JOURNAL OF COMPUTATIONAL PHYSICS 106, 30–41 (1993)

Processing of Computed Vector Fields for Visualization

SUSUMU SHIRAYAMA

SofTek Systems, Inc., 1-22-7, 3F, Sangenjaya, Setagaya-ku, Tokyo 154, Japan

Received March 4, 1991; revised April 6, 1992

This paper describes several methods of visualizing the vector fields in a flow analysis. A class of computational algorithms determining the structure of vector fields are stated. These algorithms can be applied to other areas of computational physics. The first part of the paper concentrates on a formulation of the problem. Usually the objective vector fields are obtained on a discretized space. This makes it difficult to construct the computational algorithms. An appropriate interpolation method has to be chosen in order to reconstruct a continuous space. Since the reconstructed space is defined locally, a grid cell in which the solution moves must be found by the efficient algorithm. This becomes a crucial problem in three dimensions. The construction of schemes is performed on both physical and computational spaces in order to overcome such difficulty. © 1993 Academic Press, Inc.

INTRODUCTION

Various flow-visualization techniques have been developed for analyzing experimental or computational flow fields [1, 2]. Such techniques are divided into two groups from the standpoint of analyzing the spatial flow structure. One is the analysis of the scalar field, and the other is that of the vector field. Hesselink [1] mainly classified the flow-visualization techniques for scalar fields. In the vector field, however, the classification of approaches has not been yet settled. Two kinds of descriptions of the vector field have been proposed. One is the direct representation of vectors, i.e., direction and magnitude of the vector is presented at a certain location. The other is the Lagrangian or topological representation of the vector field; oil flow pattern, streamlines, pathlines, streaklines, etc. In most conventional methods, the particle tracing method has often been utilized as a powerful approach in understanding the complicated flow patterns. Several researchers have investigated the algorithm of the particle tracing [2–8]. An early paper on the integration method for particle tracing was published by Murman and Powell [5]. In it, the accuracy of the integration method was checked by using the model velocity field. Smith *et al.* [6] also examined the accuracy of their algorithm by using another model field. Many examples in flow simulations have been cited in other papers, but few statements about the general descriptions of this problem (the formulation, the interpolation method, the integration method, or the accuracy) can be found in their papers.

The present paper addresses the above issues. First we mention the formulation of the problem (Section 1.1). It is assumed that the visualized results in the vector fields are classified according to certain kinds of expressions for the solution of the following ordinary differential equation:

$$\frac{d\mathbf{x}}{ds} = \mathbf{b}.$$

In general, the vector $\mathbf{b}$ is defined in a discrete space, and a continuous space should be reconstructed by an appropriate interpolation method. Thus, the subject of this investigation is to analyze the solution of the above equation in a reconstructed vector field. Occasionally, the ordinary differential equation is transformed into the curvilinear coordinates system in order to make it easier to treat a discretized space (Sections 1.2 and 1.3.1). By using the transformation idea, a particle tracing algorithm that is suitable for large amounts of numerical data is introduced. Next we pursue the characteristic feature of the equation by integration numerically (Section 1.3) or expanding around a critical point (Section 1.4). In Section 2, the accuracy of our method is demonstrated by solving two model problems. A practical application (visualization of flow past a two-dimensional circular cylinder) and a Fortran code used are cited in Appendix C.

1. COMPUTATIONAL APPROACH

1.1. Governing Equations

A vector field in steady or unsteady flows is described by the following dynamical system:

$$\frac{d\mathbf{x}}{ds} = \mathbf{b}, \qquad (1)$$

where $\mathbf{x}$ defines a space in which the solution moves. The system is *autonomous* if the vector field $\mathbf{b}$ is independent of time, and $\mathbf{b}$ is a function of variable $\mathbf{x}$. The system is a *nonautonomous system* if the vector $\mathbf{b}$ is a function of variables $\mathbf{x}$ and time. In the flow analysis, $\mathbf{b}$ is considered as the velocity vector or the vorticity vector. The autonomous system represents a steady flow field, and s is the distance along the streamlines or the vortex lines. In the case of a nonautonomous system, an unsteady flow field will be required, and s denotes the time increment. Note that the direct representation of vectors corresponds to the interpretation of the tangent vector at a certain point in the solution space.

In general, such ordinary differential equations cannot be integrated analytically except when the components of vector $\mathbf{b}$ exhibit special characteristics [9], because $\mathbf{b}$ is a nonlinear function of the variable $\mathbf{x}$ and s. Assuming that a certain particle in the flow field has no weight, the variable $\mathbf{x}$ can be considered as the position of such an imaginary particle. We term the variable $\mathbf{x}$ the particle position (or the particle itself).

The ordinary differential equations are integrated in a discrete space in accordance with the following process:

(a) Determining the grid cell in which each particle lies.

(b) Interpolating the vector $\mathbf{b}$, i.e., a reconstruction of continuous space.

(c1) Integrating the equation (in Section 1.3).

(c2) Linearizing around the critical points (in Section 1.4).

Usually, the algorithms for these procedures are developed in a physical space. In process (a), however, we have great difficulty finding the correct grid cell, since the grid cells are generally nonuniform. Especially in three-dimensions, this difficulty is crucial if we treat a large number of particles. In our computations, we track hundreds of thousands of particles in the grid system, which may have millions of grid points. Therefore, we have to consider the governing equation in a computational space. In next section, we introduce several transformation relations and show the formulation of the governing equation in the computational space.

1.2. Transformation Relations

As a building block for subsequent calculations, we define transformation relations from Cartesian coordinates to the general curvilinear coordinates. The curvilinear coordinates are indicated by $\boldsymbol{\xi}(\xi^1, \xi^2, \xi^3)$. The relations between Cartesian coordinates $\mathbf{x}(x^1, x^2, x^3)$ and the curvilinear coordinates are

$$\begin{aligned} x^1 &= x^1(\xi^1, \xi^2, \xi^3), \\ x^2 &= x^2(\xi^1, \xi^2, \xi^3), \\ x^3 &= x^3(\xi^1, \xi^2, \xi^3). \end{aligned} \tag{2}$$

Three tangent vectors to three coordinate lines at a certain point A_0 which coordinates are $\mathbf{x}_0$; (x_0^1, x_0^2, x_0^3) in Cartesian system and $\boldsymbol{\xi}_0$; $(\xi_0^1, \xi_0^2, \xi_0^3)$ in the curvilinear system are represented by:

$$\tau_i(A_0) = \sum_j \left(\frac{\partial x^j}{\partial \xi^i}\right) \mathbf{i}_j \qquad (i, j = 1, 2, 3), \tag{3}$$

where $\mathbf{i}_j$ is the base of Cartesian coordinates, i.e., $\mathbf{i}_1 = (1, 0, 0)$, $\mathbf{i}_2 = (0, 1, 0)$, and $\mathbf{i}_3 = (0, 0, 1)$. A transformation matrix P is defined as

$$P_{ij} = \partial x^i / \partial \xi^j. \tag{4}$$

If the determinant of matrix P is not zero, the three tangent vectors become linearly independent and make a natural base at A_0. This base is denoted by $\mathbf{e}_i$, and $\mathbf{e}_i = \tau_i$. The components of an arbitrary vector $\mathbf{v}$ at A_0 is expressed by (v^1, v^2, v^3) in Cartesian system and (V^1, V^2, V^3) in the curvilinear system; $\mathbf{v}$ can also be represented by using the base in Cartesian coordinates,

$$\mathbf{v} = v^1 \mathbf{i}_1 + v^2 \mathbf{i}_2 + v^3 \mathbf{i}_3, \tag{5a}$$

and in the curvilinear coordinates,

$$\mathbf{v} = V^1 \mathbf{e}_1 + V^2 \mathbf{e}_2 + V^3 \mathbf{e}_3. \tag{5b}$$

