JOURNAL OF COMPUTATIONAL PHYSICS 91, 278–297 (1990)

A Fast Vortex Method for Computing 2D Viscous Flow

SCOTT B. BADEN*

Lawrence Berkeley Laboratory, Berkeley, California 94720

AND

ELBRIDGE GERRY PUCKETT†

Lawrence Livermore National Laboratory, Livermore, California 94550

Received November 4, 1988; revised June 12, 1989

We present a fast version of the random vortex method for computing incompressible, viscous flow at large Reynolds numbers. The basis of this method is Anderson's method of local corrections and similar ideas for handling the potential and boundary layer flows. The goal of these ideas is to reduce the cost involved in computing the velocity field at each time step from being quadratic to linear as a function of the number of vortex elements. We present the results of a numerical study of the flow in a closed box due to a vortex fixed at its center. Our results demonstrate that the addition of the viscous portions of the random vortex method to the method of local corrections does not add appreciably to the cost. Furthermore, the cost of the resulting method is linear when $O(10^4)$ vortex elements are used, in spite of the fact that the majority of these elements lie in a thin band adjacent to the boundary.

1. INTRODUCTION

The hybrid vortex sheet/random vortex method was developed by Chorin [1–3] in order to compute incompressible, viscous flow at large Reynolds numbers. This method has been used to model a wide variety of viscous flow problems including flow past a circular cylinder [4–6], driven cavity flow [7], flow past a backward facing step [8, 9], wind flow over a building [10], and stability of the boundary

* Present address: Computer Science and Engineering Dept., U.C. San Diego, La Jolla, CA 92093.
Work done under the auspices of the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC03-76SF00098.

† Present address: Mathematics Dept., U.C. Davis, Davis, CA 95616. Work done under the auspices of the U.S. Department of Energy at Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

0021-9991/90 $3.00

layer [3]. The great advantage to this method—and vortex methods in general—is that the computational cost is concentrated in regions of greatest physical interest. This leads to excellent resolution of large scale features of the flow such as eddies and recirculation zones. However, one drawback of vortex methods is that the cost of directly computing the velocity due to N vortices at each of these N vortices is $O(N^2)$, thereby making the computation prohibitively expensive for large numbers of vortices.

Various solutions to this problem have been proposed. These solutions are usually based on the fact that the velocity due to vortices far away is much smoother than that due to vortices nearby and hence, the former is well represented by a continuum approximation. The velocity at a given point may therefore be divided into nearby and farfield components and—by finding an inexpensive approximation to the farfield component and computing the nearby interactions directly—the cost of computing the velocity at each of the N vortices becomes $O(MN)$, where M is the number of nearby interactions per vortex. These techniques can be divided into two distinct groups: particle-in-cell methods [11, 12] and adaptive multipole methods [13, 14]. A good review of vortex methods and of some of the techniques used to accelerate the velocity computation may be found in Leonard [15].

In this work we employ recent innovations to speed up the velocity computation in two dimensions. Most notable of these innovations is Anderson's method of local corrections [16]. This is a particle-in-cell method which is more accurate than previous particle-in-cell methods. The increased accuracy preserves the effect of high order vortex cutoffs and virtually eliminates the diffusive effects due to interpolating the approximate farfield velocity onto the vortices. The method is capable of economically computing with tens of thousands of vortex elements thereby permitting detailed flow visualizations in reasonable amounts of time.

We tested our method on the flow in a unit box driven by a single vortex fixed at the origin. This problem has previously been studied with the hybrid vortex method by Sethian [17] who used the standard $O(N^2)$ method for computing the vortex velocities. Our results are in good qualitative agreement with Sethian's work. The cost of our computation was linear in N and we estimate that if the direct method had been used, then it would have taken well over 10 times as long to complete. It is important to note that we find the cost to be linear in N even though the majority of vortices are concentrated in a thin band of cells adjacent to the boundary. This has important consequences for more general applications of the random vortex method, since this will be true for many flows of interest.

2. The Basic Numerical Method

In the hybrid vortex sheet/random vortex method the computational domain Ω is divided into two regions: an interior Ω_I away from the boundary $\partial\Omega$ and a sheet layer Ω_S adjacent to the boundary. (We use the term sheet layer to distinguish the

computational boundary layer from the *physical* boundary layer.) The random vortex method [1] is used to solve the incompressible Navier–Stokes equations within Ω_I, the vortex sheet method [2] is used to solve the Prandtl boundary layer equations within Ω_S. Each method is a particle method; the particles carry concentrations of vorticity and the velocity field within each of the respective regions is uniquely determined by the particle positions and the appropriate boundary conditions. Both methods are fractional step methods. One of the fractional steps transports the particles in their velocity field; the other applies a random walk to account for the diffusive effects of viscosity.

In Ω_I the particles are called vortex blobs and in Ω_S, vortex sheets. The no-flow boundary condition is satisfied on $\partial\Omega$ by imposing a potential flow on the interior region which cancels the normal component of the velocity due to the blobs. The no-slip boundary condition is satisfied by creating vortex sheets on $\partial\Omega$ which subsequently participate in the flow. The two solutions are matched by converting sheets that leave the sheet layer into blobs with the same circulation, converting blobs that enter the sheet layer into sheets with the same circulation, and letting the velocity at infinity in the Prandtl equations be the tangential component of the velocity on the boundary due to the interior flow. The sheet creation process and subsequent movement of the sheets into the interior of the flow mimics the physical process of creation of vorticity at a boundary and constitutes one of the attractive features of this numerical method.

2.1. *The Interior*

In Ω_I we solve the 2D, incompressible Navier–Stokes equations. In vorticity form these equations are:

$$\omega_t + (\mathbf{u}\cdot\nabla)\omega = R^{-1}\Delta\omega \tag{2.1a}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2.1b}$$

$$\mathbf{u} = (0, 0) \qquad \text{on } \partial\Omega, \tag{2.1c}$$

where $\mathbf{u}(\mathbf{x}, t)$ is the velocity, $\omega = u_y - v_x$ the vorticity, and R the Reynold's number. The advection part of (2.1a)–(2.1c) are Euler's equations:

$$\omega_t + (\mathbf{u}\cdot\nabla)\omega = 0 \tag{2.2a}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2.2b}$$

$$\mathbf{u}\cdot\mathbf{n} = 0 \qquad \text{on } \partial\Omega \tag{2.2c}$$

$$\Delta\psi = -\omega \tag{2.2d}$$

$$\mathbf{u} = (\psi_y, -\psi_x) \equiv \nabla^{\perp}\psi, \tag{2.2e}$$

where $\mathbf{n}$ is the outward normal to $\partial\Omega$ and ψ is the stream function.

