Journal of Computational Physics **161,** 85–113 (2000)
doi:10.1006/jcph.2000.6490, available online at http://www.idealibrary.com on IDEAL®

Vorticity Transport on a Lagrangian Tetrahedral Mesh

J. S. Marshall,* J. R. Grant,† A. A. Gossler,* and S. A. Huyer†

*Iowa Institute of Hydraulic Research and Department of Mechanical Engineering, University of Iowa, Iowa City, Iowa 52242; †Division Newport, Naval Undersea Warfare Center, Building 1302, Newport, Rhode Island 02841

Received March 15, 1999; revised February 28, 2000

An integral vorticity method for computation of incompressible, three-dimensional, viscous fluid flows is introduced which is based on a tetrahedral mesh that is fit to Lagrangian computational points. A fast method for approximation of Biot–Savart type integrals over the tetrahedral elements is introduced, which uses an analytical expression for the nearest few elements, Gaussian quadratures for moderately distant elements, and a multipole expansion acceleration procedure for distant elements. Differentiation is performed using a moving least-squares procedure, which maintains between first- and second-order accuracy for irregularly spaced points. The moving least-squares method is used to approximate the stretching and diffusion terms in the vorticity transport equation at each Lagrangian computational point. A new algorithm for the vorticity boundary condition on the surface of an immersed rigid body is developed that accounts for the effect of boundary vorticity values both on the total vorticity contained within tetrahedra attached to boundary points and on vorticity diffusion from the surface during the time step. Sample computations are presented for uniform flow past a sphere at Reynolds number 100, as well as computations for validation of specific algorithms.

Key Words: vortex methods; Navier–Stokes equations; tetrahedral mesh.

1. INTRODUCTION

The distinguishing feature of complex fluid flows observed in nature, as compared to the much simpler idealization of potential flow, is the presence of vorticity within the fluid. Vorticity is of paramount interest in incompressible fluid dynamics because, while it contains all of the dynamical information necessary for construction of the velocity and pressure fields, it nevertheless typically occupies a small subset of the flow domain. For instance, in the flow caused by a vehicle moving through an otherwise stationary, unbounded fluid, significant vorticity is contained only in the boundary layer along the vehicle surface

0021-9991/00 $35.00

and in the vehicle wake. Velocity and relative pressure, on the other hand, have significant non-zero values throughout a large region of the flow domain, decaying gradually with distance away from the vehicle. Another feature of the vorticity field is that, in the absence of viscosity, the support of vorticity moves as a material region of the flow. Viscosity causes only a gradual spreading of the vorticity support to neighboring regions of the flow field.

Accurate computation of fluid flows is only possible if the regions of the flow containing significant vorticity are sufficiently well resolved. In traditional approaches to computational fluid dynamics, the computational points are attached to some fixed (Eulerian) grid covering the entire flow field (or a truncated part of the flow field for unbounded flows). In steady flows, the location of the vorticity-containing regions may be known *a priori*, and the distribution of the computational points can be adjusted accordingly to provide sufficient resolution in these regions. In unsteady flows, the vorticity-containing regions move about and deform with time, and it is much more difficult to resolve the ever-changing vorticity support using a fixed grid. Failure to resolve vorticity with Eulerian methods gives rise to significant numerical dissipation, which results primarily from truncation errors associated with discretization of the nonlinear convection term. As discussed by Landsberg and Murman [22] and Kravchenko and Moin [21], the effects of numerical dissipation are prevalent in many of the numerical simulations reported in the literature for detached fluid flows, such as the wake behind a body, roll-up of shear layer vortices, trailing vortices behind airfoils, and large-eddy simulation of turbulent flows. In such cases, numerical dissipation causes both an enhanced spreading of the vorticity support (such as would be observed physically if the viscosity were artificially increased) and a loss of circulation of the vortex structures.

Lagrangian vorticity methods follow an alternative approach in which the computational points are placed only within the vorticity-containing regions of the flow and are allowed to move with the local fluid velocity [23, 27]. A variety of approaches have been introduced to account for the spread of vorticity support caused by viscous diffusion, which involve either advection of the computational points by an additional velocity-like quantity related to diffusion [5, 30] or adaptive methods for generation of new computational points [32, 37]. Lagrangian vorticity methods are efficient and highly self-adaptive, since computational points are advected with the vorticity-containing regions by the fluid flow. Lagrangian vorticity methods exhibit little or no numerical dissipation, since the nonlinear convection term is included in the vorticity time derivative.

Despite these advantages, Lagrangian vorticity methods experience a number of serious problems that have precluded their more general usage in the fluid dynamics community. Some of these problems have been partially resolved in recent years. For instance, computation of the Biot–Savart integral for velocity is usually performed in Lagrangian vorticity methods by interpolating the vorticity surrounding each computational point using a vorticity element centered at that point. A piecewise-continuous vorticity interpolation is obtained in cases where the vorticity is uniform across the elements, and a smooth vorticity interpolation is obtained when the vorticity associated with the elements decays smoothly with distance away from the element center (the latter case is typically called the "vortex blob" method). Although Lagrangian vorticity methods generally require many times fewer computational points than Eulerian methods to achieve a given computational accuracy, the integration procedure used to compute velocity with N elements in Lagrangian vorticity methods is $O(N^2)$, and is therefore quite slow for large N. This situation has been improved by introduction of a variety of acceleration methods [1, 6, 12], which decrease the required number of computations to $O(N \ln N)$, or in some cases to $O(N)$.

Computational efficiency is affected both by the calculation speed with a given number of computational points and by the number of computational points required to resolve the vorticity field. Because the vorticity support in many flows has the form of thin sheets (e.g., boundary layers, vortex sheets) or elongated tubes, it is highly desirable that the vorticity elements be anisotropic and able to adjust to the shape of the vorticity support. Anisotropic elements with uniform vorticity, yielding a piecewise continuous vorticity interpolation, have previously been used for this purpose by Teng [41], Bernard [3], and Huyer and Grant [16], and anisotropic elements yielding a smooth vorticity interpolation have been proposed by Marshall and Grant [24]. For problems involving flow past an immersed solid body or a density interface, a further desirable feature of the vorticity interpolation is that the vorticity field not extend over the interface surface. This condition is violated by most previous vortex method computations, resulting in inaccuracy in satisfaction of the no-slip condition at the interface.

Another problem with Lagrangian vorticity methods involves the difficulty of approximating the derivatives that occur in the viscous diffusion and (in three dimensions) the vortex stretching terms of the vorticity transport equation. The older types of Lagrangian vorticity methods simulate viscous diffusion using a stochastic "random walk" procedure [5]. While this procedure is stable for large time steps, it is only accurate for extremely small values of the time step [9] and is hence very inefficient. A variety of different deterministic diffusion procedures have been applied to Lagrangian vorticity methods in recent years. Some of these methods [7, 8] require uniformly spaced computational points in order to be accurate, which limits their usefulness for computations on Lagrangian computational points. Other methods, such as that of Shankar and van Dommelen [37], maintain good accuracy on irregular points, but require a great deal of computation time (on the same order as the velocity calculation). Approximation of the velocity derivative in the vortex stretching term is usually performed either analytically (by differentiating the velocity induced by each vorticity element and then summing over the elements) or by a finite difference approximation of the velocity derivative in the direction of vorticity [25]. The latter procedure requires two velocity calculations for each computational point (approximately doubling the computational time for large N) and the former procedure is even less efficient.

The current paper introduces a new approach to Lagrangian vorticity methods, which is able to resolve many of the remaining difficulties outlined above. Instead of employing overlapping vorticity blobs, we interpolate the vorticity field using a tetrahedral mesh that is fit to the Lagrangian computational points at each time step. A predecessor to this approach was previously described by Russo and Strain [33], who employed a triangular mesh for computation of two-dimensional, inviscid flows in an unbounded domain. This method was later extended to two-dimensional viscous flow with immersed bodies [14, 15]. The primary advantages of using a tetrahedral or triangular mesh is that (1) the interpolated vorticity field does not penetrate the surface of a solid body and (2) the vorticity elements can be fit to highly anisotropic distributions of the computational points while still yielding a continuous vorticity interpolation. The latter advantage is particularly important in three-dimensional flows with an immersed body, in which the number of isotropic elements required to resolve the body boundary layer with even modest values of the Reynolds number is excessive. When using tetrahedral elements, it is critical to employ a numerical integration procedure for calculating the velocity contribution from all but the closest elements in which the number of terms in the approximation is set to produce calculations of a prescribed accuracy.

The current paper utilizes a procedure for numerical differentiation in which a quadratic polynomial is used to provide a smooth, local interpolation of the function whose derivative is desired using a moving least-squares fit [26]. The accuracy of this procedure is second order for uniformly spaced points and between first and second order for very irregularly spaced points. Because the Lagrangian vorticity method exhibits no detectable numerical dissipation [11], we do not encounter problems with rapid diffusion commonly observed in low-order grid-based methods. The moving least- squares method is used to approximate the stretching and diffusion terms in the vorticity transport equation at each computational point.

The fundamental field equations governing the vorticity and velocity evolution and the body surface pressure are reviewed in Section 2, followed by an overview of the numerical solution procedures. Section 3 presents an efficient method for evaluation of integrals of the Biot–Savart type over the tetrahedral elements, and the accuracy and efficiency of this method are evaluated for computation of the fluid velocity field. The moving least-squares differentiation procedure is described in Section 4, and validation results are given for problems involving vorticity stretching and diffusion. A new vorticity boundary condition algorithm is described in Section 5, which makes use of the tetrahedral vorticity elements and the moving least-squares differentiation method. Issues involving mesh management and computational point generation for near-body flows are discussed in Section 6. Sample calculations for flow past a sphere are presented in Section 7. Conclusions are given in Section 8.

2. PROBLEM STATEMENT

The numerical method presented in the paper is concerned with the problem of incompressible, three-dimensional, fluid flow of uniform density occupying a region V external to an immersed body with bounding surface S. The fluid velocity can be written using the Helmholtz decomposition as the sum of an irrotational part $\mathbf{u}_I$ and a rotational part $\mathbf{u}_R$, where $\mathbf{u}_R$ is induced by the vorticity field $\vec{\omega}$ according to the Biot–Savart integral

$$\vec{\mathbf{u}}_R(\vec{\mathbf{x}}, t) = -\frac{1}{4\pi} \int_V \frac{\vec{\mathbf{r}} \times \vec{\omega}(\vec{\mathbf{x}}', t)}{r^3}\, dv' \tag{1}$$

and $r \equiv |\vec{\mathbf{r}}| \equiv |\mathbf{x} - \mathbf{x}'|$. The part $\mathbf{u}_I$ includes any irrotational flow set by the boundary conditions at infinity and any flow induced by a dilatation field (such as a source sheet on S).