Then we consider transformation relations of two bases. Since the determinant of matrix P is not zero, the inverse matrix P^{-1} can be determined. The components of matrix P^{-1} are

$$P_{ij}^{-1} = \partial \xi^i / \partial x^j. \tag{6}$$

The relation between the two bases is obtained by using the matrices P and P^{-1}. That is,

$$\mathbf{e}_i = \sum_j P_{ji} \mathbf{i}_j \tag{7a}$$

and

$$\mathbf{i}_i = \sum_j P_{ji}^{-1} \mathbf{e}_j. \tag{7b}$$

Substituting for $\mathbf{i}_i$ in Eq. (5a) from Eq. (7b), $\mathbf{v}$ becomes

$$\mathbf{v} = \left(\sum_j v^j P_{1j}^{-1}\right) \mathbf{e}_1 + \left(\sum_j v^j P_{2j}^{-1}\right) \mathbf{e}_2 + \left(\sum_j v^j P_{3j}^{-1}\right) \mathbf{e}_3. \tag{8}$$

Comparing Eq. (8) with Eq. (5b), the components of $\mathbf{v}$ in the curvilinear coordinates are

$$V^k = \sum_j v^j P_{kj}^{-1}. \tag{9a}$$

Similarly, the following equation is obtained:

$$v^k = \sum_j V^j P_{kj}. \tag{9b}$$

Finally, we define two base vectors (the covariant base vectors $\mathbf{a}_i$ and the contravariant base vectors $\mathbf{a}^i$) as

$$(\mathbf{a}_i)_j = (\mathbf{e}_i)_j = P_{ji}, \tag{10a}$$

$$(\mathbf{a}^i)_j = P_{ij}^{-1}, \tag{10b}$$

and

$$\mathbf{a}_i \cdot \mathbf{a}^j = \delta_i^j, \tag{10c}$$

where $(\cdot)_j$ denotes the component in the j-direction and δ_i^j is the Kronecker delta.

Then, we propose that Eq. (1) is transformed into the general curvilinear coordinates and that the transformed equation can be solved in order to overcome the difficulty in process (a). Since the curvilinear system is defined as the orthogonized and normalized coordinates system, it is easy to find the grid cell in which each particle lies. If matrix P^{-1} operates on Eq. (1) from the left-hand side, the equation is transformed into the curvilinear coordinates:

$$\frac{d\boldsymbol{\xi}}{ds'} = \mathbf{B}, \tag{11a}$$

$$d\boldsymbol{\xi} = P^{-1}\, d\mathbf{x}, \tag{11b}$$

$$\mathbf{B} = P^{-1}\left(\mathbf{b} - \frac{d\mathbf{x}_g}{dt}\right), \tag{11c}$$

where in the case of autonomous system s' is the distance along the streamlines or the vortex lines in the curvilinear coordinates, in the case of nonautonomous system, s' denotes the time increment and $\mathbf{x}_g$ is the grid position.

1.3. Trajectory Integration

1.3.1. *Interpolation Methods*

In solving the ordinary differential equation (1) or (11), it should be considered how to determine an adequate interpolation of the vector $\mathbf{b}$, i.e., the reconstruction of the continuous space. It is difficult to determine the best interpolation method because of two unsolved problems: (i) the relation between the geometrical behavior of the numerical solution and the discretization of the nonlinear partial differential equations; and (ii) the connectivity of the local solutions. Currently, it is popular to use the simple interpolation (linear or bilinear interpolation) with a sufficient number of grid points, and we yield to this idea. In a future paper, however, we will describe a more accurate algorithm which depends on the flow solvers and will analyze the numerical error in the interpolating process.

In the actual computation, we have to decide on the algorithm to solve Eq. (11). Then we show a specific algorithm in three-dimensions. In the following explanation, (x, y, z) represents (x^1, x^2, x^3) which appeared previously, (ξ, η, ζ) stands for (ξ^1, ξ^2, ξ^3), the component of $\mathbf{b}$ is (u, v, w) and $\mathbf{B}$ is (U, V, W). Also the superscript denotes the time step and the subscripts (k, l, m) represents the index of the grid cell. At first, the vector field is computed. Next the initial position of the particles $(x^{(0)}, y^{(0)}, z^{(0)})$ is given in the physical space (in Cartesian coordinates), and then the grid cells in which each particle lies are found. In the last process, the local transformation matrices are determined. Then, the initial position in the curvilinear system is obtained by Eq. (11b). In this stage, we face the great difficulty mentioned previously. An algorithm to find the grid cell in which a particle lies should be considered. We solve this problem by the geometrical interpretation of the two base vectors. That is, we utilize Eq. (10c), and the covariant base vectors are transformed to the unit vectors by operating on the contravariant base vectors.

Let $(x_{k,l,m}, y_{k,l,m}, z_{k,l,m})$ denote a grid point aimed at testing, and let $\mathbf{a}_i$ and $\mathbf{a}^i$ be the covariant and the contravariant base vectors at the grid point (k, l, m). A vector $\mathbf{r}$ is defined as a test vector and has three components $(x^{(0)} - x_{k,l,m}, y^{(0)} - y_{k,l,m}, z^{(0)} - z_{k,l,m})$. If the position of particle $(x^{(0)}, y^{(0)}, z^{(0)})$ is located in this grid cell indicated by (k, l, m), the following three relations are satisfied:

$$\begin{aligned} 0 &\leqslant \mathbf{r}\cdot\mathbf{a}^1 \leqslant 1,\\ 0 &\leqslant \mathbf{r}\cdot\mathbf{a}^2 \leqslant 1,\\ 0 &\leqslant \mathbf{r}\cdot\mathbf{a}^3 \leqslant 1. \end{aligned} \tag{12a}$$

The particle position in the grid cell $\delta\boldsymbol{\xi}(\delta\xi, \delta\eta, \delta\zeta)$ is

$$\begin{aligned} \delta\xi &= \mathbf{r}\cdot\mathbf{a}^1,\\ \delta\eta &= \mathbf{r}\cdot\mathbf{a}^2,\\ \delta\zeta &= \mathbf{r}\cdot\mathbf{a}^3. \end{aligned} \tag{12b}$$

The geometrical interpretation in two-dimensions is shown in Fig. 1. Strictly speaking, if the cell configuration is not a parallelogram, there are some possibilities that the algorithm may not find the correct position of the particle. We can compute the correct position of the particle by means

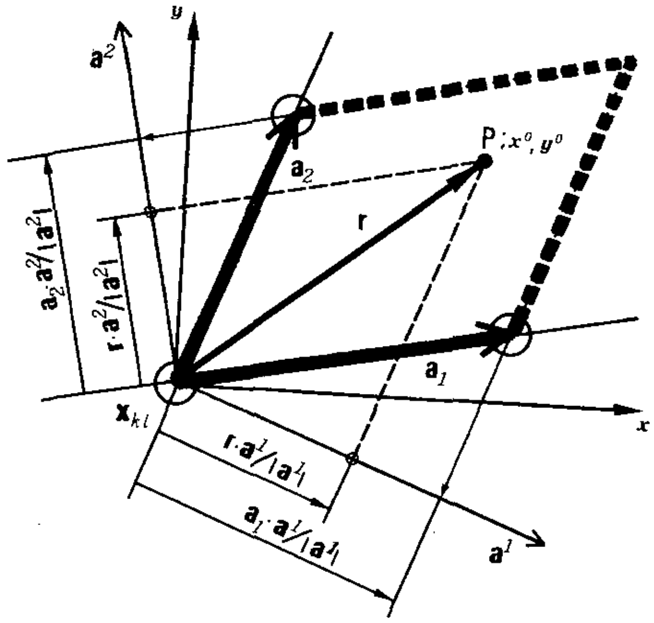
y
x
$\mathbf{a}^2$
$\mathbf{a}_2$
$\mathbf{a}_1$
$\mathbf{a}^1$
P : x^0, y^0
r
$\mathbf{X}_{kl}$
$\mathbf{a}_2 \cdot \mathbf{a}^2/|\mathbf{a}^2|$
$\mathbf{r} \cdot \mathbf{a}^2/|\mathbf{a}^2|$
$\mathbf{r} \cdot \mathbf{a}^1/|\mathbf{a}^1|$
$\mathbf{a}_1 \cdot \mathbf{a}^1/|\mathbf{a}^1|$

FIG. 1. Geometrical interpretation of relations among the test vector r, the covariant base-vectors $\mathbf{a}_i$ and the contravariant base-vectors $\mathbf{a}^i$ in a certain grid cell.