We use the vortex method to solve equations (2.2a)–(2.2e). Let Δt denote the time step. In the vortex method the vorticity field at time $k\,\Delta t$ is represented as a sum of discrete patches of vorticity called vortex blobs,

$$\tilde{\omega}^k(\mathbf{x}) = \sum_{j=1}^{N} K_\sigma(\mathbf{x}_j^k - \mathbf{x})\,\Gamma_j. \tag{2.3}$$

Here $\mathbf{x}_j^k$ is the position of the jth vortex blob at time $k\,\Delta t$, Γ_j is its *strength*, K_σ is the *cutoff function*, and σ is the *cutoff radius*. The strength Γ_j is the circulation about the jth vortex. The choice of cutoff radius and cutoff function is determined by accuracy considerations. See Hald [18] and Beale and Majda [19] for a discussion of different kinds of cutoffs and their effect on accuracy. We use the cutoff proposed by Chorin [1]:

$$K_\sigma(\mathbf{x}) = \begin{cases} (2\pi\sigma\,|\mathbf{x}|)^{-1} & |\mathbf{x}| < \sigma \\ 0 & |\mathbf{x}| \geqslant \sigma. \end{cases} \tag{2.4}$$

We compute the velocity field $\tilde{\mathbf{u}}^k$ induced by the vorticity distribution $\tilde{\omega}^k$ in two steps. First we find the *free-space* velocity $\tilde{\mathbf{u}}_f^k = \nabla^\perp \tilde{\psi}_f^k$ such that $\tilde{\psi}_f^k$ satisfies (2.2d), with ω given by (2.3), and $\tilde{\mathbf{u}}_f^k(\mathbf{x}) = 0$ at $x = \infty$. We then find a potential flow $\tilde{\mathbf{u}}_p^k = \nabla^\perp \tilde{\psi}_p^k$ such that $\tilde{\psi}_p^k = -\tilde{\psi}_f^k$ on $\partial\Omega$. The sum of the two flows $\tilde{\mathbf{u}}^k = \tilde{\mathbf{u}}_f^k + \tilde{\mathbf{u}}_p^k$ satisfies (2.2b)–(2.2e) with $\psi = \tilde{\psi}_f^k + \tilde{\psi}_p^k$.

The free-space velocity field $\tilde{\mathbf{u}}_f^k$ is given by

$$\tilde{\mathbf{u}}_f^k(\mathbf{x}) = \sum_{\substack{j=1 \\ \mathbf{x}_j^k \neq \mathbf{x}}}^{N} U_\sigma(\mathbf{x}_j^k - \mathbf{x})\,\Gamma_j, \tag{2.5}$$

where $U_\sigma(\mathbf{x})$ is the velocity induced at $\mathbf{x}$ by a vortex blob of unit-strength at the origin. The blob velocity function U_σ is determined by the choice of K_σ; the U_σ corresponding to (2.4) is

$$U_\sigma(\mathbf{x}) = \begin{cases} (-y, x)/2\pi\,|\mathbf{x}|\,\sigma, & |\mathbf{x}| < \sigma \\ (-y, x)/2\pi\,|\mathbf{x}|^2, & |\mathbf{x}| \geqslant \sigma. \end{cases}$$

The potential flow $\tilde{\mathbf{u}}_p^k$ can be found by solving Laplace's equation $\Delta\tilde{\psi}_p^k = 0$ subject to the Dirichlet boundary condition $\tilde{\psi}_p^k = -\tilde{\psi}_f^k$ on $\partial\Omega$ and then differentiating per (2.2e). There are several other ways to obtain approximations to $\tilde{\mathbf{u}}_p^k$. We discuss our choice after the description of the method of local corrections in Section 3.2 below.

Given the velocity field $\mathbf{u}^k = \tilde{\mathbf{u}}_f^k + \tilde{\mathbf{u}}_p^k$ we approximate the solution of (2.2a)–(2.2e) with initial data $\tilde{\omega}^k$ by transporting the blobs in this velocity field

$$\mathbf{x}_j^{k+1/2} = \mathbf{x}_j^k + \Delta t\,\mathbf{u}^k(\mathbf{x}_j^k),$$

where the superscript "$k + 1/2$" indicates the positions of the blobs after the first fractional step. One can improve the accuracy of the advection step by employing

a second- or fourth-order time discretization scheme and using two or more velocity evaluations per time step (see [20, 21]).

Here we used a second-order Runge–Kutta which required two velocity evaluations per time step. We employ a time step constraint described in Section 2.3 below to ensure that blobs do not leave Ω during the advection step.

The second fractional step is the solution of the diffusive part of (2.1a) subject to the no-slip boundary condition:

$$\omega_t = R^{-1} \Delta\omega \tag{2.6a}$$

$$\mathbf{u} \cdot \boldsymbol{\tau} = 0 \qquad \text{on } \partial\Omega, \tag{2.6b}$$

where $\boldsymbol{\tau}$ is a vector tangent to $\partial\Omega$. The solution of (2.6a) with initial data $\tilde{\omega}^{k+1/2}$ is obtained by letting all blobs undergo a random walk

$$\mathbf{x}_j^{k+1} = \mathbf{x}_j^{k+1/2} + \boldsymbol{\eta}_j$$

where the $\boldsymbol{\eta}_j$ are independent, Gaussian distributed random vectors with mean 0 and variance $2\,\Delta t/R$. Any blobs that end up in the sheet layer or in the image of the sheet layer as a result of the random walk become sheets, and any that end up outside the image of the sheet layer are discarded. The no-slip boundary condition (2.6b) is approximately satisfied by using the vortx sheet method to cancel the tangential velocity on $\partial\Omega$ induced by the blobs with positions $\mathbf{x}_j^{k+1}$. We will now describe this method.

2.2. *The Sheet Layer*

Let Ω_S consist of those points in Ω lying within a distance ε of $\partial\Omega$. In Ω_S we use the vortex sheet method to solve the Prandtl boundary layer equations:

$$\xi_t + u\xi_x + v\xi_y = \frac{1}{R}\,\xi_{yy} \tag{2.7a}$$

$$\xi = -u_y \tag{2.7b}$$

$$u_x + v_y = 0 \tag{2.7c}$$

$$(u, v) = (0, 0) \qquad \text{at} \quad y = 0 \tag{2.7d}$$

$$\lim_{y \to \infty} u(x, y, t) = U_\infty(x, t). \tag{2.7e}$$

Here (x, y) denotes coordinates which are, respectively, parallel and perpendicular to the boundary, (u, v) denotes the respective velocity components, ξ is the vorticity, and U_∞ is the velocity at infinity. We determine U_∞ by linearly interpolating the tangential velocity induced by the interior flow at discrete points on $\partial\Omega$. We assume that the boundary is located at $y = 0$ and identify the four walls of the domain Ω with the periodic interval $[0, 4]$. As a result of this identification, we can map Ω_S onto the rectangle $[0, 4] \times [0, \varepsilon]$. This is a convenient way of dealing with

a vortex sheet that moves into a corner, for it does not involve special treatment of the corners. Other workers (e.g., [7]) have employed special procedures for sheets that move into a corner.

In the vortex sheet method the vorticity at time $t = k\,\Delta t$ is approximated by a sum of linear concentrations of vorticity,

$$\tilde{\xi}^k(x, y) = \sum_j \xi_j b_l(x - x_j^k)\,\delta(y_j^k - y),$$

where ξ_j is the *strength* of the jth vortex sheet, (x_j^k, y_j^k) is its center, δ is the Dirac delta function, and b_l is the *smoothing* function. We use the "hat" function originally proposed by Chorin [2],

$$b_l(x) = \begin{cases} 1 - |x/l|, & |x| \leqslant l, \\ 0, & \text{otherwise.} \end{cases} \tag{2.8}$$

The parameter l is often referred to as the *sheet length*, although for b_l defined by (2.8) the sheets are of length $2l$.

