JOURNAL OF COMPUTATIONAL PHYSICS 62, 111–123 (1986)

A Method of Local Corrections for Computing the Velocity Field Due to a Distribution of Vortex Blobs

CHRISTOPHER R. ANDERSON*

Department of Mathematics and Department of Computer Science, Stanford University, Stanford, California, 94305

Received March 20, 1984; revised January 18, 1985

A computationally efficient method for computing the velocity field due to a distribution of vortex blobs is presented. The method requires fewer calculations than the straightforward vortex method velocity procedure and does not sacrifice the higher-order accuracy which can be achieved using higher-order vortex core functions.

INTRODUCTION

In the 2-dimensional vortex blob method [2, 7, 8, 16] the motion of an incompressible fluid is approximated by calculating the motion of a finite number of vortices of finite core, or vortex blobs. If we designate the centers of the vortices by $x_j(t)$ and their strength by ω_j then the evolution of $x_j(t)$ is calculated by solving the ordinary differential equations

$$\frac{dx_i(t)}{dt} = \sum_{j=1}^{N} K_\delta(x_i(t) - x_j(t))\,\omega_j \tag{1.1}$$

where K_δ is the velocity field induced by a single vortex blob, and $i = 1 \cdots N$. Equation (1.1) represents the fact that the velocity of each vortex is the sum of the velocities induced by all the other vortices. The convergence results given in [1, 3, 14] state that under certain assumptions about the core structure of the vortices, the motion of the centers of the vortices calculated using (1.1) closely approximates the exact motion of the fluid particles with which the centers are associated. It is also shown that the velocity field

$$u(x, t) = \sum_{j=1}^{N} K_\delta(x - x_j(t))\,\omega_j \tag{1.2}$$

* Partially supported by NSF Mathematical Sciences Postdoctoral Research Fellowship MCS 83-11685 and ONR Contract N00014-82-K-0335.

0021-9991/86 $3.00

581 62 1-8

closely approximates the exact velocity field. Theoretical results [1, 3] and numerical experiments [4, 18] indicate that the form of the core structure of the vortices is a critical factor in obtaining higher-order accuracy.

In the numerical solution of (1.1), it is necessary to evaluate the approximate velocity field (1.2) for each of the points $x_i(t)$. Therefore, for N vortices the method requires $O(N^2)$ operations per time step. The purpose of this paper is to present an algorithm for reducing the operation count of the method from $O(N^2)$ to approximately $O(M \log M) + O(N)$ where M is a constant independent of the number of vortices. Furthermore, if one is using a higher-order-accurate vortex method, such as those suggested in [4], then this higher-order accuracy is preserved by our algorithm.

The central idea of the method is to calculate the motion of the centers of the vortices by solving the set of equations

$$\frac{dx_i(t)}{dt} = \tilde{u}(x_i(t), t)$$

where $\tilde{u}(x, t)$ is an approximation to (1.2) and is chosen so that the evaluation of it at the points $x_i(t)$ requires less than $O(N^2)$ operations.

Our construction of a $\tilde{u}(x, t)$ is based upon the observation that the difference between the velocity field due to a point vortex and a vortex blob located at the same point in space becomes very small as one moves away from the centers of the vortices. This is evident if one compares the formula for the velocity field due to a point vortex and the formula for the velocity field of a vortex blob. In fact, if the vortex core is radially symmetric and has its support contained in a ball of radius δ, then the velocity fields of the blob and the point vortex differ only inside the ball of radius δ about their common center. (We will assume for the rest of this paper that the core functions are of compact support. For core functions that are not of compact support, but rapidly decaying, such as those presented in [4], the errors introduced by this difference are negligible and the basic results are the same.) In light of this observation, our method consists of constructing a velocity field due to a collection of point vortices and then modifying (correcting) the velocity field about the center of each vortex. The efficiency of our method is due to the fact that we use a technique for constructing and evaluating the velocity field due to a collection of point vortices that requires only $O(N) + O(M \log M)$ operations. Since the corrections to the velocity are confined to small regions about the center of each vortex, the correction step requires approximately $O(N)$ operations and we obtain a method of $O(M \log M) + O(N)$ operations. Also, we are enforcing the velocity field due to a specific core structure in the last step, so the method preserves the effects of the higher-order core functions.