of the modified algorithm cited in Appendix A (see Fig. 2 also). However, the modified algorithm consumes much computational time. In practical cases, the location of grid points or the position on the coordinate lines is often chosen as the initial value of $(x^{(0)}, y^{(0)}, z^{(0)})$, and then the initial position in the curvilinear system is obtained immediately. For convenience, the initial position is located at the grid point $\mathbf{x}_{k,l,m}$; $(\xi^{(0)}, \eta^{(0)}, \zeta^{(0)}) = (k^{(0)}, l^{(0)}, m^{(0)})$. Then Eq. (11) is integrated by using the adequate method with $\mathbf{B} = \mathbf{B}_{k^{(0)}, l^{(0)}, m^{(0)}}$, and next the values $(\xi^{(1)}, \eta^{(1)}, \zeta^{(1)})$ are obtained. The grid cell in which this particle lies is found as

$$\begin{aligned} k &= \mathrm{int}(\xi^{(1)}), \\ l &= \mathrm{int}(\eta^{(1)}), \\ m &= \mathrm{int}(\zeta^{(1)}), \end{aligned} \tag{13a}$$

where $\mathrm{int}(f)$ indicates the integer which does not exceed f. The particle position in the grid cell is

$$\begin{aligned} \delta\xi &= \xi^{(1)} - k, \\ \delta\eta &= \eta^{(1)} - l, \\ \delta\zeta &= \zeta^{(1)} - m. \end{aligned} \tag{13b}$$

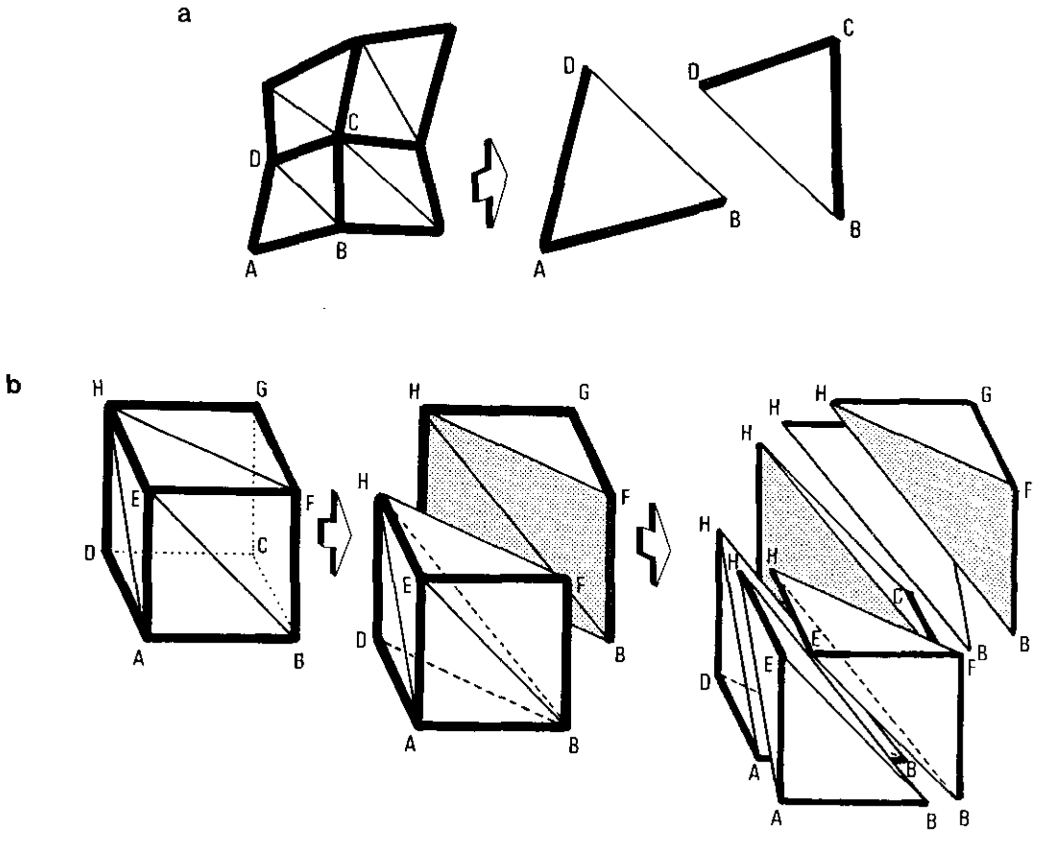

FIG. 2. (a) 2D grid cell subdivision; two-triangle subdivision. (b) 3D grid cell subdivision; six-tetrahedron subdivision.

At the next time step, the contravariant vectors must be interpolated. We use the bilinear isoparametric transformation as the interpolation method. The contravariant vector **B** at point a in Fig. 3 is obtained as follows:

$$\begin{aligned}\mathbf{B}_a = [&\mathbf{B}_{k,l,m}(1-\delta\xi)(1-\delta\eta) + \mathbf{B}_{k+1,l,m}\,\delta\xi(1-\delta\eta)\\ &+ \mathbf{B}_{k+1,l+1,m}\,\delta\xi\,\delta\eta\\ &+ \mathbf{B}_{k,l+1,m}(1-\delta\xi)\,\delta\eta](1-\delta\zeta)\\ &+ [\mathbf{B}_{k,l,m+1}(1-\delta\xi)(1-\delta\eta)\\ &+ \mathbf{B}_{k+1,l,m+1}\,\delta\xi(1-\delta\eta)\\ &+ \mathbf{B}_{k+1,l+1,m+1}\,\delta\xi\,\delta\eta\\ &+ \mathbf{B}_{k,l+1,m+1}(1-\delta\xi)\,\delta\eta]\,\delta\zeta.\end{aligned} \tag{14}$$

Repeating these procedures, the solution of Eq. (11) for $(\xi^{(0)}, \eta^{(0)}, \zeta^{(0)})$ is obtained in the curvilinear system. This solution set $[(\xi^{(1)}, \eta^{(1)}, \zeta^{(1)}), (\xi^{(2)}, \eta^{(2)}, \zeta^{(2)}), ...]$ is transformed into the physical space. For example, the position of the particle in the physical space is

$$\begin{aligned}\mathbf{x}_a = [&\mathbf{x}_{k,l,m}(1-\delta\xi)(1-\delta\eta) + \mathbf{x}_{k+1,l,m}\,\delta\xi(1-\delta\eta)\\ &+ \mathbf{x}_{k+1,l+1,m}\,\delta\xi\,\delta\eta\\ &+ \mathbf{x}_{k,l+1,m}(1-\delta\xi)\,\delta\eta](1-\delta\zeta)\\ &+ [\mathbf{x}_{k,l,m+1}(1-\delta\xi)(1-\delta\eta)\\ &+ \mathbf{x}_{k+1,l,m+1}\,\delta\xi(1-\delta\eta)\\ &+ \mathbf{x}_{k+1,l+1,m+1}\,\delta\xi\,\delta\eta\\ &+ \mathbf{x}_{k,l+1,m+1}(1-\delta\xi)\,\delta\eta]\,\delta\zeta.\end{aligned} \tag{15}$$

In this way, the trajectory of the particle is computed.

