determining the velocity vector at the edges can be constructed, the method will be highly efficient. Such an interpolation is in fact available and is given by

$$\mathbf{u}_e^{\perp} = \mathbf{t}_e \times \frac{1}{A_e} \sum^{\substack{\text{edge}\\\text{faces}}} \hat{U} \frac{L_f}{A_f} (\mathbf{x}_e - \mathbf{x}_f^*), \tag{3b}$$

where A_e is the area of the planar polygon formed by connecting all cell circumcenters that surround the edge (see Fig. 1b), $\mathbf{t}_e$ is the tangential vector pointing along the edge, and $\hat{U} = \int_{\text{face}} \mathbf{u} \cdot \mathbf{n}_f\, dA$ is the normal mass flux with the normal oriented counterclockwise with respect to the edge. This interpolation is also not intuitive, but is formally derived in Section 4. A two-dimensional derivation of the same type of interpolation is presented in Ref. [3]. Interestingly, in matrix form, the summation in Eq. (3b) is the transpose operation of the summation in Eq. (3a). It should also be pointed out that the resulting interpolated velocity vector at the edges is actually only the velocity in the plane perpendicular to the edge. The velocity component along the edge cannot be recovered from the neighboring faces, since the normal velocity does not contain any information in the direction along the edge. However, referring back to Eq. (3a) it is clear that only the velocity in the edge plane is required for the computation to proceed, and the interpolation given by Eq. (3b) is quite adequate for our purposes.

In two dimensions, the time derivative can be left just as it appears in Eq. (2), as the change in the face normal velocity. However, it is found that for three-dimensional kinetic energy conservation a symmetric "mass matrix" is required for the time derivative term which couples it with its nearest neighbors. A discrete kinetic energy can be constructed without using this mass matrix (as in the 2D case), but the resulting discrete kinetic energy is only a zeroth order accurate approximation to the true kinetic energy on arbitrary three-dimensional meshes. It would therefore result in a statement of stability but not a true statement of kinetic energy conservation. The mass matrix is constructed by interpolating the normal velocity components to construct a cell velocity vector, and then defining the normal velocity at the face as the average of the two neighboring cell velocities. The resulting face normal velocity is still first-order accurate on general meshes. Interestingly, this new normal velocity is identical to the original normal velocity when the mesh is uniform, so this apparent averaging process is *not* diffusive. Formally, the cell velocity is defined to be

$$\mathbf{u}_c = \frac{1}{V_c} \sum^{\substack{\text{cell}\\\text{faces}}} \left(\mathbf{x}_f^{\mathrm{CG}} - \mathbf{x}_c\right)\hat{U}, \tag{4}$$

where $\mathbf{u}_c$ is the velocity vector at the cell center, V_c is the volume is this particular cell, $\mathbf{x}_f^{\mathrm{CG}}$ is the position of the center of gravity of the faces surrounding this cell, $\mathbf{x}_c$ is the position of the cell center, and $\hat{U}$ is the mass flux out of the cell. This interpolation, while unintuitive, is discussed in Ref. [3] and shown to be a first-order approximation on nonuniform meshes. The normal mass flux at a face is then given by $\frac{A_f}{L_f}[\mathbf{u}_{c1} \cdot (\mathbf{x}_f^{\mathrm{CG}} - \mathbf{x}_{c1}) - \mathbf{u}_{c2} \cdot (\mathbf{x}_f^{\mathrm{CG}} - \mathbf{x}_{c2})] = \frac{A_f}{L_f} u_{\mathrm{int}}$. It is clear that this latter approximation is also first order, and is exact if the local velocity field is constant between the two cells.

In summary the three-dimensional staggered mesh discretization of the rotational form of the Navier–Stokes equations is given by

$$\frac{\partial u_{\mathrm{int}}}{\partial t} - \frac{L_f}{A_f} \sum^{\substack{\text{face}\\\text{edges}}} \hat{\omega}_e L_e \mathbf{u}_e^{\perp} \cdot (\mathbf{x}_e - \mathbf{x}_e^*) = -\left(p_{c2}^d - p_{c1}^d\right) - \frac{L_f}{A_f} \sum^{\substack{\text{face}\\\text{edges}}} \nu \hat{\omega}_e L_e \tag{5a}$$

$$\sum^{\substack{\text{cell}\\\text{faces}}} \hat{U} = 0, \tag{5b}$$

where $\hat{U}$ is the mass flux out of the cell in question, $\omega_e = \frac{1}{A_e} \sum^{\text{edge faces}} L_f \hat{u}$ is the vorticity component along an edge, $\hat{\omega}_e$ is the vorticity component along an edge and counterclockwise to the face in question, and the velocity in the edge plane is given by $\mathbf{u}_e^{\perp} = \mathbf{t}_e \times \frac{1}{A_e} \sum^{\text{edge faces}} \hat{u} L_f (\mathbf{x}_e - \mathbf{x}_e^*)$.

The memory requirements for this method are very low and the computational efficiency is very high, requiring only a local communication stencil between faces and edges and faces and cells. To obtain conservation of kinetic energy it is not possible to use upwinding in this scheme, but upwinding can be included in this type of scheme by changing the effective cell and edge positions in the interpolations.

One critical issue for this methodology is the use of cell circumcenters to define the dual mesh. On a highly distorted mesh, a cell circumcenter can lie outside its cell. For a Delaunay primary mesh, the dual mesh is still guaranteed to be regular even when the mesh is highly distorted and the analysis follows through unaltered. Remarkably, the algorithms and analysis still hold for arbitrary meshes as well, but one must be prepared for negative face lengths and negative edge areas (the dual mesh can become inverted). The real issue is not one of implementation but of accuracy. The method is first order but intolerably inaccurate for arbitrary meshes. All the meshes used in this work are Delaunay, and a number of effective algorithms exist in the literature to generate Delaunay meshes rapidly.

Equation (5a) can be interpreted as a discretization of the momentum equation integrated along the two line segments joining the neighboring cell circumcenters and the face center of gravity. The convection and diffusion terms normal to the face are appropriate approximations for these two line segments (even though the segments are not necessarily normal to the face), because the tangential components of convection and diffusion along each segment are equal and opposite. In two dimensions, the face circumcenter and center of gravity are identical and so the line segments are automatically parallel to each other and normal to the face. This is why two-dimensional or uniform three-dimensional unstructured meshes are significantly simpler than the general three-dimensional case treated in this work.

The computation of ω_e and $\mathbf{u}_e^{\perp}$ on boundary edges requires that the integration around the edge polygon be completed. In both cases, it is the integral of the velocity tangential to the edge and along the boundary that must be specified in order to complete the integration. The tangential velocity component at boundaries is assumed to be known, either explicitly at walls or static inflow conditions, or via extrapolation from the interior for outflow conditions. It is also clear from Eq. (5a) that either the boundary mass flux can be specified, or the boundary flux can be left as an unknown and the boundary pressure specified (both the convective and diffusive terms can be calculated on boundary faces). This choice of specified pressure is ideal for outflow boundaries. In fact, a potential inflow boundary condition can also be implemented in this framework. It specifies that the vorticity is zero at the inflow and the dynamic pressure is constant along the boundary.

3. CONSERVATION PROPERTIES

3.1. Conservation of Kinetic Energy

Conservation of kinetic energy is an important property for numerical simulations of turbulence. The cascade of kinetic energy from large scales to smaller scales is one of the defining characteristics of three-dimensional turbulence. Dissipation of energy at the small scales sets the rate at which energy transfers down this cascade. Numerical methods with excessive numerical dissipation can alter simulation results even if the numerical dissipation is only at small scales [8]. This section will show that the proposed method dissipates kinetic energy at the correct rate without numerical dissipation. A simpler proof of kinetic energy conservation in two dimensions can be found in Perot [3]. Conservation of kinetic energy for distorted 2D Cartesian meshes is presented in Ref. [12].

For an incompressible fluid, the specific kinetic energy $\frac{1}{2}\mathbf{u}\cdot\mathbf{u}$ can be shown to obey the following transport equation:

$$\frac{\partial(\frac{1}{2}\mathbf{u}\cdot\mathbf{u})}{\partial t} + \nabla\cdot\left[\mathbf{u}\left(\frac{1}{2}\mathbf{u}\cdot\mathbf{u}\right)\right] = -\nabla\cdot(\mathbf{pu}) + \nabla\cdot(\nu\mathbf{u}\times\boldsymbol{\omega}) - \nu\boldsymbol{\omega}\cdot\boldsymbol{\omega}. \tag{6}$$

Except for the negative definite sink term involving the product of viscosity and enstrophy, the equation is conservative. In the absence of external forces and viscosity, the kinetic energy is simply redistributed but not created or destroyed. The discrete system should mimic Eq. (6).

To prove strict kinetic energy conservation, the evolution equation for the normal velocity component (Eq. 5a) must also be discretized in time. A midpoint rule for the discretization will be assumed,

$$\frac{u_{\text{int}}^{n+1} - u_{\text{int}}^{n}}{\Delta t} - \frac{L_f}{A_f}\sum^{\substack{\text{face}\\\text{edges}}} \hat{\omega}_e L_e \mathbf{u}_e^{\perp n+1/2}\cdot(\mathbf{x}_e - \mathbf{x}_f^*) = -\left(p_{c2}^d - p_{c1}^d\right) - \frac{L_f}{A_f}\sum^{\substack{\text{face}\\\text{edges}}} \nu\hat{\omega}_e^{n+1/2} L_e. \tag{7}$$

Note that for kinetic energy conservation, the time level of the pressure and of the vorticity in the convection term is not important.

The evolution equation for the discrete kinetic energy is constructed by first multiplying Eq. (7), the evolution equation for the normal velocity component, by $U^{n+1/2} = \frac{1}{2}(U^{n+1} + U^n)$ and then summing over faces. The summation over all the faces of a single cell results in a statement of local kinetic energy conservation for that cell. A summation over all faces in the domain results in a statement of global kinetic energy conservation. Global kinetic energy conservation is also a weak statement of numerical stability because if the kinetic energy is bounded the velocity must also be bounded.

