A FAST VORTEX CODE FOR COMPUTING 2-D FLOW IN A BOX

88-3605-CP

Scott B. Baden†
Lawrence Berkeley Laboratory
Berkeley, California 94720

Elbridge Gerry Puckett‡
Lawrence Livermore National Laboratory
Livermore, California 94550

ABSTRACT

We present a fast, accurate hybrid vortex method for computing incompressible, viscous flow at large Reynolds numbers in a two-dimensional bounded domain. The random vortex method is used to model the flow away from the boundary and the vortex sheet method is used to model the flow near the boundary. Our implementation of these methods exploits the localized nature of interactions among vortex elements in each of the respective regions of the domain. A local corrections approximation is used to accelerate the velocity computation in the interior. It is substantially faster than other methods of comparable accuracy and can economically handle tens of thousands of vortex elements. We evaluate the method on the flow in a box due to a central stationary vortex. The running time for this problem is roughly linear in the number of vortex elements and results are in good qualitative agreement with other numerical solutions of the same problem.

§1 INTRODUCTION

The hybrid vortex sheet-random vortex method was introduced by Chorin [9-11] to compute incompressible, viscous flow at large Reynolds numbers. We employ recent innovations to speed up the computation in two dimensions. Most notable is the method of local corrections [1]; it is an approximation that replaces the $O(N^2)$ calculation customarily used to evaluate vortex blob velocities by a much faster one. Our code is capable of economically computing with large numbers of vortex elements, and allows us to perform detailed flow visualizations in reasonable amounts of time.

We test our code on the flow in the unit box driven by a single vortex fixed at the origin. This problem has previously been studied with the aid of another hybrid vortex method by Sethian [20] who used the $O(N^2)$ method for computing the vortex velocities. Our results are in good qualitative agreement with his work and, for the computations presented here, the cost of our method appears to be roughly linear in the number of vortices.

Hybrid vortex methods have also been applied to the flow past a circular cylinder [7, 24], driven cavity flow [8], flow past a backward facing step [14, 21], wind flow over a building [22], stability of the boundary layer [11], and the Falkner-Skan boundary layer flow [23]. See Leonard's survey [18] for a review of vortex methods.

† 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.

‡ Work done under the auspices of the U. S. Department of Energy at Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

§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 [9] is used to solve the incompressible Navier-Stokes equations within Ω_I, the vortex sheet method [10] 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 2-D, incompressible Navier-Stokes equations. In vorticity form these equations are:

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

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

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

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-c) are Euler's equations:

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

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

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

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

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

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-e). 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 its *strength*, K_σ the *cutoff function*, and σ 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 [16] and Beale and Majda [6] for a discussion of different kinds of cutoffs and their effect on accuracy. We use the cutoff proposed by Chorin [9]:

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

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

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

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 $\partial\Omega$ for $\tilde{\psi}_p^k$ and then differentiating per (2.2e). There are several ways to obtain approximations to $\tilde{\mathbf{u}}_p^k$. We discuss our choice after the description of the method of local corrections in §3.2 below.

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

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

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 that does two or more velocity evaluations per time step. We employ a time step constraint described in §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:

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

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

where τ is a tangent vector 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

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

where the η_j are independent, Gaussian distributed random numbers 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 vortex sheet method to cancel the tangential velocity on $\partial\Omega$ induced by the blobs with positions $\mathbf{x}_j^{k+1}$. We next 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:

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

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

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

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

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

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]x[0,ε]. 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. [8]) 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 [10],

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

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

$$\tilde{v}^k(x,y) = -\partial_x U_\infty(x,t)\,y$$

$$-\frac{1}{h}\sum_j \xi_j \left[b_l(x + \frac{h}{2} - x_j^k) - b_l(x - \frac{h}{2} - x_j^k) \right] Min(y, y_j^k).$$

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 ξ_{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 $-sign(u_i)\xi_{max}$, 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, and 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 ξ_{max}. We do not create sheets at a_i if $|u_i| \le \xi_{max}$ and hence the no-slip boundary condition is satisfied at a_i only up to order ξ_{max}. Other workers (e.g. [7, 8, 10, 11]) create sheets at the ith gridpoint whenever $|u_i| \ge \xi_{min}$ for some $\xi_{min} < \xi_{max}$ such that the sum of the strengths of these sheets exactly cancels u_i. However it has been shown [19] 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 thereby improving the economy of the 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 ξ_{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 [11] and Sethian [20] we set $\sigma = \pi/l$. The reader should consult [11, 19, 21, 24] 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:

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

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 a single time 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 $\tilde{\mathbf{u}}^k$ over the centers of all the vortices. We then adjust Δt accordingly before moving the blobs.