When using a Lagrangian vorticity method for a viscous flow, it is convenient to introduce an additional *diffusion velocity* $\vec{\mathbf{v}}$, which is used to advect the computational points to maintain coverage of the diffusing vorticity support. The time derivative of any quantity f evaluated at a point $\vec{\mathbf{x}}_n(t)$ that is advected by the sum of the local fluid velocity and the diffusion velocity, according to

$$\frac{d\vec{\mathbf{x}}_n}{dt} = \vec{\mathbf{u}}(\vec{\mathbf{x}}_n, t) + \vec{\mathbf{v}}(\vec{\mathbf{x}}_n, t), \tag{2}$$

is given by

$$\frac{d_v f}{dt} \equiv \frac{\partial f}{\partial t} + (\vec{\mathbf{u}} + \vec{\mathbf{v}}) \cdot \vec{\nabla} f = \frac{df}{dt} + \vec{\mathbf{v}} \cdot \vec{\nabla} f. \tag{3}$$

Here d/dt is the time derivative following a point advected with only the fluid velocity $\vec{\mathbf{u}}$ (the usual material derivative). An expression for diffusion velocity is given by Ogami and

Akamatsu [30] for two-dimensional flows as

$$\vec{\mathbf{v}} = -\nu\vec{\nabla}(\ln\omega), \tag{4}$$

where ω is vorticity magnitude and ν is the kinematic viscosity. The diffusion velocity can be chosen as any vector field that allows the computational points to maintain continuous coverage of the vorticity support as it diffuses with time. This requirement is guaranteed if the diffusion velocity satisfies the property that the circulation is invariant about any circuit C that is material with respect to the velocity field $\vec{\mathbf{u}}+\vec{\mathbf{v}}$. The expression (4) satisfies this property in two-dimensional flows but not in three-dimensional flows [19], due to the effect of curvature of the vortex lines. No expression for diffusion velocity has yet been found having this property for three-dimensional flows. However, most three-dimensional diffusion processes are either approximately one- or two-dimensional (such as diffusing vorticity sheets or tubes) or involve diffusion between two different vorticity-containing regions in which computational points already exist (such as vortex reconnection problems). For this reason, we have found the expression (4) to perform well in maintaining coverage of vorticity support in a wide variety of two- and three-dimensional flows.

The standard vorticity transport equation is

$$\frac{d\vec{\omega}}{dt} = (\vec{\omega}\cdot\vec{\nabla})\vec{\mathbf{u}} + \nu\nabla^2\vec{\omega}. \tag{5}$$

For computational points that are advected according to (2), it is more convenient to write (5) in terms of the derivative d_v/dt by adding $(\vec{\mathbf{v}}\cdot\vec{\nabla})\vec{\omega}$ to each side of (5), giving

$$\frac{d_v\vec{\omega}}{dt} = (\vec{\omega}\cdot\vec{\nabla})\vec{\mathbf{u}} + (\vec{\mathbf{v}}\cdot\vec{\nabla})\vec{\omega} + \nu\nabla^2\vec{\omega}. \tag{6}$$

It is convenient to rearrange the viscous diffusion terms to write (6) as

$$\frac{d_v\vec{\omega}}{dt} = (\vec{\omega}\cdot\vec{\nabla})\vec{\mathbf{u}} + \vec{\mathbf{D}}, \tag{7}$$

where the viscous term $\vec{\mathbf{D}}$ can be written using (4) in terms of the vorticity magnitude ω and a unit direction $\vec{\mathbf{a}}$ tangent to the vorticity vector as

$$\vec{\mathbf{D}} = \nu\omega\nabla^2(\ln\omega)\vec{\mathbf{a}} - \omega(\vec{\mathbf{v}}\cdot\vec{\nabla})\vec{\mathbf{a}} + \nu\omega\nabla^2\vec{\mathbf{a}}. \tag{8}$$

The force $\vec{\mathbf{F}}$ on an immersed body with surface S and outward unit normal $\vec{\mathbf{n}}$ is given by the sum of the pressure and viscous shear forces as

$$\vec{\mathbf{F}} = -\int_S (p\vec{\mathbf{n}} + \mu\vec{\mathbf{n}}\times\vec{\omega})\,da. \tag{9}$$

Alternatively, the force on the body can be computed using the relationship

$$\frac{d\vec{\mathbf{P}}}{dt} = -\vec{\mathbf{F}}, \tag{10}$$

where the force impulse $\vec{\mathbf{P}}$ is defined by $\vec{\mathbf{P}} \equiv \frac{1}{2}\rho\int_V \vec{\mathbf{x}}\times\vec{\omega}\,dv$.

A Poisson equation for body surface pressure can be obtained by taking the divergence of the Navier–Stokes equation as

$$\nabla^2 B = \vec{\nabla} \cdot (\vec{\mathbf{u}} \times \vec{\omega}), \tag{11}$$

where $B \equiv (p - p_\infty)/\rho + (\kappa - \kappa_\infty)$, $\kappa \equiv \vec{\mathbf{u}} \cdot \vec{\mathbf{u}}/2$ is the kinetic energy per unit mass, and p_∞ and κ_∞ are constants. Assuming that $B \to 0$ as $r \to \infty$ (for external flow with uniform velocity at infinity), the Green's function solution of (11) is

$$\frac{1}{2} B(\vec{\mathbf{x}}) + \int_S B \frac{\partial G}{\partial n}\, da = \int_S G \frac{\partial B}{\partial n}\, da + \int_V G \nabla \cdot (\vec{\mathbf{u}} \times \vec{\omega})\, dv, \tag{12}$$

where $G = 1/4\pi |\vec{\mathbf{x}} - \vec{\mathbf{x}}'|$. Taking the inner product of the Navier–Stokes equation with $\vec{\mathbf{n}}$ yields an expression for $\partial B/\partial n$ as

$$\frac{\partial B}{\partial n} = -\vec{\mathbf{n}} \cdot \left[\nu \vec{\nabla} \times \vec{\omega} + \frac{\partial \vec{\mathbf{u}}}{\partial t} - \vec{\mathbf{u}} \times \vec{\omega} \right]. \tag{13}$$

Substituting (13) into (12) and using the divergence theorem yields a boundary-integral equation for B as

$$2\pi B(\vec{\mathbf{x}}) - \int_S B \frac{\vec{\mathbf{n}} \cdot \vec{\mathbf{r}}}{r^3} da' = -\int_S \left[\nu \frac{\vec{\mathbf{n}} \cdot (\vec{\mathbf{r}} \times \vec{\omega})}{r^3} + \frac{1}{r} \vec{\mathbf{n}} \cdot \frac{\partial \vec{\mathbf{u}}}{\partial t} \right] da + \int_V \frac{\vec{\mathbf{r}} \cdot (\vec{\mathbf{u}} \times \vec{\omega})}{r^3}\, dv. \tag{14}$$

Equation (14) is a Fredholm equation of the second kind for B, where $\vec{\mathbf{r}} = \vec{\mathbf{x}} - \vec{\mathbf{x}}'$ and the variables $\vec{\mathbf{n}}$, $\vec{\mathbf{u}}$, and $\vec{\omega}$ occurring in the integrands are functions of the primed variable. The derivation of (14) presented above is a special case of the integral equation version of the full Navier–Stokes equations derived by Uhlman [42].

The governing equations of the velocity and vorticity fields and the boundary integral equation for body surface pressure involve both integration and differentiation in different places. For instance, volumetric integration is necessary to compute the velocity field from (1) and to compute a source term in the boundary-integral equation (14) for body surface pressure. In both cases, the integral has the convolution form

$$\vec{\mathbf{g}}(\mathbf{x}) = \int_V f(\mathbf{x}') \vec{\mathbf{K}}(\mathbf{x} - \mathbf{x}')\, dv', \tag{15}$$

where the kernel is given by $\vec{\mathbf{K}}(\vec{\mathbf{r}}) = \vec{\mathbf{r}}/r^3$. Differentiation is necessary to compute the derivatives on the right-hand side of the vorticity transport equation (7) and to compute the diffusion velocity from the expression (4).

Numerical solution of the flow on a set of Lagrangian points, advected according to (2), requires robust, efficient algorithms for integration and differentiation on irregularly positioned computational points. A new method for efficient approximation of volume integrals of the form (15) using a tetrahedral mesh is presented in Section 3. The tetrahedral mesh is also used to identify nearby points to a given computational point, which is employed in the moving least-squares numerical differentiation procedure described in Section 4.

3. INTEGRATION OVER TETRAHEDRAL ELEMENTS

A tetrahedral mesh is fit to the set of Lagrangian computational points at each time step using a two-step procedure. Given a set of nodal points on the body surface S and the outward unit normal of S at each of these points, we first form triangular panels on S using the Delauney triangularization procedure. The surface panels are then used as the start of an advancing front of tetrahedrons that extend out into the flow, starting with a single tetrahedron attached to each surface panel and working outwards to include all volume points. The point-insertion algorithm described by Borouchaki and Lo [4] is implemented in order to speed up formation of the volume mesh. A fast point search method is utilized, in which the computational points are initially sorted into a set of Cartesian boxes that enclose the vorticity support.

At the end of each time step, a computational point is added at the centroid of any existing tetrahedral element whose maximum side length exceeds a prescribed value l_1. This procedure is used to maintain a desired resolution of the flow field. As a new mesh is formed at the beginning of each time step, any tetrahedral elements with maximum side length greater than a second prescribed value l_2 are eliminated (where $l_2 > l_1$). This procedure is used to avoid connecting the tetrahedral mesh across disjoint vorticity regions or distant parts of a given vorticity region. For instance, when fitting a mesh to a vortex ring, it is necessary to use this procedure to eliminate tetrahera that span the center region of the ring. The choice of l_1 and l_2 depends on the length scales of the flow geometry and on the desired computational resolution. At present, these numbers are prescribed for a given computation, but some adaptive method of specifying these parameters is a desirable future development.

3.1. Direct integration procedure. The direct integration procedure computes the contribution to the integral (15) from each of the M tetrahedra of the mesh, so that by summing over these tetrahedra (15) becomes

$$\vec{\mathbf{g}}(\mathbf{x}) = \sum_{m=1}^{M} \int_{V_m} \frac{f(\mathbf{x}')\vec{\mathbf{r}}}{r^3}\, dv', \tag{16}$$

where V_m is the volume occupied by the mth tetrahedron. If we now assume that $f(\vec{\mathbf{x}})$ has a constant value f_m over each tetrahedron, the integral in (16) can be simplified using $\vec{\nabla}'(1/r) = \vec{\mathbf{r}}/r^3$ and the divergence theorem to write

$$\vec{\mathbf{g}}(\mathbf{x}) \cong \sum_{m=1}^{M} f_m \int_{S_m} \frac{\vec{\mathbf{n}}'}{r}\, da', \tag{17}$$

where S_m denotes the bounding surface of the mth tetrahedron and $\vec{\mathbf{n}}$ is the outward unit normal of S_m. The surface integral in (17) is the same as that which arises in computing the potential due to a source distribution of uniform strength and can be evaluated analytically [29] for any piecewise planar surface S_m.

The "analytical" method (17) for evaluating the convolution integral is very inefficient, since it requires evaluation of 12 logarithms and 24 arc tangents for each of the M tetrahedra. Typically M is about six times the number of computational points in the flow. In order to speed up the velocity calculation, we evaluate (16) using numerical integration for

tetrahedra that are sufficiently far away from the point at which the velocity is desired. The numerical integration procedure uses a three-dimensional Gaussian quadrature approach for a generalized tetrahedral domain [44], where the integral over each tetrahedron in (16) is approximated by the sum

$$\int_{V_m} \frac{f(\mathbf{x}')\vec{\mathbf{r}}}{r^3}\,dv' \cong \sum_{l=1}^{G} f(\vec{\xi}_l)\frac{\vec{\mathbf{r}}_l}{r_l^3} W_l V_m. \tag{18}$$

The summation formula (18) employs G Gauss points within the tetrahedron, located at positions $\vec{\xi}_l$ and having weights W_l. The volume of the mth tetrahedron is denoted by V_m, and $\vec{\mathbf{r}}_l = \vec{\mathbf{x}} - \vec{\xi}_l$ is the position vector of the field point $\vec{\mathbf{x}}$ relative to the position of the lth Gauss point. If the positions of the four nodes of the tetrahedron are denoted by $\vec{\eta}_\alpha$, $\alpha = 1, \ldots, 4$, relative to a global coordinate system, then the positions of the Gauss points can be written as

$$\vec{\xi}_l = \sum_{\alpha=1}^{4} L_{\alpha l}\vec{\eta}_\alpha, \tag{19}$$

where $L_{\alpha l}$ are the coordinates of the lth Gauss point with respect to the "tetrahedron" coordinate system [44]. For instance, L_{1l} is the ratio of the volume of the tetrahedron formed by the point $\vec{\xi}_l$ and the nodal points $\vec{\eta}_2, \vec{\eta}_3, \vec{\eta}_4$ to the total volume of the original tetrahedron.

