JOURNAL OF COMPUTATIONAL PHYSICS 89, 161–186 (1990)

A Fast Adaptive Vortex Method for Patches of Constant Vorticity in Two Dimensions

THOMAS F. BUTTKE

Program in Applied and Computational Mathematics, Princeton University, Princeton, New Jersey 08544

Received July 11, 1988; revised July 19, 1989

We present a fast numerical method for solving the incompressible Euler's equation in two dimensions for the special case when the flow field can be represented by patches of constant vorticity. The method is an adaptive vortex method in which cells (vortex blobs) of multiple scales are used to represent the patches so that the number of vortex blobs needed to approximate the patches is proportional to the length of the boundary curve of the patch and inversely proportional to the width of the smallest blob (cell) used. Points along the boundaries of the patches are advected according to the velocity obtained from the approximating vortices.

INTRODUCTION

Until the development of this method there have been two basic methods for evolving patches of constant vorticity in two dimensions. One method is contour dynamics [1–8]. In contour dynamics the velocity of any point in the fluid can be determined by integrating an appropriate kernel along the boundaries of the patches. In the numerical method one tracks points along the boundary of the patches and then advects these boundary points according to the velocity obtained by approximating the boundary integral by these same boundary points. Contour dynamics works extremely well if the contours are relatively simple; for smooth contours high order accuracy can be obtained [3–8]. However, even for smooth contours with a small maximum curvature there can be significant loss of accuracy in the evaluation of the boundary integral if the spacing of the points approximating the curve is much greater than the distance between two closely spaced parts of a contour [8]. When the contours form regions of high curvature or singularities high order methods lose their high order accuracy unless one knows the location of the singularity a priori and it becomes more difficult to accurately evaluate the boundary integral necessary to obtain the velocity of a point along the boundary.

The second method is vortex dynamics [9–14]. This method is designed to calculate the dynamics of arbitrary configurations of vorticity in two or three

0021-9991/90 $3.00

dimensions and thus much redundant work is done if the method is applied directly to the problem of patches of constant vorticity. In this method a uniform grid is drawn initially and each grid point inside of the patch is advected according to the velocity obtained by adding the contributions of each vortex element in the patch. The redundant work is the work done to advect the points which are interior to the contour. The high order accuracy of vortex dynamics cannot be obtained for the case of patches of constant vorticity since the vorticity distribution is not smooth. For the case of patches of constant vorticity the error in the vortex method is dominated by the error in approximating the initial vorticity distributions; this error occurs in approximating the vorticity near the boundary of the patches. Thus we can effectively increase the accuracy of the method by increasing the resolution of the method near the boundary of the patch.

The method which we introduce here uses features of both methods in order to develop an efficient and robust algorithm for evolving patches in two dimensions. We advect the points on the boundary contours of the patches; however, we calculate the velocity at a given point by summing the contributions of the vortex elements inside of the patch. We use vortices of different sizes in order to approximate the vortex patches in an optimal way. When using general vortex methods the number of vortex elements required to approximate an area is proportional to $1/\Delta\xi^2$, where $\Delta\xi$ is the mesh spacing. In our adaptive method we are able to approximate a patch with the number of vortex elements required being proportional to $1/\Delta\xi$. This allows us to reduce the error by increasing the effective number of vortex elements approximating the patch.

In the first section of the paper we will introduce the basic mathematics of the vortex method. This section includes the method of evaluating the velocity for a given vortex element. In the second section we describe the method used to approximate the area of the patch with the vortex elements; in the third section we present the numerical method; in the fourth section we discuss the error of the method; and in the fifth section we give an example which shows some of the complicated generic structures which appear in the patches.

1. The Basic Vortex Method

Euler's equation in two dimensions is [15]

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{\nabla P}{\rho}, \tag{1.1}$$

where $\mathbf{u}(x, y, t)$ is the velocity of the fluid, t is time, P is pressure, and ρ is the density of the fluid. We consider the case of incompressible, $(\nabla\cdot\mathbf{u}=0)$, and isentropic flow ($\nabla P/\rho = \nabla w$, for some w). We define the vorticity ω as

$$\omega \equiv \frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}; \tag{1.2}$$

and combine this definition with Eq. (1.1) to find

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega = 0. \tag{1.3}$$

Equation (1.3) implies that the vorticity is advected with the fluid velocity $\mathbf{u}$ with the magnitude of the vorticity remaining constant as we follow a given fluid particle.

We now derive an expression for the velocity $\mathbf{u}$ in terms of the vorticity. Consider a function $\phi(x, y)$, called the stream function, with continuous second derivatives; if we define the velocity $\mathbf{u}$ by

$$(u_x, u_y) = \left(-\frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial x} \right), \tag{1.4}$$

where the subscripts on u indicate the cartesian components of $\mathbf{u}$, the incompressibility condition $\nabla \cdot \mathbf{u} = 0$ is automatically satisfied by the equality of the mixed partials of $\phi(x, y)$. From the definition of vorticity equation (1.2) we immediately obtain

$$\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \Delta \phi = \omega. \tag{1.5}$$

Thus by solving Poisson's equation we can determine the velocity at any point in space if we know the vorticity.

The general vortex method in two dimensions is based on the fact that one is able to write the fluid velocity in terms of the vorticity ω and the fact that the vorticity is advected with the fluid velocity without a change in magnitude. In the general vortex method the vorticity is approximated by evaluating the initial vorticity at the points of a mesh of spacing $\Delta\xi$. For later times these vortex points are then advected according to a velocity obtained by finding an approximate solution of Eq. (1.5). The stream function ϕ is approximated by Φ, where

$$\Phi(\mathbf{r}, t) \equiv (\Delta\xi/\delta)^2 \sum_{i=1}^{N} \omega_i \phi_\delta(\mathbf{r} - \mathbf{r}_i(t)); \tag{1.6}$$

ω_i is the initial vorticity evaluated at the ith mesh point, N is the number of points approximating the vorticity, $\mathbf{r} = (x, y)$, and $\mathbf{r}_i$ is the position of the ith mesh point at time t. $\phi_\delta(\mathbf{r})$ is defined by the equation

$$\Delta \phi_\delta = f(\mathbf{r}/\delta), \tag{1.7}$$

where $f(\mathbf{r})$ is a smooth function satisfying $\int f\, dx\, dy = 1$ and $\int x^i y^j f\, dx\, dy = 0$ for all positive integers i and j such that $1 \leqslant i + j \leqslant p - 1$ for some integer p. The general vortex method has been shown to converge to the exact answer for smooth vorticity distributions with an error proportional to $(\Delta\xi)^p$ for arbitrary p provided that

$\delta = (\Delta\xi)^\alpha$ for $\alpha < 1$ [10–12]. This high order accuracy cannot be exploited for patches of constant vorticity since the error is dominated by the fact that the discontinuous vorticity distribution is not approximated accurately initially.

The method we design differs from the general vortex method because we are considering vorticity distributions which are constant inside of a region Ω_0 initially and therefore discontinuous. From Eq. (1.3) we see that the magnitude of the vorticity is unchanged as the region Ω_0 evolves into some other region $\Omega(t)$ at time t. Since the vorticity is constant we know the vorticity at evey point in the interior of $\Omega(t)$. Thus if we advect only points on the boundary of the region $\Omega(t)$ we will have fewer points to advect than if we were to advect every point in a grid in the interior of Ω_0 and we will have no loss of information, since we know that the vorticity inside of the region is constant.

From Eq. (1.6) and (1.7) we note that the approximate stream function Φ is the exact stream function for the vorticity distribution given by $\omega(\mathbf{r}) = (\Delta\xi/\delta)^2 \sum_{i=1}^{N} \omega_i f((\mathbf{r} - \mathbf{r}_i)/\delta)$. Thus if we pick $\Delta\xi = \delta$ and $f(\mathbf{r}) = f_0(\mathbf{r})$ such that

$$f_0(x, y) = \begin{cases} 1 & \text{if} \quad |x| \leqslant 1/2 \quad \text{and} \quad |y| \leqslant 1/2 \\ 0 & \text{if} \quad |x| \geqslant 1/2 \quad \text{or} \quad |y| \geqslant 1/2 \end{cases} \tag{1.8}$$

the only error in the calculated velocity field will come from grid points which lie along the boundary of $\Omega(t)$. If we were to advect the grid points as in the general vortex method at later times the grid points would no longer form a rectangular grid and our approximation to the true vorticity distribution would no longer represent the constant distribution. Thus at each time rather than advect all of the grid points independently we simply advect the boundary curve and at each time we determine the grid points which are inside of the boundary of $\Omega(t)$. We then use these grid points along with the f given in Eq. (1.8) to calculate Φ at the given time. We will discuss the method of choosing the grid points in more detail in the next section.

We now proceed to find an explicit formula for ϕ_δ as defined in Eq. (1.7) with $f = f_0$ as given in Eq. (1.8). We find ϕ_δ for the case $\delta = 2$; the general case can then be found by scaling: $\phi_\delta(\mathbf{r}) = (\delta/2)^2 \phi_2(2\mathbf{r}/\delta)$. We note that for the constant vorticity case under consideration a closed formula in terms of elementary functions may be obtained by explicitly integrating the logarithmic kernel over the square [1]. We are interested in eventually generalizing this technique so that it is applicable to more general flows in which the vorticity is not constant and so we give a method of evaluating the stream function which may be more easily generalized.

The Green function $G(z, z')$ for the Laplacian in two dimensions is $G(z, z') = (1/2\pi) \log|z - z'|$, where $z \equiv x + iy$. For $|z'| < |z|$ we can expand $G(z, z')$ in a Laurent series. We find that

$$G(z, z') = \frac{1}{2\pi} \left\{ \log(z) - \sum_{n=1}^{\infty} \frac{1}{n} \frac{z'^n}{z^n} \right\}. \tag{1.9}$$

When using Eq. (1.9) we must remember to take the real part of the right-hand side of the expression. Since $\phi_2(\mathbf{r}) = \int G(z, z') f_0(\mathbf{r}'/2)\, dx'\, dy'$ and since

$$\int_{-1}^{1}\int_{-1}^{1} z^n\, dx\, dy = \begin{cases} \dfrac{8(-4)^m}{(n+1)(n+2)} & \text{if} \quad n = 4m \\ 0 & \text{if} \quad n \neq 4m, \end{cases} \tag{1.10}$$

we have for $|z| \geq r_0 = \sqrt{2}$ that

$$\phi_2(\mathbf{r}) = \frac{2}{\pi}\log(z) - \frac{1}{2\pi}\sum_{m=1}^{\infty} \frac{8(-4)^m}{4m(4m+1)(4m+2)}\frac{1}{z^{4m}}, \tag{1.11}$$

where we mean to take the real part of the right-hand side of Eq. (1.11).