Multiplying by $U^{n+1/2}$ and summing over faces gives

$$\begin{aligned}
&\sum^{\text{faces}} U^{n+1/2} \frac{\left(u_{\text{int}}^{n+1} - u_{\text{int}}^{n}\right)}{\Delta t} - \sum^{\text{faces}} U^{n+1/2} \frac{L_f}{A_f} \sum^{\substack{\text{face}\\\text{edges}}} \hat{\omega}_e L_e \mathbf{u}_e^{\perp n+1/2} \cdot (\mathbf{x}_e - \mathbf{x}_f^*) \\
&\quad = -\sum^{\text{faces}} U^{n+1/2}\left(p_{c2}^d - p_{c1}^d\right) - \sum^{\text{faces}} U^{n+1/2} \frac{L_f}{A_f} \sum^{\substack{\text{face}\\\text{edges}}} \nu \hat{\omega}_e^{n+1/2} L_e. \qquad (8)
\end{aligned}$$

The definition of u_{int} as a summation of two cell contributions allows us to reformulate the summation over faces as a summation over cells. The time derivative becomes

$$\begin{aligned}
\sum^{\text{faces}} \frac{\left(u_{\text{int}}^{n+1} - u_{\text{int}}^{n}\right)}{\Delta t} U^{n+1/2} &= \sum^{\text{cells}} \frac{\left(\mathbf{u}_c^{n+1} - \mathbf{u}_c^{n}\right)}{\Delta t} \cdot \sum^{\substack{\text{cell}\\\text{faces}}} \left(\mathbf{x}_f^{\text{CG}} - \mathbf{x}_c\right) \hat{U}^{n+1/2} \\
&= \sum^{\text{cells}} \frac{\left(\mathbf{u}_c^{n+1} - \mathbf{u}_c^{n}\right)}{\Delta t} \cdot V_c \mathbf{u}_c^{n+1/2} \\
&= \sum^{\text{cells}} V_c \frac{\left(\mathbf{u}_c^{n+1} - \mathbf{u}_c^{n}\right)}{\Delta t} \cdot \frac{1}{2}\left(\mathbf{u}_c^{n+1} + \mathbf{u}_c^{n}\right) \\
&= \sum^{\text{cells}} V_c \frac{\frac{1}{2}\left(\mathbf{u}_c^{n+1}\right)^2 - \frac{1}{2}\left(\mathbf{u}_c^{n}\right)^2}{\Delta t}. \qquad (9)
\end{aligned}$$

Since $\mathbf{u}_c$ is a first-order approximation for the velocity vector in an irregular unstructured mesh cell, it holds that Eq. (9) is a first-order approximation for the change in kinetic energy within the cells. If the mesh consists of cells of uniform shape and size, then the approximation becomes second order. In three dimensions, tessellation of space using uniform tetrahedra are not possible, so only meshes consisting of uniform pyramids, prisms, and hexahedra can achieve formal second-order accuracy. Note however that the first-order errors are related to the nonuniformity of the mesh. If the mesh is relatively smooth, then the first-order errors are very small and only dominate when the mesh in highly refined. For practical mesh resolutions and reasonably constructed meshes, the method will appear to be essentially second-order accurate.

In the continuous case the rotational convection term is perpendicular to the velocity vector and doesn't contribute to the kinetic energy equation. The same behavior should also be true of the discrete equations. To see that this is indeed the case, the summation over faces is recast into a summation over edges, where $\hat{U}$ is the normal velocity component oriented counterclockwise with respect to the edge in question:

$$\sum^{\text{faces}} U^{n+1/2}\frac{L_f}{A_f}\sum^{\substack{\text{face}\\\text{edges}}}\hat{\omega}_e L_e \mathbf{u}_e^{\perp n+1/2}\cdot(\mathbf{x}_e-\mathbf{x}_f^*)$$
$$=\sum^{\text{edges}}\omega_e L_e\mathbf{u}_e^{\perp n+1/2}\cdot\sum^{\substack{\text{edge}\\\text{faces}}}(\mathbf{x}_e-\mathbf{x}_f^*)\frac{L_f}{A_f}\hat{U}^{n+1/2}. \tag{10}$$

Discrete kinetic energy conservation therefore requires that $\mathbf{u}_e^{\perp n+1/2}$ be perpendicular to $\sum^{\text{edge faces}}(\mathbf{x}_e-\mathbf{x}_f^*)\frac{L_f}{A_f}\hat{U}^{n+1/2}$. This is most easily arranged by defining the edge velocity as

$$\mathbf{u}_e^{\perp n+1/2}=\mathbf{t}_e\times\frac{1}{A_e}\sum^{\substack{\text{edge}\\\text{faces}}}(\mathbf{x}_e-\mathbf{x}_f^*)\frac{L_f}{A_f}\hat{U}^{n+1/2}, \tag{11}$$

where $\mathbf{t}_e$ is the unit tangential vector pointing along an edge and A_e is the area of the dual mesh polygon associated with an edge. It is shown in Section 4 that this is a first-order approximation for the velocity vector perpendicular to mesh edges (second order on uniform meshes). This very efficient vector interpolation formula uses only a local communication stencil and the normal velocity components.

The pressure term can be recast into a summation over cells

$$-\sum^{\text{faces}} U^{n+1/2}\left(p_{c2}^d-p_{c1}^d\right)=\sum^{\text{cells}} p_c^d\sum^{\substack{\text{cell}\\\text{faces}}}\hat{U}^{n+1/2}-\sum^{\substack{\text{boundary}\\\text{faces}}}\hat{U}^{n+1/2}p_b^d$$
$$=-\sum^{\substack{\text{boundary}\\\text{faces}}}\hat{U}^{n+1/2}\left(p+\frac{1}{2}\mathbf{u}\cdot\mathbf{u}\right)_b, \tag{12}$$

where $\hat{U}$ is the normal mass flux (velocity times area) pointing out of the cell in question. When using Gauss' divergence theorem it can be seen that $\frac{1}{V_c}\sum^{\text{cell faces}}\hat{U}$ is an approximation for $\nabla\cdot\mathbf{u}$ in the cell. For the incompressible flows of interest in this work, this constraint will be forced to machine precision zero in order to compute the pressure, and only the contribution from the boundary faces remains. The first part of Eq. (12) is a discrete version of the identity $-\mathbf{u}\cdot(\nabla p)=p(\nabla\cdot\mathbf{u})-\nabla\cdot(\mathbf{u}p)$. The last equality shows that both the pressure-work term and the advection of kinetic energy arise from the dynamic pressure gradient.

The viscous term must be analyzed in two steps. First, the summation over faces is recast as a summation over edges:

$$-\sum^{\text{faces}} U^{n+1/2}\frac{L_f}{A_f}\sum^{\substack{\text{face}\\\text{edges}}}\nu\hat{\omega}_e^{n+1/2}L_e=-\sum^{\text{edges}}\nu\omega_e^{n+1/2}L_e\sum^{\substack{\text{edge}\\\text{faces}}}\hat{U}^{n+1/2}\frac{L_f}{A_f}. \tag{13}$$

A first-order approximation for the vorticity component along an edge can be obtained by using Stokes' Curl theorem $\omega_e=\frac{1}{A_e}\sum^{\text{edge faces}}\hat{U}\frac{L_f}{A_f}$. This approximation is second-order accurate for uniform meshes and is possible because of the mutual orthogonality of the circumcenter dual mesh. However, the expression for the vorticity must be modified for boundary edges because the faces connected to a boundary edge do not define a complete circuit around the edge as required by Stokes theorem. So at boundary edges we have

$$\omega_e = \frac{1}{A_e} \sum^{\substack{\text{edge}\\\text{faces}}} \hat{U} \frac{L_f}{A_f} + \frac{1}{A_e} \mathbf{u}_e^{\perp} \cdot \left(\mathbf{x}_{f2}^b - \mathbf{x}_{f1}^b\right), \tag{14}$$

where $\mathbf{x}_{f1}^b$ is the circumcenter position of the first boundary face touching the boundary edge, and $\mathbf{x}_{f2}^b$ is the circumcenter position of the second boundary face (moving counterclockwise about the edge).

Combining Eqs. (13) and (14) we find that

$$\begin{aligned} &-\sum^{\text{faces}} U^{n+1/2} \frac{L_f}{A_f} \sum^{\substack{\text{face}\\\text{edges}}} v\hat{\omega}_e^{n+1/2} L_e \\ &= -\sum^{\text{edges}} v\omega_e^{n+1/2} L_e A_e \omega_e^{n+1/2} + \sum^{\substack{\text{boundary}\\\text{edges}}} v\omega_e^{n+1/2} L_e \mathbf{u}_e^{\perp} \cdot \left(\mathbf{x}_{f2}^b - \mathbf{x}_{f1}^b\right). \end{aligned} \tag{15}$$

The first term on the right-hand side can be recast as a summation over nodes, and the second term as a summation over boundary faces:

$$\begin{aligned} &-\sum^{\text{faces}} U^{n+1/2} \frac{L_f}{A_f} \sum^{\substack{\text{face}\\\text{edges}}} v\hat{\omega}_e^{n+1/2} L_e \\ &= -\sum^{\text{nodes}} v \sum^{\substack{\text{node}\\\text{edges}}} \omega_e^{n+1/2} \omega_e^{n+1/2} \tfrac{1}{2} L_e A_e + \sum^{\substack{\text{boundary}\\\text{faces}}} v \sum^{\substack{\text{face}\\\text{edges}}} \hat{\omega}_e^{n+1/2} L_e \mathbf{u}_e^{\perp n+1/2} \times \left(\mathbf{x}_{f2}^b - \mathbf{x}_e^b\right). \end{aligned} \tag{16}$$

While the summation over node edges only involves one component of the vorticity, it will be shown in Section 4 that this is a second-order approximation on uniform meshes for the enstrophy in the dual mesh polyhedron surrounding the node. The summation over boundary face edges in the last term is an approximation for $(\mathbf{u} \times \omega) \cdot \mathbf{n}_f$ defined previously for the convective term. Therefore, these two terms are discrete analogs of the last two terms in Eq. (6).