In the current calculations, either a linear (one-point) or a cubic (five-point) numerical approximation of (16) is used, depending on the distance between the tetrahedron and the field point $\vec{\mathbf{x}}$. The weight and abscissa location for the linear approximation are given by $W_1 = 1$ and $L_{\alpha 1} = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, which is equivalent to the standard trapezoidal rule. The weights and abscissa locations for the cubic approximation are given in Table I.

The integration method (linear, cubic, or analytic) is selected for each tetrahedron and for each evaluation of the integral by examining whether the error for the Gaussian quadrature schemes (given by the estimate described in Subsection 3.3) is above a prescribed value. When using this procedure to determine velocity or pressure at a computational point, the analytic method is typically used only for the 20–40 closest tetrahedra, the cubic method is used for several hundred tetrahedra surrounding the field point, and the linear method may be used for several thousand tetrahedra.

TABLE I
Gauss Weights W_l and Abscissa Locations L_{pl} for a Cubic Approximation over a Tetrahedron

		$L_{\alpha l}$			
l	W_l	$\alpha = 1$	2	3	4
1	−4/5	1/4	1/4	1/4	1/4
2	9/20	1/2	1/6	1/6	1/6
3	9/20	1/6	1/2	1/6	1/6
4	9/20	1/6	1/6	1/2	1/6
5	9/20	1/6	1/6	1/6	1/2

3.2. Indirect integration procedure. The direct integration procedure requires $O(MN)$ computations per time step to compute the velocity field induced by the M tetrahedra at the N computational points. For large values of M and N, the time required for direct integration becomes excessive. The integration can be accelerated by use of a multipole expansion procedure for sufficiently distant tetrahedra, for which the number of computations per time step reduces to approximately $O(N \ln M)$. The multipole expansion procedure first groups the computational points into a series of boxes, where the boxes are arranged in a tree structure that adapts to the point distribution. The box tree is initiated by placing all computational points in a single box. New generations of boxes are formed by a Clarke–Tutty type box division process [6] that uses the following two steps:

(1) the coordinate direction (x, y, or z) is identified along which the maximum separation distance between any two computational points in the box is greatest;

(2) the box is divided into two offspring boxes at the median computational point in the identified coordinate direction.

This division process maintains approximately the same number of computational points in each box for a given box generation. The division process is continued until the number of points in the smallest-size box is less than a prescribed value.

"Interaction lists" are formed for each of the smallest-size boxes within the tree structure, indicating other boxes with which the box interacts directly and indirectly, such that each computational point lies in exactly one box listed on either the direct or indirect interaction lists of each smallest-size box. These lists are generated by determining whether the distance between the centers of the smallest-size "target" box A and the "source" box B is less than a critical value, where the critical distance is determined for each source box based on a prescribed maximum error using the maximum allowed multipole expansion order. Only expansions through second order are considered, since in three dimensions higher-order expansions become increasingly inefficient.

The contribution to the function g in (16) at a field point $\vec{\mathbf{x}}$ due to the lth member of the L boxes contained on the indirect interaction list of the smallest-size box containing the point $\vec{\mathbf{x}}$ is given by the multipole expansion

$$\vec{\mathbf{g}}_l(\vec{\mathbf{x}}) = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}\sum_{k=0}^{\infty} \frac{(-1)^{m+n+k}}{m!n!k!} I_{l,mnk} \frac{\partial^{m+n+k}}{\partial x^m \partial y^n \partial z^k}\left(\frac{\vec{\mathbf{r}}_l}{r_l^3}\right). \tag{20}$$

The term $I_{l,mnk}$ in (20) is the moment of box l about the box centroid $\vec{\mathbf{c}}_l$, and $\vec{\mathbf{r}}_l \equiv \vec{\mathbf{x}} - \vec{\mathbf{c}}_l$ is the position of the field point relative to the box centroid. The speed-up is possible because the box moments are independent of the field point (and thus need to be computed only once for each time step) and the derivatives in (20) depend only on the box centroid positions (and not on the positions of the computational points within the boxes).

The moment $I_{l,mnk}$ of box l is defined by

$$I_{l,mnk} = \int_V f(\vec{\mathbf{x}})(x - c_{l1})^m (y - c_{l2})^n (z - c_{l3})^k \, dv. \tag{21}$$

Each tetrahedral element is associated with the box structure of one of its nodes, even though the other nodes of the tetrahedron may not lie within this box. Denoting the number of tetrahedra associated with box l by M_l, the integral in (21) can be written as a sum over

these tetrahedra as

$$I_{l,mnk} = \sum_{p=1}^{M_l} \int_{V_p} f(\vec{\mathbf{x}})(x - c_{l1})^m (y - c_{l2})^n (z - c_{l3})^k \, dv. \tag{22}$$

The integration over each tetrahedron in (22) is performed analytically using a linear interpolation for the function $f(\vec{\mathbf{x}})$ over each tetrahedron of the form

$$f(\vec{\mathbf{x}}) = \sum_{\alpha=1}^{4} f(\vec{\mathbf{x}}_\alpha) L_\alpha, \tag{23}$$

where L_α are the local tetrahedron coordinates of the point $\vec{\mathbf{x}}$. The moment arms can similarly be expanded in terms of the tetrahedron coordinates, for instance

$$x - c_{l1} = \sum_{\alpha=1}^{4} (x_\alpha - c_{l1}) L_\alpha, \tag{24}$$

and the moment integrals in (22), up through the quadrapole terms, can be evaluated using the result [44]

$$\int_{V_p} L_1^a L_2^b L_3^c L_4^d \, dv = 6V_p \frac{a!b!c!d!}{(3 + a + b + c + d)!}. \tag{25}$$

While the moments of the smallest-size boxes are computed by direct integration of (22), the moments of boxes in previous generations of the tree structure can be more efficiently computed by summing the contributions of the offspring boxes, using the binomial formula to expand the moments of the difference terms in (22). For instance, the moment of the difference term in the x-direction can be expanded about the center $\vec{\mathbf{c}}_i$ of the parent box i to box l as

$$(x - c_{i1})^m = [(x - c_{l1}) + (c_{l1} - c_{i1})]^m = \sum_{r=0}^{m} \binom{m}{r} (c_{l1} - c_{i1})^r (x - c_{l1})^{m-r}. \tag{26}$$

The contribution of offspring box l to the moment of parent box i is then given by

$$\sum_{r=0}^{m} \sum_{s=0}^{n} \sum_{t=0}^{k} \binom{m}{r} \binom{n}{s} \binom{k}{t} (c_{l1} - c_{i1})^r (c_{l2} - c_{i2})^s (c_{l3} - c_{i3})^t I_{l,(m-r)(n-s)(k-t)}. \tag{27}$$

3.3. Error estimates. The multipole expansion is truncated to include only those values of the indices such that $m + n + k \le H$, where H is called the "order" of the expansion. A theoretical upper bound for the absolute error of the multipole expansion of order H is given by Salmon and Warren [34] as

$$E_H \le \frac{1}{(d - b)^2} \left[(H + 2) \frac{B_{H+1}}{d^{H+1}} - (H + 1) \frac{B_{H+2}}{d^{H+2}} \right], \tag{28}$$

where b is an estimate of the radius of the vorticity support (i.e., a length scale associated with the box or tetrahedron to which the expansion is applied), d is the distance from the

point about which the expansion is applied (i.e., the box or tetrahedron centroid) to the point $\vec{\mathbf{x}}$ at which the integral is evaluated, and the moment magnitudes B_k are defined for a tetrahedron p by

$$B_k \equiv \int_{V_p} \|f(\vec{\mathbf{x}})\| \|\vec{\mathbf{x}} - \vec{\mathbf{c}}_l\|^k \, dv. \tag{29}$$

A slightly less tight (but simplier to implement) error bound is derived from (28) by Winckelmans *et al.* [43] as

$$E_H \leq \frac{B_0}{(d-b)^2}\left[(H+2)\left(\frac{B_2}{B_0 b^2}\right)^{(H+1)/2}\left(\frac{b}{d}\right)^{H+1} - (H+1)\left(\frac{B_2}{B_0 b^2}\right)^{H}\left(\frac{b}{d}\right)^{H+2}\right]. \tag{30}$$

The error bound (30) has the advantage that it depends only on the zeroth and second-order moments B_0 and B_2, which are simple to compute. The Newton–Raphson iteration method can be used to solve for the critical distances d from (30) with a prescribed value of absolute error. Alternatively, omitting the last term in (30) yields an explicit solution for the critical distance that typically differs from the iterative solution by only 10–20%.

The multipole expansion error estimate (30) is solved to obtain the critical distances for indirect integration with $H \leq 2$, where B_0 and B_2 are obtained by summing the moments of the tetrahedra associated with each box. The box "size" b is set equal to the maximum distance between any two computational points in the box. The critical distances are used to determine the direct and indirect interaction lists for each box and to set the multipole expansion order for each indirect interaction.

Upper bounds for the error in the Gaussian quadrature approximation (18) used in the direct part of the integration are obtained by writing the kernel $\vec{\mathbf{r}}/r^3$ of the integral (16) as a polynomial using the multipole expansion (20) about the tetrahedron centroid. The one-point and five-point Gaussian quadrature algorithms are exact up through the first- and third-order terms of the polynomial expansion, respectively. The Gaussian quadrature error is bounded by the error E_1 and E_3 of the linear and cubic multipole expansions, respectively, as given by (30). A relative error E_{rel} is prescribed for the Gaussian quadratures, such that the absolute error is estimated by $E_{abs} \cong E_{rel}(B_0/4\pi d^2)$. Expressions for the critical distances d_1 and d_5 for the one-point and five-point Gaussian quadrature approximations are obtained from (30), after omitting the second term in brackets, as

$$(d_1 - b)^2 E_{rel} \geq C_1 (B_2/B_0), \qquad (d_5 - b)^2 \, d_5^2 E_{rel} \geq C_5 (B_2/B_0)^2. \tag{31}$$

The length scale b is set equal to the maximum distance between the tetrahedron centroid and the furthest of its nodes. The constants C_1 and C_5 are obtained from (30) as 12π and 20π, respectively. Empirical tests with these constants yield relative error values that are about an order of magnitude lower than the theoretical upper bounds.

3.4. Velocity calculation test. The numerical integration procedure described above is used to compute the velocity field induced by the fluid vorticity using the Biot–Savart equation (1). In this subsection, we report on a series of test computations for Hill's spherical vortex [2] that examine the speed-up and total error produced by the indirect velocity calculation method with different box error settings. Results for tetrahedral elements are also compared to results obtained using a vortex blob method [25].