In order to find $\phi_2(\mathbf{r})$ for $r < \sqrt{2}$ ($r = |\mathbf{r}| = |z|$) we break ϕ_2 into two parts such that $\phi_2 = \phi^p + \phi^h$, where ϕ^p is a particular solution of Eq. (1.7) and ϕ^h is a harmonic function which is determined by the boundary condition that $\phi_2(\mathbf{r})$ be continuous at $r = \sqrt{2}$. We choose $\phi^p(\mathbf{r}) = r^2/4$ in the region R_0 defined as $R_0 = ((x, y): |x| < 1$ and $|y| < 1)$. Define the regions R_{I}, R_{II}, R_{III}, and R_{IV} as $R_{\mathrm{I}} = ((x, y): x > 1$ and $r < \sqrt{2})$; $R_{\mathrm{II}} = ((x, y): y > 1$ and $r < \sqrt{2})$; $R_{\mathrm{III}} = ((x, y): x < -1$ and $r < \sqrt{2})$; and $R_{\mathrm{IV}} = ((x, y): y < -1$ and $r < \sqrt{2})$. We find ϕ^p in the regions R_{I}, R_{II}, R_{III}, and R_{IV} by requiring that ϕ^p have continuous first derivatives that ϕ^p be harmonic everywhere except in R_0. We find that

$$\begin{aligned}
\phi^p(\mathbf{r}) &= \frac{r^2}{4} && \text{for} \quad (x, y) \in R_0, \\
\phi^p(\mathbf{r}) &= -\frac{1}{2} + z - \frac{z^2}{4} && \text{for} \quad (x, y) \in R_{\mathrm{I}}, \\
\phi^p(\mathbf{r}) &= -\frac{1}{2} - iz + \frac{z^2}{4} && \text{for} \quad (x, y) \in R_{\mathrm{II}}, \\
\phi^p(\mathbf{r}) &= -\frac{1}{2} - z - \frac{z^2}{4} && \text{for} \quad (x, y) \in R_{\mathrm{III}}, \\
\phi^p(\mathbf{r}) &= -\frac{1}{2} + iz + \frac{z^2}{4} && \text{for} \quad (x, y) \in R_{\mathrm{IV}},
\end{aligned} \tag{1.12}$$

where once again we must remember to take the real parts of the right-hand side in Eq. (1.12).

We now find ϕ^h. Since ϕ^h is harmonic we write $\phi^h = \sum_{n=0}^{\infty} a_n z^n$, where a_n is complex and we mean to take only the real part of the sum. We find the a_n by making $\phi_2(\mathbf{r})$ continuous at $r = \sqrt{2}$. We expand $\phi^p(\mathbf{r})$ on the circle $r = r_0 = \sqrt{2}$ in

a harmonic series; $\phi^p = \sum_{n=0}^{\infty} b_n e^{in\theta}$. Only the positive modes are considered since we are concerned with only the real part of the series. We find that $b_n = 0$ unless $n = 4m$ and that

$$b_{4m} = \begin{cases} \dfrac{-8r_0(-1)^m \sin(\pi/4)}{\pi(16m^2-1)} + \dfrac{4r_0^2(-1)^m}{\pi(16m^2-4)} & \text{for} \quad m > 0 \\ \dfrac{-1}{2} + \dfrac{4r_0 \sin(\pi/4)}{\pi} - \dfrac{r_0^2}{2\pi} & \text{for} \quad m = 0. \end{cases}$$

Substituting $r_0 = \sqrt{2}$ and simplifying the expression for b_{4m}, we obtain

$$b_{4m} = \begin{cases} \dfrac{24(-1)^m}{\pi(16m^2-1)(16m^2-4)} & \text{for} \quad m > 0 \\ \dfrac{3}{\pi} - \dfrac{1}{2} & \text{for} \quad m = 0. \end{cases} \tag{1.13}$$

From Eq. (1.11) we obtain $\phi_2 = \sum_{m=0}^{\infty} c_{4m} e^{-i4m\theta}$ at $r = r_0$, where

$$c_{4m} = \begin{cases} \dfrac{(-1)^{m+1}}{\pi m(4m+1)(4m+2)} & \text{for} \quad m > 0 \\ \dfrac{2}{\pi} \log(r_0) & \text{for} \quad m = 0. \end{cases} \tag{1.14}$$

We require that $\phi^h + \phi^p = \phi_2$ at $r = r_0$ and obtain that $a_{4m} = (c_{4m} - b_{4m})/r_0^{4m}$, where in obtaining the expression for a_{4m} we have equated only the real parts of the complex series. Thus we find that

$$\phi^h(\mathbf{r}) = \sum_{m=0}^{\infty} (c_{4m} - b_{4m}) \frac{z^{4m}}{r_0^{4m}}, \tag{1.15}$$

where b_{4m} and c_{4m} are defined in Eq. (1.13) and Eq. (1.14) and $r_0 = \sqrt{2}$.

Since we are interested in the velocity field which is given in terms of the derivatives of $\phi_\delta(\mathbf{r})$, we write down the explicit formulae for the derivatives of $\phi_2(\mathbf{r})$. It is convenient to define a complex function ψ_δ as $\psi(\mathbf{r}) \equiv \phi_x - i\phi_y$. We see then that in terms of ψ the velocity $\mathbf{u}$ is given by $u_x = \mathrm{Im}(\psi)$ and $u_y = \mathrm{Re}(\psi)$, where Re and Im denote that the real and imaginary parts of the expression are to be taken. If we recall the Cauchy–Riemann equations which relate the real and imaginary parts of the derivative of a complex function, we see that ψ_2 may be obtained in the regions, where ϕ_2 is harmonic by formally differentiating the expressions for ϕ_2 with respect to z. Thus we find that

$$\psi_2^p(\mathbf{r}) = \frac{z^*}{2} \qquad \text{for} \quad (x, y) \in R_0,$$

$$\psi_2^p(\mathbf{r}) = 1 - \frac{z}{2} \qquad \text{for} \quad (x, y) \in R_{\mathrm{I}},$$

$$\psi_2^p(\mathbf{r}) = -i + \frac{z}{2} \qquad \text{for} \quad (x, y) \in R_{\mathrm{II}},$$

$$\psi_2^p(\mathbf{r}) = -1 - \frac{z}{2} \qquad \text{for} \quad (x, y) \in R_{\mathrm{III}}, \tag{1.16}$$

$$\psi_2^p(\mathbf{r}) = i + \frac{z}{2} \qquad \text{for} \quad (x, y) \in R_{\mathrm{IV}},$$

$$\psi_2^h(\mathbf{r}) = \sum_{m=1}^{\infty} 4m(c_{4m} - b_{4m}) \frac{z^{4m-1}}{r_0^{4m}} \qquad \text{for} \quad r \leqslant r_0,$$

$$\psi_2(\mathbf{r}) = \psi_2^p(\mathbf{r}) + \psi_2^h(\mathbf{r}) \qquad \text{for} \quad r \leqslant r_0,$$

and

$$\psi_2(\mathbf{r}) = \frac{2}{\pi z} - \sum_{m=1}^{\infty} 4mc_{4m} \frac{r_0^{4m}}{z^{4m+1}}, \qquad \text{for} \quad r \geqslant r_0,$$

where z^* is the complex conjugate of z, $r_0 = \sqrt{2}$, and b_{4m} and c_{4m} are defined in Eqs. (1.13) and (1.14). We note that

$$\psi_\delta(\mathbf{r}) = (\delta/2)\, \psi_2(2\mathbf{r}/\delta). \tag{1.17}$$

We now discuss the efficient evaluation of the series in Eq. (1.16). The series in Eq. (1.16) define ψ_2 for all values of r and the series converge absolutely for all allowed r. The number of terms N required to evaluate ψ_2 to a given accuracy ε depends on the ratio $R \equiv \min(r/r_0, r_0/r)$; we find that $N \approx \log(\varepsilon(1-R))/(4 \log R)$. If we choose $\varepsilon = 10^{-16}$, we find that for $R = \frac{1}{2}$, $N \approx 14$; for $R = 0.9$, $N \approx 93$; for $R = 0.95$, $N \approx 194$; and for $R = 0.99$, $N \approx 1031$. Thus we see that as $R \to 1$ a large number of terms are required to evaluate ψ_2. For $R = 1$, 10^8 terms are required to evaluate ψ_2 to an accuracy of $\varepsilon = 10^{-16}$.

We can greatly reduce the number of terms required to evaluate ψ_2 for $R \to 1$ if we note that by superposition $\psi_{2\delta}$ can be written in terms of ψ_δ:

$$\begin{aligned}\psi_{2\delta}(\mathbf{r}) = {} & \psi_\delta(\mathbf{r} - (\delta/2)\, \mathbf{r}_{1,1}) + \psi_\delta(\mathbf{r} - (\delta/2)\, \mathbf{r}_{-1,1}) \\ & + \psi_\delta(\mathbf{r} - (\delta/2)\, \mathbf{r}_{1,-1}) + \psi_\delta(\mathbf{r} - (\delta/2)\, \mathbf{r}_{-1,-1}),\end{aligned} \tag{1.18}$$

where $\mathbf{r}_{1,1} = (1, 1)$, $\mathbf{r}_{-1,1} = (-1, 1)$, $\mathbf{r}_{1,-1} = (1, -1)$, and $\mathbf{r}_{-1,-1} = (-1, -1)$.

As a simple example of the usefulness of Eq. (1.18), we use it to express $\psi_2(1, 1)$ in terms of values of ψ_2 which are easy to evaluate. From Eq. (1.18) we have

$$\psi_2(1, 1) = \psi_1(1/2, 1/2) + \psi_1(3/2, 1/2) + \psi_1(1/2, 3/2) + \psi_1(3/2, 3/2).$$

We then use Eq. (1.17) to rewrite ψ_1 in terms of ψ_2 and simplify the resultant expression to obtain

$$\psi_2(1, 1) = \psi_2(3, 1) + \psi_2(1, 3) + \psi_2(3, 3). \tag{1.19}$$

Similar expressions can be obtained for $\psi_2(1, -1)$, $\psi_2(-1, 1)$, and $\psi_2(-1, -1)$. If we were to calculate $\psi_2(1, 1)$ to an accuracy of 10^{-16} directly using Eq. (1.16), 10^8 terms would be required; whereas using Eq. (1.19) only 33 terms are required for the same accuracy.