1.3.2. *Numerical Integration Methods*

Since the interpolation methods depend largely on the flow solvers as mentioned in the previous section, it is very difficult to improve the accuracy of the numerical solution for Eq. (1) or (11) by altering the spatial interpolation methods. In order to obtain a more accurate numerical solution for Eq. (1) or (11), the order of accuracy of the integration method of the computation has to be estimated. In this section, the accuracy of the integration methods is discussed by using the model velocity field proposed in Ref. [5]. It is expected that the process of coordinate transformation makes it complicated to evaluate the accuracy. Here we consider the accuracy of the numerical solution in the physical space. The model set of equations is

$$\begin{pmatrix} b^1 \\ b^2 \end{pmatrix} = \frac{d}{ds}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}. \tag{16}$$

This set has the following analytical solution:

$$\begin{pmatrix} x(s) \\ y(s) \end{pmatrix} = \exp(\alpha s)\begin{pmatrix} \cos(\beta s) & -\sin(\beta s) \\ \sin(\beta s) & \cos(\beta s) \end{pmatrix}\begin{pmatrix} x(0) \\ y(0) \end{pmatrix}. \tag{17}$$

For convenience, we consider that two parameters (α, β) are represented by using one parameter θ as follows: $\alpha = \cos(\theta)$; $\beta = \sin(\theta)$. Here $\mathbf{x}(s+\delta s)$ is approximated by expanding around s up through the second order:

$$\mathbf{x}(s+\delta s) = \mathbf{x} + \frac{d\mathbf{x}}{ds}\,\delta s + \frac{1}{2}\frac{d^2\mathbf{x}}{ds^2}\,\delta s^2 + O(\delta s^3). \tag{18}$$

Substituting for $d\mathbf{x}/ds$ and $d^2\mathbf{x}/ds^2$ in Eq. (18) from Eq. (16), Eq. (18) is rewritten by

$$\mathbf{x}(s+\delta s) \simeq \mathbf{x} + C\mathbf{x}\,\delta s + \tfrac{1}{2}D\mathbf{x}\,\delta s^2. \tag{19}$$

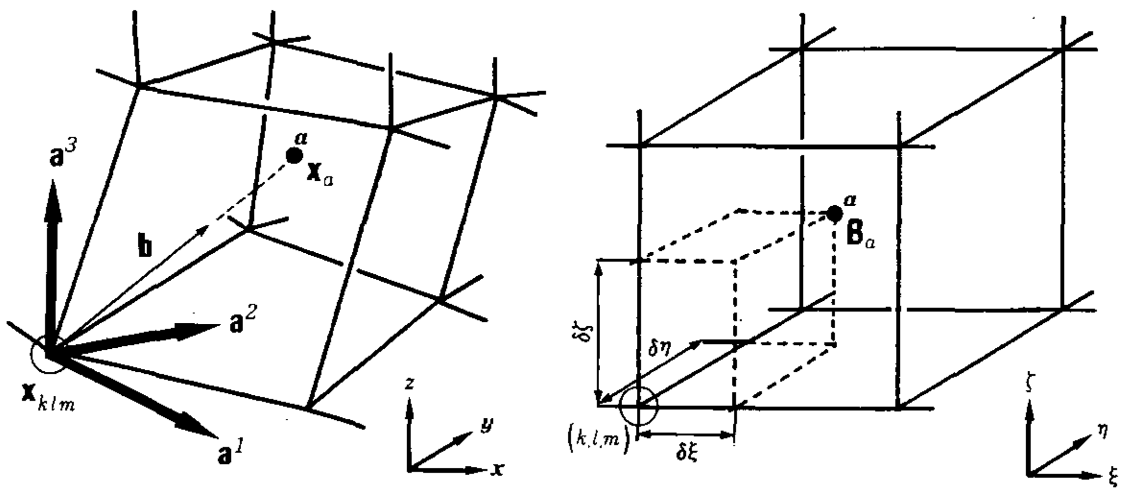

FIG. 3. Relations between Cartesian coordinates and generalized curvilinear coordinates.

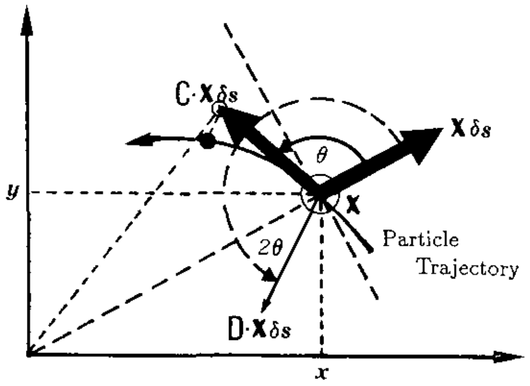

FIG. 4. Geometrical interpretation of matrices C and D.

where

$$C=\begin{pmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{pmatrix}$$

and

$$D=\begin{pmatrix}\cos(2\theta) & -\sin(2\theta)\\ \sin(2\theta) & \cos(2\theta)\end{pmatrix}.$$

Is s is taken as time t, $C\mathbf{x}$ is the velocity and $D\mathbf{x}$ is the acceleration.

Then we consider geometrical interpretation of matrices C and D. From the second term in the right-hand side of Eq. (19), matrix C rotates the vector $\mathbf{x}$ with angle θ. Also, matrix D rotates the vector $\mathbf{x}$ with angle 2θ. Figure 4 shows the geometrical interpretation of operators C and D in the case that θ takes a value between 0° and 180°. If a first-order-accurate method is used for the integration, the following remarks are derived from the above considerations:

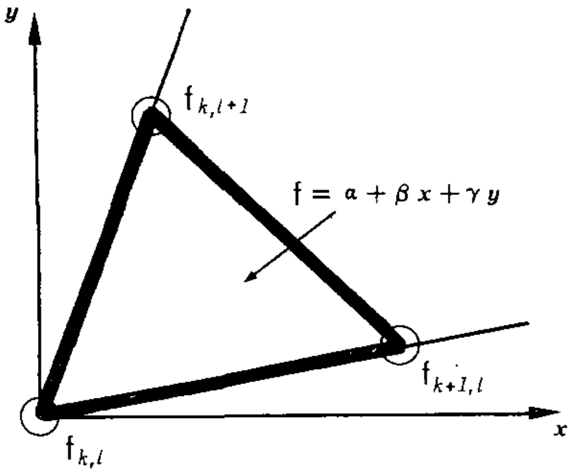

FIG. 5. Schematic of grid cell interpolated by using the linear function.

(i) if $0° < \theta < 180°$, the numerical trajectory always detaches from the exact trajectory to the outward direction of the origin.

(ii) if $180° < \theta < 360°$, the numerical trajectory always detaches from the exact trajectory to the inward direction of the origin.

In this way, if the step size δs is not small, it is found that the first-order-accurate method is not desired for the trajectory integration. Therefore, in our computation, a second-order-accurate predictor–corrector method has been utilized. Strictly speaking, this cannot be extended to any interpolated vector fields as the general description even if the vector field is represented by such a model equation. However, if bilinear interpolation is used, the second-order-accurate methods have good performance in integrating Eqs. (1) and (11) (see [5] and our results in this paper).

Also if linear interpolation is used, the first-order-accurate methods should not be used for the integration in the case of the model vector field; see appendix B (see Fig. 5 also).

1.4. Critical-Point Concepts

In previous sections, we have attempted to perform the direct integration of Eq. (1) or (11). As mentioned in Helman and Hesselink [15], the visualizations related to the direct integration reflect the ones that most closely resemble the pictures already familiar to those in the fluid dynamics field, such as oil flow pattern, smoke visualizations, etc. Helman and Hesselink have investigated the vector field topology in fluid flow by using the critical-point concept. However, in their paper [15], they have not clearly shown the relation between the conventional approaches and their critical-point concept. In this section, the global characteristic of the solution will be examined by the linearization of Eq. (1) or (11) around a critical point. Local solutions to the Navier–Stokes equations have shown that the critical points play an important role in the investigation of vector fields [10–12]. The critical points are defined as follows:

$$\frac{b^i}{b^j}=\frac{0}{0} \quad \text{for} \quad i \neq j. \tag{20}$$

The classification of the critical point indicates the global characteristic of the solution. In order to classify the critical point, Eqs. (1) and (11) are expanded and linearized around a certain point c. The linearized equations are

$$\frac{d\mathbf{x}}{ds}=E\mathbf{x}+\mathbf{b}_c, \tag{21}$$

$$\frac{d\xi}{ds'}=F\xi+\mathbf{B}_c. \tag{22}$$

where $E_{ij} = \partial b^i/\partial x^j$ and $F_{ij} = \partial B^i/\partial \xi^j$. If point c is a critical point, the classification of the critical point is made by enquiring into the eigenvalues and the eigenvectors of matrices E and F. Here we find the relation

$$F = P^{-1}EP. \tag{23}$$

As this relation means the similarity transformation of matrix E, the two matrices E and F have the same eigenvalues and eigenvectors. Therefore, we only investigate either E or F. Then, we consider how the analysis of critical points is valid in the study of the detailed structure of the vector field. According to the theorem of Hartman–Grobman [13], if the critical point is a hyperbolic point (all eigenvalues have real parts unequal to zero), these truncated systems (Eqs. (21) and (22)) are sufficient to determine the local topology of the vector field [12]. The detailed statements about the classification of the critical points are described in Refs. [10–12]. Also Helman and Hesselink have shown several examples of the critical points in the computed vector fields [15]. In this paper, we introduce the following three comments for the critical-point concept in the computed vector fields.