The problem of finding a rapid way of calculating the velocity field due to a collection of point vortices has received much attention. One popular technique is the cloud in cell method (CIC), presented by Birdsall and Fuss [5], and its improvements, such as those presented in [6, 9, 12]. Furthermore, the idea of

locally correcting the velocity field that one obtains using a CIC-type method is similar to the central idea of the particle–particle/particle–mesh (PPPM) method described in [15]. For a review of PPPM methods see [13]. Although we follow closely some of the ideas presented previously, in our calculation of the velocity field due to a collection of point vortices we take advantage of some special features of the problem that have previously been neglected. Specifically, we use extensively the fact that the velocity field induced by a point vortex is harmonic away from the center of the vortex.

In the first section we present our method and in the second section we give the results of some computational experiments. Since the basic idea of the method is to locally correct the velocity field, we shall refer to our method as a method of local corrections.

Description of the Method

Before we begin the discussion of our method it is necessary to show how to construct the velocity field from a given distribution of vorticity.

If we denote the velocity by $\mathbf{u} = (u_1, u_2)$, then the incompressibility condition, $\partial u_1/\partial x + \partial u_2/\partial y = 0$, implies that there exists a stream function Ψ such that

$$u_1 = \frac{\partial \Psi}{\partial y}, \qquad u_2 = -\frac{\partial \Psi}{\partial x}. \tag{2.1}$$

Since the vorticity ω is defined as

$$\omega = \frac{\partial u_2}{\partial x} - \frac{\partial u_1}{\partial y}$$

then we have the following relation between the stream function and the vorticity:

$$\Delta \Psi = -\omega. \tag{2.2}$$

Equations (2.1) and (2.2) are used to determine the velocity due to a distribution of vorticity ω.

In the vortex blob method, the vorticity can be considered as a sum of blobs. If the blob or core function is given by f_δ, δ a parameter that determines the core size, then the assumed vorticity distribution is given by

$$\omega(x, t) = \sum_{j=1}^{N} f_\delta(x - x_j(t))\, \omega_j \tag{2.3}$$

where $x_j(t)$ is the center of the jth blob and ω_j is the strength of the jth blob. If we

use the fact that in 2 dimensions the fundamental solution of Laplaces equation is $G=(1/2\pi)\log r$, where $r=(x^2+y^2)^{1/2}$, then

$$\Psi = G * \left[\sum_{j=1}^{N} f_\delta(x-x_j(t))\,\omega_j \right]$$
$$= \sum_{j=1}^{N} (G * f_\delta)(x-x_j(t))\,\omega_j$$

where $*$ represents convolution. Therefore, if we use (2.1), we have

$$u_1 = \sum_{j=1}^{N} -\frac{\partial(G * f_\delta)}{\partial y}(x-x_j(t))\,\omega_j, \qquad u_2 = \sum_{j=1}^{N} \frac{\partial(G * f_\delta)}{\partial x}(x-x_j(t))\,\omega_j. \tag{2.4}$$

If we denote $K_{\delta_1} = -\partial(G * f_\delta)/\partial y$, $K_{\delta_2} = \partial(G * f_\delta)/\partial x$, and $K_\delta = (K_{\delta_1}, K_{\delta_2})$, we arrive at (1.2). Thus, the velocity field in the vortex method is that which is obtained by analytically solving (2.1) and (2.2) and assuming that the vorticity distribution is a sum of blobs (2.3). We remark that the formula for K_δ can be computed explicitly if f_δ depends on r alone. (See [4, 14].)