A similar procedure can be used for evaluating ψ_2 near the corner points $(\pm 1, \pm 1)$. Consider the evaluation of $\psi_2(\mathbf{r})$ with R corresponding to $\mathbf{r}$ as defined earlier. We pick an R_0 close to 1 such that if $R \leqslant R_0$ we evaluate ψ_2 by Eq. (1.16) and if $R > R_0$ we use Eqs. (1.18) and (1.17) to rewrite $\psi_2(\mathbf{r})$ in terms of $\psi_2(\mathbf{r}_i)$. We calculate R_i corresponding to $\mathbf{r}_i$ and for any i_0 such that $R_{i_0} > R_0$ we rewrite $\psi_2(\mathbf{r}_{i_0})$ using Eqs. (1.18) and (1.17). We continue this process until we have a sum of the form

$$\psi_2(\mathbf{r}) = \sum_{j=1}^{M} \alpha_j \psi_2(\mathbf{r}_j) \tag{1.20}$$

with the condition on R_j and α_j, that given $\varepsilon > 0$,

$$\text{either } R_j \leqslant R_0 \text{ or } |\alpha_j| < \varepsilon \text{ for all } j,$$

where, as before, $R_j \equiv \min(r_j/r_0, r_0/r_j)$. The process is illustrated in Fig. 1 for $R_0 = 0.95$. The process can be carried out for any R_0, but for small R_0 the process is more complicated than for the case when R_0 is close to 1. Furthermore, for small R_0 there is no computational advantage for most of the points $\mathbf{r}$. The algorithm we illustrate will work provided $R_0 \geqslant 0.94884\ldots$. This minimum value of R_0 is determined by the condition that the shaded region in Fig. 1 must remain in the region $((x, y): x > \frac{1}{2}$ and $y < -\frac{1}{2})$ so that the point at which we are evaluating ψ will always be close to the same corner at each level of refinement. By picking $R_0 \geqslant 0.95$ we can guarantee that at each step in the process if we must rewrite one of the $\psi_2(\mathbf{r}_{i_0})$ it will only be the one corresponding to the box closest to the corner point (see Fig. 1). We write down the explicit formula corresponding to Eq. (1.20) for the case illustrated in Fig. 1, where $\mathbf{r}$ is assumed to be close to the corner point $(1, -1)$. We have for $x > 0$ and $y < 0$, where $\mathbf{r} = (x, y)$, that

$$\psi_2(\mathbf{r}) = \sum_{j=1}^{M} \left(\frac{1}{2^j} \sum_{k=1}^{3} \psi_2(2^j(\mathbf{r} - \mathbf{r}_k^j)) \right) + \frac{1}{2^M} \psi_2(2^M(\mathbf{r} - r_0^M)), \tag{1.21}$$

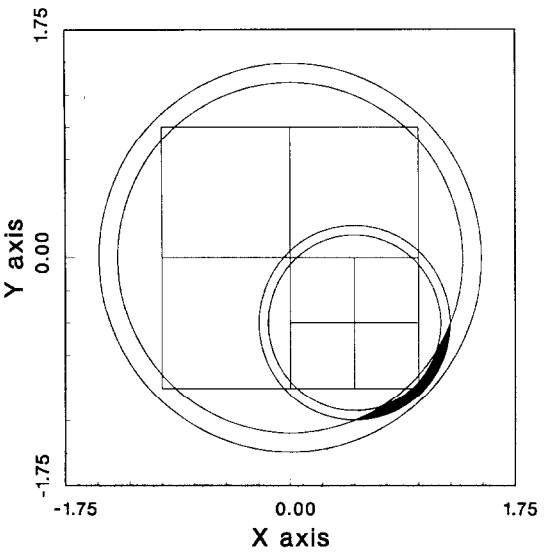


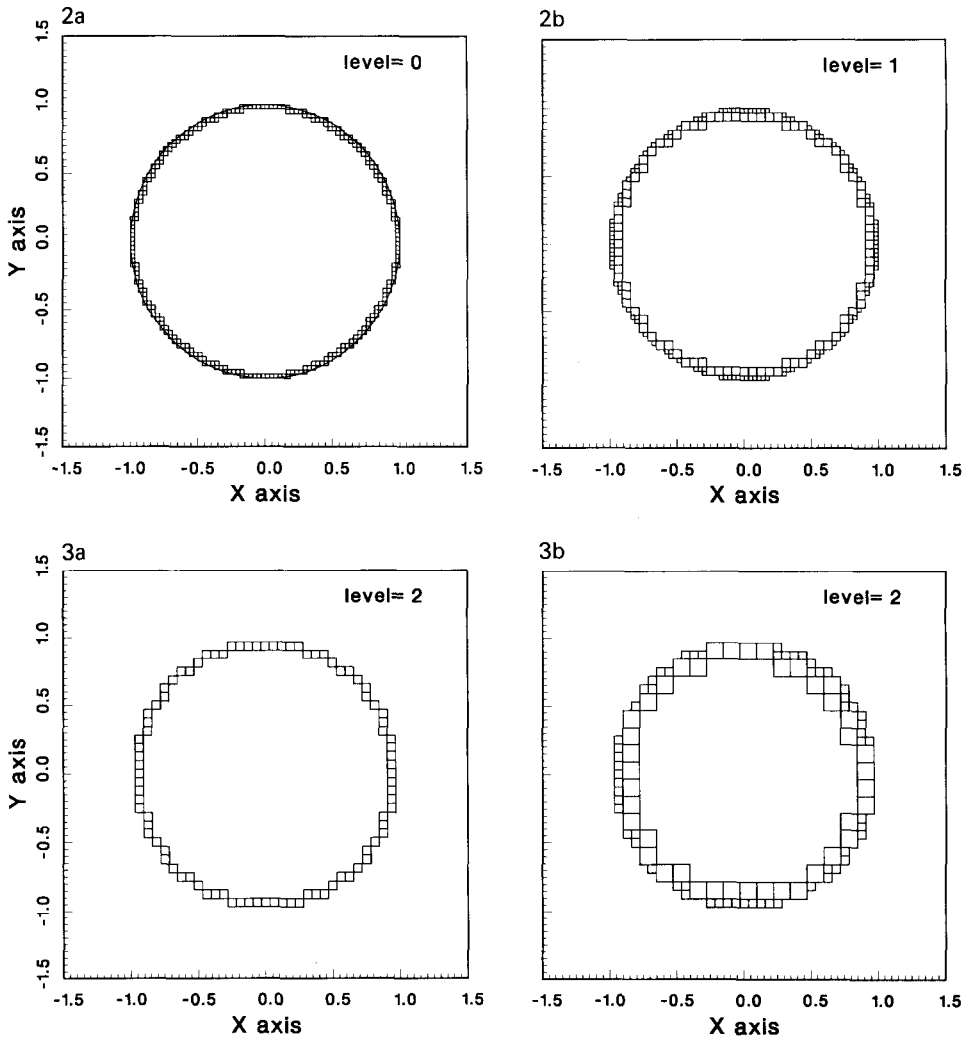
FIG. 1. The self-similar method of evaluating the function ψ_2. The points which lie in the annular regions shown in Fig. 1 correspond to points which have $R \geq 0.95$. The shaded region shows points which could not be evaluated after two levels of subdividing the basic solution.

where $\mathbf{r}_0^j = (1-(\frac{1}{2})^j)(1,-1)$, $\mathbf{r}_1^j = \mathbf{r}_0^j + (\frac{1}{2})^{j-1}(-1,0)$, $\mathbf{r}_2^j = \mathbf{r}_0^j + (\frac{1}{2})^{j-1}(-1,1)$, and $\mathbf{r}_3^j = \mathbf{r}_0^j + (\frac{1}{2})^{j-1}(0,1)$. Similar expressions can be written for $\mathbf{r}$ which lie in the other quadrants.

The number of subdivisions M required in Eq. (1.21) is determined as follows. Define $r_k \equiv |2^k(\mathbf{r}-\mathbf{r}_0^k)|$ for $k=0, 1, 2, \ldots$, where $\mathbf{r}_0^k$ is defined in Eq. (1.21) and define $R_k = \min(r_0/r_k, r_k/r_0)$. M is chosen to be the smallest integer m such that $R_m \leq R_0$ or such that $(\frac{1}{2})^m < \varepsilon$. As can be seen in Fig. 1 for most points $\mathbf{r}$ this will occur for $M \leq 2$. The worst case (the case requiring the most computational work) will occur when $|\mathbf{r}-(1,-1)| = \varepsilon \approx 0$. The number of terms N required to evaluate $\psi_2(\mathbf{r})$ for this case is $N \approx -33 \log_2 \varepsilon$. (About 33 terms are required to evaluate the three values of ψ_2 at each level.) For $\varepsilon = 10^{-16}$ we have $-\log_2 \varepsilon \approx 53$ and thus the number of terms required to evaluate $\mathbf{r}$ in this case is $33 \cdot 53 = 1749$; of course, for the corner point the exact value is given more efficiently by Eq. (1.19). This method requires orders of magnitude fewer operations to evaluate ψ_2 than if Eq. (1.16) were used directly.