§3 OUR IMPLEMENTATION OF A FAST VORTEX METHOD

3.1. The Method of Local Corrections Traditionally, vortex blob 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 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 lumped or averaged with an relatively inexpensive computation. Only interactions involving nearby vortices need to be computed directly, and these account for only a small fraction (typically 5%) of the $N(N-1)$ interactions computed by the direct method.

We use a strategy based on the above observation, known as the method of local corrections [1]. It is similar to the particle-particle, particle-mesh algorithm of Hockney et al. [17] and more accurate than Christiansen's vortex-in-cell [12], since the latter doesn't compute close interactions directly. A novel feature of this method is that it 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. (In this sense it is similar to Rokhlin and Greengard's multipole expansion method [15].) This allows one to take advantage of fourth order interpolation formulas for harmonic functions. 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 12848 vortices, distributed evenly among two patches of constant vorticity, in under 7 seconds on the Cray X-MP; the direct method takes 56 seconds. The amount of speed up one obtains with the method of local corrections depends on the distribution of the vortices in the computational domain and hence is problem dependent. See Baden [5] for further discussion on the speed and accuracy of this method.

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 [5]. The method of local corrections in predicated on the assumption that the vortex blobs behave like point vortices at distances greater than Ch from their centers. Thus, we must ensure that $\sigma \le Ch$.

In the following discussion we omit mention of the time step k for notational convenience. The algorithm first computes 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,

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

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

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

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

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 $D \le C$. 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,b) 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 space that is C boxes thick on each side, called the *correction neighborhood*, and an interpolation stencil. (We use a 5-point stencil; the interpolation procedure is accurate to fourth 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 interpolating 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.

3.2. The Potential Flow In our solution of the potential flow problem we employ a modified method of images scheme suggested by Anderson [2]. 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 [13, pg 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 image vortices that are within one correction distance of $\partial\Omega$ in the computation of $\tilde{\mathbf{u}}_f$ and hence, in the computation of $\tilde{\mathbf{u}}_f^h$. These images must be included because their influence on nearby vortices on the other side of the boundary cannot be accurately represented in a finite difference solution of $\tilde{\mathbf{u}}_p$. This is important 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 3 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 opposite strengths 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 $\tilde{\mathbf{u}}_p^h \cdot \boldsymbol{\tau}$. However, we compute the normal velocity on the boundary $\tilde{\mathbf{u}}_p^h \cdot \mathbf{n}$ ($= -\tilde{\mathbf{u}}_f \cdot \mathbf{n}$) 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. (This modification was first suggested by Chorin [10].) 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 gridpoints a_i on the boundary at which sheets are created. The ith bin extends over $a_i - l/2 \le x < a_i - l/2$ and $0 \le y < \infty$. (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.

§4 COMPUTATIONAL RESULTS

We present results for the 'spindown' problem investigated by Sethian [20]. In this problem a single vortex is fixed at the center of the box, 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 $\xi_{max} = 6.25\times10^{-3}$; and the sheet length $l = 0.1$. The initial time step was $\Delta t_0 = 0.05$. As described in §2.3 the cutoff radius was chosen to be $\sigma = l/\pi$. In the interior we use 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 [19-20]) we use only the first order Euler method (2.10) in the sheet layer.

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. However, note that we used roughly five to ten times as many computational elements as in that study, each with one-eighth the strength.

During the initial time step 3760 vortex sheets were created. During the second time step 109 sheets left the sheet layer and became blobs. The maximum (componentwise) velocity of these blobs was 0.977, so the time step was reduced to 0.018. The time step Δt slowly decreased throughout the run and attained a minimum value of 0.011. The run took 320 time steps and consumed 4107 seconds (68.4 minutes) of CPU time on a CRAY X-MP. Of this, only 2.2% of the time was spent in the sheet calculation. At the end of the run there are 13094 blobs, 7161 images, and 4034 sheets. At each time step the number of images was roughly half of the number of blobs.