With the aid of (2.7b) and (2.7e) we can express the tangential velocity u in terms of the vorticity and so obtain an approximation $\tilde{u}^k$ from $\tilde{\xi}^k$

$$\tilde{u}^k(x, y) = U_\infty(x, k\,\Delta t) + \sum_j \xi_j b_l(x - x_j^k)\,H(y_j^k - y), \tag{2.9}$$

where $H(y)$ is the Heaviside function. Similarly, we use (2.7c) and (2.7d) to write v as an integral over u_x and approximate u_x with a centered divided difference to obtain

$$\tilde{v}^k(x, y) = -\partial_x U_\infty(x, t)\,y - \frac{1}{l}\sum_j \xi_j \cdot \left(b_l\left(x + \frac{l}{2} - x_j^k\right) - b_l\left(x - \frac{l}{2} - x_j^k\right)\right) \mathrm{Min}(y, y_j^k).$$

In the advection step we evaluate the velocity $(\tilde{u}^k, \tilde{v}^k)$ at the centers of the sheets and advance each sheet one time step of length Δt accordingly. If we denote the velocity at the center of the jth sheet at time $k\,\Delta t$ by $(\tilde{u}_j^k, \tilde{v}_j^k)$, then the sheet positions after the advection step are given by

$$(x_j^{k+1/2}, y_j^{k+1/2}) = (x_j^k, y_j^k) + \Delta t(\tilde{u}_j^k, \tilde{v}_j^k). \tag{2.10}$$

To satisfy the no-slip boundary condition $u = 0$ at $y = 0$ we create sheets on the boundary as follows. Let a_i, $i = 1, \ldots, M$, denote equally spaced gridpoints at $y = 0$ with grid spacing l. The sheets at the positions given by (2.10) generally induce a non-zero tangential velocity on the boundary, $\tilde{u}^{k+1/2}(x, 0)$. Let $u_i = \tilde{u}^{k+1/2}(a_i, 0)$ and let $\xi_{\max}$ denote a computational parameter called the *maximum sheet strength.*

Then for each i we create $q_i = [|u_i|/\xi_{\max}]$ sheets with centers $(a_i, 0)$ and strengths $-\text{sign}(u_i)\,\xi_{\max}$, where $[x]$ denotes the greatest integer less than or equal to x.

The numerical solution of the diffusion equation is found by letting all sheets (new *and* old) undergo a random walk in the y direction, reflecting any that go below the boundary. The new sheet positions at time $(k+1)\,\Delta t$ are thus given by

$$(x_j^{k+1}, y_j^{k+1}) = (x_j^{k+1/2}, |y_j^{k+1/2} + \eta_j|)$$

where the η_j are independent Gaussian distributed random numbers with mean 0 and variance $2\,\Delta t/R$. At the end of the diffusion step any sheets which have left the sheet layer become blobs.

In our implementation all sheets have magnitude $\xi_{\max}$. We do not create sheets at a_i if $|u_i| < \xi_{\max}$ and hence the no-slip boundary condition is satisfied at a_i only up to order $\xi_{\max}$. Other workers (e.g., [2, 3, 5, 7]) create sheets at the ith gridpoint whenever $|u_i| \geqslant \xi_{\min}$ for some $\xi_{\min} < \xi_{\max}$ and require that the sum of the strengths of these sheets exactly cancels u_i. However it has been shown [22] that this greatly increases the number of sheets created without improving the accuracy of the computation. The sheet creation algorithm presented here significantly reduces the total number of vortex elements in the computation. This results in a more economical method.

2.3. *Choosing the Computational Parameters*

There are four computational parameters in this method: the time step Δt, the sheet length l, the maximum sheet strength $\xi_{\max}$, and the cutoff σ. Since the circulation remains constant when a sheet becomes a blob we have $|\Gamma_j| = l\xi_{\max}$. Following Chorin [3] and Sethian [17] we set $\sigma = l/\pi$. The reader should consult [3, 6, 9, 22] for a more detailed discussion of the relationship between the various parameters.

The only generally agreed upon constraint that the parameters in the vortex sheet method must satisfy is the so called "CFL" condition:

$$\Delta t \max U_\infty \leqslant l. \tag{2.11}$$

The justification usually given for (2.11) is that one wants to ensure that sheets move downstream at a rate of no more that one grid point per time step. This is an accuracy condition (as opposed to a stability condition) which ensures that information propagating in the streamwise direction will influence all features of the flow which are at least $O(l)$.

To ensure that vortex blobs do not exit the box during the advection step we enforce a constraint similar to (2.11) in the interior; no vortex is allowed to move more than a distance 0.9ε (where ε is the sheet layer thickness) in any direction during an advection step. We incorporate these two constraints into one global constraint on the time step as follows. At each time step we determine the maximum velocity component of $\mathbf{u}^k$ over the centers of all vortices and the points a_i on $\partial\Omega$. We then adjust Δt accordingly before moving any vortex element.

3. The Fast Vortex Method

3.1. *The Method of Local Corrections*

Traditionally vortex methods entail solving an N-body problem directly, at a cost that is quadratic in N, the number of vortices. This limits the number of elements that can be handled in a reasonable amount of computer time, perhaps to no more than a few thousand vortices. It turns out that there are faster ways of computing the mutually-induced velocity field on a collection of vortices. These methods are based on the idea that interactions involving distant length scales can be effectively approximated with a relatively inexpensive computation. Only interactions involving nearby vortices need to be computed directly, and these account for only a small fraction of the N^2 interactions computed by the direct method.

We use a strategy based on the above observation, known as the method of local corrections [16]. It is similar to the particle–particle, particle–mesh algorithm of Hockney *et al.* [12] and more accurate than either this algorithm or Christiansen's vortex-in-cell [11] (the latter method does not compute close interactions directly). The method of local corrections exploits the fact that a vortex blob behaves like a point source of vorticity outside the cutoff radius σ, and hence induces a harmonic velocity field there. This allows one to take advantage of high order interpolation formulas for harmonic functions. The procedure for obtaining the velocities is similar in approach to that employed by Mayo [23] for obtaining the potential due to a charge distribution on the boundary of an irregular domain.

The local corrections algorithm is nearly as accurate and considerably faster than the direct method. For example, it can perform a velocity evaluation on a collection of 12,848 vortices—distributed evenly among two patches of constant vorticity—in under 7s on the Cray X-MP; the direct method takes 56s. The amount of speedup one obtains with the method of local corrections depends on the distribution of the vortices in the computational domain, with maximum speedup occurring when the vortices are uniformly distributed in the domain. Note, however, that one of the conclusions of this study is that the speedup due to this method is still significant even though the vortices are *not* uniformly distributed in the computational domain. See Anderson [16] and Baden [24, Sect. 3] for a more detailed discussion of the speed and accuracy of the method of local corrections.

The method of local corrections distinguishes between two kinds of vortex interactions: (1) far-field interactions approximated by solving a discrete Poisson equation, (2) N-body interactions computed exactly for vortices close enough to one another. A finite difference mesh, with spacing h, is superimposed on the domain; it is used to compute the far-field interactions. A second mesh of spacing h called the *chaining mesh*, with boxes whose centers coincide with the grid points of the first mesh, is also used. The edges of the chaining mesh coincide with $\partial\Omega$; the edges of the first mesh extend beyond $\partial\Omega$ by $h/2$ in each direction. We denote this extended domain and its boundary by Ω' and $\partial\Omega'$, respectively.