First, there is the case where first-order interpolation is selected for **b** and P^{-1}. In this case, the structure of the vector field in the reconstructed space is fully described by the critical-point concept (See Eqs. (B3) and (B5) in Appendix B).

The second case is about the model vector field in Section 1.3.2. The eigenvalues at critical points in the vector field, represented by the model equation, are the conjugate complex. If $\alpha > 0$, the solution has the unstable focus. If $\alpha < 0$, the solution is characterized by the stable focus. In the case of $\alpha = 0$, the solution is represented by the circle. Since these critical points play an important role in investigation of the vortical flow, it is required in the particle tracing algorithms that the model vector field should be precisely solved by numerical experiments.

The third comment is about the property of critical points in the case of two-dimensional incompressible velocity fields. At a certain critical point (x_c, y_c), Eq. (21) becomes

$$\frac{d}{ds}\begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} = \begin{pmatrix} \dfrac{\partial u}{\partial x} & \dfrac{\partial u}{\partial y} \\ \dfrac{\partial v}{\partial x} & \dfrac{\partial v}{\partial y} \end{pmatrix}_{x_c, y_c} \begin{pmatrix} x - x_c \\ y - x_c \end{pmatrix}. \tag{24}$$

The condition for incompressibility is

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{25}$$

Let d denote $(\partial u/\partial x)(\partial v/\partial y) - (\partial u/\partial y)(\partial v/\partial x)$. The eigenvalues are as follows:

If $d \leqslant 0$ then

$$\lambda_1 = \sqrt{-d}, \qquad \lambda_2 = -\sqrt{-d}; \tag{26a}$$

the critical point is the saddle.

If $d > 0$ then

$$\lambda_1 = i\sqrt{d}, \qquad \lambda_2 = -i\sqrt{d}; \tag{26b}$$

the critical point is the center.

Therefore, we should not have node and spiral points in the velocity field. In the incompressible flow field, d is related to the pressure field:

$$d = \frac{1}{2}\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right). \tag{26c}$$

According to Eq. (26), the pressure value at the saddle point is higher than the mean value surrounding the point, and the value at the center is lower. These comments are useful in verifying the numerical solution.

2. RESULTS FOR MODEL VECTOR FIELDS

We show the accuracy of our method by using the model velocity fields in two dimensions. Two model fields are tested. The first is the model vector field in Ref. [5]. The other is the potential flow field around a circular cylinder with a circulation. The velocity field is given by

$$\begin{aligned} u &= U_0 + U_0 \frac{a^4}{r^4}\left(-(x-c_x)^2 + (y-c_y)^2\right) - \frac{\Gamma}{2\pi}\frac{(y-c_y)}{r^2}, \\ v &= -2U_0 \frac{a^4}{r^4}(x-c_x)(y-c_y) + \frac{\Gamma}{2\pi}\frac{(x-c_x)}{r^2}, \end{aligned} \tag{26}$$

where $r = \sqrt{(x-c_x)^2 + (y-c_y)^2}$, U_0 is the uniform flow velocity, Γ represents the circulation, a is the radius of the cylinder, and (c_x, c_y) is the center of the cylinder. The first case is referred to as Model I and the second case as Model II.

For these fields, Eq. (11) is solved in the discretized space. Two kinds of discrete space are considered. One consists of the grid points distributed randomly and is termed as *RG* (in Fig. 6a). The other is the smoothed space, and is termed as *SG* (in Fig. 6b). Figures 7a and 8a show the exact solutions of Model I ($\theta = 120°$) and Model II ($U_0 = 1.0$, $a = 0.5$, $c_x = c_y = 0.0$, and $\Gamma = 6.0$), respectively. In these figures, the character p indicates the initial values of ξ in Eq. (11), and the thick lines represent the numerical solutions on *RG* after

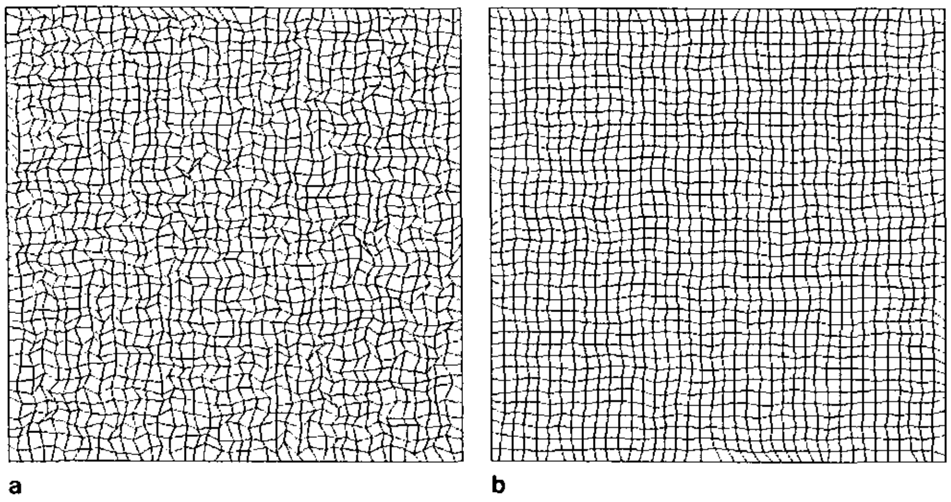

FIG. 6. (a) Grid system with randomly distributed grid points (RG). The number of grid points is 40×40. (b) Smoothed grid system (SG).