Figure 3 shows that the total number of vortex blobs and their images steadily increases with time but that the number of sheets is roughly constant. Figure 4 shows that the computational cost is roughly a linear function of the number of vortex blobs. The times shown in Figure 4 are the *total* cost per time step. In addition to the vortex blob velocity evaluations, the times include all sources of overhead such as: the potential flow, the random walks, and the sheet computation. In particular, this includes a third vortex blob velocity evaluation at each of the $\mathbf{x}_j$ for the purpose of computing U_∞. (This third velocity evaluation could be eliminated by a redesign of the algorithm.)

Figure 5 shows the computational cost per time step and compares it with an estimated cost of using the direct method to compute blob velocities. The estimated speedup (per time step) of the local corrections algorithm increases with time; by the end of the run it is roughly 10. The speedup averaged over the entire run is about 8. 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 0.4 μsec on the CRAY X-MP. Using the statistics obtained from our trial run we determined that, if the

direct method had been used for the run shown in Figures 1 and 2, then it would have computed a total of 8.53×10^{10} interactions, at a cost of 3.41×10^{4} seconds 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 an additional 5% to the running time of the computation. Thus, the running time of the vortex blob velocity evaluations gives a rough estimate of the overall running time of the direct method-based calculation. We arrive at the estimated speedup of 8.3 by dividing the 4107 CPU seconds for the local corrections-based code with the estimated time of 3.41×10^{4} CPU seconds for the direct method-based code.

§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. Our test run modeling the flow induced by a central stationary vortex in a square box is in good qualitative agreement with the earlier results of Sethian [20]. A typical run of the type shown here (beginning with no vortex elements, running for 320 time steps, and ending with 13094 blobs, 7161 images, and 4034 sheets) consumed roughly 68 minutes of CPU time on a CRAY X-MP. The run would take at least 8 times longer to complete if vortex blob velocities were evaluated using the direct method instead of the method of local corrections. Moreover, the speedup improves as the number of computational elements increases. Finally, we have shown that the cost of our method is effectively linear in the number of vortices. (The algorithm is presumably $O(N \log N)$, but $\log N$ is, in practice, bounded by 6.) This represents a substantial improvement in speed over previous implementations of the hybrid vortex sheet-random vortex algorithm.

So far our efforts to assess the accuracy of the method of local corrections have shown no appreciable loss of precision. Future work will include an application of this method to the flow in a driven cavity. Our code can be readily modified to execute in parallel on a multiprocessor like the Cray X-MP, as discussed in [3,4].