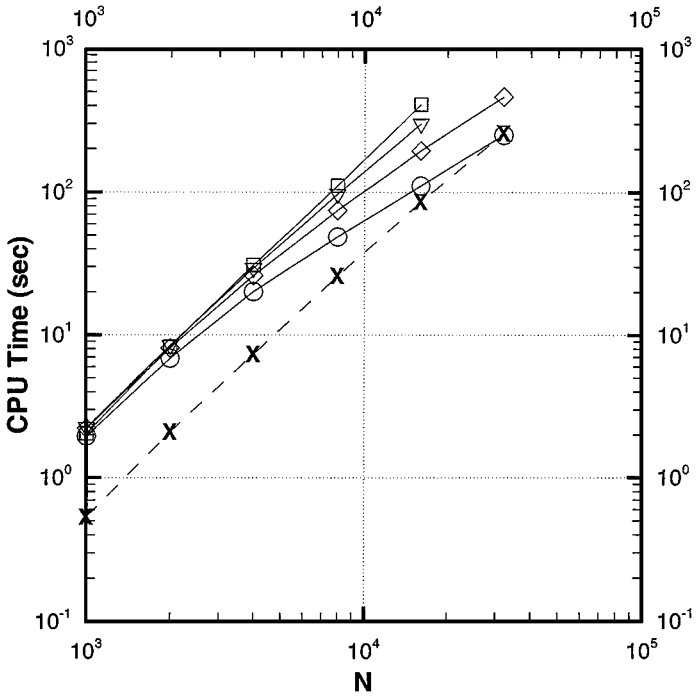

FIG. 1. Plot showing the CPU time required for velocity calculation using the tetrahdral elements (solid lines) with a fully direct calculation (squares) and for indirect calculation with absolute box error $\varepsilon = 10^{-n}$, with $n = 2$ (circles), $n = 3$ (diamonds), and $n = 4$ (gradients). Also shown is results of a calculation using vortex blobs with indirect calculation and $n = 3$ (crosses, dashed line).

The computational points are initially placed randomly within a sphere of unit radius, where the number of points ranges between 1000 and 32,000. Computations are performed with an absolute box error $\varepsilon = 10^{-n}$, with $n = 2, 3$, and 4, which is used in determining the critical box separation distance and the order of the multipole expansion in the indirect velocity calculation.

The CPU time for calculations on a Cray C-90 is plotted versus the number of computational points, N, in Fig. 1. The CPU time for the fully direct calculation increases with N at a rate slightly less than $O(N^2)$ (approximately proportional to $N^{1.88}$), reflecting the fact that a larger fraction of the tetrahedra are computed with 1-point Gaussian quadratures than with 5-point Gaussian quadratures as N increases. When N becomes large enough that nearly all the computational time is used for the 1-point quadratures, the CPU time in the direct calculation increases in proportion to N^2. For cases using the multipole acceleration method for distant points, the number of points performed with the direct calculation method is approximately constant as N varies, where the value of this constant increases as the specified box error ε decreases. For sufficiently small N, most of the velocity calculation is performed directly and there is little difference between cases using the accelerated method and the fully direct calculation. For relatively high values of ε (such as for the case with $n = 2$), the CPU time increases nearly in proportion to $N \ln N$ for the largest values of N examined. For smaller values of ε (such as for the case with $n = 4$), the values of N considered in these tests are not high enough to reach the $N \ln N$ asymptotic dependence. Calculations with vortex blobs are faster than those with tetrahedral elements by a factor of approximately 4 for $N = 1000$. However, as N increases, the velocity calculation is

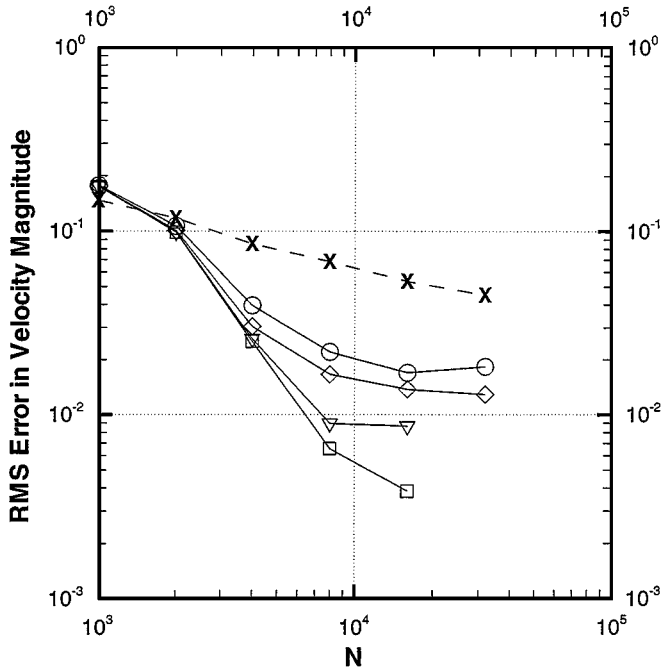

FIG. 2. Plot showing the root-mean square error for a Hill's spherical vortex, for the same cases listed in the caption for Fig. 1.

primarily performed by indirect (box-point) interactions, so that by $N = 32{,}000$ the CPU time is nearly identical for the tetrahedra with $n = 2$ and for the vorticity blobs.

The root-mean square (rms) error in velocity magnitude is plotted as a function of N in Fig. 2 for these same five cases. For the fully direct calculation with tetrahedral elements, the error decreases rapidly with increase in N, at a rate approximately proportional to $N^{-1.5}$. The rms error for computations using the accelerated method with tetrahedral elements is also observed to initially decrease rapidly with increase in N, but then to eventually asymptote to an approximately constant value at large N. This asymptotic value of the error is controlled not by the tetrahedral element size, but rather by the error incurred by approximation of the velocity induced by the tetrahedrals within a box with the multipole expansion. As expected, the asymptotic value of the rms velocity error increases with increase in the specified absolute box error ε.

The rms error for computations with vorticity blobs decreases at a rate approximately proportional to $N^{-0.4}$. The error with vortex blobs elements is much larger than for any of the cases considered with tetrahedral elements. For instance, the rms error for a computation with vorticity blobs with $N = 32{,}000$ is found to be about the same as for a calculation with tetrahedral elements with $N = 3000$ and $n = 2$. The CPU time in these two computations, however, is less for the tetrahedral elements than for the vorticity blobs by a factor of nearly 18. Much of the error observed in the calculation with vorticity blobs arises from the fact that blobs associated with points near the outer spherical boundary extend outside the nominal radius of the vortex. These results demonstrate the utility of tetrahedral elements for fitting the vorticity field in situations where the vorticity has an abrupt discontinuity. The vorticity at the surface of a solid body immersed in a fluid stream is an especially important example of such a flow.

4. NUMERICAL DIFFERENTIATION ON LAGRANGIAN COMPUTATIONAL POINTS

The numerical procedure used to approximate derivatives is required to remain accurate even when the control points are very irregularly spaced, as would typically be the case in Lagrangian calculations. A differentiation method exhibiting this property, called the "moving least-squares method," was recently described by Marshall and Grant [26] in the context of axisymmetric flows, and this method is summarized in the current section for arbitrary three-dimensional flows.

4.1. Differentiation procedure. Let us suppose that the derivative of a function $f(\vec{\mathbf{x}}, t)$ is desired at a computational point m located at $\vec{\mathbf{x}}_m$. The value of $f(\vec{\mathbf{x}}, t)$ is known only on the set of irregularly spaced points $\vec{\mathbf{x}}_n$, $n = 1, \ldots, N$. In the moving least-squares method, the values f_n of $f(\vec{\mathbf{x}}, t)$ on these computational points are interpolated locally by a quadratic polynomial in the components of the position difference $\vec{\mathbf{x}} - \vec{\mathbf{x}}_m$, or

$$q_m(\vec{\mathbf{x}}, t) = f_m + \sum_{i=1}^{9} F_{m,i} W_{m,i}(\vec{\mathbf{x}} - \vec{\mathbf{x}}_m). \tag{32}$$

In (32), the index m denotes the computational point about which the interpolation is performed, $F_{m,i}$ denotes a set of nine undetermined coefficients of the polynomial, and $W_{m,i}(\vec{\mathbf{x}} - \vec{\mathbf{x}}_m)$ are the associated weight functions, defined by

$$\begin{aligned}
&W_{m,1} = \frac{x - x_m}{R_m}, \qquad W_{m,2} = \frac{y - y_m}{R_m}, \qquad W_{m,3} = \frac{z - z_m}{R_m}, \\
&W_{m,4} = \left(\frac{x - x_m}{R_m}\right)\left(\frac{y - y_m}{R_m}\right), \qquad W_{m,5} = \left(\frac{x - x_m}{R_m}\right)\left(\frac{z - z_m}{R_m}\right), \\
&W_{m,6} = \left(\frac{y - y_m}{R_m}\right)\left(\frac{z - z_m}{R_m}\right), \\
&W_{m,7} = \left(\frac{x - x_m}{R_m}\right)^2, \qquad W_{m,8} = \left(\frac{y - y_m}{R_m}\right)^2, \qquad W_{m,9} = \left(\frac{z - z_m}{R_m}\right)^2.
\end{aligned} \tag{33}$$

The parameter R_m is a length scale associated with the computational point m, called the point "radius." The value of R_m is set for each point m based on the volume $\bar{V}_m$ of the "module" of m, which consists of all tetrahedra attached to point m, according to

$$R_m = c(3\bar{V}_m/4\pi)^{1/3}, \tag{34}$$

where the "overlap" parameter c is typically set equal to 2.

The coefficients of the polynomial (32) are obtained by a localized least-square procedure, in which the function to be minimized, E_m, is expressed as

$$E_m \equiv \sum_{n=1}^{N} L_{mn}[f_n - q_m(\vec{\mathbf{x}}_n, t)]^2. \tag{35}$$

The "localization parameter" L_{mn} serves to weight the error so as to give most importance to points that are closest to the point $\vec{\mathbf{x}}_m$ at which the derivative is desired. In the present study, L_{mn} is set equal to unity for all points n that are nodes of tetrahedra in the module of m (the "first neighbors" of m) or are nodes of tetrahedra that are in the modules of the first neighbors

(the "second neighbors" of m). The set of first and second neighbors of a point is called the "local list" of that point. For all points n not on the local list of point m, the localization function L_{mn} is set to zero and the points are not included in the sum (35). An alternative choice for L_{mn} is to use a Gaussian function that decays away from the point $\vec{\mathbf{x}}_m$ [26].

Minimization of E_m with respect to each of the coefficients $F_{m,i}$ yields a 9×9 system of linear equations, which has the form

$$\sum_{j=1}^{9} D_{m,ij} F_{m,j} = U_{m,i}, \qquad i = 1, \ldots, 9, \tag{36}$$

where

$$D_{m,ij} = \sum_{n=1}^{N} L_{nm} W_{nm,i} W_{nm,j}, \qquad U_{m,i} = \sum_{n=1}^{N} (f_n - f_m) L_{nm} W_{nm,i} \tag{37}$$

and $W_{nm,i} \equiv W_{m,i}(\vec{\mathbf{x}}_n - \vec{\mathbf{x}}_m)$. Solution of the system (36) yields the coefficients $F_{m,i}$ as

$$F_{m,i} = \sum_{j=1}^{9} D^{-1}_{m,ij} U_{m,j}. \tag{38}$$

The derivatives of f_m can be computed simply by differentiating the polynomial fit (32). It is noted that when computational points become very isolated, due to inadequate spatial resolution, the condition number of the matrix $D_{m,ij}$ in (36) becomes very large. In such cases, the best option is to either eliminate the point or to add additional points so as to improve the matrix conditioning.