The first step in our method is to compute the velocity field due to a distribution of point vortices and evaluate this velocity field at the centers of all the other vortices. Specifically, we wish to compute

$$u(x_i(t), t) = \sum_{j=1}^{N} K(x_i(t)-x_j(t))\,\omega_j \tag{2.5}$$

where $K=(1/2\pi r^2)(-y, x)$ and $i=1\cdots N$. We remark that this is the velocity field that is due to a distribution of vorticity given by

$$\omega(x, t) = \sum_{j=1}^{N} \delta(x-x_j(t))\,\omega_j$$

where in this context δ is Dirac's delta function. For clarity of exposition we will only consider the problem of finding the first component, u_1, of the velocity field. The first step of our method consists of two parts:

(1a) Obtain values of the velocity at the nodes of a grid covering the computational domain.

(1b) Interpolate the values of the velocity from these grid nodes to the centers of the vortices.

Our procedure for step (1a) is most easily presented if we assume that we have only one pont vortex. We assume that our computational grid is centered at the origin and has mesh width h in each direction. Without loss of generality we also assume that the vortex is located in the region $[-h/2, h/2]\times[-h/2, h/2]$ about the

origin. (It need not be centered on a grid node.) For a collection of vortices one uses the principle of superposition and the linearity of the problem.

Our goal in step (1a) is to find the first component of the velocity field, u_1, at the nodes of the computational grid. Assume, for the sake of discussion, that we have the values of the velocity at the nodes and consider the grid function defined by

$$\Delta^h u_1(i_1 h, i_2 h) \tag{2.6}$$

where Δ^h is an approximation to the discrete Laplacian. Since u_1 is harmonic at all points but the vortex center, $\Delta^h u_1(i_1 h, i_2 h)$ will be a function whose values become small away from that point. In fact, if the discrete Laplacian is of order $O(h^k)$, then the values of $\Delta^h u_1(i_1 h, i_2 h)$ will be $O(h^k)$. Therefore, if one considers the function $g_D(i_1 h, i_2 h)$ defined by

$$\begin{aligned} g_D(i_1 h, i_2 h) &= \Delta^h u_1(i_1 h, i_2 h) \qquad \text{for} \quad i_1, i_2 \text{ s.t. } |i_1 h| < D \text{ and } |i_2 h| < D \\ &= 0 \qquad\qquad\qquad\quad \text{for} \quad i_1, i_2 \text{ s.t. } |i_1 h| \geqslant D \text{ or } |i_2 h| \geqslant D \end{aligned} \tag{2.7}$$

then g_D is an $O(h^k)$ approximation to $\Delta^h u_1(i_1 h, i_2 h)$.

One can obviously recover the values of u_1 by inverting the discrete Laplacian with (2.6) as a right-hand side. Since $g_D(i_1 h, i_2 h)$ is an approximation to (2.6), one can obtain an approximation, $\tilde{u}_1(i_1 h, i_2 h)$, to $u_1(i_1 h, i_2 h)$ by solving

$$\Delta^h \tilde{u}_1(i_1 h, i_2 h) = g_D(i_1 h, i_2 h) \tag{2.8}$$

and setting $\tilde{u}_1(i_1 h, i_2 h) = u_1(i_1 h, i_2 h)$ for points on the boundary of the computational grid. Choosing larger values of D and solving (2.8) will yield better and better approximations to $u_1(i_1 h, i_2 h)$. Also for a fixed D, increasing the accuracy of the Laplacian, and hence the accuracy with which g_D approximates (2.6), will increase the accuracy of the approximation of $\tilde{u}_1$.

In our calculations, we used the above scheme to generate approximations to the velocity at the grid nodes. A 9-point "box" approximation to the Laplacian [11] was used. Since we are approximating the Laplacian of a harmonic function, the "box" approximation is $O(h^4)$. To obtain the values of u_1 necessary to construct g_D and the boundary values for $\tilde{u}_1$, we used (2.4). In the second section we present computational results concerning the accuracy of the velocities determined in this way for various values of D. We mention that this procedure for obtaining the velocities is similar in approach to that employed by Mayo [17] for obtaining the potential due to a charge distribution on the boundary of an irregular domain.