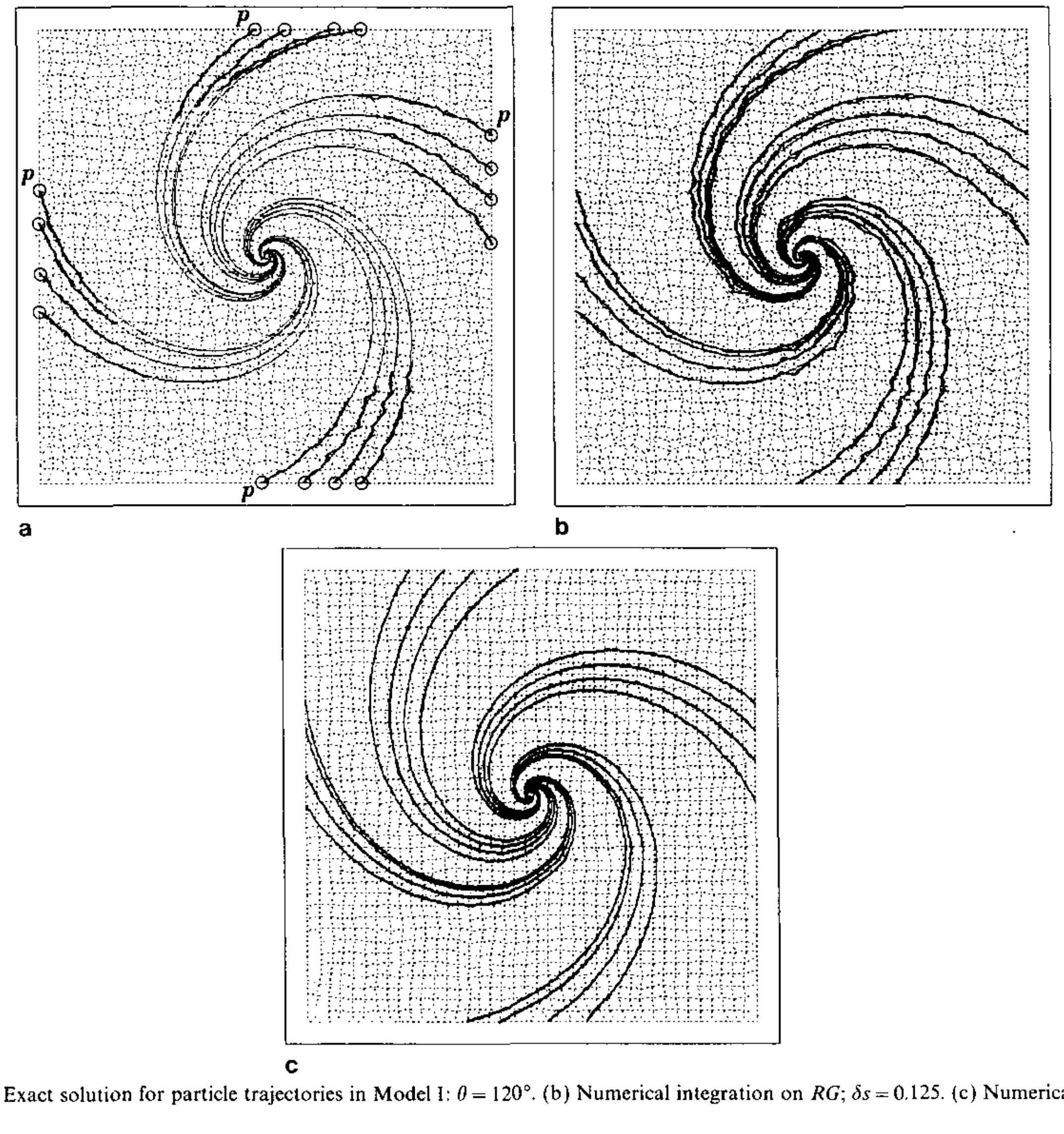

FIG. 7. (a) Exact solution for particle trajectories in Model I: $\theta = 120°$. (b) Numerical integration on RG; $\delta s = 0.125$. (c) Numerical integration on SG; $\delta s = 0.125$.

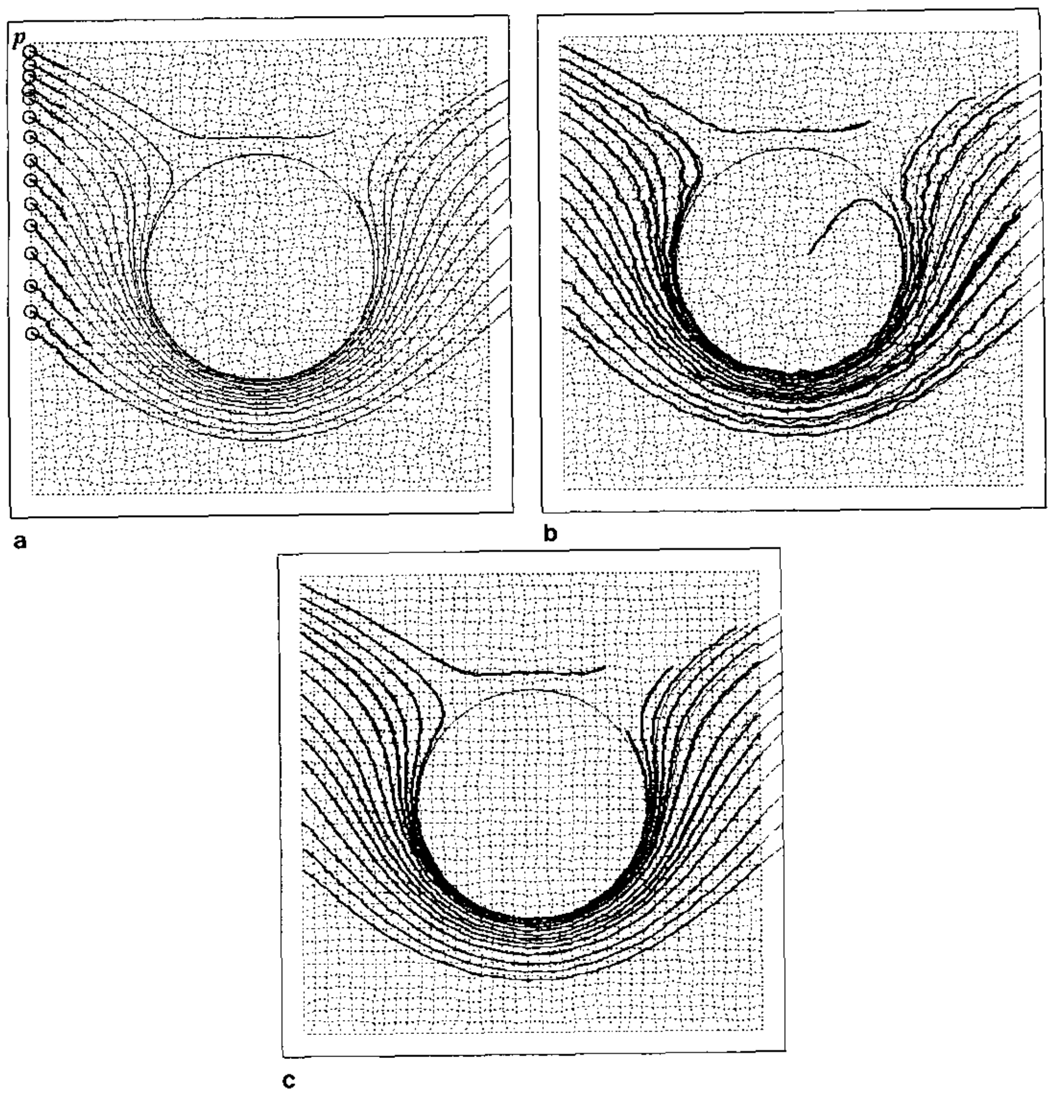

FIG. 8. (a) Exact solution for particle trajectories in Model II: $U_0 = 1.0$, $a = 0.5$, $c_x = c_y = 0.0$, and $\Gamma = 6$. (b) Numerical integration on RG; $\delta s = 0.125$. (c) Numerical integration on SG; $\delta s = 0.125$.

several time steps. The numerical results for each case are demonstrated in Figs. 7b, c and 8b, c. As seen in Figs. 7b and 8b, even if trajectory integrations are performed with the hardly distorted grid system, the qualitative feature can be captured by means of the present algorithm. If we use the smoothed grid system (in Figs. 7c and 8c), the results are in fair agreements with the exact trajectories.

Our techniques have been applied to investigating many flow fields [2]. An application and a Fortran code used in a two-dimensional field are cited in Appendix C.

CONCLUSIONS

Several techniques of visualizing the vector fields in flow analysis have been classified as enquiring into the solution of the nonlinear autonomous or nonautonomous dynamical system. Two methods are considered: the numerical integration and the linearization around the critical point.

For the implementation of computational algorithms, two difficulties are encountered because the space is discretized by grid cells. The first difficulty is the interpolation used to reconstruct the continuous space. The other is determining the grid cell in which the solution moves. It is found that the coordinates transformation into the generalized curvilinear system is useful to overcome second difficulty. Our results show that the simple interpolation is sufficient to obtain the solution qualitatively. By means of the transformation idea and the simple interpolation, a particle tracing algorithm that is suitable for large amounts of numerical data has been developed.

Additionally, we have investigated numerical integration methods. It is shown by the geometrical interpretation of the coefficients of the Taylor series expansion that the first-order-accurate method always leads the erroneous results in vector fields reconstructed by the linear or bilinear interpolation. It is suggested that more accurate methods have to be used for numerical integration in any vector field.

Also the critical-point concept has been described. Three comments mentioned in Section 1.4 are useful in verifying numerical solutions.

APPENDIX A

At first, triangulation is performed as shown in Fig. 2. The cell is divided into two triangles in two dimensions and into six tetrahedrons in three dimensions. The test vector and the contravariant base vectors are calculated at each triangle or tetrahedron. Equation (12a) is modified as follows:

$$\begin{gathered} 0 \leqslant \mathbf{r}\cdot\mathbf{a}^1, \\ 0 \leqslant \mathbf{r}\cdot\mathbf{a}^2, \\ 0 \leqslant \mathbf{r}\cdot\mathbf{a}^3, \\ \mathbf{r}\cdot\mathbf{a}^1+\mathbf{r}\cdot\mathbf{a}^2+\mathbf{r}\cdot\mathbf{a}^3 \leqslant 1. \end{gathered} \tag{A1}$$

This algorithm can be applied to the problem: "Does a point lie inside a polygon?" [14].