The final discrete kinetic energy transport equation can be written as

$$\begin{aligned} &\sum^{\text{cells}} V_c \frac{\frac{1}{2}\left(\mathbf{u}_c^{n+1}\right)^2 - \frac{1}{2}\left(\mathbf{u}_c^{n}\right)^2}{\Delta t} + \sum^{\substack{\text{boundary}\\\text{faces}}} \hat{U}^{n+1/2} \left(\frac{1}{2}\mathbf{u} \cdot \mathbf{u}\right)_f \\ &= -\sum^{\substack{\text{boundary}\\\text{faces}}} \hat{U}^{n+1/2} p_b - \sum^{\text{nodes}} v \sum^{\substack{\text{node}\\\text{edges}}} \omega_e^{n+1/2} \omega_e^{n+1/2} \tfrac{1}{2} L_e A_e \\ &\quad + \sum^{\substack{\text{boundary}\\\text{faces}}} v \sum^{\substack{\text{face}\\\text{edges}}} \hat{\omega}_e^{n+1/2} L_e \mathbf{u}_e^{\perp n+1/2} \cdot \left(\mathbf{x}_{f2}^b - \mathbf{x}_e^b\right) \end{aligned} \tag{17}$$

if the cell kinetic energy is defined as

$$K_c = \frac{1}{2}(\mathbf{u}_c)^2 \tag{18}$$

Then Eq. (17) can be written in a more concise notation which highlights the similarity with the continuous kinetic energy equation, Eq. (6),

$$\sum^{\text{cells}} V_c \frac{K_c^{n+1} - K_c^n}{\Delta t} + \sum^{\substack{\text{boundary}\\ \text{faces}}} \hat{U}^{n+1/2} K_f$$
$$= -\sum^{\substack{\text{boundary}\\ \text{faces}}} \hat{U}^{n+1/2} p_b - \sum^{\text{nodes}} \nu V_n E^{n+1/2} + \sum^{\substack{\text{boundary}\\ \text{faces}}} \nu A_f (\mathbf{u} \times \boldsymbol{\omega} \cdot \mathbf{n})_f^{n+1/2}, \tag{19}$$

where $E^{n+1/2} = \frac{1}{V_n} \sum^{\text{node edges}} \boldsymbol{\omega}_e^{n+1/2} \boldsymbol{\omega}_e^{n+1/2} \frac{1}{2} L_e A_e$ is the node enstrophy. If desired, the summation over nodes could be reapportioned as summation over cells using a volume weighting. More importantly, this is the only term in the equation that causes a decrease in the discrete kinetic energy. It is negative semi-definite and the lack of other negative definite terms implies that no artificial damping of the kinetic energy occurs. The lack of any other source terms is a weak proof of the numerical stability of the overall solution method. The kinetic energy conservation property applies to the entire domain (global conservation) and each individual mesh cell (local conservation). Local kinetic energy conservation is a rare property of numerical methods, and unknown to the authors in the context of numerical methods for unstructured meshes, with the exception of the proposed formulation.

Note that the time level of the boundary face pressure and face kinetic energy do not affect kinetic energy conservation. However, the viscous term must be evaluated at the half-time level in order to guarantee positive definite dissipation, and the velocity in the convection term must be evaluated at the half-time level in order for the rotational convection term to be correctly eliminated from the discrete kinetic energy evolution equation.

3.2. Conservation of Vorticity

Conservation of vorticity is just as critical as kinetic energy conservation in large-eddy and direct numerical simulations of turbulence. Turbulence is an inherently vortical phenomenon, and the critical energy transfer among different spatial scales is intimately connected with three-dimensional vortex stretching. Vorticity is also a critical physical parameter in many laminar viscous flows, particularly those involving boundary layers, shear layers, and bluff body separation.

The continuous equation governing the evolution of vorticity is determined by taking the curl of the momentum equation. The curl operation eliminates the dynamic pressure and splits the rotational convection into two terms. One represents the convection of vorticity by the mean flow, and the other represents the increase or decrease of vorticity due to stretching or contraction of the vortex along its length. Note that the vortex stretching term is still conservative in the sense that total vorticity is not created or destroyed by this term. Stretching of a vortex tube increases the vorticity with the tube, but the stretching also decreases the cross-sectional area of the tube, so that the total vorticity remains the same.

The equation for vorticity evolution in a constant viscosity, incompressible flow is given by

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \mathbf{u} \cdot \nabla \boldsymbol{\omega} = \boldsymbol{\omega} \cdot \nabla \mathbf{u} + \nabla^2(\nu \boldsymbol{\omega}). \tag{20a}$$

Due to the incompressibility of the velocity vector and the vorticity vector, this can be put into conservative form, which is unambiguously written in Cartesian tensor notation as

$$\frac{\partial \omega_i}{\partial t} + (u_j \omega_i - \omega_j u_i)_{,j} = (\nu \omega_i)_{,jj}. \tag{20b}$$

This section will endeavor to construct the discrete analog of Eq. (20b). The discrete vorticity equation is constructed by taking the discrete curl of the discrete momentum equation normal to mesh faces, Eq. (5a), and then interpolating this result (the vorticity equation along edges) to a vector vorticity equation at mesh nodes.

The modified vorticity along an edge is taken to be $\tilde{\omega}_e = \frac{1}{A_e} \sum^{\text{edge faces}} L_f \hat{u}_{\text{int}}$. This is a slightly different definition for the edge vorticity than used previously in the convection and diffusion terms, since it employs the mass matrix, but it is equally accurate. The definition of vorticity must be augmented at boundary edges, but boundary edges will be ignored for the present. The equation for the evolution of the discrete vorticity along interior edges is then

$$A_e \frac{\tilde{\omega}_e^{n+1} - \tilde{\omega}_e^n}{\Delta t} + \sum^{\substack{\text{edge}\\\text{faces}}} L_f \hat{c}_f = -\sum^{\substack{\text{edge}\\\text{faces}}} L_f \hat{d}_f, \tag{21}$$

where $c_f = -\frac{1}{A_f} \sum^{\text{face edges}} \hat{\omega}_e L_e \mathbf{u}_e^{\perp} \cdot (\mathbf{x}_e - \mathbf{x}_f^*)$ is the rotational convection term and $d_f = \frac{1}{A_f} \sum^{\text{face edges}} \nu \hat{\omega} L_e$ is the rotational diffusion term. The hat on each term indicates that the term is evaluated in the direction counterclockwise to the edge in question. As in the continuous case, the pressure term has been eliminated. The fact that a discrete curl can be defined which identically zeros any gradient terms is another property of the unstructured staggered mesh methods related to the local orthogonality of the circumcenter dual mesh.

The equation for edge vorticity components is transformed into an equation for the discrete vorticity vector evolution by employing an interpolation to the mesh nodes

$$\sum^{\substack{\text{node}\\\text{edges}}} \left(\mathbf{x}_e^{\text{CG}} - \mathbf{x}_n\right) A_e \frac{\tilde{\omega}_e^{n+1} - \tilde{\omega}_e^n}{\Delta t} + \sum^{\substack{\text{node}\\\text{edges}}} \left(\mathbf{x}_e^{\text{CG}} - \mathbf{x}_n\right) \sum^{\substack{\text{edge}\\\text{faces}}} L_f(\hat{c}_f + \hat{d}_f) = 0, \tag{22}$$

where the edges are assumed to point away from the node in question, and $\mathbf{x}_e^{\text{CG}}$ is the position of the center of gravity of the dual mesh polygon associated with the edge (see Fig. 1b). It is shown in Section 4 that $\frac{1}{V_n} \sum^{\text{node edges}} (\mathbf{x}_e^{\text{CG}} - \mathbf{x}_n) A_e \tilde{\omega}_e$ is a first-order approximation for the vorticity vector at the nodes (second order on uniform meshes). So the first term of Eq. (22) becomes an approximation for the change in time of the vorticity vector. The second term, involving the convection and diffusion, requires more consideration, particularly in three dimensions. It simplifies as

$$\sum^{\substack{\text{node}\\\text{edges}}} \left(\mathbf{x}_e^{\text{CG}} - \mathbf{x}_n\right) \sum^{\substack{\text{edge}\\\text{faces}}} L_f(\hat{c}_f + \hat{d}_f) = \sum^{\substack{\text{node}\\\text{faces}}} \left[\left(\mathbf{x}_{e1}^{\text{CG}} - \mathbf{x}_n\right) - \left(\mathbf{x}_{e2}^{\text{CG}} - \mathbf{x}_n\right)\right] L_f(\hat{c}_f + \hat{d}_f), \tag{23}$$

where $(\hat{c}_f + \hat{d}_f)$ points counterclockwise with respect to edge 1, and all edges point away from the node in question. The important consideration is that each face has exactly two edges which touch a particular node and the node position therefore cancels out. Substituting the face position for node position and reformulating back to a summation over node edges gives

$$\sum^{\substack{\text{node}\\ \text{faces}}} \left[\left(\mathbf{x}_{e1}^{CG} - \mathbf{x}_f^*\right) - \left(\mathbf{x}_{e2}^{CG} - \mathbf{x}_f^*\right)\right] L_f(\hat{c}_f + \hat{d}_f) = -\sum^{\substack{\text{node}\\ \text{edges}}} \sum^{\substack{\text{edge}\\ \text{faces}}} \left[\left(\mathbf{x}_f^* - \mathbf{x}_{e1}^{CG}\right)\right] L_f(\hat{c}_f + \hat{d}_f). \tag{24}$$

The last summation is a first-order approximation of the convection and diffusion terms interpolated to the edges and lying in the plane perpendicular to the edge. The normal component of convection and diffusion cannot be determined from neighboring face information, which is entirely perpendicular to the edge. Interestingly, it is also not necessary. The same interpolation was used to determine the velocity vector at the edges for use in the convection term. With this transformation, it is possible to write the equation for the discrete vorticity vector at a node as

$$V_n \frac{\boldsymbol{\omega}_n^{n+1} - \boldsymbol{\omega}_n^n}{\Delta t} + \sum^{\substack{\text{node}\\ \text{edges}}} \mathbf{t}_e \times \mathbf{c}_e^\perp A_e = -\sum^{\substack{\text{node}\\ \text{edges}}} \mathbf{t}_e \times \mathbf{d}_e^\perp A_e, \tag{25}$$

where $\mathbf{t}_e$ is the unit vector pointing outward along the edge, and $\mathbf{c}_e^\perp$ and $\mathbf{d}_e^\perp$ are first-order approximation to the vector convection and diffusion terms interpolated to the edges and lying in the plane perpendicular to the edge.

In the case of convection, $\mathbf{c} = \boldsymbol{\omega} \times \mathbf{u}$ and $\mathbf{c}_e^\perp = \mathbf{t}_e \times (\boldsymbol{\omega}_e \mathbf{u}_e^\perp - \boldsymbol{\omega}_e^\perp u_e)$, so

$$\mathbf{t}_e \times \mathbf{c}_e^\perp = -\omega_e \mathbf{u}_e^\perp + u_e \boldsymbol{\omega}_e^\perp = -(\omega_e \mathbf{u}_e^\perp + \omega_e \mathbf{t}_e u_e) + (u_e \boldsymbol{\omega}_e^\perp + \omega_e \mathbf{t}_e u_e) = -\omega_e \mathbf{u}_e + u_e \boldsymbol{\omega}_e, \tag{26}$$

where ω_e is the vorticity along the edge, $\mathbf{u}_e^\perp$ is the velocity in the plane perpendicular to the edge, u_e is the velocity component along the edge, and $\boldsymbol{\omega}_e^\perp$ is the vorticity in the plane perpendicular to the edge. This form of the convection term is equivalent to the convection term found in the continuous vorticity equation, Eq. (20b).