The next part is to interpolate the velocity field to the centers of the other vortices. This is essentially the problem of interpolating the velocities to intragrid points. Any of the family of interpolation schemes presented in [13] could be used in this step. However, one can use a special feature of the velocity field due to a point vortex to obtain high-order-accurate interpolation formulas. Specifically,

away from the center of the vortex the two components of the velocity field form the real and imaginary parts of a complex analytic function given by

$$F(z) = u_1 - iu_2. \tag{2.9}$$

Here we are identifying the complex plane with the x-y plane, i.e., $z = x + iy$. That the function defined by (2.9) is analytic follows from the fact that the velocity field satisfies the equations

$$\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} = 0, \qquad \frac{\partial u_2}{\partial x} - \frac{\partial u_1}{\partial y} = 0$$

i.e., the Cauchy–Riemann equations for the function (2.9). The usefulness of using the complex representation of the velocity field is that one can obtain an interpolation formula for the velocity field by taking the real and imaginary parts of any interpolation formula for complex functions. Interpolating a function in the complex plane is analogous to the interpolation of real valued functions of one variable. In particular, the Lagrangian interpolation formula holds (see [10]). Given n points in the complex plane, z_i, $i = 1, n$, an interpolation formula for the function $F(z)$ is

$$F(z) \cong \sum_{j=1}^{n} \left(\prod_{i \neq j} \frac{(z - z_i)}{(z_j - z_i)} \right) F(z_j). \tag{2.10}$$

If the points z_i are at adjacent nodes of a grid with sides of length h, then the accuracy of the interpolation formula (2.10) is approximately $O(h^n)$. Thus by using the real and imaginary parts of the interpolation formula (2.10), we are able to determine each component of the velocity field to $O(h^n)$ accuracy with just n points. For example, with four points one gets fourth-order accuracy. This should be contrasted with bilinear interpolation applied to each component separately. Such an interpolation also uses four points but is only second-order accurate.

We remark that very near the center of a vortex, the interpolation formula will not do well because of the singularity in the complex function defined by a point vortex. The constants in the error estimate for a formula of the type (2.10) are inversely proportional to the distance between the interpolation points and the nearest singularity. Fortunately, this inaccuracy in the interpolation can be tolerated, since the velocity field is going to be corrected in the neighborhood of the center of the vortex in the second step of the method. This particular error is not solely related to the interpolation formula (2.10); we would also expect large errors if we used other interpolation formulas at points near the center of the vortex. This completes the description of the first step.

The second step in the method is to locally correct the velocities obtained in the first step. For a given vortex, the correction consists of subtracting the component of its velocity which is due to interpolating the velocity field of nearby vortices and

then adding the correct velocity due to the nearby vortices, i.e., the correction step for the jth vortex is defined by

$$\begin{aligned}\text{velocity correction for } j\text{th vortex} \\ = \begin{array}{c}\text{exact velocity contribution} \\ \text{due to vortices such} \\ \text{that } |x_i(t)-x_j(t)| \leqslant C\end{array} - \begin{array}{c}\text{interpolated velocity contribution} \\ \text{due to vortices such} \\ \text{that } |x_i(t)-x_j(t)| \leqslant C\end{array}\end{aligned} \tag{2.11}$$

and C is a parameter to be determined.

The exact velocity contribution is computed using (2.4). To calculate the interpolated velocity contribution, one first finds the values of the velocity induced by nearby vortices at the set of nodes used in the interpolation formula for the jth vortex. These velocities are evaluated using (2.4). The interpolation formula for the jth vortex is applied to these node velocities and the result is the local contribution to the interpolated velocity field.