A test of the moving least-squares differentiation procedure was performed for a columnar vortex with axial vorticity variation of Gaussian form $\omega_z = A \exp(-r^2/\sigma_0^2)$. The computational points are arranged in each cross-section in a series of concentric rings, with a radial spacing of Δr_0 and azimuthal spacing also approximately equal to Δr_0. Figure 3 shows

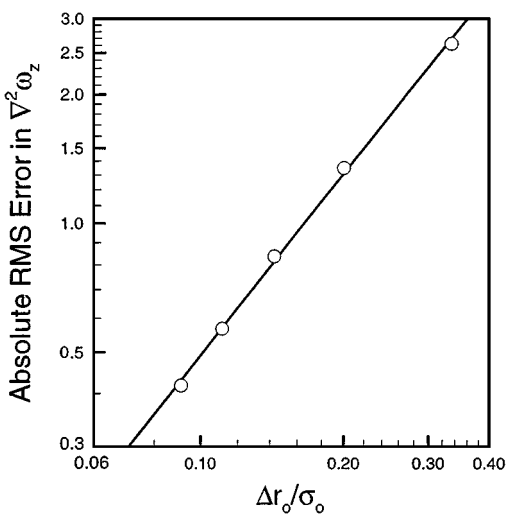

FIG. 3. Root-mean square error in $\nabla^2\omega_z$ for a columnar vortex with Gaussian vorticity profile for moving least-square method. Best-fit line has slope of 1.42.

the absolute error in the Laplacian of ω_z as a function of $\Delta r_0/\sigma_0$ on a log–log scale. The slope of the line fit in Fig. 3 is approximately 1.42, which indicates that when applied to irregularly spaced points, the method is between first- and second-order accurate.

4.2. Time stepping and point amalgamation. The moving least-squares differentiation procedure reduces to the second-order centered finite-difference procedure for uniformly spaced points when only the first neighbors are used. The computation of diffusion with the explicit moving least-squares method is subject to a stability limitation on the time step, similar to that for the explicit heat equation with centered finite-difference [31], of the form

$$\frac{2\nu\,\Delta t}{sl^2} < 1, \tag{39}$$

where l is a measure of the minimum distance between any two computational points and s is a parameter (called the "stability parameter") that is found empirically to be about 7–9.

For some fluid flows, such as high Reynolds number boundary layers, the restriction (39) often requires the time step to be much smaller than it would otherwise need to be based on the other (inviscid) terms in the vorticity and point advection equations. Since calculation of the viscous diffusion term in (7) requires far less CPU time than the velocity calculation, the computation can be considerably speeded up by use of a diffusion substep that is embedded within the larger inviscid time steps. In the current paper, we use Adams–Bashforth second-order explicit time stepping for the inviscid terms in (2) and (7) and a second-order predictor-corrector algorithm for the viscous terms.

Amalgamation of the Lagrangian computational points is necessary in cases where two points come so close that the explicit diffusion scheme becomes unstable over the diffusion substeps. For given values of stability parameter s, minimum time step $\Delta t_{\min}$, and number of diffusion substeps N_D, the critical distance between two Lagrangian control points is

$$l_{crit}^2 = \frac{2\nu\,\Delta t_{\min}}{sN_D}. \tag{40}$$

When the distance between any two points is less than l_{crit}, the points are amalgamated by eliminating both points and creating a new point at the centroid of the two original points. The vorticity on this new point is set to be the average of the vorticity from the two points that are eliminated.

4.3. Vortex stretching term. Use of the moving least-squares differentiation method for approximation of the velocity gradient in the vorticity stretching term in the vorticity transport equation (7) provides considerable time savings compared to computing this term analytically. A validation computation was performed for the vortex stretching term for the problem of a thin vortex ring that is subject to stretching by a line source aligned along the ring central axis (Fig. 4). The vorticity is aligned entirely in the azimuthal direction in this problem, and in the absence of viscosity, the vorticity at a computational point changes in direct proportion to the radial position of the point. A plot of the time variation of the azimuthal vorticity component at a point initially in the center of the vortex core as the ring radius approximately doubles under the velocity field induced by the line source is given in Fig. 5. All length and time scales are non-dimensionalized by the vortex core radius σ_0 and the vortex rotation time σ_0^2/Γ, respectively. The data in Fig. 5 are obtained for an inviscid computation with 35 points within each cross-section of the core and 3920 total

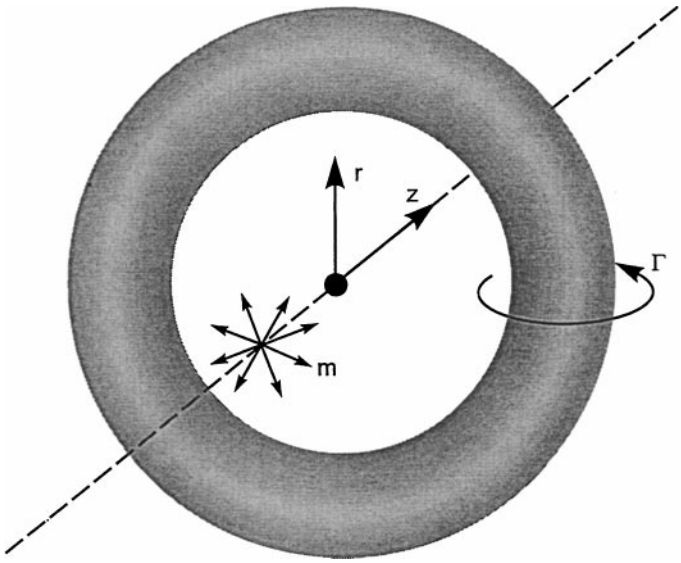

FIG. 4. Schematic diagram of a vortex ring with a center line source.

computational points. The time step is held fixed at $\Delta t = 0.01$, and the source strength is adjusted to maintain a vortex axial stretching rate of 0.05. The computational prediction in Fig. 5 is found to lie within 1.2% of the analytical solution.

4.4. Vorticity diffusion term. A series of validation computations for approximation of vorticity diffusion is reported for the case of a columnar vortex with Gaussian variation in both the axial vorticity and the axial velocity fields, the latter of which is generated by a non-zero azimuthal vorticity component. Computational points are initially placed in

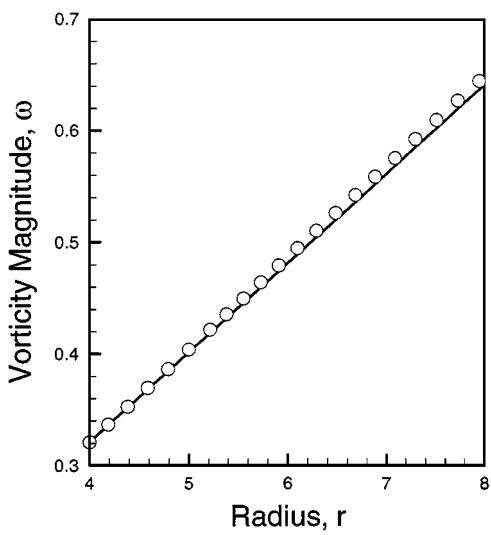

FIG. 5. Variation of vorticity magnitude with point radial position for a vortex ring that is stretched by a line source along its axis, comparing the analytical solution (solid line) with computational predictions (circles).

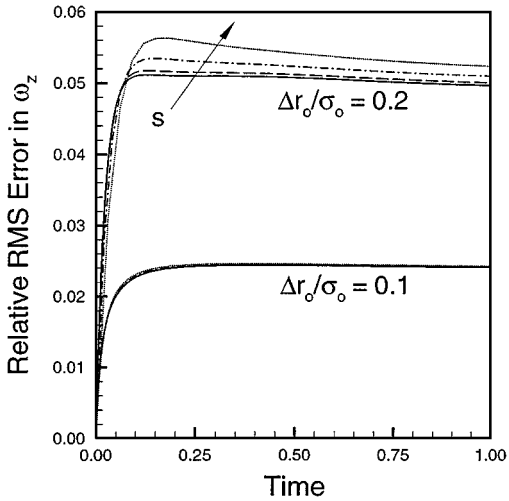

FIG. 6. Relative root-mean square error in axial vorticity for a diffusing columnar vortex with Gaussian axial vorticity profile. Curves are given for cases with radial point spacing of both $\Delta r_0/\sigma_0 = 0.1$ and 0.2, and for cases with stability parameter s of 0.1 (solid curve), 0.5 (dashed curve), 1.0 (dashed-dotted curve), and 1.5 (dotted curve).

concentric circles in each cross-section of the columnar vortex, where the rings are separated by a radial distance Δr_0. The relative root-mean square (rms) error in ω_z is plotted versus time in Fig. 6 for cases with Δr_0 of 0.1 and 0.2 and for values of the stability parameter s of 0.1, 0.5, 1.0, and 1.5. All length and time scales are non-dimensionalized by the vortex core radius σ_0 and the vortex rotation time σ_0^2/Γ, respectively. The results indicate that the relative error quickly asymptotes to a constant value which is not sensitive to the value of the stability parameter. The asymptotic value of the relative error for cases with $\Delta r_0 = 0.1$ is slightly less than half of that for cases with $\Delta r_0 = 0.2$. A series of plots are given in Figs. 7a–7c showing variation of axial and azimuthal vorticity profiles with time for cases with $\Delta r_0 = 0.2$ (circles) and $\Delta r_0 = 0.1$ (triangles) in comparison to the analytical solution (solid and dashed curves, respectively). During the time period considered, the maximum axial and azimuthal vorticity components decrease by factors of ten and forty, respectively.

5. VORTICITY BOUNDARY CONDITION

A change in the value of vorticity at the surface of an immersed rigid body affects the vorticity within the volume of the fluid both by changing the vorticity contained within tetrahedra that are attached to the surface and by changing the vorticity diffusion flux normal to the surface. The vorticity boundary condition is set so as to continually force the tangential velocity at the body surface to zero. The vorticity boundary condition employed in the present paper is related to that described by Koumoutsakos *et al.* [20], with the difference that it is implemented in the context of the least-squares differentiation method and it is modified to include the effect of direct vorticity transport to tetrahedra attached to the surface.