§6 REFERENCES

1. C. R. Anderson, "A Method of Local Corrections for Computing the Velocity Field Due to a Distribution of Vortex Blobs," *J. Comput. Phys. 62*(1986), pp. 111-123.
2. C. R. Anderson, private communications.
3. S. B. Baden, "Run-Time Partitioning of Scientific Continuum Calculations Running On Multiprocessors," LBL-23625, Lawrence Berkeley Laboratory, June 1987. (Ph. D. Dissertation in the Computer Science Division at the U. of Calif., Berkeley, # 87/366)).
4. S. B. Baden, "Programming Abstractions for Run-Time Partitioning of Scientific Continuum Calculations Running on Multiprocessors," *Proc. of the Third SIAM Conference on Parallel Processing for Scientific Computing*, Los Angeles, California, December 1-4, 1987.
5. S. B. Baden, "Very Large Vortex Calculations in Two Dimensions," in *Lecture Notes in Mathematics*, Springer-Verlag, New York, 1988. Proceedings from the UCLA Workshop on Vortex Methods, Los Angeles, Calif., May 20-22, 1987.
6. J. T. Beale and A. Majda, "The Design and Numerical Analysis of Vortex Methods," PAM-48, Center for Pure and Applied Mathematics, University of California, Berkeley, 1981.
7. A. Y. Cheer, "Unsteady Separated Wake Behind an Impulsively Started Cylinder in Slightly Viscous Fluid," *manuscript*, U. C. Davis, 1986.
8. Y. Choi, J. A. C. Humphrey and F. S. Sherman, "Random Vortex Simulation of Transient Wall-Driven Flow in a Rectangular Enclosure," *submitted to J. Comput. Phys.*, 1986.
9. A. J. Chorin, "Numerical Study of Slightly Viscous Flow," *J. Fluid Mech. 57*(1973), pp. 785-796.
10. A. J. Chorin, "Vortex Sheet Approximation of Boundary Layers," *J. of Comput. Phys 27*,3 (June 1978), pp. 428-442.
11. A. J. Chorin, "Vortex Models and Boundary Layer Instability," *SIAM J. Sci. Stat. Comput. 1*,1 (March 1980), pp. 1-21.
12. J. P. Christiansen, "Numerical Simulation of Hydrodynamics by the Method of Point Vortices," *J. Comput. Phys. 13*(1973), pp. 363-379.
13. R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Interscience, New York, 1962.
14. A. F. Ghoniem, A. J. Chorin and A. K. Oppenheim, "Numerical Modeling of Turbulent Flow in a Combustion Tunnel," *Philos. Trans. Roy. Soc. London A304*(1982), pp. 303-325.
15. L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations," YALEU/DCS/RR-459, Yale Univ., Dept. of Computer Science, April 1986.
16. O. Hald, "Convergence of Vortex Methods, II," *SIAM J. Numer. Anal 16*(1979), pp. 726-755.
17. R. W. Hockney, S. P. Goel and J. W. Eastwood, *J. Comput. Phys. 14*(1974), pp. 148.
18. A. Leonard, "Vortex Methods for Flow Simulation," *J. Comput. Phys. 37*(1980), pp. 289-335.
19. E. G. Puckett, "A Study of the Vortex Sheet Method and Its Rate of Convergence," *Siam J. of Sci. and Stat. Comp.*, (to appear).
20. J. Sethian, "Turbulent Combustion in Open and Closed Vessels," *J. Comput. Phys. 54*,3 (June 1984), pp. 425-456.
21. J. A. Sethian and A. F. Ghoniem, "Validation Study of Vortex Methods," *J. Comput. Phys. 74*(1988), pp. 283-317.
22. D. M. Summers, T. Hanson and C. B. Wilson, "A Random Vortex Simulation of Wind-Flow Over a Building," *Int. J. for Num. Meth. in Fluids 5*(1985), pp. 849-871.
23. D. M. Summers, "A Random Vortex Simulation of Falkner-Skan Boundary Layer Flow," *submited to J. C. P.*, 1987.
24. E. C. Tiemroth, *The Simulation of the Viscous Flow Around a Cylinder by the Random Vortex Method*, Dept. of Naval Architecture and Offshore Engineering, U. of Calif., Berkeley, California, May 1986. Ph. D. Dissertation.