The diffusion term is $\mathbf{d} = \nabla \times v\boldsymbol{\omega}$. If for simplicity we assume the viscosity to be constant, then

$$\begin{aligned} \frac{1}{v}\left(\mathbf{t}_e \times \mathbf{d}_e^\perp\right) &= -\frac{\partial \boldsymbol{\omega}^\perp}{\partial n} + (\nabla \omega_e)^\perp = -\frac{\partial \boldsymbol{\omega}^\perp}{\partial n} - \frac{\partial(\mathbf{n}\omega_e)}{\partial n} + (\nabla \boldsymbol{\omega}_e)^\perp + \frac{\partial(\mathbf{n}\omega_e)}{\partial n} \\ &= -\frac{\partial \boldsymbol{\omega}}{\partial n} + (\nabla \omega_e), \end{aligned} \tag{27}$$

where $(\nabla\omega_e)^\perp$ is the gradient of the outward pointing vorticity component in the plane perpendicular to the edge. The first equality can be obtained by hypothesizing an orthogonal coordinate system aligned with the edge so that $\mathbf{t}_e = (0, 0, 1)$ and $\mathbf{d}_e^\perp = (\frac{\partial \omega_e}{\partial t_2} - \frac{\partial \omega_2}{\partial n}, \frac{\partial \omega_1}{\partial n} - \frac{\partial \omega_e}{\partial t_1}, 0)$. The final expression is equivalent to $-(\omega_{i,j} - \omega_{j,i})n_j$ in Cartesian tensor notation. The first part constitutes the standard diffusion term. When multiplied by the edge area and summed over the node edges the second part is equivalent to $(\omega_{j,i})_{,j}$ which is identically zero. It is unlikely that this term will be identically zero in the discrete implementation but it still constitutes a flux term and will not affect conservation.

In summary, the vorticity evolution equation (Eq. (25)) can be written as

$$V_n \frac{\boldsymbol{\omega}_n^{n+1} - \boldsymbol{\omega}_n^n}{\Delta t} + \sum^{\substack{\text{node}\\\text{edges}}} A_e[\hat{u}_e \boldsymbol{\omega}_e - \hat{\omega}_e \mathbf{u}_e] = \sum^{\substack{\text{node}\\\text{edges}}} A_e v \left[\frac{\partial \boldsymbol{\omega}}{\partial \hat{n}} - (\nabla \hat{\omega}_e) \right], \tag{28}$$

which is the discrete equivalent of Eq. (20b). It indicates that the unstructured staggered mesh discretization of the rotational form the Navier–Stokes equations is identical to a particular *unstaggered* control volume discretization of the vorticity evolution equation. As a result, the method conserves vorticity. Note however that the method differs from standard vorticity evolution methods in two very fundamental ways. First, boundary conditions are imposed on the primative variables, not the vorticity. Second, the vorticity is a derived quantity, not a fundamental quantity.

The relevant control volumes for the vorticity evolution equation are no longer the original mesh cells, but rather the associated Veronio dual control volumes associated with the mesh nodes, and the vorticity vector is defined at mesh nodes, not at cell centers. The exact discretization of the convection term and its associated variables (such as $\mathbf{u}_e$) is not important for vorticity conservation. However, it was shown previously that kinetic energy conservation is closely tied to the specific definitions of these variables.

The analysis presented above is for interior nodes. It is clear that a similar analysis can be performed on boundary nodes as long as the dual mesh polygonal volume surrounding a boundary node is supplemented by the appropriate boundary information.

Global vorticity conservation is a direct consequence of the fact that the convective and diffusive vorticity fluxes at interior edges are equal and opposite for the two nodes touching each edge. Consequently, the contributions from all edges cancel out, and only the boundary information associated with boundary nodes remains.

4. ACCURACY

In this section, the accuracy of the various unstructured staggered mesh approximations is evaluated. Determining accuracy of discrete operations on arbitrary unstructured meshes is extremely difficult to do using Taylor series expansions. This is particularly true of the operators described in this work which use minimal neighbor information and require extensive cancellation of large errors due to the neglected terms. There are a host of geometric identities which must be known and appropriately utilized. The use of staggering and the highly unintuitive nature of many of the proposed approximations makes the task of analyzing accuracy even more complex. The standard finite element practice of showing that the solution lies in a function space of piecewise polynomials of a certain order is not straightforward either. We choose therefore to approach the issue of accuracy from a rather unusual viewpoint, relying heavily on fundamental integral relations of multidimensional calculus such as Gauss' divergence theorem and Stokes curl theorem. This section should make it clear that this novel approach is a useful tool in the analysis of unstructured mesh accuracy.

4.1. Rotational Convection Term

This section analyses the accuracy of the approximation for the rotational convection term normal to the face, $(\boldsymbol{\omega} \times \mathbf{u}) \cdot \mathbf{n}_f$. A similar, but two-dimensional analysis is found in Ref. [3]. We begin with Stokes theorem applied to the vector quantity $[\mathbf{u} \cdot (\mathbf{x} - \mathbf{x}_0)]\boldsymbol{\omega}$ where $\mathbf{u}$ is the velocity vector, $\boldsymbol{\omega}$ is the vorticity vector, $\mathbf{x}$ is position, and $\mathbf{x}_0$ is an arbitrary origin for the position

$$\int_S \nabla \times \{[\mathbf{u} \cdot (\mathbf{x} - \mathbf{x}_0)]\boldsymbol{\omega}\} \cdot \mathbf{n}\, dA = \int_{\partial S} [\mathbf{u} \cdot (\mathbf{x} - \mathbf{x}_0)](\boldsymbol{\omega} \cdot \hat{\mathbf{t}})\, dL, \tag{29}$$

where S is a surface with normal $\mathbf{n}$, and ∂S is the boundary of the surface with unit tangential vector $\hat{\mathbf{t}}$ oriented in a counterclockwise direction around the boundary with respect to the face normal.

The surfaces of interest in this case are the faces of the mesh, and the boundary of this surface is the edges that form the face. Since the face normal does not change and the edges are linear, the integral can be rewritten as a summation of the line integral along each edge:

$$\mathbf{n}_f \cdot \int_{\text{face}} \nabla \times \{[\mathbf{u} \cdot (\mathbf{x} - \mathbf{x}_0)]\boldsymbol{\omega}\}\, dA = \sum^{\substack{\text{face}\\\text{edges}}} \hat{\mathbf{t}}_e \cdot \int_{\text{edge}} \boldsymbol{\omega}[\mathbf{u} \cdot (\mathbf{x} - \mathbf{x}_0)]\, dL. \tag{30}$$

In Cartesian tensor notation, the left-hand side of this equation can be simplified as

$$\begin{aligned} n_i \int_S \varepsilon_{ijk}(u_s r_s \omega_k)_{,j}\, dA &= n_i \int_S \varepsilon_{ijk}[u_s \omega_k r_{s,j} + r_s(u_s\omega_k)_{,j}]\, dA \\ &= n_i \int_S \varepsilon_{ijk}[u_j \omega_k + r_s(u_s \omega_k)_{,j}]\, dA, \end{aligned} \tag{31}$$

where $\mathbf{r} = \mathbf{x} - \mathbf{x}_0$ is the relative position vector, and the identity that the gradient of the position vector is the identity matrix $(r_{s,j} = \delta_{sj})$ has been used. Using the previous notation that $\hat{\omega}_e$ is the vorticity component oriented along an edge and counterclockwise to the face we can write

$$-\mathbf{n}_f \cdot \int_s \boldsymbol{\omega} \times \mathbf{u}\, dA + n_i \int_s \varepsilon_{ijk} r_s (u_s \omega_k)_{,j}\, dA = \sum^{\substack{\text{face}\\\text{edges}}} \int_{\text{edge}} \hat{\omega}_e[\mathbf{u} \cdot \mathbf{r}]\, dL. \tag{32}$$

Note that this is an exact equation. The second term cannot be unambiguously written in vector notation so it has been left in Cartesian tensor notation.

If we make the first-order assumption that the velocity field $\mathbf{u}$ and the vorticity field $\boldsymbol{\omega}$ are constant along the face and the edges, then the second term is zero and the integrals can be evaluated to give

$$A_f(\boldsymbol{\omega} \times \mathbf{u})_f \cdot \mathbf{n}_f = -\sum^{\substack{\text{face}\\\text{edges}}} L_e \hat{\omega}_e \mathbf{u}_e \cdot \{\mathbf{x}_e - \mathbf{x}_0\} = -\sum^{\substack{\text{face}\\\text{edges}}} L_e \hat{\omega}_e \mathbf{u}_e \cdot \{\mathbf{x}_e - \tilde{\mathbf{x}}_f\}, \tag{33}$$

where the position origin $\tilde{\mathbf{x}}_f$ is arbitrary for each face and need not even lie on the face itself. For conservation of kinetic energy it was shown that $\tilde{\mathbf{x}}_f$ is not arbitrary and must be the midpoint between neighboring cell circumcenters. Using first-order approximations for the velocity and the edge tangential vorticity will not affect the first-order accuracy of the approximation.

4.2. Edge Velocity

The formula for the velocity in the plane perpendicular to the edge is also derived via the Stokes theorem. In this case we consider the vector $(\mathbf{a}\cdot\tilde{\mathbf{r}})\mathbf{u}$ where $\tilde{\mathbf{r}} = \mathbf{n}\times(\mathbf{x}-\mathbf{x}_0)$, $\mathbf{u}$ is the velocity vector, $\mathbf{n}$ is the surface normal vector, $\mathbf{x}$ is position, and $\mathbf{x}_0$ is an arbitrary origin for the position.

$$\int_S \nabla\times\{(\mathbf{a}\cdot\tilde{\mathbf{r}})\mathbf{u}\}\cdot\mathbf{n}\,dA = \int_{\partial S}(\mathbf{a}\cdot\tilde{\mathbf{r}})(\mathbf{u}\cdot\hat{\mathbf{t}})\,dL, \tag{34}$$

where S is a surface and ∂S is the boundary of the surface with unit tangential vector $\hat{\mathbf{t}}$ oriented in a counterclockwise direction around the boundary with respect to the face normal.