There is one aspect of this last step that requires some care. In particular, one must determine which vortices are near to, and which are far from, a given vortex. If one determines this by computing the distance between a vortex and all its neighbors, then one increases the operation count of the method to $O(N^2)$. A solution to this difficulty is a link-listing technique which is described in [15]. The basic idea is to cover the computational domain with a grid of mesh width k, a "chaining mesh." (The width h used to calculate the velocities need not be equal to k.) Each vortex is given a tag which indicates which grid box of the chaining mesh the vortex resides in. The tags, and hence the vortices, are sorted, and all the vortices with the same tag are linked together with a linked list. To determine which vortices are within a given radius of a vortex reduces to finding which grid boxes are within that radius. Once the grid boxes are found, then using the linked list of vortices for those boxes, one knows the vortices that are within the radius of the given vortex. A description, more detailed than that in [15], is given in [13].

In summary, to compute the velocity of each of the vortices we consider the computational domain covered by a grid of mesh spacing h. We then use a fast elliptic solver to find the solution to

$$\Delta^h \tilde{u}_1(i_1 h, i_2 h) = \sum_{j=1}^{N} g_{D_j} \tag{2.12}$$

where g_{D_j} is the grid function for the jth vortex defined by (2.7). Similarly, we find an approximation to the second component of the velocity field, $\tilde{u}_2$. We then interpolate $\tilde{u}_1$ and $\tilde{u}_2$ using an interpolation formula of the type (2.10), and obtain an approximation to the velocity at the vortices. In order to improve the accuracy of the approximate velocities we use formula (2.11) to correct the velocity for each vortex. Accumulating the right-hand sides of (2.12) requires $O(N)$ operations and the correction calculation requires approximately $O(N)$ operations. If we assume that the discrete Laplacian in (2.12) is solved in $O(M \log M)$ operations (M = the

TABLE I

D	Relative error in u velocity
$1h$	2.52×10^{-1}
$2h$	6.06×10^{-4}
$3h$	1.99×10^{-5}
$4h$	3.16×10^{-6}

number of nodes of the grid of mesh width h in the computational domain), then the total operation count is $O(N) + O(M \log M)$.

Computational Examples

To test the effect of the parameter D in g_D defined by (2.7), we calculated the velocity field due to a single point vortex of unit strength. We solved (2.12) for various values of D and compared it to the exact velocity field computed using (2.4). We measured the error in the discrete L^2 norm, i.e., the norm defined by

$$\|u_1\|^2 = \sum_{i_1, i_2} u_1(i_1 h, i_2 h)^2 h^2$$

where the sum is over all grid points in the computational domain. The grid used was a 30×30 grid with mesh width $h = 0.1$. The results of this calculation are presented in Table I. Clearly, D need not be chosen large to achieve accuracy in the computed velocity.

To illustrate the combined effect of the parameter D and the interpolation error, in Fig. 1 we plot the logarithm of the absolute error in the first component of the velocity field along a ray emanating from the center of the vortex and parallel to the x-axis. The different graphs represent the error for different values of D. The grid points labeled are those corresponding to the computational grid used in the discretization of (2.12). The velocities between the grid nodes were obtained using a 5-point interpolation formula given by (2.10).

From the figure we see that the error is concentrated near the center of the vortex. This is to be expected since it is near this point that the error in the interpolation formula becomes significant.

The effect of correcting the velocity field about the center of the vortex is exhibited in Fig. 2. Here we present graphs of the errors for the same problem as that used to obtain Fig. 1, but with the correction calculation preformed. As expected, the correction calculation reduces the errors in the velocity field about the center of the vortex.