The computation is organized around the boxes of the chaining mesh. An integer C, called the *correction distance*, is chosen to distinguish nearby vortices from

distant ones. Vortices interact directly only if both indices of the boxes containing them differ by no more than C. It has been observed that, for a given level of accuracy, C is a constant which is independent of N. The accuracy of the algorithm improves with increasing C, but this increases the cost; $C=2$ appears to effect a reasonable tradeoff between speed and accuracy. These issues have been studied in great detail by Baden [24, Sect. 3] and the interested reader is referred there for a more complete discussion of the relationship between C and the speed and accuracy of the method. Note that the method of local corrections in predicated on the assumption that vortex blobs behave like point vortices at distances greater than Ch from their centers and hence we must ensure that $\sigma \leqslant Ch$.

In the following discussion we omit mention of the time step k for notational convenience. In the method of local corrections we first compute an approximation $\tilde{\mathbf{u}}_f^h$ to the free-space velocity $\tilde{\mathbf{u}}_f$ by solving a discrete Poisson equation on the first finite difference mesh,

$$\Delta^h \mathbf{u}^h(\mathbf{x}) = \sum_{j=1}^{N} g_D(\mathbf{x}-\mathbf{x}_j), \qquad \mathbf{x}\in\Omega' \tag{3.1a}$$

$$\mathbf{u}^h(\mathbf{x}) = \sum_{j=1}^{N} (-(y-y_j), x-x_j)/2\pi\,|\boldsymbol{x}-\mathbf{x}_j|^2, \qquad \mathbf{x}\in\partial\Omega'. \tag{3.1b}$$

Here Δ^h is the discrete Laplacian, $\mathbf{x}_j$ is the center of the jth vortex, and

$$g_D(\mathbf{x}) = \begin{cases} \Delta^h((-y, x)/2\pi\,|\mathbf{x}|^2), & |x| \leqslant Dh \quad \text{and} \quad |y| \leqslant Dh \\ 0, & |x| > Dh \quad \text{and} \quad |y| > Dh. \end{cases}$$

The function g_D approximates the discrete Laplacian of the velocity field due to a *point* vortex at the origin, and is zero outside a square neighborhood of the vortex. The parameter D is an integer called the *spreading distance* and must satisfy $D \leqslant C$. Thus, like C, D is also independent of N, and the cost of computing the right-hand side of (3.1a) is proportional to N. To compute the boundary condition (3.1b) we evaluate the velocity induced on $\partial\Omega'$ by point sources of vorticity centered at the $\mathbf{x}_j$.

Having set up the right-hand side and boundary conditions for (3.1a), (3.1b), we use a fast Poisson solver to obtain $\tilde{\mathbf{u}}_f^h$. (We used a solver that was accurate to fourth order in the mesh spacing h.) This velocity field will be interpolated onto the centers of the vortices; but first it must be corrected to account for the influence of the nearby vortices which do not act like point sources of vorticity.

The local corrections are done one box at a time. Associated with each box is a surrounding region of Ω that is C boxes thick on each side—called the *correction neighborhood*—and an interpolation stencil. (We use a five-point stencil; the interpolation procedure is accurate to fifth order.) The local corrections are done in two steps. In the first step we compute the point vortex velocities at each point of the interpolation stencil which are due to the vortices in the correction neighborhood and subtract these values from $\tilde{\mathbf{u}}_f^h$. We use these corrected values of $\tilde{\mathbf{u}}_f^h$ when inter-

polating onto the vortices in the box. In the second step we compute the influence of each vortex in the correction neighborhood on each vortex in the box using the exact blob velocity function U_σ.

Several extensive studies have been performed to determine the accuracy of the method of local corrections [16, 24, 25]. These studies have demonstrated the effectiveness of the method in preserving the accuracy of high order cutoffs when the vortex method is used to approximate a solution of the Euler equations. Furthermore, a very careful investigation by Baden [24] of the flow field due to two patches of vorticity of opposite sign has shown that any diffusive effects due to the mesh are negligible.

3.2. *The Potential Flow*

In our solution of the potential flow problem we employ a modified method of images scheme suggested by Anderson [26]. This method is based on the observation that the potential flow $\tilde{\mathbf{u}}_p$ is the flow due to an infinite set of images of the vortices in the box [27, p. 378]. The positions of these images may be found by periodically extending the box in the plane and reflecting each vortex about the walls of the boxes. The idea is to include any images that are within one correction distance of $\partial\Omega$ in the computation of $\tilde{\mathbf{u}}_f$ and hence, in the computation of $\tilde{\mathbf{u}}_f^h$. The reason for including these images is because their influence on nearby vortices inside Ω cannot be accurately represented in a finite difference solution of $\tilde{\mathbf{u}}_p$. This is because of the sharp gradients in the velocity field near the boundary due to the images. We eliminate the contributions of these images to $\tilde{\mathbf{u}}_p$ by explicitly including them in the computation of $\tilde{\mathbf{u}}_f^h$, where they can be locally corrected.

To accommodate the image vortices in the computation of $\tilde{\mathbf{u}}_f^h$ we extend Ω' by $D + C$ boxes in all directions. For a vortex in Ω, which is within C boxes of the wall and away from a corner, one image is generated by reflecting the vortex in the plane of the wall and taking the negative of the strength. For a vortex in a corner three images are generated: one reflected in the plane of each of the two adjacent walls and one reflected through the corner. The first two images have strengths of opposite sign from that of the original vortex, while the third image has the same strength as the original.

We compute $\tilde{\psi}_p^h$, an approximation to $\tilde{\psi}_p$ on the unextended domain Ω, as follows. We first solve the discrete Laplace equation $\Delta^h \tilde{\psi}_p^h = 0$ subject to the Dirichlet boundary condition $\tilde{\psi}_p^h = -\tilde{\psi}_f$ on $\partial\Omega$, taking care to include the influence of the image vortices when setting up the boundary conditions. We use divided differences to obtain $\tilde{\mathbf{u}}_p^h$ at the grid points and then interpolate to obtain approximate values for $\tilde{\mathbf{u}}_p$ at arbitrary $\mathbf{x} \in \Omega$ (here we use a four-point stencil). All of the finite difference formulas we used are accurate to fourth order. We take a single-sided divided difference of $\tilde{\psi}_p^h$ at the boundary to obtain the tangential velocity $\tilde{u}_p^h$ directly, since we know of no fourth-order formula for computing the tangential derivative of $\tilde{\psi}_p^h$ at the boundary. Since the stream function induced by a vortex and its image(s) algebraically cancel one another on the wall(s) closest to them, we do

not compute such influences when setting down the boundary conditions for $\tilde{\psi}_p^h$. This is done to avoid a possible loss of accuracy due to roundoff errors. We also employ algebraic cancellation in the direct computation of $\tilde{\mathbf{u}}_p^h \cdot \mathbf{n}$.

3.3. *Speedup of the Vortex Sheet Method*

We have employed one relatively simple modification of the original vortex sheet algorithm which significantly speeds up the computation of the velocity of a sheet which is due to the other sheets. From (2.9) it is apparent that the velocity of a given sheet is affected only by those sheets within a distance $2l$ of its center. We divide the sheet layer Ω_S into M "bins," where M is the number of grid points a_i on the boundary at which sheets are created. The ith bin extends over $a_i - l/2 \leqslant x < a_i - l/2$ and $0 \leqslant y < \infty$. (Recall that $a_i - a_{i-1} = l$.) Thus, sheets in the ith bin are influenced only by other sheets in the ith bin and the two adjoining bins. At the end of each time step we sort the sheets by bin. Assuming the sheets are uniformly distributed (in the x-direction) over Ω_S, the cost of the sheet velocity evaluations is now $O(N_S)$ rather than $O(N_S^2)$, where N_S is the number of sheets in the flow.