However, unlike the surfaces of interest in the previous case, the surface of interest for the edge velocity is the dual mesh edge area. This area surrounding an edge has a normal vector and $\mathbf{t}_e$ and unit tangential vectors pointing along is periphery which are $\hat{\mathbf{n}}_f$—one for each face touching the edge:

$$\mathbf{t}_e\cdot\int_{\text{edge}}\nabla\times\{(\mathbf{a}\cdot\tilde{\mathbf{r}})\mathbf{u}\}\,dA = \sum^{\substack{\text{edge}\\\text{faces}}}\hat{\mathbf{n}}_f\cdot\int_{\text{face}}\mathbf{u}(\mathbf{a}\cdot\tilde{\mathbf{r}})\,dL. \tag{35}$$

Note that the edge integral is an integral over the plane perpendicular to the edge, and the face integrals are line integrals over the lines connecting neighboring cell circumcenters.

In Cartesian tensor notation, the left-hand side of this equation is written as

$$\begin{aligned} t_i\int_S \varepsilon_{ijk}(a_s\varepsilon_{snp}t_nr_pu_k)_{,j}\,dA &= \int_S \varepsilon_{ijk}\varepsilon_{snp}a_st_it_n[u_kr_{p,j}+r_pu_{k,j}]\,dA\\ &= \int_S[\varepsilon_{jki}\varepsilon_{jsn}a_st_it_nu_k+\varepsilon_{snp}a_st_it_nr_p\omega_i]\,dA\\ &= \int_S[(\delta_{ks}\delta_{in}-\delta_{kn}\delta_{is})a_st_it_nu_k+a_st_i\tilde{r}_s\omega_i]\,dA\\ &= \int_S a_s[u_s-t_st_nu_n+t_i\tilde{r}_s\omega_i]\,dA. \end{aligned} \tag{36}$$

Using the fact that $\mathbf{a}$ is an arbitrary constant vector, we can now write that

$$\int_{\text{edge}}[\mathbf{u}-\mathbf{t}(\mathbf{t}\cdot\mathbf{u})+\tilde{\mathbf{r}}(\mathbf{t}\cdot\boldsymbol{\omega})]\,dA = \sum^{\substack{\text{edge}\\\text{faces}}}\int_{\text{face}}(\hat{\mathbf{n}}_{\mathbf{f}}\cdot\mathbf{u})\tilde{\mathbf{r}}\,dL. \tag{37}$$

Since the surface in question is planar the normal vector, **t**, is constant, and

$$\int_{\text{edge}} [\mathbf{u}^{\perp} + \tilde{\mathbf{r}}(\mathbf{t}\cdot\boldsymbol{\omega})]\, dA = \mathbf{t} \times \sum^{\substack{\text{edge}\\ \text{faces}}} \int_{\text{face}} (\hat{\mathbf{n}}_{\mathrm{f}} \cdot \mathbf{u})\mathbf{r}\, dL, \tag{38}$$

where $\mathbf{u}^{\perp}$ is the velocity vector in the edge plane. If the velocity is constant on the face, then

$$\mathbf{u}_e^{\perp} A_e = \mathbf{t}_e \times \sum^{\substack{\text{edge}\\ \text{faces}}} L_f \hat{u}(\mathbf{x}_f^* - \mathbf{x}_e). \tag{39}$$

The second term on the left-hand side of Eq. (38) can also be eliminated by choosing the arbitrary position origin to be the center of gravity of the edge polygon. This gives a second-order approximation for the velocity vector at the edge polygon's center of gravity, but this is still a first-order approximation to the velocity at the edge itself.

The interpolation of a vector quantity from its face components has also been described by Shashkov *et al.* [13] and Hyman and Shashkov [14] in the context of two-dimensional distorted Cartesian meshes.

4.3. Node Enstrophy

The enstrophy (vorticity squared) is a quantity which is similar to the kinetic energy (velocity squared). The derivation resembles the two-dimensional formula for kinetic energy found in Ref. [13]. The primary difference is that kinetic energy is located at cell centers and enstrophy is located at mesh nodes (cell corners). The derivation starts with Gauss' divergence theorem applied on the dual mesh volume surrounding a node:

$$\int_V \nabla \cdot \{(\boldsymbol{\omega}\cdot\mathbf{r})\boldsymbol{\omega}\}\, dV = \int_{\partial V} (\boldsymbol{\omega}\cdot\mathbf{r})(\boldsymbol{\omega}\cdot\mathbf{n})\, dA, \tag{40}$$

where V is a volume, ∂V is the surface of the volume with outward pointing normal vector $\mathbf{n}$, and $\mathbf{r} = \mathbf{x} - \mathbf{x}_0$ is the position vector relative to an arbitrary origin.

In Cartesian tensor notation, the left-hand side of this equation is written as

$$\int_V (\omega_p r_p \omega_k)_{,k}\, dV = \int_V (\omega_p r_{p,k} + \omega_{p,k} r_p)\omega_k\, dV = \int_V (\omega_k + \omega_{p,k} r_p)\omega_k\, dV. \tag{41}$$

Assuming a constant vorticity in the volume gives the relation

$$(\boldsymbol{\omega}\cdot\boldsymbol{\omega})V_n = \sum^{\substack{\text{node}\\ \text{edges}}} \left\{\boldsymbol{\omega}\cdot\left(\mathbf{x}_e^{\mathrm{CG}} - \mathbf{x}_n\right)\right\}\hat{\omega}_e A_e, \tag{42}$$

where $\hat{\omega}_e$ is the vorticity component along the edge and pointing away from the node. The edge position $\mathbf{x}_e$ is taken to be halfway between two nodes, and is not the same as the center of gravity of the edge polygon, $\mathbf{x}_e^{\mathrm{CG}}$. Reformulating based on the edge position gives a position vector that is normal to the edge polygon,

$$(\boldsymbol{\omega}\cdot\boldsymbol{\omega})V_n = \sum^{\substack{\text{node}\\\text{edges}}} \left\{\boldsymbol{\omega}\cdot\left(\mathbf{x}_e^{\mathrm{CG}}-\mathbf{x}_e\right)+\boldsymbol{\omega}\cdot(\mathbf{x}_e-\mathbf{x}_n)\right\}\hat{\omega}_e A_e$$
$$= \sum^{\substack{\text{node}\\\text{edges}}} \boldsymbol{\omega}\cdot\left(\mathbf{x}_e^{\mathrm{CG}}-\mathbf{x}_e\right)\hat{\omega}_e A_e + \sum^{\substack{\text{node}\\\text{edges}}} \omega_e\omega_e A_e \frac{1}{2}L_e. \tag{43}$$

The second term on the right-hand side is the expression appearing in the kinetic energy conservation equation as the expression for volume weighted enstrophy. This remarkable expression says that the volume weighted average of just the edge vorticity magnitude squared is sufficient to give us the entire enstrophy at the node. Note, however, that the volume weights sum to a total of three not to the standard value of unity.

The first term on the right-hand side is not zero for a single node in a general nonuniform mesh. In that case this term should properly be included in the viscous flux term in the kinetic energy equation (final term in Eq. (15)). However, if we sum over all nodes in the mesh (for the total enstrophy) this term cancels out at every edge connecting two nodes and leaves no net contribution (even at the boundaries). The cancellation is a result of the fact that this is a flux term. Note that a similar contribution will be obtained from each node touching a particular edge, but the value of $\hat{\omega}_e$ is equal and opposite for each of the two nodes (since it always points outward). Since all other values are identical at the two-node, contributions at an edge will exactly cancel. And since every edge has exactly two nodes associated with it, the final result is complete cancellation of the second term of Eq. (43) if we consider a complete mesh not just a single node. The mesh in question could consist of a single cell if so desired.

4.4. Node Vorticity Vector

The formula for the vorticity vector at nodes is also a product of Gauss' divergence theorem applied to the dual mesh control volume surrounding the node. In this case, we start with

$$\int_V \nabla\cdot\{(\mathbf{a}\cdot\mathbf{r})\boldsymbol{\omega}\}\,dV = \int_{\partial V}(\mathbf{a}\cdot\mathbf{r})(\boldsymbol{\omega}\cdot\mathbf{n})\,dA, \tag{44}$$

where the vector $\mathbf{a}$ is an arbitrary constant vector. In Cartesian tensor notation the left-hand side of this equation is written as

$$\int_V (a_p r_p \omega_k)_{,k}\,dA = \int_V a_p r_{p,k}\omega_k\,dA = \int_V a_k\omega_k\,dA. \tag{45}$$

Since $\mathbf{a}$ is arbitrary, this becomes

$$\boldsymbol{\omega}_n V_n = \sum^{\substack{\text{node}\\\text{edges}}} \left(\mathbf{x}_e^{\mathrm{CG}}-\mathbf{x}_n\right)\hat{\omega}_e A_e \tag{46}$$

if we assume that vorticity is constant in the dual mesh control volume. This is a first-order approximation for the vorticity vector at the node given the vorticity component along the edges. Any first-order approximation for the vorticity component along the edges can be used and the expression remains first order. A similar approximation is used to compute the velocity vector in cells from the face normal velocity components.

5. NUMERICAL TESTS OF CONSERVATION AND ACCURACY

In order to test the conservation properties of these schemes a problem was chosen that has zero mass flux at the boundaries, but is inherently unsteady. The initial flow field of this problem involves a Rankine vortex located in the bottom left quadrant of a box. Although the problem is tested in a 3D domain (1.0 m $\times$ 1.0 m $\times$ 0.1 m) and using an unstructured tetrahedral mesh, it is a two-dimensional flow since the motion only occurs in **X–Y** plane and only the z-component of the vorticity vector is not zero. The domain is meshed with 7578 tetrahedra. The viscosity of the fluid is 0.01 m^2/s and the maximum initial velocity magnitude is 0.16 m/s. The initial tangential velocity reaches its maximum at radius $R = 0.01$ m for an initial circulation Reynolds number of 1. The initial velocity profile and computational mesh are shown in Figs. 2 and 3. All the boundaries are slip walls.

5.1. Conservation of Kinetic Energy

The discrete kinetic energy within each mesh cell was calculated at every time step using Eq. (18). The total discrete kinetic energy was then evaluated by calculating the volume weighted sum over all the cell kinetic energies,

$$KE_{\text{total}} = \sum^{\text{cells}} V_c K_c.$$

Since there is no flow across the domain boundaries, the change of this quantity with time is only due to the effect of dissipation.