We recall that the body surface is divided into a set of flat triangular panels, where a "boundary" point is placed at each vertex of the panels. An area $\hat{A}_m$ is defined as one-third

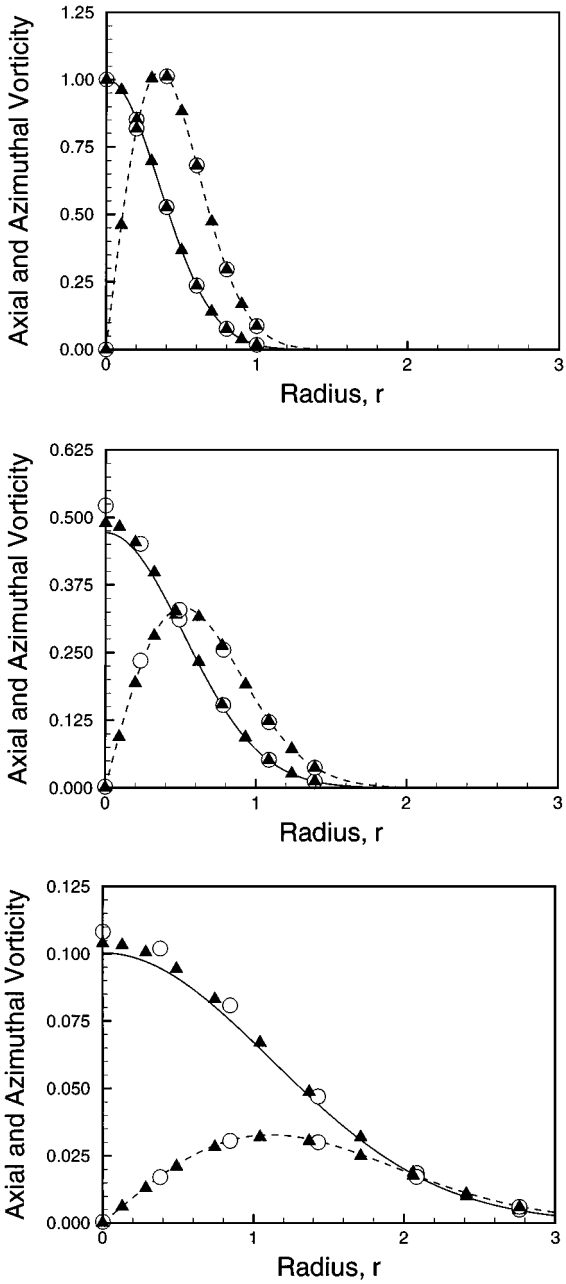

FIG. 7. Variation of axial and azimuthal vorticity within a diffusing columnar vortex at times $t = 0$, 0.07 and 0.52, showing the computational results for $\Delta r_0/\sigma_0 = 0.2$ (circles), $\Delta r_0/\sigma_0 = 0.1$ (triangles), and the exact solution (solid and dashed curves).

the area of all panels that are connected to the mth boundary point $\vec{\mathbf{x}}_m$, such that the sum of $\hat{A}_m$ for all boundary points is equal to the total body surface area. Similarly, a volume $\hat{V}_m$ is defined as one-quarter of the sum of the volume of all tetrahedra attached to a boundary point m. The vortex sheet strength $\vec{\gamma}_m$ associated with the slip velocity $\vec{\gamma}_m \times \vec{\mathbf{n}}$ at $\vec{\mathbf{x}}_m$ is evaluated, where $\vec{\mathbf{n}}$ is the outward unit normal of the body surface S at $\vec{\mathbf{x}}_m$. The

total vorticity associated with slip at $\vec{\mathbf{x}}_m$ is $\vec{\gamma}_m \hat{A}_m$. A change $\Delta\vec{\omega}_m$ in vorticity at $\vec{\mathbf{x}}_m$ causes a change in the total vorticity associated with tetrahedra in the flow of $\hat{V}_m \Delta\vec{\omega}_m$. Additionally, the total amount of vorticity to diffuse into the flow from the region of the body surface with area $\hat{A}_m$ surrounding point $\vec{\mathbf{x}}_m$ during the time interval $(t, t+\Delta t)$ is $-\nu \hat{A}_m \Delta t (\partial\vec{\omega}/\partial n)|_m$. Balancing the slip vorticity with the vorticity transport from these two mechanisms gives

$$\vec{\gamma}_m \hat{A}_m = \hat{V}_m \Delta\vec{\omega}_m - \nu \hat{A}_m \Delta t (\partial\vec{\omega}/\partial n)|_m. \tag{41}$$

The normal derivative of vorticity at the surface can be written in terms of the vorticity values at surrounding computational points as

$$\left.\frac{\partial\vec{\omega}}{\partial n}\right|_m = \sum_{p=1}^{N} J_{pm}(\vec{\omega}_p - \vec{\omega}_m), \tag{42}$$

where

$$J_{pm} \equiv \frac{1}{R_m} \sum_{j=1}^{9} \left(\sum_{i=1}^{3} n_i D^{-1}_{m,ij} \right) W_{pm,j}, \tag{43}$$

and the matrices $D_{m,ij}$ and $W_{pm,j}$ are defined in (37) and (33). Substituting (42) into (41), and denoting the time step by a superscript, gives

$$\vec{\gamma}^n_m \hat{A}_m = \hat{V}_m \left[\vec{\omega}^{n+1}_m - \vec{\omega}^n_m\right] - \nu \hat{A}_m \Delta t \sum_{p=1}^{N} J_{pm} \left(\vec{\omega}^{n+1}_p - \vec{\omega}^{n+1}_m\right). \tag{44}$$

Equation (44) is solved for the vorticity at the body surface using the fixed-point iteration procedure

$$\left[\hat{V}_m + \nu \hat{A}_m \Delta t \sum_{\substack{p=1 \\ p\neq m}}^{N} J_{pm} \right] \vec{\omega}^{n+1}_m(q+1) = \hat{V}_m \vec{\omega}^n_m + \vec{\gamma}^n_m \hat{A}_m + \nu \hat{A}_m \Delta t \sum_{\substack{p=1 \\ p\neq m}}^{N} J_{pm} \vec{\omega}^{n+1}_p(q), \tag{45}$$

where q is an iteration index. This iteration procedure converges with a relative error of less than 10^{-4} in only 3–4 iterations.

6. EULER LAYER AND POINT CREATION

In computing flow past an immersed body, it is necessary to maintain a sufficient number of points above each body panel to resolve the boundary layer. In particular, if the total number of points in the local list of a computational point m falls below about 20, the matrix $D_{m,ij}$ in (36) becomes ill-conditioned, making the differentiation procedure inaccurate. To ensure resolution of the boundary layer for all times, a thin "Euler layer" of fixed points is employed above each body panel. Typically, the points within the Euler layer are staggered, such that the computational points on the body coincide with the vertices of the body panels, the computational points in the next layer are placed a prescribed distance above the panel centroids, and subsequent layers are alternately placed above the panel vertices and centroids. Vorticity evolution for points within the Euler layer is performed using the

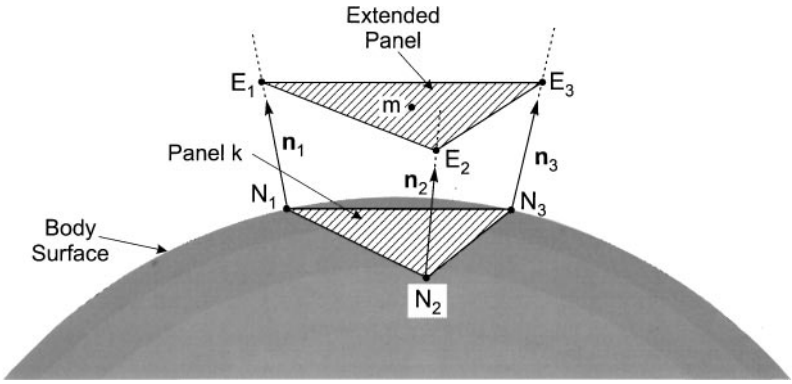

FIG. 8. Illustration of the construction of an *extended panel* passing through a point *m* oriented over a panel *k*.

Lagrangian form (7) of the vorticity transport equation by first advecting the points according to (2) and then interpolating the vorticity back onto the original point positions.

New Lagrangian points are created at the outer surface of the Euler layer above each panel. The first step in the point creation process is to determine the closest Lagrangian computational point that lies "above" each of the panels. This determination is performed using the concept of an "extended panel," illustrated in Fig. 8. The extended panel of a point *m* relative to a panel *k* is constructed first by assigning a unit normal to each vertex of *k* (using a weighted average of the unit normal of the attached panels), and then forming a triangle that passes through the point *m*, lies parallel to panel *k*, and has vertices on the lines coincident with the normal of the vertices of panel *k*.

The closest Lagrangian point above a panel *k* is determined by constructing the extended panel for each nearby Lagrangian point. Three triangles are then drawn on the plane of the extended triangle, each with one vertex at the point *m* and two vertices coinciding with the vertices of the extended panel. If the sum of the areas of these three triangles is equal to the area of the extended panel, then the point *m* is considered to be "above" the panel *k* (Fig. 9a). If this sum is greater than the area of the extended panel, then point *m* is considered to be above some other panel (Fig. 9b).

For each Lagrangian point that is found to be above the panel *k*, the normal distance of the point to the panel is computed. The Lagrangian point (not including points in the Euler layer) with the least normal distance to the panel is considered to be the "closest" point to panel *k*. A new Lagrangian computational point is created above panel *k* at any time step for which the normal distance of the closest point to the panel is greater than some prescribed value. The new point is located at the centroid of the tetrahedron formed by the closest point and the three points located above the nodes of panel *k* at the outer surface of the Euler layer, and the vorticity value at the new point is set to be the average of the vorticity at the vertices of this tetrahedron. If no Lagrangian point is found above a panel, then a new point is created above this panel at a prescribed distance from the outer surface of the Euler layer and the vorticity at this point is set by extrapolation from points within the Euler layer.

7. FLOW PAST A SPHERE

The Lagrangian vorticity method described in the preceding sections is utilized for computation of impulsive flow past a sphere at Reynolds number 100 for cases with two different ("medium" and "high") flow field resolutions. The flow variables are nondimensionalized

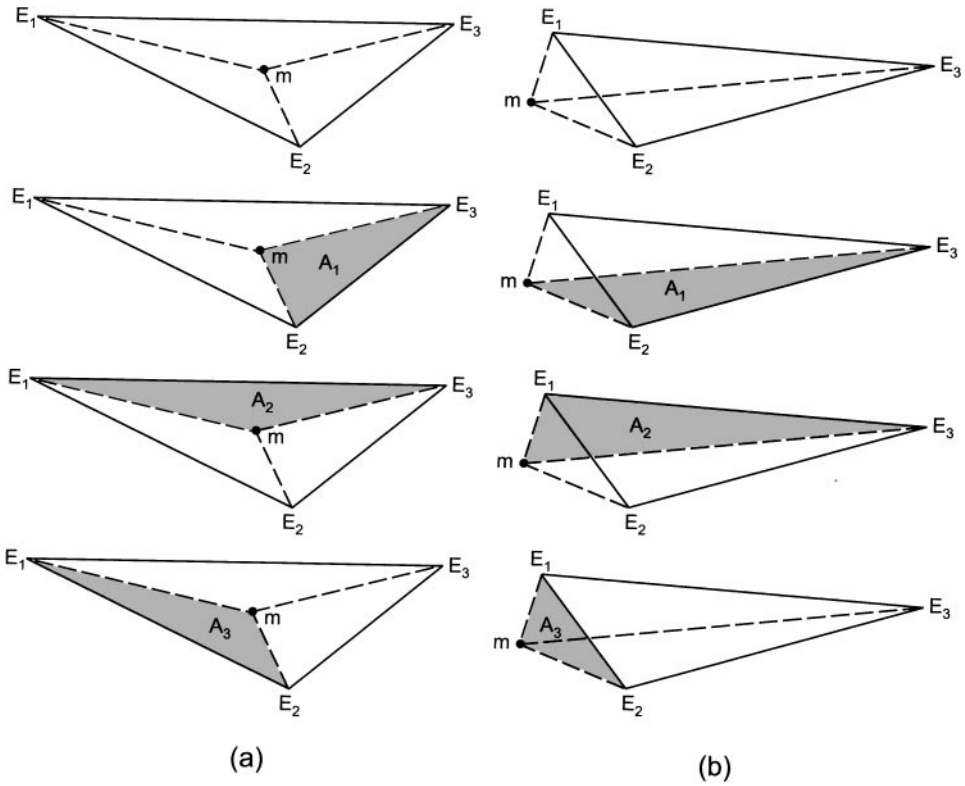

FIG. 9. Illustration of a method for determining whether a point m lies in a triangle with vertices E_1, E_2, E_3, by determining whether the sum of the areas A_1, A_2, A_3 of the three sub-triangles formed by two vertices and the point m is (a) equal to or (b) greater than the area of the original triangle.

using the sphere diameter and the free-stream velocity for length and velocity scales, and the free-stream velocity is oriented in the positive x-direction. The sphere surface is discretized using 1280 panels in the medium resolution case and 5210 panels in the high resolution case. The panels have the form of equilateral triangles of equal size (Fig. 10a). The computations are started by placing computational points in seven layers above the panels separated by a uniform radial distance of 0.01, with points placed above the body nodes and above the panel centers in alternate layers. The first five layers are composed of fixed "Euler layer" points and the remaining two layers are free "Lagrangian" points. The mesh generation parameters l_1 and l_2 are set equal to 0.2 and 0.3, respectively.