APPENDIX B

The vector **b** is interpolated by using the linear function as shown in Fig. 5. Equation (1) becomes

$$\frac{d}{ds}\begin{pmatrix} x-x_{k,l} \\ y-y_{k,l} \end{pmatrix} = C\cdot D\cdot \begin{pmatrix} x-x_{k,l} \\ y-x_{k,l} \end{pmatrix},$$

$$C=\begin{pmatrix} \Delta_k^+ b^1 & \Delta_l^+ b^1 \\ \Delta_k^+ b^2 & \Delta_l^+ b^2 \end{pmatrix},$$

and

$$D=\frac{1}{J_a}\begin{pmatrix} \Delta_l^+ y & -\Delta_l^+ x \\ -\Delta_k^+ y & \Delta_k^+ x \end{pmatrix}, \tag{B1}$$

where $\Delta_k^+ f$ means $f_{k+1,l}-f_{k,l}$, $\Delta_l^+ f$ means $f_{k,l+1}-f_{k,l}$, and $J_a=\Delta_k^+ x\cdot\Delta_l^+ y-\Delta_k^+ y\cdot\Delta_l^+ x$.

In the case of the model vector field, substituting for b^1 and b^2 in Eq. (B1) from Eq. (16), Eq. (B1) becomes

$$\frac{d}{ds}\begin{pmatrix} x-x_{k,l} \\ y-y_{k,l} \end{pmatrix} = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix}\begin{pmatrix} x-x_{k,l} \\ y-y_{k,l} \end{pmatrix}. \tag{B2}$$

Equation (B2) is same as Eq. (15). Therefore, in the linear interpolated field, comments about the model vector field are rigorously satisfied.

In Eq. (B1), matrix D is equivalent to the transform matrix P^{-1} discretized to first order. Eq. (B1) becomes

$$\frac{d}{ds}\begin{pmatrix} x-x_{k,l} \\ y-y_{k,l} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial u}{\partial x} & \dfrac{\partial u}{\partial y} \\ \dfrac{\partial v}{\partial x} & \dfrac{\partial v}{\partial y} \end{pmatrix}_{x_{k,l},\,y_{k,l}} \begin{pmatrix} x-x_{k,l} \\ y-x_{k,l} \end{pmatrix}. \tag{B3}$$

We operate matrices D and D^{-1} on Eq. (B3) as

$$D\frac{d}{ds}\begin{pmatrix} x-x_{k,l} \\ y-y_{k,l} \end{pmatrix} = D\begin{pmatrix} \dfrac{\partial u}{\partial x} & \dfrac{\partial u}{\partial y} \\ \dfrac{\partial v}{\partial x} & \dfrac{\partial v}{\partial y} \end{pmatrix}_{x_{k,l},\,y_{k,l}} D^{-1}D\begin{pmatrix} x-x_{k,l} \\ y-x_{k,l} \end{pmatrix}. \tag{B4}$$

We obtain

$$\frac{d}{ds'}\begin{pmatrix} \xi-\xi_{k,l} \\ \eta-\eta_{k,l} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial U}{\partial \xi} & \dfrac{\partial U}{\partial \eta} \\ \dfrac{\partial V}{\partial \xi} & \dfrac{\partial V}{\partial \eta} \end{pmatrix}_{\xi_{k,l},\,\eta_{k,l}} \begin{pmatrix} \xi-\xi_{k,l} \\ \eta-\eta_{k,l} \end{pmatrix}. \tag{B5}$$

Thus, the method implemented in the computational space has the same property with the method in the physical space.

APPENDIX C

We simulate the flow past a circular cylinder in two dimensions. The flow field is computed by solving the incompressible Navier–Stokes equations. The velocity field is visualized by solving Eq. (11) in two dimensions, where **B** is set as the contravariant velocity. In a steady flow field, we can obtain a streamline pattern. In an unsteady flow field, we have two options to visualize this field. One is the instantaneous streamline. This pattern can be obtained by using a frozen velocity field; that is, we use instantaneous values of **B**. The other is the computed streakline.

To explain the computed streakline, consider the grid system shown in Fig. 9a. The particles are released at every time step from several points near the body surface. That is, the initial value of Eq. (11) in two dimensions is $(\xi^{(0)}, \eta^{(0)}) = (2, 8l)$, $l = 1, 2, 3, 4, 6, 7, 8, 9$. Equation (11) is solved at every time step for all particles. The pattern of particles at $t = 50.0$ is presented in Fig. 9b. This pattern corresponds to the experimental streaklines and is called the computed streakline. The Fortran program we used, which

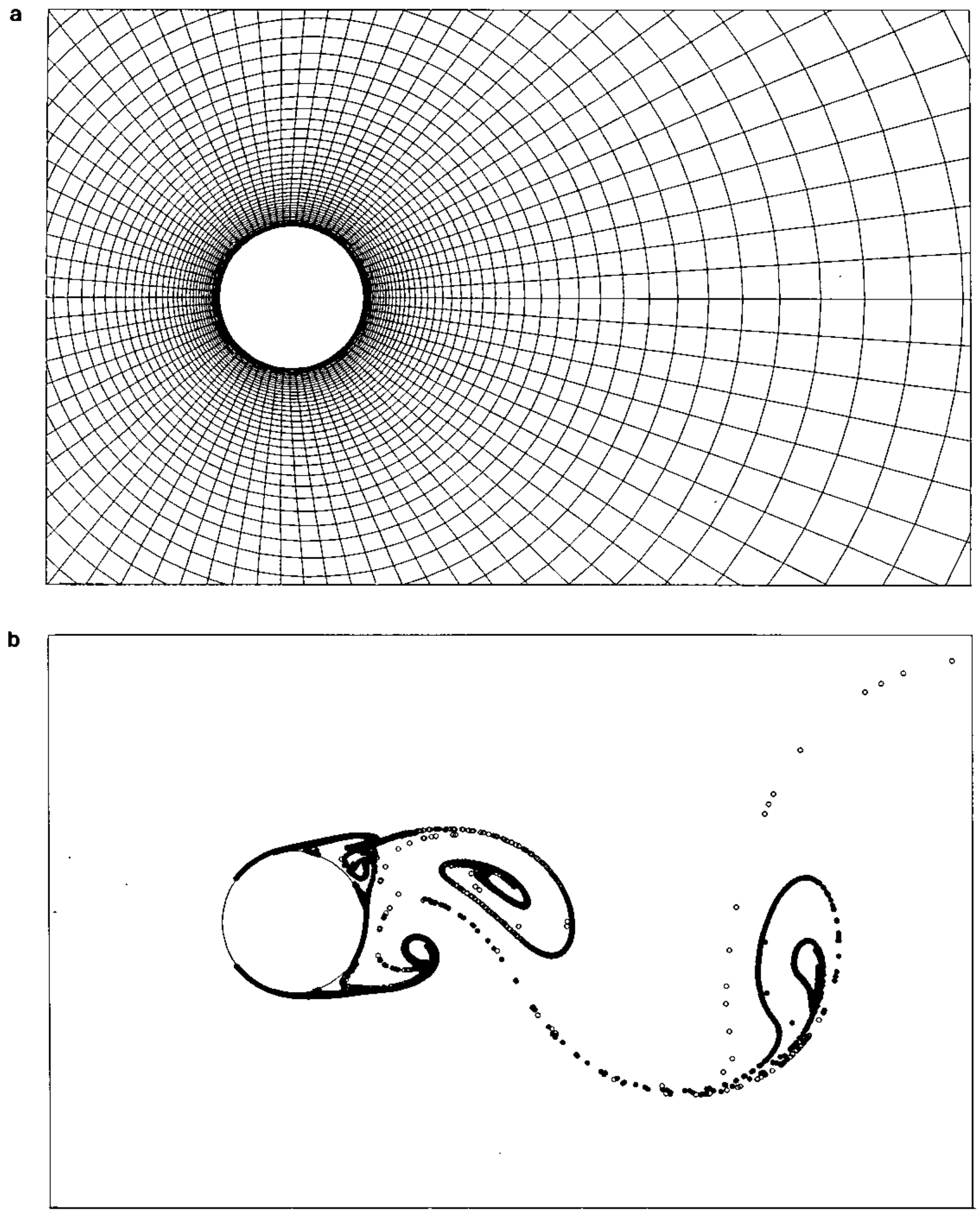

FIG. 9. (a) Grid system for the computation of flow past a two-dimensional circular cylinder. The number of grid points is 79 × 80. (b) Pattern of particle traces (computed streaklines) at Re = 1000.

was included in the flow solver, is listed in the Algorithm section of this paper. The arrays *X*, *Y* are the values of grid points; the arrays AT1 and AT2 denote $\mathbf{a}^1$ and $\mathbf{a}^2$, respectively; and the arrays *U*, *V* and *UN*, *VN* are the velocity values at the current and new step. These arrays are set in the flow solver. In the case that grid data and the time series of velocity data are stored in a file, you can utilize this program if you calculate the arrays AT1 and AT2. First the velocity field is transformed to the computational space in Subroutine TRNSV. The new generated particles are released in Subroutine GENEP. Then, new positions of the particles are computed in Subroutine INTGP. The position of particles in the computational space is stored in the arrays PK and PL. Finally, the particle positions are transformed to the physical space and are plotted on the graphic display in Subroutine PLOTP.