Errors in velocity along other rays varied in a qualitatively and quantitatively similar fashion. The only noticeable difference in results occurred in the analog of Fig. 1. For other ray directions, the rapid decrease in error at distances which were

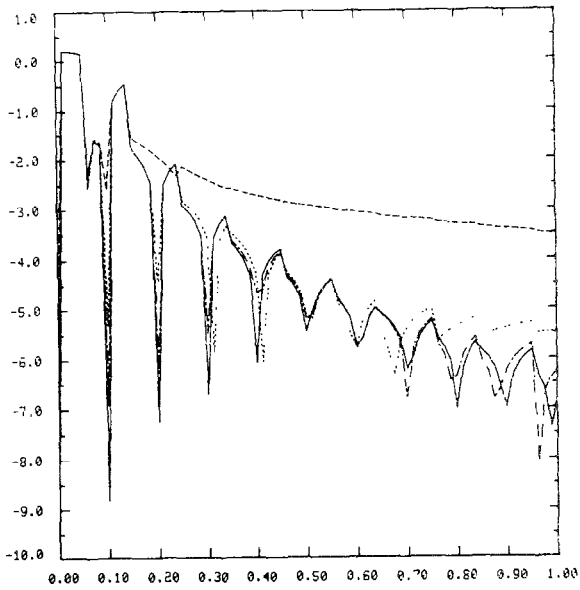

FIG. 1. Log_{10} of the absolute error in the first component of the velocity as a function of distance away from the center of a vortex blob. The curves correspond to different choices of the parameter D in (2.7). $D = h$: – – –; $D = 2h$: ···; $D = 3h$: ——; $D = 4h$: · – ·.

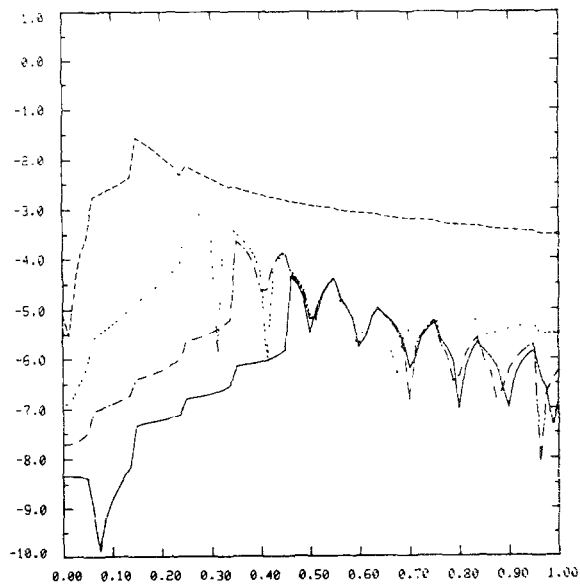

FIG. 2. Log_{10} of the absolute error in the first component of the velocity as a function of the distance away from the center of a vortex blob, when the correction calculations are preformed. The curves correspond to different choices of the parameter C defined by (2.11). $C = h$: – – –; $C = 2h$: ···; $C = 3h$: ——; $C = 4h$: · – ·.

integral multiples of h did not occur. This is to be expected, since a ray along the x-axis crosses nodes of the interpolation formula, and at these points the error of the interpolation formula is zero. (The error is only due to that of approximating the velocities at the grid nodes.)

To determine if our method is capable of preserving the effects of using higher-order-accurate blobs, such as those presented in [4], we compared two approximate solutions obtained with the vortex method on the same test problem. In one approximation, (1.2) was used to calculate the velocity field in the vortex method, and in the other approximation, our method was used. The test problem consisted of finding the solution of Euler's equations with an initial vorticity distribution given by

$$\begin{aligned} \omega(x, 0) &= \alpha(1-(2r)^2)^7, & r \leqslant \tfrac{1}{2} \\ &= 0, & r \geqslant \tfrac{1}{2} \end{aligned}$$

with $r=(x^2+y^2)^{1/2}$ and $\alpha=4\pi$. Since this initial vorticity distribution is radially symmetric, one can find an exact solution to this problem with which to compare the computed solutions. We tested two different blob functions:

$$\begin{aligned} f_\delta &= \frac{1}{2\pi r\delta}, & r < \delta \\ &= 0, & r \geqslant \delta \end{aligned} \tag{3.1}$$

$$f_\delta = \frac{e^{-(r/\delta)^2}}{\pi\delta^2}\left[2-\frac{r^2}{\delta^2}\right]. \tag{3.2}$$