The vorticity field is initialized by first computing the potential flow past the body, with no vorticity assigned to the computational points, using a standard source panel method [13]. The vorticity associated with the surface slip is then distributed to the points in the boundary layer using a Gaussian distribution above each panel of the form $\vec{\omega} = (2\vec{\gamma}/\sqrt{\pi}\sigma)$ $\exp(-y_n^2/\sigma^2)$, where $\vec{\gamma}$ is the panel vorticity sheet strength, y_n is the normal distance from the body surface, and the length scale σ is chosen as 0.055. The surface slip velocity is recomputed with this vorticity distribution, and the vorticity associated with the slip is again distributed to the boundary layer points above each panel. This process is repeated until the maximum slip velocity is less than a specified value. The slip velocity remains small during the course of the computations, with typical root-mean square value of about 1% of the

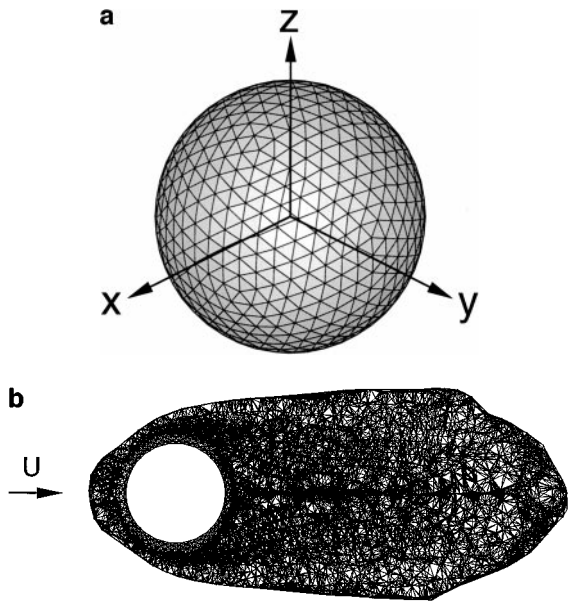

FIG. 10. Illustration showing (a) placement of source panels on the sphere surface and (b) cross-sectional view of the tetrahedral mesh in the x-y plane at $t = 3.5$.

free-stream velocity for both the medium and high resolution cases. The surface slip is associated with the vorticity that is diffused off the surface during the time step and is hence not expected to approach zero even in the limit of very high spatial resolution. Doubling the time step is observed to approximately double the surface slip. The medium and high resolution results generally remain similar throughout the computations, as shown in the comparison in Table II for time $t = 4.0$.

The computations proceed from this initial state with a fixed time step of $\Delta t = 0.01$. Fifteen viscous substeps are used for each of the larger time steps. New points are created outside of the Euler layer surrounding the sphere whenever the closest Lagrangian point exceeds a normal distance of 0.02 away from the outer edge of the Euler layer. A pair of Lagrangian points is amalgamated when their separation distance is less than 0.01. A cross-sectional slice of the tetrahedral mesh formed during a typical calculation is shown in Fig. 10b. The number of tetrahedral elements varies from about 35,000 to 600,000 for the medium resolution case and from about 130,000 to 750,000 for the high resolution case

TABLE II
Comparison of Results for High and Medium Resolution Computations of Flow Past a Sphere at Re = 100 and $t = 4.0$

Quantity	Medium resolution	High resolution
Maximum velocity	1.1361	1.1380
Linear impulse, P_x	2.8068	2.7551
Kinetic energy	4.6689	4.6020
Enstrophy	19.157	18.807
Helicity	1.1750	1.2269

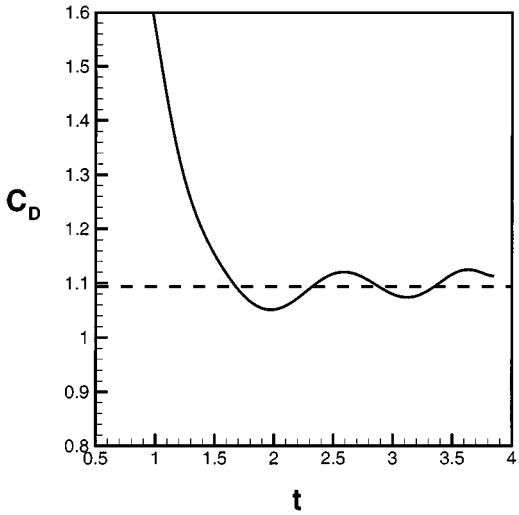

FIG. 11. Time variation of the drag coefficient impulsively started flow past a sphere at $\mathrm{Re} = 100$, showing computed results (symbols) and steady-state reference value (dashed line).

during the course of the computation. The number of tetrahedral elements can be modified by adjusting the parameters that control point amalgamation and point creation above the surface and within large-size tetrahedra.

The computed drag coefficient $C_d \equiv D/(0.5\rho U^2 \pi R^2)$ is plotted as a function of time in Fig. 11. The drag is computed from the expression (10), and the linear impulse is computed by analytically integrating over the tetrahedral elements using (25) with linear vorticity variation over each element. The drag coefficient results are compared to the steady-state value 1.09 from the Schiller–Naumann formula [35] $C_d = (24/\mathrm{Re})[1 + 0.15\,\mathrm{Re}^{0.687}]$, shown by a dashed line in Fig. 11. Similarly, the highly precise computations of Tabata and Itakura [39] give the steady-state drag coefficient as 1.0895 ± 0.0005. We do not attempt to compare to transient drag values for impulsively started flow, since these will depend on the initial conditions of the computation. The computed drag coefficient is observed to initially decrease with time, slightly overshoot the steady-state value, and then to oscillate in the range 1.09 ± 0.04 for times later than about 2.0.

The computational predictions for surface pressure coefficient and azimuthal component of the surface vorticity as a function of θ at time $t = 4.0$ are compared with steady-state reference data from fixed-grid finite-difference computations of Johnson [17] (using the velocity-pressure formulation) and Shen and Loc [38] (using the vorticity-velocity formulation) and in Figs. 12 and 13. The computed surface pressure is close to the reference values in the front of the sphere and fairly close in the rear of the sphere, with maximum deviation of about 10% of the reference value near $\theta = 110^\circ$. Similarly, the azimuthal component of the surface vorticity is close to the values obtained by Johnson [17] in the front of the sphere but exhibits a lag of about 10° in the rear of the sphere. This lag in surface vorticity value is consistent with the observation that boundary layer separation in the current computations at this time occurs at about $\theta = 135^\circ$, whereas the steady-state reference value is 127° [18, 28, 37].

The current computations are continued to time $t = 4.0$. At this time, many of the features of the computed flow around the body, such as the body drag and the surface vorticity and

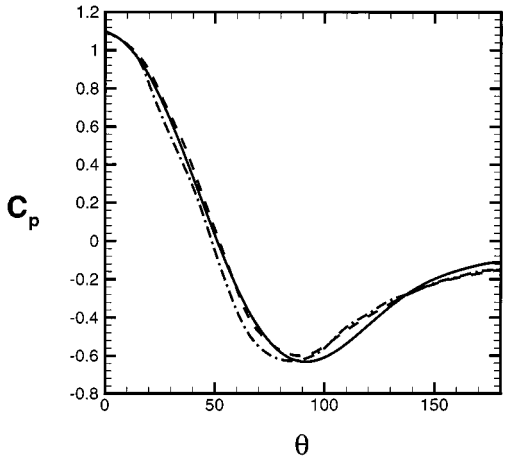

FIG. 12. Comparison of computed surface pressure variation (solid curve) for flow past a sphere at Re = 100 and $t = 4.0$ with steady-state computational data of Johnson [17] (dashed curve) and Shen and Loc [38] (dash-dotted curve).

pressure fields, appear to be approaching the steady-state values. A plot of the length S of the bubble of reversed flow behind the sphere versus time is given in Fig. 14. The bubble length grows steadily during the entire computation and clearly has not reached an asymptotic state by the ending time $t = 4.0$. A rough estimate obtained by extrapolating the bubble length growth curve to the reference value 0.87 and from the results of transient-flow computations by other investigators [17, 38] suggests that the near wake will not reach an approximate steady-state condition until a time of about 8–12.

Lagrangian vorticity methods in general are designed for computation of transient flows and are not particularly efficient for computation of steady-state flow fields due to the restrictions imposed by explicit time-stepping. Steady-state flows are particularly difficult to

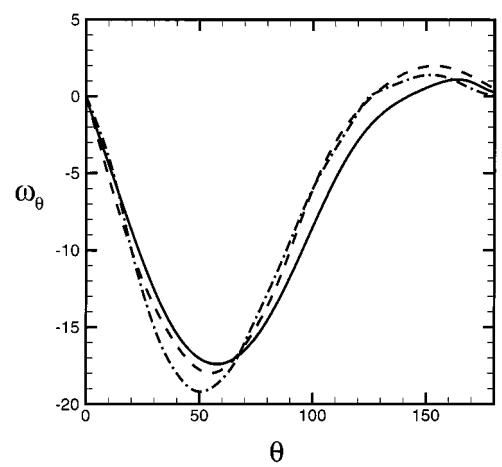

FIG. 13. Comparison of computed azimuthal vorticity variation (solid curve) for flow past a sphere at Re = 100 and $t = 4.0$ with steady-state computational data of Johnson [17] (dashed curve) and Shen and Loc [38] (dash-dotted curve).

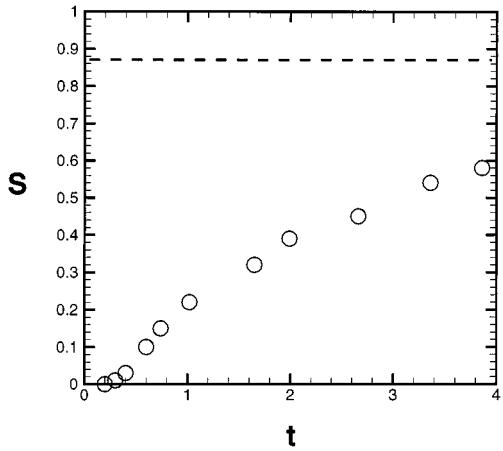

FIG. 14. Time variation of computed wake bubble length S for flow past a sphere at $\mathrm{Re} = 100$.

compute with the method described in the current paper due to the stability restrictions imposed by use of the moving least-squares differentiation method. We are currently working on development of an implicit time-stepping algorithm for use with the moving least-squares differentiation method that should resolve this problem.