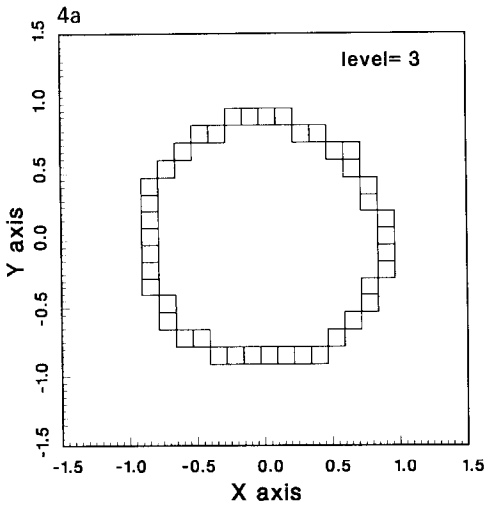
2. APPROXIMATION OF THE PATCHES

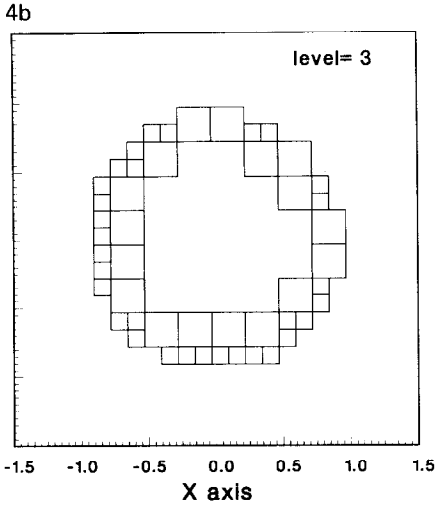
We now describe the algorithm for determining the grid cells which should be chosen to approximate a given patch of constant vorticity. We consider the xy-plane to be covered by families of square cells of size $(\frac{1}{2})^m$, $m = \ldots -2, -1, 0, 1, 2, \ldots$. The xy-plane is covered by placing a cell with a corner coinciding with

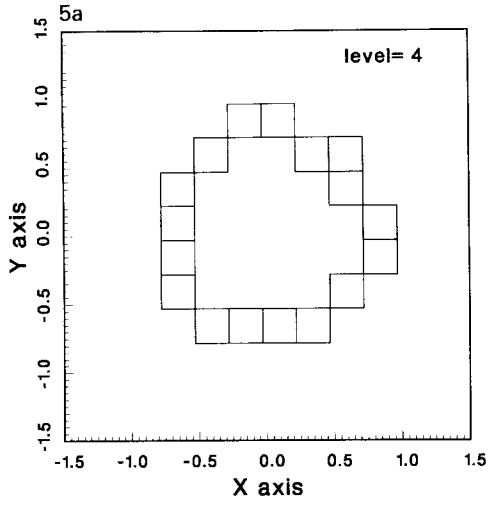


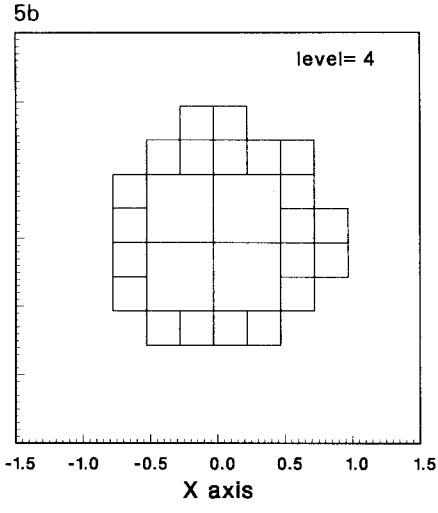
FIGS. 2–6. Approximating a patch with cells of increasing size. The steps are shown which are required to approximate a patch with cells of increasing size. At each step the size of cell used to approximate the patch is increased by a factor of two.

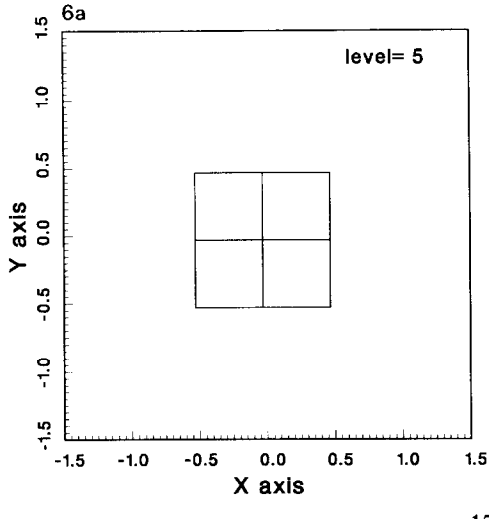
the origin and the sides of the cell chosen parallel to the x-axis and y-axis. Given a grid of size $\Delta\xi$ and a closed curve C we want to determine all of the grid cells which lie inside of the curve C. We consider a grid cell to lie inside of C if the center of the cell lies inside of C. The curve C is approximated by N points $\mathbf{r}_j$ which are distributed along C. We use linear interpolation to approximate the curve between the given points $\mathbf{r}_j$. We assume the points are listed in order so that the curve is traversed in the right-hand sense as j goes from 1 to N.

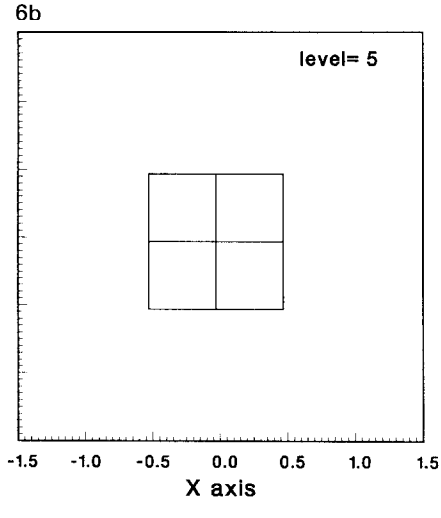

4a
level= 3
Y axis
X axis
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5


4b
level= 3
X axis
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5


5a
level= 4
Y axis
X axis
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5


5b
level= 4
X axis
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5


6a
level= 5
Y axis
X axis
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5


6b
level= 5
X axis
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5

The origin of the coordinate system is chosen so that the curve C lies in the quadrant in which both x and y are positive. Once the origin is specified the cells can be labeled by integer coordinates. The box whose corner coincides with the origin is labeled $(0, 0)$ and the coordinates of the other cells are obtained by counting along the coordinate directions from the $(0, 0)$ box. This method of labeling the cells is chosen so that one can easily determine the coordinates of the cells of a larger size which cover a given cell. The coordinates of the larger cell are obtained by dividing the coordinates of the smaller cell by the ratio of the size of the larger cell to the size of the smaller cell. (The remainder is dropped when dividing.) For instance consider the cell $(5, 9)$ of size $\frac{1}{16}$; this cell is contained in the cell $(5, 9)/2 = (2, 4)$ of size $\frac{1}{8}$; and also contained in the cell $(5, 9)/4 = (1, 2)$ of size $\frac{1}{4}$.

The first step in the process of approximating the patch is to find the sequence of cells which approximates the closed curve but is just inside of it; see Fig. 2a. We pick a cell c_0 which lies just inside of the curve as a starting cell; i.e., a cell inside of the curve such that one of the cell's nearest neighbors lies outside of the curve. In general we can determine if a cell is inside of the polygon by determining if the cell is to the left of the line segment joining the points $\mathbf{r}_N$ and $\mathbf{r}_1$ and to the left of the next line segment in the polygon, $\mathbf{r}_1$ and $\mathbf{r}_2$. In order to find the next cell in the sequence c_1, we look at the two nearest neighbors to c_0, n_x and n_y, which point in the same general direction as the local directed segment of the curve $\mathbf{t}_1 = \mathbf{r}_1 - \mathbf{r}_N$. If R_0 denotes the position of the cell c_0 then the position of n_x is given by $R_0 + \text{sign}(t_x)(1, 0)$ and the position of n_y is given by $C_0 + \text{sign}(t_y)(0, 1)$, where $\text{sign}(t_x)$ is the sign of the x component of $\mathbf{t}_1$ and $\text{sign}(t_y)$ is the sign the y component. We then choose for the next cell in the sequence the one of the two cells which is closer to $\mathbf{t}_1$ and which is inside the curve. Eventually the cells will reach the end of a segment and at this point the next segment of the polygon is considered in determining whether a cell is inside of the polygon. We continue the process in this manner keeping a list of the cells by keeping a list of their integer coordinates. When we return to the starting cell c_0 we have a sequence of cells which gives us what we call a discrete curve which approximates the closed curve C.

Once we have this discrete approximation of the curve by cells of size $\Delta\xi$ we find the cells of size $2\Delta\xi$ which lie along or inside of the discrete boundary of size $\Delta\xi$. The process is shown in Figs. 2a and 2b. For each cell in the list of cells of size $\Delta\xi$ we consider the cells of size $2\,\Delta\xi$ which covers the cells of size $\Delta\xi$. (The integer coordinate of this larger cell is found by dividing the coordinate of the smaller cell by 2, as described earlier.) We must determine if this larger cell of size $2\Delta\xi$ lies inside of the discrete curve of size $\Delta\xi$. In order to do this at the beginning of each size doubling we determine the extreme values of the coordinates of the cells of size $\Delta\xi$ which make up the discrete curve. We store these extreme values in two arrays of numbers I_x and I_y. The ith entry of I_x contains the minimum and maximum y coordinates of all cells with their x coordinate equal to i; and the ith entry of I_y contains the minimum and maximum x coordinates of all cells with their y coordinate equal to i. The arrays I_x and I_y are formed by going through the list of

cells once and placing the x and y coordinates of the cells in the appropriate places in the arrays. Once these arrays are formed one can check whether a given cell lies inside of the discrete curve by simply checking whether the coordinates of the cell lie within the ranges given by the arrays I_x and I_y.

If the larger cell lies inside of the discrete curve we place the coordinates of this cell in an array B; these cells eventually will make up a new discrete boundary of size $2\,\Delta\xi$. If the larger cell does not lie inside of the curve, we place the coordinates of the smaller underlying cell in an array A. Once we have gone through the complete list of cells of size $\Delta\xi$ the array B will contain a discrete approximation of the curve C of size $2\,\Delta\xi$. The array A contains cells which cannot be replaced by cells of a larger size.

At this point we can iterate the process. We use the cells given in array B and determine which cells of size $4\Delta\xi$ fit inside of the discrete curve of size $2\Delta\xi$. We always append the small cells which cannot be replaced by the larger cells to the array A. The process stops when at some doubling level the array B contains no cells. The array A will then contain cells of all the different sizes which will approximate the area enclosed by the polygon exactly the same as if cells of the smallest size were used throughout the area. Some cells may be repeated, however, and to prevent the possibility of double-counting some cells, we linearly sort all of the cells in A according to coordinate and size, and delete all cells which are listed more than once.

In Fig. 2a we show a circle with radius equal to one as it is represented by cells of size $\frac{1}{32}$. In Fig. 2b we show the resultant configuration of cells of size $\frac{1}{32}$ and $\frac{1}{16}$ after we have replaced all smaller cells with larger ones where it is possible. In Figs. 3 through 6 we show the process of doubling the cell size around the

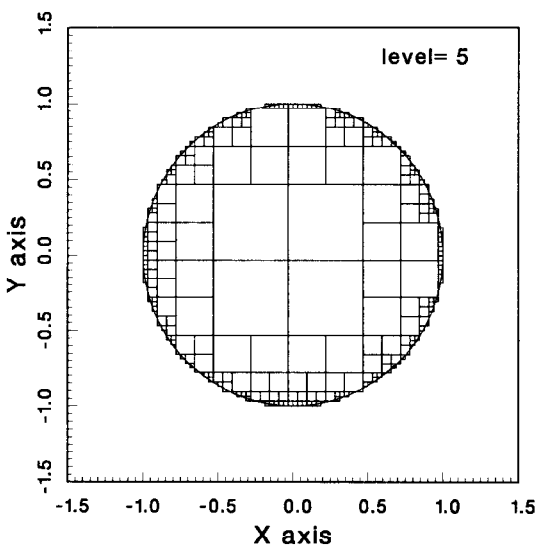


FIG. 7. Cells approximating a circular patch. The configuration of cells approximating a circular patch is shown. The size of the smallest cells is $\frac{1}{32}$.

boundary of the patch. The process stops in Fig. 6 when no cells of size 1 fit inside of the patch.