4. Computational Results

4.1. *The Results of a Numerical Study*

We present results for the "spinup" problem. In this problem a single vortex is fixed at the center of the unit box Ω_I with sufficient strength to induce a unit velocity at the center of each wall. We set the numerical parameters as follows: the Reynolds $R = 1000$; the sheet layer thickness $\varepsilon = 0.02$; the maximum sheet strength $\xi_{\max} = 6.25 \times 10^{-3}$; and the sheet length $l = 0.1$. The initial time step was $\Delta t_0 = 0.05$. As described in Section 2.3 the cutoff radius was chosen to be $\sigma = l/\pi$. In the interior we used a second-order Runge–Kutta time integration scheme. This requires two velocity evaluations per time step, a fact which should be kept in mind when we discuss the computation time below. Due to doubts about the effectiveness of a higher order time discretization in the vortex sheet method (see [17, 22]) we used only the first-order Euler method (2.10) in the sheet layer. We used a 60×60 grid for both the chaining mesh and the Poisson solver; so the parameter denoted h in Section 3.1 is $h = 0.016\overline{6}$. We set both the correction distance C and the spreading distance D equal to 2.

We ran the calculation until time $t = 5.0$ on a Cray X-MP. Figures 1 and 2 show a series of snapshots taken at various times during the run. The formation of eddies is quite clear and our results appear to be in good qualitative agreement with those of Sethian [17]. However, in the computation shown here there are roughly 10 times as many computational elements, each with one-eighth the strength.

During the initial time step 3760 vortex sheets were created. During the second time step 156 sheets left the sheet layer and became blobs. The maximum (componentwise) velocity of these blobs was 0.98, so the time step was reduced to 0.018.

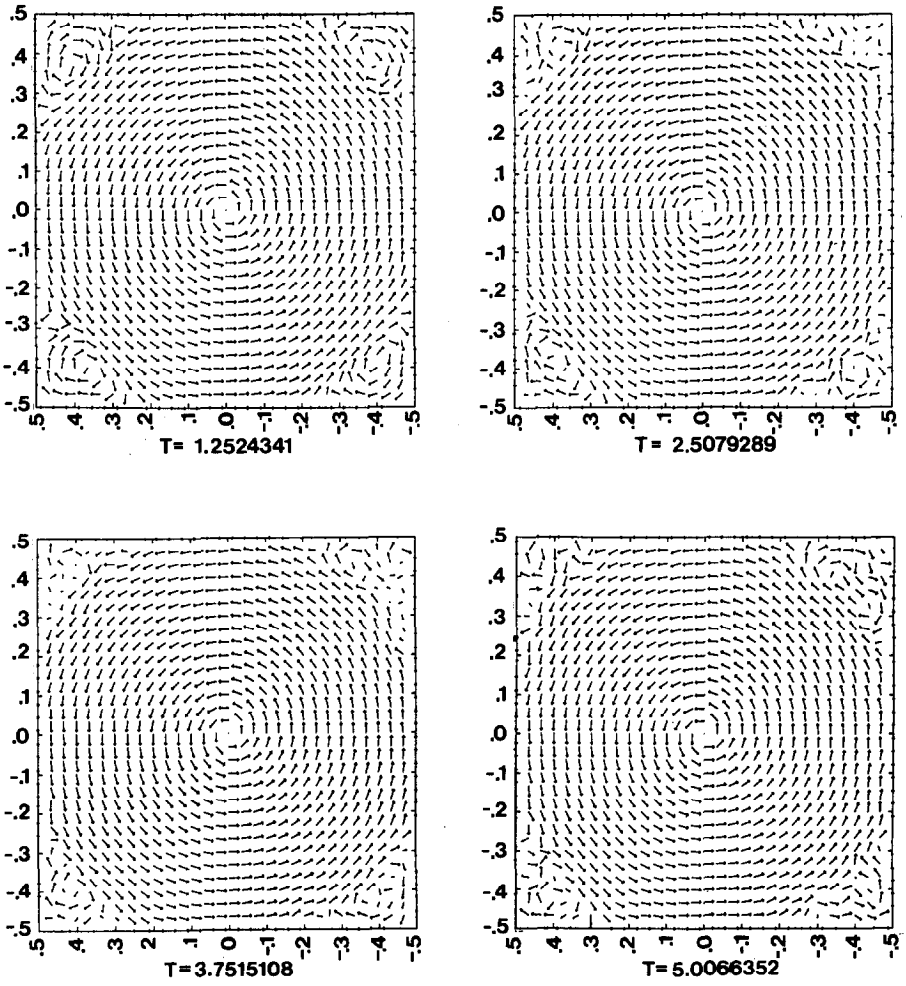

FIG. 1. A single stationary vortex induces a counterclockwise flow. Vector velocity plots clearly show the formation of counterrotating eddies. To emphasize the details near the walls, the vector lengths have been scaled so that the vectors in the center region have a constant length. Thus the vectors near the walls appear relatively larger than they actually are.

The time step slowly decreased throughout the run and attained a minimum value of 0.0074. This run took 380 time steps and consumed 2889 s (48.15 min) of CPU time on a Cray X-MP. Of this, only about 2.2% of the time was spent in the sheet calculation. At the end of the run there are 13,175 blobs, 2365 images, and 4267 sheets.

In Fig. 3 we plot the number of computational elements versus the time step. It is apparent that the total number of vortex blobs and images steadily increases with time but that the number of sheets is roughly constant.