8. CONCLUSIONS

Lagrangian vortex methods are commonly used for two- and three-dimensional inviscid flows and for viscous flows with no solid boundaries. These methods offer the advantages that they directly follow the evolving vorticity field and they exhibit little or no numerical dissipation. The present paper describes one approach to extending Lagrangian vorticity-based methods to three-dimensional viscous flows with a no-slip boundary. The proposed tetrahedral vorticity element method deviates from the usual blob and filament basis employed in vortex methods by interpolating the vorticity over a tetrahedral mesh that is continuously refit to the Lagrangian computational points. Tetrahedral elements allow efficient discretization of highly anisotropic vorticity support regions, such as boundary layers and thin vortex sheets and tubes, in such a manner that vorticity does not bleed over the surface of an immersed body.

A fast algorithm is presented for computation of the velocity induced by the tetrahedral elements, consisting of a combination of analytic integration for the nearest few elements, numerical integration with Gaussian quadratures for tetrahedra at intermediate distances, and a box-point multipole acceleration method for distant elements. This fast integration method is also employed for calculation of the inhomogeneous term occurring in the boundary-integral equation for surface pressure on an immersed body. A moving least-squares algorithm is employed for approximation of derivatives occurring in the stretching and diffusion terms of the vorticity transport equation, and a diffusion velocity approach is used to adaptively follow the vorticity support in the presence of diffusion. Validation calculations for Hill's spherical vortex demonstrate that the velocity calculation in the tetrahedral mesh formulation converges much faster than the conventional vortex blob method. Validation calculations for a thin vortex ring with a line source along its axis and for diffusion of a Gaussian columnar vortex with axial flow demonstrate that the moving least-squares

algorithm accurately approximates vorticity stretching and diffusion on Lagrangian points without point regridding. A new algorithm for specification of the vorticity boundary condition is also described that takes into account both normal vorticity diffusion during the time step and direct vorticity transport to tetrahedra attached to boundary points.

The tetrahedral vorticity element method is employed to compute impulsive flow past a sphere at Reynolds number 100 with two different flow resolutions. The computations demonstrate the functioning of the method and suggest areas for future improvement. The computed drag coefficient is found to quickly asymptote to close to the reference value, and the computed pressure and vorticity profiles were in excellent agreement with reference values on the front side of the sphere. However, the explicit time-stepping algorithm used in the current version of this method is not efficient for computation of steady-state flow fields, and the computations were discontinued before the near wake had fully achieved steady state. Development of implicit time-stepping for the moving least-square differentiation method should improve the computational efficiency, as would implementation of the method in a parallel computing environment. Nevertheless, the current form of the tetrahedral vorticity element method is well suited for time-accurate computation of transient incompressible flow problems, as demonstrated for computation of three-dimensional vortex interaction with a circular cylinder in the recent Ph.D. dissertation research of Gossler [10].

ACKNOWLEDGMENTS

Research support for J.S.M. and A.A.G. was provided by the U.S. Army Research Office under Grant DAAH04-96-1-0081, Thomas Doligalski program manager, with The University of Iowa. J.S.M. was also supported by the ASEE/Navy Summer Faculty Research Program during the summer of 1997. Research support for J.R.G. and S.A.H. was provided by the U.S. Office of Naval Research under Grant N0001494WX35105, James Fein program officer, and by internal funding from the Naval Undersea Warfare Center, Division Newport. Computer time was provided by the National Partnership for Advanced Computing Resources, the National Computational Science Alliance, the Navel Undersea Warfare Center, and the Department of Defense High Performance Computing Center. We thank Dr. Gregoire Winckelmans for making available his routine for dividing the surface of a sphere into equilateral triangles.

REFERENCES

1. A. S. Almgren, T. Buttke, and P. Colella, A fast adaptive vortex method in three dimensions, *J. Comput. Phys.* **113**, 177 (1994).
2. G. K. Batchelor, *An Introduction to Fluid Dynamics* (Cambridge Univ. Press, Cambridge, UK, 1967).
3. P. S. Bernard, A deterministic vortex sheet method for boundary layer flow, *J. Comput. Phys.* **117**, 132 (1995).
4. H. Borouchaki and S. H. Lo, Fast Delauney triangularization in three dimensions, *Comput. Methods Appl. Mech. Eng.* **128**, 153 (1995).
5. A. J. Chorin, A numerical study of slightly viscous flow, *J. Fluid Mech.* **57**, 785 (1973).
6. N. R. Clarke and O. R. Tutty, Construction and validation of a discrete vortex method for the two-dimensional incompressible Navier–Stokes equations, *Comput. & Fluids* **23**(6), 751 (1994).
7. P. Degond and S. Mas-Gallic, The weighted particle method for convection-diffusion equations. Part 1. The case of an isotropic viscosity, *Math. Comput.* **53**, 485 (1989).
8. D. Fishelov, A new vortex scheme for viscous flows, *J. Comput. Phys*. **86**, 211 (1990).
9. A. F. Ghoniem and F. S. Sherman, Grid-free simulation of diffusion using random walk methods, *J. Comput. Phys.* **61**, 1 (1985).
10. A. A. Gossler, *A Tetrahedral Element Lagrangian Vorticity Method with Application to Vortex-Cylinder Interaction*, Ph.D. dissertation, University of Iowa, Iowa City, 1999.

11. J. R. Grant and J. S. Marshall, Inviscid interaction of vortex rings: Approach to singularity, in *Third International Workshop on Vortex Flows and Related Numerical Methods, Toulouse, France, Aug. 24–27, 1998*.
12. L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, *J. Comput. Phys.* **73**, 325 (1987).
13. J. Hess and A. Smith, Calculation of potential flows about arbitrary bodies, *Prog. Aeronaut. Sci.* **8**, 1 (1967).
14. S. A. Huyer and J. R. Grant, Computation of incipient separation via solution of the vorticity equation on a Lagrangian mesh, in *European Series in Applied and Industrial Mathematics: Vortex Flows and Related Numerical Methods*, edited by Y. Gagnon, G.-H. Cottet, D. G. Dritschel, A. F. Ghoniem, and E. Meiburg (http://www.emath.fr/proc/Vol.1/1996).
15. S. A. Huyer and J. R. Grant, Examination of unsteady flow past multiple bodies by solution of the vorticity equation on a Lagrangian mesh, in *35th Aerospace Sciences Meeting & Exhibit*, AIAA 97-0661, Reno, Nevada, Jan. 6–10, 1997.
16. S. A. Huyer and J. R. Grant, Computation of unsteady separated flow fields using anisotropic vorticity elements, *J. Fluids Eng.* **118**, 839 (1996).
17. T. A. Johnson, *Numerical and Experimental Investigation of Flow Past a Sphere up to a Reynolds Number of 300*, Ph.D. dissertation, University of Iowa, Iowa City, Iowa, 1996.
18. T. A. Johnson and V. C. Patel, Flow past a sphere up to a Reynolds number of 300, *J. Fluid Mech.* **378**, 19 (1999).
19. S. N. Kempka and J. H. Strickland, *A Method to Simulate Viscous Diffusion of Vorticity by Convective Transport of Vortices at a Non-solenoidal Velocity*, Sandia Nat. Lab. Tech. Rep. SAND93-1763, 1993.
20. P. Koumoutsakos, A. Leonard, and F. Pépin, Boundary conditions for viscous vortex methods, *J. Comput. Phys.* **113**, 52 (1994).
21. A. G. Kravchenko and P. Moin, On the effect of numerical errors in large eddy simultations of turbulent flows, *J. Comput. Phys.* **131**, 310 (1997).
22. A. Landsberg and E. Murman, Control of numerical diffusion in computational modeling of vortex flows, *Progr. Astronaut. Aeronaut.* **146**, 205 (1991).
23. A. T. Leonard, Computing three-dimensional incompressible flows with vortex elements, *Ann. Rev. Fluid Mech.* **17**, 523 (1985).
24. J. S. Marshall and J. R. Grant, A method for determining the velocity induced by highly anisotropic vorticity blobs, *J. Comput. Phys.* **126**, 286 (1996).
25. J. S. Marshall and J. R. Grant, Penetration of a blade into a vortex core: Vorticity response and unsteady blade forces, *J. Fluid Mech.* **306**, 83 (1996).
26. J. S. Marshall and J. R. Grant, A Lagrangian vorticity collocation method for viscous, axisymmetic flows with and without swirl, *J. Comput. Phys.* **138**, 302 (1997).
27. E. Meiburg, Three-dimensional vortex dynamics simulations, in *Fluid Vortices*, edited by S. I. Green (Kluwar Academic, Dordrecht, The Netherlands, 1995), p. 651.
28. I. Nakamura, Steady wake behind a sphere, *Phys. Fluids* **19**(1), 5 (1976).
29. J. N. Newman, Distributions of sources and normal dipoles over a quadrilateral panel, *J. Eng. Math.* **20**, 113 (1986).
30. Y. Ogami and T. Akamatsu, Viscous flow simulation using the discrete vortex model—The diffusion velocity method, *Comput. & Fluids* **19**, 433 (1991).
31. R. Peyret and T. D. Taylor, *Computational Methods for Fluid Flow* (Springer-Verlag, New York, 1990), p. 40.
32. L. F. Rossi, Resurrecting core spreading vortex methods: A new scheme that is both deterministic and convergent," *SIAM J. Sci. Comput.* **17**(2), 370 (1996).
33. G. Russo and J. A. Strain, Fast triangulated vortex methods for the 2D Euler equations, *J. Comput. Phys.* **111**, 291 (1994).
34. J. K. Salmon and M. S. Warren, Skeletons from the treecode closet, *J. Comput. Phys.* **111**, 136 (1994).
35. L. Schiller and A. Naumann, Über die grundlegenden Berechungen bei der Schwerkraftaufbereitung, *Ver. Deut. Ing.* **77**, 318 (1933).
36. H. Schlichting, *Boundary Layer Theory* (McGraw–Hill, New York, 1979), 7th ed., p. 17.

37. S. Shankar and L. van Dommelen, A new diffusion procedure for vortex methods, *J. Comput. Phys.* **127**, 88 (1996).
38. W. Z. Shen and T. P. Loc, Numerical method for unsteady 3D Navier–Stokes equations in velocity-vorticity form, *Comput. & Fluids* **26**(2), 193 (1997).
39. M. Tabata and K. Itakura, A precise computation of drag coefficients of a sphere, *Int. J. Comput. Fluid Dynam.* **9**, 303 (1998).
40. S. Taneda, Experimental investigation of the wake behind a sphere at low Reynolds numbers, *J. Phys. Soc. Japan* **11**(10), 1104 (1956).
41. Z.-H. Teng, Elliptic-vortex method for incompressible flow at high Reynolds number, *J. Comput. Phys.* **46**, 54 (1982).
42. J. S. Uhlman, *An Integral Equation Formulation of the Equations of Motion of an Incompressible Fluid*, Naval Undersea Warfare Center, Technical Report NUWC-NPT TR 10,086, 1992.
43. G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren, Three-dimensional vortex particle and panel methods: Fast tree-code solvers with active error control for arbitrary distributions/geometries, in *Forum on Vortex Methods for Engineering Applications, Sandia Nat. Lab., Albuquerque, NM, Feb. 22–24, 1995*.
44. O. Zienkiewicz, *The Finite Element Method* (McGraw–Hill, New York, 1977).