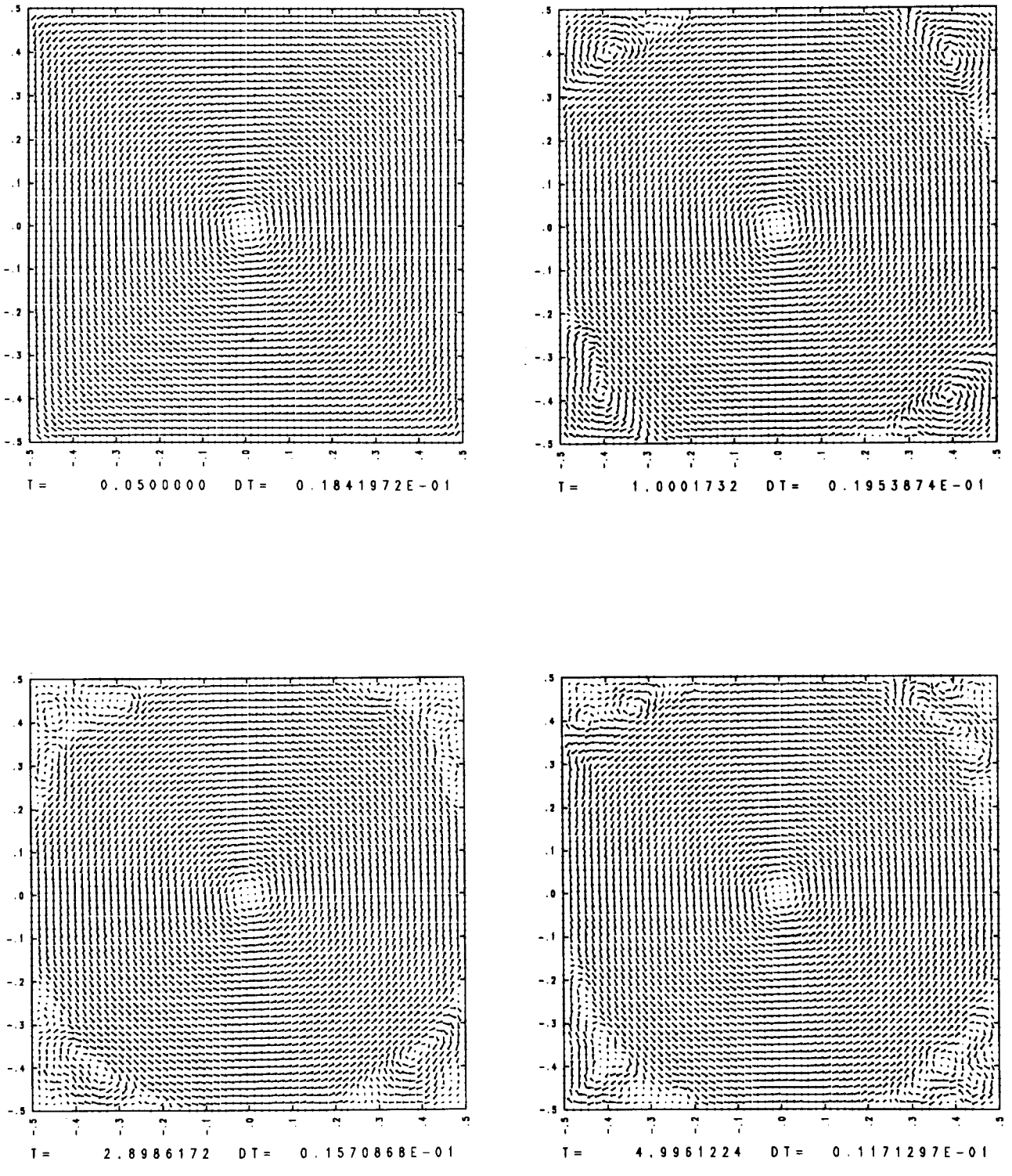

Figure 1. Vector velocity plots clearly show the formation of counterrotating eddies. To emphasize the details near the walls, the vector lengths have been scaled so that the vectors in the center region have a constant length, and so that the vectors near the wall have been somewhat enlarged. A single stationary vortex induces a counterclockwise flow.