The first function was suggested by Chorin in [7] and the second is a higher-order blob function suggested by Beale and Majda in [4]. To implement the vortex method we initially placed the vortices on the nodes of a grid of mesh width $\Delta x = 0.043$. Designating the nodes by x_i, we set each of the strengths, ω_i, equal to $\Delta x^2\, \omega(x_i, 0)$. We chose the core parameter to be $\delta = (\Delta x)^{0.95}$. The ordinary differential equations (1.1) were integrated using a 4th-order Runge–Kutta scheme, with a time step $\Delta t = 0.1$. This time step was small enough so that a decrease in step size had an insignificant effect on the results. We calculated the solution of the problem to time $t = 1.0$. This time corresponded to a maximal point rotation of 2π radians, i.e., one revolution. In the implementation of our method for calculating the velocity we used a 20×20 grid with mesh width $h = 0.1$ in each direction, and the parameter D of (2.7) was equal to the correction distance C of (2.11).

To measure the error we compared the positions of the centers of the vortex blobs with the positions of the material points associated with them at time $t = 0$ and which were moved according to the exact solution. We defined the error to be

$$(\Delta x)\left(\sum_{j=1}^{N} (\text{computed } x_j(t) - \text{exact } x_j(t))^2\right)^{1/2}$$

TABLE II
Error in Particle Positions at time $t = 1$

	Formula (1.2)	Local corrections		
		$C = 1h$	$C = 2h$	$C = 3h$
Chorin function (3.1)	8.36×10^{-3}	8.59×10^{-3}	8.35×10^{-3}	8.36×10^{-3}
Beale–Majda function (3.2)	4.15×10^{-3}	4.32×10^{-3}	4.14×10^{-3}	4.14×10^{-3}

where the number of vortices, N, is the number of nodes of mesh width Δx in the support of the initial vorticity distribution. The results of the computations for both of the blob functions and for several values of the correction radius C are given in Table II.

The results demonstrate that our method is capable of preserving the effects of higher-order-accurate vortex blobs. We also see that one need not choose the correction distance very large to obtain an accurate solution. It appears that the choice $C = D = h$ is sufficient to obtain an accurate approximation to the velocity.

Finally, as a measure of the computational efficiency of our method we compared the amount of time spent in using (1.2) and our method to compute the velocity of various numbers of vortices which were uniformly distributed in the region $[0, 1] \times [0, 1]$. In Fig. 3 we have plotted the CPU time (in CDC 7600 seconds) versus the number of vortices for both methods of calculation. As expected, the amount of time it takes to do the calculation using (1.2) is proportional to N^2. For our method we see that the amount of time is nearly linear with N, while increasing the number of corrections shifts the whole curve upwards. Of particular interest is the number of vortices for which the amount of computational time for both methods is the same. This number is approximately 350, 500, and 825 vortices for C equal to h, $2h$, and $3h$, respectively. Clearly, for numbers of vortices greater than this number it is more advantageous to use our method instead of (1.2). Although this number is dependent upon this specific problem as well as the choice of parameters D, C, and the mesh size, we believe that these results give reasonable estimates of it.

In summary, for computations that involve more than a few hundred vortices our method reduces the amount of computational time used in the vortex method without sacrificing too much accuracy. Our method is also capable of preserving the effects of using higher-order-accurate vortex blobs. The method described here is useful for calculating the velocity field due to a distribution of vortex "blobs" in an unbounded domain. A simple extension of the method would make it applicable to problems in which periodic or no-flux boundary conditions are present. (Essentially one needs to change the boundary conditions in the fast solver used to invert the discrete Laplacian in (2.7).) The use of other boundary conditions can be expec-