Figure 7 shows the final configuration of cells which approximate the circle. The area inside of the circle is approximated exactly the same as if we had used a uniform cell size of $\frac{1}{32}$; were we to use a uniform cell size, however, 3228 cells would be required to approximate the same area.

In Table I we list the number of cells required to approximate a circle as we vary the minimum cell size $\Delta\xi$. We also list the number of cells which would be required if we were to use a uniform cell size to give the same approximation: this number is the number of vortex blobs which would be required in the standard vortex method. One should note that the actual number of cells is proportional to $1/\Delta\xi$, whereas the number of cells required to approximate the area when using a single sized cell is proportional to $1/\Delta\xi^2$. When looking at the values of the total area of the cells in Table I it should be noted that the values do not converge to π because the circle is being approximated by an 80-sided polygon. The exact area of the polygon is 3.1383638...: it can be seen that the approximating areas do converge to this value.

In Fig. 8 we show the procedure of approximating a patch which is more complicated than a circle. The figure shows the sequence in which the cells are chosen; starting at the smallest scale and increasing the cell size by a factor of two until we reach the largest cell size, after five levels of refinement, in Fig. 8i. We see that in general several disconnected regions may develop as in Figs. 8g and h.

The algorithm works as described above for the curves shown in Figs 2 and 8; however, if the curves are spiraled too much, the algorithm for determining whether a cell is inside of the discrete curve may fail. There are several ways to prevent this

TABLE I

Number of Cells Required to Approximate a Polygon
Exact Area of the 80-gon Is 3.13836382...

Size of cells $\Delta\xi$	Actual number of cells	Number of cells if uniform of size $\Delta\xi$	Total area of cells
1/8	49	208	3.25000000
1/16	122	812	3.17187500
1/32	237	3228	3.15234375
1/64	545	12884	3.14550781
1/128	1034	51428	3.13891602
1/256	2097	205668	3.13824463
1/512	4097	822728	3.13845825
1/1024	8092	3290776	3.13832855
1/2048	16629	13163256	3.13836479
1/4096	32762	52653032	3.13836527
1/8192	65697	210612060	3.13836426
1/16384	131489	842448140	3.13836388

from happening. One way is to keep track of several extrema in the arrays I_x and I_y. The way we chose to prevent this is to break up a large spiraled region into several smaller regions so that the algorithm as described works on the smaller regions. We break up the regions, as can be seen in Figs. 10e–h, by considering the polygons obtained by drawing lines between two boundary points which are on opposite sides of the boundary curve. Breaking up the patches in this way introduces no additional boundary error in the method because the artificial boundary introduced by breaking the patch into several pieces is filled exactly. There is only a slight additional computational effort due to the fact that cells of a smaller size have to be used at these cut regions and therefore there are more cells present than had the cuts not been present.

3. The Numerical Method

We can now describe the numerical method in detail. We will describe the method for a single patch of constant vorticity; the generalization to any number of patches follows immediately.

Assume initially that we are given a region Ω_0. The vorticity is constant inside of the region and zero outside of the region. We wish to determine the shape of the region for later times as it evolves according to Euler's equation. Denote the region at time t by $\Omega(t)$ and denote the boundary curve of this region by $C(s, t)$, where s is a lagrangian parameter; we usually take s to be the arclength at $t=0$. Let $\mathbf{r}_j^n$ be an approximation to $C(j\,\Delta s, n\,\Delta t)$, where Δt is the time step, Δs is the spatial step and $j=1, 2, ..., N$.

Given a polygon described by the N vertices $\mathbf{r}_j^n$ we define a velocity at the point $\mathbf{x}$ which we denote by $\mathbf{u}(\mathbf{x}, \mathbf{r}_j^n, \Delta\xi)$. The first step in defining $\mathbf{u}$ is to find an approximation of the N sided polygon $\mathbf{r}_j^n$ by cells of size $\Delta\xi$ and larger as described in Section 2. We denote the positions of the centers of the M approximating cells by ξ_k and we denote the size of the kth cell by δ_k. We then define the complex function $\Psi(\mathbf{x})$ by

$$\Psi(\mathbf{x}) = \omega_0 \sum_{k=1}^{M} \psi_{\delta_k}(\mathbf{x} - \xi_k), \tag{3.1}$$

where ω_0 is the vorticity of the patch and ψ_δ is defined in Eqs. (1.16) and (1.17). We define the cartesian components u_x, u_y of $\mathbf{u}$ in terms of Ψ as

$$u_x(\mathbf{x}, \mathbf{r}_j^n, \Delta\xi) = \mathrm{Im}(\Psi(\mathbf{x})) \tag{3.2}$$

and

$$u_y(\mathbf{x}, \mathbf{r}_j^n, \Delta\xi) = \mathrm{Re}(\Psi(\mathbf{x})),$$

where Im and Re denote the real and imaginary parts of Ψ. As the final step in the method we integrate in time using fourth-order Runge–Kutta [16]. We define $\mathbf{r}_i^{n+1}$ as follows:

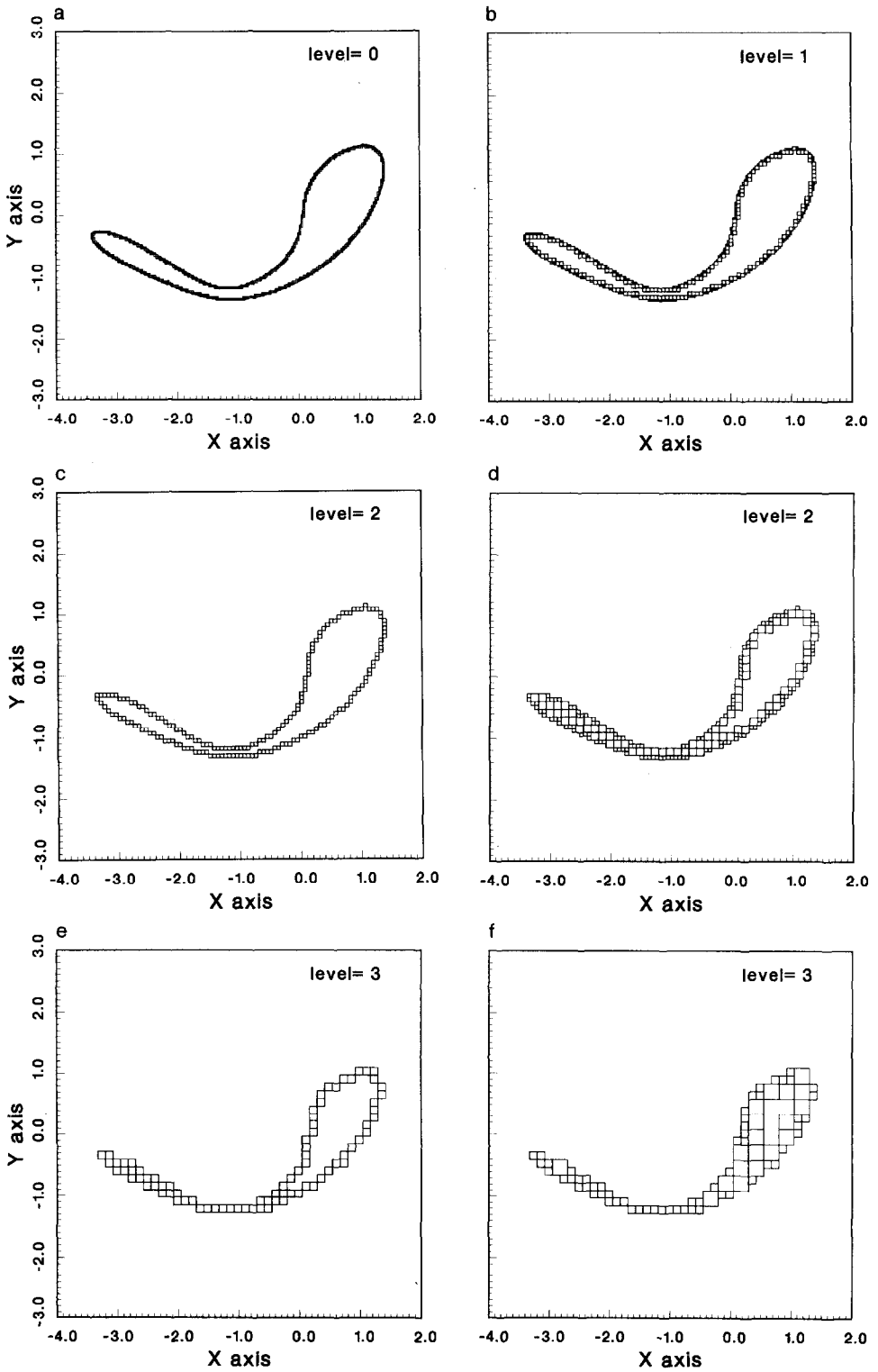


FIG. 8. Determining the cells which approximate a patch. The steps are shown which are required to determine the cells which approximate a given patch. The size $\Delta\xi$ of the smallest cells used is $\Delta\xi = \frac{1}{32}$.