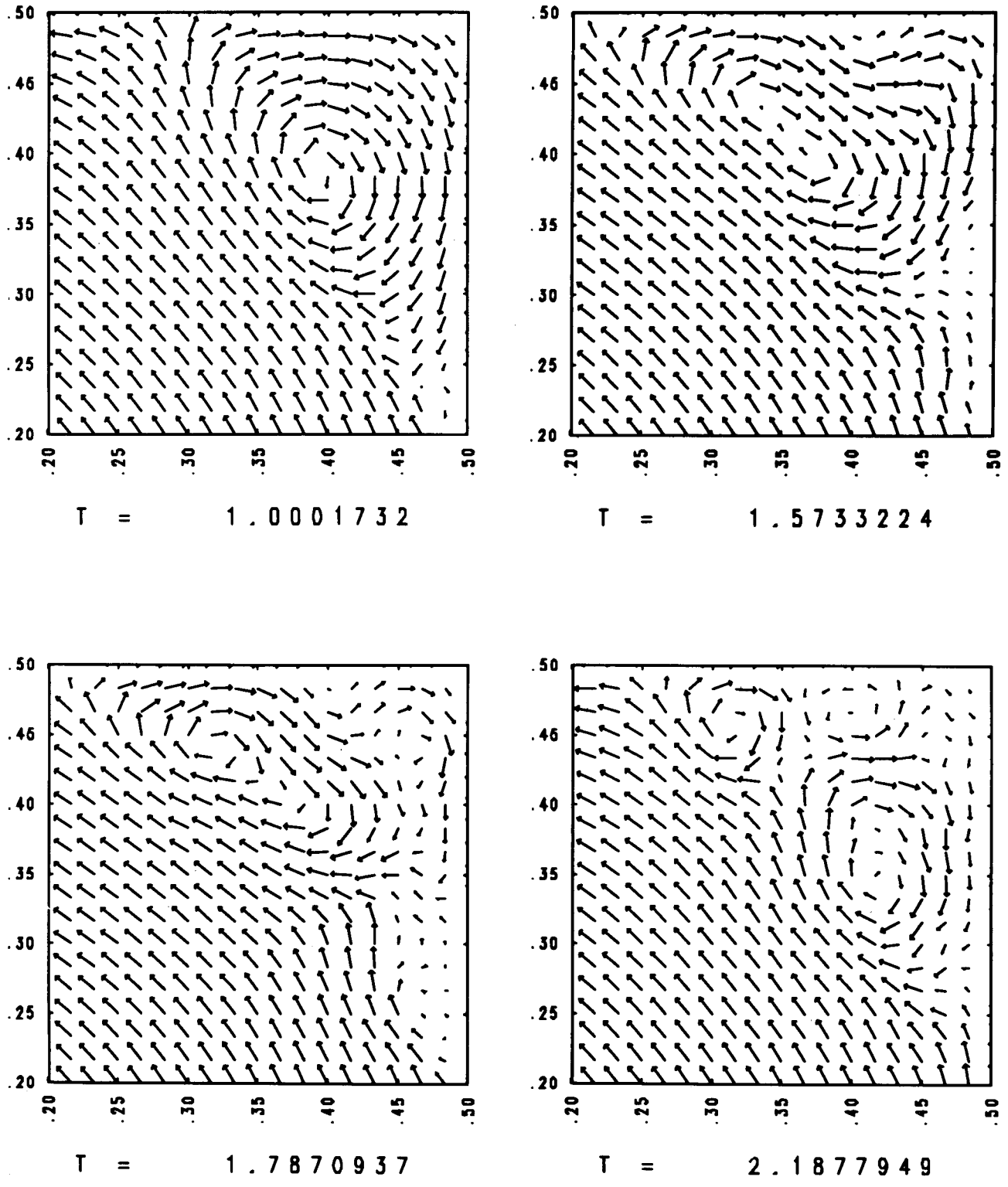

Figure 2. These plots show finer detail than in Figure 1, but only for the upper right hand corner of the domain. The figure continues on the next page.

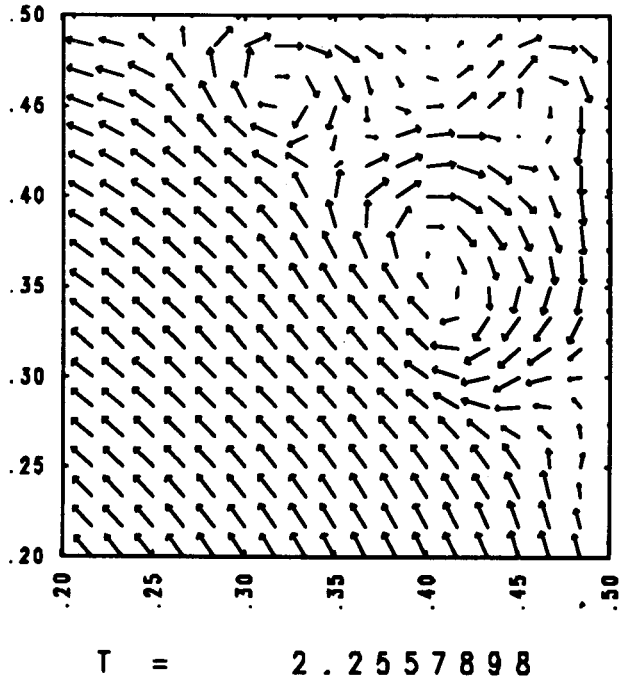

Figure 2 (continued).