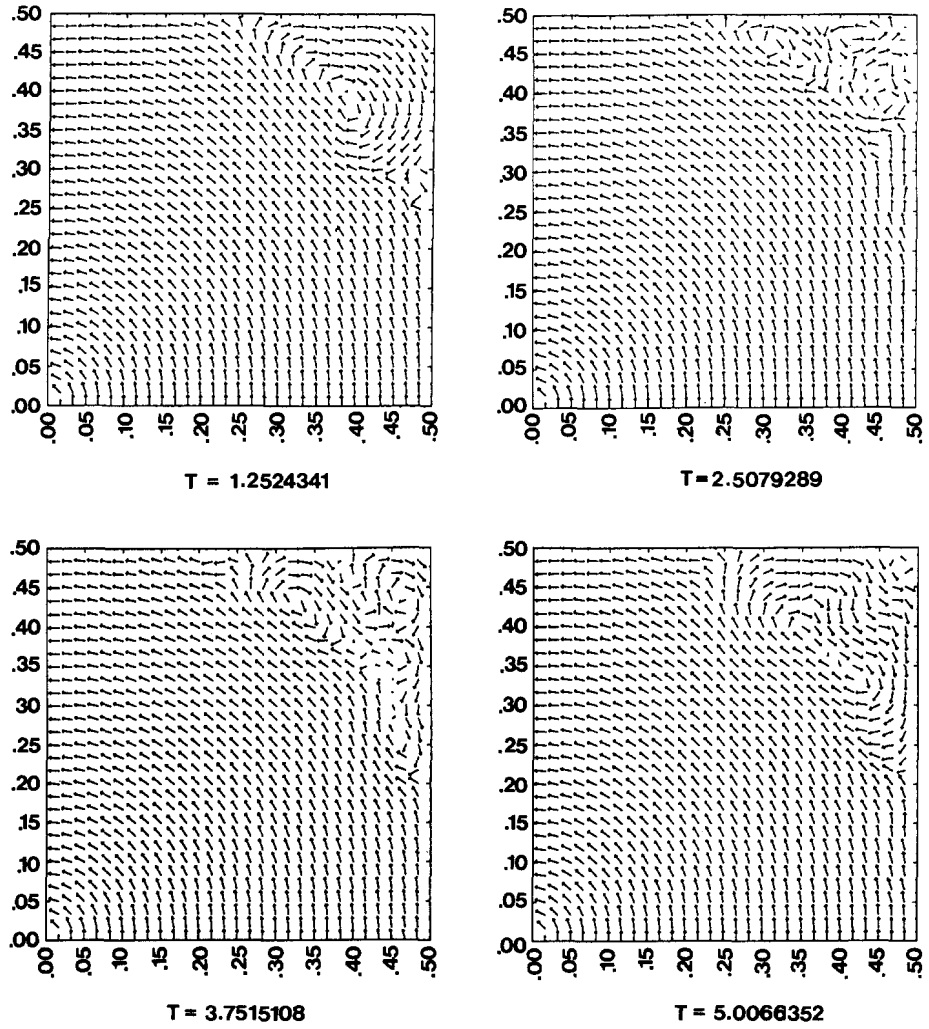

FIG. 2. These plots show finer detail than in Fig. 1, but only for the upper right-hand corner of the domain.

Figure 4 shows the total computational cost per time step and compares it with the estimated cost of using the direct method to compute blob velocities. In order to estimate the cost of using the direct method to do velocity evaluations we timed a simple program that directly computed the free-space velocity function (2.5) for various values of N. We found that the cost of computing just one interaction of a velocity evaluation was at least 0.4 μs on the Cray X-MP when the vector capabilities of the machine were being used in the most efficient manner. (The cost depends on the length of the vector.) Taking this as the cost for one velocity evalua-

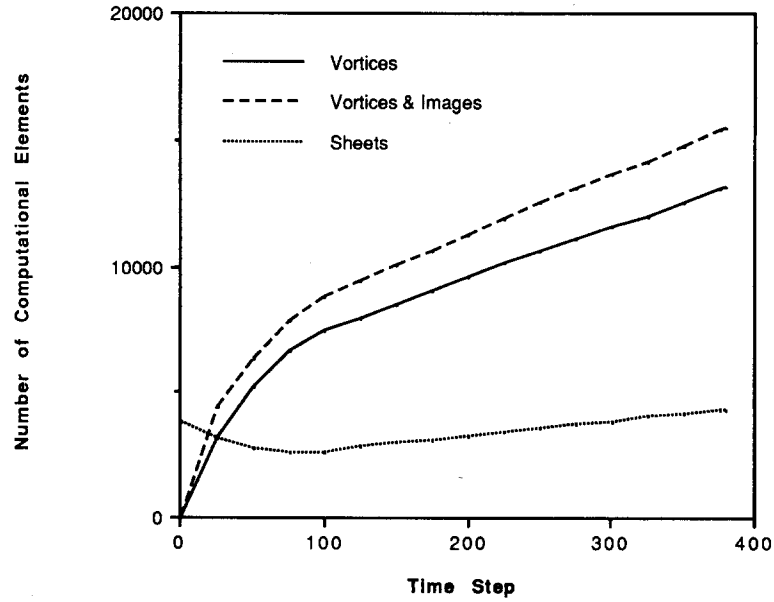

FIG. 3. The number of vortex elements varies as a function of time. The number of vortex blobs and images steadily increase while the number of vortex sheets remains relatively stable.

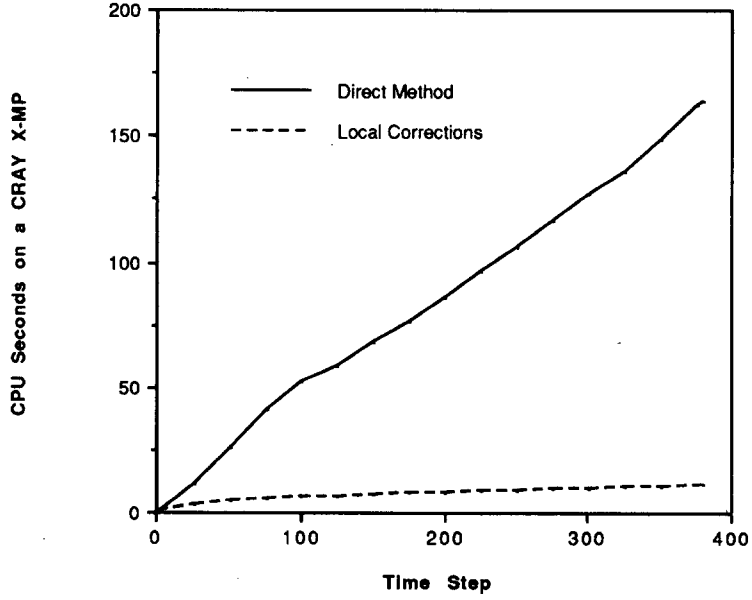

FIG. 4. The cost of a time step evaluation drops substantially when the local corrections algorithm is used to evaluate velocities instead of the direct method. The relative speedup of our method increases with the number of vortices N, and N is increasing with time. The times are reported in seconds of CPU

tion and using the statistics obtained from our trial run we estimate that, if the direct method had been used for the run shown in Figs. 1 and 2, then it would have computed a total of 3.9×10^{10} interactions, at a cost of at least 31,471 s (524.5 min) of CPU time. We estimate that the cost of any additional computation, e.g., the potential flow, the random walks, and the sheet velocities, would add only about 5% to the running time of the computation. Thus, our method took less than one tenth as long as the direct method would have. Furthermore, the speedup is an increasing function of N; by the end of the run our method is 14.54 times faster.

In Fig. 5 we plot the total computational cost per time step as a function of N, the number of vortex blobs. Remember, the times shown in Figs. 4 and 5 are the *total* cost per time step. In addition to the vortex blob velocity evaluations this includes all overhead (such as data structures), the Poisson solves, the random walks, and the sheet computation. For comparison we have plotted this against a linear function of N defined by $f(N) = 0.0008 \cdot N + 0.663$. It is clear that the cost of our method is linear in N over the range of N shown.