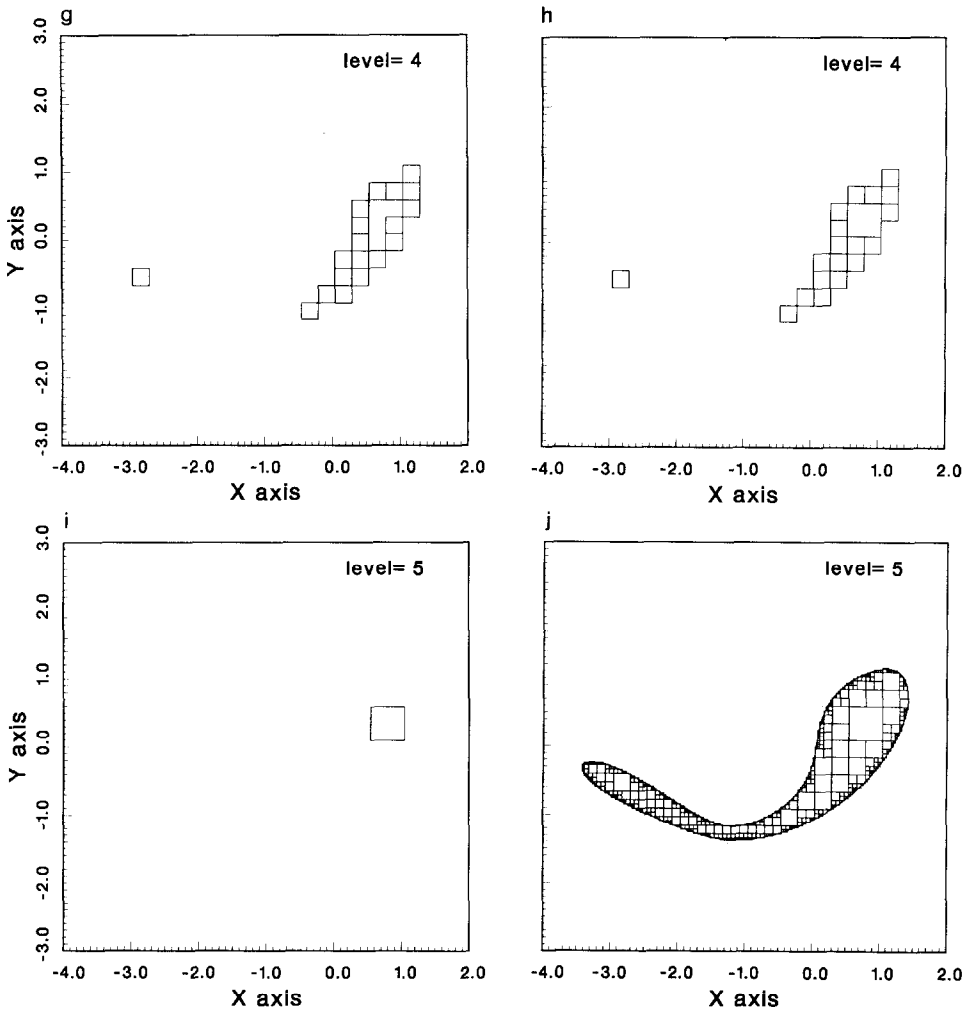


FIG. 8—*Continued*

$$
\begin{aligned}
\Delta \mathbf{r}_i^0 &= \Delta t\, \mathbf{u}(\mathbf{r}_i^n, \mathbf{r}_j^n, \Delta\xi) \\
\mathbf{r}_i^1 &= \mathbf{r}_i^n + \tfrac{1}{2}\Delta \mathbf{r}_i^0 \\
\Delta \mathbf{r}_i^1 &= \Delta t\, \mathbf{u}(\mathbf{r}_i^1, \mathbf{r}_j^1, \Delta\xi) \\
\mathbf{r}_i^2 &= \mathbf{r}_i^n + \tfrac{1}{2}\Delta \mathbf{r}_i^1 \\
\Delta \mathbf{r}_i^2 &= \Delta t\, \mathbf{u}(\mathbf{r}_i^2, \mathbf{r}_j^2, \Delta\xi) \\
\mathbf{r}_i^3 &= \mathbf{r}_i^n + \Delta \mathbf{r}_i^2 \\
\Delta \mathbf{r}_i^3 &= \Delta t\, \mathbf{u}(\mathbf{r}_i^3, \mathbf{r}_j^3, \Delta\xi) \\
\mathbf{r}_i^{n+1} &= \mathbf{r}_i^n + \tfrac{1}{6}(\Delta \mathbf{r}_i^0 + 2\,\Delta \mathbf{r}_i^1 + 2\,\Delta \mathbf{r}_i^2 + \Delta \mathbf{r}_i^3).
\end{aligned} \tag{3.3}
$$

We pick the points $\mathbf{r}_j^n$ evenly spaced along the curve C initially. As the curve $C(s, t)$ evolves there is a tendency for the curve to stretch and the distance between the approximating points $\mathbf{r}_j^n$ will change. In order to maintain a given resolution throughout the calculation it is necessary to add and delete points. When two adjacent points become separated by a distance of more than $2\Delta s$; where Δs is the initial separation of the points, we introduce another point between the two points by linearly interpolating between the two points. At certain points of $C(s, t)$ the points $\mathbf{r}_j^n$ can become very closely spaced. In order to keep the number of points approximating the curve C small when two points are closer together than some minimum distance $d_{\min}$, we simply remove one of the points. We usually take $d_{\min} = \Delta\xi/4$ in the test problems considered. This value was small enough so that points were very seldom eliminated. We intentionally picked such a small value since we were interested in investigating the formation of singularities. If one were not interested in such fine structures much computational effort could be saved by picking $d_{\min}$ much larger.

There are many higher order interpolation methods which could be used to redistribute the points; however, we did not want to introduce any smoothing into the method and so we chose a second-order method. For instance, Shelley and Baker [5] use spline techniques to redistribute the points in order to maintain a smooth parametrization of the mesh $\mathbf{r}_j^n$. Such techniques are more expensive than the one we employ and the accuracy of our method does not depend on maintaining a smooth parameterization of the curve as is required in [5]. We wish to emphasize that because of the robustness of the algorithm any kind of mesh refinement can be used along the boundary without affecting the method as long as the usual precautions are taken to maintain the accuracy of the mesh refinement.

The final part of the numerical method is the evaluation of the sum in Eq. (3.1). Since each of the functions ψ_{δ_k} is defined in terms of a multipole expansion it is natural to use a technique similar to that introduced by Greengard and Rokhlin [17] to efficiently evaluate the sum in Eq. (3.1). If the full scheme as presented in [17] is used the computational cost of our method is $O(N)$, where N is the number of points approximating the boundary of the patch. This is in contrast to contour dynamics which have a computational cost which scales as $O(N^2)$. In our implementation of the adaptive vortex method we use a slightly modified version of [17] in which we make a multipole expansion at only one scale rather than a number of scales. We pick a box size intermediate between the largest and smallest blob sizes and form the multipole expansion for all boxes of this one size which contain smaller blobs by summing the contributions from all of the blobs contained in the expansion box. When we evaluate the velocity at points outside the radius of convergence of the multipole expansion we can evaluate the contribution from all of the smaller blobs by evaluating the one multipole expansion. This modification simplifies the general method of Greengard and Rokhlin, but the method is no longer $O(N)$, but scales roughly as $O(N^{3/2})$. There is still a great savings in computational cost over that represented by the direct evaluation of the sum in Eq. (3.1).

We wish to emphasize that the speed of our method is not dependent on the multipole summation technique; this only improves the performance of the method. The fact that we approximate our flow initially with vortex blobs of different scales whose multipole expansions are already known is the reason that the method is successful. We do not have to build the multipole expansions for large blobs of the smallest size.

4. Error Analysis

In this section we discuss the numerical errors of the method. There are three sources of error in the method: the error associated with the time integration, the error associated with the discrete approximation of the boundary of the patch, and the error associated with the approximation of the area inside of the polygonal region defined by the discrete approximation of the patch. We discuss briefly why the error E should be of the form

$$E \approx C_1 \Delta\xi + C_2(\Delta s)^2 + C_3(\Delta t)^4 \tag{4.1}$$

The error due to the Runge–Kutta time integration should be fourth-order and we will not discuss this source of error in any detail. We will discuss the other two sources of error in more detail and we will verify the form of the error given in Eq. (4.1) for some specific examples.

The spatial error in the numerical method arises solely from the fact that we are not representing the area of the patch exactly. We calculate the exact velocity produced by the approximate vortex distribution given by the cells which approximate a given vortex patch. Since the velocity is given as a solution of Poisson's equation with the vorticity as the source term, we can bound the error in the velocity at any given time by a constant times the difference in area between the approximate vortex patch and the exact vortex patch. Thus we expect that the error in the method should be solely due to the error in approximating the area of the exact vortex patch by the vortex cells. This error can be broken into two parts.

The first part of the error is that due to the fact that we are approximating a continuous curve by a polygon. We expect this part of the error to contribute a term proportional to $(\Delta s)^2$, where Δs is the maximum distance between adjacent points of the polygon. To show that the error in the area has this form consider the curve $\mathbf{r}(s)$, where $0 < s < \Delta s$ and s is the arclength. Now consider the area inside of the closed curve bounded by $\mathbf{r}(s)$, $0 \leqslant s \leqslant \Delta s$ and the line segment connecting the points $\mathbf{r}(0)$ and $\mathbf{r}(\Delta s)$. The area inside of a closed curve in two dimensions is given by $\frac{1}{2}\int \mathbf{r} \times \mathbf{t}\, ds$, where $\mathbf{t} = d\mathbf{r}/ds$ is the tangent to the curve. The formula for the area is invariant under translations and thus we are free to pick $\mathbf{r}(0)$ as the origin of the coordinate system. With this choice of the origin the contribution to the integral due to the straight segment vanishes and we have

$$\text{Area} = \frac{1}{2}\int_0^{\Delta s} \mathbf{r} \times \mathbf{t}\, ds.$$

We expand the integrand in a Taylor series about the point $s=0$ and obtain

$$\mathbf{r}\times\mathbf{t}=\frac{s^2}{2}\mathbf{t}_0\times\mathbf{t}_0'+\frac{s^4}{4!}(2\mathbf{t}_0'\times\mathbf{t}_0''+3\mathbf{t}_0\times\mathbf{t}_0''')+\cdots,$$

where $'$ denotes a derivative with respect to arclength and the subscript denotes evaluation at $s=0$. If we integrate this expression we find that

$$\text{Area}=\frac{\kappa_0(\Delta s)^3}{12}\mathbf{t}_0\times\mathbf{n}_0+\cdots, \tag{4.2}$$

where κ is the curvature and $\mathbf{n}$ is the unit normal vector defined by $\kappa\mathbf{n}=\mathbf{t}'$. Equation (4.2) gives us the error due to one segment of the polygon; we sum this error over the N segments of the polygon and find

$$\text{Area}\leqslant\frac{L[\kappa(\Delta s)^2]_{\max}}{12}, \tag{4.3}$$

where $[\kappa\,\Delta s^2]_{\max}$ denotes the maximum over the N points of the polygon and L is the perimeter of the curve. Equation (4.3) gives us the desired form of the error. The error can be kept small by reducing the size of Δs in regions of the curve where the curvature is large.