In numerical tests of the vortex motion in the absence of viscosity, the total discrete kinetic energy remained constant to within six significant digits after 5000 time steps (0.05 seconds). This is about as constant as can be expected given the tolerance prescribed for the iterative solver. When viscosity is present (0.01 m^2/s), the total discrete kinetic energy as a function of time is shown in Fig. 4a. The rate of change of the kinetic energy obtained

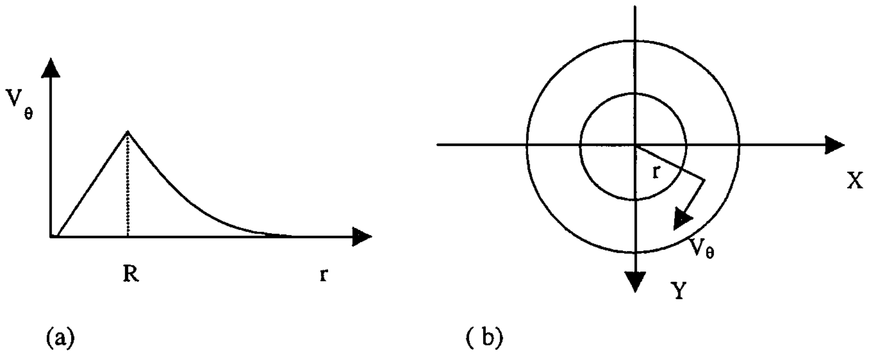

FIG. 2. Two-dimensional Rankine vortex. (a) Tangential velocity vs. radius. (b) Streamlines.

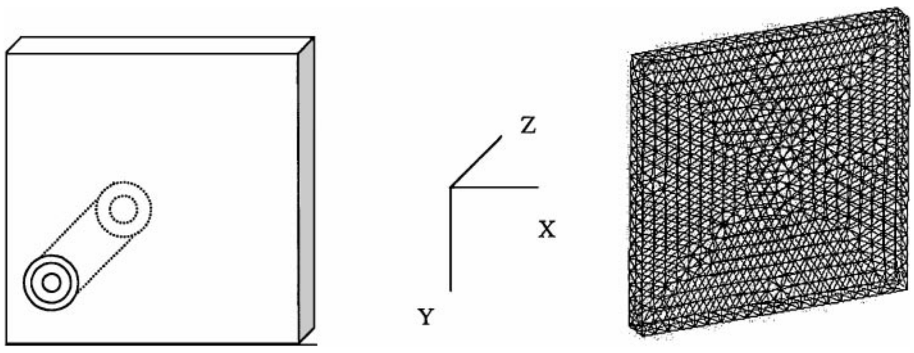

FIG. 3. Three-dimensional domain and surface mesh for the vortex test problem.

by differentiating this curve is compared with the calculated physical dissipation. A perfect match is shown in Fig. 4b. This test indicates that the theoretical analysis of Section 3.2 is well-founded and that there is no artificial dissipation in the method.

5.2. Conservation of Vorticity

The discrete vorticity within each mesh cell was calculated at every time step. The total discrete vorticity was then evaluated by calculating the volume weighted sum over all the individual node vorticity values,

$$\Omega_{\text{total}} = \sum^{\text{nodes}} V_n \omega_n .$$

Since there is no flow across the domain boundaries, and the diffusive vorticity flux is zero at the domain boundaries, this quantity should be constant, even in the presence of viscosity.

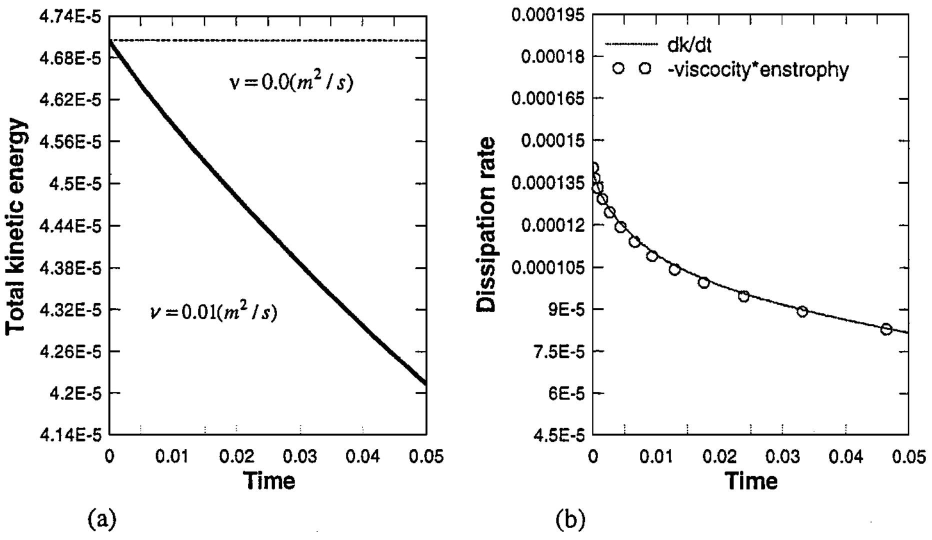

FIG. 4. Kinetic energy conservation test. (a) Total kinetic energy vs. time. (b) Rate of change of kinetic energy versus time (solid line) and total physical dissipation versus time (circles) for $v = 0.01$ m^2/s.

In this numerical test, the initial total vorticity z-component is -0.00033 ($\sec^{-1}$); all other components are zero. Final result shows that this scheme conserves the vorticity to three significant digits after the solution has evolved for 0.05 seconds (5000 time steps, roughly one vortex revolution), which is again all that can be expected given the solution tolerances of the iterative solver and the fact that vorticity has higher errors associated with it than the velocity.

5.3. Conservation of Momentum

The discrete velocity (momentum) within each mesh cell was calculated at every time step using Eq. (4). The total discrete velocity was then evaluated by calculating the volume-weighted sum over all the individual cell velocity values, $U_{\text{total}} = \sum^{\text{cells}} V_c \mathbf{u}_c$. Since there is no flow across the domain boundaries, and because we are using slip walls, this quantity should remain constant even in the presence of viscosity.

It was found that the proposed discretization conserved the total x-, y-, and z-momentum to machine precision. In this problem, the z-component velocity is always zero and the x- and y-components are symmetric about vortex axis, thus the initial total momentum is zero. The result shows that these three components remain zero (to machine procession) after 5000 time steps. The global conservation of momentum of the rotational discretization was not shown analytically in the text, but was determined to be a result of the fact that the streamfunction on the boundaries does not change with time.

5.4. Numerical Tests of Accuracy

This section confirms the accuracy assessments of the preceding sections using a series of numerical tests. In each case, a sinusoidal function is assigned to the input variables. The exact solution is computed analytically and the approximate solution is computed numerically using the approximations described in the text. Rather than changing the mesh size, which is a cube (1.0 m $\times$ 1.0 m $\times$ 1.0 m) meshed with 8838 tetrahedra (shown in Fig. 5), mesh refinement has been performed by changing the wavelength of the input sine functions. Small wavelengths are equivalent to a coarse mesh solution.

The stream function was set to be $\frac{1}{2\pi N}\sin(2\pi Nx)\cos(2\pi Ny)\vec{\mathbf{k}}$. Exact velocities and vorticity were then derived from this streamfunction. The maximum velocity magnitude is

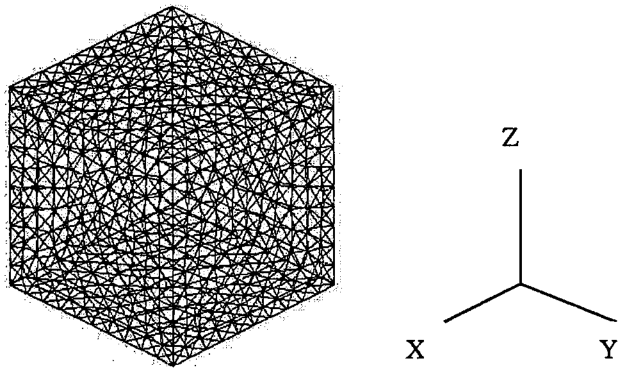

FIG. 5. Surface mesh for the accuracy tests.

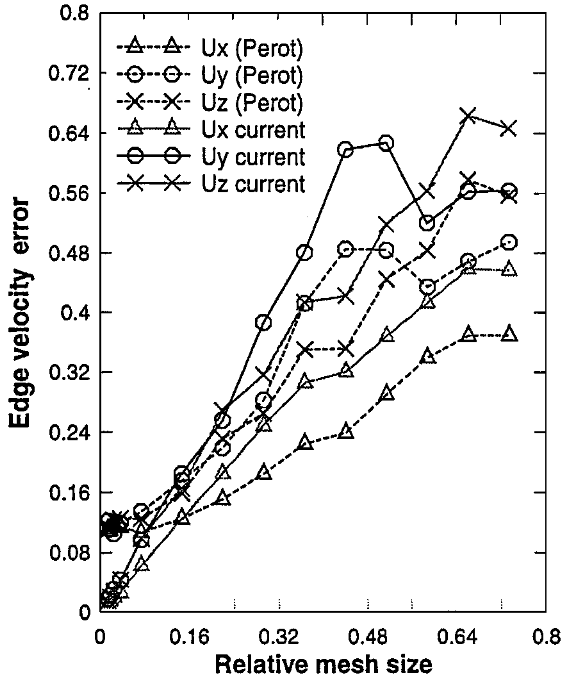

FIG. 6. Error in the edge velocity interpolation.

unity. The average mesh volume was calculated first, and then the average mesh spacing, ΔX_{avg}, was taken to be $\sqrt[3]{3.0 * V_{\text{avg}}}$ and was found to be 0.07. The effective length of the domain is $1.0/N$, where N is a variable integer value. Larger values of N correspond to a coarser effective mesh. The relative mesh size is defined to be $\Delta X_{\text{avg}}/L_{\text{eff}} = 0.07 * N$. So a relative mesh size of 1 corresponds to roughly 14 cells per wavelength. The error defined in this accuracy test is the root square mean error. Figure 6 shows the accuracy of the proposed velocity interpolation, Eq. (39), to the mesh edges. Triangle symbols are the error in the x-component of velocity, circles are the error in the y-component, and crosses are the error in the z-component. The dashed lines represent the error in the velocity interpolation required by the standard rotational form (described in Perot [3]) in which the formula is similar to Eq. (39) but $\mathbf{x}_f^*$ is replaced by $\mathbf{x}_f^{\text{cc}}$. For coarse meshes, the error of the standard scheme is