Note that the actual speedup achieved by the method of local corrections depends on the distribution of the vortices in Ω_I. For example, if all of the vortices are concentrated in one box, then all vortex interactions will be computed directly and the method of local corrections is no faster than the direct method. Conversely, the method of local corrections should achieve its maximum speedup when the vortices are uniformly distributed in Ω_I. Similar remarks apply to the bin data

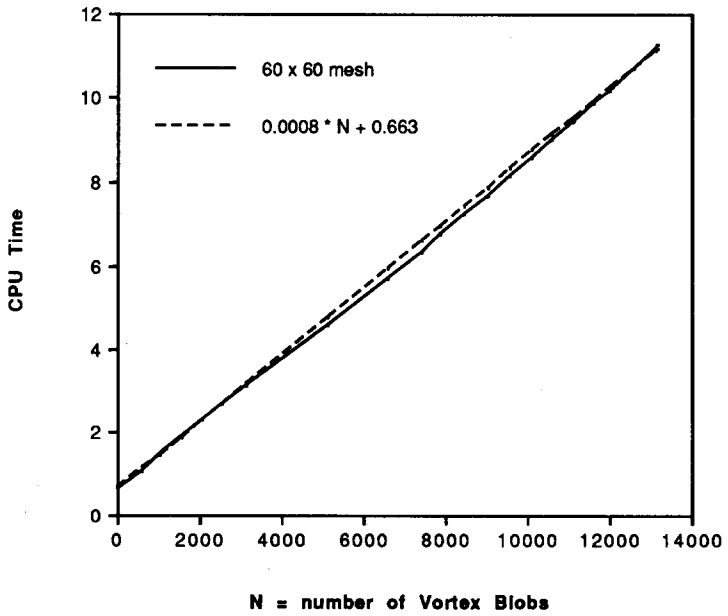

FIG. 5. For the problem presented here the total cost of a time step evaluation is roughly a linear function of the number of vortex blobs. The dotted line plots a hypothetical linear cost function.

structure used to increase the speed of the vortex sheet method. It is important to note that in the problem presented here the vast majority of vortices lie within a thin band adjacent to the boundary and therefore the vortices are not uniformly distributed in Ω_I; yet we still obtain significant speedups over the direct method.

4.2. *The Effect of the Computational Parameters on Cost and Accuracy*

There are essentially two groups of computational parameters in our method: those associated with the standard random vortex method and those associated with the method of local corrections. Included in the former group are the maximum sheet strength $\xi_{\max}$, the sheet length l, the time step Δt, and the blob core radius σ. The choice of core function, the effect of σ on the accuracy, and the relation between Δt and σ has been well documented in the literature. We refer the reader to [18–20, 28] for further information.

The effect that $\xi_{\max}$, l, and Δt have on the accuracy of the vortex sheet method has been studied by Puckett [22]. Briefly, the conclusions drawn there are that $\xi_{\max}$ is the key parameter, the error behaves roughly like $O(\sqrt{\xi_{\max}})$, and that one should use caution when decreasing the sheet length l. Here we have adopted the point of view that to attain greater accuracy one should generally decrease $\xi_{\max}$ and leave l fixed. These conclusions seem to be in agreement with Ghoniem and Sethian's exhaustive study of the effect of these parameters on the random vortex method when it is used to model the flow past a backward facing step [9].

As mentioned in Section 2.3 we choose $\sigma = l/\pi$. This relation is due to Chorin [3] and is based on the observation that for this choice of σ the velocity due to a blob tends toward that due to a sheet with the same circulation as the blob approaches the boundary.

The method of local corrections has three important parameters: the mesh size h, the spreading distance D, and the correction distance C. The effect of these parameters on the accuracy and speed of the method has been extensively studied by Anderson [16] and Baden [25]. The interested reader is referred there for a detailed discussion of this issue. Based on the above-mentioned work we choose $C = D = 2$ as this appears to result in the most cost-effective trade-off between speed and accuracy.

In order to assess the effect of the sheet strength $\xi_{\max}$ and the mesh size h on the cost of the method we ran a sequence of runs on a Cray 2. All parameters except for $\xi_{\max}$ and h have the values that were used for the results discussed in Section 4.1 above. In Fig. 6a we plot the total CPU time per time step as a function of $\xi_{\max}$ for a set of four runs with $h = 0.033\overline{3}$. Note that the cost is roughly $O(\xi_{\max}^{-1})$. Since decreasing $\xi_{\max}$ by 2 roughly doubles the number of computational elements, this corresponds to a method for which the cost is linear in the number of computational elements. After the 200th time step or so the increase in cost from $\xi_{\max} = 0.0125$ to $\xi_{\max} = 0.00625$ is somewhat more than 2. This may be because we are now computing too many direct interactions. In other words, we may have saturated the correction neighborhoods near the boundary with vortices.

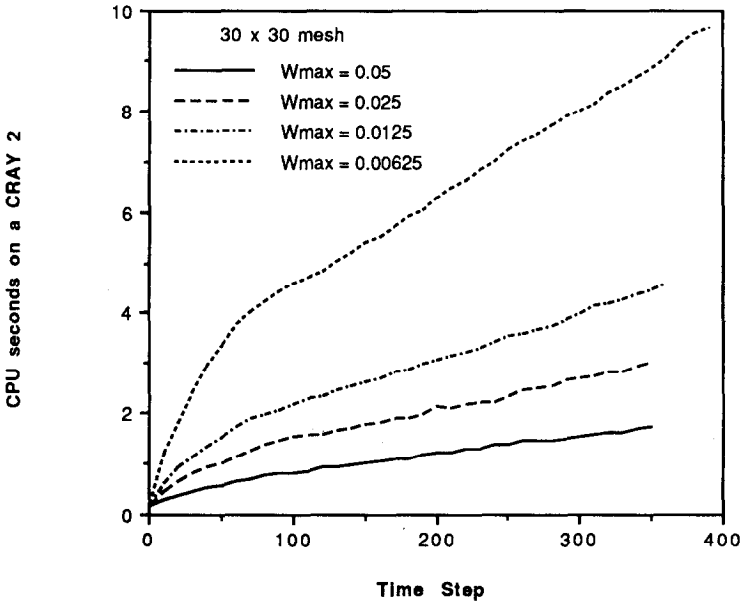

FIG. 6a. The actual cost per time step on a 30×30 mesh for four values of ξ_{max}.

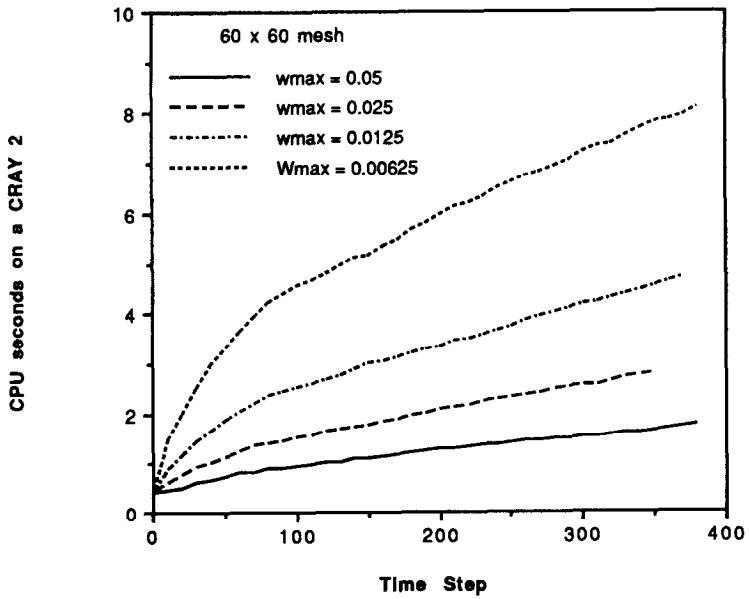

FIG. 6b. The actual cost per time step on a 60×60 mesh for four values of ξ_{max}. Note that the cost doubles when ξ_{max} is halved. Since halving ξ_{max} doubles the number of computational elements, this corresponds to a method which is linear in the number of computational elements.