The last part of the error to consider is the error due to the fact that we are approximating the area inside of an arbitrary polygon by squares of a finite size $\Delta\xi$. The error in the area is solely due to the cells which are intersected by a sement of the polygon. If we consider a segment of length l_j then the most cells of size $\Delta\xi$ that this segment can intersect is $l_j/\Delta\xi$. Thus the total error introduced in the area inside of the polygon is

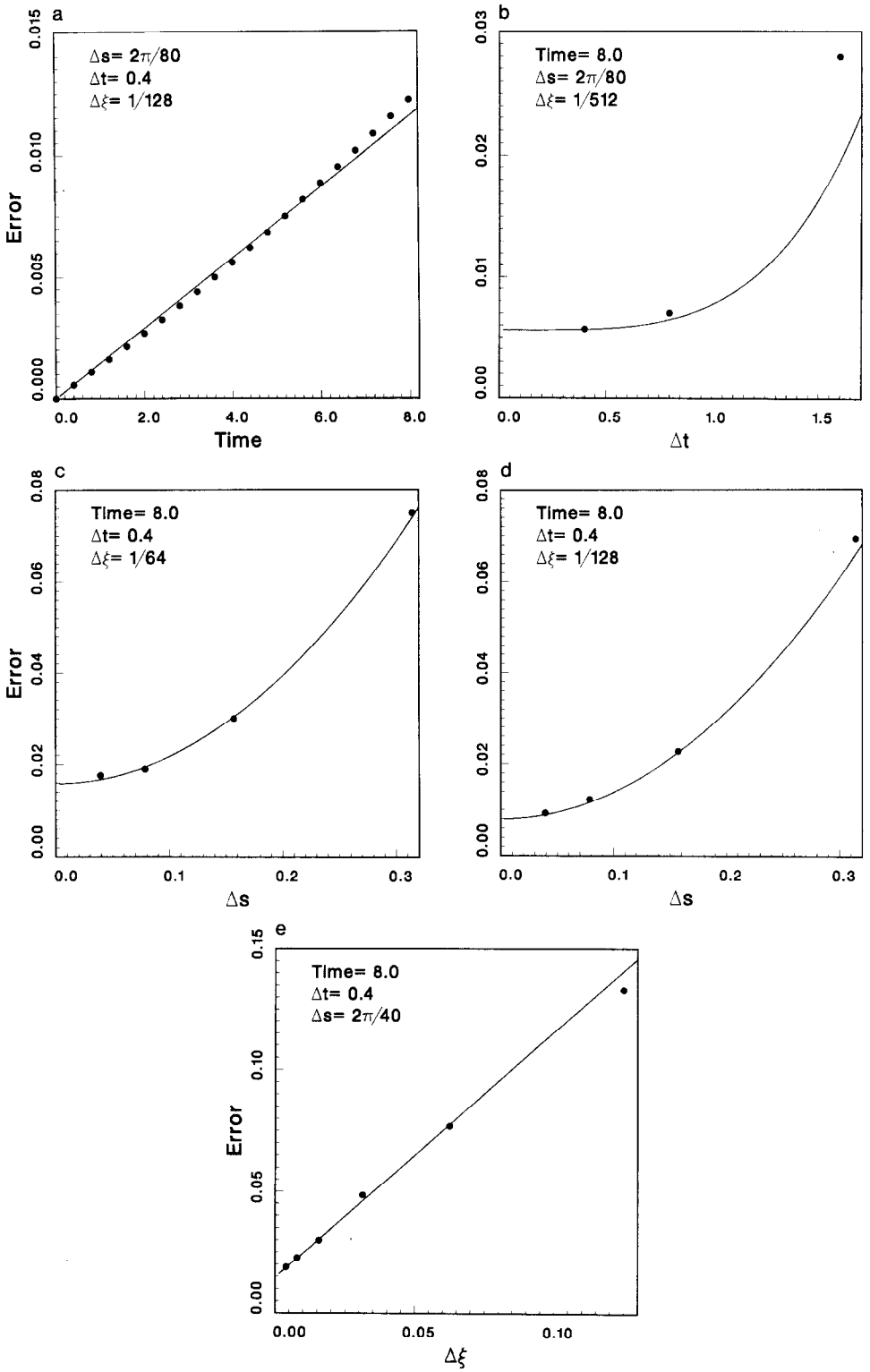
$$\text{Area}\leqslant\sum_{j=1}^{N} l_j\,\Delta\xi=L\,\Delta\xi, \tag{4.4}$$

where L is the perimeter of the polygon. Equation (4.4) gives us the desired form of the error.

We now demonstrate the validity of Eq. (4.1) numerically. We compare the exact solutions for elliptical patches given by Lamb [18] to the approximate solutions obtained by our numerical method. The position (x, y) of a fluid particle inside the elliptic patch of constant vorticity ω at time t is given by

$$x=\tfrac{1}{2}k(a+b)\cos(\Omega t+\varepsilon)+\tfrac{1}{2}k(a-b)\cos(\varepsilon) \tag{4.5}$$

FIG. 9. The error in the numerical method as a function of time, Δt, Δs, and $\Delta\xi$. The L_2 error between the calculated and exact solutions is shown for an ellipse with semi-major and semi-minor axis equal to 2.0 and 1.0, respectively. The error is plotted as a function of one of the parameters: time, Δs, $\Delta\xi$, or Δt for specific values of the other parameters shown in the figure.


a
Δs= 2π/80
Δt= 0.4
Δξ= 1/128
Error
Time
b
Time= 8.0
Δs= 2π/80
Δξ= 1/512
Δt
c
Time= 8.0
Δt= 0.4
Δξ= 1/64
Error
Δs
d
Time= 8.0
Δt= 0.4
Δξ= 1/128
Δs
e
Time= 8.0
Δt= 0.4
Δs= 2π/40
Error
Δξ

and

$$y = \tfrac{1}{2}k(a+b)\sin(\Omega t+\varepsilon) - \tfrac{1}{2}k(a-b)\sin(\varepsilon),$$

where a and b are the lengths of the semi-major and semi-minor axes of the ellipse; k and ε, $0 \leqslant k \leqslant 1$, $0 \leqslant \varepsilon < 2\pi$ are two parameters which label the different fluid particles inside of the ellipse; and $\Omega = 2ab/(a+b)^2$ is the rotational frequency of a fluid particle. The boundary of the patch is given by the curve obtained by setting $k = 1$. We have compared the exact and approximate solutions for several values of a and b and we find that the L_2 error E_{L_2} of the positions of particles along the boundary of the patch satisfies Eq. (4.1). The L_2 error is defined as

$$E_{L_2}^2 = \Delta s \sum_{j=1}^{N} |\mathbf{r}_j^n - \mathbf{r}(j\,\Delta\varepsilon, n\,\Delta t)|^2,$$

where $\mathbf{r}(\varepsilon, t) = (x, y)$ is the solution given by Eq. (4.5) for $k = 1$.

We now discuss the case $a = 1$ and $b = 2$ in detail; we pick $\omega = 1$ since the time can be scaled by the magnitude of the vorticity. For this case we pick the points along the ellipse so that they are uniformly spaced in ε and thus $\Delta\varepsilon = 2\pi/N$ and we choose $\Delta s = \Delta\varepsilon$. We calculate the L_2 error for $\Delta t = 0.4,\ 0.8,\ 1.6$; $\Delta s = 2\pi/N$ where $N = 20,\ 40,\ 80,\ 160$; and $\Delta\xi = 1/2^m$, where $m = 3, 4, \ldots, 8, 9$; we calculate the error at each time step up to a time $t = 8.0$. As can be seen in Fig. 9a over the range of times considered the error is roughly linear as a function of time. In addition to the dependence of the error as given in Eq. (4.1) we assumed a linear dependence of the error on time and thus we assume the error has the form

$$E_{L_2} = T(C_1\,\Delta\xi + C_2(\Delta s)^2 + C_3(\Delta t)^4), \tag{4.6}$$

where T is the time. We obtained the error at 810 points and fit the data to the functional form given by Eq. (4.6) with a least squares fit. For the ellipse considered we obtain

$$C_1 = 0.126, \qquad C_2 = 0.0735, \qquad C_3 = 0.000266. \tag{4.7}$$

In Fig. 9 we show the error for some specific values of the parameters T, $\Delta\xi$, Δs, and Δt. In Fig. 9 the solid curve is that given by Eqs. (4.6) and Eq. (4.7); the solid points are the actual error. In Fig. 9b we show the error as a function of Δt; in Figs. 9c–d we show the error as a function of Δs; and in Fig. 9e we show the error as a function of $\Delta\xi$. We see that in all cases the error follows the dependencies given by Eq. (4.6).

We note that for a more generic example the time dependence may be more complicated, but the error will have the same form for a given fixed time. From Eqs. (4.3) and (4.4) we see that the coefficients of the error grow linearly with the length and in generic examples the length increases as a function of time.

5. Numerical Results

In this section we will show the results of a calculation which was done using the method described in the first part of this paper. We pick the example because it shows the generic structures which evolve in patches of constant vorticity. The strengths of the adaptive vortex method presented here are immediately apparent in the way in which the method automatically adapts to resolve these long and thin features without loss of accuracy and without the introduction of any instabilities in the arm.

The example shows the versatility and robustness of the method. Figure 10 shows the evolution of two patches which are circular initially. We show the configurations of the cells which are used to approximate the area contained inside of the contours. Initially the patches are circles of radius 1.0 and are separated by a distance of 0.1. The top patch has vorticity equal to 2.0 and the bottom patch has vorticity equal to 1.0. The calculation was done with $\Delta\xi = 1/128$, $\Delta s = 2\pi/160$, and $\Delta t = 0.2$ and represents the converged result to an accuracy equal to the resolution of the plot. The calculation took 12 h on a Ridge 3200 computer; the current code runs 3 times faster on the Personal Iris 4D/20 than on the Ridge 3200. At $t = 0$ there are 320 points approximating the boundary curves and it takes approximately 2000 cells to approximate the area which is equivalent to 103,000 cells of a uniform cell size. At $t = 6.0$ there are 489 points approximating the boundaries; and it takes 3700 points to approximate the area equivalent to 103,000 cells of uniform size. The calculation from $t = 0$ to $t = 6$ takes 3 h of computer time. At $t = 13.0$ there are 1059 points approximating the boundaries; and it takes 7200 points to approximate the area equivalent to 103,000 cells of uniform size; the perimeter has increased from 12.57 at $t = 0$ to 50.06 at $t = 13.0$.