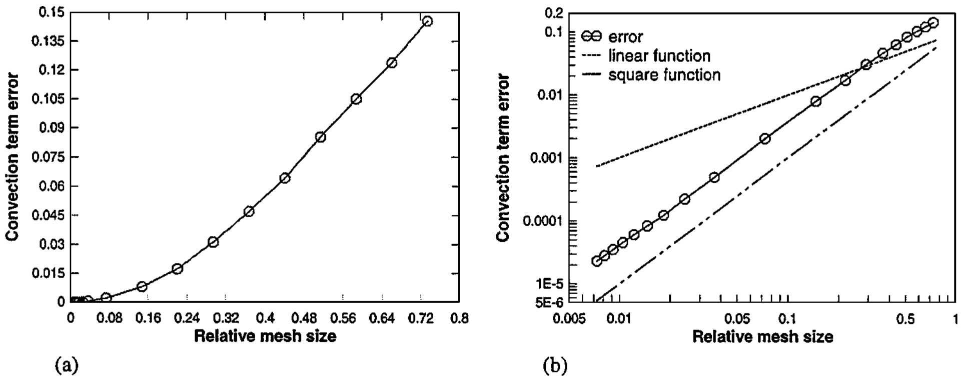

FIG. 7. Error in convection term. (a) Error versus relative mesh size. (b) Log scaling.

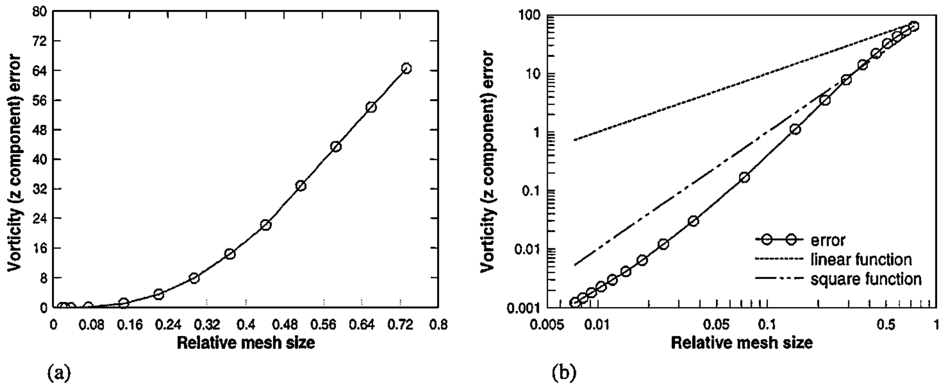

FIG. 8. Error in node vorticity. (a) Error versus mesh size. (b) Log scaling.

slightly smaller than the velocity interpolation described in this work (Eq. 39), but as the mesh is refined, it does not converge to zero. The solid lines are the current interpolation scheme, which is shown to be first-order accurate.

The approximation for the convective term is was tested by assuming exact vorticity and velocity at the mesh edges, and using Eq. (33) to calculate the normal component of $\boldsymbol{\omega} \times \mathbf{u}$ at the faces. The convective term interpolation, as shown in Fig. 7, is second-order accurate. Figure 8 shows the accuracy of the vorticity vector interpolation, Eq. (46), to the nodes. Since x- and y-components are zero to machine procession, only the accuracy of the z-component vorticity is plotted. Second-order accuracy is clearly shown in the figure. Figure 9 shows the accuracy of total node enstrophy interpolation. Second-order accuracy is achieved when Eq. (43) is used to calculate the total enstrophy. The mesh used in this test is a nearly uniform one as can be seen in the surface mesh shown in Fig. 5. On a highly nonuniform mesh we would expected to see first-order accuracy for these interpolations.

6. DRIVEN-CAVITY PROBLEM

The driven-cavity problem was chosen to test the validity of this scheme for actual fluid simulation. The computational domain and mesh are the same as that used in the conservation properties tests of Section 5. As shown in Fig. 10, the top lid is moving with a constant

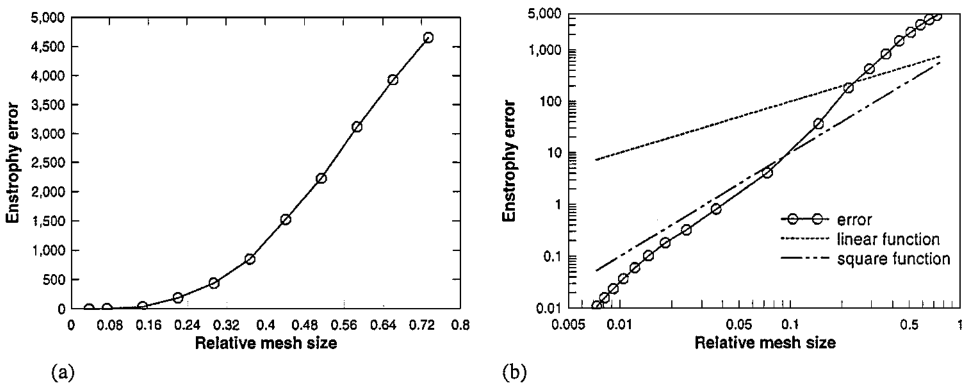

FIG. 9. Error in enstrophy. (a) Error vs. mesh size. (b) Log scaling.

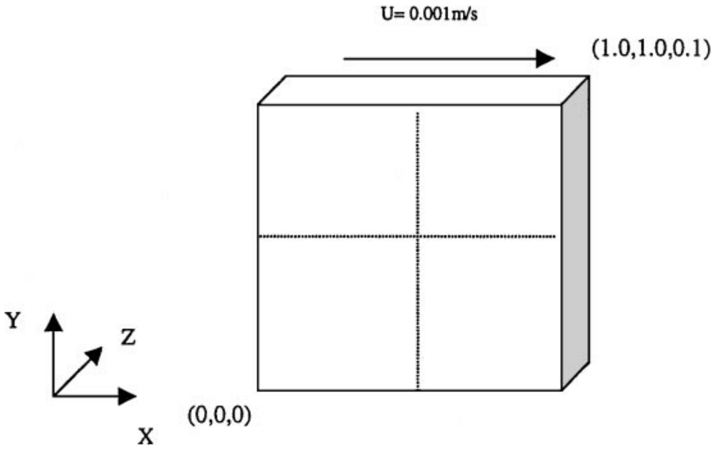

FIG. 10. Driven cavity problem.

x-component velocity ($U = 0.001$ m/s). Left, bottom, and right boundaries are solid walls. The front and back are slip walls. The viscosity of the fluid is 10^{-5} m^2/s. The Reynolds number defined as $\mathrm{Re} = \frac{UL}{v}$ is 100. After 20,000 time steps (2000 s), a steady solution is achieved. While the underlying solution is essentially two-dimensional so that it can easily be compared with other solution methods, the underlying numerical scheme is fully unstructured and three-dimensional, and does not take advantage of two-dimensionality in any way. Figure 11a shows the x-component velocity as a function of y through the center. Figure 11b shows the y-component velocity as function of x through the center. In both figures, data were extracted from the symmetrical plane $z = 0.05$.

In Fig. 11, solid lines represent the simulation results and circles represent the results from Ref. [15], in which a more refined structured Cartesian mesh is used. Given that this tetrahedra mesh has a spacing roughly equivalent to a 30×30 Cartesian mesh, the results are quite acceptable. Figure 12 shows the velocity vectors on the symmetry plane $z = 0.5$ of the steady solution. The location of the primary vortex center predicted by this scheme is (0.652, 0.717) which is reasonably close to (0.617, 0.734), the location predicted by [15]. Furthermore, by carefully studying the streamlines obtained from the integration of velocity

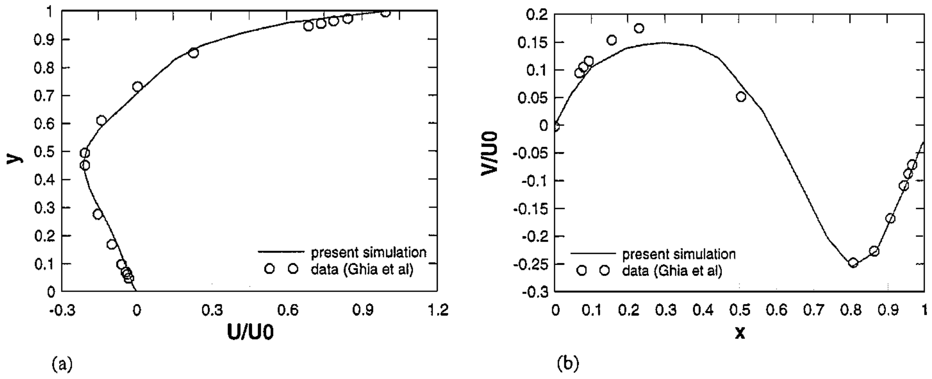

FIG. 11. Velocity profile of the driven cavity problem. (a) U velocity profile on vertical line through the center (b) V velocity profile on horizontal line through the center.

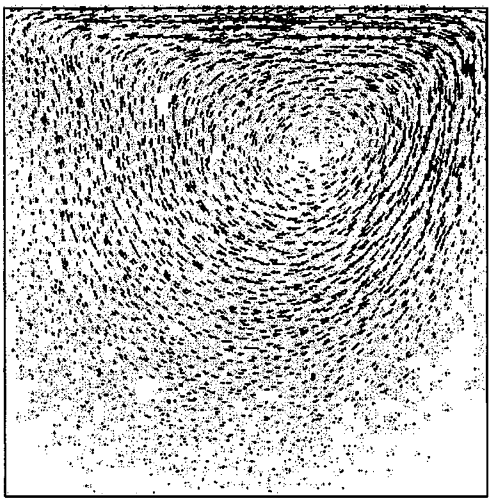

FIG. 12. Velocity vector plots for cavity flow at Re = 100.

vectors, we found that a small vortex at the right bottom corner was successfully captured (not seen in the figure) even though the mesh is relatively coarse. Due the resolution of the mesh, an even smaller vortex at the left bottom corner predicted by Ref. [15] is not captured.