Algorithm

```
C----- KMAX, LMAX are the number of grid points.
C----- DS is the time-increment.
C----- NP is the number of particles; NP=0 at first time step.
      SUBROUTINE PATH2D(KMAX, LMAX, DS, NP, X, Y, AT1, AT2, U, V, UN, VN)
      DIMENSION X(79, 80), Y(79, 80), AT1(2, 79, 80), AT2(2, 79, 80)
      DIMENSION U(79, 80), V(79, 80), UN(79, 80), VN(79, 80)
      DIMENSION PK(10000), PL(10000), IPJUMP(10000)
      CALL TRNSV(KMAX, LMAX, AT1, AT2, U, V)
      CALL TRNSV(KMAX, LMAX, AT1, AT2, UN, VN)
      CALL GENEP(KMAX, LMAX, NP, PK, PL, IPJUMP)
      CALL INTGP(KMAX, LMAX, NP, DS, U, V, UN, VN, PK, PL, IPJUMP)
      DO 10 N=1, NP
        IF(IPJUMP(N).EQ.0) CALL PLOTP(PK(N), PL(N), X, Y)
   10 CONTINUE
      RETURN
      END
C
      SUBROUTINE TRNSV(KMAX, LMAX, AT1, AT2, U, V)
      DIMENSION AT1(2, 79, 80), AT2(2, 79, 80), U(79, 80), V(79, 80)
      DO 10 L=1, LMAX
      DO 10 K=1, KMAX
        UC=AT1(1, K, L)*U(K, L)+AT1(2, K, L)*V(K, L)
        VC=AT2(1, K, L)*U(K, L)+AT2(2, K, L)*V(K, L)
        U(K, L)=UC
        V(K, L)=VC
   10 CONTINUE
      RETURN
      END
C
      SUBROUTINE GENEP(KMAX, LMAX, NP, PK, PL, IPJUMP)
      DIMENSION PK(10000), PL(10000), IPJUMP(10000)
      IF(NP.GT.9992) RETURN
      DO 10 L=1, 9
      IF(L.NE.5) THEN
        NP=NP+1
        PK(NP)=2.
        PL(NP)=8.*FLOAT(L)
        IPJUMP(NP)=0
      END IF
   10 CONTINUE
      RETURN
      END
C
      SUBROUTINE INTGP(KMAX, LMAX, NP, DS, UC, VC, UCN, VCN, PK, PL, IPJUMP)
      DIMENSION UC(79, 80), VC(79, 80), UCN(79, 80), VCN(79, 80)
      DIMENSION PK(10000), PL(10000), IPJUMP(10000)
      DS2=DS*.5
      DO 10 N=1, NP
      IF(IPJUMP(N).EQ.0) THEN
        PKW=PK(N)
        PLW=PL(N)
        CALL CALVP(PKW, PLW, UC, VC, PUW, PVW)
        PK(N)=PK(N)+DS*PUW
        PL(N)=PL(N)+DS*PVW
        CALL JDGEP(KMAX, LMAX, PK(N), PL(N), IJDGE)
        IF(IJDGE.EQ.0) THEN
          CALL CALVP(PK(N), PL(N), UCN, VCN, PU, PV)
          PK(N)=PKW+DS2*(PUW+PU)
          PL(N)=PLW+DS2*(PVW+PV)
          CALL JDGEP(KMAX, LMAX, PK(N), PL(N), IJDGE)
        END IF
        IPJUMP(N)=IJDGE
      END IF
   10 CONTINUE
      RETURN
      END

C
      SUBROUTINE CALVP(POK, POL, UC, VC, PU, PV)
      DIMENSION UC(79, 80), VC(79, 80)
      K=INT(POK)
      L=INT(POL)
      DK=POK-FLOAT(K)
      DL=POL-FLOAT(L)
      A1=(1.-DK)*(1.-DL)
      A2=DK*(1.-DL)
      A3=DK*DL
      A4=(1.-DK)*DL
      PU=A1*UC(K, L)+A2*UC(K+1, L)+A3*UC(K+1, L+1)+A4*UC(K, L+1)
      PV=A1*VC(K, L)+A2*VC(K+1, L)+A3*VC(K+1, L+1)+A4*VC(K, L+1)
      RETURN
      END
C
      SUBROUTINE JDGEP(KMAX, LMAX, POK, POL, IJDGE)
        L=INT(POL)
        IF(L.GE.LMAX-1) POL=POL-FLOAT(LMAX-2)
        IF(L.LT.1) POL=POL+FLOAT(LMAX-2)
      IJDGE=0
      K=INT(POK)
      KHAN=(K-1)*(KMAX-1-K)
      IF(KHAN.LT.0) IJDGE=1
      RETURN
      END
C
      SUBROUTINE PLOTP(POK, POL, X, Y)
      DIMENSION X(79, 80), Y(79, 80)
      K=INT(POK)
      L=INT(POL)
      DK=POK-FLOAT(K)
      DL=POL-FLOAT(L)
      A1=(1.-DK)*(1.-DL)
      A2=DK*(1.-DL)
      A3=DK*DL
      A4=(1.-DK)*DL
      GX=A1*X(K, L)+A2*X(K+1, L)+A3*X(K+1, L+1)+A4*X(K, L+1)
      GY=A1*Y(K, L)+A2*Y(K+1, L)+A3*Y(K+1, L+1)+A4*Y(K, L+1)
      CALL DRAW(GX, GY)
      RETURN
      END
```

REFERENCES

1. L. Hesselink, *Annu. Rev. Fluid Mech.* **20**, 421 (1988).
2. S. Shirayama and K. Kuwahara, *Int. J. Supercomput. Appl.* **4.2**, 66 (1990).
3. P. G. Buning and J. L. Steger, in *Proceedings, AIAA 7th Computational Fluid Dynamics Conference, Cincinnati, Ohio, June 1985*, p. 162.
4. S. E. Rogers, P. G. Buning, and F. J. Merritt, *Int. J. Supercomput. Appl.* **1.4**, 96 (1987).
5. E. M. Murman and K. G. Powell, *AIAA J.* **27**, 982 (1988).
6. M. H. Smith, W. R. Van Dalsen, F. C. Dougherty, and P. G. Buning, AIAA Paper 89-0139, Jan. 1989.
7. G. Volpe, AIAA Paper 89-0140, Jan. 1989.
8. R. Löhner, P. Parikh, and C. Gumbert, in *Proceedings, AIAA 9th Computational Fluid Dynamics Conference, Buffalo, NY, June 1989*, p. 495.
9. T. Dombre, U. Frisch, J. M. Greene, M. Hénon, A. Mehr, and A. M. Soward, *J. Fluid Mech.* **167**, 353 (1986).
10. A. E. Perry and M. S. Chong, *Annu. Rev. Fluid Mech.* **19**, 125 (1987).
11. U. Dallmann, *Fluid Dyn. Res.* **3**, 183 (1988).
12. M. E. M. de Winkel and P. G. Bakker, Delft University of Technology Report LR-541, March 1988.
13. J. Guckerheimer and Ph. Holmes, *Non-linear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (Springer-Verlag, New York, 1983).
14. M. S. Milgram, *J. Comput. Phys.* **84**, 134 (1989).
15. J. L. Helman and L. Hesselink, *IEEE Computer Graphics and Applications, May 1991*, p. 36.