This example shows the extreme range of scales which appear in short times in the evolution of patches of constant vorticity. The long narrow structures which appear in Fig. 10h are typical of the structures which appear in vortex patch calculations. In the method presented here no restriction need be imposed on the spacing of the contours relative to the point spacing approximating the contours. We see in Fig. 10 that the long narrow tail contains a relatively small fraction of the curve.

6. Conclusion

We have presented a numerical method for evolving patches of constant vorticity. The method is a mixture of contour dynamics and vortex methods. This method is the first fully adaptive vortex method. We emphasize that because of the adaptivity of the method we are able to calculate numerical solutions of vortex patches of constant vorticity to a much higher resolution than was possible with vortex

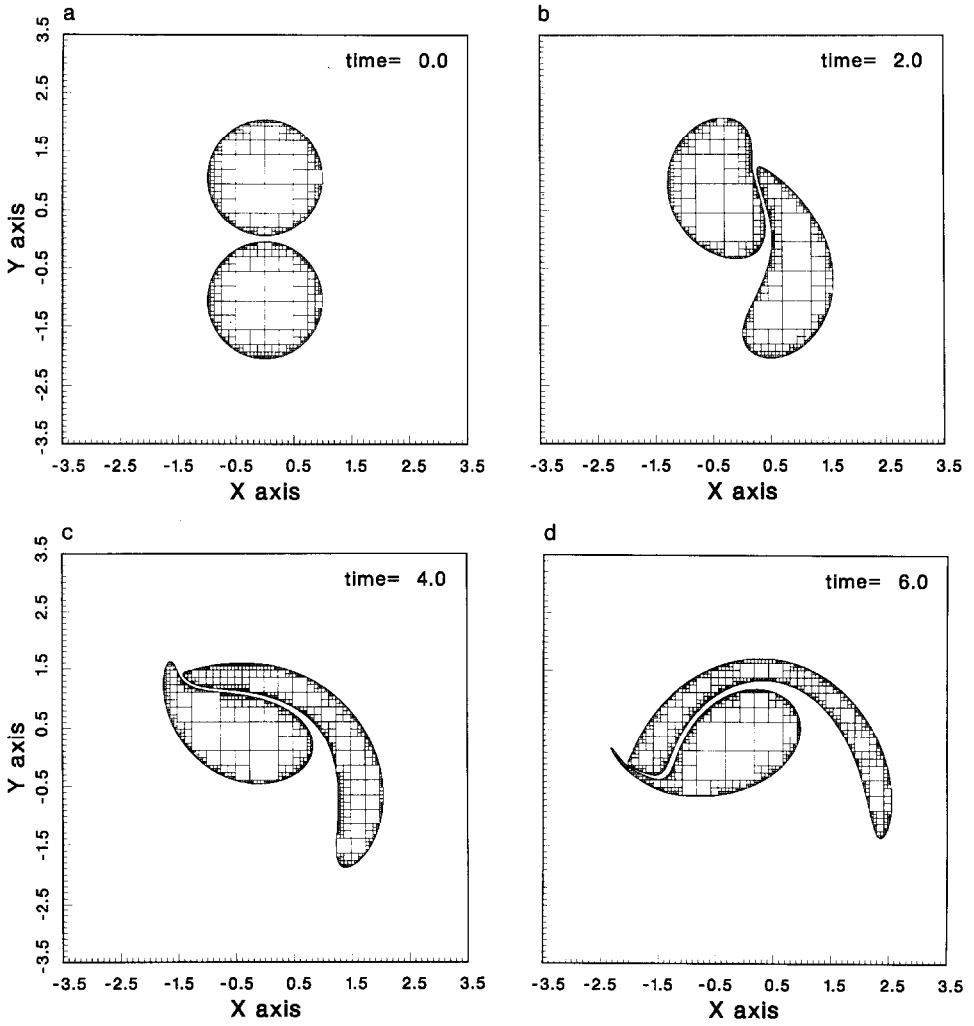


FIG. 10. The evolution of two circular patches of constant vorticity. The patches have radius $r = 1.0$ and are separated by a distance $d = 0.1$ initially. The top patch has vorticity $\omega = 2.0$ and the bottom patch has $\omega = 1.0$. We show the adaptive approximation of the two patches of constant vorticity as they evolve.

methods before the development of this method. We are able to do calculations which would require millions of vortex elements to obtain a similar resolution [19].

Our method compares favorably with standard contour dynamics methods. In our method there is no evidence of instabilities and no restrictions need be placed on the spacing of the points approximating the contour relative to the separation of the contours. The accuracy of our method is not sensitive to the formation of singularities in the boundary curve and also the accuracy does not degrade as long

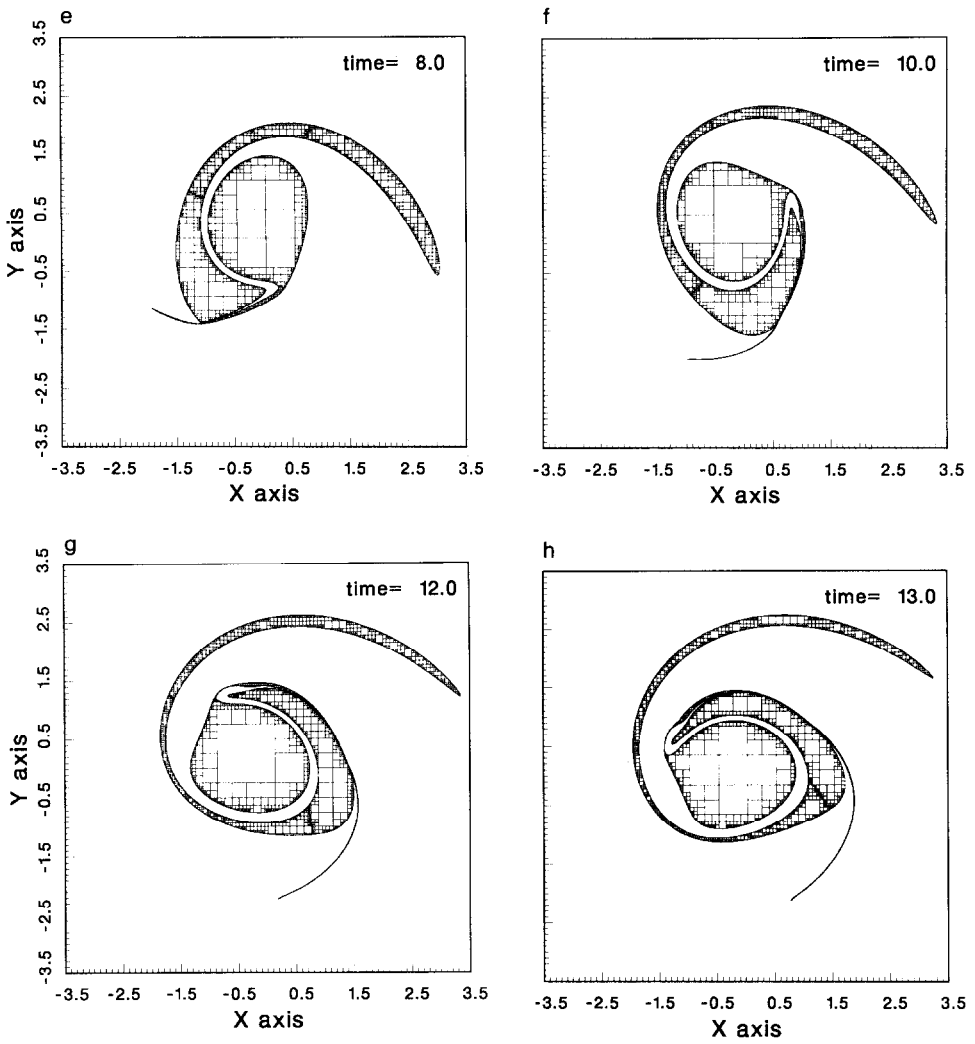


FIG. 10—*Continued*

thin finger structures form or as contours approach each other. These are extremely important features of our method since these structures are generic in vortex patch calculations and must be handled routinely.

ACKNOWLEDGMENTS

We thank Mike Shelley and Andy Majda for numerous helpful discussions. This work was performed while the author was a National Science Foundation Mathematical Sciences Postdoctoral Fellow at Princeton University.

REFERENCES

1. N. J. ZABUSKY, M. H. HUGHES, AND K. V. ROBERTS, *J. Comput. Phys.* **30**, 96 (1979).
2. Q. ZOU, E. A. OVERMAN, H. M. WU, AND N. J. ZABUSKY, *J. Comput. Phys.* **78**, 350 (1988).
3. D. G. DRITSCHEL, *J. Fluid Mech.* **172**, 157 (1986).
4. D. G. DRITSCHEL, *J. Comput. Phys.* **77**, 240 (1988).
5. M. J. SHELLEY AND G. R. BAKER, On the connection between thin vortex layers and vortex sheets, *J. Fluid. Mech.*, to appear.
6. C. POZRIKIDIS AND J. J. L. HIGDON, *J. Fluid. Mech.* **157**, 225 (1985).
7. C. POZRIKIDIS AND J. J. L. HIGDON, *Phys. Fluids* **30**, 2965 (1987).
8. G. R. BAKER AND M. J. SHELLEY, *J. Comput. Phys.* **64**, 112 (1986).
9. A. J. CHORIN, *J. Fluid Mech.* **57**, 785 (1973).
10. C. ANDERSON AND C. GREENGARD, *SIAM J. Numer. Anal.* **22**, 413 (1985).
11. O. HALD, *SIAM J. Numer. Anal.* **16**, 726 (1979).
12. J. T. BEALE AND A. MAJDA, *Math. Comput.* **39**, 1 (1982).
13. A. F. GHONIEM *et al.*, *J. Comput. Phys.* **79**, 135 (1988).
14. J. A. SETHIAN AND A. F. GHONIEM, *J. Comput. Phys.* **74**, 283 (1988).
15. A. J. CHORIN AND J. E. MARSDEN, *A Mathematical Introduction to Fluid Mechanics* (Springer-Verlag, New York, 1979).
16. G. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, (Prentice–Hall, Englewoods Cliffs, NJ, 1971).
17. L. GREENGARD AND V. ROKHLIN, *J. Comput. Phys.* **73**, 325 (1987).
18. H. LAMB, *Hydrodynamics* (Dover, New York, 1945).
19. T. F. BUTTKE, *Phys. Fluids. A* **1**, 1283 (1989).