7. CONCLUSIONS

The Cartesian staggered mesh scheme of Harlow and Welch [1] remains a popular scheme for the solution of the incompressible Navier–Stokes equations to this day. This is perhaps because the scheme displays a remarkable number of very attractive mathematical, physical, and practical properties. This work presents a generalization of the staggered mesh scheme to general unstructured three-dimensional meshes that displays many of these same mathematical, physical, and practical attributes. It requires low memory storage, uses very compact interpolation and discretization operators, and conserves important physical parameters beyond the primary discretization variables. The accuracy of the unstructured implementation is equivalent to the standard Cartesian staggered mesh method—first order on arbitrary meshes but effectively second order on reasonably smooth meshes and practical mesh sizes.

Other unstructured staggered mesh schemes exist in the literature [16, 17], even for three-dimensional problems [18–20]. These methods can differ in numerous ways but the fundamental differences appear to be in how vectors (particularly velocity) are interpolated from component information and what form of the Navier–Stokes equations the discretization is based on. This work describes a number of novel interpolation schemes and describes how they lead to conservation properties. It also describes a discretization based on the rotational form of the Navier–Stokes equations whereas all other previous work has focused on discretizations starting from the divergence form of the governing equations. It is entirely possible that other interpolations and other schemes also display conservation properties. Previous work [3] shows that this is certainly the case in two dimensions. It is hoped that the new methods for analyzing accuracy and conservation that are developed in this work will aid in the analysis of other unstructured staggered mesh methods.

The fact that this method can be shown to be equivalent to an unstaggered control volume discretization of the vorticity equation (Section 3.2) suggests that it is closely related to streamfunction–vorticity or velocity–vorticity methods. There is a strong connection, but also some very fundamental differences. The present formulation poses all boundary conditions on the velocity or the pressure. Unlike streamfunction–vorticity methods, boundary conditions on the vorticity or the streamfunction are not required. In addition, because the method really only advances the component of vorticity along mesh edges, the cost of the method does not triple when going from two dimensions to three dimensions, as is the case with standard streamfunction–vorticity methods. The underlying reason for these very significant differences is because the vorticity vector is a derived, rather than a primary variable. Alternatively, the distinction is the fact that the equations were discretized first and then manipulated into a vorticity-like form, rather than analytical manipulation of the exact equations into a vorticity equation followed by discretization.

The major limitations of the proposed scheme are its low-order accuracy (effectively second order), and the reliance on cell circumcenters and the local orthogonality properties of the Delaunay dual mesh. Current research is directed toward increasing the accuracy of the discrete operators, and allowing the use of nonorthogonal dual meshes such as the median dual mesh which connects cell and face centroids.

APPENDIX

A variety of methods is available to calculate center of gravity (centroid) and circumcenter locations. The best choice is largely determined by the available data structures and mesh connectivity information. In this work, we use methods that can be applied to an arbitrary mesh configuration. The center of gravity of a face is calculated as the area weighted sum of the center of gravity of each subtriangle formed by the face edge and a central point on the face (which is arbitrary). The formula is

$$\mathbf{x}_f^{\mathrm{CG}} = \frac{1}{A_f}\sum^{\substack{\text{face}\\\text{edges}}} \frac{1}{2} L_e L_{ef}^{\perp}\left(\frac{1}{3}\mathbf{x}_f + \frac{2}{3}\mathbf{x}_e\right) = \frac{1}{3}\mathbf{x}_f + \frac{2}{3}\sum^{\substack{\text{face}\\\text{edges}}} \frac{A_{ef}}{A_f}\mathbf{x}_e, \tag{A1}$$

where $L_{ef}^{\perp} = |\mathbf{t}_e \times (\mathbf{x}_e - \mathbf{x}_f) \cdot \mathbf{n}_f|$ is the perpendicular distance from the edge to the central point on the face. The center of gravity for the cell is calculated similarly, using subtetrahedra associated with each face,

$$\mathbf{x}_c^{\mathrm{CG}} = \frac{1}{V_c}\sum^{\substack{\text{cell}\\\text{faces}}} \frac{1}{3} A_f L_{fc}^{\perp}\left(\frac{1}{4}\mathbf{x}_c + \frac{3}{4}\mathbf{x}_f^{\mathrm{CG}}\right) = \frac{1}{4}\mathbf{x}_c + \frac{3}{4}\sum^{\substack{\text{cell}\\\text{faces}}} \frac{V_{fc}}{V_c}\mathbf{x}_f^{\mathrm{CG}}, \tag{A2}$$

where $L_{fc}^{\perp} = |(\mathbf{x}_c - \mathbf{x}_f^{\mathrm{CG}}) \cdot \mathbf{n}_f|$ is the perpendicular distance from the face to the central point in the cell.

Explicit formulas exist for finding the circumcenter of a triangle or a tetrahedron, but these formulas do not generalize to an arbitrary mesh (such as rectangles), which might still have well-defined circumcenters. For this reason, iterative methods are used to find the face and cell circumcenters. These methods are based on the property that the face circumcenter is located perpendicular to the edge midpoints. The iteration is based on the formula

$$\mathbf{x}_f^{\mathrm{CC}} = \mathbf{x}_f^{\mathrm{CC}} + \alpha \sum^{\substack{\text{face}\\ \text{edges}}} \mathbf{t}_e \left(\mathbf{x}_f^{\mathrm{CC}} - \mathbf{x}_e\right) \cdot \mathbf{t}_e, \tag{A3}$$

where α is a convergence parameter usually set to 1. The convergence is quite rapid, usually taking less than five iterations. The iterative formula for cell circumcenter position is similar:

$$\mathbf{x}_c^{\mathrm{CC}} = \mathbf{x}_c^{\mathrm{CC}} + \alpha \sum^{\substack{\text{cell}\\ \text{faces}}} \left\{\left(\mathbf{x}_c^{\mathrm{CC}} - \mathbf{x}_f^{\mathrm{CC}}\right) - \mathbf{n}_f \left(\mathbf{x}_c^{\mathrm{CC}} - \mathbf{x}_f^{\mathrm{CC}}\right) \cdot \mathbf{n}_f \right\}. \tag{A4}$$

This formula converges equally rapidly. The formulas are applicable to arbitrary meshes, even those without well-defined circumcenters. If a circumcenter does not exist, local orthogonality will not be achieved by these locations, which is an easy check on the mesh. The algorithm will still work on such a mesh without local orthogonality but accuracy and conservation properties are unlikely to continue to apply.

ACKNOWLEDGMENTS

This work was supported in part by the Center for Integrated Turbulence Simulation at Stanford University, by the Office of Naval Research, and by the Air Force Office of Scientific Research.

REFERENCES

1. F. H. Harlow and J. E Welch, Numerical calculations of time dependent viscous incompressible flow of fluid with a free surface, *Phys. Fluids* **8**, 2182 (1965).
2. D. K. Lilly, On the computational stability of numerical solutions of time-dependent non-linear geophysical fluid dynamics problems, *Mon. Weather Rev.* **93**(1), 11 (1965).
3. J. B. Perot, Conservation properties of unstructured staggered mesh schemes, *J. Comput. Phys.* **159**, 58 (2000).
4. J. B. Perot, Direct numerical simulation of turbulence on the Connection Machine, in *Parallel Computational Fluid Dynamics '92*, edited by R. B. Pelz, A. Ecer, and J. Hauser (North-Holland, Amsterdam, 1993).
5. Y. Na and P. Moin, *Direct Numerical Simulation of Turbulent Boundary Layers with Adverse Pressure Gradient and Separation*, Report TF-28 (Thermosciences Division, Dept. of Mech. Eng., Stanford University, 1996).
6. H. Le, P. Moin, and J. Kim, Direct numerical simulation of turbulent flow over a backward facing step, *J. Fluid Mech.* **330**, 349 (1997).
7. K. Akselvoll and P. Moin, Large eddy simulation of turbulent confined coannular jets, *J. Fluid Mech.* **315**, 387 (1996).
8. R. Mittal and P. Moin, Suitability of upwind-biased finite difference schemes for large-eddy simulation of turbulent flows, *AIAA J.* **35**(8), 1415 (1998).
9. R. A. Nicolaides, *Direct Discretization of Planar Div-Curl Problems*, ICASE Report 89-76 (1989).
10. C. A. Hall, J. S. Peterson, T. A. Porsching, and F. R. Sledge, The dual variable method for finite element discretizations of Navier/Stokes equations, *Int. J. Numer. Meth. Eng.* **21**, 883 (1985).
11. R. A. Nicolaides, The covolume approach to computing incompressible flow, in *Incompressible Computational Fluid Dynamics*, edited by M. D. Gunzburger and R. A. Nicolaides (Cambridge Univ. Press, Cambridge, UK, 1993).
12. Y. V. Apanovich and E. D. Lyumkis, Difference schemes for the Navier–Stokes equations on a net consisting of Dirichlet cells, *U.S.S.R. Comput. Maths. Math. Phys.* **28**(2), 57 (1988).
13. M. Shashkov, B. Swartz, and B. Wendroff, Local reconstruction of a vector field from its normal components on the faces of grid cells, *J. Comput. Phys.* **139**, 406 (1998).

14. J. M. Hyman and M. Shashkov, The orthogonal decomposition theorems for mimetic finite difference methods, *SIAM J. Numer. Anal.* **36**(3), 788 (1999).

15. 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).

16. C. A. Hall, J. C. Cavendish, and W. H. Frey, The dual variable method for solving fluid flow difference equations on Delaunay triangulations, *Comput. Fluids* **20**(2), 145 (1991).

17. R. A. Nicolaides, The covolume approach to computing incompressible flows, in *Algorithmic Trends in Computational Fluid Dynamics*, edited by M. Y. Hussaini, A. Kumar, and M. D. Salas (Springer-Verlag, Berlin/New York, 1993), pp. 295–333.

18. R. A. Nicolaides and X. Wu, Covolume solutions of three-dimensional div-cural equations, ICASE Report 95-4 (1995).

19. J. C. Cavendish, C. A. Hall, and T. A. Porsching, A complementary volume approach for modelling three-dimensional Navier–Stokes equations using dual Delaunay/Voronoi tessellations, *Int. J. Numer. Meth. Heat Fluid Flow* **4**, 329 (1994).

20. A. Lippolis, G. Vacca, and B. Grossman, Incompressible Navier–Stokes solutions on unstructured grids using a co-volume technique, in *13th International Conference on Numerical Methods in Fluid Dynamics*, edited by M. Napolitano and F. Sabetta (Springer-Verlag, Berlin/New York, 1992), pp. 270–274.