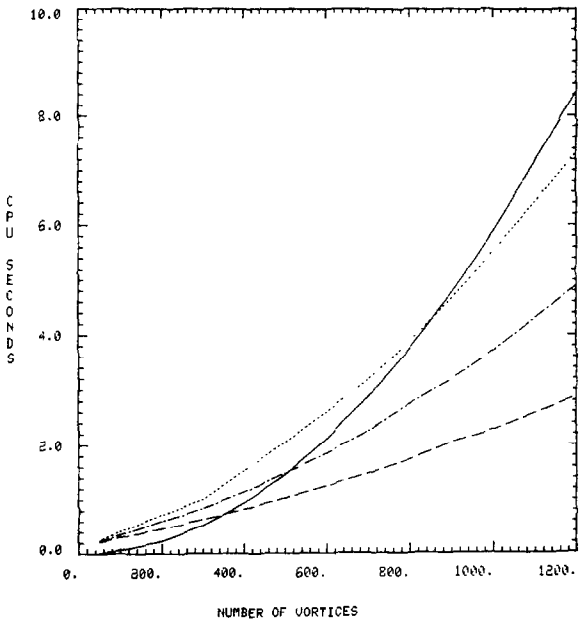

FIG. 3. CPU seconds versus the number of vortices for different values of the correction parameter C defined by (2.11). Formula (1.1): ——; local corrections, $C = h$: – – –; local corrections, $C = 2h$: · – ·; local corrections, $C = 3h$: ···.

ted to decrease the amount of computational time since the velocities on the boundaries need not be calculated in these instances. We believe that many of the ideas presented here can be used in other situations; in particular they might be incorporated in existing fast-velocity algorithms (such as particle in cell methods), or be used in algorithms for accelerating vortex methods in three dimensions.

ACKNOWLEDGMENTS

I would like to thank Professor Alexandre Chorin and Claude Greengard for their helpful comments on an early version of this work.

REFERENCES

1. C. R. ANDERSON AND C. GREENGARD, *SIAM J. Numer. Anal.* **22** (1985), 413.
2. J. T. BEALE AND A. J. MAJDA, "Transonic, Shock, and Multidimensional Flows: Advances in Scientific Computing," p. 329, Academic Press, New York, 1982.
3. J. T. BEALE AND A. J. MAJDA, *Math. Comp.* **39** (1982), 59.
4. J. T. BEALE AND A. J. MAJDA, *J. Comput. Phys.*, in press.
5. C. K. BIRDSALL AND D. FUSS, *J. Comput. Phys.* **3** (1969), 494.
6. O. BUMEMAN, *J. Comput. Phys.* **11** (1973), 250.

7. A. J. CHORIN, *J. Fluid Mech.* **57** (1973), 785.
8. A. J. CHORIN, *SIAM J. Sci. Statist. Comput.* **1** (1980), 1.
9. J. P. CHRISTIANSEN, *J. Comput. Phys.* **13** (1973), 363.
10. J. P. DAVIS, "Interpolation and Approximation," Dover, New York, 1975.
11. G. DAHLQUIST AND A. BJORCK, "Numerical Methods," Prentice–Hall, Englewood Cliffs, N.J., 1974.
12. J. W. EASTWOOD AND R. W. HOCKNEY, *J. Comput. Phys.* **16** (1974), 342.
13. J. W. EASTWOOD AND R. W. HOCKNEY, "Computer Simulation Using Particles," McGraw–Hill, New York, 1981.
14. O. HALD, *SIAM J. Numer. Anal.* **16** (1979), 726.
15. R. W. HOCKNEY, S. P. GOEL, AND J. W. EASTWOOD, *J. Comput. Phys.* **14** (1974), 148.
16. A. LEONARD, *J. Comput. Phys.* **37** (1980), 289.
17. A. MAYO, *SIAM J. Numer. Anal.* **21** (1984), 285.
18. M. PERLMAN, Ph.D. thesis, University of California at Berkeley, 1983.