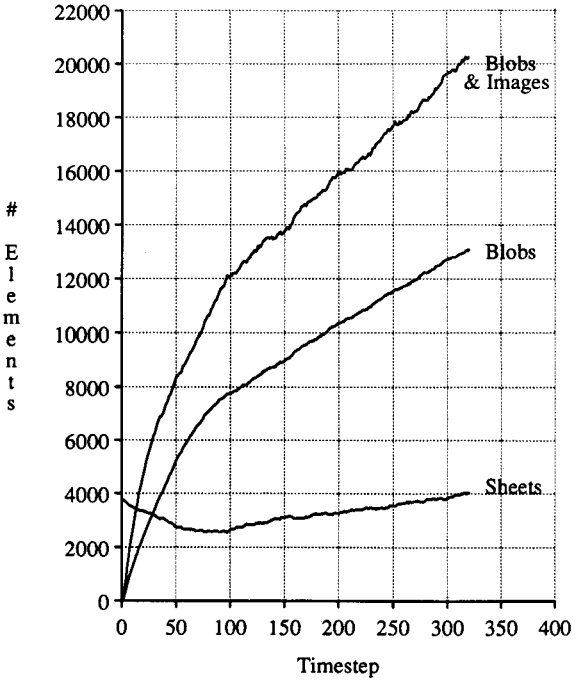

Figure 3. The number of vortex elements varies as a function of time. The vortex blobs steadily increase (second curve), but the number of vortex sheets remains relatively stable (bottom). The top curve plots the number of blobs as well as their images.

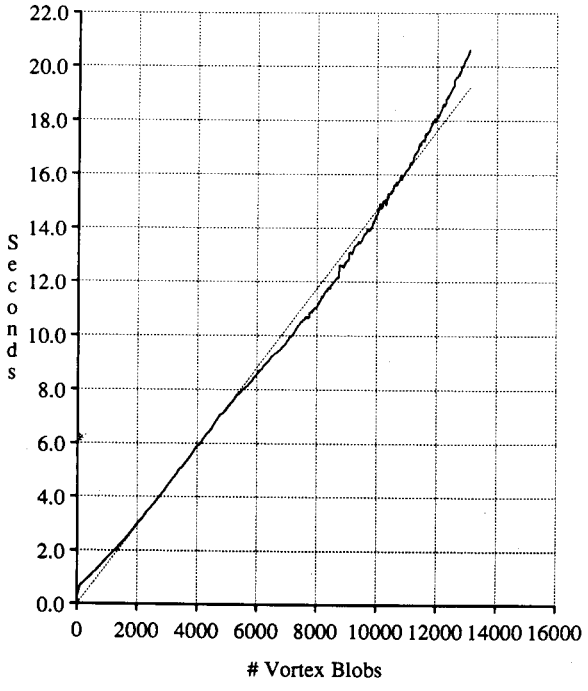

Figure 4. The total cost of a time step evaluation is roughly a linear function of the number of vortex blobs. The dotted line plots the ideal linear cost function.

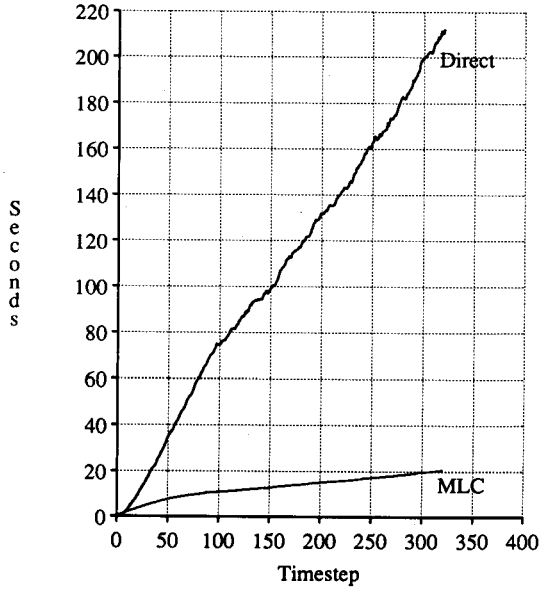

Figure 5. The cost of a time step evaluation drops substantially when the local corrections algorithm is used to evaluate velocities instead of the direct method. The speedup of the local corrections algorithm increases with the number of vortices N, which increases with time. The times are reported in seconds of CPU time and were measured with the `second` routine on a Cray X-MP.