To test this hypothesis we decreased h by 2 and repeated the four runs. It is apparent from the results in Fig. 6b that the cost is now $O(\xi_{\max}^{-1})$ over the entire range of $\xi_{\max}$. It should also be remarked that since the number of images is a function of Ch there are more images for $h = 0.033\overline{3}$. This will increase the cost somewhat. However as h is increased by 2 the number of images is observed to increase by roughly a factor of 4 for all values of $\xi_{\max}$. Yet for all values of $\xi_{\max}$ but 0.00625 the cost with $h = 0.016\overline{6}$ is the virtually the same as with $h = 0.033\overline{3}$. Therefore we conclude that the increase in cost after the 200th time step when $h = 0.033\overline{3}$ and $\xi_{\max} = 0.00625$ is because there are too many vortices per interaction neighborhood.

It is important to note that this test problem is a grueling one for the method since vortices are never thrown away. In many applications vortices will be discarded after reaching a certain point downstream. In these problems the total number of vortex elements in the flow tends toward a fixed number rather than always increasing with time.

4.2. *Generalization of the Method to More Complex Domains*

The method presented here can readily be generalized to other, more complex geometries. In most cases the details of the algorithm are essentially the same. As with the standard random vortex method, it is necessary to solve a potential problem at each time step in order to satisfy the no flow boundary condition. The actual details of the implementation of this Poisson solve always depends on the particular geometry of the domain. However, there are a wealth of techniques for solving such problems, many of which have been developed specifically for vortex methods (e.g. [4, 5, 9, 10, 14]). For domains which can described as the union of rectangles (such as a backward facing step) one could easily apply a domain decomposition method such as one of those described in [29]. We also remark that finite element methods have been receiving some attention lately as a means of programming a random vortex method for an arbitrary geometry.

The generalization of the method of local corrections to any of these geometries can be organized around the solution of the potential flow problem. One must simply devise an efficient scheme for determining which vortices are close to one another and then find a stencil upon which to base the interpolation of the farfield velocity. There is no need for a rectangular grid structure since the interpolation formulas are based on the fact that the components of the velocity are the real and complex parts of an analytic function in the complex plane (see [16]). Hence the accuracy of the interpolation scheme is only a function of the maximum distance between the points on the stencil.

The implementation of the vortex sheet method is the same as it would be with a standard $O(N^2)$ random vortex method, with the addition of the bin data structure for reducing the number of comparisons as described in Section 3.3 above. In other words, one must simply divide the boundary up into bins and devise a scheme for keeping track of which sheets are in a given bin.

5. Conclusions

The goal of this paper has been to demonstrate a fast, accurate vortex method for computing two-dimensional, incompressible, viscous flow at large Reynolds numbers. We have shown that the cost of the method remains linear even though most of the computational elements are concentrated in a thin band adjacent to the boundary. This remains true even for computations with more than 20,000 com-

Sethian [17].

A typical run of the type shown here—beginning with no vortex elements, running for 380 time steps, and ending with 13,175 blobs, 2365 images, and 4267 sheets—consumed 48.15 min of CPU time on a Cray X-MP. This run would have taken at least 10 times longer to complete if the vortex blob velocities were computed using the direct method. Moreover, the speedup improves as the number of computational elements increases. At the end of the run the cost of one complete time step for nearly 20,000 vortex elements—including two velocity evaluations, one random walk, and one Poisson solve in the interior and one velocity evaluation and random walk in the sheet layer—was 11.27 s of CPU time. The direct method alone would take at least 163.79 s—nearly 15 times as long.

We remark that our code can be readily modified to execute in parallel on a multiprocessor such as the Cray X-MP as discussed by Baden [25, 30]. Future work in this area should include a careful comparison on a Cray or similar vector computer between the type of algorithm presented here and a hybrid vortex code based on an adaptive multipole algorithm such as described in [13]. It will also be a great interest to extend these methods to three dimensions.

Acknowledgments

The authors would like to thank Chris Anderson, Alexandre Chorin, Phil Colella, and Jamie Sethian for their advice and support during the course of this work.

References

1. A. J. Chorin, *J. Fluid Mech.* **57**, 785 (1973).
2. A. J. Chorin, *J. Comput. Phys.* **27**, 428 (1978).
3. A. J. Chorin, *SIAM J. Sci. Stat. Comput.* **1**, 1 (1980).
4. A. Y. Cheer, *SIAM J. Sci. Stat. Comput.* **4**, 685 (1983).
5. A. Y. Cheer, *J. Fluid Mech.* **201**, 485 (1989).
6. E. Tiemroth, Thesis, U. C. Berkeley Naval Arch. Dept., 1986 (unpublished).
7. Y. Choi, J. A. C. Humphrey, and F. S. Sherman, *J. Comput. Phys.* **75**, 359 (1988).
8. A. F. Ghoniem, A. J. Chorin, and A. K. Oppenheim, *Philos. Trans. Roy. Soc. London A* **304**, 303 (1982).
9. J. A. Sethian and A. F. Ghoniem, *J. Comput. Phys.* **74**, 283 (1988).

10. D. M. SUMMERS, T. HANSON, AND C. B. WILSON, *Int. J. Numer. Methods Fluids* **5**, 849 (1985).
11. J. P. CHRISTIANSEN, *J. Comput. Phys.* **13**, 363 (1973).
12. R. W. HOCKNEY, S. P. GOEL, AND J. W. EASTWOOD, *J. Comput. Phys.* **14**, 148 (1974).
13. L. GREENGARD AND V. ROKHLIN, *J. Comput. Phys.* **73**, 325 (1987).
14. L. GREENGARD, *SIAM J. Sci Stat. Comput.* **11**, 603 (1990).
15. A. LEONARD, *J. Comput. Phys.* **37**, 289 (1980).
16. C. R. ANDERSON, *J. Comput. Phys.* **62**, 111 (1986).
17. J. A. SETHIAN, *J. Comput. Phys.* **54**, 425 (1984).
18. O. H. HALD, *SIAM J. Numer. Anal.* **16**, 726 (1979).
19. J. T. BEALE AND A. MAJDA, *J. Comput. Phys.* **58**, 188 (1985).
20. C. R. ANDERSON AND C. A. GREENGARD, *SIAM J. Numer. Anal.* **22**, 413 (1985).
21. O. H. HALD, *SIAM J. Numer. Anal.* **24**, 538 (1987).
22. E. G. PUCKETT, *SIAM J. Sci. Stat. Comput.* **10**, 298 (1989).
23. A. MAYO, *SIAM J. Numer. Anal.* **21**, 285 (1984).
24. S. B. BADEN, in *Lecture Notes in Mathematics*, Vol. 1360 (Springer-Verlag, New York, 1988).
25. S. B. BADEN, Thesis, Lawrence Berkeley Laboratory Report No. LBL-23625, 1987 (unpublished).
26. C. R. ANDERSON, UCLA Mathematics Department, Los Angeles, private communication (1987).
27. R. COURANT AND D. HILBERT, *Methods of Mathematical Physics* (Interscience, New York, 1962).
28. M. B. PERLMAN, *J. Comput. Phys.* **59**, 200 (1985).
29. *Proceedings, of the Second International Symposium on Domain Decomposition Methods, Los Angeles, 1988*, edited by T. F. Chan, Roland Glowinski, Jacques Periaux, and Olof B. Widlund (SIAM, Philadelphia, 1989).
30. S. B. BADEN, in *Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing, Los Angeles, 1987*, edited by Garry Rodrigue (SIAM, Philadelphia, PA, 1988).